FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Throughput Forecasting for Crowdsourced Text-Enrichment

Inês Isabel Correia Gomes

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: João Mendes Moreira

July 3, 2019

Throughput Forecasting for Crowdsourced Text-Enrichment

Inês Isabel Correia Gomes

Mestrado Integrado em Engenharia Informática e Computação

July 3, 2019

Abstract

Data-driven applications and intelligent systems, such as chatbots or autonomous cars, require large amounts of structured data. When these applications involve solving problems linked to human perception (vision, speech and natural language), data needs to be enriched with human knowledge. However, hiring a single group of annotators is slow and expensive. Crowdsourcing platforms address these scalability limitations (both in time and money) by assigning a set of tasks to an existing pool of non-expert contributors in exchange for monetary reward.

The nature of crowdsourcing environments introduces uncertainty into the annotation costs and completion timelines. The crowd requirements (availability and eligibility), contributors characteristics (e.g. expertise or learning curve), task complexity (cognitive load) and the size of the information processed are variables that when dealing with millions of data units, significantly affect the production life-cycle. For this reason, it is necessary to evaluate the workflow over time, so that we can deal with crowd performance variations and manage expectations in real-time.

In this dissertation, we address these scalability limitations by investigating both cost and time constraints in named entity enrichment workflows. Regarding the former, we estimate cost in human hours put in by the crowd to complete the tasks. This definition allows us to measure the total annotation effort, so that future monetary costs can be estimated. Regarding the latter, we forecast the foreseeable crowd throughput given the annotation workflow historical data. This definition allows to estimate throughput variations over time, so that completion timelines expectations can be met.

To answer the first research question, we build a baseline formula that reaches 19% MAPE (Mean Absolute Percentage Error) using only two variables: the number of tokens (corr=0.86) and the number of entities involved (corr=0.57). To answer the second research question, we compare a naive forecasting (persistence) with Box-Jenkins seasonal ARIMA and Exponential Smoothing for daily throughput forecasting considering hourly observations. The Exponential Smoothing method attained the best results achieving 0.39 RMSLE (Root Mean Squared Logarithmic Error).

We conclude that measuring task cognitive load, amount of the information to be processed and the available number of contributors are important factors for scalability estimation, both in time and cost. Also, we noticed diversity in behaviour among workflows, leading to adjustments in the prediction continuously over time.

Resumo

Aplicações baseadas em dados e sistemas inteligentes, como chatbots ou carros autónomos, exigem grandes quantidades de dados estruturados. Quando essas aplicações estão relacionadas com a solução de problemas ligados à percepção humana (visão, fala e linguagem natural), os dados precisam ser enriquecidos com o conhecimento humano. No entanto, a contratação de um único grupo de anotadores é lento e dispendioso. As plataformas de crowdsourcing abordam essas limitações de escalabilidade (em tempo e dinheiro), atribuíndo um conjunto de tarefas a um grupo existente de colaboradores não especializados em troca de uma recompensa monetária.

A natureza dos ambientes de crowdsourcing introduz incerteza nos custos de anotação e respetivas datas de conclusão. Os requisitos do grupo de colaboradores registados na platforma (elegibilidade e disponibilidade), características dos colaboradores (por exemplo, especialização ou curva de aprendizagem), complexidade das tarefas (carga cognitiva) e tamanho da informação processada são variáveis que, ao lidar com milhões de unidades de dados, afetam significativamente o ciclo de vida da produção. Por esse motivo, é necessário avaliar o fluxo de trabalho ao longo do tempo, para que possamos lidar com variações de desempenho dos colaboradores e gerir expectativas em tempo real.

Nesta dissertação, abordamos essas limitações de escalabilidade investigando as restrições de custo e tempo em fluxos de trabalho de enriquecimento de entidades nomeadas. Em relação às restrições de custo, estimamos o custo em horas humanas colocadas pelos colaboradores para completar as tarefas. Essa definição permite nos medir o esforço total de anotação, para que os custos monetários futuros possam ser estimados. Em relação às restrições de tempo, prevemos o trabalho esperado para um futuro próximo dado o a informação sobre o trabalho produzido recentemente durante o fluxo de anotação de dados. Assim, as expectativas para os prazos de conclusão podem ser atendidas.

Para responder ao primeiro problema posto, construímos uma fórmula de base que atinge 19% de MAPE usando apenas duas variáveis: o número de *tokens* (corr = 0.86) e o número de entidades envolvidas (corr = 0.57). Para responder ao segundo problema posto, comparamos uma previsão básica (persistência) com o ARIMA sazonal de Box-Jenkins e o *Exponential Smoothing* para a previsão diária do trabalhado produzido em horas, considerando as observações por hora. O método de *Exponential Smoothing* atingiu os melhores resultados, com 0.39 RMSLE.

Concluímos que medir a carga cognitiva de uma tarefa, a quantidade da informação a ser processada e o número de colaboradores disponíveis são fatores importantes para a estimativa da escalabilidade, tanto em tempo como custo. Além do mais, notamos diversidade no comportamento entre fluxos de trabalho, levando a ajustes na previsão ao longo do tempo.

Acknowledgements

From the day I have born to this degree, I thank my parents, Alzira e Simão, for raising me, guiding me, inspiring me, encouraging me and advising me the best they could every day of my life. I thank my brother Diogo for taking part in most of my adventures and cheering me up every time, and also my grandparents for wishing me the best, believing in me and giving me the wisest advice.

In the last five years, I have made wonderful friends. Catarina Ramos and Mário Gustavo, I must thank you, from the bottom of my heart, the fellowship, laughs, tears, encouragement, advice... You have changed me for the best. From the days of study in the library to all the exciting projects that we have made, my degree was built with you and my grades depended on you. I am grateful to have met you. Thank you.

To my supervisor, João Mendes Moreira, who has guided me through this dissertation. His good mood and expertise were the perfect combination to help me reach my goals. Thanks for always giving me new perspectives, remembering my timeline and narrowing my scope.

To DefinedCrowd, who gave me the smoothest transition from the academic to the business world, providing me with everything that I could have asked that improved my labour well-being. To DefinedCrowd - Porto, for spoiling me and supporting me in every occasion. Thank you, João Freitas, for believing in me from the first day, giving me the freedom and making me comfortable to explore my thesis theme the best I could. A special thank note to Jorge Ribeiro, that has always asked for the extra mile. Thanks for never pushing me aside and for encouraging my work. Thanks for teaching me that a good data scientist is the one who listens and asks the right questions. Thanks for teaching me how to sell my solution to the business. Thanks for helping me to start my career.

And last but not least, to Miguel Freitas. From the very beginning, you are my biggest supporter. Thanks for challenging me and pushing me to succeed. You have taught me that the sky is the limit. Thanks for cheering me up when I was down and celebrating all my achievements as they were yours. Thanks for helping me find the balance between work and leisure and inspiring me to create new milestones that I have never thought of before. You have made me the person that I am today: strong and persistent. I love you.

Inês Isabel Correia Gomes

The only way for humans to stay in the game will be to keep learning throughout their lives, and to reinvent themselves repeatedly - Yuval Noah Harari, Homo Deus

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

AIC	Akaike Information Criterion
ACF	Autocorrelation Function
BIC	Bayesian Information Criterion
CRISP-DM	CRoss Industry Standard Process for Data Mining
IQR	Interquartile Range
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
max	Maximum
min	Minimum
NET	Named Entity Tagging
NaN	Not a Number
PACF	Partial Autocorrelation Function
RF	Random Forest
RMSLE	Root Mean Squared Logarithmic Error
ARIMA	Autoregressive Integrated Moving Average
SARIMA	Seasonal Autoregressive Integrated Moving Average
ARIMAX	Autoregressive Integrated Moving Average with exogenous variables
SARIMAX	Seasonal Autoregressive Integrated Moving Average with exogenous variables
std	Standard Deviation

Terms

Autocorrelation	Correlation between values of the same time series at different time periods
Cognitive Load	Combination between inherent difficulty in the content of the presented material and its organisation and presentation [SC94]
Contributor	Crowdsourcing platform user that performs tasks in exchange of monetary reward
Crowd	Group of contributors
Crowdsourcing	General-purpose problem-solving method [DRH11] that outsources the phase of idea generation to a potentially large and unknown population in the form of an open call [PS12]
Crowd Effort	Cumulative time that the crowd must put in to complete a job
First order differenced series	Difference between observations separated by *one* time period, i.e. $Y_t' = Y_t - Y_{t-1}$ [MWH98]
Goodness of fit	Property of how well the forecasting model is able to reproduce the data that are already known [MWH98]
Heteroscedasticity	Property of the errors that do not have a constant variance across an entire range of values [MWH98]
Job	Set of tasks characterised by having the same requester, subject, micro-task type, being created together at a given point in time and having the same end. A job is a workflow that finishes when all tasks are completed
Lag	Difference in time between an observation and a previous observation. Thus Y_{t-k} lags Y_t by k periods [MWH98]
Micro-Task	Consists of small chunks of lower complexity tasks that can be distributed with ease across a mass of people
Real-World job length	Number of hours that a given job takes to complete considering the first and last tasks timestamps
Speed on task	Length of the information to be processed by the contributor divided by the given contributor time on task, measured in tokens/minute
Seasonal	Pattern type observed when a series is influenced by seasonal seasonal factors (e.g., the quarter of the year, the month, or day of the week) [MWH98]
Stationary	Property of a time series data for when it is in equilibrium around a constant value (the underlying mean) and the variance around the mean remains constant over time [MWH98]
Throughput	Amount of work expected in hours from a group of contributors assigned to some job in a given span of time

Time on task	Time elapsed from the moment that the task is visible to the contributor and it is submitted, measured in milliseconds
Time Series	Ordered sequence of values of a variable observed at equally spaced time intervals [MWH98]
Trend	Type of pattern observed when there is a long term increase or decrease in the data

Chapter 1

Introduction

The current chapter serves as an introduction to this dissertation. In Section 1.1 we explain the crowdsourcing context. In Section 1.2 we describe the short comings that motivate this dissertation. In Section 1.3 we explain the thesis scope briefly explaining this collaboration and narrowing down to our case study. In Section 1.4 we describe our problem, establishing our research questions and in Section 1.5 we define our objectives. Finally, the document's structure is presented in Section 1.6.

1.1 Context

The ongoing fourth industrial revolution, also known as Industry 4.0, is strongly related to data-driven applications and intelligent systems [Lu17]. The industry 4.0 goals are to achieve a higher level of operational efficiency, productivity and automation. Well-known examples of this revolution are personal assistants and autonomous cars.

As a result of this technological revolution, the integration of information and communication technologies generates massive amounts of data [Lu17], mostly unstructured, that is usually difficult to analyse [KYA+14]. Having the capability of leveraging unstructured data lead to more accurate and robust models with better performance and successively resulting in better applications.

However, when applications need to generalise from human judgement (e.g. named entity recognition or image classification) they require data labelled by humans to train and evaluate algorithms [EGWW10]. Given the lack of labelled data, research datasets must be generated either by users or automatic data generators [KYA+14]. Research datasets drive research in new and more challenging directions, as they create ground truths and push the fields towards more complex problems [LM+14].

The traditional approach to perform human-in-the-loop data collections is employing a single group of human experts creating the relevant material. However, perform these tasks is slow and expensive [EdV13]. These restrictions, allied to intrinsic and extrinsic crowd motivations as enjoyment, personal improvement or generalised payment methods [YLW+14], lead crowdsourcing to establish itself as a viable model to enrich and structure data.

Micro-task crowdsourcing's core concept is that of assigning tasks to an existing pool of non-expert contributors in order to solve a problem in exchange of a monetary reward. Micro-tasks consist of dividing work into smaller chunks of lower complexity that can be distributed with ease across a mass of people. Crowdsourcing relies on the belief that crowds can yield more accurate answers than an individual [YLW+14]. As opposed to traditional employee models, crowdsourcing can access the vast potential of contributors with various backgrounds and levels of expertise [GS14], increasing its popularity among the organisations.

1.2 Motivation

The size of the datasets to be annotated can vary from tens/hundreds of data units to millions. As a result, the necessary timeline for annotating them varies significantly. As an example, MS Coco, a dataset with 2.5 million labeled instances in 328k images, even built resorting to crowdsourcing community took around 60,000 worker hours to complete [LM+14]. Along with this sort of scale, arises the need for effectively measuring and estimate the costs of enriching different datasets in order to manage expectations and pay the contributors fairly.

How long a data collection or enrichment workflow takes is influenced by a combination of factors:

- *Crowd requirements* (eligibility and availability): the former determines the selected set of contributors to complete the tasks and the latter their opportunity to contribute. As an example, if a certain text must be translated from German to English, a crowd requirement is that contributors must be both German and English fluent. This restriction already narrows the crowd to a subset of contributors. Availability, on the other hand, is influenced by the open nature of the crowdsourcing platforms that allows contributors to chose if they want to contribute, and if so, when and the extent of the contribution.

- *Contributors characteristics* : contributor knowledge and skills that influence their activity in the platform, as the expertise level in certain task types or the learning curve when executing new workflows.

- *Tasks inherent complexity* (cognitive load) and *amount of information* to be processed.

The trade-off between task and contributor characteristics affects the time needed to complete a task.

These described factors introduce uncertainty with respect to the time needed for a dataset to be annotated. Therefore, project managers struggle to estimate costs and timelines. Moreover, when dealing with extensive annotation assignments that can reach months to be completed, there are several unpredictable circumstances that can affect timelines. Thus, managers must monitor closely the annotation process and have mechanisms to evaluate out of expectations behaviours. As an example, a small decrease in throughput over days could be unnoticed when checking raw data. As a result, managers would have the false perspective that the established timelines would be met.

Summing up, establishing baselines of working hours expected and provide crowd statistics allow project managers to create realistic timelines, have pricing perspectives and coordinate the crowd as needed. Moreover, forecast increasing or decreasing throughput trends, considering on-going annotation assignments, allows to anticipate short comings and act accordingly.

1.3 Thesis Scope

This thesis is the result of a collaboration between *Faculdade de Engenharia da Universidade do Porto* (FEUP) with *DefinedCrowd* [1]. *DefinedCrowd* is a smart data platform that enables data scientists to collect, refine, and structure training data for Artificial Intelligence and Machine Learning applications. *Neevo* [2] is one of the products, a crowdsourcing platform where contributors sign up to execute micro-tasks in exchange of a monetary reward. These small tasks include text, audio or images and can be used to improve virtual assistants, autonomous cars, video-games or even predictive medical diagnosis.

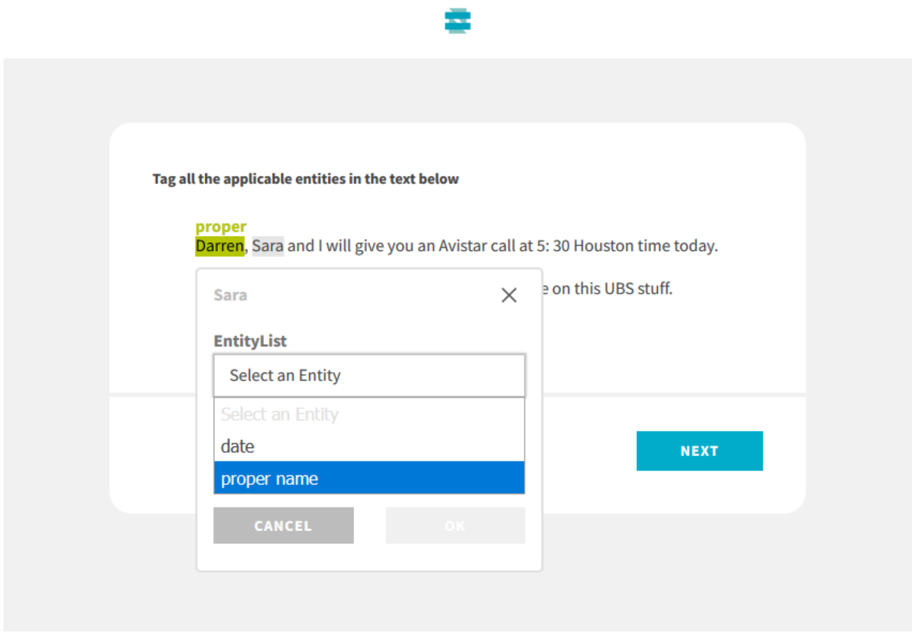
To carry out our studies, we focus on Named Entity Tagging (NET) jobs. A *job* is a work-flow of tasks executed by contributors with the same subject and the same end. NET job type is characterised by recognising information units, i.e. named entities, in unstructured text [NS07]. A named entity is any reference, whether in full by proper name, by the nickname, shortened version, abbreviation, or acronym, to a unique entity in the world. That entity may be a Person, a Date, a Commercial Product, or any other well-defined real or metaphysical object.

Figure 1.1 shows an example of a task belonging to a NET job with two entities to tag (*proper name* or *date*). The contributor must select the word to tag and then select the corresponding entity. In this example, *Darren* was already tagged by the contributor as a *proper name*, *Sara* is being tagged while *5:30* was not yet tagged with *date* entity.

1.4 Problem Statement

We address crowdsourcing large dataset annotation scalability, both in cost and time, limitation by estimating *crowd effort* and forecasting *crowd throughput*. When annotating large NET datasets, there are several factors that can influence the effort needed to put in by the crowd. We designate this effort as the cumulative sum of time on task by the crowd measured in hours, i.e. human computation time. This measurement is more accurate than number of tasks completed per job, considering that it takes into consideration the trade-off between contributors characteristics, task cognitive load and the size of the information to be processed.

[1] https://www.definedcrowd.ai/
[2] https://neevo.definedcrowd.com/en-us/community/


Tag all the applicable entities in the text below

proper
Darren, Sara and I will give you an Avistar call at 5: 30 Houston time today.

e on this UBS stuff.

Figure 1.1: *Neevo* by DefinedCrowd Crowdsourcing Platform with NET task example. The contributor must select, with the mouse, the words from the sentence that suits the available *Named Entities*.

However, factors as the crowd eligibility and availability introduce uncertainty with respect to completion timelines. We define *crowd throughput* as the amount of effort expected in hours put in by the crowd in a given span of time. In other words, we establish a span of time, that in this context is *one hour*, and we calculate the effort considering the tasks completed within that hour.

The throughput over time is a valuable measure as it provides insights about:

- The effort that the tasks completed required;
- Seasonality: as an example, if our crowd requirements establish that the contributors must inhabit in Portugal, it will be possible to observe maximum throughput between 6 p.m and 10 p.m and lower throughput during the night;
- Trend: larger jobs can take months to finish. It will be common to identify increasing, decreasing or horizontal throughput trends over time;
- If the project manager influences some how the crowd, it will be possible to observes its effects in short-time.

Given these points, we consider that estimate the future throughput is a business asset. According to Makridakis in his book *'Forecasting: Methods and Applications'* [MWH98], forecasting is used when there is a time lag between awareness of an impending event and occurrence of that event. This interval is called lead time, and it is the main reason for planning and forecasting.

When the lead time is sufficiently long and the outcome of the final event is conditional, planning has an important role. This thesis relies on the fact that forecasting applies directly to the uncontrollable external events (throughput) while decision making applies directly to the controllable internal events (e.g. crowd management) and that planning is the link that integrates both [MWH98].

As a result, we established two research questions, one per each scalability measure that we want to estimate. The former, relates to the **crowd effort** while the latter refers to the **crowd throughput**.

RQ1 : *How to measure the crowd effort of a Named Entity Tagging job?*

RQ2 : *How to forecast the foreseeable crowd throughput for a ongoing Named Entity Tagging job considering its historical data?*

1.5 Objectives

To answer **RQ1** regarding the crowd effort estimation, we established the following objectives:

- Understand time on task data distribution so that it is possible to define out of normality tasks;
- Define criteria to remove tasks out of what is human plausible time on task (too slow and too fast completion times);
- Extract features to measure the task cognitive load;
- Assess that the *contributors' characteristics*, *tasks cognitive load* and *size of the information to be processed* influence the *crowd effort*;
- Create a simple formula that estimates the crowd effort using the extracted features.

To answer **RQ2** regarding the crowd throughput forecasting, we established the following objectives:

- Create a time series dataset that aggregates the observations within one hour. We create one time series per job;
- Observe the throughput over time and draw conclusions about how different (or similar) our time series are (e.g. considering seasonality, trend, stationarity and others);
- Create a relation between real-time job length (the number of hours that a given job takes to complete considering the first and last task timestamps) and crowd effort, so that we can assess the completion timelines unpredictability;
- Find which variables influence the throughput over time;

- Study contributor daily throughput per job over time in order to find similarities among contributors and explain possible crowd throughput variations;
- Compare forecasting horizons of one hour and one day (24 hours) to understand the model deterioration;
- Compare several forecasting methods and discuss results.

1.6 Document Structure

In Chapter 2, we make an overview of the background including state of the art regarding crowdsourcing and throughput prediction studies. Finally, we detail describe the machine learning concepts used in the following chapters.

In Chapter 3, we explore and describe the data, including cleaning and preparation. While doing so, we explain and create ways to measure human computation time and task complexity concerning **RQ1**. Then, create a naive formula and assess its performance. In the end, we discuss the results achieved.

In Chapter 4, we prepare the data to hold hourly observations and explore the workflow's progress in order to understand what influences the crowd throughput concerning **RQ2**. We start by describing the experimental setup and test design. Then, we model our problem starting by identifying our modelling methodology and then building the models. In the end, we assess, evaluate and discuss the results.

Finally, Chapter 5 summarizes our study, draw the final conclusions, describe the major contributions, discuss limitations and point to future work directions.

Chapter 2

Background

This chapter describes previous work related to this dissertation. Our goal is to find concepts in the state of the art that can explain the effort that the crowd may put in to complete the tasks and the factors that can influence the cumulative time spent in the platform over time.

This chapter is organized as follows: in Section 2.1, we start by explaining the scope of our study describing what is crowdsourcing, narrowing to Micro-Task platforms in 2.1.1 and then briefly explaining the platform workflow in 2.1.2. Then, in Section 2.2 we start by exploring the contributor (see Section 2.2.1) and job (see Section 2.2.2) related concepts that can explain the crowd effort and throughput. In Section 2.2.3, we analyse others studies related to the throughput prediction in crowdsourcing and their performance metrics in Section 2.2.4.

2.1 What is Crowdsourcing

Crowdsourcing is a general-purpose problem-solving method [DRH11] that outsources the phase of idea generation to a potentially large and unknown population in the form of an open call [PS12]. Crowdsourcing is a relatively recent concept, coined by Jeff Howe in 2005 and derives from outsourcing. As Jeff Howe affirms [How12],

> "Crowdsourcing is a business practice that means literally to outsource an activity to the crowd"

Crowdsourcing has been used for a wide group of activities from knowledge-intensive processes to marketplaces for customer support services [VB10].

Crowdsourcing is a scalable solution that allow tasks to finish faster and with lower costs [KTK12] when comparing to hiring a single group of non-expert contributors. Crowdsourcing relies on the fact that they can approximate expert judgements by leveraging multiple non-expert responses [YLW$^+$14].

Nowadays, the crowdsourcing community has grouped into five groups [HC13]:

- *Microtasks*: outsources tasks that can be completed within minutes by an average contributor;
- *Macrotask*: manage large projects requiring weeks of effort with multiple skilled contributors;
- *Crowdfunding*: allow people to gather money from the crowd, for a specific project or cause;
- *Contest*: allow organisations to solicit best solutions or ideas by offering rewards per solution;
- *Open Innovation*: content generation that uses the community belonging incentives instead of monetary reward, e.g. Wikipedia.

In this dissertation, we focus on micro-task platforms that will be further explained in Section 2.1.1.

2.1.1 Micro-task Definition

A micro-task is a task (e.g. sentence translation, object detection, entity tagging) that can be completed within a short time by an average contributor [HC13] for monetary reward. There are several types of micro-tasks within a crowdsourcing platform that require different contributor skills and knowledge.

Over the last decade, micro-tasking has been used to validate, evaluate and annotate large volumes of data, because of the data-intensive nature of emerging tasks [GKDD15]. These tasks typically include labelling objects in an image, transcribing audio, or judging the relevance of a search result [DCD+15]. These type of tasks are characterised by being completed by contributors as otherwise would be extremely difficult for computers to perform [KCS08].

Some of the advantages of using micro-task crowdsourcing to data annotation are the immediate access to a large pool of contributors, contributors with vast diversity and characteristics, flexibility in work assignment and faster results.

2.1.2 Micro-task Crowdsourcing Platform

Crowdsourcing is constituted by a set of relationships that connect organisations, individuals, technologies and work activities [AVNSB+13]. These are the four main actors of a popular micro-task crowdsourcing platform [DCD+15]. Figure 2.1 shows an example of a micro-task crowdsourcing platform structure and interaction.

Crowdsourcing platforms serve as a place where contributors perform micro-tasks in exchange of a monetary reward [DCD+15]. Platforms are responsible for the decomposition of larger tasks into micro-tasks, to give access to the crowds and to aggregate the micro-tasks answers [HKH+14].

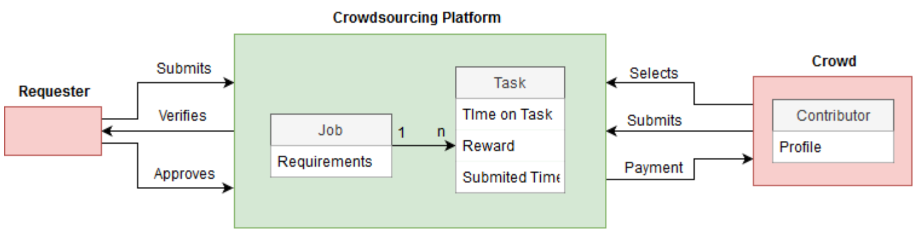


Figure 2.1: A micro-task crowdsourcing platform structure example

Some examples of micro-task crowdsourcing platforms are the Amazon *Mechanical Turk* [1], *Figure Eight* [2], *Click Worker* [3] and *Neevo* by DefinedCrowd [4].

In a platform, the requester is the actor who proposes tasks [YLW+14]. Depending on the platform, the requester can be an individual, group or organization [AVNSB+13] and can have different responsibilities. Also, according to the platform, the requester can be a project manager and can have information about the workflow completion state over time and can have influence on it. In Crowdsourcing platforms the requester can usually either choose between standard workflows for the most common labelling tasks, or to design workflow's sections. A workflow is called a *job* and corresponds to a set of tasks usually with the same subject and micro-task type that finishes when all tasks have been completed by the contributors.

A *contributor* is an individual who is rewarded per each task performed. This reward is usually monetary [RKK+11]. Further comprehension about the contributor's motivations is explained later in Section 2.2.1. On the other hand, *crowd* is the group of contributors signed up in a micro-task crowdsourcing platform. The contributors have varying profiles that can be categorised according to characteristics such as language proficiency, demographics or even performance based on their history in the platform. Consequently, it is possible to match tasks requirements with contributors capabilities.

2.2 Crowd Effort and Throughput State of the art

To predict a job completion time, other metrics beyond the contributor characteristics and abilities must be considered. With this in mind, task characteristics, workflow stages and crowd availabilities are discussed.

Jiang defines throughput as the number of tasks completed by the system per time unit under steady state condition [Jia16]. Mapping this concept to crowdsourcing, the throughput is the amount of work expected given the a set of factors, as:

- *Internal factors* as job settings. For example, number of tasks in the job, language requirement, minimum contributor proficiency;

[1] https://www.mturk.com/
[2] https://www.figure-eight.com/platform/
[3] https://www.clickworker.com/
[4] https://neevo.definedcrowd.com/en-us/community/

- *External factors*. For example the contributors availability or contributor characteristics (as expertise or learning curve).

In Sections 2.2.1 and 2.2.2 we study metrics and parameters that are fundamental to control the system effort and throughput.

2.2.1 Crowd-Related Dimensions

The contributors play a major role in a crowdsourcing platform as they are the business workforce. In order to understand the impact of their work, it is important to explain contributor characteristics and capabilities concerning the individual, instead of the crowd. For that reason, some state of the art has been studied.

According to Hassan and Curry [HC13],

> "Capability is defined as the ability of humans to do things in terms of both the capacity and the opportunity."

They also defend three main types of capabilities: Knowledge, Skill and Ability, followed by other miscellaneous factors like motivation, attitude and social relations [HC13].

Sauter and Bohm studied data quality enforcement techniques that consider throughput and they concluded that there is a correlation between the capability and honesty of contributors and crowdsourcing throughput [SB13].

The contributor **knowledge** can be measured by the knowledge acquired in the real-world [GS14]. On the other hand, the contributor skill can be learned through observations as contributors complete tasks [HV12]. Vaughan and Ho stated that each contributor is assumed to have an underlying skill level for each task and that, on average, contributors with higher skill levels add more value [HV12]. Additionally, Allahbakhsh et al. explain that a high skilled contributor is expected to have a high reputation as well [ABI$^+$13]. In detail, **reputation** is the feedback about contributors' activities in the system [ABI$^+$13]. Finally, the contributor **ability** can be measured with factors as *comprehension*, *bilingualism*, *comparison*, *judgement*, *perception*, *identification* and *reasoning*. Different subsets of abilities are meaningful to different micro-task types.

Other studies have been carried out in order to understand contributor specific characteristics. As an illustration, Vreede et al. made a study about **user engagement** in Crowdsourcing. They define user engagement as the quality of effort contributors devote to open collaboration activities that contribute directly to desired outcomes [DVNDV$^+$13]. They identified three crucial factors that precede user engagement: personal interest in topic, goal clarity, and motivation to contribute. Vreede et al. propose [DVNDV$^+$13],

> "Goal clarity moderates relationship between personal interest in topic and intrinsic motivation to contribute."

Personal interest can be topical (developed through personal experiences and emotions) or situational (context-dependent). **Goal clarity** refers to which extent the task objectives are clear and simply stated. Finally, **motivation** is a force that can determine the form, direction and duration of the task to be committed.

Yin et al. make a reliable distinction between intrinsic and extrinsic motivation. Intrinsic motivation is related to enjoyment, altruism, personal improvement and habit or preference. On the other hand, extrinsic motivation is related to **payment**, social factors (e.g. games) or even requirement (e.g. ReCAPTCHA) [AMM+08]. Rogstadius et al. did a deep study about motivation and concluded that contributors' performance and effort are affected by varying the levels of intrinsic and extrinsic motivation in a task [RKK+11]. As an illustration, paying people more did not lead to increases in their accuracy; intrinsic motivation did not impact the job completion times; intrinsic motivation has a strong positive effect on contributor accuracy, but only when extrinsic factors become the main motivator; higher payment leads to quicker results [RKK+11]. Mason and Watts support this statement declaring that increasing financial incentives increase the quantity, but not the quality of work [MW10]

Concluding about user engagement, Juin et al. carry out a contributor study where they inferred about having two types of sources, the *dedicated workforce*, performing 80% of the tasks, and an *on-demand workforce* performing the remaining ones. The dedicated workforce corresponds to just 10% of the workforce [JSPW17]. This observation indicates that contributor interest and engagement are crucial to reinforce the dedicated workforce, because these are the ones that allow marketplaces to handle fluctuating workloads better [JSPW17].

The previous conclusions are closely related to the definition of contributor **activeness**. Contributor activeness is the availability of the contributor on the crowdsourcing platform [KS18]. Kurup and Sajeev carried out a study where activeness was the time from which the contributor submitted his first task to the time till he submits his last task [KS18]. They concluded that the contributor activeness decreases exponentially over their experience and the activeness plays a crucial role to improve productivity [KS18]. Other metrics related to contributor availability were studied. Jain et al. define the contributor **lifetime** and **working days**. The contributor lifetime is the number of days between their first and last task execution and the working days are the number of days, within the lifetime, so that at least one task has been performed [JSPW17]. With this study, Jain et al. concluded that a majority of contributors perform only one or a few tasks, having short lifetimes [JSPW17].

Another previously referred metric was a contributor **effort**. Jain et al. [JSPW17] and identically Eickhoff and Vries [EdV13] solely measure the effort as the amount of time taken to complete a task. Sautter and Bohm also hold this definition and state that when contributors receive some reward for their input, they might cheat to reduce their effort [SB13]. Also, they conclude that effort reduction, not only reduces the probability of cheating but also increases the workflow throughput. However, Cheng et al. explain that task duration is not a reliable measure as is a noisy signal as contributors tend to switch between tasks or ignore them for periods of time [CTB15]. Rogstadius et al. have a different effort definition. They define effort as an aggregation of total completed assignments, total working hours and mean time per task [RKK+11]. Finally, Cheng et al. measure the effort as the impact of time on the number of errors contributors make [CTB15].

Their strategy is to calculate a task's error-time trade-off curve by giving contributors varying time constraints to finish the task and measure the probability they make an error within each time limit [CTB15].

Similarly to Cheng et al., Hirth et al. affirm that it is possible to detect low performing contributors by defining time thresholds. With this in mind, they define *reading* and *answering* **speed**. They define 200 words/min as reading speed and 5 sec/answer [HSH+14]. Per each task, they calculate the *completion times* based on the baseline speeds and can differentiate between *fast* and *slow* contributors.

