Project_Py_ML_LR_Evaluation_Beer Consumption Analysis

The objective of this project will be to demonstrate the impacts of variables on beer consumption in a given region and the consumption forecast for certain scenarios

Note-Rename the column names Respectively:

Data': 'Date', Temperature Media (C)': 'Medium Temp', Temperatura Minima (C)':'Min Temp', Temperatura Maxima (C)':'Max Temp', Precipitacao (mm)': 'Precipitation(mm)', Final de Semana':'End of week', Consumo de cerveja (litros)':'Beer consumption (liters)

importing the required libraries

In [230…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

load the given dataset

In [231…

```
df = pd.read_csv("Project_Py_ML_LR_Evaluation.csv")
df.head()
```

Out[231…

	Data	Temperatura Media (C)	Temperatura Minima (C)	Temperatura Maxima (C)	Precipitacao (mm)	Final de Semana	Consumo de cerveja (litros)
0	01/01/2015	27,3	23,9	32,5	0	0	25.461
1	02/01/2015	27,02	24,5	33,5	0	0	28.972
2	03/01/2015	24,82	22,4	29,9	0	1	30.814
3	04/01/2015	23,98	21,5	28,6	1,2	1	29.799
4	05/01/2015	23,82	21	28,3	0	0	28.900

rename all the column names

In [232…

```
df.columns = ['Date','Medium Temp','Min Temp','Max Temp','Precipitation(mm)','End of week', 'Beer consumption (li
df.head()
# for renaming a specific column- df.rename(columns = {'old1':'new2', 'old2':new2}, inplace=True)
```

Out[232…

	Date	Medium Temp	Min Temp	Max Temp	Precipitation(mm)	End of week	Beer consumption (liters)
0	01/01/2015	27,3	23,9	32,5	0	0	25.461
1	02/01/2015	27,02	24,5	33,5	0	0	28.972
2	03/01/2015	24,82	22,4	29,9	0	1	30.814
3	04/01/2015	23,98	21,5	28,6	1,2	1	29.799
4	05/01/2015	23,82	21	28,3	0	0	28.900

In [233…

```
df.info() # getting the datframe information like null count and datatype of each column.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 7 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Date                       365 non-null    object
 1   Medium Temp                365 non-null    object
 2   Min Temp                   365 non-null    object
 3   Max Temp                   365 non-null    object
 4   Precipitation(mm)          365 non-null    object
 5   End of week                365 non-null    int64
 6   Beer consumption (liters)  365 non-null    float64
```

```
dtypes: float64(1), int64(1), object(5)
memory usage: 20.1+ KB
```

Replacing , with . in values of continuous variables of the dataframe

In [234…
```
df['Medium Temp'].replace(',','.', inplace=True, regex=True)
```

In [235…
```
df['Min Temp'].replace(',','.', inplace=True, regex=True)
```

In [237…
```
df['Max Temp'].replace(',','.', inplace=True, regex=True)
```

In [238…
```
df['Precipitation(mm)'].replace(',','.', inplace=True, regex=True)
```

In [239…
```
df.head()
```

Out[239…

	Date	Medium Temp	Min Temp	Max Temp	Precipitation(mm)	End of week	Beer consumption (liters)
0	01/01/2015	27.3	23.9	32.5	0	0	25.461
1	02/01/2015	27.02	24.5	33.5	0	0	28.972
2	03/01/2015	24.82	22.4	29.9	0	1	30.814
3	04/01/2015	23.98	21.5	28.6	1.2	1	29.799
4	05/01/2015	23.82	21	28.3	0	0	28.900

changing the datatypes of continuous variable from object to float

In [240…
```
df['Medium Temp'] = df['Medium Temp'].astype('float')
```

In [241…
```
df['Min Temp'] = df['Min Temp'].astype('float')
```

In [242…
```
df['Max Temp'] = df['Max Temp'].astype('float')
```

In [243…
```
df['Precipitation(mm)'] = df['Precipitation(mm)'].astype('float')
```

Splitting the 'Date' variable in separate columns of Day and Month

In [244…
```
df['Day']= pd.to_datetime(df.Date, format="%d/%m/%Y").dt.day
```

In [245…
```
df['Month'] = pd.to_datetime(df.Date, format = "%d/%m/%Y").dt.month
```

