Project Report On
Bike Renting Prediction

Made by: SAKSHI RAPUT

22 April, 2019

Chapter 1

Introduction

A bike renting system is a service in which users can rent/use bicycles available for shared use on a short-term basis for a price or free. 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 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 bikes that will be used, using available information about that time/day.

1.1 Problem Statement

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

1.2 Data

In the given problem, data from a bike rental company is provided that contains 13 features or predictor variables like date, season, year, temperature, humidity. The data set also provide the target variable 'cnt' which is the count of bike rentals for that particular day.

To predict the target variable, we need to make a regressor model on the day.csv (dataset) provided.

Below is an insight to the dataset:

index	instant	dteday	season	yr	mnth	holiday	weekday	Working day
0	1	2011-01-01	1	0	1	0	6	0
1	2	2011-01-02	1	0	1	0	0	0
2	3	2011-01-03	1	0	1	0	1	1
3	4	2011-01-04	1	0	1	0	2	1
4	5	2011-01-05	1	0	1	0	3	1

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.3441	0.3636	0.8058	0.1604	331	654	985
2	0.3634	0.3537	0.696	0.2485	131	670	801
1	0.1963	0.1894	0.4372	0.2483	120	1229	1349
1	0.2	0.2121	0.5904	0.1602	108	1454	1562
1	0.2269	0.2292	0.4369	0.1869	82	1518	1600

1.2.1 Data Overview:

The following are the predictor variables:

- **instant**: Record index
- **dteday**: Date
- **season**: Season (1:springer, 2:summer, 3:fall, 4:winter)
- **yr**: Year (0: 2011, 1:2012)
- **mnth**: Month (1 to 12)
- **hr**: Hour (0 to 23)
- **holiday**: weather day is holiday or not (extracted fromHoliday Schedule)
- **weekday**: Day of the week
- **workingday**: If day is neither weekend nor holiday is 1, otherwise is 0.
- **weathersit**: (extracted fromFreemeteo)

  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_maxt_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)

The following are the target variables:

- **casual**: count of casual users
- **registered**: count of registered users
- **cnt**: count of total rental bikes including both casual and registered

Note: cnt is total of casual and registered

From the above observation we can separate the categorical and continuous variables:

S.No	Categorical features	Continuous features
1	season	instant
2	yr	temp
3	mnth	atemp
4	holiday	hum
5	weekday	windspeed
6	workingday	
7	weatherlist	

Chapter 2

Methodology

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

2.1 Exploratory Data Analysis (EDA)

Exploratory Data Analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

2.1 Data Preparation

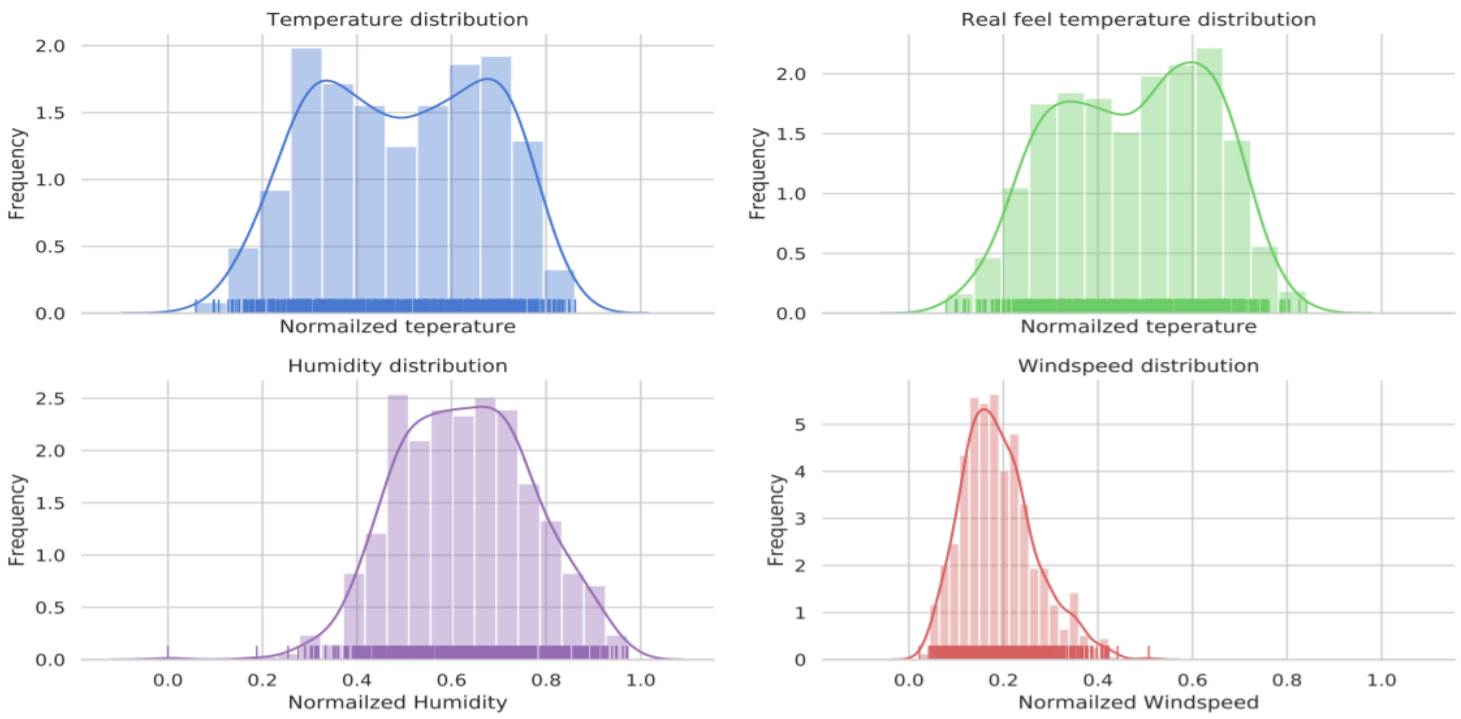
Check and verify that data types of feature, target variables provided in the dataset and data descriptor file are same or not.

```
#Function to segregate the different data types present in the dataframe.

def seperate_datatypes (df, col_names):

    obj = []

   num = []

   bool_d = []

   unknown = []

   for i in range(len(col_names)):

      if df.iloc[:,i].dtype == 'O':

         obj.append(col_names[i])

      elif df.iloc[:,i].dtype == 'int64' or df.iloc[:,i].dtype == 'float64':
```

```
num.append(col_names[i])
    elif df.iloc[:,i].dtype == 'bool':
        bool_d.append(col_names[i])
    else:
        unknown.append(col_names[i])
 return obj, num, bool_d, unknown
```

- After getting the list of different data types, it was noted that the features season, yr, mnth, holiday, weekday, workingday and weathersit were incorrectly saved as numerical data type and hence they are to be converted back to categorical data type.
- From viewing the meta data file and the data set, it can seen that the feature instance is just the index of the records and causal and registered are the count of the respective type of users which is not required as we have the total count. Hence instance, causal and registered columns are dropped from the data set.
- The dteday is the date of the records which is not required as we are not conducting a time series analysis. Hence even dteday column is dropped from the data set.
- Now there is eleven features and one label.

Next the probability distribution of each numerical column is plotted for visual examination to see if the data was normally distributed



It can be see that they are all mostly normally distributed but windseed is slightly skewed.

Code to analyse the skewness of the continous variables distribution.

```
for i in num_dtype:
from scipy import stats
skewness = stats.describe(df.loc[:,i])
print("statistical properities of {0:5s}:".format(i))
print(skewness)
print("**********************************************************")
```

OUTPUT:

```
In [307]: print("**********************************************************************")
          for i in num_dtype:
              from scipy import stats
              skewness = stats.describe(df.loc[:,i])
              print("statistical properities of {0:5s}:".format(i))
              print(skewness)
              print("**********************************************************************")
```

```
**********************************************************************
statistical properities of temp :
DescribeResult(nobs=731, minmax=(0.0591304, 0.861667), mean=0.495384788508892, variance=0.03350766717740828, skewness=-
0.05440902400571610, kurtosis=-1.1194225400473057)
**********************************************************************
statistical properities of atemp:
DescribeResult(nobs=731, minmax=(0.0790696, 0.8408959999999999), mean=0.47435398864569084, variance=0.026556345661055517
4, skewness=-0.1308188980737412, kurtosis=-0.9866019052943136)
**********************************************************************
statistical properities of hum  :
DescribeResult(nobs=731, minmax=(0.0, 0.9725), mean=0.6278940629274967, variance=0.02028604714193028, skewness=-0.06964
015783152368, kurtosis=-0.07228631791987006)
**********************************************************************
statistical properities of windspeed:
DescribeResult(nobs=731, minmax=(0.0223917, 0.507463), mean=0.190486211627907, variance=0.006005919960192755, skewness=
0.6759547264275362, kurtosis=0.39992023832685497)
**********************************************************************
statistical properities of cnt  :
DescribeResult(nobs=731, minmax=(22, 8714), mean=4504.3488372093025, variance=3752788.2082828926, skewness=-0.047255557
55362063, kurtosis=-0.8145762269613592)
**********************************************************************
```

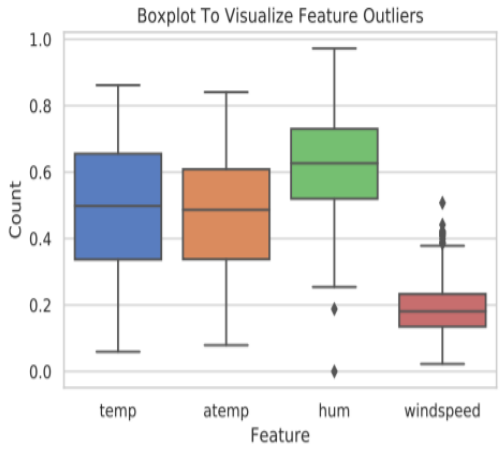
It can said that the continuous variables are all mostly normally distributed but there is a slight skew in all of them. This could be due to some outliers present in the data set. Those outliers will be dealt with in further parts of this report. The data distribution of the categorical variables were also checked and it was found all the variables except workingdays, holiday and weathersit where almost equally distributed.

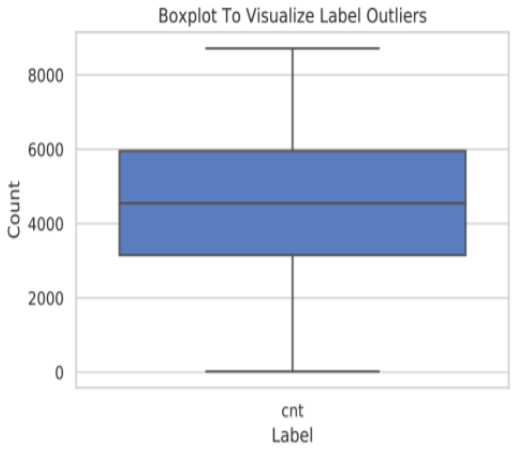
2.2 Missing Value Analysis

The data set was checked for missing values and was found that there were no missing values present in the data set.

2.3 Outlier Analysis

In order to visualize outliers, the classic approach of using boxplot is used.



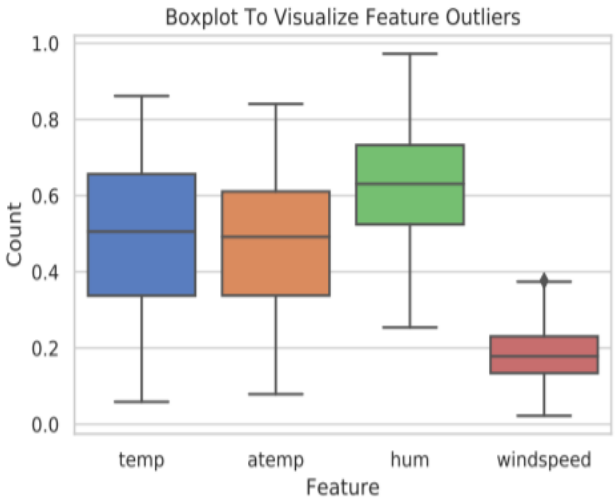


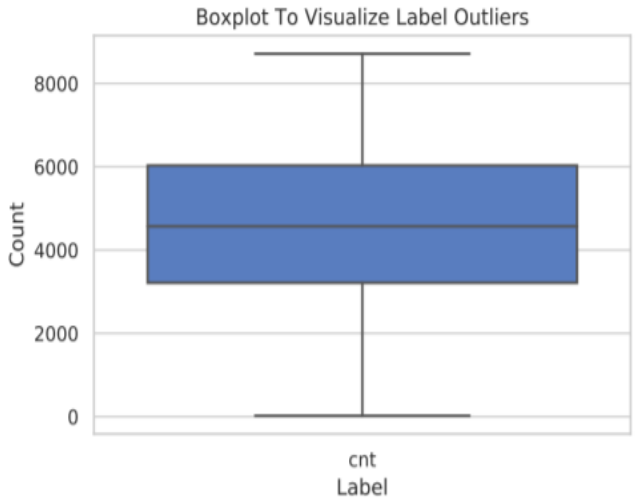
In order to decide as to how to deal with the outliers in the data set, the percentage of outliers in the data set is first calculated by sample code in listing below. The output of listing shows that the percentage of outliers present in the data set is 2.052%. In such a case it would be best to omit the records with outliers from the data set as it would not have much affect on the overall data. After removal of the outliers from the data set, the boxplot analysis is again conducted on the new data set. This is because when data is removed from the data set the mean of the data set and other statistical properties also changes and this may give is to new outliers.

Function to seperate the variables names into their respective data types.

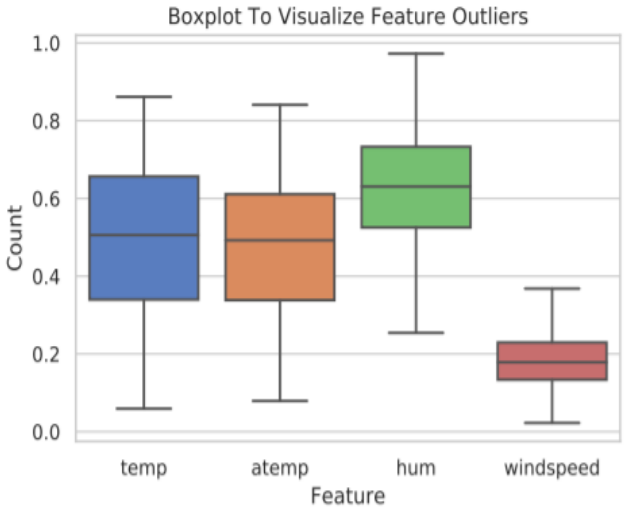
```
def outlier_percent(df):
    l_q = df.quantile(0.25)
    u_q = df.quantile(0.75)
    iqr = u_q - l_q
    total_outliers = ((df<(l_q - 1.5*iqr))|(df>(u_q + 1.5*iqr))).sum()
    outliers = total_outliers.sum()
    print('Total number of outliers present =',outliers)
    r,c = df.shape
    outlier_percentage = (outliers/r)*100
    print("Outlier percentage :",round(outlier_percentage,3))
```

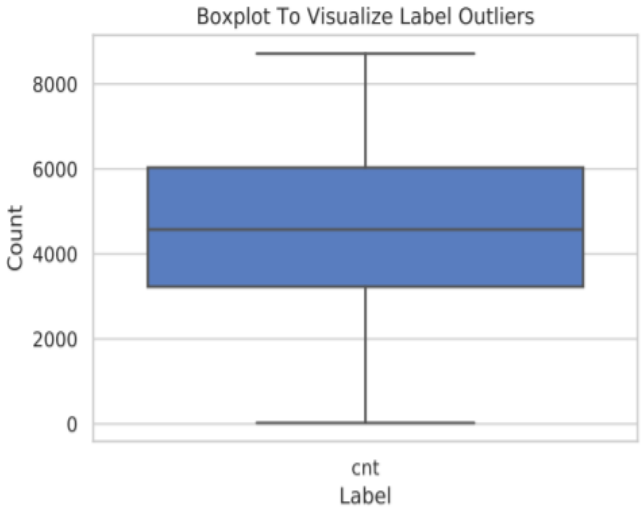
By running THE ABOVE MENTIONED FUNCTION on the new data set it can be seen that the percentage of outliers present in the data set in 0.418%.





Now we remove these outliers from the data set and repeat the process of plotting the new data set 10 in box plot and examining for outliers. this time only one outlier is found in windspeed variable. That amounts to 0.14% percentage of outliers present. This too is removed and the data set is again checked for outliers. Below Image shows that the data set is free from outliers. The data set is now reduced from 731 records to 713 records.

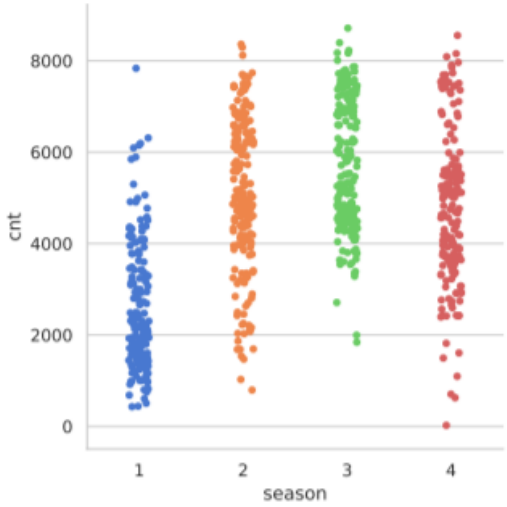




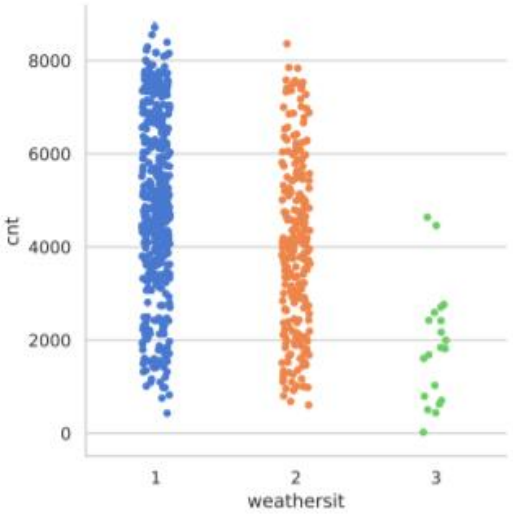
2.4 Data Understanding

In order to get further insight and understand of the data set, one should see how different features interact with each other and the target. First the amount of bike rental counts for each day of the week is analyzed.

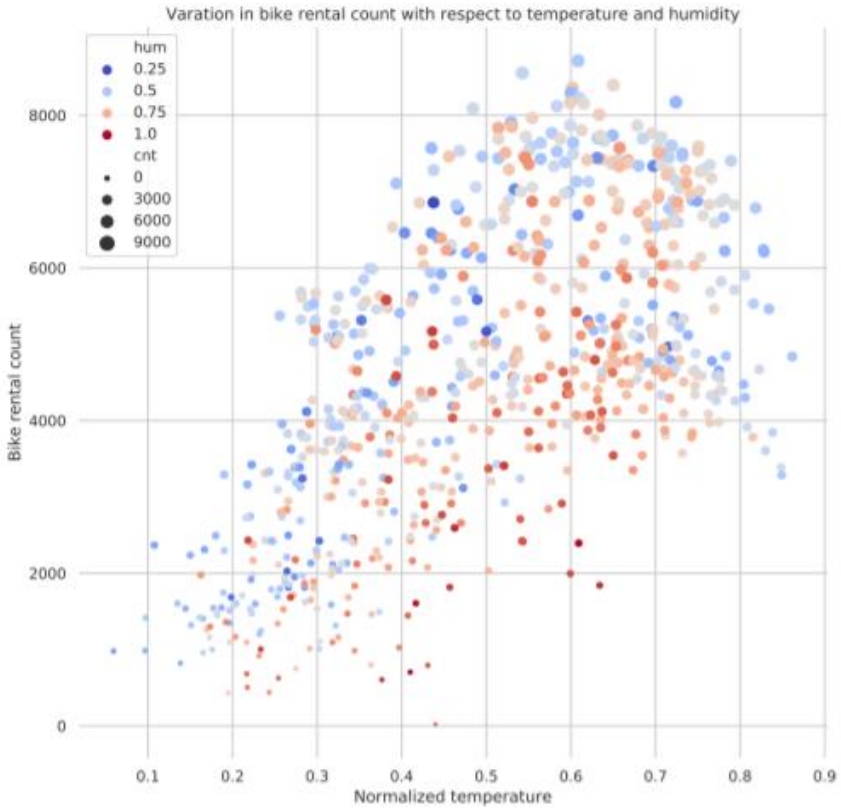
- From first figure it can be said that the majority of the bike rental each day is between 3500 - 6000. Whereas days 0, 5 and 6 is almost evenly distributed along the cnt (’bike count’) axis.
- Figure 2 shows that the maximum amount of rentals happen in the months 4 - 10. These months could reach rentals counts of 4000 to 8000 and even more.



