Université de Strasbourg

Master CSMI

Developing Innovative Metrics for Applaudo Studios' Data Lakehouse

Applaudo Studios

Author:
Javier CLADELLAS

Internship supervisor:
Jairo URBINA
Course lecturer:
Cristophe PRUD'HOMME

August 20, 2023

Contents

List of Figures

I Introduction

I.1 Applaudo Studios

During my internship, I had the opportunity to work at Applaudo Studios as a Data Engineer Intern. Applaudo Studios is a technology company that provides a wide range of services, including digital transformation, web and mobile development, cloud computing, AI and machine learning solutions, and more. Its goal is to help brands streamline their IT solutions, optimize delivery costs, and speed up their digital transformation.

The company has around a thousand employees working remotely on more than 200 different projects, from over 30 countries around the world. While the company's headquarters are located in El Salvador, most of the employees are based in south america and the United States.

Applaudo was founded in 2013 by José Luis Giammattei and Darwin Romero. Its CEO is Cesar Bendeck since 2019. The company has been exponentially growing since its foundation, starting with 15 employees and reaching over 1000 in 2023.

Applaudo's purpose is to transform business through code. Its mission is to accelerate growth by developing and executing digital transformation solutions, while its vision is to lead admired brands to success through its solutions. The company has four core values:

- Empowering Excellence
- Effective Communication
- Collaborative Teamwork
- Unsolicited Respect

Applaudo Studio's culture focuses on the well-being of its employees, providing them with a functional training program, personal professional coaches, and guidance to help them grow and develop their skills and careers. The perks and benefits of working at Applaudo can be seen in the figure 1.

Figure 1: Applaudo Studios' perks and benefits.

As the company provides their services to multiple industries (including sports, media, entertainment, retail, banking financial services, insurance and fintech), some of the most notable clients are Walmart, Miami Heat, LifeMiles and Holiday Inn.

I.2 The Data Engineering Team

As a Data Engineer Intern, I was part of the Data Engineering team responsible for the internal project of developing and maintaining the company's data lakehouse.

The team for this project is composed of 6 members, including myself, and is led by Jairo Urbina, the Data Lead. Each one of the members is responsible for a data source, and they are in charge of the data ingestion, transformation and storage of their respective source.

Most of Applaudo's teams work using the agile methodology, which has the main characteristic of being iterative and incremental. As it is known, the agile methodology is based on the idea of dividing the project into smaller parts, called sprints, and each one of them is planned, executed and reviewed in a short period of time.

In the case of this project, the sprints are one week long. They were planned on Monday, executed during the week and the status was reviewed on the same meeting the following Monday. The work of each team member was briefly presented on a daily basis, on a meeting called "Daily Standup", where each member had to present the difficulties encountered the day before, as well as the progress made and the results achieved.

I.3 Context

Most companies accumulate vast amounts of data from different sources, for example, sales, marketing, customer service, human resources, etc. Currently in Applaudo, this information is stored on separated databases, which makes it difficult to access and analyze it. As there is a constant need to improve and accelerate the company's internal processes, and to improve the productivity of multiple departments, having fragmented data can be troubling. Also, data is currently being manually downloaded and processed from all data sources, which takes a considerable amount of effort and time.

In order to overcome these problems, Applaudo has decided to implement a Data Lakehouse structure, which will not only allow the company to centralize its information, but will also automate the data ingestion and transformation processes and provide an easier way to analyze and take profit of the data.

For a Data Lakehouse to function properly, multiple points need to be considered. First, the data lakehouse should be scalable, allowing incremental growth of the data quantity. Then, it must be assured that the data is clean, standard and consistent, compliant with the Data Quality standards. Also, as the Google Cloud Platform will be used for this effect, the lakehouse should be cost-efficient. This can be done by performing cost analysis and optimizing the data storage and processing. Finally, security and privacy measures must be taken to ensure that only authorized users can access the data.

I.4 Objectives

Each member of the project's team is responsible for extracting, transforming and loading the data from a specific source. In my case, I was given the task of proposing,

designing and developing multiple metrics from all the data sources, that will be useful for the company's internal processes and decision-making. As the data lakehouse is still in development, the internship's main objective is to develop and facilitate the future implementation of the metrics. The tasks to be performed during the internship can be summarized as follows:

- Propose informative metrics that can benefit the company's internal processes, such as the hiring processes, the employee's performance and the project's status.
- Develop the metrics by proposing a pre-processing and transformation pipeline, and code the necessary scripts to generate figures and aggregated information.
- Propose a way to integrate the metrics into the data lakehouse, using Google Cloud Platform, as well as the necessary scripts to generate the figures on a Power BI dashboard.
- Perform a cost analysis for the needed calculations, aggregations and models for Google Cloud Platform.
- Optimize the code and the data storage to reduce the costs.
- Document the process and the results obtained to be able to replicate the process when the data lakehouse is deployed.

The work will be divided by the metrics, after defining them, working on one of them at a time. For each metric, a description will be written, and possible uses will be defined. Then, all the previous steps will be performed, iterating over the metrics until all of them are developed.

In this report, all the tools, methods and models used will be explained in detail on the section Prerequisites (II), as well as the necessary definitions and concepts needed to contextualize the work. Then, the results for each metric will be presented on the Results section (III), this being proposed preprocessing and transformation pipelines, an explanation of the code, the cost analysis and the results obtained. Finally, the conclusions on the results and on the internship will be presented on the Conclusions section (IV).

II Prerequisites

II.1 The Data Lakehouse Architecture

A data lakehouse is data architecture that is a combination between a data warehouse and a data lake. In order to understand its definition, we must first understand what a data warehouse and a data lake are.

A data warehouse is a centralized repository of structured data. It can be seen as a database that is optimized for data analysis and reporting. A warehouse stores data from multiple sources through an ETL (Extract, Transform, Load) process and is optimized

for performing queries and analysis. It must be noted that a data warehouse only stores structured data, which is information that is organized and well-defined. An example of structured data is any information organized in rows and columns (such as a CSV file or a spreadsheet).

On the other hand, a data lake is a database that stores semi-structured and unstructured data, which is information that does not have a predefined schema and cannot be easily inserted into a traditional database. Some examples of unstructured data are images, audio files and social media posts. The purpose of a data lake is to retain large volumes of data in its original format, without the need to transform or preprocess it.

Contrary to a data warehouse, the final users of a data lake are intended to be data scientists and analysts, whereas the final users of a data warehouse are business professionals. [1], [2].

The main differences between these two concepts can be seen in the following figure (fig. 2).

DATA WAREHOUSE	vs.	DATA LAKE
structured, processed	DATA	structured / semi-structured / unstructured, raw
schema-on-write	PROCESSING	schema-on-read
expensive for large data volumes	STORAGE	designed for low-cost storage
less agile, fixed configuration	AGILITY	highly agile, configure and reconfigure as needed
mature	SECURITY	maturing
business professionals	USERS	data scientists et. al.

Figure 2: Differences between a data warehouse and a data lake. [3]

As previously stated, a data lakehouse combines the features of a data warehouse and a data lake into a single architecture. This means that big amounts of structured, semi-structured and unstructured data from various sources can be stored without considerable preprocessing. However, the lakehouse allows the users to query its information with SQL. Data can be also treated on demand, transforming the raw data in usable information to perform analysis and generate reports.

A data lakehouse diagram proposed by the Amazon Web Services platform can be seen in the following figure (fig. 3).

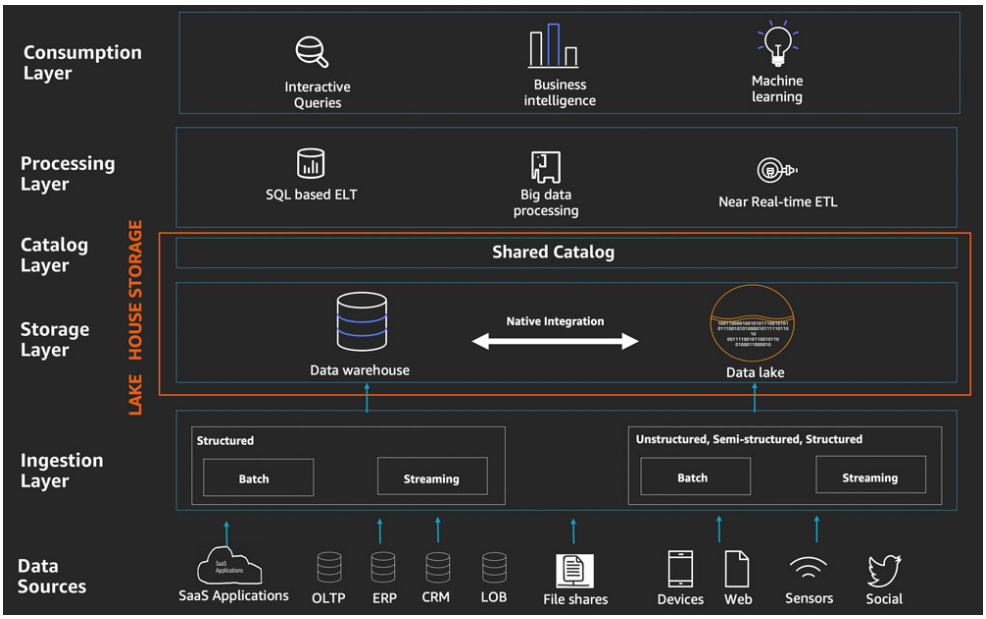


Figure 3: Data lakehouse architecture from AWS. [4]

The advantages of using a data lakehouse are the following, according to AWS [4]:

- Scalable Data Lakes
- Purpose-built Data Services
- Seamless Data Movement
- Unified Governance
- Performant and Cost-effective

II.2 Tools

In this section, the tools used during the internship will be described and their purpose on the project will be explained.

Python Stack

In order to explore, process and visualize the data, as well as to train, evaluate and predict machine learning models in this project, the following stack of Python libraries was used:

- **Pandas**: Used for reading, treating and processing data in a tabular format (DataFrame). It allows to perform operations on tables to pre-process and transform the data. Pandas vectorizes the operations, which optimizes operations such as aggregation, filtering or joining.
- **Statsmodels**: Used to implement the ARIMA models. This library provides the necessary tools to facilitate fitting time series models, perform statistical tests and predict future values.
- **Scipy**: Used particularly for the box-cox method of the **stats** module. The box-cox transformation was used to fit ARIMA models, and will be explained in detail in the ARIMA models section of this report.
- **Scikit-learn**: Used to implement the K-means algorithm and to perform a principal component analysis (PCA) on the data.
- **Tensorflow and Keras**: Used to tune, train, test and store an LSTM neural network (long short-term memory).
- **Plotly**: Used for generating interactive figures during the data exploration phase and for visualizing the results of each metric.

Google Cloud Platform

The Google Cloud Platform (GCP) is a set of cloud computing services offered by Google. It consists of physical assets such as computers, hard disk drives and virtual machines [5]. In order to interact with the GCP services, Google provides the Google Cloud console, which is a web-based interface to manage projects and resources [5]. The figure 4 shows a preview of the Google Cloud console.

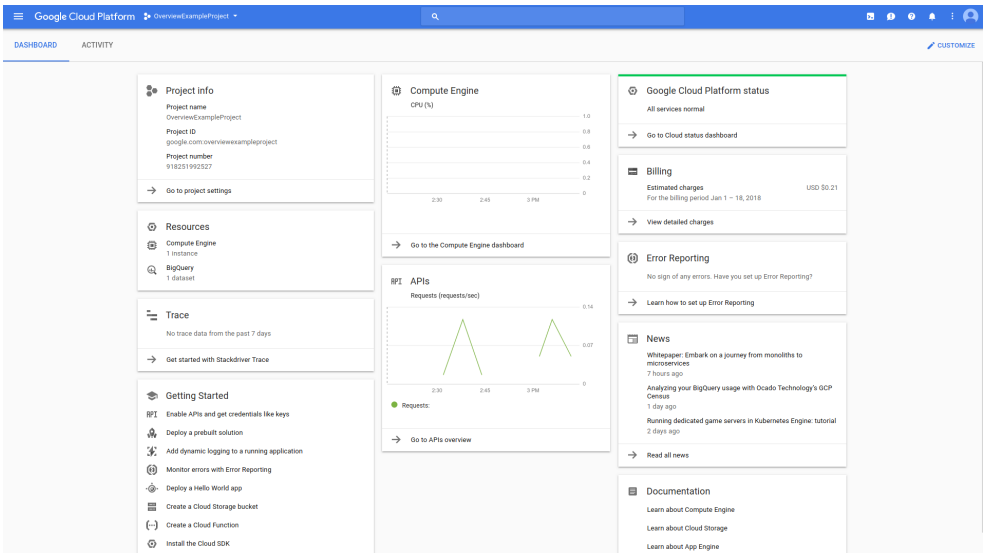


Figure 4: Google Cloud console. [5]

GCP is the default tool used by Applaudo Studios for any cloud computing task, which can be storing and managing data, or training machine learning models.

