BIKE DEMAND PREDICTION

P. Akshay Kumar, MT19094
Indraprastha Institute of Information Technology
New Delhi, India
akshay19094@iiitd.ac.in

Abstract

Aim is to predict the number of bikes bookings on an hourly basis in the state of Los Angeles. Dataset from LA bike website and Kaggle dataset corresponding to weather information were pre-processed and merged. Correlation and Grid Search CV techniques were used for feature selection and parameter tuning. Random Forest, Gradient Boost and XGBoost regression techniques were applied on the dataset and best prediction was obtained of that of XGBoost.

I. Introduction

1. Problem Statement

Predicting the number of bike rentals in Los Angeles on an hourly basis.

2. Datasets

The bike dataset is fetched from the LA Metro Bike dataset corresponding to the years 2016 and 2017.The weather and external factors, US holidays dataset is fetched from Kaggle.

Training Dataset - Last Quarter of 2016, First 3 Quarters of 2017

Testing Dataset - Last Quarter of 2017

II. Data Preprocessing

The bike rentals dataset consisted of data corresponding to trips between various stations throughout the day.

Only the columns corresponding to timestamp and the type of passed were retained and rest all were dropped since these columns seemed relevant to the problem statement.

Bike rental type in original dataset were ‘walkup users’ and ‘flex pass’ users and these are mapped to ‘Casual’ and ‘Registered’ respectively.

The various timestamps in the dataset were rolled up to the nearest hours and the count of registered, casual users, total users were updated in the dataset.

The weather dataset was of 36 different cities. The data corresponding to city of Los Angeles was fetched.

The weather and other external datasets corresponding to humidity, wind speed, pressure, humidity was rolled up to the nearest hour to map to the original dataset from LA bike website.

The temperature in temperature.csv was in Kelvin which was converted to centigrade.

The weather dataset had values such as ‘scattered clouds’, ‘heavy thunderstorms’ were encoded to integers based on ideal weather for cycling. For example: 1- Ideal weather for cycling…4- Extreme weather – Unsuitable for cycling.

The holiday dataset consisted of US public holidays from the year 1966. The data corresponding to 2016 and 2017 were fetched.

The weather, humidity, wind speed, humidity, pressure, holidays dataset was merged with the original dataset.

A new csv corresponding to season was created and merged since bike riding is highly popular in the spring season and gives a high correlation.

III. Feature creation

The day, hour, year and day of the week attributes were fetched from the timestamp field in the dataset.

The following other features were created.

'hour_workingday_casual' – Number of casual users riding on bikes on a working day during the working hours.

'hour_workingday_registered' – Number of registered users riding on bikes on a working day during the working hours.

‘is Weekend’ – Check if a day is weekend or not.

IV. Feature Selection

1. Correlation:

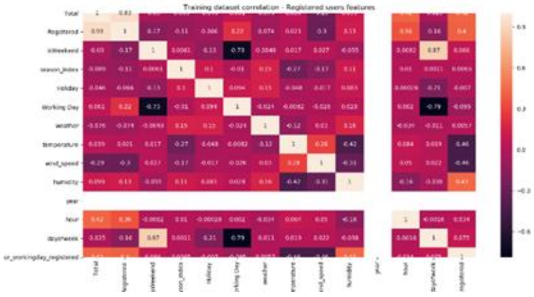
Initially correlation matrix was constructed corresponding to all the features with respect to Total.

‘Wind Speed’ was highly negatively correlated with both Casual and Registered set of users.

‘Working Day’ attribute was highly negatively correlated with Casual set of users since casual users are more likely to use the bikes on holidays and registered users use it more on working days.

‘isWeekend’ attribute was highly negatively correlated with registered set of users since registered users are more likely to avail the service on working days than weekends.

The ‘windspeed’ attribute was dropped for both casual and registered users. ‘Working Day’ was dropped for casual and ‘isWeekend’ was dropped for registered users.



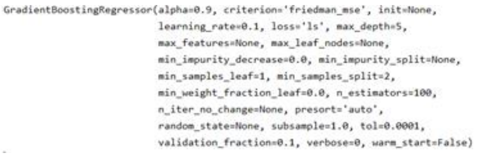
Training dataset correlation - Registered users features
Total
Registered
isWeekend
season_index
Holiday
Working Day
weather
temperature
wind_speed
humidity
year
hour
dayofweek
ur_workingday_registered

2. Grid Search CV:

Grid Search CV was used for parameter tuning in order to find the best estimator to be used for various models and the best scores.

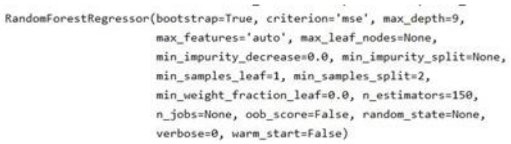
The best estimator corresponding to each model which is to be applied on the dataset are as follows:

Gradient Boost Regressor



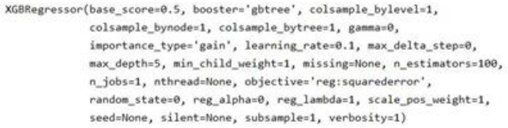
```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=0.1, loss='ls', max_depth=5,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=100,
                          n_iter_no_change=None, presort='auto',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

Random Forest Regressor



```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=9,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=150,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

XGBoost Regressor



```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=5, min_child_weight=1, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:squarederror',
             random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
             seed=None, silent=None, subsample=1, verbosity=1)
```

V. Regression Models:

The following models were applied in order to get the least possible root mean square logarithmic error.

- Gradient Boost Regressor
- Random Forest Regressor
- XGBoost Regressor.

