DATA1030 Project Report

Guanjie Linghu

October 2021

1 Introduction

Bike sharing has been regarded as the most popular environmental-friendly traffic mode in recent years and developed all around the world, especially in developed countries. Therefore, governments or bike sharing companies need to know the number of bikes they should launch, which means they need to estimate the bike usage. It has traits of low usage barrier and easily being disturbed by external factors, such as weather. Therefore, The objective of this project is to analyze the relation between the bike usage and several kinds of external factors and develop regression models to make a prediction on the sharing bike usage amount in London given the external factors.

My data set contains 17414 data and each data has the following 9 attributes: timestamp field, count of the new bike shares, real temperature, "feels like" temperature, humidity, wind speed, weather, season, and boolean attribute of whether that day is holiday or weekend. A detailed description of the data set can be accessed from Kaggle website given as reference[1].

There are several public projects based on this data set. Most of them are merely exploratory data analysis. In *Bike sharing prediction*[2], the author break down the timestamp into several distinct features and develop an XGBoost regressor without considering time-series property. It gives a good performance, an R2 score of 0.96. In *London Bike Share - Prophet, XGB, LSTM*[3], the author did a great job on developing models with four different algorithms and finally got the best result from the LSTM model, with an RMSE score of 414. However, the author mainly focused on a time series forecasting problem but not a superviesd regression problem.

2 Exploratory Data Analysis

After carefully investigate each feature, I have already had a good grasp on the overall pattern of the data set as a time series and the relation between every two features. Another thing that I need to mention here is that to perform EDA thoroughly, I create four new categorical features, Month, Day of Month, Day of the week, and Hour. I will discuss it later in the Feature Engineer Section. This section will cover some interesting facts that I found by performing EDA.

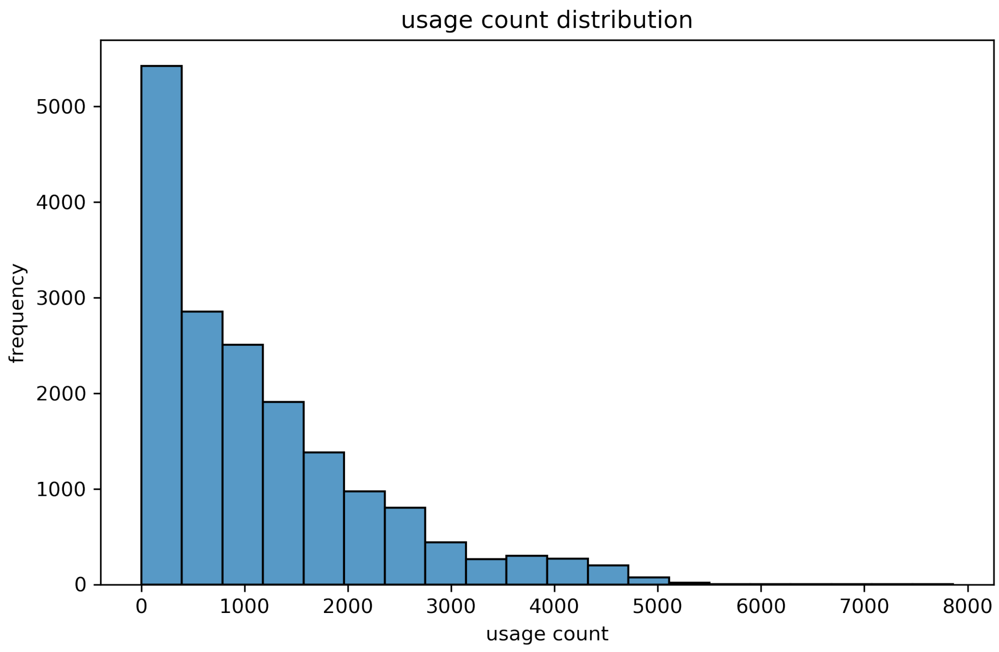

Figure 1: Distribution of target variable

Figure 1 is the histogram that shows the distribution of the target variable ”cnt”. From this graph, we can see that the distribution is highly skewed to the right. Combined with the descriptive result, we have a mean of 1143. The first interval contains more than 25% of values. The maximum value is 7860.

