City, University of
London MSc in Data Science

Project Report

2021

AIOps4B: Creating an AIOps Framework for Business Decision-Making

Julian Gonzalez Galvez

Supervised by: Ernesto Jimenez Ruiz

21 December 2021

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.

Signed: Julian Gonzalez Galvez

Abstract

AIOps4B is a framework for business decision-making that allows companies to forecast their KPIs, analyse their trend, find the root-causes for that trend, and provide deep analysis based on the metrics and external events. The Brazilian E-Commerce Public Dataset was used in the analysis. Several metrics were engineered from the different datasets and then selected using feature engineering. The Greykite Python library was then used to create the forecasting model and detect trend changepoints. The root-cause analysis was based on the impact of the metrics and external events on the trend. AIOps4B was then deployed from a local machine to the Microsoft Azure cloud platform.

Keywords: AIOps, forecast, trend, root-cause, cloud.

Table of Contents

1. Introduction and Objectives

1.1. Introduction and Research Question

This report is the result of the work I did while working on an internship at everis UK (now NTT DATA UK). NTT DATA is a consulting firm with a strong focus on Data Science and Machine Learning. The Data and Analytics area of the company has a subsection dedicated to developing Data Science and Data Engineering projects like this one. These projects are created by students with the help of the company, and after their internships are finished, the work on the projects can be resumed. The framework I produced is called AIOps4B. After my internship, I gave a presentation (NTT DATA Tech Talk) on the current implementation of AIOps4B.

The purpose of this project is to create a framework for business decision making. The main goal of the framework is for a business to control its revenue KPI. These projects are part of AIOps (Artificial Intelligence for IT Operations), which will be defined in Chapter 2 of the report. The framework should be able to take a time-series dataset as input, forecast the revenue, and analyse its trend and the root-causes for its changes, in order to avoid the decline of the revenue. The following research question is posed:

Can an AIOps framework for businesses to optimize their revenue be developed?

1.2. Objectives

The AIOps framework developed for this project was called AIOps4B (Artificial Intelligence for IT Operations for Business). The main goal of the AIOps4B framework will be to maintain a healthy revenue KPI by analysing historical data, understanding the trends, forecasting the future revenue trend, and if the forecast indicates a negative trend, understand the root-cause and provide deeper analysis of the root-causes. The generated product of the work will be a framework that optimizes a business' revenue KPI. We can divide the main goal into several objectives, which we will return to at the end of the project:

- **Analyse historical data**. This is the first step in the project. We need to find a suitable dataset for our endeavour, fully understand its context, and create appropriate metrics for the following analysis.
- **Understand revenue trends**. With the help of a Python library we will be able to visualise the revenue trends and fully comprehend them.

- **Forecast the future trend**. This is one of the main objectives of the project. Forecasting the revenue will allow the company that uses AIOps4B to foresee a decline in revenue.
- **Find root-causes**. Finding the appropriate root-causes to a decline in revenue will help the company to fix the issue. The root-causes can be a cause of the metrics or external events.
- **Provide deeper analysis**. Further explanation about the root-causes should be given in order to better understand the root-causes of a decline in revenue.

1.3. Beneficiaries

The beneficiaries of the work include:

- Businesses, as they use the revenue KPI as a measure to achieve their objectives. With AIOps4B, they will be able to foresee a decline in revenue and remedy it before it happens.
- Data scientists, who will be able to download the AIOps4B Python SDK and the APIs for free and use them for their trend analysis and forecast projects.

1.4. Work Plan

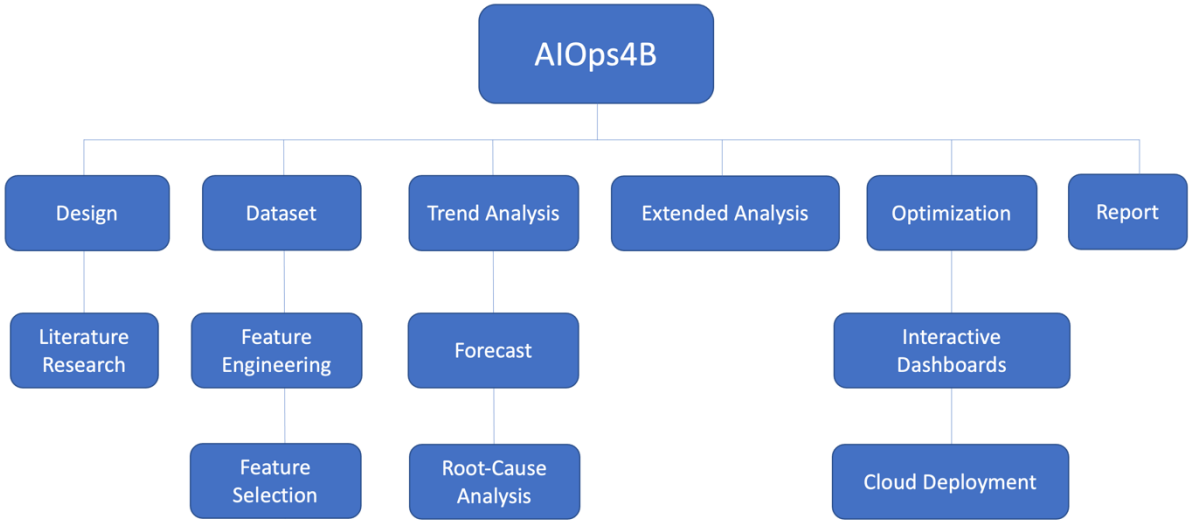

Figure 1 - Work plan

The pipeline of the project was divided into the steps that Figure 1 shows. First, a Design phase was set to organise the pipeline and start with the literature research. The next step was finding a suitable dataset, creating the metrics and applying feature selection. The Trend Analysis

phase is the core of the project, where the revenue trend was studied and forecasted, and the root-cause analysis strategy was planned and executed. The Extended Analysis was performed after that. The Optimization phase started following that, in order to fix some of the issues of the framework, while implementing the dashboard and deploying the platform to the cloud. The final step of the project was writing the report.

1.5. Report Structure and Deliverables

The report will be structured as follows. The main chapters of the report are:

- Chapter 1 introduces the work, states the research question and objectives, identifies the beneficiaries, and shows the work plan that was followed.
- Chapter 2 studies the critical context by providing an overview to the field of AIOps and to similar technologies to AIOps that are already available in the market.
- Chapter 3 explains the methods used in the development of the project. These methods include metrics creation and selection, how the trend was studied and analysed, how the forecast was made, and the cloud deployment.
- Chapter 4 presents the results of the trend analysis and provides an overview of the resulting dashboard.
- Chapter 5 returns to the objectives set at the start of the project and evaluates how they have been met. It also finds out to what extent has the research question been answered.
- Chapter 6 identifies the limitations of the project, proposes some future work for later implementations of the project, and presents some personal experiences with the project.
- The Glossary includes the definition of terms that are not explained in the report.

The appendices will be structured as follows:

- Appendix A is the Internship Based Project Proposal, which was submitted on 1 August 2021.
- Appendix B is the source code for the *data_pipeline.py* Python script, with which the *metrics_pvalue_ord.csv* metrics dataset was created.
- Appendix C is the source code for the *application.py* Python script, which contains most of the implementation of AIOps4B, including the trend analysis, forecasting, and root-cause analysis parts of the project.

- Appendix D is the source code for the *deep_analysis_geolocation.py* Python script, which contains the geolocation part of the Deep Analysis of AIOps4B.
- Appendix E contains the code for the *Dockerfile* which was used in the cloud deployment of AIOps4B.
- Appendix F is the *requirements.txt* file, which lists the dependencies used in the deployment of AIOps4B.

Other deliverables also included in the submission, along with the report:

- A video demo of AIOps4B.
- The slides of the NTT DATA Tech Talk.

These scripts and files will be included in the submission along with the report, and can also be found at my Github repository: https://github.com/julianglezg/aiops4b. The current implementation of AIOps4B can be seen in https://aiops4b.azurewebsites.net/.

2. Context

2.1. Overview

The current state of affairs and literature review will be provided in this section of the report. It will be divided into the four major areas that require a critical context analysis, which will inform the methods presented in the following chapter of the report.

The first part of the chapter tackles the concept of AIOps. The second part investigates some of the closest technologies to AIOps4B that are already available in the market. The third part provides an overview of the most important trend analysis and forecast libraries. The final part concerns cloud technologies, assessing their benefits and drawbacks.

2.2. AIOps

The area of Information Technology (IT) has started to permeate all industries, and modern companies now rely heavily on large and complex IT infrastructures. Their computer systems require an increasing efficiency, and their operators are more challenged. The need for a new solution has arisen. Levin et al. (2019) provide a paradigmatic case. Cloud-scale services must meet increasing availability, performance, and security demands on their cloud solutions. According to Levin et al. (2019), there was a significant disparity between the advanced, complex data-driven solutions they provided to their clients and the old-fashioned processes they used to run their own platform. This situation has prompted the investigation towards finding computing systems that automate these processes, while improving the efficiency of IT services.

This is how the concept of AIOps appeared. AIOps originally stood for Algorithmic IT Operations but was later changed to Artificial Intelligence for IT Operations. Lerner (2017) defines AIOps as platforms that study the use of Artificial Intelligence and other modern, data-driven technologies in the management and improvement of IT services. Some of the most important operations that AIOps platforms help businesses in is monitoring and automating IT processes. AIOps use Machine Learning and Big Data to monitor situations and provide recommendations to fix issues and failures that arise in the platform. They also allow for the concurrent use of several different technologies that are currently popular, such as different data sources, data collection processes, and analytical technologies focused on data.

Nowadays, AIOps solutions are used in a wide variety of applications. Some of their processes include collecting data and analysing it (Sigelman et al., 2010). This can be used in other cases for more complex tasks, such as failure prediction or anomaly detection. The latter, which is studied by Goldberg and Shan (2015), is especially relevant in this project, as it is one of the major steps of the pipeline.

2.3. Similar AIOps Technologies

AIOps solutions have been in development for years now. Some new products have been created recently for small operations, while other, already existing IT products have been overhauled as AIOps, such as Moogsoft (2021), a platform that performs anomaly detection using machine learning and advanced correlation; OpsRamp (2021), an IT operations management platform that uses AIOps for streamlined event resolution; or Zenoss (2021), a full-stack monitoring platform that applies AIOps solutions too. However, the two most important AIOps platforms that are already available are ThoughtSpot and Outlier, which we will analyse in detail.

2.3.1. ThoughtSpot

Thoughtspot is an AI-focused platform that allows non-expert users to perform data analysis on their projects. Their platform revolves around two major components: a consumer-grade search box and AI technology, which we will further study later. These two components help deliver self-service analytics that anyone can use (Thoughtspot, 2021). Nevertheless, there are many more important features that work towards the data analytics. We will provide a description of the most relevant of them following the nomenclature that appears in Thoughtspot's website.

- Search: Thoughtspot boasts a search engine, Search Data, that allows the user to write queries in natural language for analytics purposes, without the need to know how to write SQL language queries. Thoughtspot takes this fact as the claim that any user can perform data analytics with their platform. Figure 2 shows an example of a Search Data

query.

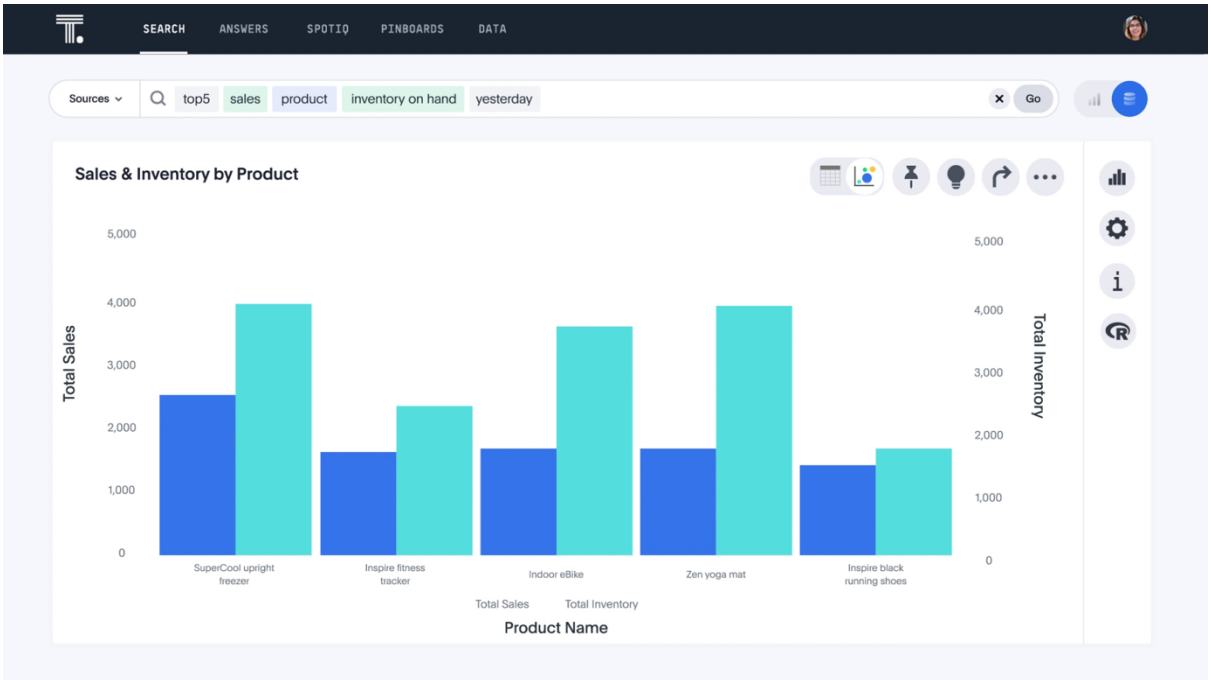

Figure 2 - Search Data query (Thoughtspot, 2021)

Search Answers is a feature that allows users to search for insights that has already been created by their peers, providing personalized, analytical content that is useful for analysis. These answers can be explained in detail by using Answer Explainer. The most interesting feature in this area is DataRank, Thoughtspot's machine learning algorithm that showcases data recommendations as you type, according to Thoughtspot (2021), which guides the user to the correct data based on its properties. Unfortunately, Thoughtspot does not provide further information about the algorithm, which would be the ideal situation in order to compare it to the one used in this project. Overall, the search features in Thoughtspot seem useful for the general public and non-expert users. However, since our platform will be aimed at data-driven companies and individual data scientists, these features would be unnecessary.

- Auto Analyze: The analysis section of an AI platform is the most important part of it. Thoughtspot uses SpotIQ, their AI engine. SpotIQ uses AI to run algorithms on the data and provides insights that might be significant to the user. According to Thoughtspot, their algorithm can discover anomalies, identify leading indicators, isolate trends, and segment data. This is relevant to this project, as AIOps4B intends to identify anomalies and trends as well. SpotIQ is connected to the search features, and uses machine learning to learn from the users' searches in order to provide better insights. SpotIQ is the most relevant part of Thoughtspot in terms of this project, as our objectives are

similar to the ones SpotIQ accomplishes. However, once again, Thoughtspot does not provide an in-depth exploration of their algorithm.

- Visualize: Thoughtspot allows the users to write queries on the search bar that create charts automatically. Figure 3 shows an example of a query-generated graph. The visualisations can be customized and arranged into a dashboard that can be prepared for a presentation.

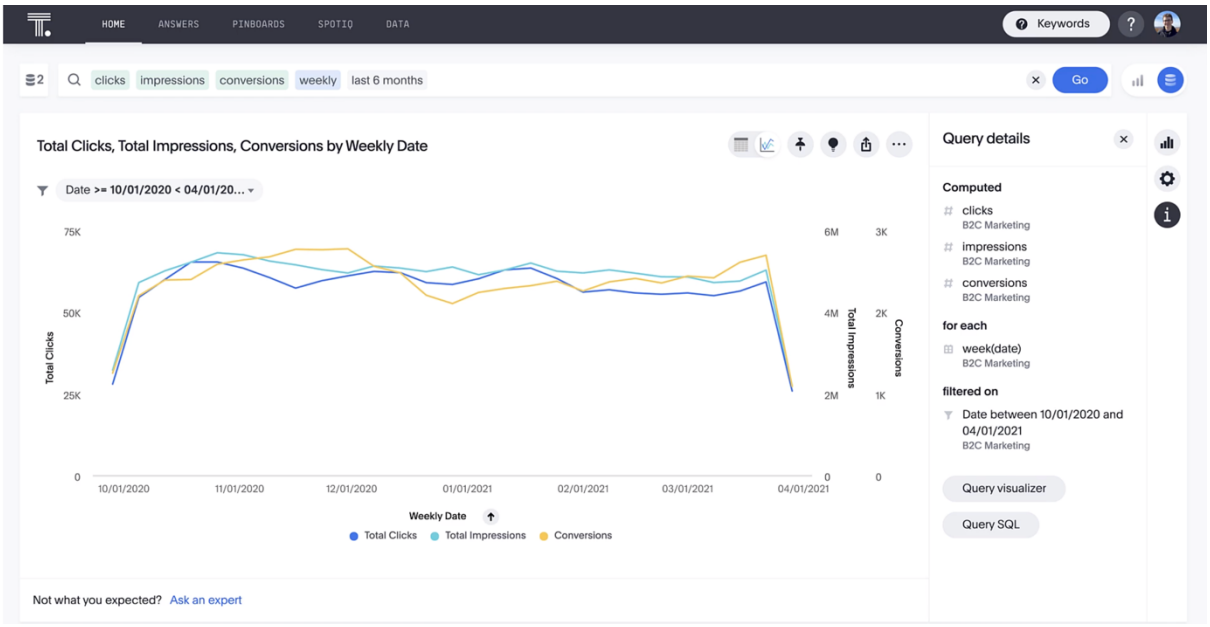

Figure 3 - Thoughspot visualisation

Thoughtspot offers other features such as cloud integration, data warehouse connectivity, and mobile capabilities. Overall, it is a powerful tool that, however, comes at a high price. This is one of the main advantages of AIOps4B; our platform will be open source.

2.3.2. Outlier

Outlier is a similar AIOps service that automatically analyses data features that the user inputs to the system, and delivers insights to them (Outlier, 2021). Their most important feature is that Outlier's systems perform data analysis, choose the most important insights using their recommender systems, and send them via email to the user every morning. Outlier also has a root-cause feature that identifies the root causes of the changes in business metrics, and displays them in a dashboard (Petersen, 2018). Nonetheless, the issue with Outlier is the same as ThoughtSpot: while they flag the problem in the future trend and identify the root-cause, they do not recommend a specific solution to the problem.

2.4. Trend Analysis and Forecast

The core of the project is the trend analysis and forecasting section. The analysis and forecast of a time series have been studied in detail in the past decades, with several models being developed, such as ARIMA (Hyndman, 2014) or other machine learning models (Ahmed et al., 2010). Since the pipeline for this project is implemented in Python, time series analysis libraries for Python are required. In the two following subsections, two of the most important libraries will be introduced: Prophet and Greykite.

2.4.1. Prophet

The most popular Python library for this trend analysis and forecasting for the past years has been Prophet. Prophet, developed by Facebook, is used for time-series forecasting and analysis. According to Žunić et al. (2020), Prophet fits non-linear trends with yearly, weekly, and daily seasonality, and it includes holiday effects. Prophet works well with outliers and, following Taylor and Letham (2018), it is used across Facebook. However, Prophet is not able to incorporate new, special, external events to the forecast, which is a relevant aspect of a project like this.

