Statistic Machine Learning Project

Zhiwei Y. [*]zhiwei.yan@angstrom.uu.se

Abstract

To fulfill the demand of public bike that more fossil based transportation cans be replaced by bike and contribute to alleviating climate change, the city of Washington D.C. has recorded the observations high bike demand along with temporal and meteorological features. With this data, this project study is dedicated to help the city to predict the necessity of increasing number of bikes at certain hours. To achieve this goal, methods including Logistic Regression, KNN, bagging and boosting are deployed to predict the target label with given feature.

Number of group members: 4.

1 Study case and Data

To further understand and practice the knowledge we have obtained from lectures. A case of classification problem was given so that we can apply different methods on the data and understand the characteristics and behaviors of these methods.

1.1 Case and data description

To help the city of Washington, D.C. understand whether increasing the number of public bikes is necessary at some certain hour, a machine learning model is expected to predict the public bike demand for given temporal and meteorological features.

A data set contains 1600 random observations is provided for the model training. The target variable is binary for "high" or "low" demand for increasing bikes' number. The description of the features are given in Table **??**

1.2 Exploratory data analysis

Expolratory data analysis has been conducted to gain the knowledge of relations among features in the dataset. Also, following questions in the project introduction are answered in this section.

1. Which are the numerical features and which are the categorical features?
2. Is there any trend to need increase in the availability of bicycles?

Out of that the ways to treat categorical features are different from numerical features, the features are sorted into two groups, numerical and categorical. The kind of them can be identified according to Table **??**. The two groups are shown in Table **??**, which also answers Question **??**. It is important to note that the temporal features can be also treated as numeric features.

For categorical features, first step was to see the label balance. The result is shown in Figure **??**. One thing to be noted is that the feature "snow" has only one value. A flat feature will not have input to the model, thus it was excluded from training set. This is be also seen in the correlation analysis. Similarly, histogram plot was generated to visualize the distribution of numerical features (shown in Figure **??**).

[*]Deploying boosting method to given data and writing *Abstrect, Introduction, and Exploratory data analysis*

Table 1: Labels and features in the data set [**?**]

Feature Name	**Description**
midrule increase_stock (prediction label)	**low_bike_demand** – no need to increase the number of bikes **high_bike_demand** – the number of bikes needs to be increased
midrule hour_of_day	Hour of the day (from 0 to 23)
day_of_week	Day of the week (from 0 – Monday to 6 – Sunday)
month	Month (from 0 – January to 12 – December)
holiday	If it is a holiday or not (0 – no holiday, 1 – holiday)
weekday	If it is a weekday or not (0 – weekend, 1 – weekday)
summertime	If it is summertime or not (0 – no summertime, 1 – summertime)
temp	Temperature in Celsius degrees
dew	Dew point in Celsius degrees
humidity	Relative humidity (percentage)
precip	Precipitation in mm
snow	Amount of snow in the last hour in mm
snow_depth	Accumulated amount of snow in mm
windspeed	Wind speed in km/h
cloudcover	Percentage of the city covered in clouds
visibility	Distance in km at which objects or landmarks can be clearly seen and identified

Table 2: Categorical and numerical features.

Categorical features	**Numerical features**
holiday	month
weekday	temp
summertime	dew
increase_stock	humidity
hour_of_day	precip
day_of_week	snow_depth
	windspeed
	cloudcover
	snow

To understand the correlation among features, mostly between targets and other features, correlation analysis has been done ploting corrolation maps, shown in Figure **??**

Dropping out the smaller correlated features ($correlation \geq 0.1$), the left ones are: "temp", "humidity", "hour_of_day", "summertime", "dew", "weekday", and "visibility". To more intuitively see if any features can help separate the target label. Figure **??** was plotted.