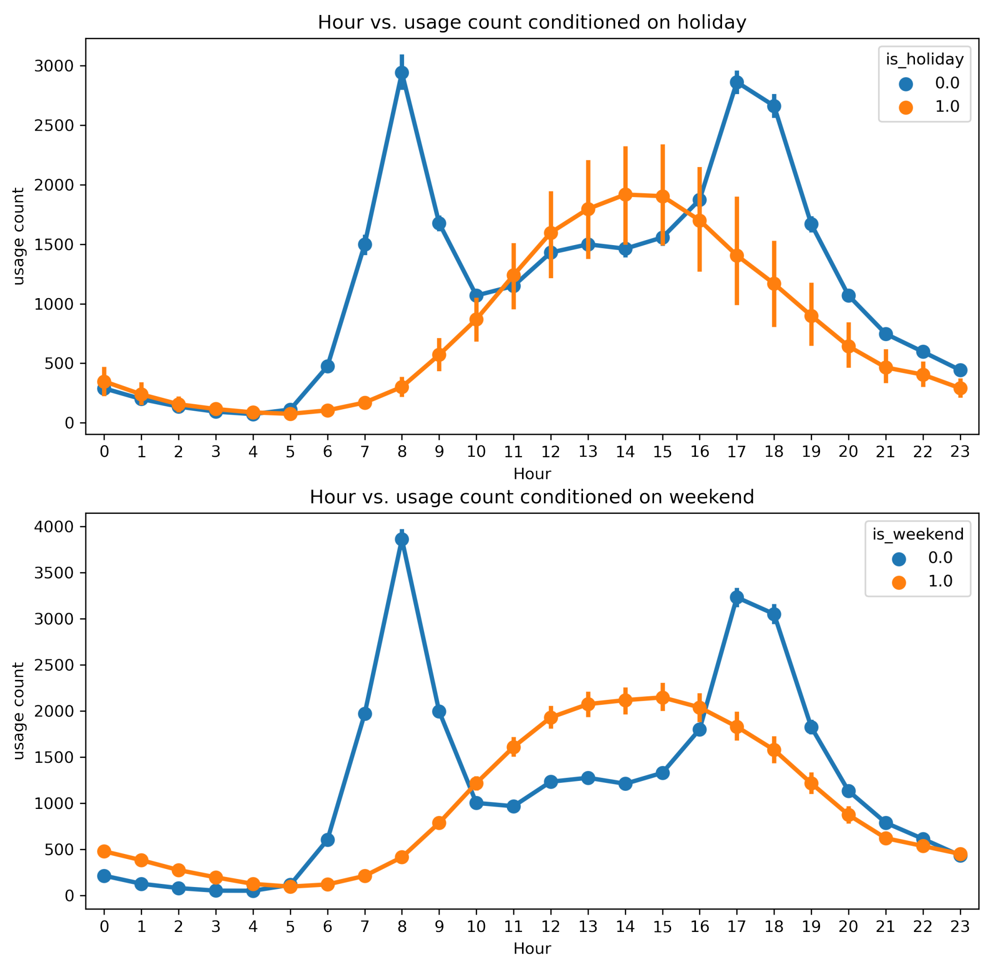

Figure 2: Hour vs. usage count conditioned on holiday and weekend

Figure 2 includes two graphs that show the average amount of shared bike use each hour. Each point with a vertical bar represents an estimate of the central tendency for usage amount of each hour and provides some indication of the uncertainty around that estimate using error bars. The first graph is conditioned on feature is_holiday and the second one is conditioned on feature is_weekend. An interesting and useful fact is that during workdays, people tend to use the shared bike around 8 am and 5-6 pm, which are the rush hour traffic times. While during the break, people usually use the shared bike in the afternoon. It might be useful to extract the pattern as a new feature to improve the performance of machine models.

