Bike Rental Count Prediction- Python

Chaitrali A Deokar

April 2019

Contents

Chapter 1

INTRODUCTION

1.1 Problem Statement

Bike rental count per day varies greatly based on the environmental settings like weather (sunny or raining) , whether it is a working day or holiday, etc. The objective of this analysis is to Predication of bike rental count on daily based on the environmental and seasonal settings. This will help the renting agencies identify the demand (increase or decrease) for a particular day and ensure adequate supply of bikes to optimise porift.

1.2 Data

The Analysis involves building a Regression model to predict the Bike rental count depending on a number of input parameters. The given data is a CSV file that consists 16 variables and 731 Observation. Given below is a sample of the data that we develop our model on

Instant	Dteday	season	Yr	mnth	holiday	weekday	Workingday	weathersit
1	01-01-2011	1	0	1	0	6	0	2
2	02-01-2011	1	0	1	0	0	0	2
3	03-01-2011	1	0	1	0	1	1	1
4	04-01-2011	1	0	1	0	2	1	1
5	05-01-2011	1	0	1	0	3	1	1

Table 1.1 Columns 1- 9 of the "day.csv" data

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
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

Table 1.2 Columns 9-16 of the "day.csv" data

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

1. instant: Record index
2. dteday: Date
3. season: Season (1:springer, 2:summer, 3:fall, 4:winter)
4. yr: Year (0: 2011, 1:2012)
5. mnth: Month (1 to 12)
6. holiday: weather day is holiday or not (extracted fromHoliday Schedule)
7. weekday: Day of the week
8. workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
9. weathersit: (extracted fromFreemeteo)
    i. 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    ii. 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    iii. 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

iv. 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

10. temp: Normalized temperature in Celsius. The values are derived via
    i. (t-t_min)/(t_max-t_min), t_min=-8,

    ii. t_max=+39 (only in hourly scale)
11. atemp: Normalized feeling temperature in Celsius. The values are derived via
    i. (t-t_min)/(t_maxt_min), t_min=-16,

    ii. t_max=+50 (only in hourly scale)
12. hum: Normalized humidity. The values are divided to 100 (max)
13. windspeed: Normalized wind speed. The values are divided to 67 (max)
14. casual: count of casual users
15. registered: count of registered users
16. cnt: count of total rental bikes including both casual and registered

From the above sample it is seen that the 2 to 13 are the the seasonal setting variables like temperature, the humidity etc on which the rental count per day is dependent. Hence, the are the predictor variables
The dependent variables are 3 namely, casual, registered and cnt. Among these our target variable is “cnt” which is basically a sum of the “casual” and “registered variables”
Hence below is a list of the dependent variables

Dependent Variables
Dteday
Season
Yr
Mnth
Holiday
Weekday
Workingday
Weathersit
Temp
Atemp
Hum
Windspeed

Table 1.3 Independent Variables

And, the target variables whose output depends on these are

Target variable
Casual
Registered
Cnt

Table 1.3 Dependent Variables

Chapter 2

Methodology

2.1 CRISP DM Process

CRISP-DM stands for cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining project. The project follows CRISP Dm process to develop the model for the given problem. It involves the following steps:

1. Business understanding
2. Data understanding
3. Data preparation
4. Modeling
5. Evaluation
6. Deployment

2.2 Business Understanding

1. Set objectives - This primary objective from a business perspective would be " How much do the seasonal settings affect the Bike rental count?" and "to Predict the Bike rental count wit variation in the external seasonal settings".
2. Produce project plan – The plan will involve Understanding the data and converting into a proper shape , i.e., a data free from any missing values, outliers that can possibly produce errors; scaling the ; applying various models and choosing the best model to predict the bike rental count
3. Business success criteria – The success would be related to able to accurately predict the bike rental count for a particular input , with maximum accuracy . On being able to predict it accurately, sufficient bikes will be made available to ensure uninterrupted supply of bikes to meet the demand each day.

2.3 Data Understanding

The given data is a CSV file that consists 16 variables and 731 Observation.