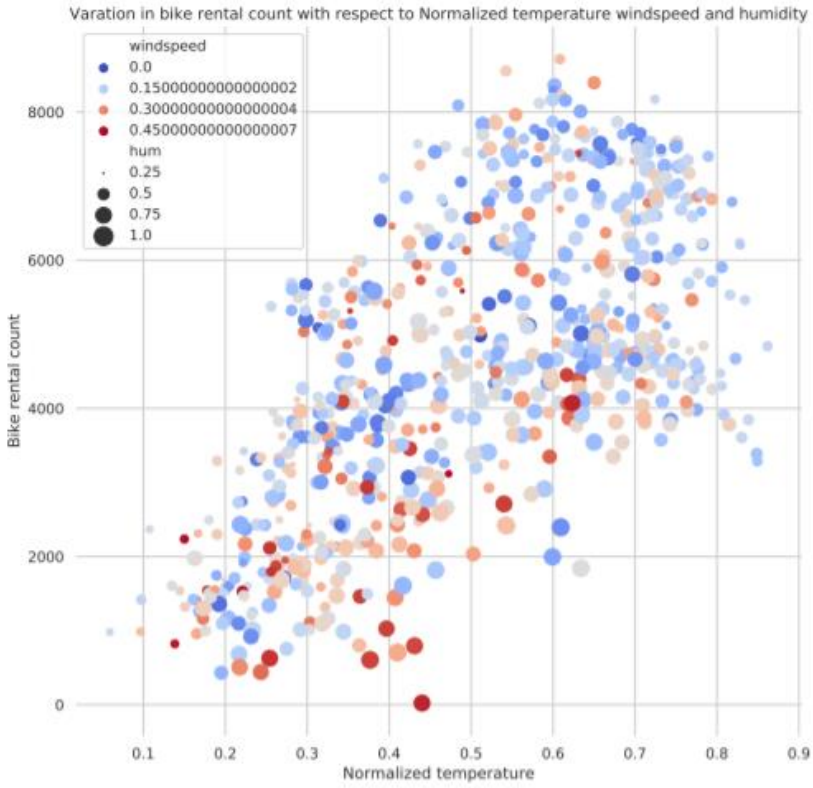
The above figure shows the variation in bike rentals based on the season. It can be observed that season 3 which is fall is the most busiest, having majority of the daily bike count between 4000 - 8000 rentals. On the other hand season 1 which is spring is the least where the majority of the daily bike count is mostly in the range 500 - 4000.



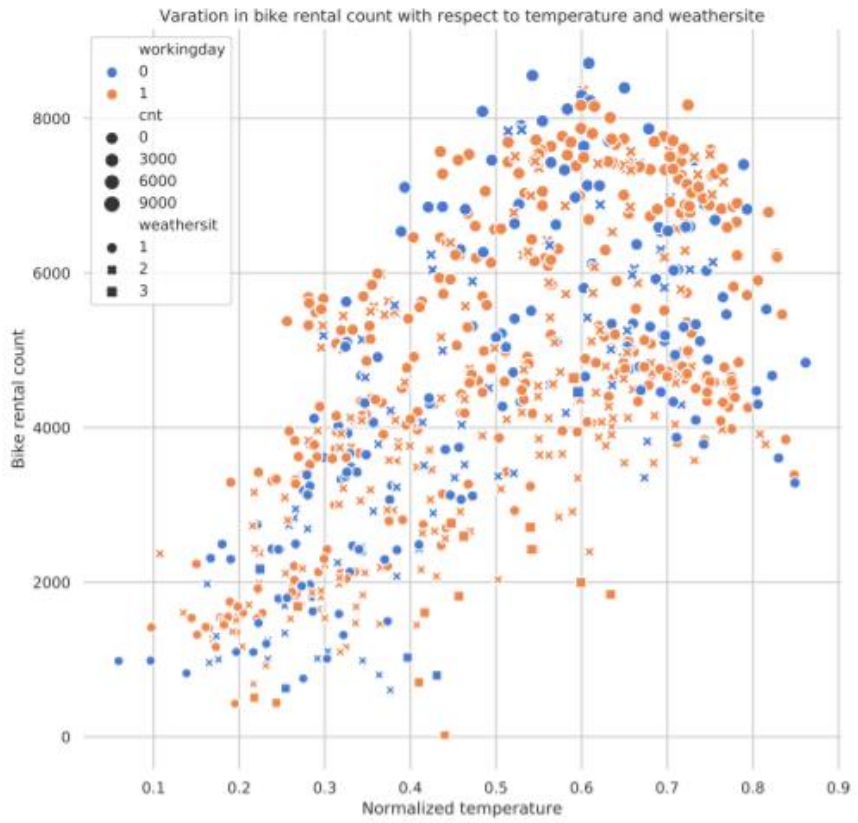
From above figure, it can be seen that weather site forecast 3 has barley any bike rentals. This weather corresponds to Light Snow, Light Rain, Thunderstorm or or Scattered clouds.



From above figure,it can be seen that majority of the bike rental counts are between humidity levels of 0.25 - 0.75 and normalized temperature levels of 0.4 - 0.8. For humidity levels of 0.75-1 there is significantly lesser bike rental counts.



From above figure, the variation in bike rental based on the change in normalized temperature, humidity and wind speed is shown. Based on both figures, it could be inferred that the optimal conditions for maximum bike count is when the normalized temperature is between 0.4 - 0.8, normalized humidity is below 0.75 and normalized winspeed is below 0.15.



To see how the bike rental count varies based on working day and the weather site information above figure was generated. Here bike rental count is plotted against normalized temperature and categorical variables 'workingday' and 'weathersit'. From figure 2.15 it can be seen that majority of the bike rental counts were on working day 1 which is neither a weekend nor a holiday. Therefor it can be said that majority of the bike rental counts are from working days Monday to Friday. It can also be noted that the weather site forecast is 2 (Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist) is maximum between the region of rentals counts 0 - 6000 (y-axis) . However the weathersite forecast 1 (Clear, Few clouds, Partly cloudy) dominates the region of bike counts of 6000-8500. Weather site 3 (Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds.) has the least amount of bike rental counts.

2.5 Feature Selection

To develop an efficient model, it is required to know which are the features which are of most importance in predicting the target variable. It is possible that many variable in our data set is less important in predicting the target than others or they my just be redundant. In this case it is required that we remove these variables from our data set to reduce its complexity. At times two features may carry the same information. In such case it is required to remove one of the features to avoid multi collinearity problems. There are many way to perform feature selection or dimensionality reduction, but the method selected is this report is Correlation Analysis for continuous features and ANOVA test for categorical features.

2.5.1 Correlation Analysis

Correlation Analysis is a technique which helps to determine how strongly two features are related to each other(i.e their co-variance). As the co-variance can vary from - infinity to + infinity, the correlation is used as it is a scaled version of the co-variance having values ranging from -1 to +1. A correlation plot is shown in figure 2.16. A correlation threshold of 0.8 is set and feature pairs of which exceeds this threshold, one of them is dropped. Table 2.1 corresponds to the values of the correlation figure 2.16. By examining table 2.1 it can be observed that features temp and atemp are highly correlated. Hence the feature atemp is dropped from the analysis.

Table 2.1: Correlation analysis table

	temp	atemp	hum	windspeed	cnt
temp	1.0	0.99	0.1	-0.13	0.62
atemp	0.99	1.0	0.11	-0.15	0.63
hum	0.1	0.11	1.0	-0.2	-0.13
winsdpeed	-0.13	-0.15	-0.2	1.0	-0.2
cnt	0.6	0.63	-0.13	-0.2	1.0

2.5.2 Analysis Of Variance (ANOVA)

Analysis of Variance (ANOVA) is a statistical technique used to compute and compare the mean between two or more groups of observations. ANOVA makes use of two variables which are categorical variables and numeric variables of the data set. The python code is used to compute the p values of the feature and target variables. Also the output p- values to the threshold value of 0.05 and saves the feature names to be dropped in drop_feat variable. Code and the output of the code is shown below :

Here is the python code for calculation of p values of the features and target variables.

```
label = 'cnt'
## ANOVA TEST FOR P VALUES
import statsmodels.api as sm
from statsmodels.formula.api import ols

anova_p = []
for  i in obj_dtype:
     buf = label + ' ~ ' + i
     mod = ols(buf,data=df).fit()
     anova_op = sm.stats.anova_lm(mod, typ=2)
     print(anova_op)
     anova_p.append(anova_op.iloc[0:1,3:4])
     p = anova_op.loc[i,'PR(>F)']
     if p >= 0.05:
          drop_feat.append(i)
```

	sum_sq	df	F	PR(>F)
season	9.305377e+08	3.0	127.350479	5.482108e-66
Residual	1.726865e+09	709.0	NaN	NaN

	sum_sq	df	F	PR(>F)
yr	8.887398e+08	1.0	357.27215	7.173240e-65
Residual	1.768663e+09	711.0	NaN	NaN

	sum_sq	df	F	PR(>F)
mnth	1.049040e+09	11.0	41.565535	3.197860e-69
Residual	1.608363e+09	701.0	NaN	NaN

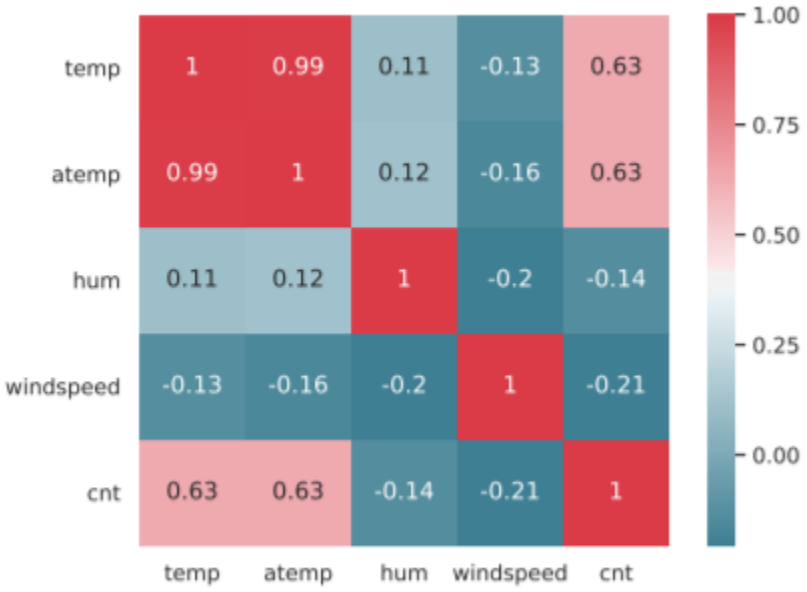
	sum_sq	df	F	PR(>F)
holiday	1.415864e+07	1.0	3.808499	0.051385
Residual	2.643244e+09	711.0	NaN	NaN

	sum_sq	df	F	PR(>F)
weekday	1.970414e+07	6.0	0.878994	0.509776
Residual	2.637698e+09	706.0	NaN	NaN

	sum_sq	df	F	PR(>F)
workingday	7.117269e+06	1.0	1.909371	0.167467
Residual	2.650285e+09	711.0	NaN	NaN

	sum_sq	df	F	PR(>F)
weathersit	2.672556e+08	2.0	39.694529	4.553787e-17
Residual	2.390147e+09	710.0	NaN	NaN

From above figure, we can see that features holiday, weekday and workingday are greater than the p value threshold (0.05) and hence they are dropped along with atemp. Now the number of features and target variables have reduced from 13 and 3 to 7 and 8 respectively.



2.6 Feature Engineering

Now that most of the pre processing is over, the last thing left to do is to convert the categorical variables into their respective dummy variables by creating a feature for each category group in the categorical feature. As our categorical variables are in the form of numbers and not strings we need not convert them to numbers, but it is required that we create dummy variables for each categorical group. As the categorical features in our data set are nominal categorical data and not ordinal categories the model shouldn't give higher precedence to a higher number. For example, the categories present in the feature seasons is 1, 2, 3 and 4 which corresponds to spring, summer, fall and winter respectively. As the categories are numeric, the model would give a higher precedence to 4 as it is numerically greater than the rest. Which is not so as they are not ordinal categories and each number refers to a specific season. Hence each group is converted to a binary category. Where 1 imply the presence of that group and 0 represents absence.

To make it less complex the first dummy variable is dropped, as when all the other groups are absent it imply that the dropped group is present. This is achieved by running the command in code below :

```
1
2 ## Creating dummy variables
3 # Where x is all the features in our analysis.
4
5 X_lr = pd.get_dummies(X, columns = obj_dtype, drop_first = True)
```

Chapter 3

Modeling

3.1 Model Selection

It has be noted in previous stages of our analysis that for different combinations of the independent variables, the count is different. It can be seen that the dependent variable is a continuous variable and hence the type of model that would be developed for this problem is a regression model.

3.2 Methodology:

Model Evaluation is an integral part of the model development process as it helps us find the best model for representing our data. It also helps to evaluate as to how it would on new data. In order to develop an efficient and accurate model to predict our target variable we shall use a combination of three different methods

The three different methods used in our analysis is given below.

- Hold-Out Method
- Hyper parameter Tuning
- Cross-Validation Technique

3.2.1 Hold-Out Method

As evaluating model performance on training data set may lead to develop an over fitted model. Due to this is is required to test the model on a separate data set. Hence the original data set is split into training and testing data. The training data set is used to build a predictive model and the testing data is used to evaluate the model performance.

3.2.2 Hyper parameter Tunning

Hyper parameter Tuning is a method in which the parameters of the model is to be set before training. Scikit-Learn implements a set of sensible default hyper parameters but it may not be optimal. In our analysis we shall use two types of hyper parameter tuning methodology which are Random Search CV and Grid Search CV. The major difference between them is the run time.

As randomized search is drastically lower than grid search. Random search is faster than grid search as we do not provide a set of discrete values to be searched. Rather we provide a statistical distribution for each parameter from which values may be randomly sampled. Where as in grid search a discrete set of values are provided for each parameter and several models are built on each of its combinations which makes it very laboursome.

3.2.3 K-Fold Cross Validation Technique

In K fold cross validation technique the data is divided into k subsets of equal size. We build the model K times, each time leaving out a subset from training. This sub set which was left out will be used for testing. This method helps us develop an unbiased estimate of the model performance.

3.3 Approach

At first thee data is split to training and testing data set as per the hold out method. Then Hyperparameter tuning and k fold cross validation are performed together. First Random Search CV is performed on a wide range of parameter values. This is performed first as its a hyperparameter tuning cross validation technique in which the run time required is less. Based on its results we can narrow are search parameters and perform grid search to come up with an optimal model.

3.4 Model Building

We Start building our model by using the simplest model to the most complex model. Therefor we start our model building using the multiple linear regression model.

3.4.1 Multivariate Linear Regression Model

Before the model is built in multivariate linear regression it is required to do some processing on the data before hand. The equation around which the multivariate regression in statsmodels works is shown below.

$y = \beta 0 + \beta 1 * X1 + \beta 2 * X2 + ... + \beta n * Xn$

Where β1, β2 till βn are the coefficients for columns X1, X2 till Xn respectively and β0 represents the intercept. Unfortunately in statsmodel the β0 column is not added by default to out model and hence we are required to add a column with a constant value of one to the data set. Now that this column is added and the dummy variables are created we can go ahead with building the model. Code below shows the Multivariate Linear Regression Modeling. Figure shows the model summary.

Creating dummy variables for multivarient linear regression

```
X_l = pd.get_dummies(df, columns = obj_dtype, drop_first = True)
X_l["b0"]=1
col_name = X_l.columns.tolist()
col_name = col_name[21:22] + col_name[0:3]+ col_name[4:21]+ col_name[3:4]
X_l = X_l.loc[:, col_name]
#Divide data into train and test
train_stat, test_stat = train_test_split(X_l, test_size=0.2)
np.random.seed(0)
# ## Linear Regression
#Import libraries for LR
import statsmodels.formula.api as sm
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

# Train the model using the training sets
model = sm.OLS(train_stat.iloc[:,21], train_stat.iloc[:,0:21]).fit()
```

OLS Regression Results

Dep. Variable:	cnt	**R-squared:**	0.845
Model:	OLS	**Adj. R-squared:**	0.840
Method:	Least Squares	**F-statistic:**	150.1
Date:	Fri, 04 Jan 2019	**Prob (F-statistic):**	1.67e-207
Time:	13:07:27	**Log-Likelihood:**	-4587.5
No. Observations:	570	**AIC:**	9217.
Df Residuals:	549	**BIC:**	9308.
Df Model:	20		
Covariance Type:	nonrobust		

It can be seen from figure above that the Adjusted R-squared is 0.83 which is pretty impressive, as it means 83% of the variance of the data is explained by the model. Looking at the F-statistic and combined p-value we can reject the null hypothesis that target variable does not depend on any of the predictor variables.

3.4.2 Decision Tree Regressor

Now a decision tree regressor model is built on to our data set. At first a model is built on the default parameters of the Decision tree regressor. The default values for the parameters controlling the size of the trees such as max_depth, min_samples_leaf, etc lead to fully grown and unpruned trees which could require large memory consumption and increase the model complexity. It can also cause overfitting of the model. To reduce the complexity and memory consumption the size of the trees should be controlled by setting those parameter values. In our analysis we are going to tune the max_depth parameter which controls the maximum depth of the tree. The default model without any tuning is taken as the base line model.

After this a grid of intervals of 5 is taken from 5 - 50 for max_depth are given to random search cv. The k fold cross validation parameter given is K is 5 and n_iter is 5, which the cross validation is to be repeated5times. The optimal depth from random searchcv is 10. Now that we narrowed down our search criteria, grid search cv can be conducted on range 5 - 20 max_depth with an interval of 2. Here it was found that the best parameters for max_depth is 5.

3.4.3 Random Forest Regressor

Random forest is an ensemble that consists of many decision trees At first a model is built on the default parameters of the Random Forest regressor. To reduce the complexity and memory consumption the max_depth and n_estimators parameters is tuned n_estimators tell the amount of trees to be used in the model. The default model without any tuning is taken as the base line model. After this a grid of intervals of 2 is taken from 1 - 20 max_depth and a grid of intervals of 2 is taken from 1 - 100 for n_estimators given to random search cv. The k fold cross validation parameter given is K is 5 and n_iter is 5, which the cross validation is to be repeated 5 times. The optimal depth from random search cv is 18 and n_estimators is 15. Now that we narrowed down our search criteria, grid search cv can be conducted on 15 - 25 for max_depth and intervals of 2 is taken from 11 - 17 for n_estimators. Here it was found that the best parameters for max_depth is 12 and n_estimators is 16.

