Business Analytics Report

Massimiliano Nardone, Chi Chun Matthew Chan,
Vedat Yasar, Didem Uysal, Habiba Sultana

December 2023

1 Introduction

1.1 Understanding the Bike Sharing

As urban centers continue to expand and populations surge, the pressing matter of transportation has emerged as a paramount concern for residents in our modern era. Factors such as traffic congestion, environmental pollution, and time loss, particularly, have increased the need for alternative transportation options. In this context, bike sharing stands out as a sustainable transportation solution. Many major cities are adopting bike sharing programs to make urban transportation more sustainable. In this regard, large metropolises such as Seoul, the capital of South Korea, actively use bike sharing systems to address transportation problems in urban areas. Seoul, in this context, not only promotes bike usage for work and leisure purposes but also encourages city residents to adopt a healthy lifestyle by using bicycles even on holidays.

1.2 Waves of Difficulty: Confronting Bike Rental Industry Challenges

The primary challenge we face is the accurate prediction of bike rental demand, a task that necessitates careful consideration of various influencing factors such as weather conditions, time of day, and holidays. To tackle this objective, we will leverage the Seoul Bike Sharing Demand dataset (2020) from the UCI Machine Learning Repository, focusing specifically on predicting bike demand at each hour of the day. We assume that the weather data is uniformly collected from the same weather station for all instances in the dataset. Additionally, we presume that the total number of bike stations and bikes remained constant throughout the data collection period. From a business perspective, a precise analysis can empower a company to strategically plan resource allocation based on the newly predicted distribution of bikes. This not only enhances user experience by ensuring a reliable and swift supply of bikes but also presents economic advantages. Timely bike availability, coupled with reduced maintenance costs, provides a competitive edge through lower operational expenses, increased reliability, and more efficient decision-making strategies.

While acknowledging that this analysis may not be a silver bullet for the bike forecasting problem, our team proposes several future improvements for the company to consider. One such suggestion involves integrating a GPS module to gather geo-location data. This additional information can significantly impact future decisions, enabling a deeper understanding of usage patterns based on geographical coordinates. Identifying high-demand locations for customer usage becomes more accurate, allowing the business to address the challenge of relocating used bikes to bike sharing stations both precisely and efficiently.

In conclusion, our analysis aims not only to predict bike demand but also to provide actionable insights for strategic planning and resource optimization. As we move forward, incorporating additional data sources, such as geo-location information, can further refine the decision-making process, ultimately leading to a more resilient and responsive bike rental system.

1.3 Igniting Profitability: The Transformative Impact of Analytics

Performing business analytics on bike rental data as a bike provider company can be highly beneficial for several reasons: Demand Analysis, Pricing Optimization, Operational Efficiency, Customer Segmentation, Forecasting and Planning. Optimal pricing strategies can be identified by analyzing rental data and understanding the price sensitivity of customers. e major questions you wish to answercan be examined to adjust rates and maximize revenue. For example, dynamic pricing can be implemented during high-demand periods to capture additional profit or discounts can be offered during low-demand periods. Bike distribution, maintenance schedules, and station capacities can be optimized based on demand patterns. Predictive analytics can be used to anticipate maintenance needs and proactively address issues before they impact customer experience. Operations can be streamlined, costs can be reduced, and resource utilization can be improved through data-driven optimization. Key performance indicators (KPIs) such as revenue per bike, customer acquisition cost, customer lifetime value, and customer satisfaction can be regularly monitored. The impact of strategies and initiatives can be tracked, and necessary adjustments can be made based on performance metrics. A culture of continuous improvement and learning from data can be embraced.

1.4 Expectations About Implications and Findings

This study aims to explore the user rental pattern for bike demand and extract valuable knowledge to improve business insights. By analyzing user behavior, preferences, and external factors, it aims to gain a deep understanding of the underlying patterns driving bike demand. Leveraging advanced data analysis techniques and machine learning algorithms, it is aimed to extract meaningful insights from the collected data. These insights have the potential to optimize resource allocation, enhance service offerings, and maximize customer satisfaction, thereby contributing to the growth and success of bike rental businesses. The findings presented in this report have implications for various stakeholders, including bike rental service providers, urban transportation planners, and non-technical business stakeholders in the field of analysis.

1.5 Methodological Approaches

-Methodologically, we aim to diversify our dataset using feature engineering techniques based on our existing and acquired domain knowledge. By applying these techniques, we will generate diversified datasets, and subsequently, develop models on these diversified datasets to obtain results. Our goal is to compare the obtained results and derive an analytical conclusion.

The models we will employ fall into three categories: linear regression models such as Ridge, Lasso, and ElasticNet Regression [1]; decision tree models such as Decision Trees and Random Forest; and finally, more advanced models such as K-Nearest Neighbors, Support Vector Machine and XGBoost, which possess a more sophisticated structure.

1.6 The Dataset

The Seoul Bike Sharing Demands dataset has been retrieved from the UC Irvine Machine Learning Repository website and was donated on 2/29/2020 [2]. This dataset includes information on the hourly bike rental counts and date details within the Seoul Bike Sharing System. Additionally, it includes weather-related features such as temperature, humidity, wind speed, visibility, dew point, solar radiation, snowfall, and precipitation. The dataset also incorporates contextual information such as season, holidays, and functional day indicators.

1.7 Overview of the Data

The primary objective of the Seoul Bike Sharing Demands dataset is to predict the hourly aggregated bike rental counts within the Seoul Bike Sharing System. With its rich features, this dataset contains various variables for analysis and modeling. Providing a comprehensive foundation, this dataset combines both temporal and weather factors, offering a robust basis for understanding how bike demand responds to diverse conditions. Here is an overview of the dataset:
Source:

- UC Irvine Machine Learning Repository.
- DOI: 10.24432/C5F62R

Size of Data:
Data consists of 8760 observations with 14 variables.

Features of Dataset:

- Date: Year-month-day format.
- Rented Bike Count: Count of bikes rented at each hour (target variable)
- Hour: Hour of the day.
- Temperature: Temperature in Celsius
- Humidity: Percentage
- Windspeed: Wind speed in metres per second.
- Visibility: Visibility in 10 metres.

- Dew Point Temperature: Dew point temperature in Celsius
- Solar Radiation: Solar radiation in MJ/m2
- Rainfall: Rainfall in millimetre
- Snowfall: Snowfall in centimetre
- Seasons: Winter, Spring, Summer, Autumn (Categorical)
- Holiday: Holiday/No holiday (Categorical)
- Functional Day: No (Non-Functional Hours), Yes (Functional hours)- Categorical

Data Types:
This dataset includes a mix of numerical and categorical features. After considering all the above information, analysing and modelling could involve regression techniques and time series analysis to visualise and understand patterns and make predictions.

1.8 Dataset can answer the questions

The dataset used in this study is a collection of variables, including both temporal and weather factors. It provides a solid foundation for understanding how bike demand responds to diverse conditions. By analyzing the temporal variables, insights into the fluctuations of bike demand throughout different times of the day, days of the week, and seasons can be gained. The inclusion of weather factors allows to explore how bike demand is influenced by various weather conditions, such as temperature, precipitation, wind speed, and humidity. This dataset offers valuable information to answer specific questions and test hypotheses related to bike demand. It provides framework to investigate the complex relationship between bike usage patterns and a wide range of conditions to understand the factors that drive bike demand.

1.9 CRISP-DM Approach

In this study, the CRISP-DM (Cross-Industry Standard Process for Data Mining) approach to analyze the dataset is applied. by using the dataset consisting of variables, it offered a solid basis for understanding how bike demand responds to different conditions. The project followed several phases of CRISP-DM approach. First, for data understanding and preparation, the dataset is being explored and understood. This phase also included identifying relevant variables and performing necessary data transformations. Then, various algorithms techniques have been applied, such as regression analysis and time series modeling, to investigate the relationship between bike demand and the identified factors. In the evaluation phase, the performance and accuracy of our models have been calculated in order to compare. Finally, with the interpreted results, meaningful conclusions and insights regarding the factors influencing bike demand were observed. By applying the CRISP-DM approach, we ensured a systematic and structured analysis of the dataset, enabling us to gain comprehensive insights into the dynamics of bike demand.

2 Journey Begins: Acquiring Bike Rental Data

Embarking on a data exploration journey, we delve into the realm of acquiring bike rental data. In this section, we undertake the role of intrepid data analysts, navigating through various aspects such as missing values, outliers, preprocessing techniques, and data transformations. In this phase, we successfully tackled one of the initial challenges we encountered, which involved addressing the issue of special characters within the column names of the dataset. By employing the check.names = FALSE parameter, we effectively resolved this challenge and eliminated any potential complications related to special characters throughout the preprocessing process.

2.1 Insights and Visualization : Exploratory Data Analysis (EDA)

In the preliminary phase of our Exploratory Data Analysis (EDA), we commenced by gaining insights into the descriptive statistics of our features, aiming to understand the characteristics and distributions of our dataset. Additionally, we utilized a dedicated plotting function, referred to as "plot_data" to illustrate the distribution patterns of bike rentals at an hourly level (Figure 3), as well as during different seasons and holidays. Furthermore, we generated a pie chart (Figure 4) to effectively visualize the variations in bike usage across different seasons.

The visual representations provided an initial assessment of the behavioral patterns associated with the target variable, Rental Bike Count, across various time frames. A noteworthy observation is the presence of a bimodal

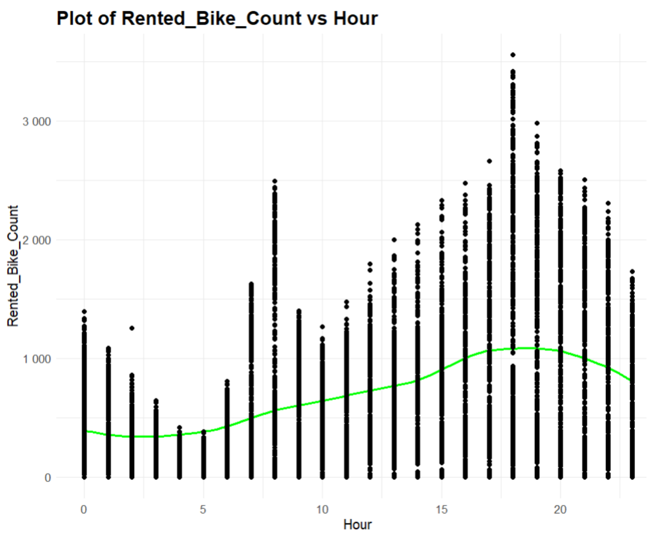


Figure 1: Plot of Rented_Bike_Count over Hour

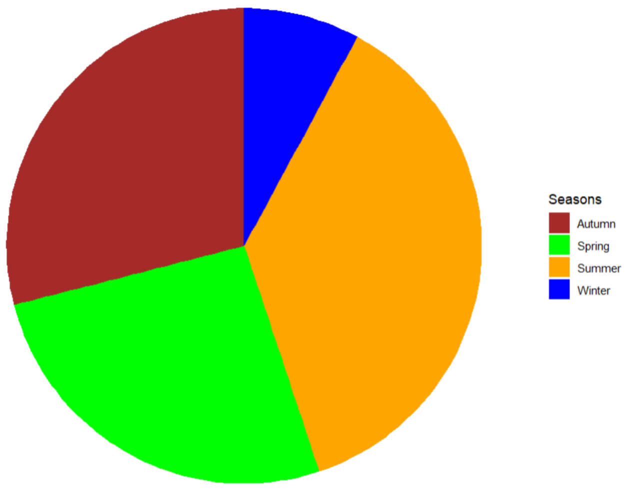


Figure 2: Seasonal Pie Chart

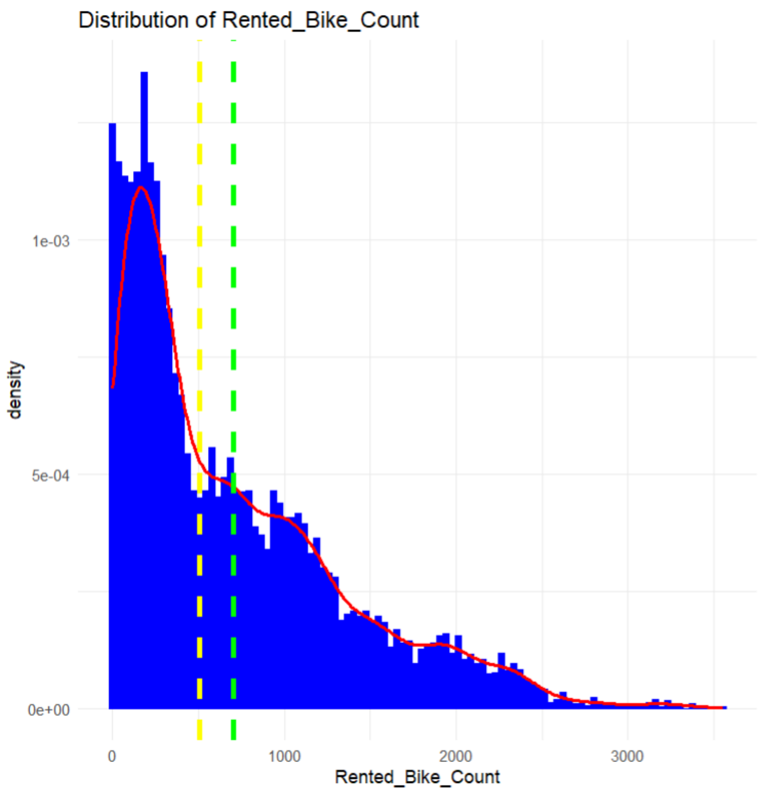


Figure 3: Distribution of Rented_Bike_Count

peak in bike usage, predominantly occurring at 8 AM and 6 PM. Additionally, a decrease in bike rentals during the winter season was evident.