These metrics studied by Vreede et al., e.g. motivation and personal interests, were already mentioned by Hassan and Curry as other miscellaneous factors that included **social factors**. Vukovic and Bartolini [VB10] gathered the relevant state of the art concerning this topic as demographics, ethnicity, gender or tech savviness. Martin et al. studied age, work devices (e.g. laptop, mobile), workplaces (e.g. college, office, cafe) economic status and educational level [MCG+17] Social networks or games are also a social factor that can motivate the user, as stated previously.

Concerning **demographics**, Rogstadius et al. studied the differences between North Americans and Asians with respect to intrinsic and extrinsic motivations [RKK+11]. Kurup et Sajeev analysed the distribution of tasks over the United States, India, United Kingdom and Canada [KS18]. Both studies show that contributors' demographics affect somehow the job workflow.

Finally, it is possible to characterise a contributor given his/her **productivity**. The contributor productivity, according to Kurup and Sajeev, can be computed as the trade-off between the number of tasks submitted and the tasks average performance given an active period [KS18].

2.2.2 Job-Related Dimension

Tasks have temporal dynamics that consequently affect the workflow completion times. Task **arrival rate** is the number of tasks arriving over a time period [JSPW17]. This behaviour analysis allows to identify trends and cycles that combined with the contributor availability and demographics can sum up to a workflow trend analysis, e.g. identify busy hours and days of the week per each country or busy countries or regions. The task **incentives** are the payments associated with task completion. In Section 2.2.1 extrinsic motivations as the monetary reward, are studied with the aim of optimising the trade-off between price and contributor performance. In this section, we analyse the trade-off between price and task characteristics, e.g. complexity or topic. As an illustration, Kurup and Sajeev affirm that requesters prefer to post more complex tasks containing multiple tasks and offering a better reward [KS18]. At the same time, contributors have been preferring higher rewards over the last few years [KS18], but still, micro-payments as suggest by Gadiraju et al. that did a survey where 30% of the contributors claim preference "easy to complete" tasks [GKD14].

Sweller and Chandler studied the **cognitive load** theory and propose two sources: **intrinsic** and **extraneous**. The former refers to the inherent difficulty in the content of the presented material, that is, the task complexity [SC94]. The latter refers to the organisation and presentation of the material, that is, task design and clarity [SC94]. Finnerty et al. affirm that a high cognitive demand generally leads to worse performance [FKTC13], so both intrinsic and extraneous cognitive load must be reduced.

Complexity reflects the temporal demands and the real cognitive effort that performers need to put into the task completion [YRDB16]. Concerning task complexity, it has been found previously that the complexity and the effort needed [CTB15] can contribute to the workers ability to perform a task well [FKTC13]. Yang et al. studied the factors that influence task complexity. They start with a set of six factors (*Mental, Physical, Temporal, Performance, Effort, Frustration*), weighted representing their relevance. The overall task complexity was studied through a questionnaire. Yang et al. conclude that mental complexity and effort to complete a task are the most relevant features to judge the task complexity, as opposed to frustration that obtained the lowest weight [YRDB16]. Finally, they also conclude that complexity is reflected from the point of view of required actions and consequently by the number of task elements (text, images, links) available [YRDB16].

Clarity is the task quality in terms of its comprehensibility. It can be described by goal clarity, that is, the extent to which the objective of the task is clear, and by the role clarity, that is, the extent to which the steps or activities to be carried out in the task is clear [GYB17]. Gadiraju et al. carried a survey to understand which factors make tasks unclear. Contributor complained both about the task instructions (e.g. *vague*, *inconsistent*, *ambiguous*, *poor*) and the language use (e.g *too many words*, *high standard of English*, *broken English*, *spelling*) [GYB17]. Among other conclusions, Gadiraju et al. deduced that the features that influence the most the task clarity are *task title*, *instructions*, *description*, and *keywords* associated with the task; goal clarity was slightly more influential than role clarity in determining the task clarity; longer words decrease task clarity, while longer sentence enhance it; tasks with high clarity can be highly complex and tasks with low clarity can have low task complexity at the same time [GKD14].

A common mechanism to ensure quality among tasks is **redundancy**, that is, it consists of several copies that are executed by a number of separate contributors and the final result can be achieved by a majority voting rule [Jia16]. Unfortunately, redundancy has a severe impact on throughput, combined with complex decisions it leads to relatively low throughput [SB13].

A task pipeline starts as soon as the job is created. First, the contributor must accept the task, then the contributor must complete it. The former is defined by Jain et al. as the **pickup time**. The pickup time determines how quickly tasks are picked up by contributors, i.e. the difference between the job and task start time [JSPW17]. The latter is defined by Hirth et al. as the **completion time**. The completion time determines the time interval between the start and finishes time [HSH$^+$14] and can be divided into two working phases: *reading instructions time* and *answering the question time* [HSH$^+$14]. The completion time is not reliable because it includes inactive time. As an illustration, in the Hirth et al. study they show that the completion times can vary between 1 and 56 minutes, with an average of 20 minutes [HSH$^+$14]. The reading time is estimated based on the average reading speed (already defined in Section 2.2.1), scrolls and clicks. The answering time is the difference between the first time the contributor could consider the answer and the contributor final answer [HSH$^+$14].

In order to understand the platform workflow, Kurup and Sajeev analyse the task and worker behaviours. Concerning the tasks, the task arrival rate was already explained in Section 2.2.1. Concerning the contributors, the **task acceptance rate** is the rate at which the task is accepted by the contributor. This rate is measured by the number of micro-tasks available in each batch and their difficulty level [KS18]. Jain et al. compared the contributor availability with the task arrival rate and concluded that there is a limitation in the supply of task rather than in the availability of contributors [JSPW17].

2.2.3 Machine Learning applied to Throughput Prediction

After analysing the contributor and task factors that can influence the throughput, we study the state of the art about the throughput prediction.

Difallah et al. studied the throughput of a batch of tasks at time T, given some factors that influence the batch progress [DCD$^+$15]. The batch is a set of tasks in Amazon Mechanical Turk. They train a *Random Forest* model, with samples taken in the range [T - δ, T] and, where δ is the size of the time window, e.g. $\delta = 4hours$. Some of the metrics used on this prediction model were the number of *Tasks available* per each batch, *reward*, *location*, *time allocated* per task. They have concluded that the prediction works best for larger batches and the best prediction features were the *Tasks available* and *Age of batch in minutes* [DCD$^+$15].

Yang et al. also researched throughput prediction, inspired by Difallah et al. studies. Their goal is to predict the number of tasks executed in a predefined time span [YRDB16]. The dataset was divided into three subgroups with batch size varying in the range of [1,10), [10,100) and [100, 1000) and four type of features: *Metadata* (i.e. title, description, initial Tasks), *Dynamic* (i.e. marketplace context), *Content* (i.e. number of words, keywords), *Content LDA* (i.e. content extracted features). The *Mean Absolute Error* (MAE) was used to measure the prediction performance. Some of the metrics used included the total number of *Available* Tasks; the total number and relative size of *Published* and *Completed* Tasks; total *reward*. The models used were *Linear Regression*, *Lasso*, *Random Forest* regression and their own predictor (*MFLR*). The experiment results say that *dynamic* features provide the least prediction support; *content* features achieve the best performance with *Lasso* with small batches; *content LDA* features perform better than regular *content* with the same conditions; *metadata* features achieves better performance with *Random Forest* regression and big batches [YRDB16].

Yang et. al finally conclude that further investigation focusing on the relationship between task complexity, market dynamics, and execution performance is needed [YRDB16].

2.2.4 Performance Metrics

When validating and assessing predictive models, is important to select the right evaluation metrics. In this study, both research questions correspond to regression problems, that is, the response variable is a continuous numerical outcome. As a result, we explore the following metrics:

- **Mean Absolute Error (MAE)**: measures the difference between values observed and values predicted by a model, penalising errors just proportionally to the size of the error itself [Bos14]. It is given in the form of a positive number in the same units as the original data and it is calculated using the 2.1 formula, where n represents the total number of observations, Y_{true} is the observed value and $Y_{predicted}$ is the predicted value by the model.

$$MAE = \frac{1}{n}\sum \mid Y_{true} - Y_{predicted} \mid \tag{2.1}$$

- **Mean Absolute Percentage Error (MAPE)** : is similar to MAE, but it is a relative to the true error. It is given in the form of a percentage positive number and it is calculated using the 2.2 formula. This metric has the advantage of being intuitive to interpret.

$$MAPE = \frac{100}{n}\sum \frac{\mid Y_{true} - Y_{predicted} \mid}{Y_{true}} \tag{2.2}$$

- **Root Mean Square Logarithmic Error (RMSLE)**: measures the difference between values observed and values predicted by a model, not penalising huge differences when both actual and predicted errors are larger. Also, it can be used when penalising more under estimates than over estimates. It is calculated using the 2.3 formula. This metric is valuable in our context because it is a ratio between predicted and true values that allows fair penalisation when comparing jobs with different scales.

$$RMSLE = \sqrt{\frac{1}{n}\sum (log(Y_{true}+1) - log(Y_{predicted}+1))^2} \tag{2.3}$$

- **Akaike Information Criterion (AIC)**: is an estimator of the relative quality of statistical models for a given set of data. AIC avoids overfitting by considering both goodness of fit and model complexity [MSA18]. The AIC does not have much meaning by itself. It is only useful in comparison to the AIC value for another model fitted to the same data set. A lower AIC means a model is considered to be closer to the truth. The AIC is calculated using equation 2.4, where k is the number of estimated parameters in the model and $\hat{L}$ is the maximum value of the likelihood function for the model.

$$AIC = 2K - 2ln(\hat{L}) \tag{2.4}$$

- **Bayesian Information Criterion (BIC)**: is similar to AIC but has higher penalisation for complex models. The BIC is calculated using equation 2.5, where n is the number of observations, k is the number of estimated parameters in the model and $\hat{L}$ is the maximum value of the likelihood function for the model.

$$BIC = ln(n)k - 2ln(\hat{L}) \tag{2.5}$$

Table 2.1: Comparing performance metrics when true and predicted values have the same MAPE

True	Predicted	MAPE	MAE	RMSLE
1	0.8	0.2	0.2	-0.04
10	8	0.2	2	-0.08
		1x	**10x**	**2x**

In order to compare the performances advantages, Table 2.1 shows an example of actual and predicted values when both predicted and actual values are short (one) and large (ten). For the same MAPE, that is, the same relative error, MAE says that the second prediction has *10 times* worse performance than the first one, while RMSLE says that it is only *2 times* worse than the first one.

Comparing AIC and BIC based on their definitions, their advantages are similar so they mostly agree in the model selected. However, AIC is widely used in the literature, so we pick it.

2.3 Conclusion

In this section we study the crowdsourcing concept, narrowing to micro-tasks platforms explaining their core concept, the fundamental entities and why is crowdsourcing a reliable solution. Then, we study diverse contributor and job related metrics. We verify that the *requester* has higher influence in the **effort** that the crowd must put in while the *contributor* influences the **throughput**.

In Sections 2.2.1 and 2.2.2 we study several external factors that can explain the crowd throughput. We consider that several metrics as the *user engagement*, *demographics*, *motivation* and *working days* can explain some unpredictable behaviours and are important metrics to have in mind either in Future Work and when establishing claims during this dissertation. In Section 2.2.3 we study previous work in throughput prediction and conclude that our work has two major differences: first, the throughput is measured in time (hours) instead of number of completed tasks; second, in our approach we use time series instead of traditional machine learning approaches. Considering the first research question, the effort estimation is unique as the state of the art in the field is sparse.

In the end, we consider that completion time is our target variable in both research question and that the speed and cognitive load as the most important metrics to answer **RQ1**, regarding the crowd effort estimation, and throughput and availability to **RQ2** regarding the throughput forecasting.

Chapter 3

Crowd Effort

In this chapter, we focus on the **crowd effort** in human hours, that is, the total time spent on task by contributors on specific tasks, with regard to **RQ1**. This labelling measurement allows to accurately estimating the time it takes to complete a job so that managers can estimate labelling costs and therefore, provide fair payments to the crowd.

In Section 3.1 we describe the data provided by *DefinedCrowd* and clean it in Section 3.2. While doing so, explain and create ways to measure human computation time and task complexity and measuring their correlations with our target variable. Then, in Section 3.3 we define the technologies used and the experimental setup to estimate the crowd effort. In Section 3.4 we create and evaluate a naive formula that explains the extracted features correlations. The results are discussed in Section 3.5.

3.1 Dataset Description

As mentioned in Section 1.3, this thesis is built with the collaboration of *DefinedCrowd*, that provides data collected by their product, *Neevo*. The *Neevo* platform has a similar structure to the micro-task crowdsourcing platforms described in Section 2.1.2. In what concerns the scope of this thesis, there are three main entities: the *Contributor*, the *Task* and the *Job*. The relations between the actors are shown in Figure 3.1.

A *Job* is a set of tasks that share the same subject, standard annotation type and language requirement. We use two different datasets: The *Job* dataset includes jobs of NET annotation type from 2017-02-10 to 2019-01-25 that have more than 1000 tasks; The *Task* dataset includes all the annotated tasks from the jobs previously described.

3.1.1 Job Dataset

The job dataset included 51 jobs with seven different language requirements: Dutch, English, French, German, Italian, Portuguese and Spanish. Table 3.1 describes the dataset initial attributes.

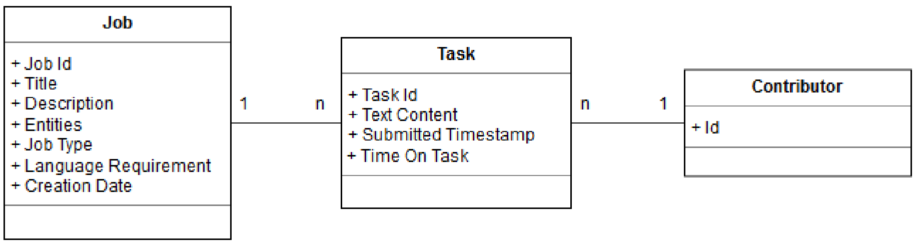


Figure 3.1: *Neevo* by *DefinedCrowd* crowdsourcing platform simplified class diagram showing the relations between *Job, Task and Contributor* and their main properties

3.1.2 Task Dataset

The Task dataset with up to 2.6 million tasks. Table 3.2 describes the dataset initial attributes.

3.2 Data Preparation

In this section, we explain our process to prepare the data to answer the established research questions from Section 1.4 and fulfill the objectives from Section 1.5. During this process we clean, explore the data and extract new features in order to understand the business and decide about the following steps. This step is crucial as the quality of the data and the amount of useful information that it contains are key factors to improve machine learning models accuracy [Ras15].

3.2.1 Handling Missing Values

When loading the data made available, one of the first concerns was dealing with missing data. Observing each metric, already described in 3.1 and 3.2 tables, we realised that *TimeOnTaskInMs* had *Not a Number* (NaN) placeholders. As our research questions imply measuring the completion time of the tasks, handling these missing values was mandatory.

There are diverse techniques to deal with missing values. The most known ones are eliminating samples, eliminating the feature or adding the missing values. In this case, eliminating the feature is impractical as it is our target variable. In order to choose between the other two techniques, further data exploration is needed.

We started by analysing how many tasks were affected, realising that 43% of the *Task* dataset had invalid values in the *TimeOnTaskInMs* field. Figure 3.2 helps to understand the invalid values distribution along the jobs. The jobs are ordered by creation date, and it shows that there is a point in time where task times commence to be recorded. Further investigation leads to 2018-11-27 as the first day with the time recorded.

In previous studies, tasks are compared to *ground truth* in order to understand if they fit in the normal job standards [HAGM15] [GKDD15]. These quality assessment methods allow one to measure quality attributes. For example, accuracy may be measured comparing outputs with a ground truth [DKC$^+$18] that was executed by experts in the same context. As an example, we found that there were tasks with timestamps before the job creation date. As the job creation date

Table 3.1: Job dataset initial attributes

Designation	Description
JobId	Job identifier
Title	Job title as view by the contributors
Description	Job description as view by the contributors
Entities	Set of terms that are related to some subject and represent the different topics that the contributors annotate
LanguageRequirement	The language that the contributors must be proficient to complete the tasks
CreationDate	Date in which the job was published on the platform
JobType	Job standard annotation type (in this case is only NET)

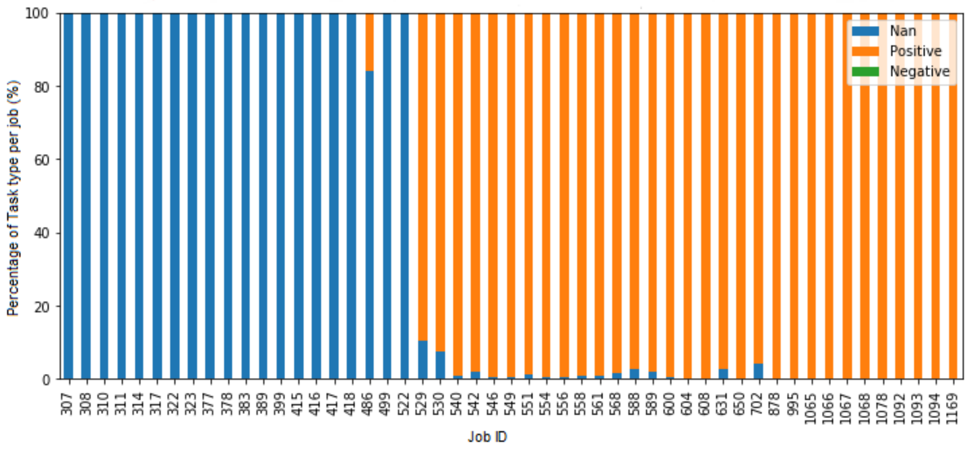


Figure 3.2: Percentage of tasks with *NaNs, Positive or Negative* time on task, per job

is the moment at which we start creating tasks, we consider that these tasks do not fit in the normal job standards and must be removed.

Observing the same Figure (3.2), we also state that there are instances with negative values. Exploring this result, we conclude that there are 79 negative time on task that represents 0.0031% of the dataset. After a short analysis, we conclude that there is no relation among tasks as they belong to different jobs, different contributors and different days. Therefore, we can conclude that these negative samples are measurement errors and should be removed.

Removing the tasks with *task execution times* leads to an inconsistency between the *Job* and *Task* dataset. As a result, we defined jobs that contain more than 50% of NaN time on task as **invalid jobs**. Those jobs and the correspondent tasks were removed as it is not possible to predict the job total execution time. Other jobs that are valid, may still have NaN time on task, so we removed them too as those tasks were not reliable.

After this analysis, we proceed to inspect other attributes and concluded that there were some samples with the *Text Content* empty. As every task should have at least one sentence to be annotated, we concluded that these tasks had a reading error and should be removed. Those tasks

Table 3.2: Task dataset initial attributes

Designation	**Description**
TaskId	Task identifier
JobId	Job identifier
ContributorId	Contributor identifier
TextContent	Set of sentences to be annotated by the contributor
TaskSubmitedTimestamp	Timestamp when the contributor submits the task
TimeOnTaskInMs	Time elapsed from the moment that the task is visible to the contributor and it is submitted it in milliseconds

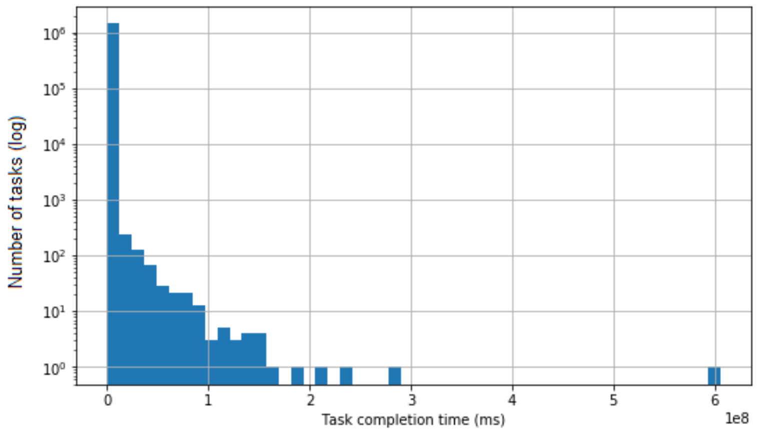


Figure 3.3: Initial distribution of *TimeOnTaskInMs* feature

corresponded to 0.15% of the dataset.

Altogether, we finish up this Section with 24 jobs between 2018-02 and 2019-01 and 955,810 tasks.

3.2.2 Time on Task Exploration

To measure the total human effort for enriching a given text corpus, we need to get accurate measurements of time on task for each individual task by each contributor. This metric, *TimeOnTaskInMs*, is observed in order to analyse which factors influence it the most.

Figure 3.3 shows the time on task distribution considering the *Task* dataset. It is possible to observe that the distribution has a long tail, with minimum and maximum values that may be extreme, when compared to the remaining samples in the distribution. We also observe that although most of the tasks have times around the mean (52 milliseconds), the standard deviation is significant (18 minutes) as several tasks can reach up to 1,680 hours to be completed. Finally, the large standard deviation also corroborates the fact that the task completion times have significant variance.

We were expecting a distribution close to a bell shape, with a long tail representing tasks with more information to process. As a result, we hypothesise that unreasonably short/long completion

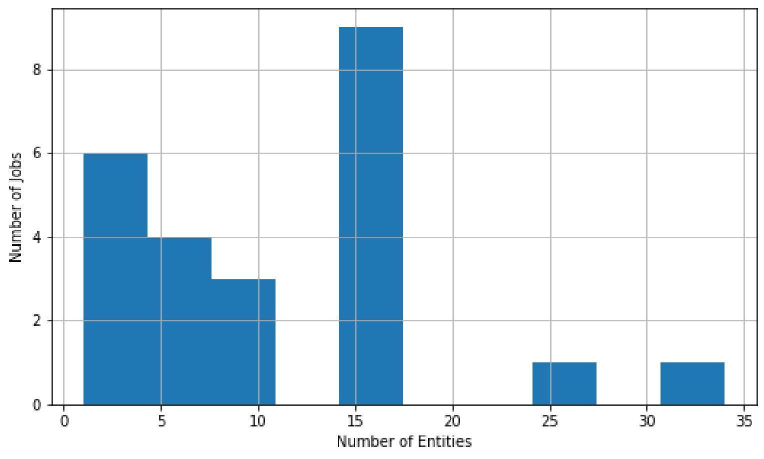


Figure 3.4: Number of entities distribution over jobs

times (see Section 3.2.4) and tasks with different **cognitive load** affect the current distribution shape.

Cognitive load (see Section 2.2.2), is the combination of task complexity and clarity [SC94] and affects directly the task performance [FKTC13]. In Section 2.2.1, Hirth et al. define *speed* as a metric to assess contributor's poor performance. The speed is a measure that incorporates the length of the input a contributor needs to go through it, given that it is measured in tokens read by the contributor per minute. With this in mind, we create the hypothesis that the speed variation may explain the task cognitive load. On the other hand, considering that we are dealing with NET jobs, we can also consider the *number of entities to tag* as a complexity measure. Our hypothesis is that the more entities there is to tag, the more complex is the task.

Summing up, we have created the hypothesis that *contributor speed on task* and *number of entities* are two factors that influence the time a NET job takes to complete. During the next sections, we assess our hypothesis.

3.2.3 Feature Extraction

To answer **RQ1** regarding the crowd effort estimation per job, we must extract new features, as previously identified in Section 3.2.3, related to the *cognitive load* (number of entities) and *speed on task*.

Firstly, the number of entities is extracted at the job level, counting the number of entities on the *Entities* metric (see Table 3.1). Observing Figure 3.4, it is possible to deduce that, usually, jobs have between one and fifteen entities to annotate with few cases of up to 30 entities to annotate. We verify that the distribution is not balanced, given that we have few observations. This fact can influence the relations between the number of entities and other variables.

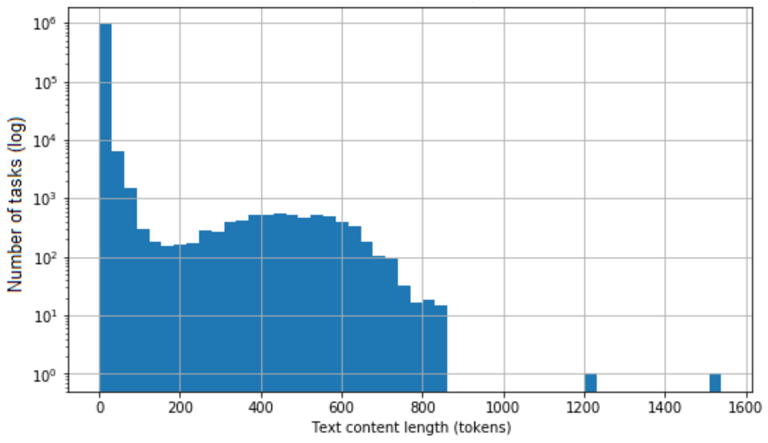


Figure 3.5: Distribution of text content length in tokens over tasks

Secondly, the *speed on task* is calculated using the formula

$$Speed(tokens/min) = \frac{NumberOfTokens}{TimeOnTaskInMinutes} \tag{3.1}$$

The *speed* feature is valuable as it offers a more accurate approach to distinguish between what may be implausible instances of time on task instances. While shorter/longer completion times can be justified by short/large text contents, the speed creates an abstraction to the real task size. As a result, it is possible to create a range of feasible speeds [HSH⁺14] from what is humanly possible that explain the task cognitive load and discard the tasks out of range (see Section 3.2.4).

The *speed* formula implies the extraction of the *text content length in tokens* feature. We decide to use tokens instead of characters so that it is possible to compare with the literature. The number of tokens distribution per task over jobs can be observed in Figure 3.5. We observe that the number of tokens can vary between one and almost 1,600 tokens per Text Content, with most common length varying between one and 800 tokens.

With the *Text content length in tokens*, it is possible to apply the speed formula. The calculated mean speed on task is 69 tokens/min, under the 200 tokens/min stated by Hirth et al. [HSH⁺14]. This divergence is justified by disparate definitions of speed. While Hirth et al. separate the reading from the answering speeds, we do include both speeds into the task completion speed.

Figure 3.6 shows the speed in tokens per minute distribution over tasks. We observe that the distribution has a long tail of unfeasible speed on task's, both too slow or too fast. In the next Section (3.2.4) we cover how to deal with these values.

3.2.4 Detection and Treatment of Outliers

An outlier is a data point that deviates markedly from the others in a sample [Bos14]. These inconsistent or erroneous data not only skews descriptive measures such as mean and variance but also bias our model causing distorted predictions [Bos14].

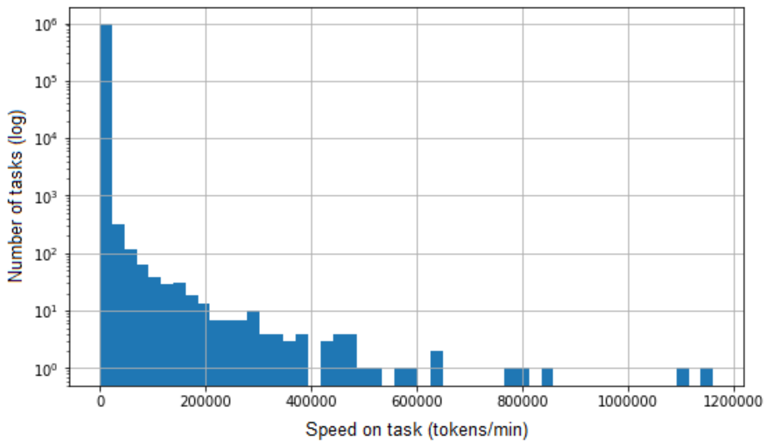


Figure 3.6: Speed on task (tokens/min) distribution over tasks

As we have suggested in Section 3.2.3, the *speed on task* may be a valuable feature to classify unreasonable tasks when their *time on task* is beyond what is human plausible. On one hand, *too fast* speeds imply, for an average text content length in tokens, too short task completion times. These speeds may be a consequence of contributors not be fully committed to creating quality results. Some examples:

- contributor did not read the text content
- contributor read the text content partially
- contributor did not answer to the task

On the other hand, **too slow** speeds imply, for an average text content length in tokens, too long task completion times. These speeds may be a consequence of contributors leaving tasks open, without completing it. As a result, those tasks do not reflect the true effort needed to complete the tasks because we may be considering not only the reading and answering time but also contributor idle time.

Herein, we apply the commonly used interquartile range (IQR) strategy to filter out outliers. IQR is a measure of the spread of a distribution as it is the difference between the 75th (i.e third quartile, Q3) and 25th (i.e first quartile, Q1) percentiles [Dow12]. The IQR can be used to identify outliers, defined as observations that fall below $Q1 - 1.5 * IQR$ (minimum fence) or above $Q3 + 1.5 * IQR$ (maximum fence) (see Figure 3.7 [1]).

Applying this technique, the new range of admissible speeds, i.e. the minimum and maximum fences, are 0 tokens/min and 224 tokens/min. The maximum fence is reasonable as it falls around the average reading speed, that is, tasks that do not have words to be annotated or even extremely straightforward tasks. On the other hand, this minimum speed allows unreasonable behaviours as extremely long time on task, so we decided to define a reliable minimum speed of *5 tokens/min*

[1]https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

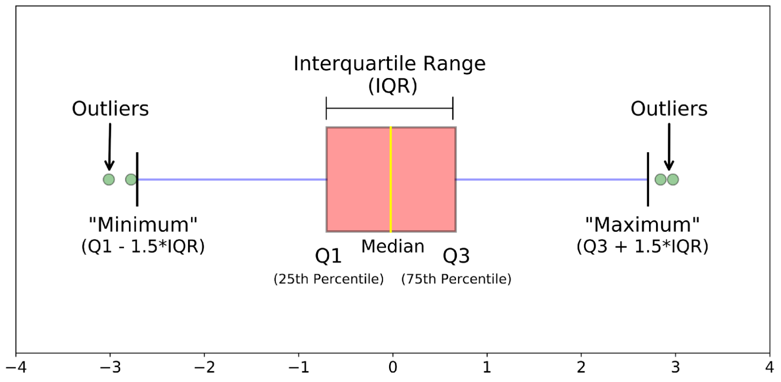


Figure 3.7: Outliers definition according to IQR

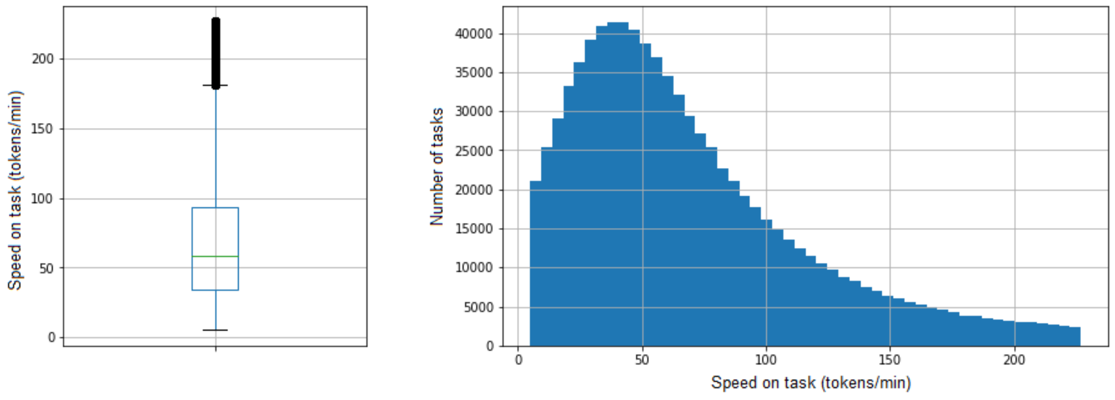
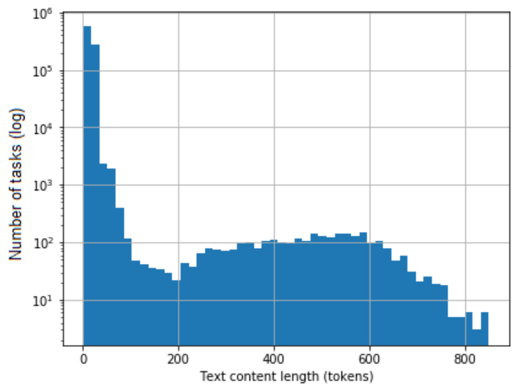


Figure 3.8: Boxplot (left) and Distribution (right) of speed on tasks (tokens/min) over tasks, after discarding outliers based on IQR

and discard tasks with lower speed.The minimum speed reflects a real-world minimum value considering task comprehension, answering consideration or even a contributor learning curve. After truncating the dataset to remove tasks with task completion speed lower than 5 tokens/min, we have discarded 11.76% of the *Task* dataset.

The new speed distributions are depicted in Figure 3.8. Compared to 3.6, we can observe a bell shape with a right long tail, as we were expecting. Also the mean speed decrease from *197.2 tokens/min* to *69.7 tokens/min* and the variance from *4,191.9 tokens/min* to *47.3 tokens/min.*

Figure 3.9 also shows the features text content length in tokens and task completion time after discarding the outliers. Comparing the text content length in tokens with the previous distribution (see Figure 3.5), we observe that excessive long text contents were removed and that the outliers were equally distributed over the text content lengths. Comparing task completion time with the previous distribution (see Figure 3.3), we conclude that long completion times were removed successfully and that the new shape is similar to a power-law distribution.