2.4.2. Greykite

Recently, LinkedIn developed a new time series forecasting and analysis Python library in response to Facebook's Prophet: Greykite. Greykite's main advantage over Prophet is that it offers the possibility of including special events or anomalies on the forecasting model (Hosseini et al., 2021). Its algorithm for forecasting is called Silverkite. Figure 4 shows its architecture.

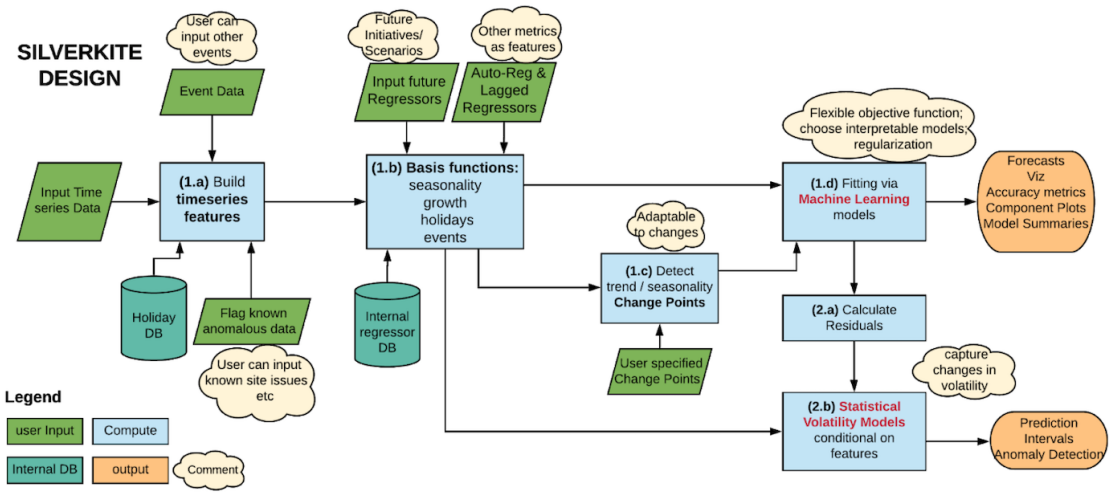

Figure 4 - Silverkite architecture (LinkedIn Engineering, 2021)

In Figure 4, the green parallelograms represent the user inputs, the blue rectangles indicate the computational steps that Silverkite performs, the blue prisms denote internal databases that Silverkite uses, and the orange ovals are the outputs. The computational steps of the algorithm are divided into two phases: Phase 1, which represents the conditional mean model, and Phase 2, which represents the error model (Hosseini, 2021).

Phase 1 starts with the user inputting their time-series data, along with other known events and flagging anomalies. In the second step of Phase 1, Silverkite transforms the features to appropriate basis functions. The most important step in Phase 1 is the third one, in which Silverkite applies a changepoint detection algorithm to discover changes in the trend and seasonality over time. The fourth, and possibly final step, is applying an appropriate machine learning algorithm to fit the features, The outputs of this process are the forecasts, visualisations, and performance metrics and model summaries. There is also Phase 2, in which Silverkite calculates residuals and applies a model that can be fitted to them in order to find predictions or perform anomaly detection.

While Prophet is still the most popular time-series analysis Python library, Greykite offers a new alternative that boasts some features that Prophet cannot offer. Greykite also allows us to work with the most modern and advanced trend analysis Python library.

2.5. Cloud Technologies

While developing the platform, which was meant to be a local application, the possibility arose to migrate the framework to the cloud. Apart from the personal experience of having the opportunity to work with cloud technologies in a professional project for the first time, this would allow AIOps4B to be hosted online and enabling anyone to access the final application. We will provide the definition of cloud computing technologies, some of their benefits and drawbacks, and a brief overview of the most popular technologies.

2.5.1. Definition

Google Cloud (2021 - a) defines cloud computing as “the on-demand availability of computing resources as services over the internet”. Cloud computing eliminates the need of having to procure and manage the resources needed for a project, allowing for the use of servers, storage, and other resources online instead of on premise. Another important feature of cloud computing is that users pay only for what they use.

Companies are increasingly adopting cloud computing for their projects. According to Flexera's 2021 State of the Cloud Report (Luxner, 2021), out of the 750 IT professionals surveyed, 92 percent of their enterprises have a multi-cloud strategy. Moreover, enterprise cloud spend is growing, accelerated by the COVID-19 pandemic, and most believe that this growth will be increased over time. Gartner (2021) forecasted that public cloud end-user spending would grow 23 percent in 2021.

There are three types of cloud services:

- Infrastructure as a Service (IaaS): IaaS provides resources that the user can take advantage of online, instead of on premise, and pay only for what they use. According to Gartner (2021), IaaS would see the highest growth of 2021 out of the other cloud services, of 38.5 percent.
- Platform as a Service (PaaS): PaaS offers a cloud platform for developing and running applications online, without the need of on-premise facilities. Developers operate on a monthly fee, making use of on demand resources supplied by the provider.
- Software as a Service (SaaS): SaaS offers software that is hosted and runs in the cloud. The user connect to it via the internet, and can access their data and the application for a monthly fee. SaaS is the most common form of cloud computing, and will continue to be in 2021 and 2022, according to Gartner (2021).

2.5.2. Benefits

The benefits of cloud computing when compared to traditional, local computing are:

- Cost. The model of paying only for what is used allows users to reduce their costs significantly. Companies can forget the expense on creating and maintaining a datacentre and other infrastructure and can focus on allocating their IT resources into more productive work.
- Efficiency. Developers can create and deploy servers and applications much faster on cloud, thus increasing the company's efficiency.
- Scalability. Cloud providers allow users to scale their services, up or down, as required. Moreover, not only can companies scale their computing services, but they can also scale geographically, as providers such as Google Cloud Platform (GCP) offer data

servers around different locations of the world, in order to increase performance globally.

- Security. Cloud security is generally considered higher than data centre security, as shown by a report from McAfee, where 52 percent of organizations state that they experience better security in the cloud than on-premise IT environments (Businesswire, 2019). The major cloud providers usually have a system for controlling the access of other users to cloud resources. For instance, GCP's Identity and Access Management (IAM) lets administrators manage who can access the different resources from a centralized platform (Google Cloud, 2021 - b). This is especially relevant for enterprises with several projects on the cloud and numerous users.

2.5.3. Drawbacks

However, cloud technologies are still not a completely reliable replacement for on-premise cloud computing. Some of the main drawbacks of cloud computing include:

- Downtime: Cloud computing services can have an outage or a slowdown and stop cloud operations that are currently running, and this can have major consequences for companies. For instance, in 2017, an outage in Amazon Web Services (AWS) had an impact of $150 million in the S&P 500 index companies (Hersher, 2017). There are ways to minimize these problems, such as setting multi-region deployments, an option available in the major cloud computing solutions, that have automated failover, so the data is replicated in several regions.
- Security: Even though cloud security is held in higher consideration than data centre security, external cloud providers always have security risks. However, using the security tools that the major cloud technologies offer, such as the aforementioned IAM service and proper authentication for the users, those risks can be avoided.

3. Methods

3.1. Overview

This section of the report presents the methods that were used in this project. An overview of the architecture of the AIOps4B framework will be provided, explaining some of the crucial steps of the pipeline: feature engineering and selection, trend analysis and forecast, root-cause analysis, and cloud deployment.

3.2. Architecture

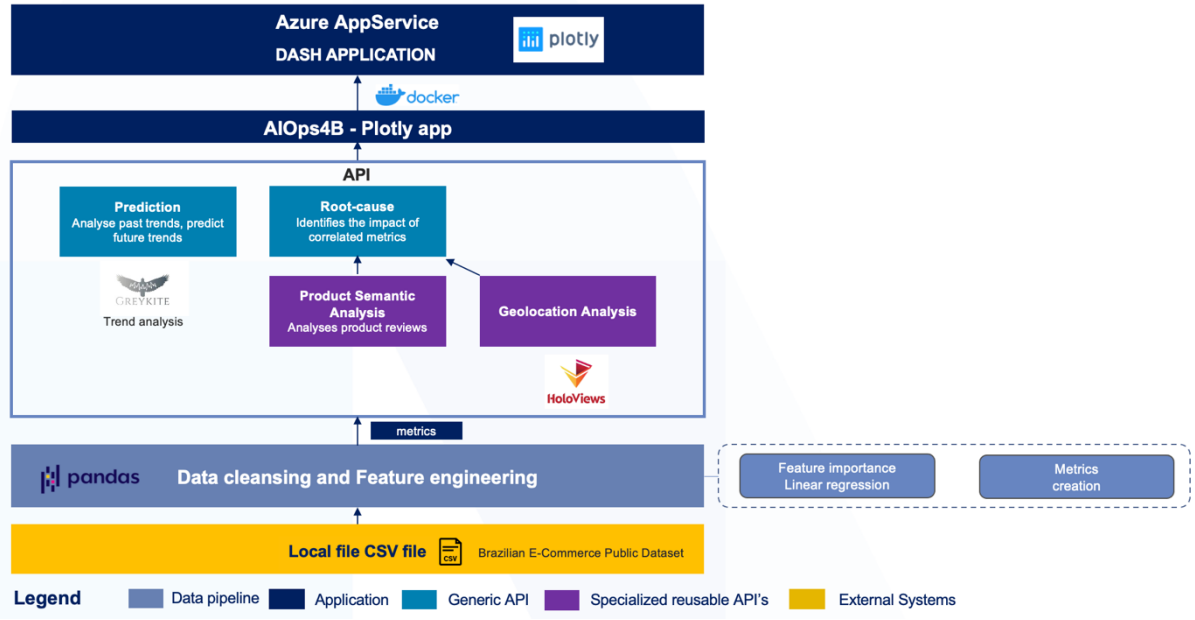

Figure 5 - AIOps4B High-level architecture

Figure 5 shows the high-level architecture of the AIOps4B platform. This graph was designed for the presentation I gave on AIOps4B to NTT DATA UK, thus the more commercial-oriented structure. However, the code follows the same pipeline. A rundown of the major steps will now be provided, and the specific details of them will be presented in the subsequent sections of the report.

The project starts with a dataset, in this case the Brazilian E-Commerce Public Dataset. It will be further analysed in Chapter 3.3 of the report. The next step was data cleansing and feature engineering. From the multiple CSV files that the Brazilian E-Commerce Public Dataset comprises, several metrics were created. In order to facilitate the subsequent steps of the pipeline, feature importance was applied to the metrics. The approach that was used will be explained in Chapter 3.4. The final set of metrics were brought together in a new dataset which will be informally called the "metrics" dataset (*metrics_pvalue_ord.csv* file).

The core of the project is what is represented in Figure 5 as the APIs, i.e., the trend analysis, forecast, and root-cause analysis. The blue rectangles represent the two generic APIs that were implemented: the Prediction API, which plots the time series and performs trend analysis, and the Root-Cause API, which performs root-cause analysis on the metrics. We first input the metrics dataset into the Greykite Python library in order to perform trend analysis and the forecast. The root-cause analysis was divided into the impact of the metrics on the revenue and the impact of external events on the revenue. This process will be further explained in Chapter 3.6. The Root-Cause API also applies what we defined as Deep Analysis, which is a more concrete explanation of the root-cause for the decline in revenue. The purple rectangles in Figure 5 represent the two main areas of Deep Analysis: product semantic analysis, or natural language processing, was applied to the order reviews with the objective of finding out the problems that customers had, and geolocation analysis, which was used to discover the regions where there was more delay in the orders and higher delivery costs.

All this information, graphs, and analysis was brough together in a dashboard, implemented locally with Dash, a Python library that is a part of the Plotly library. The final step of the project was the cloud deployment. By creating a Docker image of the Dash app, and uploading it to the Docker Hub, the dashboard was easily deployed to the Microsoft Azure cloud platform as a web application. We will further elaborate on this process in Chapter 3.7.

3.3. Dataset

The first step in the pipeline is acquiring the dataset. The initial approach was choosing a dataset for an e-commerce use case. Since the main objective of the AIOps4B framework is to keep the revenue KPI healthy, an e-commerce use case would be ideal for the project. Another requirement was decided when searching for a dataset, which was that the dataset should have some features that could be optimized or that would allow for interesting analysis. The dataset that was chosen for the final implementation was the Brazilian E-Commerce Public Dataset, which can be downloaded in Kaggle. This is a public dataset of orders that the Olist Store, a Brazilian website that allows companies to create their own marketplaces, received. According to Kaggle (2018), Olist connects small, Brazilian businesses to channels. The merchants are then able to sell their products through the Olist Store and ship them directly to the customers with the help of Olist logistics. The dataset contains information about 99441 unique orders, placed between 2016 and 2018. The Brazilian E-Commerce dataset, in fact, is really a database,

with the data for the orders divided into multiple datasets, which are organized as the following data schema shows:

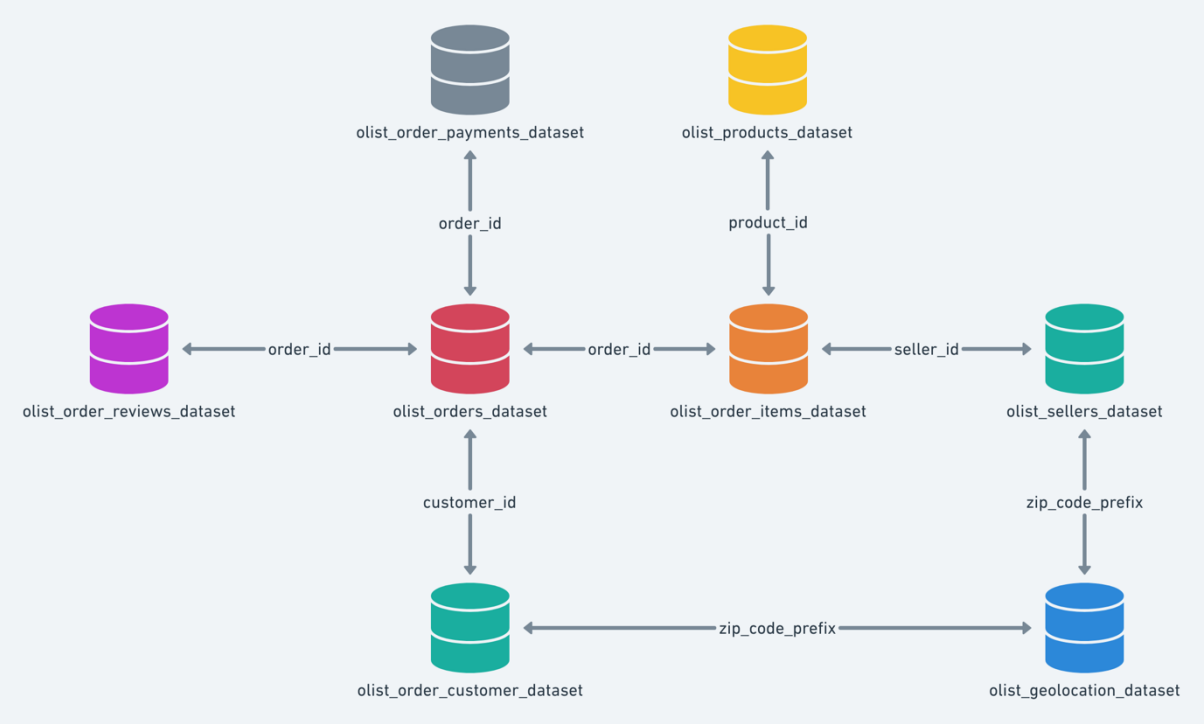

Figure 6 - Brazilian E-Commerce Dataset data schema (Kaggle, 2018)

The multiple datasets contain information about the customers and their location, Brazilian zip-codes and coordinates, items, payment options, reviews, products, and sellers. For the purpose of this project, the following CSV files were selected and imported into the Python script:

- *olist_orders_dataset.csv*
- *olist_order_payments_dataset.csv*
- *olist_order_reviews_dataset.csv*
- *olist_order_items_dataset.csv*
- *olist_order_customer_dataset.csv*
- *olist_geolocation_dataset.csv*

3.4. Feature Engineering and Selection

The next step in the pipeline is a typical one in Data Science projects: data cleansing and feature engineering. Since the Brazilian E-Commerce Dataset offers many variables and features to work with, a first selection of them was made. This selection includes some of the features that are basic in a task like this, for instance, daily orders and revenue, as well as other features which allow for interesting analysis, one of our objectives when choosing a dataset, such as review scores or order delay. The first set of metrics includes the following features:

- Date: each element of this metrics corresponds to a day between September 4, 2016, and October 17, 2018.
- Revenue: the revenue for each day.
- Total New Customers: the total new customers for each day.
- Total Returning Customers: the total returning customers for each day.
- Average Review Score: the average review score for each day.
- Total Good Reviews: the total number of reviews with a score of 3 (out of 5) or higher for each day.
- Total Bad Reviews: the total number of reviews with a score of 2 or lower for each day.
- Good/Bad Review Ratio: the ratio of good and bad reviews for each day.
- Average Freight Value: the average delivery cost for all the orders for each day.
- Average Delivery Score: the average score (each order gets a score of 1 if it is on time or early and -1 if it is late).

Since the number of metrics is significant, feature importance was applied to reduce the number of them and to simplify the task. Feature selection, or feature importance, is the process of selecting the relevant metrics and discarding the redundant ones. Following the definition provided by Yu and Liu (2003), a feature is good if it is important to the class, but it is not redundant to the other relevant features. The approach chosen in this project is to use correlation between variables as the metric for feature selection. Therefore, a feature will be relevant if it has a high correlation with the class but not with the other features. The feature selection task then comprises finding a suitable measure of correlations between features and a procedure to select features based on this measure, according to Yu and Liu (2003).

In order to measure the correlation between features, the chosen measure was the linear correlation. We performed linear regression with the OLS (Ordinary Least Squares) function from the *statsmodels* Python library. The results of the OLS can be seen in Figure 7.

