LAB Assignment No 2

Topic: Multiple and Logistic Linear Regression

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:

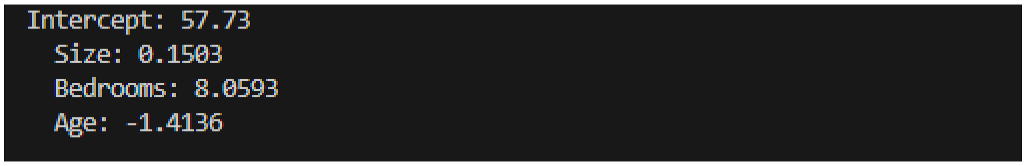
1. Fit a **multiple linear regression model**.
2. Predict the price of a house with: Size = 2000 sqft, Bedrooms = 3, Age = 10 years.
3. Print coefficients and interpret them.

Code:

```
import pandas as pd

from sklearn.linear_model import LinearRegression
import numpy as np
data = pd.read_csv('house_features_data.csv')
data['Price'] = 50 + (data['Size_sqft'] * 0.15) + (data['Bedrooms'] * 10) - (data['Age'] * 1.2) +
np.random.randint(-20, 20, len(data))

X = data[['Size_sqft', 'Bedrooms', 'Age']]
y = data['Price']

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

```
new_house = np.array([[2000, 3, 10]])
predicted_price = model.predict(new_house)[0]

print(f"Intercept: {model.intercept_:.2f}")
print(f"  Size: {model.coef_[0]:.4f}")
print(f"  Bedrooms: {model.coef_[1]:.4f}")
print(f"  Age: {model.coef_[2]:.4f}")
print(f"\nPredicted price for a 2000 sqft, 3-bed, 10-year-old house: ${predicted_price *
1000:.2f}")
```

output:



```
Intercept: 57.73
  Size: 0.1503
  Bedrooms: 8.0593
  Age: -1.4136
```

Q2. Multiple Linear Regression – Student Performance

Dataset columns:

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

👉 Task:

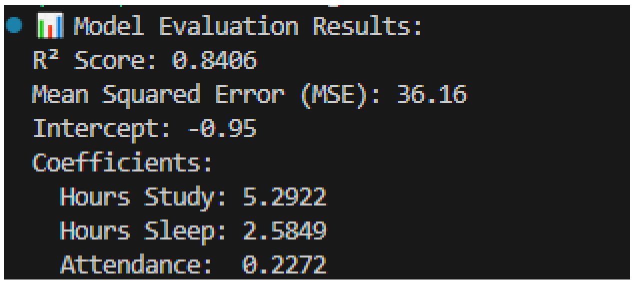
1. Train a regression model.
2. Plot actual vs predicted marks.
3. Compute **R^2 score** and **Mean Squared Error (MSE)**.

Code:

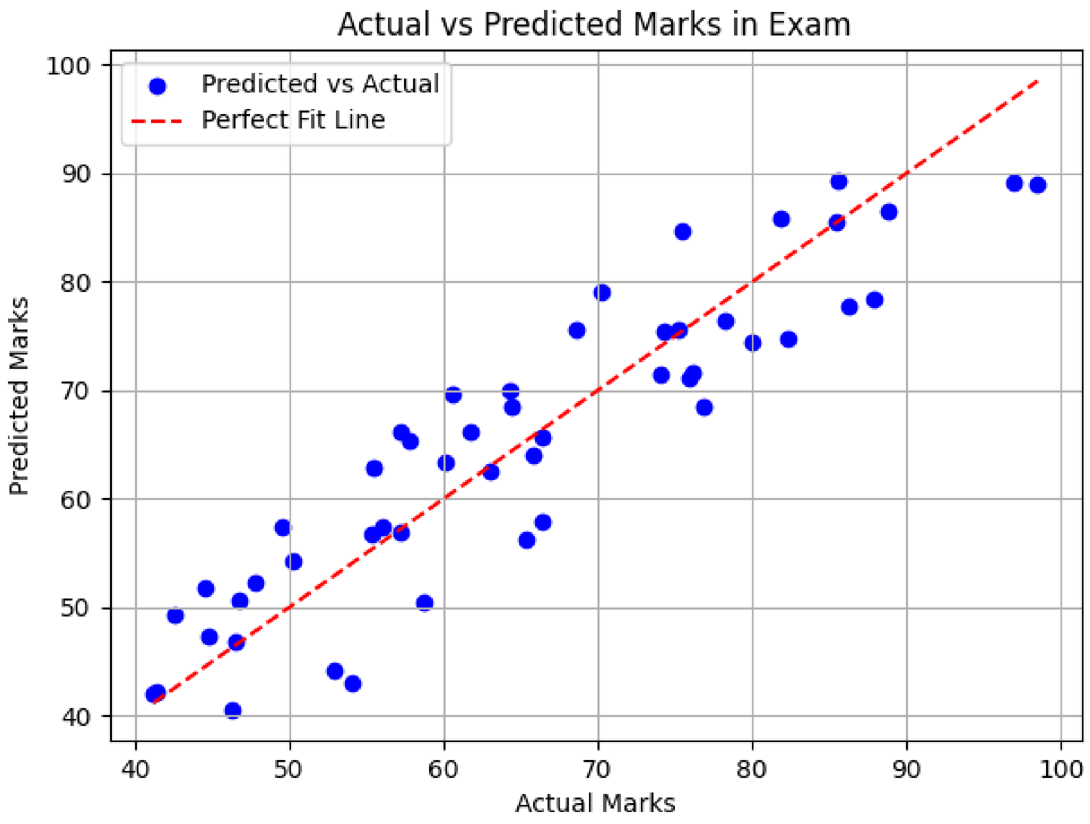
```
    import pandas as pd
    import numpy as np
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import r2_score, mean_squared_error
    import matplotlib.pyplot as plt
    data = pd.read_csv("student_performance_data.csv")
X = data[["Hours_Study", "Hours_Sleep", "Attendance_%"]]
```

