DATA1030 Final Report

By

Yingyi Zhu

Data Science Institute

https://github.com/yingyi0422/DATA1030-Final-Project.git

Introduction

As an advanced form of urban transportation and the fruit of environmental activism, bike sharing has become a heated topic nowadays and has been deployed in many cities around the globe. Not only do they provide a convenient and affordable way to live, but they also positively contribute to the environment, health, and urban sustainability. For example, bike sharing reduces traffic congestion, improves air quality, and encourages people to adopt a healthier lifestyle. Moreover, they could help reduce the fight over limited parking space during rush hours, creating more public space in cities and improving urban livability.

In this project, I aim to delve into how to better allocate the number and timing of shared bikes by operating a machine learning pipeline on the London Bike Sharing dataset on Kaggle. A potential goal of this project could be to help companies and governments decide on the number of shared bikes to place at different times and locations to optimize resource utilization and service efficiency. By analyzing shared bike usage patterns, user demand, weather conditions, and other relevant factors, shared bike allocation strategies can help ensure that the shared bike system better addresses the travel needs of tourists and promotes green transportation. Moreover, it could provide business solutions to bike-sharing companies since the market for bike-sharing has become a competitive field.

The data used in this analysis was obtained from three primary sources:

1. Transport for London (TfL) cycling data contains information on bike sharing, including start times, and is copyrighted alongside OS data and Geomni UK map data. The data covers the period from January 1, 2015, to December 31, 2016, and focuses on the number of new bike-share bikes per hour but excludes long-duration bike-share bikes.

2. WEATHER DATA from freemeteo.com, including actual temperature (t1), "felt" temperature (t2), humidity (hum), and wind speed, and categorized weather codes (weather_code). This weather data is critical to understanding how weather conditions affect shared cycle patterns.

3. UK GOVERNMENT BANK HOLIDAYS: Data on public holidays was obtained from the official UK government website (https://www.gov.uk/bank-holidays) from January 1, 2015, to December 31, 2016.

An author named Hristo Mavrodiev on Kaggle built machine-learning pipelines and created a random forest and an XGBoost regressor model on the data and discovered that the predictive power of the XGBoost regressor model is characterized by achieving its best Root Mean Squared Logarithmic Error (RMSLE) score of 0.2611, which translates to a standard deviation of 1.30 when expressed as a multiplier. This standard deviation indicates that the model's predictions will likely fall within 1.30 times greater or lesser than the actual values. Also, if we consider the mean of the transformed predictions from both algorithms, the RMSLE score improves to 0.2557, corresponding to a standard deviation of 1.29 times the prediction. However, it's important to note that the XGBoost model exhibits overfitting issues, even when it performs well on the test set.

Exploratory Data Analysis

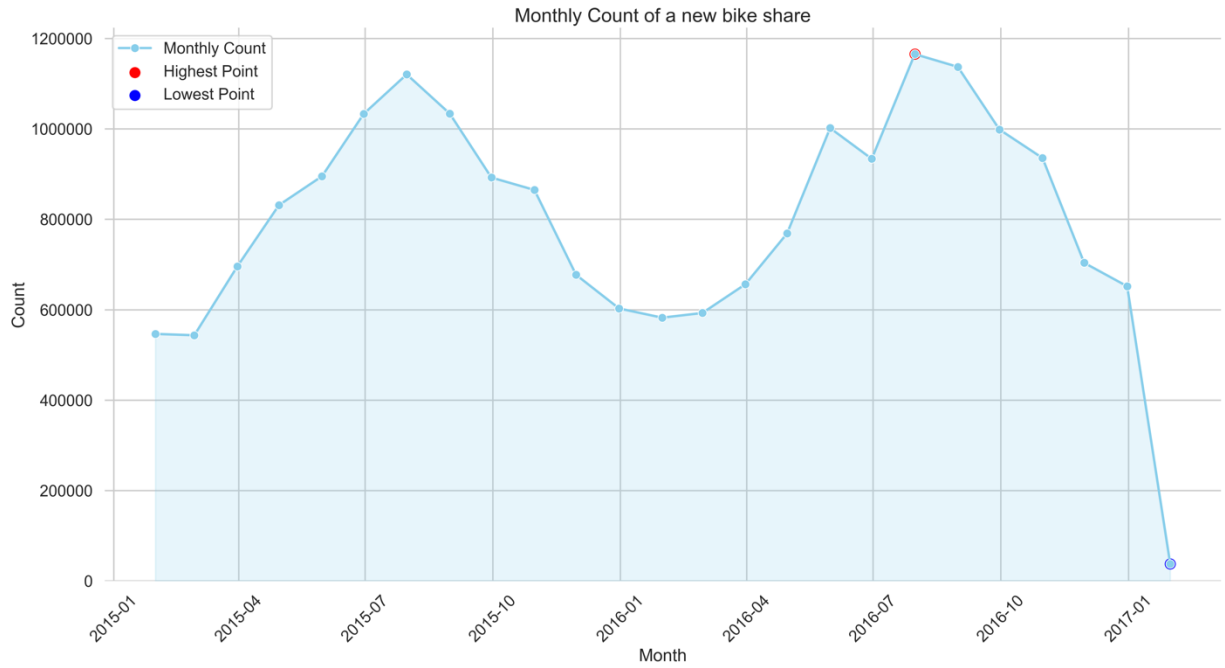

Figure 1 Monthly Count of New Bike Shares

This figure shows the monthly count of new bike shares, obtained by grouping all the counts in the same month. It follows the seasonal pattern: the highest point is in the summer when the temperature is high, and the weather is warm, and the lower end is in the winter when it is relatively cold outside.