Given the considerable fluctuations in temperature, it is plausible to assume that temperature variations contribute to a portion of the differences in bicycle rental numbers. This trend suggests a preliminary correlation between Rental Bike Count and the Temperature feature. To further investigate this potential relationship, we conducted a comprehensive statistical analysis using the summary() function. This analysis facilitated an overview of key statistical measures, including minimum, first quartile, median, mean, third quartile, and maximum values, for each column in the dataset.

Insights from this analysis:

- A notable discrepancy between the median (504.5) and mean (704.6) of the Rented Bike Count, indicating a right-skewed distribution. This skew- ness suggests that specific hours or conditions contribute to substantially elevated rental counts.
- The percentiles provide information about the spread of the data and the majority of Rented_Bike_Count values fall within the 25th to 75th percentiles, ranging from 191.0 to 1065.2.
- The mean (average) temperature is 12.88 with range of temperatures is quite wide, with the lowest value being -17.8 and the highest value being 39.4. The majority of temperature values fall within the 25th to 75th percentiles, ranging from 3.5 to 22.5.
- The mean (average) humidity is 58.23 with range of humidity values is from 0 to 98, covering a wide range of humidity levels. The majority of humidity values fall within the 25th to 75th percentiles, ranging from 42 to 74.
- The average wind speed is relatively mild, registering only 1.7 meters per second, with the peak wind speed reaching only a gentle to moderate level.
- The majority of other features exhibit distribution patterns that align with a normal distribution.

In order to enhance visual interpretability, we employed the 'plot distributions in batches' function [5]. This tool concurrently presents two charts, thereby enabling a more refined analytical perspective. The visual depiction of the Rented Bike Count reinforces our earlier observation regarding its right-skewed nature, as the median,

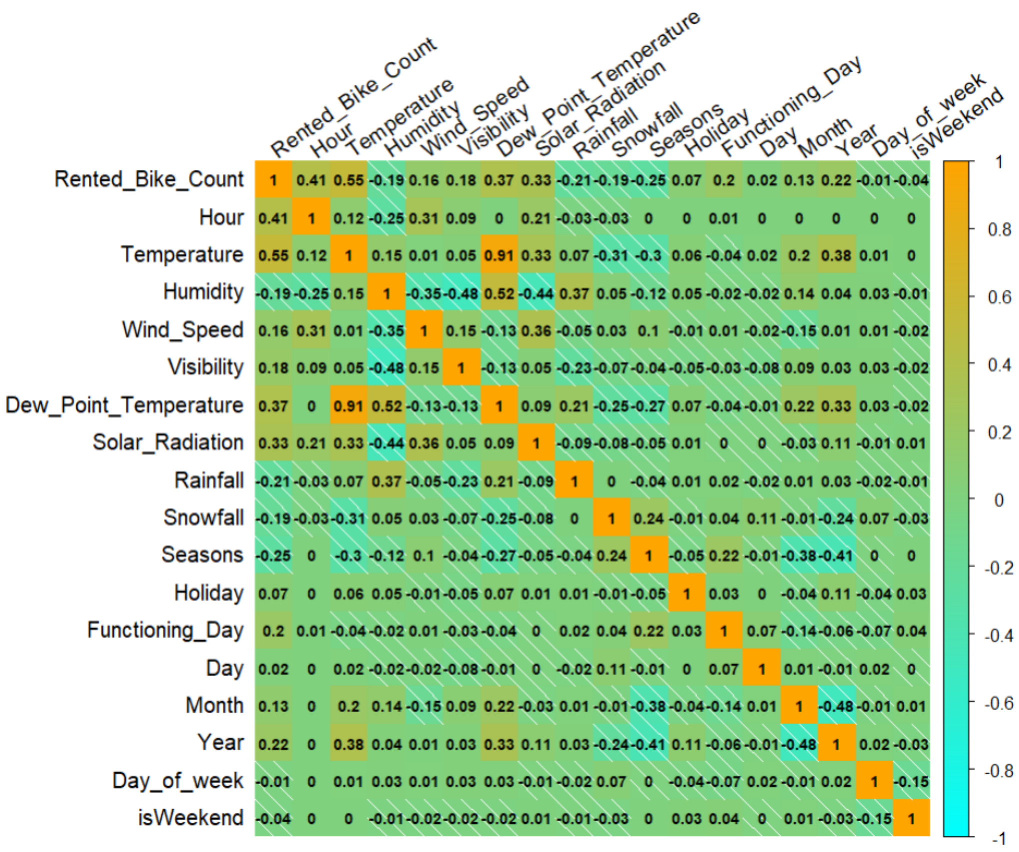


Figure 4: Correlation Matrix

denoted by a yellow dashed line, is positioned below the mean represented by the red dashed line (Figure 5). This particular characteristic of the data will be revisited in the Feature Engineering section of our analysis.
Our investigation delved into the generation of a Correlation Matrix using the 'plot correlation matrix' function (Figure 6). Findings from this matrix unveil a strong correlation (0.91) between Dew Point Temperature and Temperature, warranting further exploration during the Feature Engineering phase. Key variables demonstrating robust correlation with our target variable include Temperature (0.55), Hour (0.41), Dew Point Temperature (0.37), and Solar Radiation (0.33). This analysis raises the possibility of constructing a distinct dataset, potentially excluding features with a correlation coefficient below 0.1. It's crucial to recognize that correlation coefficients exclusively signify linear relationships, and non-linear associations may not be evident in this matrix.

2.2 Dusting Off the Data: Cleaning and Polishing Bike Rental Data

We initiated the process by safeguarding the original dataset under the name 'original_SeoulDataset' to prevent any unintended modifications. A duplicate of this dataset was then created in another object, 'NonClean_dataset,' where various cleaning tasks were initiated. The first step involved renaming columns to mitigate potential issues arising from special characters and punctuation. We opted against merging this dataset with others, as it already resulted from a combination of datasets.

Upon scrutinizing the dataset for missing values using the is.na() function in R, we observed no instances of missing data. As a result, our custom function "check_missing" designed for mean imputation, was not employed.Upon employing the sum(duplicated()) function for verifying duplicate entries, no duplications were detected, rendering our "remove_duplicates" function unnecessary. Subsequently, we documented the dataset's dimensions, consisting of 8760 rows and 14 features, as previously mentioned in the introduction.
In the realm of Practical Business Analytics, managing outliers holds significant importance for enhancing accuracy, improving model performance, and focusing on core trends. Outliers have the potential to markedly skew results, leading to inaccurate conclusions that fail to capture the underlying patterns in the data. To address this, we utilized the "plot_outliers()" function, generating bar charts for all numerical columns [3]. While we did detect outliers, a conscious choice was made to keep them, understanding the importance of mirroring real-world situations in our analysis and enhancing our model's capacity to deal with extreme scenarios effectively.
Consequently, our "remove_outliers" function, designed to discard outliers outside the 5th and 95th percentile range, went untouched. Despite recognizing alternative outlier-handling techniques such as trimming, capping,

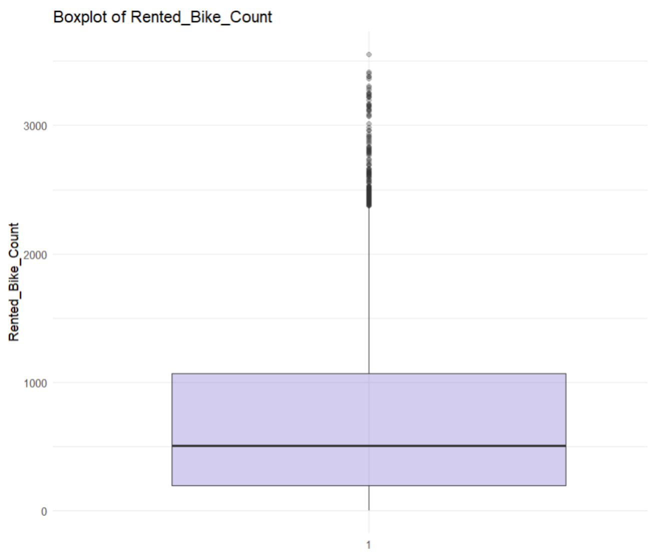


Figure 5: Boxplot of Rented_Bike_Count Outliers

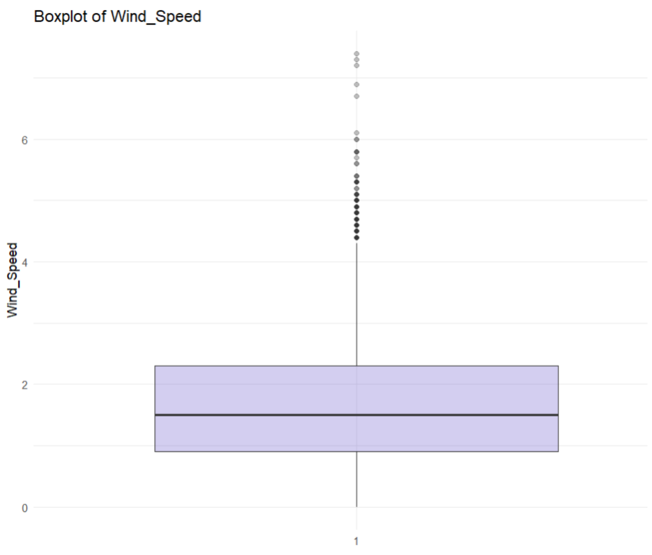


Figure 6: Boxplot of Wind_Speed Outliers

and imputation, we decided to focus our efforts on different facets of the analysis. This initial phase of data cleaning affirms the dataset's overall high quality.
We can assess that by these Key factors:

- Completeness: all the data is present, there are no incomplete records or missing values.
- Accuracy: the dataset is free from errors.
- Consistency: there are no duplicated records and there's no different formatting.
- Variety: The dataset is a good mix of numerical and categorical features.
- Accessibility: The dataset is open source and available to all.
- Trust: the dataset is from a trusted source.

2.3 Data Transformation

In this stage, we transformed the Date column into a Datetime object and decomposed it into new columns: Day, Month, and Year. We introduced three supplementary features: "Day_of_Week" (a categorical feature denoting the day's name), and "is Weekend" (a categorical feature marked "Yes" for Saturdays and Sundays, and "No" otherwise). These additions were made to enhance our comprehension of the temporal aspects associated with bike rentals.

Following the extraction, the Date column was eliminated, although it was retained in the "Non_Clean_dataset" for subsequent Time Series analysis. Our custom function, "determine column type," was then applied to classify columns as either "Categorical" or "Numerical." Additionally, we utilized the "classifyAndNormalizeNumericFields" function _cite33 to further categorize and normalize numerical columns as either "Ordinal" or "Discrete."

