Machine Learning Assignment 1

Linear Regression

Analysis

- We used Solar Radiation dataset which is to predict solar radiation. In this assignment Radiation is our Y variable which is to be predicted. Our dataset have Temperature, Pressure and humidity since Pressure is constant so we used only Temperature and Humidity as our X Variable
- We used sklearn library for linear regression, splitting data and to find mean square error. We used pandas to read data from csv file and numpy to convert it into numpy array
- Our mean square error from linear.fit is 19984.37105046212 and from Gradient Descent it is 19544.575972453513 and their diffrence is very low which is 439.79507800860665
- For linear regression with multivariable we used both temperature and humidity as our X variable and our mean square error is 18735.895356990874 which is lower than with one variable and indicates that the prediction of radiation becomes better by increasing variables

In [1]:

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import (StandardScaler, PolynomialFeatures)
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
```

Reading Data

In [2]:

```
filepath ="C:/Users/k-ali/Desktop/FYP/fyp/ML sem/s2.csv"
data = pd.read_csv(filepath)
data.head()
```

Out[2]:

	UNIXTime	Data	Time	Radiation	t1	Temperature	Pressure	Humidity	Wind
0	1475229326	9/29/2016 12:00:00 AM	13:12:00	1.21	23.55	48	30.46	59	
1	1475229023	9/29/2016 12:00:00 AM	23:50:23	1.21	23.50	48	30.46	58	
2	1475228726	9/29/2016 12:00:00 AM	23:45:26	1.23	23.45	48	30.46	57	
3	1475228421	9/29/2016 12:00:00 AM	23:40:21	1.21	23.40	48	30.46	60	
4	1475228124	9/29/2016 12:00:00 AM	23:35:24	1.17	23.35	48	30.46	62	

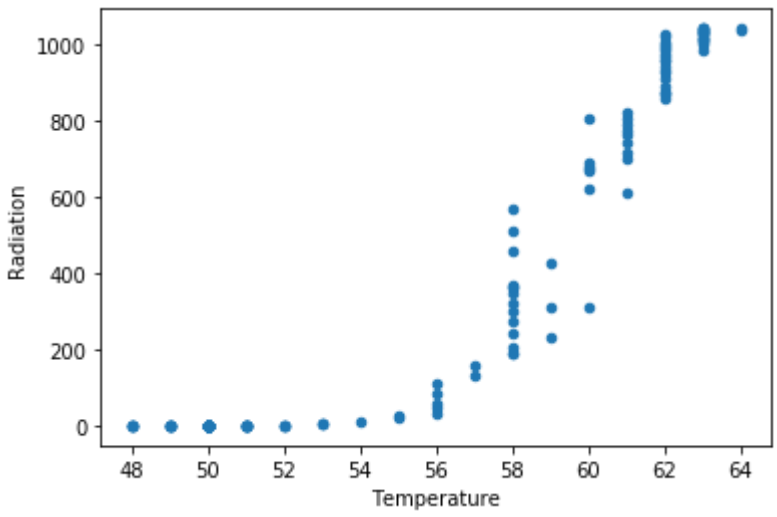
Scatter Plot

In [3]:

```
data.plot(kind='scatter', y='Radiation', x='Temperature')
```

Out[3]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2548116ff48>
```



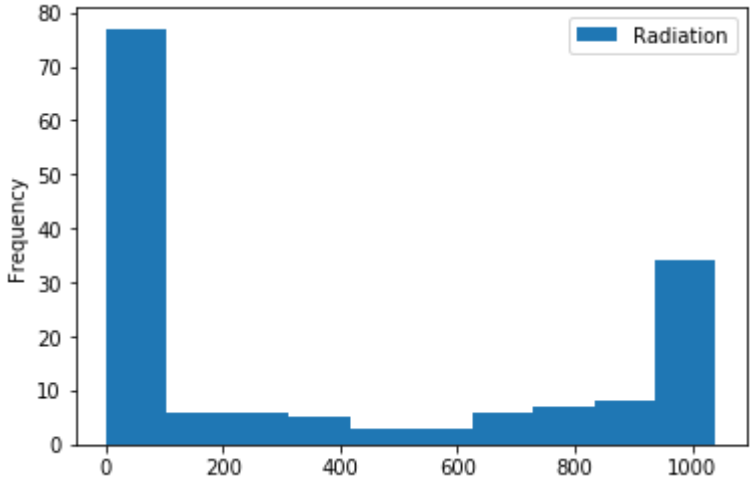
Histogram

In [4]:

```
data.plot(kind='hist', y='Radiation', x='Temperature')
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x254818d6fc8>
```



In [5]:

```
data.shape
```

Out[5]:

```
(155, 12)
```

In [6]:

```
Y = pd.DataFrame(data['Radiation'])
Y.head(2)
```

Out[6]:

	Radiation
0	1.21
1	1.21

In [7]:

```
X = pd.DataFrame(data['Temperature'])
X.head(2)
```

Out[7]:

	Temperature
0	48
1	48

Splitting Data into Test and Training Set

In [8]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=
1)
```

In [9]:

```
X_train.head(3)
```

Out[9]:

	Temperature
104	61
135	63
77	56

Linear Regression with One Variable

In [10]:

```
linear = LinearRegression()
linear.fit(X_train,Y_train)
```

Out[10]:

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

In [11]:

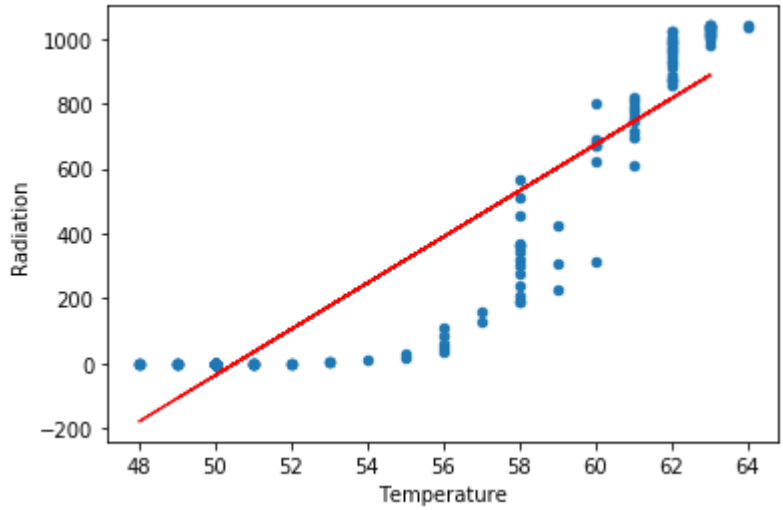
```
Y_predicted = linear.predict(X_test)
pd.DataFrame(Y_predicted).head()
```

Out[11]:

	0
0	816.124295
1	389.578437
2	-36.967420
3	-36.967420
4	-36.967420

In [12]:

```
data.plot(kind='scatter', x='Temperature', y='Radiation')
plt.plot(X_test, Y_predicted, color='red', linewidth=1)
plt.show()
```



In [13]:

```
linear.coef_
```

Out[13]:

```
array([[71.09097626]])
```

In [14]:

```
linear.intercept_
```

Out[14]:

```
array([-3591.51623317])
```

Mean Square Error

In [15]:

```
mse = mean_squared_error(Y_test,Y_predicted)
mse
```

Out[15]:

```
19984.37105046212
```

In [16]:

```
X_test.head()
```

Out[16]:

	Temperature
117	62
75	56
51	50
31	50
35	50

Gradient Descent

In [17]:

```
X_test_np = np.array(X_test,dtype = 'float64')
Y_test_np = np.array(Y_test,dtype = 'float64')



t0 = linear.intercept_
t1 = linear.coef_
L = 0.000001
iterations = 1000
cost_s = []
iteration=[]
n = 1
m = len(X_test_np)
for i in range(iterations):
    y_predicted = t1 * X_test_np + t0

    t1_n = (-2/m) * sum(X_test_np * (Y_test_np - y_predicted))
    t0_n = (-2/m) * sum(Y_test_np - y_predicted)

    cost = (1/m) * sum(val**2 for val in (Y_test_np - y_predicted))

    cost_s.append(cost)
    iteration.append(n)
    t1 = t1 - L * t1_n
    t0 = t0 - L * t0_n
    n=n+1



print(t1,t0,cost)
```

```
[[71.46272533]] [-3591.51539474] [19544.57602201]
```

In [18]:

```
y_predicted = t1 * X_test_np + t0
pd.DataFrame(y_predicted).head()
```

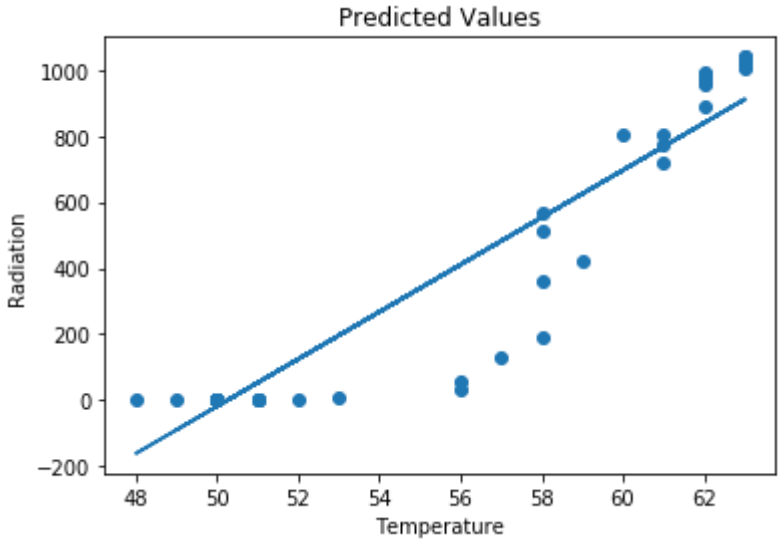
Out[18]:

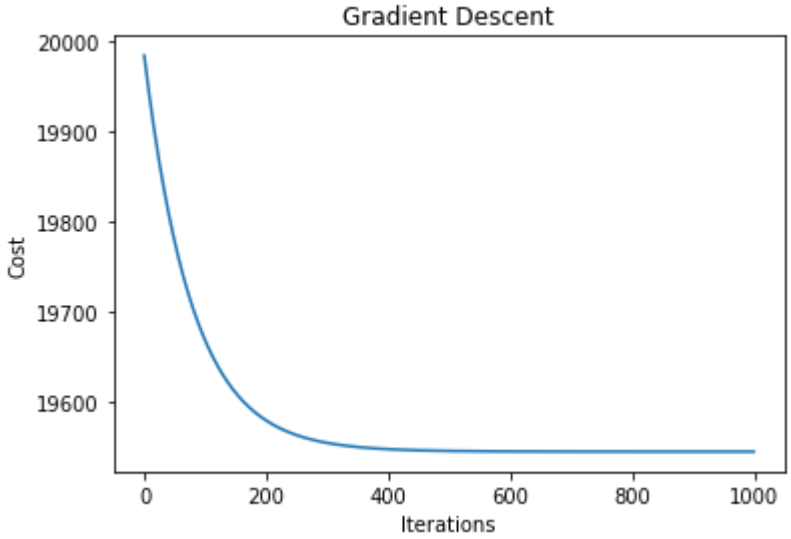
	0
0	839.173576
1	410.397224
2	-18.379128
3	-18.379128
4	-18.379128

In [19]:

