**PROJECT REPORT**

**CMPE 255 - Data Mining**

**SF bay area bike share analytics and predict**

**Team 5**

**Team Information**

Jie Liu 015331121

Xialu Zou 011353316

Lingxiang Hu 015230631

Chaoran Lei 015264119

**TABLE OF CONTENTS**

1. **Introduction**
   1. Motivation
   2. Objective
2. **System Design & Implementation details**
   1. Algorithms considered/selected
   2. Technologies & Tools used (and why)
   3. Architecture-related decisions (if applicable)
3. **Experiments / Proof of concept evaluation**
   1. Dataset(s) used
   2. Preprocessing
   3. Methodology followed(e.g. n-fold-cross validation, number of folds, size of training/test set etc.)
   4. Analysis of results
4. **Discussion & Conclusions**
   1. Decisions made
   2. Difficulties faced
   3. Things that worked
   4. Things that didn’t work well
   5. Conclusion
5. **Project Plan / Task Distribution**
6. **References**
7. **Appendix**
8. **Introduction**
   1. **Motivation**

Bike Sharing Network is a newly emerging industry especially in large cities and downtowns. This industry not only provides convenience in commuting, but also helps in environmental protection. However bike sharing is not necessarily profitable in every city. This depends on the local climate, topography, population composition, and other elements. Bay area is an area we are familiar with. The flat terrain and low rainfall climate are also ideal for bike sharing. It has one of the largest city in USA, San Francisco. That’s why we decided to explore bike sharing data in the bay area.

* 1. **Objective**
     1. Reformat the data properly, retrieve useful information from the dataset. Eliminate the noises.
     2. Explore bike sharing information in five cities: Mountain View, San Francisco, Palo Alto, Redwood City, San Jose. Find the patterns and develop different models based on different algorithms.
     3. Evaluate different models results to decide on the best choice.
     4. Make conclusions based on our works.

1. **System Design & Implementation details**
   1. **Algorithms considered/selected**
      1. ***Linear Regression***

Linear regression is a linear approach for modelling the relationship between a scalar response and one or more explanatory variables

* + 1. ***Logistic Regression***

The logistic model is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

* + 1. ***Random Forest Regressor***

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

* + 1. ***Gradient Boosting Regressor***

Gradient boosting builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

* + 1. ***AdaBoost Regressor***

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

* + 1. ***Xgboost***

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework.

* + 1. ***Multi-layer Perceptron regressor***

This model optimizes the squared error using LBFGS or stochastic gradient descent. MLPRegressor trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting. This implementation works with data represented as dense and sparse numpy arrays of floating point values.

* 1. **Technologies & Tools used (and why)**

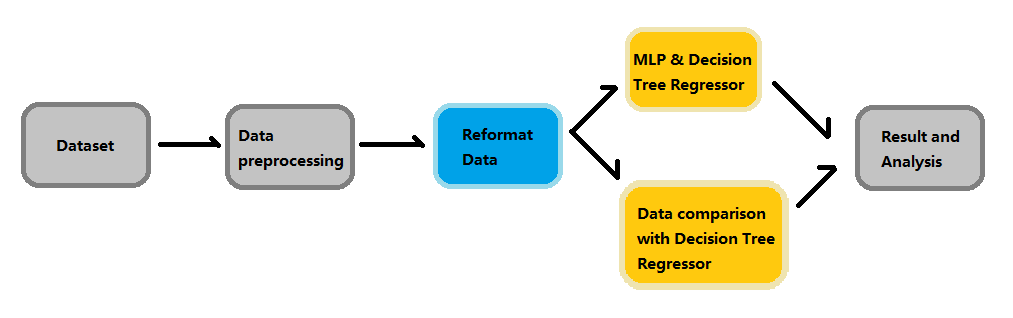
***Pandas***: Specially designed for processing tables and mixed data, which is more in line with the table structure in statistical analysis.