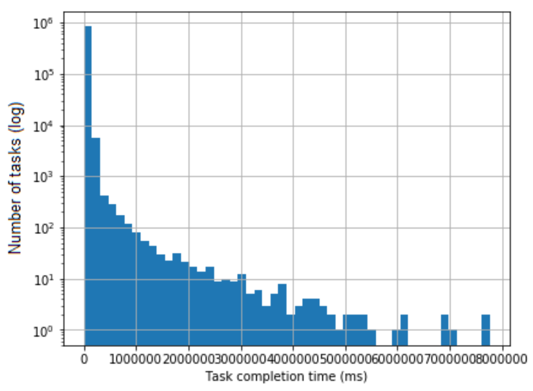


Figure 3.9: Text content length in tokens (left) and Task completion time (right) distributions over tasks, after discarding outliers

3.3 Experimental Setup

When assessing the **crowd effort** that given job may require, there are three factors having the most influence: the *tasks cognitive load*, the *length of the information to be processed* and the *contributors characteristics* as expertise level. After studying how these factors influence the effort that a contributor must put in to complete a task, we create a naive formula to estimate the crowd effort to complete a job.

This experiment dataset is based on the job dataset (see Section 3.1.1) and uses the features extracted regarding the job cognitive load and length of the information to be processed. The resulting dataset has 24 data points that correspond to the number of jobs in the dataset job. The extracted features are:

- *CrowdEffortInHours* is the sum of tasks completion time, converted to hours;
- *JobTextContentLengthInTokens* is the sum of tasks size of the information to be processed in tokens;
- *NumberOfEntities* is the number of entities to tag in the tasks.

The dataset description can be found in Table 3.3. Our target variable is the *CrowdEffortIn-Hours* and the others are the predictors.

To assess a predictive model's performance, it is important to create adequate conditions in which the model is tested. In this case, our goal is to establish a baseline performance for estimating the crowd effort in hour through a naive formula. As a result, we do not want to generalise our formula so we do not hold a test set. This formula is a baseline for further iterations described in Future work. In Section 3.4, we assess our model with the train set, i.e. the total dataset described in 3.3 using the MAPE performance metric (see Section 2.2.4).

Table 3.3: Job dataset variables description for crowd effort estimation in hours

Designation	**mean**	**std**
CrowdEffortInHours	241.58	381.67
JobTextContentLengthInTokens	598745.22	880159.35
NumberOfEntities	11.25	8

3.3.1 Technologies Used

Our solution was developed in Python Programming Language [2], a clear and object-oriented programming language. Among other advantages, its open-source nature led to the rising of an ecosystem for scientific computing [PGH11]. This ecosystem has several Python libraries that simplify numerical computing, data analysis, interactive visualisation and machine learning. The following list contains some of the Python packages used:

- Numpy [3] for scientific computing
- Pandas [4] for high-performance, easy-to-use data structures and data analysis tools
- Matplotlib [5] for 2D plotting
- Scikit-learn [6] for data mining, data analysis and machine learning tools
- Statsmodels [7] for statistical models, statistical tests and statistical data exploration

3.4 Crowd Effort Results

We start by assessing our hypothesis that the cognitive load is an important factor to crowd effort estimation. To assess our hypothesis, we have measured the correlations between our target variable and the features created. The *correlation coefficient* is a standardised measure of the association or mutual dependence between two variables. Its values range from $-1 to +1$, indicating strong negative relationship, through zero, to strong positive association [MWH98]. Our target variable is the cumulative sum of time on task for all tasks within a job. The *total task length* is the cumulative sum of all text to be processed in all tasks and it observed a correlation coefficient of *0.85* with the target variable. The *number of entities* to tag is the number of entities that a contributor may tag along the tasks with the same job, defined before the workflow start and it observed a correlation coefficient of *0.57* with the target variable. Although this correlation apparently it is not strong enough, we have stated in Section 3.2.3 that it is related to the unbalanced distribution. We expect that a balanced distribution would create a stronger correlation.

[2]https://www.python.org/
[3]https://www.numpy.org/
[4]https://pandas.pydata.org/
[5]https://matplotlib.org/
[6]https://scikit-learn.org/stable/
[7]https://www.statsmodels.org/stable/index.html

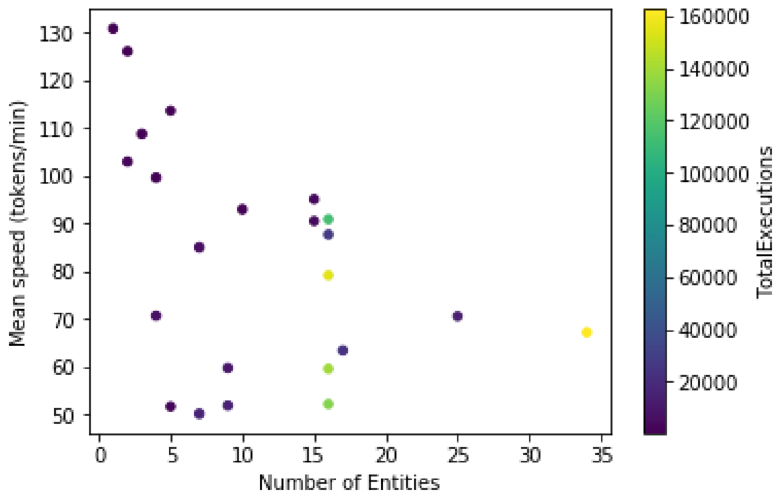


Figure 3.10: Speed (tokens/minute) versus the number of entities per job

$$CrowdEffortInHours = \frac{TotalTasksLength}{JobMeanSpeed} \tag{3.2}$$

The knowledge gained during this chapter allowed to chose the *total task length*, *number of entities* and *speed on task* as the most important factors that we then use to build our baseline formula. We base our formula on the *speed on task* leading to Equation 3.2.

$$CrowdEffortInHours = \frac{TotalTasksLength}{a + b * NumberOfEntities} \tag{3.3}$$

However, we have seen before that different jobs tend to overall different speeds, because of their differences in cognitive load. Observing Figure 3.10, we can observe that there is a correlation between the cognitive load considering the number of entities and the speed on task. Given the number of entities unbalanced distribution, the correlation is not clear to establish. Even though in our context, as the number of entities increase, it is expected that the job mean speed decreases. As a result, we have decided to use linear regression to calculate the speed based on the number of entities, leading to the final formula from equation 3.3, where a is the intercept and b the slope.

We use a Grid Search method to find the optimal a and b values, i.e. the values that minimize our MAPE, the evaluation metric chosen for this problem. Considering that we only have two hyper-parameters to be optimized and that we know the extent of their values, the Grid Search technique allows a straight-forward solution for hyper-parameter tuning. We define the following ranges a = [50 55 60 65 70 75 80 80 85] and b = [0.2 0.4 0.6 0.8 0.85 0.9 0.95 1 1.1 1.2] leading to *90* combinations. The optimal performance of $MAPE = 18.99\%$ was achieved with $a = 65, b = 0.85$ combination.

This result already allows putting an upper boundary on total expected effort, which is useful because it gives an expectation of length more accurate than the total number of tasks. Further discussion can be found in Section 3.5.

3.5 Discussion

In regard to **RQ1**, we want to estimate a NET job crowd effort before a job starts. Concerning the data preparation, we have defined a range of admissible speed on task's. As an example, a task with a high cognitive load is expected to have longer completion times. However, the task completion time can vary greatly considering the contributor's profile, e.g. proficient contributors can have higher speeds while new contributors may be influenced by the learning curve leading to slower speeds. This range of admissible speeds is also valuable to define abnormal tasks, i.e. tasks with speeds higher than the maximum admissible speed on task indicates that the task may have been completed too fast to the demanded cognitive load. As a result, the quality of the task may be compromised. Finally, we consider this measurement essential when dealing with crowdsourcing quality measures, as it identifies, in real time, malicious contributors.

The baseline model achieved to estimate the crowd effort put in to complete a job already allows to define an upper boundary on effort, which is useful because it gives an expectation of length more accurate than the total number of tasks. As an example, considering the same number of documents to label, the fact that the documents are twitter sentences or news articles have a huge impact on the total effort. Studies pinpoint that tweets have an average of 9.1 tokens while news articles have an average of 1106.79 tokens [JLZ+11], that is, **110 times more** tokens than tweets. Thus, considering the same amount of documents, the costs of annotating news articles will be much higher than annotating tweets. Our goal is to further develop this solution (see Section 5.4) so that we can have a more informed result.

We believe that measuring the crowd effort is a big step ahead in the crowdsourcing community. It is possible to extract new features according to each job type and create accurate crowd effort predictions that can guide future research about estimating job pricing and consequently, fair contributors payments.

3.6 Conclusion

In this chapter, we discussed the provided data from its context until its results. We extracted several features as the *number of entities*, *speed on task* and *text content length in tokens*. We verify that these features correlates with our target variable *time on task*.

Then, we defined our experimental setup presenting the technologies used. In the end, we both model and evaluate **RQ1** that achieves 19% MAPE when predicting the crowd effort put in to complete a job when using a naive formula. This performance means that the cognitive load measured in number of entities to tag and the length of the information to be processed are relevant factors to estimate the crowd effort.

Chapter 4

Crowd Throughput

The eligibility and availability of the crowd affects greatly the predictability of working contributors on the job. First, jobs go through demographic requirements that select subsets of contributors. Then, the open nature of a crowdsourcing platform influences availability and stability of contributor throughput over time. Contributors are not expected to complete micro-tasks equally or in predefined hours. This factor allied to context specific aspects that we study throughout this chapter, lead us to believe that establishing a deterministic completion timeline before a job starts can be unfeasible in our case.

This chapter goal is to answer **RQ2** regarding the **crowd throughput forecasting**. We start by preparing our data so that we can hold hourly observations in Section 4.1 and exploring the time series dataset in Sections 4.1.1, 4.1.2 and 4.1.3. In Section 4.2, we define the modelling steps: in Section 4.3 we explain the experimental setup, including the investigated models and the test design; In Section 4.4 we model and evaluate our research question, fitting the models in Section 4.4.1; assessing the models in Section 4.4.2; Finally we evaluate the models regarding the business perspective in 4.4.3. In the end, we discuss our results in Section 4.5.

4.1 Data Preparation

Measuring and correctly evaluating the current throughput for an ongoing job allows managing expectations when these timelines are unpredictable. As we have already stated in Section 2.2.3, previous work on throughput prediction use machine learning regression models. However, considering that we have a collection of tasks obtained from sequential measurements over time [EA12], it is possible to formulate our problem as a **time series forecasting problem**.

To analyse our crowd throughput temporal dimension, we sort the tasks by periods of **one hour**. A one-hour time span was also used in the past by Difallah et al. [DCD+15]. The new time series dataset with hourly observations has two indexes, the *job id* and the *period*. One period corresponds to one hour of observations, so we need to aggregate the tasks that were completed

Table 4.1: Extracted features related to throughput, number of completed tasks, number of contributors and size of the information to be processed

Designation	**Definition**
ThroughputInHours	Sum of task completion times of completed tasks within the same hour
TotalTextContentLengthInTokens	Sum of total tokens of completed tasks within the same hour
TotalContributors	Number of different contributors that were completing tasks within the same hour
TotalExecutions	Number of completed tasks within the same hour
CumulativeCrowdEffortInHours	Sum of task completion times of completed tasks from the job beginning until the current period
CumulativeTextContentLengthInTokens	Sum of total tokens of completed tasks from the job beginning until the current period
CumulativeContributors	Number of different contributors that were completing tasks from the job beginning until the current period
CumulativeExecutions	Number of completed tasks from the job beginning until the current period

within the same hour. For example, if we have three tasks, one submitted at 14h10, other at 14h30 and other at 15h20, the first two belong to the first period and the third one to the second period. As a preprocessing consequence, spans of time without executions exist but with the indication that no work has been completed.

The extracted features are described in Table 4.1. The table has two distinct attributes: the total and the cumulative sums. The total sums aggregate the tasks information within the same hour, while the cumulative sums aggregate the tasks information from the job beginning until the current period.

Finally, we consider **24 hours** as the minimum job time span. This time span represents one day cycle, assuming that jobs may have daily seasonality. After removing jobs having time spans shorter than 24 hours, we have a remainder of 18 jobs for this part of our study. The shortest job has 50 periods (nearly two days) and the longest one 671 (nearly one month).

4.1.1 Throughput Exploratory Analysis

In this section, we perform an exploratory time series analysis considering our target variable, the crowd throughput in hours (*ThroughputInHours*). Table 4.2 shows some descriptive statistics about our target variable. We verify that the hourly throughput may vary between 0 and 17.18 hours, but the mean is 1 hour, meaning that our distribution is right skewed. Also, the variation is significant, considering that it is almost two times the mean value.

Our goal is to find patterns or correlations between different time points so that we can justify the use of time series to forecast the foreseeable throughput considering our recent past. As a result, in this section we analyse if time series have patterns or abrupt changes.

Table 4.2: *ThroughputInHours* descriptive statistics

Designation	**min**	**max**	**mean**	**std**
ThroughputInHours	0	17.18	1	1.93

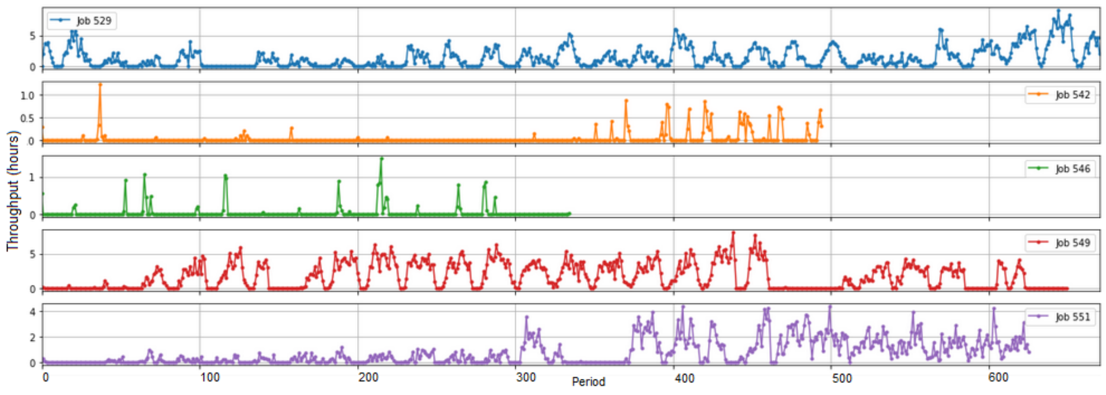


Figure 4.1: Sample of five time series with different behaviours and lengths

It is common to distinguish between four time series data patterns [MWH98].

- *Horizontal* pattern exists when the data values fluctuate around a constant mean;
- *Seasonal* pattern evaluates if there is a regularly repeating pattern of highs and lows related to calendar time such as seasons, quarters, months, days of the week and so on;
- *Cyclical* pattern exists when the data exhibit rises and falls that are not of a fixed period;
- *Trend* is a measure that evaluates if the time series tend to increase (or decrease) over time.

As our goal is to create a model that can adapt to any new NET job in the platform, we are looking for similarities among time series. Figure 4.1 shows a sample of five out of our 18 time series. It is possible to observe that there is no apparent similarity among them. As an example, the second and third time series (jobs 542 and 546, respectively) have no apparent trend or seasonality while the fourth time series (job 549) shows daily seasonality. The remaining plots can be found in Appendix A.1.

Time series plots do not provide enough graphical insights about their inherent patterns. Therefore, in this section we apply several methodologies that refer to the time series graphical data exploration.

We start by using *decomposition* as a tool to split the time series into trend, seasonality and irregularity using the additive method from Equation 4.1.

$$Observation = Trend + Seasonal + Residual \tag{4.1}$$

Figure 4.2 shows an example of two time series decomposition with 24 periods frequency, that is, assuming daily seasonality. The remainder decomposition plots can be found in Appendix A.2.

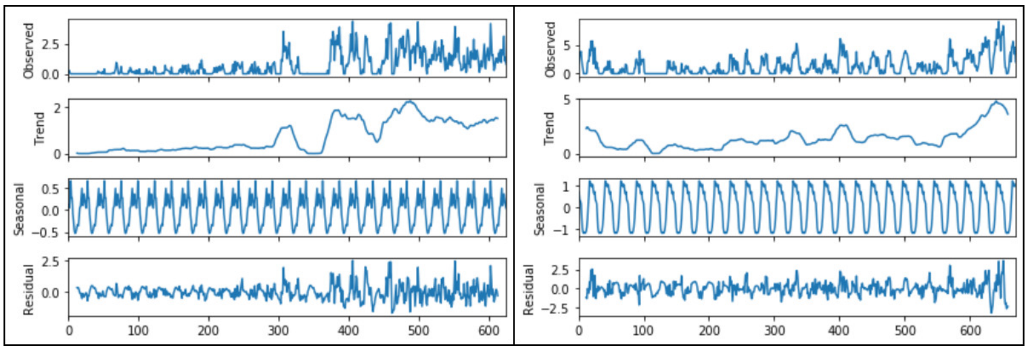


Figure 4.2: *Job 551* (left) and *job 542* (right) *decomposition* plots. The *Observed* time series is divided into *trend*, *seasonal* and *residual* time series using the additive method and assuming daily seasonality

Each decomposition plot has the observed data, the captured trend and seasonal cycles and the residuals, that is, what the model cannot explain. As we are dealing with real data, we verify that our data has noise and the trend and seasonal patterns are different per each job.

According to Makridakis, when preparing the time series data, we can apply transformations in order to stabilise the variance and difference the data to obtain stationary time series [MWH98]. Time Series are stationary if their basic statistics do not change over time [BB12]. Difference the data, that is, subtracting the value of an earlier observation from the value of a later observation, allows to remove linear or quadratic systematic trends from non-stationarity series. Time series that can be transformed into stationary time series applying differencing are called *difference-stationary* time series.

We verify if our time series are stationary or difference-stationary using the Dickey-Fuller unit root test [DF79]. This test is a hypothesis test, that is, we establish a null hypothesis that the time series is difference-stationary and the alternative hypothesis that the time series is stationary. Our significance value is $\alpha = 0.05$, so we reject the null hypothesis if the *p-value* $< \alpha$. Summing up, if *p-value* $<$ *0.05* the time series is stationary. Applying this test we have concluded that eight out of eighteen time series are not stationary. Then, we have apply a first order differencing to eliminate job's linear trend. When applying again the Dickey-Fuller test to these newly transformed time series, we observe that all time series are difference-stationary.

Figure 4.3 shows a sample of first order difference series. We observe that the differenced time series have an horizontal pattern as their differenced throughput varies around a constant mean of zero. Comparing with Figure 4.1, we observe the same periods of inactivity and we can differentiate clearly between periods of significant throughput variation. Two interesting time series are the job 549 and job 551. Observing Figure 4.1, job 549 has similar throughput variation over time while job 551 has an increase of variation. Observing Figure 4.3 we confirm our claim, verifying that job 549 varies between [-2.5, 2.5] over time while job 551 starts with short variations around the mean and then increases until variations between [-2.5, 2.5].

However, some difference time series may still have seasonality. A strategy to identify those

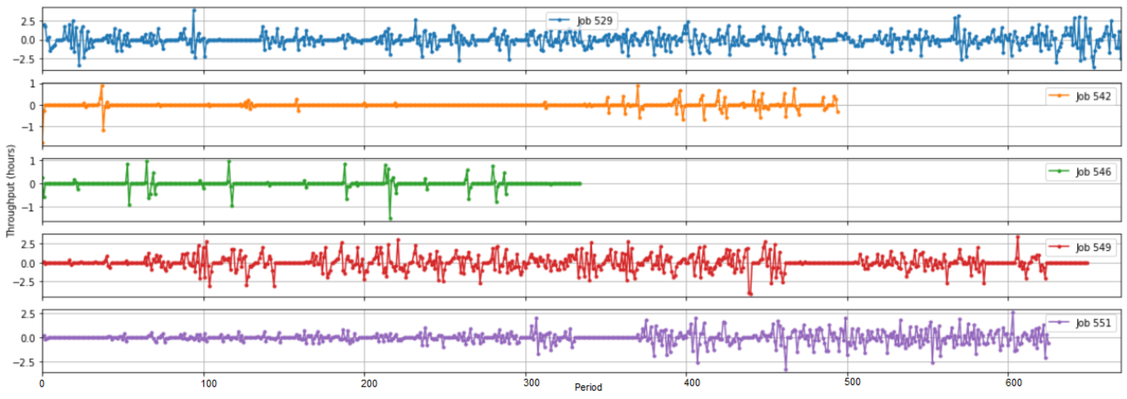


Figure 4.3: Sample of five difference time series with hourly steps

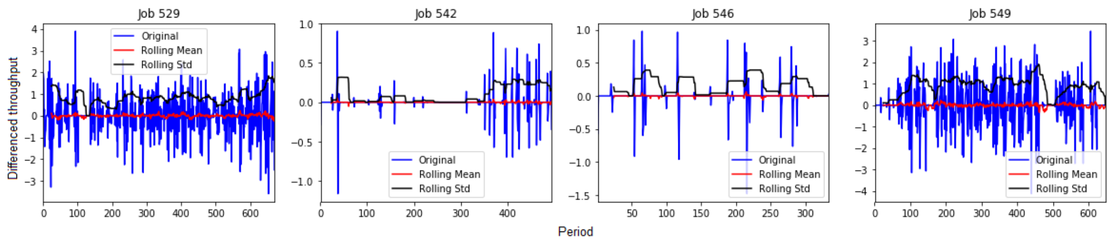


Figure 4.4: Sample of four difference time series with rolling mean and standard deviation applied

patterns in time series is to take a *rolling average*, that is, for each time point, taking the average of a *window*, i.e. a collection of the points on either side of it. This is a *smoothing* method, as it reduces the random variation, i.e. noise. Considering Figure 4.3, it is possible to identify trend patterns. As a result, we apply a rolling mean and standard variation of 24 data points, i.e. one day. Figure 4.4 shows that, after applying the rolling window, the mean varies around zero over time, meaning that the method smoothed out the noise and seasonality from the time series. See Appendix A.3 for complete visualisations.

Another graphical data exploration method is to verify the *Autocorrelation Function* (ACF) and *Partial Autocorrelation Function* (PACF) plots. These plots allow to understand the relation between the different time lags. An ACF plot represents the autocorrelation of the series with lags of itself. It is also useful to identify stationary time series as they quickly drop the autocorrelation to zero , while non-stationary time series do so more slowly. A PACF plot represents the amount of correlation between a series and a lag of itself that is not explained by correlations at all lower-order lags. This function can capture correlations of residuals and the time series lags.

Figure 4.5 shows ACF and PACF plots of a sample of five time series with a maximum lag of 50 steps, that is, it only shows the autocorrelations as far as 50 periods. Confidence intervals are drawn as a blue cone, set to a 95%, suggesting that values outside of this cone are very likely a correlation. A 95% confidence interval means that it will contain the true value of the population parameter with probability 95% [MWH98]. We can state that the plots are very different among

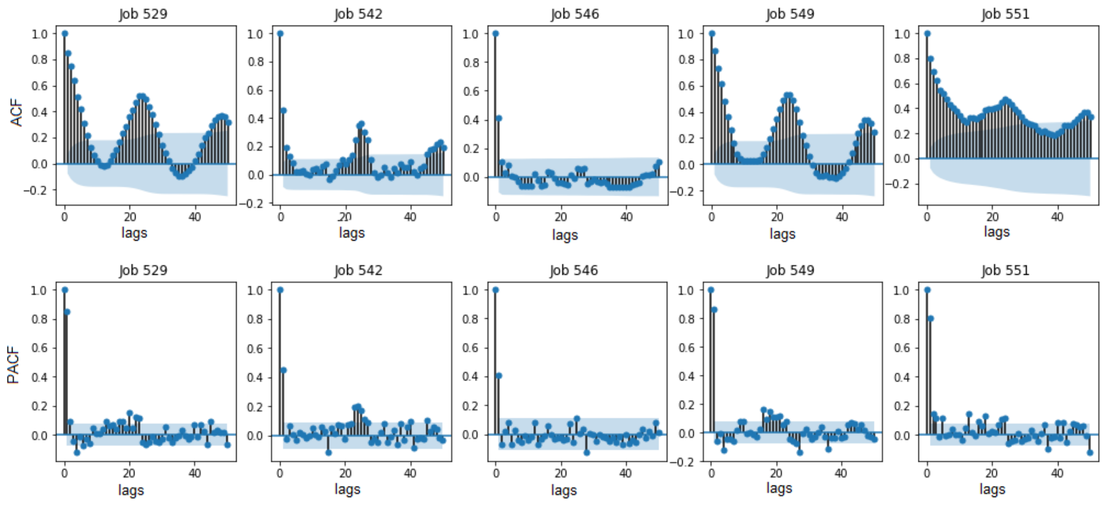


Figure 4.5: Autocorrelation (first row) and Partial autocorrelation (second row) plots with 50 lags maximum, from a sample of five time series. The blue cone represents the 95% confidence interval.

time series, reinforcing that it is difficult to find a common pattern that can fit precisely among them.

4.1.2 Job's Real-World Length Exploratory Analysis

Previously, we have analysed the throughput variation over time, trying to find similarities between time series and the correlation between periods. In addition to the time series exploration, we study other dimensions of our problem in order to further understand our time series.

For visualisation purposes, from this point on, we add up the hourly periods into daily observations. In this section, we look into the crowd throughput of each job. We start by relating the total crowd effort with the real-world job length. The total crowd effort corresponds to the sum of the all job tasks completion time, and it is measured in hours. On the other hand, the real-world job length is the number of hours that a given job takes to complete considering the first and last completed task timestamps.

When relating the computed crowd effort with the real-world job length, we are establishing boundaries of real-world length, in order to manage completion timelines expectations. In order to quantify these boundaries, we apply the Equation 4.2. This formula says that, if the computed crowd effort is higher than the job real-world length, the job finishes faster than the time it needs from the crowd, that is, there are factors as the number of contributors available and their capacity to contribute that influence the completion timelines.

$$\frac{JobRealWorldLength - JobCrowdEffort}{JobRealWorldLength} * 100 \tag{4.2}$$

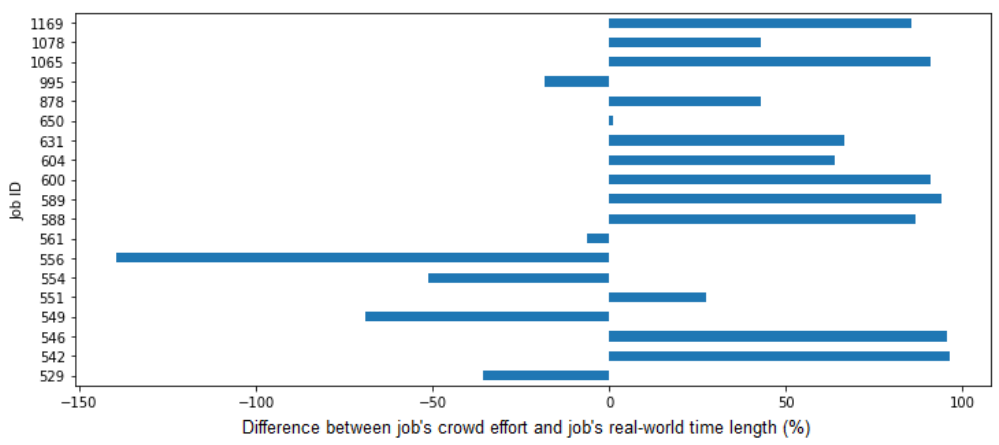


Figure 4.6: Difference in percentage between Job's computed crowd effort and the job's real-world length in hours. Difference equal to zero means equal crowd effort and real-world length, negative difference mean higher crowd effort than time took to complete the job and positive differences the other way around

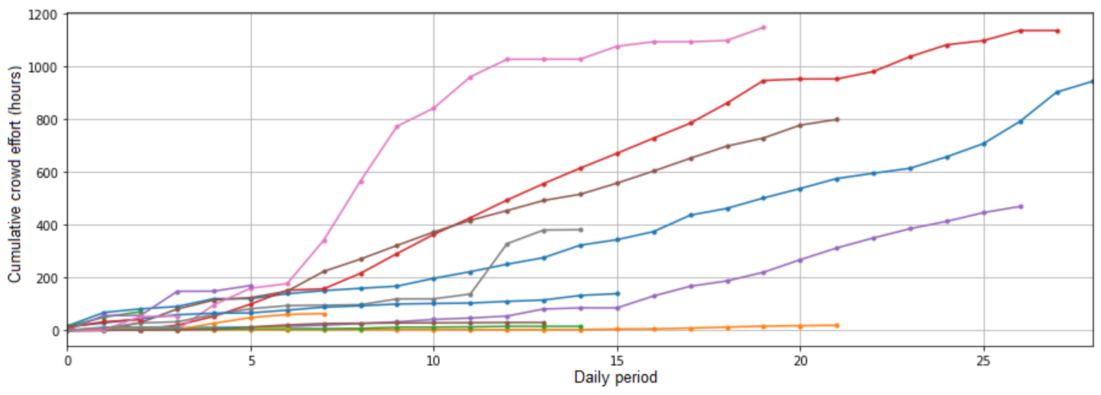


Figure 4.7: Cumulative crowd effort in hours considering daily observations, per each job

Figure 4.6 shows us a comparison between the computed crowd effort and the real-world job length using the 4.2 formula. We conclude that jobs can complete from 2.5 times faster to 2 times slower than the time it needs from the crowd. For example, if we compute 60 hours in crowd effort, we can expect that the job finish between 24h (1 day) and 120h (5 days) in real-world time.

In order to evaluate the crowd effort over time, Figure 4.7 shows the throughput evolution curves per job with daily observations. It is possible to observe that the curve varies greatly among jobs. Also, jobs with more than 400 hours in crowd effort, usually take more than 15 days to complete. We conclude that not all jobs have produced throughput equally over time. Consequently, jobs with the same crowd effort in hours will have different real-world job lengths. As an example, we have two jobs with 1100 computed crowd effort, one finished in 19 days and another in 27 days.

We create the hypothesis that the number of active contributors affects the daily throughput,

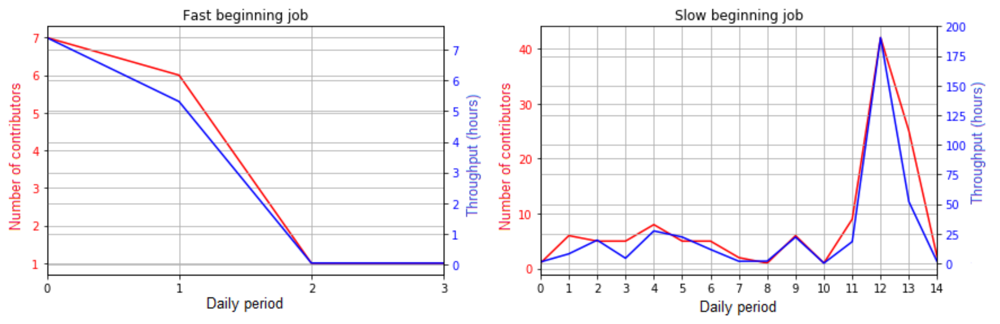


Figure 4.8: Fast beginning (left) and slow beginning (right) jobs comparing number of contributors (red) and throughput in hours (blue), per day

and consequently the real-world job length. Figure 4.8 shows two jobs: one with *fast beginning*, that is, the initial throughput is large and then it decreases, and other with *slow beginning*, that is, the other way around. We can state that our hypothesis is correct, so the throughput depends on the number of contributors executing tasks. The slow beginning job is a good example of a job with a considerable number of contributors and computed crowd effort in hours, that shows that in the beginning, the evolution is slow, and as soon as the job gets an increase in the number of active contributors, the evolution curve speeds up.

4.1.3 Mean Contributor Throughput Exploratory Analysis

As we have seen in previous sections, the daily crowd throughput in hours is highly dependent on the number of contributors completing tasks in that day. However, it is important to understand if contributors have similar throughput or if jobs do have an influence on the contributor's throughput. In order to test this hypothesis, we have calculated the mean daily throughput per contributor, that is, the mean effort put in by the contributors in a given day per job. This measurement follows the 4.3 formula.

$$MeanDailyThroughputPerContributor = \frac{DailyThroughput}{DailyNumberOfContributors} \tag{4.3}$$

Table 4.3 summarises the mean daily throughput per contributor and daily number of contributors. We verify that the minimum values are zero, meaning that there are jobs with inactive days. The maximum number of contributors was *81* in one day, while the average is *12.3*. The daily number of contributors varies greatly among jobs, given that the standard deviation is *14.04* and it is greater than the mean. The *MeanDailyThroughputPerContributor* statistics can be validated with Figure 4.9 that shows both mean daily throughput per contributor distribution and boxplot. The distribution is right skewed and with most of the values varying between 0.5 hours/contributor to 2 hours/contributor, as it can be verified by the boxplot, and the average contributor works around 1.34 hours/day.