Normalizing data is crucial for enhancing the training efficiency of algorithms, particularly those relying on gradients, like neural network architectures. It ensures that each feature contributes proportionally to predictions. This practice is essential for gradient-based methods as training on features with varying scales can lead to disparate step sizes during gradient descent.

In our code, discrete columns underwent Min-max Normalization using the 'normalizeDiscreteColumn' function, while ordinal columns were subjected to Rank-normalization via 'normalizeOrdinalColumn.' Notably, we opted not to normalize numerical columns with fewer than 32 unique values temporarily to maintain the integrity of specific columns like 'Month' and 'Hour' for later use. Categorical columns, having limited unique values, were encoded using One-hot encoding through 'encodeCategoricalColumns,' with 'Season' having a maximum of four unique values. The processed data was then stored in the 'encoded_dataset' object for easy access.

3 Feature Engineering

In the previous sections, some aspects of the dataset have been analyzed both visually and numerically. This section aims to provide a deeper explanation of the transformation of the features to fit the problem better to the machine learning models.

It is important to specify that the field of Business Analytics is characterized by continuous evolution, resulting of overlapping aspects in certain sections. For instance, it has been indicated by some resources that normalization, scaling, and encoding should be addressed within the Feature Engineering section, despite other resources suggesting that these tasks should be completed beforehand. The schema provided in the Coursework sheet will be adhered to.

In this section's details, insights derived from our Time Series analysis were explored. Subsequently, two novel features, namely 'Peak Hour' and 'Hour Bins,' were introduced. The 'Peak Hour' feature is binary, assuming a value of 1 during peak hours, which span from 7 AM to 9 PM, and 0 for all other time intervals. On the other hand, 'Hour Bins' is a feature that encompasses a range of integer values, strategically segmenting the hours into five distinct sections: morning, rush hours, midday, evening rush hours, and night.

That we tried to address the non-normal distribution observed in our target variable by implementing a logarithmic transformation. However, during the analysis, it was found that some models underperformed when using the transformed variable. Therefore, we ultimately decided not to proceed with the transformation and maintain the target variable in its original form for further analysis.

Recognizing the absence of bike rentals during inactive days in the 'Functioning Day' variable, we identified the redundancy of such data. Consequently, we proceeded to construct a new dataset named 'No FD dataset.'

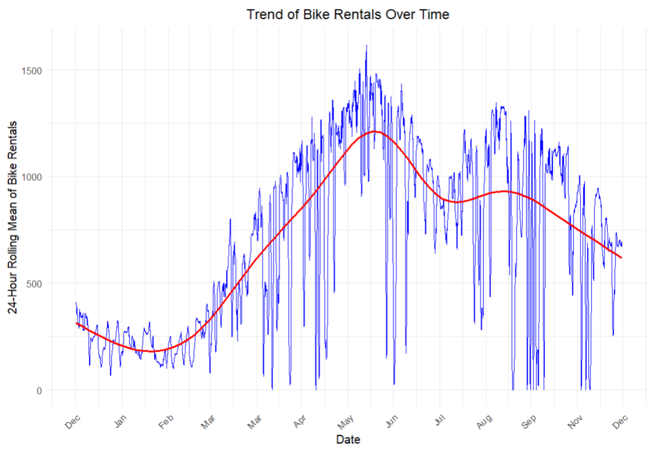


Figure 7: 24h Rolling Mean with trend line

All records where ’Functioning Day’ had a value of ’No’ (encoded as 1) were excluded, along with the removal of the column itself. Additionally, we established another dataset named ’SD No FD dataset,’ incorporating special days in the ’Holiday’ column that were not initially considered. This encompassed notable occasions such as Valentine’s Day, Blackpink Concert 2018, Woman’s Day, April Fool’s Day, among others [9] [10].

Additionally, the ’No DEW dataset’ was introduced to address the multicollinearity concerns identified in the Correlation Matrix. This was achieved by eliminating the ’Dew Point Temperature’ column strategically. The removal of this column aims to tackle multicollinearity issues and reduce the dimensionality of the dataset. The reduction in dimensionality offers several advantages for our Machine Learning models. By eliminating irrelevant features, the performance of the models is enhanced, complexity is reduced, and focus can be shifted on more relevant aspects. Furthermore, this approach mitigates the risk of overfitting, reduces computational costs, and optimizes the training time. It is worth noting that reducing dimensionality contributes to improved model explainability, thereby enhancing the model’s ability to effectively generalize on new and unseen data.

In alignment with the principle articulated by the eminent English logician Occam, who stated, ’Plurality should not be posited without necessity’ [11], an additional dataset known as the ’No Weak dataset,’ has been crafted, excluding columns with weak correlations (less than 0.1). In this context, the following columns omitted: ’Day of Week,’ ’Day,’ ’Holiday,’ ’isWeekend,’ and ’Hour Bins.’

Furthermore, a supplementary dataset, termed the ’No FD DEW dataset,’ was formulated by amalgamating attributes from both the ’No FD dataset’ and the ’No DEW dataset.’ From the original dataset, six distinct datasets emerged: the encoded dataset, No FD dataset, SD No FD dataset, No DEW dataset, No Weak dataset, and No FD DEW dataset. These datasets underwent meticulous preparation to facilitate training and testing of Machine Learning models, adhering to an 80-20 split ratio for this purpose.

4 Time Series Analysis

In this section, an attempt was made to gain further insight by applying well-known time series operations. The first step involved creating a new object called TS_dataset, which was a copy of the NonClean_dataset. This allowed for manipulations and analyses without impacting the original dataset’s integrity. Next, the date and time data were converted into the POSIXct datetime format, and the ’Date’ and ’Hour’ columns were merged into a single ’DateTime’ column. Subsequently, the dataset was restructured by removing the original ’Date’ and ’Hour’ columns and reordering the entries chronologically based on the new ’DateTime’ column. To gain additional insights, the bike rental data was aggregated by hour, day, and week. A notable technique employed was the calculation of a 24-hour rolling average, which effectively smoothed out short-term irregularities and highlighted longer-term trends in the data. Graphical analysis included various plots, with particular emphasis placed on a line graph that incorporated both a rolling mean and a trend line (Figure 7).
This visualization effectively highlighted the overarching trends in bike rentals over time. Then we applied the Dickey-Fuller test to check if the time series data is stationary. We returned a p-value less than 0.01, meaning that out series data is stationary. A key aspect is the test statistic: more negative values suggest a likelihood of stationarity. The results are interpreted using a p-value, with values less than a chosen significance level (often 0.05) indicating stationarity. The DF test focuses on one type of non-stationarity and should be interpreted within the broader context of the time series analysis [6].
Further, analyses involving Autocorrelation (ACF) and Partial Autocorrelation (PACF) were conducted to determine if past values within the bike rental series had predictive power over future values (Figure 8 and 9). ACF measures the correlation between a time series and its lagged versions over a range of time intervals. In

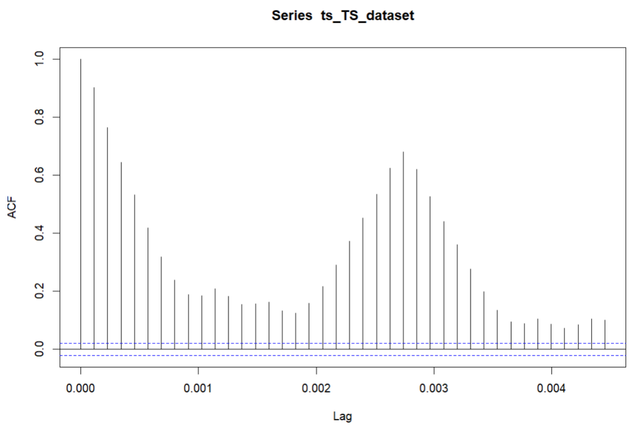


Figure 8: Autocorrelation

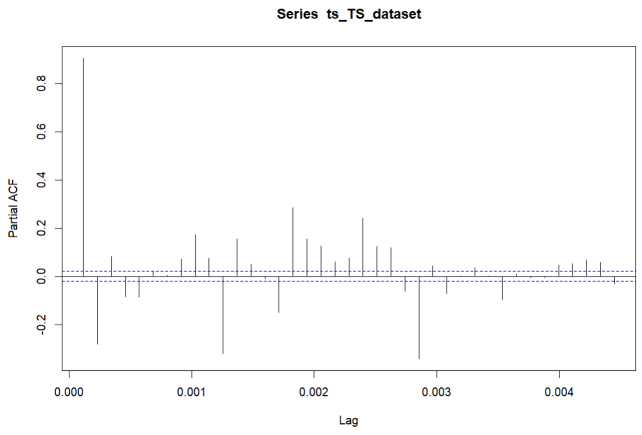


Figure 9: Partial Autocorrelation

the ACF plot, each bar represents the strength and direction of correlation at different lags. PACF shows the unique correlation between an observation and its lag, after accounting for the influence of intermediate lags. It's crucial for identifying specific lags that significantly impact a time series [7].
We developed a simple linear trend model to analyse the long-term trend of bike rentals. The model revealed a gradual increase over the observed year (The continuous red line in Figure 10). We then proceeded on creating a Heatmap [8] to have an easily interpretable format to visualize the bike rental with colour intensity indicating the magnitude of the measured variable (Figure 11). With the way the Heatmap is organised we can clearly identify recurring patterns, such as peak hours or days. We noticed that in the months of May, June and July we registered the highest amount of bikes rented as we can see from the bright yellow colour into the Heatmap image. Usually, the Heatmap is also useful to identify sudden spikes and anomalies, however we can clearly see that that's not the case here. The deep blue lines that we can notice into the image even into the previously mentioned months, represents Non Functioning days that took down the usage to 0 in these days, resulting in sudden drops in usage, thus we do not classify them as anomalies. This easy way of visualizing the data simplifies the usual plot complexity by representing data through colours, making it easier to derive insights without getting overwhelmed by numbers.
Insights gained with the Time series analysis:

- People tend to rent bicycles much more frequently during daytime hours than at night throughout the year.
- While the total number of bike rentals changes with the seasons, some core trends stay consistent. Notably, the highest demand periods remain constant in all seasons, occurring at 8 am and 6 pm.
- The trends are often present during working days. We can make the assumption that office workers are the slice of the population that mostly influences bike rental. This confirms our claims on Bike Rental made in the introduction, saying that the bikes are often used as an alternative way of commuting during rush hours to avoid traffic.

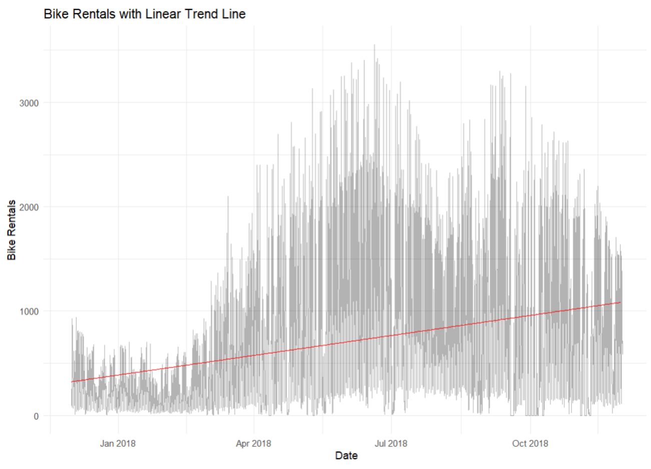


Figure 10: Yearly Trend

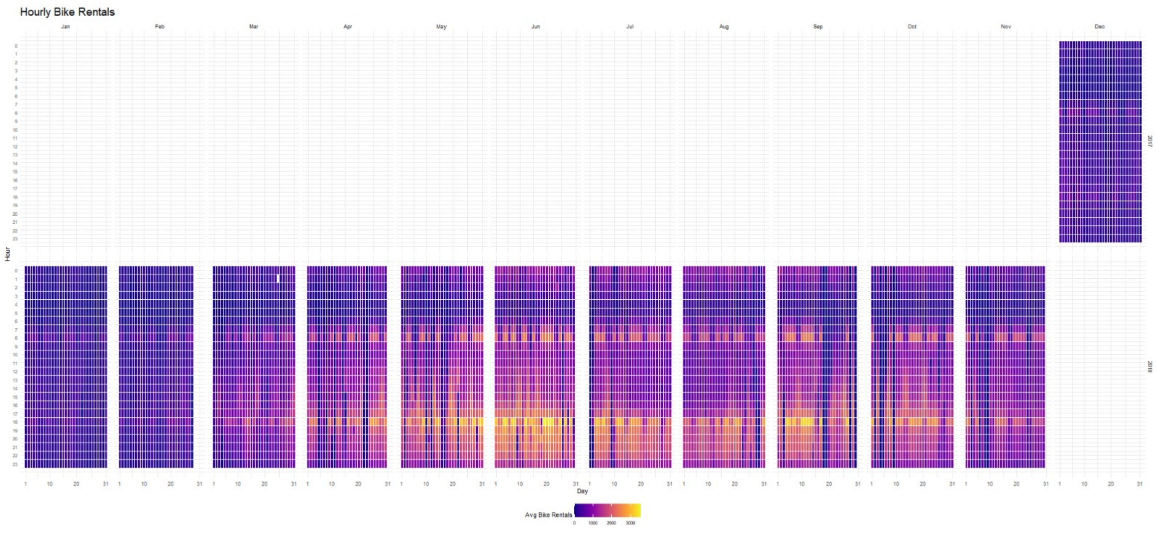


Figure 11: Yearly Heatmap of Bike Rental

We unfortunately have only one year worth of data. To make a complete Time Series analysis we could have also implemented detrending techniques and other ideas that we had (e.g. analyse the performance of the model Prophet via transfer learning). Sadly, it's difficult to isolate the trends when you have less than 2 years (cycles) of data. Many known R functions to analyse Time Series do not work with our dataset, meaning that we lack the opportunity to explore this section even further. Although we could have implemented resampling and over-sampling techniques to grow the dataset, we decided not to in order to preserve the real-world aspect of our data.

