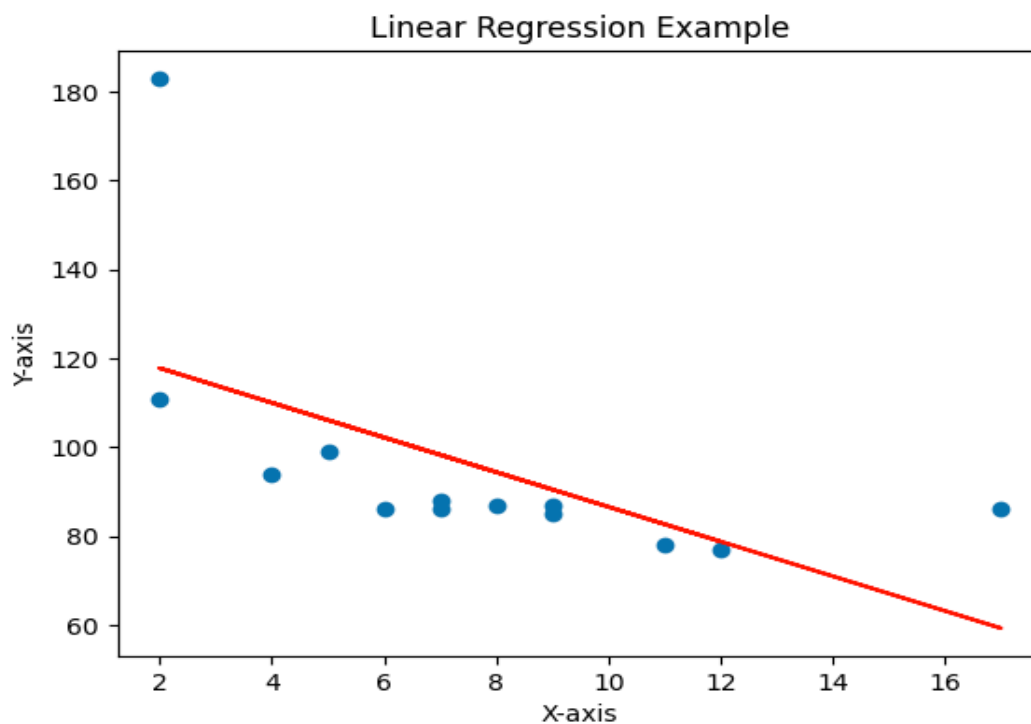


Implementation and result:

Linear Regression

```
import matplotlib.pyplot as plt
from scipy import stats
x= [5,7,8,7,2,17,2,9,4,11,12,9,6]
y=[99,86,87,88,111,86,183,87,94,78,77,85,86]
slope, intercept, r, p, std_err=stats.linregress (x, y)
print("r-value:", r)
print("p-value:", p)
print("Standard error:", std_err)
def myfunc(x):
    return slope *x + intercept
mymodel =list(map(myfunc, x))
plt.scatter(x,y)
plt.plot(x, mymodel, color='red')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Linear Regression Example')
plt.show()
```

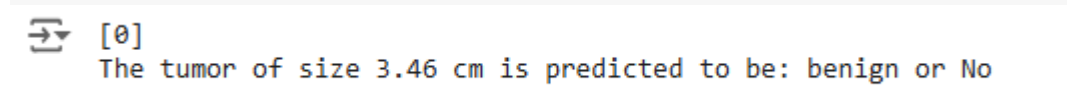
```
➤ r-value: -0.5898776643754048
p-value: 0.03383790892389773
Standard error: 1.6096693822889183
```



Logistic Regression :- Logistic regression is a statistical method used for binary classification, where the goal is to predict the probability of an outcome belonging to one of two classes. In Python, the

matplotlib library can be used to visually represent the logistic regression model. By plotting the data points and the decision boundary (the curve representing the threshold between the two classes), you can visually assess how well the model fits the data. Typically, a logistic regression curve is plotted by calculating the probability of class membership for each data point using the logistic function, and then showing this curve alongside the data points. This allows analysts to understand both the decision rule and the distribution of the classes, helping to evaluate the model's performance.

```
import numpy as np
from sklearn import linear_model
X= np.array([3.78, 2.44, 2.89, 8.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69, 5.88]).reshape(-1, 1)
y= np.array([0,0,0,0,0, 0, 1, 1, 1, 1, 1, 1])
logr=linear_model.LogisticRegression()
logr.fit(X, y)
predicted=logr.predict(np.array([3.46]).reshape(-1, 1))
print(predicted)
print(f'The tumor of size 3.46 cm is predicted to be: {'malignant or Yes' if predicted[0]== 1 else 'benign or No'})
```



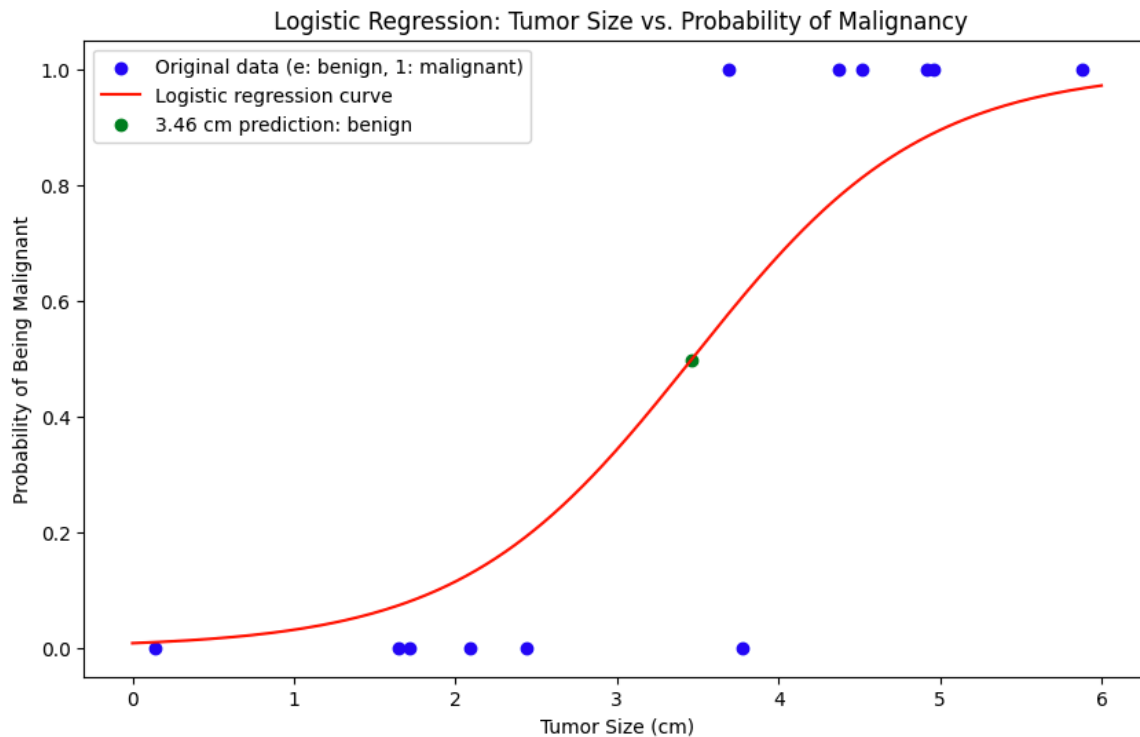
The output of the code is a Jupyter Notebook cell showing the result of the prediction. It displays the value [0] and the corresponding text: "The tumor of size 3.46 cm is predicted to be: benign or No".

2.Logistic Regression with plot. :-

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
x= np.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69, 5.88]).reshape(-1, 1)
y=np.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
logr=linear_model.LogisticRegression()
logr.fit(X, y)
probs=logr.predict_proba(X)[:,1]
new_data=np.array([3.46]).reshape(-1, 1)
predicted=logr.predict(new_data)
predicted_prob=logr.predict_proba(new_data)[:, 1]
print("Prediction for tumor size 3.46 cm: ('malignant if predicted[0] 1 else 'benign')")
print("Probability of being malignant (predicted_prob[@])")
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Original data (e: benign, 1: malignant)')
x_values=np.linspace(0, 6, 300).reshape(-1, 1)
y_values=logr.predict_proba(x_values)[:,1]
plt.plot(x_values,y_values, color='red', label='Logistic regression curve')
plt.scatter(new_data, predicted_prob, color='green', marker='o',label=f'3.46 cm prediction: {"malignant" if predicted[0] == 1 else "benign"}')
plt.xlabel('Tumor Size (cm)')
```

```
plt.ylabel('Probability of Being Malignant')
plt.title(' Logistic Regression: Tumor Size vs. Probability of Malignancy')
plt.legend()
plt.show()
```

Prediction for tumor size 3.46 cm: ('malignant' if predicted[0] 1 else 'benign')
 Probability of being malignant (predicted_prob[0])



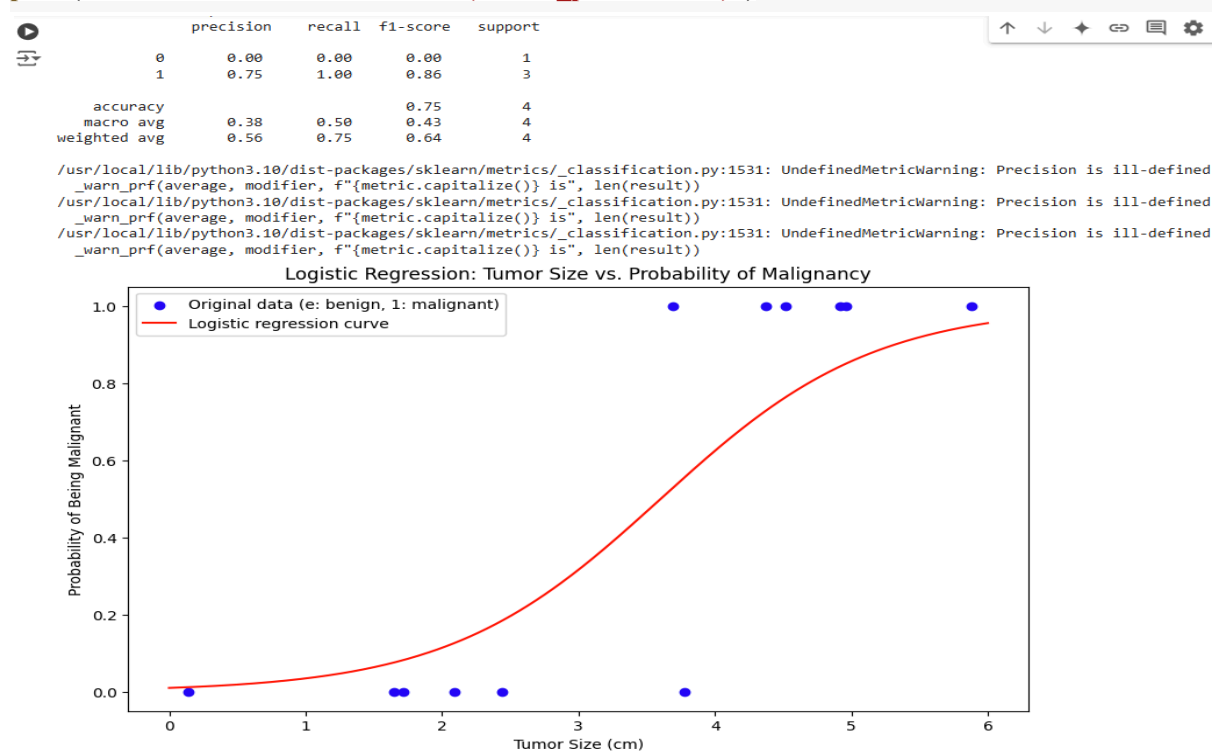
3.Logistic Regression-on unseen Data

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
x=np.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69, 5.88]).reshape(-1, 1)
y=np.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
X_train, X_test, y_train, y_test=train_test_split(x, y, test_size=0.3, random_state=42)
logr= linear_model.LogisticRegression()
logr.fit(X_train, y_train)
y_pred =logr.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report: \n", classification_report(y_test, y_pred))
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Original data (e: benign, 1: malignant)')
x_values= np.linspace(0, 6, 300).reshape(-1, 1)
```

```

y_values=logr.predict_proba(x_values)[:, 1]
plt.plot(x_values, y_values, color='red', label='Logistic regression curve')
plt.xlabel('Tumor Size (cm)')
plt.ylabel('Probability of Being Malignant')
plt.title('Logistic Regression: Tumor Size vs. Probability of Malignancy')
plt.legend()
plt.show()
unseen_data=np.array([3.46, 4.0, 2.5]).reshape(-1, 1)
unseen_predictions=logr.predict(unseen_data)
unseen_probabilities=logr.predict_proba(unseen_data)[:, 1]
print(f"Predictions for unseen data: (unseen_predictions)")
print(f"Probabilities for unseen data: (unseen_probabilities)")

```



4. Logistic Regression-Function to predict based on user input.

```

import numpy as np
from sklearn import linear_model
x= np.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69, 5.88]).reshape(-1, 1)
y=np.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
logr =linear_model.LogisticRegression()
logr.fit(X, y)
def predict_tumor_size():
    tumor_size=float(input("Enter tumor size in cm: "))
    input_data= np.array([tumor_size]).reshape(-1, 1)
    prediction= logr.predict(input_data)
    probability =logr.predict_proba(input_data)[:, 1]
    result= 'malignant' if prediction[0] == 1 else 'benign'

```

```
print(f"The tumor of size (tumor size) ce is predicted to be: (result)")
```

```
print(f"Probability of being malignant (probability[0])")
```

```
predict_tumor_size()
```



```
Enter tumor size in ca: 5
```

```
The tumor of size (tumor size) ce is predicted to be: (result)
```

```
Probability of being malignant (probability[0])
```

Colab Link :- <https://colab.research.google.com/drive/1BYGjBhbHZkvWjUZ1JMr6YGIGM6q8v6dg>