***Numpy***: numpy is a matrix-based mathematical calculation module that processes unified numerical array data.

***Sklearn***: Sklearn has methods that can be used for supervised and unsupervised learning. Estimator is used for data prediction or regression, and Transformer is used for data processing.

***Seaborn and Matplot Library***: Visualize the data and generate intuitive data graphs for analysis.

* 1. **Architecture-related decisions (if applicable)**



1. **Experiments / Proof of concept evaluation**
   1. **Dataset(s) used**

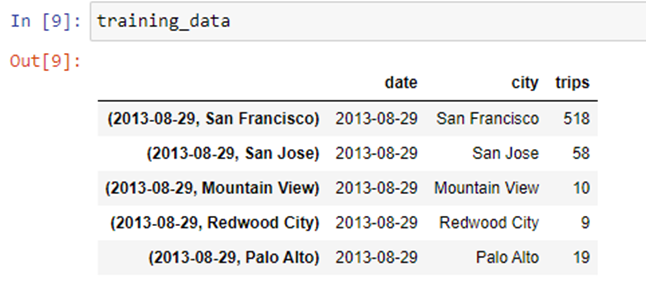
Dataset link: <https://www.kaggle.com/benhamner/sf-bay-area-bike-share>

This Datasets contains four tables:

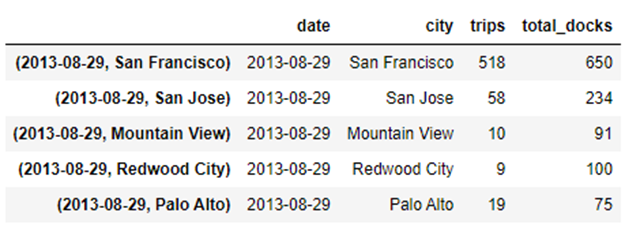
1. Station: contains information about each station, including the docks amount, location.
2. Trip: including the date, time, location,duration of each trip.
3. Weather: including weather details in each city everyday.
4. Status: including the docks and bikes available every minute in each station.
   1. **Preprocessing**

To be able to predict the trip amount from the desired city, we need to retrieve the useful information from three tables: station, trip and weather. Here are the processes of preprocessing.

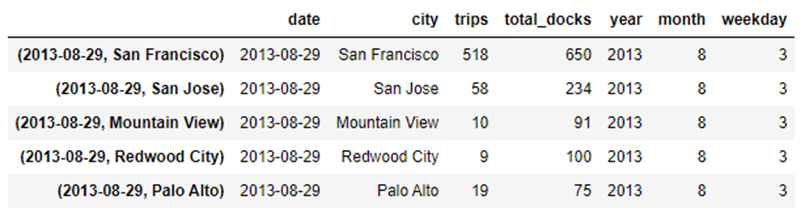
* + - * Get each trip and merge it with the station on station\_id. Only keep the not na values and get rid of the non-valid zip-code.
      * Eliminate the trips that are longer than 5 hours and less than 1 min.
      * Iterate through each trip and calculate total trips of each day in each city.



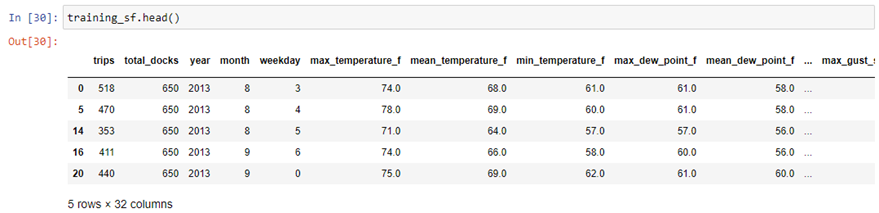
* + - * Add up docks installation in each station to find out the total docks available in each city on each date.



* + - * Change date to three attributes: Year/Month/Weekday.



* + - * Add a weekend feature to training data.
      * Processing the weather data. Change the event attribute to 5 Attributes: Fog, Fog-Rain, Rain, Rain-Thunderstorm, Normal. Use 1 to represent happens, use 0 to represent does not happen.
      * Join trips data with weather data on date and city. Then split the data based on the 5 cities: San Francisco, San Jose, Mountain View, Redwood City, Palo Alto.