5 Modelling - Machine learning Algorithms Implementation

5.1 Didem's Model: XGBoost

I have successfully implemented two distinct algorithms: Lasso and XGBoost models. Furthermore, as part of my research, with CRISP-DM method specifically, I dedicated significant effort to privately updating and refining my own unique model. This involved a exploration and modification of various components, including feature selection techniques, regularization methods. Also in the appendix section, I have added the references that I used for the model and the code.

5.1.1 Model Selection

XGBoost, or Extreme Gradient Boosting, is a popular and powerful machine learning algorithm that excels in predictive modeling. With the ability to assess feature importance and its widespread adoption in competitions and real-world applications, XGBoost is highly regarded in the field of machine learning. One of the biggest reason for me to choose is The XGBoost algorithm's ability to incorporate gradient boosting along with its scalability for handling large datasets. This approach allowed me to explore the impact of different tree depths on the model's performance and assess its ability to capture complex relationships within the data.

5.1.2 Model Evaluation

Given that one of my team members was also working on the same task, I made the decision to focus on a specific hyperparameter, namely "max_depth," which is listed In the 'params' for tuning purposes. By limiting the hyperparameter tuning to "max_depth," I aimed to ensure a fair and meaningful comparison between our respective models' evaluations and result swith one of my team member.

After loading the necessary packages, "xgboost" and "caret", which provide functions for XGBoost modeling and data preprocessing, train_xgboost function is defined to train an XGBoost model. It takes predictor variables (X), target variable (y), model parameters (params), training ratio (train_ratio), and the number of boosting rounds (nrounds) as input. The training and test data is converted to the DMatrix format required by XGBoost. The XGBoost model is trained using the xgb.train function, with the provided parameters, training data, and the number of boosting rounds, which is a constant. A loop iterates over the datasets and performs the following steps for each dataset: Separates the predictor variables (X) and the target variable (y). Prints the evaluation metrics for the current dataset. Iterates over the max_depth values and performs the following steps for each max_depth. - Sets the XGBoost parameters (params) based on the current max_depth and the specified learning_rate. -Calls the train_xgboost function to train the XGBoost model and evaluate it on the test data. -prints the evaluation metrics for the current max_depth value.

The code essentially trains XGBoost models on different datasets with various max_depth values and calculates the evaluation metrics for each combination. The evaluation metrics are printed for each max_depth value and dataset.

5.1.3 Hyperparameter Tuning and Model Strength and Weakness

While XGBoost greatly supports large datasets, it's hyperparameter allows a range of parameters that could potentially improve the model's performance, in this particular case, the decision to just the tune the "max_depth" parameter was made to facilitate a more focused comparison and evaluation process. Use hyper parameters are: Learning Rate, controls the contribution of each tree, Max_depth ,sets the maximum depth, nrounds is number of the rounds. Learning rate and nrounds kept as a constant, which are respectively 0,1 and 100. Which lead to a weakness of the model: insufficient complexity and oversimplified hyper parameters. Even though the Grid Search applied, only using max_depth did not create a sufficient score.

Encoded dataset: Max depth = 5, Best R-Squared: 0.702155 No FD dataset: Max depth = 3, Best R-Squared: 0.6750636 SD No FD dataset: Max depth = 3, Best R-squared: 0.6751853 No Dew dataset: Max depth = 3, Best R-squared: 0.6557857 No Weak dataset: Max depth = 5, Best R-squared: 0.3801834 No FD DEW dataset: Max depth = 3 Best R-squared: 0.6933624

XGBoost models were evaluated on various datasets with tuned ”max_depth” hyperparameters, achieving optimal R-squared scores for Encoded dataset with Max depth = 5 and Best R-Squared value 0.702155. However, further exploration of hyperparameter combinations is needed to improve performance due to potential oversimplification.

5.2 Didem’s Model: Lasso Regression

5.2.1 Model Selection

Lasso Regression provides benefits such as automatic feature selection, regularization, and applicability to high-dimensional datasets. However, it has limitations in terms of feature selection, biased coefficient estimates, and limited ability to capture complex relationships. Apart from its advantages and disadvantages I also wanted to compare the model’s ability with one of my team mate’s models which applied Ridge and ElasticNet models.

5.2.2 Model Evaluation

After loading the necessary package, glmnet package, which is used for Lasso regression, fit_lasso_model function is defined to train a Lasso model. It takes a dataset, target variable, training ratio, and an optional lambda grid as input.

Within the function, the dataset is split into training and test sets based on the training ratio. The predictor variables are standardized using the scale function. A lambda grid is defined, which specifies the range of lambda values to consider. The function performs grid search using cross-validation (cv.glmnet) to find the best lambda value. The Lasso model is fitted using the training data and the best lambda value. The predictor variables in the test set are standardized using the mean and standard deviation from the training set. The function returns a list containing the trained Lasso model, standardized test data, the actual target values from the test set, and the best lambda value. Within the function, metrics such as mean squared error (MSE), root mean squared error (RMSE), R-squared, and mean absolute error (MAE) are calculated. The evaluate_lasso_model function is defined to make predictions on the test set using the trained Lasso model and calculate evaluation metrics. The function returns the evaluation metrics as a result. A loop iterates over the datasets and performs the following steps for each dataset: - Fits the Lasso model using the fit_lasso_model function. -Evaluates the Lasso model using the evaluate_lasso_model function. -Stores the evaluation results in the corresponding lists.

Overall, the code trains Lasso regression models on different datasets, evaluates the models using various evaluation metrics, and prints the results.

5.2.3 Hyperparameter Tuning and Model Strength and Weakness

The model has two hyperparameters which are lambda_grid and alpha. Lambda_grid is the range of lambda values to be tested in the grid search and alpha is the he elastic net parameter which is fixed to 1, indicating Lasso regression. On the model Instead of giving values to lamba I wanted the model to find it since I know that the model might not be the best one to choose since features are correlated and also Lasso is sensitive to large and or small values of lambda. These two features are the weakness of the models.

Encoded dataset: Best lambda for the dataset = 0.1484968, Best R-Squared: 0.6284163 No FD dataset: Best lambda for the dataset = 1e-05, Best R-Squared: 0.5958205 SD No FD dataset: Best lambda for the dataset = 1e-05, Best R-Squared: 0.6088027 No Dew dataset: Best lambda for the dataset = 0.03430469, Best R-Squared: 0.5986248 No Weak dataset: Best lambda for the dataset = 4.328761e-05 , Best R-Squared: 0.6134591 No FD DEW dataset: Best lambda for the dataset = 0.2056512 , Best R-Squared: 0.5859597

XGBoost models were evaluated on various datasets with tuned ”lambda” hyperparameter, achieving optimal R-squared scores for Encoded dataset with lambda = 0.1484968 5 and Best R-Squared value 0.6284163. Removing specific features showed that even though with the best lambda that model can have, the relationship between the features are still important. Because of the feature relationship, Ridge regression, or Elastic Net might provide better evaluation.

5.3 Habiba’s Model – Random Forest Regressor

5.3.1 Model Selection

The Random Forest algorithm is a suitable choice for the Seoul Bike Rental Prediction task as its inherent ability is able to handle non-linear relationships and interactions between features without the need for explicit transformation. Its ensemble nature, aggregating decision from a variety of decision trees, enables it to become robust against overfitting whilst maintaining first-class accuracy. Considering the complexness of Random Forest, this makes it generally higher than the simple model which is the same as linear regression but offers significant advantages in interpretability over more complex, black-box models like deep neural networks. The interpretability comes from the fact that it's simpler to understand the importance of each feature contributing to the decision, thanks to the layout of the decision trees.

5.3.2 Hyperparameter Tuning

Hyperparameter tuning was conducted to optimize the Random Forest model's performance. The tuning process involved varying the number of trees (ntree) and the maximum number of features considered for splitting a node (mtry). The selected values for the ntree and mtry from the hyperparameter tuning process demonstrates a consideration of both model complexity and performance. Different datasets derived from the original data found their best parameters in varied ranges, signalling a much more tailored approach towards the optimization of each model available. The hyperparameter tuning process had extremely benefitted the model's performance and its ability to engage towards solving complex data – its training was carefully executed with extreme measure with tailored models for different data versions.

5.3.3 Model Training

The model was trained on six different versions of the dataset, which undergone various pre-processing stages, which takes into account the removal of certain features and encoding. Every single model has been trained and tested with its set of best-found hyperparameters, ensuring the training was optimized for each version of dataset. This tailored approach has suggested that there is a satisfactory amount of understanding towards the significance of the dataset-specific nuances in model training. Furthermore, taking into account Random Forest's inherent feature has been able to adapt to various types of data (numerical and categorical).

5.3.4 Model Evaluation

The evaluation metrics chosen to assess the Random Forest models were MSE, RMSE, MAE and R-squared. All these metrics provide a stable and comprehensive understanding of the model performance. MSE and RMSE Gives a sense of the average error magnitude, with RMSE being more prone to larger mistakes, therefore giving a sense that there is a more accurate depiction when higher errors are particularly undesirable. MAE Offers an easily interpretable metric representing the average absolute error across all instances. R-squared Gives a comprehensive understanding of how well measured the model's predictions are fitting towards the approximation of the real data, essentially indicating the explained variability by the model. Through the selection of these metrics a robust approach briefly reflects how well the model has been performing taking into account both accuracy and the fit of the model. The use of multiple metrics provides a holistic representation of the model's overall performance, considering various aspects of errors and fit. The comprehensive evaluation strategy provided a multi-faceted understanding of model performance, making sure that the selected models met the needs of accurate and reliable bike rental predictions.

5.4 Habiba's Model – K-Nearest Neighbours

5.4.1 Model Selection

The K-Nearest Neighbours (KNN) model is chosen for the Seoul Bike Rental Prediction task for its simplicity and effectiveness in capturing the non-linear relationships between features. KNN works by identifying the 'k' nearest data points in the feature space and predicting the output based on the majority vote or the average of the nearest neighbours. Considering its simplicity makes it a good baseline model to identify and comprehend underlying data patterns without assuming much about data distribution. KNN's reliance on the local neighbourhood makes it sensitive to the choice of 'k' and the distance metric used, making these critical hyperparameters in the model's performance.