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Total_New_Customers          -16.7185     25.695     -0.651      0.516     -67.182      33.745
Total_Returning_Customers     26.6682     65.456      0.407      0.684    -101.885     155.221
AVG_Review_Score           -1730.9939    484.948     -3.569      0.000   -2683.419    -778.569
Total_Good_Reviews           185.1437     26.662      6.944      0.000     132.781     237.507
Total_Bad_Reviews             97.4146     28.781      3.385      0.001      40.890     153.939
Ratio                        -31.1083   4074.412     -0.008      0.994   -8033.142    7970.925
AVG_Freight_Value            455.3396     52.045      8.749      0.000     353.124     557.555
AVG_Delivery_Score         -1433.0515   1733.237     -0.827      0.409   -4837.082    1970.979
==============================================================================
Omnibus:                      109.279   Durbin-Watson:                   1.614
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              263.316
Skew:                           0.945   Prob(JB):                     6.63e-58
Kurtosis:                       5.634   Cond. No.                     7.69e+03
==============================================================================
```

Figure 7 - OLS results

In Figure 7, the P>|t| column indicates the p-value of the features in this hypothesis test. If we choose an alpha value of 0.5, the only significant metrics are the average review score, the total good reviews, the total bad reviews, the average freight value, and the average delivery score. The final selected metrics are:

- Total Good Reviews
- Total Bad Reviews
- Average Review Score.
- Average Freight Value.
- Average Delivery Score.

These metrics, which we based the root-cause analysis on, were brought into a new Python dataframe, along with the date, the revenue, and the total daily orders, and we defined them as the metrics dataset. The metrics dataset has the following structure:

Date	Revenue	AVG Review Score	AVG Freight Value	AVG Delivery Score	Total Good Reviews	Total Bad Reviews	Orders
…	…	…	…	…	…	…	…

Table 1 - Metrics dataset format

Even though the total good and bad reviews are included in the metrics dataset, they will not be used in the root-cause analysis, as they are redundant when using the average review score metric.

3.5. Trend Analysis and Forecast

The next major section in the architecture of AIOps4B is the core of the project: trend analysis, forecast, and root-cause analysis. In this chapter the procedure for the trend analysis and forecast will be provided. To sum up the process until this point: an online store receives orders every day. The objective is to analyse the trend that the revenue follows. The revenue trend can be obtained with the use of the Greykite library, which has been described in Chapter 2. In addition to the revenue trend, the Greykite library detects its changepoints, i.e., the points in time where the revenue trend changes. The main focus will be to understand those changepoints by performing root-cause analysis; however, the trend will be analysed first.

In order to study the trend, a Greykite forecasting model must be created. To create a forecasting model, several parameters need to be defined. The first decision is to create a Prophet or Silverkite model; we chose to create a Silverkite model. According to the Greykite Library documentation (2021 - a), Silverkite decomposes time series into components and creates time-based features, autoregressive features. The user also provides macro-economic features, such as external events, and Silverkite then performs a machine learning regression model to find out the relationship between the time series that the user has provided and these features. Defining those features is then a vital step in the process of obtaining a correct forecast.

There are some initial parameters that need to be set, such as the name of the time column and the value column or the frequency for the rows, which is daily in this case. However, a forecasting model needs other additional parameters, which can come from 5 different features: anomalies, growth, changepoints, seasonality, and events. The defined features in our model were the following:

- **Anomalies**. An anomaly is defined by the Greykite Library documentation (2021 - a) as a deviation in the metric that is not expected to occur in the future. In our case, the day 24-11-2017 has been set as an anomaly. That day marks Black Friday, the event in which most stores around the world have a special sale. We do not know the specific

sales that the Olist store had that day, but just by looking at the graph that will be presented in Section 4 we can see that Black Friday is an outlier in the yearly sales. If the Brazilian E-Commerce Public Dataset contained data from more years, we could have probably observed the effect of Black Friday on repeated occasions; given that we have limited data, we can only see the effect once. That is the reason why Black Friday was set as an anomaly. The anomaly start was set at 23-11-2017 and the end at 26-11-2017, since the effect of Black Friday on the sales lasts for several days before and after its specific date.

- **Growth**. Growth is used to model the trend of the data over time. The *growth_term* has been set to linear, which is the default value.
- **Changepoints**. Automatic changepoint detection has been set for this model, even if we will not analyse them at this moment. There are several parameters in this feature that need to be explained. First, the algorithm does a mean aggregation to eliminate small fluctuations, according to the Greykite Library documentation (2021 - b). This effect is regulated by the *resample_freq* parameter, which has been set to 7 days. Expecting some yearly seasonality effect, although there is not enough data to detect seasonality, the *yearly_seasonality_order* parameter has been set to 2, for the 2 years of data that we have. Another important parameter is *potential_changepoint_n*, which indicates the maximum number of changepoints wanted. This parameter was set to 25, since it is not ideal to have more than that number. Complementing this parameter, the *no_changepoint_proportion_from_end* parameter indicates the proportion of the data from the end in which there will be no detected changepoints. This was set to 0.1, which corresponds to the 10% of the data. However, the most important parameter is *regularization_strength*, which regulates the amount of changepoints that will be detected. It ranges from 0.0 to 1.0, where greater values mean less changepoints. This parameter was set to 0.6.
- **Seasonality**. Seasonality describes the periodical patter of the time series, according once again to the Greykite Library documentation (2021 - a). It is defined through Fourier series with different orders, and the greater the order the more detailed pattern the model can learn. The yearly seasonality was set to 2 and weekly seasonality was set to 4.

- **Events**. Events are holidays and other occurrences that can cause an effect on the time series. All the holidays in Brazil were used as events in the model, as Silverkite provides a database with the holidays of every country.

With those parameters, the complete model was set up. A forecaster is then configured and run. A forecast horizon of 5 days gives us the forecast of the revenue for the following 5 days, which can be seen in Chapter 4.2.2.

The model defined above provides the forecast for the revenue KPI, which is one of the objectives of AIOps4B and this project, but it is not the only one. In order to understand the trend, the trend changepoints need to be detected and analysed. The changepoint detection is performed with a new Silverkite model, less complete than the one above. In this case, the only feature included is the changepoints. The only parameters that are changed from above are the *yearly_seasonality_order*, which is set to 15, and the *no_changepoint_proportion_from_end* parameter, which is set to 0.05. There is an additional custom parameter in this model, which is a ridge fit algorithm. This algorithm prevents overfitting when there are many changepoints, which could be useful for our model. A forecaster is then configured with these parameters and a forecast horizon of 5 days. The resulting graph with the trend and the detected changepoints can be seen in Chapter 4.2.2.

3.6. Root-Cause Analysis

The final step of the pipeline is to discover the root-cause of every changepoint. In other words, the objective is to understand why the trend behaves the way it does, which will help understand how it will behave in the future. Our approach for the root-cause analysis will be focused on finding out how the metrics we are working with (average review score, average freight value, and average delivery score, since the total good and bad reviews are redundant) explain the changepoints. There will be two main ways of studying this. The first one is by studying the impact of the metrics and the impact of the events on the trend. The second one is by performing what was called Deep Analysis, which uses different Machine Learning techniques on the metrics to find out a deeper root-cause. The impact of the metrics and the events was be calculated for all changepoints, while deep analysis was only performed for the changepoints in which the revenue decreased, since in a commercial situation these would be the points in time where the trend would need to be fixed.

3.6.1. Impact of Metrics and Events

The approach to calculate the impact of the metrics and events was to compute the change of revenue and orders associated to each changepoint. The average revenue before and after every changepoint was calculated, and its percentage change is displayed in the dashboard. For the percentage change of the revenue and orders, instead of computing individual days before and after each changepoint, we take the period between one changepoint and the previous one, and the period between one changepoint and the following one. Since we are not considering the first changepoint in the analysis, the "previous" period for Changepoint 2 starts at 23-1-2017. The result of this calculation is the change of revenue and orders for each changepoint and will be presented in Chapter 4.

The impact on the revenue is then considered as the sum of the impact of the metrics and the impact of the events. The impact of the metrics is calculated using the following procedure. First, a linear regression was performed on the three metrics, which gives us three illustrative coefficients. Multiplying those coefficients by the change of the respective metric will give us the impact of that metric in a certain changepoint, and the sum of all of those will give us the impact of the metrics on the trend. The results will be presented in Chapter 4. The impact of the metrics on the trend also gives the impact of external events on the trend. If the event is known, AIOps4B flags it as such. If not, AIOps4B flags the decrease as unknown. Some examples of this will be provided in Chapter 4.

3.6.2. Deep Analysis

As mentioned before, we have defined Deep Analysis as an advanced root-cause analysis based on the distinctive features of the metrics that were chosen in the feature selection: average review score, average freight value, and average delivery score. It is important to mention that the Deep Analysis feature is only applied to the changepoints where the trend is negative.

The deep analysis of the average review score was inspired by Thiago Panini's (Panini, 2021) work on Kaggle. We used Natural Language Processing techniques with the *NLTK* and *scikit-learn* Python libraries. The aim was to simplify the review comments in order to analyse them more easily. Regular expressions were used, stopwords were removed and stemming was applied to the reviews. The end products were simplified versions of the reviews. The next step was to extract the top unigrams, bigrams, and trigrams by sentiment (for negative and positive reviews). The top negative trigrams were used to analyse the reasons for poor review scores.

The most popular trigrams were translated from Portuguese to English and are showcased in the Deep Analysis section of the dashboard. The results will be shown in Chapter 4.

The deep analysis of the freight value and delivery score metrics consists of geolocation analysis of those two metrics. The *Plotly* library was used to deploy two maps of Brazil: one shows the freight value of every order and the other shows the delivery delay. The maps are not included in AIOps4b, but their conclusions are. They will be shown in Chapter 4 of the report.

3.7. Cloud Deployment

AIOps4B was first implemented locally, following the steps of the pipeline that have been described in the previous sections. The next step would be to deploy it to the cloud. There are several reasons that indicated that cloud deployment was the next step. One of the ideas for the creation of AIOps4B was that it would be an open-source application that any user could download and include it in their projects. Moreover, everis UK had easy access to cloud technologies, Microsoft Azure specifically. This allowed AIOps4B to be deployed to the Azure cloud platform, thus incorporating to the project the new and popular technologies that have been discussed in Chapter 2.5.

The cloud deployment of a Dash application to the Azure cloud platform is a straightforward process. We use Docker, an application development platform, in the cloud deployment. The first step is to create a Dockerfile, which is a text document that contains the commands a user could call on the command line to assemble an image (Docker Documentation, 2021). The Dockerfile can be found in Appendix D. It provides a series of instructions to the system to create a Docker image of the Dash application. It requires a *requirements.txt* file, which lists the installed libraries in the Python environment where the Dash application was implemented. The *requirements.txt* file can be found in Appendix E. A Docker image is a file used to execute the code inside a Docker container, which in this case contains the Dash application. With the Dockerfile, the requirements.txt file, and the Dash application script, and after fixing some inconsistences in the library dependencies, a Docker image was created.

The Docker image contains the code needed to deploy the dashboard. The next step is to upload the image to the Docker Hub. The Docker Hub is a repository of container images (Docker, 2021) where a user can upload their Docker images in public or private repositories. After

creating an account and a repository, we can publish the Docker image to the repository. Once that is done, we manually deployed it to the Azure App Service. The final step is to create a web app from a container that comes from the Docker Hub. The web app must be configured so that the image and tag of the container matches the ones in our repository in the Docker Hub. Once the web app is created, the deployment begins. The deployment was followed through the Azure Deployment Centre logs. After several attempts changing some parameters, for instance changing the website port and increasing the container start time limit (since the default limit is not enough for this web app to be deployed), the web app was finally deployed and operational, and thus AIOps4B deployed on the cloud. The current implementation of AIOps4B can be found at *aiops4b.azurewebsites.net.*

4. Results

4.1. Overview

In this chapter of the report, the results of the project will be presented. The first subsection will explore the different parts of the AIOps4B dashboard, giving detailed explanations of the different graphs and results. The second subsection will describe and show the metrics for the models deployed in the trend analysis and forecast. Furthermore, a video demonstration of the AIOps4B app will be included in the submission along with this report and the code, showcasing the results of the project in a clearer way.

4.2. AIOps4B App

The AIOps4B app is the main result of this project. It was first developed and implemented on local but was then deployed to the cloud and can be accessed at *aiops4b.azurewebsites.net.* In this section, a detailed explanation of every part of the app will be provided.

4.2.1. Dashboard Overview

Figure 8 shows a general view of the AIOps4B dashboard. The dashboard is divided into three sections:

- Revenue KPI: this subsection includes the revenue time-series plot with its trend and the detected changepoints, as well as the forecast data.
- Root-Cause Analysis: this subsection includes the root-cause analysis divided into the two areas defined in Section 3, impact of metrics on the trend and impact of events on the trend.
- Deep Analysis: this subsection, which is only shown if the selected changepoint has a negative trend, includes the most common problems that the orders had based on the three metrics.

These three sections of the dashboard will be described in the following subsections.

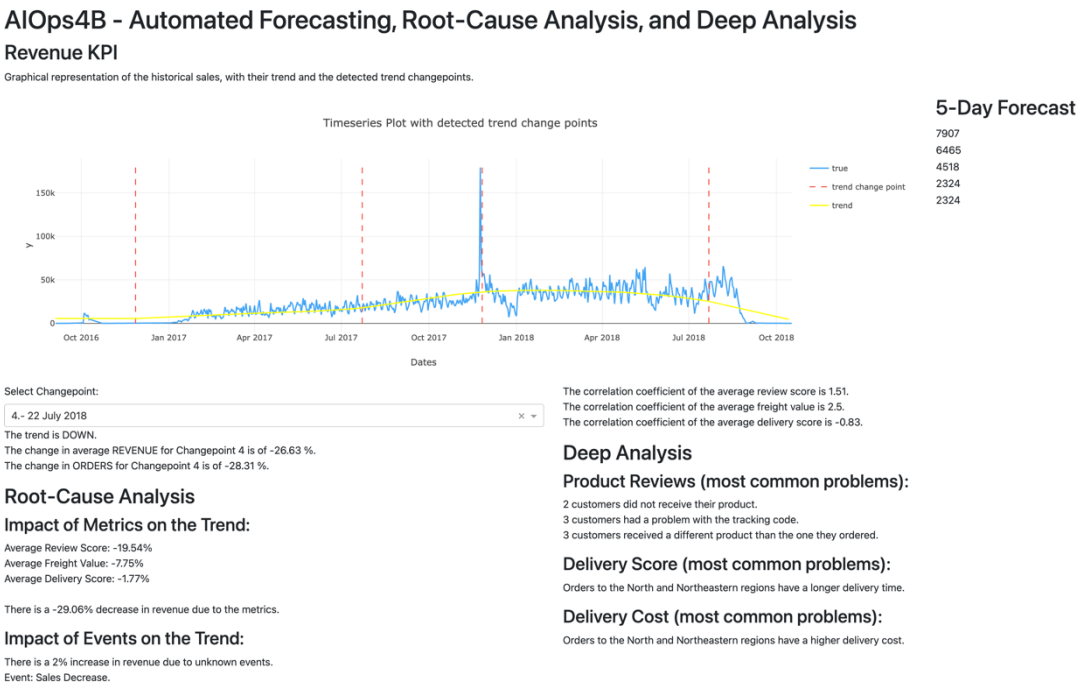

Figure 8 - AIOps4B general view (with Changepoint 4 selected)

4.2.2. Revenue KPI: Time Series, Changepoints, and Forecast

The first and most prominent section of the AIOps4B application is what we called the Revenue KPI, which includes the time-series plot of the revenue, its trend, the detected changepoints, and the forecast. Figure 9 shows the Revenue KPI in detail.

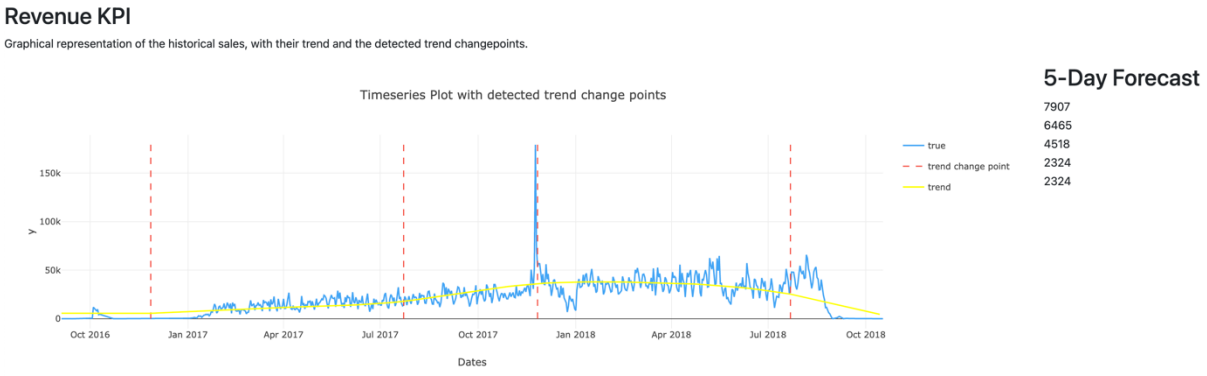

Figure 9 - Revenue KPI graph and forecast

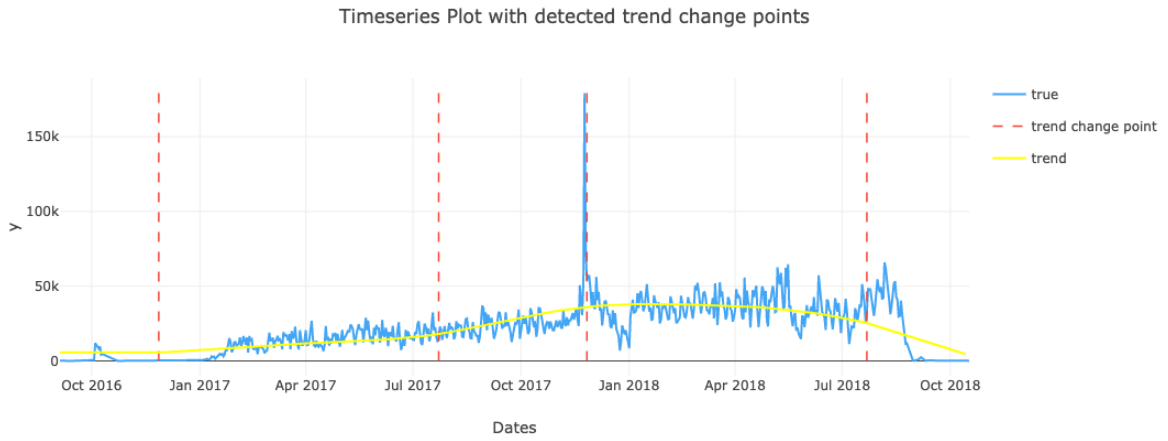

Figure 10 - Revenue time-series plot with trend and detected changepoints

First, we will discuss the graph. Figure 10 shows only the revenue time-series plot. The blue line represents the daily revenue that the Olist store has obtained, ranging from October 2016 to October 2018. Even though there is clearly some missing data between October 2016 and January 2017, and in October 2018, the existing points needed to be left in for the model to work. This process will be discussed in depth in Chapter 6.2. The interesting elements of Figure 10 are the yellow line, which represents the revenue trend, and the red, dashed lines, which represent the changepoints. These lines, both the trend and the changepoints, have been detected by the Greykite library, following the model defined with the parameters presented in Chapter 3.5. The changepoints are the points in time in which the trend changes, and the analysis will be based on them. From here on, the changepoints will be denominated as Changepoint 1, 2, 3, and 4, which correspond to the dates 27 November 2016, 23 July 2017, 26 November 2017, and 22 July 2018, respectively. Since Changepoint 1 has been detected in a period where most data are missing, it will be omitted from the analysis, and we will only focus on Changepoints 2, 3, and 4.

5-Day Forecast

7907
6465
4518
2324
2324

Figure 11 - 5-day forecast for the revenue

The second part of the Revenue KPI is the forecast, one of our main objectives of AIOps4B. The numbers in Figure 11 represent the forecast for the revenue for the 5 following days of the dataset (the forecast has been set to include 5 days only as a demo but can be changed in the code to any number of days). The forecast has been obtained using the complete Greykite model that was defined in Chapter 3.5. These data will help businesses to predict their revenue and anticipate a decline.

4.2.3. Root-Cause Analysis

One of the main parts of AIOps4B is the root-cause analysis section. However, before exploring the root-cause analysis section of the application, there is a section above it that needs to be explained.

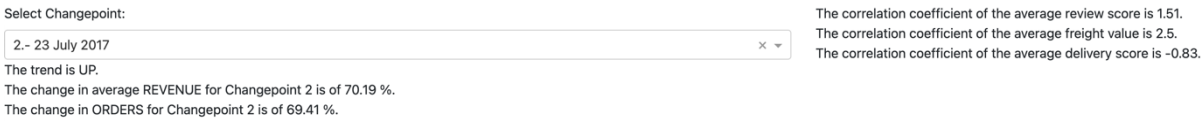

Figure 12 - Further information for Changepoint 2

Figure 12 shows some more information that is contained in the app. First, there is a dropdown menu that allows the user to select a changepoint from the four that have been detected. This selection causes the app to show the information and the root-cause analysis for that specific changepoint. In Figure 12, the selected changepoint is Changepoint 2, which corresponds to 23 July 2017. Below the dropdown menu, there is some general information about the revenue, the trend, and the orders for the selected changepoint. It tells the user whether the trend goes up or down, the change in average revenue, and the change in orders. These data have been calculated following the methods presented in Chapter 3.6. On the right, the correlation coefficients of the three metrics are showed. These coefficients have been calculated as shown in Chapter 3.6 and do not change when switching the changepoint in the dropdown menu.

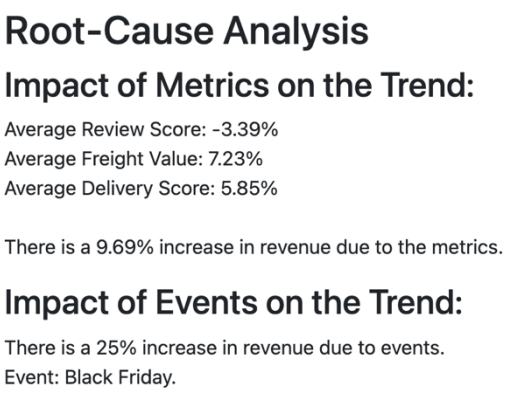

Figure 13 - Root-cause analysis of Changepoint 3

Figure 13 shows the root-cause analysis of Changepoint 4. As we mentioned in Chapter 3.6.1, the root-cause analysis was divided into two parts: the impact of the metrics on the trend and the impact of external events on the trend. The metrics are the average review score, average freight value, and average delivery score. Their changes were calculated with the percentage change that was explained in Chapter 3.6.1. The total increase or decrease in revenue due to the metrics is shown below. The impact of external events was calculated using the correlation coefficients that appear in Figure 12. For instance, in Figure 13, there is a 25% increase in revenue in Changepoint 3 due to external events. In this case, the external event is Black Friday.

4.2.4. Deep Analysis

