BIKE RENTING PREDICTION

Ipsita Sahu

Contents

Chapter 1

Introduction

1.1 Problem Description –

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

1.2 Data-

The details of data attributes in the dataset are as follows –

Table 1.1 Bike Renting Reduction Data (Columns 1-5)

Instant	Dteday	Season	Yr	Mnth
1	01-01-2011	1	0	1
2	02-01-2011	1	0	1
3	03-01-2011	1	0	1
4	04-01-2011	1	0	1
5	05-01-2011	1	0	1

Table 1.2 Bike Renting Reduction Data (Columns 6-10)

Holiday	Weekday	Working day	Weathersit	Temp
0	6	0	2	0.344
0	0	0	2	0.363
0	1	1	1	0.196
0	2	1	1	0.2
0	3	1	1	0.227

Table 1.3 Bike Renting Reduction Data (Columns 11-16)

Atempt	Hum	Windspeed	Casual	Registered	Cnt
0.364	0.806	0.160	331	654	985
0.354	0.696	0.249	131	670	801
0.189	0.437	0.248	120	1229	1349
0212	0.590	0.160	108	1454	1562
0.229	0.437	0.187	82	1518	1600

The Predictors provided are as follows

- 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_max- t_min), t_min=-16, t_max=+50 (only in hourly scale)

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

Target Variable : cnt

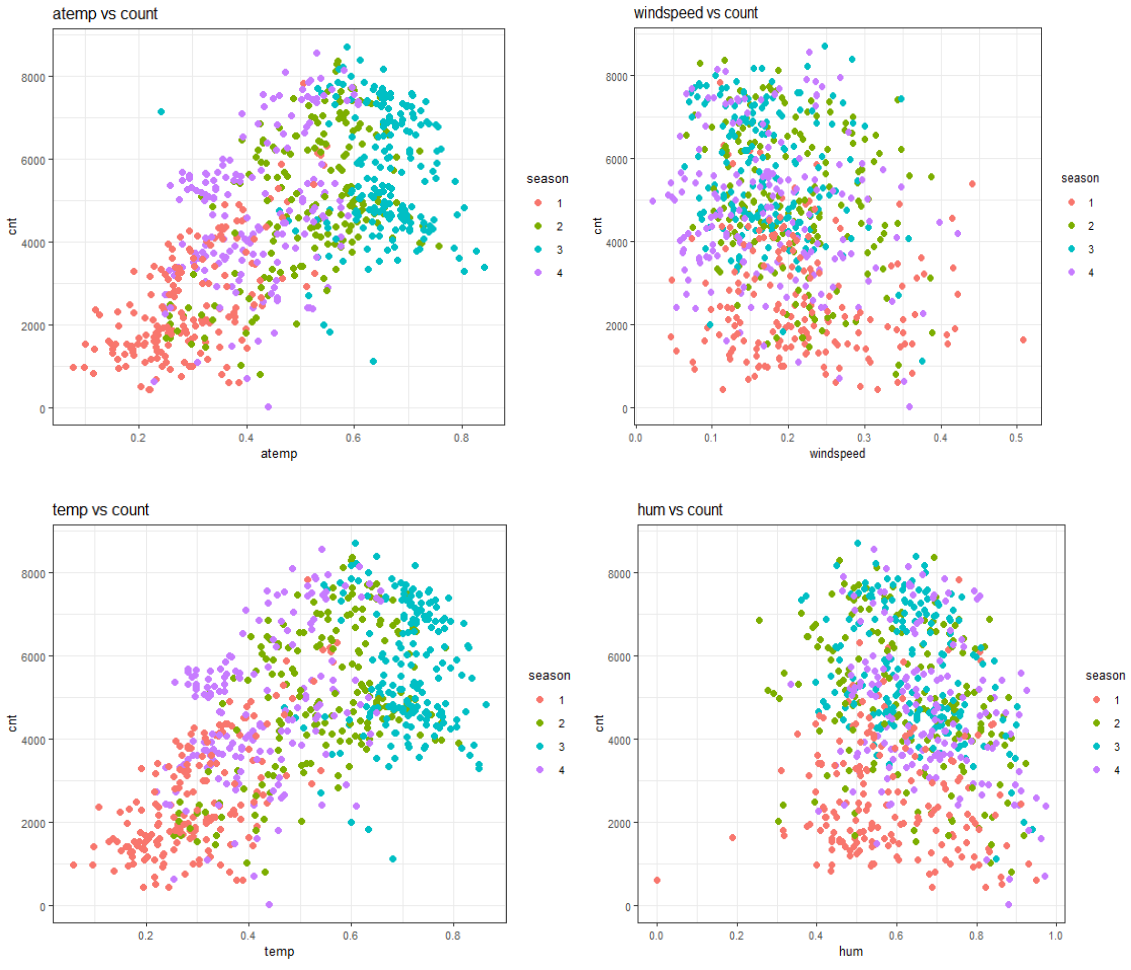
We have to predict the count of bike rental on daily basis

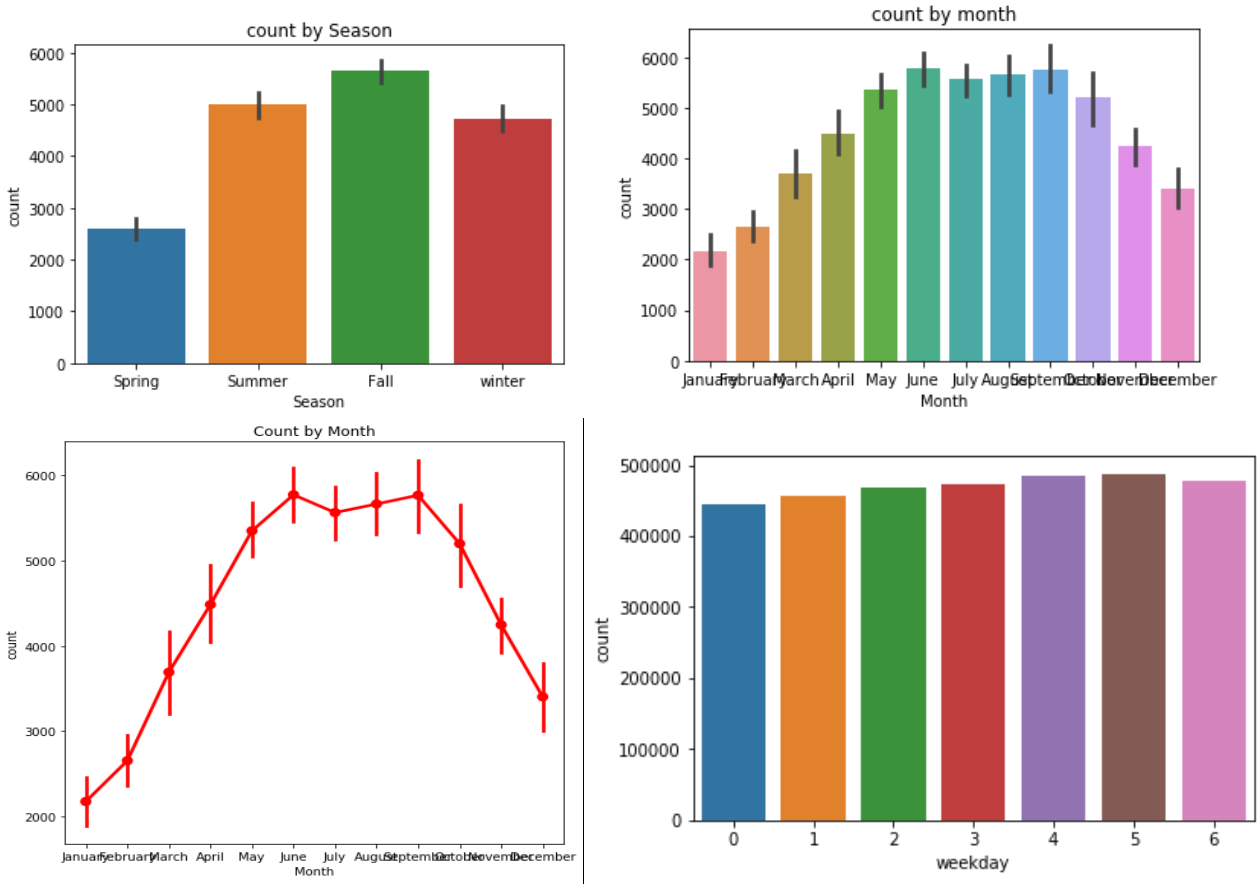
Chapter 2

Methodology

2.1 Pre Processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Data pre-processing includes cleaning, outlier deduction, normalization, feature extraction and selection, etc. The product of data pre-processing is the final data. This is often called as Exploratory Data Analysis.





2.1.1 Missing value analysis

Imputation simply means replacing the missing values with an estimate, then analyzing the full data set as if the imputed values were actual observed values. The best method to impute missing value for a data are imputation using mean, median and KNN method. We can calculate the mean/median of the non-missing values in a column and then replacing the missing values within each column separately. In KNN (distance based method) imputation find the nearest neighbor based on the existing attribute and then find the Euclidean distance. In KNN it will try to find the distance score between missing value and observation and select those observations which are close to the missing value and impute that value. The Euclidean distance or Euclidean metric is the "ordinary" (i.e. straight-line) distance between two points in Euclidean space. The Manhattan distance defined as the sum of the lengths of the projections of the line segment between the points onto the coordinate axes. The dataset which has been given has no missing value.

2.1.2 Outliers Analysis

An outlier is a data point that differs significantly from other observations.

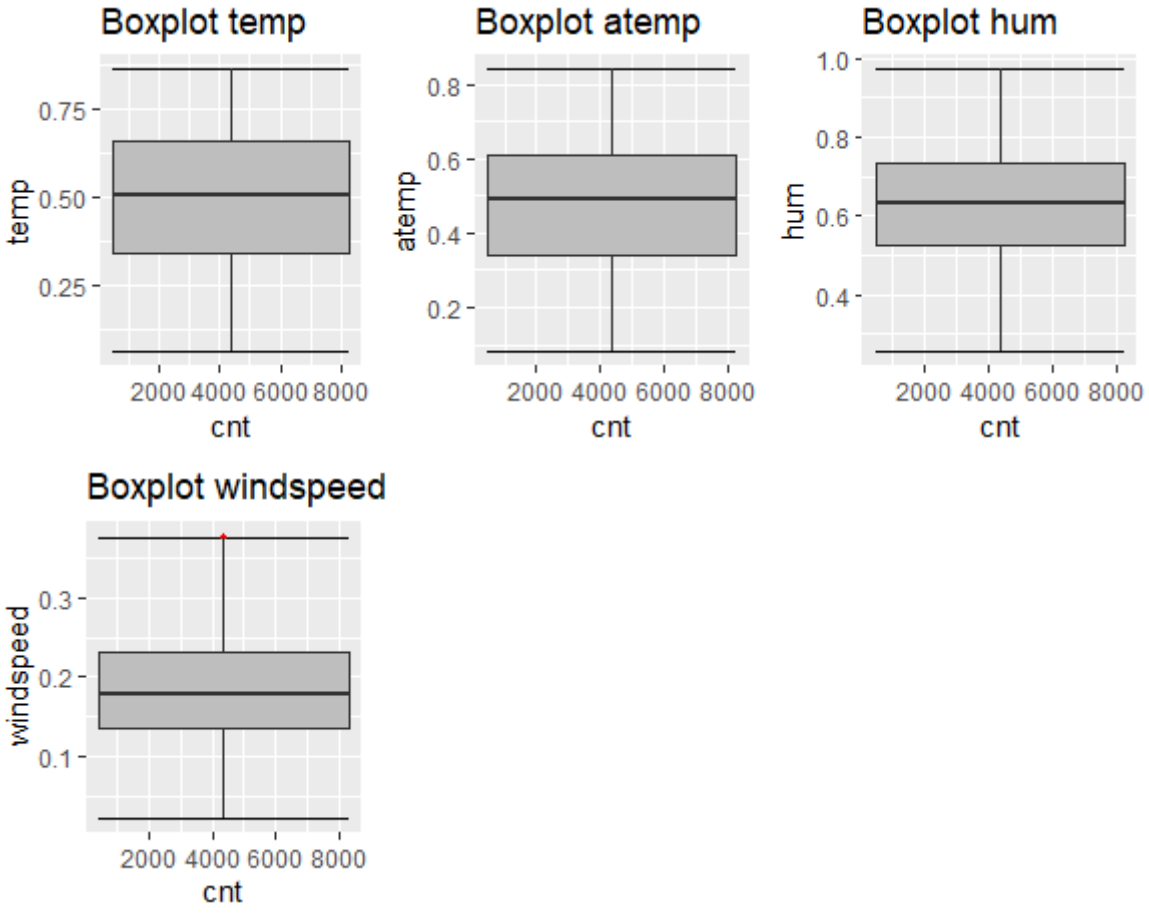
Causes of Outliers

•Poor data quality / contamination

•Low quality measurements, malfunctioning equipment, manual error

•Correct but exceptional data

Box plot diagram is a graphical method typically depicted by quartiles and inter quartiles that helps in defining the upper limit and lower limit beyond which any data lying will be considered as outliers. The purpose of this diagram is to identify outliers and discard it from the data series before making any further observation so that the conclusion made from the study gives more accurate results not influenced by any extremes or abnormal values.



2.1.3 Feature selection

In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem.

Correlation –

1.It show whether and how strongly pairs of variables are related to each other.

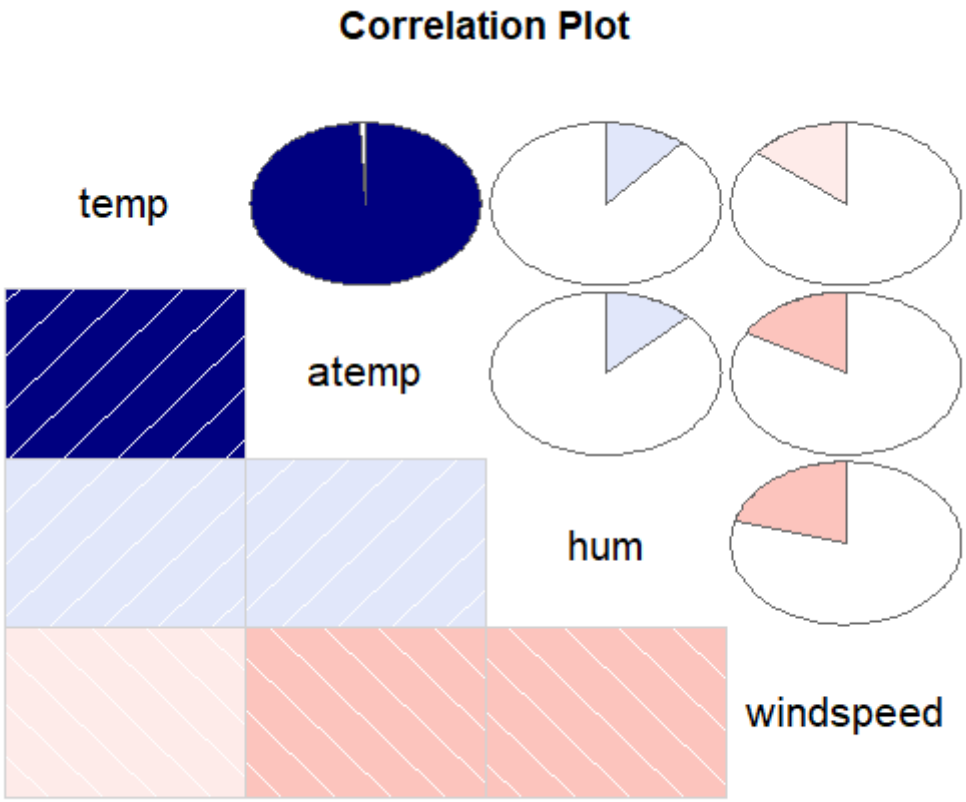
2.Correlation takes values between -1 to +1, wherein values close to +1 represents strong positive correlation and values close to -1 represents strong negative correlation.

3.In this variable are indirectly related to each other.

4.It gives the direction and strength of relationship between variables.

(A)Correlation analysis

A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (Xi) in the table is correlated with each of the other values in the table (Xj). This allows you to find and predict which pairs have the highest correlation. If two variables are highly correlated we have two drop one variable .Here temp and atemp are highly correlated.