5.4.2 Hyperparameter Tuning

In this approach, 'k' was the primary hyperparameter tuned for the KNN model. The different values of 'k' were tried to understand its effect on the model performance. The decision to choose 'k' has affected the bias-variance trade-off of the model; smaller 'k' values can lead to overly generalized models (high bias). The best 'k' was selected based on the evaluation metrics over the validation set, ensuring the model is neither overfitting not underfitting. Its performances get impacted significantly on the choice of 'k' and the distance metric, with hyperparameter tuning playing a crucial role in optimizing these aspects

5.4.3 Model Training

The KNN model, was trained on several pre-processed versions of the dataset produced which reflects on the different featured engineering strategies. This has included datasets with removed features based on correlation analysis and feature importance, as well as different encodings for categorical variables. Each dataset version was trained with its specified 'k', reflecting the hyperparameter tuning phase's results. The training process in KNN is generally fast as it involves storing the feature vectors and their corresponding labels; however, it is critical to ensure that the data is normalized or standardized so that all features contribute equally to the distance calculations.

5.4.4 Model Evaluation

The KNN Models were evaluated using the following metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), R-Squared and Root Mean Squared Error (RMSE). These metrics collectively offer a comprehensive view of the model's performance, considering different aspects of error and fit. The model's effectiveness was demonstrated across various versions of the dataset offered, which every single part of the dataset representing different pre-processing solutions. The evaluation process has confirmed the model's ability to accurately predict bike rentals reasonably well, with an opportunity to further stabilize and enhance the bike rentals by introducing more nuanced feature engineering and hyperparameter tuning. Overall, the KNN model's application showed a level of satisfactory agreement and understanding of the fundamentals of model training and evaluation. The KNN model serves as a straightforward yet powerful method for the Seoul Bike Rental Prediction task.

5.5 Matthew's Model - Support Vector Machine Regressor

5.5.1 Background

The Support Vector Machine (SVM) Regressor is a supervised machine learning algorithm that can be a suitable choice for analysing the Seoul Bike dataset. The SVM Regressor aims to find the hyperplane that maximizes the margin between the closest data points of the target values. The optimization problem involves finding the optimal set of coefficients and the bias term that defines the hyperplane. To optimise the performance of SVM Regressor, a grid search with cross-validation is done with suitable hyperparameters. Hence, the metrics used in the model training are MAE, RMSE and R-squared.

5.5.2 Strength & Weakness

First of all, SVM Regressor demonstrates decent performance in dataset with a number of features, which is suitable for the Seoul Bike dataset that has more than 13 features, since the kernel function is SVM Regressor can transform the data input into the required form. Besides SVM Regressor is robust to outliers and has excellent generalisation capability given that SVM Regressor is aiming to find the best decision boundary, which helps to avoid overfitting. Furthermore, SVM Regressor is able to capture complex and high-dimensional patterns in the data, by using different kernel functions.

However, SVM Regressor is sensitive to the hyperparameter tuning, especially kernel function. If it is not properly selected, SVM Regressor is unable to deliver optimal performance. On the other hand, SVM regressor can be computationally expensive, especially for large datasets. Essentially, the SVM Regressor would consider all the data for finding the hyperplane, which leads to high long computation time when the number of samples and features grows.

5.5.3 Hyperparameters

SVM Regressor offers various kernel functions, for instance linear, polynomial, radial basis function (RBF), and sigmoid. These kernels enable capturing non-linear relationships in the data by mapping it to a higher-dimensional feature space. A prior test is conducted to choose the best kernel function to reduce the experiment time for grid search in the later stage. The experiment showed that the RBF demonstrated outstanding R-squared performance (0.85) compared to the other kernel functions, polynomial (0.65) and sigmoid (-7.3). Therefore, RBF is chosen as the primary kernel function.

Besides, one of the major concerns of training a machine model is to generalise the trend and minimise the likelihood of overfitting. SVM regressor offers cost which is a penalty parameter for misclassified data points. It controls the balance between achieving a smoother margin which may result in more misclassifications and minimizing the training error and potentially leading to overfitting. A smaller value of C allows for a smoother margin and more tolerance for misclassifications, while a larger value of C enforces a stricter margin and aims for more correct training data. To achieve a high R2 score, value 10, 20, 40 are considered in the grid search.

On the other hand, SVM regressor also provides shrinking heuristic as a hyperparameter which is an optimisation technique to accelerate the training process. It finds and excludes certain training samples that are not likely to affect the decision boundary significantly. Instead of considering the entire training dataset at each iteration, the shrinking heuristic selectively updates the model only with the samples close to the decision boundary. This reduces the computational complexity and speeds up the training process. The boolean value is being tested in the grid search.

Furthermore, epsilon defines the margin of tolerance within which errors are considered negligible. Any prediction falling within this margin is considered accurate and does not contribute to the loss function. Epsilon helps control the trade-off between achieving a tight fit to the training data and allowing some level of error tolerance. A smaller epsilon value leads to a stricter fit, while a larger value allows for more errors within the tolerance margin. Value of 0.05, 0.1 are selected for the grid search.

RBF function is selected as the function kernel in the model training, and gamma parameter plays an important role to affect the model performance. Gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. Value of 1/4, 1/8, 1/16 are considered for the grid search.

5.5.4 Evaluation

SVM Regressor achieved excellent results after grid search with cross-validation. The best result was achieved using the following parameter values: cost = 40, epsilon = 0.05 and gamma = 0.0625 and shrinking = True. The best performance are achieved with the encoded dataset with the following metrics: Average MAE: 102.76, Average RMSE: 177.67 and Average R-squared: 0.9239. From the result, the higher the cost, the better the performance it achieves. While smaller the epsilon, the better the result. On the other hand, it is expected that setting shrinking to True would outperform setting it to False. Overall, the grid search is able to optimise the performance of SVM regressor.

5.6 Matthew's Model - Random Forest Regressor

5.6.1 Background

The Random Forest Regressor is a supervised machine learning algorithm which creates an ensemble of decision trees. Each tree makes a prediction independently, the final value are calculated by averaging the total prediction. Generally, some factors will affect the intention of people renting bikes, for example, time, temperature, snow and holidays. Therefore, Random Forest Regressor seems to be a sensible choice for outlining the important factors to make predictions for this dataset. Hence, the Random Forest Regressor employs various hyperparameters to optimize its performance. These hyperparameters include ntree, mtry, maxdepth, and nodesize.

5.6.2 Strength & Weakness

First of all, Random Forest Regressor reduces the likelihood of overfitting which happens in the decision tree. Since Random Forest is averaging the result from decision trees, it can produce a more reliable prediction. Secondly, Random Forest indicates the relative contribution of each feature in the ensemble's predictions, outlying the key features variable. This would help in feature selection and minimise the impact of the least important feature towards the prediction. However, Random Forest is memory expensive. It requires more memory compared to other algorithms especially if the number of trees and the depth of trees are high. This also leads to a computationally expensive issue where Random Forest Regressor can take a longer training time with large datasets and large numbers of trees. Last but not least, unlike decision trees, random forests are difficult to interpret. The ensemble of the decision tree is complicated to understand the relationship between the model and the target variable.

5.6.3 Hyperparameter

The hyperparameter ntree determines the total number of decision trees in the random forest. Each decision tree is trained on a different randomly selected subset of the training data, introducing randomness and diversity in the model. Increasing the number of trees in the ensemble can lead to improved accuracy and robustness of the model. With more trees, the random forest can capture a greater range of patterns and relationships in the data, reducing the risk of overfitting. The ensemble nature of the random forest helps to average out biases and errors in individual trees, resulting in more reliable predictions. In this study, values of 500 and 1000 were considered.

Furthermore, the mtry hyperparameter controls the number of randomly selected features which is considered at each split during tree construction. The hyperparameter mtry in the Random Forest Regressor refers to the number of randomly selected features considered at each split when constructing an individual decision tree within the ensemble. Since the maximum number of features in the dataset is 16, the values explored were 6, 8, 12, and 16.

Moreover, a decision tree grows by recursively partitioning the data based on selected features and splitting criteria. The depth of a tree refers to the number of levels from the root node to the deepest leaf node. How deep the tree can grow during the training process is controlled by limiting the maxdepth. For example, setting a lower maxdepth value constrains the tree to be shallower, resulting in simpler and less complex models. Shallower trees are less likely to capture intricate or fine-grained patterns in the data. Values of 5, 10, and 15 were tested in this experiment.

Additionally, the nodesize hyperparameter determines the minimum number of samples required to split an internal node in a tree. It controls the granularity of the tree structure and affects the tree's ability to capture both general trends and specific patterns in the data. When growing a decision tree, at each internal node, the algorithm evaluates different splitting criteria to determine if a node should be further divided into child nodes. Values of 1, 2, and 4 were investigated.

Grid search was conducted to systematically evaluate different combinations of these hyperparameters and identify the optimal configuration that maximizes the Random Forest Regressor's performance on the Seoul Bike dataset. This approach allows for fine-tuning of the model to strike a balance between accuracy, generalization, and computational efficiency.

5.6.4 Evaluation

Random Forest Regressor achieved excellent results after grid search. The best result was achieved using the following parameter values: ntree = 1000, mtry = 12, maxdepth = 5, and nodesize = 1. The best performance are achieved with no dew point dataset with the following metrics: Average MAE: 102.76 Average RMSE: 177.67, Average R-squared: 0.9239. From the result, the greater number of trees, the better the result. For mtry, not all features are needed for model training, 12 features are selected for the best result. On the other hand, 5 max depth is found to be the best parameter to limit the tree depth. The minimum nodesize is 1 to split the internal node in a tree. Overall, the grid search is able to optimise the performance of Random Forest regressor.

5.7 Massimiliano's Model - XGBoost

The eXtreme Gradient Boosting (XGBoost) is a Machine Learning algorithm that is used for regression in this case study. It's an ensemble method that combines the predictions for a multitude of models (Weak Classifiers) that in this case are trees. The framework that supports all these predictions is gradient boosting. For the XGBoost, after a series of predictions, it computes the errors of the classifiers used. These errors are used to implement new models that support the existing classifiers to improve performance and correct their errors. In short, we start by using a single tree model and iteratively add more tree models to compensate the errors. The gradient descent is the optimization algorithm used to minimize the loss (difference between predicted values and actual values). Hence, the new trees added to support the existing ones are selected with Gradient descent. Furthermore, this model uses a technique called "Pruning" that simplifies the trees by removing branches that are no longer useful for the predictions and another technique called Regularization that helps to prevent overfitting (Overfitting: when a machine learning model can only predict correctly data that has previously seen and cannot generalize well with unseen data, effectively it has memorized the training data). I chose the XGBoost for a multitude of reasons. For a regression analysis, due to its ensemble nature and gradient boosting, this model can learn various complex and non linear relationships, where other simpler models may struggle, resulting in a higher accuracy in predictions. This model can easily handle large datasets because of its scalability, making it a sustainable choice for a regression task in terms of computational cost. This model, however, does not come without his limitations: We have a lack of interpretability given the model complexity. Even thou some functions in R can provide a graphical representation of the relationships and trees created, the image plotted is undiscernible (After some trials, I decided not to plot a chart for the model's interpretability). Another weakness of this model is the Hight risk of overfitting the data given and the computationally intensive nature of the algorithm. To further improve this model I implemented a Grid Search that fine tunes the hyperparameters by making combinations and trials, returning the best result on the R-squared metric. The hyperparameters tested are:

- Eta (Learning Rate): this parameter controls the contribution of each tree. Tested with values of 0.1, 0.15, 0.2, 0.25, 0.3 . Smaller values make the model more robust.

- Max depth: It's the hyperparameter that sets the maximum depth of the trees. The deeper the tree, the more complex it is, but it can lead to overfitting. Tested with values of 6, 9, 10.
- Min child weight: this hyperparameter defines the minimum sum of weights of a child on all the observations, this parameter is useful to reduce overfitting in larger trees. Tested with values of 1,3, 4 .

After the Grid Search, the best parameters found for the XGBoost model are:

- Encoded dataset: eta= 0.1, Max depth = 10, Min child weight = 1; R-squared increase 0.0547, Best R-Squared 0.99849.
- No FD dataset: eta= 0.1, Max depth = 9, Min child weight = 3; R-squared increase of: 0.04881, Best R-squared : 0.995312
- SD No FD dataset: eta= 0.1, Max depth = 9, Min child weight = 4; R-squared increase of: 0.05129, Best R-squared: 0.99626
- No Dew dataset: eta= 0.1, Max depth = 9, Min child weight = 3; R-squared increase of: 0.05073, Best R-squared: 0.99593
- No Weak dataset: eta= 0.1, Max depth = 6, Min child weight = 4; R-squared increase of: 0.02926, Best R-squared: 0.92996
- No FD DEW dataset: eta= 0.1, Max depth = 9, Min child weight = 3; R-squared increase of: 0.05071 Best R-squared: 0.99464

The Gris search had a great impact on the model's performance, increasing the performance of the R-square metric of 0.05 on average. The best performances are on the "encoded dataset" with these metrics: RMSE: 25.290 , MAE: 16.16, MSE: 639.6, R-squared: 0.99849.