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

1. instant: Record index
2. dteday: Date
3. season: Season (1:springer, 2:summer, 3:fall, 4:winter)
4. yr: Year (0: 2011, 1:2012)
5. mnth: Month (1 to 12)
6. holiday: weather day is holiday or not (extracted fromHoliday Schedule)
7. weekday: Day of the week
8. workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
9. weathersit: (extracted fromFreemeteo)
    1. 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    2. 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    3. 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    4. 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
10. temp: Normalized temperature in Celsius. The values are derived via
    1. (t-t_min)/(t_max-t_min), t_min=-8,

2. t_max=+39 (only in hourly scale)

11. atemp: Normalized feeling temperature in Celsius. The values are derived via
    1. (t-t_min)/(t_maxt_min), t_min=-16,

    2. t_max=+50 (only in hourly scale)
12. hum: Normalized humidity. The values are divided to 100 (max)
13. windspeed: Normalized wind speed. The values are divided to 67 (max)
14. casual: count of casual users
15. registered: count of registered users
16. cnt: count of total rental bikes including both casual and registered

2.4 Data Pre Processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

2.4.1 Missing Value Analysis

Missing value is the occurance of incomplete observations in asset of data. Missing values can arise due to many reasons, error in uploading data, unable to measure a particular observation etc.

Due to presence of missing values, that observation either gets value as 0 or NA. These affect the accuracy of model. Hence it is necessary to check for any missing values in the given data.

Our given data does not have any missing values, hence we can skip this

2.4.2 Outlier Analysis

An Outlier is any inconsistent or abnormal observation that deviates from the rest of the observations. These inconsistent observation can be due to manual error, poor quality/ malfunctioning equipments or correct but exceptional data. It can cause an error in prediciting the target variables.

Hence we need to check for the outliers and either remove the observations containing them or replace them with NA and later impute values.

We visualize the outliers using boxplots. We plot the continuous variables using Box plot and the following are the results:

Box Plot in Python

The following code helps to plot box plot for each variable:

```
%matplotlib inline
plt.boxplot(rental['temp'])
plt.boxplot(rental['atemp'])
plt.boxplot(rental['hum'])
plt.boxplot(rental['windspeed])
```

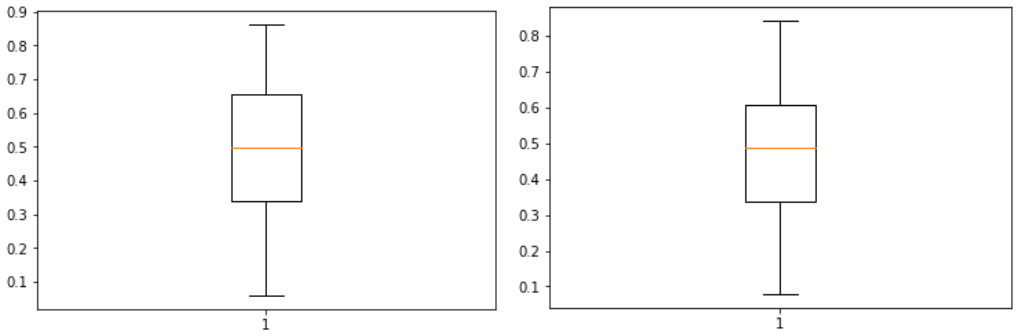

Fig 1 Box plot of temp and atemp variables

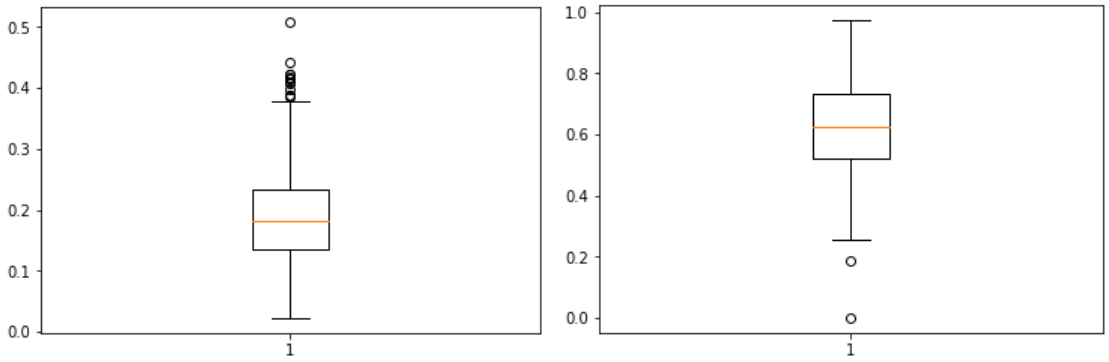

Fig 2 Box plot of windspeed and hum variables

The figures Shows that there are no outliers in the temp, atemp variable whereas there are outliers in

the "hum" and "windspeed" variable

These outliers can be removed by 2 ways:

1. Removing the observations consisting of the Outliers
2. Replacing outliers with NA and the imputing values.

In our case, we chose to Replace the outliers with NA. The values can be imputed using 3 methods namely

1. Mean method
2. Median Method
3. KNN imputation method

We did a trial and the Mean method worked better, hence we imputed the NAs in Hum with the Mean value of the particular variable.

2.4.3 Feature Selection

Before creating a model we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of prediction. Hence, feature selection can help in reducing the time for computation of model as well as the complexity of the model. Also, few models require the independent variables to be free from multicollinear effect, hence it is needed to perform various procedures to ensure that the independent variables are not collinear.

The 1st step involved is to do the correlation analysis for the categorical variables and continuous variables. Since our Predictor variable is also continuous we will plot a correlation table that would predict the correlation strength between the dependent variable and the 'cnt', 'casual' and 'registered' variable

Correlation Analysis in Python For Continuous variables

In similar method, we do correlation analysis in python for continuous variable using thefollowing code:

```
            #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, ax=ax)
```

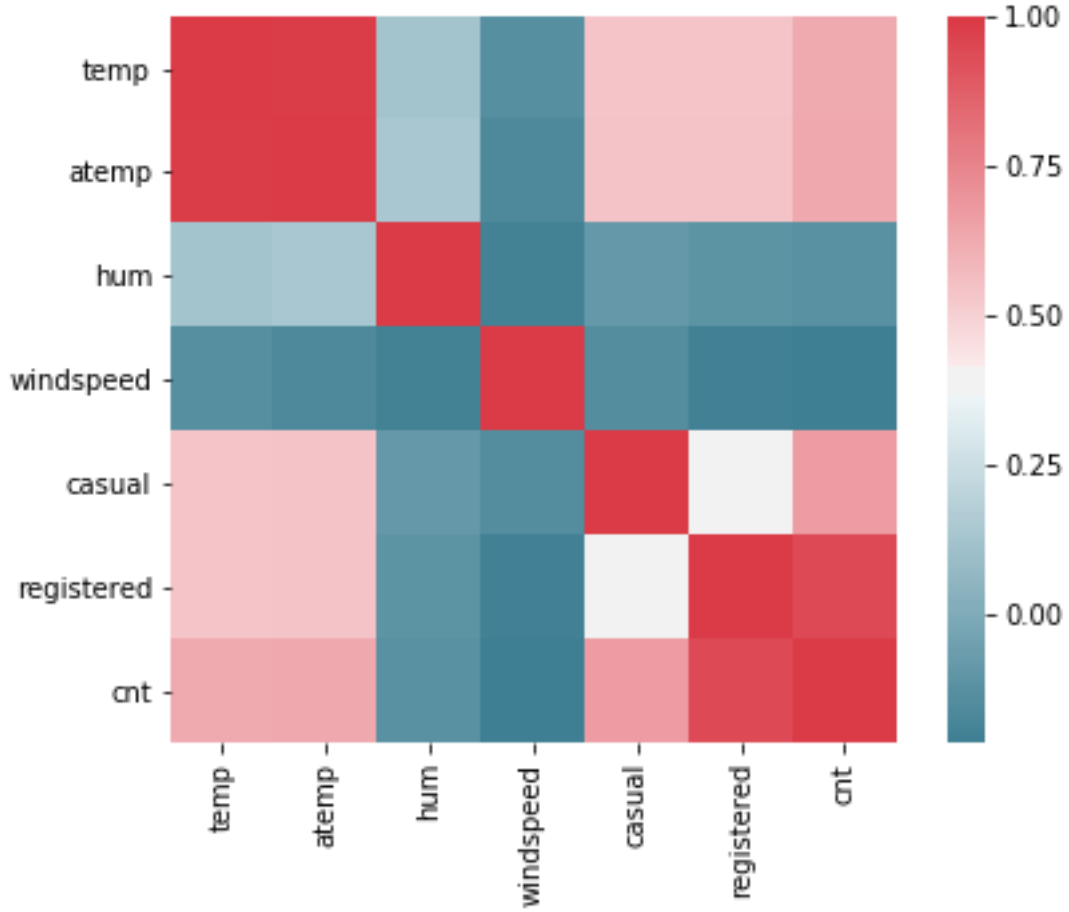

Fig 3 Correlation plot between continuous variables

Similarly, to get the numerical values for better understanding, we use the code:

print(corr)

	temp	atemp	hum	windspeed	Casual	registered	cnt
Temp	1	0.991702	0.123703	-0.14	0.543285	0.540012	0.627494
Atemp	0.991702	1	0.137293	-0.165315	0.543864	0.544192	0.631066
Hum	0.123703	0.137293	1	-0.201894	-0.0878	-0.11224	-0.121518
Windspeed	-0.14	-0.16532	-0.20189	1	-0.14693	-0.20411	-0.216473
Casual	0.543285	0.543864	-0.0878	-0.146933	1	0.395282	0.672804
Registered	0.540012	0.544192	-0.11224	-0.204112	0.395282	1	0.945517
Cnt	0.627494	0.631066	-0.12152	-0.216473	0.672804	0.945517	1

Table 2.1 Correlation Value of continuous variables

For Categorical Variables,

For categorical variables, Chi-sqaure test of independence is applied

```
#Chisquare test of independence
#Save categorical variables
cat_names = ['season', 'mnth','yr', 'holiday', 'weekday', 'workingday',
  'weathersit', 'dteday']
#loop for chi square
for i in cat_names:

  for k in cat_names:
    print(i)
    print(k)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(rental[k] , rental[i]))
  print(p)
```

From the above python correlation, the following is seen:

1. Temp and atemp are highly related
2. Temp and atemp are highly correlated with the target variable “cnt”

Hence from the above we can choose to remove the following variables for our model development:

1. dteday- since they are just date instants
2. The “casual” variable, because it is not needed to be predicted as cnt is the target variable
3. The “Registered” variable, because it is not needed to be predicted as cnt is the target variable
4. Though temp and atemp are highly correlated, since the are highly correlated to cnt, we do not remove them.

2.5 Modeling in Python

We know that thedependent variabes are “cnt” , “casual” , “registered” . The “cnt” variable is a sum of the “casual” and “registered” variable. Hence we can choose to take only the “cnt” variable as the output to simplify the model. The model is to predict count, hence it is a **Regression** problem statement.

The various models that can be used for prediction problem statement are Decision trees, Random Forest, Linear Regression and KNN imputation.

2.5.1 Decision Tree

The Decision tree algorithm is applied

```
#Load libraries
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor

# Divide the data into train and test
train, test = train_test_split(rental_train, test_size=0.2)

fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:, 0:11], train.iloc[:,11])
```

```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
        max_leaf_nodes=None, min_impurity_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=None, splitter='best')
```

2.5.2 Random Forest

The following code is applied to develop the Random Forest Algorithm and predict the values:

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier

RF_model = RandomForestClassifier(n_estimators = 30).fit(train.iloc[:, 0:11], train.iloc[:,11])

RF_Predictions = RF_model.predict(test.iloc[:, 0:11])
```

2.5.3 KNN imputation model

The KNN model is predicted throught the following source of code

```
#KNN implementation
from sklearn.neighbors import KNeighborsClassifier
KNN_model = KNeighborsClassifier(n_neighbors = 3 ).fit(train.iloc[: , 0:11], train.iloc[:, 11])
#predict test cases
KNN_Predictions = KNN_model.predict(test.iloc[: , 0:11])
```

2.5.4 Linear Regression

Since our data is a mix of categorical and numerical data types, Linear Regression model cannot work. This is because of presence of categorical data. In order to fit the model, we need to create dummy variables from our input data.

