BIKE RENTING **PREDICTIONS BASED ON WEATHER CONDITIONS**

BY:

RAHUL RANJAN

Contents

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.
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)
hr: Hour (0 to 23)
holiday: weather day is holiday or not (extracted fromHoliday Schedule)
weekday: Day of the week
workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
weathersit: (extracted fromFreemeteo)
1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp: Normalized temperature in Celsius. The values are derived via
(t-t_min)/(t_max-t_min),
t_min=-8, t_max=+39 (only in hourly scale)
atemp: Normalized feeling temperature in Celsius. The values are derived via
(t-t_min)/(t_maxt_
min), t_min=-16, t_max=+50 (only in hourly scale)
hum: Normalized humidity. The values are divided to 100 (max)
windspeed: Normalized wind speed. The values are divided to 67 (max)
casual: count of casual users
registered: count of registered users
cnt: count of total rental bikes including both casual and registered

1.2 Data

Data is attached as csv

Sample data s shown below :

```
dtype='object')
```

In [237]:
```
df1.head()
```

Out[237]:

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

2.1 Pre Processing

We start with Data Exploratory Analysis and changing the way data looks
We change the behavioral data into categorical columns

2.2 Missing Value Analysis

Missing value analysis is done to check is there any missing value present in given dataset. Missing values can be easily treated using various methods like mean, median method, knn method to impute missing value.

As shown below in the given data set there was no missing value

Lets check for any kind of null or missing values

```
[239]: df1.isnull().sum()
```

```
[239]: instant       0
       dteday        0
       season        0
       yr            0
       mnth          0
       holiday       0
       weekday       0
       workingday    0
       weathersit    0
       temp          0
       atemp         0
       hum           0
       windspeed     0
       casual        0
       registered    0
       cnt           0
       dtype: int64
```

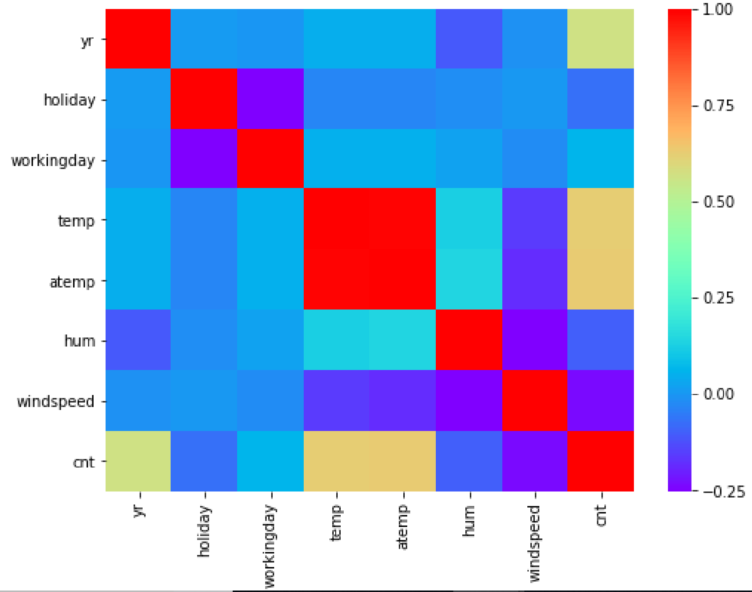
NO there are no null values

2.3 Feature Selection

Feature selection analysis is done to Select subsets of relevant features (variables, predictors) to be in model construction.
As our target variable is continuous so we can only go for correlation check. We use a heatmap to see co relation.

Co relation can also be observed by finding co relation of the data frame

```
In [251]: corr = df1.corr()
          plt.figure(figsize=(8,6))
          sns.heatmap(corr,mask = np.zeros_like(corr,dtype=np.bool),cmap='rainbow')

Out[251]: <matplotlib.axes._subplots.AxesSubplot at 0x26507eb7cf8>
```

