Bike Renting Prediction

Ankit Saxena

11 April 2019

Table of Contents

Chapter 1

1. INTRODUCTION

With the rapid development of the global low-carbon movement and the increase in the number of private cars, city planning to expand the number of vehicles is increasing rapidly, traffic congestion and environmental pollution problems are becoming more and more serious, the public bicycle rental system is as the new public transport system to flourish in the world, the public bicycle can not only in the short distance travel play the flexible and efficient advantages, but also can effectively extend the scope of public transport service. Real time to master the number of bicycle rental bicycles to guide the needs of the public, is conducive to the formation and planning departments to develop rental policy.

1.1 Problem Statement and Project Goal

A bike-sharing system is a service in which users can rent/use bikes available for shared use on a short term basis for a price or free. Currently, there are over 500 bike-sharing programs around the world. Such systems usually aim to reduce congestion, noise, and air pollution by providing free/affordable access to bicycles for short-distance trips in an urban area as opposed to motorized vehicles. The number of users on any given day can vary greatly for such systems. The ability to predict the number of hourly users can allow the entities (businesses/governments) that oversee these systems to manage them in a more efficient and cost-effective manner.

Our goal is to use and optimize Machine Learning models that effectively predict the number of ride-sharing bikes that will be used in any given 1 hour time-period, using available information about that time/day.

The objective of this Case is to prediction of bike rental count on daily based on the environmental and seasonal settings.

We are provided with dataset of bike rental based on environmental conditions and seasons for two year i.e. 2011 and 2012. We need to predict what could be the counting of bike rental for a given season, month and year with environmental condition.

1.2 Data

Our model will be regression analysis, which will predict total bike rental for a condition.
Let us see few data points of our dataset.

instant	dteday	season	yr	mnth	weekday	workingday	weathersit
1	1/1/2011	1	0	1	6	0	2
2	1/2/2011	1	0	1	0	0	2
3	1/3/2011	1	0	1	1	1	1
4	1/4/2011	1	0	1	2	1	1
5	1/5/2011	1	0	1	3	1	1

temp	atemp	hum	windspeed	casual	registered	cnt
0.344167	0.363625	0.805833	0.160446	331	654	985
0.363478	0.353739	0.696087	0.248539	131	670	801
0.196364	0.189405	0.437273	0.248309	120	1229	1349
0.2	0.212122	0.590435	0.160296	108	1454	1562
0.226957	0.22927	0.436957	0.1869	82	1518	1600

All columns have name as they normally stands for. Details are below:

Attributes	Description
dteday	Date
yr	Year - (0: 2011, 1:2012)
mnth	Month (1 to 12)
weekday	Day of the week
season	Season (1:spring, 2:summer, 3:fall, 4:winter)
holiday	weather day is holiday or not
workingday	If day is neither weekend nor holiday is 1, otherwise it is 0
weathersit	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	Normalized temperature in Celsius. The values are derived via (t - t_min) / (t_max - t_min), t_min=-8, t_max=+39 (only in hourly scale)
atemp	Normalized feeling temperature in Celsius. The values are derived via (t - t_min) / (t_max - t_min), t_min=-16, t_max=+50 (only in hourly scale)

hum	Normalized humidity. The values are divided to 100 (max)
windspeed	Normalized wind speed. The values are divided to 67 (max)
casual	count of casual users
registered	count of registered users
cnt	count of total rental bikes including both casual and registered

Table 1: Organization of attributes

Sets	Description
Objects	{ 731 distinct objects} in dataset
Conditional Attributes	{instant, dteday, season, yr, mnth, holiday, weekday workingday, weathersit, temp, atemp, hum, windspeed} (13)
Decision Attributes	Casual, registered, cnt (3)

Chapter 2

2. METHODOLOGY

2.1 Data Pre-processing: (Exploratory Data Analysis)

Exploratory Data Analysis is an initial process of analysis, in which we can summarize characteristics of data such as pattern, trends, outliers, and hypothesis testing using descriptive statistics and visualization. We cannot remove noise/error data completely from our dataset but we can reduce it with the help of EDA (Exploratory Data Analysis).

2.1.1 Understanding the Data

Getting a look at our data and datatype:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
instant       731 non-null int64
dteday        731 non-null object
season        731 non-null int64
yr            731 non-null int64
mnth          731 non-null int64
holiday       731 non-null int64
weekday       731 non-null int64
workingday    731 non-null int64
weathersit    731 non-null int64
temp          731 non-null float64
atemp         731 non-null float64
hum           731 non-null float64
windspeed     731 non-null float64
casual        731 non-null int64
registered    731 non-null int64
cnt           731 non-null int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

Types of datatype present in our dataset:

We have datatype as object for dteday and rest others have int and float.

i) **Time based Variable: ‘**dteday’, ‘yr’, ‘mnth’, ‘season’, ‘weekday’ are time based variables in our dataset.

‘dteday’ :has date of everyday, so all unique values hence it could not be categorical.

‘yr’: we have two value 0 for 2011 and 1 for 2012, so we can consider it as categorical.

‘mnth’: We will make bins for this variable in feature engineering section as this variable constitutes 31 unique values.

‘season’: has four unique values, so we would consider it as category.

‘weekday’: It’s same like mnth, so we will make bins for this in feature engineering section as this variable constitutes 7 unique values.

ii) **Categorical Variable:**

‘Holiday’ having value 0 for non-holiday & 1 for holiday. So it would be our categorical variable.

Similarly, ‘Working day’ having value 0 for non-working day & 1 for working day. So it would also be our categorical variable.

‘weathersit’ containing 4 unique values, defining four different condition of weather.

iii) **Continuous Variable:**

‘temp’, ‘atemp’, ‘hum’, ‘windspeed’ are continuous values, and in our dataset they are in normalized format.

iv) **Target/Dependent Variable:**

‘casual’, ‘registered’ and ‘cnt’ are our target variables and in continuous form. So, our problem would be regression problem.

Analysis of dataset:

i) Analysis of numerical data:

	temp	atemp	hum	windspeed	casual	registered	cnt
count	731.0	731.0	731.0	731.0	731.0	731.0	731.0
mean	0.4953	0.4743	0.6278	0.1904	848.1764	3656.1723	4504.3488
std	0.1830	0.1629	0.14242	0.0774	686.6224	1560.2563	1937.2114
min	0.0591	0.0790	0.0	0.0223	2.0	20.0	22.0
25%	0.3370	0.3378	0.520	0.1349	315.50	2497.0	3152.0
50%	0.4983	0.4867	0.6266	0.1809	713.0	3662.0	4548.0
75%	0.6554	0.6086	0.7302	0.2332	1096.0	4776.50	5956.0
max	0.8616	0.8408	0.9725	0.5074	3410.0	6946.0	8714.0

Numerical data contains four independent continuous variables (temp, atemp, hum, windspeed) and three dependent continuous variables (casual, registered, cnt).

As we can observe from above table, all independent variable are between 0 and 1. Dataset contain normalized values for all four columns.

'cnt' i.e. target variable have minimum 22 counting and maximum 8714 counting.

ii) Analysis of categorical data:

In categorical variables, first we will find number of unique values in each category:

```
season          4
yr              2
mnth           12
holiday         2
weekday         7
workingday      2
weathersit      3
dtype: int64
```

Now, we will count each unique value in each categorical variable.

```
count of values in each categorical variable

season
3    188
2    184
1    181
4    178
Name: season, dtype: int64

yr
1    366
0    365
Name: yr, dtype: int64

mnth
12    62
10    62
8     62
7     62
5     62
3     62
1     62
11    60
9     60
6     60
4     60
2     57
Name: mnth, dtype: int64

holiday
0    710
1     21
Name: holiday, dtype: int64

weekday
6    105
1    105
0    105
5    104
4    104
3    104
2    104
Name: weekday, dtype: int64

workingday
1    500
0    231
Name: workingday, dtype: int64

weathersit
1    463
2    247
3     21
Name: weathersit, dtype: int6
```

In this dataset, we almost have similar counting of unique value for ‘season’, ‘yr’, ‘mnth’ and ‘weekday’. As these data are related to time and we have two year’s data, that is why we have almost similar counting.

