PROJECT – BIKE RENTING

By Anchit Chaturvedi

INDEX

INTRODUCTION

Problem statement:
You are provided with a historical dataset, on the daily count of bike rentals, based on various environmental and seasonal times. Using this data, you need to predict the count of bike rentals for a day based on the environmental and seasonal conditions of that particular day.

Data:
The task is to predict the bike rentals for a day based on the environmental settings and the atmospheric conditions of the day. The various variables that are available to us are the following:

- **instant:** This is nothing but the index of the record in the data.
- **dteday:** It is the date on which the data was recorded.
- **Season:** It is the season in which the record was recorded. It is a categorical data type. It has the following 4 values:
    1. Spring
    2. Summer
    3. Fall
    4. Winter
- **Yr:** It is the year in which the data was recorded. It is a categorical data, which has just two values:
    0. 2011
    1. 2012
- **mnth:** It is the month of recording the data. It is also of categorical type, which has 12 values corresponding to the various months.
- **Holiday:** Tells whether the day was a holiday or not.
- **Weekday:** Tells which day of the week was the particular record taken on. It is a categorical variable, and the various values correspond to the various days of the week.
- **Workingday:** Tells whether the particular day was a working day or a weekend.
- **Weathersit:** Tells the weather condition of the particular day. It is a categorical variable and the various values are as follows:
    1 Clear, few clouds, partly cloudy.
    2 Mist, Mist + Cloudy, Mist + broken clouds, Mist + few clouds.
    3 Light snow, Light rain + thunderstorm + scattered clouds, Light rain + scattered clouds
    4 Heavy rain + ice palette + thunderstorm + Mist, snow + fog
- **Temp:** It is the normalized temperature in degree Celsius, using the formula:
  $(t - t\_min)/(t\_max - t\_min)$,
  where $t\_min = -8$ and $t\_max = 39$.
- **Atemp:** It is the normalized feeling temperature in degree Celsius. It is derived using the formula:
  $(t - t\_min)/(t\_max - t\_min)$
  where, $t\_min = -16$ and $t\_max = 50$.
- **Hum:** It is the normalized value of humidity, obtained by dividing all the values by 100.
- **Windspeed:** It is the value of the normalized windspeed on that particular day, obtained by dividing all the values by 67.

- **Casual:** It is the number of casual users, renting the bikes on that day.
- **Registered:** It is the number of registered users, renting the bikes on that day.
- **Count:** It is the total number of all the users renting bike on the day, including both registered and casual users.