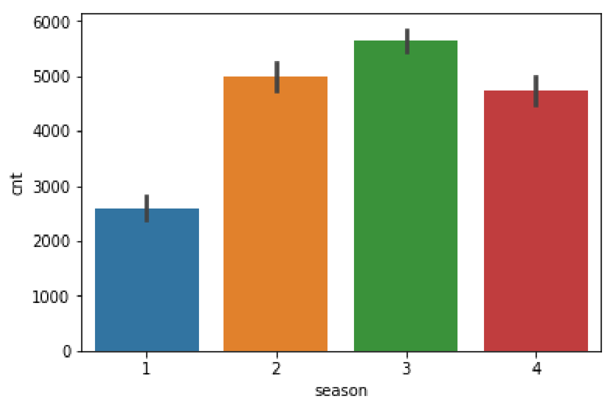


Things confirmed for co relation of cnt column :

1. Renters numbers depends upon the season.
2. Renters depended upon the weekday too . Slightly higher on weekends.
3. Weather or temperature plays an important role for bike renting.

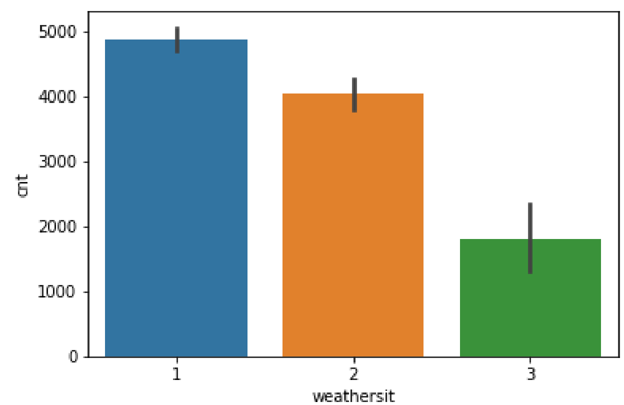
Bar Plot of count vs season

```
In [370]: #x=df1.groupby(['season'])['cnt'].sum()

          sns.barplot(x=df1['season'],y=df1['cnt'])

Out[370]: <matplotlib.axes._subplots.AxesSubplot at 0x2650c158ac8>
```



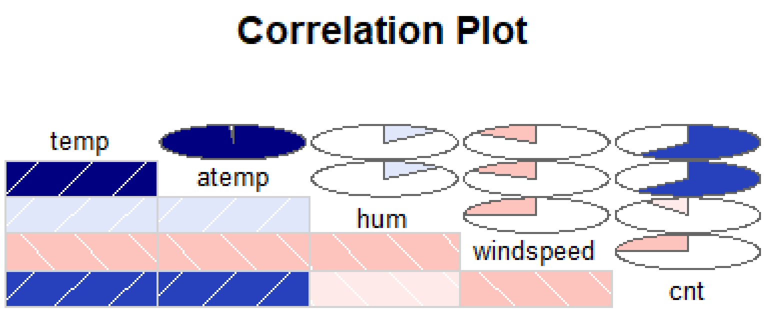
Bar Plot of count vs weather

```
: sns.barplot(x=df1['weathersit'],y=df1['cnt'])

: <matplotlib.axes._subplots.AxesSubplot at 0x291ee512128>
```

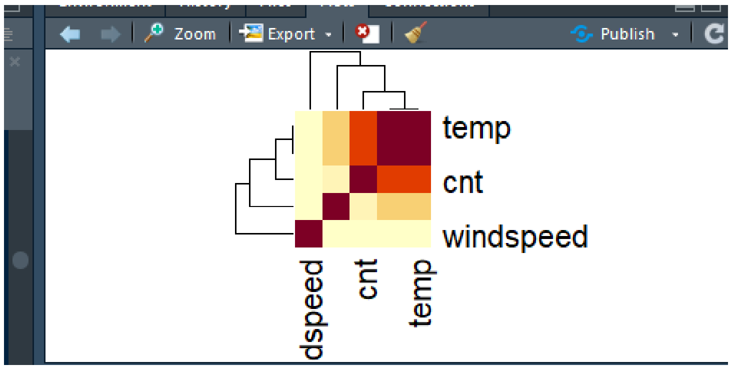


Co relation in R

```
numeric_index = sapply(data_frame,is.numeric)

corrgram(data_frame[,numeric_index], order = F,
         upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
corr = cor(data_frame[,numeric_index], method = c("pearson", "kendall", "spearman"))

heatmap(d, scale = "none")


## Dimension Reduction####
```



Heatmap of R



2.4 Modeling

2.5 Model Selection

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For Continuous we can use various Regression models. Model having less error rate and more accuracy will be our final model.

