Machine Learning Engineer Nanodegree
Capstone Project

Akshaya Rane

July 27, 2018

1 Definition

1.1 Project Overview

Community bike-sharing systems are public transportation programs which enable tourists as well as local residents to rent bicycles from one location and return it to a different location as needed. Their central motivation is to provide affordable access to bicycles for short-distance trips in an urban area as a way to enhance mobility, alleviate automotive congestion, thereby reducing air pollution, noise, and boost health. The first bike-share project began in Amsterdam in 1965 and today more than 500 cities in 49 countries host advanced bike-sharing programs with over 500,000 bicycles (1).

Figure 1: Bike-sharing program in Washington DC, Source: https://www.capitalbikeshare.com/

These systems are fully automated via a network of kiosk locations throughout a city and hence they generate digital footprints that reveal mobility in a city. Therefore machine learning can be used to study this data that can help in designing and planning policy in urban transportation, to optimize the service by forecasting rental demand thereby regulating the availability.

1.1.1 Motivation

First of all, there is a growing interest in these systems today as they play important role in reducing traffic and improving environment as well as health in urban areas. Being a regular bike user myself, I am very curious to investigate this system in detail.

Many major cities are aiming to become bike-friendly day-by-day. Bike sharing programs are emerging worldwide as can be seen in Figure 2 (3). This increasing trend indicates that we would soon see this service in most major cities alongside other public transport services. So, my personal goal is to use this project as a first step to create a machine learning framework using supervised learning techniques to forecast bike rental demand in a city. The problem and solution statements are briefly stated in Section 1.2 and the corresponding evaluation metrics are described in Section 1.3. Section 2 describes the analysis including data exploration, exploratory visualization, algorithms, and benchmark models. Data preprocessing, implemention, and refinement are discussed in Section 3. The results are stated in the Section 4 and finally Section 5 states conclusions.

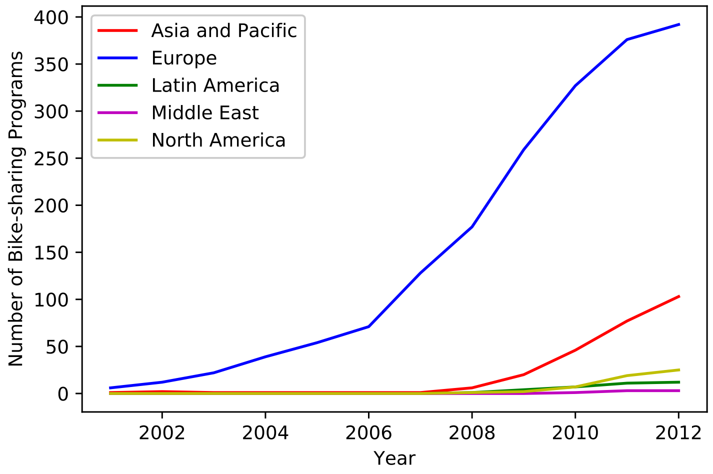

Figure 2: Number of bike-sharing programs across the world from 2000 to 2012.

1.2 Problem Statement

1.2.1 Problem description

The goal of this project is to predict bike rental demand in the Capital Bikeshare program in Washington, D.C. by combining historical usage patterns with weather data and annual holidays. In other words, we need to predict the total count of rental bikes during each hour covered by the test set, using only information available prior to the rental period. This could be useful for improving the bike usage by providing a better client experience.

1.2.2 Solution description

This is a well-defined supervised learning problem, more specifically a regression problem, with hourly records of weather, holiday information as features and total number of bike rentals as target variable. To solve this problem and to create an accurate model, the following steps are taken:

- As mentioned above, the total number of bike rentals is used as a label with supervised learning. The registered and casual bike rental numbers can also be used as labels individually if we are particularly interested in improving the usage of one of these.
- Data is prepared for the analysis.
- An appropriate splitting method is chosen to divide the data set into training, cross-validation, and testing set.
- Regression algorithms like Linear Regression, SVM, Decision trees, Random Forest, and Gradient Boosting are applied. If available, the feature selection is applied to determine the most important features and the regression analysis is carried out.
- Learning curves are created to better understand the learning algorithms.
- Parameter space is investigated for the best suited model to get the best test metric.
- Perform PCA with the optimal model in order to reduce feature dimensions while capturing as much variance as possible. The hope is that regression algorithm might work better on the PCA transformed data set than with the original features.
- Implement feature importance if applicable for the best fit model.

