Linear Regression

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df1 = pd.read_csv("weatherAUS.csv")
```

In [3]:
```
df1.isnull().sum()
```

Out[3]:
```
Date                0
Location            0
MinTemp          1485
MaxTemp          1261
Rainfall         3261
Evaporation     62790
Sunshine        69835
WindGustDir     10326
WindGustSpeed   10263
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am     1767
WindSpeed3pm     3062
Humidity9am      2654
Humidity3pm      4507
Pressure9am     15065
Pressure3pm     15028
Cloud9am        55888
Cloud3pm        59358
Temp9am          1767
Temp3pm          3609
RainToday        3261
RainTomorrow     3267
dtype: int64
```

In [4]:
```
df1.dropna(subset = ['RainTomorrow'],inplace =True)
```

In [5]:
```
df1['Date'].dtypes
```

Out[5]:
```
dtype('O')
```

We can see that the data type of `Date` variable is object. I will parse the date currently coded as object into datetime format.

In [6]:
```
df1['Date'] = pd.to_datetime(df1['Date'])
```

In [7]:
```
df1['Year'] = df1['Date'].dt.year

df1['Year'].head()
```

Out[7]:
```
0    2008
1    2008
2    2008
3    2008
4    2008
Name: Year, dtype: int64
```

In [8]:
```
df1['Month'] = df1['Date'].dt.month

df1['Month'].head()
```

Out[8]:
```
0    12
1    12
2    12
3    12
4    12
Name: Month, dtype: int64
```

In [9]:
```
df1['Day'] = df1['Date'].dt.day

df1['Day'].head()
```

Out[9]:
```
0    1
1    2
2    3
3    4
4    5
Name: Day, dtype: int64
```

In [10]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 142193 entries, 0 to 145458
Data columns (total 26 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           142193 non-null  datetime64[ns]
 1   Location       142193 non-null  object
 2   MinTemp        141556 non-null  float64
 3   MaxTemp        141871 non-null  float64
 4   Rainfall       140787 non-null  float64
 5   Evaporation    81350 non-null   float64
 6   Sunshine       74377 non-null   float64
 7   WindGustDir    132863 non-null  object
 8   WindGustSpeed  132923 non-null  float64
 9   WindDir9am     132180 non-null  object
 10  WindDir3pm     138415 non-null  object
 11  WindSpeed9am   140845 non-null  float64
 12  WindSpeed3pm   139563 non-null  float64
 13  Humidity9am    140419 non-null  float64
 14  Humidity3pm    138583 non-null  float64
 15  Pressure9am    128179 non-null  float64
 16  Pressure3pm    128212 non-null  float64
 17  Cloud9am       88536 non-null   float64
 18  Cloud3pm       85099 non-null   float64
 19  Temp9am        141289 non-null  float64
 20  Temp3pm        139467 non-null  float64
 21  RainToday      140787 non-null  object
 22  RainTomorrow   142193 non-null  object
 23  Year           142193 non-null  int64
 24  Month          142193 non-null  int64
 25  Day            142193 non-null  int64
dtypes: datetime64[ns](1), float64(16), int64(3), object(6)
memory usage: 29.3+ MB
```

In [11]:
```
df1.drop('Date', axis=1, inplace = True)
```

In [12]:
```
df1.head()
```

Out[12]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed
0	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0
1	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0
2	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0
3	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0
4	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0

5 rows × 25 columns

In [13]:
```
def remove_outlier(i,df1):
    q1=df1[i].quantile(0.25)
    q3=df1[i].quantile(0.75)
    iqr=q3-q1
    ll=q1-3*iqr
    ul=q3+3*iqr
    return df1[~((df1[i]<ll) | (df1[i]>ul))]
```

```
col_list = ['MinTemp' ,'MaxTemp' ,'Rainfall','Evaporation','WindGustSpeed','WindSpe
            'Pressure9am', 'Pressure3pm' , 'Temp9am' ,'Temp3pm']

for i in col_list:
    df1 = remove_outlier(i,df1)
```

In [14]:
```
df1.shape
```

Out[14]: (121051, 25)

In [15]:
```
categorical = [col for col in df1.columns if df1[col].dtypes == 'O']
```

In [16]:
```
import category_encoders as ce
encoder2 = ce.OrdinalEncoder(cols=categorical)
df1 = encoder2.fit_transform(df1)
```

In [17]:
```
df1.head()
```

Out[17]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed
0	1	13.4	22.9	0.6	NaN	NaN	1	44.0
1	1	7.4	25.1	0.0	NaN	NaN	2	44.0
2	1	12.9	25.7	0.0	NaN	NaN	3	46.0
3	1	9.2	28.0	0.0	NaN	NaN	4	24.0
4	1	17.5	32.3	1.0	NaN	NaN	1	41.0