Variable holiday constitutes of 21 holidays and 710 non-holiday.

Variable working day constitutes of 500 working day and 231 non-working day and also non-working day constitutes of holiday part.

Variable weathersit has only three unique values with counting of value 3 (Light raining) is 21, counting of value 2(cloudy weather) is 247 & counting of value 1(clear weather) is 463.

2.1.2 Missing Value Analysis

In this dataset, we don’t have any missing values.

In case we have missing values then we should impute it using different methods such as mean, median, KNN, linear regression etc.

Missing values for each column in our dataset:

```
instant       0
dteday        0
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

2.1.3 Outlier Analysis

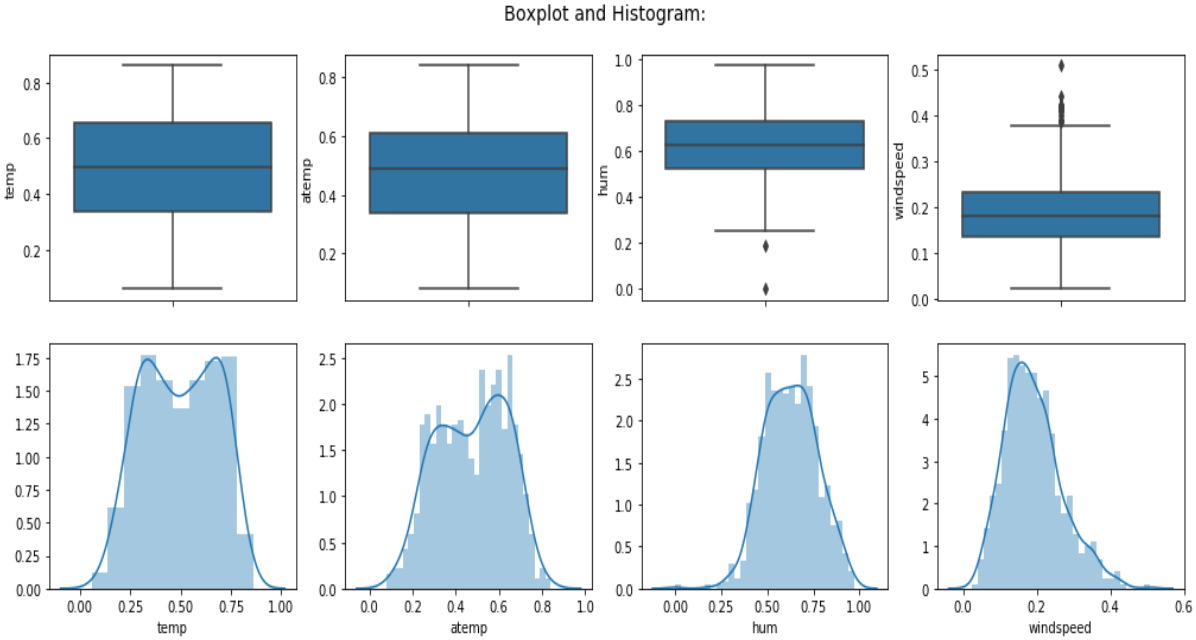
Outlier detection is important especially when our dataset is small. The box plot method detects outlier if any value is present greater than (Q3 + (1.5 * IQR)) or less than (Q1 – (1.5 * IQR)) where IQR represents Inter-Quartile Range.

Q1 > 25% of data are less than or equal to this value
Q2 or Median -> 50% of data are less than or equal to this value
Q3 > 75% of data are less than or equal to this value
IQR(Inter Quartile Range) = Q3 – Q1

So, Boxplot method find approx. 1 % of data as outliers.

Let us check for outliers in our numerical dataset and will also look at distribution of data.
So we have numerical columns as temp, atemp, hum and windspeed.
Abnormal values may directly affect count of bike renting, so removing outlier would not be a good idea for this dataset as per domain knowledge.

Let us see boxplot and histogram for our numerical data.



So, we have no outliers for ‘temp’ and ‘atemp’ and very less outlier for ‘humidity’ (hum) and few outliers for ‘windspeed’. Distribution is almost normal with little skewness for hum and windspeed and near to normal for ‘temp’ and ‘atemp’ with little bimodal effect.

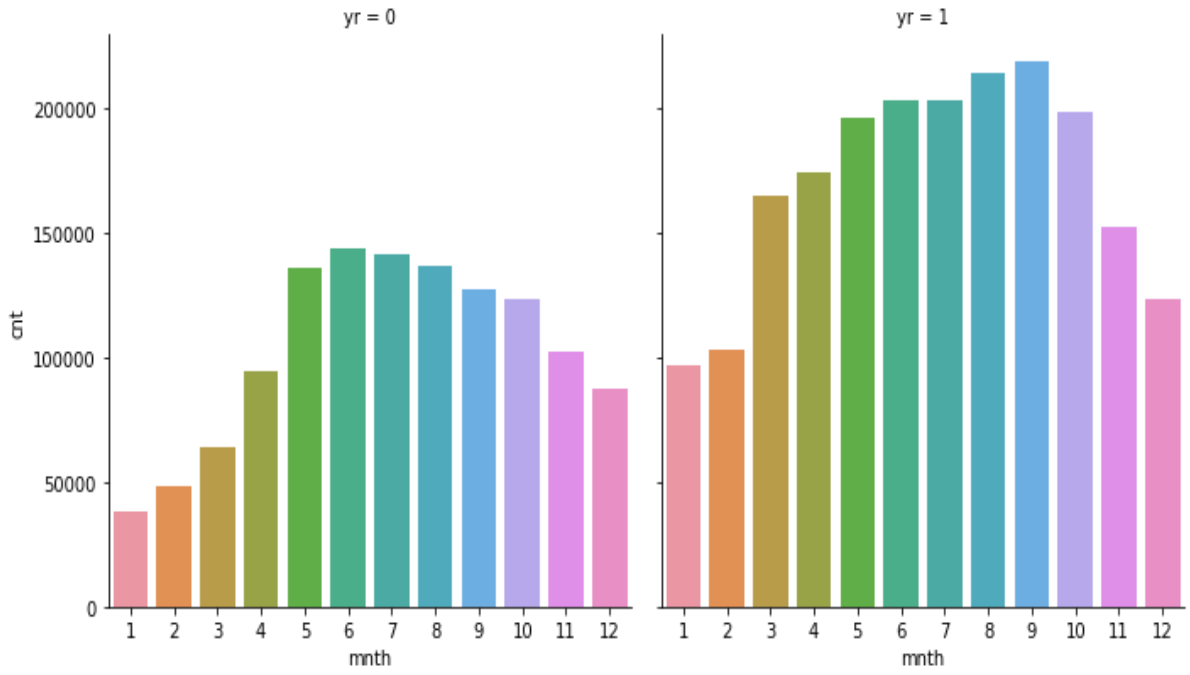
So, we will make our model with whole data points.

2.1.4 Feature Engineering

We decided to make binning for month and weekday column as per the distribution of data in respective columns.

For mnth: In month column we have range from 1 to 31. There are two ways in which we can look this, if we choose this column as numerical then it would be like 31 is greater than 30 and 25 is greater than 24, which is not the case. Also choosing every month as an category, then we may have curse of dimensionality and our model will underperform. So, to counteract this problem, we will make a new column for 'mnth' as per current values.

Let us see Distribution of counting of bike renting w.r.t each year i.e for 2011 and 2012 separately according to each month

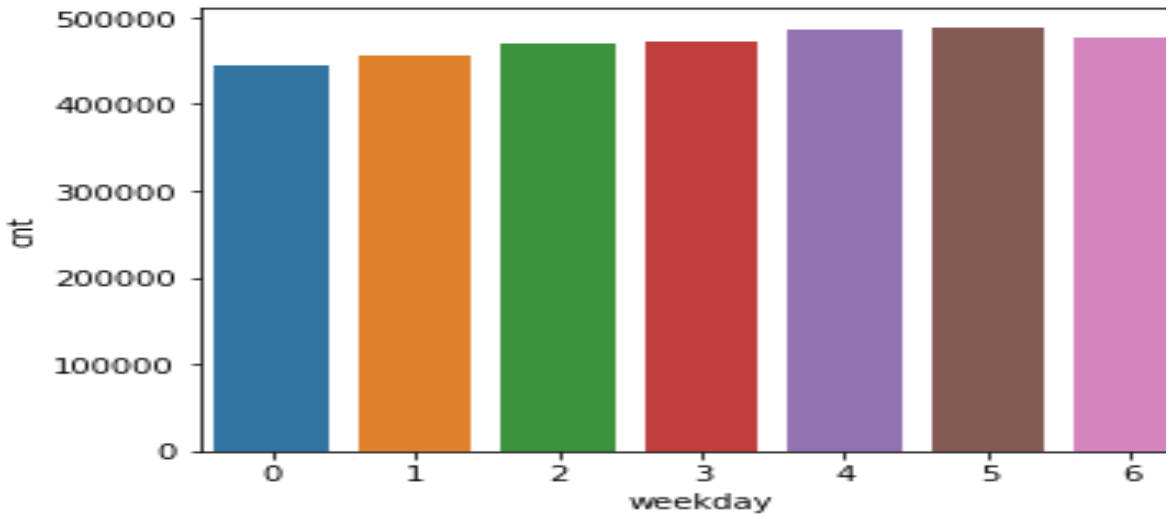


We can see a trend in counting of bike renting for month column. 5th to 10th month in both the year has high renting and other months have less bike renting in comparison.

So, we will categories month in two categories i.e. '1' for month 5th to 10th and '0' for rest. As 5th to 10th having high values for both 2011 and 2012 and rest have less values as indicated above.

For weekday: In weekday we have values ranging from 0 to 6.

Checking bar plot of counting for bike-rent for each day of week:



Here, we can clearly observe that there is a trend for weekday. We will make two bins for weekday. Value ‘0’ for weekday and value ‘1’ for rest of the days.

2.1.5 Feature Selection

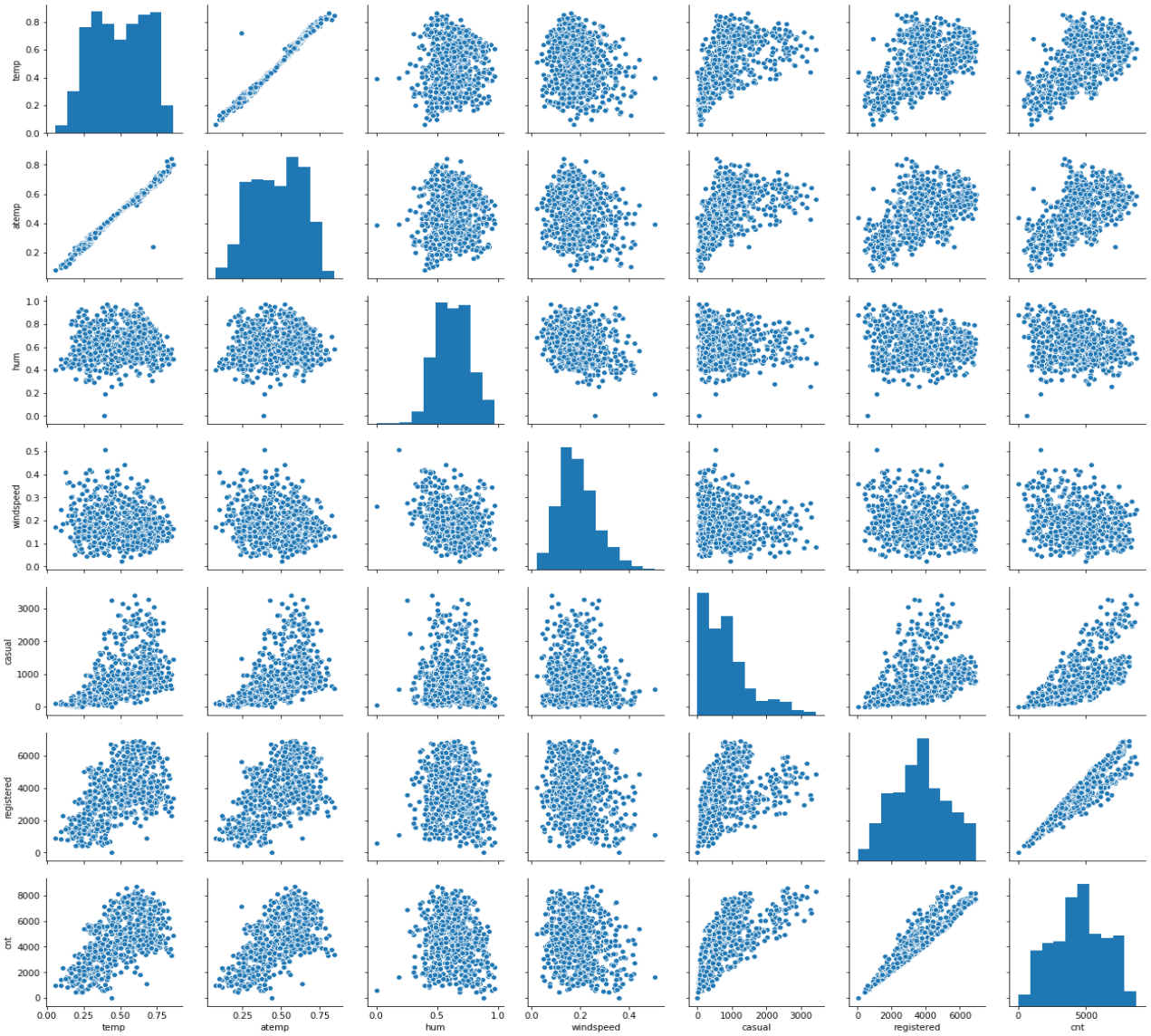
