Importing Pandas and Numpy library-

In [1]:
```
import numpy as np
import pandas as pd
```

1. Read csv file-

In [2]:
```
file=pd.read_csv("D:\\Downloads\\Heart.csv")
```

2. Read first and last 5 rows-

In [6]:
```
file.head(5)
```

Out[6]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No

In [7]:
```
file.tail(5)
```

Out[7]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.0	reversable	Yes
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversable	Yes
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.0	reversable	Yes
301	302	57	0	nontypical	130	236	0	2	174	0	0.0	2	1.0	normal	Yes
302	303	38	1	nonanginal	138	175	0	0	173	0	0.0	1	NaN	normal	No

3. Summary of the file

In [8]:
```
file.describe()
```

Out[8]:

	Unnamed: 0	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	299.000000
mean	152.000000	54.438944	0.679868	131.689769	246.693069	0.148515	0.990099	149.607261	0.326733	1.039604	1.600660	0.672241
std	87.612784	9.038662	0.467299	17.599748	51.776918	0.356198	0.994971	22.875003	0.469794	1.161075	0.616226	0.937438
min	1.000000	29.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000
25%	76.500000	48.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	152.000000	56.000000	1.000000	130.000000	241.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000
75%	227.500000	61.000000	1.000000	140.000000	275.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	303.000000	77.000000	1.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000

4. Shape of file-

In [9]:
```
file.shape
```

Out[9]: (303, 15)

5. Name of all attributes-

```
In [14]: file.columns
```

```
Out[14]: Index(['Unnamed: 0', 'Age', 'Sex', 'ChestPain', 'RestBP', 'Chol', 'Fbs',
                'RestECG', 'MaxHR', 'ExAng', 'Oldpeak', 'Slope', 'Ca', 'Thal', 'AHD'],
               dtype='object')
```

6. Datatypes of each attribute-

```
In [12]: file.dtypes
```

```
Out[12]: Unnamed: 0      int64
         Age             int64
         Sex             int64
         ChestPain      object
         RestBP          int64
         Chol            int64
         Fbs             int64
         RestECG         int64
         MaxHR           int64
         ExAng           int64
         Oldpeak       float64
         Slope           int64
         Ca            float64
         Thal           object
         AHD            object
         dtype: object
```

7. Rename a column (MaxHR -> MaxHRNew)-

```
In [19]: file.rename(columns={'MaxHR':'MaxHRNew'})
```

Out[19]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHRNew	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.0	reversable	Yes
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversable	Yes
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.0	reversable	Yes
301	302	57	0	nontypical	130	236	0	2	174	0	0.0	2	1.0	normal	Yes
302	303	38	1	nonanginal	138	175	0	0	173	0	0.0	1	NaN	normal	No

8. Show missing values in dataset-

```
In [22]: file.isnull().sum()

Out[22]: Unnamed: 0    0
         Age           0
         Sex           0
         ChestPain     0
         RestBP        0
         Chol          0
         Fbs           0
         RestECG       0
         MaxHR         0
         ExAng         0
         Oldpeak       0
         Slope         0
         Ca            4
         Thal          2
         AHD           0
         dtype: int64
```

9. Mean of age of patients

```
In [24]: file['Age'].mean()

Out[24]: 54.43894389438944
```

10. Max and Min of age of patients

```
In [25]: file['Age'].max()

Out[25]: 77

In [26]: file['Age'].min()

Out[26]: 29
```

11. How many 0's is there in file?

```
In [28]: file.count(0).sum()

Out[28]: 4539
```

12. Replace Yes by 1 and No by 0 in AHD columns-

```
In [32]: file['AHD'].replace({'Yes':1,'No':0})

Out[32]: 0      0
         1      1
         2      1
         3      0
         4      0
               ..
         298    1
         299    1
         300    1
         301    1
         302    0
         Name: AHD, Length: 303, dtype: int64
```