The final section of the AIOps4B application is the Deep Analysis. This section only appears in the app when a changepoint with a negative trend is selected in the dropdown menu. In the use case we are working with, the Deep Analysis only appears when Changepoint 4 is selected, since it is the only one with a negative revenue trend. Figure 14 shows the Deep Analysis section.

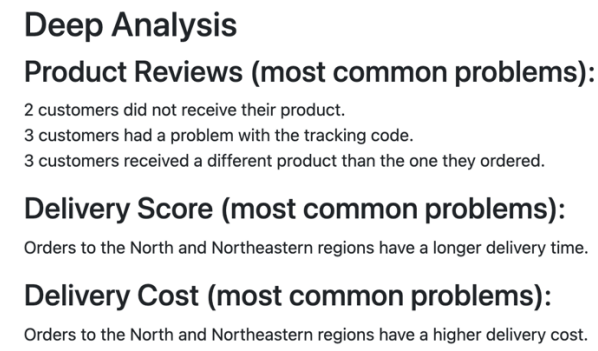

Figure 14 - Deep Analysis of Changepoint 4

The Deep Analysis, like the first section of the root-cause analysis, is divided into three subsections, corresponding to the three metrics. This section displays the most common problems that the users have encountered in their orders and have caused the revenue trend to be negative. These problems have been discovered following the methods that were explained in Chapter 3.6.2; i.e., natural language processing for the product reviews and geolocation analysis for the delivery score and cost. For instance, for the product reviews, some customers did not receive their products, others had trouble with the tracking code, while others received different products than the ones they ordered. In the case of the delivery score and cost, we found out through geolocation analysis of the orders that the orders to the North and North-eastern regions of Brazil have a longer delivery time and a higher delivery cost.

5. Discussion

5.1. Overview

This section will examine to what extent have the objectives set out in Chapter 1.2 been fulfilled. A rundown of the objectives and the way AIOps4B accounts for them will be provided. Finally, we will analyse to what extent has the research question posed in Chapter 1.1 been answered.

5.2. Fulfilment of the Objectives

In this subsection the choice of the objectives and the extent to which they have been fulfilled in the project will be discussed. Apart from the five objectives that were defined in Chapter 1.2, an extra objective that was present in the project proposal but was not carried out in the project will also be studied.

5.2.1. Analyse historical data

This was the first step of the project pipeline and a somewhat generic objective. The search for a dataset was not a task as simple as it was initially thought. Our requirements for the dataset were that it should be an e-commerce dataset, as this would suit most use cases for AIOps4B, it should have interesting features that could be optimized and analysed, and it should include several years of sales. Many datasets, both public and paid, were researched, and we finally settled for the Brazilian E-Commerce Public Dataset, since it fits the preestablished requirements. Although the Brazilian E-Commerce Dataset has some limitations, as we will see in Chapter 6.2, it can be considered as a suitable dataset for the project. Its context and characteristics were fully understood, as Chapter 3.3 shows, where we provided a detailed explanation of the use case and the different datasets that comprise the Brazilian E-Commerce Dataset. Several metrics were created and then selected, as shown in Chapter 3.4. The objective of analysing historical data can be regarded as achieved.

5.2.2. Understand Revenue Trends

Understanding the revenue trends is one of the main objectives of the project, and, as we defined it, it includes both the visualisation and the comprehension of the trends. This objective was achieved mainly with the help of the Greykite library, which was introduced in Chapter 2.4.2. After obtaining the final set of metrics following the methods of Chapter 3.4, a Greykite model was used to find the revenue trend and plot it. With the addition of revenue changepoint detection, and the analysis that was applied afterwards, we can conclude that the objective of

understanding revenue trends was met successfully. Figure 10 shows the result of this objective.

5.2.3. Forecast the Future Trend

Forecasting the revenue allows companies that use AIOps4B to foresee a decline in revenue. This was accomplished with another Greykite model, as shown by Chapter 3.5. Figure 11 shows the 5-day forecast for the revenue; however, any number of days can be set in the code. This objective has also been met.

5.2.4. Find Root-Causes

Finding the root-causes for the decline in revenue was another major part of the project. Our approach was to base the root-cause analysis on the trend changepoints that had been detected with the Greykite library. The root-cause analysis itself was based on two aspects: the metrics, which had been defined before, and the external events that happened in the period of time that covers the dataset. This objective was met following the methods presented in Chapter 3.6 and the results were showcased in Chapter 4.3.3. Not only were the root-causes found, but they were also quantified and analysed based on the metrics and the external events.

5.2.5. Provide Deeper Analysis

Deeper root-cause analysis was also provided in the AIOps4B app. It is based on the particularities of the metrics, and its approach varied depending on them. Natural language processing and geolocation analysis were the methods, as presented in Chapter 3.6.2, that were used for what we called Deep Analysis. The aim was to further understand the real root-cause of a potential decline in revenue. As we showed in Chapter 4.3.4 for Changepoint 4, this objective was met successfully.

5.2.6. Other Objectives

One of the main objectives that was set in the project proposal was to provide recommendations for actions that would fix a decline in revenue. These recommendations would be based on the metrics for each use case, in the same way as the root-cause analysis. Since the implementation was not completed, this objective will be set as future work and further explained in Chapter 6.3.

5.3. Answering the Research Question

The research question, which asked whether an AIOps framework for businesses to optimize their revenue be developed, was answered positively. AIOps4B, an open-source project, was developed successfully with that purpose. Even though it is only tailored for one use case in its current implementation, it could be generalized for its use with other datasets. This topic will be discussed in Chapter 6.3.

6. Evaluation, Reflections, and Conclusions

6.1. Overview

This section will serve as a conclusion of the project and the dissertation. The limitations of the project will be studied, as well as some possibilities for future work. Finally, some personal reflections and experiences regarding this project will be presented.

6.2. Limitations of the Project

Although the development and implementation of AIOps4B and the overall project was successful, there were still some limitations that need to be addressed. These are aspects of the work that possibly were not the ideal approaches or things that could be improved in a future implementation of AIOps4B, given that the application is still a work in progress and could be continued in the future by myself or by NTT DATA.

The first major limitation of the project concerns the dataset. As we indicated in Figure 10 of Chapter 4.3.2, there are some missing data between October 2016 and January 2017, and in October 2018. These missing data needed to be left in for the model to work. Nevertheless, working with a dataset that has that amount of missing data causes the model to not be as precise as we would want it. This is reflected in the model metrics, which we have not included in the results section as they were not the focus of the project. When taking all the data points and computing the model metrics of the Greykite forecasting model, which we defined in Chapter 3.5, the mean test MAPE (Mean Absolute Percentage Error) was immense, 21077.3. However, when removing from the metrics dataset the last set of missing data, in October 2018, the MAPE is reduced significantly, up to 11.7. For our model and the later analysis to work correctly, we decided to leave the missing data and continue with the analysis. This is one of the most important limitations of the project, which could be fixed in a later implementation using a better dataset, with more complete data and less missing data. In that case, the Greykite models that have been defined in AIOps4B will work better.

Moreover, the application is tailored only to the Brazilian E-Commerce Dataset. Although AIOps4B works and is functional for this use case, in its current implementation it cannot accommodate other datasets and use cases. However, this could be easily fixed in later implementations of the application. The app could be made more generic to accommodate other datasets. This situation will be discussed in the future work section of the report, Chapter 6.3.

The main limitation of the project concerns recommendations. One of the objectives set out in the project proposal was for AIOps4B to provide recommendations for actions to fix a negative revenue forecast. The implementation for the recommendations was not completed, and it will be mentioned as future work in Chapter 6.3 of the report. The idea was to provide recommendations for specific actions based on the metrics for each use case so that a business can fix a negative trend.

Another limitation of the project was the approach taken for the feature selection, which we discussed in Chapter 3.4. As we presented there, it was not the ideal method for selecting the most relevant metrics. In future work, other methods of feature selection could be explored.

6.3. Future Work

Since an application like the one developed is an ongoing project, several possibilities for future work will be listed below. They have been divided into two categories: future implementation of the app, which would be suitable next steps for further AIOps4B implementation, and optimizations, which are changes or improvements that could be applied to the already existing AIOps4B implementation.

The most important future step in AIOps4B would be a recommendations feature. Although this topic has already been addressed in Chapter 6.2, it would be the ideal first step in future work. This feature would allow AIOps4B to provide recommendations based on the use case and the metrics when the revenue is predicted to go down, so that the users can remedy the problem before it happens. A recommendations feature would also further differentiate AIOps4B from the rest of the similar technologies, i.e., Thoughtspot and Outlier, who do not provide this feature. The recommendations would ideally be tailored to the metrics that each use case would have, so that the users would know exactly which action to perform in order to fix the decline in revenue (or any other KPI).

Another possible future step in the implementation of AIOps4B would be the integration of chatbots to answer business questions directly, providing an alternative to KPI dashboards. This could be an alternative step to the recommendations feature or the following step after the recommendations feature implementation. If it is the latter, businesses could ask the app if there will be an increase in revenue the following month, for instance, and the app could provide the predicted revenue as well as recommendations to achieve the revenue objective. The

implementation of the chatbot feature could be done with Amazon chatbots (Amazon Web Services, 2021).

These two features are some possibilities that would expand the AIOps4B app in new directions. However, there are also several optimizations or improvements that could be applied to the current implementation of AIOps4B. The first optimization concerns the data, specifically the missing data. As we showed in Figure 10 of Chapter 4.3.2, there are some missing data between October 2016 and January 2017, and in October 2018. These missing data needed to be left in for the model to work. Nonetheless, this situation is not ideal for the project. One possible future solution would be to add synthetic data. With synthetic data, those two spots could be filled in, improving the model. Moreover, synthetic data could be created for future years, so that other effects such as seasonality could be studied.

Another possible optimization comes from the trend analysis section. We have used the Greykite library since it is the most advanced trend analysis and forecasting library. However, the use of the Prophet library could be beneficial and yield better results. Following van der Merwe's (2018) article, a new AIOps4B implementation with Prophet could improve the models. Prophet is the classic time-series analysis library, but new ones are currently being developed. New solutions attempt to bind together classical methods and the new deep learning models. For instance, Triebe et al. (2021) present NeuralProphet, a new library which, serving as a successor to Prophet, is based on PyTorch and incorporates deep learning methods, which allow developers to scale the framework. An implementation with NeuralProphet would bring a new deep learning approach to the project.

Finally, a further automation of end-to-end business processes would optimize the application and make it more accessible to users. In other words, the current implementation of AIOps4B is not highly customizable, as it is adapted for the Brazilian E-Commerce Dataset. Future work could include increasing the adaptability of the app to other datasets and use cases, making the root-cause analysis and deep analysis more generic or adaptable. The final step in this new implementation would be the automation of the whole pipeline, having the application create the metrics itself and performing the trend and root-cause analysis, therefore automating the whole end-to-end process.

6.4. Personal Experience

From a personal perspective, this project was a challenging, albeit rewarding one. Moreover, it not only served as my first full, end-to-end Data Science project, it was also a crucial part of my first professional experience. It has allowed me to delve into the field of time-series analysis, which I did not have much experience with, and has arisen interest in trend analysis and time-series forecasting, accompanied by an in-depth analysis of the current libraries. This project has also given me some more experience in working with cloud technologies, which I believe will be vital in the short-term future of Data Science and my future work. The cloud deployment of the AIOps4B application was a challenging but gratifying process which allowed me to dive into the cloud technologies.

Regarding my professional endeavour, the experience was a pleasant and hugely educational one. It allowed me to work for the first time in a compelling project in a professional environment. The experience was satisfying for both the company and me. As a sidenote, on November 18, 2021, I gave a presentation (Tech Talk) for some colleagues from NTT DATA UK along with my supervisor from my time in the company, which was a magnificent way of showing the work I did on the project and showcasing the results, along with a demo of AIOps4B. A similar video demo will be provided in the submission files long with the code and this report.

As a final thought, taking into account the possibilities of future work that could be implemented, AIOps4B could become a useful, open-source application for trend analysis and forecasting. Even though its current implementation is not fully optimised for any dataset or use case, AIOp4B is already a functional trend analysis and forecasting application. Since the approach from the start of the project was to create an open-source platform that would have the potential to become a library, the APIs that form the main section of the code were uploaded to the Python Package Index (PyPI) and can be downloaded and used at *https://pypi.org/project/aiops4b-NTTDataUK/*. Therefore, any user can download the library and use the AIPs to obtain a similar dashboard on their computers.

Glossary

- AIOps: Artificial Intelligence for IT Operations, the application of big data and machine learning techniques to automate IT processes.
- Alpha value: also known as significance level, a value that is set as threshold and below which the null hypothesis, H_0, is rejected.
- ARIMA: autoregressive integrated moving average, a model for time series forecasting.
- AWS: Amazon Web Services, Amazon's cloud platform.
- Docker image: a standalone software package that contains all the code needed to run an application.
- GCP: Google Cloud Platform, Google's cloud service.
- Hypothesis test: a statistical method of determining some statement starting from two different hypotheses, the null hypothesis and the alternative hypothesis.
- KPI: key performance indicator, a measure of success of a business or of a particular project or product.
- MAPE: mean absolute percentage error, a metric that shows the accuracy of a forecasting method.
- OLS: ordinary least squares, a statistical method for obtaining the parameters in a linear regression.
- p-value: a value that indicates the level of significance in a hypothesis test.

References

- Ahmed, N. K., Atiya, A. F., Gayar, N. E., and El-Shishiny, H. (2010) 'An empirical comparison of machine learning models for time series forecasting', *Econometric Reviews*, *29*(5-6), pp. 594-621.
- Amazon Web Services (2021) *Conversational AI and Chatbots - Amazon Lex.* Available at: https://aws.amazon.com/lex/?nc1=h_ls (Accessed: 7 December 2021).
- Businesswire (2019) *New McAfee Report Finds Eighty-Seven Percent of Companies Experience Business Acceleration from Use of Cloud Services*. Available at: https://www.businesswire.com/news/home/20190617005945/en/New-McAfee-Report-Finds-Eighty-Seven-Percent-Companies (Accessed: 13 November 2021).
- Docker (2021) *Docker Hub - Container Image Library*. Available at: https://www.docker.com/products/docker-hub (Accessed: 23 September 2021).
- Docker Documentation (2021) *Dockerfile reference*. Available at: https://docs.docker.com/engine/reference/builder/ (Accessed: 25 September 2021).
- Gartner (2021) *Gartner Forecasts Worldwide Public Cloud End-User Spending to Grow 23% in 2021*. Available at: https://www.gartner.com/en/newsroom/press-releases/2021-04-21-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-23-percent-in-2021 (Accessed: 13 November 2021).
- Goldberg, D. and Shan, Y. (2015) 'The importance of features for statistical anomaly detection', *7th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 15)*.
- Google Cloud (2021 - a) *What is Cloud Computing?*. Available at: https://cloud.google.com/learn/what-is-cloud-computing (Accessed: 13 November 2021).
- Google Cloud (2021 - b) *Identity and Access Management (IAM)*. Available at: https://cloud.google.com/iam (Accessed: 13 November 2021).
- Greykite Library documentation (2021 - a) *Tune your first forecast model.* Available at: https://linkedin.github.io/greykite/docs/0.1.0/html/gallery/tutorials/0100_forecast_tutorial.html#sphx-glr-gallery-tutorials-0100-forecast-tutorial-py (Accessed: 20 July 2021).

- Greykite Library documentation (2021 - b) *Changepoint Detection.* Available at: https://linkedin.github.io/greykite/docs/0.1.0/html/gallery/quickstart/0200_changepoint_detection.html#sphx-glr-gallery-quickstart-0200-changepoint-detection-py (Accessed: 21 July 2021).
- Hersher, R. (2017) *Amazon And The $150 Million Typo*. NPR. Available at: https://www.npr.org/sections/thetwo-way/2017/03/03/518322734/amazon-and-the-150-million-typo (Accessed: 13 November 2021).
- Hosseini, R. (2021) *Greykite: A flexible, intuitive, and fast forecasting library*. LinkedIn Engineering. Available at: https://engineering.linkedin.com/blog/2021/greykite--a-flexible--intuitive--and-fast-forecasting-library (Accessed: 20 July 2021).
- Hosseini, R., Yang, K., Chen, A., and Patra, S. (2021) 'A flexible forecasting model for production systems'.
- Hyndman, R. J. (2014) *Forecasting: principles and practice*. Available at: https://robjhyndman.com/uwafiles/fpp-notes.pdf (Accessed: 15 November 2021).
- Kaggle (2018) *Brazilian E-Commerce Public Dataset by Olist*. Available at: https://www.kaggle.com/olistbr/brazilian-ecommerce (Accessed: 10 July 2021).
- Lerner, A. (2017) *AIOps Platforms*. Gartner. Available at: https://blogs.gartner.com/andrew-lerner/2017/08/09/aiops-platforms/ (Accessed: 9 November 2021).
- Levin, A., Garion, S., Kolodner, E., Lorenz, D., Barabash, K., Kugler, M. and McShane, N. (2019) 'AIOps for a Cloud Object Storage Service', *2019 IEEE International Congress on Big Data (BigDataCongress)*, pp. 165-169.
- Luxner, T. (2021) *Cloud Computing Trends: 2021 State of the Cloud Report*. Flexera Blog. Available at: https://www.flexera.com/blog/cloud/cloud-computing-trends-2021-state-of-the-cloud-report/ (Accessed: 13 November 2021).
- Moogsoft (2021). Available at https://www.moogsoft.com/ (Accessed: 10 November 2021).
- Notaro, P., Cardoso, J., and Gerndt, M. (2020) 'A Systematic Mapping Study in AIOps', *ICSOC 2020 Workshops. ICSOC 2020. Lecture Notes in Computer Science*, vol 12632.
- OpsRamp (2021). Available at https://www.opsramp.com/ (Accessed: 10 November 2021).

- Outlier (2021). Available at https://outlier.ai/ (Accessed: 10 November 2021).
- Panini, T. (2021) *E-Commerce Sentiment Analysis: EDA + Viz + NLP*. Kaggle. Available at: https://www.kaggle.com/thiagopanini/e-commerce-sentiment-analysis-eda-viz-nlp (Accessed: 30 August 2021).
- Petersen, M. (2018) *Outlier Introduces AI Powered Automated Business Analysis Platform*. Outlier. Available at: https://outlier.ai/2018/10/10/outlier-introduces-ai-powered-automated-business-analysis-platform/ (Accessed: 10 November 2021).