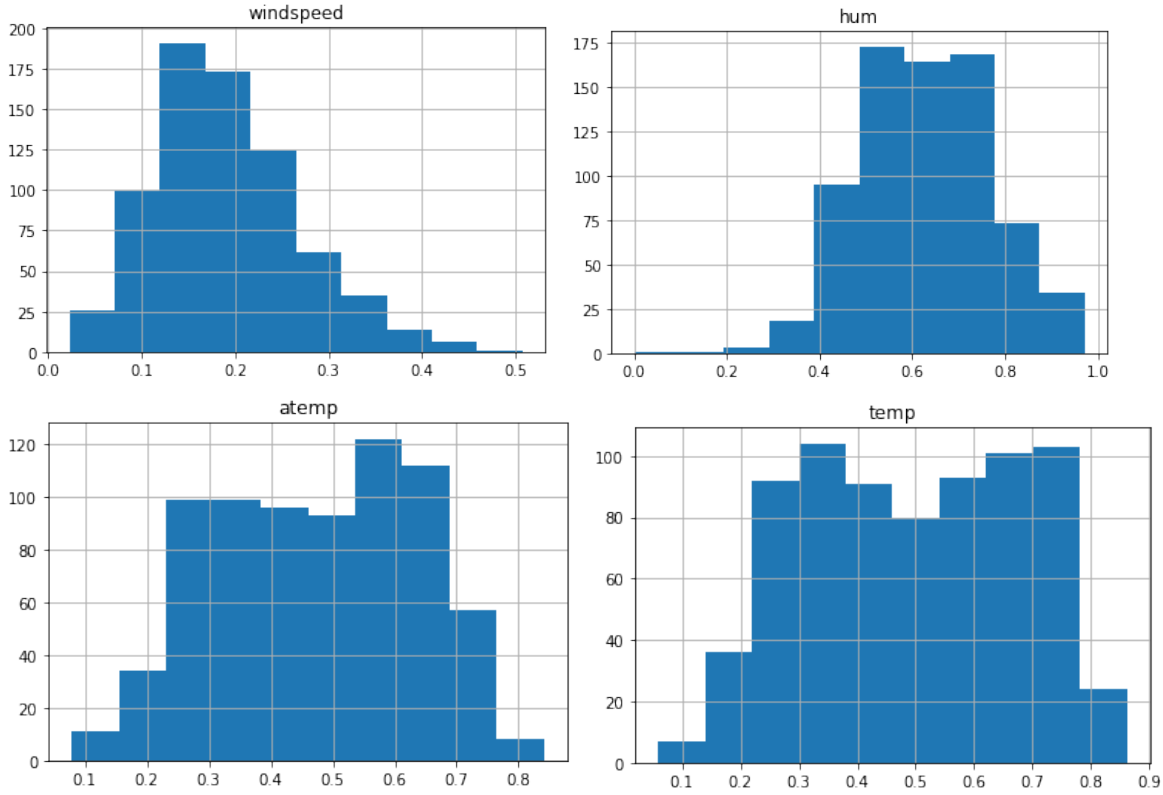
To correctly predict the count of the total bike rentals on a day, the following 11 predictor variables are used:

1. Season
2. Yr
3. Mnth
4. Holiday
5. Weekday
6. Workingday
7. Weathersit
8. Temp
9. Atemp
10. Hum
11. Windspeed

EXPLORATORY DATA ANALYSIS & DATA PRE-PROCESSING

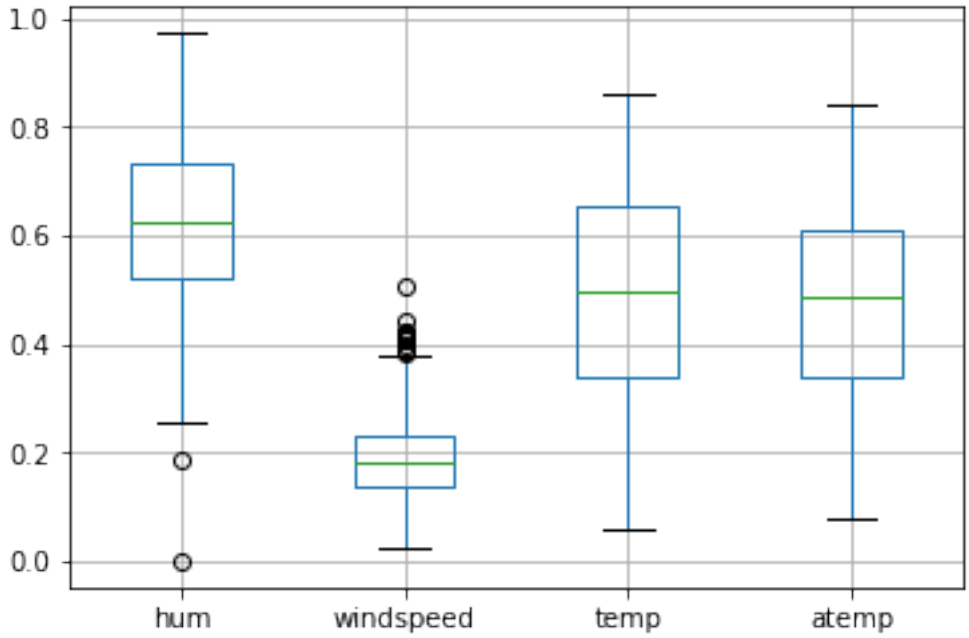
In exploratory data analysis, the focus is on checking the distribution of the values of a particular variable. Here, we try to see if the data is normally distributed or not. This is because some of the algorithms need to have the data normally distributed for a variable. Also, it is determined what the mean and the median values of the data. Along with that, the 25^{th} and the 75^{th} quartiles are determined, using which the inner and outer boundaries can be calculated, and thus the outliers in the data can be detected easily.