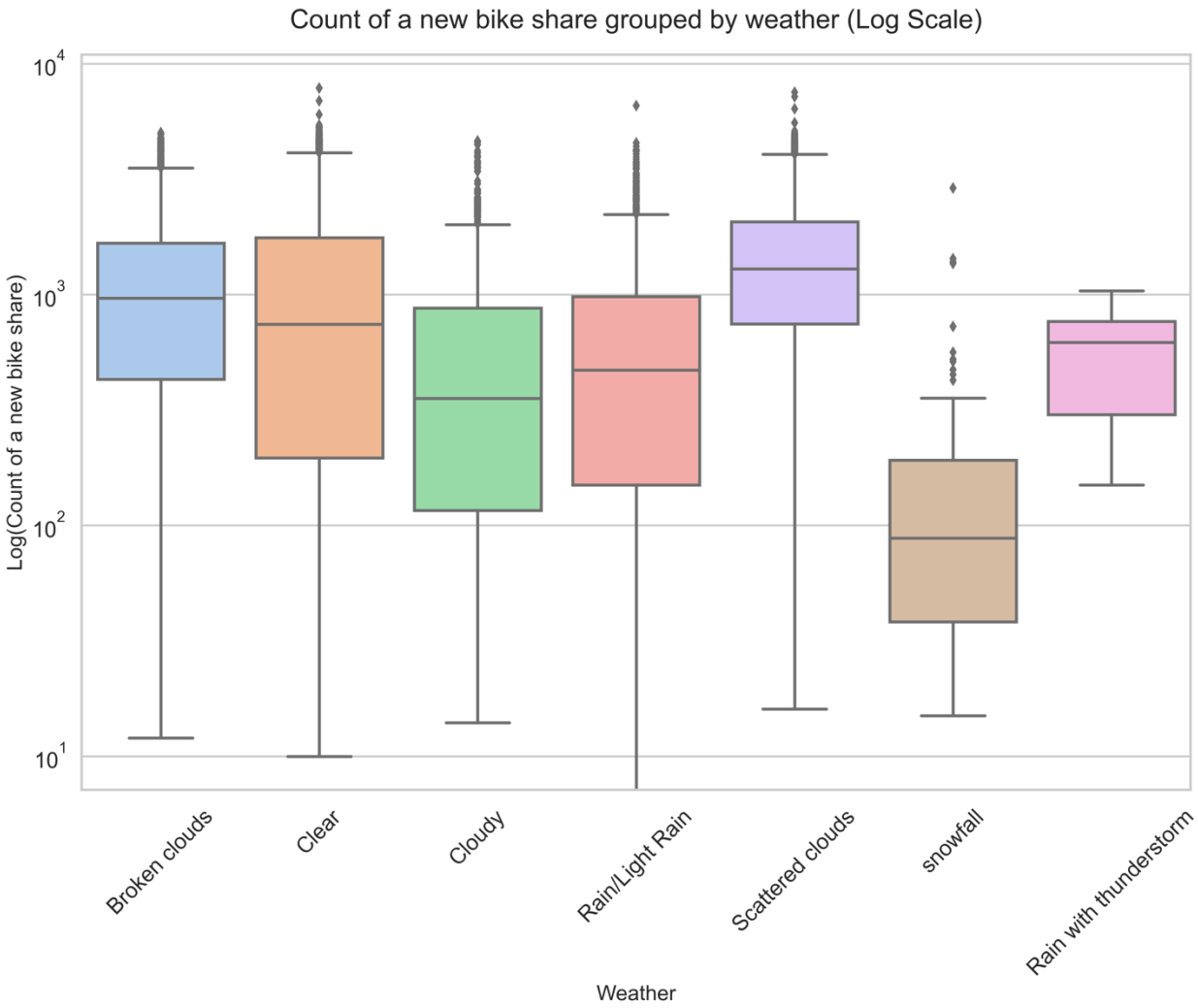

Figure 2 A Boxplot of the Count of a New Bike Share Grouped by Weather

This figure is a boxplot depicting the number of new bike shares grouped by weather conditions. It is intuitive since the count has the lowest mean when it snows outside, which makes sense since it would be the least ideal condition to ride a bike out. However, the average is not low when there is rain with thunderstorms, which is counterintuitive.