Feature selection is an important step in knowledge discovery process, to identify those relevant variables or attributes from the large number of attributes in a dataset which are too relevant and reduce the computational cost. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction.

For Numerical Variable-

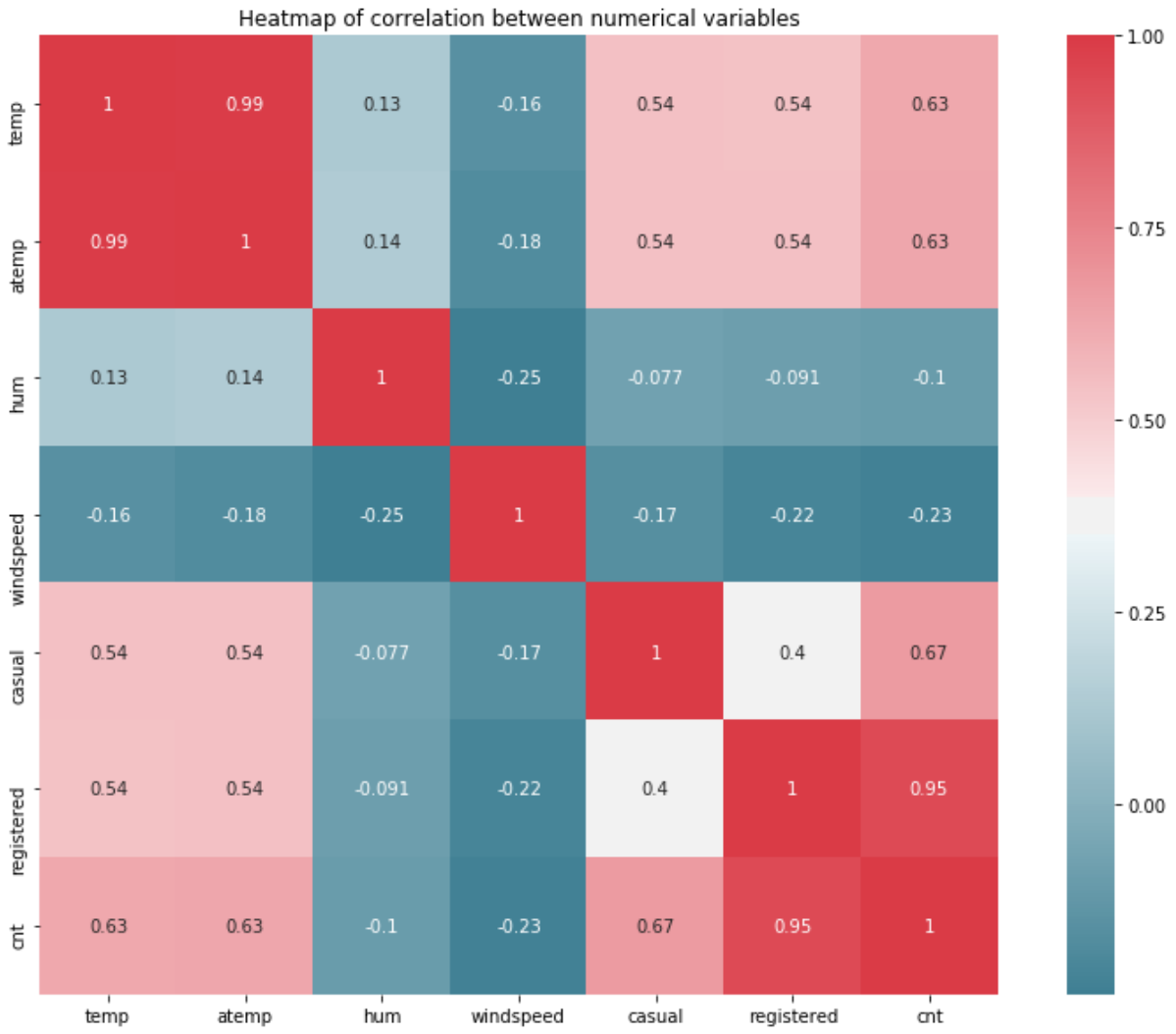
For feature selection of numerical variable, we will see if there is multicollinearity between our continuous independent variables.

For this we will plot scatter plot & heat map between our continuous independent variables.

Scatter plot between continuous variables-



Heatmap of correlation between continuous variables-



From heatmap and correlation plot we can conclude that, there is multicollinearity present between variables ‘temp’ and ‘atemp’, so we will drop one of those variables as per importance of features(whose importance in model prediction would be less). Also, there is multicollinearity present between registered and cnt, but these are our dependent variable. We will only use total counting i.e. cnt as our dependent variable and would drop casual and registered variable as they redundant for our model prediction.

For Categorical Variable-

We would not use instant and dteday column for prediction of our model. As instant is index only and information of dteday i.e. yr, month and day we have already columns for that.

For getting dependency between categorical variables, we would use chi-square test of independence test. Basically. we have mnth , yr, weekday, holiday, workingday, weathersit and season as categorical variables. We will use week_bins (i.e. after making bins of week) and month_bins (i.e. after making bins of month).

Null Hypothesis for Chi-square test of Independence:

Two variables are independent.

Alternate Hypothesis:

Two variables are not independent.

So, we want that our categorical variable should be independent. If we get p-value less than 0.05, it means we need to reject null hypothesis and accept alternate hypothesis saying that two variables are dependent. So, we would check for every combination and would get p-value.

	season	yr	month_feat	holiday	week_feat	workingday	weathersit
season	-	1.0	0.0	0.683	0.985	0.887	0.021
yr	1.0	-	0.971	0.995	0.954	0.98	0.127
month_feat	0.0	0.971	-	0.359	0.972	0.657	0.362
holiday	0.683	0.995	0.359	-	0.0	0.0	0.601
week_feat	0.985	0.954	0.972	0.0	-	0.0	0.227
workingday	0.887	0.98	0.657	0.0	0.0	-	0.254
weathersit	0.021	0.127	0.362	0.601	0.227	0.254	-

Here, from the table we can see some values are less than 0.05 indicating that they are dependent. Holiday is showing collinearity with week and workingday. Month is showing collinearity with season. Weathersit showing collinearity with season.

Now, we need to drop certain variables to remove multicollinearity. But we have to be sure that we are not losing any information.

So, for that let us now check important of features in our dataset:

	Feature	Importance
0	yr	0.315006
1	temp	0.159318
2	atemp	0.148559
3	month_feat	0.124084
4	season	0.105923
5	weathersit	0.059089
6	hum	0.034752
7	windspeed	0.022865
8	workingday	0.013670
9	week_feat	0.010718
10	holiday	0.006015

So, we will drop only holiday variable as we have working day variable which has information of holiday variable. Also, from the above table of feature importance it is clear that holiday variable has least importance while predicting our model.

Now, Checking VIF(Variance Inflation Factor) for our numerical variables:

As VIF value of 1 indicates that predictor is not correlated with other variables.