Table 4.3: *MeanDailyThroughputPerContributor* and *DailyContributors* features descriptive statistics

Designation	min	max	mean	std
MeanDailyThroughputPerContributor	0	5.95	1.34	1.10
DailyContributors	0	81	12.30	14.04

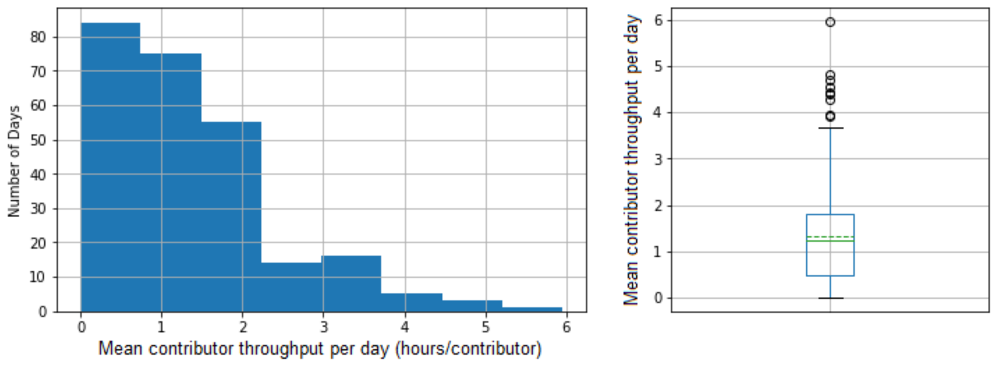


Figure 4.9: Mean contributor throughput per day distribution (left) and distribution boxplot (right)

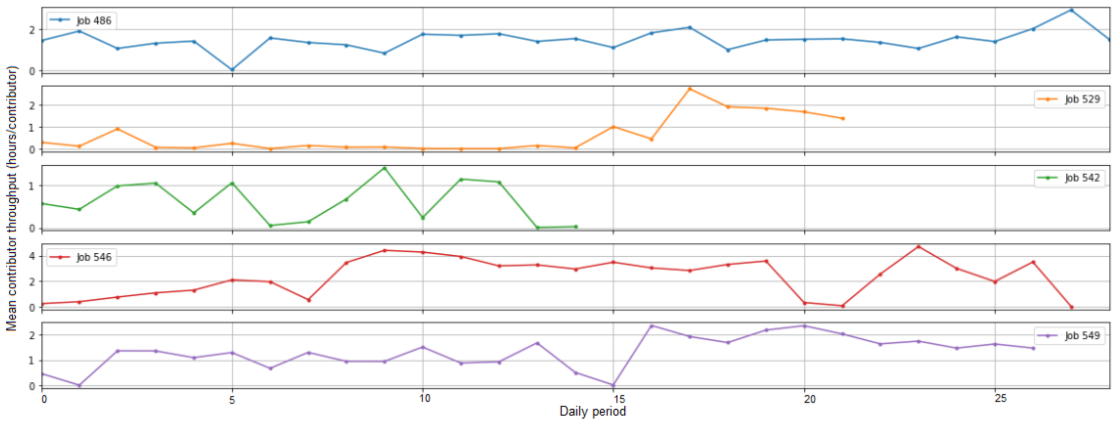


Figure 4.10: Mean daily throughput per contributor over a sample of five jobs

Figure 4.10, shows the mean daily throughput per contributor over a sample of five jobs. *Job 486* has mainly short variance over time, that is, in consecutive days the variation is usually less than 0.5 hours, in contrast to *job 542* that has significant variance over time, that is, two consecutive days can reach 4 hours of difference in throughput. Also, jobs usually have at least one abrupt variation. Finally, the mean daily throughput per contributor is not dependent on the job age, that is, we cannot see ascending/descending trend over time.

How many contributors execute tasks over a span of time is an important factor to estimate the job throughput in the span of time referred. Therefore, we study the contributor's distribution over time and their **joining rhythm**. The joining rhythm corresponds to the number of contributors completing tasks for the first time over time. Understanding the rate at which new contributors

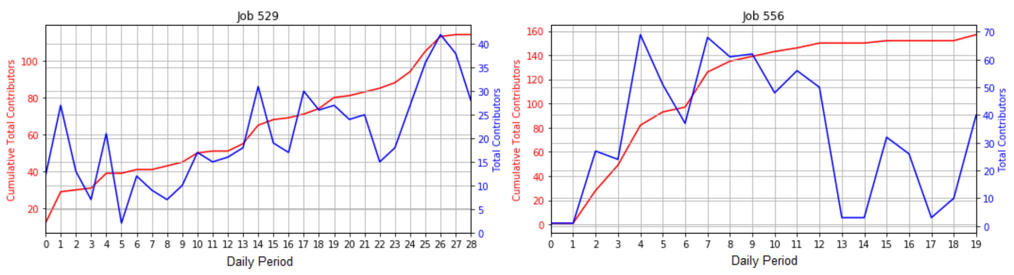


Figure 4.11: Cumulative total new contributors joining the job (red) versus daily total contributors executing tasks (blue) considering two distinct jobs

join the job, it is possible to evaluate if the number of contributors executing tasks is predictable.

Figure 4.11 compares the daily number of contributors executing tasks with the cumulative new contributors joining the job. This figure has two different jobs with two distinct contributors distribution over time. *Job 529* has an increasing number of new contributors joining the job converted in an increasing trend of total contributors completing tasks. On the other hand, *job 556* has a quick beginning in new contributors joining the job, followed by a moderate period that it is converted to a decreasing trend of total contributors completing tasks per day.

In the final analysis, we consider that the increasing/decreasing trend of contributors completing tasks varies between uneven values, in non related moments and without an evident pattern between jobs. We also found no relation between these conclusions and other variables or events in the platform. We suspect that these abrupt changes may be related to deadlines imposed by the business or even requirements adjustment.

4.2 Methodology

According to the CRISP-DM methodology, in order to model and evaluate **RQ2** regarding the crowd throughput forecasting, we should follow the steps described in Figure 4.12. In Section 4.3 we select the modelling technique and define the experimental setup in Section 4.3 and generate test design in Section 4.3.4. In Section 4.4.1 we build the models and in Section 4.4.2 we assess them. Finally, Section 4.4.3 is to evaluate the model in a business perspective.

4.3 Experimental Setup

After the exploratory analysis, in this Section, we prepare the dataset, methods and test design in which we model **RQ2**. As a result, we use **time series models** to forecast the crowd throughput in hours for the following 24 hours of a job, in other words, our *forecasting horizon* is of 24 periods. In the state of the art, throughput is measured as the number of tasks expected to be completed within the next hour [DFI18]. However, this definition lacks in understanding the variation in task complexity and length. As a result, we define throughput for a given time span as

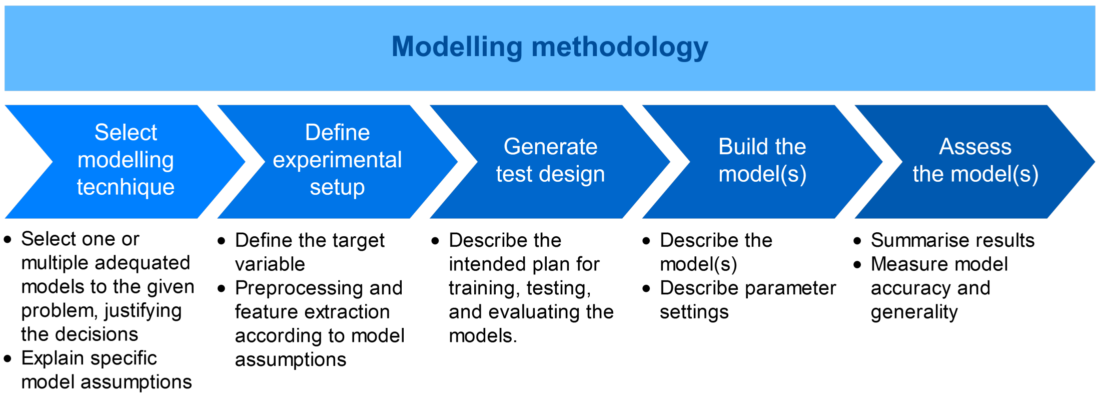

Modelling methodology
Select modelling tecnhique
Define experimental setup
Generate test design
Build the model(s)
Assess the model(s)
Select one or multiple adequated models to the given problem, justifying the decisions
Explain specific model assumptions
Define the target variable
Preprocessing and feature extraction according to model assumptions
Describe the intended plan for training, testing, and evaluating the models.
Describe the model(s)
Describe parameter settings
Summarise results
Measure model accuracy and generality

Figure 4.12: Modelling pipeline followed along this chapter

the expected cumulative time spent on task by the crowd because it explains factors as the number of contributors working and their profiles and the tasks cognitive load. The throughput prediction allows measuring scalability and crowd engagement in real-time.

In this case, the most accurate machine learning approach is *Quantitative Time Series Models*. *Time series* implies that we have a collection of values obtained from sequential measurements over time. Quantitative forecasting implies that information about the past is available and can be quantified in the form of numerical data and that some aspects of the past pattern will continue in the future [MWH98].

Difallah et al. uses used Random Forest to forecast the throughput in number of tasks completed [DCD+15]. However, as we measure the throughput in crowd hours the results cannot be directly compared. Also, it is the first extensive study and experiment in *DefinedCrowd*'s research and analytics team, so we do not have any results to compare with. Given these points, we start by establishing a *baseline* model, that simply estimates the throughput for the next hour to be the same as for the previous.

In the next sections, we summarise some of the most important methods to forecast with time series. We focus on linear methods, that is, the forecast models can all be expressed as a linear combination of past observations. Non-linear time series models, such as the ones that can be represented by neural networks, are out of the scope of the current thesis. The linear methods that we focus our studies belong to Exponential Smoothing methods (see Section 4.3.1), time series regression (see Section 4.3.2) and Box-Jenkins methods (see Section 4.3.3).

4.3.1 Exponential Smoothing

This group of methods is based on the fact that past observations are given weights, that is, a relative importance included in forecasting [MWH98], that typically decay in an exponential manner from the most recent to the most distant data point.

All methods in this group includes four parameters that must be defined:

- α : level (i.e. horizontal pattern) smoothing parameter

- β : trend smoothing parameter
- γ : seasonality smoothing parameter
- ϕ: damping parameter, i.e. a parameter that smooths the trend over time when forecasting so that constant trends do not increase/decrease indefinitely into the future

These parameters determine the unequal weights to be applied to past data and their values lie between 0 and 1, where zero indicates that all weights are equal and one that we use only the last value to forecast. An approach for determining these values is to use a non-linear optimisation algorithm to find optimal parameter values.

The exponential smoothing methods are:

- Simple Exponential Smoothing (one parameter : α)
- Holt's linear method (suitable for trends)
- Holt-Winter's method (suitable for trends and seasonality)

The major **advantages** of widely used smoothing methods are their simplicity and low cost [MWH98] and it produces accurate forecasts. The major **disadvantage** is that it is only optimal when we have short term forecasting and large numbers for forecasting.

The Holt-Winters' method is based on three smoothing equations: one for the level, one for trend, and one for seasonality [MWH98]. Moreover, there are two different Holt-Winters' methods, depending on whether seasonality is modelled in an additive or multiplicative way. The former says that the various terms are added together [MWH98] so the seasonal variations are roughly constant over time. The latter says that the various terms are multiplied together [MSA18] so the seasonal variations change proportionally to the level of the series. In Section 4.1.1, we verify that we do not have strong seasonality, so the additive model would have better fit than the multiplicative.

Equations 4.4, 4.5, 4.6 and 4.7 are the basic equations for Holt-Winters' additive method, where Y_t is the current observation, m the forecasting horizon and s is the length of seasonality.

$$Level : L_t = \alpha(Y_t - S_{t-s}) + (1-\alpha)(L_{t-1} + b_{t-1}) \tag{4.4}$$

$$Trend : b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1} \tag{4.5}$$

$$Seasonal : S_t = \gamma(Y_t - L_t) + (1-\gamma)S_{t-s} \tag{4.6}$$

$$Forecast : F_{t+m} = L_t + b_t m + S_{t-s+m} \tag{4.7}$$

Equation 4.6 is based on the difference between the trend L_t that is a smoothed value of the series without seasonality and the observation Y_t that contains both randomness and seasonality.

The equation adds up the new season with the most recent one, both weighted with γ in order to smooth the randomness [MWH98]. Equation 4.5 updates the trend in the first term and then adds up with the most recent one, both weighted with β in order to smooth the randomness [MWH98]. In Equation 4.4, the first term is divided by the seasonal number S_{t-s} to eliminate seasonal fluctuations from Y_t. The second term adjusts the trend of the previous period to eliminate the lag [MWH98]. Finally, Equation 4.7 is used to forecast ahead.

4.3.2 Time Series Regression

In this group of methods, a forecast is expressed as a function of a certain number of factors that influence its outcome that is not necessarily time dependent [MWH98]. As a result, we should come up with variables (e.g. seasonal effects and explanatory variables) that relate to the data series of interest and then develop a model that expresses the functional relationship among them.

The main **advantage** of this group of methods is that they are widely used given their popularity among Machine Learning studies which may allow you to compare your results to other benchmarks. The main **disadvantages** are [MSA18]:

- there is a possible lack of independence in the residuals, which breaks the assumptions of these methods;
- when forecasting for our target variable, we need to first have forecasts for each of the explanatory variables which can be difficult sometimes;
- it is very easy to create long lists of variables bringing more complexity to the model that requires feature selection and consequently large amounts of data.

In our context, the disadvantages overlap the advantages so that we decide to not use this group of methods.

4.3.3 Box-Jenkins Methodology

The Box-Jenkins methodology consists of a group of methods that include ARIMA (Autoregressive Integrated Moving Average) models and their upgrades. This approach consists of three phases: identification, estimation and testing and application. Figure 4.13 shows a schematic representation by Makridakis that can be found in his book *Forecasting: Methods and Applications* [MWH98]. We follow this approach throughout this chapter. The identification phase, started in Section 4.1.1 while differencing data and examining the ACF and PACF plots. The estimation and testing will be assessed in Section 4.4.1, while the application will be assessed in the evaluation phase in Section 4.4.2.

The general seasonal ARIMA model is known as $ARIMA(p,d,q)(P,D,Q)_s$, where:

AR : p = order of the autoregressive part (AR)

I : d = degree of first differencing involved

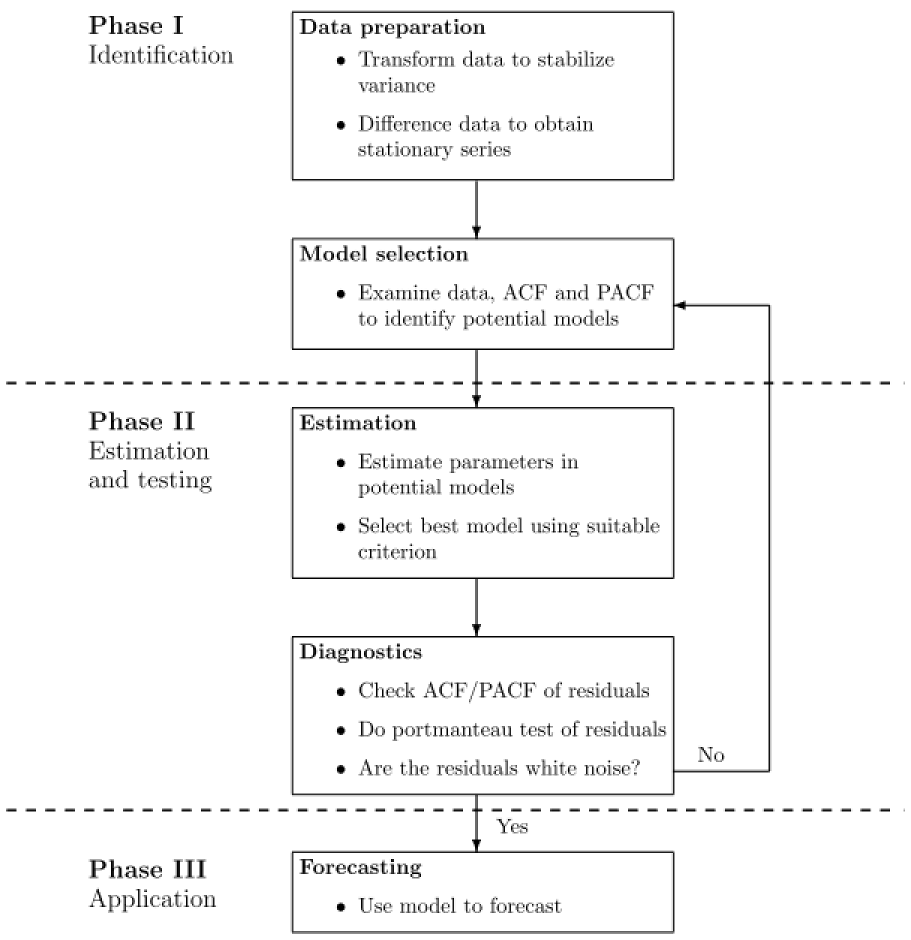

Phase I
Identification
Data preparation
Transform data to stabilize variance
Difference data to obtain stationary series
Model selection
Examine data, ACF and PACF to identify potential models
Phase II
Estimation and testing
Estimation
Estimate parameters in potential models
Select best model using suitable criterion
Diagnostics
Check ACF/PACF of residuals
Do portmanteau test of residuals
Are the residuals white noise?
No
Yes
Phase III
Application
Forecasting
Use model to forecast

Figure 4.13: Schematic representation of the Box-Jenkins methodology for time series modelling by Makridakis that can be found in the book *Forecasting: Methods and Applications* [MWH98]

MA: q = order of the moving average part

(P,D,Q) = order of the seasonality and s = number of periods per season

Note that if some of the orders are zero, the model can also be written in a shorthand notation, e.g. $ARIMA(p,d,q)(0,0,0)_s$ is $ARIMA(p,d,q)$. Figure 4.14 shows the mathematical formulation for $ARIMA(1,1,1)(1,1,1)_4$ example model, where B is the backward shift, that is, the differencing operator (see equation 4.8). ϕ_x and θ_x are the autoregressive and moving average parameters, where x corresponds to the order.

$$(1-B)Y_t = Y_t - Y_{t-1} \tag{4.8}$$

Box-Jenkins major **advantage** is the need of only 50 periods to have adequate forecasts [HK14]. Box-Jenkins methods major **disadvantages** are the high costs and instability, that is,

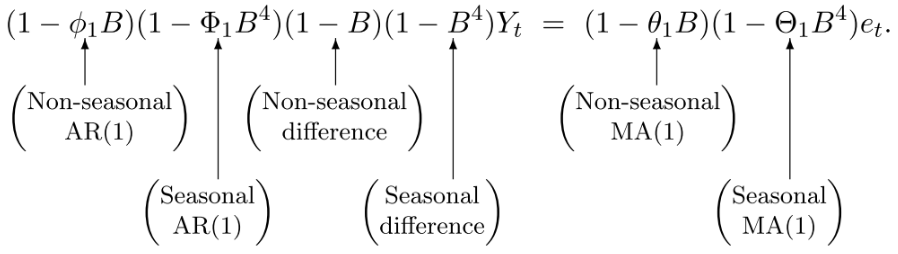


Figure 4.14: Mathematical formulation for $ARIMA(1,1,1)(1,1,1)_4$ model

changes in observations or in model specification affects deeply the forecasting. Also, these methods have the assumption that time series must be weakly stationary or some order integrated. As we have stated in Section 4.1.1, our time series are first order differenced so we fulfil the assumption.

4.3.3.1 Seasonal ARIMAX

Although univariate time series, as seasonal ARIMA, refers to a single variable observation over time, that is, the target variable, seasonal ARIMAX allows explanatory variables. This method follows a similar formula to Seasonal ARIMA from Figure 4.14, but instead of Y_t it uses the W_t, defined in 4.9 formula, where β is the coefficient value for the explanatory input variable and x_t the explanatory variable at time t.

$$W_t = Y_t - \beta x_t \tag{4.9}$$

Considering that we should make a prediction for 24 hours, our explanatory variables must relate to the past, i.e. more than one day lagged values. As a result, we assess which variables at the same hour in the previous day correlate the most with the target variable.

Figure 4.15 shows a correlation matrix between our target variable and one day lagged explanatory variables that we found useful during Chapters 3 and 4. We verify that the number of active contributors, task length in tokens and total executions with one day lag correlates with the target variable.

4.3.4 Test Design

With the different nature of time series and forecasts types, choosing the method that performs well and yield accurate forecasts vary, as well as the evaluation techniques that guarantee robust and meaningful error estimates [BB12]. One important aspect of the predictive models is that they should generalise, that is, we can expect that the model correctly predicts any new data [Bos14]. As a result, we should use out-of-sample methodologies where data is divided into training and test sets. The training set can be also divided into train and validation set so we can tune the model's hyper-parameters. Then, we compare the **fitting errors** (the difference between the fitted model

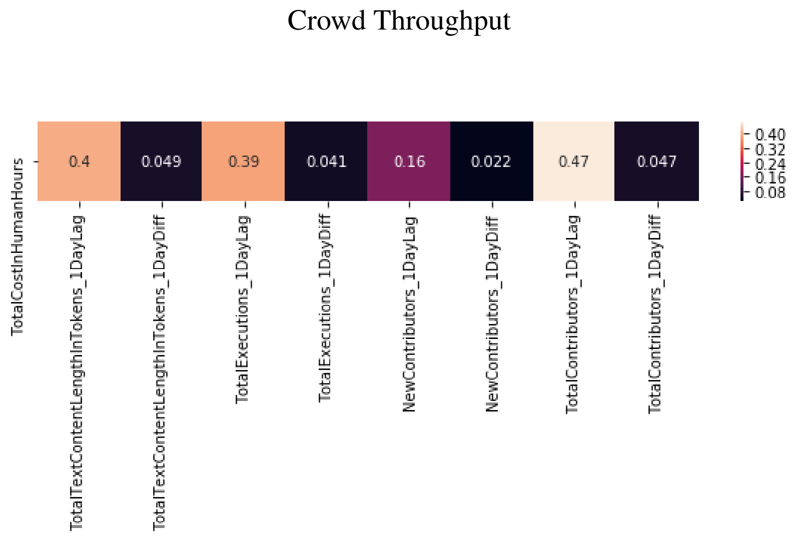


Figure 4.15: Correlation Matrix between target and explanatory variables

and the true model) with the **forecasting errors** (the difference between the predicted values and the true values) in order to assess the performance. This out-of-sample methodology avoids over-fitting the model [MWH98], that is, include randomness as part of the generating process.

When dealing with time series models, most of the times the hyper-parameters are selected according to visualisations (e.g. ACF or PACF). However, in this scenario, we have several time series with different data patterns. As a result, the plots give us an intuition but we need to tune the hyper-parameters with a *Grid Search* on the validation set.

When dealing with time series modelling techniques, it is important to split to take into consideration the temporal dimension so that the natural dependencies in the data are respected when splitting between train, validation and test set. Considering that we want to forecast the next day for an on-going job, we reserve the last 24 data points from each time series for evaluation purposes, that is, to the test set. This technique is known as last-block validation [BB12]. In order to split between train and validation sets, we can use in-sample evaluation. To perform it, we use *TimeSeriesSplit* scikit-learn Python library to perform the time series split. As we have 18 time series, we decide to split them with the minimum value allowed, i.e. three leading to $18 * 3 = 54$ folds. Considering that ARIMA is a medium to long length time series that requires a minimum of 50 time points [HK14], we only accept new folds that have at least 50 training points. As a result, some folds were removed leading to a total of 34 folds and 13 time series. Figure 4.16 summarises the train/validation/test splitting per each time series.

To validate the models, we train with the training set and validate the model with the validation set. We average the 34 folds performances. When tuning hyperparameters we select the model with the lowest AIC, as it yields more accurate assessment than BIC (see Section 2.2.4).

To evaluate the models' performance, we use the training and validation set to train our model and then the test set to forecast. Finally, we average the 34 folds performances to achieve the model evaluation. We then compare the out-of-sample performance between the models and the baseline, using the MAE and RMSLE metrics as forecasting errors. RMSLE allows a fair comparison between different jobs while MAE allows measuring for how many minutes/hours are we failing

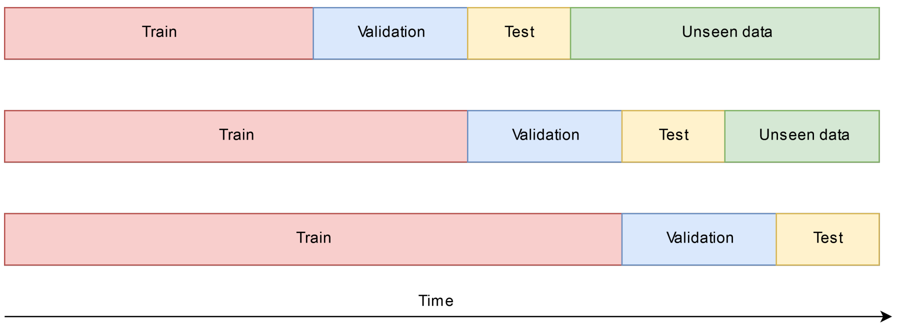


Figure 4.16: Division of time series into three folds

our prediction (see Section 2.2.4).

4.4 Crowd Throughput Results

After exploring and establishing the experimental setup regarding the **crowd throughput** forecasting (**RQ2**), we must model and evaluate our question in order to achieve the final results.

4.4.1 Models Fitting

In this section we model and assess a baseline formula and three time series models (seasonal ARIMA, seasonal ARIMAX and Additive Holt-Winter's) to answer **RQ2**. We start by fitting the models, then we evaluate their performances both in a research and business perspective.

4.4.1.1 Persistence

As we have no previous models to forecast the daily throughput in human computation time, we have no line of comparison to evaluate the goodness of fit of the model. As a result, we create a naive forecast, that is, forecasting with a minimal amount of effort and data manipulation and it is based solely on the most recent information available [MWH98]. Then, we can compare the models achieved performances and estimate the relative improvement regarding the baseline.

When using time series, the simplest prediction is forecast the next value equal to the current one, i.e. $\hat{y}_{t+1} = y_t$. This model is named persistence and it has no validation phase, so we only forecast the values and evaluate them.

4.4.1.2 Seasonal ARIMA

Seasonal ARIMA is a Box-Jenkins method (see Section 4.3.3). If the time series contains a seasonal component and the lag is coincident with the periodicity of the data, the model is called seasonal $ARIMA(p,d,q)(P,D,Q)_s$, where (P,D,Q) is the seasonal part of the model and s is the seasonality of the model. We decide to use this model given the trend and seasonal properties that

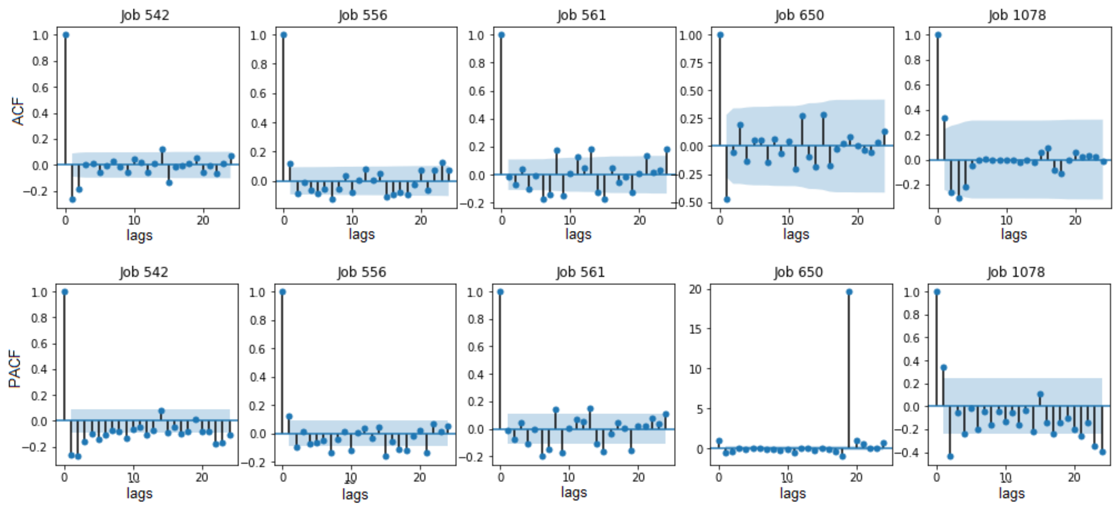


Figure 4.17: Autocorrelation plot sample (top) and Partial autocorrelation plot (bottom) sample for first order differenced series with a maximum of 24 lags

we identify and had previously discussed in Section 4.1.1 (see Appendix A.1 for complete time series visualisations).

In order to choose the seasonal ARIMA order, we both look into the ACF and PACF plots to use intuition and used hyperparameter tuning to choose the best fitting model. According to Section 4.1.1, our time series are first order difference-stationary, that is, $d = 1$ and $D = 1$, and has daily seasonality, that is, $s = 24$.

Autoregressive (AR) process calculates an estimate for a future value using determined lagged values from the past [BB12]. The order of the AR model is the lag value after which the PACF plot crosses the upper confidence interval for the first time. Figure 4.17 shows the partial autocorrelation plot for a sample of five first order differenced series with a maximum of 24 lags. The remaining PACF plots can be found in Appendix A.4. The first order differenced PACF plots show usually no values crossing the upper confidence interval.

Moving average (MA) process is a process where the present value of series is defined as a linear combination of past errors [BB12]. Order q of the MA process is obtained from the ACF plot, this is the lag after which ACF crosses the upper confidence interval for the first time. Figure 4.17 shows the partial autocorrelation plot for a sample of five first order differenced series with a maximum of 24 lags. The remaining ACF plots can be found in Appendix A.4. Again, some time series show no pertinent correlations, others show relevant correlations around the 24 hours. This can be explained by the seasonality that some time series reveal.

As we have no evident seasonal ARIMA orders that can fit all time series, we decide to do a *Grid Search* varying the parameters (p,q) between [0:7] and (P,Q) between [0:2]. The chosen model was the one with the lowest AIC. The grid search results can be found in Appendix B.1.

The chosen model was **seasonal ARIMA(6,1,2)(2,1,0) with s=24** that achieved a mean *AIC = 386.13*. In order to check the goodness of fit of the model, it is possible to assess several estimated errors diagnostics. Figure 4.18 shows a sample of four time series diagnostics, i.e. standardised residuals, histogram and estimated density, QQ plot and correlogram, i.e. a plot of the ACF against the lag. For a model have a good fit, the residuals should be simply white noise, that is, the data must have small and no systematic or predictable patterns. Consequently, the following rules must apply:

- The residuals should not exhibit *heteroscedasticity*, that is, the errors do not have a constant variance across an entire range of values. Any residual less than -3 or greater than 3 is an outlier.
- Kernel Density Estimation (KDE) consistent with the normal distribution (N(0,1))
- The QQ-plot points lie approximately on the red line, meaning that the residuals are similar to the normal distribution (N(0,1))
- Autocorrelation plot with low correlation at lags > 0 because it indicates that the forecasting method has removed all of the pattern from the data [MWH98]

Observing Figure 4.18, we can state that the last three time series comply with the previous rules, while the first one shows a slight lack of normality in the data. In Appendix B.3 we have the full time series of diagnostic plots. Observing those plots, it is possible to see that most of the time series comply with the normality requirements, so we can conclude that the model fits the data, but not precisely. Finally, considering the outliers, the diagnostics show that there are outliers in some time series that may affect our results.

Figure 4.19 shows an example of two time series comparing the training set with the model fitted values. The remaining visualisations can be found in Appendix B.3. While the first time series has a precise fitting, the second one has greater variance. Actually, the mean training set MAE is *0.427* that is, on average, the fitted values have 26 minutes difference from the actual throughput. This result confirms that the goodness-of-the-fit is not the same over time series, and consequently, the accuracy will be higher or at least equal to 0.427 (MAE).

4.4.1.3 Seasonal ARIMAX

Seasonal ARIMAX is based on seasonal ARIMA, but uses explanatory variables to assist the prediction. In this case, we use the previous seasonal ARIMA calculated orders but with the new dataset including explanatory variables. According to Figure 4.15, the one day lagged variables that correlate with our objective variable, i.e. total throughput in human hours, are the *TotalContributors*, *TotalTextContentLengthInTokens* and *TotalExecution*. As a result, we apply a $ARIMAX(6,1,2)(2,1,0)_{24}$ to multiple combinations of the explanatory variables and compare the mean AIC achieved.

