Name: ***Aditya Kangune***

Roll number: ***33323***

Batch: ***K11***

Multiple Linear Regression

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Importing the dataset

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
dataset = pd.read_csv('/content/drive/MyDrive/Datasets/temperat
print(dataset)
```

```
     YEAR    JAN    FEB    MAR  ...  JAN-FEB  MAR-MAY  JUN-SEP  OCT-DEC
0    1901  22.40  24.14  29.07  ...    23.27    31.46    31.27    27.25
1    1902  24.93  26.58  29.77  ...    25.75    31.76    31.09    26.49
2    1903  23.44  25.03  27.83  ...    24.24    30.71    30.92    26.26
3    1904  22.50  24.73  28.21  ...    23.62    30.95    30.66    26.40
4    1905  22.00  22.83  26.68  ...    22.25    30.00    31.33    26.57
..    ...    ...    ...    ...  ...      ...      ...      ...      ...
112  2013  24.56  26.59  30.62  ...    25.58    32.58    31.33    27.83
113  2014  23.83  25.97  28.95  ...    24.90    31.82    32.00    27.81
114  2015  24.58  26.89  29.07  ...    25.74    31.68    31.87    28.27
115  2016  26.94  29.72  32.62  ...    28.33    34.57    32.28    30.03
116  2017  26.45  29.46  31.60  ...    27.95    34.13    32.41    29.69

[117 rows x 18 columns]
```

```
dataset.head()
```

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DE
0	1901	22.40	24.14	29.07	31.91	33.41	33.18	31.21	30.39	30.47	29.97	27.31	24.4
1	1902	24.93	26.58	29.77	31.78	33.73	32.91	30.92	30.73	29.80	29.12	26.31	24.0
2	1903	23.44	25.03	27.83	31.39	32.91	33.00	31.34	29.98	29.85	29.04	26.08	23.6
3	1904	22.50	24.73	28.21	32.02	32.64	32.07	30.36	30.09	30.04	29.20	26.36	23.6
4	1905	22.00	22.83	26.68	30.01	33.32	33.25	31.44	30.68	30.12	30.67	27.52	23.8

```
# X = dataset.iloc[:, :-1].values
# y = dataset.iloc[:, -1].values
```

```
dataset.dtypes
```

```
YEAR          int64
JAN         float64
FEB         float64
MAR         float64
APR         float64
MAY         float64
JUN         float64
JUL         float64
AUG         float64
SEP         float64
OCT         float64
NOV         float64
DEC         float64
ANNUAL      float64
JAN-FEB     float64
MAR-MAY     float64
JUN-SEP     float64
OCT-DEC     float64
dtype: object
```

```
dataset.columns
```

```
Index(['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP',
       'OCT', 'NOV', 'DEC', 'ANNUAL', 'JAN-FEB', 'MAR-MAY', 'JUN-SEP',
       'OCT-DEC'],
      dtype='object')
```