For the assigned tasks of this internship, two GCP services were used: Google BigQuery and Google Cloud Storage.

Google Cloud storage allows the users to store unstructured and semi-structured information on objects called buckets [7]. In the case of this project, this service was used to store trained machine learning models.

All structured data used in this project was stored in Google BigQuery. The Google BigQuery service provides a storage system for structured data, using a columnar storage format [6]. This data can be queried using SQL or the GoogleSQL dialect.

Additionally, Google BigQuery provides a machine learning service called BigQuery ML, which was employed to load machine learning models and make predictions on the time series.

It must be noted that, as the lakehouse is still in development, a simulation of the data model was used to work on the different metrics, with the objective of reducing the amount of work needed to implement the metrics once the data lakehouse is deployed. This said, some changes will still need to be made to the code at the final stage the project.

Finally, to train some machine learning models, a virtual machine was used to periodically execute python scripts.

Power BI

Power BI is a business analytics and data visualization platform developed by Microsoft [8]. It provides interactive visualizations to create dashboards and reports, which can be shared with other users. The platform allows connecting to multiple sources of data, as well as to transform and model information, using the DAX (Data Analysis Expressions)

language and the M language [8]. However, on the scope of this internship, Power BI was used only to visualize the results of the metrics in a dynamic way by connecting to the Google BigQuery Storage.

The dashboards created in Power BI provide a consolidated view of the visualizations, helping the final users to quickly and easily understand the presented information and facilitate the decision-making process.

As the results of the metrics will be stored on the cloud in a way that minimizes the amount of transformations needed to generate the respective visualizations, the reporting team of Applaudo Studio's will be able to easily generate dashboards using the provided data.

Concerning this internship, Power BI will be used to test and visualize the results, making sure that they can be correctly interpreted by the reporting team using this tool.

II.3 Data Sources

As stated on the introduction, the calculated metrics that are the subject of this internship are based on data coming from multiple platforms. Each tool provides a way to obtain the needed information, which can be by manually downloading a csv file, or through a supplied API.

In this section, the different tools used by Applaudo, that provide valuable information, will be explained in detail. Also, the data present on each source will be described.

Zoho People

Zoho People is a Human Resource Management System that is designed to manage a company's employee information [9]. This platform provides many features, such as employee self-service, leave management, attendance management, HR Process automation and more.

The data downloaded from Zoho People is a csv file where each row represents an employee, and each column represents part of its personal information and work related information, such as their age, title, position, location and others. The file is formed by 80 columns and 1573 rows, and can be seen as Applaudo's roster from July 2014 to May 2023.

Some insights of the data can be seen in the figure 5, visualized using a Power BI dashboard. The report shows the distribution of employees by some categories, such as their gender, department, location and seniority level.

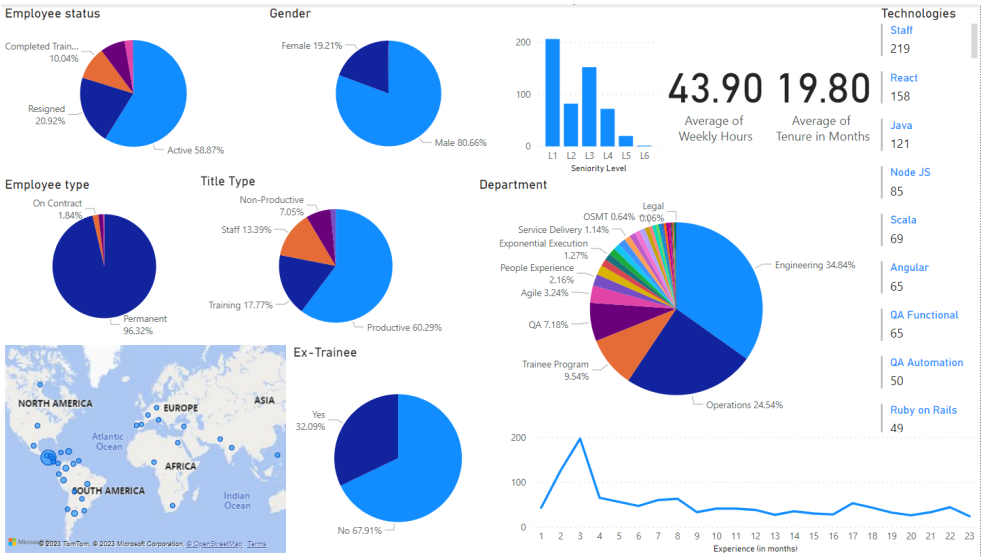


Figure 5: Zoho People data exploration dashboard.

Harvest

Harvest is a time tracking and project management tool for companies that allows them to track the time spent on tasks, projects and clients. It is a useful platform to keep track of billable hours, manage projects and analyze team's productivity [10].

This tool generates downloadable Excel time reports by year, in which each row of the table represents a time entry for a given day, which is entered by each one of the employees. The data can be indexed by the employee's identifier, the date, client and project.

The report includes features such as the number of worked hours, the billable hours and rate, and the cost amount and rate.

Some insights can be seen in the figure 6, visualized using a Power BI dashboard. The dashboard shows three pie charts representing the distribution of hours depending on if they are approved, billable and already invoiced. Additionally, there are two tree maps showing the distribution of the generated income and the distribution of worked hours by client.

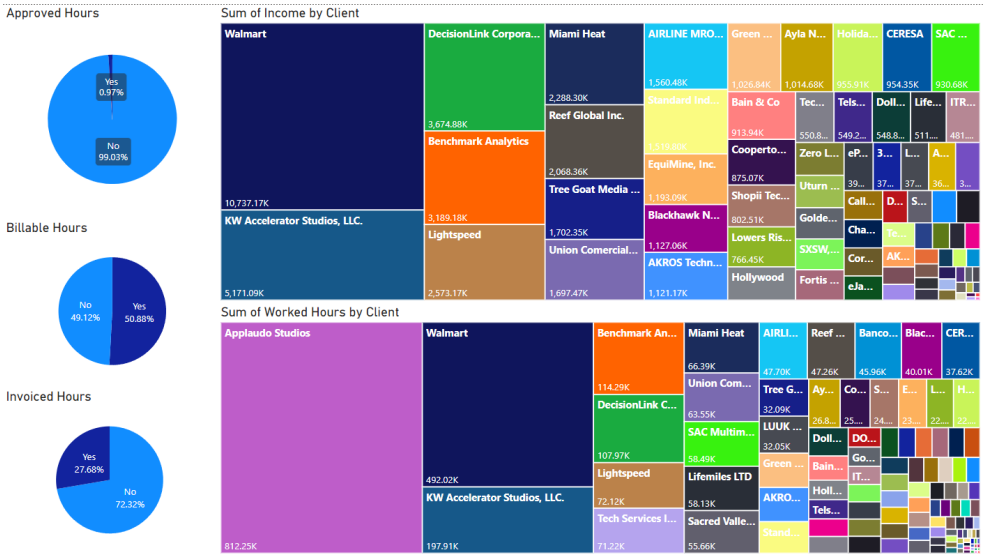


Figure 6: Harvest data exploration dashboard.

Forecast

Harvest also provides a tool to plan and schedule project on a timeline, which is called Forecast. [11] It is useful for assigning time to the team members for specific projects and tasks.

The provided data is a time report by year in Excel format, indexed by the employee's ID, a client and a project. The report's column represent a time period (a date, week or month) containing the planned hours for each entry within the given period. One row can be seen as a timeline for a given employee working on a project. For example, we can have the following row:

Employee	Client	Project	2021-01-01	2021-01-08	...	2023-01-01
EM-1	Walmart	Cotuza	44	16	...	36

Forecast information can be used, combined with Harvest data, to determine a project's progress.

II.4 ARIMA models

As some of the project's metrics include time series forecasting, it is important to understand the different models that can be used to make predictions on time series data. ARIMA stands for "AutoRegressive Integrated Moving Average". It is a class of models used to forecast time series that can be made to be "stationary" by differencing the data [12], and combines an autoregressive and a moving average component. These type of models can capture the trend a seasonality of a time series, and are commonly used in finance and economics.

ARIMA models are defined by three parameters p, d and q, so we can write an ARIMA model as ARIMA(p,d,q). The parameters are defined as follows:

As stated, an ARIMA(p,d,q) model is a combination of three components: an autoregressive component, a moving average component and an integrated component. In order to properly understand the model, we should first define each one of the components. Let's note Y a time series and Y_t the value of the time series at time t.

- **Autoregressive** $AR(p)$ **component**: It is a regression model that is based on the assumption that the current value Y_t can be predicted as a linear combination of the past p values. In an equivalent way,

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t$$

  Where c is a constant, ε_t is the error term and ϕ_i are weights.

- **Moving Average** $MA(q)$ **component**: This component captures the effect of the past forecasting errors, or residuals, on the current value of the time series. It means that the current value Y_t can be predicted as a linear combination of the past q errors. i.e.

$$Y_t = c + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t$$

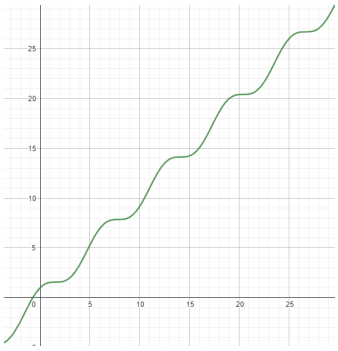
  Where c is a constant term, ε_t is the error term at the time t and θ_i are weights.

- **Integrated** $I(d)$ **component**: As the autoregressive and moving average components assume that their coefficients are constant over time, the time series must be stationary. This is due to the fact that stationarity on a time series implies that its statistical properties (such as mean and variance) do not change over time, which allows the constant components weights to accurately describe the relationships between the values of the time series.

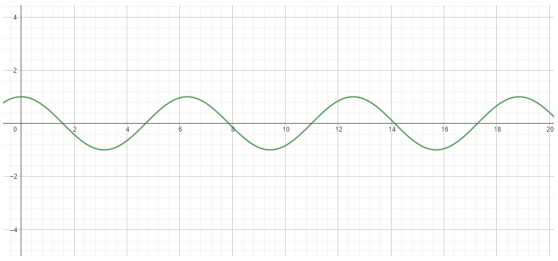
  This component involves differencing the data, with the objective of making the time series stationary.

  The parameter d represents the number of times that the time series needs to be differenced for it to become stationary.

  On the following figure (fig. 7), we can appreciate the difference between a non-stationary and a stationary time series.



(a) Non-stationary time series.



(b) Stationary time series.

Figure 7: Difference between a non-stationary and a stationary time series.

Let Y^d be the d-th difference of the time series Y. Then, using an ARIMA(p,d,q) model, the predicted value of the differenced time series at time t (noted $\hat{Y_t^d}$) can be written as

$$\hat{Y_t^d} = c + \sum_{i=1}^{p} \phi_i Y_{t-i}^d + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t$$

To accurately predict future values in a time series with an ARIMA model, we must determine the appropriate values for the model parameters.