1.3 Metrics

The mean squared error measures the average squared difference between actual and predicted values and incorporates both the variance and bias of an algorithm (12). The root mean square logarithmic error (RMSLE) is used to evaluate the accuracy of an algorithm. RMSLE is defined as the ratio between the actual and predicted values:

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

where n is the number of hours in the test set, p_{i} is the predicted count, and a_{i} is the actual count.
RMSLE score is used to avoid penalizing huge differences between the actual and predicted values when both the values are large numbers.

2 Analysis

2.1 Data Exploration

This project will use the bike sharing data set obtained from the UCI machine learning repository (4) which combines the hourly rental data between years 2011 and 2012 in Capital bikeshare system (5) with weather information (6), and holiday schedule (7).

This data set contains a total of 17,389 instances that are hourly time stamps. Each instance is described by 16 features:

- dteday: date
- season: season (1:spring, 2:summer, 3:fall, 4:winter)
- yr : year (0: 2011, 1:2012)
- mnth : month (1 to 12)
- hr : hour (0 to 23)

- holiday : if holiday:0,, else 1
- weekday : day of the week starting Sunday (0 to 6)
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0
- weathersit : 1 if Clear, Few clouds, Partly cloudy, Partly cloudy;
  2 if Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist;
  3 if Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds;
  4 if Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are derived via $(t - t_{\text{min}})/(t_{\text{max}} - t_{\text{min}})$, $t_{\text{min}} = -8, t_{\text{max}} = +39$
- atemp: Normalized feeling temperature in Celsius. The values are derived via $(t - t_{\text{min}})/(t_{\text{max}} - t_{\text{min}})$, $t_{\text{min}} = -16, t_{\text{max}} = +50$
- hum: Normalized humidity. The values are divided by 100 (max)
- windspeed: Normalized wind speed. The values are divided by 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered (target variable)

If we consider the casual and registered users separately, their statistical description shows that rentals bikes usage is much higher with registered users as compared to casual users and tourists.

Table 1: Statistical description

Feature	Mean
Casual users	35.7
Registered users	153.8

To explore this further, we compare the usage with the hour of the day as shown in Figure 3. Here the average bike rentals every hour is plotted. While the casual usage is a bit more during early afternoons, there is a clear trend in the usage patterns of registered users implying that people rent bikes for morning/evening commutes. This information can be used to come up with separate solutions to increase the usage in each of these categories.

2.2 Exploratory Visualization

It is very important to visualize the input features to see how they affect the target variable. To explore some of the features, the plot in Figure 4 compares the 'feels-like' temperature with the total number of bike rentals. The normalized values are converted to actual temperatures for better understanding, however, in the data analysis the normalized values are used. It can be noticed that the number of bike rentals peak at a temperature of $\sim$ 25 C which is reasonable. The best way to understand the dataset, a scatter matrix is constructed for the continuous variables as shown in Figure 5. As can be seen, some of the distributions are close to normal, while some others including the labels and windspeed are not normally distributed.