5.8 Massimiliano's Model - Conditional Inference Tree

The Conditional Inference Tree (CI Tree) Regressor is a part of the "party" package in R. This algorithm is a type of decision tree with some differences. The decision trees rely on heuristics for splitting the nodes, the CI Tree uses statistical testing. The algorithm works with a recursive statistical testing to determine the links between features and target variables. Hence, the variable with the smallest p-value is the one selected for the binary split of the node. To prevent overfitting, the algorithm has a stopping mechanism that helps creating a tree structure that stops when no meaningful associations between variables can be found. This means that there is no need for pruning, unlike the standard decision Tree, to handle such problems. The CI Trees are easier to interpret than other more complicated models and can also handle non linear relationships. Unlike XGBoost, the CI Tree is easier to implement and interpret. The CI Tree however cannot perform as well as an ensemble method on large datasets and cannot handle very complex patterns. To further improve the model's performance, I applied 2 ways of hyperparameter tuning: Grid Search and Bayesian Search. The Bayesian Search is an optimization technique that uses Bayesian optimization techniques. The hyperparameters tested is mincriterion. The "mincriterion" is the parameter that regulates the threshold for the value p used to decide splits. With lower value, the tree can grow bigger and have even more splits and lower values of "mincriterion" result in simpler trees. If the values of R-squared after the tuning is lower than the original hyperparameters, the originl R-squared is reported as the best. After the Grid Search and Bayesian Opti mization the best parameters found for the CI Tree model are:

- Encoded dataset
  - Grid Search mincriterion = 0.8 R-squared: 0.89758;
  - Bayesian Optimization: mincriterion = 0.801564 R-squared of: 0.8499;
  - Best performance recorded (R-squared): 0.92262
- No FD dataset
  - Grid Search mincriterion = 0.8 R-squared: 0.89871;
  - Bayesian Optimization: mincriterion = 0.80661 R-squared of: 0.84971;
  - Best performance recorded (R-squared): 0.92398
- SD No FD dataset

  - Grid Search mincriterion = 0.8 R-squared: 0.89747
  - Bayesian Optimization: mincriterion = 0.83851 R-squared of: 0.84839
  - Best performance recorded (R-squared): 0.92273

- No DEW Dataset
  - Grid Search mincriterion = 0.8 R-squared: 0.87546
  - Bayesian Optimization: mincriterion = 0.8427 R-squared of: 0.8480
  - Best performance recorded (R-squared): 0.92172

- No Weak Dataset
  - Grid Search mincriterion = 0.8 R-squared: 0.85353
  - Bayesian Optimization: mincriterion = 0.80493 R-squared of: 0.81896
  - Best performance recorded (R-squared): 0.87162

- No FD DEW dataset
  - Grid Search mincriterion = 0.8 R-squared: 0.87585
  - Bayesian Optimization: mincriterion = 0.8063 R-squared of: 0.84671
  - Best performance recorded (R-squared): 0.92302

5.9 Vedat's Model: Ridge Regression Model

Ridge Regression is an appropriate choice when dealing with multicollinearity and expecting all predictors to contribute, offering the convenience of straightforward implementation. I used the functionalities of the glmnet (General Linear Models) package for Ridge Regression and Elastic Net models.

The train_ridge_regression function is designed to train a Ridge Regression model. It takes three input parameters: dataset, target, and lambda. The train_ridge_regression function trains a Ridge Regression model using a structured dataset that includes both the predictor variables and the target variable. The dataset parameter represents this dataset. The target parameter specifies the column name of the target variable, which the Ridge Regression model aims to predict. The lambda parameter is the regularization parameter used in Ridge Regression. It controls the level of regularization applied to the model to prevent overfitting.

The function returns a list with the trained Ridge Regression model, the original dataset used for training (test_set), and the column name of the target variable (target_column). The model element in the output represents the trained Ridge Regression model, which can be utilized for making predictions on new data or for further analysis. The test_set element corresponds to the original dataset employed for training the model, enabling additional examination if required. Additionally, the target_column element specifies the column name of the target variable in the dataset, serving as a reference for identifying the target variable used in the model.

The gri_search_ridge function performs a grid search to find the optimal hyperparameters for Ridge Regression. It takes four input parameters: dataset, target, lambda_values, and nfolds. During the grid search, the function evaluates different lambda (regularization) values specified in the lambda_values vector, which includes the values 0.001, 0.01, 0.1, and 1 to determine the best hyperparameters, the function employs cross-validation with a 5-fold setup.

In summary, the train_ridge_regression function is used to train a Ridge Regression model using a specified dataset, target variable, and regularization parameter. It provides the trained model, the original dataset, and the target variable column name as output. Then the grid_search_ridge function conducts a grid search to identify the best hyperparameters for Ridge Regression. It evaluates different lambda values (0.001, 0.01, 0.1, and 1) using a 5-fold cross-validation setup and returns an evaluation object with the optimal parameters and evaluation metrics which include mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE), and R-squared. evaluate_model_ridge_net function which takes two input parameters (predictions and observed_values), is used to evaluate the performance of Ridge Regression and Elastic Net Regression models. The code iterates over the train datasets, trains Ridge Regression and Elastic Net Regression models on each dataset and evaluates the performance of the models on the respective test datasets. The evaluation metrics for each model and dataset combination are printed, allowing for the assessment of the models' performance across various datasets.

According to results, it could be said that Ridge regression is underperforming and explains around 60% of the variance in the target variable. It seems to be model is underfitting, it can not capture the complexity of the underlying relationships present in the data. If we look at whether the assumptions of the Ridge Regression model are met, we can say that the assumptions such as independence and homoscedasticity are met, but the linearity assumption is not completely met, which affects the accuracy of the model's predictions.

5.10 Vedat's Model: Elastic Net Regression Model

Elastic Net Regression is well-suited for variable selection and high-dimensional data, providing the convenience of user-friendly implementation and making it accessible for researchers and practitioners seeking efficient and effective regression models. The train_elastic_net function is designed to train an Elastic Net Regression model. It takes three input parameters: dataset, target, lambda, and alpha. The train_elastic_net function trains an Elastic Net Regression model using a structured dataset that includes both the independent variables and the target variable. The dataset parameter represents this dataset. The target parameter specifies the column name of the target variable, which the Elastic Net Regression model aims to predict. The regularization parameter lambda controls the extent of regularization applied to the model, serving to prevent overfitting. The alpha parameter determines the balance between the L1 (Lasso) and L2 (Ridge) regularization components within the Elastic Net model. A value of 0 for alpha corresponds to Ridge Regression, while a value of 1 corresponds to Lasso Regression. By adjusting these parameters, the model's regularization, and the relative contribution of L1 and L2 regularization can be controlled to suit the specific requirements of the problem at hand.

The function returns a list with the trained Elastic Net Regression model, the original dataset used for training (test_set), and the column name of the target variable (target_column). The model element in the output represents the trained Elastic Net Regression model, which can be utilized for making predictions on new data or for further analysis. The test_set element corresponds to the original dataset employed for training the model, enabling additional examination if required. Additionally, the target_column element specifies the column name of the target variable in the dataset, serving as a reference for identifying the target variable used in the model.

The grid_search_elastic_net function performs a grid search to find the optimal hyperparameters for Elastic Net Regression. It takes four input parameters: dataset, target, lambda_values, alpha_values, and nfolds. During the grid search, the function evaluates different lambda (regularization) values specified in the lambda_values vector, which includes the values 0.001, 0.01, 0.1, and 1, additionally, the function explores various combinations of the mixing parameter (alpha) between 0 and 1, including 0, 0.5, and 1 to determine the best hyperparameters, the function employs cross-validation with a 5-fold setup.

In summary, the train_elastic_net function is used to train a Elastic Net Regression model using a specified dataset, target variable, and regularization parameter. It provides the trained model, the original dataset, and the target variable column name as output. Then the grid_search_ elastic function conducts a grid search to identify the best hyperparameters for Elastic Net Regression. As same with Ridge Regression, it evaluates different lambda values (0.001, 0.01, 0.1, and 1) and also alpha values (0, 0.5, 1) using a 5-fold cross-validation setup and returns an evaluation object with the optimal parameters and same evaluation metrics (MSE, RMSE, MAE and R-squared).

Based on the results, it can be concluded that both the Ridge Regression and Elastic Net Regression models performed similarly (underfitted), achieving comparable results. This similarity can be attributed to the fact that, during the grid search, the alpha value of 0 (corresponding to Ridge Regression) yielded better outcomes, resulting in both models obtaining similar results (both explain 60% of the data). Considering the similar performance of both models, it can be inferred that they provide satisfactory results within the linear model class.

Overall, the evaluation suggests that both the Ridge Regression and Elastic Net Regression models exhibit similar underperformance in explaining the data. The models struggle to capture the complexity present in the data, potentially due to the linearity assumption not being fully met. It may be necessary to apply higher-complexity models instead of linear models which may yield better results capturing the underlying relationships in the data.

6 Model Evaluations from Project Members:

6.1 Didem's Evaluation:

R-sqaured is chosen as the mutual agreement of the team member's to evaluate the models on the same parameter. I wanted to experiment the two different structured models. After coding and evaluating the models it is seen that the best performance for both the models is on the dataset Encoded dataset. The values of R-squared are for XGBoost 0.702155 and 0.6284163 for Lasso Regression. For the other datasets with missing features lead the models to perform worse according to the evaluation matrices. Based on my two models I can observe that linear model is slightly worse than decision tree structure. In order to make a fair comparison with Massimiliano's model, I kept the number of the hyperparameter which is tuned stable in order to compare to him to see the effects of the tuning. By keeping the Learning Rate, nrounds, min child weight constant and only chancing the max_depth, I had an opportunity to observe the performance of the high and low tuned models. The difference is approximately 28% on the best performance with the lack of hyperparameters which shows that only using max_depth to tune the model is not efficient enough on the comparison for the dataset. With the highest R_squared of the Lasso Model, it is seen that the model is not able to perform with high performance even with the optimal tuning. On the other hand with the comparison of Vedat's both model Ridge Regression, another Linear Regression model, and Elastic Net, combination of Lasso and Ridge models, it is also seen that the performance of that two models are similar to Lasso. On the highest performance both of the models' R-squared value was 0.6206 while Lasso performed 0.6123. As I also observed that not only in my models but also amongs the team members decision tree structured models are better than the linear models for the given dataset. As a conclusion, it is seen that the models that I have with not highly hyperparameter tuned or linearly based are performing but not succesfully and not given the expected predictions and stays underperformed. On the given project Random Forest and CI Tree had a significant performance. Therefore with better hyperparameter tuning these two models can perform well for the dataset.

6.2 Matthew's Evaluation:

There are 8 algorithms implemented in this project and 2 are shared between members. R-squared, which is a measure of how well the regression model fits into the data, is chosen to be the uniform metric to evaluate the models. For R-squared metrics, 0 implies that the regression model is poorly fit with the data; while 1 means the model is perfectly fit.

Overall, according to Figure 12, the most outstanding model would be XGBoost achieved by Max, with R-squared 0.9984, which implies that the model can precisely predict the rented bike trend. On the other hand, Didem may have proceeded the XGboost training with a different set of parameters which delivers a less accurate model. Furthermore, Random Forest model, the second best algorithm in this project, achieved nearly the same result across all the datasets conducted by Habiba and I. This result, where R-squared around 0.93, implies that Random Forest is able to combine multiple decision trees to make predictions. Hence, it is observed that CI Tree and SVM Regressor would be the third best algorithms for this project, also achieving excellent performance. In short, XGBoost, Random Forest, CI Tree and SVM Regressor demonstrates exceptional performance in predicting the bike rent, while KNN, ridge regression, elastics regression, lasso achieved satisfactory performance, with R-squared between 0.5 to 0.75.

