Machine Learning Fundamentals Lab-2

Name: Gaurav Prasanna
Reg No: 19BEC1315

Aim:

a) To show Logistic Regression without using sci-kit learn and using the inbuilt formula and using matplotlib to visualize the regression lines.
b) Using sci-kit learn library of logistic Regression for array data using numpy.
c) Using sci-kit learn to use the Logistic Regression library and the using the metrics from scikit learn to evaluate and then plotting to visualize.(Using 50-startups dataset)

Software Required:

1) Ananconda Navigator
2) Jupyter Notebook

Libraries Required: Numpy, Matplotlib, Scik-kit learn, Pandas

Code and Outputs:

a) Logistic Regression Mathematically:

```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import mean_squared_error, r2_score
```

```
In [4]: x = np.array([1,2,3,4,5])
        y = np.array([7,14,15,18,19])
        n = np.size(x)
        print(n)

        5
```

```
In [5]: x_mean = np.mean(x)
        y_mean = np.mean(y)
        print(x_mean, y_mean)

        3.0 14.6
```

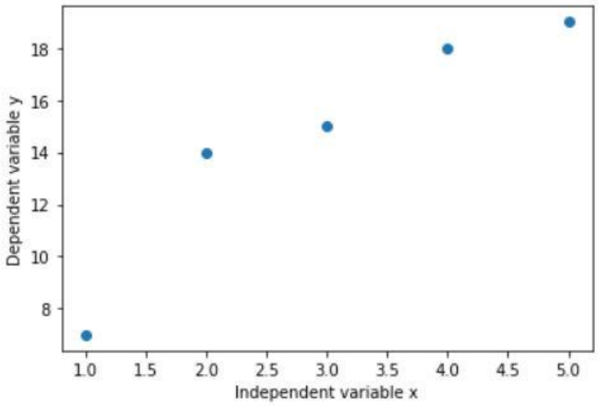
```
In [6]: Sxy = np.sum(x*y) - n*x_mean*y_mean
        Sxx = np.sum(x*x) - n*x_mean*x_mean

        print(Sxy, Sxx)

        28.0 10.0
```

```
In [10]: b1 = Sxy/Sxx
         b0 = y_mean-b1*x_mean

         print('Slope is:', b1)
         print('Intercept is:', b0)

         Slope is: 2.8
         Intercept is: 6.200000000000001
```

In [8]:
```
%matplotlib inline
```

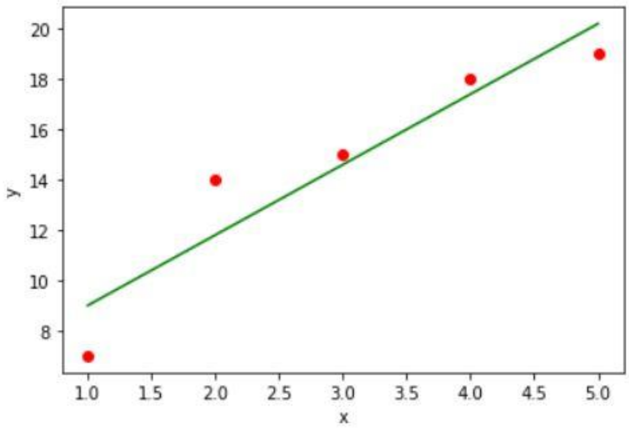
In [9]:
```
plt.scatter(x,y)
plt.xlabel('Independent variable x')
plt.ylabel('Dependent variable y')
```

Out[9]: Text(0, 0.5, 'Dependent variable y')



In [11]:
```
y_pred = b1 * x + b0

plt.scatter(x, y, color = 'red')
plt.plot(x, y_pred, color = 'green')
plt.xlabel('x')
plt.ylabel('y')
```

Out[11]: Text(0, 0.5, 'y')



In [12]:
```
error = y - y_pred
se = np.sum(error**2)
print('squared error is:', se)
```

```
squared error is: 10.800000000000004
```

In [13]:
```python
mse = se/n
print('mean squared error is:', mse)
```

```
mean squared error is: 2.160000000000001
```

In [14]:
```python
rmse = np.sqrt(mse)
print('root mean squared error is:', rmse)
```

```
root mean squared error is: 1.4696938456699071
```

In [15]:
```python
sst = np.sum((y - y_mean)** 2)
r2 = 1 - (se/sst)
print('R square is:', r2)
```

```
R square is: 0.8789237668161435
```