```
y_predicted = t1 * X_test_np + t0
plt.scatter(X_test,Y_test)
plt.plot(X_test,y_predicted)
plt.xlabel('Temperature')
plt.ylabel('Radiation')
plt.title('Predicted Values')
plt.show()

plt.plot(cost_s)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Gradient Descent')
plt.show()
```





Difference

In [20]:

```
mse2 = mean_squared_error(Y_test,y_predicted)
mse2
```

Out[20]:

```
19544.575972453513
```

In [21]:

```
Difference = mse - mse2
Difference
```

Out[21]:

```
439.79507800860665
```

Linear Regression with Multiple Variable

In [22]:

```
x1 = pd.DataFrame(data['Temperature'])
x2 = pd.DataFrame(data['Humidity'])
```

In [23]:

```
multi_x = pd.concat([x1,x2], axis=1)
multi_x.head(3)
```

Out[23]:

	Temperature	Humidity
0	48	59
1	48	58
2	48	57

In [24]:

```
X_train, X_test, Y_train, Y_test = train_test_split(multi_x, Y, test_size=0.30, random_
state=1)
```

In [25]:

```
X_train.head(3)
```

Out[25]:

	Temperature	Humidity
104	61	53
135	63	43
77	56	49

In [26]:

```
linear = LinearRegression()
linear.fit(X_train,Y_train)
```

Out[26]:

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

In [27]:

```
Y_predicted = linear.predict(X_test)
```

In [28]:

```
pd.DataFrame(Y_predicted).head()
```

Out[28]:

	0
0	825.541945
1	366.386932
2	-97.288922
3	4.429993
4	2.169573

In [29]:

```
mse3 = mean_squared_error(Y_test,Y_predicted)
mse3
```

Out[29]:

```
18735.895356990874
```

In [30]:

```
X1 = pd.DataFrame(X_test['Temperature'])
```

In [31]:

```
X2 = pd.DataFrame(X_test['Humidity'])
X2.head()
```

Out[31]:

	Humidity
117	50
75	50
51	48
31	93
35	92

Gradient Descent

In [32]:

```
X1_test_np = np.array(X1,dtype = 'float64')
X2_test_np = np.array(X2,dtype = 'float64')
Y1_test_np = np.array(Y_test,dtype = 'float64')


t0 = linear.intercept_
t1 = 0
t2 = 0
L = 0.00001
iterations = 1000
cost_s = []
iteration=[]
n = 1
m = len(X1_test_np)
for i in range(iterations):
    y_predicted =  t0 + (t2 * X2_test_np) + (t1 * X1_test_np)

    t2_n = (-2/m) * sum(X2_test_np * (Y1_test_np - y_predicted))
    t1_n = (-2/m) * sum(X1_test_np * (Y1_test_np - y_predicted))
    t0_n = (-2/m) * sum(Y1_test_np - y_predicted)

    cost = (1/m) * sum(val**2 for val in (Y1_test_np - y_predicted))

    cost_s.append(cost)
    iteration.append(n)

    t2 = t1 - L * t2_n
    t1 = t1 - L * t1_n
    t0 = t0 - L * t0_n

    n=n+1


print(t2,t1,t0,cost)
```

```
[38.51090698] [38.95989346] [-4032.89965225] [852492.87490417]
```

In [33]:

```
y_predicted = (t2 * X2_test_np) + (t1 * X1_test_np) + t0
pd.DataFrame(y_predicted).head()
```

Out[33]:

	0
0	308.159091
1	74.399731
2	-236.381444
3	1496.609370
4	1458.098463

In [34]:

```
mse4 = mean_squared_error(Y_test,y_predicted)
mse4
```

Out[34]:

852493.2309950344

END