Table 4.4 shows the comparison between mean AIC considering the different created datasets with explanatory variables combinations. We can conclude that the model with the lowest AIC is

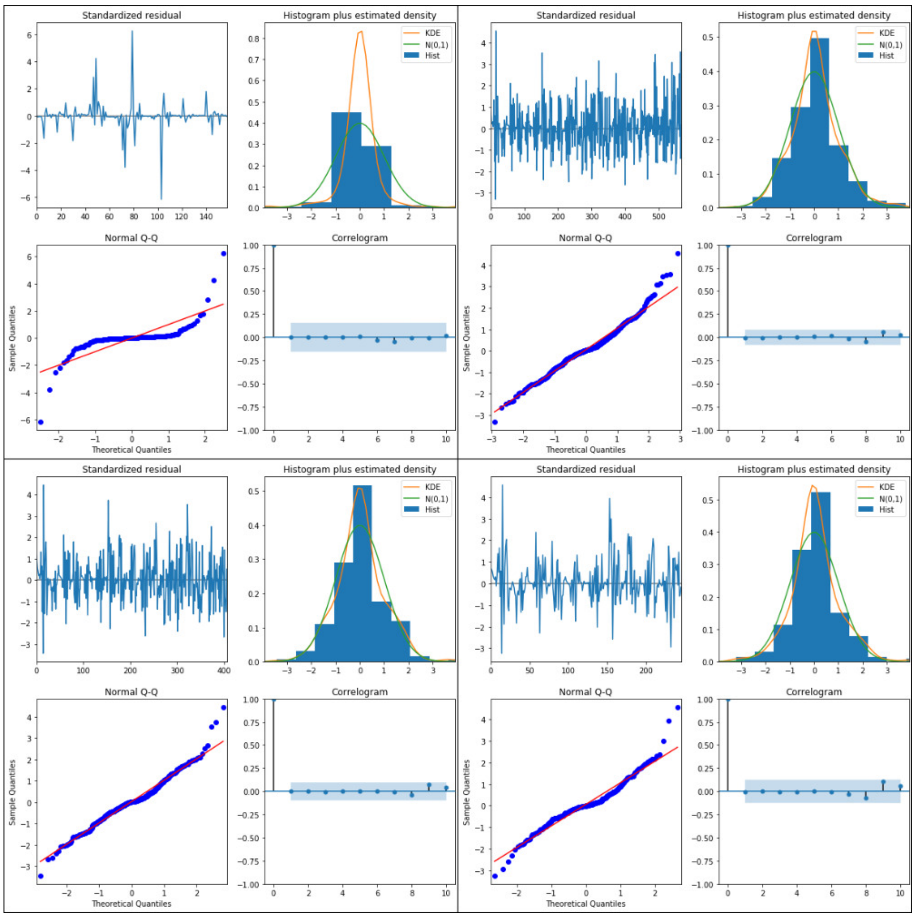


Figure 4.18: $ARIMA(6,1,2)(2,1,0)_{24}$ sample of time series diagnostics. From top left to bottom right: standardised residuals, histogram and estimated density, QQ plot and correlogram

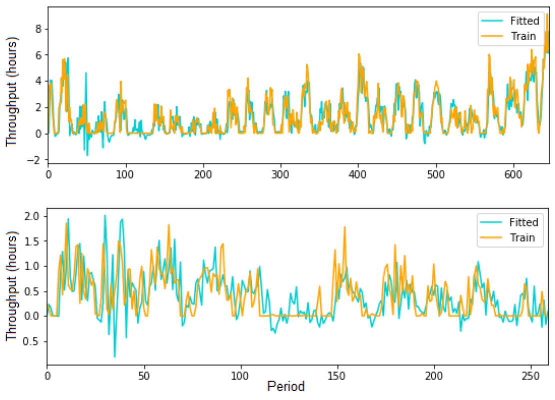


Figure 4.19: Training values versus fitted values by $ARIMA(6,1,2)(2,1,0)_{24}$ for throughput in hours over time considering a sample of two jobs

the one with the exogenous variable *TotalContributors* one day lagged that achieved $AIC = 384.14$. As a result, we analyse the goodness of the fit following Section 4.4.1.2 established rules.

Table 4.4: Comparison between AICs when feeding the seasonal ARIMAX model with different exogenous variables

Exogenous Variables	**AIC**	**BIC**
TotalContributors	384.14	422.36
TotalTextContentInTokens	397.21	435.43
TotalExecutions	387.69	425.92
TotalTextContentInTokens TotalContributors	396.95	438.36
TotalContributors TotalExecutions	386.34	427.75
TotalTextContentInTokens TotalExecutions	401.98	443.38
TotalTextContentInTokens TotalContributors	402.37	446.96

Observing Figure 4.20 we have a similar diagnostic to Section 4.4.1.2. In this case, we have one example of normally distributed residuals (bottom left diagnostics), while the remaining ones are *heteroscedasticity*, i.e. sub-populations that have different variabilities from others.The remaining plots in Appendix B.3 validates the goodness of the fit.

Figure 4.21 shows an example of two time series comparing the training set with the model fitted values. The remaining visualisations can be found in Appendix B.3. While the first time series has an accurate fitting, the second one has significant variance. Actually, the mean training set MAE is *0.424* that is, on average, the fitted values have 26 minutes difference from the actual throughput. This result confirms that the model is not fitted accurately among time series, and consequently, the accuracy will be higher or at least equal to 0.424 (MAE).

4.4.1.4 Additive Holt-Winter's

Holt-Winter's method belongs to the category of exponential smoothing methods (see Section 4.3.1). These methods assign exponentially decreasing weights for past observations so that the more recent the observation is, the larger weight would be assigned. Holt-Winters' method was chosen by being suitable for data with trends and seasonalities.

The model was built with the *statsmodel* library for Python, that already optimises the smoothing parameters α, β, γ and ϕ. In our context, we are using the additive method, as explained in Section 4.3.1, and a seasonal order of 24 periods, as in previous models.

In our context, we fit 34 models as we have 34 time series and we present the achieved mean AIC of -390.26 and the mean optimal values in Table 4.5. The full parameters fitted per each time series can be found in Appendix B.2. We observe that the mean model is almost absent from seasonality. The level parameter can vary between zero and one where one is the maximum value says that the last values have maximum weight, i.e. it corresponds to the persistence, and zero means that all values from the training data have similar weights. In this case, the level is set to 0.56 meaning that the recent past is more significant than the aged past. The trend parameter is

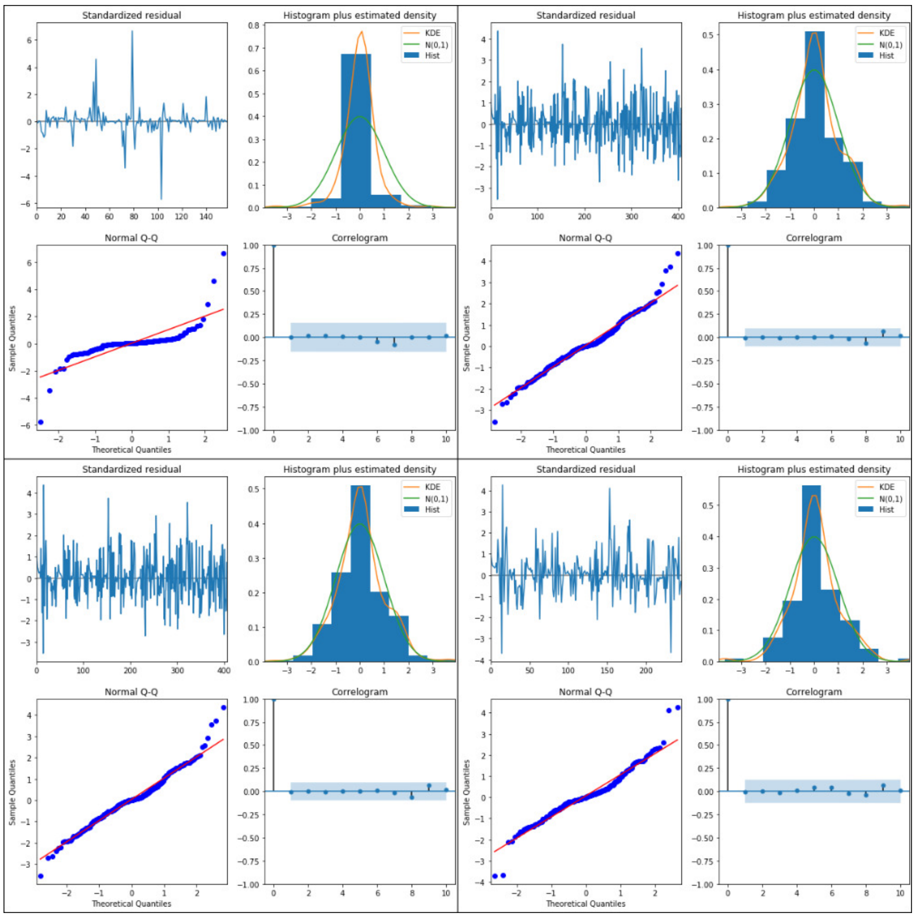


Figure 4.20: $ARIMAX(6,1,2)(2,1,0)_{24}$ sample of time series diagnostics. From top left to bottom right: standardised residuals, histogram and estimated density, QQ plot and correlogram

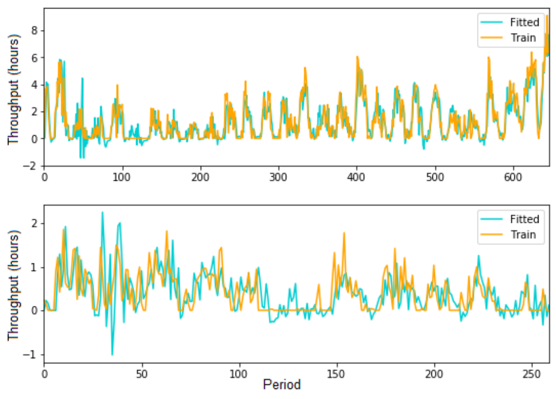


Figure 4.21: Training values versus fitted values by $ARIMAX(6,1,2)(2,1,0)_{24}$ for throughput in hours over time considering a sample of two jobs

close to 0.32 meaning that the model capture more than only the recent past trend. Observing the Tables B.3, B.4 and B.5 from Appendix B.2 we observe that there are several fitted models with

Table 4.5: Optimal parameters achieved with Holt-Winter's Exponential Smoothing method

mean α	mean β	mean γ	mean ϕ
0.5643	0.3276	0.0427	0.0252

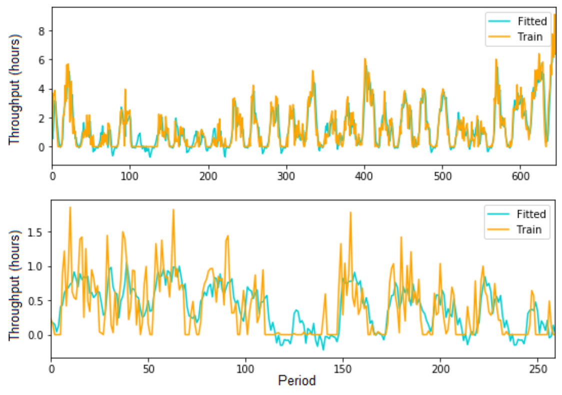


Figure 4.22: Training values versus fitted values by Additive Holt-Winter's model for throughput in hours over time considering a sample of two jobs

trend close to one and absent of seasonality.

Figure 4.22 shows a sample of two time series folds comparing the training set with the model fitted values. The remaining visualisations can be found in Appendix B.3. We can state that the model can capture the cyclical patterns and trends properly, but not equally among time series. Actually, the mean training set MAE is *0.38* that is, on average, the fitted values have 23 minutes difference from the actual throughput. This result confirms that the model is not fitted accurately among time series, and consequently, the accuracy will be, at least, equal to 0.38 (MAE).

4.4.2 Models Assessment

So far, we have built several models that appear to fit the data properly. Now, we assess the models according to the established test design discussed in Section 4.3.4. Although our goal is to forecast a horizon of 24 periods, we assess the results for one-step-ahead forecasting horizon in order to compare with the previous horizon and discuss the model's deterioration.

We start by assessing our results one-step-ahead. Table 4.6 presents the performance results. We verify that all methods achieve better performance than Persistence with Holt-Winter's achieving 9% better performance.

Then, we forecast with a 24 period horizon. Table 4.7, shows the performances achieved when assessing the models with this forecasting horizon. The table shows that the model with the best performance is the **Additive Holt-Winter's** (both considering MAE and RMSLE), with close performances to the baseline.

Figure 4.23 shows the forecast values for two time series (columns) per each model (rows). The orange values correspond to the training set, the blue to the real values and the green to the

Table 4.6: Forecasting one step ahead models assessment

Model	**AIC**	**BIC**	**Fitted MAE**	**Forecasting MAE**	**Forecasting RMSLE**
Persistence	-	-	-	0.4795	0.2173
$ARIMA(6,1,2)(2,1,0)_{24}$	386.13	421.17	0.4266	0.4731	0.2078
$ARIMAX(6,1,2)(2,1,0)_{24}$	397.21	408.23	0.4244	0.4876	0.2123
Holt-Winter's	-390.26	-285.21	0.3798	0.4324	**0.1983**

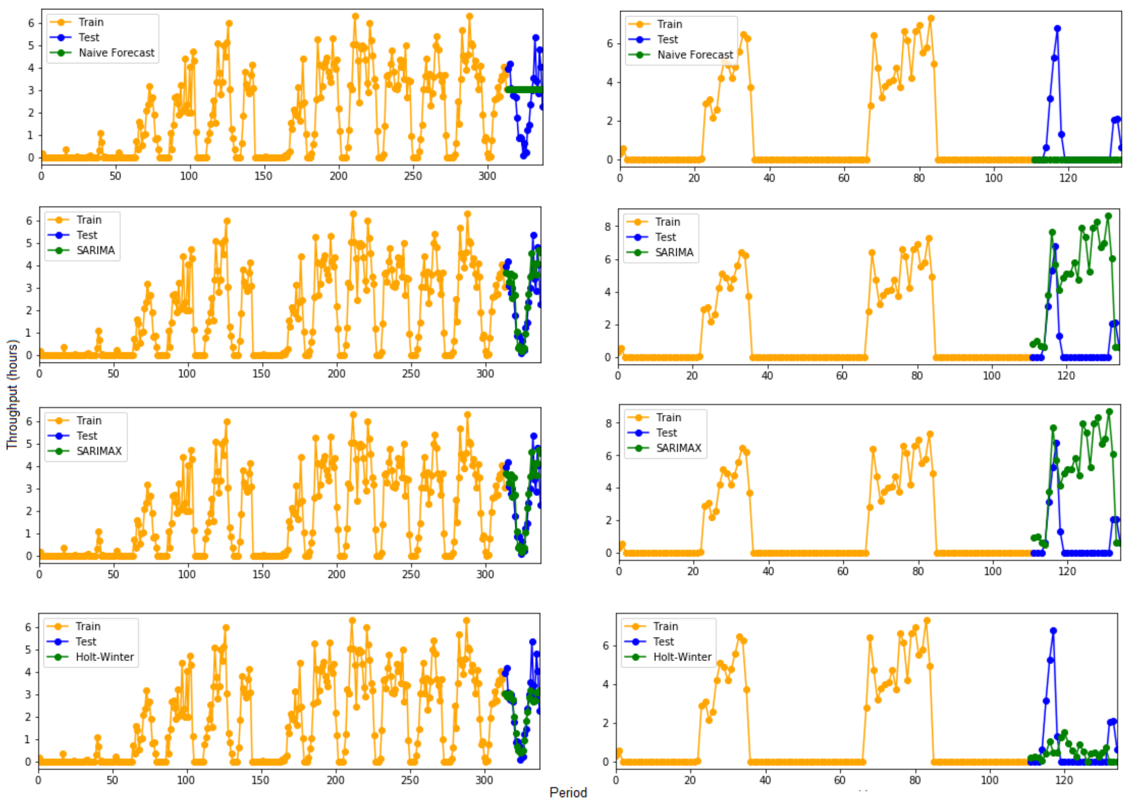


Figure 4.23: Train and forecast visualisation sample of two time series (columns) and our models (Baseline, seasonal ARIMA/ARIMAX, Holt-Winter's)

forecasting values. The remaining visualisations can be found in Appendix B.4. It is possible to observe that the seasonal ARIMA and seasonal ARIMAX have similar fittings. In the first time series, their prediction is the most suitable, capturing the trend and seasonality as expected. However, in the second time series, they both fail considerably, as the model behaves out of expectations. On the other hand, the exponential smoothing method creates an accurate prediction concerning the first time series, as good as the seasonal ARIMA/ARIMAX models. However, the second time series has more accurate predictions comparing to seasonal ARIMA/ARIMAX models for the reason that this model gives more weight to the recent past.

Overall, we state that persistence attains better performances than most of the algorithms. This result can be explained by the fact that we have time series with different trends and seasonality.

Table 4.7: Forecasting 24 period horizon models assessment

Model	**AIC**	**BIC**	**Fitted MAE**	**Forecasting MAE**	**Forecasting RMSLE**
Persistence	-	-	-	0.8391	0.4073
$ARIMA(6,1,2)(2,1,0)_{24}$	386.13	421.17	0.4266	1.0551	0.4954
$ARIMAX(6,1,2)(2,1,0)_{24}$	397.21	408.23	0.4244	1.0081	0.4769
Holt-Winter's	-390.26	-285.21	0.3798	0.8214	**0.39**

Also, our time series have some unpredictable factors that influence the predictions. Consequently, predicting the next value equal to the previous one is a strong statement that produces accurate predictions.

Seasonal ARIMA and ARIMAX have similar performances, meaning that adding an explanatory variable only increases the model's complexity without increasing its accuracy, so that we favor ARIMA over ARIMAX.

Considering the model generalisation, we can state that our models do not generalise as expected for the 24 period forecasting horizon. Generalisation corresponds to the model's ability to adapt properly to unseen data drawn from the same distribution as the one used to create the model. Comparing the training and test sets MAE, we conclude that the test set error is approximately *2.3* times larger than the training error. Models that have good generalisation usually have similar train and test set errors.

Finally, if we were forecasting with the Additive Holt-Winter's fitted model, we could say that we could be mistaken by 50 minutes (0.8214 of one hour). Comparing to Table 4.2 that shows the throughput in hours distribution, we believe that this error is significant, considering that it is close to the throughput mean of one hour. However, we should also observe that the throughput can vary between zero and 17.18 hours, so 50 minutes is reasonable when comparing to large throughput.

When comparing the forecasting horizons, we observe that the one-step-ahead performance is better than 24 period horizon. As a result, we can state that our model's performance degrades over the 24 step horizon. As a result, we have more confidence when forecasting for the following hour than a whole 24 step horizon.

4.4.3 Models Evaluation

In previous sections, we have modelled our problem dealing with factors such as accuracy and model generalization only considering data analytics. According to CRISP-DM methodology, in the evaluation step we must determine if there is some important business issue that has not been sufficiently considered [Wir]. At the end of this phase, a data mining strategy and model must be chosen and the process must be reviewed in order to determine the next steps.

Considering **RQ2** in regard to the crowd throughput forecasting, we have assessed the models in Section 4.4.2, concluding that the model that achieved the best performance was the **Additive Holt-Winter's**. However, the data mining strategy used to train the model is not suitable for a live

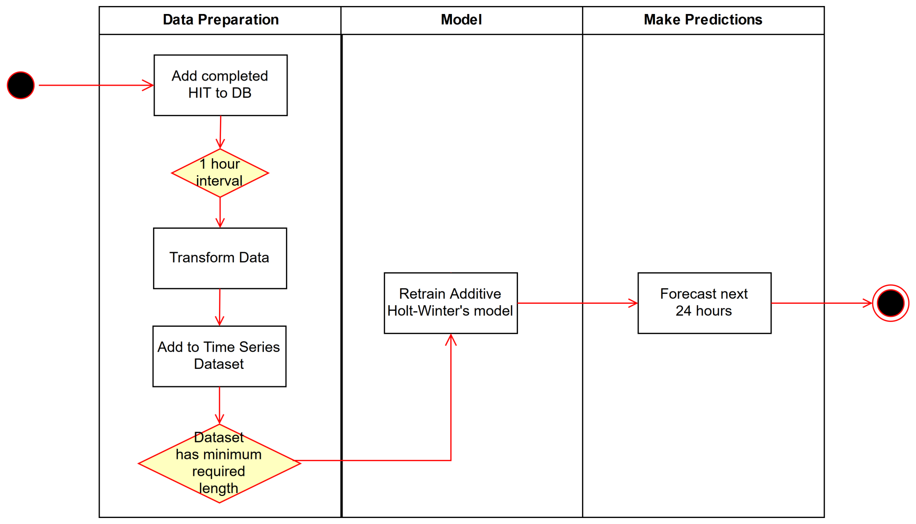


Figure 4.24: Model implementation business strategy from task arrival to forecasting

model with constant information arrival that must update the forecast. As a result, a proper data mining strategy must be applied.

Figure 4.24 shows a possible data preparation and modelling strategy for successful business implementation. While the information that tasks have been completed arrives, we update our database until one hour has passed by. Then, we aggregate the data as described in Section 4.1. As soon as we have the established 50 hours of minimum data, the model is ready to train and forecast the next 24 hours. Then, the model is trained hourly with the new tasks arrived, with the *rolling origin recalibration* evaluation. This method forecasts for a fixed horizon, i.e. 24 data points, by sequentially moving values from the test set to the training set, and changing the forecast origin accordingly. For each forecast, the model is recalibrated using all available data in the training set, which means complete retraining of the model [BB12]. This method is very useful as it is not static, and can be continuously updated. As an example, if when predicting the next 24 hours will have a given trend, update the model for the next hour and we have an unexpected throughput, the model is able to recover and change the predictions for the next 24 hours.

This solution is a business asset as it is appropriate for jobs that demand large crowd effort, given that we need at least 50 hours running in order to make the first prediction. Given how unpredictable the crowd throughput can be, jobs with high computed crowd effort are the ones that require project managers attention in order to manage their expectations throughout the process.

Predicting a foreseeable future gives insights about the expected throughput trend so that we can, in the first place, understand the crowd behaviours (e.g. daily seasonality or the most productive hours), and then be aware beforehand about throughput variations (e.g throughput slow down) allowing to take actions ahead.

Although we have concluded in Section 4.4.2 that the model can significantly degrade over a longer horizon, we still consider that the model is a business asset given that our goal is not to predict exactly the work that will be concluded by the contributors but to anticipate the the crowd's throughput trend for the near future.

4.5 Discussion

In the previous section, we discussed our research questions results in *terms of business*, discussing how could we implement the solution and why it is a business asset. In this section, we discuss other results, as the use of time series to model the **RQ2** about the crowd throughput forecasting, issues and directions of this solution both considering the business and the crowdsourcing community.

This thesis was strongly influenced by working with real data. In our context, we verify that data is noisy and influenced by external factors. In a business context, it is usual that the actual completion timelines are indirectly influenced by business deadlines (whose variations we could not consider into our models), rather than the other way around. Consequently, in particular scenarios the crowd gets incentives that are external to our data, and likely not aligned with any past patterns. Observing the time series, it is usual to observe increasing trends close to the deadline, although we had no access to deadline information explicitly as a part of our data. These factors cannot be modelled, however, they can be considered when interpreting the data. Moreover, with the new crowd effort estimation, it is likely that the timelines can be more accurate and consequently the crowd management less intrusive.

When assessing the models, we state that persistence is a reliable approach. We can explain it by the fact that we have different time series with different trends and seasonality. Also, our time series have some unpredictable factors that influence the predictions.

Using time series was a unique solution, that was not previously considered in the state of the art and we further explore in Section 4.5.1. Considering our goal of predicting the foreseeable future considering the recent past, we believe that it was the wisest solution. Forecasting a 24 period horizon allows to understand the throughput evolution over time and minimizes the impact of the forecasting errors.

Although the hypothesis of using regression in time series was studied, it was not possible to formulate given the profound differences among time series. Time series regression success depends on the features extracted, and consequently on the time series graphical exploration. However, we have concluded that different time series have different patterns, so finding the right set of features became difficult. Finally, considering multivariate time series the state of the art suggests deep learning, that was out of the scope of this thesis considering that it requires a much larger number of observations than the previously discussed methods [MWH98] that was not available.

4.5.1 Comparison with State of the art

Table 4.8 reflects a comparison of our solution with the Section 2.2.3 state of the art. We verify that our solution is very unique with a different end result. While in the state of the art the throughput is measured at the *platform* level, we study the throughput per job. Then, as they use all jobs, regarding the job type, they have *130 times* more tasks than our solution, allowing an accurate

Table 4.8: Comparison of our solution with past approaches in the literature to predict throughput in crowdsourcing environment

Publication	[DCD+15]	[YRDB16]	Our solution
Crowdsourcing Platform	Mechanical Turk	Mechanical Turk	Neevo
Dataset Size	2.5M jobs 130M tasks	2.5M jobs 130M tasks	24 jobs 1M tasks
Throughput Measured in	Number of tasks	Number of tasks	Effort in hours
Throughput Span of time	1 hour	1hour	1 hour
Throughput Measured per	Platform	Platform	Job
Job type	General	General	NET
Most important features used	Job age in minutes Number of tasks left	Number of tasks left Published tasks Completed tasks Total reward	1 day lagged number of contributors
Evaluation Metric	R-squared	MAE	RMSLE MAE
Machine Learning Algorithms	RF regression	RF Regression Lasso Linear Regression	SARIMA SARIMAX Holt-Winter's

prediction using methods as Random Forest. In our solution, our dataset limitations (NET and the amount of tasks) reduce our scope to less complex methods. Finally, we use the same span of time to measure throughput but a different measurement: *crowd effort* instead of *number of tasks*.

4.6 Conclusion

This chapter is committed to analyse **RQ2** in regard to the crowd throughput forecasting. The crowd eligibility and availability influence profoundly the expected throughput. As a result, in this chapter we analyse the throughput over time, starting by creating our dataset with hourly observations. Then, we analyse the throughput over time per each job. We conclude that different jobs have different trend and seasonality and that a part of jobs are stationary while the others are difference-stationary.

Then, we conduct a job investigation in order to find new features or behaviours that could shape the different time series. We define the boundaries that a job can finish between 2.5 times faster to 2 times slower than the time it needs from the crowd, in real-world time. Then, we demonstrate that the number of contributors has a direct influence on throughput in human computation hours. Finally, we conduct an analysis of the mean daily throughput by contributors, achieving that an average contributor contributes approximately 1.3 hours per day and per job.

Considering the modelling and evaluation results, we conclude that the Additive Holt-Winter's method achieves the best performance with 0.39 RMSLE when forecasting 24 periods and 0.198 RMSLE when forecasting one step ahead. While one-step ahead our forecasting is 9% better than the baseline, 24 periods horizon is only 1% better, meaning that increasing the forecasting period degrades our forecasting. As a result, during the evaluation phase, we consider a new method that creates new predictions over time so we can learn over time and adapt to unexpected situations.

Chapter 5

Conclusions and Future Work

In this chapter summarise our work giving a perspective of objectives satisfaction in Section 5.1, discussing limitations in Section 5.2, listing our most important contributions in Section 5.3 and future work in Section 5.4.

5.1 Summary

Although crowdsourcing is a scalable solution, when we are dealing with enriching and annotating different datasets with millions of data units, several factors introduce uncertainty into the costs estimation and completion timelines. As a result, in this dissertation, we study the estimated crowd effort and crowd throughput in order to give managers a reliable solution to improve their pricing estimation, crowd management and create trustworthy expectations.

We start by defining and measuring **crowd effort** in hours in regard to **RQ1**. This metric brings a new perspective of throughput measurement in the research crowdsourcing community. It varies according to contributors characteristics (e.g. expertise), task cognitive load (e.g. task complexity) and length of the information to be processed.

Thereafter, we have computed the *speed on task* in tokens per minute. This new metric can abstract the size of a certain task focusing on the trade-off between task complexity and contributor ability to execute the task. While studying the speed on task, we established an admissible range of speeds between 5 and 223 tokens/min, discarding tasks out of the range. This classification of *too slow* and *too fast* task execution can be further explored for purposes of quality assessment and spam prevention. Also, we found out a relation between the speed on task and the number of entities to tag. We use this relation to build a naive formula to predict the crowd effort needed for a job with certain characteristics. The formula achieved 19% MAPE, concluding that the *total task length* in tokens and the *number of entities* are key factors to predict the crowd effort.

Concerning **RQ2** our goal is to forecast the **crowd throughput** for an on-going NET job, for an horizon of 24 hours. To proceed with the crowd throughput research, we start by creating a time series dataset and exploring its behaviour. We discovered that different jobs have different trend and seasonal patterns influenced by their own requirements (e.g. language proficiency), crowd availability and characteristics among other unpredictable factors.

Within the crowd throughput research, we study the relation between the estimated crowd effort and the job's real-world length to assess the completion times unpredictability. We verified that the job's real-world length has a significant variance considering that it can vary between *2.5 times* slower to *2 times* faster than the estimated crowd effort.

To study the factors that influence the throughput, we computed the *mean daily throughput by the contributor* (in hours/contributor per day) of a given job. This metric is also useful to the crowdsourcing community considering that it gives insight into the contributors work patterns and engagement. We have concluded that on average contributors work between half an hour to two hours per day in any job, with a mean daily throughput of 1.3 hours/contributor.

To forecast the crowd throughput for an on-going job we decided to use time series so we could find patterns and trends useful to analyse crowd behaviours in real-time. We verified that each job has different trends and seasonal patterns, with frequent abrupt changes. We have then assessed four models: persistence (baseline), seasonal ARIMA/ARIMAX and Holt-Winter's. These methods were chosen given the seasonality observed when exploring their behaviour. In the end, the model that achieved the best performance was the *Additive Holt-Winter's* with *0.39 RMSLE* when forecasting a 24 period horizon and *0.198 RMSLE* when forecasting one step ahead.

5.2 Limitations

This thesis was based on Named Entity Tagging jobs. As a result, the cognitive load features are specific for this job type and the size of the information for text-enrichment jobs. However, the micro-task crowdsourcing dataset annotation has several types as audio and image. Consequently, in order to generalize the crowd effort estimation, new features associated to the cognitive load and size of the information must be add.

Then, we only considered jobs having text from Latin alphabet languages and with more than 1000 tasks to be completed, in order to deal with great crowd effort, which reduced the amount of tasks and jobs available. As a result, our last limitation that we faced during this thesis was the amount of data. While the number of tasks was approximately one million, in the end we were limited to 18 jobs and consequently 18 time series that is not sufficient to capture all different time series patterns.

5.3 Conclusions

Our approach was successful in answering our two research questions.

RQ1 : *How to measure the crowd effort of a Named Entity Tagging job?*

To measure the crowd effort of a NET job, that is, the cumulative time that the crowd must put in to complete the job, we extracted the features related to the cognitive load (number of entities), the size of the tasks and the average speed on task (tokens/minute). Then, we relate these features into a naive formula that attained 19% MAPE estimating the crowd effort in hours.

This estimated effort gives a viewpoint of job length and complexity, solving problems such as estimating job pricing and making sure contributors are paid fairly.

RQ2 : *How to forecast the foreseeable crowd throughput for a ongoing Named Entity Tagging job considering its historical data?*

To forecast the crowd throughput for an ongoing NET job, we measured the crowd effort in spans of time of one hour and established a forecasting horizon of 24 hours, i.e. one day. We attained 0.39 RMSLE using Additive Holt-Winter's to forecast a 24 period horizon of crowd throughput. We verify that different jobs have different throughput patterns over time such as seasonality our trend. Also, abrupt changes are frequent and we deduce that they are related to deadline adjustments and crowd management in real time. Additionally, we verify that the number of active contributors influences the crowd throughput and that the results deteriorate over the forecasting horizon, that is, the next period forecasting is more reliable than the last period.

We conclude that an accurate throughput estimation allows managing expectations even when completion timelines are unpredictable. As an example, if the expected throughput is below the estimated timeline, it is possible to react and intervene, generating some type of incentive that will realign the crowd throughput with some desired deadline/timeline.

All in all, we met the objectives defined in Section 1.5, created new and important contributions to the crowdsourcing community and a new strategy to improve the *DefinedCrowd*'s management team.

5.4 Future work

For future work, there is room for improvement concerning the **crowd effort** estimation by:

- Use machine learning techniques to predict the crowd effort. Our baseline shows a strong correlation between the extracted features. As a result, a machine learning model would improve the performance achieved;
- Assessing natural language related problems such as readability. The task cognitive load is influenced by its inherent complexity. A hypothesis is that if the task is difficult to read, is also difficult to interpret leading to higher time on task. If we could prove this hypothesis, the cost estimation would be improved. An example of readability test is the Flesh-Kincaid [SZGG17];
- Comparing these studies to other than Latin alphabets jobs. The job's languages are one of our limitations as the data received was related to only eight languages. The language requirement affects directly the crowd eligibility. It would be interesting to understand the relationship between different languages and crowd effort.

Concerning the **crowd throughput** forecasting:

- Add contributor's personal information. We create the hypothesis that different contributors have different speed on task and availability, which influences directly the throughput. The contributors' seasonality (i.e. works every day) or expertise (e.g. works only in some type of jobs), affects both the time on task and the job completion timeline. As a result, this information would improve our throughput predictability;
- Assess ensemble methods. When exploring our time series, we realise that there are groups of time series, e.g. daily seasonal and without trend or seasonality. As a result, we create the hypothesis that having an ensemble of two models that are optimal at each time series type would improve our performance;
- Study the model deterioration analysing each step forecasting error from one step ahead to 24 steps;
- Implement the business evaluation strategy described in Figure 4.24.