```
dataset.describe()
```

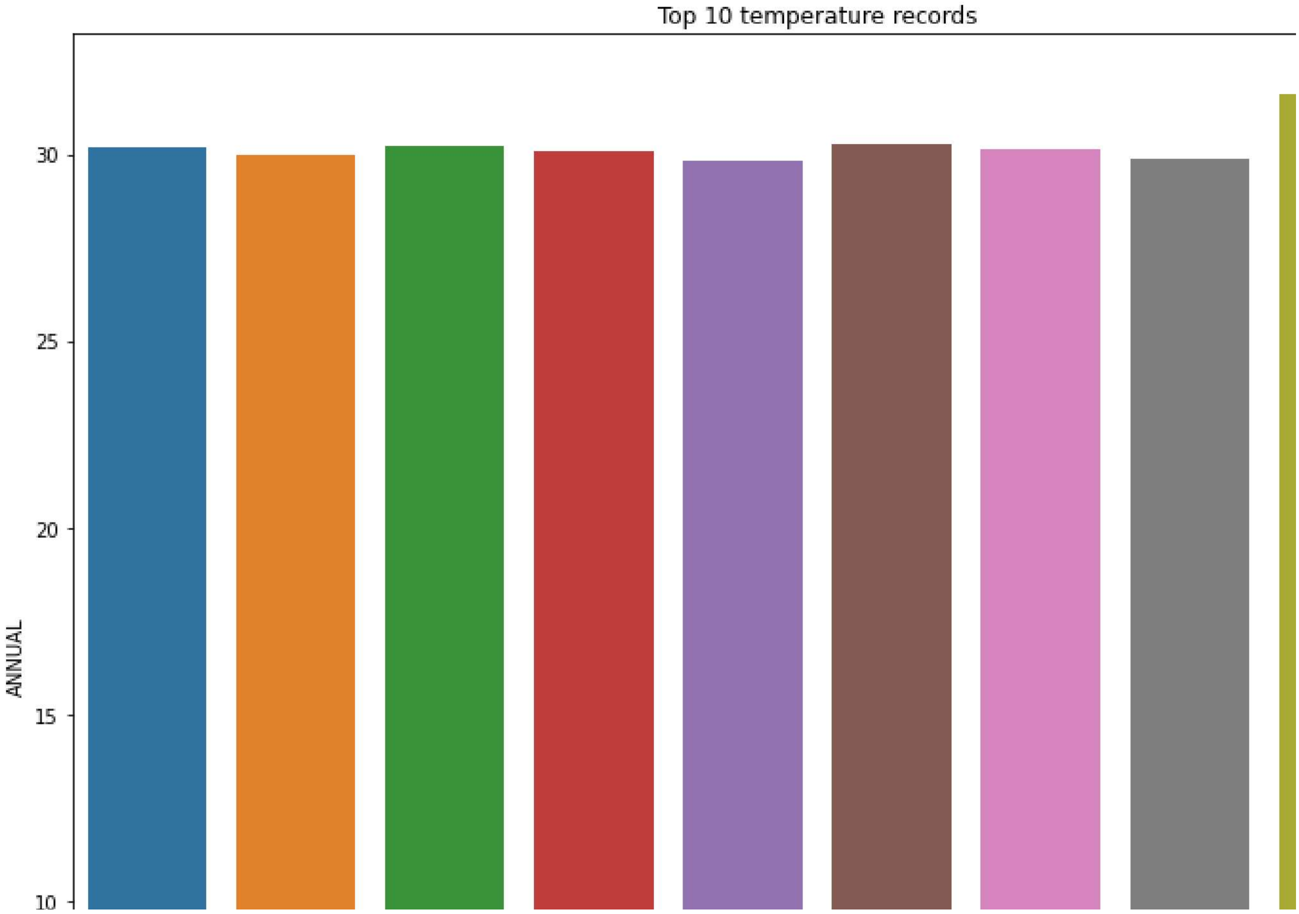
	YEAR	JAN	FEB	MAR	APR	MAY	J
count	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.0000
mean	1959.000000	23.687436	25.597863	29.085983	31.975812	33.565299	32.7742
std	33.919021	0.834588	1.150757	1.068451	0.889478	0.724905	0.6331
min	1901.000000	22.000000	22.830000	26.680000	30.010000	31.930000	31.1000
25%	1930.000000	23.100000	24.780000	28.370000	31.460000	33.110000	32.3400
50%	1959.000000	23.680000	25.480000	29.040000	31.950000	33.510000	32.7300
75%	1988.000000	24.180000	26.310000	29.610000	32.420000	34.030000	33.1800

```
dataset.isnull().sum()
```

```
YEAR       0
JAN        0
FEB        0
MAR        0
APR        0
MAY        0
JUN        0
JUL        0
AUG        0
SEP        0
OCT        0
NOV        0
DEC        0
ANNUAL     0
JAN-FEB    0
MAR-MAY    0
JUN-SEP    0
OCT-DEC    0
dtype: int64
```

```
top_10_data = dataset.nlargest(10, "ANNUAL")
plt.figure(figsize=(14,12))
plt.title("Top 10 temperature records")
sns.barplot(x=top_10_data.YEAR, y=top_10_data.ANNUAL)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2bf41ec410>
```



Taking in the required month for prediction