3.4.4 Gradient Boosting

Gradient boosting is an ensamble boosting method in which a colection of regressors are built in series.It is a method of converting asequence of weak learners to a complex model. Each regressors prediction is based on th previous regressors by adding weights accordingly. We shall now implement the Gradint boosting model on our data. At first a model is built on the default parameters of the Random Forest regressor.
Toreducethecomplexityandmemoryconsumptionthemax_depthandn_estimatorsparameters is tuned n_estimators tell the amount of trees to be used in the model. The default model without any tuning is taken as the base line model. After this a grid of intervals of 2 is taken from 1 - 10 max_depth and a grid of intervals of 10 is taken from 50 - 150 for n_estimators given to random search cv. The k fold cross validation parameter given is Kis5andn_iteris5,whichthecrossvalidationistoberepeated5times. Theoptimaldepthfromrandom search cv is 23 and n_estimators is 15. Now that we narrowed down our search criteria, grid search cv can be conducted on range 1 - 5 for max_depth and intervals of 10 is taken from 120 - 140 for n_estimators. Hereit wasfound thatthe bestparametersfor max_depthis 3and n_estimatorsis 75

3.4.5 Extreme Gradient Boosting

The efficiency and accuracy of the gradient boosting model can further be improved on, by running the xgboost model on the data. Extreme Gradient Boosting, which is an efficient implementation of the gradient boosting frame work. The package xgboost is used to build the xgboost model on the data.

Chapter4

Conclusion

4.1 Model Evaluation

Now that a few models have been created on our data set, it is required that a suitable evaluation matrix is selected it compare the different model. As our problem statement is a regression problem. There are a few popular evaluation matrix which are commonly used. These matrix are : MAPE - Mean absolute percentage error, is the measure accuracy as a percentage of error, MSE - Mean squared error, it is the square root of the mean squared errors and RMSE - Root mean squared error, it is the square root of the mean squared errors. RMSE is mostly used in time series analysis and hence it is not used. There is a choice to choose between MAE and MAPE. As MAPE delivers the values in for of error percentage, it is selected as it is easy to understand. MSE is also selected as it gives us the goodness of fit of the model. The smaller the MSE the better the fit.

4.2 Model Selection

From table 4.1 it can be seen that Grid Search CV of Gradient Boost Machine, having the parameters for max_depth- 3 and n_estimator - 100, has the least MAPE of 18.06%. It has a R-Squared value of 0.88 which means it explains 88% of the variance, which is very good. The second best model is the XGBoosting model which has a MAPE of 18.25% and it also has a Rsquared value of 0.88 explaining 88%ofthevariance. We see that both the models perform comparitively well while comparing their MSE, R-Squared value and MAPE. Hence either Grid Search CV of Gradient Boost Machine with max_depth-3 and n_estimator-100 or XGBoosting model can be selected as the preferred model.

	Model_name	MSE	MAPE	R^2
0	Multivarient linear regression	5.715082e+05	18.437683	0.866874
1	Decision tree default	1.079306e+06	26.748537	0.748588
2	Decision tree Random Search CV	1.040601e+06	26.572396	0.757604
3	Decision tree Grid Search CV	9.135586e+05	25.073871	0.787197
4	Random Forest Default	6.151233e+05	20.272507	0.856714
5	Random Forest Random Search CV	5.962079e+05	20.508065	0.861120
6	Random Forest Grid Search CV	6.082217e+05	20.578772	0.858322
7	Gradient Boosting Default	5.141599e+05	18.506449	0.880232
8	Gradient Boosting Random Search CV	5.223807e+05	18.406009	0.878317
9	Gradient Boosting Grid Search CV	5.143945e+05	18.063431	0.880178
10	XGBOOST	4.991374e+05	18.254574	0.883732

Appendix A

Python code and its corresponding output:

Functions:

In [76]:

```
#Function to segregate the different data types present in the dataframe.
def seperate_datatypes(df, col_names):
    objct = []
    num = []
    bool_d = []
    unknown = []
    for i in range(len(col_names)):
        if df.iloc[:,i].dtype == 'O':
            objct.append(col_names[i])
        elif df.iloc[:,i].dtype == 'int64' or df.iloc[:,i].dtype == 'float64':
            num.append(col_names[i])
        elif df.iloc[:,i].dtype == 'bool':
            bool_d.append(col_names[i])
        else:
            unknown.append(col_names[i])
    return objct, num, bool_d, unknown
```

In [77]:

```
#Function to view the categories present in each categorical feature
def view_categorical_features(objct):

    for i in range(len(objct)):
        print('Feature:',objct[i])
        print(df[str(objct[i])].value_counts())
```

In [78]:
```
def percentage_of_outliers(df):
    lower_quartile = df.quantile(0.25)
    upper_quartile = df.quantile(0.75)
    iqr = upper_quartile - lower_quartile
    total_outliers = ((df<(lower_quartile - 1.5*iqr))|(df>(upper_quartile + 1.5*iqr))).sum()
    outliers = total_outliers.sum()
    print('Total number of outliers present =',outliers)
    r,c = df.shape
    outlier_percentage = (outliers/r)*100
    print("percentage_of_outliers :",round(outlier_percentage,3))
```

In [79]:
```
#Function to perform outlier analysis
def analysis_of_outlier(df, cnames, method):
    if method == 0:
        #Detect and delete outliers from data
        for i in cnames:
            q75, q25 = np.percentile(df.loc[:,i], [75 ,25])
            iqr = q75 - q25
            low_qutr = q25 - (iqr*1.5)
            up_qutr = q75 + (iqr*1.5)
            df = df.drop(df[df.loc[:,i] < low_qutr].index)
            df = df.drop(df[df.loc[:,i] > up_qutr].index)
        return df
    elif method == 1:
        for i in cnames:
            #Detect and replace with NA
            # #Extract quartiles
            q75, q25 = np.percentile(df[i], [75 ,25])

            #Calculate IQR
            iqr = q75 - q25

            #Calculate inner and outer fence
            low_qutr = q25 - (iqr*1.5)
            up_qutr = q75 + (iqr*1.5)

            #Replace with NA
            df.loc[df[i] < low_qutr,:i] = np.nan
            df.loc[df[i] > up_qutr,:i] = np.nan
            #Impute with KNN
            df = pd.DataFrame(KNN(k = 3).fit_transform(df[i]), columns = df.columns)
        return df
    else:
        print('Error: incorrect input')
```

In [80]:
```
def find_features_to_be_dropped(columns, rows, treshold):

    drop = []
    drop_val = []
    for i in list(range(0,len(columns))):
        for j in list(range(0, len(rows))):
            if columns[i] != rows[j]:
                if abs(corr.loc[rows[j],columns[i]]) >= abs(treshold):
                    drop.append(rows[j]) if rows[j] and columns[i] not in drop else print('repeat')
                    drop_val.append(corr.loc[rows[j],columns[i]])
    return drop
```

In [81]:
```
def drop_features_from_variables(drop_feat, obj_dtype, num_dtype):
    obj_del = []
    num_del = []
    for i in range(0, len(drop_feat)):
        #print(i)
        if(drop_feat[i] in obj_dtype):
            #print('obj in',i)
            ## Dropping deleted object variable name from obj_dtype variable
            obj_del.append(drop_feat[i])

        elif(drop_feat[i] in num_dtype):
            #print('num in',i)
            ## Dropping deleted numeric variable name from num_dtype variable
            num_del.append(drop_feat[i])

        else:
            print('Data type', i ,'is not an object or numeric.')
    obj_d= list(set(obj_dtype).difference(set(obj_del)))
    num_d= list(set(num_dtype).difference(set(num_del)))
    return obj_d, num_d
```

In [82]:
```
#Function to compute MAPE
def get_MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
```

Exploratory Data Analysis:

In [83]:
```
#Load data
df = pd.read_csv("day.csv")
```

In [84]:
```
df.head()
```

Out[84]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

In [87]:
```
#Calling function 'seperate_datatypes' to segregate different data types
obj_dtype, num_dtype, bool_dtype, unknown_dtype = seperate_datatypes(df, df.columns)
```

In [88]:
```
df.dtypes
```

Out[88]:
```
season          int64
yr              int64
mnth            int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
windspeed     float64
cnt             int64
dtype: object
```

In [89]:
```
## Note:- while seperating data types it was noticed that several of the categorical variable are stored as int.
## Hence these variables are to be converted to object data type.

## Create a list of variables which are incorrectly classified as numeric
convert_obj = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit']
```

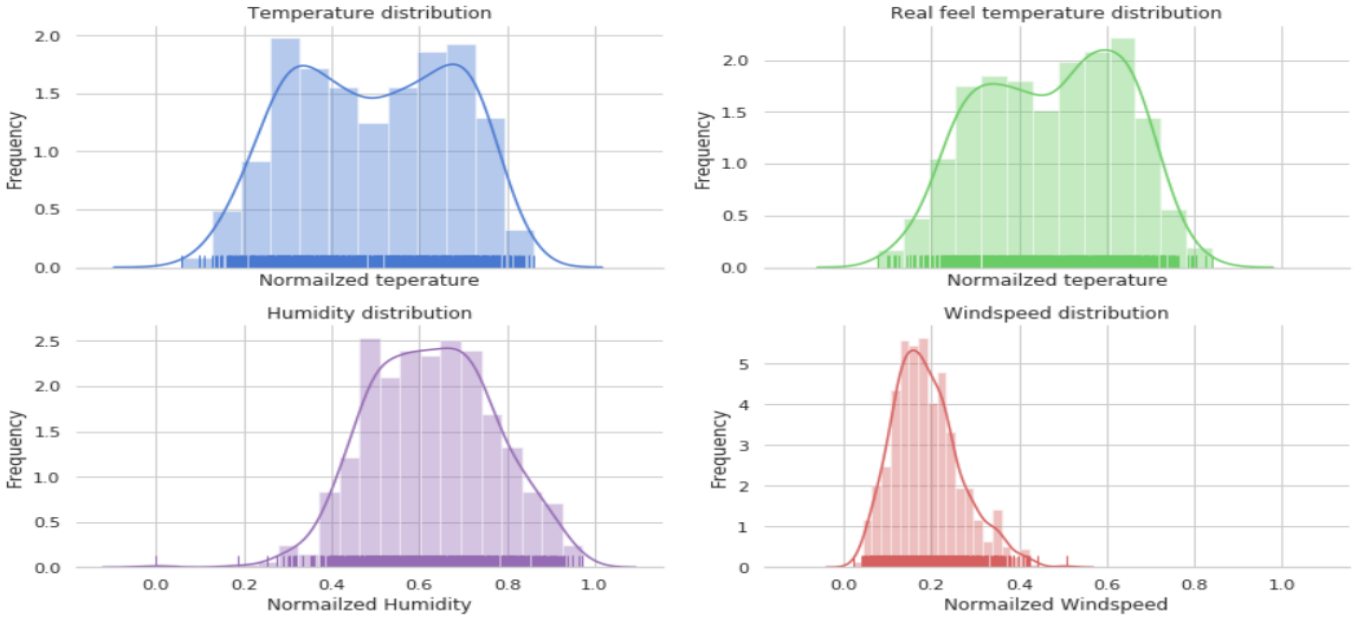
In [90]:
```
for i in convert_obj:
    df.loc[:,i] = df.loc[:,i].astype("object")

df.dtypes
```

Out[90]:
```
season         object
yr             object
mnth           object
holiday        object
weekday        object
workingday     object
weathersit     object
temp          float64
atemp         float64
hum           float64
windspeed     float64
cnt             int64
dtype: object
```

In [91]:
```
#Calling function 'seperate_datatypes' to segregate different data types
obj_dtype, num_dtype, bool_dtype, unknown_dtype = seperate_datatypes(df, df.columns)
```

Distribution Analysis Of Continous Variables

In [92]:
```
# Set up the matplotlib figure
sns.set(style="whitegrid", palette="muted", color_codes=True)
f, axes = plt.subplots(2, 2, figsize=(12, 7), sharex=True)
sns.despine(left=True)
# Plot a historgram and kernel density estimate
ax1 = sns.distplot(df['temp'], rug=True, color="b", ax=axes[0, 0])
ax1.set_title('Temperature distribution')
ax1.set_ylabel('Frequency')
ax1.set_xlabel('Normailzed teperature')
# Plot a historgram and kernel density estimate
ax2 = sns.distplot(df['atemp'], rug=True, color="g", ax=axes[0, 1])
ax2.set_title('Real feel temperature distribution')
ax2.set_ylabel('Frequency')
ax2.set_xlabel('Normailzed teperature')
# Plot a historgram and kernel density estimate
ax3 = sns.distplot(df['hum'], rug=True, color="m", ax=axes[1, 0])
ax3.set_title('Humidity distribution')
ax3.set_ylabel('Frequency')
ax3.set_xlabel('Normailzed Humidity')
# Plot a historgram and kernel density estimate
ax4 = sns.distplot(df['windspeed'], rug=True, color="r", ax=axes[1, 1])
ax4.set_title('Windspeed distribution')
ax4.set_ylabel('Frequency')
ax4.set_xlabel('Normailzed Windspeed')
plt.tight_layout()
plt.savefig('distribution_plot.pdf')
```



In [93]:
```
print("*****************************************************************")
for i in num_dtype:
    from scipy import stats
    skewness = stats.describe(df.loc[:,i])
    print("statistical properities of {0:5s}:".format(i))
    print(skewness)
    print("*****************************************************************")
```

```
*****************************************************************
statistical properities of temp :
DescribeResult(nobs=731, minmax=(0.0591304, 0.861667), mean=0.495384788508892, variance=0.03350766717740828, skewness=-0.0544
0902480571618, kurtosis=-1.1194225488473057)
*****************************************************************
statistical properities of atemp:
DescribeResult(nobs=731, minmax=(0.0790696, 0.8408959999999999), mean=0.47435398864569084, variance=0.026556345661055174, ske
wness=-0.1308188980737412, kurtosis=-0.9866019052943136)
*****************************************************************
statistical properities of hum  :
DescribeResult(nobs=731, minmax=(0.0, 0.9725), mean=0.6278940629274967, variance=0.02028604714193028, skewness=-0.06964015783
152368, kurtosis=-0.07228631791987006)
*****************************************************************
statistical properities of windspeed:
DescribeResult(nobs=731, minmax=(0.0223917, 0.507463), mean=0.190486211627907, variance=0.006005919960192755, skewness=0.6759
547264275362, kurtosis=0.39992023832685497)
*****************************************************************
statistical properities of cnt  :
DescribeResult(nobs=731, minmax=(22, 8714), mean=4504.3488372093025, variance=3752788.2082828926, skewness=-0.047255557553620
63, kurtosis=-0.8145762269613592)
```

Missing Values Analysis

In [95]:
```
## Missing values analysis
pd.DataFrame(df.isnull().sum())
```

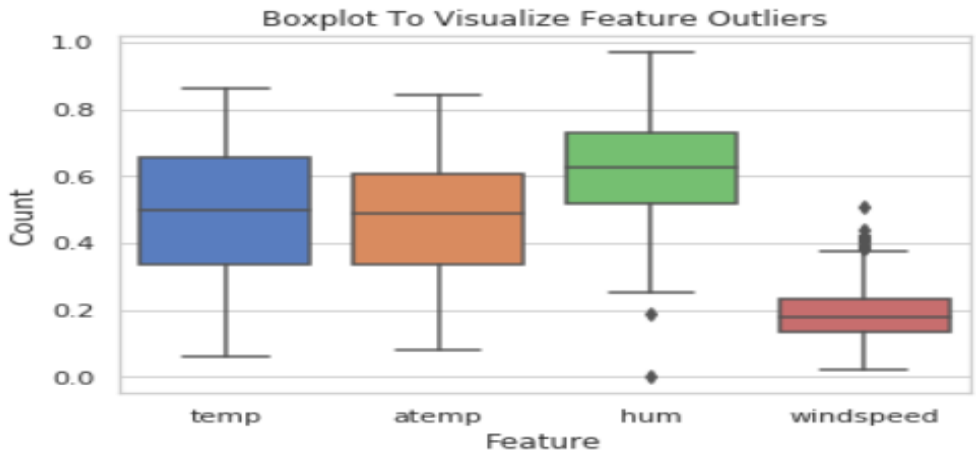
Out[95]:

	0
season	0
yr	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	0
cnt	0

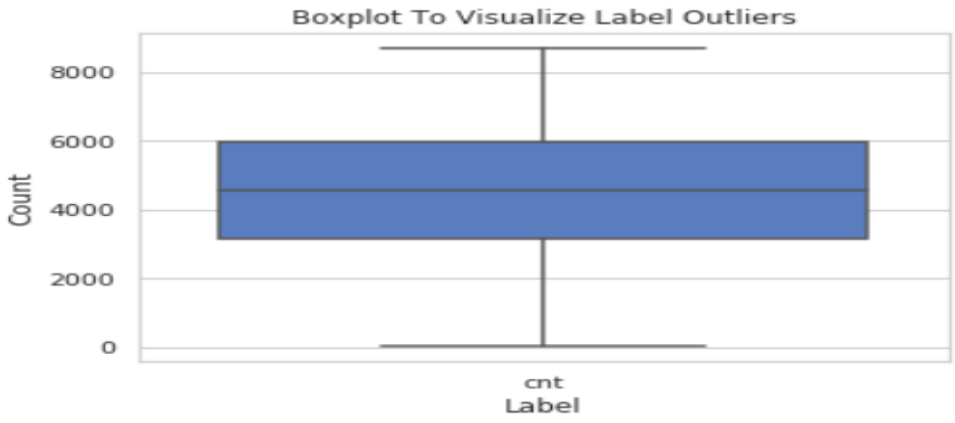
It is found that there is no missing values in the data set

Outlier Analysis

In [96]:
```
#Plot boxplot to visualize Feature variable Outliers
ax_o1 = sns.boxplot(data= df.iloc[:,7:11])
ax_o1.set_title('Boxplot To Visualize Feature Outliers')
ax_o1.set_ylabel('Count')
ax_o1.set_xlabel('Feature')
plt.savefig('feature_outlier1.pdf')
```



In [97]:
```
# #Plot boxplot to visualize Target variable Outliers
ax_o2 = sns.boxplot(data= df.iloc[:,11:12])
ax_o2.set_title('Boxplot To Visualize Label Outliers')
ax_o2.set_ylabel('Count')
ax_o2.set_xlabel('Label')
plt.savefig('label_outlier1.pdf')
```



In [98]:
```
#Calling the outlier function to compute and print outlier percent
percentage_of_outliers(df)
```

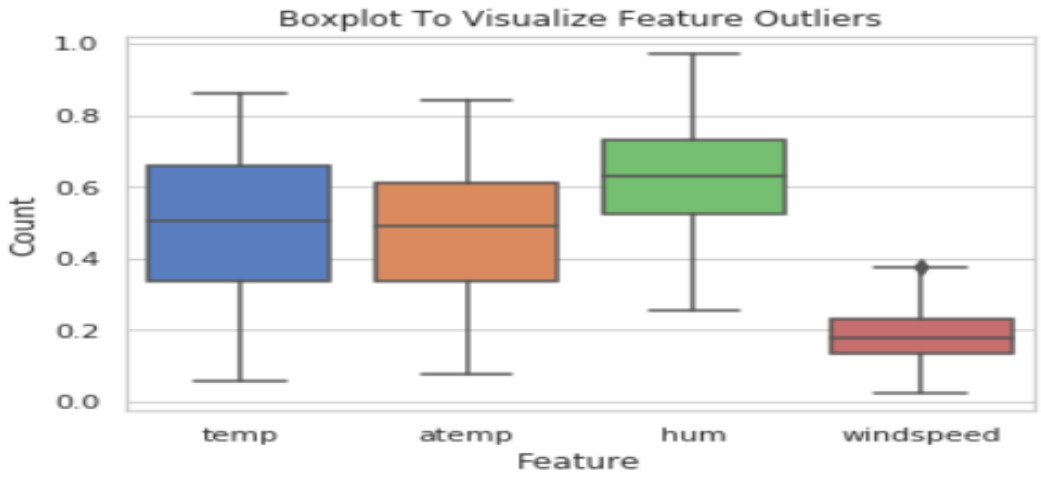
```
Total number of outliers present = 15
Outlier percentage : 2.052
```

In [99]:
```
#Calling the outlier function to perform outlier analysis
method = 0
df = analysis_of_outlier(df, num_dtype, method)
df.describe()
```