```
#Load libraries
import os
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split

linear = rental_train.iloc[:, 7:11]
cat_names = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday','weathersit']
for i in cat_names:
   temp = pd.get_dummies(rental_train[i], prefix = i)
   linear = linear.join(temp)
linear= linear.join(rental_train['cnt'])

import statsmodels.api as sm
# Train the model using the training sets
model = sm.OLS(train1.iloc[:, 36 ], train1.iloc[:, 0:36 ]).fit()

# Print out the statistics
model.summary()
```

OLS Regression Results

Dep. Variable:	cnt	**R-squared:**	0.838			
Model:	OLS	**Adj. R-squared:**	0.829			
Method:	Least Squares	**F-statistic:**	102.3			
Date:	Wed, 17 Apr 2019	**Prob (F-statistic):**	9.88E-199			
Time:	15:47:33	**Log-Likelihood:**	-4710.2			
No. Observations:	584	**AIC:**	9478			
Df Residuals:	555	**BIC:**	9605			
Df Model:	28					
Covariance Type:	nonrobust					
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Temp	2711.6209	1498.646	1.809	0.071	-232.09	5655.332
Atemp	2048.2741	1553.545	1.318	0.188	-1003.273	5099.821
Hum	-1477.96	357.652	-4.132	0	-2180.477	-775.443
Windspeed	-2348.7184	515.392	-4.557	0	-3361.075	-1336.362
season_1	-479.3525	141.365	-3.391	0.001	-757.028	-201.677
season_2	125.6893	145.334	0.865	0.388	-159.783	411.162
season_3	386.9613	150.098	2.578	0.01	92.132	681.79
season_4	1113.9502	146.239	7.617	0	826.701	1401.2
yr_0	-415.7834	82.519	-5.039	0	-577.87	-253.697
yr_1	1563.0318	80.648	19.381	0	1404.619	1721.444
mnth_1	-338.4498	206.483	-1.639	0.102	-744.033	67.133
mnth_2	-190.4273	193.86	-0.982	0.326	-571.216	190.361

mnth_3	251.5407	151.781	1.657	0.098	-46.596	549.677
mnth_4	402.7009	179.693	2.241	0.025	49.739	755.663
mnth_5	721.395	195.247	3.695	0	337.882	1104.908
mnth_6	459.0443	187.342	2.45	0.015	91.058	827.031
mnth_7	-285.0334	214.989	-1.326	0.185	-707.325	137.258
mnth_8	112.7212	204.445	0.551	0.582	-288.858	514.301
mnth_9	655.8473	161.088	4.071	0	339.43	972.264
mnth_10	202.9996	177.103	1.146	0.252	-144.874	550.873
mnth_11	-461.3188	191.96	-2.403	0.017	-838.375	-84.263
mnth_12	-383.7713	166.515	-2.305	0.022	-710.847	-56.696
holiday_0	818.6531	100.126	8.176	0	621.981	1015.325
holiday_1	328.5953	114.912	2.86	0.004	102.88	554.311
weekday_0	-128.9968	70.348	-1.834	0.067	-267.177	9.184
weekday_1	60.867	83.415	0.73	0.466	-102.98	224.714
weekday_2	180.572	84.509	2.137	0.033	14.576	346.568
weekday_3	234.0469	88.346	2.649	0.008	60.514	407.58
weekday_4	154.6466	85.473	1.809	0.071	-13.244	322.537
weekday_5	290.6338	83.501	3.481	0.001	126.616	454.651
weekday_6	355.4789	69.955	5.082	0	218.071	492.887
workingday_0	555.0775	93.412	5.942	0	371.594	738.561
workingday_1	592.1709	67.852	8.727	0	458.893	725.449
weathersit_1	1284.9095	84.741	15.163	0	1118.458	1451.361
weathersit_2	783.2974	85.382	9.174	0	615.586	951.009
weathersit_3	-920.9586	184.875	-4.982	0	-1284.099	-557.818
Omnibus:	105.176	**Durbin-Watson:**	2.054			
Prob(Omnibus):	0	**Jarque-Bera (JB):**	300.227			
Skew:	-0.876	**Prob(JB):**	6.41E-66			
Kurtosis:	6.045	**Cond. No.**	1.09E+16			

The Linear regression calculates the coefficients for each of the dummy variables

For all the p vales < than 0.05 the target variable is dependent corresponding variable. On this we predict the values of test cases and check the error

The R square and adj R Square values are near 83, which means 83% of variable can explain the target variable.

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 *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 Error (MAE)

MAE is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have generated in the previous section.

In Python, Define MAPE using function