To get a better sense of correlations, the heatmap is produced (See, Figure 6. The most expected correlation is between *temp* and *atemp* which is obvious. Some of the prominent negative correlations are between casual-working day which is also expected, hr-humidity which makes sense because the humidity reduces in the day time compared to late night and early mornings, humidity-count is also reasonable because

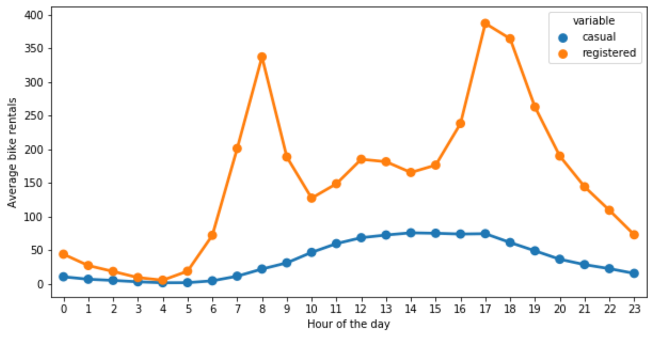

Figure 3: Average bike rentals every hour.

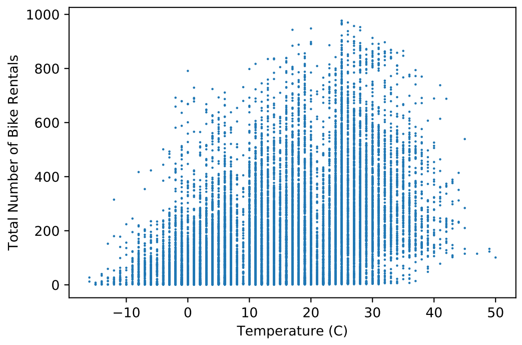

Figure 4: Total number of bike rentals as a function of 'feels-like' temperature.

people will not likely bike when it is very humid which is also apparent between weather situation and count. Weather categories from 1 to 4 indicate that it gets worse and it is less likely that people will rent bikes. Both casual and registered bikes are related to the total number of rentals as expected with registered bikes showing a very high correlation indicating that most of the bikes that are rented are registered as seen before.

2.3 Algorithms and techniques

A number of supervised learning algorithms are applied on the dataset as discussed below.

2.3.1 Linear Regression

Linear regression is one of the most well known and well understood algorithms in supervised machine learning. It is represented as a linear equation that combines a specific set of input values (x), the solution to which is the predicted output for that set of input values (y).
Different techniques can be used to train the linear regression equation from data, the most common of which is called Ordinary Least Squares. In this method, given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimize.