```
const        45.6
temp         63.0
atemp        63.6
hum           1.1
windspeed     1.1
dtype: float64
```

Now after checking VIF, we would remove atemp variable as it is multicollinear with temp variable and having less importance than temp variable which we can see from variable importance table.

Now, we have good value of VIF and our dataset don’t have multicollinearity.

2.1.6 Feature Scaling

Feature Scaling is a method used to normalize or standardize the range of independent variables or features of data. Basic function of feature scaling is to bring all the variables in same range. Feature scaling is important for those variables for which values are too high and too low.

Since in our dataset, All variable are already in same range. So we don't require feature scaling for our dataset.

2.1.7 Data after Exploratory Data Analysis

So, we would remove 'instant', 'holiday', 'dteday', 'atemp', 'casual', 'registered' variables from our dataset and would make bins for 'mnth' and 'weekday' as described earlier in Feature Engineering step of Data pre-processing. Now, we have 731 observations of 10 variables.

	season	yr	workingday	weathersit	temp	hum	windspeed	cnt	month_feat	week_feat
0	1	0	0	2	0.344167	0.805833	0.160446	985	0	1
1	1	0	0	2	0.363478	0.696087	0.248539	801	0	0
2	1	0	1	1	0.196364	0.437273	0.248309	1349	0	0
3	1	0	1	1	0.200000	0.590435	0.160296	1562	0	1
4	1	0	1	1	0.226957	0.436957	0.186900	1600	0	1

Chapter 3

3. MACHINE LEARNING - BIKE RENT PREDICTION

Total of 5 different classification models were used to predict the bike rent.

These models were:

- Random Forest
- Decision Tree
- Linear Regression
- KNN
- Support Vector Machine

3.1 Random Forest

Random forests (RF) are an ensemble learning technique that can support classification and regression. It extends the basic idea of single classification tree by growing many classification trees in the training phase. To classify an instance, each tree in the forest generates its response (vote for a class), the model choses the class that has receive the most votes over all the trees in the forest. One major advantage of RF over traditional decision trees is the protection against overfitting which makes the model able to deliver a high performance.

3.2 Decision Tree

Decision Tree (DT) is a model that generates a tree-like structure that represents set of decisions. DT returns the probability scores of class membership. DT is composed of: a) internal Nodes: each node refers to a single variable/feature and represents a test point at feature level; b) branches, which represent the outcome of the test and are represented by lines that finally lead to c) leaf Nodes which represent the class labels. That is how decision rules are established and used to classify new instances. DT is a flexible model that supports both categorical and continuous data.

3.3 Linear Regression

Linear Regression establishes a relationship between dependent variable (Y) and one or more independent variables (X) using a best fit straight line (also known as regression line). In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.

- There must be linear relationship between independent and dependent variables
- Multiple regression suffers from multicollinearity, autocorrelation, heteroskedasticity.
- Linear Regression is very sensitive to Outliers. It can terribly affect the regression line and eventually the forecasted values.

3.4 Support Vector Machines

Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

3.5 Instance – based learning

Also known as memory based learning, new instances are labelled based on previous instances stored in memory. The most widely used instance based learning techniques for classification is K-nearest neighbor (KNN). KNN does not try to construct an internal model and computations are not performed until the classification time. KNN only stores instances of the training data in the features space and the class of an instance is determined based on the majority votes from its neighbors. Instance is labelled with the class most common among its neighbors. KNN determine neighbors based on distance using Euclidian, Manhattan or Murkowski distance measures for continuous variables and hamming for categorical variables. Calculated distances are used to identify a set of training instances (k) that are the closest to the new point, and assign label from these. Despite its simplicity, KNN have been applied to various types of applications.

3.6 Hyperparameter tuning for Model development

Hyperparameters are hugely important in getting good performance with models. In order to understand this process, we first need to understand the difference between a model parameter and a model hyperparameter.

Model parameters are internal to the model whose values can be estimated from the data and we are often trying to estimate them as best as possible whereas hyperparameters are external to our model and cannot be directly learned from the regular training process. These parameters express "higher-level" properties of the model such as its complexity or how fast it should learn.

In this model development, I chose to demonstrate using grid search.

3.6.1 Grid Search

Grid Search takes a dictionary of all of the different hyperparameters that we want to test, and then feeds all of the different combinations through the algorithm for us and then reports back to us which one had the highest accuracy.

Chapter 4

4. EVALUATION MEASURES FOR REGRESSION

4.1 Mean Absolute Error (MAE)

MAE is the average of the absolute difference between the predicted values and observed value. The MAE is a linear score which means that all the individual differences are weighted equally in the average. For example, the difference between 10 and 0 will be twice the difference between 5 and 0. Mathematically, it is calculated using this formula:

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

4.2 Root Mean Square Error (RMSE)

RMSE represents the sample standard deviation of the differences between predicted values and observed values (called residuals). Mathematically, it is calculated using this formula:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

4.3 Mean Absolute Percentage Error (MAPE)

The mean absolute percentage error (MAPE) is the percentage equivalent of MAE. The equation looks just like that of MAE, but with adjustments to convert everything into percentages. MAPE is how far the model's predictions are off from their corresponding outputs on average.

Mathematically, it is calculated using this formula:

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|,$$

where A_t is the actual value and F_t is the forecast value.

4.4 R-Squared

R-squared is a statistical measure of how close the data are to the fitted regression line. It is the percentage of the response variable variation that is explained by a linear model. R-squared increases with increasing terms even though the model is not actually improving.

R-squared = Explained variation / Total variation

R-squared is always between 0 and 100%

- 0% indicates that the model explains none of the variability of the response data around its mean.
- 100% indicates that the model explains all the variability of the response data around its mean.

Mathematically, R Squared is given by:

$$\hat{R}^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

The numerator is MSE (average of the squares of the residuals) and the denominator is the variance in Y values. Higher the MSE, smaller the R_squared and poorer is the model.

4.5 Adjusted R-Squared

Adjusted R-Squared gives exact and accurate information about the variance of independent variable. Adjustment of R-squared that penalizes the addition of extraneous predictors to the model, i.e it will penalizes the effect of extra variables.

Just like R^2, adjusted R^2 also shows how well terms fit a curve or line but adjusts for the number of terms in a model. It is given by below formula:

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(n-1)}{n-k-1}\right]$$

where n is the total number of observations and k is the number of predictors.
Adjusted R^2 will always be less than or equal to R^2.
There is one main difference between R Square and the adjusted R square: R square assumes that every single variable explains the variation in the dependent variable. ... The R-Squared increases only if the new term improves the model more than would be expected by chance.

4.6 Validation of the Model using K-fold Cross Validation

K-fold cross validation is used to check or review performance of model which is checked on K different test dataset. So in this, K-fold cross validation helps, it would divide our training data in k sets and will build a model using k-1 training set and one left set would be used to test our model performance. In this way it would build k times model and each time there would be different test dataset to check performance and at the end all k model's accuracy(or any other metric) mean value would be considered as model accuracy(any mertric). So, we would use K-fold cross validation technique to get performance & validation of our model.

Chapter 5

5. MODEL DEVELOPMENT USING PYTHON

We will build one by one regression models and will check performance of our model and then will decide final model which we should use for our project.

5.1 Decision Tree

Using DecisionTreeClassifier algorithm we designed our bike rent prediction model with ‘cnt’ as target variable.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
0.737662334988456

on train data R-Squared
1.0

on test data R-Squared
0.8235176614838063
```

