

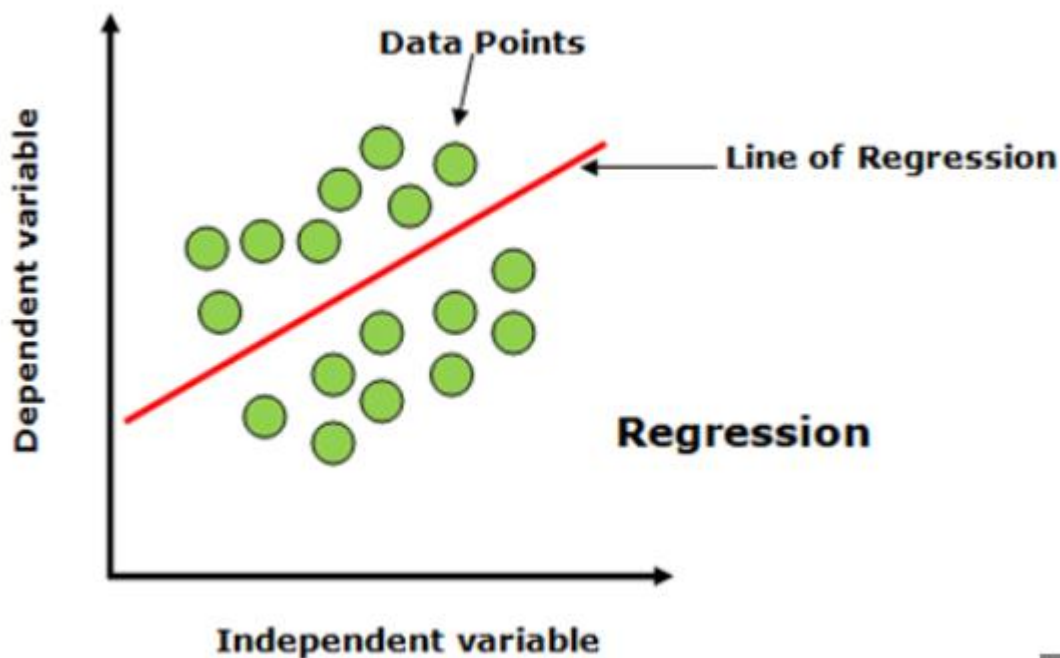
LAB No. 2

Implementation of Regression Analysis using python

Regression techniques are used to analyze relationships between variables and make predictions. Multiple Linear Regression predicts a continuous output using multiple input variables, while Logistic Regression is used for classification problems with binary outcomes. In this lab, students will apply both methods using Python to understand data modeling, prediction, and basic performance evaluation.

Introduction

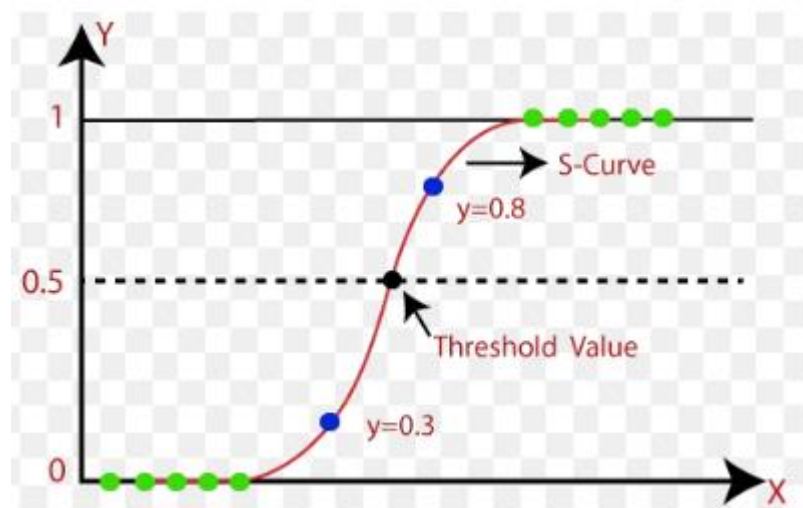
Regression analysis is a fundamental technique in machine learning and data science used to understand the relationship between variables and to make predictions. In this lab, students will explore **Multiple Linear Regression** and **Logistic Regression** using Python.



Multiple Linear Regression is used when a dependent (output) variable is continuous and depends on two or more independent (input) variables. It helps in modeling and analyzing how changes in multiple features affect a numerical

outcome, such as predicting house prices based on area, number of rooms, and location.

Logistic Regression is used for classification problems where the dependent variable is categorical, usually binary (such as Yes/No, True/False, or 0/1). It estimates the probability that an input belongs to a particular class and is widely used in applications such as disease prediction, spam detection, and customer churn analysis.



