Project Name - Bike Rental Count

By: Deepanshu Vashisth

Contents

Chapter 1

Introduction

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

Our task is to build Regression Models which will predict the count of bike rental depending on various factors such as temperature, humidity, weekday, etc.

Given below is a sample of the data set that we are using to predict the rental counts:

Table 1.1: Bike Rental Data Sample(Columns: 1-8)

instant	dteday	season	yr	mnth	holiday	weekday	workingday
1	01-01-2011	1	0	1	0	6	0
2	02-01-2011	1	0	1	0	0	0
3	03-01-2011	1	0	1	0	1	1
4	04-01-2011	1	0	1	0	2	1

Table 1.2: Bike Rental Data Sample(Columns: 9-16)

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562

As you can see in the table below we have the following 15 variables, using which we have to correctly predict the Bike rental:

Table 1.3: Predictor Variables

S.No	Predictor
1	instant
2	dteday
3	season
4	yr
5	mnth
6	holiday
7	weekday
8	workingday
9	weathersit
10	temp
11	atemp
12	hum
13	windspeed
14	casual
15	registered

The details of data attributes in the dataset 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)

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 values are divided to 67 (max)

casual: count of casual users

registered: count of registered users

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

Chapter 2

Methodology

2.1 Pre Processing

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.

2.1.1 Taking Care of Date

In the dataset we have an Date variable(dteday). Since machine learning model doesn’t accept date variable, so we have divided the date in day, month and year variables and since month and year variables are already present(mnth, yr) thus, only day variable is replaced with dteday.

R Code:

```
data$dteday = as.Date(data$dteday, "%Y-%m-%d")

data$dteday = as.numeric(format(data$dteday, "%d"))
```

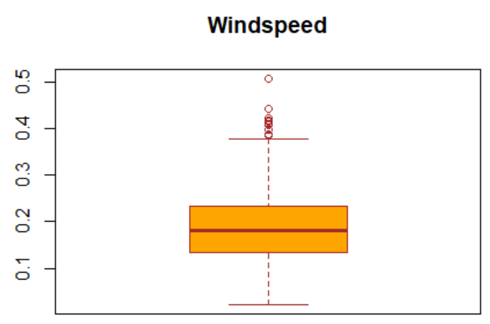
Python Code:

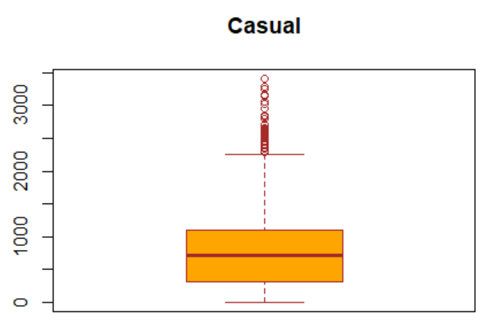
```
data['dteday'] = pd.to_datetime(data['dteday'], format = "%Y-%m-%d")

data['dteday'] = data['dteday'].dt.day
```

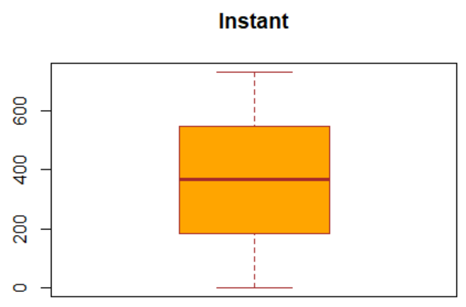
2.1.2 Outlier Analysis

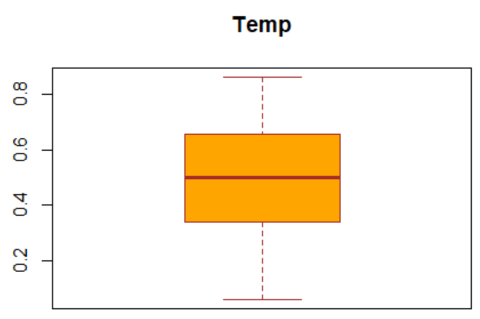
An outlier is an element of a data set that distinctly stands out from the rest of the data. In other words, outliers are those data points that lie outside the overall pattern of distribution.