```
#Calculate MAPE
def MAPE(y_true, y_pred):
   mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
   return mape            }

MAPE(test.iloc[:,11], predictions_DT) #Mape for Decision Tree
Out[] 29.75
MAPE(test.iloc[:,11], RF_Predictions) #Mape For Random Forest at n=200
Out[] 24.85
MAPE(test[,12], predictions_LR)   #MApe for Linear Regression
Out[] 16.37
MAPE(test.iloc[:,11], KNN_Predictions)   #Mape for KNN method
Out[]22.14738
```

3.1.2 Accuracy

It is the ratio of number of correct predictions to the total number of predictions made.

Accuracy= number of correct predictions / Total predicitions made

It can also be calculated from MAE as

Accuracy = 100 – MAE

1. Accuracy fro **Decision tree** = 100-29.75= **70.25%**
2. Accuracy for **Random forest** = 100-24.85= **75.15%**
3. Accuracy for **Linear Regression** = 100-16.37=**83.36%**
4. Accuracy for **KNN method** = 100-22.96 =**77.04%**

3.2 Model Selection

From the values of Error and accuracy, it is seen that all the models perform similar in terms of Error. As such any model can be used, but **Linear Regression** gives better results compared to other algorithms.

Appendix

Entire Python Code

```
# In[1]:
#Load libraries
import os
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform

# In[2]:
os.chdir("C:/Users/Chaitrali/Downloads/Data/data01s2l1/Edwisor-Project")

# In[11]:
rental=pd.read_csv('day.csv', header = 0 ).iloc[:,1:16]
train = pd.read_csv('trial1.csv', header = 0 )
test = pd.read_csv('trialte1.csv', header = 0 )

# In[16]:
#concert the required variavles to category
convert_dic={'dteday' : 'category','season': 'category','mnth': 'category','yr': 'category','holiday': 'category',
'weekday': 'category', 'workingday': 'category', 'weathersit': 'category'}
rental=rental.astype(convert_dic)
print(rental.dtypes)


# In[18]:
# Missing Value Analysis
#Create dataframe with missing percentage
missing_val = pd.DataFrame(rental.isnull().sum())

#Reset index
missing_val = missing_val.reset_index()

#Rename variable
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_percentage'})

#Calculate percentage
missing_val['Missing_percentage'] = (missing_val['Missing_percentage']/len(missing_val))*100

#descending order
missing_val = missing_val.sort_values('Missing_percentage', ascending = False).reset_index(drop = True)
# missing_val
#save output results
# missing_val.to_csv("Miising_perc.csv", inex = False)
```

```
# # Outlier Analysis

# In[40]:
#save it in another name, incase we need it again
df = rental.copy()
rental = df.copy()


# In[25]:
# #Plot boxplot to visualize Outliers
get_ipython().run_line_magic('matplotlib', 'inline')
plt.boxplot(rental['windspeed'])


# In[47]:
#save numeric names
coutliers = [ 'hum', 'windspeed']
for list in coutliers:
   #Detect and replace with NA
   # #Extract quartiles
   q75, q25 = np.percentile(rental[list], [75 ,25])

   #Calculate IQR
   iqr = q75 - q25

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

   # #Replace with NA
   rental.loc[rental[list] < minimum,list] = np.nan
   rental.loc[rental[list] > maximum,list] = np.nan

   # #Calculate missing value
   missing_val = pd.DataFrame(rental.isnull().sum())
   missing_val


# In[49]:
#Impute with mean
rental['windspeed'] = rental['windspeed'].fillna(rental['windspeed'].mean())


# In[50]:
#Impute with mean
rental['hum'] = rental['hum'].fillna(rental['hum'].mean())


# ## Feature Selection
```

```
# In[52]:
##Correlation analysis
#Correlation plot
numeric=['temp', 'atemp', 'hum', 'windspeed' ,'casual','registered', 'cnt']
df_corr = rental.loc[:,numeric]


# In[53]:
#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix
corr = df_corr.corr()
print(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, ax=ax)


# In[54]:
#Chisquare test of independence
#Save categorical variables
cat_names = ['season', 'mnth','yr', 'holiday', 'weekday', 'workingday',
      'weathersit', 'dteday']