Solved Examples

Example 1

A dataset contains information about students' **study hours** and **attendance percentage**. Predict the **final marks** using Multiple Linear Regression.

Solution:

```
# Import required libraries
import pandas as pd
from sklearn.linear_model import LinearRegression

# Create dataset
```

```

data = {
    'Study_Hours': [2, 4, 6, 8, 10],
    'Attendance': [60, 70, 80, 90, 95],
    'Marks': [50, 60, 70, 85, 90]
}

df = pd.DataFrame(data)

# Independent and dependent variables
X = df[['Study_Hours', 'Attendance']]
y = df['Marks']

# Create and train model
model = LinearRegression()
model.fit(X, y)

# Prediction
prediction = model.predict([[7, 85]])

print("Predicted Marks:", prediction[0])

```

Example 2

Predict the **house price** based on **area (sq ft)** and **number of bedrooms** using Multiple Linear Regression.

Solution:

```

import pandas as pd
from sklearn.linear_model import LinearRegression

# Dataset
data = {
    'Area': [800, 1000, 1200, 1500, 1800],
    'Bedrooms': [1, 2, 2, 3, 3],
    'Price': [50000, 65000, 75000, 90000, 110000]
}

df = pd.DataFrame(data)

```

```
X = df[['Area', 'Bedrooms']]
y = df['Price']

model = LinearRegression()
model.fit(X, y)

# Predict price
predicted_price = model.predict([[1400, 3]])

print("Predicted House Price:", predicted_price[0])
```

Example 3

A dataset contains **study hours** and **attendance** information. Predict whether a student will **pass (1)** or **fail (0)** using Logistic Regression.

Solution:

```
import pandas as pd
from sklearn.linear_model import LogisticRegression

# Dataset
data = {
    'Study_Hours': [1, 2, 4, 6, 8, 10],
    'Attendance': [50, 55, 65, 75, 85, 95],
    'Result': [0, 0, 0, 1, 1, 1] # 0 = Fail, 1 = Pass
}

df = pd.DataFrame(data)

X = df[['Study_Hours', 'Attendance']]
y = df['Result']

# Create and train model
model = LogisticRegression()
model.fit(X, y)

# Predict pass/fail
result = model.predict([[5, 70]])
```

```
print("Predicted Result (1=Pass, 0=Fail):", result[0])
```

LAB Assignment No. 2

Lab Practice Questions

Q1. Multiple Linear Regression – House Price Prediction

A dataset contains:

- Size (sqft),
- Number of Bedrooms,
- Age of House (years)

and the target variable is **House Price**.

Task:

1. Fit a **multiple linear regression model**.
2. Predict the price of a house with: **Size = 2000 sqft, Bedrooms = 3, Age = 10 years**.
3. Print coefficients and interpret them.

```
df = pd.read_csv("task1.csv")
print(df.head())
```

	Size (sqft)	No. of Bedrooms	Age of House (Years)	Price
0	1901	3	3	4000000
1	1950	4	6	4500000
2	2243	5	4	5200000
3	1245	1	2	3200000
4	1874	2	1	3500000

```
X = df[['Size (sqft)', 'No. of Bedrooms', 'Age of House (Years)']]
y = df['Price']
```

```
new_house = np.array([[2000, 3, 10]])  
predicted_price = model.predict(new_house)  
print("Predicted House Price:", predicted_price[0])
```

```
Predicted House Price: 3293916.3664325615
```

Q2. Multiple Linear Regression – Student Performance

Dataset columns:

- Hours Study,
 - Hours Sleep,
 - Attendance (%),
- Target: **Marks in Exam**

Task:

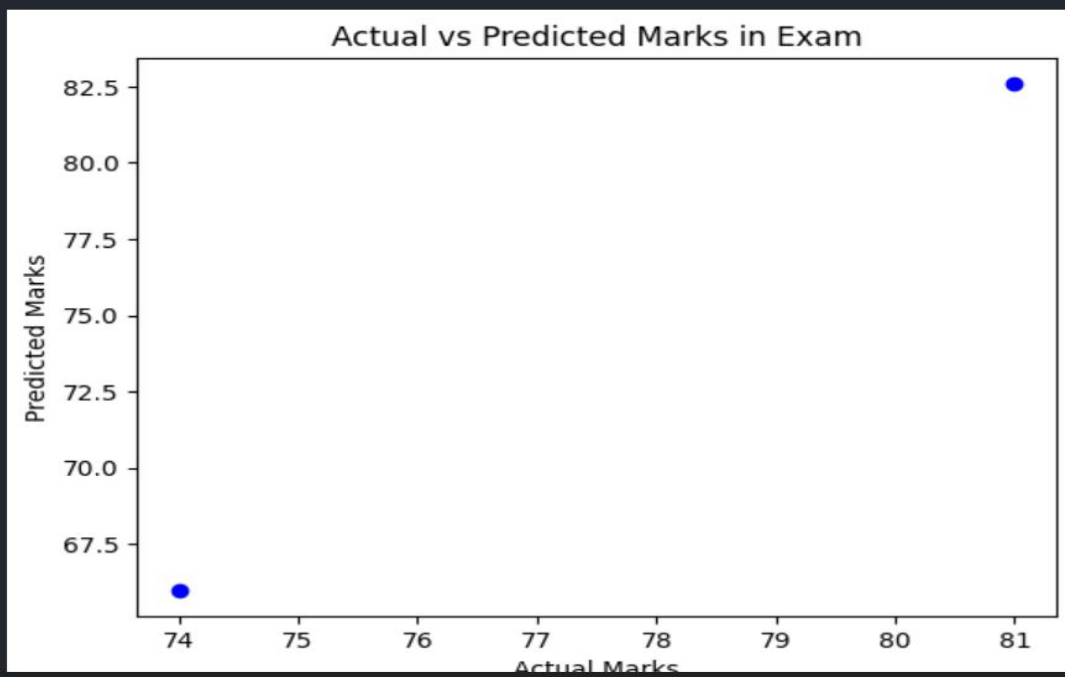
1. Train a regression model.
2. Plot actual vs predicted marks.
3. Compute **R^2 score** and **Mean Squared Error (MSE)**.

```
df = pd.read_csv("task2.csv")
print(df.head())
```

	Study Hours	Sleep Hours	Attendance %	Marks
0	7	12	82	64
1	5	13	78	60
2	9	11	84	81
3	6	14	80	73
4	7	10	82	78

```
X = df[['Study Hours', 'Sleep Hours', 'Attendance %']]
y = df['Marks']
```

```
plt.title("Actual vs Predicted Marks in Exam")
plt.show()
```



```
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print("R2 Score:", r2)
print("Mean Squared Error:", mse)
```

```
R2 Score: -1.730753953626862
Mean Squared Error: 33.45173593192906
```

Q3. Logistic Regression – Pass/Fail Classification

Dataset columns:

- Hours Study
 - Hours Sleep
- Target: Pass (1) / Fail (0)

Task:

1. Fit a **logistic regression classifier**.
2. Predict the probability of passing if a student studies 30 hours and sleeps 6 hours.
3. Plot the **decision boundary** (pass vs fail).

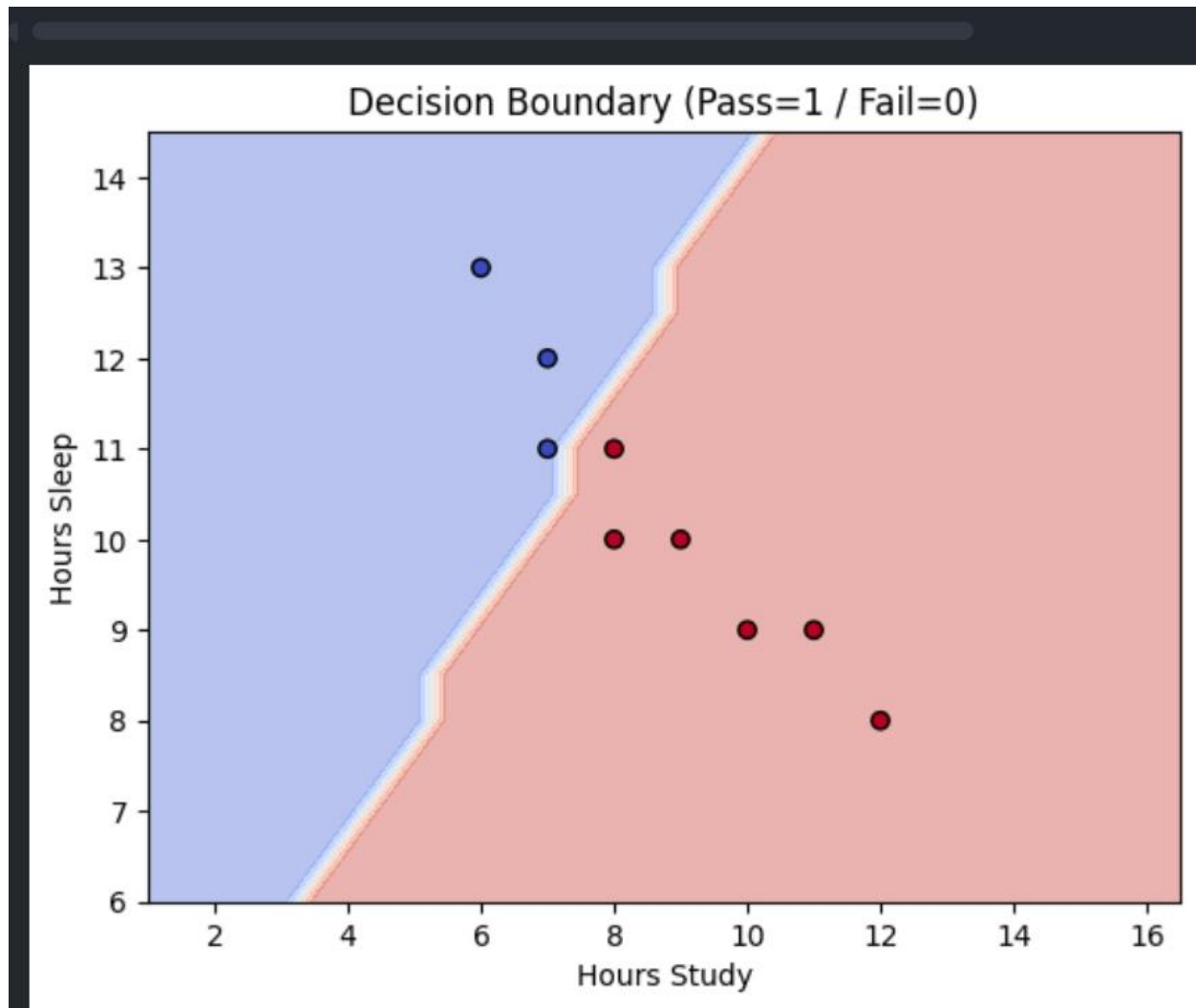
```
df = pd.read_csv("task3.csv")
print(df.head())
```