We can observe from above results, Decision tree regressor is explaining K-fold variance 73.76% for whole dataset.

Another thing we can observe is that Decision tree explaining variance 100% for training dataset. So, overfitting is present here. So, we will use next Random forest which would remove overfitting of decision Tree.

5.2 Random Forest

Using RandomForestClassifier algorithm, we designed our bike rent prediction model with ‘cnt’ as target variable.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
    0.860380694575986

on train data R-Squared
   0.9719772555444962

on test data R-Squared
   0.8887112469492335
```

We can observe that in Random Forest model, R-Squared for k-fold is 86.03% for whole dataset. We also have less R-Squared for training dataset than decision tree and high score for test dataset. **So, Random forest helped in removing overfitting.**

5.3 Linear Regression

Using LinearRegression algorithm, we designed our bike rent prediction model with ‘cnt’ as target variable.

Performance of Linear Regression built on X_train, y_train and tested on X_test.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
    0.7934309409817804

on train data R-Squared
   0.7977073584880527

on test data R-Squared
   0.7936861362478682
```

We can see from above results K-fold explained variance for whole dataset is 79.34%.

We also have R-Squared for training dataset is 79.77% & for test dataset is 79.36% which is least in comparing to other models.

5.4 KNN

Using KNeighborsRegressor algorithm, we designed our bike rent prediction model with 'cnt' as target variable.

Performance of KNN built on X_train, y_train and tested on X_test.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
    0.8119688131112855

on train data R-Squared
   0.8777866118450446

on test data R-Squared
   0.8156584724137148
```

We can see from above results K-fold explained variance for whole dataset is 81.11%.

We also have R-Squared for training dataset is 87.77% & for test dataset is 81.56% which is less than Random forest model.

5.5 Support-vector machines

Using SVR algorithm, we designed our bike rent prediction model with 'cnt' as target variable.

Performance of support vector machine built on X_train, y_train and tested on X_test.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
   0.022557971435676914

on train data R-Squared
   0.02480882896991921

on test data R-Squared
   0.02515226238064705
```

Support vector regressor is not giving good result for our dataset due to the fact that SVM is mostly used in classification problems.

5.6 Hyperparameter tuning for Model development

Now, we will tune our best model i.e. Random Forest using hyperparameter tuning. With the help of hyperparameter tuning we would find optimum values for parameter used in function that would increase the accuracy of model.