Below, we can see the distribution of all the numerical variables in our data:



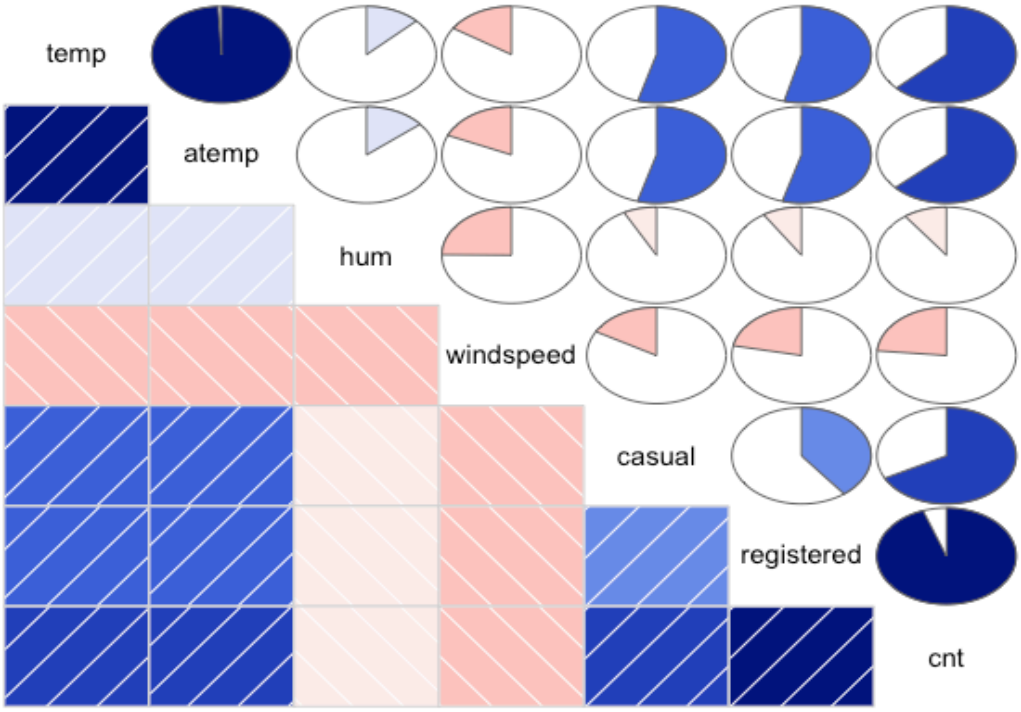
As can be seen in the distribution of all numerical data variables, there is not much skewness in the data, and they are almost normally distributed.

Now, we can have a look at the box plots of the numerical variables to determine whether there are outliers in the data or not.



From the figure, we seem to find outliers in the wind speed variable. But we can ignore this based on the domain knowledge, because there can be days where the wind speed is very high. So, we don't need to change these values.
But, in the humidity column there are values which are 0. This is actually not possible, so we can change this value and impute with a proper value.

Now, we can see the correlation between the different columns of the data:



The darker colours show that there is a lot of correlation between two columns. As can be seen, there is a lot of correlation between the ‘temp’ and the ‘atemp’ column. This makes sense as the feeling temperature is almost equivalent to the actual temperature. And we can make use of both the columns, to determine the bike rental counts.
Also, there seem to be correlation between the ‘registered’ and the ‘count’ column. This is also because ‘count’ column is, in fact, the sum of the ‘registered’ column and the ‘casual’ column.