- Sigelman, B. H., Barroso, L. A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., Jaspan, S., and Shanbhag, C. (2010) 'Dapper, a large-scale distributed systems tracing infrastructure', *Google Technical Report.*
- Taylor, S. J. and Letham, B. (2018) 'Forecasting at scale', *The American Statistician*, *72*(1), pp. 37-45.
- Thoughtspot (2021). Available at https://www.thoughtspot.com/ (Accessed: 10 November 2021).
- Triebe, O., Hewamalage, H., Pilyugina, P., Laptev, N., Bergmeir, C., and Rajagopal, R. (2021) 'NeuralProphet: Explainable Forecasting at Scale'.
- van der Merwe, R. (2018) *Implementing Facebook Prophet efficiently*. Medium. Available at: https://towardsdatascience.com/implementing-facebook-prophet-efficiently-c241305405a3 (Accessed: 7 December 2021).
- Yu, L. and Liu, H. (2003) 'Feature selection for high-dimensional data: A fast correlation-based filter solution', *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 856-863.
- Zenoss (2021). Available at https://www.zenoss.com/ (Accessed: 10 November 2021).
- Žunić, E., Korjenić, K., Hodžić, K. and Đonko, D. (2020) 'Application of Facebook's Prophet Algorithm for Successful Sales Forecasting Based on Real-world Data', *International Journal of Computer Science and Information Technology*, 12(2), pp. 23-36.

Appendix A: Project Proposal

AIOps4B Framework

Julian Gonzalez Galvez

Introduction

The purpose of this project is to create a framework for business decision making. The objective will be to maintain a healthy revenue KPI by analysing historical data, understanding the trends, forecasting the future revenue trend, and if the forecast indicates a negative trend, understand the root cause and recommend actions to fix the issue.

The generated product of the work will be a framework that optimizes a business' revenue KPI. The beneficiaries of the work include businesses, which use the revenue KPI as a measure to achieve their objectives. With AIOps4B, they will be able to foresee a decline in revenue and remedy it before it happens.

The following research question is posed:

Can an AIOps framework for businesses to optimize their revenue be developed?

Critical Context

AIOps

Following the definition by Notaro, Cardoso, and Gerndt (2020), AIOps (Artificial Intelligence for IT Operations) studies the use of Artificial Intelligence in the management and improvement of IT services. AIOps use Machine Learning and Big Data to monitor situations and provide recommendations to fix issues and failures, such as the revenue trend in this project.

Similar Products

AIOps has been in development for some years now, and there are several products that perform a similar pipeline as this project: automatically detect trends in data, perform root-cause analysis, and make recommendations to fix an issue. However, they do not offer the same solution as AIOps4B. The two most important products are ThoughtSpot and Outlier.

ThoughtSpot

ThoughtSpot is a service that offers interactive apps that help the user to extract insight from their data (ThoughtSpot, 2021). Their main claims are that users can search their cloud data, discover hidden insights from them, and create interactive dashboards. Their SpotIQ algorithm identifies anomalies, finds the relationships between the features and the outcome, find the trends, and delivers insights, without being a black box to the user. However, even though

ThoughtSpot would identify the problems leading to a decrease in revenue, taking the example of this project, it does not recommend a specific solution to the problem, it just identifies it. The final objective of AIOps4B is to provide the user with the specific measure to revert a negative future trend in revenue to a positive one.

Outlier

Outlier is a similar AIOps service that automatically analyses data features that the user inputs to the system, and delivers insights to them (Outlier, 2021). Their most important feature is that Outlier's systems perform data analysis, choose the most important insights using their recommender systems, and send them via email to the user every morning. Outlier also has a root-cause feature that identifies the root causes of the changes in business metrics, and displays them in a dashboard (Petersen, 2018). Nonetheless, the issue with Outlier is the same as ThoughtSpot: while they flag the problem in the future trend and identify the root-cause, they do not recommend a specific solution to the problem.

Trend Analysis, Forecast, and Root-Cause Analysis

Since AIOps4B will not use any of these products, new technologies must be found for the most important parts of the pipeline: trend analysis, forecast, and root-cause analysis. The main implementation of the project will be in Python, so Python libraries that tackle these topics must be found. These three different tasks can be assembled into one: time-series forecasting and analysis. There are two main Python libraries for this task: Prophet and Greykite.

Prophet

Prophet, developed by Facebook, is used for time-series forecasting and analysis. According to Žunić, Korjenić, Hodžić, and Đonko (2020), Prophet fits non-linear trends with yearly, weekly, and daily seasonality, and it includes holiday effects. Prophet works well with outliers and, following Taylor and Letham (2017), it is used across Facebook. However, Prophet is not able to incorporate new, special, external events to the forecast, which is an important aspect of a project like this.

Greykite

Greykite is Linkedin's response to Facebook's Prophet. It is also a time-series forecasting library that includes seasonality and holiday effects on the model. However, most importantly, Greykite also offers the possibility of including special events or anomalies, as Hosseini, Yang, Chen, and Patra (2021) show in their work. Figure 1 shows the architecture of Silverkite, Greykite's algorithm for forecasting. Silverkite is fast and highly customizable, so it will be adapted to our use case. Greykite also has integration with Prophet, so both algorithms may be used in the implementation if the results are better with Prophet.

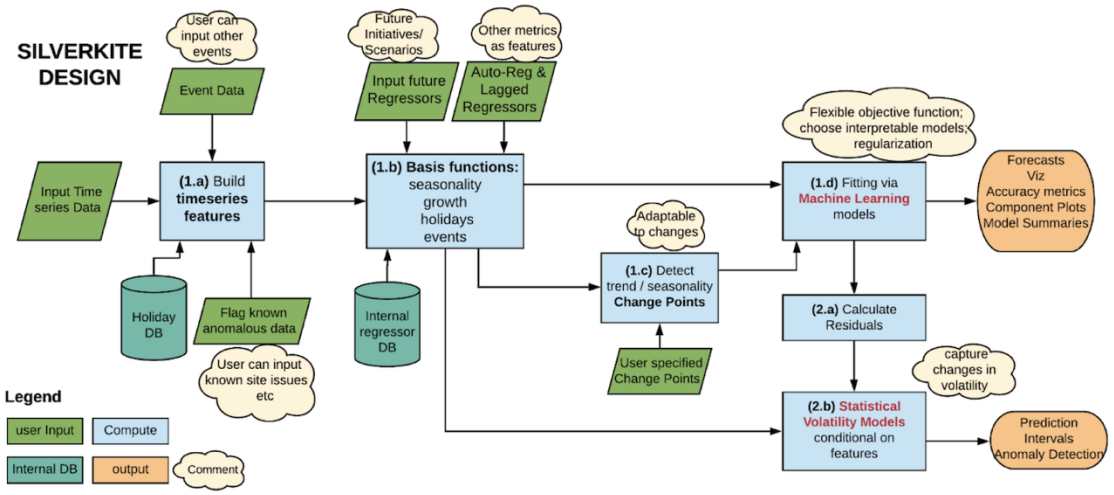

Figure 1 - Architecture of Silverkite (Hosseini, 2021)

Approaches: Methods & Tools for Design, Analysis & Evaluation

Design

The design of the project started with devising the work plan that can be seen in Figure 3. After that, a thorough literature research was conducted to find the best technologies and libraries for each step of the pipeline.

Literature Research

The current literature research was performed looking online mostly for the best technologies available for each step of the pipeline, as well as investigating about similar projects to this one. After some days of research, some initial technologies have been selected for the project. Nonetheless, throughout the process, further literature research and review will be performed, according to the needs of the project. Moreover, this is a fast-changing field, and new technologies are presented frequently, so staying up to date is vital. Research papers will be searched in Google Scholar and following references from other papers, as well as new publications. Keeping up with new libraries, such as Greykite, will also be essential.

Dataset

After the literature research, which will realistically continue throughout the whole project, the search for a dataset began. The ideal dataset would include several years of ecommerce sales, would capture special events such as holidays, and would ideally include information about marketing campaigns or events. A preselection of datasets was made according to these parameters, and *Brazilian E-Commerce Public Dataset* was chosen (Kaggle, 2018). The rest of the datasets will be kept in case there is need for a backup.

Feature Engineering

The first step of the data pipeline, after selecting a dataset, is data cleansing. Once the data is cleaned and ready to use, feature engineering begins. The objective is to create a dataframe that can be input to the Greykite library, which will be used for the trend analysis and forecast. This dataframe needs to be grouped by a certain time frame (daily, in our case), must also include the daily revenue as the target column, and the metrics grouped by their daily values. The Brazilian E-Commerce Dataset is a database which contains several datasets, which need to be joined to create the different metrics. Initially, the metrics were to be created using the Featuretools and Autonormalize libraries, but after some tests with them, the metrics were created manually using Pandas.

The metrics are divided by new customer sales and returning customer sales, and are based on product performance and delivery performance. One of the objectives of the feature engineering phase is to find a way to explain the daily revenue based on the metrics, i.e., find the coefficients of the following equation:

$$REVENUE = New\ Customer\ Sales * [(x_1 * Product\ Performance) + (x_2 * Delivery\ Performance)] + Returning\ Customer\ Sales * [(x_3 * Product\ Performance) + (x_4 * Delivery\ Performance)].$$

Feature Selection

After creating the metrics and organising them into the dataframe, feature selection will take place. Two approaches will be tested: a p-value approach and a Pearson correlation coefficient approach. They will both be applied to see which one yields better results.

The p-value approach is based on a linear regression. Although this approach is not the ideal one for feature selection according to the literature, it will be tested, nonetheless. First, (Ordinary Least Squares) regression with the statsmodels library and all the metrics is performed against the revenue. The *summary* function provides the p-value of every metric; it indicates which metrics are significant for the revenue. An alpha value of 0.05 is set, which marks several metrics as significant. The summary function also shows the coefficients for those metrics.

The Pearson Correlation Coefficient approach is a more accepted approach in the literature, although there are still some irregularities such as having to choose the number of best metrics wanted. Here, the *scikit-learn* library is used. The *SelectKBest* function is used, with a score function of *f_regression* and *k*, the number of metrics that the user wants, as inputs. The algorithm is then fit to the metrics. The result is an array of the *k* selected features, which can be compared to the metrics dataframe to see which ones they are.

Trend Analysis

The objective is to understand how the revenue trend changes based on the metrics selected before, as well as other internal and external events. The daily revenue can be plotted against the time, and in this visualization we will see how the revenue trend has changed.

Forecast

A Python library that is able to analyse past trends and special events, incorporate new events, forecast the monthly revenue, and interpret the metrics' impact, is needed in this step of the

project. Prophet is a framework for forecasting time series which can handle the analysis of past trends and the forecast of monthly revenue, while considering seasonality. However, the Greykite library will instead be used in this project, as it can incorporate new events too.

The pipeline for the forecast is the following. The already prepared dataframe is input to Greykite. We define external events such as holidays, we include seasonality, tune the model, and train it with the metrics. Greykite can also detect the changepoints were the trend changes; these will be relevant in the Root-Cause Analysis section. The result will be a plot which represents the revenue trend, with its changepoints, and the forecast for the next 30 days. After this, a certain objective for a revenue growth will be set, so that the prediction can be flagged as positive or negative.

Root-Cause Analysis

Greykite can detect the changes in the trend. The root-cause analysis phase of the project comprises two parts: the past trend changepoints must be explainable based on events and metrics, or else they will be identified as anomalies, and, if the revenue growth objective for the forecast is not met, the root-cause for this must be provided through the revenue drivers. If the trend changepoints are found to be caused by a metric that the user can control, such as delivery price, this metric can be optimized, which leads into the next phase of the project.

Recommendations

If the revenue in the forecast does not meet the previously set objective, the root-cause must be found. If it is found that the cause is a metric controlled by the user, the framework will study how to optimize this driver. AIOps4B will find the best value for this feature and display the results in a graph, in order to show the user the optimal value to fix the future revenue trend.

Optimization

This phase of the project will consist of several tasks to improve and finalize the framework. The aim is to have an operational framework by this stage of the project, and spend the allocated time for this phase to improve it and possibly add new features, such as an incorporation of semantic data. The other two main optimizations are providing interactive dashboards and a deployment to the cloud. These optimizations will be added if there is enough time left in the project, but they are not vital to the framework.

Interactive Dashboards

The final visualization of the framework should be an interactive dashboard where the user can see the most important information, especially the recommendations for the future revenue trend growth. The potential Python library to be used for these dashboards is Dash.

Cloud Deployment

The other main optimization is a cloud deployment to Azure. The framework will first be implemented locally with PyCharm, but the use of cloud is a way of incorporating new technologies to the project.

Report

The report will be started as soon as the project begins. Even though the framework will not have yet been implemented, the composition of the report could be started for instance in literature review matters. Eight weeks have been allotted at the end of the project for the completion of the report, redacting several drafts and reviewing them.

Additional Time

Since the eight-week period for the report is a prolonged period of time, the first two weeks can be used for delays in any part of the project, and to study and analyse any new, unexpected findings. If there are no delays in the development of the project, these extra days will be allocated to the writing of the Project Report.

Project Meetings

Project meetings have been set once every two weeks with the project supervisor. The meetings are telematic, through Microsoft Teams, and are used to catch up and keep the supervisor updated on how the project is going. Further along the project, the meetings will help with the writing of the report.

Risks

The following risk register includes the foreseeable risks that the project could suffer, the likelihood for them to happen, their consequences and impact, and the ways to solve those problems.

Risks	**Likelihood (1-3)**	**Consequences (1-5)**	**Impact (L * C)**	**Solution**
Chosen libraries for trend analysis do not work	2	5	**10**	Search for other libraries
Implementation takes longer than expected	3	5	**15**	Some extra days have been allocated
Dataset is not ideal	3	4	**12**	Search for alternative datasets
Code is lost	1	5	**5**	Use Gitlab
Cannot deploy to cloud	2	4	**8**	Work only on local
Results are not as expected	2	4	**8**	Perform a comparison

Table 1 - Risk Register

Work Plan

Figure 2 shows the graphical work plan that was devised for this project. It includes a representation of every step described in the Approaches section of the proposal.

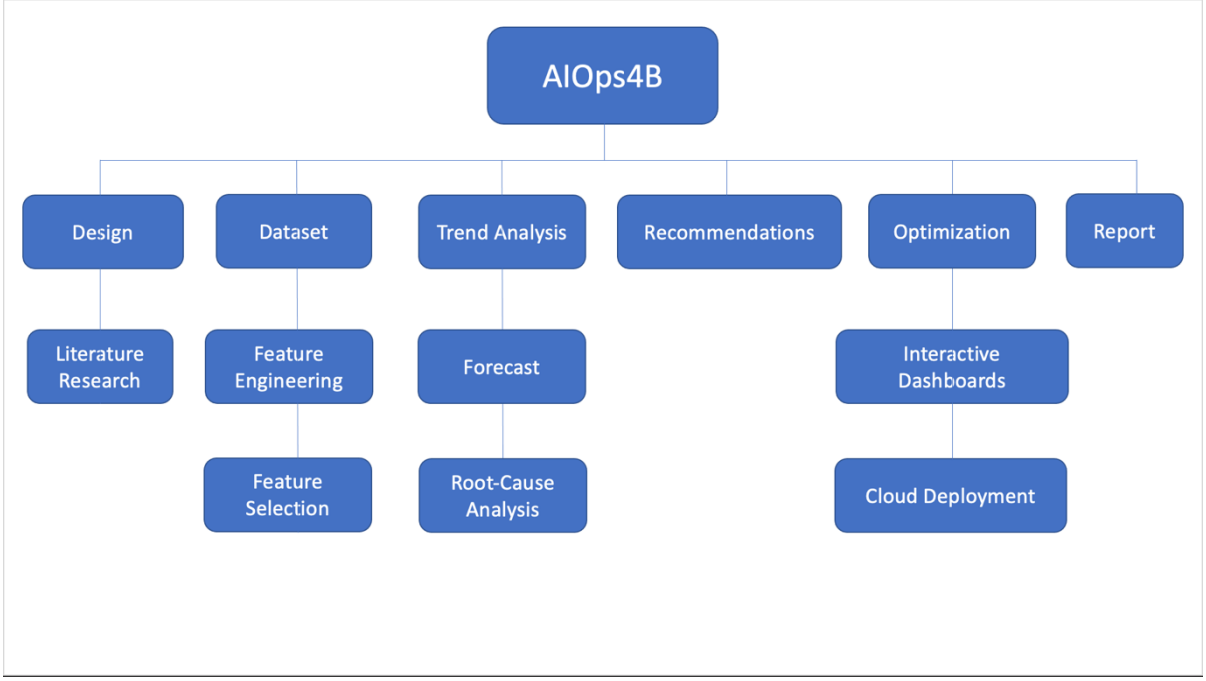

Figure 2 - Graphical Work Plan

Figure 3 shows a Gantt chart that indicates the project timeline.

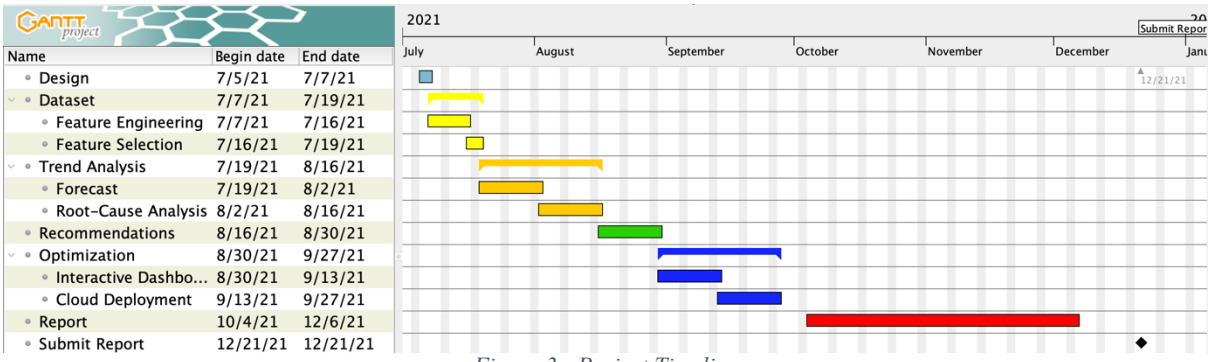

Figure 3 - Project Timeline

References

- Hosseini, R. (2021) 'Greykite: A flexible, intuitive, and fast forecasting library' Available at: https://engineering.linkedin.com/blog/2021/greykite--a-flexible--intuitive--and-fast-forecasting-library (Accessed: 6 July 2021).

- Hosseini, R., Yang, K., Chen, A., and Patra, S. (2021) 'A flexiblee forecasting model for production systems'.

- Kaggle (2018) 'Brazilian E-Commerce Public Dataset by Olist' Available at: https://www.kaggle.com/olistbr/brazilian-ecommerce/home?select=olist_products_dataset.csv (Accessed: 8 July 2021).

- Notaro, P., Cardoso, J., and Gerndt, M. (2020) 'A Systematic Mapping Study in AIOps' *ICSOC 2020 Workshops. ICSOC 2020. Lecture Notes in Computer Science*, vol 12632.

- Outlier (2021) Available at: https://outlier.ai/ (Accessed: 5 July 2021).

- Petersen, M. (2018) 'Outlier Introduces AI Powered Automated Business Analysis Platform' Available at: https://outlier.ai/2018/10/10/outlier-introduces-ai-powered-automated-business-analysis-platform/ (Accessed: 5 July 2021).

- Taylor, S. J. and Letham B. (2018) 'Forecasting at Sale' *The American Statistician*, 72:1, 37-45.

- ThoughtSpot (2021) Available at: https://www.thoughtspot.com/product (Accessed: 5 July 2021).

- Žunić, M., Korjenić, K., Hodžić, K., and Đonko, D. (2020) 'Application of Facebook's Prophet Algorithm for Successful Sales Forecasting Based on Real-World Data' *International Journal of Computer Science & Information Technology (IJCSIT)* Vol 12, No 2.

Research Ethics Review Form: BSc, MSc and MA Projects

Computer Science Research Ethics Committee (CSREC)

http://www.city.ac.uk/department-computer-science/research-ethics

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

PART A: Ethics Checklist. All students must complete this part.
The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be ***provisional*** – *identifying the planned research as* likely to involve MINIMAL RISK. In such cases you must additionally seek ***full approval*** from the supervisor as the project progresses and details are established. ***Full approval*** must be acquired in writing, before beginning the planned research.

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
1.1	Does your research require approval from the National Research Ethics Service (NRES)? *e.g. because you are recruiting current NHS patients or staff?* *If you are unsure try - https://www.hra.nhs.uk/approvals-amendments/what-approvals-do-i-need/*	**NO**
1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act? *Such research needs to be approved by an external ethics committee such as NRES or the Social Care Research Ethics Committee - http://www.scie.org.uk/research/ethics-committee/*	**NO**
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation? *Such research needs to be authorised by the ethics approval system of the National Offender Management Service.*	**NO**
A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		*Delete as appropriate*
2.1	Does your research involve participants who are unable to give informed consent? *For example, but not limited to, people who may have a degree of learning disability or mental health problem, that means they are unable to make an informed decision on their own behalf.*	**NO**
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	**NO**

2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	**NO**
2.4	Does your project involve participants disclosing information about special category or sensitive subjects? *For example, but not limited to: racial or ethnic origin; political opinions; religious beliefs; trade union membership; physical or mental health; sexual life; criminal offences and proceedings*	**NO**
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study? *Please check the latest guidance from the FCO - http://www.fco.gov.uk/en/*	**NO**
2.6	Does your research involve invasive or intrusive procedures? *These may include, but are not limited to, electrical stimulation, heat, cold or bruising.*	**NO**
2.7	Does your research involve animals?	**NO**
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	**NO**
A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/ **Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.**		*Delete as appropriate*
3.1	Does your research involve participants who are under the age of 18?	**NO**
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)? *This includes adults with cognitive and / or learning disabilities, adults with physical disabilities and older people.*	**NO**
3.3	Are participants recruited because they are staff or students of City, University of London? *For example, students studying on a particular course or module.* *If yes, then approval is also required from the Head of Department or Programme Director.*	**NO**
3.4	Does your research involve intentional deception of participants?	**NO**
3.5	Does your research involve participants taking part without their informed consent?	**NO**
3.5	Is the risk posed to participants greater than that in normal working life?	**NO**
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	**NO**
A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK. **If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.**		*Delete as appropriate*

If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.		
4	Does your project involve human participants or their identifiable personal data? *For example, as interviewees, respondents to a survey or participants in testing.*	**NO**

Appendix B: *data_pipeline.py* script

The following code refers to the first Python script in the project. This script takes as input the Brazilian E-Commerce datasets and creates the metrics we described in Chapter 3. The output is a csv file containing the metrics called **'metrics_pvalue_ord.csv'.**