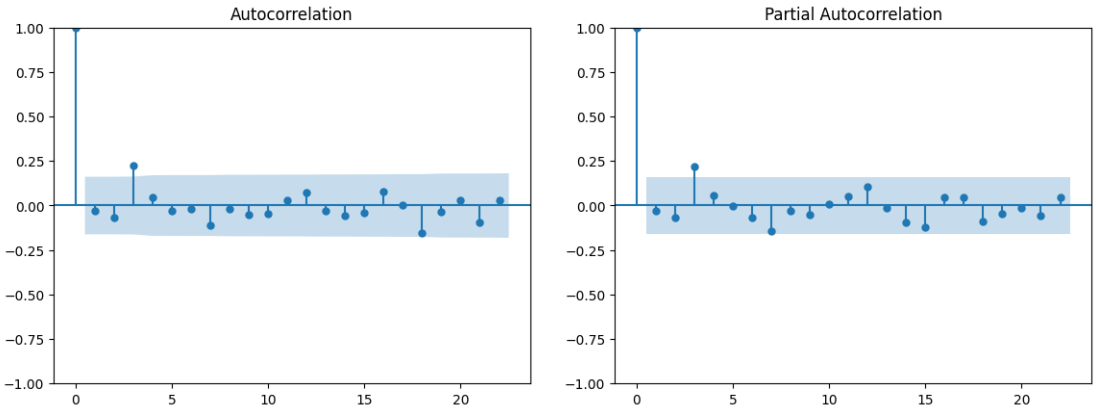
The first step is to find the value of the integrated parameter d that makes the time series stationary. For this, we must start by differencing the time series. We can plot the differenced time series and visually inspect its stationarity. Then, we should perform the Augmented Dickey-Fuller test (ADF), which provides a p-value inferring the likelihood of the time series being stationary. If the p-value is less than a chosen threshold (usually 0.05), we can reject the null hypothesis, which is that the time series is non-stationary. If the differenced data does not seem stationary and/or does not pass the ADF test, we need to difference the data again and iterate the process.[13]

After the d parameter has been set, the autoregressive parameter p and the moving average parameter q can be determined by observing the autocorrelation function (ACF) and the partial autocorrelation function (PACF) of the differenced time series.

The ACF function measures the correlation between the time series and its lagged values. This means that the ACF at lag k is given by $corr(Y_t, Y_{t-k})$. On the other hand, the PACF function measures the correlation uniquely between the time series and its lagged values, without the effects of the intermediate lags. This means that the PACF at lag k is given by $corr(Y_t, Y_{t-k}|Y_{t-1}, Y_{t-2}, ..., Y_{t-k+1})$. [14] A significance limit (confidence interval) is usually plotted on the plots of both functions to help determine the values of p and q.

The autoregressive p parameter is given by the number of significant lags (that are outside the confidence interval) that are before the first non-significant lag on the PACF plot. The same logic is applied for the moving average q parameter, but using the ACF plot instead. [13]

On the following example, we can see the ACF and PACF plots of a time series, with the corresponding significance limits.



(a) ACF plot. (b) PACF plot.

Figure 8: ACF and PACF plots of a time series.

From the plots, we can deduce that $p = 1$ and $q = 1$ because there is only one lag outside the confidence interval on each plot. They could even be conservatively fixed at 0 because the non-significant lags are very close to the significance limit.

To stabilize the variance of the time series, and making the data look like it is normally distributed, it can be useful to normalize it using a Box-Cox transformation. This transformation is given by the following expression:

$$\begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(y) & \text{if } \lambda = 0 \end{cases}$$

With λ being the transformation parameter, which is usually chosen to maximize the log-likelihood function of the transformed data. [15]

Additionally, if the time series present seasonality, the Seasonal ARIMA (SARIMA) model can be used. The SARIMA model introduces a seasonal component to the ARIMA model, while still being able to handle non-stationary data. To do this, a seasonal autoregressive and a seasonal moving average component are added to the model. Also, a seasonal integrated component should also be considered. This implies that the SARIMA model has 7 parameters: p, d, q, P, D, Q and S. Where P, D and Q are the seasonal counterparts of p, d and q, respectively, and S is the seasonal period of the time series (e.g. 7 for weekly data, 30 for monthly, etc.). [16]

We can then define the SARIMA model as:

$$\hat{Y}_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \sum_{i=1}^{P} \Phi_i Y_{t-iS} + \sum_{i=1}^{Q} \Theta_i \varepsilon_{t-iS} + \varepsilon_t$$

It must be noted that exogenous variables can be added to the model, allowing to consider multiple features other than the time series itself.

II.5 The LSTM model

The Long Short Term Memory (LSTM) model is a type of Recurrent Neural Network (RNN) that is able to learn information on long time series. To understand how the LSTM model works, we must start by defining a neural network, then explaining the concept of recurrent neural networks on time series, and finally explaining the LSTM model.

A simple neural network model can be defined as a parametric function (that is usually the composition of multiple intermediate functions). These intermediate functions are called layers. The nodes (neurons) of each layer are interconnected and receive inputs from the previous layer. Each neuron performs a weighted sum of the previous layer's outputs, and applies a non-linear function (called activation function). It can be written as

$$y_j = S(\sum_{i:i\to j} w_{ij}x_i + b_i)$$

[18] Where S is the activation function, w_{ij} is the weight of the connection between the i-th neuron of the previous layer and the j-th neuron of the current layer, x_i is the output of the i-th neuron of the previous layer, b_i is the bias of the i-th neuron of the current layer, and y_j is the output of the j-th neuron of the current layer.

On the following figure (fig 9) we can see the architecture of a neural network.

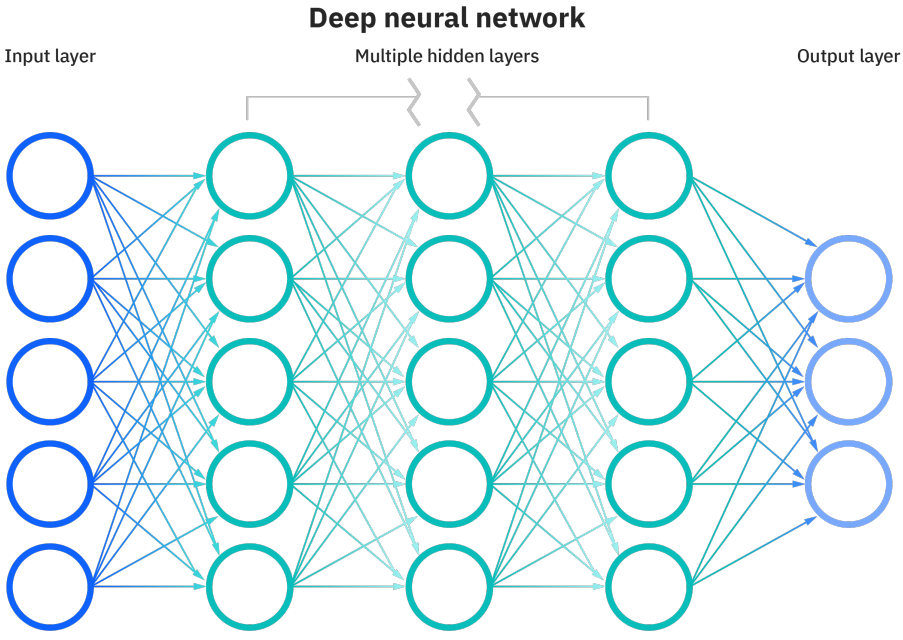


Figure 9: Neural network architecture [19].

The shown network contains an input layer, three hidden layers and an output layer. The input layer will receive the input features of a dataset, and the output layer will produce the final predictions of the model [19].

In order to train the model, a loss function must be defined, and it must be minimized to find the optimal weights of the network (generally using a gradient descent).

A simple RNN model can handle sequential data (such as a time series), by looping back on the previous outputs of each layer. This allows the existence of a hidden state that is used to carry information across time steps [17].

If h_t denotes the output series of the RNN at time t, and x_t denotes the input series at time t, then the RNN can be defined as:

$$h_t = S(x_t U + h_{t-1} W + b)$$

Where U, W, b are matrices of parameters independent of t. [17] We can appreciate this architecture on the following figure (fig 10).

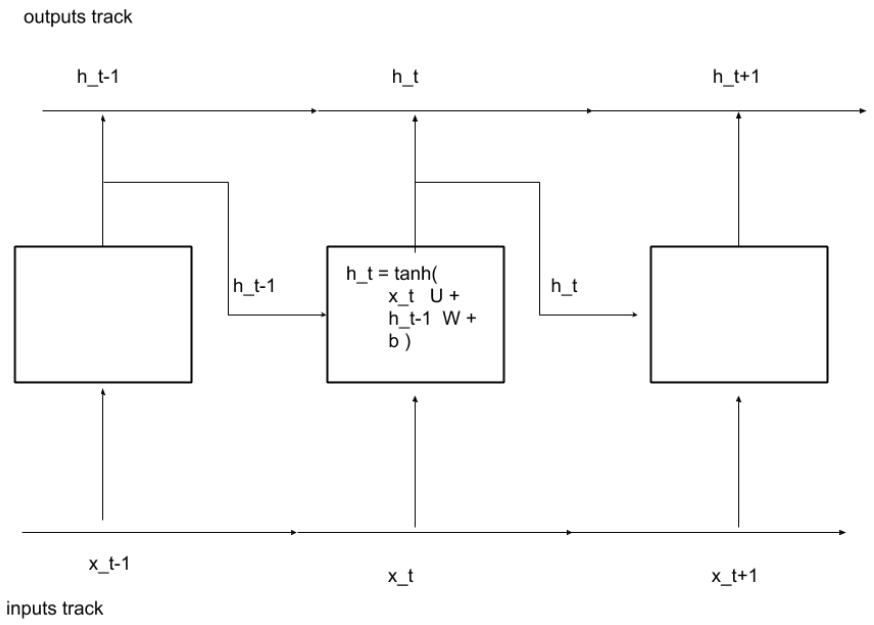
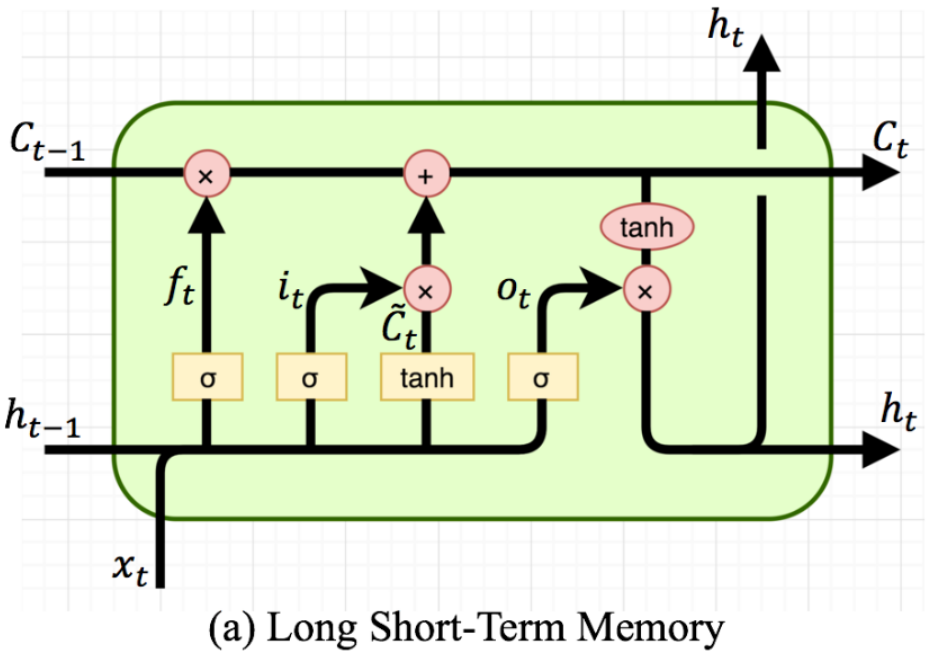


Figure 10: RNN architecture [17].

Finally, the LSTM model is a specific type of RNN that incorporates mechanisms to learn or forget information over time. This is not possible on a simple RNN model, as the information is lost on gradient backpropagation (vanishing gradient problem). [17] This can be done by adding a "carry track" C_t that will carry information across time steps. A schema of an LSTM model can be seen on the following figure (fig 11).



(a) Long Short-Term Memory

Figure 11: LSTM architecture [17].