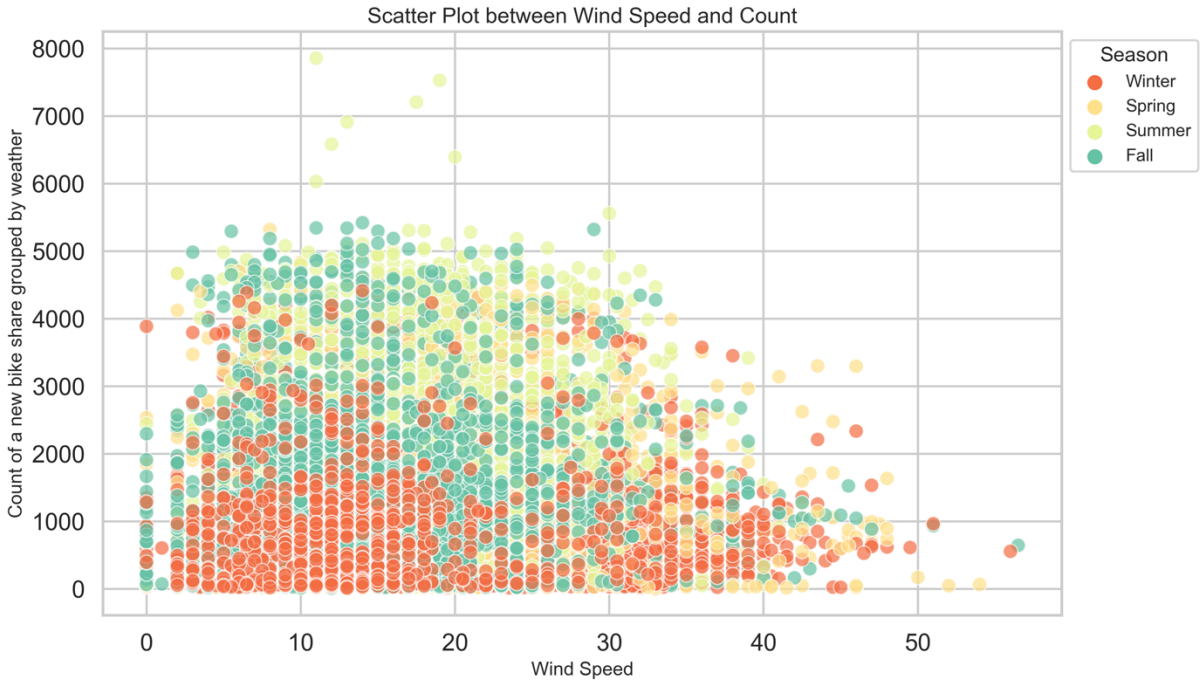

Figure 3 Scatter Plot of Windspeed and Count, Grouped by Seasons

This plot shows the relationship among three features: season, wind speed, and the target variable. The pattern indicates that the count tends to be smaller in winter compared to the other three seasons. Moreover, as the windspeed increases, there tends to be less count, which is intuitive.

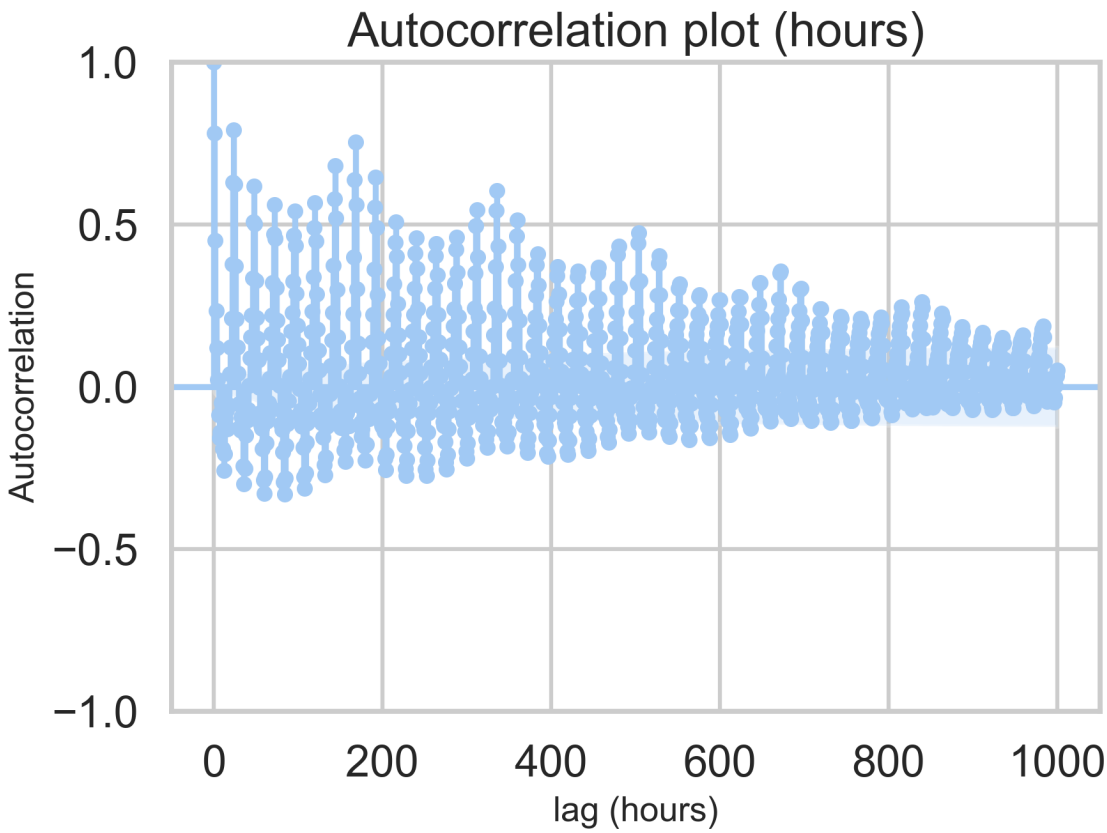

Figure 4 Autocorrelation

This figure is an autocorrelation plot showing the degree of similarity between a time series and a lagged version over successive time intervals. The plot suggests a strong autocorrelation at initial lags that gradually decreases but remains positive, indicating periodicity or seasonality, which proves that the dataset is a typical time series.

Methods

First, I manipulated data to replace season, holiday, weather, and weekend codes from numbers to their actual context. For example, 1 in weather codes was replaced by "clear," and 2 was replaced by "scattered clouds." Then, the initial split was to separate the dataset into a training set and a test set, with 80% of the data distributed to the training set and the last 20% allocated to the test set. This initial split differs from the fundamental train test split as I manually ensured that the first 80% and the last 20% strictly followed the sequential order, which helped prevent data leakage.

Preprocessing, on the other hand, involves several steps:

1. The first step was to extract features from the "timestamp" features in the dataset, such as hour, day, month, year, and day of the week. The reason is that it could potentially help uncover patterns linked to time and prevent data leakage.
2. The second step was to create three lagged features: one week (160 hours), two weeks (320 hours), and three weeks (480 hours), which is essential when dealing with time series, as this would help capture historical patterns linked in time in the dataset.
3. The third step was to drop the rows caused by creating lagged features so that the following pipelines could function correctly.
4. The fourth step was to analyze all variables and allocate them to categorical, ordinal, and numeric features. Then, I scaled them either using a standard scaler or a one-hot encoder, ensuring that all features contribute equally to the models.

Before preprocessing, the training set contained 13,547 rows and 10 features, and after the process, the number of elements became 21.

The primary metric for model evaluation was the Root Mean Squared Error (RMSE), which penalizes more significant errors and could provide a better picture of model performances, especially when capturing the robustness of time series data and the compatibility with cross-validation.

The function tune_and_evaluate_model is the most crucial step out of this machine learning pipeline. First, it takes in a model, a parameter to tune, preprocessed training data (X_train_prep and y_train), the model's name, and expanding window size for cross-validation. Inside the function, it first performs a time series split to define the number of folds for cross-validation, which suits time series data by ensuring that the validation set always comes after the training set, which helps retain the temporal order. Then, it creates custom expanding window splits, and in each split, the size of the training set expands while the test set remains the same size. After that, the function employs "GridSearchCV" to search over specified parameter values for the model, which optimizes model performance based on negative mean squared error.

For each split, the best model (with tuned parameters) is fitted on the training subset and used to predict the corresponding test subset. The RMSE, mean, and standard deviation are calculated for each test subset across different periods. Lastly, the function prints out the best parameters found, the average, and the standard deviation of RMSE and returns the best models.

I chose four models, ensuring a balance between linear and non-linear algorithms. The four selected models and their parameters are illustrated in Figure 5.

ML Algorithm	Linear	Parameter Tuning	
Ridge	Yes	'alpha': [0.1, 1, 10, 50, 100, 200	
Random Forest	No	'n_estimators': [100, 200]	'max_depth': [10, 20, None]
K-Neighbors	No	'svr__C': [0.1, 1, 10]	'svr__gamma': ['scale', 'auto']
SVR	Yes	'n_neighbors': [3, 5, 7]	'weights': ['uniform', 'distance']

Figure 5 Models and Parameters Chart

Among these models, Ridge regression was selected for its ability to handle multicollinearity and prevent overfitting. Random forest was chosen to model complex non-linear relationships. Support Vector Regression was fixed for high-dimensional spaces and robustness to outliers. Lastly, the K-neighbor regressor was selected for its simplicity and ability to capture the local structure of the data.

I incorporated time series split and expanded windows to handle and measure uncertainty to ensure the model stays robust to different periods. Moreover, to mitigate uncertainty caused by inherent randoms like the random forest, I used a fixed random state and examined the standard deviation of RMSE across different periods.

Results

ML Algorithm	Best Parameter	RMSE	Mean RMSE	Std RMSE
Ridge	'alpha': 0.1	908.21	879.35	66.41
Random Forest	'max_depth': 20, 'n_estimators': 200	849.6	826.67	86.38
K-Neighbors	'n_neighbors': 5, 'weights': 'distance'	977.86	916.63	81.71
SVR	'svr__C': 10, 'svr__gamma': 'scale'	993.06	970.68	149.70
Baseline			1120	

Figure 6 Model Outcomes Chart

Based on Figure 6, the random forest model appears to be the most predictive compared to the baseline score 1120, as the mean of 826.67 is around 3.42 standard deviations below the baseline score.

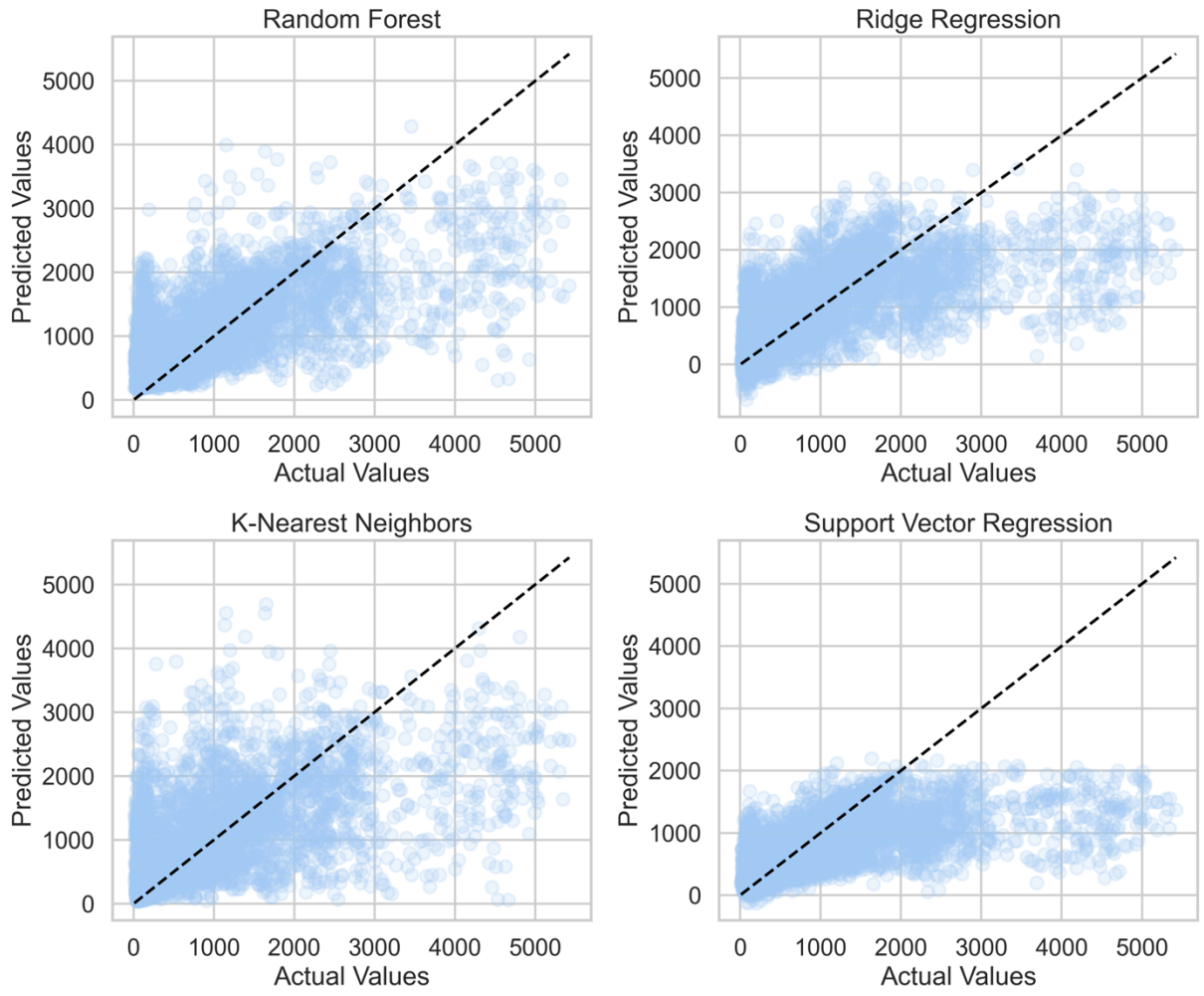

Figure 7 Predicted Values vs. Actual Values

Figure 7 indicates that all models show some degree of variance for the predicted line. Still, the random forest and ridge regression appear closer to the line than the other two, which aligns with the scores as they also have the two lowest mean RMSEs.

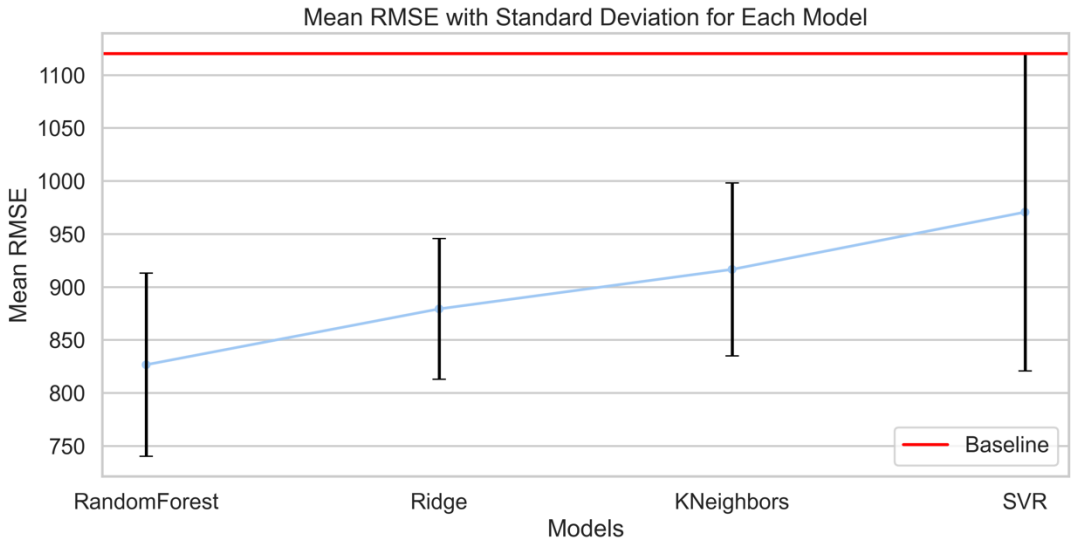

Figure 8 Mean RMSE with standard deviation.

Figure 8, on the other hand, is a plot mean RMSE with standard deviation for each model, corresponding to the scores in Figure 6. It also indicates that all four models are under the baseline, meaning that they perform well, but random forest and ridge tend to be the better two.

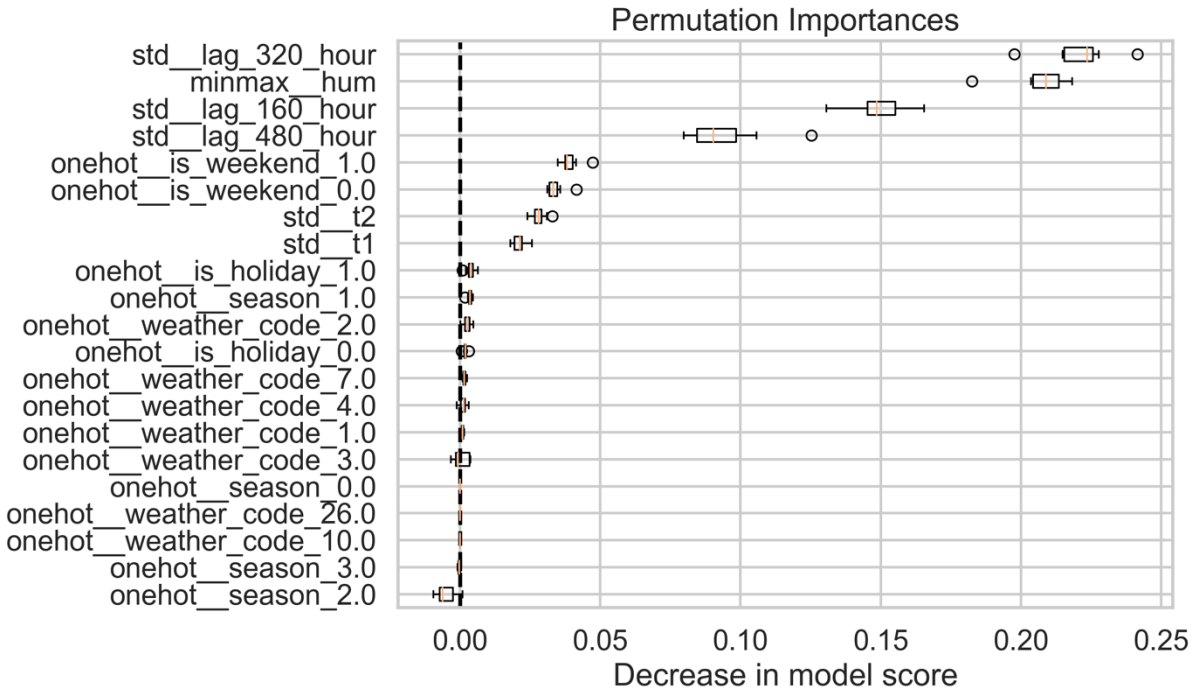

Figure 9 Permutation Importance

Figure 9 illustrates the features that most significantly impact the random forest model's predictions in the context of permutation importance. It is shown that features related to count lagged two weeks and humidity are among the most influential, which means that lagged values and weather conditions are the most predictive.

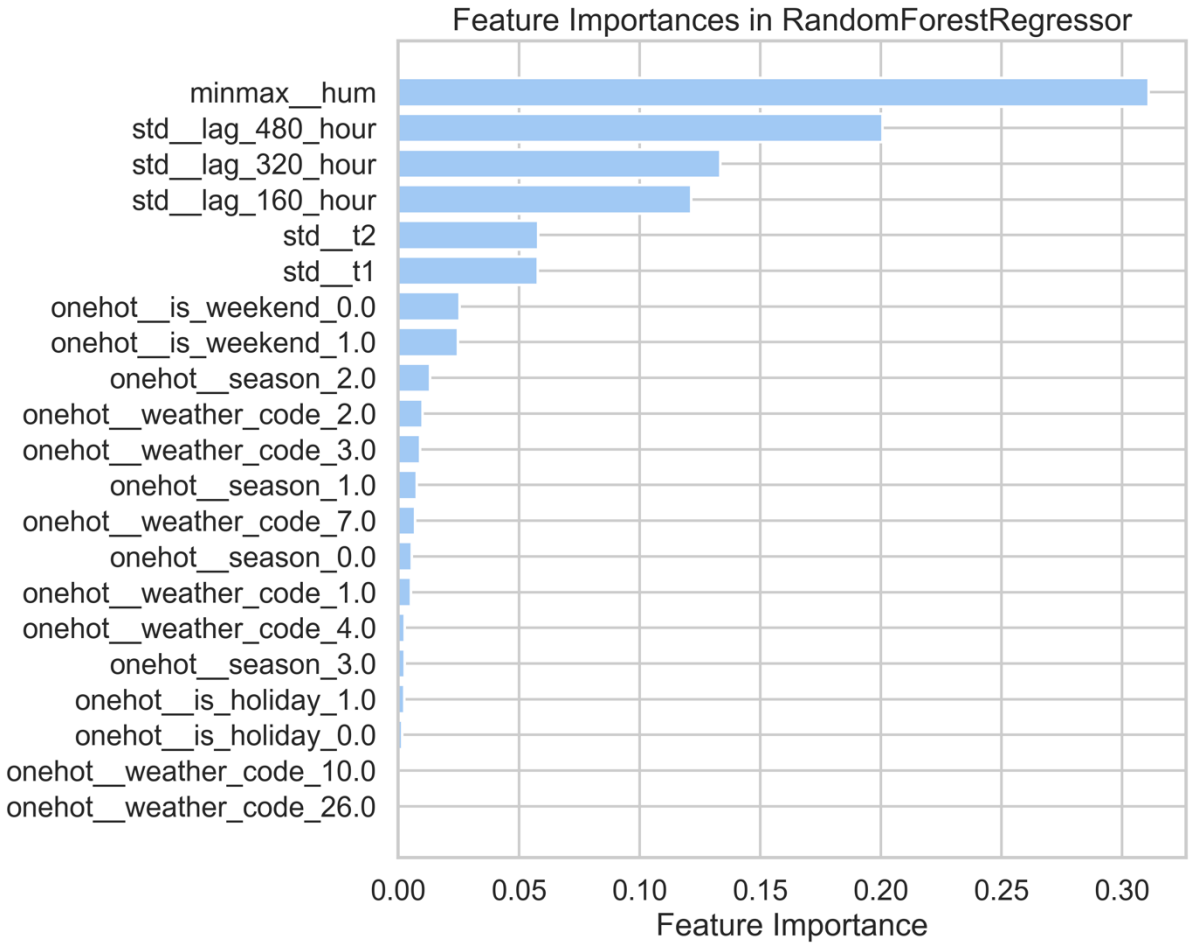

Figure 10 Random Forest Regressor Feature Importance

Figure 10 shows the feature importance achieved by RandomForestRegressor, and it has similar results to the permutation importance results, with humidity and lagged features being the most influential.

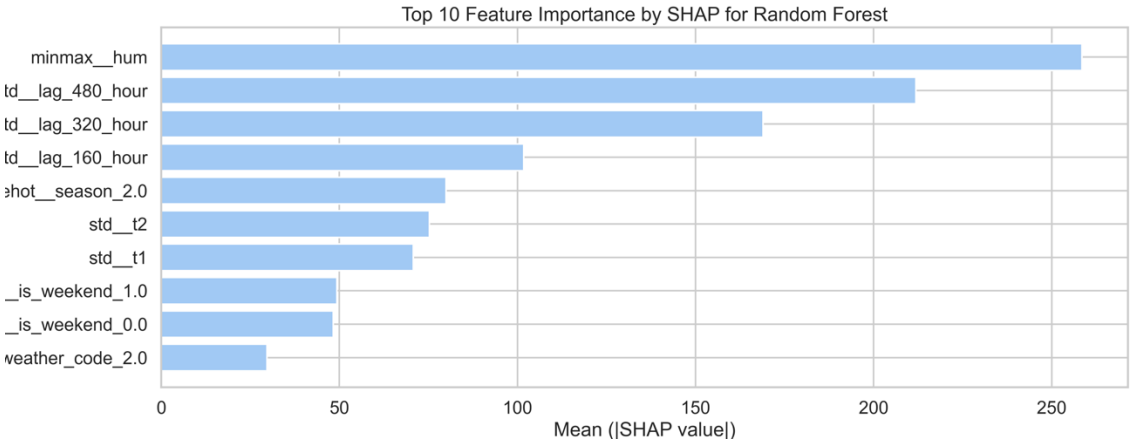

Figure 11Top 10 Feature Importance by SHAP for Random Forest

Figure 11 depicts that, like the previous two global feature importance plots, lag features and humidity are the most essential features.

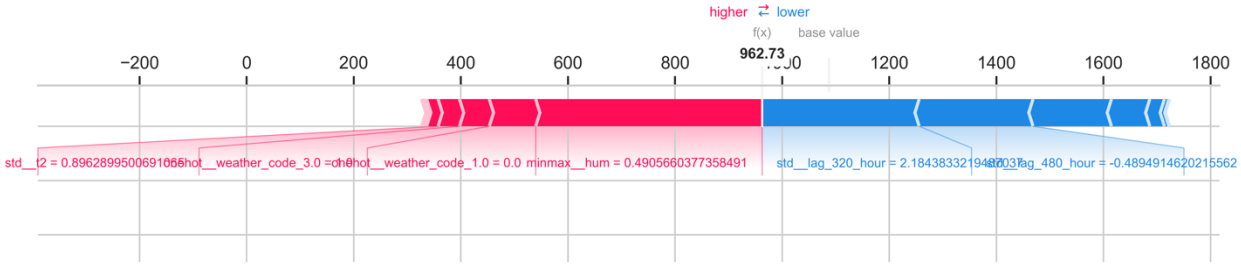

Figure 12 Shap Force Plot Index 0

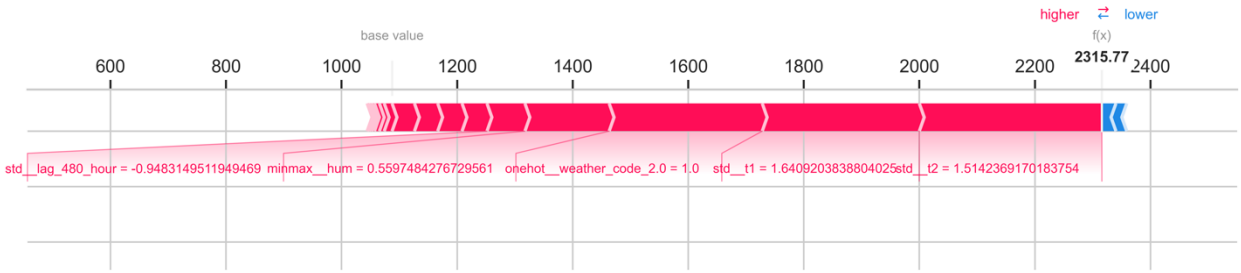

Figure 13 Shap Force Plot Index 100

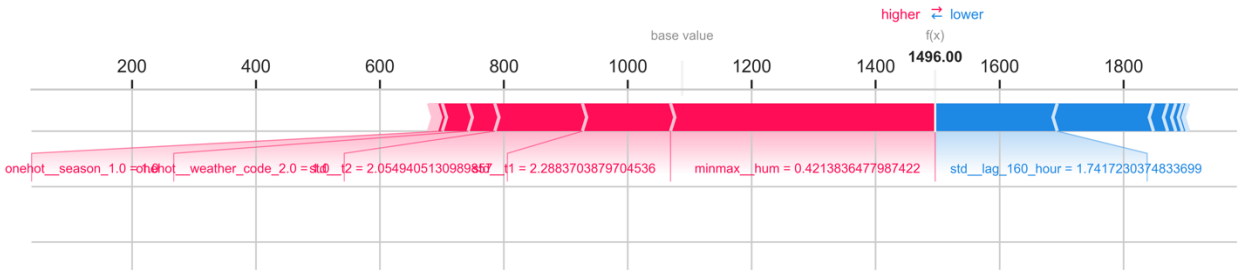

Figure 14 Shap Force Plot Index 200

At index 0, features like temperatures push the prediction up, while lag features pull it down. At index 100, the other temperature variable strongly contributes to the prediction. At index 200, t1 is again pushing the prediction. In sum, t1 and t2 positively contribute to the forecast, while lag hours pull down the projections.

Outlook

Based on what I have discovered, it is difficult to determine why the lag features are the most important since, intuitively thinking, temperature, humidity, and weather would be the most crucial. Therefore, doing more experiments when I preprocess the data would be helpful. For example, I could add interactive terms and experiment with more models, especially those designed for time series like ARIMA. Moreover, although my current conclusion is that data leakage is unlikely, given the dataset's characteristics, I would love to find out if I needed to include any spots.

Moreover, I would like to try other evaluation metrics, such as RMLSE (the logarithm form) and MSE, to see how my scores could vary. It would also be ideal to dig deeper into the parameter tuning part and see if there are parameters that could better fit the models I formatted.

Reference

Cycling Data: Transport for London. (n.d.). Cycling data. Retrieved from https://cycling.data.tfl.gov.uk/

Weather Data: Freemeteo. (n.d.). Weather data. Retrieved from https://freemeteo.com

Bank Holidays in the UK: Government Digital Service. (n.d.). UK bank holidays. Retrieved from https://www.gov.uk/bank-holidays

Kaggle Notebooks and Dataset: Mavrodiev, H. (n.d.). Bike sharing prediction: RF & XGBoost [Data set]. Kaggle. Retrieved from https://www.kaggle.com/code/hmavrodiev/bike-sharing-prediction-rf-xgboost

Mavrodiev, H. (n.d.). London bike sharing dataset [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/hmavrodiev/london-bike-sharing-dataset