Final Project: Data Modeling and Analysis Results for the "Capital Bikeshare System, Washington DC, U.S.A" (Term Project)

Submitted to:

Professor. Mai Abdelhakim
School of Computing and Information and Intelligent Systems Program
University of Pittsburgh

Report Prepared By:

Jiayu Chen(jic117@pitt.edu)
Xuechun Dong(xud8@pitt.edu)
Zhiming Yang (zhy61@pitt.edu)
Graduate Student, School of Computing and Information
University of Pittsburgh

April 15, 2018

Table of Content

Executive Summary

As a newly emerging public transportation in cities, bike sharing systems, the product of a new era of technologies, gradually replace the traditional bike rentals where the whole process starting from rentals to return the bike is connected online. The bike sharing systems accounts update real-time recording and personalize choices such that users have access to unlock the chosen bike at one particular location and lock the bike to terminate the short-term rental at another location. Based on the daily usage of bike sharing systems at Washington D.C., U.S.A, along with the respective weather condition and the first twenty days of each month from 2011 to 2012, our team will analyze the prediction of the total count of bikes rented for the rest of days in each month. More precisely, our team will apply the tools of machine learning to predict bike sharing demand and how likely users are willing to rent bikes in certain situation.

Introduction

Our team's report will concentrate on "Bike Sharing Demand" competition provided by Kaggle Inc. (https://www.kaggle.com/c/bike-sharing-demand).

In the "Bike Sharing Demand" competition raised by Kaggle Inc., this report will address out the accurate prediction of total count of bike rentals in the test set (test.csv) given the training set (train.csv), by Root Mean Squared Logarithmic Error (RMSLE).

Our team initially concentrates on the particular variables which influence mostly on the probability of total count of bike rentals. In other words, some of the unrelated variables are ignored while cleaning the data.

After processing both sets, it is observed that there are 10886 values with 12 columns in the training set and 6493 values with 9 columns in the test set. Before implementing the analysis, it is necessary to mention that the characters in the columns named as "season", "holiday", "workingday", "weather" have been converted into numbers.

Afterwards, three methods are considered to model the prediction of total count of bike rentals: Linear Regression Model in Python, Adaboost in Python, Random Forest in Python, and Gradient Boost and XGBoost in Python. The specific procedures and description will be provided in the next section.

Finally, the test set will evaluate each model that will generate different accuracy of analysis because of the different variables chosen. The submission on Kaggle's website will return the score, which is the percentage of passengers our team correctly predicts, known as "RMSLE".

Background

Bike sharing systems adopt dynamic pricing model regarding to environmental and seasonal settings, including weather conditions, season of year, day of week, hour of the day, etc. to maximize the market value. The settings are all considered as influential features in the dataset. The following are the features in both sets and the descriptive statistics of each feature generated in Python.

datetime	hourly date + timestamp
season	1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday	whether the day is considered a holiday
workingday	whether the day is neither a weekend nor holiday
weather	1: Clear, Few clouds, Partly cloudy, Partly cloudy
	2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
	3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
	4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp	temperature in Celsius
atemp	"feels like" temperature in Celsius
humidity	relative humidity
windspeed	wind speed
casual	number of non-registered user rentals initiated
registered	number of registered user rentals initiated
count	number of total rentals

Table 1.1 The variables and definitions of each variable

	season	holiday	workingday	weather	temp	atemp
count	10/20/1929 0:00	10886	10886	10886	10886	10886
mean	1/2/1900 12:09	0.028568804	0.680874518	1.418427338	20.23085982	23.65508405
std	1/1/1900 2:47	0.166598851	0.466159169	0.633838586	7.791589844	8.474600626
min	1/1/1900 0:00	0	0	1	0.82	0.76
25%	1/2/1900 0:00	0	0	1	13.94	16.665
50%	1/3/1900 0:00	0	1	1	20.5	24.24
75%	1/4/1900 0:00	0	1	2	26.24	31.06
max	1/4/1900 0:00	1	1	4	41	45.455