```
import pandas as pd

# Raw Data - we import the datasets from the Brazilian E-Commerce Dataset

orders = pd.read_csv(r'olist_orders_dataset.csv')

payments = pd.read_csv(r'olist_order_payments_dataset.csv')
payments['payment_id'] = range(0, 103886)

reviews = pd.read_csv(r'olist_order_reviews_dataset.csv')

items = pd.read_csv(r'olist_order_items_dataset.csv')

customers = pd.read_csv(r'olist_customers_dataset.csv')

geolocation = pd.read_csv(r'olist_geolocation_dataset.csv')

# METRICS - we will now create the different metrics

# Date

date = orders['order_purchase_timestamp']
date = date.sort_values(ascending=True)
timestamp = date
date = date.str.slice(stop=10)
date = date.unique()

# Daily total revenue

df = pd.merge(orders, payments, on='order_id')
df['order_purchase_timestamp'] =
df['order_purchase_timestamp'].str.slice(stop=10)
df = df.sort_values(by='order_purchase_timestamp')
rev = df[['order_purchase_timestamp', 'payment_value']]
revenue = rev.groupby(['order_purchase_timestamp']).sum()
###print(revenue.head())

# STD Total Customers

rev_std = rev.groupby('order_purchase_timestamp').agg(['std'])
###print(rev_std.head(15))

# STD New Customers

df = pd.merge(orders, customers, on='customer_id')
df['new_customers_orders'] = 1
df['returning_customers_orders'] = 1
df['total_customer_orders'] = 1
df['order_purchase_timestamp'] =
df['order_purchase_timestamp'].str.slice(stop=10)
```

```
df = df.sort_values(by='order_purchase_timestamp')
# print(df.head())

df2 = df.duplicated('customer_unique_id', 'first')
# print(df2.tail())
customer_list = list(df2)
new_customers = [i for i, j in enumerate(customer_list) if not j]

new_customers_df = df.loc[df.index[new_customers]]
new_customers_df2 = pd.merge(new_customers_df,payments,on='order_id')
nco = new_customers_df2[['order_purchase_timestamp', 'payment_value']]
rev_new_std = nco.groupby('order_purchase_timestamp').agg(['count','sum'])
###print(rev_new_std.head(15))

## Daily returning customers orders ##

pd.options.mode.chained_assignment = None

returning_customers = [i for i, j in enumerate(customer_list) if j]

returning_customers_df = df.loc[df.index[returning_customers]]
returning_customers_df2 =
pd.merge(returning_customers_df,payments,on='order_id')
rco = returning_customers_df2[['order_purchase_timestamp',
'payment_value']]
rev_ret_std = rco.groupby('order_purchase_timestamp').agg(['count','sum'])
###print(rev_ret_std.head(15))

ratio = pd.merge(rev_ret_std,rev_new_std,on='order_purchase_timestamp')
ratio['returning/new_customers_ratio'] =
ratio['payment_value_x']['count']/ratio['payment_value_y']['count']
ratio = ratio[['order_purchase_timestamp','returning/new_customers_ratio']]
#print(ratio.head())


# Total customer orders

df = pd.merge(orders, customers, on='customer_id')
df['new_customers_orders'] = 1
df['returning_customers_orders'] = 1
df['total_customer_orders'] = 1
df['order_purchase_timestamp'] =
df['order_purchase_timestamp'].str.slice(stop=10)
df = df.sort_values(by='order_purchase_timestamp')
# print(df.head())


# Total daily customer orders

tco = df[['order_purchase_timestamp', 'total_customer_orders']]
total_customer_orders = tco.groupby(['order_purchase_timestamp']).sum()
#print(total_customer_orders.head(10))



## Average/STD daily product popularity ##

pp = pd.merge(orders, reviews, on='order_id')
pp['order_purchase_timestamp'] =
pp['order_purchase_timestamp'].str.slice(stop=10)
pp2 = pp[['order_purchase_timestamp', 'review_score']]
```

```
avg_review_score = pp2.groupby(['order_purchase_timestamp']).agg(['mean'])
###print(product_popularity.head(10))

# Total good/bad reviews #

good = []
bad = []

for i in pp['review_score']:
    if i>=3:
        good.append(1)
    else:
        good.append(0)

for i in pp['review_score']:
    if i<3:
        bad.append(1)
    else:
        bad.append(0)

pp['good_review'] = good
pp['bad_review'] = bad

pp3 = pp[['order_purchase_timestamp', 'good_review','bad_review']]
good_bad_reviews = pp3.groupby(['order_purchase_timestamp']).agg(['sum'])
##print(good_bad_reviews.head(10))


## Average/STD delivery value ##

afv = pd.merge(orders, items, on='order_id')
afv['order_purchase_timestamp'] =
afv['order_purchase_timestamp'].str.slice(stop=10)
afv = afv[['order_purchase_timestamp', 'freight_value']]
afv_total = afv.groupby(['order_purchase_timestamp']).agg(['mean'])
###print(afv_total.head(10))

## Average/STD delivery delay ## we are not going to use this

delivery = orders.copy()
delivery['order_purchase_timestamp'] =
delivery['order_purchase_timestamp'].str.slice(stop=10)
delivery['order_delivered_customer_date'] =
delivery['order_delivered_customer_date'].str.slice(stop=10)
delivery['order_estimated_delivery_date'] =
delivery['order_estimated_delivery_date'].str.slice(stop=10)

delivery['order_delivered_customer_date'] =
pd.to_datetime(delivery['order_delivered_customer_date'])
delivery['order_estimated_delivery_date'] =
pd.to_datetime(delivery['order_estimated_delivery_date'])

delivery['delivery_delay'] = delivery['order_delivered_customer_date'] -
delivery['order_estimated_delivery_date']

delay = delivery[['order_purchase_timestamp', 'delivery_delay']]
delay = delay.sort_values(by='order_purchase_timestamp')
delay = delay.dropna()
#print(delay.head(15))

score = []
```

```
# The score is 1 if there is no delay and -1 is there is any delay

for i in delay['delivery_delay']:
    if i<=pd.Timedelta(0):
        score.append(1)
    else:
        score.append(-1)

delay['delivery_score'] = score

delivery_score = delay[['order_purchase_timestamp', 'delivery_score']]
delivery_score =
delivery_score.groupby(['order_purchase_timestamp']).agg(['mean'])



### METRICS DATAFRAMES ###

dataframes = [revenue, rev_new_std, rev_ret_std, avg_review_score,
good_bad_reviews, ratio,
              afv_total, delivery_score, total_customer_orders]

from functools import reduce

metrics = reduce(lambda left, right: pd.merge(left, right,
on=['order_purchase_timestamp'],
                                              how='outer'), dataframes)

metrics.reset_index(inplace=True)
#print(metrics.columns)

col_names =
['Time_col','Value','Total_New_Customers','Total_New_Customers_Value',
             'Total_Returning_Customers',
             'Total_Returning_Customers_Value', 'AVG_Review_Score',
             'Total_Good_Reviews','Total_Bad_Reviews','Ratio',
             'AVG_Freight_Value',
             'AVG_Delivery_Score','total_customer_orders']

metrics.columns = col_names
#metrics = metrics.iloc[14:-1,:]
metrics.reset_index(inplace=True)
metrics = metrics.drop(['index'],axis=1)


### FEATURE IMPORTANCE ### we use OLS to find the most relevant metrics

from sklearn.linear_model import LinearRegression

X = metrics[[
             'Total_New_Customers',
             'Total_Returning_Customers',
             'AVG_Review_Score',
             'Total_Good_Reviews',
             'Total_Bad_Reviews',
             'Ratio',
             'AVG_Freight_Value',
             'AVG_Delivery_Score'
             ]]
```

```
X = X.iloc[:601,:]

y = metrics[['Value']]
y = y.iloc[:601,:]

X = X.fillna(0)

import numpy as np
from scipy import stats
import statsmodels.api as sm

est = sm.OLS(y, X)
est2 = est.fit()
print(est2.summary())


## Final metrics dataset ##

from collections import defaultdict
import warnings

warnings.filterwarnings("ignore")

metrics_pvalue =
metrics[['Time_col','Value','Total_Good_Reviews','Total_Bad_Reviews','AVG_R
eview_Score','AVG_Freight_Value','AVG_Delivery_Score','total_customer_order
s']]

metrics_pvalue.to_csv('metrics_pvalue_ord.csv')
```

Appendix C: *application.py* script

This script contains the main Python application for the project.

```
# Import libraries

import pandas as pd

from greykite.algo.forecast.silverkite.constants.silverkite_holiday import
SilverkiteHoliday
from greykite.framework.templates.autogen.forecast_config import
ForecastConfig, MetadataParam, ModelComponentsParam, \
    EvaluationPeriodParam
from greykite.framework.templates.forecaster import Forecaster
from greykite.framework.templates.model_templates import ModelTemplateEnum
from greykite.common import constants as cst

from plotly import graph_objs as go

import warnings
warnings.filterwarnings("ignore")

from ml_utils import *

import nltk
nltk.download('stopwords')
nltk.download('rslp')

from nltk.corpus import stopwords
import re
from nltk.stem import RSLPStemmer
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer

import statsmodels.api as sm

# We read the metrics file and the reviews dataset

with open(f'metrics_pvalue_ord.csv', 'rb') as f:
    data = f.read()

metrics = pd.read_csv(f'metrics_pvalue_ord.csv',index_col=[0])
#metrics = metrics.iloc[:609,:]

with open(f'olist_order_reviews_dataset.csv', 'rb') as f:
    data = f.read()

reviews = pd.read_csv(f'olist_order_reviews_dataset.csv',index_col=[0])

# These are the parameters that have been explained in Chapter 3

anomaly_start = "2017-11-23"
anomaly_end = "2017-11-26"
country = "Brazil"
prediction_days = 5



# 1. Forecasting model
```

```
# 1. Anomalies
anomaly_df = pd.DataFrame({
    # start and end are included
    cst.START_DATE_COL: [anomaly_start],
    cst.END_DATE_COL: [anomaly_end],
    cst.ADJUSTMENT_DELTA_COL: [np.nan]
})

anomaly_info = {
    "value_col": "Value",
    "anomaly_df": anomaly_df,
    "adjustment_delta_col": cst.ADJUSTMENT_DELTA_COL
}

# 2. Growth

growth = {
    "growth_term": "linear"
}

# 3. Changepoint detection

changepoints = {
    "changepoints_dict": dict(
        method="auto",
        yearly_seasonality_order=2,
        regularization_strength=0.6,
        resample_freq="7D",
        potential_changepoint_n=25,
        yearly_seasonality_change_freq="365D",
        no_changepoint_proportion_from_end=0.1
    )
}

# 4. Seasonality

yearly_seasonality_order = 2
weekly_seasonality_order = 4
seasonality = {
    "yearly_seasonality": yearly_seasonality_order,
    "quarterly_seasonality": False,
    "monthly_seasonality": False,
    "weekly_seasonality": weekly_seasonality_order,
    "daily_seasonality": False
}

# 5. Holidays and events

events = {
    "holidays_to_model_separately":
SilverkiteHoliday.ALL_HOLIDAYS_IN_COUNTRIES,
    "holiday_lookup_countries": [country],  # only look up holidays in
Brazil
    "holiday_pre_num_days": 2,  # also mark the 2 days before a holiday as
holiday
    "holiday_post_num_days": 2,  # also mark the 2 days after a holiday as
holiday
}

# Complete model
```

```
metadata = MetadataParam(
    time_col="Time_col",
    value_col="Value",
    freq="D",
    anomaly_info=anomaly_info,
)

model_components = ModelComponentsParam(
    seasonality=seasonality,
    growth=growth,
    events=events,
    changepoints=changepoints,
)

evaluation_period = EvaluationPeriodParam(
    test_horizon=28,
    cv_horizon=prediction_days,
    cv_max_splits=3,
    cv_min_train_periods=300
)

# running the forecast
forecaster = Forecaster()

result = forecaster.run_forecast_config(
    df=metrics,
    config=ForecastConfig(
        model_template=ModelTemplateEnum.SILVERKITE.name,
        forecast_horizon=prediction_days,
        coverage=0.95,
        metadata_param=metadata,
        model_components_param=model_components,
        evaluation_period_param=evaluation_period
    )
)

# fig2 is the forecasting graph, which has not been included in the app

forecast = result.forecast
fig2 = forecast.plot()
fig2 = go.Figure(fig2)

from greykite.framework.utils.result_summary import
summarize_grid_search_results

# this applies cross validation and outputs the test MAPE

cv_results = summarize_grid_search_results(
    grid_search=result.grid_search,
    decimals=1,
    cv_report_metrics=None,
    column_order=["rank", "mean_test", "split_test", "params"])
cv_results["params"] = cv_results["params"].astype(str)
cv_results.set_index("params", drop=True, inplace=True)
print(cv_results.transpose())

# forecast_df contains the revenue forecast dataframe, from which we will
display 5 elements

forecast_df = forecast.df
```

```
df = pd.DataFrame(forecast_df.iloc[((-prediction_days) - 1):-1,
2]).reset_index(drop=True)
df.index += 1


# 2.1 Recommendation - analyse. This is the changepoint detection model

# this is the dataset metadata
metadata = dict(
    time_col="Time_col",
    value_col="Value",
    freq="D"  # our frequency is daily
)
# these are the changepoint parameters
model_components = dict(
    changepoints={
        "changepoints_dict": {
            "method": "auto",
            "yearly_seasonality_order": 15,
            "regularization_strength": 0.6,
            "resample_freq": "7D",
            "potential_changepoint_n": 25,
            "no_changepoint_proportion_from_end": 0.05
        }
    },
    custom={
        "fit_algorithm_dict": {
            "fit_algorithm": "ridge"}})  # we use ridge to prevent
overfitting

config = ForecastConfig.from_dict(
    dict(
        model_template=ModelTemplateEnum.SILVERKITE.name,
        forecast_horizon=5,
        coverage=0.95,
        metadata_param=metadata,
        model_components_param=model_components))