In terms of dataset, most of the datasets are observed to have similar performance. The most notable dataset would be no weak correlation dataset which records a significant performance drop in most of the algorithms used except ElasticNet Regression. This suggests that the weak correlation variables are actually contributing towards the bike rental prediction should be kept in the dataset. Besides, most of the algorithms are achieving the best performance using the encoded dataset, this indicates that all the features shall be kept and encoded in order to deliver the best prediction. Yet, some algorithms are better using other dataset, for example, KNN achieved the best with the data set 3 with special days and no functioning days dataset. This implicated that it is also crucial to generate new dataset with justified features.

6.3 Habiba's Evaluation:

The evaluation of machine learning models developed by our group members was conducted rigorously using the coefficient of determination (R2) as the chosen evaluation metric for consistency and comparability across all models and datasets. Between my two model, Random Forest and KNN, Random Forest with the highest R2(0.9381) on Dataset 3(SD_No_FD_dataset), indicating a robust model fit. It was observed that Random Forest and XGBoost models generally offered superior performance, suggesting their robustness in handling the datasets provided.

6.4 Massimiliano's Evaluation:

The best performance of the CI Tree is on the dataset "No_FD_dataset" with and R-squared of 0. 92398 . The absence of the "Functioning_Day" feature boosted the performances of the CI Tree regressor while having no positive effect on the XGBoost model. The XGBoost model had a significant increase in performances over both my own CI Tree and my peers' models. The difference in performances is mostly due to the nature of the model. The XGBoost regressor is especially capable when working with datasets like the one in use, especially after the Feature Engineering changes that boosted the performances of all models. The XGBoost was fine tuned with grid search, showing a preference for the model in building deeper trees with a very low "eta" value. The low value of "eta" means that the model is taking a slow but robust training to reduce the risk of overfitting, also making conservative choices when growing deeper. The growth (max_depth) can model complex relationships this way while balancing out bias and variance given the combination of the two fine-tuned parameters.

The performance uplift of 27% to 30% form Didem's XGBoost model can be related to a series of factors. Possibly the choice of the hyperparameters, the xgboost package selection and other adjustments to the code have benefitted the R-squared metric.

6.5 Vedat's Evaluation:

In this evaluation, I aim to assess the performance and effectiveness of the models by analyzing their results. We chose R-squared as the evaluation metric for all our models because we wanted to assess the models' ability to explain the data. By using this metric, we aimed to make more accurate predictions for future outcomes based on how well the models captured the underlying patterns in the data. Based on the R-squared results, I can observe that linear models fall slightly behind in explaining the data, while models with the ability to capture complex relationships and structures (such as Random Forest, Decision Tree, KNN, SVM, and XGBoost) tend to perform better. This performance difference can be attributed to the complexity and relationships present in the data, causing linear models to lag behind in data explanation. Among the models with the ability to capture nonlinear structures, XGBoost stands out with an outstanding performance ranging from 0.93 to 0.99. Undoubtedly, XGBoost demonstrates the best ability to explain the data. However, another noteworthy aspect is the solid and robust performance of decision tree-based models (two different Random Forests and one Decision Tree), all achieving an average performance of approximately 0.93. This exemplifies the ease of implementation and the ability of decision tree structures to capture complex relationships and patterns within the data. Therefore, decision tree-based models continue to be a viable option.

As for the datasets, except for the 5th dataset (No_Weak_train), all others have improved the model performances. Among the remaining 5 datasets, the 1st and 2nd datasets have relatively contributed to more accurate model predictions by making the data more meaningful. Consequently, either the 1st (encode_train) or 2nd (No_FD_train) dataset can be chosen for the XGBoost model, while decision tree models (CI Tree and RF models) remain as an option.

7 Conclusion of the Analytics

The evaluations conducted by the team members focus on assessing the performance of different machine learning models using the coefficient of determination (R-squared) as the evaluation metric. The goal is to determine how well the models fit the data and make predictions.

The evaluations highlight the performance of various models, including XGBoost, Random Forest, CI Tree, SVM Regressor, KNN, and linear regression models. Decision tree-based models, particularly XGBoost and Random Forest, consistently demonstrate strong performance in capturing complex relationships in the data. Linear models tend to fall behind in explaining the data.

The impact of dataset preprocessing is also discussed, with encoded datasets generally yielding better results. The inclusion of relevant features improves model performance, while the absence of certain features can boost the performance of specific models. Hyperparameter tuning is found to have a significant impact on model performance, with fine-tuning leading to improved results.

In conclusion, decision tree-based models, such as XGBoost and Random Forest, are often the preferred choice due to their ability to capture complex relationships. Dataset preprocessing and feature selection play a crucial role in improving model performance. Hyperparameter tuning is essential for optimizing model results.

References

[1] Zou, Hui, and Trevor Hastie. "Regularization and Variable Selection via the Elastic Net." Journal of the Royal Statistical Society. Series B (Statistical Methodology), vol. 67, no. 2, 2005, pp. 301–320, http://www.jstor.org/stable/3647580.

[2] Seoul Bike Sharing Demand. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5F62R.

[3] Function inspired by the tutorials on: https://r-graph-gallery.com/262-basic-boxplot-with-ggplot2.html

[4] Function inspired by Prof. Alaa Marshan's code from laboratory 3, COMM053, University of Surrey.

[5] Inspired by the ggplot2 r-graph gallery: https://r-graph-gallery.com/223-faceting-with-ggplot2.html

[6] https://analyticsindiamag.com/complete-guide-to-dickey-fuller-test-in-time-series-analysis/

[7] https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/

[8] Inspiration from: https://r-graph-gallery.com/283-the-hourly-heatmap.html

[9] Source for special dates: https://www.timeanddate.com/holidays/south-korea/2018

[10] Source for special events: https://kpoppersguide.wordpress.com/2019/02/02/2018-kpop-calendar/

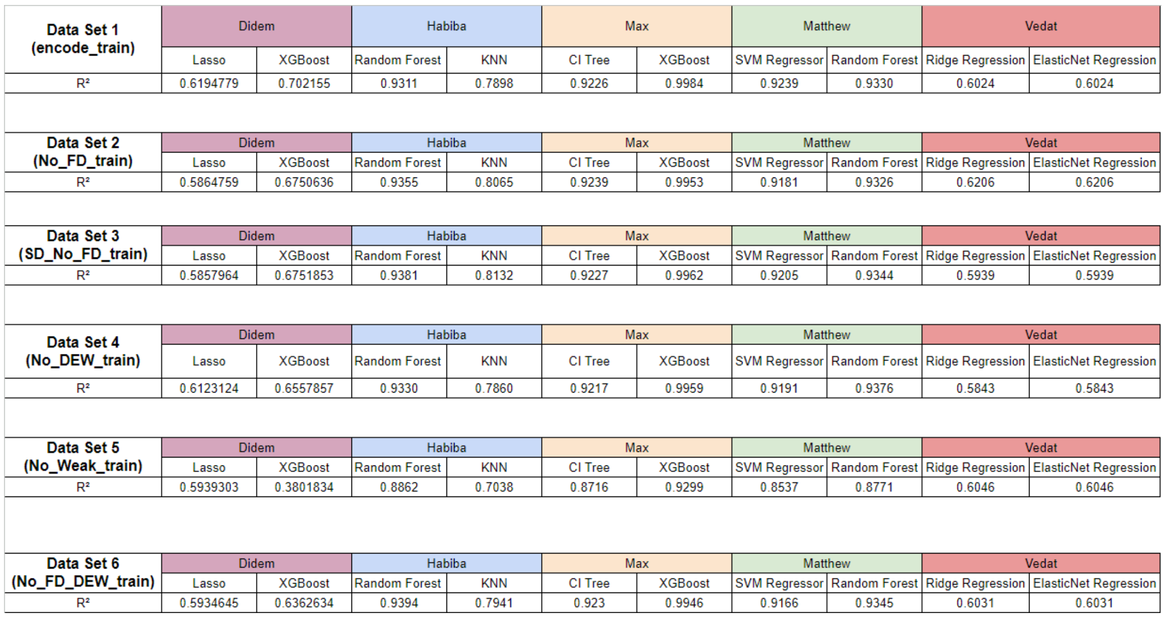
[11] Occam's Razor, William of Ockham, 1347.

[12] https://scikit-learn.org/stable/modules/generated/sklearn.linear$_{m}odel.Lasso.htmlhttps$: $//www.mygreatlearning.com/blog/understanding - of - lasso - regression/$

[13] https://towardsdatascience.com/hyperparameter-tuning-in-lasso-and-ridge-regressions-70a4b158ae6d

[14] https://xgboost.readthedocs.io/en/stable/parameter.html

[15] https://www.kaggle.com/code/prashant111/a-guide-on-xgboost-hyperparameters-tuning



Data Set 1 (encode_train)	Didem		Habiba		Max		Matthew		Vedat	
	Lasso	XGBoost	Random Forest	KNN	CI Tree	XGBoost	SVM Regressor	Random Forest	Ridge Regression	ElasticNet Regression
R^2	0.6194779	0.702155	0.9311	0.7898	0.9226	0.9984	0.9239	0.9330	0.6024	0.6024

Data Set 2 (No_FD_train)	Didem		Habiba		Max		Matthew		Vedat	
	Lasso	XGBoost	Random Forest	KNN	CI Tree	XGBoost	SVM Regressor	Random Forest	Ridge Regression	ElasticNet Regression
R^2	0.5864759	0.6750636	0.9355	0.8065	0.9239	0.9953	0.9181	0.9326	0.6206	0.6206

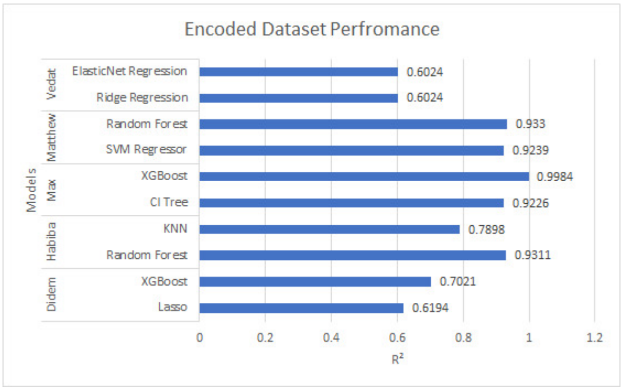
Data Set 3 (SD_No_FD_train)	Didem		Habiba		Max		Matthew		Vedat	
	Lasso	XGBoost	Random Forest	KNN	CI Tree	XGBoost	SVM Regressor	Random Forest	Ridge Regression	ElasticNet Regression
R^2	0.5857964	0.6751853	0.9381	0.8132	0.9227	0.9962	0.9205	0.9344	0.5939	0.5939

Data Set 4 (No_DEW_train)	Didem		Habiba		Max		Matthew		Vedat	
	Lasso	XGBoost	Random Forest	KNN	CI Tree	XGBoost	SVM Regressor	Random Forest	Ridge Regression	ElasticNet Regression
R^2	0.6123124	0.6557857	0.9330	0.7860	0.9217	0.9959	0.9191	0.9376	0.5843	0.5843

Data Set 5 (No_Weak_train)	Didem		Habiba		Max		Matthew		Vedat	
	Lasso	XGBoost	Random Forest	KNN	CI Tree	XGBoost	SVM Regressor	Random Forest	Ridge Regression	ElasticNet Regression
R^2	0.5939303	0.3801834	0.8862	0.7038	0.8716	0.9299	0.8537	0.8771	0.6046	0.6046

Data Set 6 (No_FD_DEW_train)	Didem		Habiba		Max		Matthew		Vedat	
	Lasso	XGBoost	Random Forest	KNN	CI Tree	XGBoost	SVM Regressor	Random Forest	Ridge Regression	ElasticNet Regression
R^2	0.5934645	0.6362634	0.9394	0.7941	0.923	0.9946	0.9166	0.9345	0.6031	0.6031