In [246…
```
df.info() # getting the dataframe information after above changes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Date                       365 non-null    object 
 1   Medium Temp                365 non-null    float64
 2   Min Temp                   365 non-null    float64
 3   Max Temp                   365 non-null    float64
 4   Precipitation(mm)          365 non-null    float64
 5   End of week                365 non-null    int64  
 6   Beer consumption (liters)  365 non-null    float64
 7   Day                        365 non-null    int64  
 8   Month                      365 non-null    int64  
dtypes: float64(5), int64(3), object(1)
memory usage: 25.8+ KB
```

Dropping the Date column, since is it splitted in Day and Month

In [247…

```
df.drop(["Date"], axis = 1, inplace = True)
```

Describing the data

In [248…

```
df.describe() # getting the statistical information from the continuous variables of dataframes
```

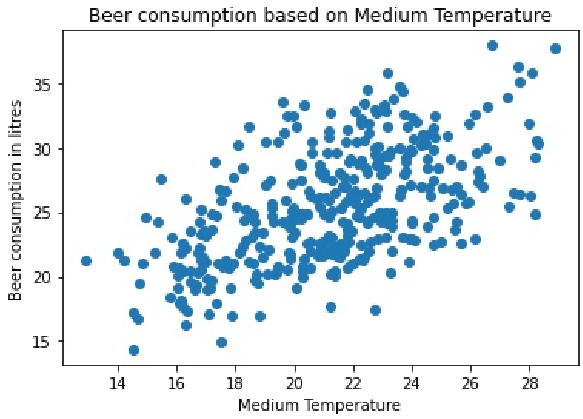
Out[248…

	Medium Temp	Min Temp	Max Temp	Precipitation(mm)	End of week	Beer consumption (liters)	Day	Month
count	365.000000	365.000000	365.000000	365.000000	365.000000	365.000000	365.000000	365.000000
mean	21.226356	17.461370	26.611507	5.196712	0.284932	25.401367	15.720548	6.526027
std	3.180108	2.826185	4.317366	12.417844	0.452001	4.399143	8.808321	3.452584
min	12.900000	10.600000	14.500000	0.000000	0.000000	14.343000	1.000000	1.000000
25%	19.020000	15.300000	23.800000	0.000000	0.000000	22.008000	8.000000	4.000000
50%	21.380000	17.900000	26.900000	0.000000	0.000000	24.867000	16.000000	7.000000
75%	23.280000	19.600000	29.400000	3.200000	1.000000	28.631000	23.000000	10.000000
max	28.860000	24.500000	36.500000	94.800000	1.000000	37.937000	31.000000	12.000000

Plotting some of the variables to find the relation with the Beer consumption variable

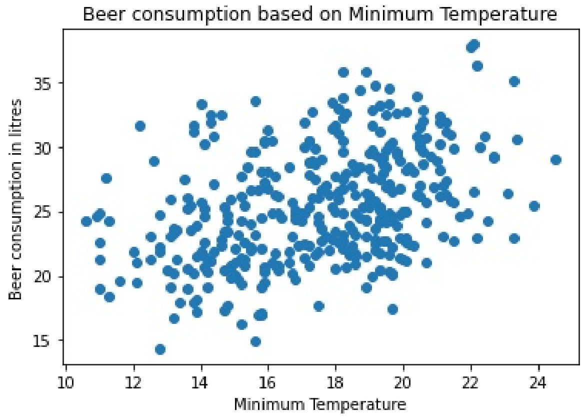
In [249…

```
plt.scatter(df['Medium Temp'], df['Beer consumption (liters)'])
plt.xlabel('Medium Temperature')
plt.ylabel('Beer consumption in litres')
plt.title('Beer consumption based on Medium Temperature');
```



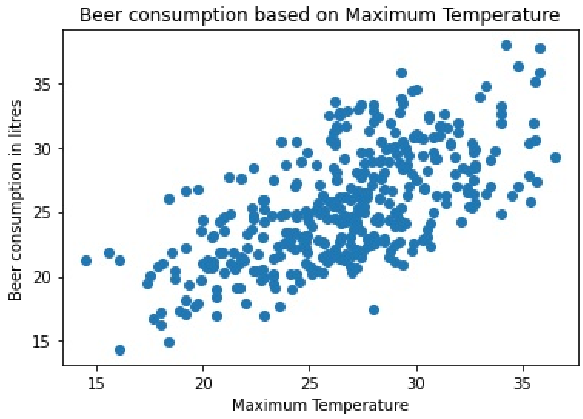
In [250…

```
plt.scatter(df['Min Temp'], df['Beer consumption (liters)'])
plt.xlabel('Minimum Temperature')
plt.ylabel('Beer consumption in litres')
plt.title('Beer consumption based on Minimum Temperature');
```