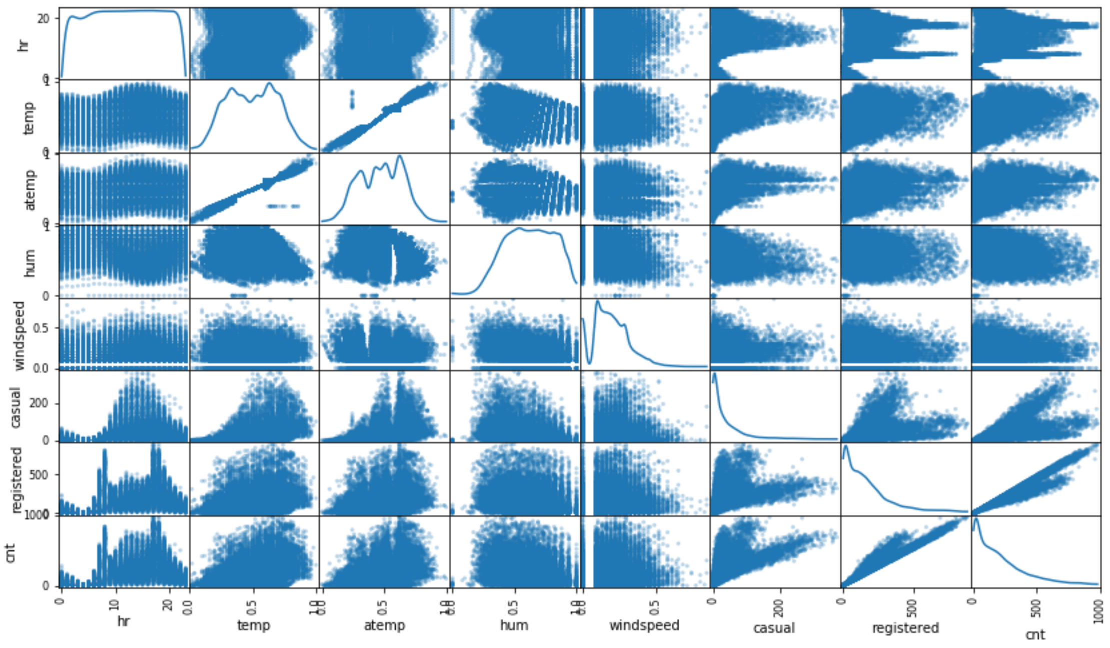

Figure 5: Scatter matrix plot for the continuous features in the dataset.

2.3.2 Random Forest Regressor

Random Forest is a flexible, easy to use supervised machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is a type of additive model that makes predictions by combining decisions from a sequence of base models such as Decision Trees. The random forest model is very good at handling tabular data with numerical features. Unlike linear models, random forests are able to capture non-linear interaction between the features and the target variable.
Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

2.3.3 Gradient Boost Regressor

Gradient boosting is a supervised machine learning algorithm and is one of the most powerful techniques for building predictive model in the form of an ensemble of weak prediction models, typically decision trees (Wikipedia). It works as a optimization problem where the objective is to minimize the loss of the model by adding weak learners using a gradient descent like procedure.

2.4 Benchmarks

A few scikit-learn regressors (linear regressor, random forest regressor) were implemented initially. The performances are listed below:

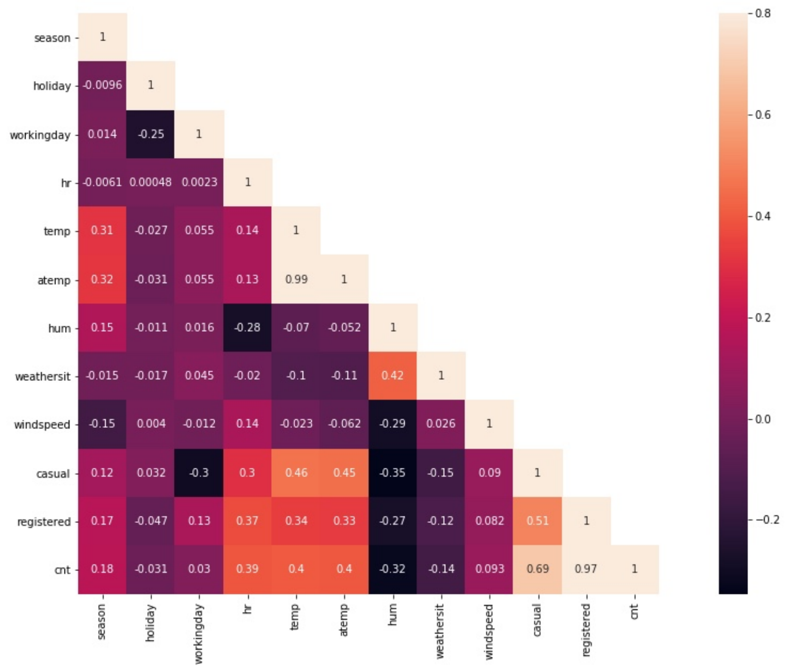

Figure 6: Correlation matrix plot for the continuous features in the dataset.

Table 2: Benchmark performances

Algorithm	RMSLE
Linear regressor	0.99
Random forest regressor	0.16

3 Methodology

3.1 Data Preprocessing

In order to implement supervised learning algorithms on the dataset, several steps are carried out listed below:

- Collect and extract the data from UCI machine learning repository
- Load full dataset (csv format)
- Remove any NaN/missing values
  - Fortunately, there are no missing values in this dataset as found by

    ```
    data.isnull().sum()
    ```
- Normalization/Scaling
  - The temperature values are normalized in the original dataset.
- Outlier detection and removal

    – All data points that are greater than the third quartile + 1.5 * inter-quartile range and smaller than the first quartile - 1.5 * inter-quartile range are considered outliers.
    – After analyzing boxplots for some of the features, the outliers are removed from the dataset.

```
data[np.abs(data["cnt"]-data["cnt"].mean())<=(3*data["cnt"].std())]
```

    – Figure 7 shows one such boxplot. As can be seen, most of the outlier points are contributed from "Working day" feature.