METHODOLOGY

To proceed with the prediction of the count of the bike rentals, we need to predict the count of the casual bike rentals as well as the registered bike rentals. For this, we need to proceed by doing two different regressions: one for predicting the count of the casual bike rentals and the other for predicting the count of the registered bike rentals, using the predictor variables. Once, the two values are calculated, they are added to find the final value of the total bike rentals of the day.

Train – test split: Since, there is no external dataset available for the testing of the trained models, we need to split our data into two different datasets - one, that can be used for training the model, and the other that can be used to evaluate the performance of the trained model. About 70% of the data can be used for training the model, and the remaining 30% of the values can be used to test the performance of the model.

Models
Once, we have performed the split, it is time to implement the machine learning regression models on the data, to predict the bike rental values. The various regression models used are as follows:

- **Linear regression model:** The linear regression model is the simplest of all the regression models. It tries to make use of a linear equation to determine the value for new data. During the training phase, it tries to determine various coefficients for the independent variables. The equation that a linear regression model works on is:
  $y = a0 + a1*x1 + a2*x2 + ...$
  where a0, a1, a2 are all coefficients that the model tries to determine while training, and x1, x2, x3 are all the independent variables and y is the final answer of the dependent variable.
  So, the linear regression model tries to determine these co-efficients during the training phase, and then make the linear equation as stated above, and then make use of this equation to perform prediction for new data.
- **Decision tree model:** A decision tree regression model works by creating a tree structure and then determining the new values by passing it through this tree structure and finding out the value obtained at the leaf nodes. At each step, information gain or gini index is calculated to determine the best possible column for a split in the decision tree. At the first step, we calculate gini index or information gain to determine which is the best variable to perform the first split. Then, it is done again, to determine the variable for the best split. This process is done until we do cannot make any more splits. Finally, the leaf node is created, by taking average of the values of the dependent variable, of the variables present at the previous stage. So, now, whenever a new value comes, it is passed through this tree structure, and whichever leaf node this data ends up in, is the value for its dependent variable.

- **Random forest regression:** This model is just like decision trees, except for the fact that instead of creating one single tree, it creates a number of trees, and the final answer is either the mean or the median of all the answers of the trees. The trees are

all different from each other. To create a tree, the model takes up random number of variables and a random subset of the training data. This process is performed for all of the trees. We need to explicitly pass the number of trees that the random forest model should create.

CONCLUSION

Model Evaluation: To evaluate the performance of a model, we have various error metrics available. The metrics differ based on the type of task in hand. There are different metrics for evaluating the performance of a classification model and that for a regression model. For our task, we calculate the error in the prediction of the values. The lesser the value of the error, the better is the model. Here, we discuss two of the error metrics and why one is better than the other for our task in hand:

- **Root mean squared error (RMSE):** This is the square root of the MSE value. The MSE value is the mean squared error, which calculates the difference between the actual value and the predicted values, square it, and then finally take mean of all the values obtained. The formula to calculate RMSE is:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

  where: N = total number of observations
  y_i = actual value
  $\hat{y}_i$ = predicted value
    - For this task, RMSE or any such error metrics is not a suitable option. This is because, here, the values are very large, and to calculate the difference between such large values and then squaring it, will result in a huge value. This will not give a clear picture of how the models are performing.
- **Mean absolute percentage error (MAPE):** This is like MAE. But here, the percentage error is calculated as well. The formula to calculate MAPE is:

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

  where:
  n = total number of observations
  A_t = actual value
  F_t = forecasted/predicted value
    - The MAPE is a suitable metrics for this task. This is because it does not have any squared term in it. So, it gives a better picture of how the models are performing. Also, there are not '0' values in the actual values so division by zero is not an issue.