(B) ANOVA

Inferential statistics are used to determine if observed data we obtain from a sample (i.e., data we collect) are different from what one would expect by chance alone. A more simple answer is that we want to determine if the relationships among variables or differences between groups that we see in our sample data are occurring in the entire population. In ANOVA, we have two or more group means (averages) that we want to compare. In an ANOVA, one variable must be categorical and the other must be continuous.

```
> m1=aov(cnt~season+yr+mnth+holiday+weekday+workingday+weathersit)
> summary(m1)
             Df    Sum Sq   Mean Sq  F value  Pr(>F)
season        3 921846604 307282201  427.956 < 2e-16 ***
yr            1 871757387 871757387 1214.108 < 2e-16 ***
mnth         11 184091208  16735564   23.308 < 2e-16 ***
holiday       1   3612964   3612964    5.032 0.02520 *
weekday       6  14576781   2429463    3.384 0.00269 **
weathersit    2 184071169  92035585  128.179 < 2e-16 ***
Residuals   692 496871695    718023
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

From the above summary it is clearly seen that holiday and weekday are carrying low weightage.

2.1.4 Scaling method

When you’re working with a learning model, it is important to scale the features to a range which is centered around zero. It is done to bring the feature on a same scale. This is done so that the variance of the features are in the same range. If a feature’s variance is orders of magnitude more than the variance of other features, that particular feature might dominate other features in the dataset, which is not something we want happening in our model. Before choosing the data scaling method, we need to check the distribution of data. If the data is uniformly distributed, then Standardization is the suitable method for the scaling purpose. On the other hand, if the data is not normally distributed, we go with Normalization scaling method. Normalization is bringing all the variables into proportion with one another. Normalization is the process of reducing unwanted variation either within or between variables. It range between 0 to 1. The given dataset is in normalized form.

2.2Modeling

2.2.1 Model Selection

Before applying the model we have to divide the data into train and test sets.

2.2.2 Multiple linear regression

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable. In essence, multiple regression is the extension of ordinary least-square (OLS) regression that involves more than one explanatory variable.

The co coefficient of determination (R-square) is a statistical metric that is used to measure how of the variation in outcome can be explained by the variation in the independent variables.R2 always increases as more predictions are added to the MLR model even though the predictors may not be related to the outcome variable. R2 by itself can not thus be used to identify which predictors should be included in a model and which should be excluded.R2 can only be between 0 and 1 ,where 0 indicates that outcome cannot be predicted by any of the independent variable and 1 indicates that the outcomes can be predicted without error from the independent variable.

```
> lm_mod=lm(cnt~.,train)
> summary(lm_mod)

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

Residuals:
    Min      1Q  Median      3Q     Max
-3949.1  -339.1    59.1   487.1  2855.5

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2067.166    186.620  11.077  < 2e-16 ***
season2       779.206    202.896   3.840 0.000137 ***
season3       777.200    249.891   3.110 0.001968 **
season4      1487.820    213.466   6.970 9.20e-12 ***
yr1          2084.713     67.136  31.052  < 2e-16 ***
mnth2         180.059    171.198   1.052 0.293376
mnth3         675.313    189.872   3.557 0.000408 ***
mnth4         683.912    281.067   2.433 0.015284 *
mnth5         940.187    299.260   3.142 0.001771 **
mnth6         822.162    313.167   2.625 0.008900 **
mnth7         316.038    350.676   0.901 0.367867
mnth8         767.074    338.565   2.266 0.023864 *
mnth9        1205.478    302.116   3.990 7.51e-05 ***
mnth10        694.732    283.437   2.451 0.014555 *
mnth11        -53.241    267.960  -0.199 0.842579
mnth12          4.709    215.367   0.022 0.982562
weekday1     -297.672    208.520  -1.428 0.153994
weekday2     -272.225    230.220  -1.182 0.237541
weekday3     -149.969    229.775  -0.653 0.514240
weekday4     -232.532    230.016  -1.011 0.312495
weekday5      -98.985    228.689  -0.433 0.665305
weekday6      516.332    124.843   4.136 4.10e-05 ***
workingday1   610.950    197.326   3.096 0.002061 **
weathersit2  -486.544     89.923  -5.411 9.42e-08 ***
weathersit3 -1681.400    224.744  -7.481 2.96e-13 ***
atemp         746.735     82.106   9.095  < 2e-16 ***
hum          -224.267     48.637  -4.611 5.00e-06 ***
windspeed    -182.298     35.524  -5.132 4.00e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 780.5 on 545 degrees of freedom
Multiple R-squared:  0.8468,    Adjusted R-squared:  0.8392
F-statistic: 111.5 on 27 and 545 DF,  p-value: < 2.2e-16
```