In [251…

```
plt.scatter(df['Max Temp'], df['Beer consumption (liters)'])
plt.xlabel('Maximum Temperature')
plt.ylabel('Beer consumption in litres')
plt.title('Beer consumption based on Maximum Temperature');
```



In [252…

```
df.corr() # gives the correlation among each continuous variable of the dataframe
```

Out[252…

	Medium Temp	Min Temp	Max Temp	Precipitation(mm)	End of week	Beer consumption (liters)	Day	Month
Medium Temp	1.000000	0.862752	0.922513	0.024416	-0.050803	0.574615	0.012382	-0.103169
Min Temp	0.862752	1.000000	0.672929	0.098625	-0.059534	0.392509	-0.011206	-0.172923
Max Temp	0.922513	0.672929	1.000000	-0.049305	-0.040258	0.642672	0.035079	-0.074866
Precipitation(mm)	0.024416	0.098625	-0.049305	1.000000	0.001587	-0.193784	-0.003414	0.007089
End of week	-0.050803	-0.059534	-0.040258	0.001587	1.000000	0.505981	0.006254	-0.006526
Beer consumption (liters)	0.574615	0.392509	0.642672	-0.193784	0.505981	1.000000	0.025969	0.039908
Day	0.012382	-0.011206	0.035079	-0.003414	0.006254	0.025969	1.000000	0.011893
Month	-0.103169	-0.172923	-0.074866	0.007089	-0.006526	0.039908	0.011893	1.000000

Starting creating the Model

splitting the data into predictor(x) and target (y)

In [253…

```
x = df.drop('Beer consumption (liters)',axis=1)
y = df['Beer consumption (liters)']
```

In [254…

```
from sklearn.model_selection import train_test_split
```

creating the train and test data of the predictor and target

In [255…

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

creating the linear regression model

In [256…

```
lrm = LinearRegression()
```

In [257…

```
lrm.fit(x_train, y_train)
```

Out[257…

```
LinearRegression()
```

In [258…

```
lrm.score(x_train, y_train) #training score of the created model
```

Out[258…

```
0.7222896489940203
```

In [259…

```
lrm.score(x_test, y_test) #test score of the created model
```

Out[259…

```
0.7406013522605527
```

Trying feature selection to select the best variables for prediction and try increasing the model accuracy

In [260…

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_regression
```

In [261…

```
bestfeatures = SelectKBest(score_func=f_regression, k='all')
```

In [262…

```
fit = bestfeatures.fit(x,y)
```

In [263…

```
fit.scores_
```

Out[263…

```
array([1.78938298e+02, 6.61100241e+01, 2.55428041e+02, 1.41633735e+01,
       1.24913906e+02, 2.44963789e-01, 5.79047775e-01])
```

In [264…

```
x.columns
```

Out[264…

```
Index(['Medium Temp', 'Min Temp', 'Max Temp', 'Precipitation(mm)',
       'End of week', 'Day', 'Month'],
      dtype='object')
```

In [265…

```
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
```

In [266…

```
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores
```

Out[266…

	0	0
0	Medium Temp	178.938298
1	Min Temp	66.110024
2	Max Temp	255.428041
3	Precipitation(mm)	14.163373
4	End of week	124.913906
5	Day	0.244964
6	Month	0.579048

In [267…

```
featureScores.columns = ['effect_on_consumption','Score']
featureScores
```

Out[267…

	effect_on_consumption	Score
0	Medium Temp	178.938298
1	Min Temp	66.110024
2	Max Temp	255.428041
3	Precipitation(mm)	14.163373
4	End of week	124.913906
5	Day	0.244964
6	Month	0.579048

In [268…