Finally, with all analysis and figures above, the Question **??** can be answered.
It is seen that the label "1" ("high_bike_demand") is concentrated around daytime, growth starts from early morning, drops till late evening after reaching its peak at 15:00 to 16:00. Weekday has more "high_bike_demand" than weekends.
Temperature also has impact on bike demand, more bike are needed when temperature is in a comfortable region, 20 $\tilde{3}$0 degC in data. This also interprets the trend in the summertime, temperature is higher in summer, same as dew point.

report/figures/count_plot_category.png

Figure 1: Label counts for categorical features

2 Methodology

2.1 Data splitting

The given data is splitting into 0.8 of all data as train set and the rest as test set. The process was done with *sklearn.model_selection.train_test_splitting* function.

2.2 Machine learning models and Traning

2.2.1 Boosting

Boosting is an ensemble method which is built on the idea that even a week model can capture some patterns between target variables and given features. Starting from a base model, optmizing it based on the returned error can produce a new model which would become the base for next model. By ensembling the predictions from these models, the intention of boosting is to reduce bias[**?**].

In this study, three maintream boosting algorithm were deployed, **AdaBoost**[**?**], **GradientBoost**[**?**], and **CatBoost**[**?**]. The methods can expressed as following equations, in which α is weight in **AdaBoost** and learning rate in both **GradientBoost** and **CatBoost**. While **GradientBoost** and **Catboost** have the same mathematical expression, they differ fundamentally in how each base learner is trained. In Gradient Boosting, each learner fits the negative gradient of a specified loss function report/figures/num_dist.png

Figure 2: histgrams of numerical features

(f1-score in this study), whereas **Catboost** further procedure through ordered boosting and symmetric trees to prevent target leakage[**?**].

$$\text{AdaBoost:}\;\; \hat{y}^{(B)}_{\text{boost}}(\mathbf{x}) = \operatorname{sign}\left(\sum_{b=1}^{B} \alpha^{(b)} \hat{y}^{(b)}(\mathbf{x})\right) \tag{1}$$

$$\text{GradientBoost:}\;\; f^{(B)}(\mathbf{x}) = \sum_{b=1}^{B} \alpha^{(b)} f^{(b)}(\mathbf{x}), \tag{2}$$

$$\text{CatBoost:}\;\; f^{(B)}(\mathbf{x}) = \sum_{b=1}^{B} \alpha^{(b)}\, T^{(b)}(\mathbf{x}) \tag{3}$$

To identify the best-performing boosting model among the three, multiple experiments were conducted using different feature sets, including highly correlated features (correlation threshold > 0.1 or > 0.2) and configurations with and without cyclic encoding of temporal features. By comparison, 'temp', 'humidity', 'hour_of_day', 'summertime', 'dew', 'weekday', 'visibility' were chosen to be the features for the boosting models.

report/figures/corr_map.png

Figure 3: Corrolation among features

2.2.2 Logistic Regression

Logistic Regression is one of the most common supervised machine learning models for binary classification problems. Unlike Linear Regression, it uses a sigmoid function, where the raw output value is passed through and converted into a probability between 0 and 1. Since our problem has two possible outcomes, this model is specifically a binomial logistic regression.

Logistic Regression first calculates the linear combination of the input features:

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Here, $x_1, x_2, \ldots, x_n$ are the input feature values, $w_0, w_1, \ldots, w_n$ are the model's learned weights, and z is the model's internal score. This value is then passed through the sigmoid function to convert it into a probability between 0 and 1.

2.2.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for both classification and regression problems. It works by finding the K nearest points to a given data point and making predictions based on the majority class for classification or the average value for regression.

`report/figures/box_and_hist_features.png`

Figure 4: Boxes and Histograms for features with different targets labels

KNN is a simple, non-parametric, and lazy learning algorithm because it performs computation during prediction rather than during training.

The distance between two data points, $\mathbf{x}_1$ and $\mathbf{x}_2$, is measured using the Euclidean distance formula:

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + \cdots + (x_{1n} - x_{2n})^2} \tag{4}$$

For a new data point $\mathbf{X}_{\text{new}}$, the algorithm calculates its distance from all instances in the dataset and selects the K instances with the smallest distances. The final predicted class is determined based on the majority vote among these K nearest neighbors.