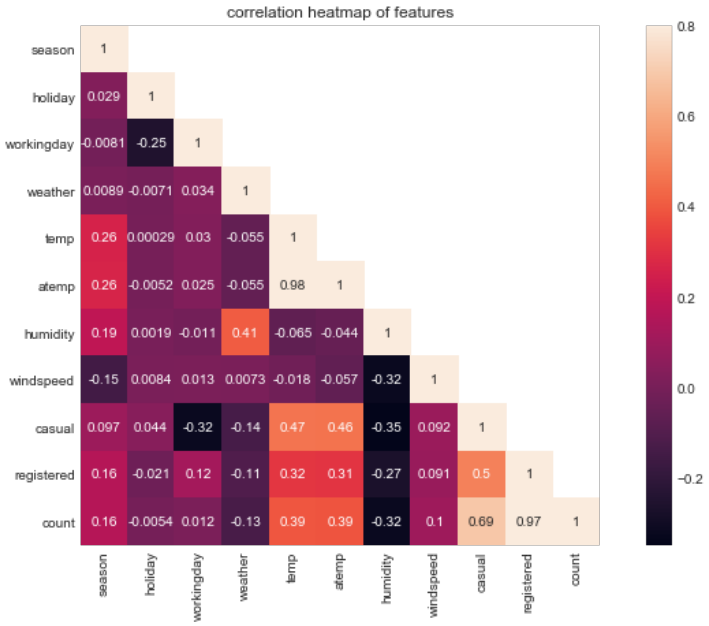
	humidity	windspeed	casual	registered	count
count	10886	10886	10886	10886	10886
mean	61.88645967	12.79939541	36.0219548	155.5521771	191.5741319
std	19.24503328	8.164537327	49.96047657	151.0390331	181.1444538
min	0	0	0	0	1
25%	47	7.0015	4	36	42
50%	62	12.998	17	118	145
75%	77	16.9979	49	222	284
max	100	56.9969	367	886	977

Table 1.2 & 1.3 The variables and definitions of each variable

Limitations

According to characteristics of variables, our team selects features to build machine learning models, which are used to learn and find relationships between features and target. While cleansing both train and test data, our team runs correlation analysis between features and target, however, none of features have strong relationship (more than 0.5) with target, which raise a concern whether the accuracy can be expected ideally.

Correlation Figure



Although there does not exist missing values in both datasets after reviewing data info in Python, the missing values might be prefixed by data provider with inappropriate method. For example, the "windspeed" feature has many zero (0) values, leading to reconsider whether zero values represent that windspeed cannot be detected or are the fillings of empty values. Accordingly, tuning the parameter ("windspeed") seems to be a challenge hard to handle properly.

Feature Engineering

Data Source: https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset

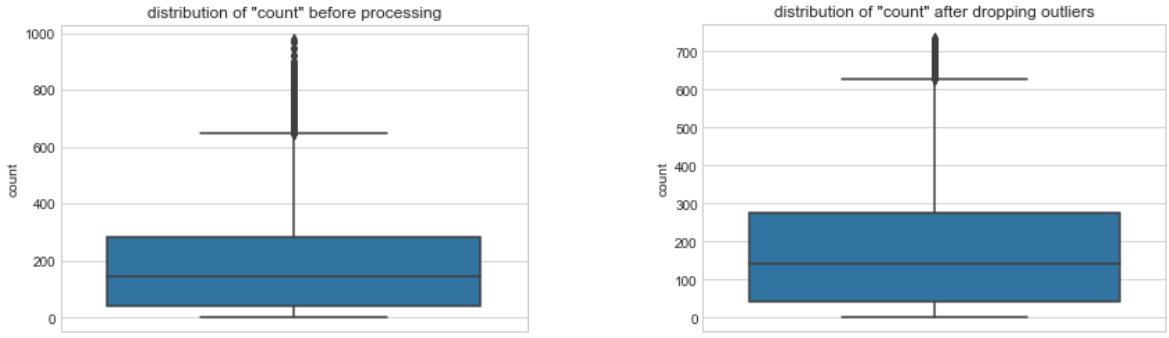
Data Type:

- Qualitative features: season, holiday, workingday, weather
- Quantitative features: temp, atemp, humidity, windspeed, casual, registered

1. remove the outliers for the train set.

From the above Table 1.3, our team finds that the 50 percentile of 'count' is about 141, the 75 percentile of 'count' is about 284. However, the max of 'count' is about 977. So our team estimates that there are some outliers in datasets.