```
featureScores.nlargest(3,'Score') #getting these best three variable among 7 after feature selection
```

Out[268…

	effect_on_consumption	Score
2	Max Temp	255.428041

0	Medium Temp	178.938298
4	End of week	124.913906

In [269...
```
x1 = df[['Max Temp','Medium Temp','End of week']] #predictor suggested by feature selection
y1 = df['Beer consumption (liters)']
```

In [270...
```
x1_train, x1_test, y1_train, y1_test = train_test_split(x1,y1,test_size=0.2,random_state=42)
```

In [271...
```
lrm1 = LinearRegression()
```

In [272...
```
lrm1.fit(x1_train, y1_train)
```

Out[272...
```
LinearRegression()
```

In [273...
```
lrm1.score(x1_train, y1_train)
```

Out[273...
```
0.6954059612625987
```

In [274...
```
lrm1.score(x1_test, y1_test) # accuracy didn't improve using the suggested 3 varibales after feature selection
```

Out[274...
```
0.6886564844629807
```

In [275...
```
featureScores.nlargest(5,'Score') #getting top 5 features and will try model creation using these features to get
```

Out[275...

	effect_on_consumption	Score
2	Max Temp	255.428041
0	Medium Temp	178.938298
4	End of week	124.913906
1	Min Temp	66.110024
3	Precipitation(mm)	14.163373

In [276...
```
x1 = df[['Max Temp','Medium Temp','End of week','Min Temp','Precipitation(mm)']] #predictor suggested by feature
y1 = df['Beer consumption (liters)']
```

In [277...
```
x1_train, x1_test, y1_train, y1_test = train_test_split(x1,y1,test_size=0.2,random_state=42)
```

In [278...
```
lrm1 = LinearRegression()
lrm1.fit(x1_train, y1_train)
```

Out[278...
```
LinearRegression()
```

In [279...
```
lrm1.score(x1_train, y1_train)
```

Out[279...
```
0.7104068706547428
```

In [280...
```
lrm1.score(x1_test, y1_test) # Accuracy score increased by 6 percent after using 5 varibles (feature select) in t
```

Out[280...
```
0.7427938181737366
```

Trying Standard Scaling on train and test data of all predictors in order to try getting more accuracy

Trying Standard Scaling on train and test data of all predictors in order to try getting more accuracy

In [281…
```
from sklearn.preprocessing import StandardScaler
```

In [282…
```
s = StandardScaler()
```

In [283…
```
x_train = s.fit_transform(x_train)
x_test = s.fit_transform(x_test)
```

In [284…
```
col = x.columns
```

In [285…
```
x_train = pd.DataFrame(x_train, columns=[col])

x_test = pd.DataFrame(x_test, columns=[col])

x_train.head()
```

Out[285…

	Medium Temp	Min Temp	Max Temp	Precipitation(mm)	End of week	Day	Month
0	-0.577526	-1.569856	0.031792	-0.425154	-0.593171	1.725816	0.125584
1	-0.329620	0.237186	-0.590171	3.512292	-0.593171	-0.992053	1.616253
2	-2.040166	-1.569856	-2.386953	-0.425154	-0.593171	0.706615	0.125584
3	0.252957	0.445691	-0.198564	1.568281	-0.593171	0.140392	-1.066952
4	0.581431	0.341439	0.768934	-0.425154	-0.593171	1.159593	-1.066952

In [286…
```
lrm2 = LinearRegression()
```

In [287…
```
lrm2.fit(x_train, y_train)
```

Out[287…
```
LinearRegression()
```

In [288…
```
lrm2.score(x_train, y_train)
```

Out[288…
```
0.7222896489940203
```

In [289…
```
lrm2.score(x_test, y_test) # scaling didn't worked in getting the more accuracy, on the contrary the accuracy dec
```

Out[289…
```
0.6777625864500789
```

Trying the different regressors

In [290…
```
X = df.drop('Beer consumption (liters)',axis=1)
Y = df['Beer consumption (liters)']
```

In [291…
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [292…
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.svm import SVR
```

In [293…
```
model = [DecisionTreeRegressor,SVR,RandomForestRegressor,KNeighborsRegressor,AdaBoostRegressor]

for mod in model:
    reg = mod()
    reg = reg.fit(X_train,Y_train)
    print(mod , 'accuracy',reg.score(X_test,Y_test))
```

```
<class 'sklearn.tree._classes.DecisionTreeRegressor'> accuracy 0.47627933938585965
<class 'sklearn.svm._classes.SVR'> accuracy 0.359657180864009
<class 'sklearn.ensemble._forest.RandomForestRegressor'> accuracy 0.6931273024263127
<class 'sklearn.neighbors._regression.KNeighborsRegressor'> accuracy 0.3153496691022142
<class 'sklearn.ensemble._weight_boosting.AdaBoostRegressor'> accuracy 0.7031416451200945
```