CREATING 'TEMPERATURE VARIATION' MODEL FOR 'JAN' MONTH

1. Importing Libraries and Reading .csv file

In [1]:
```python
import numpy as np
import pandas as pd
```

In [2]:
```python
file=pd.read_csv("C:\\Users\\hp\\Downloads\\temperatures.csv")
```

2. Reading Dataset

In [3]:
```python
file
```

Out[3]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	JAN-FEB	MAR-MAY	JUN-SEP	OCT-DEC
0	1901	22.40	24.14	29.07	31.91	33.41	33.18	31.21	30.39	30.47	29.97	27.31	24.49	28.96	23.27	31.46	31.27	27.25
1	1902	24.93	26.58	29.77	31.78	33.73	32.91	30.92	30.73	29.80	29.12	26.31	24.04	29.22	25.75	31.76	31.09	26.49
2	1903	23.44	25.03	27.83	31.39	32.91	33.00	31.34	29.98	29.85	29.04	26.08	23.65	28.47	24.24	30.71	30.92	26.26
3	1904	22.50	24.73	28.21	32.02	32.64	32.07	30.36	30.09	30.04	29.20	26.36	23.63	28.49	23.62	30.95	30.66	26.40
4	1905	22.00	22.83	26.68	30.01	33.32	33.25	31.44	30.68	30.12	30.67	27.52	23.82	28.30	22.25	30.00	31.33	26.57
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
112	2013	24.56	26.59	30.62	32.66	34.46	32.44	31.07	30.76	31.04	30.27	27.83	25.37	29.81	25.58	32.58	31.33	27.83
113	2014	23.83	25.97	28.95	32.74	33.77	34.15	31.85	31.32	30.68	30.29	28.05	25.08	29.72	24.90	31.82	32.00	27.81
114	2015	24.58	26.89	29.07	31.87	34.09	32.48	31.88	31.52	31.55	31.04	28.10	25.67	29.90	25.74	31.68	31.87	28.27
115	2016	26.94	29.72	32.62	35.38	35.72	34.03	31.64	31.79	31.66	31.98	30.11	28.01	31.63	28.33	34.57	32.28	30.03
116	2017	26.45	29.46	31.60	34.95	35.84	33.82	31.88	31.72	32.22	32.29	29.60	27.18	31.42	27.95	34.13	32.41	29.69

117 rows × 18 columns

3. Assigning variables (Dependent & Independent)

In [4]:
```python
x=file[['YEAR']]
y=file[['JAN']]
```

In [5]:
```python
x
```

Out[5]:

	YEAR
0	1901
1	1902
2	1903
3	1904
4	1905
...	...
112	2013
113	2014
114	2015
115	2016
116	2017

117 rows × 1 columns

In [6]:
```python
y
```

Out[6]:

	JAN
0	22.40
1	24.93
2	23.44
3	22.50
4	22.00
...	...
112	24.56
113	23.83
114	24.58
115	26.94
116	26.45

117 rows × 1 columns

4. **Splitting the data into Training and Testing data**

```
In [7]: from sklearn.model_selection import train_test_split
```

```
In [8]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

5. **Printing Size of Training & Testing**

```
In [9]: print(len(x_train))
93
```

```
In [10]: print(len(x_test))
24
```

6. **Importing & Initiating Linear Regression Model**

```
In [11]: from sklearn import linear_model, metrics
```

```
In [12]: reg=linear_model.LinearRegression()
```

7. **Fitting the Model and Displaying Regression Coefficients**