```
> library(Metrics)
> rmse(test[,10],lm_prd)
[1] 758.0349
```

There are three popular regularization techniques, each of them aiming at decreasing the size of the coefficients:

- Ridge Regression, which penalizes sum of squared coefficients (L2 penalty).
- Lasso Regression, which penalizes the sum of absolute values of the coefficients (L1 penalty).
- Elastic Net, a convex combination of Ridge and Lasso.

Ridge: -

In Ridge Regression, the OLS loss function is augmented in such a way that we not only minimize the sum of squared residuals but also penalize the size of parameter estimates, in order to shrink them towards zero:

$$L_{ridge}(\hat{\beta}) = \sum_{i-1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda\sum_{j-1}^{m}\hat{\beta}_j^2 = ||y - X\hat{\beta}||^2 + \lambda||\hat{\beta}||^2.$$

Lasso: -

Lasso, or Least Absolute Shrinkage and Selection Operator, is quite similar conceptually to ridge regression. It also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients (the so-called L2 penalty), lasso penalizes the sum of their absolute values (L1 penalty). As a result, for high values of λ, many coefficients are exactly zeroed under lasso, which is never the case in ridge regression.

Under lasso, the loss is defined as:

$$L_{lasso}(\hat{\beta}) = \sum_{i-1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda\sum_{j-1}^{m}|\hat{\beta}_j|.$$

Elastic Net:-

Elastic Net first emerged as a result of critique on lasso, whose variable selection can be too dependent on data and thus unstable. The solution is to combine the penalties of ridge regression and lasso to get the best of both worlds. Elastic Net aims at minimizing the following loss function:

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i-1}^{m}(y_i - x_i'\hat{\beta})^2}{2n} + \lambda(\frac{1-\alpha}{2}\sum_{j-1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j-1}^{m}|\hat{\beta}_j|),$$

where α is the mixing parameter between ridge ($\alpha = 0$) and lasso ($\alpha = 1$).

K-Fold cross validation:-

Cross-validation is a statistical method used to estimate the skill of machine learning models.It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. K-Fold CV is where a given data set is split into a ***K*** number of sections/folds . Each fold is then used once as a validation while the k - 1 remaining folds form the training set. The algorithm is trained and tested K times, each time a new set is used as testing set while remaining sets are used for training. Finally, the result of the K-Fold Cross-Validation is the average of the results obtained on each set.

Grid Search:-

Hyper parameters are hugely important in getting good performance with models. In order to understand this process, we first need to understand the difference between a model parameter and a model hyper parameter. Model parameters are internal to the model whose values can be estimated from the data and we are often trying to estimate them as best as possible. whereas hyper parameters are external to our model and cannot be directly learned from the regular training process. These parameters express "higher-level" properties of the model such as its complexity or how fast it should learn. Hyper parameters are model-specific properties that are 'fixed' before you even train and test your model on data. The process for finding the right hyper parameters is still somewhat of a dark art, and it currently involves either random search or grid search across Cartesian products of sets of hyper parameters. Grid Search takes a dictionary of all of the different hyper parameters that you want to test, and then feeds all of the different combinations through the algorithm for you and then reports back to you which one had the highest accuracy.

2.2.3 Decision Tree

Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables.

Types of Decision Trees

Types of decision tree is based on the type of target variable we have. It can be of two types:

Categorical Variable Decision Tree: Decision Tree which has categorical target variable then it called as categorical variable decision tree.

Continuous Variable Decision Tree: Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.

Regression Tree:-

1.Regression trees are used when dependent variable is continuous.

2. In case of regression tree, the value obtained by terminal nodes in the training data is the mean response of observation falling in that region. Thus, if an unseen data observation falls in that region, we'll make its prediction with mean value.

Classification Tree:-

1. Classification trees are used when dependent variable is categorical.
2. In case of classification tree, the value (class) obtained by terminal node in the training data is the mode of observations falling in that region. Thus, if an unseen data observation falls in that region, we'll make its prediction with mode value.

Entropy is the measures of impurity**,** disorder or uncertainty in a bunch of examples.

Information gain (IG) measures how much "information" a feature gives us about the class. An attribute with highest Information gain will tested/split first.

```
> #Decision Tree
> fit = rpart(cnt ~ ., data = train, method = "anova")
> fit
n= 573

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

 1) root 573 2166402000 4554.555
   2) atemp< -0.2782751 236  571808600 3127.508
     4) yr=0 121  118835500 2233.339
       8) season=1,2 83   27960660 1758.542 *
       9) season=4 38   31295580 3270.395 *
     5) yr=1 115  254437000 4068.330
      10) season=1 58   59373540 3156.034
        20) atemp< -1.112213 32   22888610 2589.938 *
        21) atemp>=-1.112213 26   13608600 3852.769 *
      11) season=2,3,4 57   97671690 4996.632
        22) hum>=0.9614467 8   17732040 3317.625 *
        23) hum< 0.9614467 49   53705100 5270.755 *
   3) atemp>=-0.2782751 337  777422400 5553.911
     6) yr=0 156  116982200 4269.872
      12) hum>=1.828909 13    5317775 2681.615 *
      13) hum< 1.828909 143   75889950 4414.259 *
     7) yr=1 181  181553800 6660.597
      14) hum>=0.7998345 31   43352560 5524.258 *
      15) hum< 0.7998345 150   89899300 6895.440 *
> predictions_DT = predict(fit, test[,-9])
> library(Metrics)
> rmse(test[,9],predictions_DT)
[1] 956.3683
```

2.2.4 Random Forest

Random forest is one of the most popular and most powerful machine learning algorithm. Random forest or random decision forest are an ensemble learning method for classification, regression and other tasks that operates by constructing multitude of decision trees at training time and outputting the class that is the mode of the classes(classification) and mean prediction(regression) of the individual trees .Random decision forests correct for decision trees' habit of overfitting to their training set.

Ensemble learning, in general, is a model that makes predictions based on a number of different models. By combining individual models, the ensemble model tends to be more flexible.

Two most popular ensemble methods are bagging and boosting.

1.Bagging: Training a bunch of individual model in parallel way. Each model is trained by a random subset of the data.

2.Boosting: Training a bunch of individual models in a sequential way. Each individual model learns from mistakes made by the previous model.

```
> RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 500)
> RF_model

Call:
 randomForest(formula = cnt ~ ., data = train, importance = TRUE,      ntr
ee = 500)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 489921.2
                    % Var explained: 87.04
> treeList = RF2List(RF_model)
> exec = extractRules(treeList, train[,-9])
4437 rules (length<=6) were extracted from the first 100 trees.
> exec[1:2,]
[1] "X[,1] %in% c('1') & X[,3] %in% c('1','2','12') & X[,6]<=-1.2996577963
5625 & X[,6]<=-1.38439326688248 & X[,7]<=1.36996265781195 & X[,8]<=-0.0653
127182392825"
[2] "X[,1] %in% c('1') & X[,3] %in% c('1','2','12') & X[,6]<=-1.2996577963
5625 & X[,6]>-1.38439326688248 & X[,7]<=1.36996265781195 & X[,8]<=-0.06531
27182392825"
> readableRules = presentRules(exec, colnames(train))
> readableRules[1:2,]
[1] "season %in% c('1') & mnth %in% c('1','2','12') & atemp<=-1.2996577963
5625 & atemp<=-1.38439326688248 & hum<=1.36996265781195 & windspeed<=-0.06
53127182392825"
[2] "season %in% c('1') & mnth %in% c('1','2','12') & atemp<=-1.2996577963
5625 & atemp>-1.38439326688248 & hum<=1.36996265781195 & windspeed<=-0.065
```

```
> ruleMetric = getRuleMetric(exec, train[,-9], train$cnt)
> ruleMetric[1:2,]
     len freq    err
[1,] "6" "0.024" "245713.107142857"
[2,] "6" "0.005" "285216"
     condition


[1,] "X[,1] %in% c('1') & X[,3] %in% c('1','2','12') & X[,6]<=-1.299657796
35625 & X[,6]<=-1.38439326688248 & X[,7]<=1.36996265781195 & X[,8]<=-0.065
3127182392825"
[2,] "X[,1] %in% c('1') & X[,3] %in% c('1','2','12') & X[,6]<=-1.299657796
35625 & X[,6]>-1.38439326688248 & X[,7]<=1.36996265781195 & X[,8]<=-0.0653
127182392825"
     pred
[1,] "1457.5"
[2,] "2435"
> rf_prd=predict(RF_model,test[,-9])
> summary(rf_prd)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1403    3377    4324    4423    5293    7590
> library(Metrics)
> rmse(test[,9],rf_prd)
[1] 664.2774
```

Chapter 3

3. Conclusion

3.1 Model Evaluation

Evaluating the model accuracy is an essential part of the process in creating machine learning models to describe how well the model is performing in its predictions. As it is a time series data we use RMSE for the evaluation purpose.The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. RMSE is an absolute measure of fit.

Model	RMSE
Multiple linear regression	780.63
Decision Tree	816.54
Random Forest	652.11

3.2 Model Selection

Various models are created and trained using different types of algorithm. We have created model using K fold cross validation technique, grid search(hyper parameter tuning).We have also calculated the error metrics in terms of RMSE to evaluate the efficiency of the models. RMSE rate of random forest regression is relatively less than the other model.

Appendix A – Code

R code

```
#set the working directory
setwd("G:/edwisor_project2")
getwd()
#load the data
dataset=read.csv("day.csv",header = T,na.strings = c(" ","",NA))
str(dataset)
dim(dataset)


#factorization of data
factor=c("season","yr","mnth","holiday","weekday","workingday","weathersit")
for(i in factor){
  dataset[,i]=as.factor(dataset[,i])
}

#Dropping of casual and registered count
dataset=dataset[,-c(14,15)]

#Distributionn of data
library(ggplot2)

ggplot(data=dataset, aes_string(x = dataset$atemp, y = dataset$cnt)) +
  geom_point(aes_string(colour = dataset$season),size = 2) +
  theme_bw()+ ylab("cnt") + xlab("atemp") + ggtitle("atemp vs count") +
  theme(text=element_text(size=10)) +
  scale_x_continuous(breaks = scales::pretty_breaks(5)) +
  scale_y_continuous(breaks = scales::pretty_breaks(5))+
  scale_colour_discrete(name="season")


ggplot(data=dataset, aes_string(x = dataset$windspeed, y = dataset$cnt)) +
  geom_point(aes_string(colour = dataset$season),size = 2) +
  theme_bw()+ ylab("cnt") + xlab("windspeed") + ggtitle("windspeed vs count") +
  theme(text=element_text(size=10)) +
  scale_x_continuous(breaks = scales::pretty_breaks(5)) +
  scale_y_continuous(breaks = scales::pretty_breaks(5))+
  scale_colour_discrete(name="season")


ggplot(data=dataset, aes_string(x = dataset$temp, y = dataset$cnt)) +
  geom_point(aes_string(colour = dataset$season),size = 2) +
  theme_bw()+ ylab("cnt") + xlab("temp") + ggtitle("temp vs count") +
```

```
  theme(text=element_text(size=10)) +
  scale_x_continuous(breaks = scales::pretty_breaks(5)) +
  scale_y_continuous(breaks = scales::pretty_breaks(5))+
  scale_colour_discrete(name="season")

ggplot(data=dataset, aes_string(x = dataset$hum, y = dataset$cnt)) +
  geom_point(aes_string(colour = dataset$season),size = 2) +
  theme_bw()+ ylab("cnt") + xlab("hum") + ggtitle("hum vs count") +
  theme(text=element_text(size=10)) +
  scale_x_continuous(breaks = scales::pretty_breaks(5)) +
  scale_y_continuous(breaks = scales::pretty_breaks(5))+
  scale_colour_discrete(name="season")

#ggplot(dataset, aes(x=dataset$season)) +

#geom_bar(position="dodge")+theme_bw()+ggtitle("month vs working day")


ggplot(dataset, aes(x = atemp, y = cnt, color = weekday)) +

geom_smooth(method = "loess", fill = NA, size = 1) +

theme_light(base_size = 11) +

xlab("atemp") +

ylab("count of Bike Rentals") +

ggtitle("count VS atemp") +



scale_color_discrete(name = "Weekday:",

                     breaks = c(1, 2, 3, 4, 5, 6, 7),

                     labels = c("sun","mon","tue","wed","thurs","fri","sat"))+

theme(plot.title = element_text(size = 11, face="bold"))

ggplot(dataset,aes(x=season,y=cnt,fill=season))+
geom_boxplot(outlier.color ="red",outlier.size = 3)+ggtitle("season vs count")




#checking of missing value
```

```
sum(is.na(dataset))

#saving numeric varible in cnames
numeric_index = sapply(dataset,is.numeric)
numeric_data = dataset[,numeric_index]
cnames = colnames(numeric_data)
cnames

#saving categorical variable in cnames1
cnames1=c("temp","atemp","hum","windspeed")

#outlier plot
library(ggplot2)
for (i in 1:length(cnames1)) {
  assign(paste0("gn",i), ggplot(aes_string( y = (cnames1[i]), x= "cnt") , data = subset(dataset)) +
         stat_boxplot(geom = "errorbar" , width = 0.5) +
         geom_boxplot(outlier.color = "red", fill = "grey", outlier.shape = 20, outlier.size = 1, notch =
FALSE)+
         theme(legend.position = "bottom")+
         labs(y = cnames1[i], x= "cnt")+
         ggtitle(paste("Boxplot" , cnames1[i])))
  #print(i)
}
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,ncol=3,nrow=2)

#removing of outlier from dataset
for(i in cnames1){
val = dataset[,i][dataset[,i] %in% boxplot.stats(dataset[,i])$out]
print(length(val))
dataset=dataset[which(!dataset[,i] %in% val),]
}

#corrlation plotting
library(corrgram)
corrgram(dataset[,cnames1], order = F,
       upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
attach(dataset)

#ANOVA Test
m1=aov(cnt~season+yr+mnth+holiday+weekday+workingday+weathersit)
summary(m1)


#standardization
for(i in cnames1){
   print(i)
   dataset[,i] = (dataset[,i] - mean(dataset[,i]))/
                          sd(dataset[,i])
```

```
}
View(dataset)
#Final dataset after dropping of certain features
dataset=subset(dataset,select=-c(instant,dteday,holiday,temp,weekday))

library(rpart)
library(MASS)
set.seed(123)
train_index = sample(1:nrow(dataset), 0.8 * nrow(dataset))
train = dataset[train_index,]
test = dataset[-train_index,]

# linear regression model
lm_mod=lm(cnt~.,train)
summary(lm_mod)
lm_prd=predict(lm_mod,test[,-9])
lm_prd
library(Metrics)
rmse(test[,9],lm_prd)


#kFold cross validation
library(caret)
custom <-trainControl(method="repeatedcv",
                      number=10,
                      repeats=5,
                      verboseIter=T)

set.seed(123)
lm <-train(cnt ~.,train,method ='lm',trControl=custom)
lm$results
lm
summary(lm)


#Ridge Regression
set.seed(123)

library(glmnet)
ridge <-train(cnt~.,
              train,
              method='glmnet',
              tuneGrid=expand.grid(alpha=0,lambda=seq(0.0001,2,length=10)),
              trControl=custom)
plot(ridge)
ridge
plot(varImp(ridge,scale = T))
```

```
#lasso regression
set.seed(123)
lasso <-train(cnt~.,
          train,
          method='glmnet',
          tuneGrid=expand.grid(alpha=1,lambda=seq(0.0001,0.2,length=5)),
          trControl=custom)
lasso
plot(varImp(lasso,scale = T))

#elastic
set.seed(123)
en <-train(cnt~.,
          train,
          method='glmnet',
          tuneGrid=expand.grid(alpha=seq(0,1,length=10),lambda=seq(0.0001,1,length=5)),
          trControl=custom)
en

model_list <-list(LinearModel=lm,Ridge=ridge,Lasso=lasso,ElasticNet=en)
res <-resamples(model_list)
summary(res)
xyplot(res,metric = 'RMSE')
en$bestTune




#Decision Tree
fit = rpart(cnt ~ ., data = train, method = "anova")
fit
predictions_DT = predict(fit, test[,-9])
predictions_DT
library(Metrics)
rmse(test[,9],predictions_DT)



#Random forest
library(randomForest)
library(ggplot2)
library(inTrees)
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 500)
RF_model
```

```
treeList = RF2List(RF_model)
exec = extractRules(treeList, train[,-9])
exec[1:2,]
readableRules = presentRules(exec, colnames(train))
readableRules[1:2,]
ruleMetric = getRuleMetric(exec, train[,-9], train$cnt)
ruleMetric[1:2,]
rf_prd=predict(RF_model,test[,-9])
rf_prd
summary(rf_prd)
library(Metrics)
rmse(test[,9],rf_prd)


#xg boost

features=c("season","yr","mnth","weekday","workingday","weathersit","atemp","hum","windspeed
")
xgb_mod <- xgboost(data = data.matrix(train[,features]),label =
train$cnt,objective="reg:linear",eval_metric="rmse",max.depth=5,nround = 10)
y_pred = predict(xgb_mod, newdata = data.matrix(test[,features]))
y_pred
library(Metrics)
rmse(test$cnt,y_pred)
```

Python code

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform

os.chdir("G:\edwisor_project2") //Set the directory path
os.getcwd()
dataset=pd.read_csv("day.csv") //Load the data
dataset.shape
dataset.head(10) # First 10 row
dataset.describe() # about the dataset
dataset = dataset.rename(columns = {'dteday':'datetime','yr':'year',
'mnth':'month','hum':'humidity','cnt':'count'})  #Rename of columns
dataset.dtypes # datatypes of columns
dataset['season'] = dataset['season'].astype('category')
dataset['year'] = dataset['year'].astype('category')
dataset['month'] = dataset['month'].astype('category')
dataset['holiday'] = dataset['holiday'].astype('category')
dataset['weekday'] = dataset['weekday'].astype('category')
dataset['workingday'] = dataset['workingday'].astype('category')
dataset['weathersit'] = dataset['weathersit'].astype('category')    # conversion of datatypes of
columns

dataset.dtypes
Visualizations
1.fig, ax = plt.subplots()
sns.barplot(data=dataset[['season','count']],
        x='season',
        y='count',
        ax=ax)

plt.title('count by Season')
plt.ylabel('count')
plt.xlabel('Season')

tick_val=[0, 1, 2, 3]
tick_lab=['Spring','Summer','Fall',"winter"]
plt.xticks(tick_val, tick_lab)

plt.show()

2.fig, ax = plt.subplots()
sns.barplot(data=dataset[['month','count']], x='month', y='count', ax=ax)
```

```
plt.title('count by month')
plt.ylabel('count')
plt.xlabel('Month')

tick_val=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
tick_lab=['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October',
'November', 'December']
plt.xticks(tick_val, tick_lab)

plt.show()


3.by_week = dataset.groupby(['weekday'])['count'].sum().reset_index()
ax = sns.barplot(x = by_week['weekday'], y = by_week['count'])
ax.set(xlabel='weekday', ylabel='count')
plt.show()

4.fig, ax = plt.subplots(figsize=(7,7))
sns.pointplot(data=dataset[['month', 'count']],
        x='month',
        y='count',
        ax=ax,
        color='red')
tick_val=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
tick_lab=['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October',
'November', 'December']
plt.xticks(tick_val, tick_lab)

plt.title('Count by Month')
plt.ylabel('count')
plt.xlabel('Month')

plt.show()
#Missing value
missing_val = pd.DataFrame(dataset.isnull().sum())
missing_val

#Visualization(Outliers)
1.%matplotlib inline
plt.boxplot(dataset['humidity'])

2.%matplotlib inline
plt.boxplot(dataset['windspeed'])

3.%matplotlib inline
plt.boxplot(dataset['temp'])
```

```
4.%matplotlib inline
plt.boxplot(dataset['atemp'])
# Saving continuous feature in cnames
cnames=['temp','atemp','humidity','windspeed']
# Outlier detection
1.q75, q25 = np.percentile(dataset['humidity'], [75 ,25])
iqr = q75 - q25
min = q25 - (iqr*1.5)
max = q75 + (iqr*1.5)
dataset = dataset.drop(dataset[dataset['humidity'] < min].index)
dataset = dataset.drop(dataset[dataset['humidity'] > max].index)

2.q75, q25 = np.percentile(dataset['windspeed'], [75 ,25])
iqr = q75 - q25
min = q25 - (iqr*1.5)
max = q75 + (iqr*1.5)
dataset = dataset.drop(dataset[dataset['windspeed'] < min].index)
dataset = dataset.drop(dataset[dataset['windspeed'] > max].index)
#Rechecking the rows and columns
dataset.shape
#Correlation plot
df_corr = dataset.loc[:,cnames]
f, ax = plt.subplots(figsize=(10, 10))
corr = df_corr.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),annot=True,
cmap=sns.diverging_palette(220, 10, as_cmap=True),
        square=True, ax=ax)
#Saving categorical features in cat_names
cat_names=['season','year','month','holiday','weekday','workingday','weathersit']
#ANOVA test
from statsmodels.formula.api import ols
import statsmodels.api as sm
anova=ols('count~season+year+month+holiday+weekday+workingday+weathersit',data=dataset).fit(
)
anova_table=sm.stats.anova_lm(anova,typ=2)
anova_table
#standardization
for i in cnames:
  print(i)
  dataset[i] = (dataset[i] - dataset[i].mean())/dataset[i].std()
#Final dataset after dropping of some features
dataset = dataset.drop(['instant', 'datetime', 'temp','holiday', 'casual', 'registered','weekday'], axis=1)

dataset_logit = pd.DataFrame(dataset['count'])
colnames=['atemp','humidity','windspeed']
dataset_logit = dataset_logit.join(dataset[colnames])
dataset_logit.head()
cat_names = ["season", "year", "month", "workingday", "weathersit"]
```

```
for i in cat_names:
   temp = pd.get_dummies(dataset[i], prefix = i)
   dataset_logit = dataset_logit.join(temp)
dataset_logit

from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
#spiliting of data
X = dataset_logit.iloc[:, 1:28]
y = dataset_logit.iloc[:,0]

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.2,random_state = 0)

#Linear regression model
from sklearn.linear_model import LinearRegression
lm_model=LinearRegression().fit(X_train, y_train)
print(lm_model.intercept_)
print(lm_model.coef_)
lm_predict=lm_model.predict(X_test)

from sklearn import metrics
print(np.sqrt(metrics.mean_squared_error(y_test,lm_predict)))

#Ridge regression model
from sklearn.linear_model import Ridge

ridgeReg = Ridge(alpha=0.01, normalize=True)

ridgeReg.fit(X_train,y_train)
print(ridgeReg.coef_)
pred = ridgeReg.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(y_test,pred)))
#Lasso regression model
from sklearn.linear_model import Lasso

lassoReg = Lasso(alpha=0.01, normalize=True)

lassoReg.fit(X_train,y_train)

pred = lassoReg.predict(X_test)
print(lassoReg.coef_)
print(np.sqrt(metrics.mean_squared_error(y_test,pred)))
#K-Fold cross validation
from sklearn.model_selection import cross_val_score
results=cross_val_score(estimator=lassoReg,X=X_train,y=y_train,cv=10,scoring='neg_mean_squared
_error')
```

```
print(results.mean())
print("MSE: %.3f (%.3f)" % (results.mean(), results.std()))
#Grid search
from sklearn.model_selection import GridSearchCV
parameters=[{'alpha':[0.0001,0.001,0.01,.05,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5]}]
grid_search=GridSearchCV(estimator=lassoReg,param_grid=parameters,cv=10)
grid_search=grid_search.fit(X_train, y_train)
best_parameters=grid_search.best_params_
best_parameters
y_pred=grid_search.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
#Decision tree model
from sklearn.tree import DecisionTreeRegressor
fit_DT = DecisionTreeRegressor(max_depth=10).fit(X_train, y_train)
predictions_DT = fit_DT.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(y_test,predictions_DT)))

#Random forest
from sklearn.ensemble import RandomForestRegressor
rf_model=RandomForestRegressor(n_estimators=500).fit(X_train, y_train)
predictions_RF=rf_model.predict(X_test)
print(np.sqrt(metrics.mean_squared_error(y_test,predictions_RF)))
```