Another possibility for future work would be generalise our problem to include other than NET jobs. Such generalisation would create more data and consequently would need more feature work and data cleaning.

In the end, we consider that crowd effort and throughput forecasting are promising solutions to be applied in crowdsourcing platforms. This tool enhance the project managers practice and can reduce business costs. Nonetheless, further exploration on this topic is fundamental to enhance the results achieved.

References

[ABI+13] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2):76–81, 2013.

[AMM+08] Luis Von Ahn, Benjamin Maurer, Colin Mcmillen, David Abraham, and Manuel Blum. reCAPTCHA : Human-Based Recognition via Web Character. *Science*, 321(5895):1465–1468, 2008.

[AVNSB+13] Kittur Aniket, Jeffrey V. Nickerson, Michael S. Bernstein, Elizabeth M. Gerber, Aaron Shaw, John Zimmerman, Matthew Lease, and John J. Horton. The Future of Crowd Work. *CSCW*, pages 1301–1317, 2013.

[BB12] Christoph Bergmeir and José M. Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

[Bos14] A. Boschetti. *Python Data Science Essentials*. Number 1. 2014.

[CTB15] Justin Cheng, Jaime Teevan, and Michael S. Bernstein. Measuring Crowdsourcing Effort with Error-Time Curves. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, pages 1365–1374, 2015.

[DCD+15] Djellel Eddine Difallah, Michele Catasta, Gianluca Demartini, Panagiotis G. Ipeirotis, and Philppe Cudre-Mauroux. The Dynamics of Micro-Task Crowdsourcing. *Www*, pages 238–247, 2015.

[DF79] David A. Dickey and Wayne A. Fuller. Distribution of the Estimators for Autoregressive Time Series with a Unit Root. *Journal of the American Statistical Association*, 74(366a):427–431, 1979.

[DFI18] Djellel Difallah, Elena Filatova, and Panos Ipeirotis. Demographics and Dynamics of Mechanical Turk Workers. (August):135–143, 2018.

[DKC+18] Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh. Quality Control in Crowdsourcing: A Survey of Quality Attributes, Assessment Techniques and Assurance Actions. 51(1), 2018.

[Dow12] Allen B Downey. *Think Stats - Exploratory Data Analysis*. 2012.

[DRH11] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing systems on the World-Wide Web. *Communications of the ACM*, 54(4):86, 2011.

[DVNDV$^+$13] Triparna De Vreede, Cuong Nguyen, Gert Jan De Vreede, Imed Boughzala, Onook Oh, and Roni Reiter-Palmon. A theoretical model of user engagement in crowdsourcing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8224 LNCS:94–109, 2013.

[EA12] Philippe Esling and Carlos Agon. Time-Series Data Mining. *ACM Computing Surveys*, 45(1):34, 2012.

[EdV13] Carsten Eickhoff and Arjen P. de Vries. Increasing cheat robustness of crowdsourcing tasks. *Information Retrieval*, 16(2):121–137, 2013.

[EGWW10] Mark Everingham, Luc Van Gool, Christopher K I Williams, and John Winn. The P ASCAL Visual Object Classes (VOC) Challenge. pages 303–338, 2010.

[FKTC13] Ailbhe Finnerty, Pavel Kucherbaev, Stefano Tranquillini, and Gregorio Convertino. Keep it simple: Reward and Task Design in Crowdsourcing. *Proceedings of the Biannual Conference of the Italian Chapter of SIGCHI on - CHItaly '13*, (September):1–4, 2013.

[GKD14] Ujwal Gadiraju, Ricardo Kawase, and Stefan Dietze. A taxonomy of microtasks on the web. pages 218–223, 2014.

[GKDD15] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. Understanding Malicious Behavior in Crowdsourcing Platforms. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, pages 1631–1640, 2015.

[GS14] David Geiger and Martin Schader. Personalized task recommendation in crowdsourcing information systems - Current state of the art. *Decision Support Systems*, 65(C):3–16, 2014.

[GYB17] Ujwal Gadiraju, Jie Yang, and Alessandro Bozzon. Clarity is a Worthwhile ality – On the Role of Task Clarity in Microtask Crowdsourcing. pages 5–14, 2017.

[HAGM15] Daniel Haas, Jason Ansel, Lydia Gu, and Adam Marcus. Argonaut : Macrotask Crowdsourcing for Complex Data Processing. *Vldb*, 8(12):1642–1653, 2015.

[HC13] Umair ul Hassan and Edward Curry. A Capability Requirements Approach for Predicting Worker Performance in Crowdsourcing. *Proceedings of the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pages 429–437, 2013.

[HK14] Rob J Hyndman and Andrey V. Kostenko. Minimum sample size Requirements for seasonal forecasting models. 2014.

[HKH$^+$14] Tobias Hossfeld, Christian Keimel, Matthias Hirth, Bruno Gardlo, Julian Habigt, Klaus Diepold, and Phuoc Tran-Gia. Best practices for qoe crowdtesting: Qoe assessment with crowdsourcing. *IEEE Transactions on Multimedia*, 16(2):541–558, 2014.

[How12] By Jeff Howe. The Rise of Crowdsourcing. (14):1–5, 2012.

[HSH^{+}14] Matthias Hirth, Sven Scheuring, Tobias Hossfeld, Christian Schwartz, and Phuoc Tran-Gia. Predicting result quality in Crowdsourcing using application layer monitoring. *2014 IEEE 5th International Conference on Communications and Electronics, IEEE ICCE 2014*, (July):510–515, 2014.

[HV12] Chien-Ju Ho and Jennifer Wortman Vaughan. Online Task Assignment in Crowdsourcing Markets. *Proceedings of the 2012 AAAI Conference on Artificial Intelligence*, pages 45–51, 2012.

[Jia16] Yichuan Jiang. A Survey of Task Allocation and Load Balancing in Distributed Systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):585–599, 2016.

[JLZ^{+}11] Ou Jin, Nathan N. Liu, Kai Zhao, Yong Yu, and Qiang Yang. Transferring topical knowledge from auxiliary long texts for short text clustering. page 775, 2011.

[JSPW17] Ayush Jain, Akash Das Sarma, Aditya Parameswaran, and Jennifer Widom. Understanding Workers, Developing Effective Tasks, and Enhancing Marketplace Dynamics: A Study of a Large Crowdsourcing Marketplace. 10(7):829–840, 2017.

[KCS08] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with Mechanical Turk. *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, page 453, 2008.

[KS18] Ayswarya R Kurup and G P Sajeev. A Trace Driven Analysis of Incentive Based Crowdsourcing Workload. *2018 International Conference on Data Science and Engineering (ICDSE)*, pages 1–6, 2018.

[KTK12] Hiroshi Kajino, Y Tsuboi, and H Kashima. A Convex Formulation for Learning from Crowds. *Aaai*, pages 73–79, 2012.

[KYA^{+}14] Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker, Targio Hashem, Zakira Inayat, Waleed Kamaleldin, Mahmoud Ali, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani. Big Data : Survey , Technologies , Opportunities , and Challenges. *The Scientific World Journal*, 2014:18, 2014.

[LM^{+}14] Tsung-Yi Lin, , Michael Maire, , Serge Belongie, , James Hays, , Pietro Perona, , Deva Ramanan, , Piotr Doll{\'a}r, , C. Lawrence" Zitnick, David Editor="Fleet, , Tomas Pajdla, , Bernt Schiele, , and Tinne Tuytelaars. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, number June, pages 740–755. Springer International Publishing, Cham, 2014.

[Lu17] Yang Lu. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10, 6 2017.

[MCG^{+}17] David Martin, Sheelagh Carpendale, Neha Gupta, Tobias HoBfeld, Naderi Babak, Judith Redi, Ernestasia Siahaan, and Ina Wechsung. *Understanding the Crowd: Ethical and Practical Matters in the Academic Use of Crowdsourcing*, volume 10264. 2017.

[MSA18] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE*, 13(3):1–26, 2018.

[MW10] Winter Mason and Duncan J. Watts. Financial incentives and the "performance of crowds". *ACM SIGKDD Explorations Newsletter*, 11(2):100, 2010.

[MWH98] Spyros G. Makridakis, Steven C. Wheelwright, and Rob J. Hyndman. *Forecasting: Methods and Applications*. 1998.

[NS07] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[PGH11] Fernando Pérez, Brian Granger, and John Hunter. Python: An Ecosystem for Scientific Computing. pages 13–21, 2011.

[PS12] Marion K. Poetz and Martin Schreier. The value of crowdsourcing: Can users really compete with professionals in generating new product ideas? *Journal of Product Innovation Management*, 29(2):245–256, 2012.

[Ras15] Sebastian Raschka. *Python Machine Learning : unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. 2015.

[RKK$^+$11] Jakob Rogstadius, Vassilis Kostakos, Aniket Kittur, Boris Smus, Jim Laredo, and Maja Vukovic. An Assessment of Intrinsic and Extrinsic Motivation on Task Performance in Crowdsourcing Markets. *Fifth International AAAI Conference on Weblogs and Social Media*, page 328, 2011.

[SB13] Guido Sautter and Klemens Böhm. High-throughput crowdsourcing mechanisms for complex tasks. *Social Network Analysis and Mining*, 3(4):873–888, 2013.

[SC94] John Sweller and Paul Chandler. Why some material is Difficult to Learn. *Cognition and Instruction*, 12(3):185–233, 1994.

[SZGG17] Marina I Solnyshkina, Radif R Zamaletdinov, Ludmila A Gorodetskaya, and Azat I Gabitov. Evaluating Text Complexity and Flesch-Kincaid Grade Level. *www.jsser.org Journal of Social Studies Education Research Sosyal Bilgiler Eğitimi Araştırmaları Dergisi*, 2017(3):238–248, 2017.

[VB10] Maja Vukovic and Claudio Bartolini. Towards a research agenda for enterprise crowdsourcing. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6415 LNCS(PART 1):425–434, 2010.

[Wir] Rüdiger Wirth. CRISP-DM : Towards a Standard Process Model for Data Mining. (24959).

[YLW$^+$14] Xu Yin, Wenjie Liu, Yafang Wang, Chenglei Yang, and Lin Lu. What? How? Where? A Survey of Crowdsourcing. *Frontier and Future Development of Information Technology in Medicine and Education*, pages 221–232, 2014.

[YRDB16] Jie Yang, Judith Redi, Gianluca Demartini, and Alessandro Bozzon. Modeling Task Complexity in Crowdsourcing. *The Fourth AAAI Conference on Human Computation and Crowdsourcing*, (October):249–258, 2016.

Appendix A

Time Series Exploratory Analysis

In this Appendix we can check with detail the visualisations created during the time series exploratory analysis, including rolling statistics, autocorrelation and partial autocorrelations.

A.1 Time Series Visualisations

The next two figures shows the the throughput in human hours over jobs. Figure A.1 is the original data, while FigureA.2 corresponds to the first order differenced data. This data holds hourly observations, so the x axis corresponds to the time series steps over time.

A.2 Decomposition Plots

The next three figures represent the decomposition plots from our 18 time series. The decomposition plots are composed by: the observed plot, the captured trend, the seasonal pattern and the residual. We are dealing with seasonal patterns of 24 periods length.

A.3 Rolling Statistics

In this section, we show the rolling statistics visualisations both in the original (see FigureA.6) and first order differenced (see FigureA.7) series. In order to calculate these statistics we a moving window of time, and calculate the mean and standard deviations of that time period as the current value. In our case, with hourly observations and the suspicion of daily seasonality, our window is of 24 steps. Each time series is one job with hourly observations of the throughput in hours.

A.4 ACF and PACF Plots

In this section, we show the ACF (figures A.8 and A.9) and PACF (figures A.10 and A.11) visualisations both in the original and first order differenced series. We limit the autocorrelation plots to 50 lags and the partial autocorrelation plots to 24 lags, just for visualisation purposes.

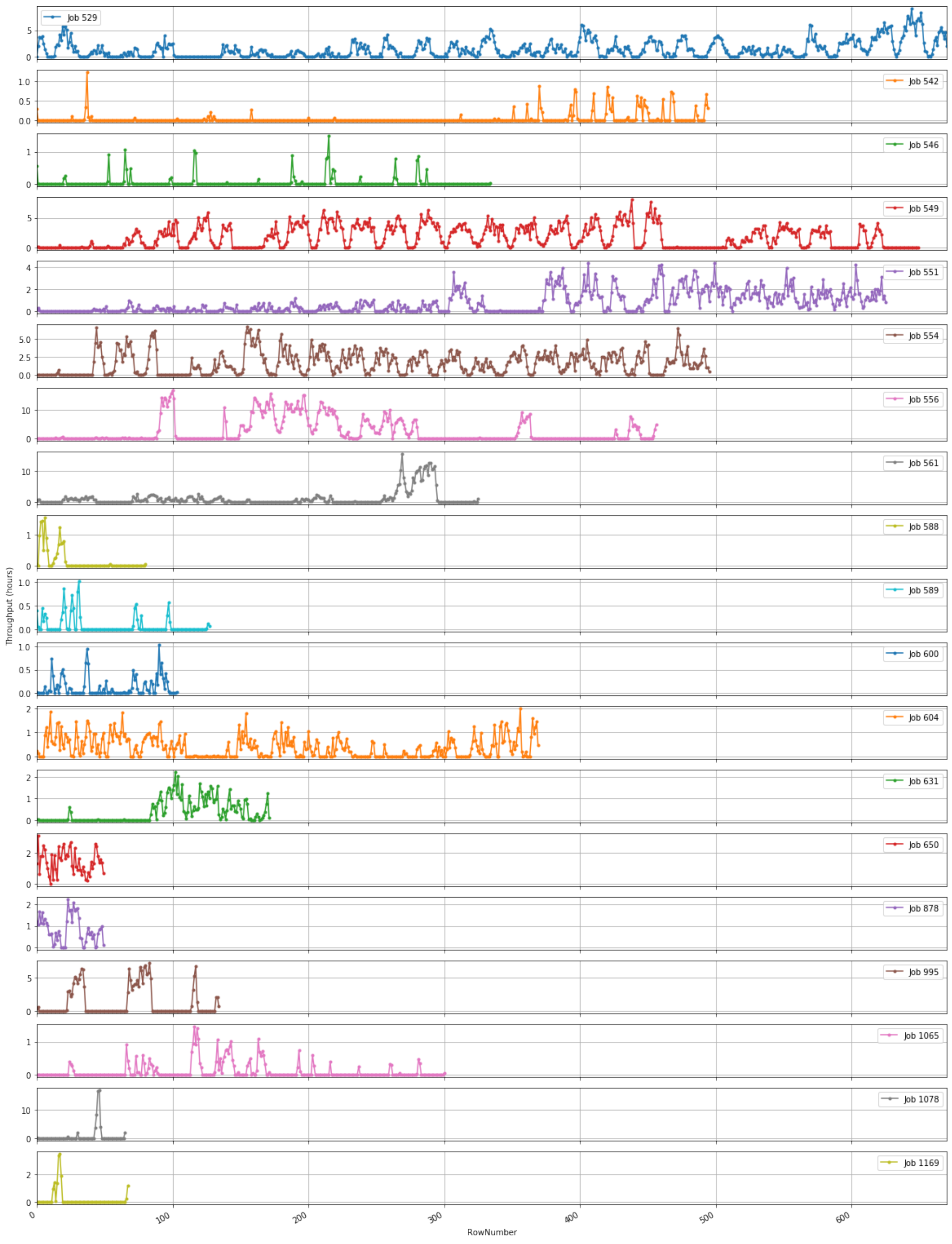


Figure A.1: Time Series throughput in hours over jobs with hourly observations

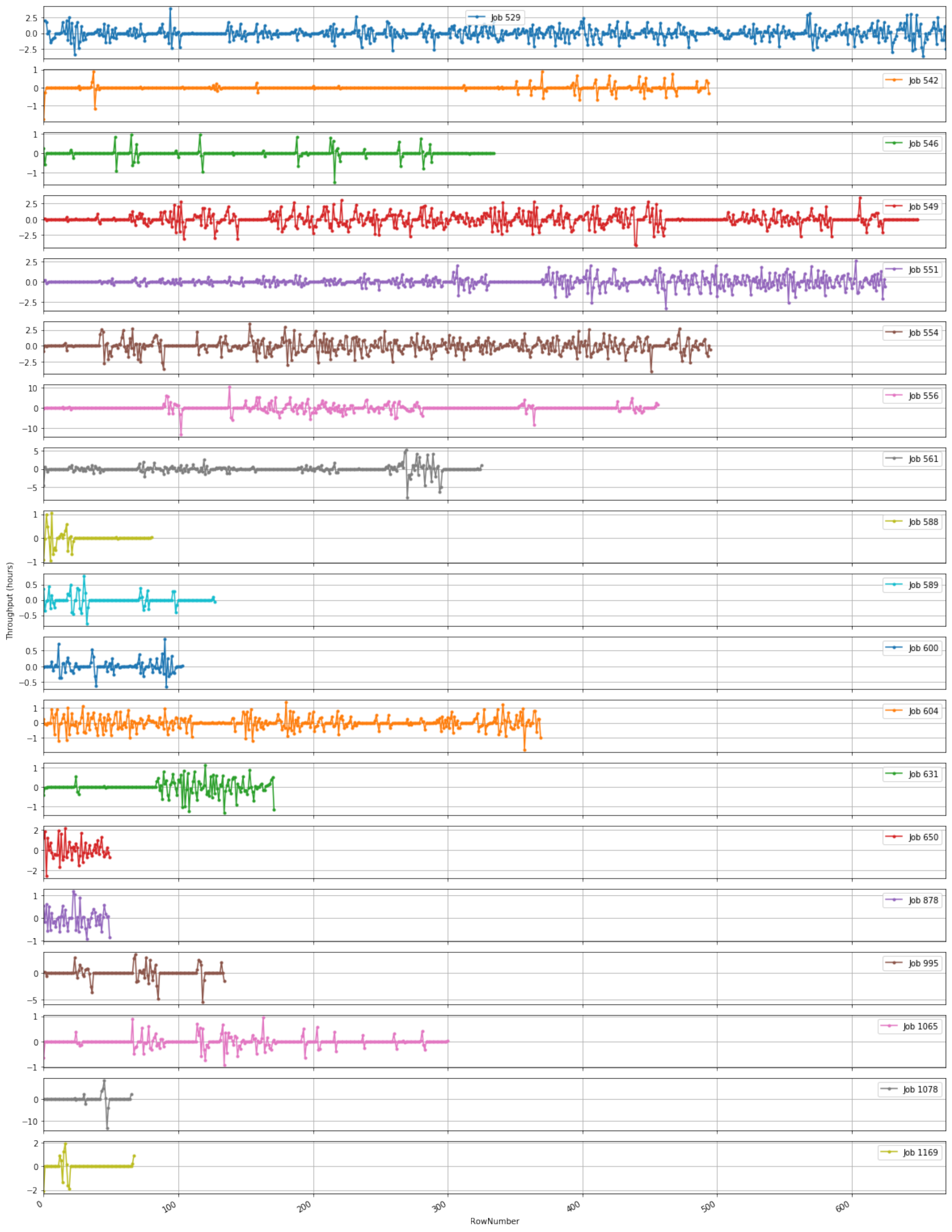


Figure A.2: First order differenced series throughput in hours over jobs with hourly observations

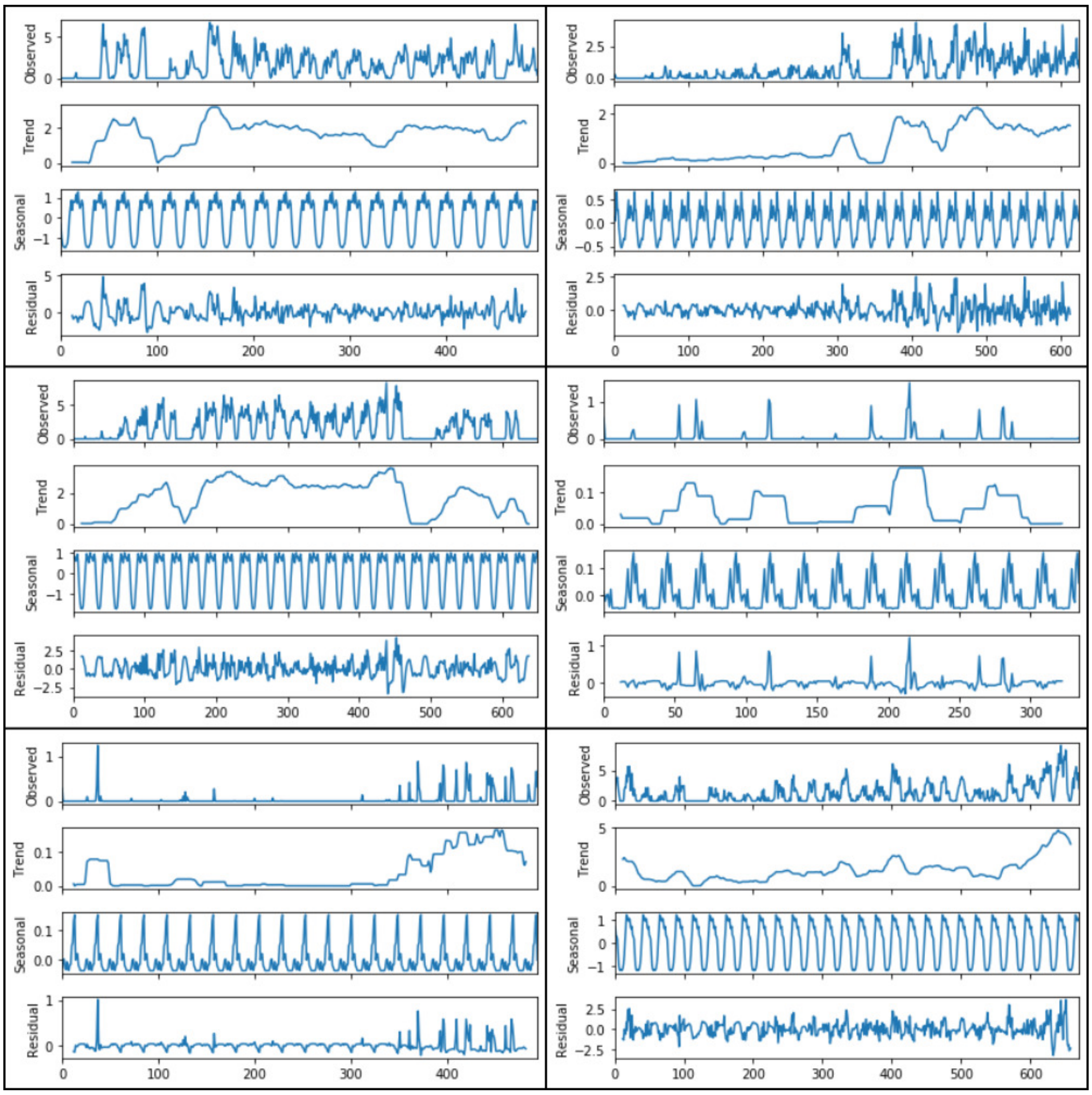


Figure A.3: Six jobs additive decomposition plots into trend, seasonal and residual time series (part 1/3)

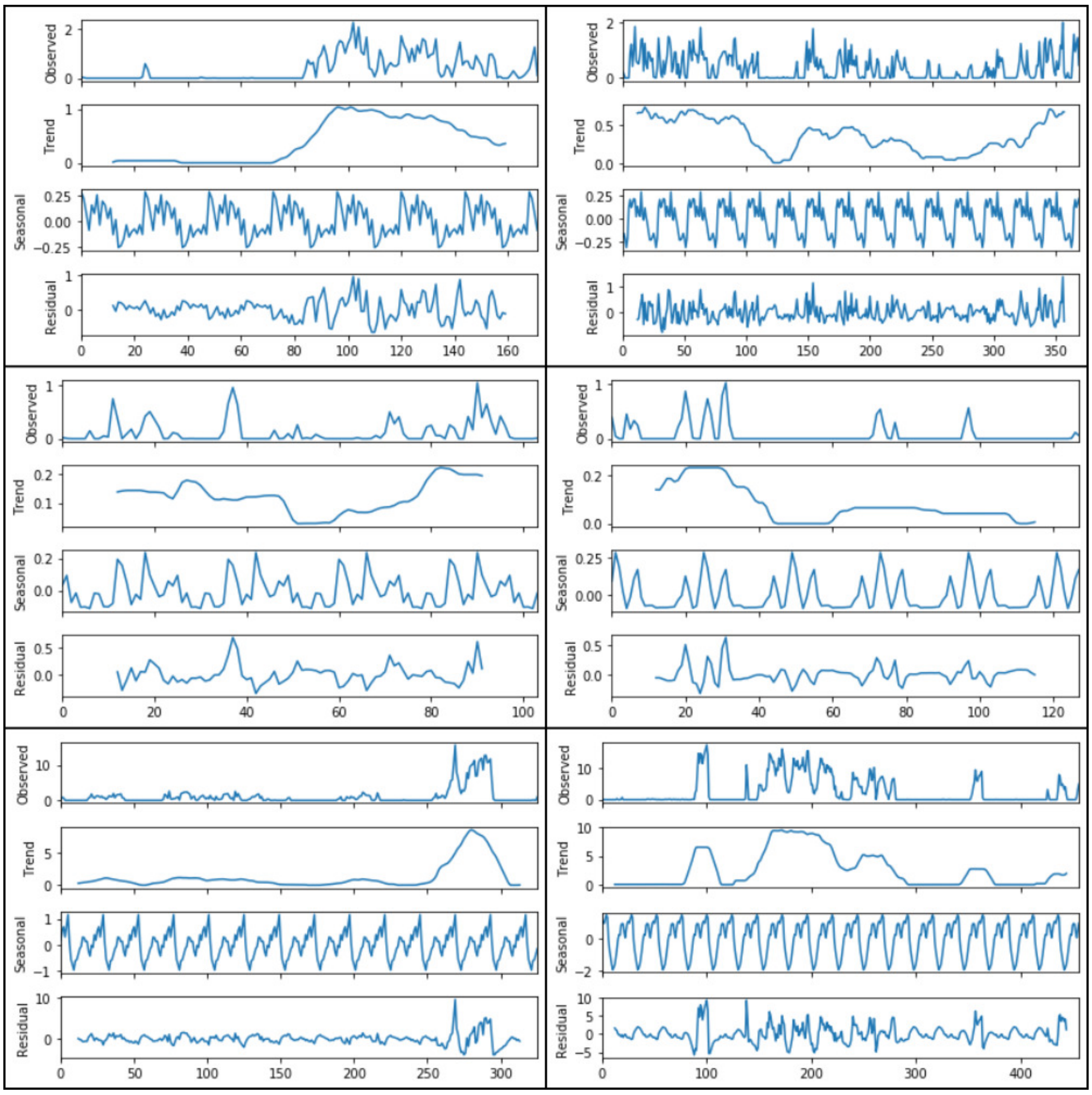


Figure A.4: Six jobs additive decomposition plots into trend, seasonal and residual time series (part 2/3)

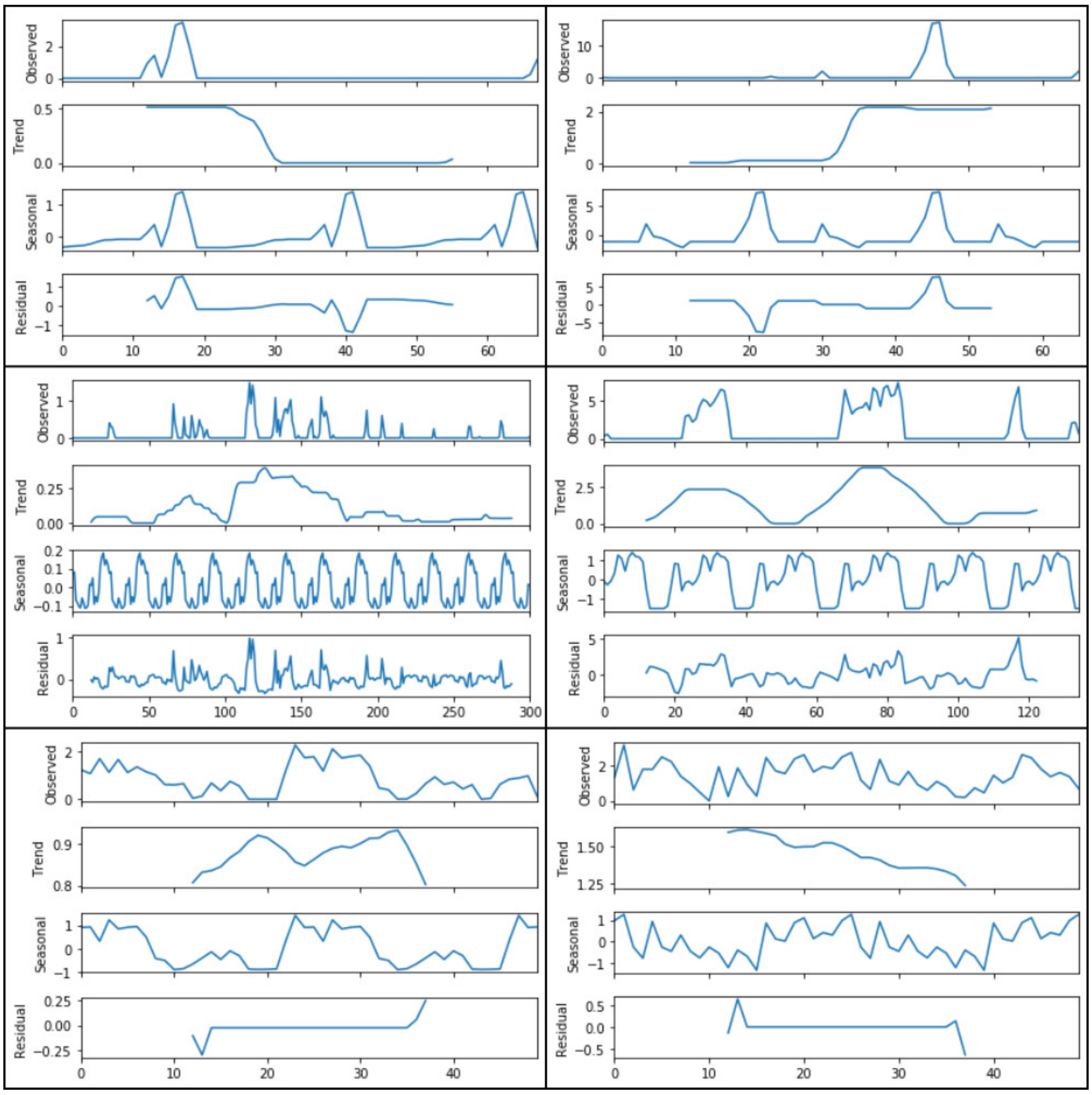


Figure A.5: Six jobs additive decomposition plots into trend, seasonal and residual time series (part 3/3)