# the forecaster is run with the changepoints configuration
forecaster = Forecaster()
result = forecaster.run_forecast_config(
    df=metrics,
    config=config)

import plotly

plotly.io.renderers.default = 'browser'

# this fig is the changepoint graph that we can see in AIOps4B

fig = result.model[-1].plot_trend_changepoint_detection(dict(plot=False))
#print(fig.show())

fig = go.Figure(fig)


# 2.2 Root-Cause Analysis and Deep Analysis

#reviews = pd.read_csv(reviews_path)
```

```
## START - Adapted from https://www.kaggle.com/thiagopanini/e-commerce-
sentiment-analysis-eda-viz-nlp ##

## 1. REVIEW COMMENTS DATASET

reviews = reviews.sort_values(by='review_answer_timestamp')
df_comments = reviews.loc[:, ['review_score', 'review_comment_message']]
df_comments = df_comments.iloc[99332:, :]
df_comments = df_comments.dropna(subset=['review_comment_message'])
df_comments = df_comments.reset_index(drop=True)
df_comments.columns = ['score', 'comment']

# print(df_comments.head())

## 2. REGULAR EXPRESSIONS

# we remove line breaks

def re_breakline(text_list):
    return [re.sub('[\n\r]', ' ', r) for r in text_list]

reviews = list(df_comments['comment'].values)

reviews_breakline = re_breakline(reviews)
df_comments['re_breakline'] = reviews_breakline

## 3. HYPERLINKS

# we replace hyperlinks with "link"

def re_hyperlinks(text_list):
    pattern = 'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-
fA-F][0-9a-fA-F]))+'
    return [re.sub(pattern, ' link ', r) for r in text_list]

reviews_hyperlinks = re_hyperlinks(reviews_breakline)
df_comments['re_hyperlinks'] = reviews_hyperlinks

## 4. DATES

# we replace dates with "data"

def re_dates(text_list):
    pattern = '([0-2][0-9]|(3)[0-1])(\/|\.)(((0)[0-9])|((1)[0-
2]))(\/|\.)\d{2,4}'
    return [re.sub(pattern, ' data ', r) for r in text_list]

reviews_dates = re_dates(reviews_hyperlinks)
df_comments['re_dates'] = reviews_dates

## 5. MONEY

# we replace currency with "dinheiro", the Portuguese word for money

def re_money(text_list):
    pattern = '[R]{0,1}\$[ ]{0,}\d+(,|\.)\d+'
    return [re.sub(pattern, ' dinheiro ', r) for r in text_list]

reviews_money = re_money(reviews_dates)
df_comments['re_money'] = reviews_money
```

```
## 6. NUMBERS

# we replace numbers with "numero", the Portuguese word for number

def re_numbers(text_list):
    return [re.sub('[0-9]+', ' numero ', r) for r in text_list]

reviews_numbers = re_numbers(reviews_money)
df_comments['re_numbers'] = reviews_numbers

## 7. NEGATION

# we replace negation or negative expressions with "negação", the
Portuguese word for negation

def re_negation(text_list):
    return [re.sub('([nN][ãÃaA][oO]|[ñÑ]| [nN] )', ' negação ', r) for r in
text_list]

reviews_negation = re_negation(reviews_numbers)
df_comments['re_negation'] = reviews_negation

## 8. SPECIAL CHARACTERS

# we remove special characters

def re_special_chars(text_list):
    return [re.sub('\W', ' ', r) for r in text_list]

reviews_special_chars = re_special_chars(reviews_negation)
df_comments['re_special_chars'] = reviews_special_chars

## 9. WHITESPACES

# we remove whitespaces

def re_whitespaces(text_list):
    white_spaces = [re.sub('\s+', ' ', r) for r in text_list]
    white_spaces_end = [re.sub('[ \t]+$', '', r) for r in white_spaces]
    return white_spaces_end

reviews_whitespaces = re_whitespaces(reviews_special_chars)
df_comments['re_whitespaces'] = reviews_whitespaces

## 10. STOPWORDS

# we remove Portuguese stopwords

pt_stopwords = stopwords.words('portuguese')

def stopwords_removal(text,
cached_stopwords=stopwords.words('portuguese')):
    return [c.lower() for c in text.split() if c.lower() not in
cached_stopwords]

reviews_stopwords = [' '.join(stopwords_removal(review)) for review in
reviews_whitespaces]
df_comments['stopwords_removed'] = reviews_stopwords

## 11. STEMMING
```

```
def stemming_process(text, stemmer=RSLPStemmer()):
    return [stemmer.stem(c) for c in text.split()]

reviews_stemmer = [' '.join(stemming_process(review)) for review in
reviews_stopwords]
df_comments['stemming'] = reviews_stemmer

# Feature Extraction

# we define a function for feature extraction

def extract_features_from_corpus(corpus, vectorizer, df=False):
    corpus_features = vectorizer.fit_transform(corpus).toarray()
    features_names = vectorizer.get_feature_names()
    df_corpus_features = None
    if df:
        df_corpus_features = pd.DataFrame(corpus_features,
columns=features_names)
    return corpus_features, df_corpus_features

## 1. CountVectorizer

count_vectorizer = CountVectorizer(max_features=300, min_df=7, max_df=0.8,
stop_words=pt_stopwords)
countv_features, df_countv_features =
extract_features_from_corpus(reviews_stemmer, count_vectorizer, df=True)

## 2. TF-IDF

tfidf_vectorizer = TfidfVectorizer(max_features=300, min_df=7, max_df=0.8,
stop_words=pt_stopwords)
tfidf_features, df_tfidf_features =
extract_features_from_corpus(reviews_stemmer, tfidf_vectorizer, df=True)

# scores of 1-2 will be considered negative and scores of 3-5 will be
considered positive

score_map = {
    1: 'negative',
    2: 'negative',
    3: 'positive',
    4: 'positive',
    5: 'positive'
}
df_comments['sentiment_label'] = df_comments['score'].map(score_map)

def ngrams_count(corpus, ngram_range, n=-1,
cached_stopwords=stopwords.words('portuguese')):
    vectorizer = CountVectorizer(stop_words=cached_stopwords,
ngram_range=ngram_range).fit(corpus)
    bag_of_words = vectorizer.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in
vectorizer.vocabulary_.items()]
    words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
    total_list = words_freq[:n]
    count_df = pd.DataFrame(total_list, columns=['ngram', 'count'])
    return count_df

positive_comments = df_comments.query('sentiment_label ==
"positive"')['stemming']
```

```
negative_comments = df_comments.query('sentiment_label ==
"negative"')['stemming']

# extracting the top 10 unigrams by sentiment
unigrams_pos = ngrams_count(positive_comments, (1, 1), 10)
unigrams_neg = ngrams_count(negative_comments, (1, 1), 10)

# extracting the top 10 bigrams by sentiment
bigrams_pos = ngrams_count(positive_comments, (2, 2), 10)
bigrams_neg = ngrams_count(negative_comments, (2, 2), 10)

# extracting the top 10 trigrams by sentiment
trigrams_pos = ngrams_count(positive_comments, (3, 3), 10)
trigrams_neg = ngrams_count(negative_comments, (3, 3), 10)

## END - adapted from https://www.kaggle.com/thiagopanini/e-commerce-
sentiment-analysis-eda-viz-nlp ##

# here we select the dates from the previous changepoint to the next
changepoint. Like we explained in Chapter 3,
# the metrics' change will be measured in these periods of time

# from 2017-01-23 to Changepoint 2

d21 = '2017-01-23'
d22 = '2017-07-23'

# from Changepoint 2 to Changepoint 3

d31 = '2017-07-23'
d32 = '2017-11-26'

# from Changepoint 3 to Changepoint 4

d41 = '2017-11-26'
d42 = '2018-07-22'

# these parameters are the calculations for the root-cause analysis

# these are the metrics for 2017-01-23

review1 = 3.69230769230769
freight1 = 18.0077777777777
delivery1 = 0.888888888888888
revenue1 = 7553.89
orders1 = 39

# these are the metrics for the period between 2017-01-23 and Changepoint 2

review2 = pd.DataFrame(metrics.iloc[32:213,4])
freight2 = pd.DataFrame(metrics.iloc[32:213,5])
delivery2 = pd.DataFrame(metrics.iloc[32:213,6])

revenue2 = pd.DataFrame(metrics.iloc[32:213,1])
orders2 = pd.DataFrame(metrics.iloc[32:213,7])

# these are the metrics for the period between Changepoint 2 and
Changepoint 3

review3 = pd.DataFrame(metrics.iloc[213:339,4])
freight3 = pd.DataFrame(metrics.iloc[213:339,5])
```

```
delivery3 = pd.DataFrame(metrics.iloc[213:339,6])

revenue3 = pd.DataFrame(metrics.iloc[213:339,1])
orders3 = pd.DataFrame(metrics.iloc[213:339,7])

# these are the metrics for the period between Changepoint 3 and
Changepoint 4

review4 = pd.DataFrame(metrics.iloc[339:577,4])
freight4 = pd.DataFrame(metrics.iloc[339:577,5])
delivery4 = pd.DataFrame(metrics.iloc[339:577,6])

revenue4 = pd.DataFrame(metrics.iloc[339:577,1])
orders4 = pd.DataFrame(metrics.iloc[339:577,7])

# these are the metrics for the period between Changepoint 4 to the end

review0 = pd.DataFrame(metrics.iloc[577:,4])
freight0 = pd.DataFrame(metrics.iloc[577:,5])
delivery0 = pd.DataFrame(metrics.iloc[577:,6])

revenue0 = pd.DataFrame(metrics.iloc[577:,1])
orders0 = pd.DataFrame(metrics.iloc[577:,7])

# these parameters are the metrics between certain periods of time that
will be used in the linear correlation,
# since only 3 data points is not enough

# the metrics for 01-03-17

review5 = metrics.iloc[69,4]
freight5 = metrics.iloc[69,5]
delivery5 = metrics.iloc[69,6]

revenue5 = metrics.iloc[69,1]

# the metrics from 02-03-17 to 15-04-17

review6 = pd.DataFrame(metrics.iloc[70:115,4])
freight6 = pd.DataFrame(metrics.iloc[70:115,5])
delivery6 = pd.DataFrame(metrics.iloc[70:115,6])

revenue6 = pd.DataFrame(metrics.iloc[70:115,1])

# the metrics for 01-03-18

review7 = metrics.iloc[434,4]
freight7 = metrics.iloc[434,5]
delivery7 = metrics.iloc[434,6]

revenue7 = metrics.iloc[434,1]

# the metrics from 02-03-18 to 15-04-18

review8 = pd.DataFrame(metrics.iloc[435:480,4])
freight8 = pd.DataFrame(metrics.iloc[435:480,5])
delivery8 = pd.DataFrame(metrics.iloc[435:480,6])

revenue8 = pd.DataFrame(metrics.iloc[435:480,1])

###
```

```
# now we calculate the change in metrics for Changepoint 2, for the period
before and after it

review21 = round((review3.mean()[0]-
review2.mean()[0])/review2.mean()[0]*100,2)
freight21 = round((freight3.mean()[0]-
freight2.mean()[0])/freight2.mean()[0]*100,2)
delivery21 = round((delivery3.mean()[0]-
delivery2.mean()[0])/delivery2.mean()[0]*100,2)
revenue21 = round((revenue3.mean()[0]-
revenue2.mean()[0])/revenue2.mean()[0]*100,2)
orders21 = round((orders3.mean()[0]-
orders2.mean()[0])/orders2.mean()[0]*100,2)

# now we calculate the change in metrics for Changepoint 3, for the period
before and after it

review32 = round((review4.mean()[0]-
review3.mean()[0])/review3.mean()[0]*100,2)
freight32 = round((freight4.mean()[0]-
freight3.mean()[0])/freight3.mean()[0]*100,2)
delivery32 = round((delivery4.mean()[0]-
delivery3.mean()[0])/delivery3.mean()[0]*100,2)
revenue32 = round((revenue4.mean()[0]-
revenue3.mean()[0])/revenue3.mean()[0]*100,2)
orders32 = round((orders4.mean()[0]-
orders3.mean()[0])/orders3.mean()[0]*100,2)

# now we calculate the change in metrics for Changepoint 4, for the period
before and after it

review43 = round((review0.mean()[0]-
review4.mean()[0])/review4.mean()[0]*100,2)
freight43 = round((freight0.mean()[0]-
freight4.mean()[0])/freight4.mean()[0]*100,2)
delivery43 = round((delivery0.mean()[0]-
delivery4.mean()[0])/delivery4.mean()[0]*100,2)
revenue43 = round((revenue0.mean()[0]-
revenue4.mean()[0])/revenue4.mean()[0]*100,2)
orders43 = round((orders0.mean()[0]-
orders4.mean()[0])/orders4.mean()[0]*100,2)

# these calculations are for the other periods, for the linear regression

review65 = round((review6.mean()[0]-review5)/review5*100,2)
freight65 = round((freight6.mean()[0]-freight5)/freight5*100,2)
delivery65 = round((delivery6.mean()[0]-delivery5)/delivery5*100,2)
revenue65 = round((revenue6.mean()[0]-revenue5)/revenue5*100,2)

review87 = round((review8.mean()[0]-review7)/review7*100,2)
freight87 = round((freight8.mean()[0]-freight7)/freight7*100,2)
delivery87 = round((delivery8.mean()[0]-delivery7)/delivery7*100,2)
revenue87 = round((revenue8.mean()[0]-revenue7)/revenue7*100,2)

# we can now build a dataframe with the metrics changes

df2 =
pd.DataFrame({'review':[review21,review32,review43,review65,review87],

'freight':[freight21,freight32,freight43,freight65,freight87],
```

```
'delivery':[delivery21,delivery32,delivery43,delivery65,delivery87],

'revenue':[revenue21,revenue32,revenue43,revenue65,revenue87]})

X = df2[['review','freight','delivery']]
Y = df2['revenue']

# we apply a linear regression

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

# these parameters are the linear correlation coefficients that will be
present in the dashboard

params = model.params


# 3. Visualization

import dash_table

# Inspired by https://medium.com/analytics-vidhya/building-a-dashboard-app-
using-plotlys-dash-a-complete-guide-from-beginner-to-pro-61e890bdc423

def create_data_table(dataframe):
    data_table = dash_table.DataTable(
        id = 'table',
        data = dataframe.to_dict(),
        columns = [{'id':c,'name':c} for c in dataframe.columns],
        style_table = {'overflowY':'scroll'},
        style_cell = {'width':'100px'}
    )


def generate_table(dataframe, max_rows=10):
    return html.Table([
        html.Tbody([
            html.Tr([
                html.Td(round(dataframe.iloc[i][col])) for col in
dataframe.columns
            ]) for i in range(min(len(dataframe), max_rows))
        ])
    ])

import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import dash_bootstrap_components as dbc

app = dash.Dash(external_stylesheets=[dbc.themes.BOOTSTRAP])
application = app.server

app.layout = html.Div([
    html.Div(
        className="header",
        children=[
            html.Div(
                className="div-info",
```

```
                children=[
                    html.H1(className="title", children="AIOps4B -
Automated Forecasting, Root-Cause Analysis, and Deep Analysis"),
                ],
            ),
        ],
    ),

    html.H2(["Revenue KPI"]),
    html.P(
        """
        Graphical representation of the historical sales, with their trend
and the detected trend changepoints.
        """
    ),

    html.Div(
        [
            dbc.Row(
                [
                    dbc.Col(dcc.Graph(id="graph", figure=fig)),
                    dbc.Col([html.H2(str(prediction_days) + '-Day
Forecast'),
                             html.Div(generate_table(df))],width=2)
                ]
            ),
        ]
    ),

    dbc.Row(
        [
            dbc.Col([
                html.Label('Select Changepoint:'),
                dcc.Dropdown(
                    id="dropdown-component",
                    options=[
                        {'label': '1.- 27 November 2016', 'value': 'cp1'},
                        {'label': '2.- 23 July 2017', 'value': 'cp2'},
                        {'label': '3.- 26 November 2017', 'value': 'cp3'},
                        {'label': '4.- 22 July 2018', 'value': 'cp4'}
                    ],
                    value='cp4'
                ),

                html.P(
                    className="root-cause", id="root-cause", children=[""]
                )
            ]),
            dbc.Col(
                html.P(
                    className="recommendations", id="recommendations",
children=[""]
                )
            )
        ]
    ),
])

# Callbacks #

# Inspired by https://github.com/plotly/dash-sample-
```

apps/blob/main/apps/dash-daq-satellite-dashboard/app.py

```
@app.callback(
    Output("root-cause", "children"),
    [Input("dropdown-component", "value")],
)
def update_root_cause(val):
    if val == "cp1":
        text = (
            "-"
        )
    elif val == "cp2":
        text = (html.P(["The trend is UP.",html.Br(),
                        "The change in average REVENUE for Changepoint 2 is
of {}".format(revenue21), " %.",
                        html.Br(),
                        "The change in ORDERS for Changepoint 2 is of
{}".format(orders21), " %.",
                        ]),
                html.H2("Root-Cause Analysis"),
                html.H3("Impact of Metrics on the Trend:"),
                html.P(["Average Review Score:
{}%".format(round(params[0]*review21, 2)),html.Br(),
                        "Average Freight Value:
{}%".format(round(params[1]*freight21, 2)),html.Br(),
                        "Average Delivery Score:
{}%".format(round(params[2]*delivery21, 2)),html.Br(),
                        html.Br(),
                        "There is a {}% increase in revenue due to the
metrics.".format(round(round(params[0]*review21, 2) +
round(params[1]*freight21, 2) + round(params[2]*delivery21, 2), 2))
                        ]),
                html.H3("Impact of Events on the Trend:"),
                html.P(["There is a {}% increase in revenue due to
events.".format(round(revenue21-(round(params[0]*review21, 2) +
round(params[1]*freight21, 2) + round(params[2]*delivery21,2)))),
html.Br(),
                        "Event: Sales Increase."])
                )

    elif val == "cp3":
        text = (html.P(["The trend is UP.",html.Br(),
                        "The change in average EVENUE for Changepoint 3 is
of {}".format(revenue32), " %.",
                        html.Br(),
                        "The change in ORDERS for Changepoint 3 is of
{}".format(orders32), " %.",
                        ]),
                html.H2("Root-Cause Analysis"),
                html.H3("Impact of Metrics on the Trend:"),
                html.P(["Average Review Score:
{}%".format(round(params[0]*review32, 2)),html.Br(),
                        "Average Freight Value:
{}%".format(round(params[1]*freight32, 2)),html.Br(),
                        "Average Delivery Score:
{}%".format(round(params[2]*delivery32, 2)),html.Br(),
                        html.Br(),
                        "There is a {}% increase in revenue due to the
metrics.".format(round(round(params[0]*review32, 2) +
round(params[1]*freight32, 2) + round(params[2]*delivery32, 2), 2))
                        ]),
```