In [16]:
```python
x = x.reshape(-1, 1)
regression_model = LinearRegression()
```

In [18]:
```python
regression_model.fit(x, y)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```python
y_predict = regression_model.predict(x)
```

In [20]:
```python
mse_fun = mean_squared_error(y, y_predict)
print('MSE using sci-kit learn is:', mse_fun)
```

```
MSE using sci-kit learn is: 2.160000000000001
```

In [21]:
```python
rmse_fun = np.sqrt(mean_squared_error(y, y_predict))
print('RMSE using sci-kit learn is:', rmse_fun)
```

```
RMSE using sci-kit learn is: 1.4696938456699071
```

In [22]:
```python
r2_fun = r2_score(y, y_predict)
print('R2 score using sci-kit learn is:', r2_fun)
```

```
R2 score using sci-kit learn is: 0.8789237668161435
```

In [23]:
```python
print('Slope is:', regression_model.coef_)
print('Intercept is:', regression_model.intercept_)
```

```
Slope is: [2.8]
Intercept is: 6.1999999999999975
```

---

b) Logistic Regression for Numpy array fitting and Predicting:

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics  import mean_squared_error, r2_score
```

```
In [2]: x = np.array([1,2,3,4,5])
        y = np.array([7,14,15,18,19])
        n = np.size(x)
```

```
In [3]: x
Out[3]: array([1, 2, 3, 4, 5])
```

```
In [4]: x = x. reshape(-1,1)
        regression_model = LinearRegression ()
```

```
In [5]: # Fit the data (train the model)
        regression_model.fit(x, y)
Out[5]: LinearRegression()
```

```
In [6]: x
Out[6]: array([[1],
               [2],
               [3],
               [4],
               [5]])
```

```
In [7]: # Predict
        y_predicted = regression_model.predict(x)
```

```
In [6]: x
Out[6]: array([[1],
               [2],
               [3],
               [4],
               [5]])
```

```
In [7]: # Predict
        y_predicted = regression_model.predict(x)
```


```
In [8]: # model evaluation
        mse=mean_squared_error(y,y_predicted)
        rmse = np.sqrt(mean_squared_error(y, y_predicted))
        r2 = r2_score(y, y_predicted)

        #printing values
        print('Slope:', regression_model.coef_)
        print('Intercept:', regression_model.intercept_)
        print('MSE:', mse)
        print('Root mean squared error: ', rmse)
        print('R2 score: ', r2)

        Slope: [2.8]
        Intercept: 6.1999999999999975
        MSE: 2.160000000000001
        Root mean squared error:  1.4696938456699071
        R2 score:  0.8789237668161435
```

---

c) <u>Logistic Regression using Sci-kit learn and 50 startups dataset:</u>

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
```

In [10]:
```
dataset = pd.read_csv('50_Startups.csv')
X = dataset.iloc[:,: -1]
y = dataset.iloc[:, 4]
```

In [11]:
```
print(dataset.head(10))
```

```
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
2  153441.51       101145.55        407934.54     Florida  191050.39
3  144372.41       118671.85        383199.62    New York  182901.99
4  142107.34        91391.77        366168.42     Florida  166187.94
5  131876.90        99814.71        362861.36    New York  156991.12
6  134615.46       147198.87        127716.82  California  156122.51
7  130298.13       145530.06        323876.68     Florida  155752.60
8  120542.52       148718.95        311613.29    New York  152211.77
9  123334.88       108679.17        304981.62  California  149759.96
```

In [13]:
```
states = pd.get_dummies(X['State'], drop_first = True)
```

In [14]:
```
X = X.drop('State', axis=1)
X = pd.concat([X, states], axis=1)
```

In [18]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[18]:
```
LinearRegression()
```

In [22]:
```
y_pred = regressor.predict(X_test)
score = r2_score(y_test, y_pred)
print('R2 is:',score)
```

```
R2 is: 0.9603594733442932
```

In [21]:
```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print('MSE is:', mse)
print('RMSE is:', rmse)
```

```
MSE is: 43035873.733029984
RMSE is: 6560.173300533301
```

Inference:
So from the above two parts, we understand logistic regression mathematically by implementing the mathematical logic in the code and in the second part we have used in the inbuilt libraries using scikit learn and various metrics and the graphs are visualized using matplotlib.

Results:
Logistic Regression is proved and the metrics scored are verified both theortically and the graphs are visualized and plotted using matplotlib.