```
month_input = input("Enter month for prediction (Eg:'JAN'):")
```

```
Enter month for prediction (Eg:'JAN'):APR
```

```
X = dataset[month_input].values
y = dataset["ANNUAL"].values

# X = np.array(X, ndmin=2)
# y = np.array(y, ndmin=2)

print(X)
```

```
[31.91 31.78 31.39 32.02 30.01 31.93 31.79 32.42 30.79 31.42 31.27 31.29
 32.02 30.96 31.36 31.99 30.61 30.68 31.55 30.89 32.47 31.85 32.37 32.07
 32.53 30.42 31.5  31.7  31.59 30.98 32.65 32.08 30.53 31.95 30.24 31.52
 30.96 32.33 31.03 31.38 32.8  31.98 30.75 30.93 30.89 32.19 32.28 31.93
 31.85 31.16 30.3  32.22 32.1  32.47 31.12 32.04 30.91 32.51 32.33 31.9
 31.8  31.76 30.99 32.12 30.6  31.95 31.7  31.59 32.4  32.98 32.19 31.46
 33.3  32.68 32.36 31.71 31.7  32.23 33.11 33.36 32.26 31.65 31.1  32.19
 32.72 32.11 32.18 32.15 31.76 31.59 31.51 31.89 32.33 31.57 32.83 32.4
```

```
 31.   32.6  33.77 33.17 32.54 33.51 32.91 32.97 32.37 32.59 33.57 32.13
 33.09 34.07 31.7  32.46 32.66 32.74 31.87 35.38 34.95]
```

```
print(y)
```

```
[28.96 29.22 28.47 28.49 28.3  28.73 28.65 28.83 28.38 28.53 28.62 28.95
 28.67 28.66 28.94 28.82 28.11 28.66 28.66 28.76 28.86 28.8  28.74 28.8
 28.67 28.7  28.59 28.98 28.76 28.65 29.15 29.09 28.49 29.03 28.76 28.71
 28.7  28.7  28.85 28.88 29.46 28.98 28.8  28.89 28.97 29.37 28.84 28.73
 28.89 28.47 29.09 29.16 29.43 28.92 28.76 28.63 28.64 29.34 29.02 29.31
 28.72 28.89 29.04 29.09 29.16 29.41 29.14 29.07 29.61 29.47 29.15 29.31
 29.44 29.26 28.89 29.27 29.41 29.23 29.63 29.58 29.32 29.12 29.11 29.28
 29.61 29.33 29.72 29.55 29.18 29.14 29.32 29.23 29.55 29.46 30.18 29.58
 29.05 29.7  29.81 29.75 29.99 30.23 29.75 29.79 29.6  30.06 29.84 29.64
 30.3  30.13 29.82 29.81 29.81 29.72 29.9  31.63 31.42]
```

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
```

Training the Linear Regression model on the Training set

```
X_train= X_train.reshape(-1, 1)
y_train= y_train.reshape(-1, 1)
X_test = X_test.reshape(-1, 1)
```

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Predicting the Test set results

```
y_pred = regressor.predict(X_test)
# np.set_printoptions(precision=2)
# print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.r

# print(y_test.reshape())
```

```
print(y_test)
```

```
[28.62 29.31 29.58 29.23 28.83 29.72 28.59 29.81 28.74 30.18 30.23 28.47
 29.09 28.67 31.42 29.04 29.46 28.89 28.89 29.26 28.11 30.3  28.66 28.89]
```

```
print(y_pred)
```

```
[[28.85675571]
 [29.13776779]
 [29.36079324]
 [29.13330728]
 [29.36971426]
 [29.26266204]
 [28.95934742]
 [29.47676648]
 [29.34741171]
 [29.55259513]
 [29.85590975]
 [28.91028182]
 [28.42408633]
 [29.41877986]
 [30.49822306]
 [28.73186146]
 [28.99057099]
 [29.3429512 ]
 [29.07532066]
 [29.48568749]
 [28.56236211]
 [29.66856837]
 [28.71847993]
 [28.70509841]]
```

```
y_test = np.array(y_test)
y_pred = np.array(y_pred)
```