# In[57]:
for i in cat_names:

    for k in cat_names:
        print(i)
        print(k)
        chi2, p, dof, ex = chi2_contingency(pd.crosstab(rental[k] , rental[i]))
        print(p)


# In[192]:
rental_train=rental.drop([ 'dteday','casual', 'registered'], axis=1)


# In[193]:
rental_train.head(5)




# # Model Development
```

```
# # Decision Tree

# In[60]:
#Load libraries
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor


# In[115]:
# Divide the data into train and test
# train, test = train_test_split(rental_train, test_size=0.2)
# Already loaded


# In[249]:
# Decision tree for regression
fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:, 0:11], train.iloc[:,11])


# In[70]:
fit_DT


# In[250]:
#Apply model on test data
predictions_DT = fit_DT.predict(test.iloc[:,0:11])


# In[72]:
#Calculate MAPE
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape


# In[251]:
MAPE(test.iloc[:,11], predictions_DT)


# In[ ]:
#Error 29.75
#Accuracy 70.25


# In[257]:
#Save_pred as csv
pd.DataFrame(predictions_DT, columns = ['DT_Predictions']).to_csv("DT_pred-Py.csv", index = False)
```

```
# # Random Forest

# In[74]:
#Random Forest
from sklearn.ensemble import RandomForestClassifier


# In[84]:
RF_model = RandomForestClassifier(n_estimators = 30).fit(train.iloc[:, 0:11], train.iloc[:,11])


# In[85]:
RF_Predictions = RF_model.predict(test.iloc[:, 0:11])


# In[86]:
MAPE(test.iloc[:,11], RF_Predictions)


# In[ ]:
#error is 24.85 for n=30
#accuracy 75.16


# In[258]:
#Save_pred as csv
pd.DataFrame(RF_Predictions, columns = ['PF_Predictions']).to_csv("RF_pred-Py.csv", index = False)


# # KNN Imputation
# In[87]:
#KNN implementation
from sklearn.neighbors import KNeighborsClassifier

KNN_model = KNeighborsClassifier(n_neighbors = 1).fit(train.iloc[: , 0:11], train.iloc[:, 11])


# In[88]:
#predict test cases
KNN_Predictions = KNN_model.predict(test.iloc[: , 0:11])


# In[89]:
MAPE(test.iloc[:,11], KNN_Predictions)


# In[ ]:
#error is 22.96 for K-1
#accuracy is 77.06
```

```
# In[259]:
#Save_pred as csv
pd.DataFrame(KNN_Predictions, columns = ['KNN_Predictions']).to_csv("KNN_pred-Py.csv", index = False)


#
# # Linear Regression
#

# In[55]:
#Load libraries
import os
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split


# In[210]:
rental_train.head(0)


# In[211]:
linear = rental_train.iloc[:, 7:11]


# In[213]:
cat_names = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday','weathersit']
for i in cat_names:
    temp = pd.get_dummies(rental_train[i], prefix = i)
    linear = linear.join(temp)


# In[214]:
linear= linear.join(rental_train['cnt'])


# In[224]:
#already loaded
train1, test1 = train_test_split(linear, test_size=0.2)


# In[217]:
#Import libraries for LR
import statsmodels.api as sm


# In[228]:
# Train the model using the training sets
model = sm.OLS(train1.iloc[:, 36 ], train1.iloc[:, 0:36 ]).fit()
```

```
# In[229]:
# Print out the statistics
model.summary()


# In[230]:
# make the predictions by the model
predictions_LR = model.predict(test1.iloc[:,0:36])


# In[232]:
#Calculate MAPE
MAPE(test1.iloc[:,36], predictions_LR)


# In[ ]:
#MAPE 16.37
#Accuracy 83.63


# In[233]:
train1.to_csv("py_LR_train.csv", index = False)
test1.to_csv("py_LR_test.csv", index= False)


# In[260]:
#Save_pred as csv
pd.DataFrame(predictions_LR, columns = ['LR_Predictions']).to_csv("LR_pred-Py.csv", index = False)
```

Reference

- **Websites**
  - www.edwisor.com
  - www.analyticsvidhya.com
  - www.tuitorialspoint.com
  - www.geeksforgeeks.com