Then, our team creates the box plot for the ‘count’ in the training dataset. From the obvious show in the box plot, our team confirm that there are some outliers and our team must remove them to prevent its impact on our predictions. Our team choose to remove the elements which go beyond 3-std scope.

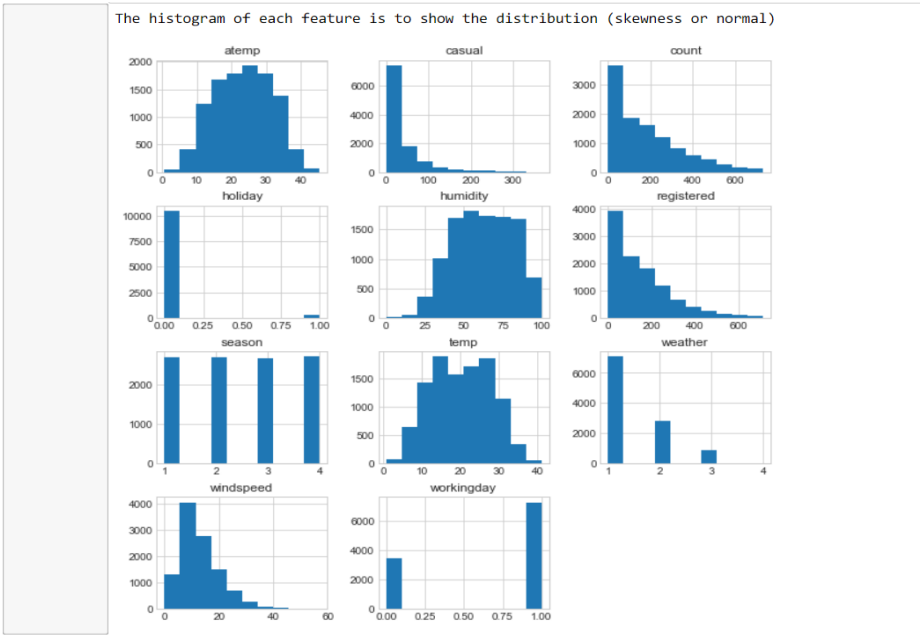


2. separate “datetime” feature into applicable formats in both training and test sets.

Due to the feature of datetime is object type, our team separates it into four columns which are year, month, weekday and hour.

		season	holiday	workingday	weather	temp	atemp	humidity	windspeed	hour	year	weekday	month
train	**0**	1	0	0	1	9.84	14.395	81	0.0	0	2011	5	1
	1	1	0	0	1	9.02	13.635	80	0.0	1	2011	5	1
	2	1	0	0	1	9.02	13.635	80	0.0	2	2011	5	1
	3	1	0	0	1	9.84	14.395	75	0.0	3	2011	5	1
	4	1	0	0	1	9.84	14.395	75	0.0	4	2011	5	1

3. visualize data and examine the distribution of each numerical feature.



Predict “windspeed” using Random Forest Model. From train and test datasets, our team finds a large percentage of zero values in the feature of “windspeed”. According to our view, there are several assumptions for this phenomenon:

1. These zero values are missing values
2. These zero values are wind speeds under 5, nearly round to zero
3. These zero values are exactly zero

After discussion, our team thinks the second hypothesis can be solved by filling zero values with random numbers between 0 to 5. However, this kind of method has no basis, which is same as zero values in the beginning. And the third hypothesis leads to a strange distribution. Therefore, our team chooses the first assumption to process zero values of “windspeed”.

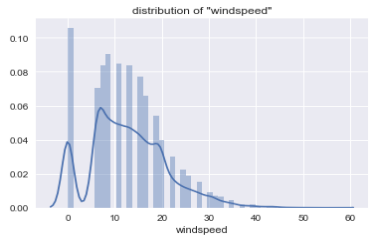


Figure: “windspeed” distribution in train dataset before processing

After dropping outliers in train dataset, our team calculates the correlation coefficient value between “windspeed” and count which is 0.12068. Our team combines train and test datasets as a full dataset because larger data volume would get a better train model and do an optimal prediction of missing values in “windspeed”. Then our team extracts the part where “windspeeds” are larger than zero as known_wind and split it into train and test parts, which can avoid overfitting. The model our team chooses to predict missing wind speeds is RandomForestRegressor in sklearn.ensemble because this feature is continuous. The method of tuning parameters our team uses is grid search with cross-validation to find the best parameters’ combination of our model.