* 1. **Methodology followed**

To find the appropriate algorithm that could fit the training data set and make the prediction better, we decided to apply several different algorithms from sklearn and xgboost to our project. At first, we split the whole dataset into a train and test dataset in a ratio of 8/2. We used the score method in sklearn to validate the model on the test data set. However, most of the time, the scores are so similar that we cannot identify which one is better than the others. Thus we adopted the cross validation score from sklearn, with folds of 5 and median absolute error regression loss for validate. Once validation was complete, we picked top 2~3 models to predict on the test dataset.

**Parameters setting:**

**Parameters of the Random Forest Regression:**

Random forest is one of the ensemble methods which performs Regression, where estimators are a large number of small independent decision trees. We have considered the no of estimators as 55.

**Parameters of the Gradient Boost Regressor:**

Gradient boosting is an ensemble method. It produces a strong learner through a combination of weak learners in an iterative fashion. We are using this model because the accuracy of the predictive results is higher as it minimizes the predictive error altogether. In this project we set the maximum depth as 8, no of estimators as 150 and minimum sample leaf as 1.

**Parameters of the Logistic Regression:**

Logistic regression is a predictive analysis method. It is a linear model used for classification and before applying it to this model we scaled our data. In this project we set the solver parameter is ‘***liblinear’***.

**Parameters of the AdaBoostregressor**:

AdaBoostregressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. In this project we set the estimators is 100 and loss is ‘***linear’***.

**Parameters of the Decision Tree Regressor:**

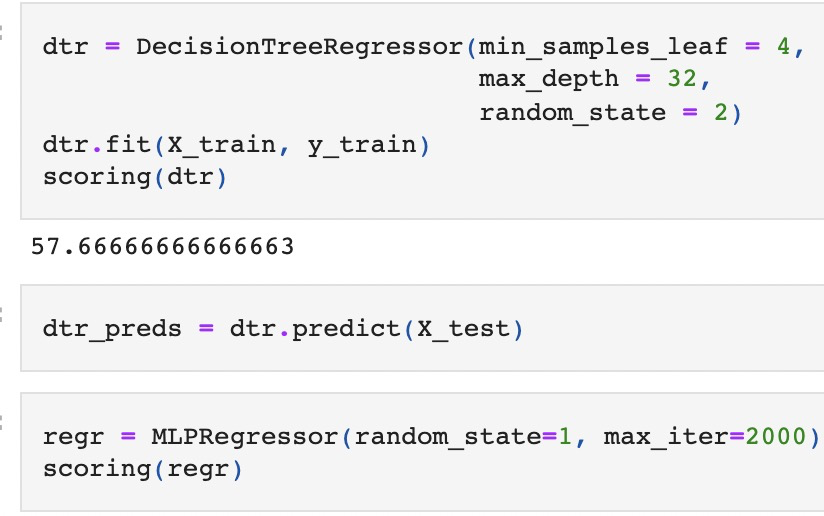
Decision tree regression is a supervised learning which splits the data into a format of tree based on test condition given. We have chosen maximum depth of tree as 32 in this project.

**Parameters of the Multi Layer Perceptron:**

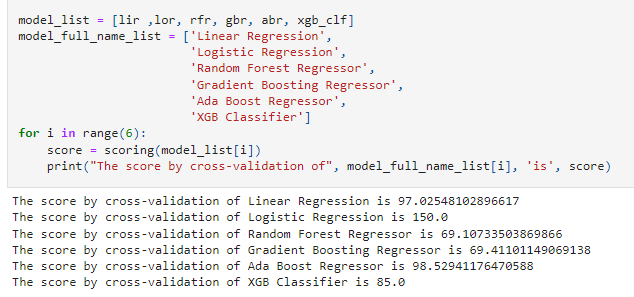
Multi-layer perceptron is a deep learning methodology. This deep neural network learns non- linear functions. It has hidden layers. The layers of input nodes produce different sets of outputs. We are using this algorithm to learn the non-linearity factor in the data. The maximum number of iterations taken is 2000.