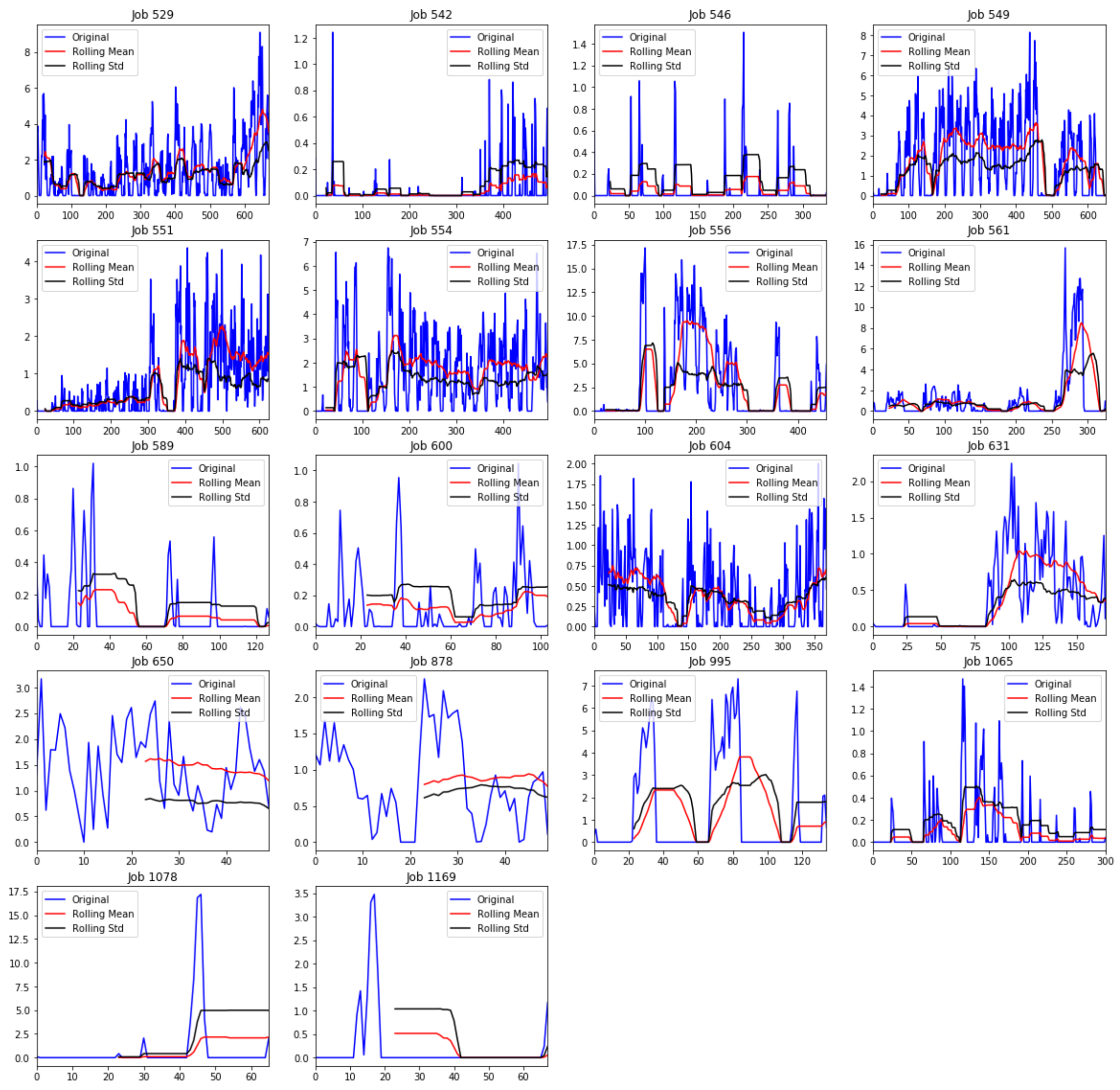


Figure A.6: Rolling mean, rolling standard deviation and original time series

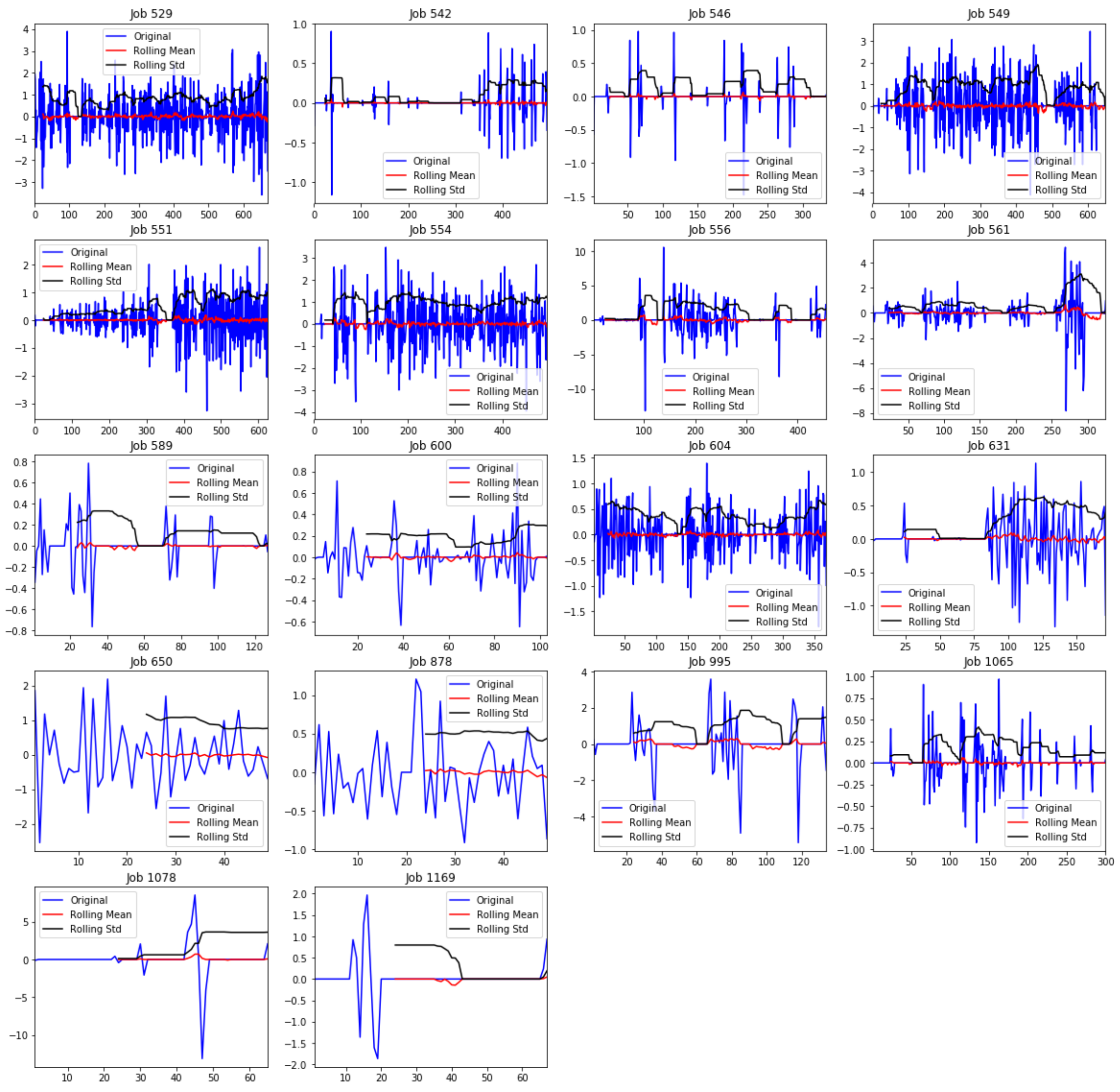


Figure A.7: Rolling mean, rolling standard deviation and first order differenced series

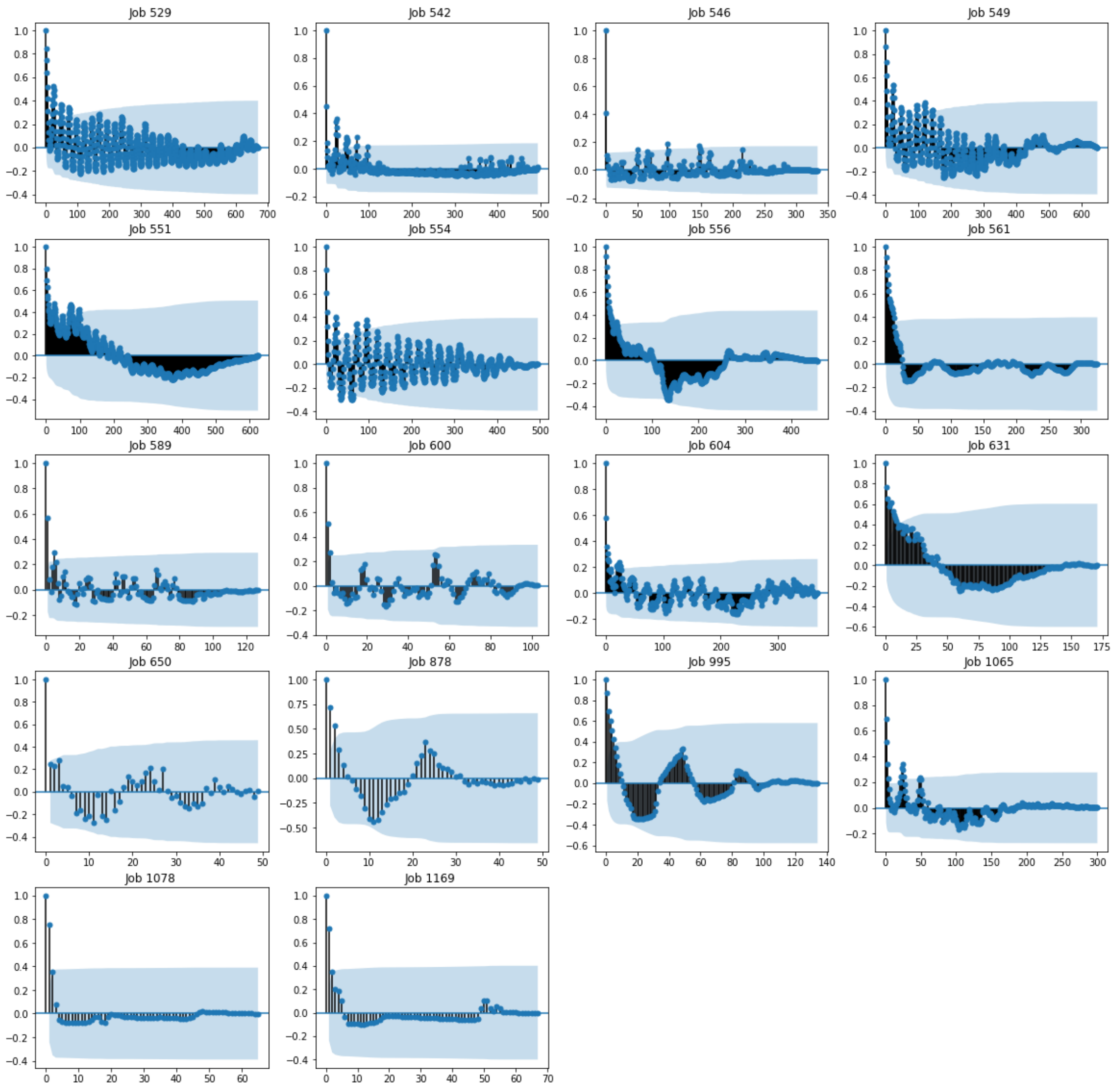


Figure A.8: Autocorrelation plots per each time series with maximum of 50 lags

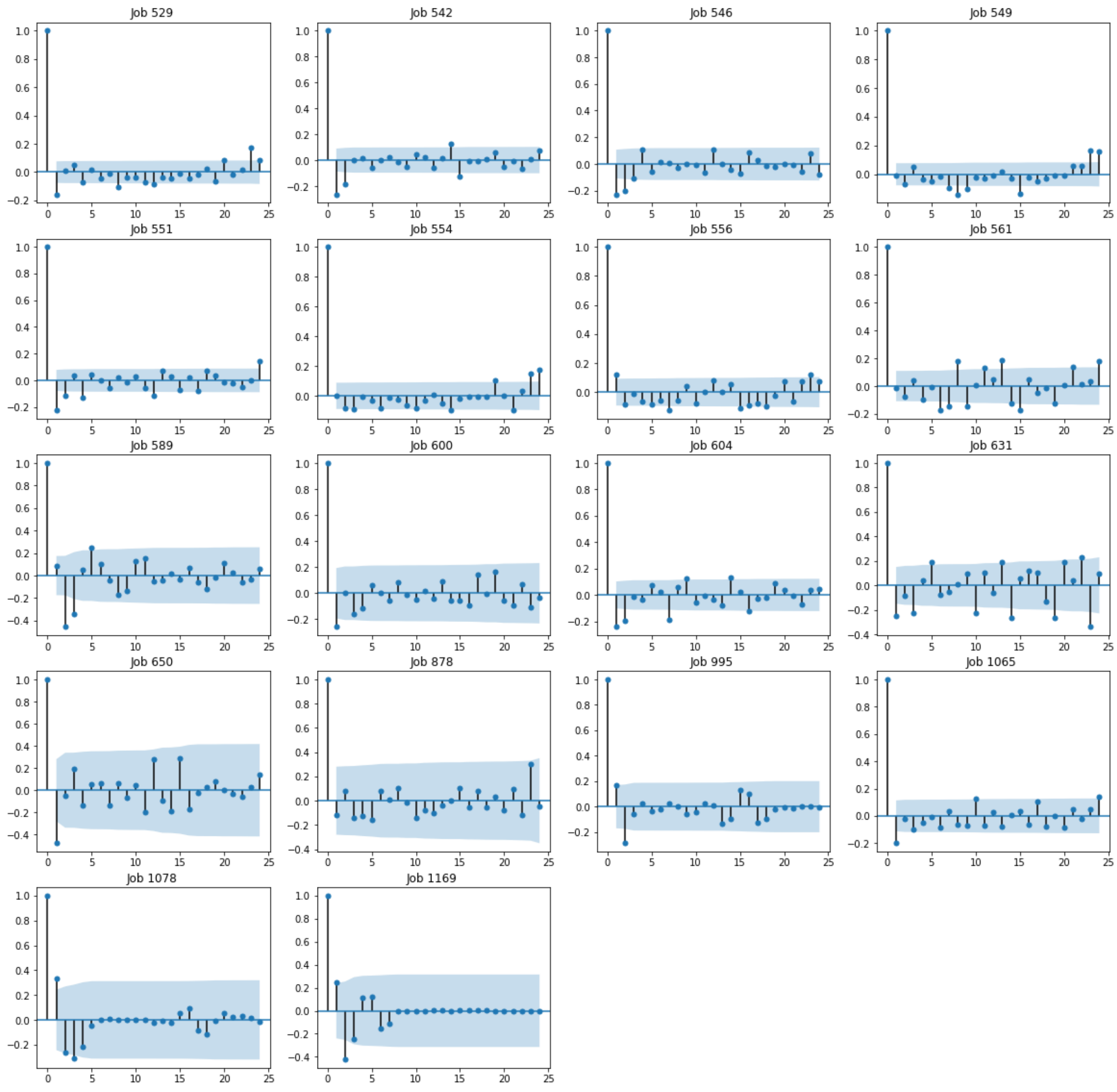


Figure A.9: Autocorrelation plots per each first order differenced series with maximum of 50 lags

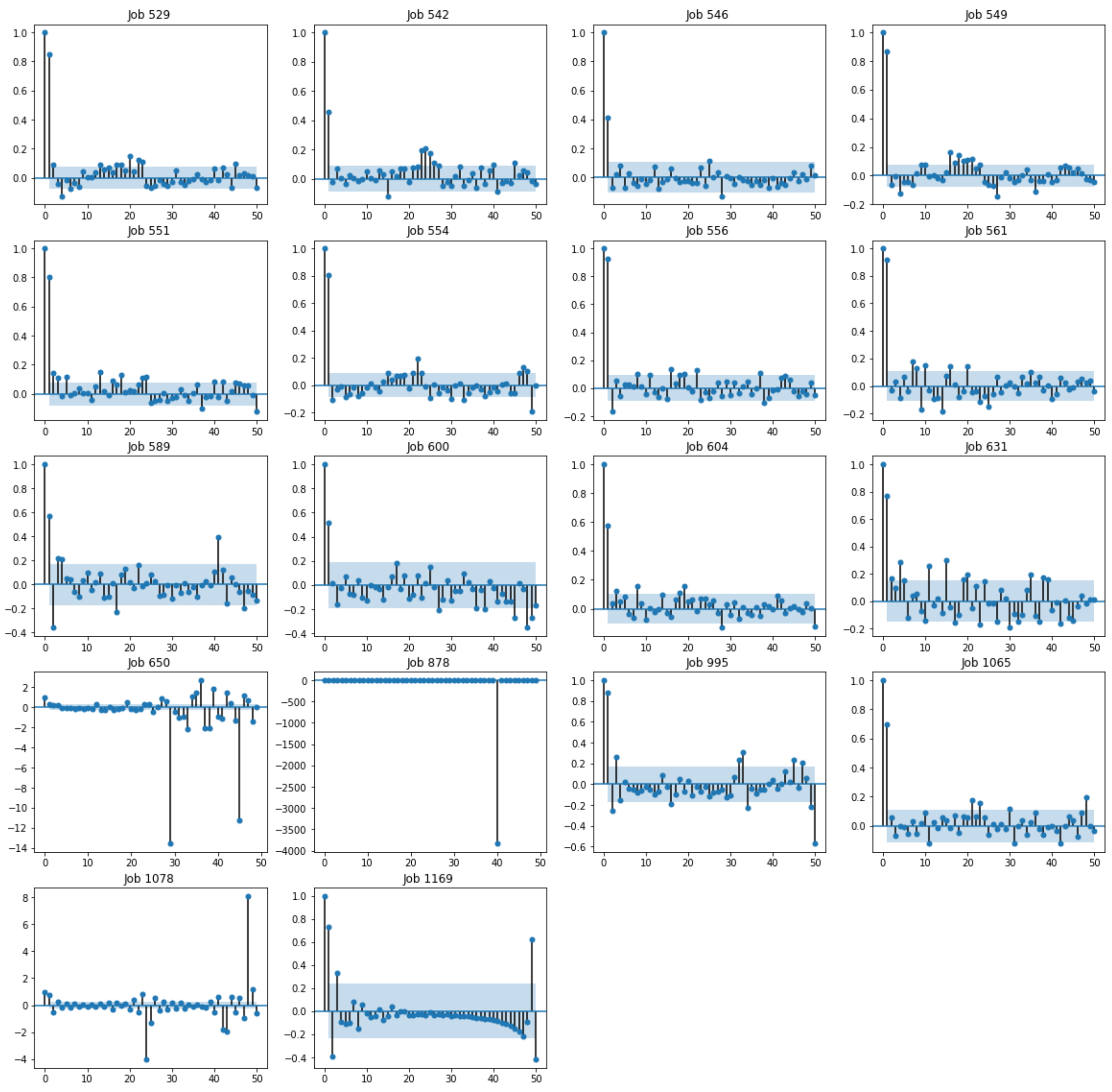


Figure A.10: Partial autocorrelation plots per each time series with maximum of 24 lags

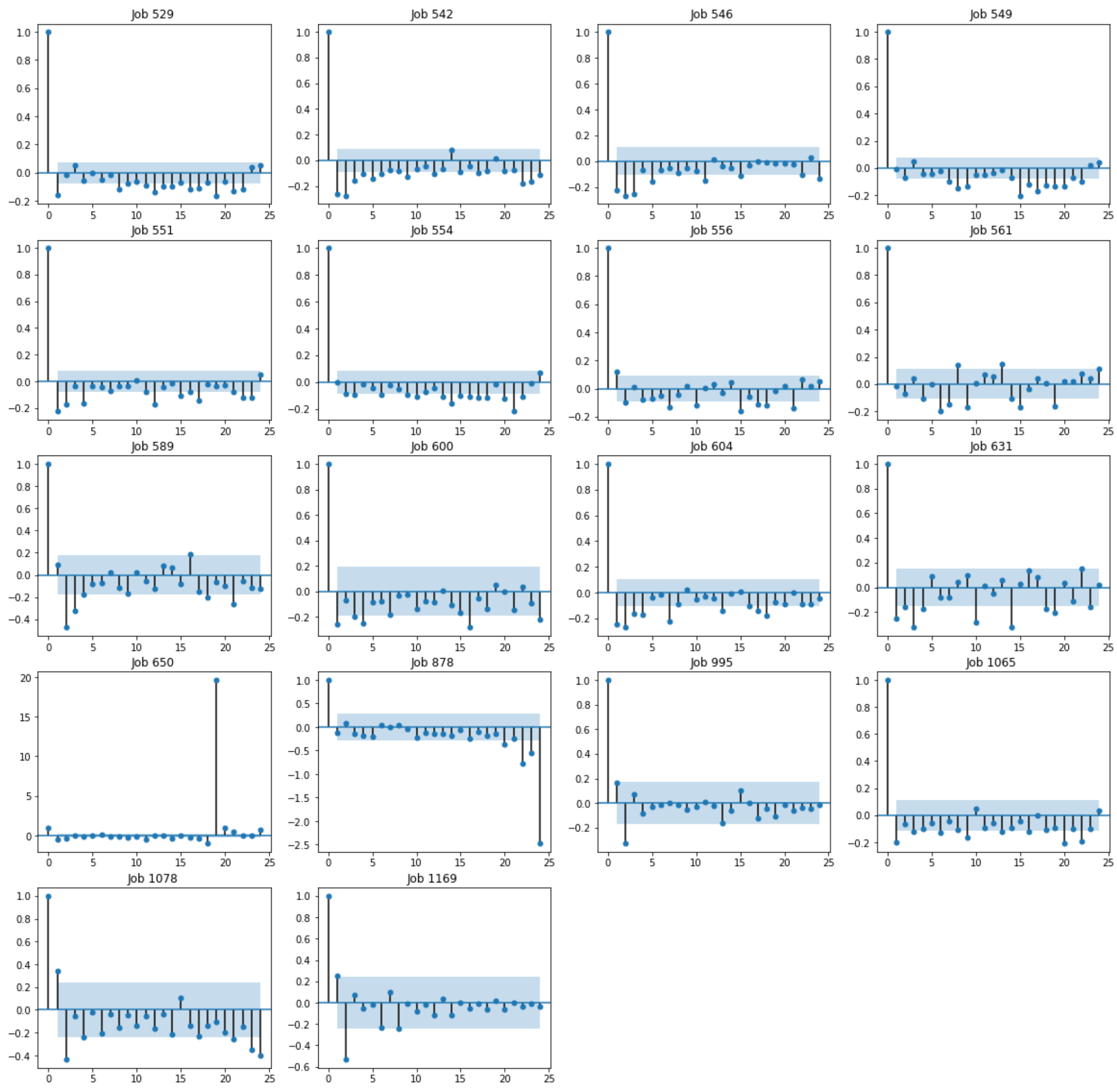


Figure A.11: Partial autocorrelation plots per each first order differenced series with maximum of 24 lags

Appendix B

Modelling Results

In this Appendix we can check with detail the visualisations created during the modelling and evaluation phases, including grid search tables, model fit evaluation and forecasts.

B.1 Grid Search Results

According to Makridakis, the process of identifying a Box-Jenkins ARIMA model requires experience and good judgment, so there are some rules of the thumb:

1. Make the series stationary
2. Consider non-seasonal aspects
3. Consider seasonal aspects

As a result, to create our seasonal ARIMA model, we start by finding the order values for the ARIMA (Table B.1), and then find the seasonal order (Table B.2) considering the previous ARIMA orders.

B.2 Holt-Winter's Parameters

Holt-Winter's Seasonal method has four parameters that are optimised in order to fit the time series. Per each time series, Holt-Winter's optimises the parameters accordingly. The next tables present the optimal values per each time series fold.

B.3 Diagnostic Checking

In this section, we have further visualisations concerning the seasonal ARIMA/ARIMAX and Holt-Winter's Exponential Smoothing models fitting. Per each model, we show the obtained visualisations of train set versus fitted values (see Figures B.1, B.4 and B.7) in order to conclude about

Table B.1: Grid Search results for ARIMA(p,d,q) order values

p	**d**	**q**	**AIC**
0	1	0	498.66
0	1	1	474.17
0	1	2	465.26
1	1	0	486.95
2	1	0	478
2	1	1	458.3
3	1	0	475.27
3	1	1	461.48
4	1	0	473.03
4	1	1	456.54
5	1	0	473.15
5	1	1	452.58
6	1	0	472.79
6	1	1	451.39
6	1	2	451.26
7	1	0	471.72
7	1	**1**	452.26

the goodness of the fit. Each plot corresponds to one fold, with hourly observations of the throughput in hours. The orange line corresponds to the original data while the blue line corresponds to the fitted values by the model.

Another method to check the goodness of the fit, when using Box-Jenkins ARIMA models, is to check the residuals normality. Figures B.2 and B.3 shows the $ARIMA(6,1,2)(2,1,0)_{24}$ diagnostics plots. Figures B.5 and B.6 shows the $ARIMAX(6,1,2)(2,1,0)_{24}$ diagnostics plots. Each diagnostic plot is composed by (ordered clockwise from top left) [1]:

- Standardized residuals over time
- Histogram plus estimated density of standardised residuals, along with a Normal(0,1) density plotted for reference
- Normal Q-Q plot, with Normal reference line
- Correlogram

B.4 Forecasting Results

In this section, we present the forecasting plots per each fold and per each model. The time series hold hourly observations of the throughput in hours. The orange line corresponds to the training set (train + validation) and the blue line to the test set. The green line corresponds to the forecast values.

[1]https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.mlemodel.MLEResults.plot_diagnostics.html

Table B.2: Grid Search results for $ARIMA(p,d,q)(P,D,Q)_{24}$ order values

p	**d**	**q**	**P**	**D**	**Q**	**AIC**
6	1	2	0	1	0	567.4
6	1	2	0	1	1	625.37
6	1	2	1	1	0	455.04
6	1	2	1	1	1	475.4
6	1	2	2	1	0	386.14
6	1	2	2	1	1	403.0

Table B.3: Holt-Winter's optimal parameters per fold (part 1/3)

α	0.539	0.601	0.625	0.052	0.052	0.052	0.052	0.052	0.801	0.721	0.780
β	0.020	0.029	0.095	0.052	0.052	0.052	0.052	0.052	0.053	0.047	0.251
γ	0	0	0	0.157	0.157	0.263	0.157	0.210	0	0	0
ϕ	0	0	0	0	0	0	0	0	0	0	0

Table B.4: Holt-Winter's optimal parameters per fold (part 2/3)

α	0.052	0.675	0.604	0.894	0.894	0.894	0.988	0.992	0.994	0.583	0.598
β	0.052	0.625	0.052	0.894	0.627	0.627	0.894	0.988	0.992	0.994	0.052
γ	0.263	0	0	0.105	0.105	0.105	0	0	0	0	0
ϕ	0	0	0	0.173	0.096	0.131	0.114	0.115	0.104	0.016	0.001

Table B.5: Holt-Winter's optimal parameters per fold (part 3/3)

α	0.894	0.222	0.252	0.239	0.157	0.631	0.536	0.894	0.759	0.723	0.697
β	0.894	0.052	0.030	0.052	0.157	0.472	0.472	0.894	0.031	0.524	0.365
γ	0.105	0	0	0	0.315	0	0	0.105	0	0	0
ϕ	0.130	0	0	0	0.105	0	0	0.097	0	0	0

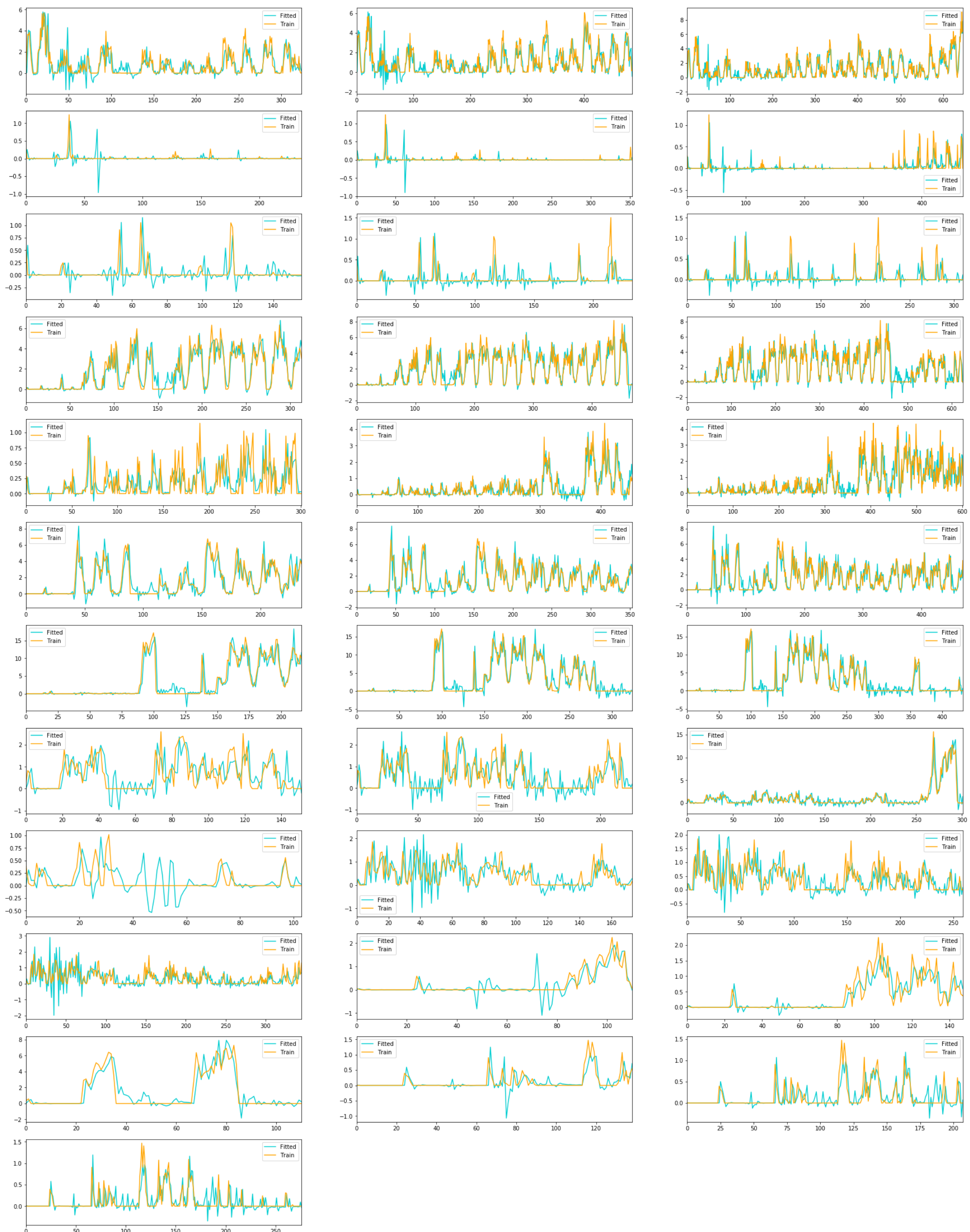


Figure B.1: Training values (orange) versus fitted values (blue) by $ARIMA(6,1,2)(2,1,0)_{24}$ considering all folds

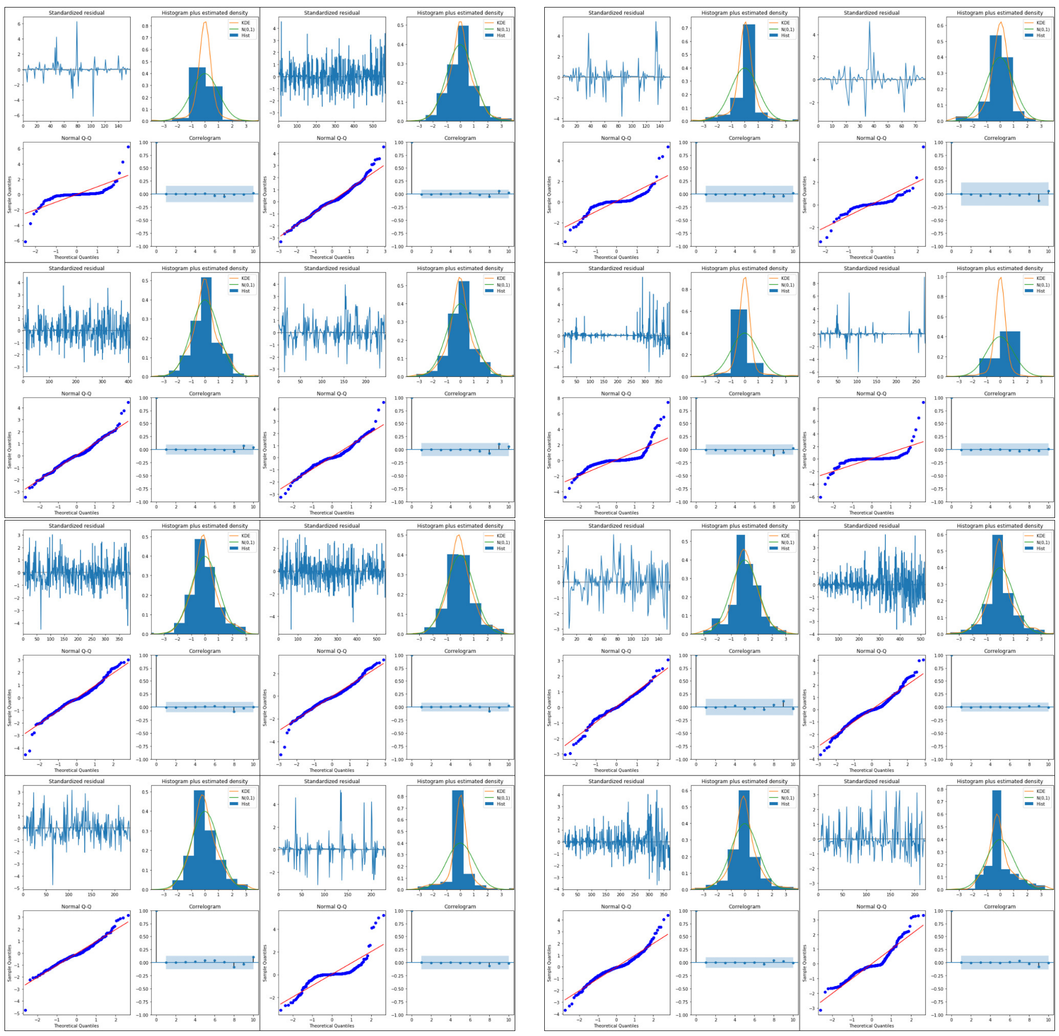


Figure B.2: 16 $ARIMA(6,1,2)(2,1,0)_{24}$ model diagnostic plots (part 1/2)