```
y = data["Marks_in_Exam"]

# Step 3: Train model
model = LinearRegression()
model.fit(X, y)

# Step 4: Predict
y_pred = model.predict(X)

# Step 5: Model evaluation
r2 = r2_score(y, y_pred)
mse = mean_squared_error(y, y_pred)

print("📊 Model Evaluation Results:")
print(f"R² Score: {r2:.4f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Intercept: {model.intercept_:.2f}")
print(f"Coefficients:")
print(f"  Hours Study: {model.coef_[0]:.4f}")
print(f"  Hours Sleep: {model.coef_[1]:.4f}")
print(f"  Attendance:  {model.coef_[2]:.4f}")

# Step 6: Plot actual vs predicted marks
plt.figure(figsize=(7,5))
plt.scatter(y, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', label='Perfect Fit
Line')
plt.xlabel('Actual Marks')
plt.ylabel('Predicted Marks')
plt.title('Actual vs Predicted Marks in Exam')
plt.legend()
plt.grid(True)
plt.show()
```

output



```
📊 Model Evaluation Results:
R² Score: 0.8406
Mean Squared Error (MSE): 36.16
Intercept: -0.95
Coefficients:
  Hours Study: 5.2922
  Hours Sleep: 2.5849
  Attendance:  0.2272
```



Q3. Logistic Regression – Pass/Fail Classification

Dataset columns:

- Hours Study

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

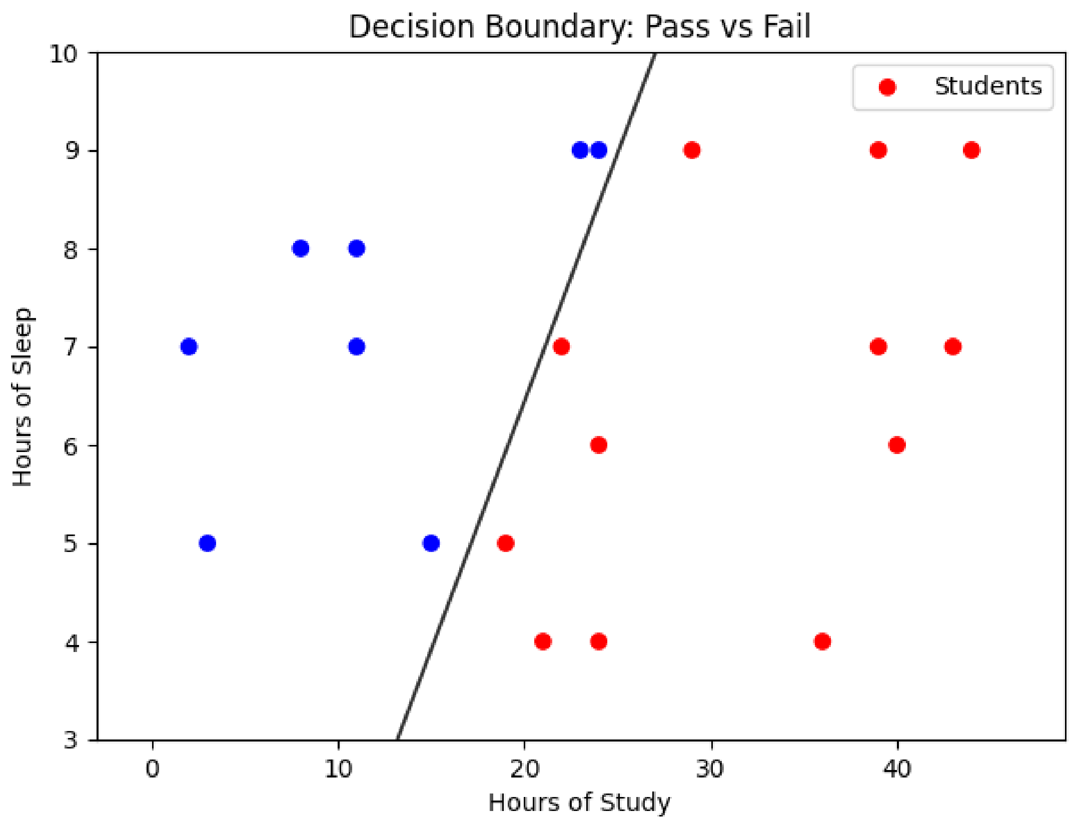
👉 Task:

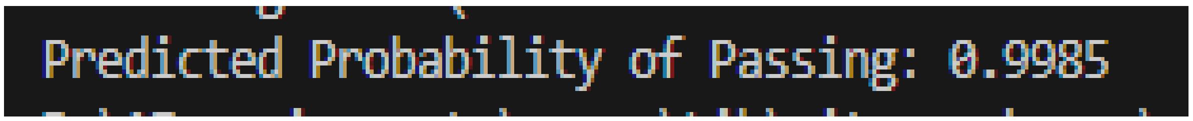
1. Fit a **logistic regression classifier**.
2. Predict the probability of passing if a student studies 30 hours and sleeps 6 hours.