Let x_t be the input at time t, h_{t-1} the input at the previous time, and C_{t-1} the carry track at the previous time. The new quantities h_t and C_t are computed as follows [17]:

$$
\begin{aligned}
i_t &= \sigma(x_t U^i + h_{t-1} W^i + b^i) \text{ (input gate)} \\
f_t &= \sigma(x_t U^f + h_{t-1} W^f + b^f) \text{ (forget gate)} \\
o_t &= \sigma(x_t U^o + h_{t-1} W^o + b^o) \text{ (output gate)} \\
\tilde{C}_t &= \tanh(x_t U + h_{t-1} W + b) \text{ (candidate carry)} \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \text{ (updated carry)} \\
h_t &= o_t \cdot \tanh(C_t) \text{ (output)}
\end{aligned}
$$

II.6 The K-means algorithm

The k-means algorithm is a clustering technique within the unsupervised learning category.

This method aims to create k different clusters of data points, where each point belongs to the cluster with the nearest mean (called centroid). The grouped data will share similar

characteristics from the features used.

It is an iterative algorithm in which the centroids position are recalculated at each iteration until the distance between the centroids and the data points is optimized.

First, the number of clusters k must be defined. Usually, the optimal number of clusters is found by using the elbow method. This technique consists of calculating the sum of distances within each cluster for different values of k, and analysing the resulting curve to see at which value the curve starts to flatten, generating the shape of an elbow. An example can be seen on the following figure (fig 12).

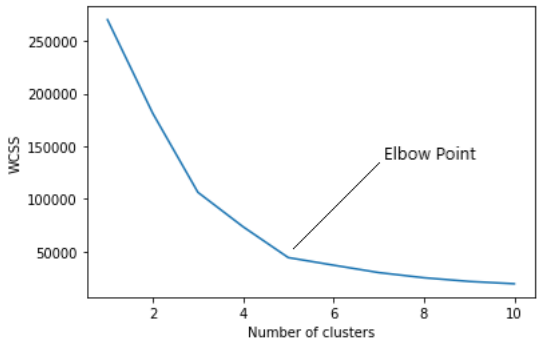


Figure 12: Elbow method example [21].

Once the k is defined, the centroids are randomly initialized. At each iteration, the distance between each point and each centroid is calculated, and the point is assigned to cluster corresponding to the nearest centroid. Then, the centroids are recalculated as the mean of the points in each cluster. When the centroids position stops significantly changing, we can consider that the algorithm has converged.

An example of clustering with the k-means method can be seen on the following figure (fig 13).

Figure 13: K-means clustering example [22].

Additionally, to be able to visualize the results of the clustering on a 2D or 3D space when the number of features is superior to 3, the Principal Component Analysis (PCA) can be used to reduce the dimensionality of the data. The PCA method transforms data

to a lower-dimensional space by projecting the data onto the eigenvectors of the covariance matrix of the data. This allows to keep the maximum amount of information from the original data. [23]

During this internship, this technique was only used for visualization purposes and not for the clustering itself.

III Results

III.1 Headcount Forecasting

The objective of this metric is to predict the future number of employees of Applaudo Studios, by department or technology. The headcount can be extracted from the Zoho People dataset.

Knowing the future headcount of the company can be useful for workforce planning, as it is insightful information for knowing when to hire or train new employees. It can also be used to plan the budget and resources needed in the future.

In order to compute the headcount, the columns representing the start date and the exit date of each employee will be considered.

A **PreProcessing** class was created in Python, which contains a pipeline responsible for reading, and cleaning the data. This class also computes and plots the daily headcount of the company, with the possibility of filtering by department, technology or any categorical column of the dataset.

To compute the headcount, the following steps are performed:

- Calculate the count of employees that started in each day.
- Calculate the count of employees that exited in each day.
- If data is filtered by a given category, these counts are stored in a time series where the columns are the different values of the category.
- Create an empty pandas DataFrame indexed with a **date_range**, with a daily frequency, from the minimum start date to the maximum exit date.
- The previous "join" and "exit" counts are re-indexed to the date range, and null values are filled with 0.
- The daily headcount is then computed as the cumulative sum of the difference between the "join" and "exit" counts.

On the following figures (fig. 14a, fig. 14b), we can see the total daily headcount of the company and the daily headcount separated by department, respectively. It must be noted that, as there is a considerable amount of departments, only the top 5 are shown in the figure.

(a) Total headcount (b) Filtered by department

Figure 14: Daily headcount of the company

The **SARIMAPipeline** class is in charge of fitting the data to a Seasonal ARIMA model to predict the future headcount of the company.

As it can be seen on the previous figures, the headcount function is clearly not stationary, so differencing is required. Additionally, as most new hires start on the first day of the week, the assumption of a weekly seasonality can be made. Thus, the parameter S can be set to 7. The first step to be done is to determine the parameters of the model.

Finding d and D

As the data is not stationary, at least one differencing step is required. The figure 15 shows the daily headcount after applying a first order differencing.

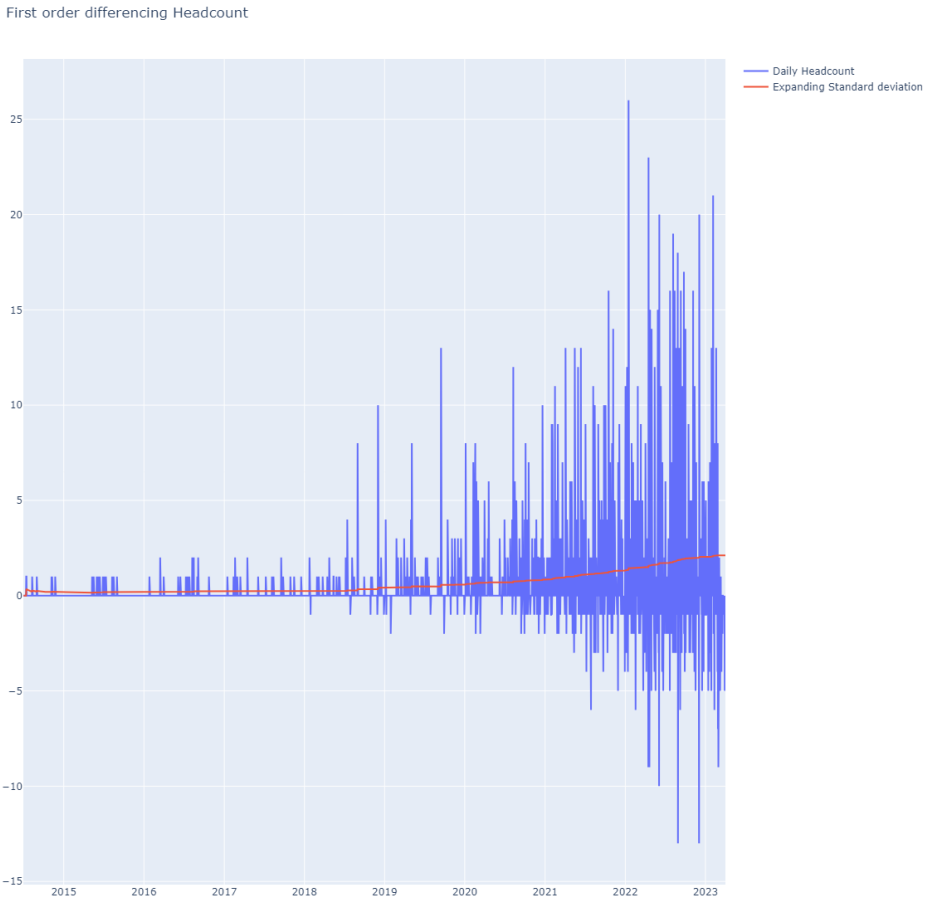
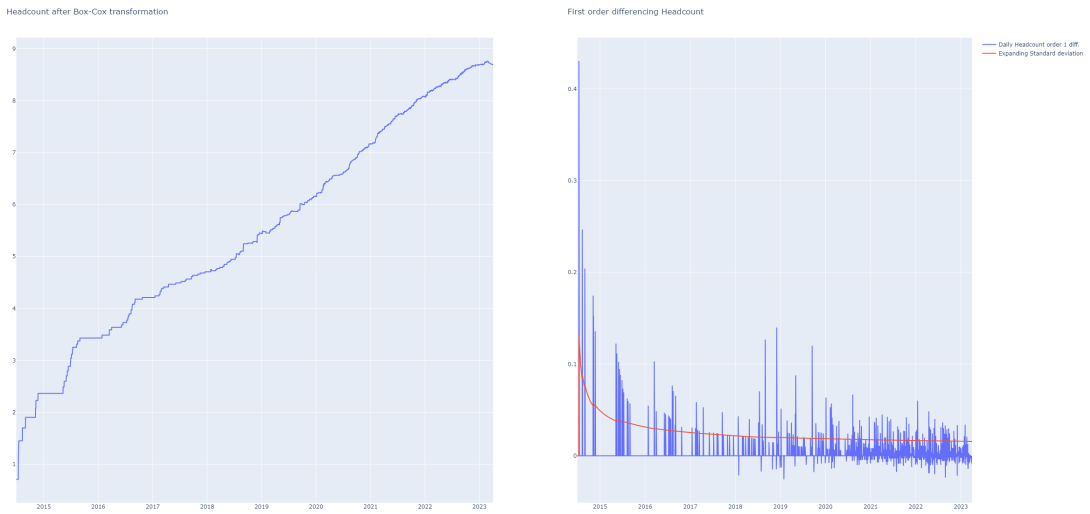


Figure 15: Daily headcount after applying a first order differencing

We can see in the figure that the variance of the data is not constant, so in order to reduce it, a Box-Cox transformation will be applied.

On the following figures (16a, 16b), we can see the headcount after applying a box cox transformation, and the first order differencing of the transformed data, respectively.



(a) Headcount after applying a Box-Cox transformation

(b) First order differencing of the transformed data

Figure 16: Headcount after applying a Box-Cox transformation and first order differencing

These figures show that the variance of the data becomes constant over time after applying the transformation, and we can now fix $d = 1$. As the data is now stationary, the parameter D can be set to 0, as no seasonal differencing terms are needed.

To be certain that the right choice of the differencing order was made, the **Augmented Dickey-Fuller** test was performed on the transformed data using the **adfuller** function from the **statsmodels** library. The test returned a p-value of approximately 6.68-16. As the p-value is lower than the 0.05 threshold, the null hypothesis of the data being non-stationary is rejected.

Finding p and P

As stated on the ARIMA models section, the autoregressive parameters p and P can be found by observing the partial auto correlation function (PACF) of the data. This function of the transformed data can be seen on the figure 17.

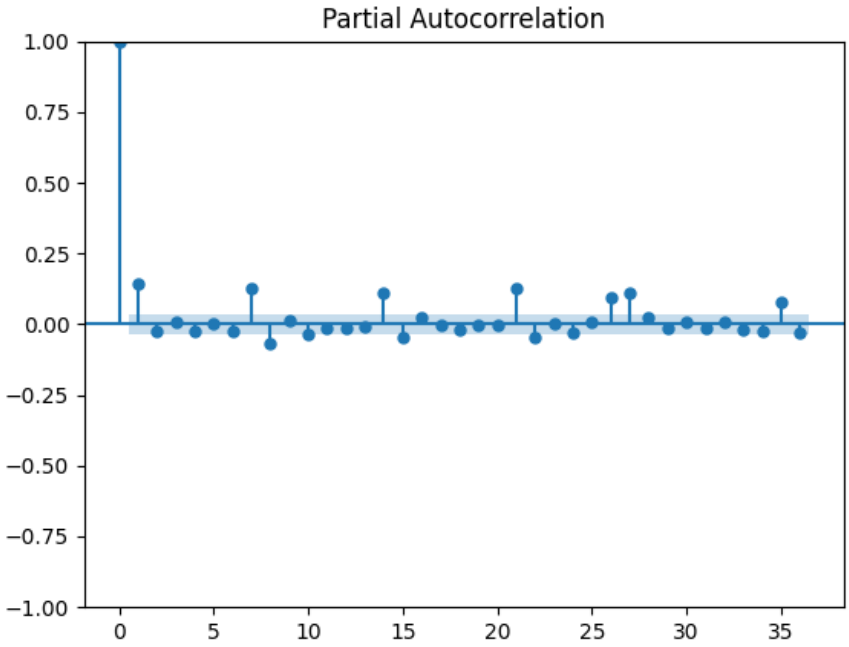


Figure 17: Partial auto correlation of the transformed headcount

From this figure, we can see that as the second lag directly crosses the significance limit, and the first lag is outside the limit, p can be set to 1. Concerning the seasonal autoregressive parameter, the seasonal lags at a weekly frequency (7) are significant, so we can add one seasonal AR term, and set P to 1. However, we can see that the lag 27 is also outside the confidence interval, this can indicate a monthly seasonality. We have decided to ignore this lag, in order to avoid overfitting the model.

Finding q and Q

The moving average parameters q and Q can be found by observing the auto correlation function (ACF) of the data. The following figure 18 shows the ACF of the headcount after the Box-cox transformation.

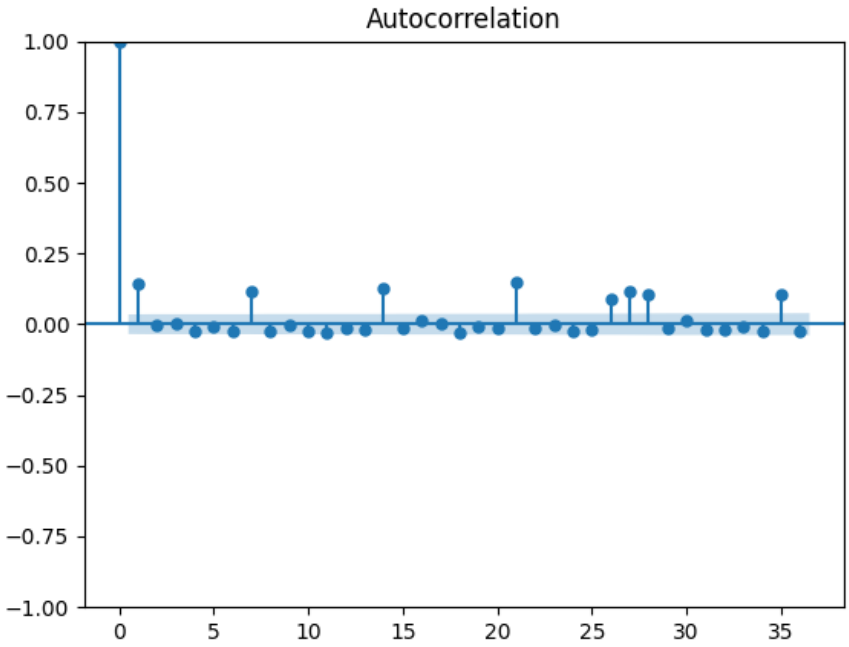


Figure 18: Auto correlation of the transformed headcount

The same logic as the AR terms is applied for finding the number of MA terms, so we can deduce that $q = 1$ and $Q = 1$.

We can summarize this information in the following table:

Number of terms	Non-seasonal	Seasonal
Autoregressive	$p = 1$	$P = 1$
Moving average	$q = 1$	$Q = 1$
Differencing	$d = 1$	$D = 0$

Table 1: Parameters of the SARIMA model

Now, the SARIMA model is ready to be fitted to the data.

To test the model, the last 90 days will be used as test data, so they will be removed from the training set.

The predictions are automatically calculated with the **get_forecast** method of the fitted SARIMA model class.

The following figure 19 shows the predicted values against the real values of the test set. The left figure shows the whole data and includes a confidence interval, while the right figure only shows a zoomed in version of the last 90 days.

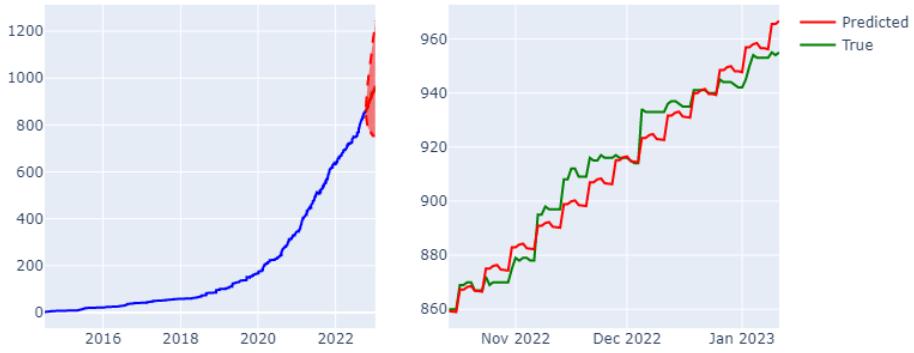


Figure 19: Predicted against test values of the SARIMA model

The model seems to be able to capture the trend and seasonality of the data, and the results are promising.

The following figure 20 presents the relative error, absolute error and squared error of the test set against the predicted values over time.

Figure 20: Errors of the SARIMA model over time

As the relative error is always almost below 1%, we can be lead to believe that this model is appropriate for forecasting this type of data.

The mean errors of the model are shown on the following table:

Metric	Value
Mean absolute error	5.19
Mean squared error	39.66
Root mean squared error	6.29
Mean absolute percentage error	0.5%

Table 2: Mean errors of the SARIMA model

It can be said that, on average, the model predicts the headcount to be 5.19 people off the real value, on a 3-month forecast. These results imply that the headcount is a metric which can be easily predicted by a time series forecasting model.

The same process is applied for when the headcount is filtered by a category. However, one SARIMA model is fitted for each category value, and the predictions are made for each model.

An example of the resulting figure for the headcount forecast by department is shown on the figure 21, filtered by department.

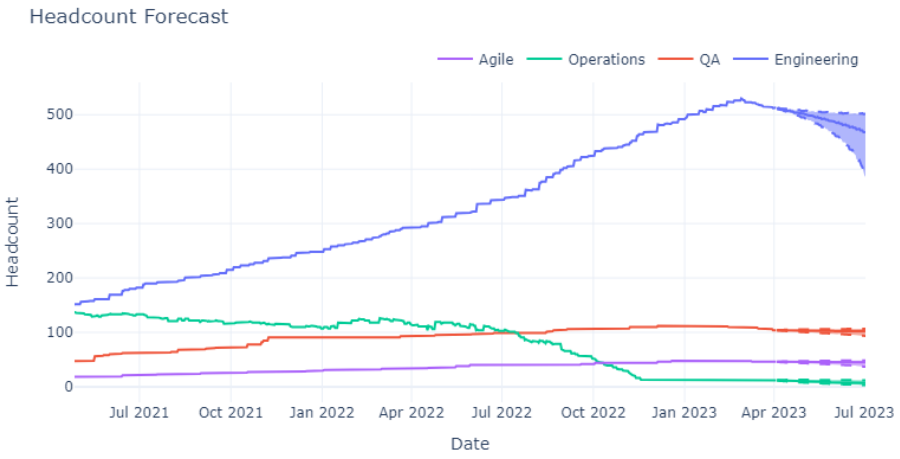


Figure 21: Forecast of the headcount by department

III.2 Income Forecasting

The main purpose of this metric is to predict the income of the company, as well as the worked hours over a certain period of time.

This is useful to help the company to financially plan the future, to help in the decision-making processes and to better evaluate the performance of employees.

This metric is calculated using the Harvest time report dataset. As previously stated, the time report is indexed by each one of the daily worked hours entries, that is entered for each working day of the employees, by project and client.

The dataset contains three main target columns for this metric: the billable amount (income), the worked hours and the cost amount of each employee, by project and client. However, the cost amount is not up-to-date in the dataset, so it will not be considered for this part.

First, the data needs to be aggregated and re-indexed by a date range of daily frequency. To start, the aggregations will be done by the total sum of the billable amount and worked hours, by day. When needed, the data will be filtered by a specific project, client, or any categorical feature.

The first step of the preprocessing phase yields the following figures (fig. 22)

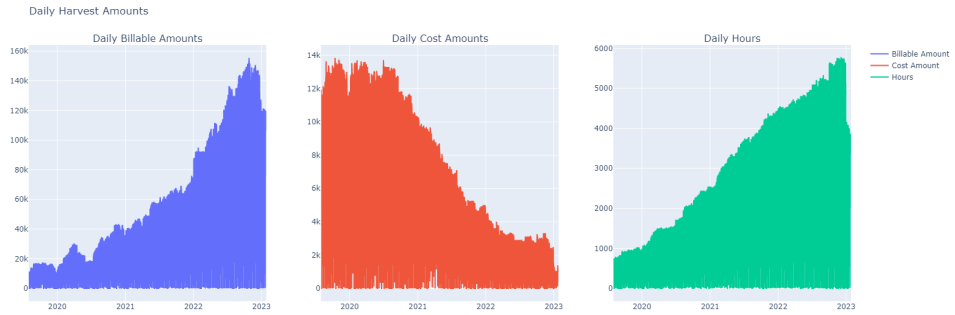


Figure 22: First step of the preprocessing phase of the daily quantities of the Harvest Dataset

As seen on the line plots, the data contains a lot of noise. The values are reaching 0 very frequently. This can be explained by the fact that 0 is assigned to null values, which usually correspond to weekends and holidays.

After replacing the 0 values and weekends by the previous value of the time series, the following figures are obtained (fig. 23)

Now, it can be observed that the information is now less noisy. However, there are still a number of outliers that need to be treated. These outliers represent some holidays, as some employees worked on those days but not the majority of them. Since Applaudo's employees work from different countries around the world, and there is no public worldwide holiday dataset available, a statistical method will be used to fix the outliers. The aberrant values will be detected by stating that a value is an outlier if it is located below or above

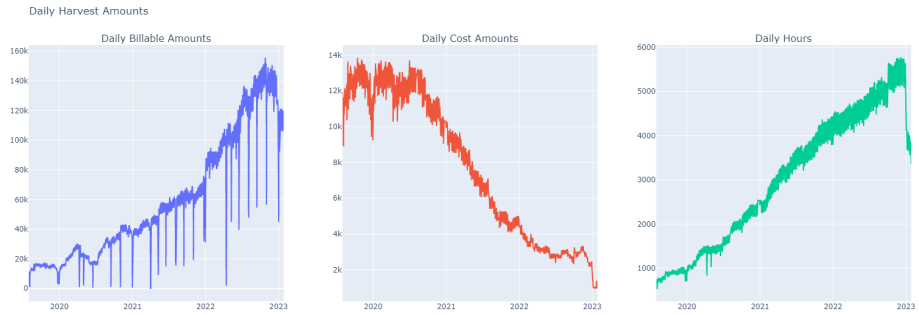


Figure 23: Second step of the preprocessing phase of the daily quantities of the Harvest Dataset

the 30-day moving average plus or minus two times the 30-day rolling standard deviation of the data. This can be appreciated, for the income column, on the following figure (fig. 24)

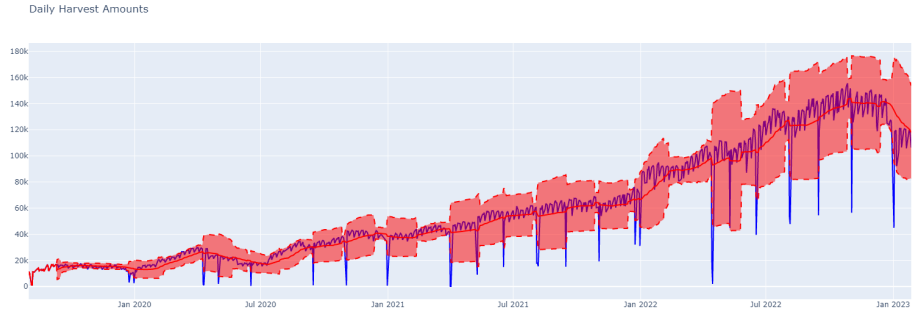


Figure 24: Statistical method to detect outliers of the income column

Some outliers are not detected by this technique but can still be observed on the plots. For example, Mother's Day in El Salvador for 2022. This aberrant values will be removed by hand, since they are not very frequent.