Performance of hyperparameter tuning in Random forest model built on X_train, y_train and tested on X_test.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
    0.8668317334704365

on train data R-Squared
    0.965607969664463

on test data R-Squared
    0.8898201610699171
```

Thus, with the help of hyperparameter tuning, we have increased our model R-Squared for k-fold from 86.03% to 86.68%. Also we can observe that previously it was giving accuracy for training dataset as 97.19% and now it is giving 96.5% and on test dataset we have increased accuracy from 88.87% to 88.98%.

So, with the help of hyperparameter tuning we reduced overfitting in our model.

Chapter 6

6. MODEL DEVELOPMENT USING R

We will build one by one regression models in R and will check performance of our model and then will decide final model which we should use for our project.

6.1 Decision Tree

Using ‘rpart2’ library present in R environment we designed our bike rent prediction model with ‘cnt’ as target variable.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
          0.7467

on train data R-Squared
          0.7872

on test data R-Squared
          0.7895
```

We can observe from above results, Decision tree model is explaining K-fold variance 74.67% for whole dataset.

Another thing we can observe is that Decision tree explaining variance 78.72% for training dataset & 78.95% for test dataset.

6.2 Random Forest

Using ‘rf’ library present in R environment, we designed our bike rent prediction model with ‘cnt’ as target variable.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
          0.8705

on train data R-Squared
          0.9747

on test data R-Squared
          0.894
```

We can observe that in Random Forest model, R-Squared for k-fold is 86.03% for whole dataset.

6.3 Linear Regression

Using ‘lm’ library present in R environment, we designed our bike rent prediction model with ‘cnt’ as target variable.

Performance of Linear Regression built on X_train, y_train and tested on X_test.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
          0.8191

on train data R-Squared
          0.8213

on test data R-Squared
          0.8574
```

We can see from above results K-fold explained variance for whole dataset is 81.91%.

We also have R-Squared for training dataset is 82.13% & for test dataset is 85.74%.

6.4 KNN

Using ‘knn’ library present in R environment, we designed our bike rent prediction model with ‘cnt’ as target variable.

Performance of KNN built on X_train, y_train and tested on X_test.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
          0.7977

on train data R-Squared
          0.8353

on test data R-Squared
          0.8331
```

We can see from above results K-fold explained variance for whole dataset is 79.77%.

We also have R-Squared for training dataset is 83.53% & for test dataset is 83.31%.

6.5 Support-vector machines

Using ‘svmLinear3’ library present in R environment, we designed our bike rent prediction model with ‘cnt’ as target variable.

Performance of support vector machine built on X_train, y_train and tested on X_test.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
          0.8099

on train data R-Squared
          0.8208

on test data Adjusted R-Sqaured
          0.8584
```

We can see from above results K-fold explained variance for whole dataset is 80.99%.

We also have R-Squared for training dataset is 82.08% & for test dataset is 85.84%.

6.6 Hyperparameter tuning for Model development

Now, we will tune our best model i.e. Random Forest using hyperparameter tuning. With the help of hyperparameter tuning we would find optimum values for parameter used in function that would increase the accuracy of model.

Performance of hyperparameter tuning in Random forest model built on X_train, y_train and tested on X_test.

Showing K-fold cross validation, R-Squared score for train and test dataset.

```
K-fold (K = 10) R-Squared
          0.8789

on train data R-Squared
          0.9736

on test data R-Squared
          0.8956
```

Thus, with the help of hyperparameter tuning, we have increased our model R-Squared for k-fold from 87.05% to 87.89%. Also we can observe that previously it was giving accuracy for training dataset as 97.47% and now it is giving 97.36% and on test dataset we have increased accuracy from 89.44% to 89.56%.

So, with the help of hyperparameter tuning we have reduced overfitting in our model.

Chapter 7

7. CONCLUSION

Our goal was to use and optimize Machine Learning models that effectively predicts the number of ride-sharing bikes that will be used in any given 1 hour time-period, using available information about that time/day. So, I have tried to encounter this problem statement using various machine learning algorithm through R & Python language.

Based on the findings of this model development & According to interpretation of evaluation measures, I recommended Random forest model after doing hyperparameter tuning would be most suitable for prediction of bike rent in both R & Python language gave the best result as K-fold cross validation is 87.89% & R-Squared score for train and test dataset is 97.36% & 89.56% respectively.

And from the buisness perspective view of the project, As I have already discussed in the feature engineering section 5th to 10th month in both the years has high bike renting and other months have less bike renting in comparison so, we should try to have more bikes running in the road in that period as in this 6 months period bikes are in demand. And also, In feature engineering section we can clearly observe that there is a high trend for bike rent during weekends so we should try to have more bikes during weekends as well than usual weekdays.

8. APPENDIX A-

8.1 COMPLETE PYTHON CODE

```
# import libraries
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
os.chdir("C:/Users/Ankit Saxena/Desktop/Edwisor/Project")

#read csv file
bike_data = pd.read_csv("bike_rentdata.csv")
```

```
#Getting few observation of dataset
bike_data.head()
```

```
# Getting information of dataset
bike_data.info()
```

```
#Defining Columns as categorical and numerical
numerical_columns = ['temp','atemp','hum','windspeed','casual','registe
red','cnt']
catego_columns = ['season','yr','mnth','holiday','weekday','workingday'
,'weathersit']
```

```
#Summary of numerical variables
bike_data[numerical_columns].describe()
```

```
#Count of unique values of categories variables
bike_data[catego_columns].nunique()
```

```
#Count of each unique value in each categorical variable
print("count of values in each categorical variable")
print()
for i in catego_columns:
    print(i)
    print(bike_data[i].value_counts())
```

```
# check for missing values in dataset
bike_data.isnull().sum()
```

```
# Function that will plot boxplot and histogram for four variables
def hist_and_box_plot(col1, col2, col3, col4, data, bin1=30, bin2=30, b
in3=30, bin4 = 30, sup =""):
    fig, ax = plt.subplots(nrows = 2, ncols = 4, figsize= (14,6))
```

```
    super_title = fig.suptitle("Boxplot and Histogram: "+sup,fontsize='
x-large')
    plt.tight_layout()
    sns.boxplot(y = col1, data = data, ax = ax[0][0])
    sns.boxplot(y = col2,data = data, ax = ax[0][1])
    sns.boxplot(y = col3, data = data, ax = ax[0][2])
    sns.boxplot(y = col4, data = data, ax = ax[0][3])
    sns.distplot(data[col1], ax = ax[1][0], bins = bin1)
    sns.distplot(data[col2], ax = ax[1][1], bins = bin2)
    sns.distplot(data[col3], ax = ax[1][2], bins = bin3)
    sns.distplot(data[col4], ax = ax[1][3], bins = bin4)
    fig.subplots_adjust(top = 0.90)
    plt.show()
```

```
# Plot of boxplot and histogram for numerical variables in dataset
hist_and_box_plot('temp', 'atemp', 'hum', 'windspeed', bin1 = 10, data
= bike_data)
```

```
#user defined function to plot bar plot of a column for each y i.e. y1
and y2 wrt unique variables of each x i.e. x1 and x2
def plot_bar(x1, y1,x2, y2, method = 'sum'):
    fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize= (12,4), squee
ze=False)
    super_title = fig.suptitle("Bar Plot ",  fontsize='x-large')
    if(method == 'mean'):
        gp = bike_data.groupby(by = x1).mean()
        gp2 = bike_data.groupby(by = x2).mean()
    else:
        gp = bike_data.groupby(by = x1).sum()
        gp2 = bike_data.groupby(by = x2).sum()
    gp = gp.reset_index()
    gp2 = gp2.reset_index()
    sns.barplot(x= x1, y = y1, data = gp, ax=ax[0][0])
    sns.barplot(x= x2, y = y2, data = gp2, ax=ax[0][1])
    fig.subplots_adjust(top = 0.90)
    plt.show()
```

```
#plotting Distribution of counting of bike renting w.r.t each year
gp = bike_data.groupby(by = ['yr', 'mnth']).sum().reset_index()
sns.factorplot(x= 'mnth', y = 'cnt', data = gp, col = 'yr', kind = 'bar
')
```

```
# Ploting barplot for counting wrt weekday
plot_bar('weekday', 'cnt', 'weekday', 'cnt')
```

```
# define function for making bins in mnth and weekday variable
def feat_month(row):
     if row['mnth'] <= 4 or row['mnth'] >=11:
        return(0)
     else:
        return(1)
def feat_weekday(row):
     if row['weekday'] < 2:
        return(0)
```

```
    else:
        return(1)
bike_data['month_feat'] = bike_data.apply(lambda row : feat_month(row),
axis=1)
bike_data = bike_data.drop(columns=['mnth'])
bike_data['week_feat'] = bike_data.apply(lambda row : feat_weekday(row)
, axis=1)
bike_data = bike_data.drop(columns=['weekday'])
```

```
# Correlation plot for each numerical variables
sns.pairplot(bike_data[numerical_columns])
```

```
#Heat map for each numerical variable
fig = plt.figure(figsize = (14,10))
corr = bike_data[numerical_columns].corr()
sn_plt=sns.heatmap(corr,mask = np.zeros_like(corr, dtype = np.bool),squ
are=True,
            annot= True, cmap = sns.diverging_palette(220, 10, as_cmap=
True))
plt.title("Heatmap of correlation between numerical variables")
fg = sn_plt.get_figure()
fg.savefig('heatmap.png')
```

```
#Chi-square test for each categorical variable
catego_columns=['season','yr','month_feat','holiday','week_feat','workingda
y', 'weathersit']
```