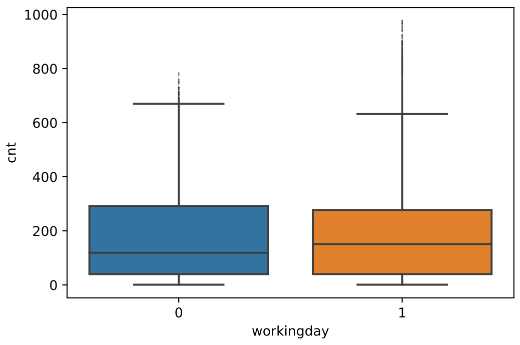

Figure 7: Boxplot showing outliers for the "working day" feature of the dataset.

- Drop unnecessary features
    – Some of the less relevant features such as Date, Year, temp are removed from the dataset.
- Split the dataset into train, cross-validation, and test set
    – Since this is a time-series dataset, the first 19 days of each month is considered as the training/cross-validation set.
    – The test set comprises the data from 20th to the end of the month.

```
data["dt"] = data.dteday.apply(lambda x : x.split()[0].split("-")[2])
data['date'] = data['dt'].astype(str).astype(int)
test_data = data[(data['date'] >= 20)]
train_data = data[(data['date'] < 19)]
```

3.2 Implementation

To start with, basic implementation for each of algorithms is carried out. Code for one example is shown below:

- Import and initiate the algorithm

```
from sklearn.linear_model import LinearRegression
LRmodel = LinearRegression()
```

- Convert the training and test labels into logarithmic values because their distribution is not normal.

```
logtrain_labels = np.log1p(y_train)
logtest_labels = np.log1p(y_test)
```

- Train the chosen model on training data (`X_train`, `logtrain_labels`).

  ```
  LRmodel.fit(X_train, logtrain_labels)
  ```

- Predictions are made on the training and test set.

  ```
  preds = LRmodel.predict(X_train)
  preds_test = LRmodel.predict(X_test)
  ```

- Compute RMSLE and print it for each set.

  ```
  print ("RMSLE Value For Linear Regression: ",rmsle(np.exp(logtrain_labels),np.exp(
  print ("RMSLE Value For Linear Regression: ",rmsle(np.exp(logtest_labels),np.exp(p
  ```

In order to test the robustness of each model, the dataset is divided into a training, validation, and testing set and learning curves are produced. A 10-fold cross validation is performed implying that the dataset is shuffled split with 10 splits. 20% of the data is chosen as the test data set.

```
from sklearn.model_selection import ShuffleSplit
cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=0)
estimator = DecisionTreeRegressor()
plot_learning_curve(estimator, title, X, y, ylim=(0.1, 1.01), cv=cv, n_jobs=4)
```

3.3 Refinement

3.3.1 Parameter estimation using grid search with cross-validation

In order to optimize the best-fit model, grid search is applied to Gradient Boosting with learning rate ranging from 0.05 to 1.0 and the maximum depth ranging from 1 to 10. The optimal values for these parameters are found to be 0.2 and 8 respectively. The results are shown in Table 3.

3.3.2 Feature Importance

The Gradient Boosting algorithm provides the feature importance implementation. Figure 8 shows the five most important features for this data set. The performance of this did not improve the model any further.