Model selection: To select a suitable model, we look at the errors that the various models produce. The errors produced by the various models are as follows:

For python code:

- Linear Regression Model: 20.24
- Decision Tree Model: 18.83
- Random Forest Model: **17.47**

For R code:

- Linear Regression Model: 20.42

- Decision Tree Model: 25.55
- Random Forest Model: **18.03**

As can be seen from the various errors in the models, the Random forest model was the best regression model out of all the models, for this task.

APPENDIX

Python code:

```
#Importing the required libraries.
import pandas as pd
from scipy.stats import chi2_contingency
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from fancyimpute import KNN

df = pd.read_csv('day.csv')
df.head()

df.dtypes

#Converting the data types of the required columns to categorical.
df.iloc[:,2:9] = df.iloc[:,2:9].astype('category')
df.dtypes

#Removing the 'instant' and 'dteday' columns from the dataframe.
df = df.drop(['instant','dteday'], axis=1)

#Histogram for the 'windspeed' column.
df.hist(column='windspeed')

#Histogram for the 'hum' column.
df.hist(column='hum')

#Histogram for the 'atemp' column.
df.hist(column='atemp')

#Histogram for the 'temp' column.
df.hist(column='temp')

#Boxplot for the numeric columns.
df.boxplot(column=['hum', 'windspeed', 'temp', 'atemp'])

#Correlation analysis for the numeric columns.
corr = df.iloc[:,7:14].corr()
corr

df.temp.describe()
df.atemp.describe()
#There is a value which is zero, which is clearly an outlier.
df.hum.describe()
df.windspeed.describe()
df.casual.describe()
df.registered.describe()
```

```
df['weathersit'].value_counts()

#Removing the outlier from the 'hum' column.
q75, q25 = np.percentile(df.loc[:,'hum'], [75 ,25])
iqr = q75 - q25
min = q25 - (iqr*1.5)
max = q75 + (iqr*1.5)
print(min)
print(max)
df.loc[df.loc[:,'hum'] < min,: 'hum'] = np.nan
df.loc[df.loc[:,'hum'] > max,: 'hum'] = np.nan
#Imputing using KNN imputation.
df = pd.DataFrame(KNN(k = 3).fit_transform(df), columns = df.columns)

df.hum.describe()

#Train - test split.
x_train, x_test, y_train, y_test = train_test_split(df.iloc[:,:-3], df['casual'],
random_state=1000)
x_train2, x_test2, y_train2, y_test2 = train_test_split(df.iloc[:,:-3], df['registered'],
random_state=1000)
x_train
x_train2

from sklearn.linear_model import LinearRegression

#Implementation of Linear Regression model.
lr = LinearRegression()
lr_mod = lr.fit(x_train, y_train)
cas = lr_mod.predict(x_test)
cas
lr_mod2 = lr.fit(x_train2, y_train2)
reg = lr_mod2.predict(x_test2)
reg
cnt = (cas + reg).round()
cnt
y_reg = y_test + y_test2
y_reg

#Calculating the MAPE for linear regression model.
y_reg, cnt = np.array(y_reg), np.array(cnt)
np.mean(np.abs((y_reg - cnt) / y_reg)) * 100

from sklearn.tree import DecisionTreeRegressor

#Implementing the decision tree model.
dt = DecisionTreeRegressor()
dt_mod = dt.fit(x_train, y_train)
cas_dt = dt_mod.predict(x_test)
```

```
cas_dt
dt_mod2 = dt.fit(x_train2, y_train2)
reg_dt = dt_mod2.predict(x_test2)
reg_dt
cnt_dt = cas_dt + reg_dt
cnt_dt

#MAPE for the decision tree model.
cnt_dt = np.array(cnt_dt)
np.mean(np.abs((y_reg - cnt_dt) / y_reg)) * 100

from sklearn.ensemble import RandomForestRegressor

#Random forest model implementation.
rf = RandomForestRegressor(n_estimators=500)
rf_mod = rf.fit(x_train, y_train)
cas_rf = rf_mod.predict(x_test)
cas_rf
rf_mod2 = rf.fit(x_train2, y_train2)
reg_rf = rf_mod2.predict(x_test2)
reg_rf
cnt_rf = (cas_rf + reg_rf).round()

#MAPE for the random forest model.
cnt_rf = np.array(cnt_rf)
np.mean(np.abs((y_reg - cnt_rf) / y_reg)) * 100
```