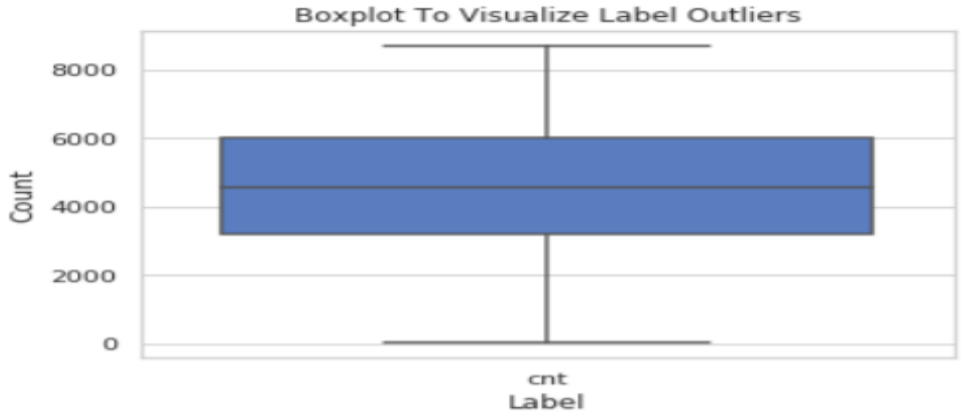
Out[99]:

	temp	atemp	hum	windspeed	cnt
count	717.000000	717.000000	717.000000	717.000000	717.000000
mean	0.497365	0.476252	0.631562	0.186287	4532.843794
std	0.183617	0.163155	0.139222	0.071786	1933.542429
min	0.059130	0.079070	0.254167	0.022392	22.000000
25%	0.337500	0.337939	0.524583	0.134329	3214.000000
50%	0.505833	0.491783	0.630833	0.178496	4570.000000
75%	0.656667	0.611121	0.732917	0.230721	6031.000000
max	0.861667	0.840896	0.972500	0.378108	8714.000000

In [100]:
```
# #Plot boxplot to visualize Feature variable Outliers
ax_o1 = sns.boxplot(data= df.iloc[:,7:11])
ax_o1.set_title('Boxplot To Visualize Feature Outliers')
ax_o1.set_ylabel('Count')
ax_o1.set_xlabel('Feature')
plt.savefig('feature_outlier2.pdf')
```

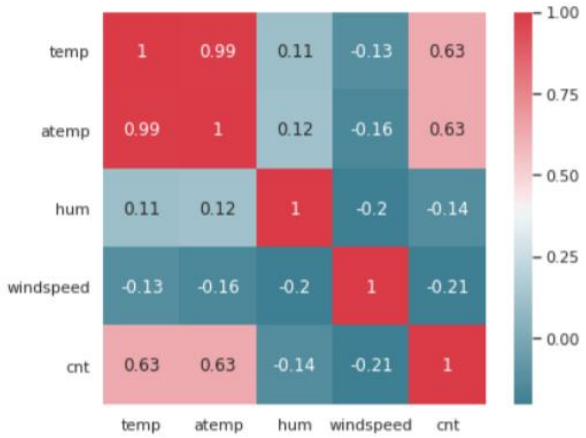


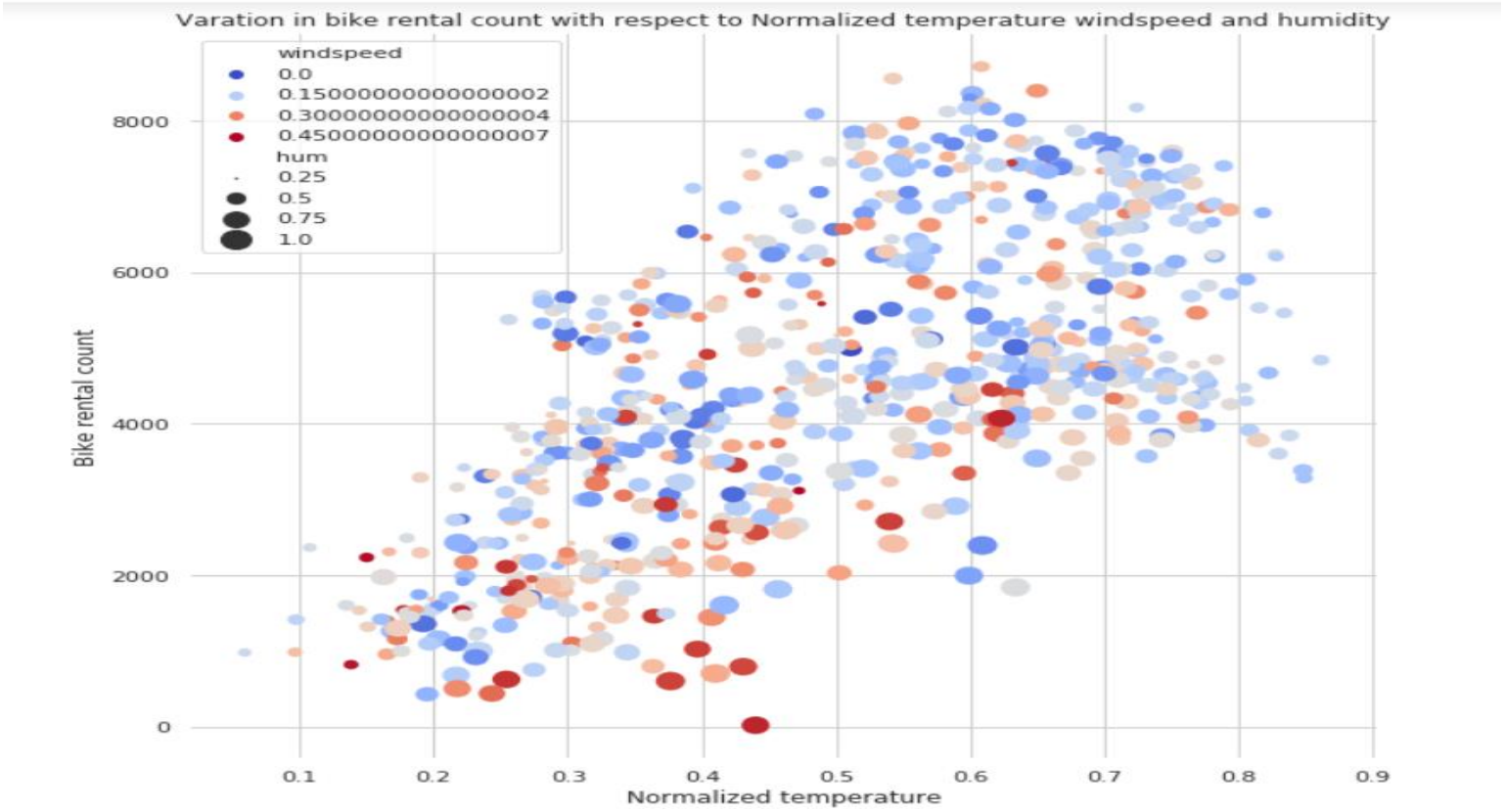
In [403]:
```
# #Plot boxplot to visualize Target variable Outliers
ax_o2 = sns.boxplot(data= df.iloc[:,11:12])
ax_o2.set_title('Boxplot To Visualize Label Outliers')
ax_o2.set_ylabel('Count')
ax_o2.set_xlabel('Label')
plt.savefig('label_outlier4.pdf')
```

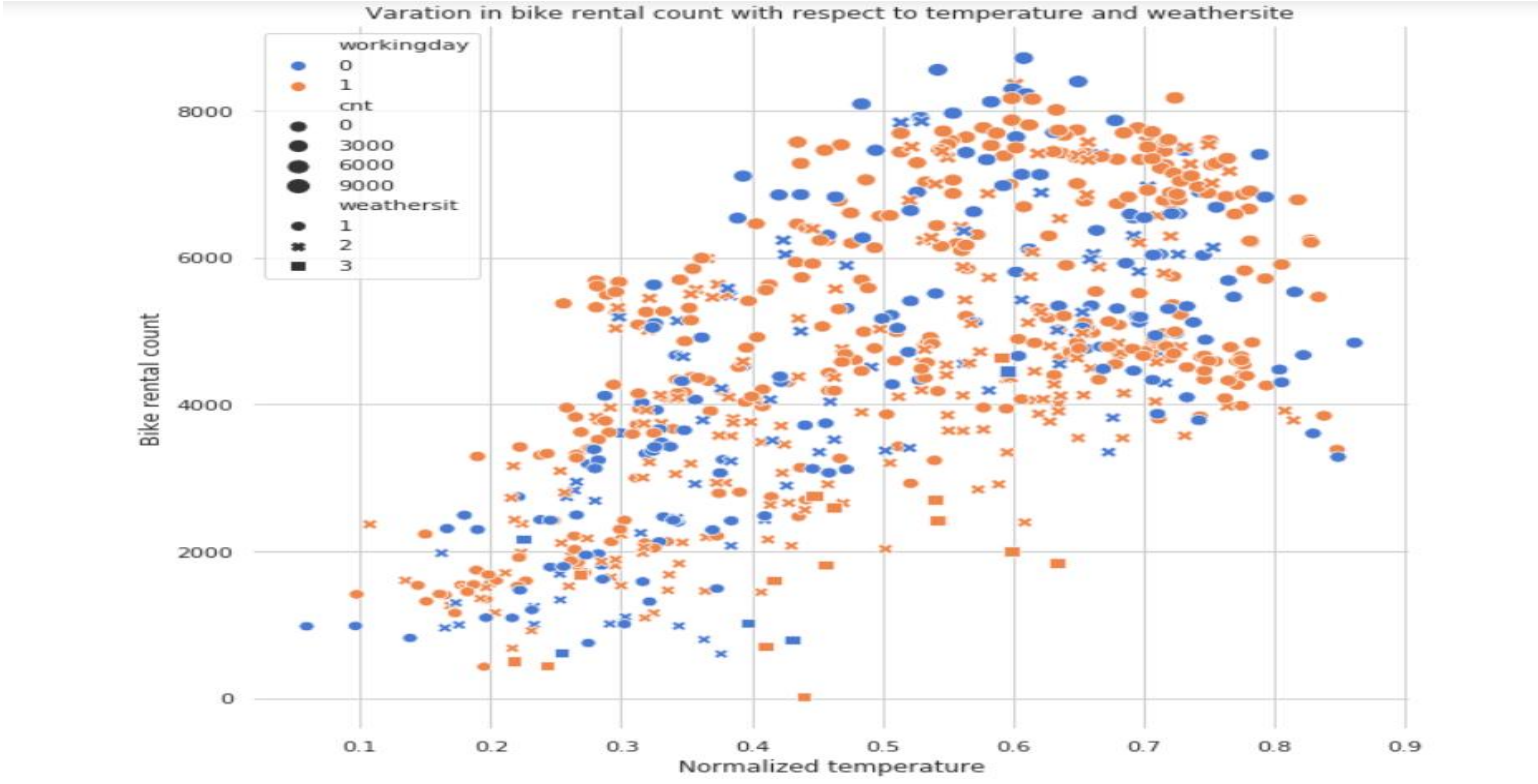


Correlation Analysis

In [409]:
```
##############################################################################
## Correlation plot
#Correlation plot
df_corr = df.loc[:,num_dtype]
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(7, 5))
#Generate correlation matrix
corr = df_corr.corr()
#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, annot=True,ax=ax)
plt.savefig('corr_plot.pdf')
```




Varation in bike rental count with respect to Normalized temperature windspeed and humidity
windspeed
0.0
0.15000000000000002
0.30000000000000004
0.45000000000000007
hum
0.25
0.5
0.75
1.0
Bike rental count
Normalized temperature


Varation in bike rental count with respect to temperature and weathersite
workingday
0
1
cnt
0
3000
6000
9000
weathersit
1
2
3
Bike rental count
Normalized temperature

ANOVA Test

Less significient variables are added to the drop_feat variable to be removed from the dataframe later.

In [412]:
```
label = 'cnt'
## ANOVA TEST FOR P VALUES
import statsmodels.api as sm
from statsmodels.formula.api import ols

anova_p = []
for  i in obj_dtype:
    buf = label + ' ~ ' + i
    mod = ols(buf,data=df).fit()
    anova_op = sm.stats.anova_lm(mod, typ=2)
    print(anova_op)
    anova_p.append(anova_op.iloc[0:1,3:4])
    p = anova_op.loc[i,'PR(>F)']
    if p >= 0.05:
        drop_feat.append(i)
```

```
                sum_sq     df           F        PR(>F)
season    9.305377e+08    3.0  127.350479  5.482108e-66
Residual  1.726865e+09  709.0         NaN           NaN
                sum_sq     df           F        PR(>F)
yr        8.887398e+08    1.0  357.27215  7.173240e-65
Residual  1.768663e+09  711.0        NaN           NaN
                sum_sq     df           F        PR(>F)
mnth      1.049040e+09   11.0  41.565535  3.197860e-69
Residual  1.608363e+09  701.0        NaN           NaN
                sum_sq     df         F    PR(>F)
holiday   1.415864e+07    1.0  3.808499  0.051385
Residual  2.643244e+09  711.0       NaN       NaN
                sum_sq     df         F    PR(>F)
weekday   1.970414e+07    6.0  0.878994  0.509776
Residual  2.637698e+09  706.0       NaN       NaN
                  sum_sq     df         F    PR(>F)
workingday  7.117269e+06    1.0  1.909371  0.167467
Residual    2.650285e+09  711.0       NaN       NaN
                sum_sq     df          F        PR(>F)
weathersit  2.672556e+08    2.0  39.694529  4.553787e-17
Residual    2.390147e+09  710.0        NaN           NaN
```

Multivariant Linear regression

[424]:
```
## Creating dummy variables for multivarient linear regression
X_l = pd.get_dummies(df, columns = obj_dtype, drop_first = True)
X_l["b0"]=1
col_name = X_l.columns.tolist()
col_name = col_name[21:22] + col_name[0:3]+ col_name[4:21]+ col_name[3:4]
X_l = X_l.loc[:, col_name]

#Divide data into train and test
train_stat, test_stat = train_test_split(X_l, test_size=0.2,random_state = 0)

# ## Linear Regression
#Import libraries for LR
import statsmodels.formula.api as sm
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
# Train the model using the training sets
model = sm.OLS(train_stat.iloc[:,21], train_stat.iloc[:,0:21]).fit()
```

In [148]:
```
# make the predictions by the model
predictions_LR = model.predict(test_stat.iloc[:,0:21])

r2 = r2_score(test_stat.iloc[:,21], predictions_LR)
mse = mean_squared_error(test_stat.iloc[:,21], predictions_LR)

#Calculate MAPE
mape = MAPE(test_stat.iloc[:,21], predictions_LR)

print('Linear Regression Model Performance:')
print('R-squared = {:0.2}.'.format(r2))
print('MSE = ',round(mse))
print('MAPE = {:0.4}%.'.format(mape))
```

```
Linear Regression Model Performance:
R-squared = 0.87.
MSE =  571508.0
MAPE = 18.44%.
```

```
Decision Regressor Model Performance:
Default Parameters =  {'criterion': 'mse', 'max_depth': None, 'max_features': None, 'max_leaf_nodes': None, 'min_impuri
ty_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf
': 0.0, 'presort': False, 'random_state': 0, 'splitter': 'best'}
R-squared = 0.75.
MSE =  1082039.0
MAPE = 26.7%.
****************************************
Random Search CV Decision Regressor Model Performance:
Best Parameters =  {'max_depth': 10}
R-squared = 0.76.
MSE =  1013272.0
MAPE = 25.95%.
****************************************
Grid Search CV Decision Regressor Model Performance:
Best Parameters =  {'max_depth': 5}
R-squared = 0.77.
MSE =  986838.0
MAPE = 25.07%.
****************************************
```

Random Search CV - Decision Tree

n [151]:
```
##Random Search CV
from sklearn.model_selection import RandomizedSearchCV
np.random.seed(0)
RDT = DecisionTreeRegressor(random_state = 0)
depth = list(range(5,50,5))
# Create the random grid
randDT_grid = {'max_depth': depth}

randomcv_DT = RandomizedSearchCV(RDT, param_distributions = randDT_grid, n_iter = 5, cv = 5, random_state=0)
randomcv_DT = randomcv_DT.fit(Xlr_train,y_train)

predictions_RDT = randomcv_DT.predict(Xlr_test)
predictions_RDT = np.array(predictions_RDT)

view_best_params_RDT = randomcv_DT.best_params_

best_model = randomcv_DT.best_estimator_

predictions_RDT = best_model.predict(Xlr_test)

#R^2
RDT_r2 = r2_score(y_test, predictions_RDT)
RDT_mse = mean_squared_error(y_test, predictions_RDT)

#Calculate MAPE
RDT_mape = MAPE(y_test, predictions_RDT)

print('Random Search CV Decision Regressor Model Performance:')
print('Best Parameters = ',view_best_params_RDT)
print('R-squared = {:0.2}.'.format(RDT_r2))
print('MSE = ',round(RDT_mse))
print('MAPE = {:0.4}%.'.format(RDT_mape))
print('****************************************')
```

```
Random Search CV Decision Regressor Model Performance:
Best Parameters =  {'max_depth': 10}
R-squared = 0.76.
MSE =  1040601.0
MAPE = 26.57%.
****************************************
```

Grid Search CV - Decision Tree

In [64]:

```
##Grid Search CV

from sklearn.model_selection import GridSearchCV
Gridregr = DecisionTreeRegressor(random_state = 0)
depth = list(range(1,20,2))

# Create the grid
grid_search = {'max_depth': depth}
## Grid Search Cross-Validation with 5 fold CV
gridcv_GDT = GridSearchCV(Gridregr, param_grid = grid_search, cv = 5)
gridcv_GDT = gridcv_GDT.fit(Xlr_train,y_train)
view_best_params_GDT = gridcv_GDT.best_params_

#Apply model on test data
predictions_GDT = gridcv_GDT.predict(Xlr_test)
GDT_r2 = r2_score(y_test, predictions_GDT)
GDT_mse = mean_squared_error(y_test, predictions_GDT)

#Calculate MAPE
GDT_mape = get_MAPE(y_test, predictions_GDT)

print('Grid Search CV Decision Regressor Model Performance:')
print('Best Parameters = ',view_best_params_GDT)
print('R-squared = {:0.2}.'.format(GDT_r2))
print('MSE = ',round(GDT_mse))
print('MAPE = {:0.4}%.'.format(GDT_mape))
print('****************************************')
```

```
Grid Search CV Decision Regressor Model Performance:
Best Parameters =  {'max_depth': 5}
R-squared = 0.79.
MSE =  894157.0
MAPE = 24.84%.
****************************************
```

Random Search CV - Random Forest

In [154]:
```
##Random Search CV
from sklearn.model_selection import RandomizedSearchCV

RRF = RandomForestRegressor(random_state = 0)
n_estimator = list(range(1,20,2))
depth = list(range(1,100,2))

# Create the random grid
rand_grid = {'n_estimators': n_estimator,
               'max_depth': depth}

randomcv_rf = RandomizedSearchCV(RRF, param_distributions = rand_grid, n_iter = 5, cv = 5, random_state=0)
randomcv_rf = randomcv_rf.fit(Xlr_train,y_train)
predictions_RRF = randomcv_rf.predict(Xlr_test)
predictions_RRF = np.array(predictions_RRF)

view_best_params_RRF = randomcv_rf.best_params_

best_model = randomcv_rf.best_estimator_

predictions_RRF = best_model.predict(Xlr_test)


#R^2
RRF_r2 = r2_score(y_test, predictions_RRF)
#Calculating MSE
RRF_mse = np.mean(( y_test - predictions_RRF)**2)
#Calculate MAPE
RRF_mape = MAPE(y_test, predictions_RRF)

print('Random Search CV Random Forest Regressor Model Performance:')
print('Best Parameters = ',view_best_params_RRF)
print('R-squared = {:0.2}.'.format(RRF_r2))
print('MSE = ',round(RRF_mse))
print('MAPE = {:0.4}%.'.format(RRF_mape))
print('**********************************************')
```

```
Random Search CV Random Forest Regressor Model Performance:
Best Parameters =  {'n_estimators': 15, 'max_depth': 23}
R-squared = 0.86.
MSE =  596208
MAPE = 20.51%.
**********************************************
```