Figure 12: Model Results on Datasets



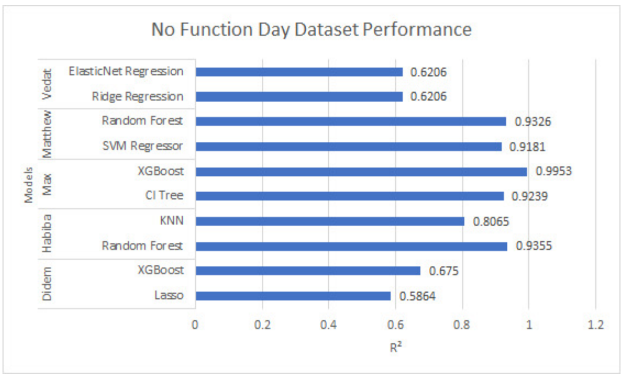
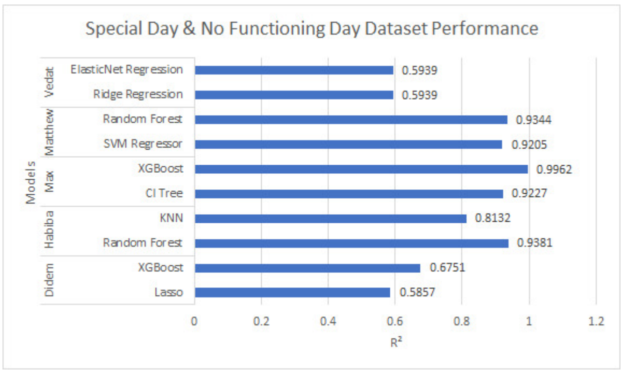


Figure 13: Model Performances for "Encoded" and " No Function Day" Dataset



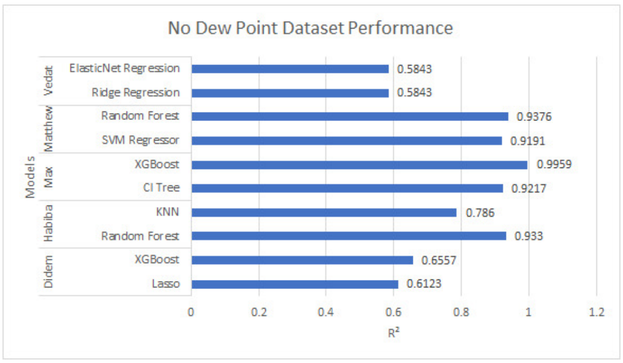


Figure 14: Model Performances for "Special Day & No Function Day" and "No Dew Point" Dataset

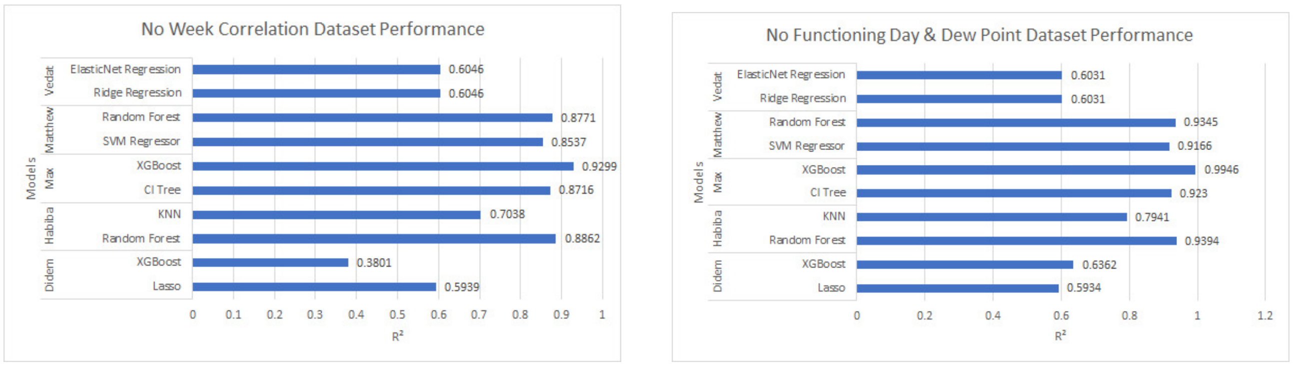


Figure 15: Model Performances for "No Weak Correlation" and "No Function Day & No Dew Point" Dataset

I hereby certify that I, Vedat Yasar, am the sole author of the tasks performed for the Practical Business Analytics course (COMM053).

I contributed to acquisition of data, analysis and interpretation of data and evaluation in the coding part. I actively contributed to the report writing, the transfer of the conducted work into the document, the proofreading process, and the collection of individual contributions to be incorporated into the report and the final submission of the report.

Date: 05.01.2024

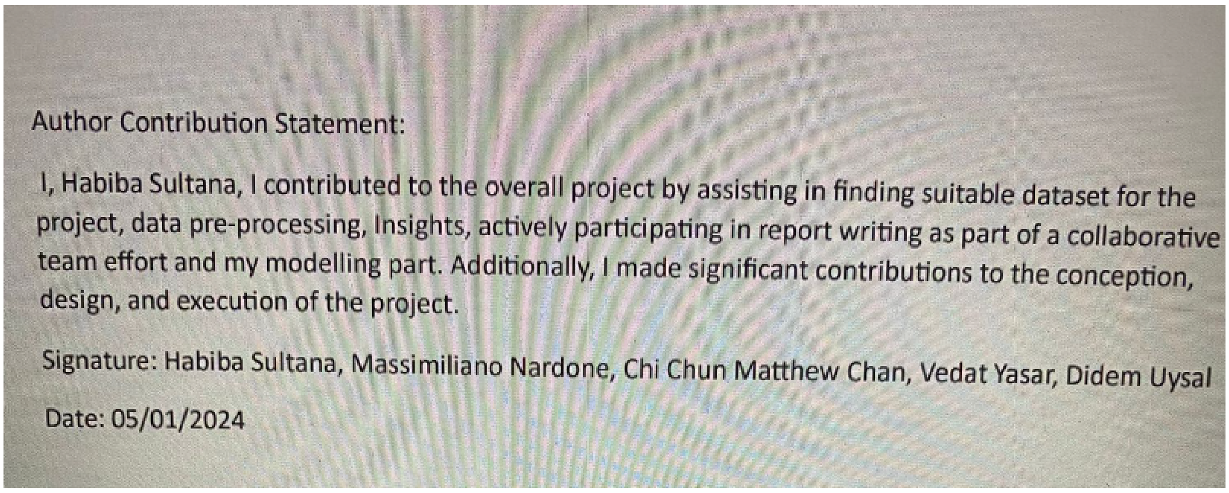
Signatures: Vedat Yasar

Didem Uysal

Massimilliano Nardone

Chan Matthew Chi Chun

Habiba Sultana

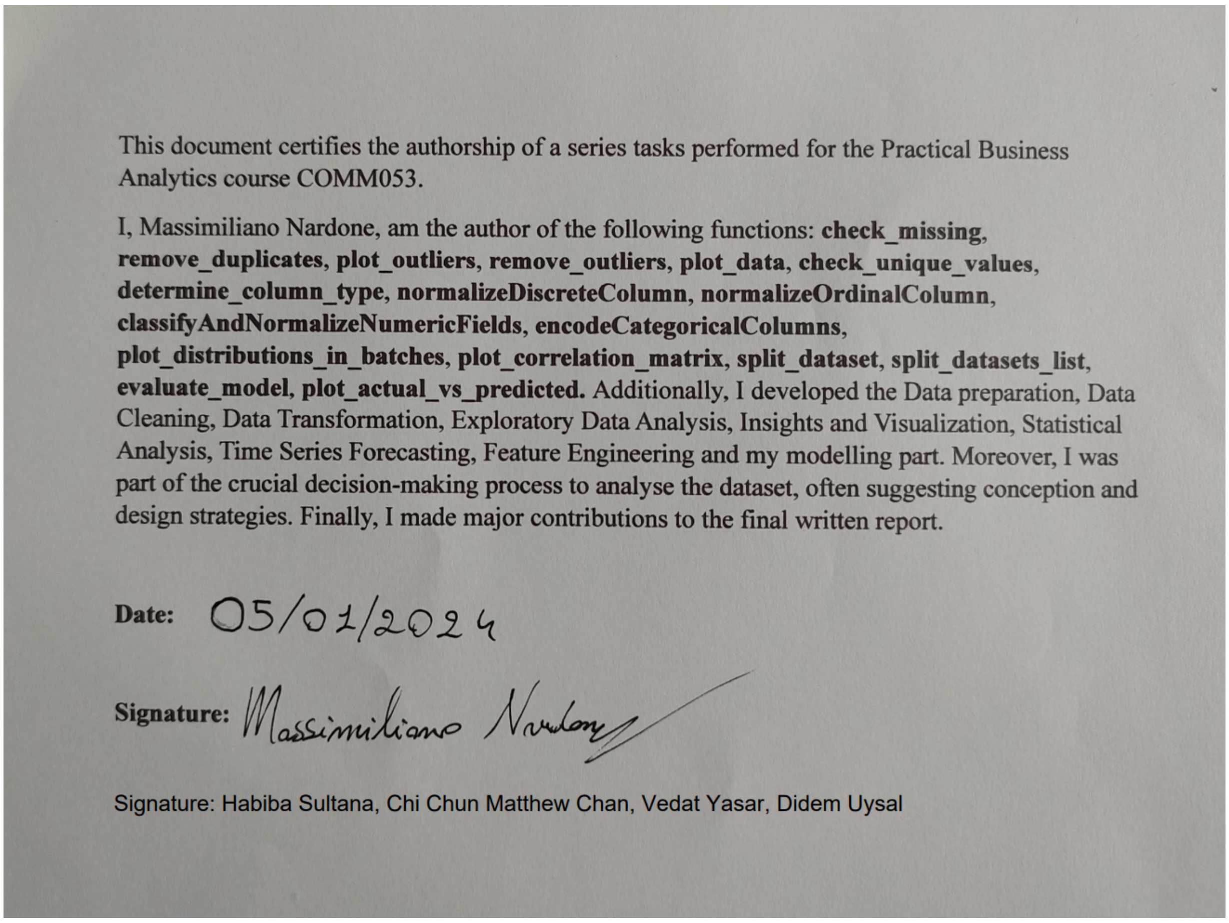


Author Contribution Statement:

I, Habiba Sultana, I contributed to the overall project by assisting in finding suitable dataset for the project, data pre-processing, Insights, actively participating in report writing as part of a collaborative team effort and my modelling part. Additionally, I made significant contributions to the conception, design, and execution of the project.

Signature: Habiba Sultana, Massimiliano Nardone, Chi Chun Matthew Chan, Vedat Yasar, Didem Uysal

Date: 05/01/2024



This document certifies the authorship of a series tasks performed for the Practical Business Analytics course COMM053.

I, Massimiliano Nardone, am the author of the following functions: **check_missing**, **remove_duplicates**, **plot_outliers**, **remove_outliers**, **plot_data**, **check_unique_values**, **determine_column_type**, **normalizeDiscreteColumn**, **normalizeOrdinalColumn**, **classifyAndNormalizeNumericFields**, **encodeCategoricalColumns**, **plot_distributions_in_batches**, **plot_correlation_matrix**, **split_dataset**, **split_datasets_list**, **evaluate_model**, **plot_actual_vs_predicted.** Additionally, I developed the Data preparation, Data Cleaning, Data Transformation, Exploratory Data Analysis, Insights and Visualization, Statistical Analysis, Time Series Forecasting, Feature Engineering and my modelling part. Moreover, I was part of the crucial decision-making process to analyse the dataset, often suggesting conception and design strategies. Finally, I made major contributions to the final written report.

Date: 05/01/2024

Signature: Massimiliano Nardone

Signature: Habiba Sultana, Chi Chun Matthew Chan, Vedat Yasar, Didem Uysal

This document certifies the authorship of a series of tasks performed for the Practical Business Analytic course COMM053 for 2023/2024.

I, Didem Uysal, contributed on the dataset research, data preparation, data visualization: and actively studied on the modelling of the code assembling, code review, results integration, and report writing.

Date: 5/1/2024

Signatures:
Massimiliano Nardon
Chi Chun Matthew Chan
Vedat Yasar
Habiba Sultana
Didem Uysal

This document certifies the authorship of a series of tasks performed for the Practical Business Analytic course COMM053.

I, Chi Chun Matthew Chan, contributed to the following tasks, dataset research, data preparation, data visualization, all models code assembling, code review, code and concept suggestions, major decision making, results integration, result evaluation, result visualization, code delivery and report writing.

Date: 5/1/2024

Signature: Habiba Sultana, Massimiliano Nardone, Vedat Yasar, Didem Uysal