In [294…

```
# on scaled data
model_s = [DecisionTreeRegressor,SVR,RandomForestRegressor,KNeighborsRegressor,AdaBoostRegressor]

for mod in model_s:
    reg = mod()
    reg = reg.fit(x_train,y_train)
    print(mod , 'accuracy',reg.score(x_test,y_test))
```

```
<class 'sklearn.tree._classes.DecisionTreeRegressor'> accuracy 0.23763566840738326
<class 'sklearn.svm._classes.SVR'> accuracy 0.6621135292581894
<class 'sklearn.ensemble._forest.RandomForestRegressor'> accuracy 0.6727914207125629
<class 'sklearn.neighbors._regression.KNeighborsRegressor'> accuracy 0.7161239938784414
<class 'sklearn.ensemble._weight_boosting.AdaBoostRegressor'> accuracy 0.6988167270799108
```

None of the above regressor has better score than the linear regression model created earlier.

creating the linear regression model with combinations of variables in the predictor to get the best accuracy

In [297…

```
x3 = df.drop(['Beer consumption (liters)','Day','Month'],axis=1)
y3 = df['Beer consumption (liters)']
```

In [298…

```
x3_train, x3_test, y3_train, y3_test = train_test_split(x3, y3, test_size=0.2, random_state=42)
```

In [299…

```
lrm3 = LinearRegression()
```

In [300…

```
lrm3.fit(x3_train, y3_train)
```

Out[300…

```
LinearRegression()
```

In [301…

```
lrm3.score(x3_train, y3_train)
```

Out[301…

```
0.7104068706547428
```

In [302…

```
lrm3.score(x3_test, y3_test)
```

Out[302…

```
0.7427938181737361
```

In [325…

```
x4 = df[['Medium Temp','Max Temp','Min Temp','End of week']]
y4 = df['Beer consumption (liters)']
```

In [326…

```
x4_train, x4_test, y4_train, y4_test = train_test_split(x4, y4, test_size=0.2, random_state=42)
```

In [327…

```
lrm4 = LinearRegression()
```

In [328…

```
lrm4.fit(x4_train, y4_train)
```

Out[328…

```
LinearRegression()
```

In [329…

```
lrm4.score(x4_train, y4_train)
```

Out[329…

```
0.6961296970163136
```

In [330…

```
lrm4.score(x4_test, y4_test)
```

Out[330… 0.6861100648174152

In [309…

```
x5 = df[['Max Temp','Medium Temp','End of week']]
y5 = df['Beer consumption (liters)']
```

In [310…

```
x5_train, x5_test, y5_train, y5_test = train_test_split(x5, y5, test_size=0.2, random_state=42)
```

In [311…

```
lrm5 = LinearRegression()
lrm5.fit(x5_train, y5_train)
```

Out[311… LinearRegression()

In [312…

```
lrm5.score(x5_train, y5_train)
```

Out[312… 0.6954059612625987

In [313…

```
lrm5.score(x5_test, y5_test)
```

Out[313… 0.6886564844629807

After trying multiple combinations for predictor variables , got the best score for below predictor variables combination.

In [314…

```
x6 = df[['Medium Temp','End of week','Max Temp','Precipitation(mm)']] # predictor
y6 = df['Beer consumption (liters)'] # target
```

In [315…

```
x6_train, x6_test, y6_train, y6_test = train_test_split(x6, y6, test_size=0.2, random_state=42)
```

In [316…

```
lrm6 = LinearRegression()
lrm6.fit(x6_train, y6_train)
```

Out[316… LinearRegression()

model training accuracy

In [317…

```
lrm6.score(x6_train, y6_train)
```

Out[317… 0.7101409516827788

model testing accuracy

In [318…

```
lrm6.score(x6_test, y6_test)
```

Out[318… 0.7443500987830509

In [320…

```
y6_prediction = lrm6.predict(x6_test) # prediction
```

mean squared error

mean squared error

In [321…

```
print('mean squared error is: ', mean_squared_error(y6_test, y6_prediction))
```

```
mean squared error is:  5.66496605013936
```

mean absolute error

In [322…

```
print('mean absolute error is: ', mean_absolute_error(y6_test, y6_prediction))
```

```
mean absolute error is:  2.05538301484382
```

root mean squared error

In [323…

```
print('root mean squared error is: ', np.sqrt(mean_squared_error(y6_test, y6_prediction)))
```

```
root mean squared error is:  2.3801189151257462
```

r2 score

In [324…

```
print('r2 score is: ', r2_score(y6_test, y6_prediction))
```

```
r2 score is:  0.7443500987830509
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js