In [20]:

```
# Making combination from cat_columns
factors_paired = [(i,j) for i in catego_columns for j in catego_columns
]
```

```
from scipy.stats import chi2_contingency
# Applying chi-square test for every combination
p_values = []
for factor in factors_paired:
    if factor[0] != factor[1]:
        chi2, p, dof, ex = chi2_contingency(pd.crosstab(bike_data[facto
r[0]],
                                                        bike_data[factor[1]
]))
        p_values.append(p.round(3))
    else:
        p_values.append('-')
p_values = np.array(p_values).reshape((7,7))
p_values = pd.DataFrame(p_values, index=catego_columns, columns=catego_
columns)
print(p_values)
```

```
# checking importance of features
drop_col = ['cnt', 'instant','dteday', 'registered', 'casual']
from sklearn.ensemble import ExtraTreesRegressor
```

```
reg = ExtraTreesRegressor(n_estimators=200)
X = bike_data.drop(columns= drop_col)
y = bike_data['cnt']
reg.fit(X, y)
imp_feat = pd.DataFrame({'Feature': bike_data.drop(columns=drop_col).co
lumns,
                         'importance':reg.feature_importances_})
imp_feat.sort_values(by = 'importance', ascending=False).reset_index(dr
op = True)
```

```
from statsmodels.stats.outliers_influence import variance_inflation_fac
tor as vf
```

```
from statsmodels.tools.tools import add_constant
```

```
numeric_df = add_constant(bike_data[['temp', 'atemp', 'hum', 'windspeed
']])
vif = pd.Series([vf(numeric_df.values, i) for i in range(numeric_df.sha
pe[1])],
                index = numeric_df.columns)
vif.round(1)
```

```
# Dropping unwanted variables from dataset
bike_data.drop(columns=['instant', 'dteday', 'holiday', 'atemp', 'casua
l','registered'], inplace=True)
```

```
bike_data.head()
```

```
print('shape of original dataset',bike_data.shape)
```

```
#Establish a function which will build model on training set and would
show result
    # for k-fold & R-Squared and also show result for training & test
from sklearn.metrics import explained_variance_score
from sklearn.model_selection import cross_val_score
def fit_predict_show_performance(regressor, X_train, y_train, X_test, y
_test):
    regressor.fit(X_train, y_train)
    y_pred = regressor.predict(X_test)
    ten_performances = cross_val_score(estimator=regressor,X=X_train, y
=y_train,
                                       cv = 10,scoring='explained_varia
nce')
    k_fold_performance = ten_performances.mean()
    print("K-fold (K = 10) R-Squared")
    print(k_fold_performance)
    print()
    print("on train data R-Squared")
    print(regressor.score(X_train, y_train))
    print()
    print("on test data R-Squared")
    print(regressor.score(X_test, y_test))
```

```
# Splitting dataset in train and test for whole dataset after EDA
X = bike_data.drop(columns=['cnt'])
y = bike_data['cnt']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0
.2, random_state = 0)
```

```
#Decision tree model
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state=1)
fit_predict_show_performance(regressor, X_train, y_train,X_test,y_test)
```

```
#Random Forest model
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(random_state=1)
fit_predict_show_performance(regressor, X_train, y_train,X_test,y_test)
```

```
#Linear Regression model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
fit_predict_show_performance(regressor, X_train, y_train,X_test,y_test)
```

```
#KNN
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor(n_neighbors=5)
fit_predict_show_performance(regressor, X_train, y_train,X_test,y_test)
```

```
#SVM
from sklearn.svm import SVR
regressor = SVR()
fit_predict_show_performance(regressor, X_train, y_train,X_test,y_test)
```

```
#Hyperparameter tuning
from sklearn.model_selection import GridSearchCV
regressor = RandomForestRegressor(random_state=1)
params = [{'n_estimators' : [500, 600, 800],'max_features':['auto','sqr
t','log2'],
            'min_samples_split':[2,4,6],'max_depth':[12, 14, 16],
            'min_samples_leaf':[2,3,5],'random_state' :[1]}]
grid_search = GridSearchCV(estimator=regressor, param_grid=params,cv=5,
                           scoring = 'explained_variance', n_jobs=-1)
grid_search = grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
```

```
#Building Random forest on tuned parameter
regressor = RandomForestRegressor(random_state=1, max_depth=14, n_estim
ators=600,
                                  max_features='auto', min_samples_leaf
=2,
                                  min_samples_split=2)
fit_predict_show_performance(regressor, X_train, y_train, X_test, y_tes
t)
```

9. APPENDIX B-

COMPLETE R CODE

```
# importing required libraries
required_library <- c('ggplot2', 'corrgram', 'corrplot',
'randomForest',
                      'caret', 'class', 'e1071', 'rpart',
'mlr','grid',
                      'DMwR','usdm','dplyr','caTools','LiblineaR')
# checking each library whether installed or not
# if not install then installing it first and then attaching to file
for (lib in required_library){
  if(!require(lib, character.only = TRUE))
  {
    install.packages(lib)
    require(lib, character.only = TRUE)
  }
}
# Set working directory
setwd("C:/Users/Ankit Saxena/Desktop/Edwisor/Project")
getwd()
#Load data in R
bike_data = read.csv("bike_rentdata.csv")

#Exploratory Data Analysis
#cheking datatypes of all columns
str(bike_data)
# Assigning column names
numeric_columns <- c('temp', 'atemp', 'hum', 'windspeed', 'casual',
'registered', 'cnt')
cat_columns <- c('season', 'yr', 'mnth', 'holiday', 'weekday',
'workingday', 'weathersit')
# Checking numerical statistics of numerical columns
```

```
summary(bike_data[,numeric_columns])
```

 # Checking unique values in each category

```
lapply(bike_data[,cat_columns], function(feat) length(unique(feat)))
```

 # counting of each unique values in each categorical columns

```
lapply(bike_data[,cat_columns], function(feature) table(feature))
```

#Missing Value Analysis

```
missing_val <- data.frame(lapply(bike_data, function(feat)
sum(is.na(feat))))
```