Grid Search CV - Random Forest

163]:
```
## Grid Search CV
from sklearn.model_selection import GridSearchCV

regr = RandomForestRegressor(random_state = 0)
n_estimator = list(range(11,17,2))
depth = list(range(15,25,1))

# Create the grid
grid_search = {'n_estimators': n_estimator,
               'max_depth': depth}

## Grid Search Cross-Validation with 5 fold CV
gridcv_rf = GridSearchCV(regr, param_grid = grid_search, cv = 5)
gridcv_rf = gridcv_rf.fit(Xlr_train,y_train)
view_best_params_GRF = gridcv_rf.best_params_

#Apply model on test data
predictions_GRF = gridcv_rf.predict(Xlr_test)


#R^2
GRF_r2 = r2_score(y_test, predictions_GRF)
#Calculating MSE
GRF_mse = np.mean(( y_test - predictions_GRF)**2)
#Calculate MAPE
GRF_mape = MAPE(y_test, predictions_GRF)

print('Grid Search CV Random Forest Regressor Model Performance:')
print('Best Parameters = ',view_best_params_GRF)
print('R-squared = {:0.2}.'.format(GRF_r2))
print('MSE = ',round(GRF_mse))
print('MAPE = {:0.4}%.'.format(GRF_mape))
print('**********************************************')
```

```
Grid Search CV Random Forest Regressor Model Performance:
Best Parameters =  {'max_depth': 18, 'n_estimators': 15}
R-squared = 0.86.
MSE =  595450
MAPE = 20.57%.
**********************************************
```

Random Search CV - Random Forest

154]:
```
##Random Search CV
from sklearn.model_selection import RandomizedSearchCV

RRF = RandomForestRegressor(random_state = 0)
n_estimator = list(range(1,20,2))
depth = list(range(1,100,2))

# Create the random grid
rand_grid = {'n_estimators': n_estimator,
               'max_depth': depth}

randomcv_rf = RandomizedSearchCV(RRF, param_distributions = rand_grid, n_iter = 5, cv = 5, random_state=0)
randomcv_rf = randomcv_rf.fit(Xlr_train,y_train)
predictions_RRF = randomcv_rf.predict(Xlr_test)
predictions_RRF = np.array(predictions_RRF)

view_best_params_RRF = randomcv_rf.best_params_

best_model = randomcv_rf.best_estimator_

predictions_RRF = best_model.predict(Xlr_test)


#R^2
RRF_r2 = r2_score(y_test, predictions_RRF)
#Calculating MSE
RRF_mse = np.mean(( y_test - predictions_RRF)**2)
#Calculate MAPE
RRF_mape = MAPE(y_test, predictions_RRF)

print('Random Search CV Random Forest Regressor Model Performance:')
print('Best Parameters = ',view_best_params_RRF)
print('R-squared = {:0.2}.'.format(RRF_r2))
print('MSE = ',round(RRF_mse))
print('MAPE = {:0.4}%.'.format(RRF_mape))
print('**********************************************')
```

```
Random Search CV Random Forest Regressor Model Performance:
Best Parameters =  {'n_estimators': 15, 'max_depth': 23}
R-squared = 0.86.
MSE =  596208
MAPE = 20.51%.
**********************************************
```

Gradient Boosting Machine

In [156]:
```
#Gradient Boost
from sklearn.ensemble import GradientBoostingRegressor

gbt = GradientBoostingRegressor(random_state= 0).fit(Xlr_train,y_train)

predictions_gbt = gbt.predict(Xlr_test)


gbt.get_params()

#R^2
GBR_r2 = r2_score(y_test, predictions_gbt)
#Calculate MSE
GBR_mse = mean_squared_error(y_test, predictions_gbt)

#Calculate MAPE
GBR_mape = MAPE(y_test, predictions_gbt)

print('Gradient Boosting Regressor Model Performance:')
print('Default Parameters = ',gbt.get_params())
print('R-squared = {:0.2}.'.format(GBR_r2))
print('MSE = ',round(GBR_mse))
print('MAPE = {:0.4}%.'.format(GBR_mape))
print('************************************************')
```

```
Gradient Boosting Regressor Model Performance:
Default Parameters =  {'alpha': 0.9, 'criterion': 'friedman_mse', 'init': None, 'learning_rate': 0.1, 'loss': 'ls', 'ma
x_depth': 3, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'm
in_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 100, 'presort': 'auto', '
random_state': 0, 'subsample': 1.0, 'verbose': 0, 'warm_start': False}
R-squared = 0.88.
MSE =  514160.0
MAPE = 18.51%.
************************************************
```

Random Search CV - Gradient Boosting

In [157]:
```
##Random Search CV
rGBR = GradientBoostingRegressor(random_state = 0)
#loss = ['ls','lad','huber','quantile']
n_estimator = list(range(50,150,10))
#max_feat = ['auto','sqrt','log2']
depth = list(range(1,10,2))

# Create the random grid
rand_GBT = {#'loss': loss,
            'n_estimators': n_estimator,
            #'max_features': max_feat,
            'max_depth': depth}

randomcv_gbt = RandomizedSearchCV(rGBR, param_distributions = rand_GBT, n_iter = 5, cv = 5, random_state=0)
randomcv_gbt = randomcv_gbt.fit(Xlr_train,y_train)
predictions_GBT = randomcv_gbt.predict(Xlr_test)


view_best_params_GBT = randomcv_gbt.best_params_

#R^2
rGBR_r2 = r2_score(y_test, predictions_GBT)
#Calculate MSE
rGBR_mse = mean_squared_error(y_test, predictions_GBT)

#Calculate MAPE
rGBR_mape = MAPE(y_test, predictions_GBT)

print('Random Search CV Gradient Boosting Regressor Model Performance:')
print('Best Parameters = ',view_best_params_GBT)
print('R-squared = {:0.2}.'.format(rGBR_r2))
print('MSE = ',round(rGBR_mse))
print('MAPE = {:0.4}%.'.format(rGBR_mape))
print('**************************************************')
##########################################################################################################################
```

```
Random Search CV Gradient Boosting Regressor Model Performance:
Best Parameters =  {'n_estimators': 60, 'max_depth': 3}
R-squared = 0.88.
MSE =  522381.0
MAPE = 18.41%.
**************************************************
```

Gride Search CV - Gradient Boosting

In [158]:
```
## Gride Search CV

gGBR = GradientBoostingRegressor(random_state=0)
#loss = ['ls','lad','huber','quantile']
n_estimator = list(range(40,80,5))
#max_feat = ['auto','sqrt','log2']
depth = list(range(1,5,1))

# Create the random grid
grid_GBT = {#'loss': loss,
            'n_estimators': n_estimator,
            #'max_features': max_feat,
            'max_depth': depth}

## Grid Search Cross-Validation with 5 fold CV
gridcv_GBT = GridSearchCV(gGBR, param_grid = grid_GBT, cv = 5)
gridcv_GBT = gridcv_GBT.fit(Xlr_train,y_train)
view_best_params_gridGRF = gridcv_GBT.best_params_

#Apply model on test data
predictions_gridGBT = gridcv_GBT.predict(Xlr_test)

#R^2
gGBR_r2 = r2_score(y_test, predictions_gridGBT)
#Calculate MSE
gGBR_mse = mean_squared_error(y_test, predictions_gridGBT)

#Calculate MAPE
gGBR_mape = MAPE(y_test, predictions_gridGBT)

print('Grid Search CV Gradient Boosting Regressor Model Performance:')
print('Best Parameters = ',view_best_params_gridGRF)
print('R-squared = {:0.2}.'.format(gGBR_r2))
print('MSE = ',round(gGBR_mse))
print('MAPE = {:0.4}%.'.format(gGBR_mape))
print('**************************************************')
```

```
Grid Search CV Gradient Boosting Regressor Model Performance:
Best Parameters =  {'max_depth': 3, 'n_estimators': 75}
R-squared = 0.88.
MSE =  514395.0
MAPE = 18.06%.
**************************************************
```

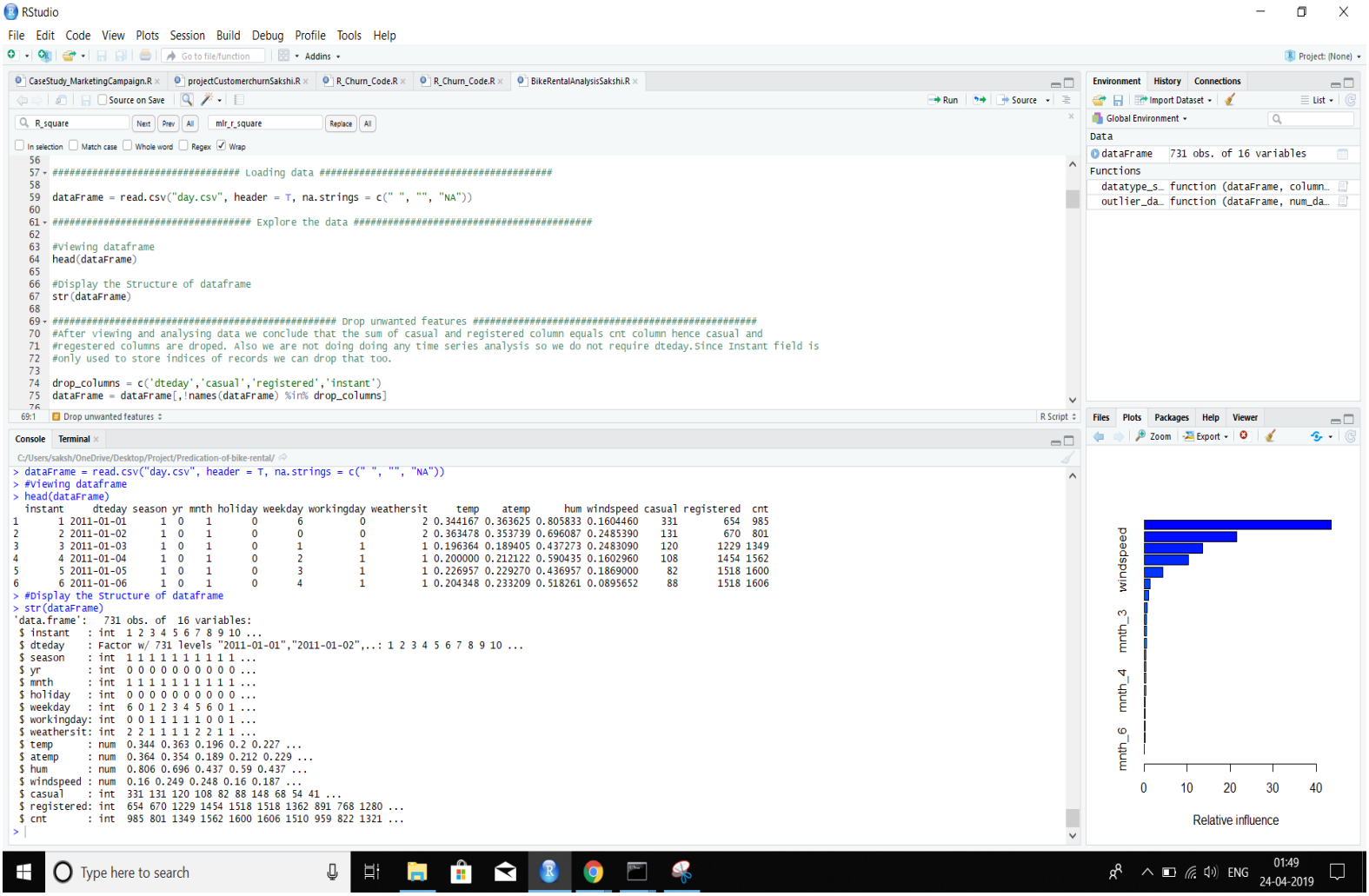
Extra Graident Boosting Machine

In [159]:
```
# Extra Graident Boosting Machine
import xgboost as xgb

xgb_reg = xgb.XGBRegressor(random_state = 0)
xgb_reg.fit(Xlr_train,y_train)


predictions_xgb = xgb_reg.predict(Xlr_test)

#R^2
XBT_r2 = r2_score(y_test, predictions_xgb)
#Calculate MSE
XBT_mse = mean_squared_error(y_test, predictions_xgb)

#Calculate MAPE
XBT_mape = MAPE(y_test, predictions_xgb)

print('XG Boosting Regressor Model Performance:')
print('XG Boosting parameters = ',xgb_reg.get_params())
print('R-squared = {:0.2}.'.format(XBT_r2))
print('MSE = ',round(XBT_mse))
print('MAPE = {:0.4}%.'.format(XBT_mape))
##########################################################################################################################
```

```
XG Boosting Regressor Model Performance:
XG Boosting parameters =  {'base_score': 0.5, 'booster': 'gbtree', 'colsample_bylevel': 1, 'colsample_bytree': 1, 'gamm
a': 0, 'learning_rate': 0.1, 'max_delta_step': 0, 'max_depth': 3, 'min_child_weight': 1, 'missing': None, 'n_estimators
': 100, 'n_jobs': 1, 'nthread': None, 'objective': 'reg:linear', 'random_state': 0, 'reg_alpha': 0, 'reg_lambda': 1, 's
cale_pos_weight': 1, 'seed': None, 'silent': True, 'subsample': 1}
R-squared = 0.88.
MSE =  499137.0
MAPE = 18.25%.
```

Appendix B

R program and its corresponding output:



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R

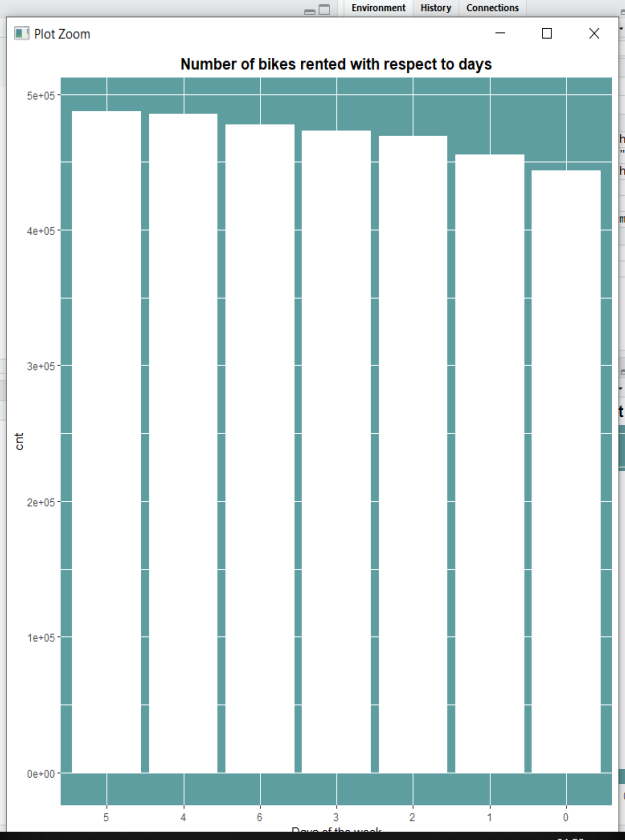
```
104 int_datatype = datatype[[3]]
105 int_datatype = unlist(int_datatype, recursive=FALSE)
106
107
108
109 ##########################################################################################################
110 # Data Visualisation and Data Understanding
111 ##########################################################################################################
112
113 #Plotting no of bikes rented with respect to the days of the week.
114 ggplot(data = dataFrame, aes(x = reorder(weekday,-cnt), y = cnt))+
115   geom_bar(stat = "identity",fill = "white")+
116   labs(title = "Number of bikes rented with respect to days", x = "Days of the week")+
117   theme(panel.background = element_rect("cadetblue"))+
118   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
119
120 #Plotting Bikes rented with respect to variation in temperature and hunidity
121 ggplot(dataFrame,aes(temp,cnt)) +
122   geom_point(aes(color=hum),alpha=0.5) +
123   labs(title = "Bikes rented with respect to variation in temperature and hunidity", x = "Normalized temperature")+
```

Console Terminal

C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

```
> dataFrame = dataFrame[,!names(dataFrame) %in% drop_columns]
> # Calling function to seperate by datatype
> datatype = datatype_separator(dataFrame,colnames(dataFrame))
> num_datatype = datatype[[2]]
> num_datatype = unlist(num_datatype, recursive=FALSE)
> int_datatype = datatype[[3]]
> int_datatype = unlist(int_datatype, recursive=FALSE)
> convert_obj = c('season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit')
> for(i in 1:length(int_datatype)){
+   if(int_datatype[i] %in% convert_obj){
+     dataFrame[,int_datatype[i]] = factor(dataFrame[,int_datatype[i]])
+   }
+ }
> # Convert target variable to numeric data type
> dataFrame[,'cnt'] = as.numeric(dataFrame[,'cnt'])
> # Calling function to seperate by datatype
> datatype = datatype_separator(dataFrame,colnames(dataFrame))
> fact_datatype = datatype[[1]]
> fact_datatype = unlist(fact_datatype, recursive=FALSE)
> num_datatype = datatype[[2]]
> num_datatype = unlist(num_datatype, recursive=FALSE)
> int_datatype = datatype[[3]]
> int_datatype = unlist(int_datatype, recursive=FALSE)
> #Plotting no of bikes rented with respect to the days of the week.
> ggplot(data = dataFrame, aes(x = reorder(weekday,-cnt), y = cnt))+
+   geom_bar(stat = "identity",fill = "white")+
+   labs(title = "Number of bikes rented with respect to days", x = "Days of the week")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
>
```

Plot Zoom



Type here to search

ENG 01:52 24-04-2019

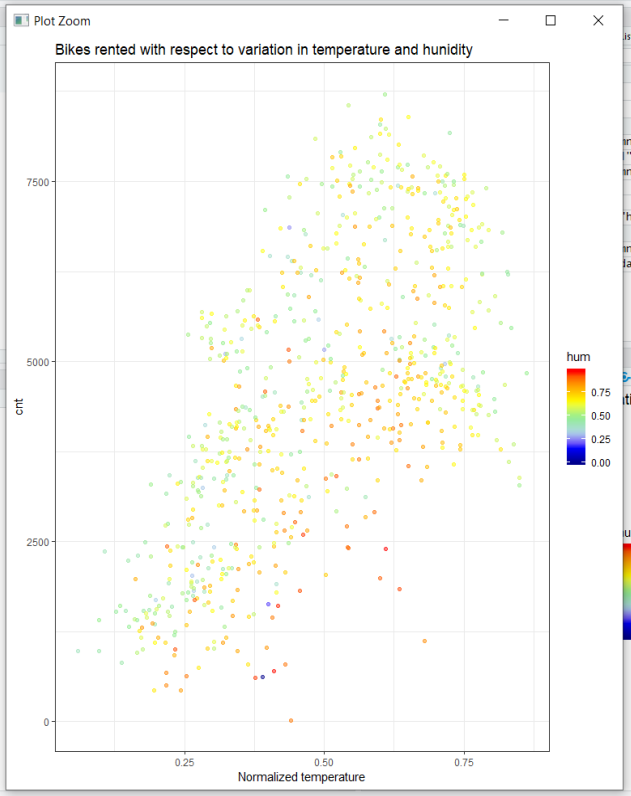
RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R

```
113 #Plotting no of bikes rented with respect to the days of the week.
114 ggplot(data = dataFrame, aes(x = reorder(weekday,-cnt), y = cnt))+
115   geom_bar(stat = "identity",fill = "white")+
116   labs(title = "Number of bikes rented with respect to days", x = "Days of the week")+
117   theme(panel.background = element_rect("cadetblue"))+
118   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
119
120 #Plotting Bikes rented with respect to variation in temperature and hunidity
121 ggplot(dataFrame,aes(temp,cnt)) +
122   geom_point(aes(color=hum),alpha=0.5) +
123   labs(title = "Bikes rented with respect to variation in temperature and hunidity", x = "Normalized temperature")+
124   scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red')) +
125   theme_bw()
126
127 #Plotting Bikes rented with respect to temperature and weathersite
128 ggplot(data = dataFrame, aes(x = temp, y = cnt))+
129   geom_point(aes(color=weathersit))+
130   labs(title = "Bikes rented with respect to temperature and weathersite", x = "Normalized temperature")+
```

```
> int_datatype = unlist(int_datatype, recursive=FALSE)
> convert_obj = c('season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit')
> for(i in 1:length(int_datatype)){
+   if(int_datatype[i] %in% convert_obj){
+     dataFrame[,int_datatype[i]] = factor(dataFrame[,int_datatype[i]])
+   }
+ }
> # Convert target variable to numeric data type
> dataFrame[,'cnt'] = as.numeric(dataFrame[,'cnt'])
> # Calling function to seperate by datatype
> datatype = datatype_separator(dataFrame,colnames(dataFrame))
> fact_datatype = datatype[[1]]
> fact_datatype = unlist(fact_datatype, recursive=FALSE)
> num_datatype = datatype[[2]]
> num_datatype = unlist(num_datatype, recursive=FALSE)
> int_datatype = datatype[[3]]
> int_datatype = unlist(int_datatype, recursive=FALSE)
> #Plotting no of bikes rented with respect to the days of the week.
> ggplot(data = dataFrame, aes(x = reorder(weekday,-cnt), y = cnt))+
+   geom_bar(stat = "identity",fill = "white")+
+   labs(title = "Number of bikes rented with respect to days", x = "Days of the week")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
> #Plotting Bikes rented with respect to variation in temperature and hunidity
> ggplot(dataFrame,aes(temp,cnt)) +
+   geom_point(aes(color=hum),alpha=0.5) +
+   labs(title = "Bikes rented with respect to variation in temperature and hunidity", x = "Normalized temperature")+
+   scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red')) +
+   theme_bw()
>
```