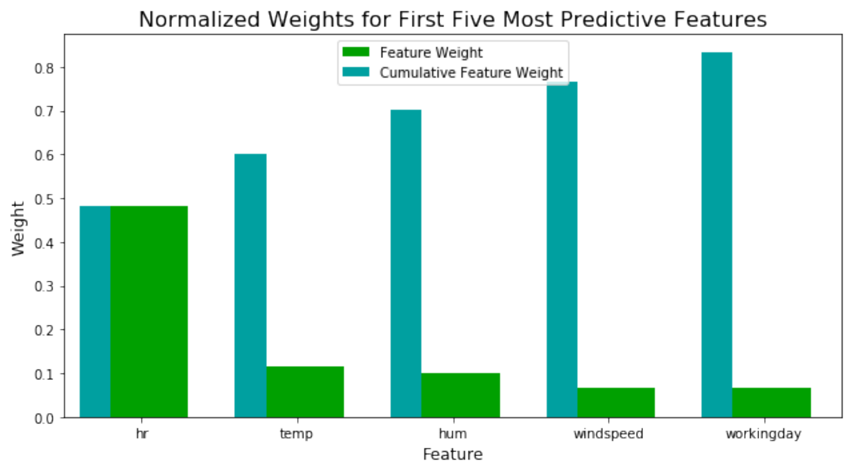

Figure 8: Normalized weights for the first five most predictive features in the dataset.

The reason might be that most of the correlated features are already removed from the data set before analysis. The results are listed in Table 4.

4 Results

4.1 Model Evaluation and Validation

The learning algorithms discussed above are implemented with basic parameters and Figure 9 shows the learning curves for four algorithms. Linear regression model performance is poor with RMSLE of 0.99 on the training data, thus lowest score. Although decision trees algorithm performes best on the training data (RMSLE = 0.009) compared to the other three algorithms, the cross-validation score is not quite high.

The cross-validation is essential to characterize the effectiveness of the algorithm on unseen data. For both, decision trees and random forest, it looks like the cross-validation score might improve with more training samples. If the score does increase with more training samples then one of these two models could be more robust than the current best fit model and therefore could be explored.

But for the current dataset, the gradient boost algorithm seem to generalize better with slightly less cross-validation score but it seems more robust than other models.