	Study Hours	Sleep Hours	Pass/Fail
0	0	12	0
1	8	10	1
2	9	10	1
3	10	9	1
4	12	8	1

```
X = df[['Study Hours', 'Sleep Hours']]
y = df['Pass/Fail']
```

```
new_student = np.array([[30, 6]])
prob = model.predict_proba(new_student)
print("Probability of Passing:", prob[0][1])
```

```
Probability of Passing: 0.9999999999829112
```



Q4. Logistic Regression – Diabetes Prediction (Binary Classification)

Use a small dataset with:

- BMI,
 - Age,
 - Glucose Level
- Target: **Diabetic (1) or Not (0)**

Task:

1. Fit logistic regression.

2. Find accuracy, precision, recall.
3. Predict whether a patient (BMI=28, Age=45, Glucose=150) is diabetic.

```
df = pd.read_csv("task4.csv")  
print(df.head())
```

	BMI	Age	Glucose Level	Diabetic(0/1)
0	0	23	91	0
1	21	31	97	0
2	19	27	81	0
3	26	29	127	1
4	25	34	130	1

```
X = df[['BMI', 'Age', 'Glucose Level']]  
y = df['Diabetic(0/1)']
```

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size=0.3, random_state=0)
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
```

```
new_data = [[28, 45, 150]]
prediction = model.predict(new_data)
probability = model.predict_proba(new_data)

print("Predicted Class (1=Diabetic, 0=Not):", prediction[0])
print("Probability [Not, Diabetic]:", probability)
```

```
Predicted Class (1=Diabetic, 0=Not): 1
```

Q5. Comparison – Linear vs Logistic Regression

Dataset columns:

- Hours Study,
- Exam Score,
- Pass/Fail

Task:

1. Use **Linear Regression** to predict exam scores.
2. Use **Logistic Regression** to predict pass/fail.
3. Compare results — explain why linear regression is unsuitable for classification.

```
mse = mean_squared_error(y_lin_test, y_lin_pred)
print("\n--- Linear Regression Results (Exam Score Prediction) ---")
print("MSE:", mse)

print("Coefficient:", lin_model.coef_[0])
print("Intercept:", lin_model.intercept_)
```

```
--- Linear Regression Results (Exam Score Prediction) ---
MSE: 95.51342586109664
Coefficient: 8.32086883349111
Intercept: 6.5723677290417655
```

```
y_log_pred = log_model.predict(X_log_test)

# Evaluate
acc = accuracy_score(y_log_test, y_log_pred)
print("\n--- Logistic Regression Results (Pass/Fail Prediction) ---")
print("Accuracy:", acc)
print("Coefficient:", log_model.coef_[0][0])
print("Intercept:", log_model.intercept_[0])
```

```
--- Logistic Regression Results (Pass/Fail Prediction) ---
Accuracy: 0.9090909090909091
Coefficient: 2.1360055245275666
Intercept: -16.15433988336347
```

Implementation Notes for Students:

- Use pandas to load small CSVs (or create toy datasets directly in code).
- Use `sklearn.linear_model.LinearRegression` and `LogisticRegression`.
- Plot with matplotlib.
- Interpret coefficients in both models.