R code:

```
rm(list=ls(all=T))
#Set the working directory.
setwd("/Users/anchitchaturvedi/Desktop/Study/Edwisor/project2")
getwd()

df = read.csv('day.csv')
head(df)
#Removing the columns 'instant' and 'dteday' because they are not required for the analysis.
df = subset(df, select = -c(instant, dteday) )
#Converting the data types of the columns to categorical.
df[,1] = sapply(df[,1], as.factor)
df[,2] = sapply(df[,2], as.factor)
df[,3] = sapply(df[,3], as.factor)
df[,4] = sapply(df[,4], as.factor)
df[,5] = sapply(df[,5], as.factor)
df[,6] = sapply(df[,6], as.factor)
df[,7] = sapply(df[,7], as.factor)

class(df$weathersit)
sapply(df,class)
```

```
#Correlation analysis of the numeric columns.
library(corrgram)
corrgram(df[], order = F,
        upper.panel=panel.pie, text.panel=panel.txt)
summary(df$hum)

#Outlier removal from the 'hum' column.
val = df[,'hum'][df[,'hum'] %in% boxplot.stats(df[,'hum'])$out]
df[,'hum'][df[,'hum'] %in% val] = NA
val

#Knn imputation.
library(DMwR)
df = knnImputation(data = df, k = 3)
#Selecting the appropriate columns to predict 'casual' bike rental values.
cas = subset(df, select = -c(registered, cnt) )
#Selecting the appropriate columns to predict 'registered' bike rental values.
reg = subset(df, select = -c(casual, cnt) )
head(cas)
head(reg)

#Train - test split.
set.seed(100)
train_index = sample(1:nrow(df), 0.75*nrow(df))
cas_train = cas[train_index,]
cas_test = cas[-train_index,]
reg_train = reg[train_index,]
reg_test = reg[-train_index,]
class(cas_train$yr)

cnt = cas_test$casual + reg_test$registered
library(MASS)
#Implementing the linear regresssion model.
lr = lm(casual ~., data = cas_train)
lr_cas = predict(lr, cas_test[,1:11])

lr_re = lm(registered ~., data = reg_train)
lr_reg = predict(lr_re, reg_test[,1:11])
cnt_lr = lr_reg + lr_cas
cnt_lr = round(cnt_lr)

#Calculating MAPE for the model.
library(Metrics)
mape(cnt,cnt_lr)

#Implementation of the decision tree model.
library(rpart)
dt_cas = rpart(casual ~., data = cas_train)
cas_dt = predict(dt_cas, cas_test[,1:11])
```

```
dt_reg = rpart(registered ~., data = reg_train)
reg_dt = predict(dt_reg, reg_test[,1:11])

cnt_dt = cas_dt + reg_dt
cnt_dt = round(cnt_dt)

#Calculating MAPE for decision tree model.
mape(cnt, cnt_dt)

#Implementing random forest regression model.
library(randomForest)
cas_rf = randomForest(casual ~., data = cas_train)
rf_cas = predict(cas_rf, cas_test[,1:11])

reg_rf = randomForest(registered ~., data = reg_train)
rf_reg = predict(reg_rf, reg_test[,1:11])

cnt_rf = rf_reg + rf_cas
cnt_rf = round(cnt_rf)

#Calculating the MAPE for Random Forest model.
mape(cnt, cnt_rf)
```