```
In [13]: model=reg.fit(x_train,y_train)
```

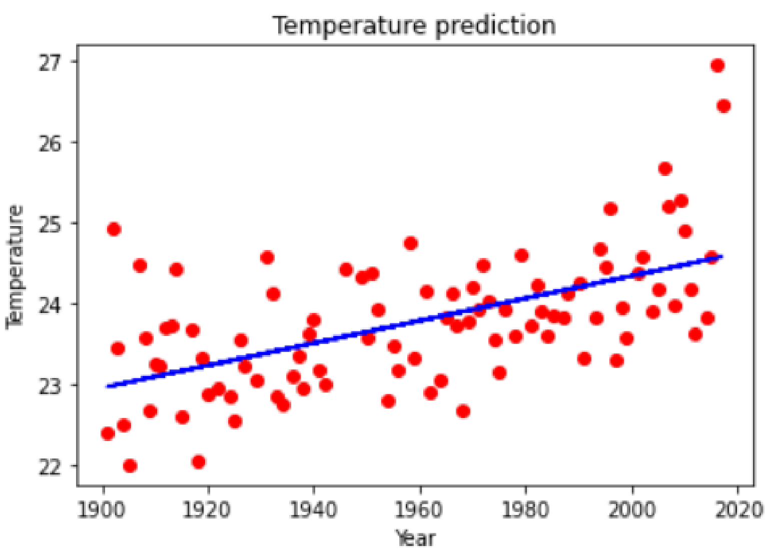
```
In [14]: print(model.intercept_,model.coef_)
[-3.34424105] [[0.01383922]]
```

8. **Creating the Model**

```
In [15]: import matplotlib.pyplot as plt
         import seaborn as sb
```

```
In [16]: plt.title("Temperature prediction")
         plt.xlabel("Year")
         plt.ylabel("Temperature")
         plt.scatter(x_train, y_train, color='red')
         plt.plot(x_train, reg.predict(x_train), color='blue')
```

9. **Plotting the Model**



10. Importing metrics library and creating predict variable

```
In [17]: import sklearn.metrics as metrics
```

```
In [18]: y_predict=reg.predict(x_test)
```

11.Calculating the Performance metrics MAE, MSE, RMSE, R-Squared, Adjusted R-Squared

```
In [23]: mse=metrics.mean_squared_error(y_test, y_predict)
         mae=metrics.mean_absolute_error(y_test, y_predict)
         rmse=np.sqrt(mse)
         r2=metrics.r2_score(y_test, y_predict)
         n=len(x_test)
         adjusted_r2=((1-r2)*(n-1)/(n-1-1))
         print("MAE:",mae)
         print("MSE:", mse)
         print("RMSE:", rmse)
         print("R-Squared:", r2)
         print("Adjusted R-Squared:", adjusted_r2)

         MAE: 0.5311430565253427
         MSE: 0.4321235802333869
         RMSE: 0.6573610729525949
         R-Squared: 0.083148146062207
         Adjusted R-Squared: 0.9585269382076926
```

CREATING ‘TEMPERATURE VARIATION’ MODEL FOR ‘FEB’ MONTH

1. Importing & Initiating Linear Regression Model

```
In [11]: from sklearn import linear_model, metrics
```

```
In [12]: reg=linear_model.LinearRegression()
```

2. Fitting the Model and Displaying Regression Coefficients

```
In [13]: model=reg.fit(x_train,y_train)
```

```
In [14]: print(model.intercept_,model.coef_)

         [-20.90316138] [[0.02374755]]
```

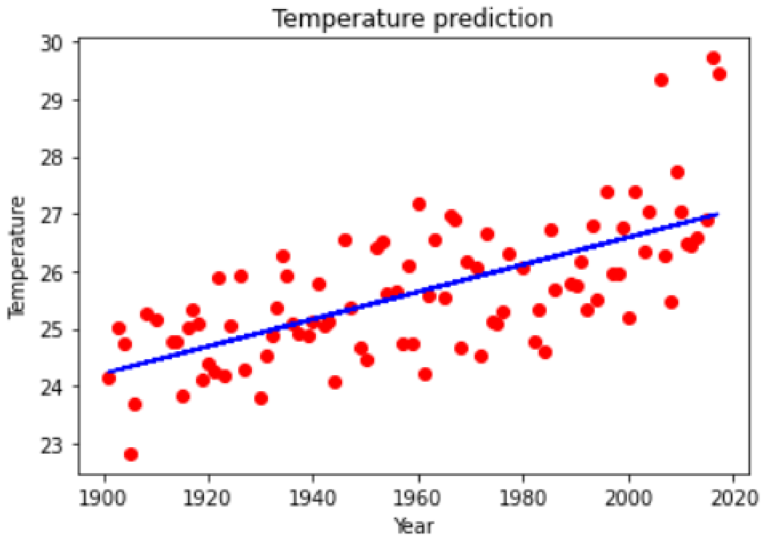
3. Creating the Model