Models built are

1. Random Forest (with 200 trees)
2. Linear regression

Random Forest

```
In [302]: ########## Random Forest ################
          df1_RF = df1.drop('atemp',axis=1)
```

```
In [340]: from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor

          train,test = train_test_split(df1_RF,test_size = 0.2)
```

```
In [346]: RFmodel = RandomForestRegressor(n_estimators = 200).fit(df1_RF.iloc[:,0:11], df1_RF.iloc[:,11])
          RF_Predictions = RFmodel.predict(test.iloc[:,0:11])
```

Linear Regression:

For applying linear Regression we are changing the categorical columns into dummies value columns .Dummy value will be in binary form if true the 1
If False then 0

In [256]:
```
from sklearn.model_selection import train_test_split


df1_linr=df1.copy()
categorical = ["season", "dteday", "weathersit", "mnth","weekday"]
for i in categorical:
    temp_c = pd.get_dummies(df1_linr[i], prefix = i)
    df1_linr = df1_linr.join(temp_c)
```

In [257]:
```
drop = ['dteday', 'season', 'weathersit', 'weekday', 'mnth','cnt']
df1_linr = df1_linr.drop(drop, axis=1)
df1_linr=df1_linr.join(df1['cnt'])
```

In [294]:
```
X, y = train_test_split(df1_linr, test_size=0.2)
model = sm.OLS(X['cnt'], X.iloc[:,0:63]).fit()
```