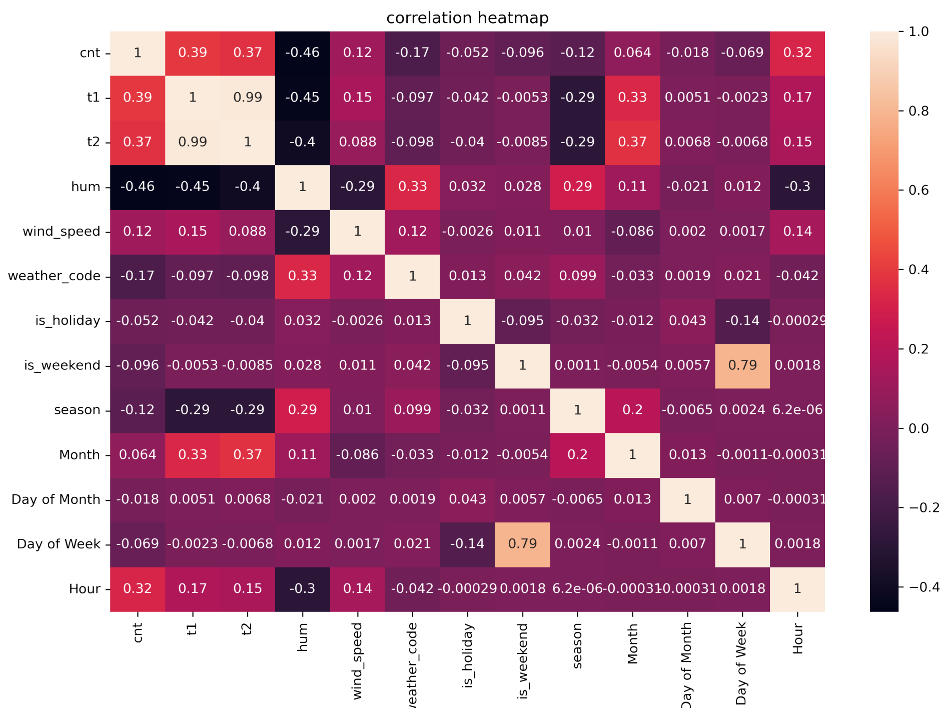

Figure 3: Correlation Heatmap

Figure 3 is the correlation heatmap that gives the correlation between each feature. From the heatmap, we can see that feature "t1", "t2", "hum", and "Hour" has the strongest correlation with target variable "cnt".

3 Method

In this section, I will discuss my pipeline, which includes splitting strategy, data preprocessing, machine learning algorithms that I choose. And then I will talk about how I combine the steps listed above together.

3.1 Feature engineering

First of all, since the original data is a time series, which is not i.i.d, I break down the time stamp into four categorical features: "Month", "Day of Month", "Day of Week", and "Hour". "Month" has 12 values which represent 12 months in a year. "Day of Month" has 31 values which are days in a month. Similarly, "Day of Week" contains 7 days in a week, and "Hour" contains 24 hours in a day. By doing so, it can mimic an i.i.d data set. The time-dependent relationship between the adjacent data points will be ignored, I will only consider the period time patterns which are captured by the four new features listed above.

3.2 Data Splitting

Since I treat the data as i.i.d, I shuffle the data set and use 20% of data as a testing set. The remaining part is used to implement 4 fold cross-validation.

3.3 Preprocessing

Standard Scaler is always a good tool to treat continuous variables so I use Standard Scaler to encode "t1", "t2", "hum", and "wind_speed", which are four continuous features. All the remaining 8 original features are categorical and don't have an ordinal pattern, so I encode them by One Hot Encoder. The four new features are tricky. All of them are time components so it seems like they should be ordinal features. However, since they don't lie on the timeline, they are sets with cyclic order, for example, December is 11 months after January but also 1 month before January. Therefore, I treat them as categorical features and implement One Hot Encoder to transform them. After preprocessing, the data set has 93 features plus 1 target variable.

3.4 Model Selection

Eight machine learning models were chosen for this task, which are the linear regression without regularization, the linear regression with L1 regularization (LASSO), the linear regression with L2 regularization (Ridge), the linear regression with L1 and L2 regularization (elastic net), the K-nearest neighbors regressor, the random forest regressor, and the multi-layer perceptron regressor. The table given below lists the hyperparameters tuned for each machine learning algorithm.

Model	Hyperparameters
Linear regression	None
LASSO	alpha: 0.001, 0.01, 0.1, 0.3, 0.7, 1, 10, 100
Ridge	alpha: 0.001, 0.01, 0.1, 0.3, 0.7, 1, 10, 100
Elastic net	alpha: 0.001, 0.01, 0.1, 0.3, 0.7, 1, 10, 100
	L1_ratio: 0.05, 0.2, 0.4, 0.6, 0.8, 0.95
K neighbors regressor	n_neighbors: 2, 3, 5, 10, 17, 30
	weights: uniform, distance
Random forest	max_features: 5, 10, 25, 50, 75
	max_depth: 3, 5, 10, 20, 30, None
Multi-layer perceptron	hidden_layer_sizes : (100),(50,50),(64,32,16),(128,64,32,16),
	(50,50,50),(100,50,25),(256,128,64,32)

Table 1: Candidate hyperparameters of each model

The metric used to evaluate the model in the hyperparameter tuning process and model comparing process is $RMSE$. It is easy to interpret since it has the same measurement units as the target variable. All the processes mentioned above in this section are integrated together using pipeline and GridsearchCV.

All the processes mentioned above were executed 10 times for 10 different random states. The random state controls the randomness in the process of splitting the data, cross-validation, and the fitting process of some machine learning algorithms, which are LASSO, Ridge, Elastic net, Random forest, and Multi-layer perceptron.