```
In [15]: import matplotlib.pyplot as plt
         import seaborn as sb
```

```
In [16]: plt.title("Temperature prediction")
         plt.xlabel("Year")
         plt.ylabel("Temperature")
         plt.scatter(x_train, y_train, color='red')
         plt.plot(x_train, reg.predict(x_train), color='blue')
```

4. Plotting the Model

```
Out[16]: [<matplotlib.lines.Line2D at 0x19b1c9e7370>]
```



5. Importing metrics library and creating predict variable

```
In [17]: import sklearn.metrics as metrics
```

```
In [18]: y_predict=reg.predict(x_test)
```

6. Calculating the Performance metrics MAE, MSE, RMSE, R-Squared, Adjusted R-Squared

```
In [19]: mse=metrics.mean_squared_error(y_test, y_predict)
         mae=metrics.mean_absolute_error(y_test, y_predict)
         rmse=np.sqrt(mse)
         r2=metrics.r2_score(y_test, y_predict)
         n=len(x_test)
         adjusted_r2=((1-r2)*(n-1)/(n-1-1))
         print("MAE:",mae)
         print("MSE:", mse)
         print("RMSE:", rmse)
         print("R-Squared:", r2)
         print("Adjusted R-Squared:", adjusted_r2)
```

```
MAE: 0.6204868628773595
MSE: 0.6959224488139789
RMSE: 0.8342196646051799
R-Squared: 0.17740203448548264
Adjusted R-Squared: 0.8599887821288136
```

CREATING 'TEMPERATURE VARIATION' MODEL FOR 'MAR' MONTH

1. Importing & Initiating Linear Regression Model

```
In [11]: from sklearn import linear_model, metrics
```

```
In [12]: reg=linear_model.LinearRegression()
```

2. Fitting the Model and Displaying Regression Coefficients

```
In [13]: model=reg.fit(x_train,y_train)
```

```
In [14]: print(model.intercept_,model.coef_)

         [-9.6210023] [[0.01975092]]
```

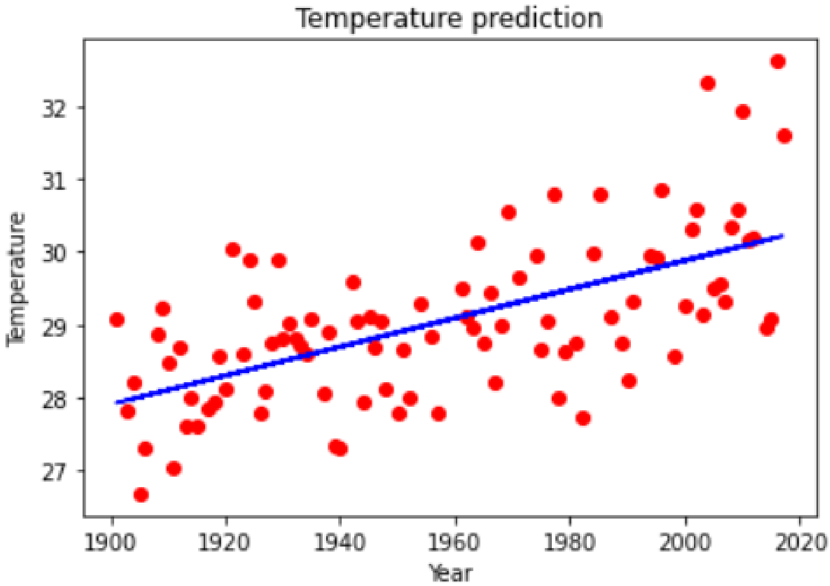
3. Creating the Model