```
                html.H3("Impact of Events on the Trend:"),
                html.P(
                    ["There is a {}% increase in revenue due to
events.".format(round(revenue32-(round(params[0]*review32, 2) +
round(params[1]*freight32, 2) + round(params[2]*delivery32,2)))),
html.Br(),
                     "Event: Black Friday."])
                )

    elif val == "cp4":
        text = (html.P(["The trend is DOWN.",html.Br(),
                        "The change in average REVENUE for Changepoint 4 is
of {}".format(revenue43), " %.",
                        html.Br(),
                        "The change in ORDERS for Changepoint 4 is of
{}".format(orders43), " %.",
                        ]),
                html.H2("Root-Cause Analysis"),
                html.H3("Impact of Metrics on the Trend:"),
                html.P(["Average Review Score:
{}%".format(round(params[0]*review43, 2)),html.Br(),
                        "Average Freight Value:
{}%".format(round(params[1]*freight43, 2)),html.Br(),
                        "Average Delivery Score:
{}%".format(round(params[2]*delivery43, 2)),html.Br(),
                        html.Br(),
                        "There is a {}% decrease in revenue due to the
metrics.".format(round(round(params[0]*review43, 2) +
round(params[1]*freight43, 2) + round(params[2]*delivery43, 2), 2))
                        ]),
                html.H3("Impact of Events on the Trend:"),
                html.P(
                    ["There is a {}% increase in revenue due to unknown
events.".format(round(revenue43-(round(params[0]*review43, 2) +
round(params[1]*freight43, 2) + round(params[2]*delivery43,2)))),
html.Br(),
                     "Event: Sales Decrease."])
                )
    return text

@app.callback(
    Output("recommendations", "children"),
    [Input("dropdown-component", "value")],
)
def update_recommendations(val):

    if val == "cp1":
        text = ("")

    elif val == "cp2":
        text = (html.P(["The correlation coefficient of the average review
score is {}.".format(round(params[0],2)),
                        html.Br(),
                        "The correlation coefficient of the average freight
value is {}.".format(round(params[1],2)),
                        html.Br(),
                        "The correlation coefficient of the average
delivery score is {}.".format(round(params[2],2))]))


    elif val == "cp3":
```

```
        text = (html.P(["The correlation coefficient of the average review
score is {}.".format(round(params[0],2)),
                        html.Br(),
                        "The correlation coefficient of the average freight
value is {}.".format(round(params[1],2)),
                        html.Br(),
                        "The correlation coefficient of the average
delivery score is {}.".format(round(params[2],2))]))


    elif val == "cp4":
        text = (html.P(["The correlation coefficient of the average review
score is {}.".format(round(params[0],2)),
                        html.Br(),
                        "The correlation coefficient of the average freight
value is {}.".format(round(params[1],2)),
                        html.Br(),
                        "The correlation coefficient of the average
delivery score is {}.".format(round(params[2],2))]),
                html.H2(["Deep Analysis"]),
                html.H3(['Product Reviews (most common problems):']),

                html.P(["{} customers did not receive their
product.".format(trigrams_neg.loc[trigrams_neg['ngram'] == 'neg receb
produt', 'count'].iloc[0]),
                        html.Br(),
                        "{} customers had a problem with the tracking
code.".format(trigrams_neg.loc[trigrams_neg['ngram'] == 'produt códig vem',
'count'].iloc[0]),
                        html.Br(),
                        "{} customers received a different product than the
one they ordered.".format(trigrams_neg.loc[trigrams_neg['ngram'] == 'outr
total difer', 'count'].iloc[0])]),

                html.H3(["Delivery Score (most common problems):"]),
                html.P(["Orders to the North and Northeastern regions have
a longer delivery time."]),

                html.H3(["Delivery Cost (most common problems):"]),
                html.P(["Orders to the North and Northeastern regions have
a higher delivery cost."])
                )

    return text

# now that we have defined the Dash app, we run it in port 80

if __name__ == '__main__':
    application.run(debug=True, host='0.0.0.0', port='80')
```

Appendix D: *deep_analysis_geolocation.py* script

This script contains the geolocation part of the Deep Analysis. Although no maps or figures that have resulted from this script have been included in the AIOps4B app, the basis for the analysis is in this script.

```
# Adapted from https://www.kaggle.com/hoonkeng/eda-understand-brazil-e-commerce-geographically

# We start by importing the required libraries

import numpy as np
import pandas as pd
import warnings
warnings.simplefilter("ignore")
import matplotlib.pyplot as plt
import plotly
import plotly.offline as offline
import plotly.graph_objs as go
from plotly.offline import plot
from plotly.graph_objs import Scatter, Figure, Layout
from plotly import tools

# We import the datasets that we will work with, which are the geolocation, payments, order items, orders, and customers

geo = pd.read_csv(r'olist_geolocation_dataset.csv',
                  dtype={'geolocation_zip_code_prefix': str})

df_payment = pd.read_csv(r'olist_order_payments_dataset.csv')
df_items = pd.read_csv(r'olist_order_items_dataset.csv')
df_orders = pd.read_csv(r'olist_orders_dataset.csv')
df_customers = pd.read_csv(r'olist_customers_dataset.csv')

# We can merge the first two on order_id and the customers one on customer_id

df = pd.merge(df_payment,
              df_items[['order_id','price','freight_value']],
              on='order_id')

df = pd.merge(df,

df_orders[['order_id','customer_id','order_purchase_timestamp','order_approved_at',

'order_delivered_customer_date','order_estimated_delivery_date']],
              on='order_id')

df = pd.merge(df,

df_customers[['customer_id','customer_state','customer_city','customer_zip_code_prefix']],
              on='customer_id')

#print(df.head())

# We can plot the orders on the map of Brazil
```

```
data = [go.Scattermapbox(
    lon = geo['geolocation_lng'],
    lat = geo['geolocation_lat'],
    marker = dict(
        size = 5,
        color = 'green',
    ))]

layout = dict(
    title = 'Olist Orders',
    mapbox = dict(
        accesstoken =
'pk.eyJ1IjoiaG9vbmtlbmc5MyIsImEiOiJjam43cGhpNng2ZmpxM3JxY3Z4ODl2NWo3In0.SGR
vJlToMtgRxw9ZWzPFrA',
        center= dict(lat=-22,lon=-43),
        bearing=10,
        pitch=0,
        zoom=2,
    )
)
fig = dict( data=data, layout=layout )
#plot(fig, validate=False)

# Now we can complete the dataframe with the correct geolocation data

df['customer_state'] = df['customer_state'].apply(lambda x : x.lower())
df['customer_city'] = df['customer_city'].apply(lambda x : x.lower())

geo_state =
geo.groupby('geolocation_state')['geolocation_lat','geolocation_lng'].mean(
).reset_index()
geo_state['geolocation_state'] =
geo_state['geolocation_state'].apply(lambda x : x.lower())

geo_city =
geo.groupby('geolocation_city')['geolocation_lat','geolocation_lng'].mean()
.reset_index()
geo_city['geolocation_city'] = geo_city['geolocation_city'].apply(lambda x
: x.lower())

geo_city.rename(columns={'geolocation_lat':
'c_lat','geolocation_lng':'c_lng'}, inplace=True)

df = pd.merge(df, geo_state, how='left',
left_on='customer_state',right_on='geolocation_state')
df = pd.merge(df,
geo_city,how='left',left_on='customer_city',right_on='geolocation_city')

#print(df.head())

# The following map shows the orders with the cities they were made from

data = [go.Scattermapbox(
    lon = geo_city['c_lng'],
    lat = geo_city['c_lat'],
    text = geo_city['geolocation_city'],
    marker = dict(
        size = 2,
        color = 'Green',
    ))]
```

```
layout = dict(
    title = 'Olist Orders Cities',
    mapbox = dict(
        accesstoken =
'pk.eyJ1IjoiaG9vbmtlbmc5MyIsImEiOiJjam43cGhpNng2ZmpxM3JxY3Z4ODl2NWo3In0.SGR
vJlToMtgRxw9ZWzPFrA',
        center= dict(lat=-22,lon=-43),
        bearing=10,
        pitch=0,
        zoom=2,
    )
)
fig = dict(data=data,layout=layout)
#plot(fig,validate=False)

# Now we can start with the analysis for the freight value

city_spend =
df.groupby(['customer_city','c_lng','c_lat'])['price'].sum().to_frame().res
et_index()
city_freight =
df.groupby(['customer_city','c_lng','c_lat'])['freight_value'].mean().reset
_index()

state_spend =
df.groupby(['customer_state','c_lng','c_lat'])['price'].sum().to_frame().re
set_index()
state_freight =
df.groupby(['customer_state','c_lng','c_lat'])['freight_value'].mean().rese
t_index()
state_freight['text'] = 'state :' + state_freight['customer_state'] + ' |
Freight: ' + state_freight['freight_value'].astype(str)

data = [go.Scattergeo(
    lon = state_spend['c_lng'],
    lat = state_spend['c_lat'],
    text = state_freight['text'],
    marker = dict(
        size = state_spend['price']/3000,
        sizemin = 5,
        color= state_freight['freight_value'],
        colorscale= 'Reds',
        cmin = 20,
        cmax = 50,
        line = dict(width=0.1, color='rgb(40,40,40)'),
        sizemode = 'area'
    ),
    name = 'State'),
    go.Scattergeo(
        lon = city_spend['c_lng'],
        lat = city_spend['c_lat'],
        text = city_freight['freight_value'],
        marker = dict(
            size = city_spend['price']/1000,
            sizemin = 2,
            color= city_freight['freight_value'],
            colorscale= 'Blues',
            reversescale=True,
            cmin = 0,
            cmax = 80,
            line = dict(width=0.1, color='rgb(40,40,40)'),
```

```
            sizemode = 'area'
        ),
        name = 'City')]

layout = dict(
    title = 'Olist Orders Freight Value',
    showlegend = True,
    autosize=True,
    width = 900,
    height = 600,
    geo = dict(
        scope = "south america",
        projection = dict(type='winkel tripel', scale = 1.6),
        center = dict(lon=-47,lat=-22),
        showland = True,
        showcountries= True,
        showsubunits=True,
        landcolor = 'rgb(155, 155, 155)',
        subunitwidth=1,
        countrywidth=1,
        subunitcolor="rgb(255, 255, 255)",
        countrycolor="rgb(255, 255, 255)"
    )
)

fig = dict(data=data,layout=layout)
plot(fig,validate=False)

# The North and North-Eastern regions have a higher freight value

# Now we can see the analysis for the delivery time

df['order_purchase_timestamp'] =
pd.to_datetime(df['order_purchase_timestamp'])
df['order_approved_at'] = pd.to_datetime(df['order_approved_at'])

df['order_estimated_delivery_date'] =
pd.to_datetime(df['order_estimated_delivery_date'])
df['order_delivered_customer_date'] =
pd.to_datetime(df['order_delivered_customer_date'])

df['delay'] = (df['order_delivered_customer_date'] -
df['order_estimated_delivery_date']).dt.total_seconds() / (3600 * 24)
df['deliver'] = (df['order_delivered_customer_date'] -
df['order_approved_at']).dt.total_seconds() / (3600 * 24)

df['delay'] = df['delay'].fillna(0)
df['deliver'] = df['deliver'].fillna(0)

city_deliver =
df.groupby(['customer_city','c_lng','c_lat'])['deliver'].mean().reset_index
()
city_delay =
df.groupby(['customer_city','c_lng','c_lat'])['delay'].mean().reset_index()

state_deliver =
df.groupby(['customer_state','c_lng','c_lat'])['deliver'].mean().reset_inde
x()
state_delay =
df.groupby(['customer_state','c_lng','c_lat'])['delay'].mean().reset_index(
)
```

```
state_deliver['text'] = 'Deliver duration: ' +
state_deliver['deliver'].astype(str) + '| Delay:' +
state_delay['delay'].astype(str)
city_deliver['text'] = 'Deliver duration: ' +
city_deliver['deliver'].astype(str) + '| Delay:' +
city_delay['delay'].astype(str)

data = [go.Scattergeo(
    lon = state_deliver['c_lng'],
    lat = state_deliver['c_lat'],
    text = state_deliver['text'],
    marker = dict(
        size = state_deliver['deliver']*20,
        sizemin = 1,
        color= state_delay['delay'],
        colorscale= 'Reds',
        cmin = -30,
        cmax = 0,
        line = dict(width=0.1, color='rgb(40,40,40)'),
        sizemode = 'area'
    ),
    name = 'state'),
    go.Scattergeo(
        lon = city_deliver['c_lng'],
        lat = city_deliver['c_lat'],
        text = city_deliver['text'],
        marker = dict(
            size = (city_deliver['deliver']+3),
            sizemin = 2,
            color= city_delay['delay'],
            colorscale= 'Blues',
            reversescale=True,
            cmin = -50,
            cmax = 50,
            line = dict(width=0.1, color='rgb(40,40,40)'),
            sizemode = 'area'
        ),
        name = 'city')]

layout = dict(
    title = 'Brazilian E-commerce Delivery and Delay (Click legend to
toggle traces)',
    showlegend = True,
    autosize=True,
    width = 900,
    height = 600,
    geo = dict(
        scope = "south america",
        projection = dict(type='winkel tripel', scale = 1.6),
        center = dict(lon=-47,lat=-22),
        showland = True,
        showcountries= True,
        showsubunits=True,
        landcolor = 'rgb(155, 155, 155)',
        subunitwidth=1,
        countrywidth=1,
        subunitcolor="rgb(255, 255, 255)",
        countrycolor="rgb(255, 255, 255)"
    )
)
```

```
fig = dict(data=data,layout=layout)
#plot(fig,validate=False) #this is now commented so that the first plot is
shown

# The North and North-Eastern regions also have higher delivery delay.
```

Appendix E: *Dockerfile* code

This is the code from the *Dockerfile*, used to create a Docker image of the *application.py* script for the cloud deployment of AIOps4B.

```
FROM python:3.7

RUN mkdir /app
WORKDIR /app
ADD requirements.txt /app/
RUN pip install -r requirements.txt
ADD . /app/

ENTRYPOINT [ "python" ]

CMD ["application.py"]
```

Appendix F: *requirements.txt* file

This file contains the dependencies that I installed in my project environment. Some libraries may not be needed for the application to work, but this is the configuration I used for building the Docker image and this worked.

```
aiops4b-NTTDataUK==0.6
aiops4b-pkg-julgonza==0.0.1
argon2-cffi==20.1.0
async-generator==1.10
atomicwrites==1.4.0
attrs==21.2.0
backcall==0.2.0
bleach==4.0.0
bokeh==2.3.3
Bottleneck==1.3.2
Brotli==1.0.9
brotlipy==0.7.0
build==0.6.0.post1
certifi==2021.5.30
cffi==1.14.6
cftime==1.5.0
chardet==4.0.0
charset-normalizer==2.0.3
click==7.1.2
click-plugins==1.1.1
cligj==0.7.2
cloudpickle==1.6.0
cmarkgfm==0.4.2
colorama==0.4.4
colorcet==2.0.6
convertdate==2.3.2
cryptography==3.4.7
cvxpy==1.1.13
cycler==0.10.0
Cython==0.29.24
cytoolz==0.11.0
dash==1.19.0
dash-bootstrap-components==0.13.0
dash-core-components==1.15.0
dash-html-components==1.1.2
dash-renderer==1.9.0
dash-table==4.11.2
dask==2021.8.0
datashader==0.13.0
decorator==5.0.9
defusedxml==0.7.1
dill==0.3.4
distributed==2021.8.0
docutils==0.17.1
ecos==2.0.7.post1
entrypoints==0.3
ephem==4.0.0.2
Fiona==1.8.13.post1
Flask==1.1.2
Flask-Compress==1.10.1
```

```
fonttools==4.25.0
fsspec==2021.7.0
future==0.18.2
geopandas==0.9.0
greykite==0.2.0
HeapDict==1.0.1
hijri-converter==2.1.3
holidays==0.11.2
holidays-ext==0.0.7
holoviews==1.14.5
idna==3.2
importlib-metadata==4.6.1
iniconfig==1.1.1
ipykernel==5.3.4
ipython==7.22.0
ipython-genutils==0.2.0
itsdangerous==2.0.1
jedi==0.17.0
Jinja2==3.0.1
joblib==1.0.1
jsonschema==3.2.0
jupyter-client==6.1.12
jupyter-core==4.7.1
jupyterlab-pygments==0.1.2
kaleido==0.2.1
keyring==23.0.1
kiwisolver==1.3.1
korean-lunar-calendar==0.2.1
lightgbm==3.1.1
llvmlite==0.36.0
locket==0.2.1
LunarCalendar-ext==0.0.1
mapclassify==2.4.3
Markdown==3.3.4
MarkupSafe==2.0.1
matplotlib==3.4.2
mistune==0.8.4
mkl-fft==1.3.0
mkl-random==1.2.2
mkl-service==2.4.0
msgpack==1.0.2
multipledispatch==0.6.0
munch==2.5.0
munkres==1.1.4
nbclient==0.5.3
nbconvert==6.1.0
nbformat==5.1.3
nest-asyncio==1.5.1
netCDF4==1.5.7
networkx==2.6.2
nltk==3.6.2
notebook==6.4.3
numba==0.53.1
numexpr==2.7.3
numpy
olefile==0.46
osqp==0.6.1
overrides==6.1.0
packaging==21.0
pandas==1.3.1
pandocfilters==1.4.3
```

```
panel==0.12.1
param==1.11.1
parso==0.8.2
partd==1.2.0
patsy==0.5.1
pep517==0.11.0
pickleshare==0.7.5
Pillow==8.3.1
pip==21.1.3
pkginfo==1.7.1
plotly==3.10.0
pluggy==0.13.1
pmdarima==1.8.2
prometheus-client==0.11.0
prompt-toolkit==3.0.17
psutil==5.8.0
py==1.10.0
pycparser==2.20
pyct==0.4.8
Pygments==2.9.0
PyMeeus==0.5.11
pyOpenSSL==20.0.1
pyparsing==2.4.7
pyproj==2.6.1.post1
pyrsistent==0.18.0
pyshp==2.1.3
PySocks==1.7.1
pytest==6.2.4
pytest-runner==5.3.1
python-dateutil==2.8.2
pytz==2021.1
pyviz-comms==2.0.2
pywin32-ctypes==0.2.0
pywinpty==0.5.7
PyYAML==5.4.1
pyzmq==22.2.1
readme-renderer==24.0
regex==2021.4.4
requests==2.26.0
requests-toolbelt==0.9.1
retrying==1.3.3
rfc3986==1.4.0
Rtree==0.9.7
scikit-learn==0.24.2
scipy==1.7.0
scs==2.1.4
seaborn==0.11.1
Send2Trash==1.5.0
setuptools==52.0.0
Shapely==1.7.1
six==1.16.0
sortedcontainers==2.4.0
statsmodels==0.12.2
tblib==1.7.0
tenacity==8.0.1
terminado==0.9.4
testfixtures==6.18.0
testpath==0.5.0
threadpoolctl==2.2.0
toml==0.10.2
tomli==1.2.1
```

```
toolz==0.11.1
tornado==6.1
tqdm==4.62.0
traitlets==5.0.5
twine==3.4.1
typing-extensions==3.10.0.0
typing-utils==0.1.0
urllib3==1.26.6
wcwidth==0.2.5
webencodings==0.5.1
Werkzeug==1.0.1
wheel==0.36.2
win-inet-pton==1.1.0
wincertstore==0.2
xarray==0.19.0
zict==2.0.0
zipp==3.5.0
```