The final product of this treatment is shown on the following figure (fig. 25)

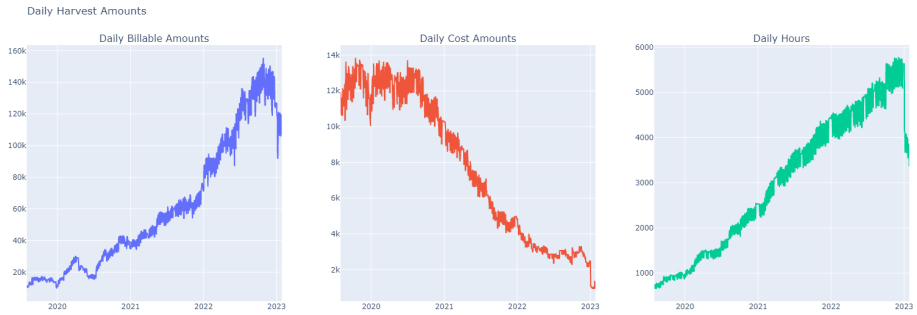


Figure 25: Final product of the preprocessing phase of the daily quantities of the Harvest Dataset

To forecast the income and the worked hours, we have decided to use an LSTM model, since the SARIMA model did not provide reliable results for this metric. Any LSTM model looks at a previous window of time steps, called lookback window, and predicts a future window, called forecast horizon.

For this metric, the considered forecast horizon will be of one year.

As the data is too noisy, it was aggregated by a weekly frequency, and the lookback window was set to 52 weeks.

An example of one window of the training data can be seen on the following figure (fig. 26)

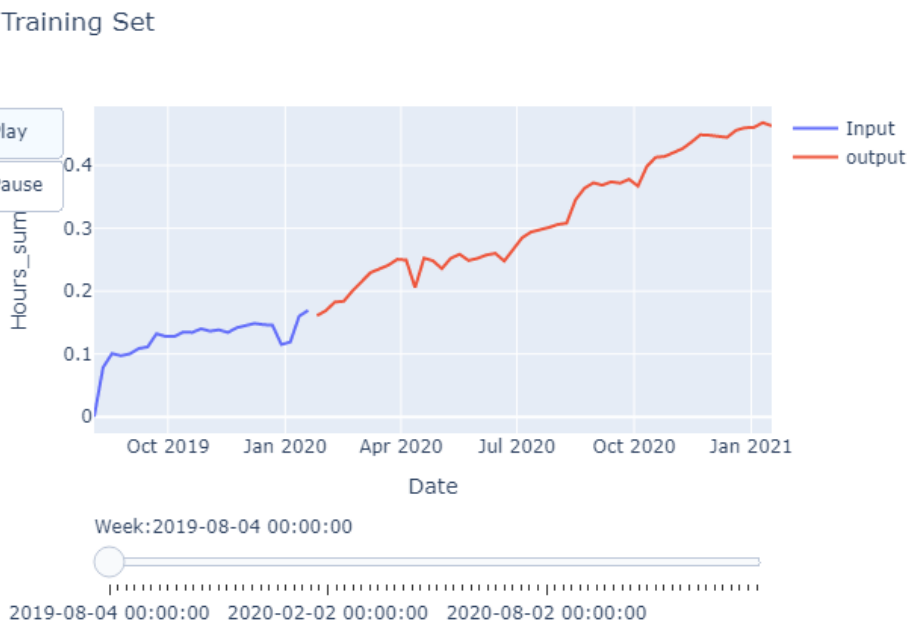


Figure 26: Example of one window of the training data for the LSTM model

Some tuning needs to be done in order to choose the correct lookback window. After a simple grid search, it was determined that 25 weeks was an appropriate lookback window. Also, the number of units of the LSTM layer was set to 128.

To transform the data into a supervised problem, the **timeseries_dataset_from_array** function from the Tensorflow library was used.

Additionally, exogenous variables like the trend of the series, the monthly difference and date related variables (week of year, month of year, year) were considered.

To avoid overfitting, the model will be trained for 55 epochs, with an Adam optimizer and a learning rate of 0.001. The Tensorflow model can be seen on the following code snippet:

```
    class OneLayerLSTM(tf.keras.Model):
    """ LSTM model with one layer (128 units)"""

    def __init__(self,lookback,input_dims,output_dims,forecast_horizon):
        super(OneLayerLSTM,self).__init__()
        self.lstm1 = tf.keras.layers.LSTM(128,input_shape = (lookback,
input_dims),activation = 'relu')
        self.dense = tf.keras.layers.Dense(forecast_horizon*output_dims)
        self.reshape = tf.keras.layers.Reshape((forecast_horizon,
output_dims))

    def call(self,x):
        x = self.lstm1(x)
        x = self.dense(x)
```

```
13          x = self.reshape(x)
14          return x
```

An example of the training loss and accuracy curves can be seen on the following figure (fig. 27)

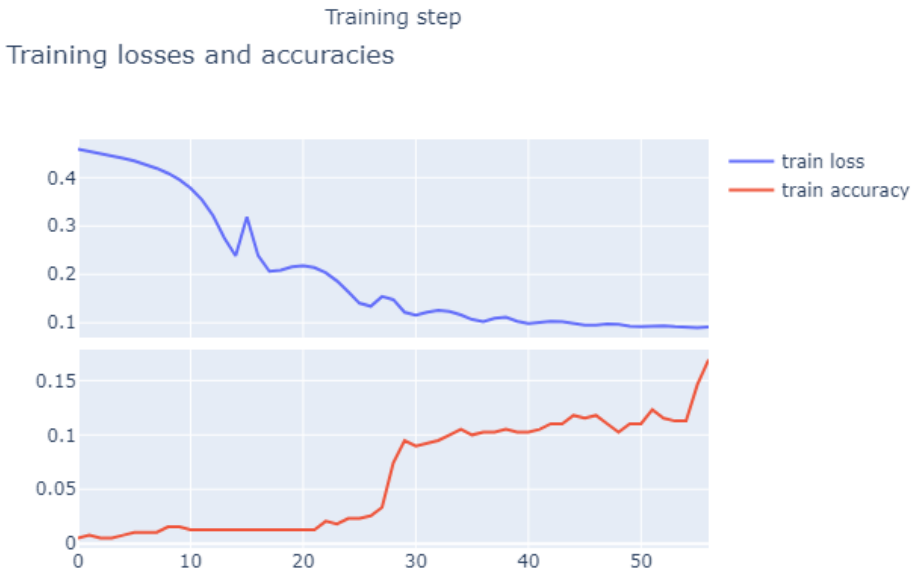


Figure 27: Training loss and accuracy curves of the LSTM model

Also, we can see on the figure 28, the predicted number of worked hours for the training data, for one window of the data. This shows how the model fits itself to the training data without overfitting.

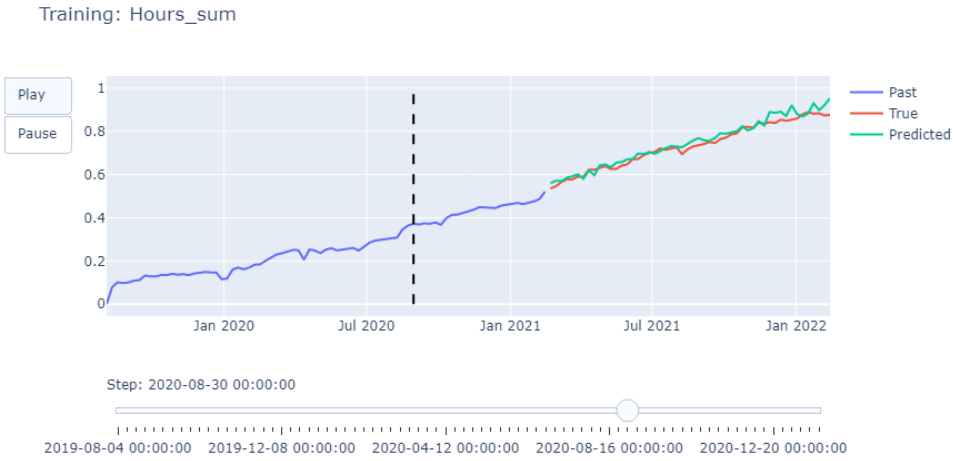


Figure 28: Predicted values of the LSTM model for the training data

To facilitate cloud implementation whenever the data lakehouse is deployed, two files are provided.

The first one is a python script that contains the preprocessing (using BigQuery), it trains the model for 55 epochs, and it saves the model on a Google Cloud Storage bucket. This script should be executed recurrently within GCP, ideally once a month, to keep the model up to date and allow it to make reliable predictions.

The second file is a SQL script that will prepare the data to be ingested by the model, then it will import the saved model and will make a prediction using BigQuery ML. When this file is executed by BigQuery, a complete view of the historical data and the predicted values is returned.

III.3 Project Clustering

The Project Clustering metric has the objective of grouping the projects by their similarities, depending on their progress, to facilitate employee relocation and project management.

To determine the progress of a project, both data from Harvest and Forecast is used. Harvest provides the number of hours worked by each employee on a given project, and Forecast provides the number of hours that are estimated to be worked on that project in the future.

A processing class was created to read the data, parse it by aggregating it by project, merge both datasets and calculate the progress of each project. Finally, the projects are split into three different progress levels. Three different progress levels were considered:

- **Starting projects**: The project has less than 33% of the estimated hours worked.
- **Halfway projects**: The project has between 33% and 66% of the estimated hours worked.

- **End stage projects**: The project has more than 66% of the estimated hours worked.

As each project contains multiple dimensions, such as the sum of the worked hours, the total income, total cost, number of employees working on it, and others, it is impossible to represent it in a two-dimensional space without losing some information. For visualization purposes, a Principal Component Analysis (PCA) will be done in order to reduce the dimensionality.

The **Clustering** class contains a method to normalize the data, perform a PCA, plot the curve to determine the number of clusters, and cluster the data using the K-Means algorithm.

In the following part, a step-by-step example of how the clustering of projects that share the same progress level is done will be shown.

First, the data must be normalized. A min-max normalization is performed to scale the data between 0 and 1. Then, the inertia curve is plotted to use the elbow method to determine the optimal number of clusters (fig. 29).

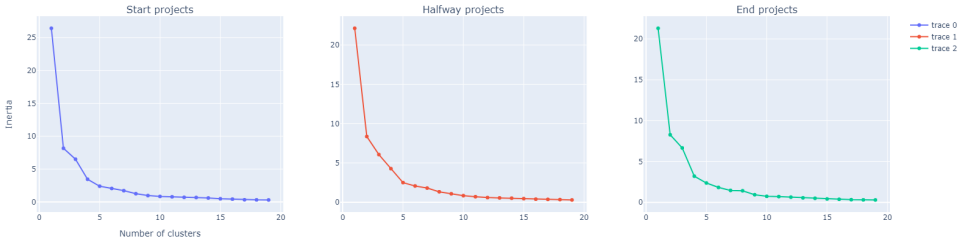


Figure 29: Inertia curve to determine the optimal number of clusters

We can choose 5 clusters for each of the project stages.

After this, the k-means algorithm is applied to the data to form the clusters. The algorithm is implemented with the **KMeans** function of the **sklearn** library. Even if it is still not possible to visualize all the data, the clusters are formed, and it is possible to show a list of the projects that belong to each group.

For example, the following image of a pandas DataFrame 30 shows the projects belonging to the cluster where the project "Cooperton Investments, LTD. / VativoRX" belongs, and their distance to this project.

		distance_to_project
Client	**Project**	
Cooperton Investments, LTD.	VativoRX - Non-Billable	0.000000
KW Accelerator Studios, LLC.	Keller Mortgage - Non-Billable	0.005068
ONLIFE HOLDING INC.	Onlife - Non-Billable	0.006278
Applaudo Studios	Cross-training	0.010168
Benchmark Analytics	Benchmark - Non-Billable	0.010773
...	...	...
LUUK HOLDINGS INC.	Catalogo maestro	0.201408
Applaudo Studios	AI / ML	0.229654
Lifemiles LTD	Maintenance contract	0.241553
LUUK HOLDINGS INC.	Luuk	0.305700
Applaudo Studios	Trainee & Training Program	0.310815

Figure 30: Projects belonging to the same cluster as "Cooperton Investments, LTD. / VativoRX"

Note that the distance is the euclidean distance between the normalized values

Finally, it is now possible to plot the clusters in a two-dimensional space by performing a PCA. The clusters can be seen on the following figure for projects that are in the starting stage (fig. 31).