```
In [15]: import matplotlib.pyplot as plt
         import seaborn as sb
```

```
In [16]: plt.title("Temperature prediction")
         plt.xlabel("Year")
         plt.ylabel("Temperature")
         plt.scatter(x_train, y_train, color='red')
         plt.plot(x_train, reg.predict(x_train), color='blue')
```

4. Plotting the Model

```
Out[16]: [<matplotlib.lines.Line2D at 0x2697124b370>]
```



5. Importing metrics library and creating predict variable

```
In [17]: import sklearn.metrics as metrics
```

```
In [18]: y_predict=reg.predict(x_test)
```

6. **Calculating the Performance metrics MAE, MSE, RMSE, R-Squared, Adjusted R-Squared**

In [19]:
```
mse=metrics.mean_squared_error(y_test, y_predict)
mae=metrics.mean_absolute_error(y_test, y_predict)
rmse=np.sqrt(mse)
r2=metrics.r2_score(y_test, y_predict)
n=len(x_test)
adjusted_r2=((1-r2)*(n-1)/(n-1-1))
print("MAE:",mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R-Squared:", r2)
print("Adjusted R-Squared:", adjusted_r2)
```

```
MAE: 0.7664045190432921
MSE: 0.8709650898493626
RMSE: 0.9332551043789488
R-Squared: -0.2574417882078337
Adjusted R-Squared: 1.3145982331263715
```

CREATING 'TEMPERATURE VARIATION' MODEL FOR 'APR MONTH

1. **Importing & Initiating Linear Regression Model**

In [11]:
```
from sklearn import linear_model, metrics
```

In [12]:
```
reg=linear_model.LinearRegression()
```

2. **Fitting the Model and Displaying Regression Coefficients**

In [13]:
```
model=reg.fit(x_train,y_train)
```

In [14]:
```
print(model.intercept_,model.coef_)
```

```
[5.66465629] [[0.01342036]]
```

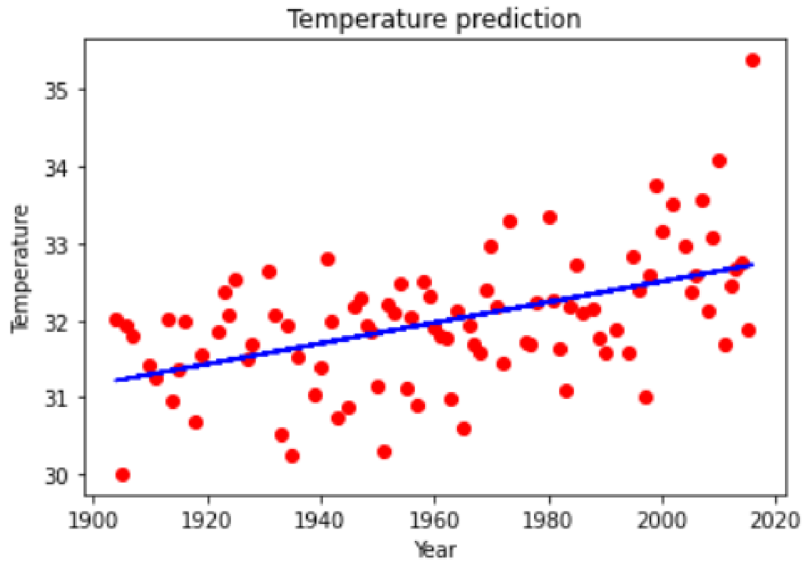
3. **Creating the Model**

In [15]:
```
import matplotlib.pyplot as plt
import seaborn as sb
```

In [16]:
```
plt.title("Temperature prediction")
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.scatter(x_train, y_train, color='red')
plt.plot(x_train, reg.predict(x_train), color='blue')
```

4. **Plotting the Model**

Out[16]: [<matplotlib.lines.Line2D at 0x24a64ddada0>]



5. **Importing metrics library and creating predict variable**

In [17]:
```
import sklearn.metrics as metrics
```

In [18]:
```
y_predict=reg.predict(x_test)
```

6. **Calculating the Performance metrics MAE, MSE, RMSE, R-Squared, Adjusted R-Squared**

In [19]:
```
mse=metrics.mean_squared_error(y_test, y_predict)
mae=metrics.mean_absolute_error(y_test, y_predict)
rmse=np.sqrt(mse)
r2=metrics.r2_score(y_test, y_predict)
n=len(x_test)
adjusted_r2=((1-r2)*(n-1)/(n-1-1))
print("MAE:",mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R-Squared:", r2)
print("Adjusted R-Squared:", adjusted_r2)
```