Type here to search

ENG 01:53 24-04-2019

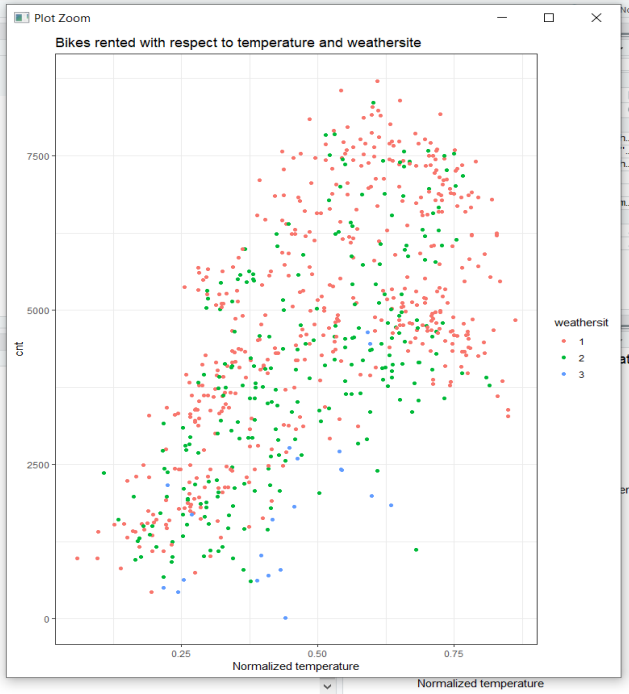
RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R

```
118   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
119
120 #Plotting Bikes rented with respect to variation in temperature and hunidity
121 ggplot(dataFrame,aes(temp,cnt)) +
122   geom_point(aes(color=hum),alpha=0.5) +
123   labs(title = "Bikes rented with respect to variation in temperature and hunidity", x = "Normalized temperature")+
124   scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red')) +
125   theme_bw()
126
127 #Plotting Bikes rented with respect to temperature and weathersite
128 ggplot(data = dataFrame, aes(x = temp, y = cnt))+
129   geom_point(aes(color=weathersit))+
130   labs(title = "Bikes rented with respect to temperature and weathersite", x = "Normalized temperature")+
131   theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
132   theme_bw()
133
134 #PLotting Bikes rented with respect to temperature and workingday
135 ggplot(data = dataFrame, aes(x = temp, y = cnt))+
136   geom_point(aes(color=workingday))+
137   labs(title = "Bikes rented with respect to temperature and workingday", x = "Normalized temperature")+
```

```
+ }
> # Convert target variable to numeric data type
> dataFrame[,'cnt'] = as.numeric(dataFrame[,'cnt'])
> # Calling function to seperate by datatype
> datatype = datatype_separator(dataFrame,colnames(dataFrame))
> fact_datatype = datatype[[1]]
> fact_datatype = unlist(fact_datatype, recursive=FALSE)
> num_datatype = datatype[[2]]
> num_datatype = unlist(num_datatype, recursive=FALSE)
> int_datatype = datatype[[3]]
> int_datatype = unlist(int_datatype, recursive=FALSE)
> #Plotting no of bikes rented with respect to the days of the week.
> ggplot(data = dataFrame, aes(x = reorder(weekday,-cnt), y = cnt))+
+   geom_bar(stat = "identity",fill = "white")+
+   labs(title = "Number of bikes rented with respect to days", x = "Days of the week")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
> #Plotting Bikes rented with respect to variation in temperature and hunidity
> ggplot(dataFrame,aes(temp,cnt)) +
+   geom_point(aes(color=hum),alpha=0.5) +
+   labs(title = "Bikes rented with respect to variation in temperature and hunidity", x = "Normalized temperature")+
+   scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red')) +
+   theme_bw()
> #Plotting Bikes rented with respect to temperature and weathersite
> ggplot(data = dataFrame, aes(x = temp, y = cnt))+
+   geom_point(aes(color=weathersit))+
+   labs(title = "Bikes rented with respect to temperature and weathersite", x = "Normalized temperature")+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
+   theme_bw()
>
```



Type here to search

ENG 01:53 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R

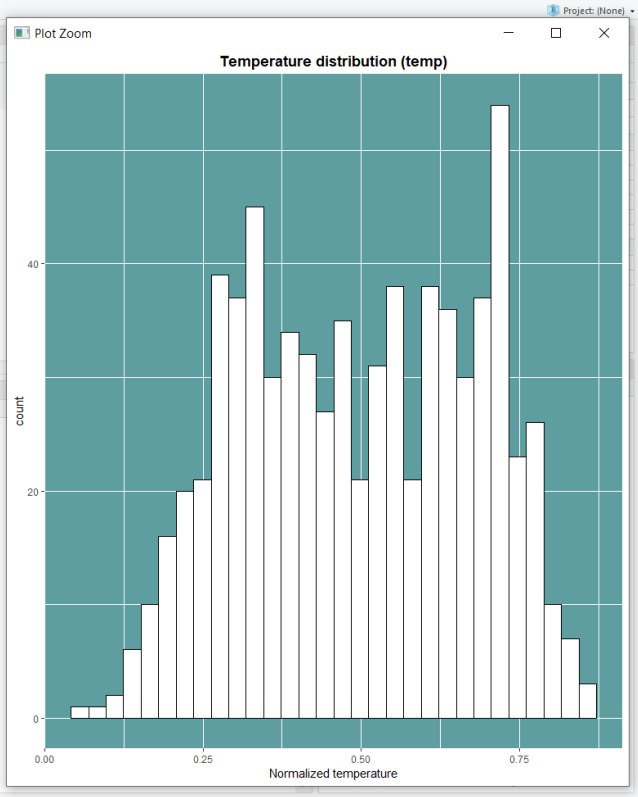
```
  labs(title = "Bikes rented with respect to temperature and workingday", x = "Normalized temperature")+
  #  theme(panel.background = element_rect("white"))+
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
  theme_bw()

#####################################################################
# Distribution Plot
#####################################################################

#Plot Temperature distribution (temp)
ggplot(data = dataFrame, aes(x = temp))+
  geom_histogram(bins = 30, fill = "white", col = "black")+
  labs(title = "Temperature distribution (temp)", x = "Normalized temperature")+
  theme(panel.background = element_rect("cadetblue"))+
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

#PLot Real feel temperature distribution (atemp)
ggplot(data = dataFrame, aes(x = atemp))+
  geom_histogram(bins = 30, fill = "white", col = "black")+
  labs(title = "Real feel temperature distribution (atemp)", x = "Normalized temperature")+
```

Console Terminal

C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

```
+   geom_bar(stat = "identity",fill = "white")+
+   labs(title = "Number of bikes rented with respect to days", x = "Days of the week")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
> #Plotting Bikes rented with respect to variation in temperature and hunidity
> ggplot(dataFrame,aes(temp,cnt)) +
+   geom_point(aes(color=hum),alpha=0.5) +
+   labs(title = "Bikes rented with respect to variation in temperature and hunidity", x = "Normalized temperature")+
+   scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red')) +
+   theme_bw()
> #Plotting Bikes rented with respect to temperature and weathersite
> ggplot(data = dataFrame, aes(x = temp, y = cnt))+
+   geom_point(aes(color=weathersit))+
+   labs(title = "Bikes rented with respect to temperature and weathersite", x = "Normalized temperature")+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
+   theme_bw()
> #Plotting Bikes rented with respect to temperature and workingday
> ggplot(data = dataFrame, aes(x = temp, y = cnt))+
+   geom_point(aes(color=workingday))+
+   labs(title = "Bikes rented with respect to temperature and workingday", x = "Normalized temperature")+
+   #  theme(panel.background = element_rect("white"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
+   theme_bw()
> #Plot Temperature distribution (temp)
> ggplot(data = dataFrame, aes(x = temp))+
+   geom_histogram(bins = 30, fill = "white", col = "black")+
+   labs(title = "Temperature distribution (temp)", x = "Normalized temperature")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
>
```

Plot Zoom



Type here to search

ENG 01:54 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R

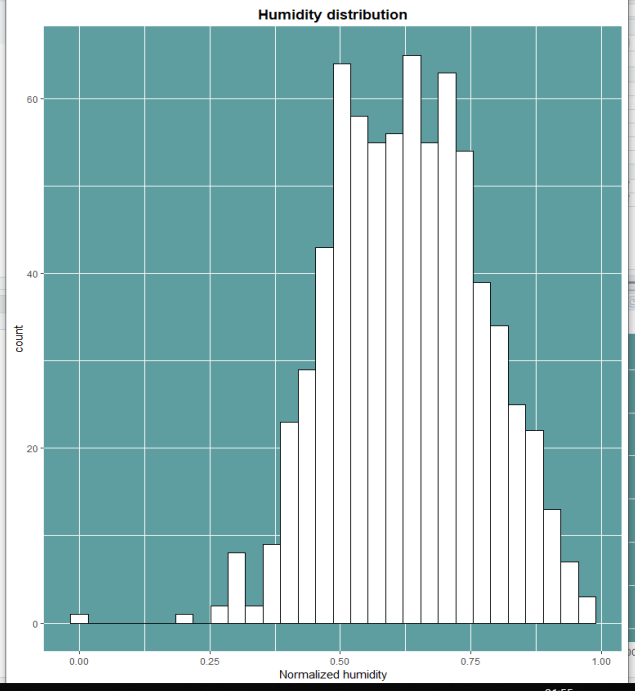
```
151   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
152
153 #PLot Real feel temperature distribution (atemp)
154 ggplot(data = dataFrame, aes(x = atemp))+
155   geom_histogram(bins = 30, fill = "white", col = "black")+
156   labs(title = "Real feel temperature distribution (atemp)", x = "Normalized temperature")+
157   theme(panel.background = element_rect("cadetblue"))+
158   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
159
160 #Plot Humidity distribution
161 ggplot(data = dataFrame, aes(x = hum))+
162   geom_histogram(bins = 30, fill = "white", col = "black")+
163   labs(title = "Humidity distribution", x = "Normalized humidity")+
164   theme(panel.background = element_rect("cadetblue"))+
165   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
166
167 #Plot Windspeed distribution
168 ggplot(data = dataFrame, aes(x = windspeed))+
169   geom_histogram(bins = 30, fill = "white", col = "black")+
170   labs(title = "Windspeed distribution", x = "Features")+
```

Console | Terminal

C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

```
+   geom_point(aes(color=weathersit))+
+   labs(title = "Bikes rented with respect to temperature and weathersite", x = "Normalized temperature")+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
+   theme_bw()
> #PLotting Bikes rented with respect to temperature and workingday
> ggplot(data = dataFrame, aes(x = temp, y = cnt))+
+   geom_point(aes(color=workingday))+
+   labs(title = "Bikes rented with respect to temperature and workingday", x = "Normalized temperature")+
+   #  theme(panel.background = element_rect("white"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
+   theme_bw()
> #Plot Temperature distribution (temp)
> ggplot(data = dataFrame, aes(x = temp))+
+   geom_histogram(bins = 30, fill = "white", col = "black")+
+   labs(title = "Temperature distribution (temp)", x = "Normalized temperature")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
> #PLot Real feel temperature distribution (atemp)
> ggplot(data = dataFrame, aes(x = atemp))+
+   geom_histogram(bins = 30, fill = "white", col = "black")+
+   labs(title = "Real feel temperature distribution (atemp)", x = "Normalized temperature")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
> #Plot Humidity distribution
> ggplot(data = dataFrame, aes(x = hum))+
+   geom_histogram(bins = 30, fill = "white", col = "black")+
+   labs(title = "Humidity distribution", x = "Normalized humidity")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
>
```

Plot Zoom



Type here to search

ENG 01:55 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R

```
158   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
159
160 #Plot Humidity distribution
161 ggplot(data = dataFrame, aes(x = hum))+
162   geom_histogram(bins = 30, fill = "white", col = "black")+
163   labs(title = "Humidity distribution", x = "Normalized humidity")+
164   theme(panel.background = element_rect("cadetblue"))+
165   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
166
167 #Plot windspeed distribution
168 ggplot(data = dataFrame, aes(x = windspeed))+
169   geom_histogram(bins = 30, fill = "white", col = "black")+
170   labs(title = "Windspeed distribution", x = "Features")+
171   theme(panel.background = element_rect("cadetblue"))+
172   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
173
174 #Distribution of unique values in a Feature
175 for(i in 1:length(fact_datatype)){
176   fact = fact_datatype[i]
177   print(fact)
```

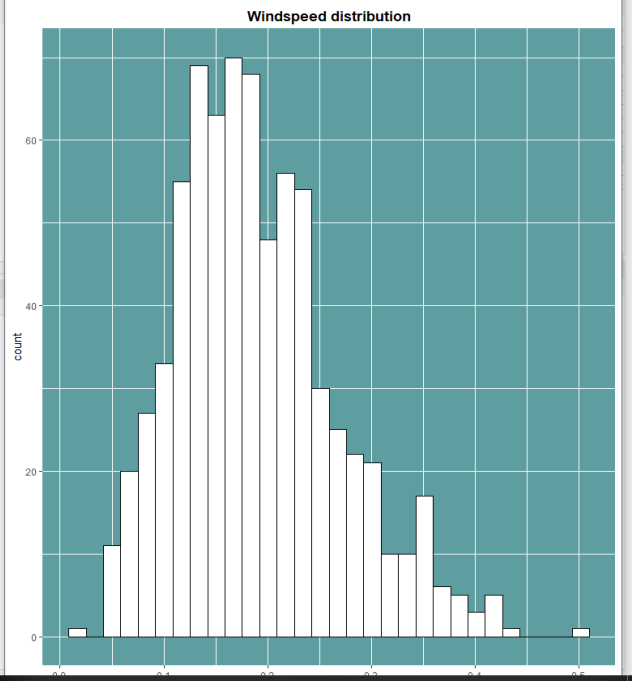
Console Terminal

C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

```
+   geom_point(aes(color=workingday))+
+   labs(title = "Bikes rented with respect to temperature and workingday", x = "Normalized temperature")+
+   #  theme(panel.background = element_rect("white"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))+
+   theme_bw()
> #Plot Temperature distribution (temp)
> ggplot(data = dataFrame, aes(x = temp))+
+   geom_histogram(bins = 30, fill = "white", col = "black")+
+   labs(title = "Temperature distribution (temp)", x = "Normalized temperature")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
> #Plot Real feel temperature distribution (atemp)
> ggplot(data = dataFrame, aes(x = atemp))+
+   geom_histogram(bins = 30, fill = "white", col = "black")+
+   labs(title = "Real feel temperature distribution (atemp)", x = "Normalized temperature")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
> #Plot Humidity distribution
> ggplot(data = dataFrame, aes(x = hum))+
+   geom_histogram(bins = 30, fill = "white", col = "black")+
+   labs(title = "Humidity distribution", x = "Normalized humidity")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
> #Plot Windspeed distribution
> ggplot(data = dataFrame, aes(x = windspeed))+
+   geom_histogram(bins = 30, fill = "white", col = "black")+
+   labs(title = "Windspeed distribution", x = "Features")+
+   theme(panel.background = element_rect("cadetblue"))+
+   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
>
```

Environment History Connections

Plot Zoom



Type here to search

ENG 01:56 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R

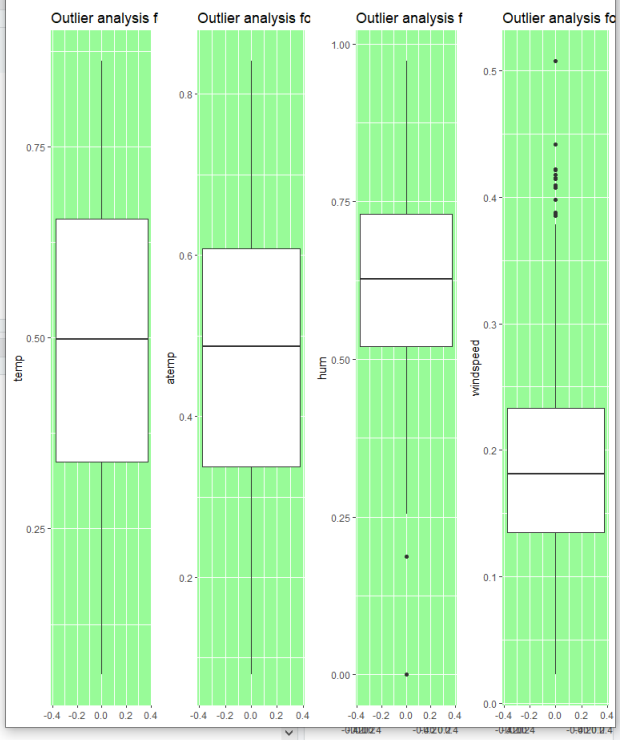
```
print(missing_val)
############################################################################################################
#### Outlier Analysis
############################################################################################################
#Plot outliers present in continous data set
for (i in 1:length(num_datatype))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (num_datatype[i])), data = subset(dataFrame))+ geom_boxplot()+
           labs(y=num_datatype[i])+
           theme(panel.background = element_rect(fill = "palegreen"))+
           ggtitle(paste("Outlier analysis for ",num_datatype[i])))
}
## Plotting plots together
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,ncol=4)


outlier_analysis = outlier_dataFrame(dataFrame,num_datatype)

ggplot(data = outlier_analysis, aes(x = reorder(feature,-total_outliers), y = total_outliers))+
  geom_bar(stat = "identity",fill = "white")+
```