The best accuracy score of predicting wind speed our team gets is nearly 0.505. Our team has tried to do a feature selection using SelectPercentile before using model but the accuracy score is not ideal. Afterwards, our team uses model with suitable parameters to fit all known_windspeed and predict and fill values of unknown “windspeed”.

Again, the correlation coefficient value between “windspeed” and count after filling missing values is 0.1267, which means that the filling of missing values is helpful to predict count.

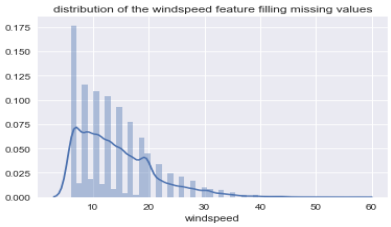


Figure: “windspeed” distribution in train dataset after filling missing values

Methodology

Our team implements four analysis techniques: Linear Regression Model in Python, Adaboost in Python, Random Forest in Python, and Gradient Boost and XGBoost in Python.

I. Linear Regression Model with MinMaxScaler method in Python

LinearRegression Model is a supervised machine learning algorithm which can be used for regression problems. Since the feature of count is continuous, our team chooses this simple model to calculate the accuracy of prediction.

our team firstly uses this model to fit train and test sets splited from Train dataset but both R^2 scores of X_train and X_test are low, which means the linear regression model is underfitting and have not been able to capture variability in data. However, linear regression model has no parameters. So there is no way to control model complexity.

In order to solve the problem of underfitting and improve the accuracy, our team does feature scaling to preprocess data in order to unify the range of each feature and make a better fit of training data. MinMaxScaler is a feature scaling method which can transform the range of data between 0 to 1.

Our main goal is to reduce overfitting, to avoid underfitting and to choose a generalized model which is accurate for new test examples. To test accuracy of the fitted model our team applies cross validation techniques like k-fold cross validation and leave-one-out cross validation. Our team also researches on ensemble learning to improve the accuracy of the fit.

II. Random Forest Model method in Python

Random Forest is a type of classification and regression algorithm which can be implemented with the enormous data. It is an ensemble learning method where two or many models are generated and merged to solve a specific computational analytical problem. Firstly, the tree structure of the project would be,

Also, our team utilizes Random Forest classification to calculate the importance of features.

Random Forest								
Features	season	holiday	workingday	weather	temp	atemp	humidity	windspeed
Feature Importance	0.0623326	0.008602	0.0457959	0.0525968	0.1665608	0.1980519	0.2553078	0.2107523
Score	0.822952192							

Class of model:

$$g(x)=f_0(x)+f_1(x)+f_2(x)+...$$

where g is the sum of simple base models f_i (base classifier: decision tree)

Since our case has large amount of training instances and mostly underfitting, out teams decides to set number of trees as large as memory allows, low minimum sum of instance weight (hessian) needed in a child and increasing maximum depth.

III. Gradient Boosting and XGBoost Model method in Python

XGBoost (Extreme gradient boosting) is a gradient boosting algorithm. The principle used in XGboost is similar to the working of gradient boosted trees which uses additive training approach to boost the performance of the algorithm. The model uses combination of regression and classification trees (CART). In such models the leaf nodes are associated with additional weights of positive classification known as leaf scores. The ultimate respective leaf score of the ensemble tree is the respective sum of the leaf scores in set of trees under consideration. Our team need to find set of trees that optimizes the leaf scores. What additive learning does is to add up one tree at a time, and to add up one new tree at another time.

T = (t1, t2, t3) where T represents the whole tree with three subset trees

X1 = x1(t1) + x1(t2) + x1(t3) where X1 represents one feature of T and x1(t1), x1(t2), x1(t3) represents features of tree t1, t2, t3 respectively.

Similarly, X2 = x2(t1) + x2(t2) + x2(t3) and X3 = x2(t1) + x2(t2) + x2(t3)

This can be mathematically written as,

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

Where, K is the number of trees, f is a function in function space F, and F is a subset of all possible CARTs.