Mean Squared Error

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print(mse)
```

```
0.21199560831993922
```

Root Mean Squared Error

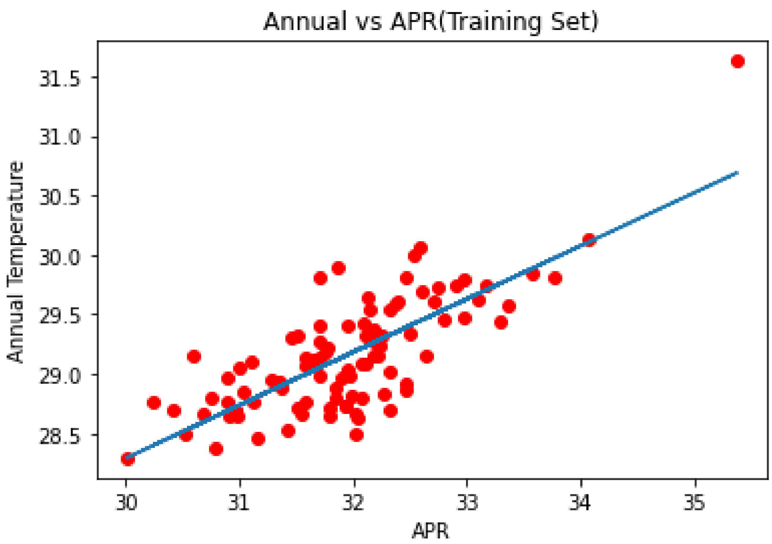
```
import math
rmse = math.sqrt(mse)
print(rmse)
```

```
0.46042980824436l2
```

Visualising the Training set results

```
plt.title("Annual vs {month}(Training Set)".format(month=month_
plt.ylabel("Annual Temperature")
plt.xlabel("{month}".format(month=month_input))
plt.scatter(X_train, y_train, color="red")
plt.plot(X_train, regressor.predict(X_train))
# show is used to display the graphic in the output
plt.show()

# Red dots are real values of salary (x_train, y_train)
# Blue line is the best fit line on training X_train and predic
```

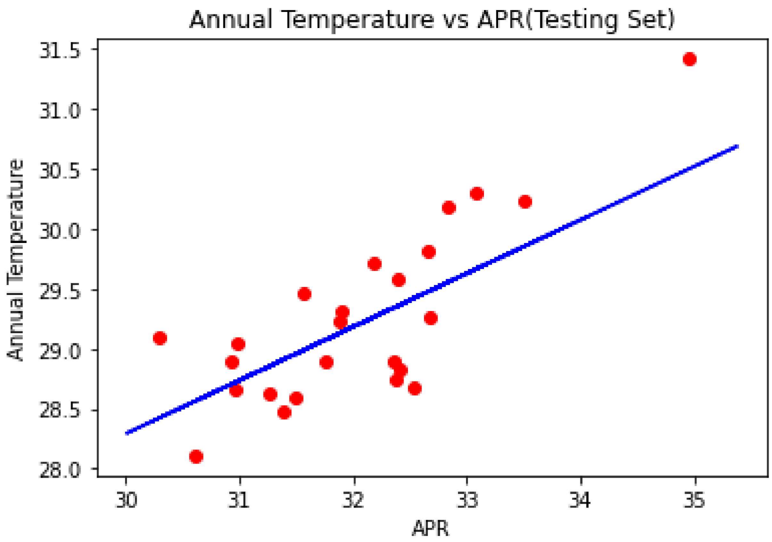


Visualising the Test set results

```
plt.title("Annual Temperature vs {month}(Testing Set)".format(m
plt.ylabel("Annual Temperature")
plt.xlabel("{month}".format(month=month_input))
plt.scatter(X_test, y_test, color="red")

# Predicted salries of test set will on the same rgression line
plt.plot(X_train, regressor.predict(X_train), color="blue")
plt.show()
```

```
# Red are new observations(test set)
# Blue is our best fit line after traning on the available trai
```




```
print(regressor.coef_)
print(regressor.intercept_)
```

```
[[0.44605091]]
[14.90874382]
```

Therefore, the equation of our simple linear regression model is:

$$\text{Annual\_temperature} = 0.44605091 \times \text{month } + 14.90874382.$$

Important Note: To get these coefficients we called the "coef_" and "intercept_" attributes from our regressor object. Attributes in Python are different than methods and usually return a simple value or an array of values.