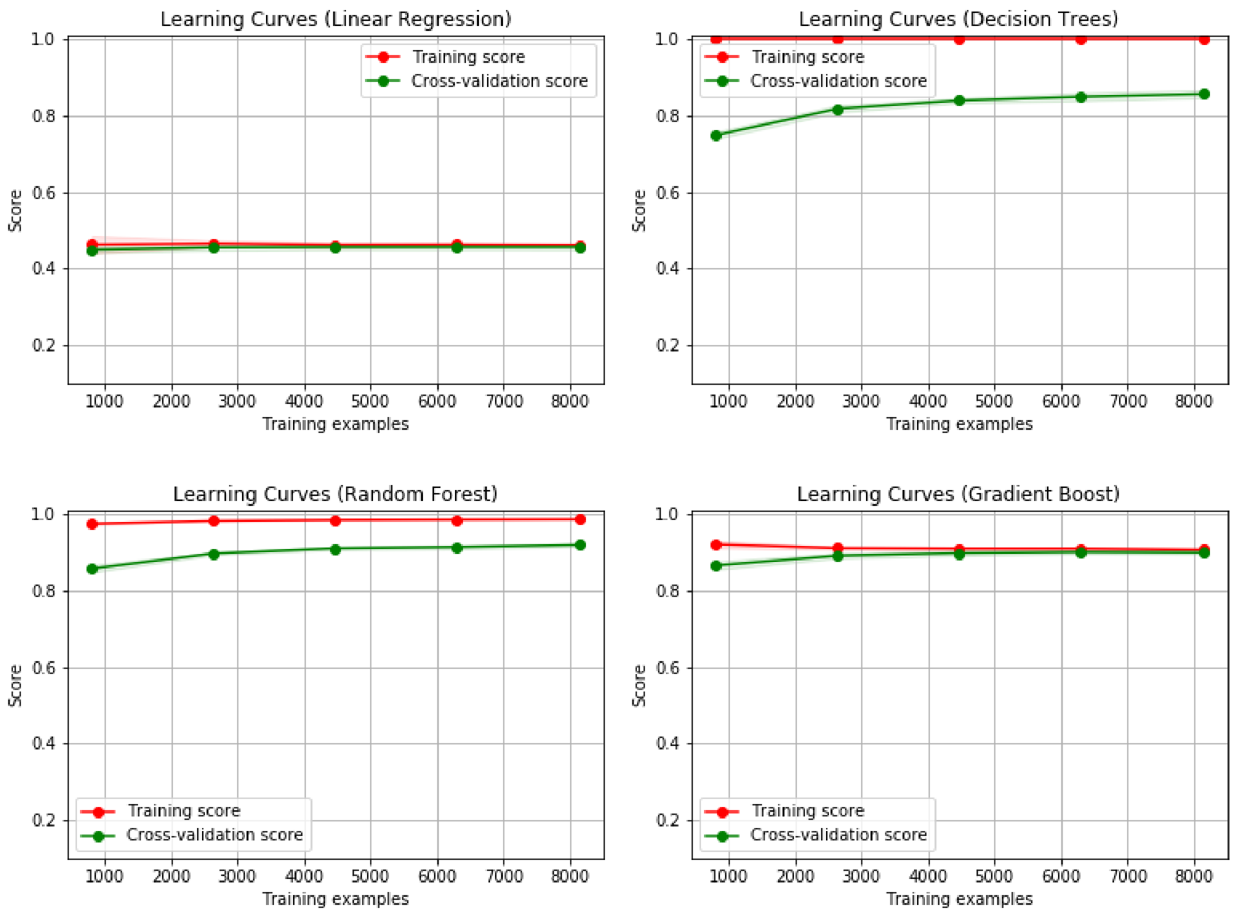

Figure 9: Learning curves with basic implementation.

As Random forest (RF) and Gradient Boosting (GB) are the best performing algorithms on this dataset, the training and prediction times are compared (Figure 10). Although the RF is a bit faster than GB in training the dataset, the prediction time for GB is less than RF.
Another advantage of GB over the other learners is that it provides feature importance functionality and therefore Gradient Boosting algorithm is chosen as the algorithm to work with for further analysis.

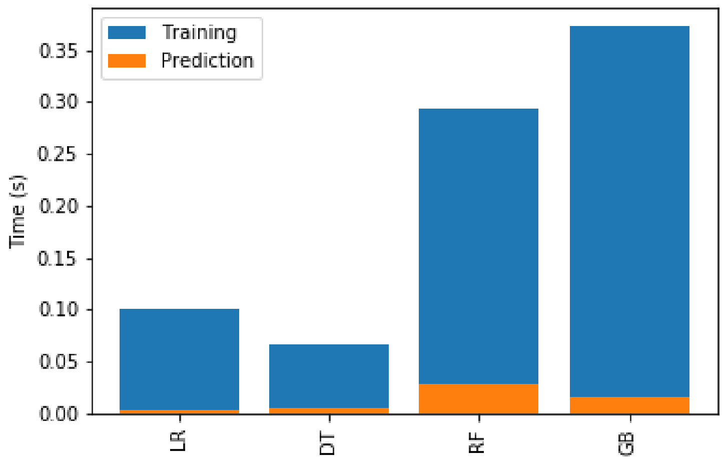

Figure 10: Training and prediction times for the four algorithms.

4.1.1 Final model performance

After tuning the parameters for the Gradient Boosting algorithm using grid search, the RMSLE is computed for the optimal parameters as:

- $learning_{\text{rate}} = 0.2$ (default: 0.1)
- $max_{\text{depth}} = 8$ (default: 3)

```
GradientBoostingRegressor(loss='ls', learning_rate=0.2, n_estimators=100,
subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_depth=8, min_impurity_decrease=0.0,
min_impurity_split=None, init=None, random_state=None, max_features=None,
alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, presort='auto')
```

Table 3: Results for the unoptimized and optimized models

	GB (un-optimized)		GB (optimized)	
	Train data	Test data	Train data	Test data
RMSLE	0.41	0.49	0.14	0.46

The RMSLE is improved for both train/validation and test sets after tuning the parameters.