In order to find the combination of trees which optimized the leaf scores our team adds one new tree to the set at a time. The tree which our team adds should optimize the objective function.

$$= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$$

Where "t" is tree which is added, and "l" is the cost function like MSE (mean square error)

IV. Adaptive Boosting Model method in Python

AdaBoost (adaptive boosting) is an ensemble learning algorithm that can be used for classification or regression. It uses multiple iterations to generate a single composite strong learner and it is more resistant to overfitting. Since the feature of count is continuous, our team chooses the Adaboost Regressor model to do the prediction and adjust the number of n_estimators to get the best score.

Instrumentation

Python built-in library:
Pandas (read the files), NumPy (create N-dimensional array object), Scikit-Learn (machine learning), Time (create random numbers), MinMaxScaler

Machine Learning Algorithms:
Linear Regression, AdaBoost Regressor, Random Forest Regressor, Gradient Boosting Regression, XGBoost Regressor

Results

Submissions are evaluated by the Root Mean Squared Logarithmic Error (RMSLE).

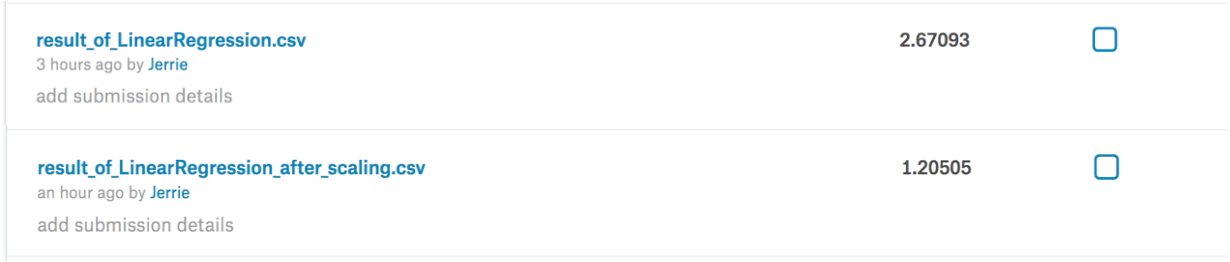
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

Where:

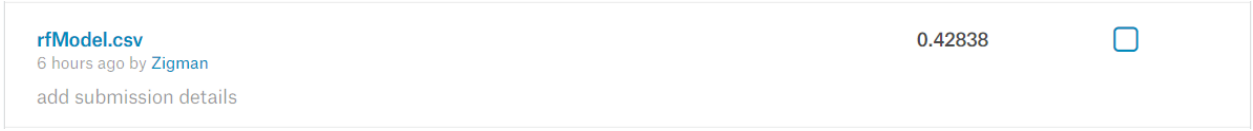
- n is the number of hours in the test set
- pi is your predicted count
- ai is the actual count
- log(x) is the natural logarithm

The following results display the scores by each methodology returned on Kaggle's website.

For the method **Linear Regression in Python**, the RMSLE score after the submission of prediction is 2.67093. Then using transformed datasets to fit Train dataset, our team obtains a better score after submission which is 1.20505.

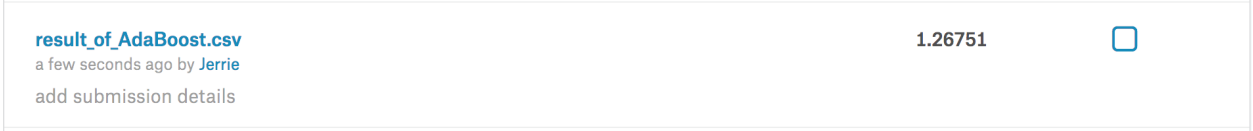


For the method **Random Forest in Python**, the RMSLE score after the submission of prediction is 0.42282.



For the method **Gradient Boosting in Python**, the RMSLE score after the submission of prediction is 0.42282. For the method **XGBoost in Python**, the RMSLE score after the submission of prediction is 0.40122.

For the method **Adaptive Boosting in Python**, the RMSLE score after the submission of prediction is 1.26751.



Conclusion

As discussed above, the "windspeed" can be regarded as three different scenarios, which is surely a huge challenge before applying any machine learning models. The procedure of dealing with missing values is only filled by Random Forest classification on the basis of the chosen features. Plus, the feature selection cannot be precisely done since all features are weakly correlated with target. Therefore, the "Limitations" section briefly describes the obstacles our team faces and the challenge our team cannot conquer to achieve higher accuracy.