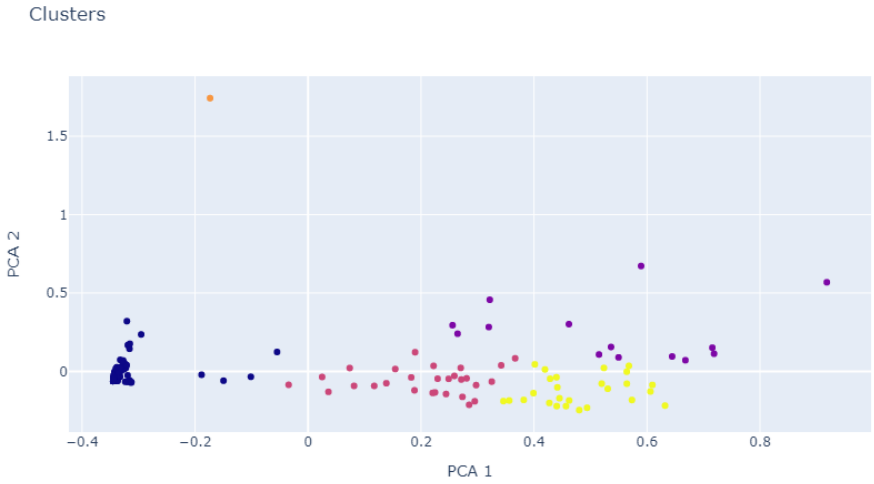


Figure 31: Clusters of projects in the starting stage

III.4 Billable Percentage Visualization

There are already some existing reports, generated by the Applaudo Studios reporting team, concerning the income of the company. However, this metric focuses on creating new visualizations and figures synthesizing more information about the income of Applaudo.

Also, one important feature that will be analyzed is the billable percentage. This can be defined by the proportion of the daily billable hours over the available hours of the day for each employee. It can be calculated with the Harvest time reports by dividing the number of worked hours by the number of available hours per day for each employee, and multiplying by 1 if the hours are billable, or 0 otherwise. We will view this metric as a percentage, so it will be multiplied by 100.

First, we can see the time series of the average billable percentage per month for the whole company (fig. 32). It can be observed that the billable percentage is usually around 50%. This implies that, on average, 50% of the worked hours are spent on billable tasks.

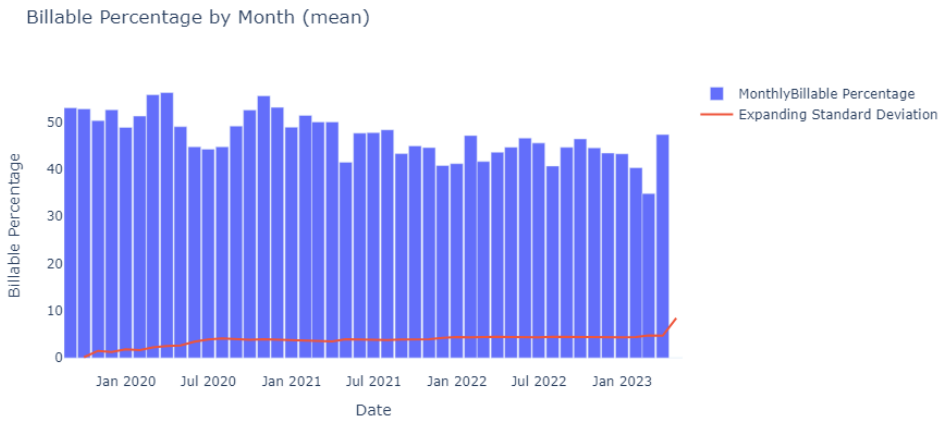


Figure 32: Average billable percentage per month

Additionally, we could group the data by the day of the week and plot the mean billable percentage for each day (fig. 33). This figure allows analyzing in which day the employees of Applaudo are more productive, concerning billable projects.

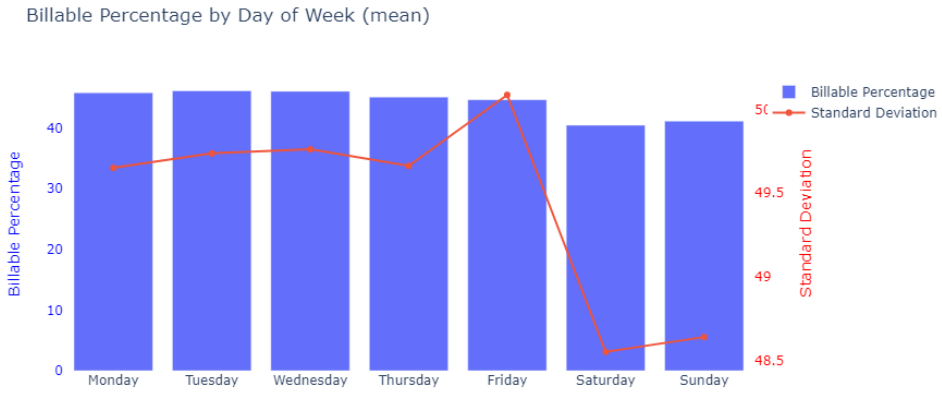


Figure 33: Average billable percentage per day of the week

Note that each generated figure can be filtered by multiple categories. For example, the function that generates the previous figure takes a **date_group** parameter that could be set to "dom" (day of month) instead of the day of the week.

Next, the evolution of the monthly mean billable percentage across the different years can be plotted (fig. 34).

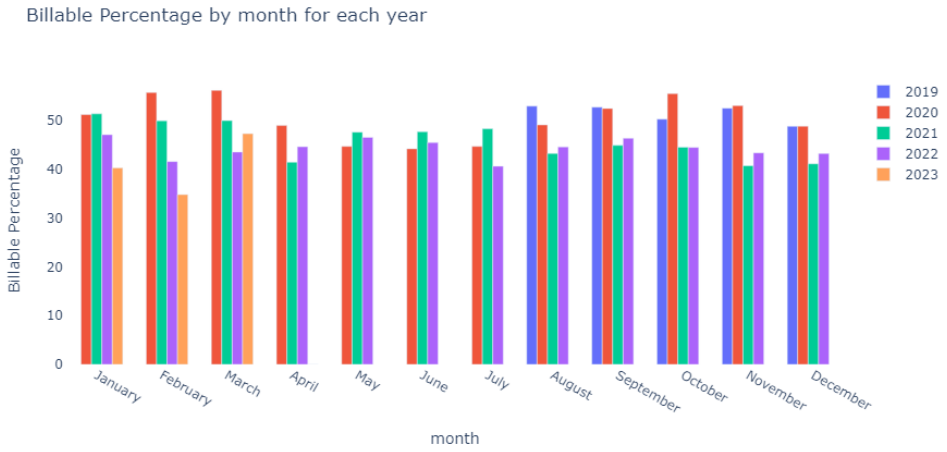
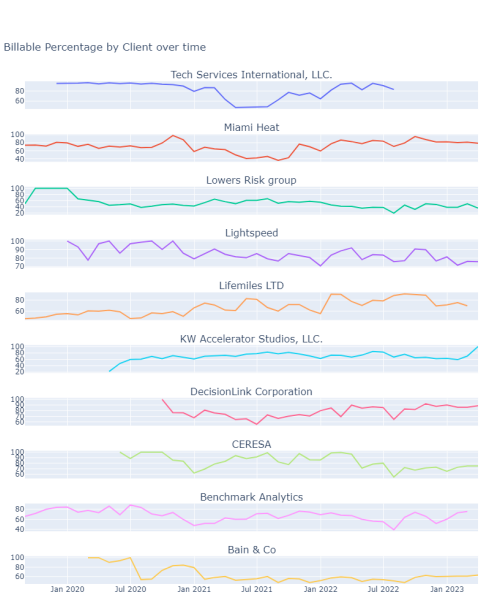
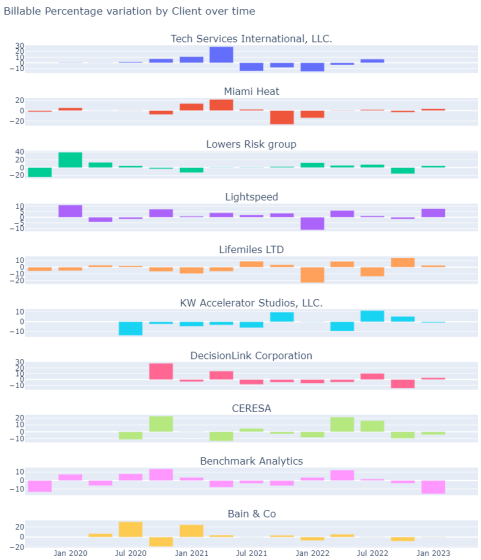


Figure 34: Average billable percentage per month and year

The evolution of the billable percentage over time can be analyzed by client or project (fig. 35). Only the top N clients by income will be plotted, as the number of clients is too elevated to generate a comprehensible figure.



(a) Billable percentage per client



(b) Variation of the Billable percentage per client

Figure 35: Billable percentage per client over time

The Harvest data can be merged with the Zoho roster in order to find some insights about the billable percentage aggregated by employee features (for example, by work expe-

rience). The following figure shows a histogram of the billable percentage per the employee's work experience (fig. 36).

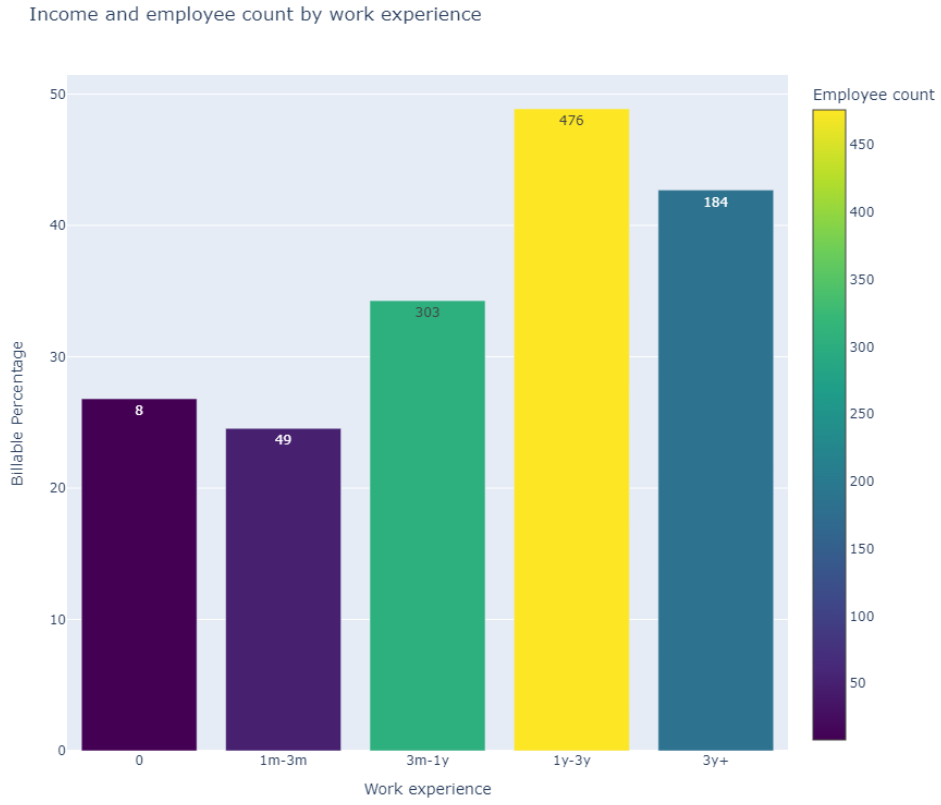


Figure 36: Billable percentage per employee's work experience

Another useful metric to consider is the occupancy rate. The occupancy can be defined as the proportion of worked hours spent on a project (not on the "Available" position) over the total worked hours. This signifies the percentage of time that an employee is actually working on a project.

The following figure shows the billable percentage over the occupancy rate, by job title (fig. 37). It allows analyzing which job titles are more productive and billable.