4.2 Justification

The best-fit model with optimal parameters performs much better than the benchmark models as expected. The labels used in the benchmark models are not logarithmic values and hence the distribution is not normal. Also, the outliers are not removed in the benchmark models.

5 Conclusion

5.1 Free-Form Visualization

Using a 2-year dataset of rental bike usuage and weather, the Gradient Boosting algorithm is capable of making predictions on the bike rentals with a RMSLE of 0.14. If the number of features is reduced from 10 to 5 most important features, the RMSLE for both the train and test data sets does not seem to improve and it actually gets worse. Since we only have 10 features in the data, the model performs better when all the features are used as seen in Table 4.

Table 4: Results for feature importance implementation

	GB (No feature selection)		GB (With feature selection)	
	Train data	Test data	Train data	Test data
RMSLE	0.14	0.46	0.21	0.49

5.2 Reflection

The goal of this project was to train a supervised learning algorithm such that it is able to predict the bike rental demand in Washington D.C., given the historical usage data, weather conditions and annual holidays. A number of algorithms were implemented such as linear regression, decision trees, random forest, SVM, and gradient boost. Linear regression performed very poorly on the dataset. SVM was extremely slow compared to all others and decision trees did very well on the training data but poorly on the validation set. Hence these 3 were not explored further in the analysis. Random forest and Gradient boost seem to work well with the data but since Gradient Boost provides feature importance add-on, it was chosen as the best-fit model. Although Gradient Boosting seem to perform well on this 2-year dataset, additional training samples could prove beneficial for the overall improvement of the bike sharing program.
Grid search to tune the parameters for the Gradient Boost was useful to determine the optimal parameters of the model and then feature importance was tested on the final model. It did not improve the final model performance but with additional data, it could be useful.
In order to increase the bike rental demand, the program can introduce some incentive offers to registered and casual users separately. For example, the prices can be lowered for specific hours during the day in winter or in the afternoons for casual users in certain months of the year. This can be achieved if more data is available or will work even better if real-time data is available.
In order to use this model in a real-world setting, a couple of improvements should be made as described in the next section.

5.3 Improvement

Currently, the labels (number of total bike rentals) are not normally distributed. But with collection of more data over several years, this will get closer to normal distribution and the prediction results will get better. The data can be studied based on seasons and this will provide more information to the learner to identify any characteristics and trends in the data that will eventually improve prediction.
The learners should be applied for the casual and registered users separately as the needs and trends for each could be different.
As mentioned in the previous section, various incentives can be offered to users and each of these can be studied as separate experiments. And based on the results of these, future offers could be determined, thereby optimizing the bike rental usage in the city.

References

[1] Bike-Sharing Programs Hit the Streets in Over 500 Cities Worldwide Earth Policy Institute; Larsen, Janet; 25 April 2013.

[2] Public bikesharing in North America: Early operator and user understanding: http://transweb.sjsu.edu/sites/default/files/1029-public-bikesharing-understanding-early-operators-users.pdf

[3] Worldwide bike-sharing data: http://www.earth-policy.org/data_center/C23

[4] UCI-bike sharing dataset:

[5] Capital bike-sharing data: http://capitalbikeshare.com/system-data

[6] Weather data: http://www.freemeteo.com

[7] Annual holiday data: http://dchr.dc.gov/page/holiday-schedule

[8] Bike sharing demand-Forecast use of a city bikeshare system: https://www.kaggle.com/c/bike-sharing-demand

[9] Bicycle-sharing system: https://en.wikipedia.org/wiki/Bicycle-sharing_system

[10] Machine Learning Mastery: https://machinelearningmastery.com

[11] Wikipedia-Gradient Boosting: https://en.wikipedia.org/wiki/Gradient_boosting

[12] Wikipedia-Mean Squared Error: https://en.wikipedia.org/wiki/Mean_squared_error