In summary, our team applies three categories, five kinds of methods to do the predictions：Linear Regression Method, Random Forest Method and Boosting Method (Gradient Boosting, XGBoost and Adaptive Boosting). The best RMSLE score our team gets is the method XGBoost in Python and its score is 0.40122.

Specifically, Linear Regression model with or without feature scaling does not perform good in general, and the reasons can be 1) the model is so simple that fitting process leads to underfitting; 2) our dataset has both numerous and categorial features that the model is unable to be flexible.

Random Forest and Boosting algorithms improves the accuracy at a much higher level, and the reasons can be 1) these ensemble models are good at handling tabular data with qualitative and quantitative features; 2) these ensemble models are able to capture non-linear interaction between features and target.

Future Plans

Instead of using GradientSearchCV, RandomizedSearchCV can optimize the methods by cross_validated search over parameter settings on continuous features.

In our project, our team crosses out "casual" and "registered" features. There should be ways to cope with the dependent variables in the dataset and to generate different results.

Responsibilities

Jiayu Chen:

- features engineering
- outlier detection and treatment
- Linear Regression model

Xuechun Dong:

- data exploration and preparation
- missing value treatment
- AdaBoost model

Zhiming Yang:

- data cleansing and find models
- Random Forest, Gradient Boost and XGBoost model
- further improvement

References

Data Science Central. Introduction to Classification & Regression Trees (CART). 3 December 2017. <https://www.datasciencecentral.com/profiles/blogs/introduction-to-classification-regression-trees-cart>.

DMLC. Introduction to Boosted Trees. 3 December 2017. <http://xgboost.readthedocs.io/en/latest/model.html>.

Fanaee, Hadi T and Joao Gama. "Event labeling combining ensemble detectors and background knowledge." Progress in Artificial Intelligence (2013): 1-15. Springer Berlin Heidelberg.

Kaggle Inc. Bike Sharing Demand. 28 May 2015. <https://www.kaggle.com/c/bike-sharing-demand>.

MathWorks. AdaBoost. n.d. 15 April 2018.

Muller, Andreas and Sarah Guido. Introduction to Machine Learning with Python. O'Reilly Media, 2016.

Ray, Sunil. "Kaggle Bike Sharing Demand Prediction." 25 June 2015. Analytics Vidhya. 15 April 2018.

scikit-learn. MinMaxScaler. n.d. 15 April 2018.

scikit-learn. RandomizedSearchCV. n.d. 15 April 2018.

turi. Turi Machine Learning Platform: Random Forest Regression&Gradient Boosted Regression Trees. n.d. 15 April 2018.

Appendix I

Tuning parameters with cross-validation applying to “windspeed” feature

```
def tuningPara_crossValid(model, param_grid, X_train, X_test, y_train, y_test):
    kfold = KFold(n_splits=5, random_state=0, shuffle=True)
    grid_search = GridSearchCV(model, param_grid, cv=kfold)
    grid_search.fit(X_train, y_train)
    accuracy = grid_search.score(X_test, y_test)
    model_after_tune = grid_search.best_estimator_
    print('Best parameters: {}'.format(grid_search.best_params_))
    print('Best cross-validation score:', grid_search.best_score_)
    print('The accuracy score is:', accuracy)
    return model_after_tune
```

```
from sklearn.ensemble import RandomForestRegressor
X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state=0)
param_grid = {'n_estimators': [25, 50, 70, 100],
              'max_depth':[5, 10, 15, 20]}
rfr = tuningPara_crossValid(RandomForestRegressor(max_features=4, n_jobs=-1),
                            param_grid, X_train, X_test, y_train, y_test)
```

```
Best parameters: {'max_depth': 20, 'n_estimators': 100}
Best cross-validation score: 0.47853688953024687
The accuracy score is: 0.5029618828061957
```

Appendix II

Function: Root Mean Squared Logarithmic Error

```
# The function of calculating RMSLE
def rmsle(y_true,y_pred):
    assert len(y_true) == len(y_pred)
    return np.square(np.log(y_pred + 1) - np.log(y_true + 1)).mean() ** 0.5
```