Following picture showing different algorithms with their hyper parameters.

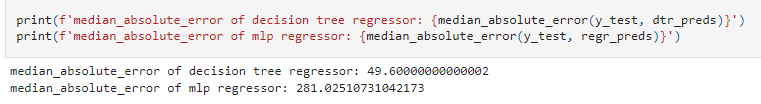


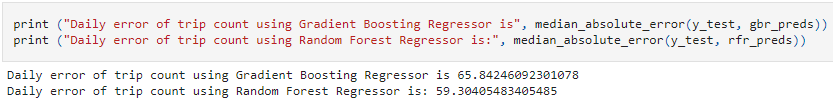


Validate different algorithms with cross validation score from sklearn, with folds of 5 and median absolute error regression loss.



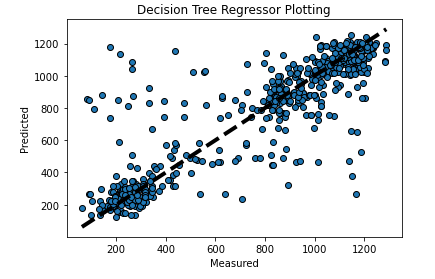
Choose top 2~3 models to predict on the test dataset of 5 different cities.

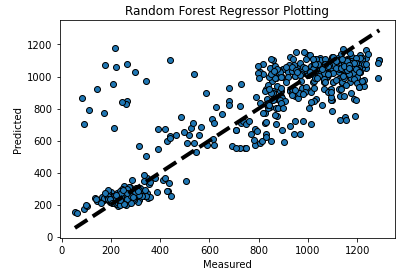




**Evaluation Metrics:**

Use SanFrancisco as an example to show the algorithms evaluated. We found that decision-tree and random-forest are the most suitable models.





* 1. **Analysis of results**

The production we made using random forest and decision tree are pretty effective for San Francisco, San Jose and Mountain View. From the plotting, we can see that the prediction value follows the trend of the real data.

|  |  |  |
| --- | --- | --- |
|  | random forest | decision tree |
| San Francisco | https://lh3.googleusercontent.com/b9ZVWwY-qb1L41w9_x14-ABWQJCG2xB9pYJPn0MaySCtgfg0xVDkMma-bYgazL45UIuDHTsyozJdZxA2XG7I3dD03YHlHfKGx8IYDMsvFFLwJcOPr46NAHt5RuyACm6KdYVaJwPC | https://lh5.googleusercontent.com/4r1ysZXWq1QE6EFrlgJFnCtYWGbO07m7mPsZJ-HayQKRLX1Jc4TEVsUm-mUeA6l0p6_JlnVbLsm-DZmCNScHwL2pMwqsZsS4hT3mBAj_q5wxUa0UDzIBjovTFEpIPQql-j1ZHoa1 |
| San Jose | https://lh4.googleusercontent.com/5CAcw3hXCJmtKn_mCJZFz03IgDgsKFHKO-0fLM09HX1gcE9Vqpr98XBJ-Imyl1T1nGkFCVtI0xaLcQAlcyh7a0f5wOAqp_STeWxOSVr_iPOLCOTBBWUWCWUdSP3-Qt14dF9cfLTI | https://lh5.googleusercontent.com/PdF0OU6HH-BVA9zy5u5r1K6pG1VBE7wUkZLkPl037co_hrl3wc8N2G88PfZ85I55twU3jYJ02KDp-_67doXy3qhNJJqBBB__81nFwszyPFaMBvSQedGfhNEiWvqYrs9IKHHHqGvU |
| Mountain View | https://lh3.googleusercontent.com/SkxM5xFtRc9aV8L8aTh9JqPCfrEDRK30-p21XBH0PHNilTBG6CxtFuvLUqDYjBeK6jNbjtI-qecaSaHezKK7cGVcTRXtU-yn8G6qfPjP0v9LnPDVZGdrAAL03JGFWjSyKiPV6liL | https://lh6.googleusercontent.com/S4DTYGDGTBRXmjbUZ-WPfbeqzIfRwSiZrXyVnXvD3b54oy6wlk7c02NROM8b8Enx_yiIqM3ZQvIBnXJPXrH0F70GadptOamL3NfYJcG5BsqgUZteuEjBVoG7RVe1690WZecAgg1w |