4 Results

4.1 Model comparison

For each machine learning algorithm, the pipeline described in the last section returns 10 best models and their corresponding testing $RMSE$ scores for 10 different random states. The evaluation metric for model comparison is the mean and standard deviation of 10 $RMSE$ scores. The mean $R2$ score is also given as an assistant metric. The table below gives the evaluation metrics of the best model of each machine learning algorithm.

model	RMSE mean	RMSE std	R2 mean
Baseline	1091.445000	11.063342	-0.000206
Linear regression	579.520826	10.615211	0.717776
LASSO	579.528827	10.651703	0.717768
Ridge	579.523936	10.616572	0.717774
Elastic net	579.510448	10.578283	0.717788
K neighbors regressor	747.860292	11.099727	0.530414
Random forest	257.837337	8.706595	0.944107
Multi-layer perceptron	160.636425	2.506457	0.978326

Table 2: Testing scores of each model

Obviously, the multi-layer perceptron gives the most predictive model. It has the lowest mean and standard deviation $RMSE$ score. It gives the best $R2$ score too. The mean $RMSE$ score is around 84 standard deviations below the mean $RMSE$ score of the baseline. The XGBoost regressor is the second-best model, which is about 78 standard deviations below the baseline. The random forest also performs very well, which is around 75 standard deviations below the baseline. The performances of the 4 linear models are almost the same, approximately 46 standard deviations below the baseline. K-nearest neighbor regressor is the worst one for this problem since it has the highest mean $RMSE$ and $R2$. Its mean $RMSE$ is only 31 standard deviations below the mean $RMSE$ score of the baseline.

4.2 Feature Importance

Since the feature importance of the neural network is not easy to interpret, the feature importance analysis is implemented to the best model of the XGBoost regressor. The global feature importance is measured by 3 methods, permutation feature importance, SHapley Additive exPlanations (SHAP), and the built-in feature importance attribute of XGBoost. XGBoost implements 5 different metrics to measure feature importance but only the feature importance plot with respect to metrics ”weight” and ”gain” are posted here. You can find the rest of the feature importance plots made by XGBoost built-in function in the figures file in my GitHub directory.

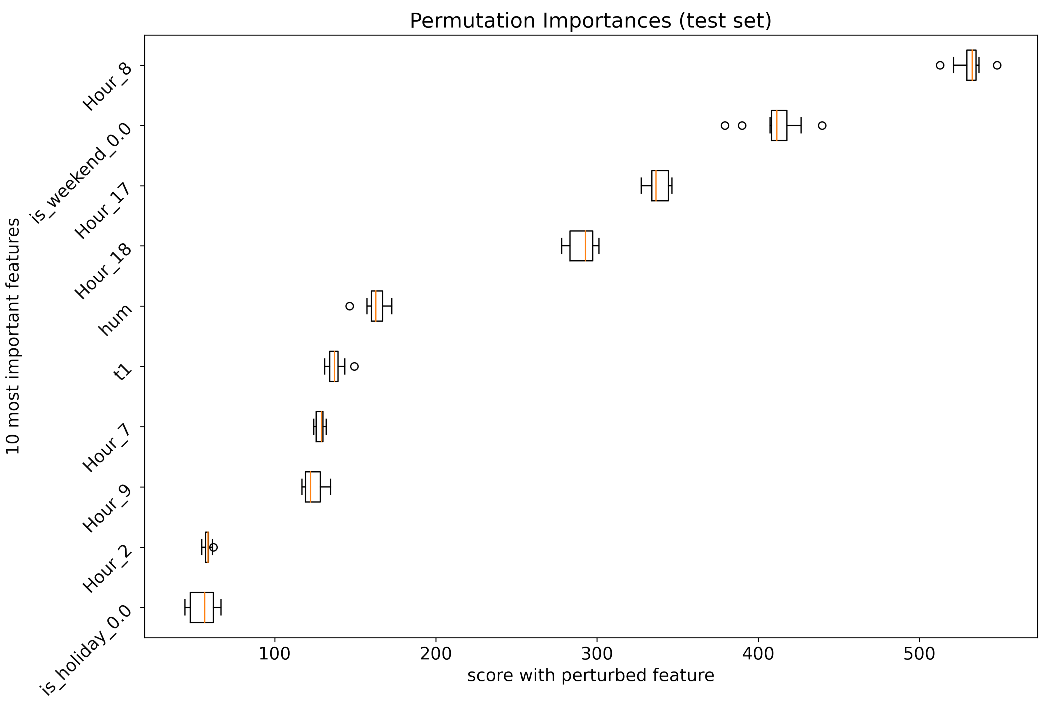

Figure 4: Top 10 permutation importance features

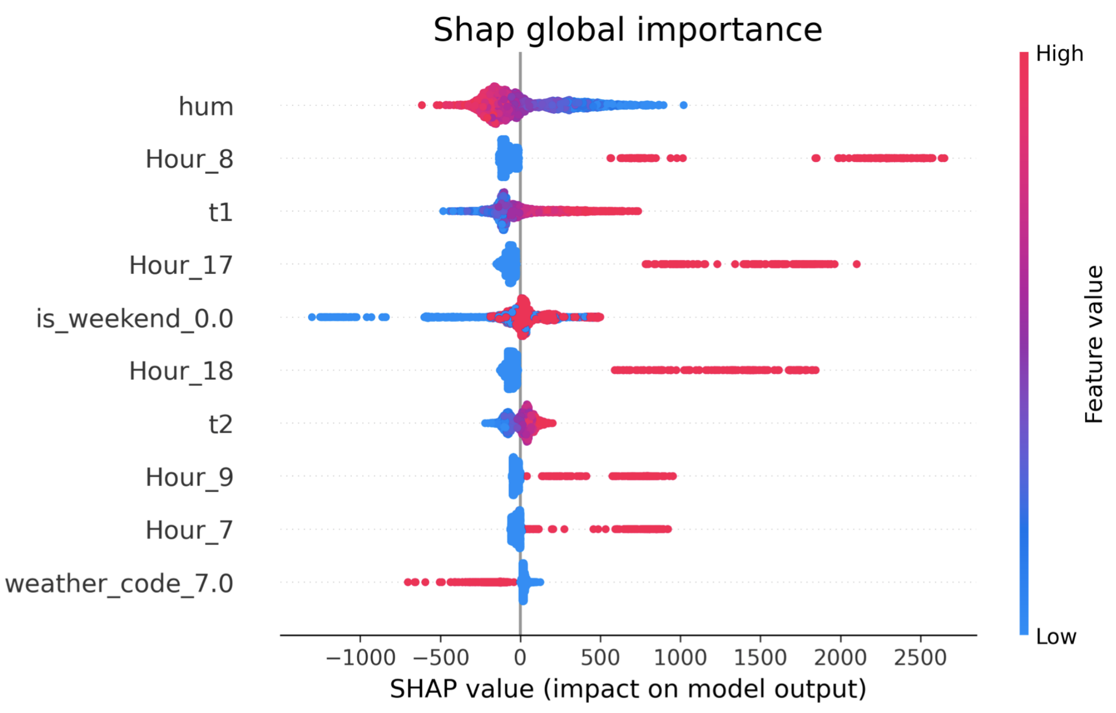

Figure 5: Top 10 SHAP importance features

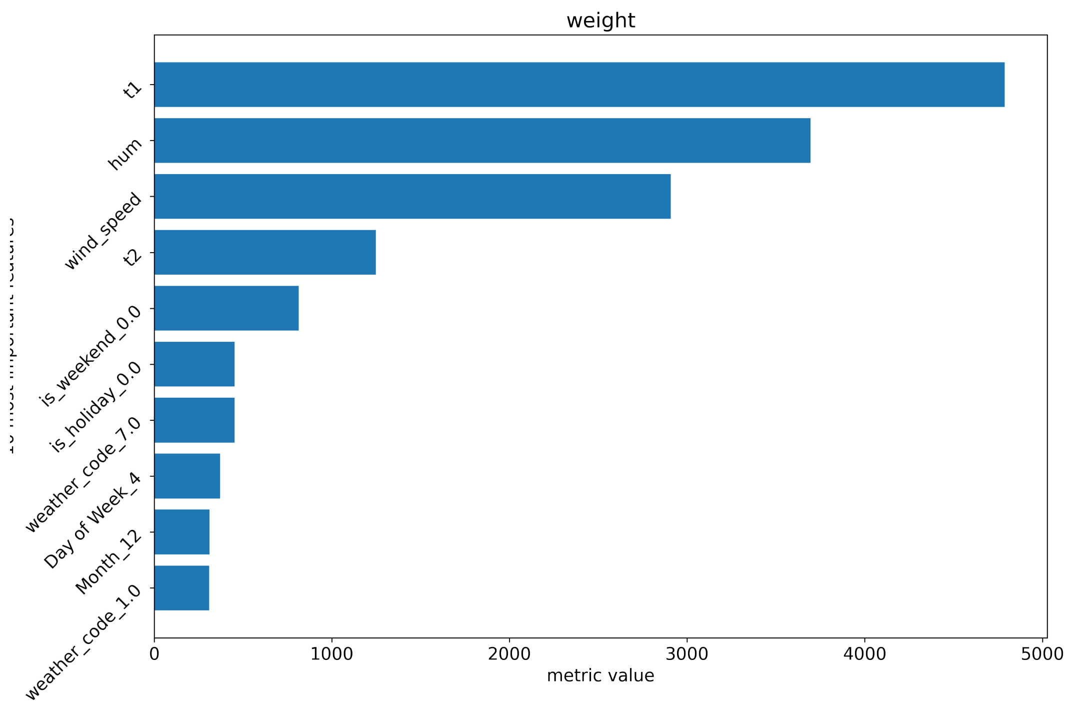

Figure 6: Top 10 XGBoost importance features w.r.t weight

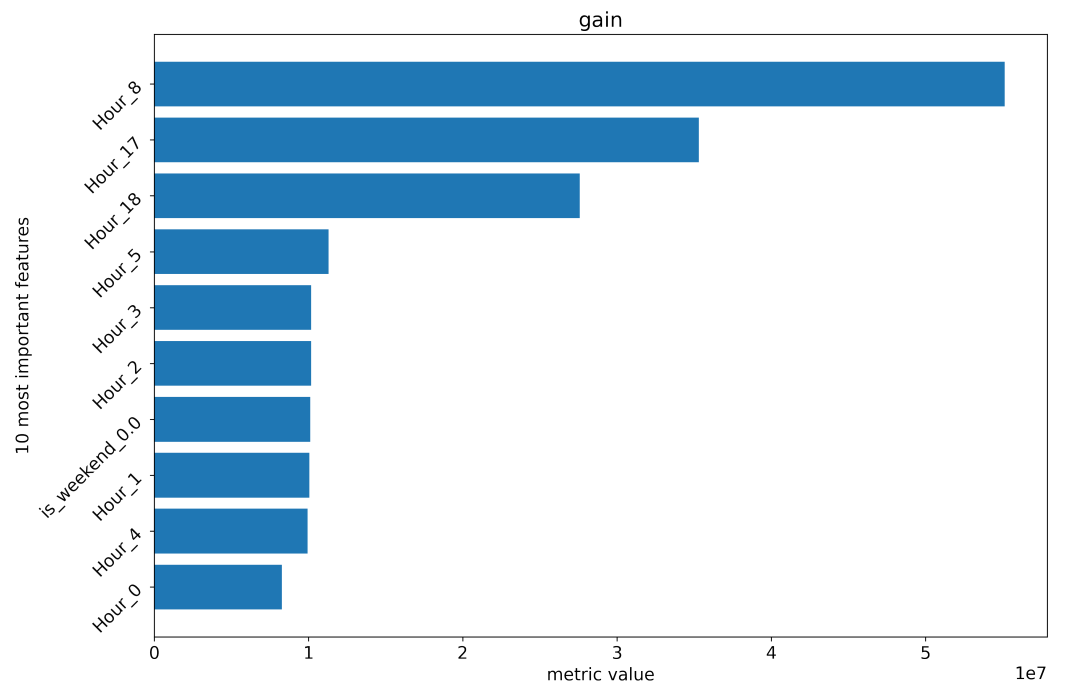

Figure 7: Top 10 XGBoost importance features w.r.t gain

We can see that the results of permutation feature importance and SHAP are almost quite consistent, 8 over 10 features are the same. And these results are also consistent with intuition.

However, the two feature importance plots generated by the built-in function of XGBoost are very different from each other. The top 10 important features selected by "weight" are continuous features and categorical features which are not very imbalanced. It makes sense because most of the features are generated by One Hot encoder using "Day of Month" with 31 values and "Hour" with 24 values, which are binary and super-imbalanced, and thus the number of times those features are used to split the data across all trees, which is the "weight", is very low. And we can see that most of the top 10 important features selected by "gain" are the hours in a day where a lot of people use shared bikes or nobody uses a shared bike and the gain of splitting on these features are very high.

The most unimportant features are the categorical features generated by One Hot encoding the feature "Day of Month".

SHAP also provides us an opportunity to analyze local feature importance so I use dependence plots to show the relationship between feature values and SHAP values. Also, every single point in the dependence plot represents the SHAP score of a specific data point so we can visualize how each data point contributes to the model given a specific feature. Since it is impossible to go over all 93 features, we only focus on 3 important features given by previous global feature importance analysis, which are "t1", "hum", and "is_weekend_0.0".