#Outlier Analysis

```
# box_plot method to plot boxplot of numerical columns
box_plot <- function(column, dataset){
  dataset$x = 1
  ggplot(aes_string(x= 'x', y = column), data = dataset)+
    stat_boxplot(geom = 'errorbar', width = 0.5)+
    geom_boxplot(outlier.size = 2, outlier.shape = 18)+
    labs(y = "", x = column)+
    ggtitle(paste(" BP :",column))
}
# hist_plot function to plot histogram of numerical variable
hist_plot <- function(column, dataset){
  ggplot(aes_string(column), data = dataset)+
    geom_histogram(aes(y=..density..), fill = 'skyblue2')+
    geom_density()+
    labs(x = gsub('\\.', ' ', column))+
    ggtitle(paste(" Histogram :",gsub('\\.', ' ', column)))
}
# calling box_plot function and storing all plots in a list
all_box_plots <- lapply(c('temp', 'atemp', 'hum',
'windspeed'),box_plot, dataset = bike_data)
# calling hist_plot function and storing all plots in a list
```

```
all_hist_plots <- lapply(c('temp', 'atemp', 'hum',
'windspeed'),hist_plot, dataset = bike_data)

# printing all plots in one go

gridExtra::grid.arrange(all_box_plots[[1]],all_box_plots[[2]],all_bo
x_plots[[3]],all_box_plots[[4]],


all_hist_plots[[1]],all_hist_plots[[2]],all_hist_plots[[3]],all_hist
_plots[[4]],ncol=4,nrow=2)
```

#Feature Engineering

```
# method which will plot barplot of a columns with respect to other
column

plot_bar <- function(cat, y, fun){

  gp = aggregate(x = bike_data[, y], by=list(cat=bike_data[, cat]),
FUN=fun)

  ggplot(gp, aes_string(x = 'cat', y = 'x'))+

    geom_bar(stat = 'identity')+

    labs(y = y, x = cat)+

    ggtitle(paste("Bar plot for",y,"wrt to",cat))

}

# plotting cnt with respect to month

plot_bar('mnth', 'cnt', 'sum')

# plotting cnt with respect to yr

plot_bar('yr', 'cnt', 'sum')


# making bins of mnth and weekday

# changing values of month 5th to 10th as 1 and others 0

bike_data = transform(bike_data, mnth = case_when(

  mnth <= 4 ~ 0,

  mnth >= 11 ~ 0,

  TRUE   ~ 1

))

colnames(bike_data)[5] <- 'month_feat'


# changing values of weekday for day 0 and 1 the value will be 0
```

```
#and 1 for rest
bike_data = transform(bike_data, weekday = case_when(
  weekday < 2 ~ 0,
  TRUE   ~ 1
))
colnames(bike_data)[7] <- 'week_feat'
```

#Feature Selection

 # correlation plot for numerical feature

```
corrgram(bike_data[,numeric_columns], order = FALSE,
         upper.panel = panel.pie, text.panel = panel.txt,
         main = "Correlation Plot for bike data set")
```

 # heatmap plot for numerical features

```
corrplot(cor(bike_data[,numeric_columns]), method = 'color', type =
'lower')


cat_columns <- c('season', 'yr', 'month_feat', 'holiday',
'week_feat', 'workingday', 'weathersit')
```

 # making every combination from cat columns

```
combined_cat <- combn(cat_columns, 2, simplify = F)
```

 # doing chi-square test for every combination

```
for(i in combined_cat){
  print(i)
  print(chisq.test(table(bike_data[,i[1]], bike_data[,i[2]])))
}
```

 # finding important features

```
important_feat <- randomForest(cnt ~ ., data = bike_data[,-
c(1,2,14,15)],
                               ntree = 200, keep.forest = FALSE,
importance = TRUE)
```

```
importance_feat_df <- data.frame(importance(important_feat, type =
1))
```

 # checking vif of numerical column withhout dropping multicollinear column

```
vif(bike_data[,c(10,11,12,13)])
```

 # Checking VIF values of numeric columns after dropping multicollinear column i.e. atemp

```
vif(bike_data[,c(10,12,13)])
```

 # Making factor datatype to each category

```
bike_data[,cat_columns] <- lapply(bike_data[,cat_columns],
as.factor)
```

 # releasing memory of R, removing all variables except dataset

```
rm(list = setdiff(ls(),"bike_data"))
# checking dimension of both dataset
dim(bike_data)
```

 # dropping unwanted columns

```
drop_col <- c('instant', 'dteday', 'holiday', 'atemp', 'casual',
'registered')
bike_data[,drop_col]<- NULL
```

#Models Development in R

```
set.seed(1)
split = sample.split(bike_data$cnt, SplitRatio = 0.80)
train_set = subset(bike_data, split == TRUE)
test_set = subset(bike_data, split == FALSE)
```

 # making a function which will train model on training data and would show

 # K-fold R squared , R squared for test dataset and train dataset

```
fit.predict.show.performance <- function(method, train_data,
test_data){
  reg_fit <- caret::train(cnt~., data = train_data, method = method)

  y_pred <- predict(reg_fit, test_data[,-10])
  print("Adjusted R squared on test dataset")
  print(caret::R2(y_pred, test_data[,10]))

  y_pred <- predict(reg_fit, train_data[,-10])
  print("R2 on train dataset")
  print(caret::R2(y_pred, train_data[,10]))

  # creating 10 folds of data
  ten_folds = createFolds(train_data$cnt, k = 10)
  ten_cv = lapply(ten_folds, function(fold) {
    training_fold = train_data[-fold, ]
    test_fold = train_data[fold, ]
    reg_fit <- caret::train(cnt~., data = training_fold, method =
method)

    y_pred <- predict(reg_fit, test_fold[,-10])
    return(as.numeric(caret::R2(y_pred, test_fold[,10])))
  })
  sum = 0
  for(i in ten_cv){
    sum = sum + as.numeric(i)
  }
  print("K-fold (K =10) explained variance")
  print(sum/10)
}
```

#Decision Tree Model

```
fit.predict.show.performance('rpart2', train_set, test_set)
```

#Random Forest model

```
fit.predict.show.performance('rf', train_set, test_set)
```

#Linear Regression

```
fit.predict.show.performance('lm', train_set, test_set)
```

#KNN

```
fit.predict.show.performance('knn', train_set, test_set)
```

#SVM

```
fit.predict.show.performance('svmLinear3', train_set, test_set)
```

#Tuning Random Forest with the help of hyperparameter tuning for bike_data dataset

```
control <- trainControl(method="repeatedcv", number=10, repeats=3)
reg_fit <- caret::train(cnt~., data = train_set, method =
"rf",trControl = control)
reg_fit$bestTune
y_pred <- predict(reg_fit, test_set[,-10])
print(caret::R2(y_pred, test_set[,10]))
```

10. REFERENCES

[1] Jay (Haijie) Gu. Using Gradient Boosted Trees to Predict Bike Sharing Demand Available at http://blog.dato.com/using-gradient-boosted-trees-to-predict-bike-sharing-demand

[2] R Data: http://cran.r-project.org/

[3] Data Mining And Business Analytics with R‖, Johannes Ledolter, Dept. of Management Studies, University of IOWA.

[4] Ting Ma, Chao Liu, Sevgi Erdoan. Bicycle Sharing and Transit: Does Capital Bikeshare Affect Metrorail Ridership in Washington, D.C.? Available at http://smartgrowth.umd.edu/assets/bikeshare_transit_for_parisws_v1.pdf

[5] https://towardsdatascience.com

[6] Elegant Graphics for Data Analysis, Wickham, H. ―http://www.springer.com/978-0-387-981040-6.

[7] Han and Kamber. Data Mining: Concepts and Techniques. Second Morgan Kaufman Publisher, 2006

[8] Wikipedia ―www.wikipedia.com‖.