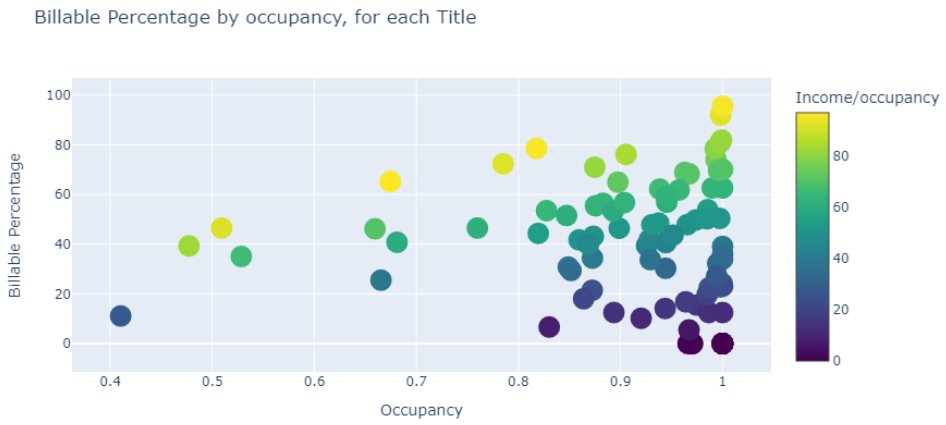


Figure 37: Billable percentage over the occupancy rate, by job title

Finally, it is interesting to analyze the employee flux between the different projects of the company. For this, a Sankey diagram is generated (fig. 38), showing a project on a given month, and the projects where the working employees come from, as well as which projects the employees go to. The following figure shows this diagram for the "available" tag (employees that are not working on a project) for the month of February 2023.

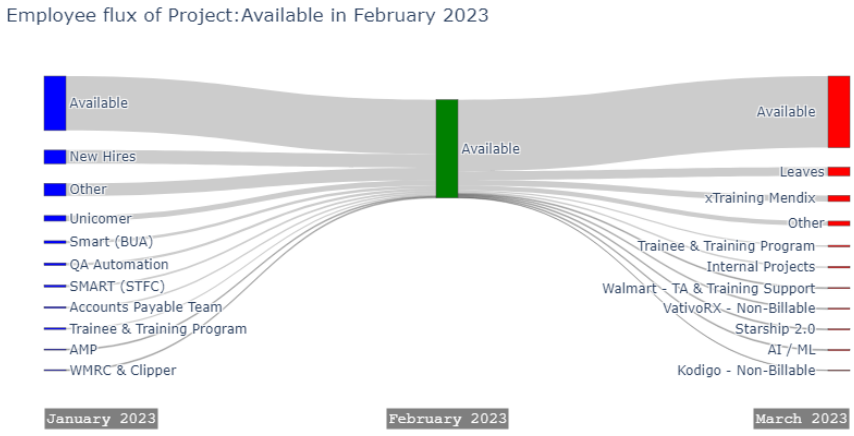


Figure 38: Sankey diagram of the employee flux between projects

All the figures generated for this visualization group are created using plotly, and are parametrized in order to be able to filter the data by any categorical feature or view different target columns.

A dashboard containing some insights was also generated using power BI (fig. 39).

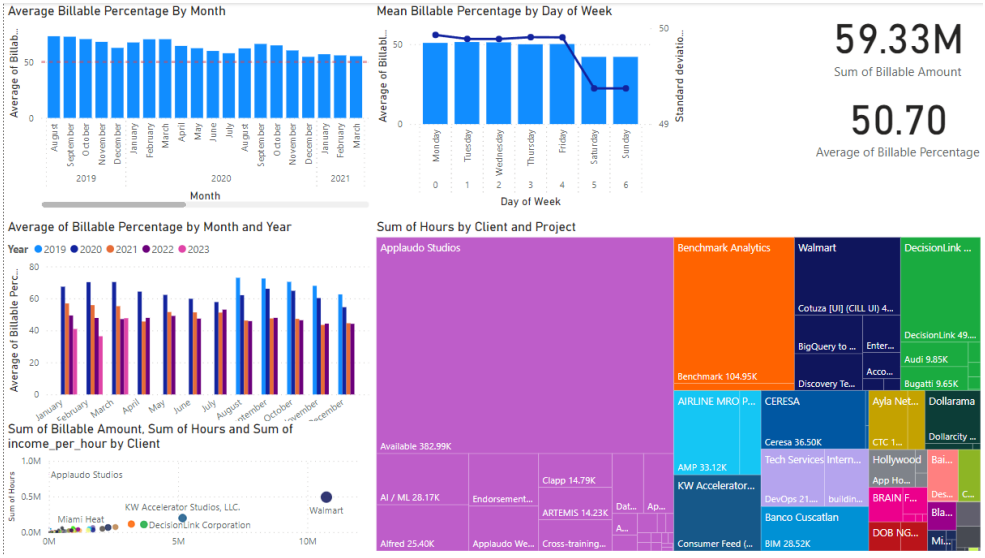
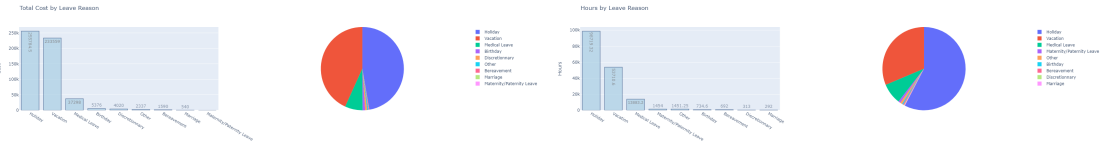


Figure 39: Power BI dashboard

III.5 Leaves Cost Visualization

This metric has for objective to analyze the impact of different leave reasons on the cost of the company. Understanding this information can help to make decisions about how can certain reasons improve without affecting the cost.

The total cost by each leave reason as well as the total hours lost can be calculated using the Harvest data and are shown in the following plotly visualizations (fig. 40).



(a) Total cost by leave reason (b) Total hours lost by leave reason

Figure 40: Total cost and hours lost by leave reason

Additionally, we can calculate the cost per hour for a specific leave reason over time, in order to analyze the times of the year when a certain leave reason has an important impact on the cost of the company.

The following figure shows the cost per hour for the "Vacation" leave reason (fig. 41).

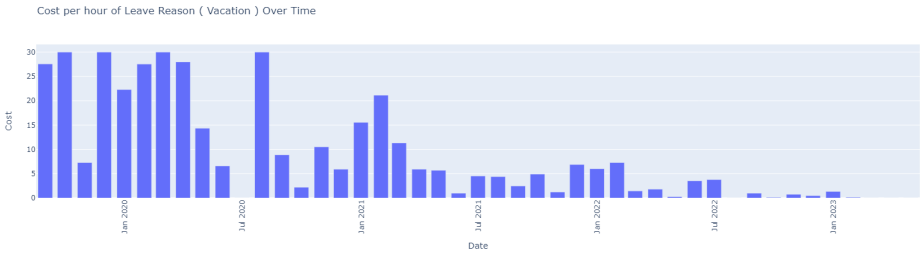


Figure 41: Cost per hour for the "Vacation" leave reason over time

Finally, the relative cost by leave reasons on a given time frequency can be analyzed on a stacked bar chart (fig. 42) to understand the evolution of the cost and prepare the company for the future. The following figure (fig. 42) shows the relative cost by leave reason on a monthly basis.

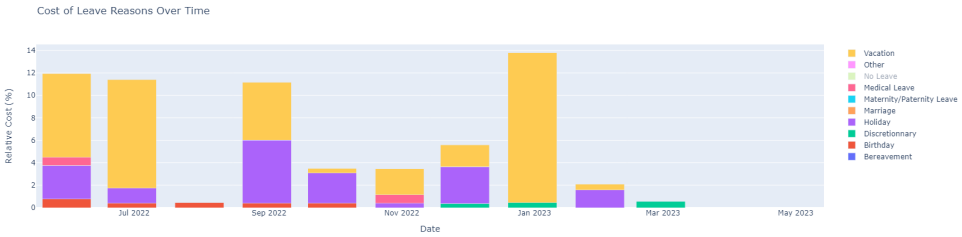


Figure 42: Relative cost by leave reasons on a monthly basis

IV Conclusions

In conclusion, five metrics to implement in the data lakehouse were proposed and developed. The metrics are insightful and propose new ways to look at the available information. The developed metrics were not only able to supply new figures to the company's report, but also to provide forecasts and insights into the future of the company.

Concerning the calculated metrics, the headcount of the company was forecasted using a seasonal ARIMA model, while the total and filtered income was forecasted using an LSTM neural network. Additionally, the company projects were clustered using the K-means algorithm by progress, facilitating the relocation of employees throughout other projects. Finally, two metrics were developed to propose new visualizations for the company's power BI dashboards, showing a new approach to the available information.

Even though Applaudo Studio's data lakehouse is still in development, the proposed metrics were coded in a way that can easily be modified to adapt to the deployed lakehouse structure.

During this internship, I had the opportunity to approach for a first time Cloud Computing. I was able to familiarize myself with the Google Cloud Platform and its services,

as well as to learn how to operate with BigQuery and make use of its machine learning capabilities. Also, I could learn how to process data using GoogleSQL and how to train a Machine Learning model on the cloud. I had also the opportunity to create dashboards and process information using Power BI, which is something that allowed me to acquire some experience in business intelligence, data modelling and data visualization. Additionally, I learned in detail how time series forecasting and their mathematics works, specially concerning the ARIMA model family and recurrent neural networks.

My contributions to the project allowed Applaudo Studios to have a better understanding of their data and to possibly have live and dynamic reports in the future. These contributions were also useful to improve the existing forecasting models available inside the company. Also, my contributions allowed me to improve my skills, learn new technologies and gain experience as a data engineer.

With the daily team meetings, I was able to share with the team the difficulties encountered, as well as to receive helpful suggestions and feedback on how to improve my work. Also, the occasional one on one meetings with the internship supervisor and with some team members were insightful and allowed me to advance in the project.

Even though I will not be continuing to work for Applaudo Studios, I am extremely grateful for the opportunity to work with them and learn from the team and the project managers.

I would like to thank the Applaudo Studios family for taking me in and making me feel welcome. I would also extend my gratitude and appreciation to the internship supervisor, Jairo Urbina, for his guidance and teaching me new skills all throughout the internship. Also, I thank the Applaudo Data Engineering team for their support and help during this project. Finally, I would like to extend my gratitude to Christophe Prud'homme, our professor and the CSMI master's supervisor, as well as to my class colleagues, who were always willing to help and support me.

This internship was a fulfilling experience, as it invigorated my passion for data engineering and data science. I am looking forward to continuing my career in similar fields to keep learning and improving my skills in the process.

Bibliography

[1] https://www.kdnuggets.com/2015/09/data-lake-vs-data-warehouse-key-differences.html

[2] https://www.databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html

[3] https://panoply.io/data-warehouse-guide/data-warehouse-vs-data-lake/

[4] https://aws.amazon.com/blogs/big-data/build-a-lake-house-architecture-on-aws/

[5] https://cloud.google.com/docs/overview

[6] https://cloud.google.com/bigquery/docs/introduction

[7] https://cloud.google.com/storage/docs/introduction

[8] https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview

[9] https://www.zoho.com/people/help/adminguide/overview.html

[10] https://www.getharvest.com/why-harvest

[11] https://www.getharvest.com/forecast

[12] https://people.duke.edu/~rnau/411arim.htm#arima010

[13] https://arauto.readthedocs.io/en/latest/how_to_choose_terms.html

[14] https://statisticsbyjim.com/time-series/autocorrelation-partial-autocorrelation/

[15] https://towardsdatascience.com/box-cox-transform-for-time-series-cc45f26082c6

[16] https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide

[17] https://colab.research.google.com/drive/1fWycFlt5ddUrALMlLoJA5ZXsATAV-oIc?usp=sharing

[18] https://colab.research.google.com/drive/1s0-vh32jTWr5Wv46jd_e6RO7t7Nb5hcr?usp=sharing

[19] https://www.ibm.com/topics/neural-networks

[20] https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-

[21] https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clusterir

[22] https://www.geeksforgeeks.org/ml-k-means-algorithm/

[23] https://colab.research.google.com/drive/1UpVHImq5bAMmPCi1AMQyl7TtRBFoVFct