However, when we apply the same model on Redwood city and Palo alto, the trip count is very limited, so the prediction is not as accurate as the ones in other cities. Even though, under this circumstance, the decision tree algorithm is doing better than random forest.

|  |  |  |
| --- | --- | --- |
|  | random forest | decision tree |
| Redwood City | https://lh6.googleusercontent.com/yiP32Q8s_DX6xI4UaihqRQ1fp5_QGFFaP7aaFL2jmxeAr1GvMU9VdyjiEe4oZLFdSOOInF497CNkc_-fuwVNxOnluMsDumr2tr16g84RM4ejUT5Ul93_BvgplztNL2tWckaGOyeO | https://lh6.googleusercontent.com/Ju9oSLCYyHRe8VhyZdbE3q3sZVBgYs1mYoI7r5g49XV3iVCWHMteKZkvN-dxb3-7mauYIX3RPjxwA2oi1sXRLRyv01-Ln1_nxT0pvoe__jC6XMt9h7_dqPGfrFM_GXuYQ7mOu4gW |
| Palo Alto | https://lh4.googleusercontent.com/OnKaKsSSOmkctb9hCc7DY-91_na8ZtiWcgNVsU5ndmA2Yom9cmB40RJwZaWwggtFpDKlxalkyUjFJWtbLa2JuL_4oyBAO0tZsphumE2Ctz4YRswcRa3Y5yttgcEzTjRQxbeF7wKM | https://lh5.googleusercontent.com/syZpURFoaa4FmOoUJkYRsATCvTddAcOvt1mALuw-fVOlrlSBfLR_j9Au2ykGUz0lGK5ycXShTpixqA5Jib4Ssz7levqSrK9mBexRo0U6cZmlOCSvUIoqQYRxiAfCRpyIRf7IoHm2 |

1. **Discussion & Conclusions**
   1. **Decisions made**
   2. **Difficulties faced**
   3. **Things that worked**
   4. **Things that didn’t work well**
   5. **Conclusion**

Through this project, we use a variety of algorithms to analyze the laws of bicycle travel in different regions and use them for prediction. For cross validation, we found that the Ada boosting algorithm works very well, indicating that it can greatly improve the overall performance. For prediction, we found that decision-tree and random-forest are the most suitable models.

1. **Project Plan / Task Distribution**

|  |  |
| --- | --- |
| Xialu Zou | Data Preprocessing, Model Plotting |
| Chaoran Lei | Model Training |
| Lingxiang Hu | Validation, Summary |
| Jie Liu | Model Training |

1. **References**
2. Hinton, Geoffrey E.“Connectionist learning procedures.” Artificial intelligence 40.1 (1989): 185-234.
3. Glorot, Xavier, and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks.” International Conference on Artificial Intelligence and Statistics. 2010.
4. He, Kaiming, et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.” arXiv preprint arXiv:1502.01852 (2015).
5. Kingma, Diederik, and Jimmy Ba. “Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014).
6. https://pandas.pydata.org/docs/ [3]<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html>
7. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html>
8. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html>
9. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html>
10. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>
11. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>
12. <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html>
13. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
14. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
15. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>
16. pandas.tseries.offsets.DateOffset — pandas 1.3.4 documentation. (n.d.). Pandas. <https://pandas.pydata.org/docs/reference/api/pandas.tseries.offsets.DateOffset.html>
17. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. <https://scikit-learn.org/stable/about.html#citing-scikit-learn>
18. sklearn.neural\_network.MLPRegressor. <https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html>
19. **Appendix**

Link to Github Repository:

<https://github.com/Xialu0803/255-SFbike-data>