It is important to note that if K is too small, the model becomes overfitted. If K is too large, the model becomes underfitted. In our project, we selected the optimal K value using the `GridSearchCV` function.

2.2.4 Tree-based methods

Tree-based methods rely on recursively splitting the feature space into smaller, more homogeneous regions using decision rules. At each internal node, the data is split based on a single feature until a stopping criteria is met and a threshold value chosen to maximize the homogeneity of the resulting subsets.

In this study, three main tree-based approaches were explored: Decision Tree, Random Forest, and Bagging Classifier. A Decision Tree is prone to overfitting due to high variance. In contrast, the

Bagging Classifier and Random Forest Classifier reduce this variance by training many independent trees on bootstrapped samples of the dataset and averaging their predictions. The Random Forest further extends to random feature selection at each split, reducing correlation among individual trees and typically improving generalization performance.

2.3 Validation and Hyperparameter Optimization

Stratified K-fold method with 5 subset was used for cross-validation to the deployed models. Considering that label are imbalanced in the target variable, metrics like accuracy would not be a good choice for performance measurement. F-score, given by equation **??**, is used for measuring the performance in this case.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \tag{5}$$

As for tuning the hyperparameter, **Grid search** technique with given parameters grid was implemented to find the hyperparameters to get best performance out of a model.

3 Results

As mentioned in Seciont **??**, different models of four family were deployed. The result is shown in Table **??**.

Table 3: F1-score of best models

Model	**CatBoost**		**KNN**		**Logistic Regression**		**Random Forest**	
	0	**1**	**0**	**1**	**0**	**1**	**0**	**1**
F1-score	0.93	0.61	0.92	0.57	0.94	0.62	**0.96**	**0.82**
Support	270	50	270	50	270	50	262	58

4 Discussion

- Across all explored methods, tree-based approaches—particularly Random Forest demonstrated the most stable performance for both the classes, likely due to their ability to capture nonlinear feature interactions without extensive preprocessing.
- The evaluation primarily relies on the F1-Score for the minority class (High Demand, Class 1) rather than overall Accuracy. This choice is critical because the dataset is imbalanced (only $\approx 18\%$ High Demand).
- The Naive Benchmark (always predicting 'Low Demand') achieves an inflated Accuracy of 82% but has an F1-Score, Precision and Recall of 0.00 for the minority class. This demonstrates that any useful model must significantly exceed this baseline in predicting true high-demand events.
- Another significant side is the Business Impact of Errors: like for False Negatives (Low Recall, FN), the model fails to predict high demand when it is needed. This leads to business loss and customer frustration. Again, for False Positives (Low Precision, FP), the model incorrectly predicts high demand which leads to unnecessary operational costs and resource deployment.

5 Conclusion

This project addressed a challenging classification task due to the dataset's class imbalance and the fact that several features did not have balanced representation across classes, making the data difficult for standard models to learn from. An extensive exploratory analysis was therefore performed to identify meaningful trends and understand when high bike demand occurs. Initial experiments using raw features produced suboptimal results, motivating the need for feature engineering. We introduced cyclic encodings for time-related variables such as hour and month, applied appropriate scaling and encoding, and retrained the models, which significantly improved performance.

We also investigated feature elimination based on correlation analysis; however, removing most features reduced performance, so only the snow feature—containing a single constant class—was excluded. To ensure fair evaluation under imbalance, Stratified K-Fold cross-validation was used across all models. Finally, we can say that, by doing great prediction in both the classes(96% F1-score for low demand and 82% F1-score for high demand), makes Random Forest our best performing model.

References

[] Do we need more bikes? project in machine learning. Technical report, Department of Information Technology, Uppsala University, November 2024. URL `N/A`. Course: Statistical Machine Learning, 1RT700.

[] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning: A First Course for Engineers and Scientists*. Cambridge University Press, 2022. Pre-publication draft, July 8, 2022. Cambridge University Press. Available at http://smlbook.org.

[] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

A Appendix