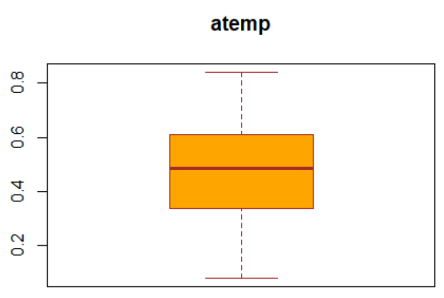
The easiest way to detect outliers is to create a graph. Plots such as Box plots, Scatterplots and Histograms can help to detect outliers. Alternatively, we can use mean and standard deviation to list out the outliers. Interquartile Range and Quartiles can also be used to detect outliers.

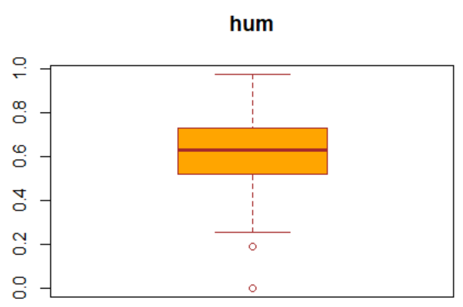


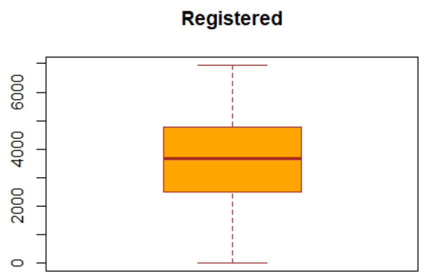












As you can see in above box plots, only casual and windspeed have many outliers and hum has only 2 outliers.

The below code is used for outlier removal:

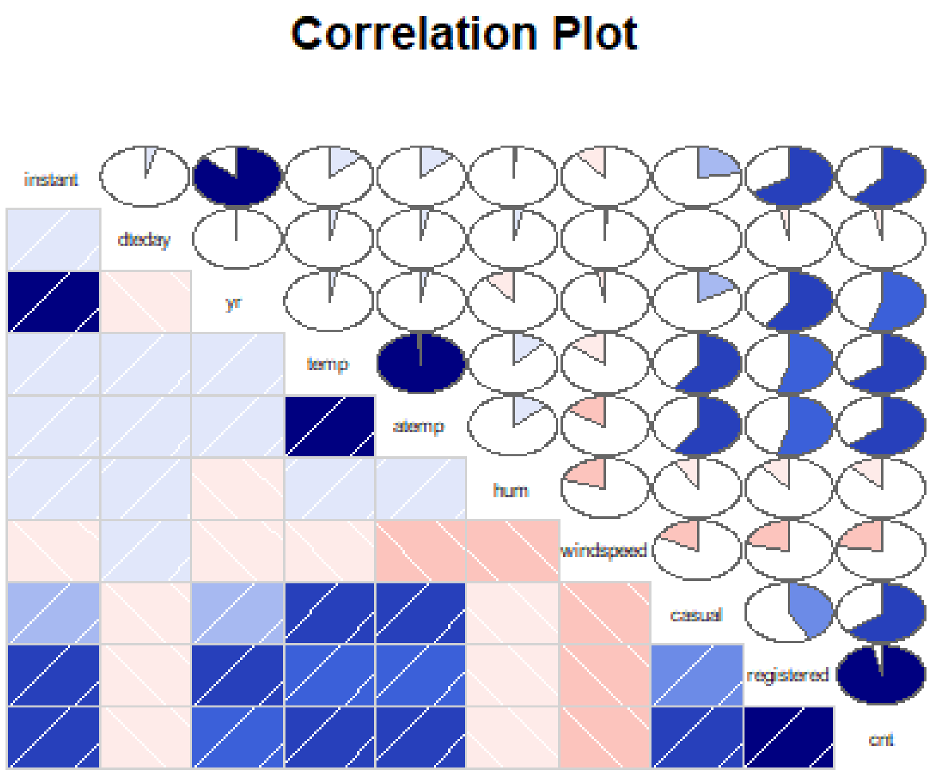
```
R Code:

num_in = sapply(data, is.numeric)

num_in = data[,num_in]

cname = colnames(num_in)


for(i in cname){

  print(i)

  val = data[,i][data[,i]%in%boxplot.stats(data[,i])$out]

  data = data[which(!data[,i]%in%val),]

}
```

```
Python Code:


Q1 = data.quantile(0.25)

Q3 = data.quantile(0.75)

IQR = Q3 - Q1


data = data[~((data < (Q1 - 1.5 * IQR)) |(data > (Q3 + 1.5 *
IQR))).any(axis=1)]
```

2.1.3 Feature Selection

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

There are several methods of doing that. Below we have used corrgram plot in R and Pearson Correlation in Python.