Console Terminal

C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

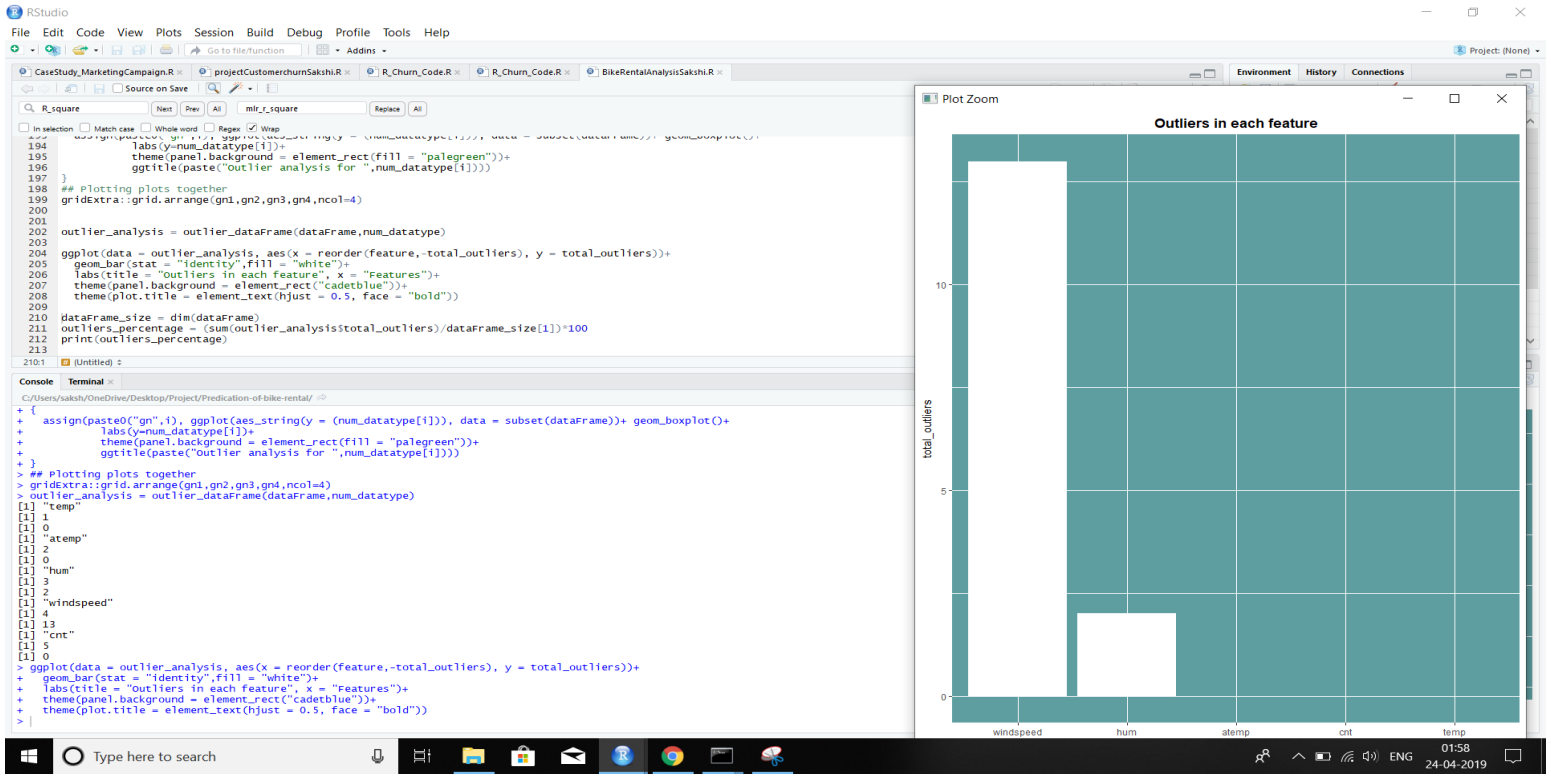
```
  0   1   2   3   4   5   6
105 105 104 104 104 104 105
[1] "workingday"

  0   1
231 500
[1] "weathersit"

  1   2   3
463 247  21
> ############################################################################################################
> # Missing Values Analysis
> ############################################################################################################
> missing_val = sum(is.na(dataFrame))
> print(missing_val)
[1] 0
> ############################################################################################################
> #### Outlier Analysis
> ############################################################################################################
> #Plot outliers present in continous data set
> for (i in 1:length(num_datatype))
+ {
+   assign(paste0("gn",i), ggplot(aes_string(y = (num_datatype[i])), data = subset(dataFrame))+ geom_boxplot()+
+            labs(y=num_datatype[i])+
+            theme(panel.background = element_rect(fill = "palegreen"))+
+            ggtitle(paste("Outlier analysis for ",num_datatype[i])))
+ }
> ## Plotting plots together
> gridExtra::grid.arrange(gn1,gn2,gn3,gn4,ncol=4)
>
```

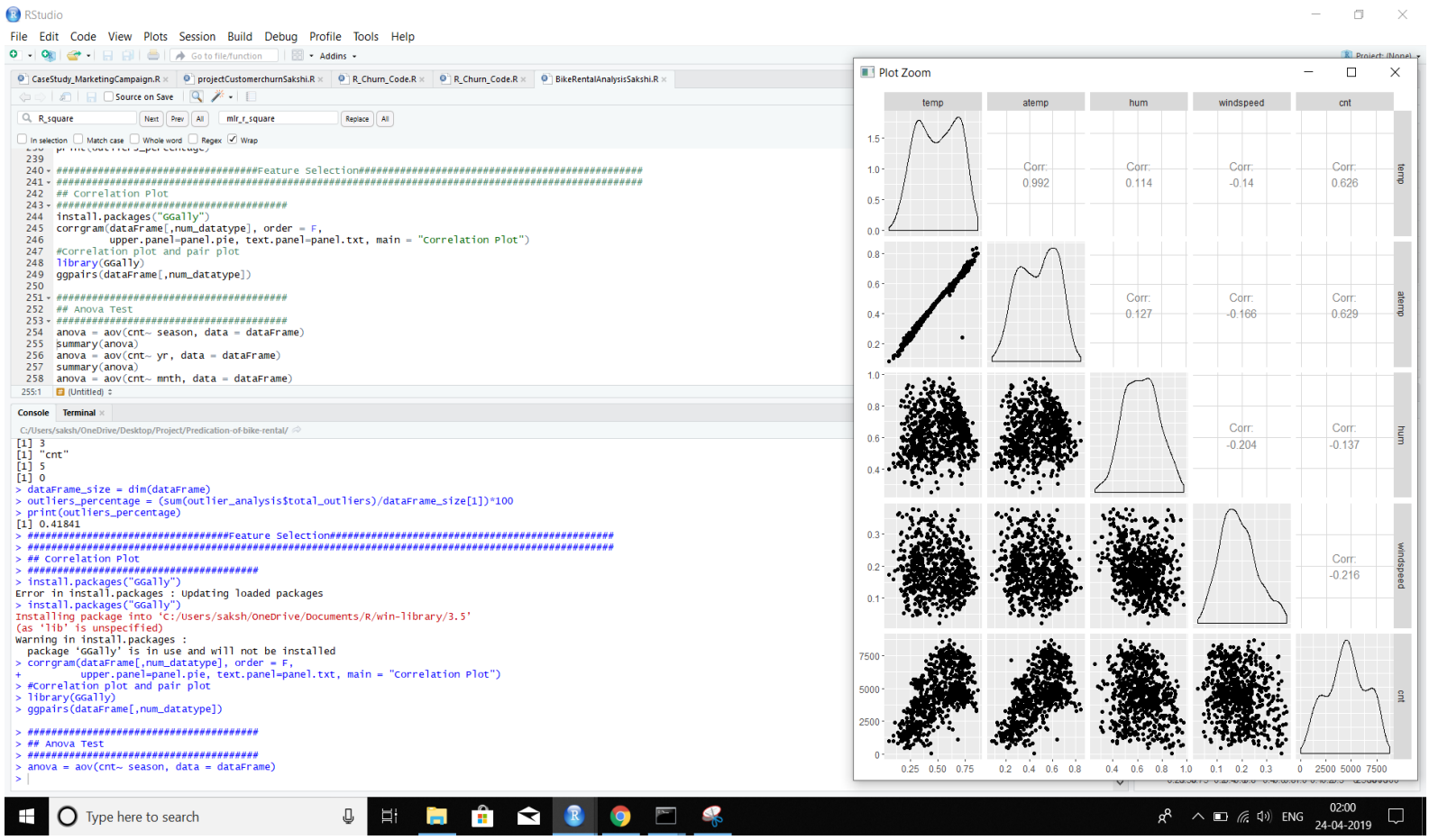
Plot Zoom



Type here to search

ENG 01:57 24-04-2019


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
CaseStudy_MarketingCampaign.R
projectCustomerchurnSakshi.R
R_Churn_Code.R
R_Churn_Code.R
BikeRentalAnalysisSakshi.R
Environment History Connections
Plot Zoom
Outliers in each feature
total_outliers
windspeed hum atemp cnt temp
Console Terminal
C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/
> ggplot(data = outlier_analysis, aes(x = reorder(feature,-total_outliers), y = total_outliers))+
+ geom_bar(stat = "identity",fill = "white")+
+ labs(title = "Outliers in each feature", x = "Features")+
+ theme(panel.background = element_rect("cadetblue"))+
+ theme(plot.title = element_text(hjust = 0.5, face = "bold"))
Type here to search
ENG
01:58
24-04-2019


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
CaseStudy_MarketingCampaign.R
projectCustomerchurnSakshi.R
R_Churn_Code.R
R_Churn_Code.R
BikeRentalAnalysisSakshi.R
Plot Zoom
temp atemp hum windspeed cnt
Corr: 0.992
Corr: 0.114
Corr: -0.14
Corr: 0.626
Corr: 0.127
Corr: -0.166
Corr: 0.629
Corr: -0.204
Corr: -0.137
Corr: -0.216
Console Terminal
C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/
> dataFrame_size = dim(dataFrame)
> outliers_percentage = (sum(outlier_analysis$total_outliers)/dataFrame_size[1])*100
> print(outliers_percentage)
[1] 0.41841
> ####################################Feature Selection##
> ## Correlation Plot
> install.packages("GGally")
Error in install.packages : Updating loaded packages
> install.packages("GGally")
Installing package into 'C:/Users/saksh/OneDrive/Documents/R/win-library/3.5'
(as 'lib' is unspecified)
Warning in install.packages :
package 'GGally' is in use and will not be installed
> corrgram(dataFrame[,num_datatype], order = F,
+ upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
> #Correlation plot and pair plot
> library(GGally)
> ggpairs(dataFrame[,num_datatype])
> ####################################
> ## Anova Test
> ####################################
> anova = aov(cnt~ season, data = dataFrame)
Type here to search
ENG
02:00
24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R

```
251 ######################################
252 ## Anova Test
253 ######################################
254 anova = aov(cnt~ season, data = dataFrame)
255 summary(anova)
256 anova = aov(cnt~ yr, data = dataFrame)
257 summary(anova)
258 anova = aov(cnt~ mnth, data = dataFrame)
259 summary(anova)
260 anova = aov(cnt~ holiday, data = dataFrame)
261 summary(anova)
262 anova = aov(cnt~ weekday, data = dataFrame)
263 summary(anova)
264 anova = aov(cnt~ workingday, data = dataFrame)
265 summary(anova)
266 anova = aov(cnt~ weathersit, data = dataFrame)
267 summary(anova)
```

Console | Terminal

C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> anova = aov(cnt~ yr, data = dataFrame)
> summary(anova)
             Df    Sum Sq   Mean Sq F value Pr(>F)
yr            1 8.813e+08 881327066     351 <2e-16 ***
Residuals   715 1.796e+09   2511190
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> anova = aov(cnt~ mnth, data = dataFrame)
> summary(anova)
             Df    Sum Sq  Mean Sq F value Pr(>F)
mnth         11 1.042e+09 94755196   40.87 <2e-16 ***
Residuals   705 1.635e+09  2318469
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> anova = aov(cnt~ holiday, data = dataFrame)
> summary(anova)
             Df    Sum Sq  Mean Sq F value Pr(>F)
holiday       1 1.377e+07 13770983   3.697 0.0549 .
Residuals   715 2.663e+09  3724555
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> anova = aov(cnt~ weekday, data = dataFrame)
> summary(anova)
             Df    Sum Sq Mean Sq F value Pr(>F)
weekday       6 1.757e+07 2928537   0.782  0.584
Residuals   710 2.659e+09 3745432
> anova = aov(cnt~ workingday, data = dataFrame)
> summary(anova)
             Df    Sum Sq Mean Sq F value Pr(>F)
workingday    1 8.494e+06 8494340   2.276  0.132
Residuals   715 2.668e+09 3731935
>
```

Environment | History | Connections

Global Environment

Data

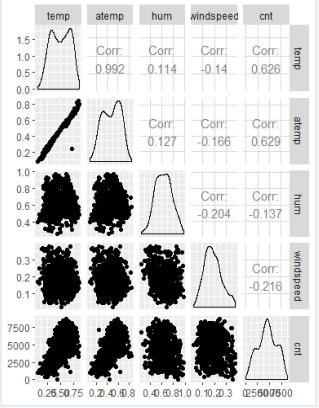
anova	List of 13
dataFrame	717 obs. of 12 variables
datatype	List of 6
gn1	List of 9
gn2	List of 9
gn3	List of 9
gn4	List of 9
gn5	List of 9
outlier_an…	5 obs. of 2 variables

Values

convert_obj	chr [1:7] "season" "yr" "mnth…
dataFrame_…	int [1:2] 717 12
drop_colum…	chr [1:4] "dteday" "casual" "…
fact	"weathersit"
fact_datat…	chr [1:7] "season" "yr" "mnth…
i	5L

Files | Plots | Packages | Help | Viewer

Zoom | Export



Type here to search

ENG 02:02 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R*

```
316 ############################################################
317 # Miltivarient Linear Regression to the Training set
318 ############################################################
319 #Building model
320 set.seed(seed)
321 linear_regressor = lm(formula = cnt ~.,data = train)
322 summary(linear_regressor)
323
324 # Predicting the Test data output
325 y_pred = predict(linear_regressor, newdata = test[,1:20])
326 install.packages("miscTools")
327 library(miscTools)
```

Console Terminal — C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

```
> linear_regressor = lm(formula = cnt ~.,data = train)
> summary(linear_regressor)

Call:
lm(formula = cnt ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-3834.5  -357.0    90.3   507.8  3169.9

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   1585.41     306.34   5.175 3.18e-07 ***
season_2      1033.25     215.93   4.785 2.19e-06 ***
season_3       885.68     249.00   3.557 0.000407 ***
season_4      1569.90     211.38   7.427 4.21e-13 ***
yr_1          1947.24      67.96  28.652  < 2e-16 ***
mnth_2         194.53     167.07   1.164 0.244771
mnth_3         358.56     199.12   1.801 0.072293 .
mnth_4         131.69     291.60   0.452 0.651735
mnth_5         420.64     313.84   1.340 0.180688
mnth_6         113.39     330.44   0.343 0.731616
mnth_7        -220.73     361.10  -0.611 0.541265
mnth_8         206.37     347.65   0.594 0.553011
mnth_9         790.99     303.10   2.610 0.009309 **
mnth_10        466.30     280.61   1.662 0.097127 .
mnth_11       -209.96     265.34  -0.791 0.429114
mnth_12        -99.17     215.16  -0.461 0.645057
weathersit_1   376.37      90.35   4.166 3.60e-05 ***
weathersit_3 -1300.35     220.28  -5.903 6.21e-09 ***
temp          5014.79     487.38  10.289  < 2e-16 ***
hum          -1966.08     359.10  -5.475 6.63e-08 ***
windspeed    -2868.65     511.39  -5.609 3.21e-08 ***
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 791.2 on 555 degrees of freedom
Multiple R-squared:  0.8373,    Adjusted R-squared:  0.8315
```

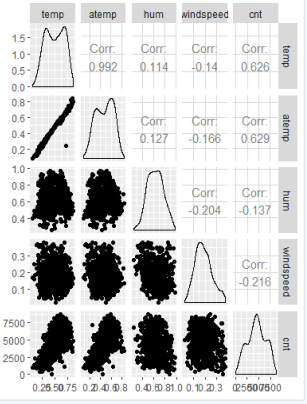
Environment | History | Connections

Import Dataset — Global Environment

Data	
dataFrame	717 obs. of 21 variables
linear_reg…	List of 12
test	141 obs. of 21 variables
train	576 obs. of 21 variables
train.index	int [1:576, 1] 2 3 6 7 8 9 …
Values	
seed	1234
Functions	
MAPE	function (y, yhat)

Files | Plots | Packages | Help | Viewer

Zoom | Export



Type here to search

ENG 02:07 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R*

```
316 ############################################################
317 # Miltivarient Linear Regression to the Training set
318 ############################################################
319 #Building model
320 set.seed(seed)
321 linear_regressor = lm(formula = cnt ~.,data = train)
322 summary(linear_regressor)
323
324 # Predicting the Test data output
325 y_pred = predict(linear_regressor, newdata = test[,1:20])
326 install.packages("miscTools")
327 library(miscTools)
```

Environment | History | Connections

Global Environment

Data	
dataFrame	717 obs. of 21 variables
linear_reg…	List of 12
test	141 obs. of 21 variables
train	576 obs. of 21 variables
train.index	int [1:576, 1] 2 3 6 7 8 9 …
Values	
seed	1234
Functions	
MAPE	function (y, yhat)

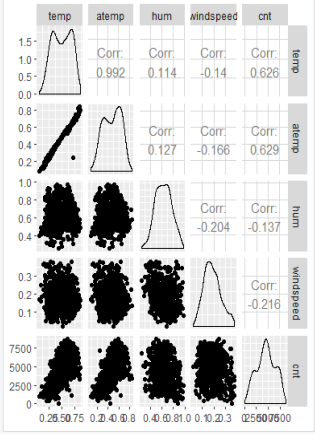
Console | Terminal

C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

```
> linear_regressor = lm(formula = cnt ~.,data = train)
> summary(linear_regressor)

Call:
lm(formula = cnt ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-3834.5  -357.0    90.3   507.8  3169.9

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   1585.41     306.34   5.175 3.18e-07 ***
season_2      1033.25     215.93   4.785 2.19e-06 ***
season_3       885.68     249.00   3.557 0.000407 ***
season_4      1569.90     211.38   7.427 4.21e-13 ***
yr_1          1947.24      67.96  28.652  < 2e-16 ***
mnth_2         194.53     167.07   1.164 0.244771
mnth_3         358.56     199.12   1.801 0.072293 .
mnth_4         131.69     291.60   0.452 0.651735
mnth_5         420.64     313.84   1.340 0.180688
mnth_6         113.39     330.44   0.343 0.731616
mnth_7        -220.73     361.10  -0.611 0.541265
mnth_8         206.37     347.65   0.594 0.553011
mnth_9         790.99     303.10   2.610 0.009309 **
mnth_10        466.30     280.61   1.662 0.097127 .
mnth_11       -209.96     265.34  -0.791 0.429114
mnth_12        -99.17     215.16  -0.461 0.645057
weathersit_1   376.37      90.35   4.166 3.60e-05 ***
weathersit_3 -1300.35     220.28  -5.903 6.21e-09 ***
temp          5014.79     487.38  10.289  < 2e-16 ***
hum          -1966.08     359.10  -5.475 6.63e-08 ***
windspeed    -2868.65     511.39  -5.609 3.21e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 791.2 on 555 degrees of freedom
Multiple R-squared:  0.8373,    Adjusted R-squared:  0.8315
```

Files | Plots | Packages | Help | Viewer

Zoom | Export



Type here to search

ENG 02:07 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

CaseStudy_MarketingCampaign.R | projectCustomerchurnSakshi.R | R_Churn_Code.R | R_Churn_Code.R | BikeRentalAnalysisSakshi.R*

```
tunegrid = expand.grid(.maxdepth=maxdepth)
dt_random = caret::train(cnt~., data=train, method="rpart2", metric="RMSE", tuneLength =10, trControl=control)

#print out summary of the model
print(dt_random)

#Best fit parameters
view_dt_rand_para = dt_random$bestTune
print(view_dt_rand_para)
#Build model based on best fit
rand_dt_model = rpart(cnt ~ ., train[,0:21], method = "anova", maxdepth = 12)
#print out summary of the model
print(rand_dt_model)
```

Console Terminal

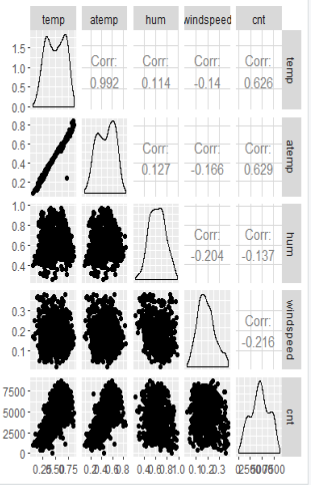
C:/Users/saksh/OneDrive/Desktop/Project/Predication-of-bike-rental/

```
> control = trainControl(method="repeatedcv", number=5, repeats=1,search='random')
> set.seed(seed)
> maxdepth = c(1:30)
> tunegrid = expand.grid(.maxdepth=maxdepth)
> dt_random = caret::train(cnt~., data=train, method="rpart2", metric="RMSE", tuneLength =10, trControl=control)
> #print out summary of the model
> print(dt_random)
CART

576 samples
 20 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 1 times)
Summary of sample sizes: 461, 460, 461, 461, 461
Resampling results across tuning parameters:

  maxdepth  RMSE      Rsquared   MAE
   1        1516.933  0.3873027  1285.0712
   2        1238.208  0.5939700   967.3788
   8        1000.413  0.7338131   745.7781
   9        1000.413  0.7338131   745.7781
  11        1000.413  0.7338131   745.7781
  13        1000.413  0.7338131   745.7781
  17        1000.413  0.7338131   745.7781
  18        1000.413  0.7338131   745.7781
  25        1000.413  0.7338131   745.7781
  26        1000.413  0.7338131   745.7781

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was maxdepth = 8.
> #Best fit parameters
> view_dt_rand_para = dt_random$bestTune
> print(view_dt_rand_para)
  maxdepth
3        8
>
```

Environment | History | Connections

Data	
control	List of 27
dataFrame	717 obs. of 21 variables
dt_r2	num [1, 1] 0.806
dt_random	Large train (23 elements, 1…
fit	List of 14
linear_reg…	List of 12
mlr_r_squa…	num [1, 1] 0.838
test	141 obs. of 21 variables
train	576 obs. of 21 variables
train.index	int [1:576, 1] 2 3 6 7 8 9 …
tunegrid	30 obs. of 1 variable
view_dt_ra…	1 obs. of 1 variable
Values	
dt_mape	26.464258414947
dt_mse	742947.357777098

Files | Plots | Packages | Help | Viewer



Type here to search

ENG 02:09 24-04-2019

RStudio

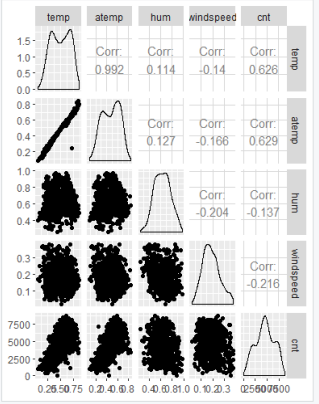
File Edit Code View Plots Session Build Debug Profile Tools Help

```
print(view_dt_rand_para)
#Build model based on best fit
rand_dt_model = rpart(cnt ~ ., train[,0:21], method = "anova", maxdepth = 12)
#print out summary of the model
print(rand_dt_model)

#Presdict test data using random forest model
rand_dt_Predictions = predict(rand_dt_model, test[,1:20])

#Compute R^2
rand_dt_r2 = rSquared(test[,21], test[,21] - rand_dt_Predictions)
print(rand_dt_r2)
#Compute MSE
```

```
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was maxdepth = 8.
> #Best fit parameters
> view_dt_rand_para = dt_random$bestTune
> print(view_dt_rand_para)
  maxdepth
3        8
> #Build model based on best fit
> rand_dt_model = rpart(cnt ~ ., train[,0:21], method = "anova", maxdepth = 12)
> #print out summary of the model
> print(rand_dt_model)
n= 576

node), split, n, deviance, yval
      * denotes terminal node

 1) root 576 2135762000.0 4546.372
   2) temp< 0.432174 231  514457900.0 3073.268
     4) yr_1< 0.5 122  124931800.0 2258.648
       8) season_4< 0.5 83   30000890.0 1732.747 *
       9) season_4>=0.5 39   23121580.0 3377.872 *
     5) yr_1>=0.5 109  217950200.0 3985.046
      10) temp< 0.2804165 32   23848190.0 2673.875 *
      11) temp>=0.2804165 77  116225900.0 4529.948
        22) season_4< 0.5 39   47758040.0 4008.487 *
        23) season_4>=0.5 38   46978890.0 5065.132 *
   3) temp>=0.432174 345  784388800.0 5532.710
     6) yr_1< 0.5 163  104166300.0 4338.196
      12) weathersit_3>=0.5 7     911356.9 2366.857 *
      13) weathersit_3< 0.5 156   74831010.0 4426.654 *
     7) yr_1>=0.5 182  23934300.0 6602.522
      14) hum>=0.7710415 22    5452910.0 5095.182 *
      15) hum< 0.7710415 160  12703370.0 6809.781 *
> #Presdict test data using random forest model
> rand_dt_Predictions = predict(rand_dt_model, test[,1:20])
>
```

Environment | History | Connections

fit	List of 14
linear_reg…	List of 12
mlr_r_squa…	num [1, 1] 0.838
rand_dt_mo…	List of 14
test	141 obs. of 21 variables
train	576 obs. of 21 variables
train.index	int [1:576, 1] 2 3 6 7 8 9 …
tunegrid	30 obs. of 1 variable
view_dt_ra…	1 obs. of 1 variable
Values	
dt_mape	26.464258414947
dt_mse	742947.357777098
maxdepth	int [1:30] 1 2 3 4 5 6 7 8 9 …
mlr_mape	17.1915802912793
mlr_mse	620864.660446473
prediction…	Named num [1:141] 1733 1733 1…
rand_dt_Pr…	Named num [1:141] 1733 1733 1…



Type here to search

ENG 02:10 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
print(view_dt_rand_para)
#Build model based on best fit
rand_dt_model = rpart(cnt ~ ., train[,0:21], method = "anova", maxdepth = 12)
#print out summary of the model
print(rand_dt_model)

#Presdict test data using random forest model
rand_dt_Predictions = predict(rand_dt_model, test[,1:20])

#Compute R^2
rand_dt_r2 = rSquared(test[,21], test[,21] - rand_dt_Predictions)
print(rand_dt_r2)
#Compute MSE
```

```
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was maxdepth = 8.
> #Best fit parameters
> view_dt_rand_para = dt_random$bestTune
> print(view_dt_rand_para)
  maxdepth
3        8
> #Build model based on best fit
> rand_dt_model = rpart(cnt ~ ., train[,0:21], method = "anova", maxdepth = 12)
> #print out summary of the model
> print(rand_dt_model)
n= 576

node), split, n, deviance, yval
      * denotes terminal node

 1) root 576 2135762000.0 4546.372
   2) temp< 0.432174 231  514457900.0 3073.268
     4) yr_1< 0.5 122  124931800.0 2258.648
       8) season_4< 0.5 83   30000890.0 1732.747 *
       9) season_4>=0.5 39   23121580.0 3377.872 *
     5) yr_1>=0.5 109  217950200.0 3985.046
      10) temp< 0.2804165 32   23848190.0 2673.875 *
      11) temp>=0.2804165 77  116225900.0 4529.948
        22) season_4< 0.5 39   47758040.0 4008.487 *
        23) season_4>=0.5 38   46978890.0 5065.132 *
   3) temp>=0.432174 345  784388800.0 5532.710
     6) yr_1< 0.5 163  104166300.0 4338.196
      12) weathersit_3>=0.5 7     911356.9 2366.857 *
      13) weathersit_3< 0.5 156   74831010.0 4426.654 *
     7) yr_1>=0.5 182  239345300.0 6602.522
      14) hum>=0.7710415 22   55452910.0 5095.182 *
      15) hum< 0.7710415 160  127033700.0 6809.781 *
> #Presdict test data using random forest model
> rand_dt_Predictions = predict(rand_dt_model, test[,1:20])
>
```

Environment	
fit	List of 14
linear_reg…	List of 12
mlr_r_squa…	num [1, 1] 0.838
rand_dt_mo…	List of 14
test	141 obs. of 21 variables
train	576 obs. of 21 variables
train.index	int [1:576, 1] 2 3 6 7 8 9 …
tunegrid	30 obs. of 1 variable
view_dt_ra…	1 obs. of 1 variable
Values	
dt_mape	26.464258414947
dt_mse	742947.357777098
maxdepth	int [1:30] 1 2 3 4 5 6 7 8 9 …
mlr_mape	17.1915802912793
mlr_mse	620864.660446473
prediction…	Named num [1:141] 1733 1733 1…
rand_dt_Pr…	Named num [1:141] 1733 1733 1…

Type here to search

ENG 02:10 24-04-2019

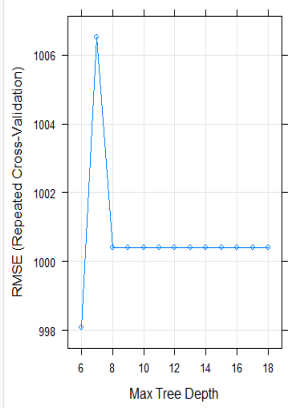
RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
#Compute R^2
grid_dt_r2 = rSquared(test[,21], test[,21] - grid_dt_Predictions)
print(grid_dt_r2)
#Compute MSE
grid_dt_mse = mean((test[,21] - grid_dt_Predictions)^2)
print(grid_dt_mse)
#Compute MAPE
grid_dt_mape = MAPE(test[,21], grid_dt_Predictions)
print(grid_dt_mape)


##########################################################################################################################
```

```
> print(grid_dt_model)
n= 576

node), split, n, deviance, yval
      * denotes terminal node

 1) root 576 2135762000.0 4546.372
   2) temp< 0.432174 231  514457900.0 3073.268
     4) yr_1< 0.5 122  124931800.0 2258.648
       8) season_4< 0.5 83   30000890.0 1732.747 *
       9) season_4>=0.5 39   23121580.0 3377.872 *
     5) yr_1>=0.5 109  217950200.0 3985.046
      10) temp< 0.2804165 32   23848190.0 2673.875 *
      11) temp>=0.2804165 77  116225900.0 4529.948
        22) season_4< 0.5 39   47758040.0 4008.487 *
        23) season_4>=0.5 38   46978890.0 5065.132 *
   3) temp>=0.432174 345  784388800.0 5532.710
     6) yr_1< 0.5 163  104166300.0 4338.196
      12) weathersit_3>=0.5 7     911356.9 2366.857 *
      13) weathersit_3< 0.5 156   74831010.0 4426.654 *
     7) yr_1>=0.5 182  239345300.0 6602.522
      14) hum>=0.7710415 22   55452910.0 5095.182 *
      15) hum< 0.7710415 160  127033700.0 6809.781 *
> #Presdict test data using model
> grid_dt_Predictions = predict(grid_dt_model, test[,1:20])
> #Compute R^2
> grid_dt_r2 = rSquared(test[,21], test[,21] - grid_dt_Predictions)
> print(grid_dt_r2)
          [,1]
[1,] 0.8061982
> #Compute MSE
> grid_dt_mse = mean((test[,21] - grid_dt_Predictions)^2)
> print(grid_dt_mse)
[1] 742947.4
> #Compute MAPE
> grid_dt_mape = MAPE(test[,21], grid_dt_Predictions)
>
```

Values	
dt_mape	26.464258414947
dt_mse	742947.357777098
grid_dt_ma…	26.464258414947



Type here to search

ENG 02:12 24-04-2019

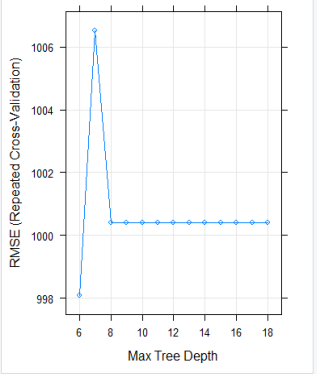
RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
set.seed(seed)
rf_random = caret::train(cnt~., data=train, method="rf", metric="RMSE", tuneLength =10, trControl=control)

#print out summary of the model
#here we see that when mtry is 5 RMSE is at its lowest
print(rf_random)

#Best fit parameters
view_rf_rand_para = rf_random$bestTune
print(view_rf_rand_para)

#Build model based on best fit
rand_rf_model = randomForest(cnt ~ ., train[,0:21], method = "anova",importance = TRUE, mtry = 5)
```

```
> ###Random Forest Regression
> ####################################
> #Building model
> RF_model = randomForest(cnt ~ ., train[,0:21], method = "anova",importance = TRUE)
> #Prints out model information
> print(RF_model)

Call:
 randomForest(formula = cnt ~ ., data = train[, 0:21], method = "anova",      importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 6

          Mean of squared residuals: 497784.9
                    % Var explained: 86.58
> #Predicting the Test data output
> RF_Predictions = predict(RF_model, test[,1:20])
> #Compute R^2
> rf_r2 = rSquared(test[,21], test[,21] - RF_Predictions)
> print(rf_r2)
          [,1]
[1,] 0.87477
> #Compute MSE
> rf_mse = mean((test[,21] - RF_Predictions)^2)
> print(rf_mse)
[1] 480074.5
> #Compute MAPE
> rf_mape = MAPE(test[,21], RF_Predictions)
> print(rf_mape)
[1] 20.83274
> # Create model with Random paramters 5 fold CV with 1 repeats
> control = trainControl(method="repeatedcv", number=5, repeats=1,search='random')
> set.seed(seed)
> rf_random = caret::train(cnt~., data=train, method="rf", metric="RMSE", tuneLength =10, trControl=control)
#print out summary of the model
#Best fit parameters
```

Environment	
control	List of 27
dataFrame	717 obs. of 21 variables
dt_grid	Large train (23 elements, 1…
dt_r2	num [1, 1] 0.806
dt_random	Large train (23 elements, 1…
fit	List of 14
grid_dt_mo…	List of 14
grid_dt_r2	num [1, 1] 0.806
linear_reg…	List of 12
mlr_r_squa…	num [1, 1] 0.838
rand_dt_mo…	List of 14
rand_dt_r2	num [1, 1] 0.806
RF_model	Large randomForest.formula …
rf_r2	num [1, 1] 0.875
test	141 obs. of 21 variables



Type here to search

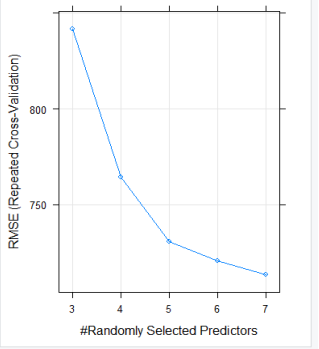
ENG 02:14 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
511 view_rf_grid_pram = rf_grid$bestTune
512 print(view_rf_grid_pram)
513
514 #Build model based on best fir value
515 grid_rf_model = randomForest(cnt ~ ., train[,0:21], method = "anova",importance = TRUE,mtry = 6)
516 #print out summary of the model
517 #from the summary we see that this model explains 87.55% of the variance
518 print(grid_rf_model)
519
520 #Presdict test data using model
521 grid_rf_predictions = predict(grid_rf_model, test[,1:20])
522
523
```

```
No pre-processing
Resampling: Cross-Validated (5 fold, repeated 1 times)
Summary of sample sizes: 461, 460, 461, 461, 461
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  3     841.8870  0.8494069  651.8550
  4     764.2342  0.8605079  573.5180
  5     730.7203  0.8654998  536.6052
  6     720.6990  0.8644885  520.1904
  7     713.1921  0.8656792  513.5886

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 7.
> #Plot RMSE Vs mtry values
> #from the plot we can see that when the randomly selected predictor is 6 RMSE is at its lowest
> plot(rf_grid)
> #Best fit parameters
> view_rf_grid_pram = rf_grid$bestTune
> print(view_rf_grid_pram)
  mtry
5    7
> #Build model based on best fir value
> grid_rf_model = randomForest(cnt ~ ., train[,0:21], method = "anova",importance = TRUE,mtry = 6)
> #print out summary of the model
> #from the summary we see that this model explains 87.55% of the variance
> print(grid_rf_model)

Call:
 randomForest(formula = cnt ~ ., data = train[, 0:21], method = "anova",      importance = TRUE, mtry = 6)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 6

          Mean of squared residuals: 502679.5
                    % Var explained: 86.44
>
```



Environment | History | Connections

grid_rf_mo...	Large randomForest.formula ...
linear_reg...	List of 12
mlr_r_squa...	num [1, 1] 0.838
rand_dt_mo...	List of 14
rand_dt_r2	num [1, 1] 0.806
rand_rf_mo...	Large randomForest.formula ...
rand_rf_r2	num [1, 1] 0.866
rf_grid	Large train (23 elements, 6...
RF_model	Large randomForest.formula ...
rf_r2	num [1, 1] 0.875
rf_random	Large train (23 elements, 6...
test	141 obs. of 21 variables
train	576 obs. of 21 variables
train.index	int [1:576, 1] 2 3 6 7 8 9 ...
tunegrid	5 obs. of 1 variable
view_dt_gr...	1 obs. of 1 variable

Type here to search

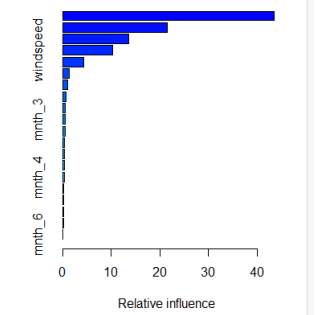
ENG 02:15 24-04-2019

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

```
550 gbm_mse = mean((test[,21] - gbm_Predictions)^2)
551 print(gbm_mse)
552 #Compute MAPE
553 gbm_mape = MAPE(test[,21], gbm_Predictions)
554 print(gbm_mape)
555 ##############################################################################################################
556 ## Random Search ##############################################################################################
557 ##############################################################################################################
```

```
> ##############################################################################################################
> ####################################
> ###Gradient Boosting
> ####################################
> #Building model
> gbm_base = gbm(cnt~., data = train,distribution = "gaussian",n.trees = 10000,
+                shrinkage = 0.01, interaction.depth = 4)
> #Print out summary of the model
> summary(gbm_base)
                    var     rel.inf
temp               temp 43.55833026
yr_1               yr_1 21.59753154
hum                 hum 13.51074477
windspeed     windspeed 10.28090551
season_4       season_4  4.35422780
weathersit_1 weathersit_1  1.34662202
weathersit_3 weathersit_3  0.99244972
mnth_9           mnth_9  0.71907786
mnth_11         mnth_11  0.56403685
mnth_3           mnth_3  0.56002913
mnth_10         mnth_10  0.52017097
season_2       season_2  0.40817432
mnth_12         mnth_12  0.29654879
season_3       season_3  0.27507632
mnth_4           mnth_4  0.27326316
mnth_5           mnth_5  0.20687279
mnth_2           mnth_2  0.20171001
mnth_7           mnth_7  0.13956857
mnth_8           mnth_8  0.13205553
mnth_6           mnth_6  0.06260407
> #Presdict test data using random forest model
> gbm_Predictions = predict(gbm_base, test[,1:20],n.trees = 10000)
> #Compute R^2
> gbm_r2 = rSquared(test[,21], test[,21] - gbm_Predictions)
> print(gbm_r2)
          [,1]
[1,] 0.864654
> #Compute MSE
> gbm_mse = mean((test[,21] - gbm_Predictions)^2)
> #Compute R^2
> gbm_r2 = rSquared(test[,21], test[,21] - gbm_Predictions)
> print(gbm_r2)
```



Environment | History | Connections

rf_grid	Large train (23 elements, 6...
RF_model	Large randomForest.formula ...
rf_r2	num [1, 1] 0.875
rf_random	Large train (23 elements, 6...
test	141 obs. of 21 variables
train	576 obs. of 21 variables
train.index	int [1:576, 1] 2 3 6 7 8 9 ...
tunegrid	5 obs. of 1 variable
view_dt_gr...	1 obs. of 1 variable
view_dt_ra...	1 obs. of 1 variable
view_rf_gr...	1 obs. of 1 variable
view_rf_ra...	1 obs. of 1 variable
Values	
dt_mape	26.464258414947
dt_mse	742947.357777098
gbm_mape	19.4038818987614

Type here to search

ENG 02:16 24-04-2019