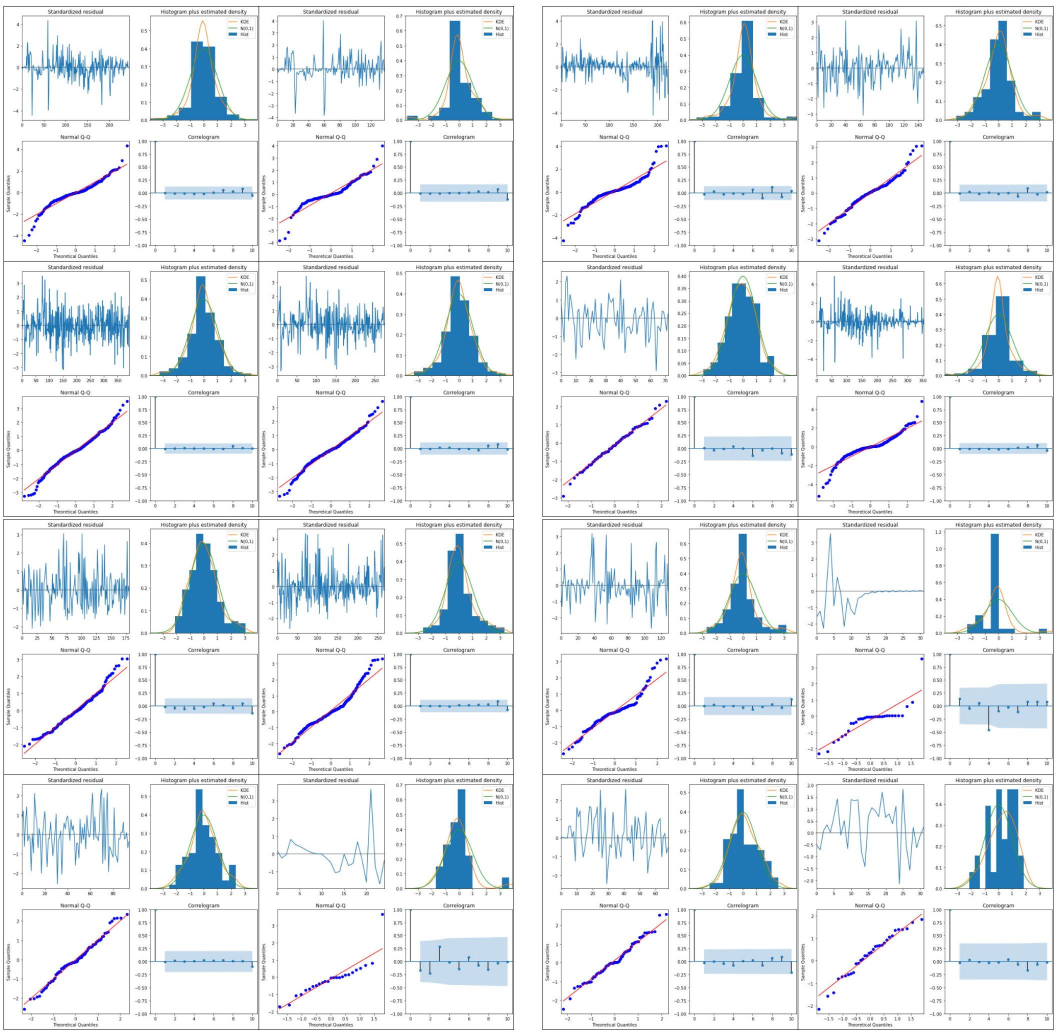


Figure B.3: 16 $ARIMA(6,1,2)(2,1,0)_{24}$ model diagnostic plots (part 2/2)

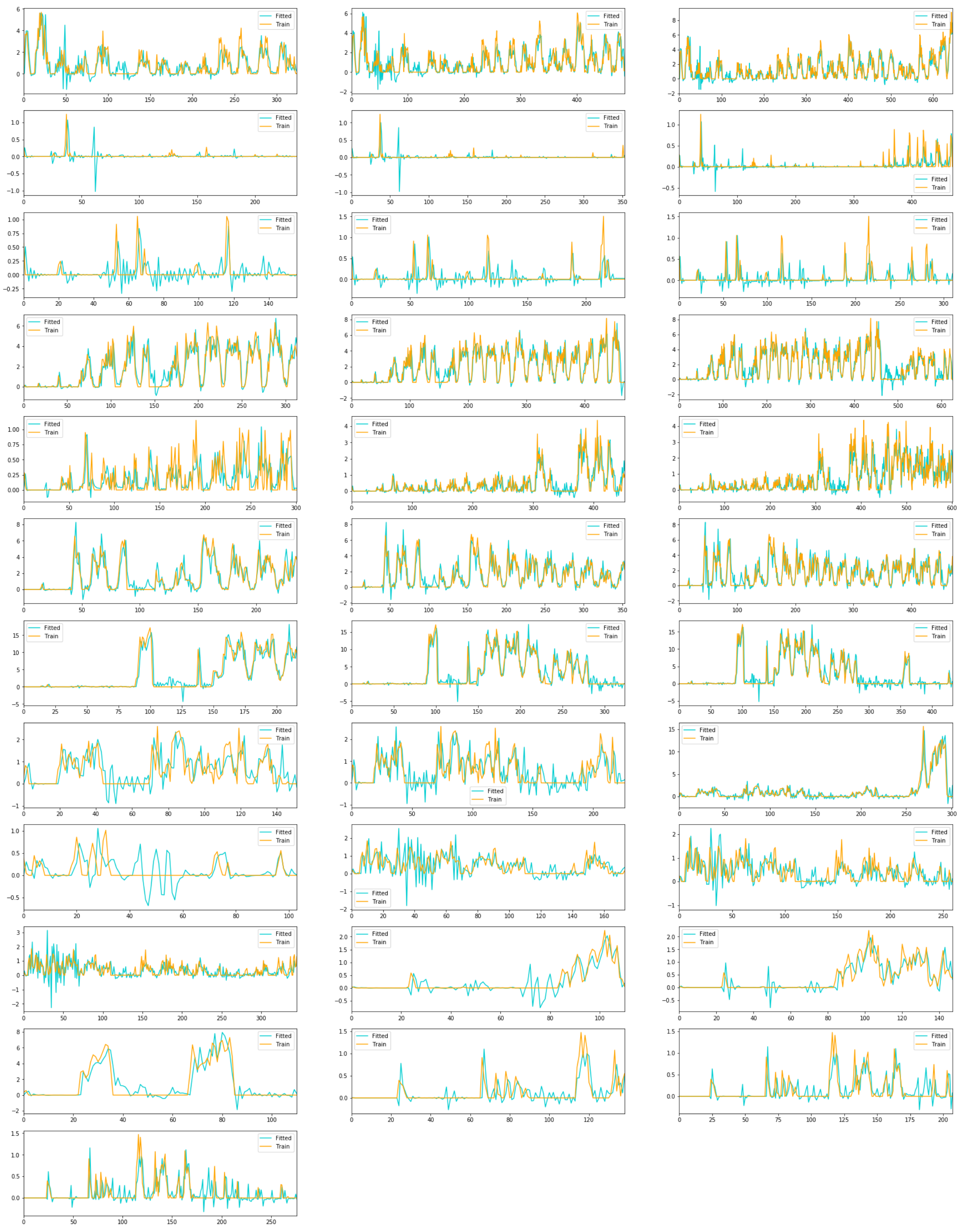


Figure B.4: Training values (orange) versus fitted values (blue) by $ARIMAX(6,1,2)(2,1,0)_{24}$ model considering all folds

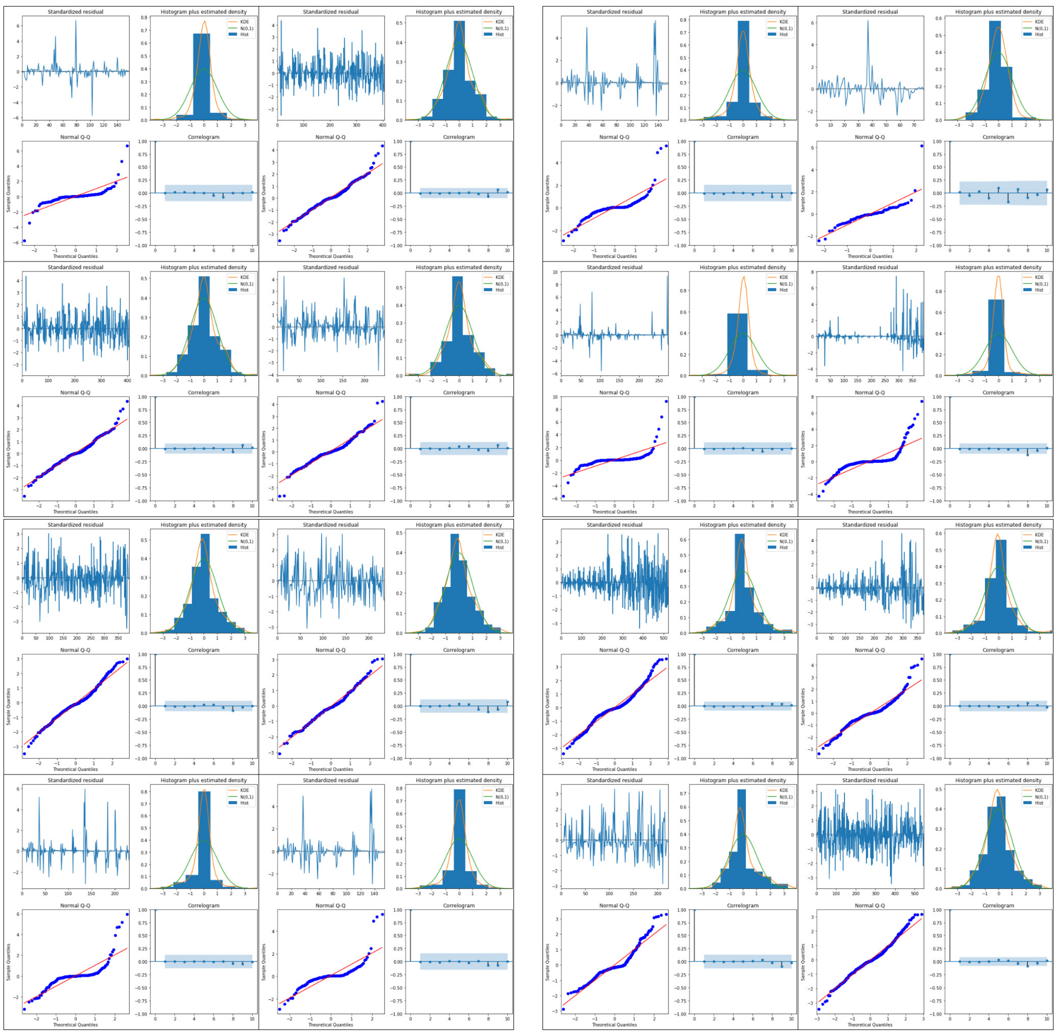


Figure B.5: 16 $ARIMAX(6,1,2)(2,1,0)_{24}$ model diagnostic plots (part 1/2)

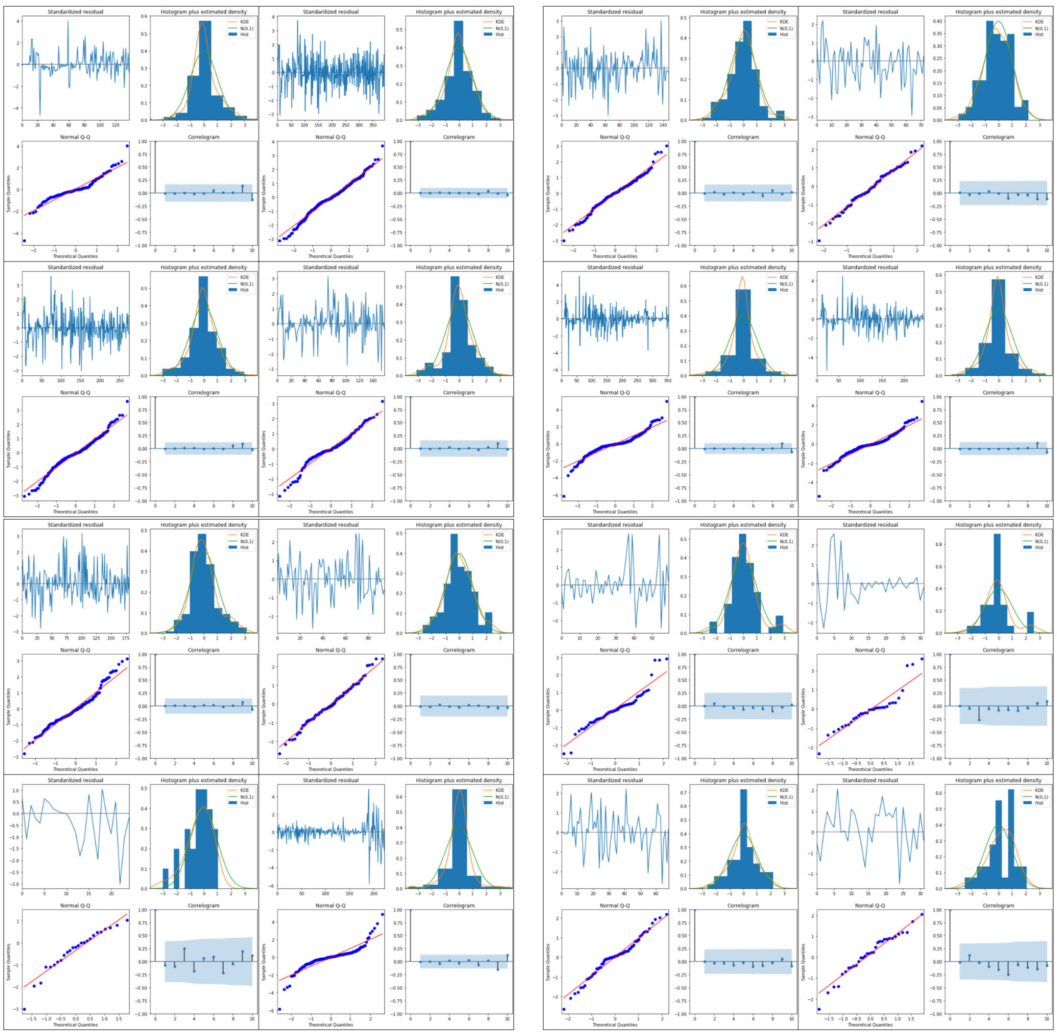


Figure B.6: 16 $ARIMAX(6,1,2)(2,1,0)_{24}$ model diagnostic plots (part 2/2)

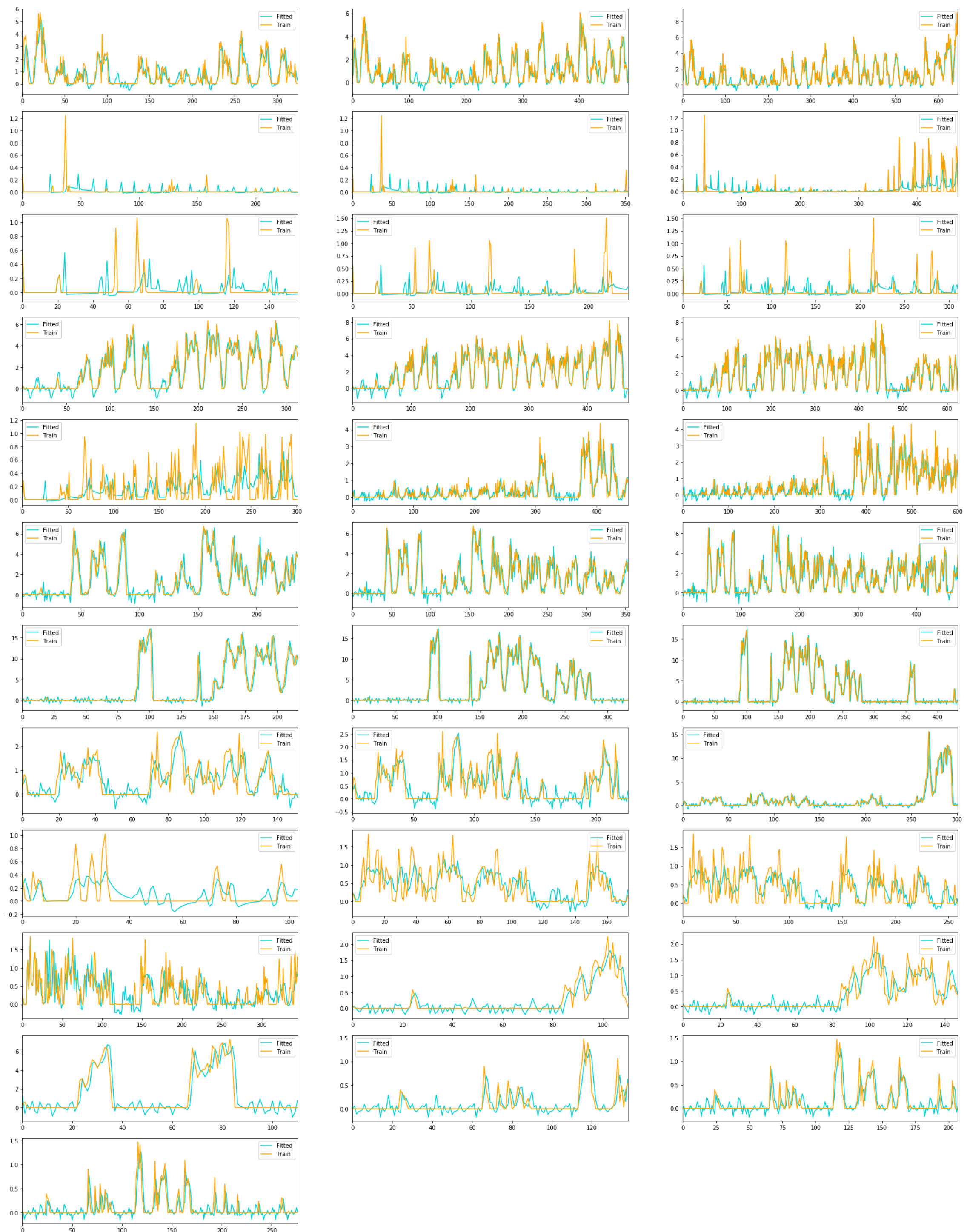


Figure B.7: Training values (orange) versus fitted values (blue) by Additive Holt-Winter's model considering all folds

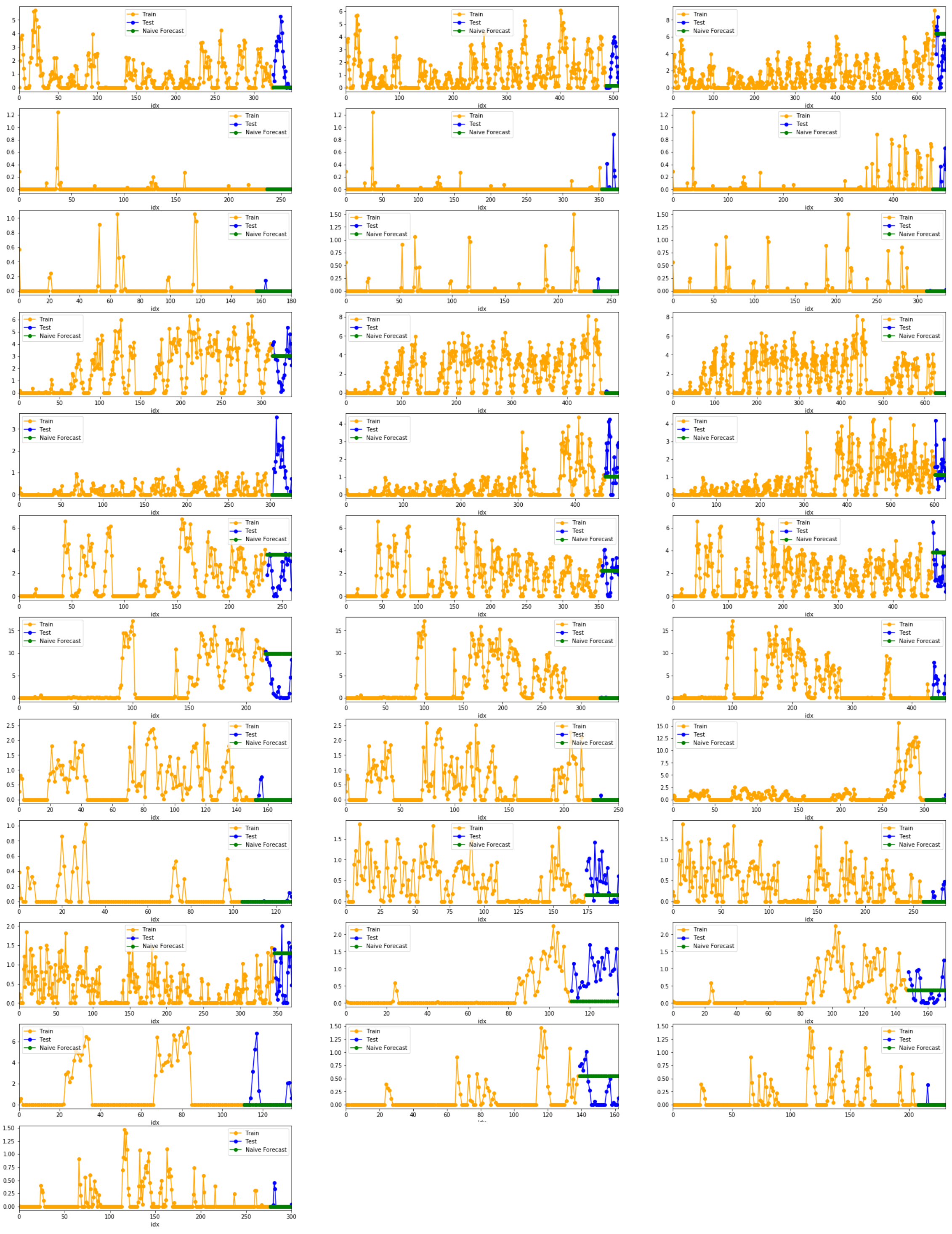


Figure B.8: Persistence model time series forecasting per each fold

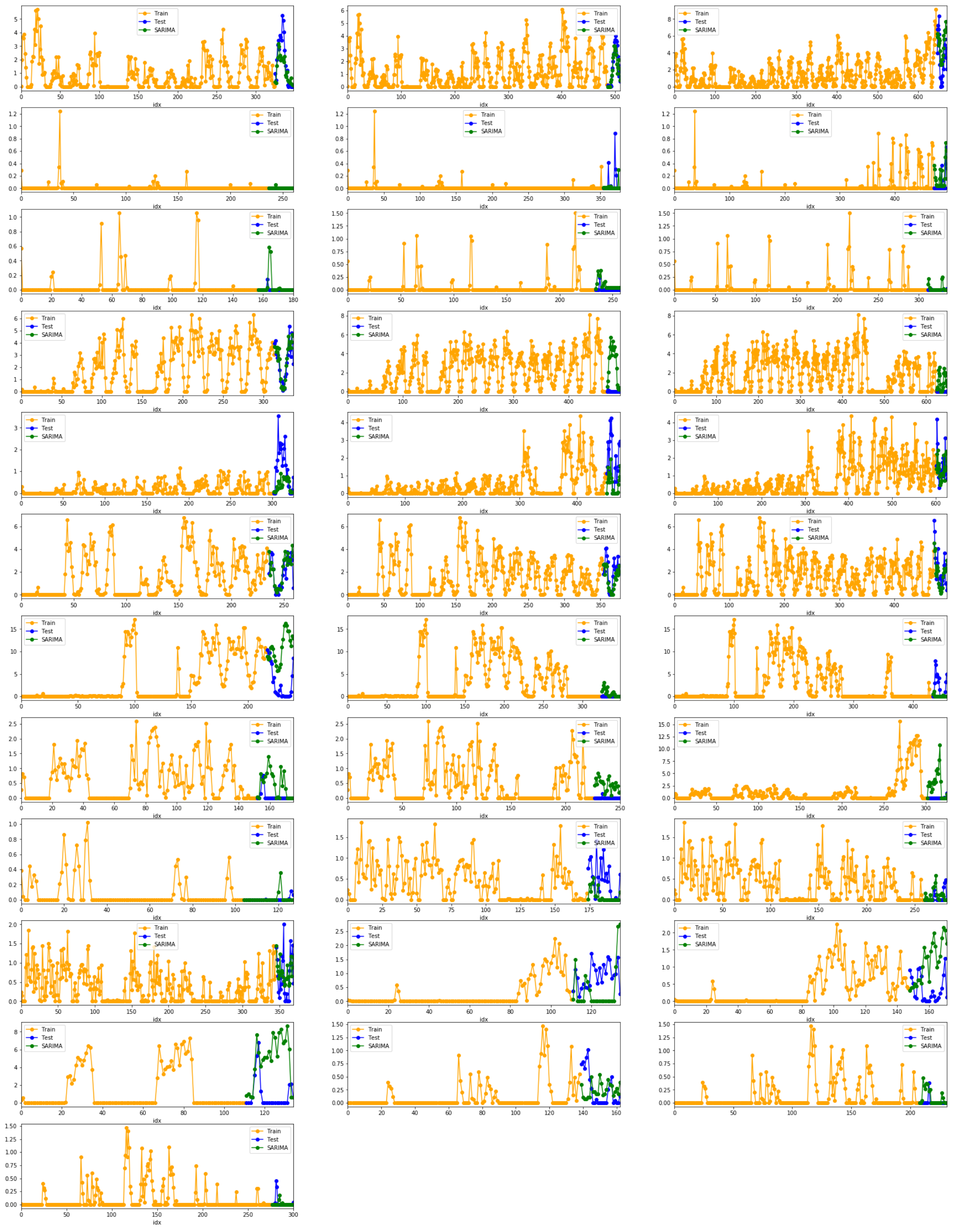


Figure B.9: $ARIMA(6,1,2)(2,1,0)_{24}$ model time series forecasting per each fold

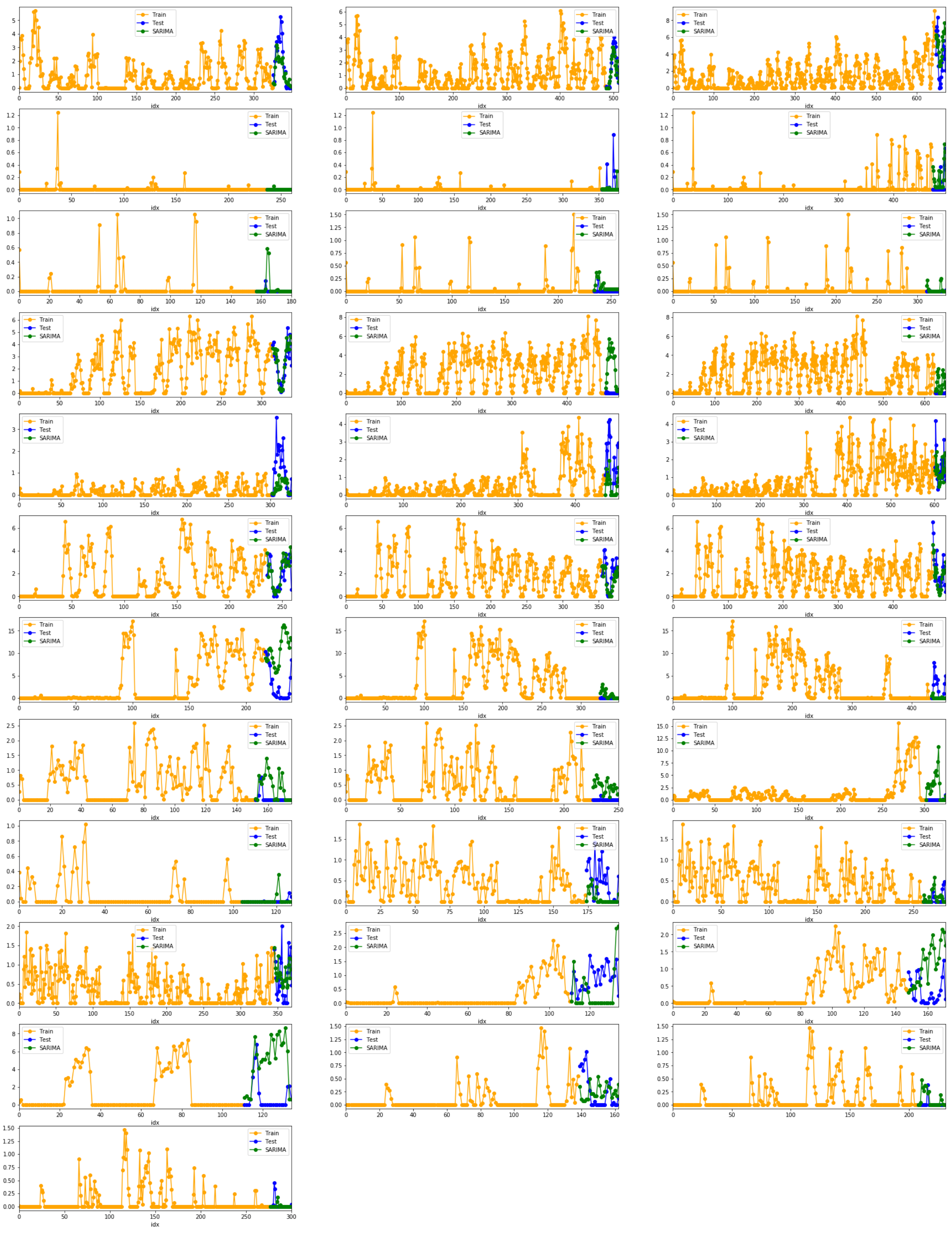


Figure B.10: $ARIMAX(6,1,2)(2,1,0)_{24}$ model time series using 1 day lagged number of active contributors forecasting per each fold

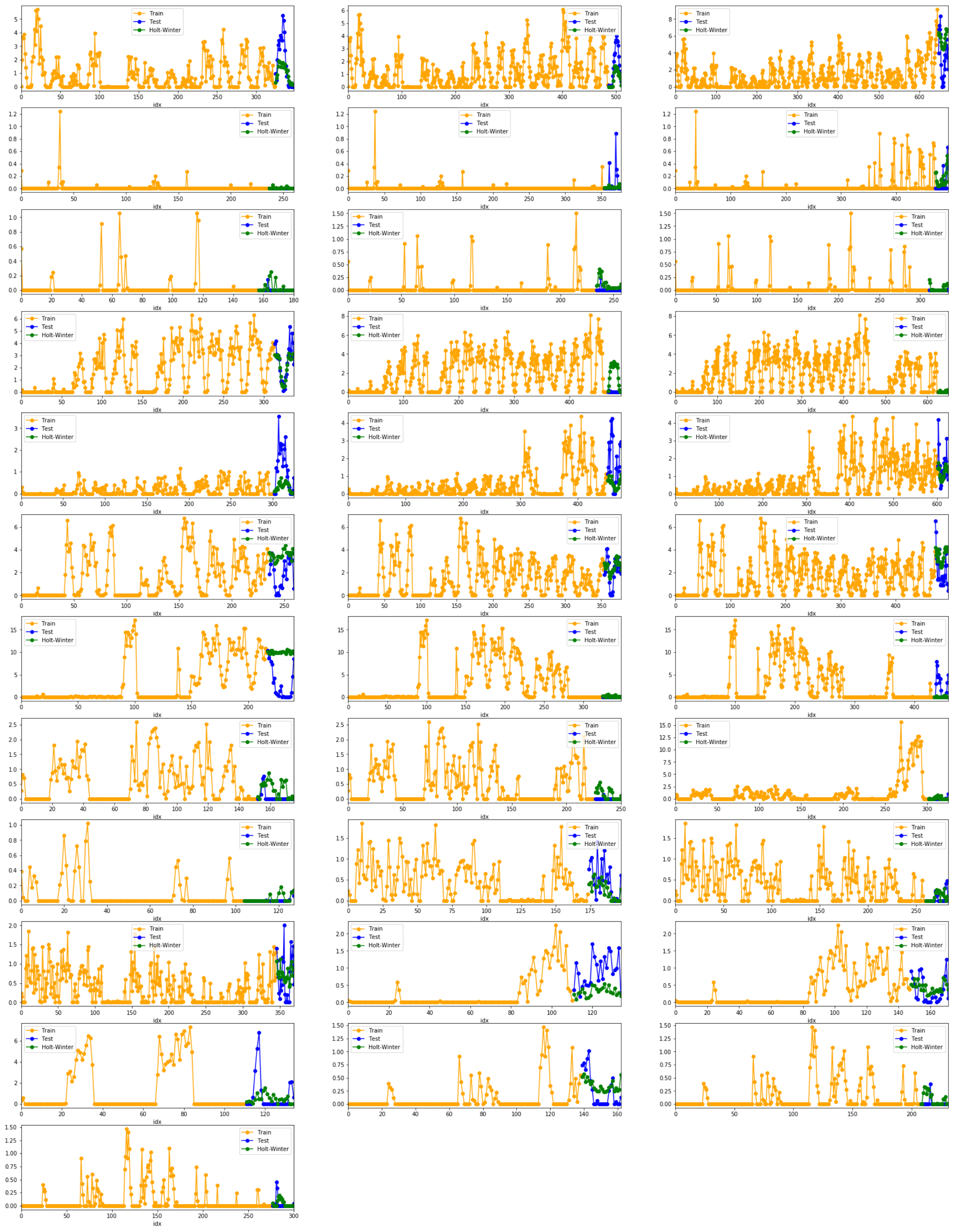


Figure B.11: Additive Holt-Winter's time series forecasting per each fold