In [295]:
```
predictions = model.predict(y.iloc[:,0:63])
```

In [278]:
```
results = sm.OLS(df1_linr['cnt'], df1_linr).fit()
```

In [279]:
```
print(results.summary())
```

```
OLS Regression Results
```

Ols Model summary :

```
In [279]: print(results.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.701e+30
Date:                Sat, 28 Sep 2019   Prob (F-statistic):               0.00
Time:                        23:51:47   Log-Likelihood:                 17984.
No. Observations:                 731   AIC:                        -3.585e+04
Df Residuals:                     671   BIC:                        -3.557e+04
Df Model:                          59
Covariance Type:            nonrobust
================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
yr           2.402e-12   6.65e-13      3.611      0.000     1.1e-12    3.71e-12
holiday      2.041e-12   1.02e-12      1.996      0.046    3.28e-14    4.05e-12
workingday  -1.755e-12   5.75e-13     -3.051      0.002   -2.88e-12   -6.25e-13
temp        -1.248e-11    9.7e-12     -1.286      0.199   -3.15e-11    6.57e-12
atemp        1.802e-11   1.01e-11      1.780      0.076   -1.86e-12    3.79e-11
hum         -1.577e-12   2.06e-12     -0.765      0.445   -5.63e-12    2.47e-12
windspeed    -1.13e-12   2.96e-12     -0.381      0.703   -6.95e-12    4.69e-12
season_1     3.411e-12   9.21e-13      3.703      0.000     1.6e-12    5.22e-12
season_2     -2.48e-12   9.46e-13     -2.621      0.009   -4.34e-12   -6.22e-13
season_3     -1.93e-12   1.01e-12     -1.915      0.056   -3.91e-12    4.91e-14
season_4    -8.384e-13   1.03e-12     -0.815      0.415   -2.86e-12    1.18e-12
dteday_01   -9.148e-13   1.06e-12     -0.863      0.388      -3e-12    1.17e-12
dteday_02   -1.119e-13   1.05e-12     -0.106      0.915   -2.18e-12    1.95e-12
dteday_03   -1.325e-12   1.05e-12     -1.258      0.209   -3.39e-12    7.42e-13
dteday_04    2.833e-13   1.06e-12      0.268      0.789   -1.79e-12    2.36e-12
dteday_05   -6.892e-13   1.06e-12     -0.653      0.514   -2.76e-12    1.38e-12
dteday_06   -1.065e-12   1.06e-12     -1.004      0.316   -3.15e-12    1.02e-12
dteday_07   -4.396e-13   1.06e-12     -0.416      0.677   -2.51e-12    1.63e-12
dteday_08   -1.148e-12   1.05e-12     -1.089      0.276   -3.22e-12    9.21e-13
dteday_09    2.323e-12   1.05e-12      2.203      0.028    2.52e-13    4.39e-12
dteday_10    1.172e-13   1.05e-12      0.111      0.911   -1.95e-12    2.18e-12
dteday_11     9.53e-13   1.05e-12      0.905      0.366   -1.12e-12    3.02e-12
mnth_4      9.486e-13   1.05e-12      0.903      0.367   -1.11e-12    3.01e-12
mnth_5      3.588e-13   1.13e-12      0.318      0.750   -1.86e-12    2.57e-12
mnth_6     -1.492e-12   1.08e-12     -1.380      0.168   -3.62e-12    6.31e-13
mnth_7     -3.126e-13   1.27e-12     -0.247      0.805    -2.8e-12    2.18e-12
mnth_8      1.023e-12   1.21e-12      0.844      0.399   -1.36e-12     3.4e-12
mnth_9      6.395e-14   9.83e-13      0.065      0.948   -1.87e-12    1.99e-12
mnth_10    -1.648e-12   1.07e-12     -1.535      0.125   -3.76e-12    4.61e-13
mnth_11    -5.684e-13   1.12e-12     -0.509      0.611   -2.76e-12    1.63e-12
mnth_12    -1.137e-13   9.67e-13     -0.118      0.906   -2.01e-12    1.78e-12
weekday_0  -9.024e-13   7.53e-13     -1.199      0.231   -2.38e-12    5.75e-13
weekday_1  -4.263e-13      5e-13     -0.853      0.394   -1.41e-12    5.55e-13
weekday_2  -2.887e-14   5.25e-13     -0.055      0.956   -1.06e-12       1e-12
weekday_3   3.388e-13   5.24e-13      0.647      0.518   -6.89e-13    1.37e-12
weekday_4  -5.951e-14   5.15e-13     -0.116      0.908   -1.07e-12    9.51e-13
weekday_5   3.428e-13   5.14e-13      0.667      0.505   -6.67e-13    1.35e-12
weekday_6  -5.116e-13   7.58e-13     -0.675      0.500      -2e-12    9.77e-13
cnt            1.0000   2.64e-16   3.79e+15      0.000       1.000       1.000
==============================================================================
Omnibus:                        8.586   Durbin-Watson:                   0.254
Prob(Omnibus):                  0.014   Jarque-Bera (JB):                5.780
Skew:                          -0.053   Prob(JB):                       0.0556
Kurtosis:                       2.577   Cond. No.                     1.61e+20
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.79e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

OLS Model Summary via R

```
> summary(linear_model)

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

Residuals:
    Min      1Q  Median      3Q     Max 
-3931.6  -636.6    53.6   720.6  2564.2 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2618.57     300.20   8.723  < 2e-16 ***
yr1          2032.75      84.35  24.098  < 2e-16 ***
holiday1     -549.06     251.24  -2.185   0.0293 *  
workingday1   -34.29      94.18  -0.364   0.7159    
temp         2371.67    1694.91   1.399   0.1623    
atemp        4365.64    1915.43   2.279   0.0230 *  
hum         -2538.40     309.18  -8.210 1.46e-15 ***
windspeed   -3884.40     583.29  -6.659 6.42e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1010 on 576 degrees of freedom
Multiple R-squared:  0.7303,    Adjusted R-squared:  0.727 
F-statistic: 222.8 on 7 and 576 DF,  p-value: < 2.2e-16
```

3.Conclusion

3.1 Mean Absolute Percentage Error

Conclusion will be taken based on MAPE score from Random Forest and Linear Regression whichever model give smaller MAPE is best suited for this problem statement.

MAPE by Linear Regression:

```
**Now find the MAE(Root mean absolute  error in %)**

In [296]: def MAPE(y_test,predictions):
              mape = np.mean(np.abs((y_test-predictions)/y_test))*100
              return mape

In [297]: MAPE(y['cnt'],predictions)