VI. Error Estimation:

The estimate used to measure the error is ‘Root mean squared logarithmic error’. This estimate was used since RMSLE incurs a larger penalty for the underestimation of the Actual variable than the Overestimation. [6]

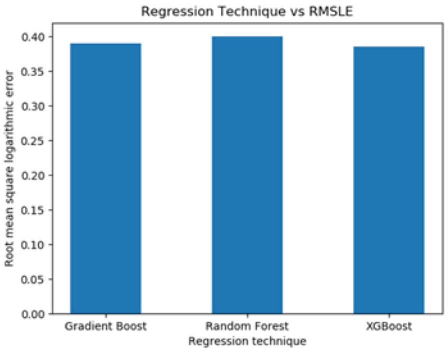
{'Gradient Boost': 0.8020686995590756, 'Random Forest': 0.7911069355108967, 'XGBoost': 0.803315714357377}

RMSLE by Gradient Boost: 0.39064946198462913

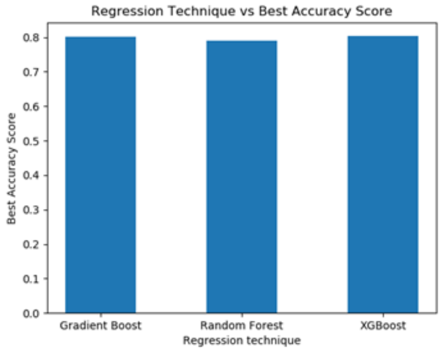
RMSLE by Random Forest: 0.39684579561720495

RMSLE by XGBoost: 0.3853878402722978

Least RMSLE is obtained from XGBoost model.



VI. Best Score for various models:

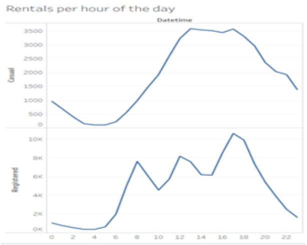


VII. Predicted Output

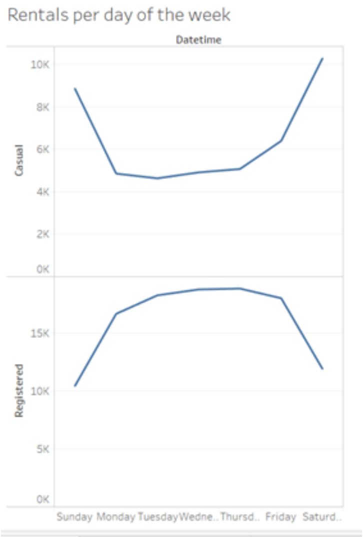
Datetime	Total
10/1/2017 0:00	14.1682
10/1/2017 1:00	9.203767
10/1/2017 2:00	5.428997
10/1/2017 3:00	3.062933
10/1/2017 4:00	2.487095
10/1/2017 5:00	2.084362
10/1/2017 6:00	3.764745
10/1/2017 7:00	7.697911
10/1/2017 8:00	15.71184
10/1/2017 9:00	26.90046
10/1/2017 10:00	40.5385
10/1/2017 11:00	46.39053
10/1/2017 12:00	51.93648
10/1/2017 13:00	57.64378
10/1/2017 14:00	56.84118
10/1/2017 15:00	57.5596
10/1/2017 16:00	65.00659
10/1/2017 17:00	65.13668
10/1/2017 18:00	61.09695
10/1/2017 19:00	45.68273
10/1/2017 20:00	31.02853
10/1/2017 21:00	20.6592
10/1/2017 22:00	14.97494
10/1/2017 23:00	8.815697

VIII. Data Visualization[Tableau]

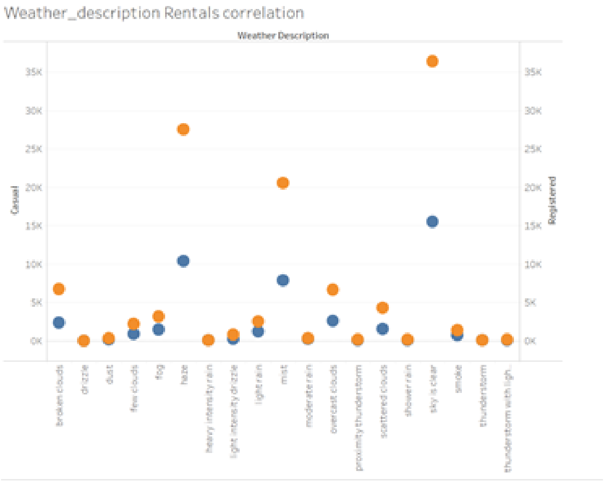
1. Rentals per hour of the day



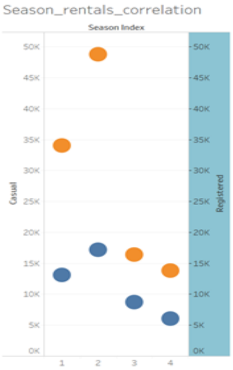
2. Rentals per day of the week



3. Weather description rentals correlation



4. Season rentals correlation



IX. References

[1] https://www.kaggle.com/selfishgene/historical-hourly-weather-data

[2] https://www.kaggle.com/cityofLA/los-angeles-metro-bike-share-trip-data

[3] https://bikeshare.metro.net/about/data/

[4]https://www.kaggle.com/c/bike-sharing-demand/notebooks

[5] https://www.kaggle.com/gsnehaa21/federal-holidays-usa-19662020

[6] https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a

[7] https://gisgeography.com/root-mean-square-error-rmse-gis/

[8] https://www.geeksforgeeks.org/exploring-correlation-in-python/