Machine Learning 💻

Linear Regression

> *By ⟹ PRINCE*❤️

Mounting Google Drive

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Importing some important libraries

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Loading and Reading Dataset

```
df = pd.read_csv('/content/drive/MyDrive/Weather.csv')
df.head()
```

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snowfall	P
0	10001	7/1/1942	1.016	NaN	25.555556	22.222222	23.888889	0.0	
1	10001	7/2/1942	0	NaN	28.888889	21.666667	25.555556	0.0	
2	10001	7/3/1942	2.54	NaN	26.111111	22.222222	24.444444	0.0	
3	10001	7/4/1942	2.54	NaN	26.666667	22.222222	24.444444	0.0	
4	10001	7/5/1942	0	NaN	26.666667	21.666667	24.444444	0.0	

5 rows × 30 columns

```
df.tail()
```

	STA	Date	Precip	WindGustSpd	MaxTemp	MinTemp	MeanTemp	Snow
119035	82506	12/27/1945	0	NaN	28.333333	18.333333	23.333333	
119036	82506	12/28/1945	9.906	NaN	29.444444	18.333333	23.888889	
119037	82506	12/29/1945	0	NaN	28.333333	18.333333	23.333333	
119038	82506	12/30/1945	0	NaN	28.333333	18.333333	23.333333	
119039	82506	12/31/1945	0	NaN	29.444444	17.222222	23.333333	

5 rows × 30 columns

```
df.shape
```

```
(119040, 30)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119040 entries, 0 to 119039
Data columns (total 30 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   STA          119040 non-null  int64  
 1   Date         119040 non-null  object 
 2   Precip       119040 non-null  object 
 3   WindGustSpd  532 non-null     float64
 4   MaxTemp      119040 non-null  float64
 5   MinTemp      119040 non-null  float64
 6   MeanTemp     119040 non-null  float64
 7   Snowfall     117877 non-null  object 
```

```
 8   PoorWeather  34237 non-null   object
 9   YR           119040 non-null  int64
 10  MO           119040 non-null  int64
 11  DA           119040 non-null  int64
 12  PRCP         117108 non-null  object
 13  DR           533 non-null     float64
 14  SPD          532 non-null     float64
 15  MAX          118566 non-null  float64
 16  MIN          118572 non-null  float64
 17  MEA          118542 non-null  float64
 18  SNF          117877 non-null  object
 19  SND          5563 non-null    float64
 20  FT           0 non-null       float64
 21  FB           0 non-null       float64
 22  FTI          0 non-null       float64
 23  PGT          525 non-null     float64
 24  TSHDSBRSGF   34237 non-null   object
 25  SD3          0 non-null       float64
 26  RHX          0 non-null       float64
 27  RHN          0 non-null       float64
 28  RVG          0 non-null       float64
 29  WTE          0 non-null       float64
dtypes: float64(19), int64(4), object(7)
memory usage: 27.2+ MB
```

Based on Minimum Temperature we have to predict Maximum Temperature.

Define independent variable :

```
# Define independent variable
x = df['MinTemp'].values.reshape(-1,1)
x
```

```
array([[22.22222222],
       [21.66666667],
       [22.22222222],
       ...,
       [18.33333333],
       [18.33333333],
       [17.22222222]])
```

Define dependent variable :

```
# Define dependent variable
y = df['MaxTemp'].values.reshape(-1,1)
y
```

```
array([[25.55555556],
       [28.88888889],
       [26.11111111],
       ...,
       [28.33333333],
       [28.33333333],
       [29.44444444]])
```

Train_Test_Split :

Used to split the dataset to train and test

```
# Train_Test_Split  : Used to split the dataset to train and test
from sklearn.model_selection import train_test_split     # sklearn => Scikit Learn
```

Splitting

```
# Splitting at 70:30 ratio
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.3, train_size = 0.7, random_state = 1)
# train_size, random_state can be exclude
```

To check ⟶

```
# To check
x_train.shape      # 0.7 * 119040
```

```
(83328, 1)
```

```
y_train.shape      # 0.7 * 119040
```

```
(83328, 1)
```

```
x_test.shape       # 0.3 * 119040
```

```
(35712, 1)
```

```
y_test.shape       # 0.3 * 119040
```

```
(35712, 1)
```

```
df.shape[0]
```

```
119040
```

Importing our first Machine Learning Model

```
# Importing our first Machine Learning Model
from sklearn.linear_model import LinearRegression
```

Create an object of the Linear Regression Model

```
# Create an object of the Linear Regression Model
model = LinearRegression()
```

Train the model ⟶ .fit( )

```
# Train the model  : .fit()
model.fit(x_train, y_train)
```

```
LinearRegression()
```

Predict the model :

```
# Predict the model
y_pred = model.predict(x_test)
y_pred
```

```
array([[20.88972865],
       [28.56451499],
       [31.63442954],
       ...,
       [31.12277711],
       [28.56451499],
       [31.12277711]])
```

R2_Score (R Square) →

```
# R2_Score (R Square)    :    model.score(x_test, y_test)
r2 = model.score(x_test, y_test)
```

```
model.score(x_train, y_train)   # It's not R2_Score
```

```
0.773970250594516
```

Accuracy →

```
# Accuracy
r2 * 100
```

```
76.5801933965467
```

Another method to calculate R2_Score :

```
# Another method to calculate R2_Score
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
0.765801933965467
```

Adjusted R2_Score = 1 - [(1 - R^2) * (N - 1) / (N - p - 1)]

where,

R^2 = R2_Score or R Square

N = Total number of samples or Total number of rows

p = Total number of training or test size of independet features(variable)

```
# Adjusted R2_Score
adj_r2 = 1 - ((1-r2)*(len(df)-1)) / (len(df)-len(x_test)-1)
adj_r2
```

```
0.6654301296976397
```

Save as an Excel File ⟶

```
# Save as an excel file
y_pred = pd.DataFrame(y_pred)        # First change y_pred to a DataFrame
max_temp_pred = y_pred.to_csv('MaxTemp Prediction.csv')
```