5 rows × 25 columns

In [18]:
```
df1.dtypes
```

Out[18]:
```
Location          int32
MinTemp         float64
MaxTemp         float64
Rainfall        float64
Evaporation     float64
Sunshine        float64
WindGustDir       int32
WindGustSpeed   float64
WindDir9am        int32
WindDir3pm        int32
WindSpeed9am    float64
WindSpeed3pm    float64
Humidity9am     float64
Humidity3pm     float64
Pressure9am     float64
Pressure3pm     float64
Cloud9am        float64
Cloud3pm        float64
Temp9am         float64
Temp3pm         float64
RainToday         int32
RainTomorrow      int32
Year              int64
Month             int64
Day               int64
dtype: object
```

In [19]:
```
df1.corr()
```

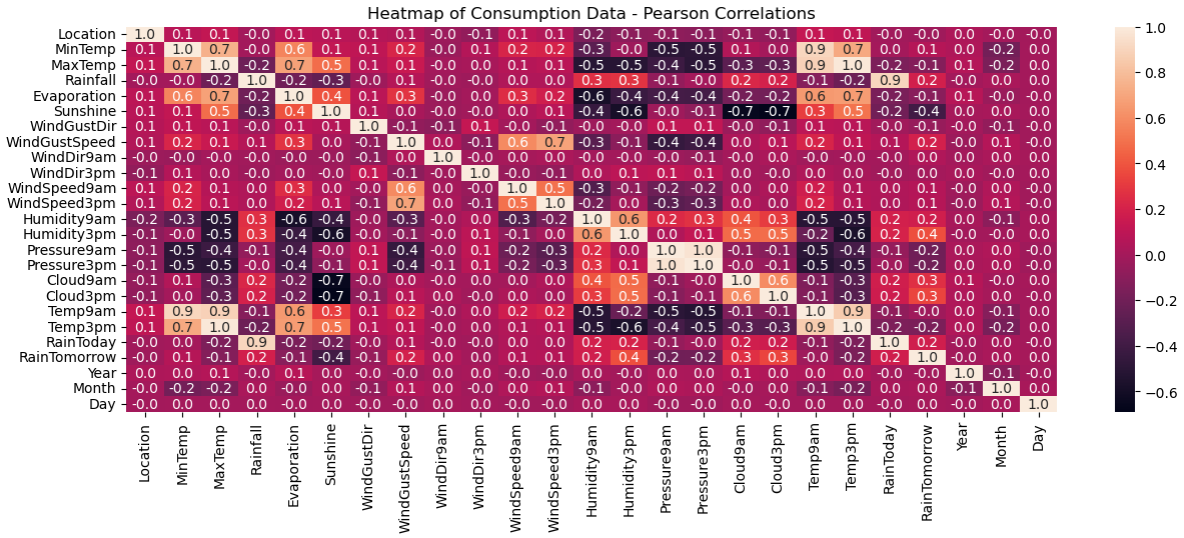
Out[19]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir
Location	1.000000	0.082188	0.117533	-0.004123	0.091788	0.080229	0.067969
MinTemp	0.082188	1.000000	0.743245	-0.008664	0.565739	0.117722	0.106106
MaxTemp	0.117533	0.743245	1.000000	-0.212981	0.679310	0.483600	0.079916
Rainfall	-0.004123	-0.008664	-0.212981	1.000000	-0.206757	-0.254538	-0.023854
Evaporation	0.091788	0.565739	0.679310	-0.206757	1.000000	0.386855	0.083937
Sunshine	0.080229	0.117722	0.483600	-0.254538	0.386855	1.000000	0.068806
WindGustDir	0.067969	0.106106	0.079916	-0.023854	0.083937	0.068806	1.000000
WindGustSpeed	0.050128	0.218491	0.143036	0.065490	0.279760	0.013017	-0.074017
WindDir9am	-0.033991	-0.039051	-0.000636	-0.013690	-0.029226	-0.034535	-0.089680
WindDir3pm	-0.072698	0.056846	0.014997	-0.022975	0.000700	-0.011929	0.123178
WindSpeed9am	0.093215	0.208871	0.063223	0.046736	0.253175	0.040674	-0.008284
WindSpeed3pm	0.054538	0.206275	0.087922	0.045687	0.179118	0.078666	-0.074224
Humidity9am	-0.156558	-0.291315	-0.516474	0.262790	-0.582430	-0.439544	-0.025111
Humidity3pm	-0.094278	-0.024130	-0.517003	0.274678	-0.429953	-0.584378	-0.003455
Pressure9am	-0.088434	-0.495142	-0.420757	-0.063110	-0.381747	-0.025264	0.109496
Pressure3pm	-0.096048	-0.490661	-0.494006	-0.001805	-0.391071	-0.080734	0.113731
Cloud9am	-0.061713	0.056368	-0.282314	0.222417	-0.183739	-0.654960	-0.041749
Cloud3pm	-0.072157	0.000219	-0.268410	0.200801	-0.190198	-0.688850	-0.079204
Temp9am	0.127117	0.903366	0.884857	-0.118867	0.644035	0.318530	0.093154
Temp3pm	0.107554	0.715376	0.984443	-0.216747	0.660576	0.505064	0.082829
RainToday	-0.014824	0.002819	-0.150403	0.903416	-0.161984	-0.198832	-0.015089
RainTomorrow	-0.017184	0.063351	-0.125049	0.184564	-0.105448	-0.397923	-0.052188
Year	0.032071	0.039499	0.059224	-0.008338	0.072619	0.004325	-0.004945
Month	-0.000835	-0.190647	-0.156971	0.020455	-0.037800	0.010696	-0.067340
Day	-0.003642	0.003236	0.002283	0.000408	-0.009165	0.000113	-0.010692