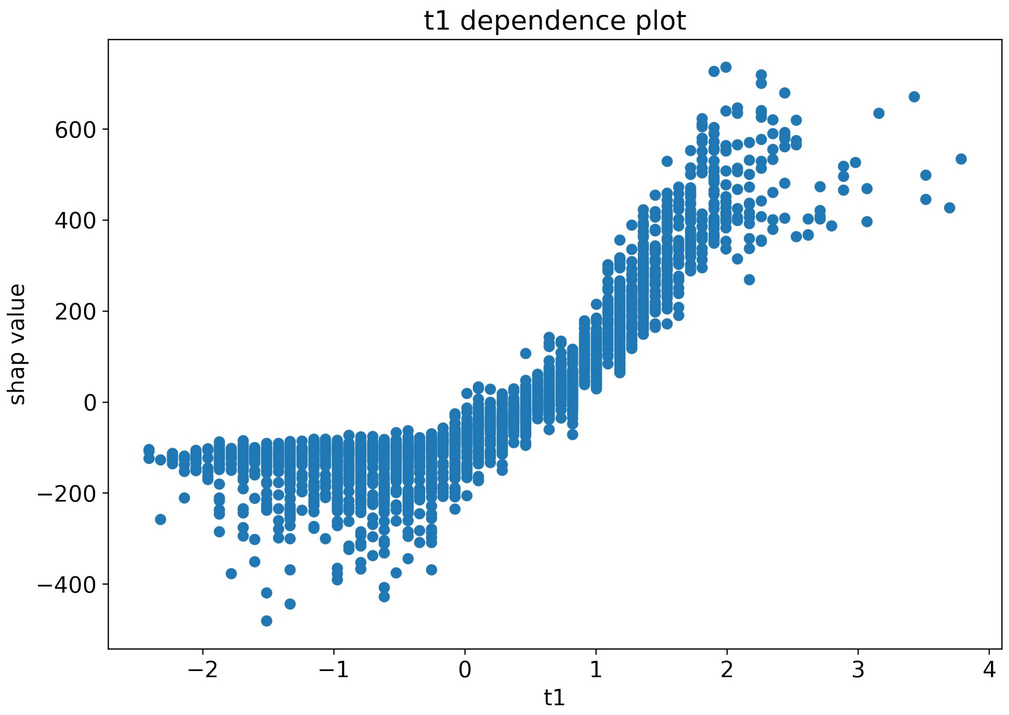

Figure 8: Local importance-temperature

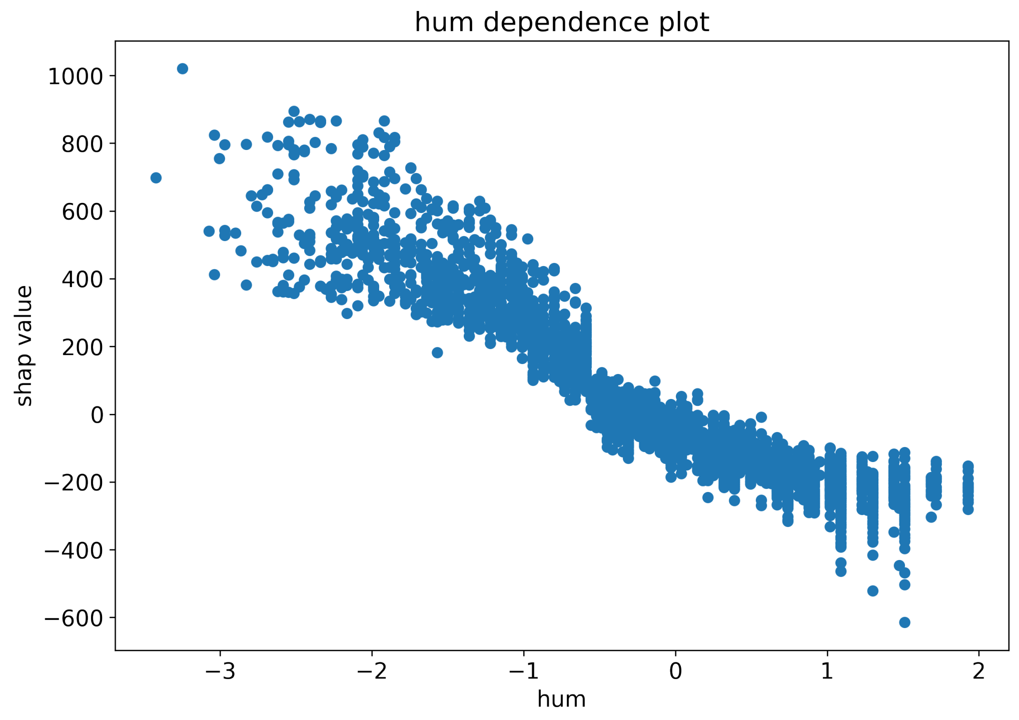

Figure 9: Local importance-humidity

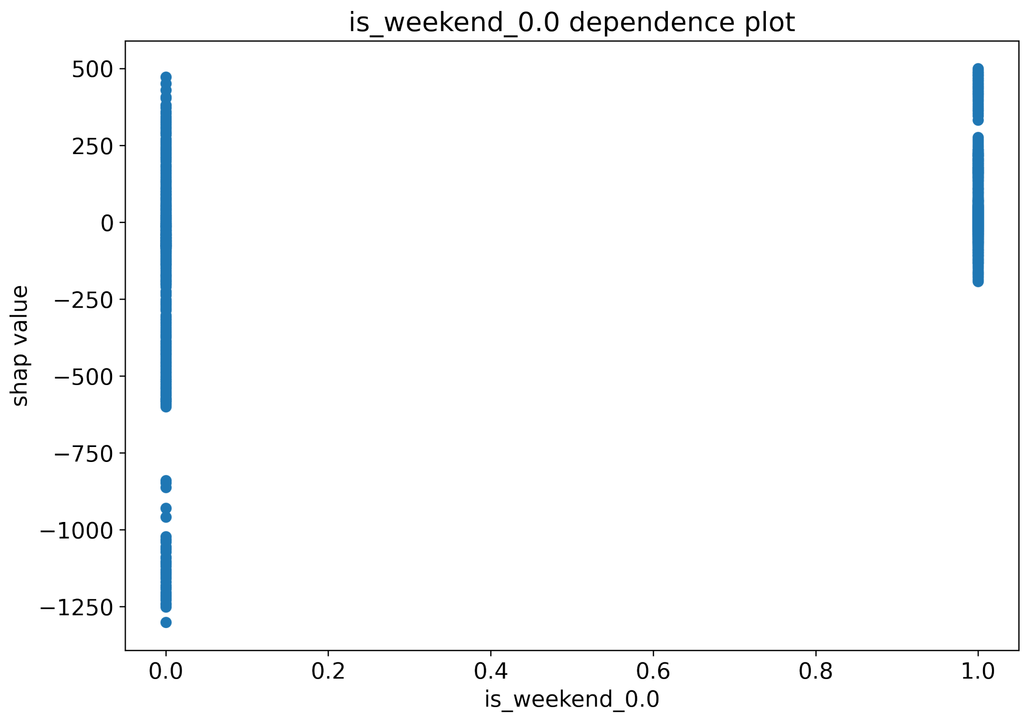

Figure 10: Local importance-is_weekend

Figure 8 shows us a strong positive correlation between the temperature and the SHAP value, which means that low temperature is associated with low bike usage and vise versa. People sleep at night so the shared bike usage must be lower than the usage in the daytime. And since the temperature at night is usually lower than the temperature during the day, it is reasonable that the temperature has a strong correlation with the prediction.

Figure 9 shows us that the humidity has a negative correlation with the SHAP value. It also makes a lot of sense since high humidity usually means raining, snowing or other bad weather and people don't tend to ride a shared bike in those kinds of weather.

Figure 10 is consistent to our intuitions. The weekend mostly contributes to high usage amount and the workday is kind of polarized.

5 Outlook

For the current solution, by combining the results of global feature importance analysis and feature selection techniques such as F-test and mutual information, we can get rid of some unimportant features and reduce the collinearity problem.

Also, this project might be improved by using time-series analysis models such as LSTM and ARIMA since the data is a time series essentially.

6 References

[1] Mavrodiev, H. (2019, October 10). London Bike Sharing Dataset. Kaggle. Retrieved October 12, 2021, from https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset.
[2] niks8411. (2021, March 29). Bike sharing prediction. Kaggle. Retrieved October 12, 2021, from https://www.kaggle.com/niks8411/bike-sharing-prediction.
[3] Maartenvandevelde. (2021, February 21). London bike share - prophet, XGB, LSTM. Kaggle. Retrieved October 12, 2021, from https://www.kaggle.com/maartenvandevelde/london-bike-share-prophet-xgb-lstm.

7 Github repository

https://github.com/lah-dee-dah/DATA1030_project.git