Out[297]: 21.39837956223615
```

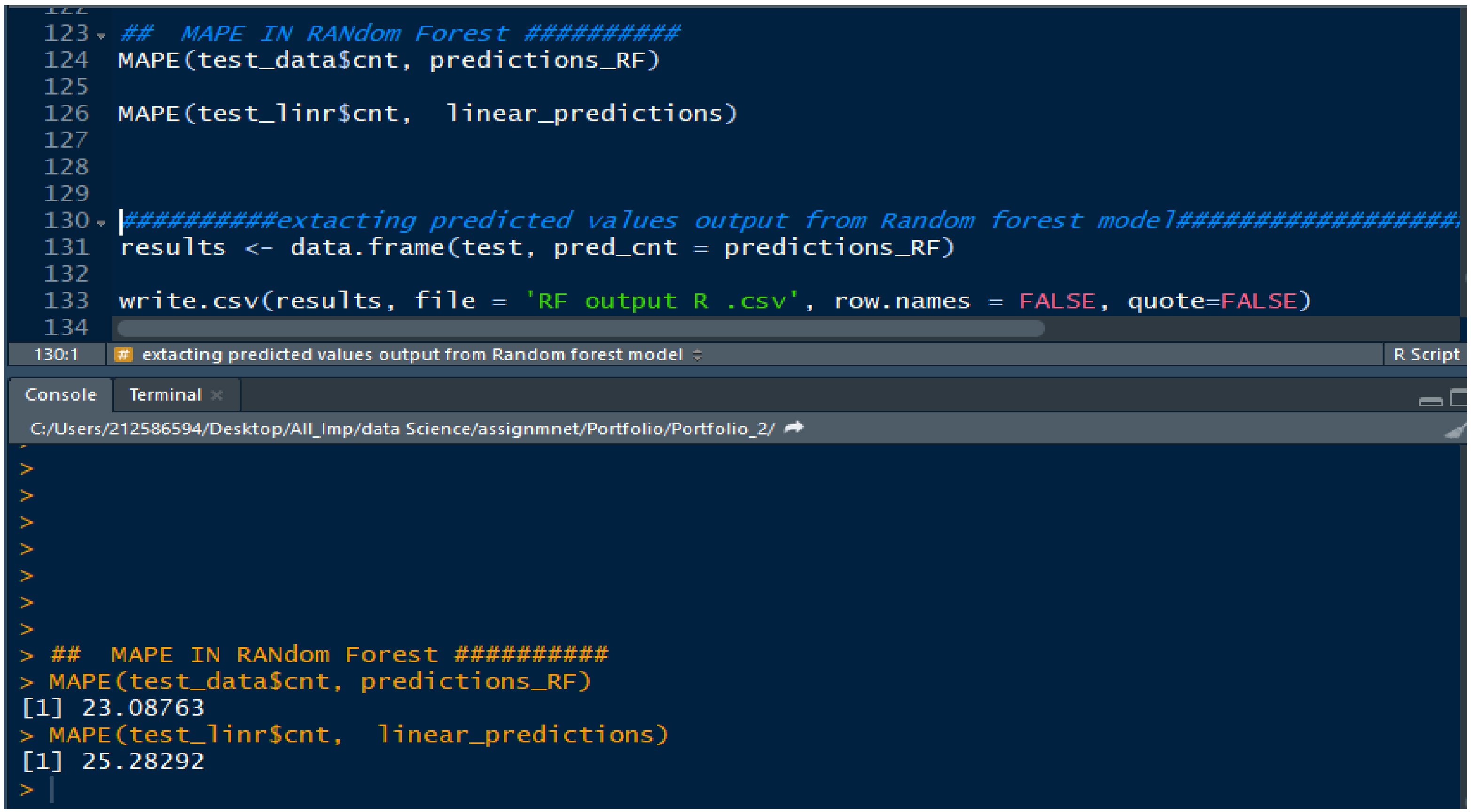
MAPE by Random Forest

```
: RFmodel = RandomForestRegressor(n_estimators = 200).fit(df1_RF.iloc[:,0:11], df1_RF.iloc[:,11])
  RF_Predictions = RFmodel.predict(test.iloc[:,0:11])

: ### Same MAPE function here

  MAPE(test['cnt'],RF_Predictions)

: 6.328395699324122
```

We clearly see in Python

The MAPE is less in Random Forest ,so this algo will give more accurate predictions.
By python

In R MAPE is less again for Random Forest while compared to Linear model



THE END