25 rows × 25 columns

In [20]:
```
fig, ax = plt.subplots(figsize=(15, 5))
correlations = df1.corr()
# Rohit Bhimani
# annot=True displays the correlation values

sns.heatmap(correlations, annot=True , fmt=".1f").set(title='Heatmap of Consumptior
```



In [21]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df2 = scaler.fit_transform(df1)
df1 = pd.DataFrame(df2, columns=df1.columns)
```

In [22]:
```
X1 = df1.drop(['RainTomorrow'], axis=1)

y1 = df1['RainTomorrow']
```

Split data into separate training and test set

In [23]:
```
from sklearn.model_selection import train_test_split

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size = 0.2, ra
```

In [24]:
```
X1_train.shape, X1_test.shape
```

Out[24]:
```
((96840, 24), (24211, 24))
```

Feature Engineering

In [25]:
```
categorical1 = [col for col in X1_train.columns if X1_train[col].dtypes == 'O']
```

In [26]:
```
numerical1 = [col for col in X1_train.columns if X1_train[col].dtypes != 'O']
```

In [27]:
```
# impute missing values in X_train and X_test with respective column median in X_tr

for df2 in [X1_train, X1_test]:
    for col in numerical1:
        col_median=X1_train[col].median()
        df2[col].fillna(col_median, inplace=True)
```

Engineering missing values in categorical variables

In [28]:
```
# impute missing categorical variables with most frequent value
```

```
for df3 in [X1_train, X1_test]:
    df3['WindGustDir'].fillna(X1_train['WindGustDir'].mode()[0], inplace=True)
    df3['WindDir9am'].fillna(X1_train['WindDir9am'].mode()[0], inplace=True)
    df3['WindDir3pm'].fillna(X1_train['WindDir3pm'].mode()[0], inplace=True)
    df3['RainToday'].fillna(X1_train['RainToday'].mode()[0], inplace=True)
```

In [29]:
```
X1_train[numerical1].describe()
```

Out[29]:

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	Winc
count	96840.000000	96840.000000	96840.000000	96840.000000	96840.000000	96840.000000	9684(
mean	0.492659	0.482982	0.540338	0.078472	0.240927	0.586001	(
std	0.291248	0.151736	0.137494	0.193910	0.122106	0.185015	(
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	(
25%	0.229167	0.375000	0.437743	0.000000	0.200000	0.606897	(
50%	0.479167	0.478774	0.531128	0.000000	0.227273	0.620690	(
75%	0.750000	0.591981	0.636187	0.000000	0.254545	0.634483	(
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1

8 rows × 24 columns

In [30]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X1_train, y1_train)
```

Out[30]:
```
LinearRegression()
```

In [31]:
```
regressor.intercept_
```

Out[31]:
```
0.08721288900808244
```

In [32]:
```
regressor.coef_
```

Out[32]:
```
array([-1.23849086e-05, -1.11245883e-01,  7.00075166e-02,  1.39751818e-02,
        5.21667485e-02, -2.82094769e-01, -9.82690533e-03,  6.28236452e-01,
        2.42754921e-02, -8.76561506e-03, -2.55242522e-02, -2.79161474e-01,
       -8.11894216e-02,  7.75396520e-01,  9.81618245e-01, -1.51822467e+00,
       -1.89887743e-02,  9.13283491e-02, -1.34434249e-01,  1.68959249e-01,
        1.89300020e-01,  1.40405979e-02,  1.38126912e-02, -1.56348021e-03])
```

In [33]:
```
y1_pred = regressor.predict(X1_test)
```

In [34]:
```
regressor.score(X1_train , y1_train)
```

Out[34]:
```
0.25824604814730534
```

In [35]:
```
regressor.score(X1_test , y1_test)
```

Out[35]:
```
0.25889665229486236
```

In [36]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(y1_test, y1_pred)
```

```
mse = mean_squared_error(y1_test, y1_pred)
error2 = rmse = np.sqrt(mse)

print(f'Mean absolute error: {mae:.2f}')
print(f'Mean squared error: {mse:.2f}')
print(f'Root mean squared error: {rmse:.2f}')
```

```
Mean absolute error: 0.23
Mean squared error: 0.11
Root mean squared error: 0.33
```

In [36]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(y1_test, y1_pred)
mse = mean_squared_error(y1_test, y1_pred)
error2 = rmse = np.sqrt(mse)

print(f'Mean absolute error: {mae:.2f}')
print(f'Mean squared error: {mse:.2f}')
print(f'Root mean squared error: {rmse:.2f}')
```

```
Mean absolute error: 0.23
Mean squared error: 0.11
Root mean squared error: 0.33
```

In [ ]: