In [1]:

```
import pandas as pd
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.drop('Scores', axis=1)
y = data['Hours']
from sklearn.model_selection import train_test_split
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, ra
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0
print(X)
print(y)
```

```
    Hours
0     2.5
1     5.1
2     3.2
3     8.5
4     3.5
5     1.5
6     9.2
7     5.5
8     8.3
9     2.7
10    7.7
11    5.9
12    4.5
13    3.3
14    1.1
15    8.9
16    2.5
17    1.9
18    6.1
19    7.4
20    2.7
21    4.8
22    3.8
23    6.9
24    7.8
0     2.5
1     5.1
2     3.2
3     8.5
4     3.5
5     1.5
6     9.2
7     5.5
8     8.3
9     2.7
10    7.7
11    5.9
12    4.5
13    3.3
14    1.1
15    8.9
16    2.5
17    1.9
18    6.1
19    7.4
20    2.7
21    4.8
22    3.8
23    6.9
24    7.8
Name: Hours, dtype: float64
```

Train set

```
In [2]: import pandas as pd
        from sklearn.model_selection import train_test_split
        data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
        X = data.drop('Scores', axis=1)
        y = data['Hours']
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
        X
```

```
Out[2]:
     Hours
 0     2.5
 1     5.1
 2     3.2
 3     8.5
 4     3.5
 5     1.5
 6     9.2
 7     5.5
 8     8.3
 9     2.7
10     7.7
11     5.9
12     4.5
13     3.3
14     1.1
15     8.9
16     2.5
17     1.9
18     6.1
19     7.4
20     2.7
21     4.8
22     3.8
23     6.9
24     7.8
```

Validation sat

In [3]:

```
import csv
from sklearn.model_selection import train_test_split
data = []
with open("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/EXCEL/stud
    csv_reader = csv.reader(file)
    for row in csv_reader:
        data.append(row)
X = [row[:-1] for row in data]
y = [row[-1] for row in data]
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, rand
X
```

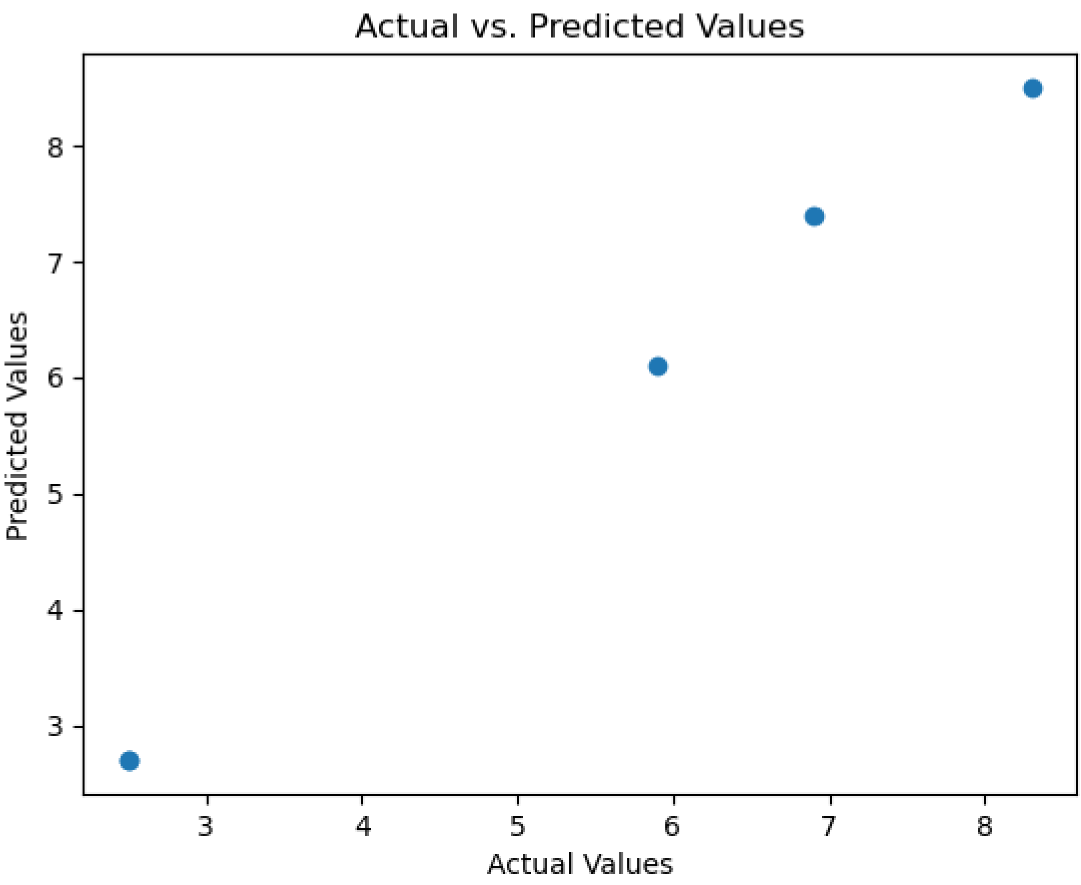
Out[3]:

```
[['Hours'],
 ['2.5'],
 ['5.1'],
 ['3.2'],
 ['8.5'],
 ['3.5'],
 ['1.5'],
 ['9.2'],
 ['5.5'],
 ['8.3'],
 ['2.7'],
 ['7.7'],
 ['5.9'],
 ['4.5'],
 ['3.3'],
 ['1.1'],
 ['8.9'],
 ['2.5'],
 ['1.9'],
 ['6.1'],
 ['7.4'],
 ['2.7'],
 ['4.8'],
 ['3.8'],
 ['6.9'],
 ['7.8']]
```

Overfitting

```
In [4]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.metrics import mean_squared_error
        import matplotlib.pyplot as plt
        data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
        X = data.drop('Scores', axis=1)
        y = data['Hours']
        X.fillna(X.mean(), inplace=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
        model = DecisionTreeRegressor(max_depth=None)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)
        print(f"Mean Squared Error: {mse}")
        plt.scatter(y_test, y_pred)
        plt.xlabel("Actual Values")
        plt.ylabel("Predicted Values")
        plt.title("Actual vs. Predicted Values")
        plt.show()
```

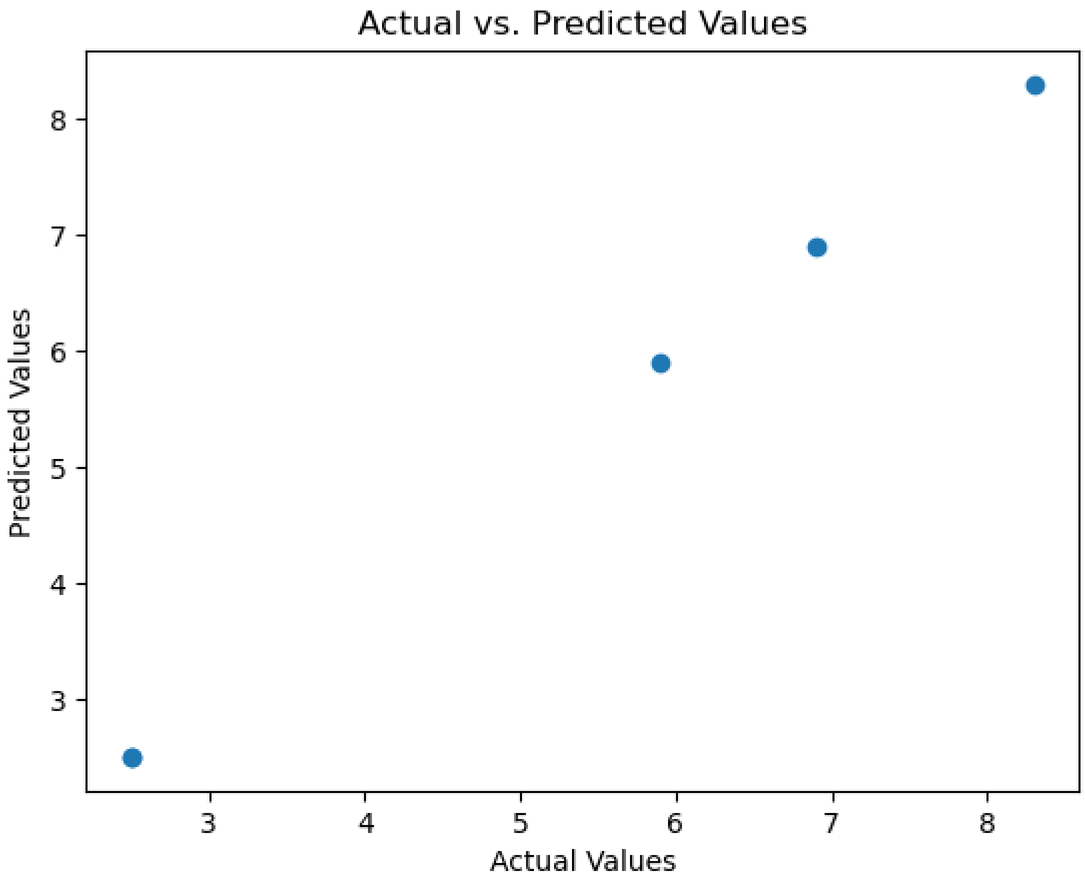
```
Mean Squared Error: 0.08199999999999992
```



Underfitting

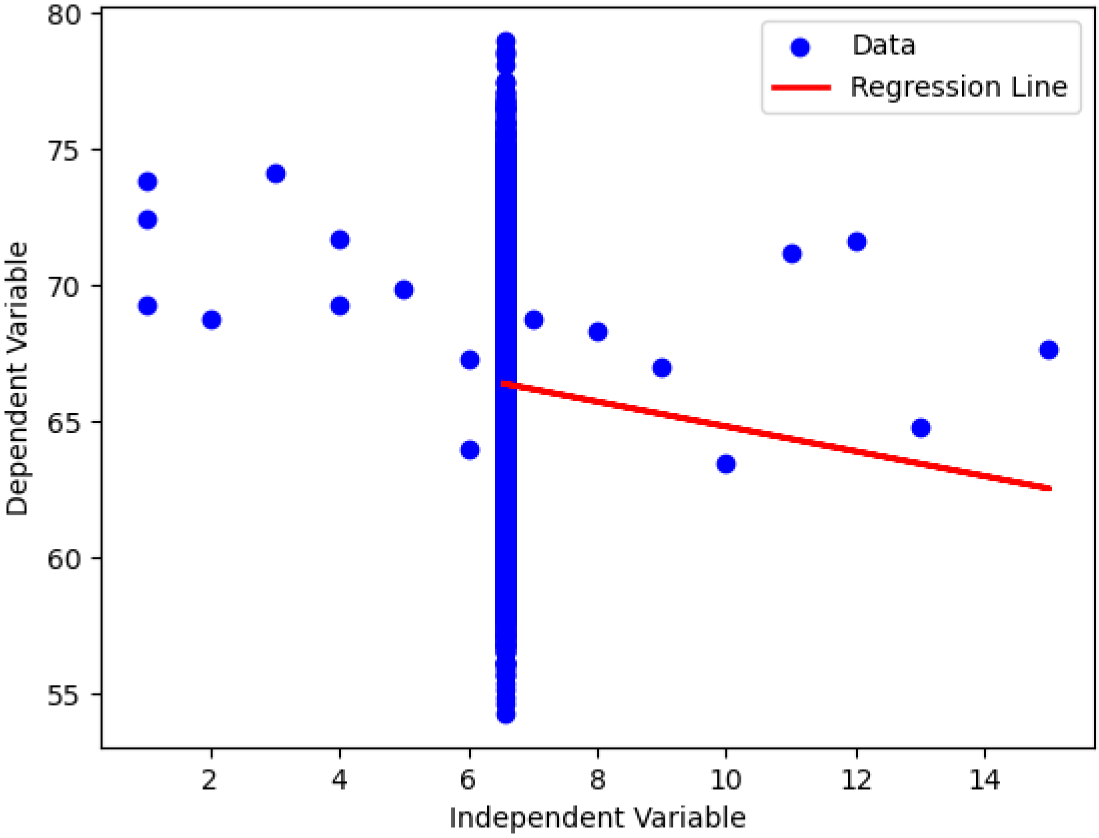
```
In [5]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import mean_squared_error
        import matplotlib.pyplot as plt
        data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
        X = data.drop('Scores', axis=1)
        y = data['Hours']
        X.fillna(X.mean(), inplace=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
        model = LinearRegression()
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)
        print(f"Mean Squared Error: {mse}")
        plt.scatter(y_test, y_pred)
        plt.xlabel("Actual Values")
        plt.ylabel("Predicted Values")
        plt.title("Actual vs. Predicted Values")
        plt.show()
```

```
Mean Squared Error: 1.262177448353619e-30
```



Simple linear regression

```
In [6]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        import matplotlib.pyplot as plt
        data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
        X = data[['Number']]
        y = data['Height']
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
        model = LinearRegression()
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        plt.scatter(X, y, color='blue', label='Data')
        plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
        plt.xlabel('Independent Variable')
        plt.ylabel('Dependent Variable')
        plt.legend()
        plt.show()
        slope = model.coef_[0]
        intercept = model.intercept_
        print(f"Slope (m): {slope}")
        print(f"Intercept (b): {intercept}")
```



```
Slope (m): -0.45626191596593185
Intercept (b): 69.36866404251712
```

Multiple linear regression¶

In [7]:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
print(data.head())
print(data.info())
X = data[['Scores']]
y = data['Hours']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r2)
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

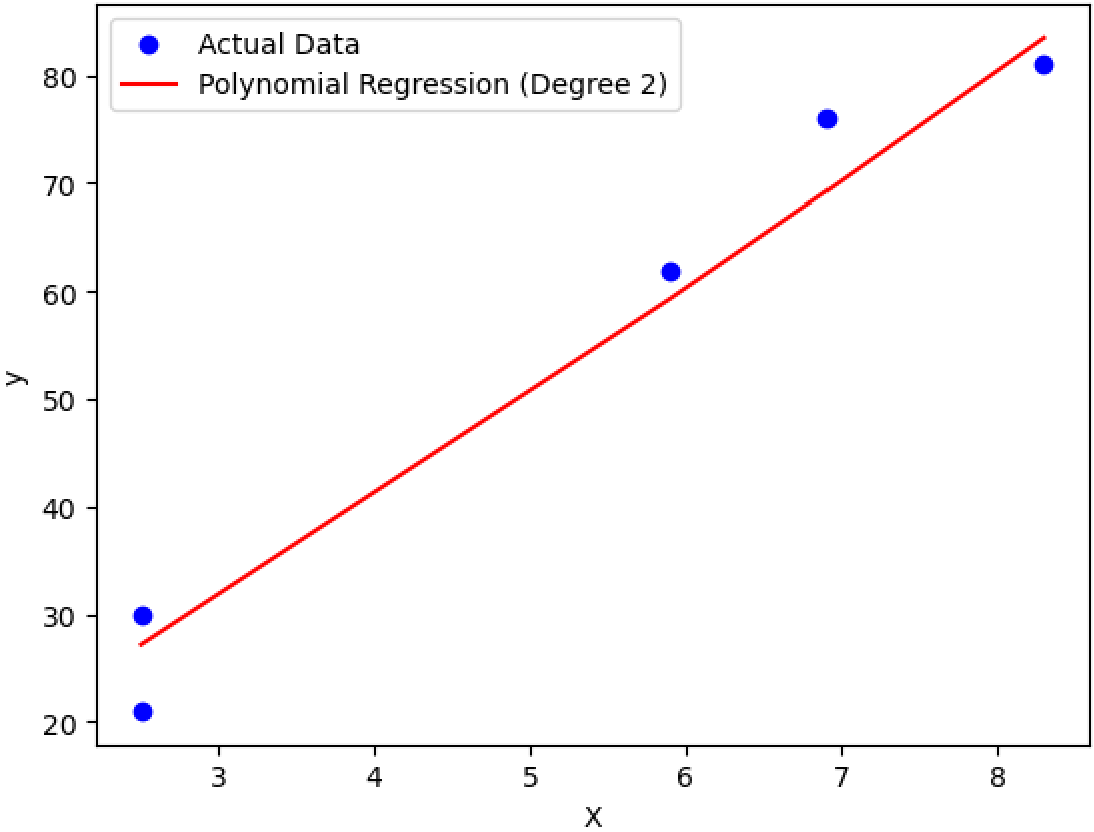
```
   Hours  Scores
0    2.5      21
1    5.1      47
2    3.2      27
3    8.5      75
4    3.5      30
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
None
Mean Squared Error: 0.16754295656677842
R-squared: 0.9696127835594206
Coefficients: [0.09802864]
Intercept: -0.024756258820576527
```

Polynomial linear regression

In [8]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data[['Hours']]
y = data['Scores']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
degree = 2  # Adjust the degree of the polynomial as needed
poly_features = PolynomialFeatures(degree=degree)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)
poly_reg = LinearRegression()
poly_reg.fit(X_train_poly, y_train)
y_pred = poly_reg.predict(X_test_poly)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)
X_test_sorted, y_pred_sorted = zip(*sorted(zip(X_test.values, y_pred)))
plt.scatter(X_test, y_test, color='blue', label='Actual Data')
plt.plot(X_test_sorted, y_pred_sorted, color='red', label=f'Polynomial Regr
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
Mean Squared Error: 21.066769638340965
R-squared (R2) Score: 0.9641965165901751
```

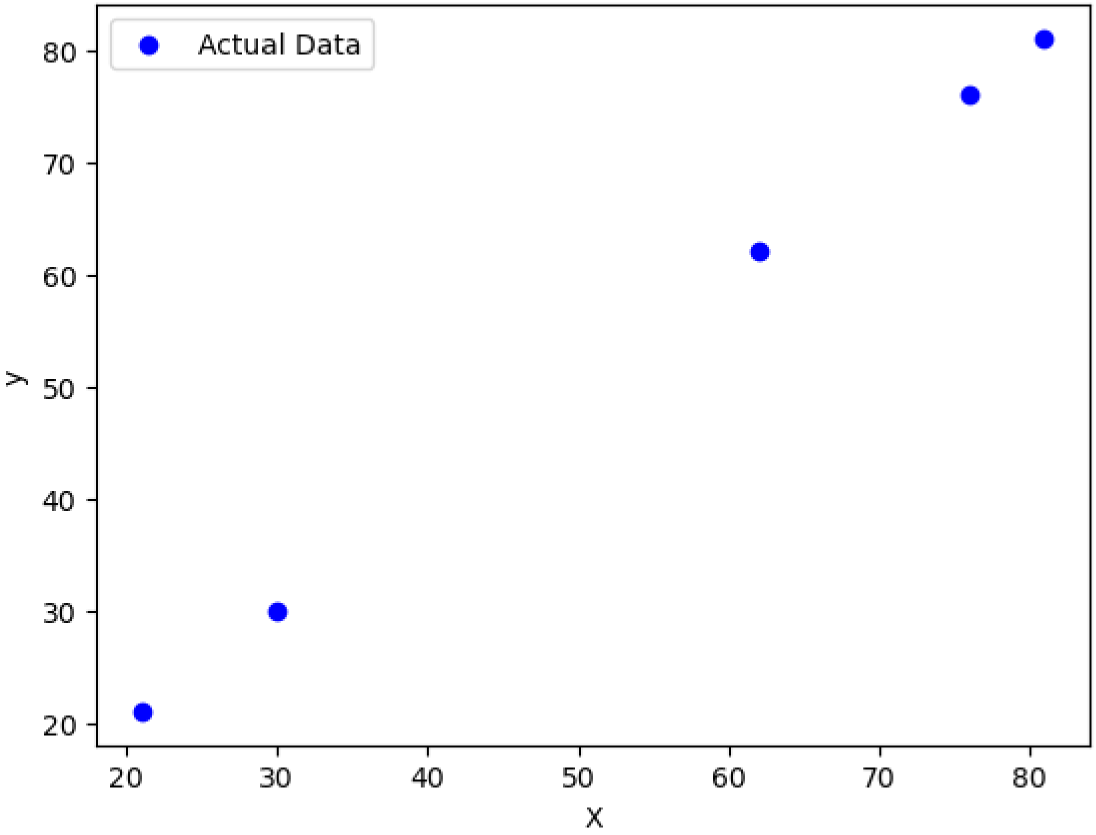


logistic Regression¶

In [9]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.drop('Hours', axis=1)
y = data['Scores']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
plt.scatter(X_test, y_test, color='blue', label='Actual Data')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
Accuracy: 0.2

C:\Users\bhava\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
```

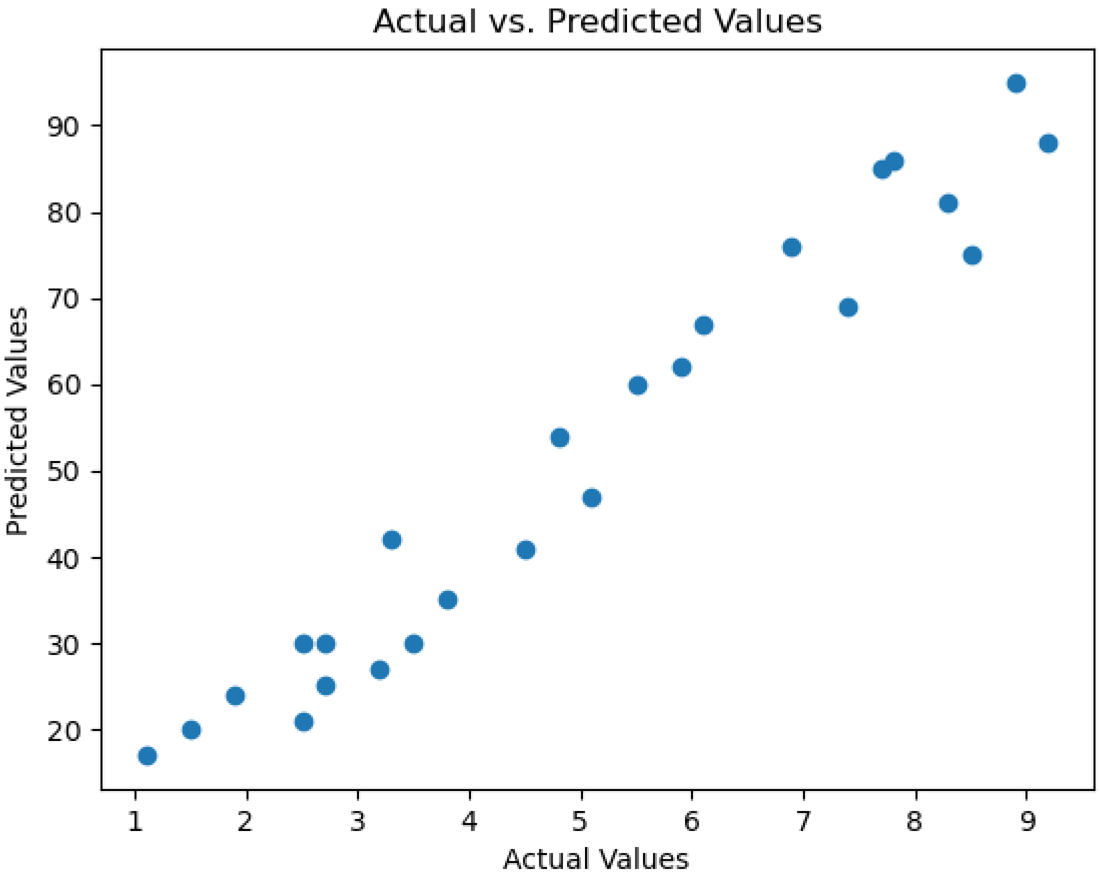


Equation regression model

In [10]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sco
import numpy as np
test_data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th
X_test = test_data[['Scores']]
y_test = test_data['Hours']
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sco

mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("Mean Absolute Error:", mae)
print("R-squared:", r2)

import matplotlib.pyplot as plt

plt.scatter(y_test, y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values")
plt.show()
```

```
Mean Squared Error: 2659.5691999999995
Root Mean Squared Error: 51.57101123693426
Mean Absolute Error: 46.468
R-squared: -433.4951590052434
```



Evaluation Matrix

In [11]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sco
import numpy as np
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.drop('Hours', axis=1)
y = data['Scores']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error (MSE): {mse:.4f}')
print(f'Root Mean Squared Error (RMSE): {rmse:.4f}')
print(f'Mean Absolute Error (MAE): {mae:.4f}')
print(f'R-squared (R²): {r2:.4f}')
```

```
Mean Squared Error (MSE): 0.0000
Root Mean Squared Error (RMSE): 0.0000
Mean Absolute Error (MAE): 0.0000
R-squared (R²): 1.0000
```

R square R2

In [12]:

```
##Root Mean Squared Error (RMSE)
import numpy as np
from sklearn.metrics import mean_squared_error
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
rmse
```

Out[12]:

```
0.0
```

Route mean squared error

In [13]:

```
## adjusted R-squared
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
num_features = X_train.shape[1]
num_data_points = len(y_test)
adjusted_r2 = 1 - (1 - r2) * (num_data_points - 1) / (num_data_points - num
adjusted_r2
```

Out[13]:

```
1.0
```

Mean absolute error

In [14]:

```
## Mean Absolute Error(MAE)
import pandas as pd
from sklearn.metrics import mean_absolute_error
data=pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/EXC
actual = data['Height']
predicted = data['Number']
mae = mean_absolute_error(actual, predicted)
print(f"Mean Absolute Error (MAE): {mae}")
```

```
Mean Absolute Error (MAE): 59.812004198866795
```

Mean squared error

In [15]:

```
## Mean Squared Error
import pandas as pd
from sklearn.metrics import mean_squared_error
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
actual = data['Number']
predicted = data['Height']
mse = mean_squared_error(actual, predicted)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 3592.3321704373852
```

Cross validation

In [47]:
```
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestRegressor  # Example model
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data[['Height', 'Weight']]  # Features
y = data['Number']  # Target variable
model = RandomForestRegressor()  # Example model
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mae_scores = -cross_val_score(model, X, y, cv=kf, scoring='neg_mean_absolut
mean_mae = mae_scores.mean()
std_mae = mae_scores.std()
```

In [48]:
```
mae_scores
```

Out[48]:
```
array([0.01570444, 0.00432278, 0.01376111, 0.01050611, 0.01677333])
```

In [49]:
```
mean_mae
```

Out[49]:
```
0.012213555555822594
```

In [50]:
```
std_mae
```

Out[50]:
```
0.004484474267726706
```

Hold out method

In [18]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
data=pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/EXC
X = data.drop('Hours', axis=1)
y = data['Scores']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.125
```

leave one out cross validation

In [20]:

```
import pandas as pd
from sklearn.model_selection import LeaveOneOut
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.drop('Hours', axis=1)
y = data['Scores']
loo = LeaveOneOut()
predicted_labels = []
true_labels = []
for train_index, test_index in loo.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    model = DecisionTreeClassifier()
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    predicted_labels.extend(y_pred)
    true_labels.extend(y_test)
accuracy = accuracy_score(true_labels, predicted_labels)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.12
```

k fold cross validation

In [21]:

```
import pandas as pd
from sklearn.model_selection import cross_val_score, KFold
from sklearn.tree import DecisionTreeClassifier
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.drop('Hours', axis=1)
y = data['Scores']
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
model = DecisionTreeClassifier()
scores = cross_val_score(model, X, y, cv=kfold)

for i, score in enumerate(scores, start=1):
    print(f'Fold {i}: {score:.4f}')
print(f'Mean Accuracy: {scores.mean():.4f}')
print(f'Standard Deviation: {scores.std():.4f}')
```

```
Fold 1: 0.2000
Fold 2: 0.0000
Fold 3: 0.2000
Fold 4: 0.2000
Fold 5: 0.0000
Mean Accuracy: 0.1200
Standard Deviation: 0.0980
```

Random forest regression

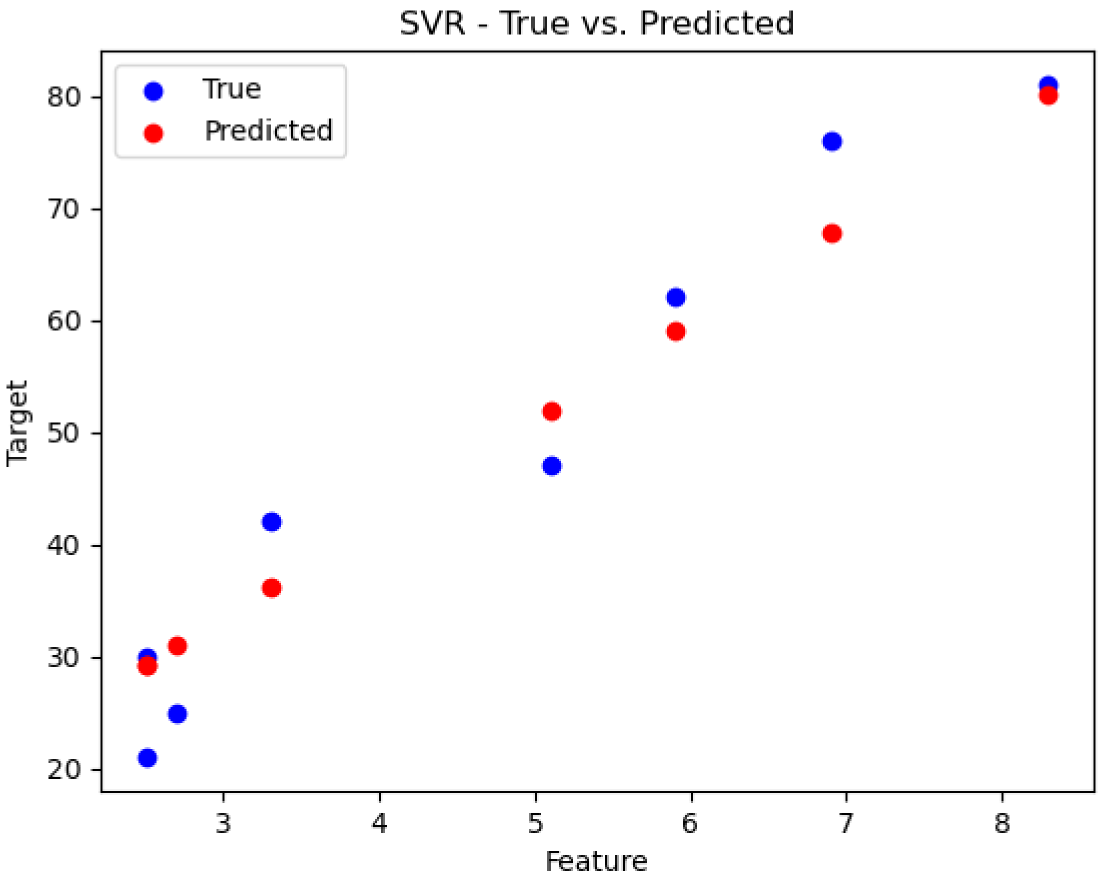
In [22]:

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
X, y = make_regression(n_features=4, n_informative=2, random_state=0, shuff
rfr = RandomForestRegressor(max_depth=3)
rfr.fit(X, y)
print(rfr.predict([[0, 1, 0, 1]]))
```

```
[36.7820371]
```

Support vector regression

```
In [23]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         from sklearn.model_selection import train_test_split
         from sklearn.svm import SVR
         from sklearn.metrics import mean_squared_error, r2_score
         data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
         X = data[['Hours']]
         y = data['Scores']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
         model = SVR(kernel='linear')
         model.fit(X_train, y_train)
         y_pred = model.predict(X_test)
         mse = mean_squared_error(y_test, y_pred)
         r2 = r2_score(y_test, y_pred)
         plt.scatter(X_test, y_test, color='b', label='True')
         plt.scatter(X_test, y_pred, color='r', label='Predicted')
         plt.title('SVR - True vs. Predicted')
         plt.xlabel('Feature')
         plt.ylabel('Target')
         plt.legend(loc='upper left')
         plt.show()

         print(f'Mean Squared Error (MSE): {mse:.4f}')
         print(f'R-squared (R²): {r2:.4f}')
```

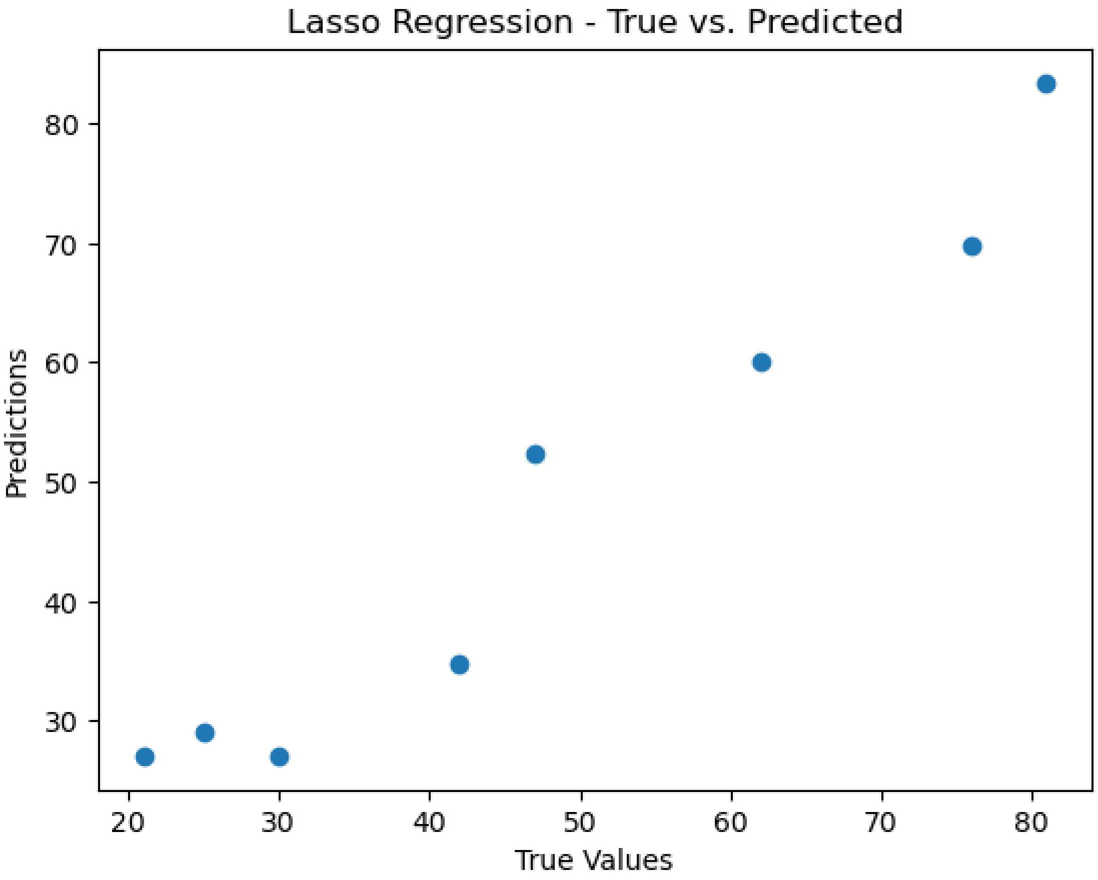


```
Mean Squared Error (MSE): 29.9038
R-squared (R²): 0.9351
```

lasso Regression

In [24]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data[['Hours']]
y = data['Scores']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ra
model = Lasso(alpha=0.01)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
plt.scatter(y_test, y_pred)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.title('Lasso Regression - True vs. Predicted')
plt.show()

print(f'Mean Squared Error (MSE): {mse:.4f}')
print(f'R-squared (R²): {r2:.4f}')
```



```
Mean Squared Error (MSE): 23.6231
R-squared (R²): 0.9488
```

Decision tree classifier

In [25]:

```python
import pandas as pd
from sklearn.model_selection import LeaveOneOut
from sklearn.tree import DecisionTreeClassifier  # Replace with your chosen
from sklearn.metrics import accuracy_score  # Replace with appropriate eval
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E

X = data.drop('Hours', axis=1)
y = data['Scores']
loo = LeaveOneOut()
model = DecisionTreeClassifier()
predicted_labels = []
true_labels = []
for train_index, test_index in loo.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

predicted_labels.extend(y_pred)
true_labels.extend(y_test)
accuracy = accuracy_score(true_labels, predicted_labels)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.0
```

Classification

In [26]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.drop('Hours', axis=1)
y = data['Scores']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.20
Classification Report:
              precision    recall  f1-score   support

          20       0.00      0.00      0.00         0
          21       0.00      0.00      0.00         1
          30       1.00      1.00      1.00         1
          60       0.00      0.00      0.00         0
          62       0.00      0.00      0.00         1
          75       0.00      0.00      0.00         0
          76       0.00      0.00      0.00         1
          81       0.00      0.00      0.00         1
          85       0.00      0.00      0.00         0

    accuracy                           0.20         5
   macro avg       0.11      0.11      0.11         5
weighted avg       0.20      0.20      0.20         5
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Binary classifier

In [27]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
logistic_classifier = LogisticRegression()
logistic_classifier.fit(X_train, y_train)
y_pred = logistic_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.20
Classification Report:
              precision    recall  f1-score   support

          21       0.00      0.00      0.00         1
          30       0.50      1.00      0.67         1
          62       0.00      0.00      0.00         1
          67       0.00      0.00      0.00         0
          69       0.00      0.00      0.00         0
          76       0.00      0.00      0.00         1
          81       0.00      0.00      0.00         1
          88       0.00      0.00      0.00         0

    accuracy                           0.20         5
   macro avg       0.06      0.12      0.08         5
weighted avg       0.10      0.20      0.13         5
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://sc
ikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression (https://scikit-learn.org/stable/modules/linear_model.html#logisti
c-regression)
  n_iter_i = _check_optimize_result(
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Multi class classifier

In [28]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.20
Classification Report:
              precision    recall  f1-score   support

          21       0.00      0.00      0.00         1
          30       0.50      1.00      0.67         1
          62       0.00      0.00      0.00         1
          67       0.00      0.00      0.00         0
          69       0.00      0.00      0.00         0
          75       0.00      0.00      0.00         0
          76       0.00      0.00      0.00         1
          81       0.00      0.00      0.00         1

    accuracy                           0.20         5
   macro avg       0.06      0.12      0.08         5
weighted avg       0.10      0.20      0.13         5
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Confusion matrix

In [29]:

```
import pandas as pd
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
# Split the data into features (X) and the target variable (y)
X = data.drop('Hours', axis=1)  # Adjust 'target_column' to your target var
y = data['Scores']
test_size = 0.33
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, y, t
model = LogisticRegression(solver='liblinear')
model.fit(X_train, Y_train)
predicted = model.predict(X_test)
matrix = confusion_matrix(Y_test, predicted)
print(matrix)
```

```
[[0 0 0 1 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0]]
```

AUC ROC

In [31]:

```
import pandas as pd
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
column_name = 'Scores'

# Calculate Gini index for a specific column
def calculate_gini(data, column):
    total_rows = len(data)
    unique_classes = data[column].unique()
    gini = 0

    for label in unique_classes:
        proportion = (len(data[data[column] == label]) / total_rows) ** 2
        gini += proportion

    gini_index = 1 - gini
    return gini_index

gini_value = calculate_gini(data, column_name)
print(f"Gini Index for '{column_name}': {gini_value:.4f}")
```

```
Gini Index for 'Scores': 0.9504
```

Gini

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
y_prob = rf_classifier.predict_proba(X_test)[:, 1]  # Considering positive
auc_roc = roc_auc_score(y_test, y_prob)
print(f"AUC-ROC Score: {auc_roc:.2f}")
```

Entropy

In [32]:

```
import pandas as pd
import math
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
column_name = 'Hours'

# Calculate entropy for a specific column
def calculate_entropy(data, column):
    total_rows = len(data)
    unique_classes = data[column].unique()
    entropy = 0

    for label in unique_classes:
        proportion = len(data[data[column] == label]) / total_rows
        entropy -= proportion * math.log2(proportion)

    return entropy

entropy_value = calculate_entropy(data, column_name)
print(f"Entropy for '{column_name}': {entropy_value:.4f}")
```

```
Entropy for 'Hours': 4.4839
```

Boston housing price from sci-kit learn

In [33]:

```
from sklearn.datasets import load_boston
boston = load_boston()
X = boston.data
y = boston.target
print(boston.DESCR)
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value
(attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,0
00 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds rive
r; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of black p
eople by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/ (http
s://archive.ics.uci.edu/ml/machine-learning-databases/housing/)


This dataset was taken from the StatLib library which is maintained at Car
negie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnost
ics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers
that address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influenti
al Data and Sources of Collinearity', Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning.
In Proceedings on the Tenth International Conference of Machine Learning,
236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:8
7: FutureWarning: Function load_boston is deprecated; `load_boston` is dep
recated in 1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refe
r to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of
this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=Non
e)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housi
ng
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

In [34]:

```
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
boston = load_boston()
X = boston.data
y = boston.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print("Mean Squared Error (MSE):", mse)
print("R-squared (R2):", r2)
```

```
Mean Squared Error (MSE): 24.29111947497374
R-squared (R2): 0.6687594935356289
```

Cricket match result - past data

In [35]:

```
import pandas as pd

data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
print(data.head())

run_a_wins = len(data[data['Runs'] == 'run A'])
hs_b_wins = len(data[data['HS'] == 'highest_scores B'])

print("Number of matches won by Team A:", run_a_wins)
print("Number of matches won by Team B:", hs_b_wins)
```

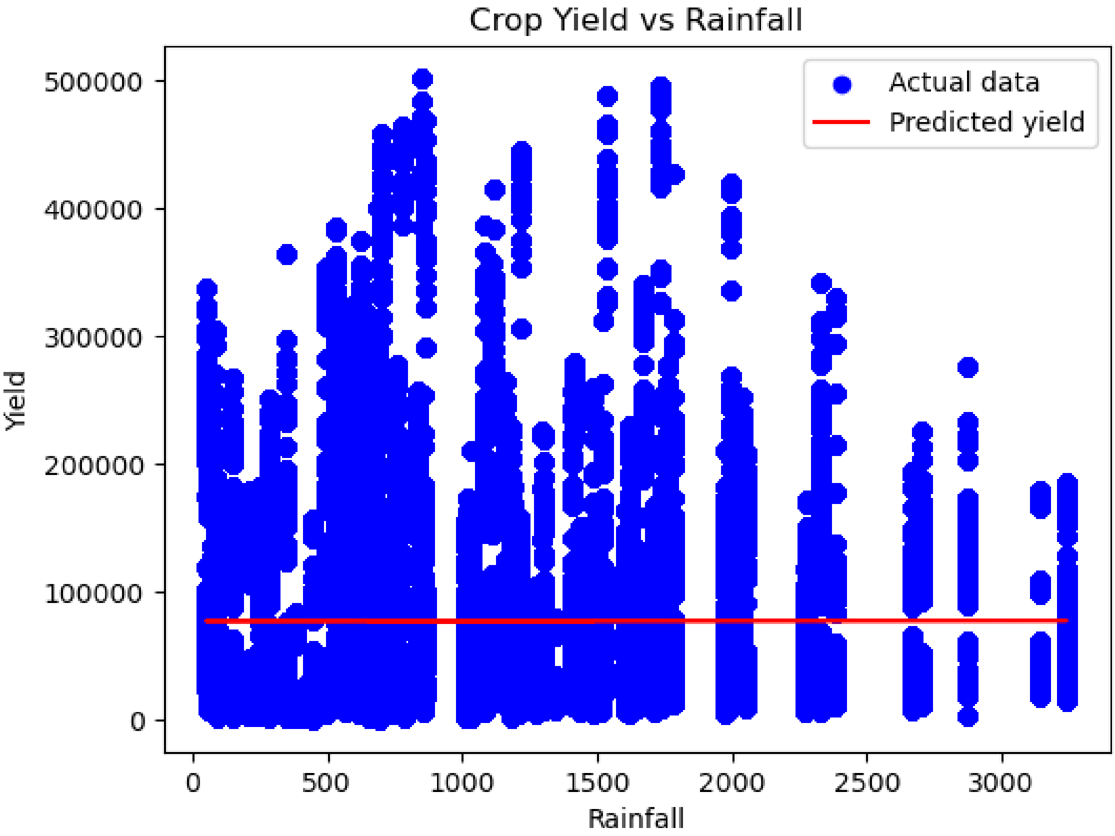
```
   Unnamed: 0                       Player       Span  Mat  Inns  NO   Runs  \
0           0           SR Tendulkar (INDIA)  1989-2012  463   452  41  18426
1           1  KC Sangakkara (Asia/ICC/SL)  2000-2015  404   380  41  14234
2           2           RT Ponting (AUS/ICC)  1995-2012  375   365  39  13704
3           3        ST Jayasuriya (Asia/SL)  1989-2011  445   433  18  13430
4           4   DPMD Jayawardene (Asia/SL)  1998-2015  448   418  39  12650

     HS    Ave     BF     SR  100  50   0  Unnamed: 13
0  200*  44.83  21367  86.23   49  96  20          NaN
1   169  41.98  18048  78.86   25  93  15          NaN
2   164  42.03  17046  80.39   30  82  20          NaN
3   189  32.36  14725   91.2   28  68  34          NaN
4   144  33.37  16020  78.96   19  77  28          NaN
Number of matches won by Team A: 0
Number of matches won by Team B: 0
```

Cricket match result-past data

In [36]:

```python
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
data = pd.read_csv('integrated_data.csv')
print(data.head())
X = data[['average_rain_fall_mm_per_year_y']]
y = data['hg/ha_yield']
model = LinearRegression()
model.fit(X, y)
predicted_yield = model.predict(X)
plt.scatter(X, y, color='blue', label='Actual data')
plt.plot(X, predicted_yield, color='red', label='Predicted yield')
plt.xlabel('Rainfall')
plt.ylabel('Yield')
plt.legend()
plt.title('Crop Yield vs Rainfall')
plt.show()
```

```
C:\Users\bhava\AppData\Local\Temp\ipykernel_13236\3090302346.py:4: DtypeWa
rning: Columns (2) have mixed types. Specify dtype option on import or set
low_memory=False.
  data = pd.read_csv('integrated_data.csv')

          Area  Year average_rain_fall_mm_per_year_x  Unnamed: 0     Area
\
0  Afghanistan  1990                             327           0  Albania
1  Afghanistan  1990                             327           1  Albania
2  Afghanistan  1990                             327           2  Albania
3  Afghanistan  1990                             327           3  Albania
4  Afghanistan  1990                             327           4  Albania

          Item  hg/ha_yield  average_rain_fall_mm_per_year_y  \
0        Maize        36613                           1485.0
1     Potatoes        66667                           1485.0
2  Rice, paddy        23333                           1485.0
3      Sorghum        12500                           1485.0
4     Soybeans         7000                           1485.0

   pesticides_tonnes  avg_temp
0              121.0     16.37
1              121.0     16.37
2              121.0     16.37
3              121.0     16.37
4              121.0     16.37
```



Crope yield pastdata

In [37]:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import pandas as pd
iris = load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
print(iris_df.head())
X = iris.data  # Features
y = iris.target  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (c
m)  \
0                5.1               3.5                1.4               0.
2
1                4.9               3.0                1.4               0.
2
2                4.7               3.2                1.3               0.
2
3                4.6               3.1                1.5               0.
2
4                5.0               3.6                1.4               0.
2

   target
0       0
1       0
2       0
3       0
4       0
X_train shape: (120, 4)
X_test shape: (30, 4)
y_train shape: (120,)
y_test shape: (30,)
```

Iris dataset from sci-kit learn perfom data exporation preprocessing and splitting

In [38]:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
breast_cancer = load_breast_cancer()
X = breast_cancer.data  # Features
y = breast_cancer.target  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.95
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        43
           1       0.96      0.96      0.96        71

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

Build randam forest based model

In [39]:

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
breast_cancer = load_breast_cancer()
X = breast_cancer.data  # Features
y = breast_cancer.target  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.96
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.93      0.95        43
           1       0.96      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

Max Voting

In [40]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassi
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
gb_classifier = GradientBoostingClassifier(n_estimators=100, learning_rate=

rf_classifier.fit(X_train, y_train)
gb_classifier.fit(X_train, y_train)
y_pred_rf = rf_classifier.predict(X_test)
y_pred_gb = gb_classifier.predict(X_test)
voting_predictions = []

for i in range(len(X_test)):
    combined_prediction = max(y_pred_rf[i], y_pred_gb[i])
    voting_predictions.append(combined_prediction)
accuracy = accuracy_score(y_test, voting_predictions)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:")
print(classification_report(y_test, voting_predictions))
```

```
Accuracy: 0.20
Classification Report:
              precision    recall  f1-score   support

          21       0.00      0.00      0.00         1
          30       0.50      1.00      0.67         1
          62       0.00      0.00      0.00         1
          67       0.00      0.00      0.00         0
          69       0.00      0.00      0.00         0
          75       0.00      0.00      0.00         0
          76       0.00      0.00      0.00         1
          81       0.00      0.00      0.00         1

    accuracy                           0.20         5
   macro avg       0.06      0.12      0.08         5
weighted avg       0.10      0.20      0.13         5
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Averaging

In [41]:

```
import numpy as np
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
predictions_model1 = np.array([0, 1, 1, 0, 1])  # Replace with actual predi
predictions_model2 = np.array([1, 1, 0, 0, 1])  # Replace with actual predi
predictions_model3 = np.array([1, 0, 1, 1, 0])  # Replace with actual predi

# Combine predictions using averaging
averaged_predictions = (predictions_model1 + predictions_model2 + predictio

# Round the averaged predictions for binary classification
rounded_averaged_predictions = np.round(averaged_predictions).astype(int)

# Display the averaged predictions and the rounded predictions
print("Averaged Predictions:", averaged_predictions)
print("Rounded Averaged Predictions:", rounded_averaged_predictions)
accuracy = accuracy_score(y_test, voting_predictions)
print(f"Accuracy: {accuracy:.2f}")
```

```
Averaged Predictions: [0.66666667 0.66666667 0.66666667 0.33333333 0.66666
667]
Rounded Averaged Predictions: [1 1 1 0 1]
Accuracy: 0.20
```

Weighted Average

In [42]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassi
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
gb_classifier = GradientBoostingClassifier(n_estimators=100, learning_rate=

rf_classifier.fit(X_train, y_train)
gb_classifier.fit(X_train, y_train)
y_pred_rf = rf_classifier.predict(X_test)
y_pred_gb = gb_classifier.predict(X_test)
weight_rf = 0.6  # Weight for Random Forest's prediction
weight_gb = 0.4  # Weight for Gradient Boosting's prediction
weighted_average_predictions = (weight_rf * y_pred_rf) + (weight_gb * y_pre

accuracy = accuracy_score(y_test, weighted_average_predictions)
print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")
print(classification_report(y_test, weighted_average_predictions))
```

```
Accuracy: 0.00
Classification Report:
              precision    recall  f1-score   support

        21.0       0.00      0.00      0.00       1.0
        28.0       0.00      0.00      0.00       0.0
        30.0       0.00      0.00      0.00       1.0
        62.0       0.00      0.00      0.00       1.0
        67.0       0.00      0.00      0.00       0.0
        69.0       0.00      0.00      0.00       0.0
        75.0       0.00      0.00      0.00       0.0
        76.0       0.00      0.00      0.00       1.0
        81.0       0.00      0.00      0.00       1.0

    accuracy                           0.00       5.0
   macro avg       0.00      0.00      0.00       5.0
weighted avg       0.00      0.00      0.00       5.0
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Stacking

In [43]:

```python
from sklearn.ensemble import StackingClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassi
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X, y = make_classification(n_samples=1000, n_features=20, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
base_estimators = [
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
    ('gb', GradientBoostingClassifier(n_estimators=100, learning_rate=0.1,
]
meta_estimator = LogisticRegression()
stacking_classifier = StackingClassifier(estimators=base_estimators, final_
stacking_classifier.fit(X_train, y_train)
predictions = stacking_classifier.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy of the stacked model: {accuracy:.2f}")
```

```
Accuracy of the stacked model: 0.90
```

Blending

In [44]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassi
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
gb_classifier = GradientBoostingClassifier(n_estimators=100, learning_rate=

rf_classifier.fit(X_train, y_train)
gb_classifier.fit(X_train, y_train)

y_pred_rf = rf_classifier.predict(X_test)
y_pred_gb = gb_classifier.predict(X_test)

blended_predictions = (y_pred_rf + y_pred_gb) / 2

final_prediction = [1 if pred >= 0.5 else 0 for pred in blended_predictions

accuracy = accuracy_score(y_test, final_prediction)
print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")
print(classification_report(y_test, final_prediction))
```

```
Accuracy: 0.00
Classification Report:
              precision    recall  f1-score   support

           1       0.00      0.00      0.00       0.0
          21       0.00      0.00      0.00       1.0
          30       0.00      0.00      0.00       1.0
          62       0.00      0.00      0.00       1.0
          76       0.00      0.00      0.00       1.0
          81       0.00      0.00      0.00       1.0

    accuracy                           0.00       5.0
   macro avg       0.00      0.00      0.00       5.0
weighted avg       0.00      0.00      0.00       5.0
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Bagging

```
In [45]: import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import accuracy_score, classification_report
         data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
         X = data.iloc[:, :-1]
         y = data.iloc[:, -1]

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

         rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

         rf_classifier.fit(X_train, y_train)

         y_pred = rf_classifier.predict(X_test)

         accuracy = accuracy_score(y_test, y_pred)
         print(f"Accuracy: {accuracy:.2f}")

         print("Classification Report:")
         print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.20
Classification Report:
              precision    recall  f1-score   support

          21       0.00      0.00      0.00         1
          30       0.50      1.00      0.67         1
          62       0.00      0.00      0.00         1
          67       0.00      0.00      0.00         0
          69       0.00      0.00      0.00         0
          75       0.00      0.00      0.00         0
          76       0.00      0.00      0.00         1
          81       0.00      0.00      0.00         1

    accuracy                           0.20         5
   macro avg       0.06      0.12      0.08         5
weighted avg       0.10      0.20      0.13         5
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Boosting

In [46]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report
data = pd.read_csv("C:/Users/bhava/OneDrive/Pictures/Desktop/GPTC/5th SEM/E
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
gb_classifier = GradientBoostingClassifier(n_estimators=100, learning_rate=

gb_classifier.fit(X_train, y_train)

y_pred = gb_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.00
Classification Report:
              precision    recall  f1-score   support

          21       0.00      0.00      0.00       1.0
          25       0.00      0.00      0.00       0.0
          30       0.00      0.00      0.00       1.0
          62       0.00      0.00      0.00       1.0
          67       0.00      0.00      0.00       0.0
          69       0.00      0.00      0.00       0.0
          75       0.00      0.00      0.00       0.0
          76       0.00      0.00      0.00       1.0
          81       0.00      0.00      0.00       1.0

    accuracy                           0.00       5.0
   macro avg       0.00      0.00      0.00       5.0
weighted avg       0.00      0.00      0.00       5.0
```

```
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Precision and F-score are ill-defined a
nd being set to 0.0 in labels with no predicted samples. Use `zero_divisio
n` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\bhava\anaconda3\lib\site-packages\sklearn\metrics\_classificatio
n.py:1318: UndefinedMetricWarning: Recall and F-score are ill-defined and
being set to 0.0 in labels with no true samples. Use `zero_division` param
eter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
In [ ]:
```

```
In [ ]:
```