```
MAE: 0.6141904741678816
MSE: 0.6047906867346705
RMSE: 0.777682896002394
R-Squared: 0.388308992678045
Adjusted R-Squared: 0.6394951440184076
```

CREATING 'TEMPERATURE VARIATION' MODEL FOR 'MAY' MONTH

1. **Importing & Initiating Linear Regression Model**

In [11]:
```
from sklearn import linear_model, metrics
```

In [12]:
```
reg=linear_model.LinearRegression()
```

2. Fitting the Model and Displaying Regression Coefficients

```
In [13]: model=reg.fit(x_train,y_train)
```

```
In [14]: print(model.intercept_,model.coef_)
```

```
[14.36622931] [[0.00979527]]
```

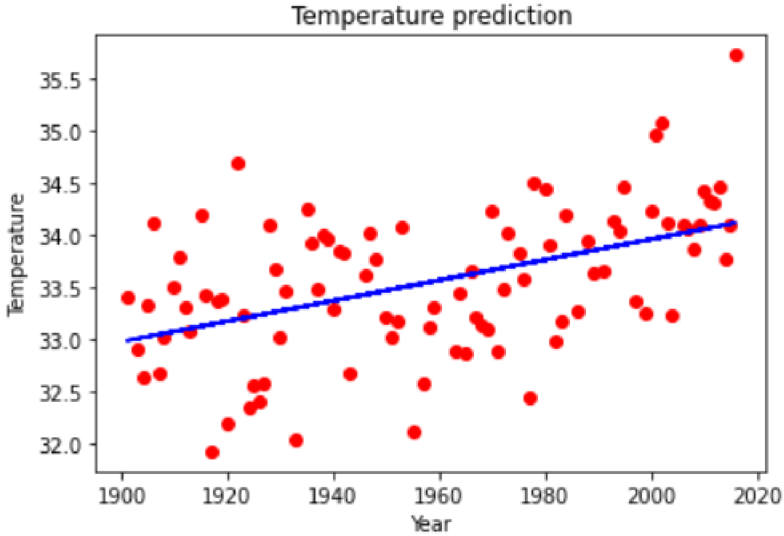
3. Creating the Model

```
In [15]: import matplotlib.pyplot as plt
         import seaborn as sb
```

```
In [16]: plt.title("Temperature prediction")
         plt.xlabel("Year")
         plt.ylabel("Temperature")
         plt.scatter(x_train, y_train, color='red')
         plt.plot(x_train, reg.predict(x_train), color='blue')
```

4. Plotting the Model

```
Out[16]: [<matplotlib.lines.Line2D at 0x154708c7370>]
```



5. Importing metrics library and creating predict variable

```
In [17]: import sklearn.metrics as metrics
```

```
In [18]: y_predict=reg.predict(x_test)
```

6. Calculating the Performance metrics MAE, MSE, RMSE, R-Squared, Adjusted R-Squared

In [19]:
```python
mse=metrics.mean_squared_error(y_test, y_predict)
mae=metrics.mean_absolute_error(y_test, y_predict)
rmse=np.sqrt(mse)
r2=metrics.r2_score(y_test, y_predict)
n=len(x_test)
adjusted_r2=((1-r2)*(n-1)/(n-1-1))
print("MAE:",mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R-Squared:", r2)
print("Adjusted R-Squared:", adjusted_r2)
```

```
MAE: 0.5782658022605768
MSE: 0.6713851636923708
RMSE: 0.8193809637112464
R-Squared: -0.04407723246399753
Adjusted R-Squared: 1.0915352884850884
```