As you can see in the above plot that dteday, hum and windspeed are least correlated with dependent variable cnt, thus can be dropped.

And there is multicollinearity between variables (instant and yr) and (temp and atemp)

Thus, variables dteday, hum, windspeed, instant and atemp can be dropped from dataset.

2.2 Modelling

2.2.1 Model Selection

Since our dependent variable is continuous thus only Regression Analysis can be done on the dataset. The models we'll be using are:

- Decision Tree
- Random Forest
- Multiple Linear Regression
- K-Nearest Neighbour

2.2.2 Decision Tree

Decision tree is one of the predictive modelling approaches used in statistics, data mining and machine learning.

Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric **supervised learning** method used for both **classification** and **regression** tasks.

In R we have used the **rpart** library for Decision Tree regression and DecisionTreeRegressor from sklearn in Python.

2.2.3 Random Forest

The random forest is a Machine learning algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

We have used randomForest library in R and RandomForestRegressor from sklearn.ensemble in Python.

2.2.4 Multiple Linear Regression

Multiple linear regression (MLR/multiple regression) is a statistical technique. It can use several variables to predict the outcome of a different variable. The goal of multiple regression is to model the linear relationship between your independent variables and your dependent variable.

We have used lm in R and LinearRegression from sklearn.linear_model in Python.

2.2.5 K-Nearest Neighbour

K-Nearest Neighbors (KNN) is one of the simplest algorithms used in Machine Learning for regression and classification problem. KNN algorithms use data and classify new data points based on similarity measures (e.g. distance function). Classification is done by a majority vote to its neighbors. The data is assigned to the class which has the nearest neighbors. As you increase the number of nearest neighbors, the value of k, accuracy might increase.

In R knnreg from class library is used and in Python KNeighborsRegressor from sklearn.neighbors.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

We will use Predictive performance as the criteria to compare and evaluate models. Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

3.1.1 Mean Absolute Percentage Error

The mean absolute percentage error (**MAPE**), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning.

```
> mape(test[,11], dt_predict)
[1] 12.07285

> mape(test[,11], rf_predict)
[1] 6.052475

> mape(test[,11], lr_predict)
[1] 1.696196e-13

> mape(test[,11], knn_predict)
[1] 1.385023
```

3.2 Model Selection

As we can see from MAPE result the least error percentage is in Multiple Linear Regression model, thus the best model for this data is Multiple Linear Regression.

Appendix A – R Code

```
data = read.csv('day.csv')

#Taking care of date

data$dteday = as.Date(data$dteday, "%Y-%m-%d")

data$dteday = as.numeric(format(data$dteday, "%d"))

#Conversion of variables

data$holiday = as.factor(data$holiday)

data$weekday = as.factor(data$weekday)

data$workingday = as.factor(data$workingday)

data$weathersit = as.factor(data$weathersit)

data$season = as.factor(data$season)

data$mnth = as.factor(data$mnth)

#Outlier Analysis usinmg boxplot

boxplot(data$windspeed, main = "Windspeed", col = "orange", border = "brown")

boxplot(data$registered, main = "Registered", col = "orange", border = "brown")

boxplot(data$casual, main = "Casual", col = "orange", border = "brown")

boxplot(data$hum, main = "Hum", col = "orange", border = "brown")

boxplot(data$temp, main = "Temp", col = "orange", border = "brown")

boxplot(data$atemp, main = "Atemp", col = "orange", border = "brown")

boxplot(data$instant, main = "Instant", col = "orange", border = "brown")

num_in = sapply(data, is.numeric)

num_in = data[,num_in]

cname = colnames(num_in)

for(i in cname){

  print(i)
  val = data[,i][data[,i]%in%boxplot.stats(data[,i])$out]

  data = data[which(!data[,i]%in%val),]

}

#Feature selection

library(corrgram)

corrgram(data[,cname], order = F, upper.panel = panel.pie, text.panel = panel.txt, main
= 'Correlation Plot')

data = subset(data, select = -c(instant, atemp, hum, windspeed, dteday))

#Splitting into test and train

library('caret')

set.seed(1234)

train.index = createDataPartition(data$cnt,p=0.80,list=FALSE)

train = data[train.index,]

test = data[-train.index,]

  #Mean Absolute Percentage Error

mape = function(y_true, y_pred){

  mean(abs((y_true-y_pred)/y_true))*100

}

#Decision Tree

library(rpart)

dt_model = rpart(cnt~., data = train, method = 'anova')


dt_predict = predict(dt_model, test[,-14])


mape(test[,11], dt_predict)

#mape = 12.07
```

```
#Random Forest

library(randomForest)

rf_model = randomForest(cnt~., train, importance = TRUE, ntree = 300)

rf_predict = predict(rf_model, test[,-11])

mape(test[,11], rf_predict)

#mape = 6.45


#Linear Regression

lr_test = subset(train, select = c(yr, mnth, temp, casual, registered, cnt))

lr_model = lm(cnt~., data = lr_test)

lr_predict = predict(lr_model, test[,-11])

mape(test[,11], lr_predict)

#mape = 1.63e-13


#Knn Model

library(class)

knn_model = knnreg(cnt~., train, k=3)

knn_predict = predict(knn_model, test[,-11])

mape(test[,11], knn_predict)

#mape = 1.37
```

Appendix B – Python Code

```
import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

data = pd.read_csv('day.csv')

#Taking care of date

data['dteday'] = pd.to_datetime(data['dteday'], format = "%Y-%m-%d")

data['dteday'] = data['dteday'].dt.day

#Conversion of variables

#data['day'] = data['day'].astype('category')

data['season'] = data['season'].astype('category')

data['mnth'] = data['mnth'].astype('category')

data['holiday'] = data['holiday'].astype('category')

data['weekday'] = data['weekday'].astype('category')

data['weathersit'] = data['weathersit'].astype('category')

#Outlier Analysis

sns.boxplot(data=data['casual'])

sns.boxplot(data=data[‘registered’])

sns.boxplot(data=data['instant'])

sns.boxplot(data=data['temp'])

sns.boxplot(data=data['atemp'])

sns.boxplot(data=data['hum'])

sns.boxplot(data=data['windspeed'])
```

```
Q1 = data.quantile(0.25)

Q3 = data.quantile(0.75)

IQR = Q3 - Q1

data = data[~((data < (Q1 - 1.5 * IQR)) |(data > (Q3 + 1.5 * IQR))).any(axis=1)]

#Feature selection

#Using Pearson Correlation

plt.figure(figsize=(12,10))

cor = data.corr()

sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)

plt.show()

#Correlation with output variable

cor_target = abs(cor["cnt"])

#Selecting highly correlated features

relevant_features = cor_target[cor_target>0.5]

relevant_features

#Droping low correlatedvariables

del data['hum']

del data['windspeed']

del data['dteday']

#Dropping multicollinear variables

del data['instant']

del data['atemp']

#Splitting in train and test

x = data.values[:,0:10]

y = data.values[:,10]
```

```
x = pd.DataFrame(x)

y = pd.DataFrame(y)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)


#Function for calculating Mean Absolute Percentage Error

def MAPE(y_true, y_pred):

   y_true = np.array(y_true)

   y_pred = np.array(y_pred)

   mape = np.mean(np.abs((y_true-y_pred)/y_true))*100

   return mape


#Decision Tree

from sklearn import tree

dt_model = tree.DecisionTreeRegressor(max_depth=1).fit(x_train, y_train)

dt_predicts = dt_model.predict(x_test)

MAPE(y_test, dt_predicts)

#mape = 68.14



#Random Forest

from sklearn.ensemble import RandomForestRegressor

RF_model = RandomForestRegressor(n_estimators = 100).fit(x_train, y_train)

rf_predict = RF_model.predict(x_test)

MAPE(y_test, rf_predict)

#mape = 71.20
```

```
#Linear Regression

from sklearn.linear_model import LinearRegression

regressor = LinearRegression().fit(x_train, y_train)

lr_predicts = regressor.predict(x_test)

MAPE(y_test, lr_predicts)

#mape = 5.784490670664497e-14




#KNN Algorithm

from sklearn.neighbors import KNeighborsRegressor

knn_model = KNeighborsRegressor(n_neighbors = 4).fit(x_train, y_train)

knn_predicts = knn_model.predict(x_test)

MAPE(y_test, knn_predicts)

#mape = 1.43
```