3. Plot the **decision boundary** (pass vs fail).

Code:

```
 import pandas as pd
 import numpy as np
 from sklearn.linear_model import LogisticRegression
 import matplotlib.pyplot as plt
 data = pd.read_excel("student_pass_fail_20.xlsx")
 X = data[["Hours_Study", "Hours_Sleep"]]
 y = data["Pass"]
model = LogisticRegression()
 model.fit(X, y)
 new_student = np.array([[30, 6]])
 prob_pass = model.predict_proba(new_student)[0][1]
 print(f"Predicted Probability of Passing: {prob_pass:.4f}")
 plt.figure(figsize=(7,5))
 plt.scatter(data["Hours_Study"], data["Hours_Sleep"], c=data["Pass"], cmap="bwr",
 label="Students")
x_min, x_max = X["Hours_Study"].min()-5, X["Hours_Study"].max()+5
 y_min, y_max = X["Hours_Sleep"].min()-1, X["Hours_Sleep"].max()+1
 xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 200))
  grid = np.c_[xx.ravel(), yy.ravel()]
  probs = model.predict_proba(grid)[:, 1].reshape(xx.shape)
  # Contour plot (boundary at probability = 0.5)
  plt.contour(xx, yy, probs, levels=[0.5], cmap="Greys", vmin=0, vmax=0.6)
  plt.xlabel("Hours of Study")
  plt.ylabel("Hours of Sleep")
  plt.title("Decision Boundary: Pass vs Fail")
  plt.legend()
  plt.show()
```

output:





```
Predicted Probability of Passing: 0.9985
```

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.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
data = pd.read_excel("student_pass_fail_20.xlsx")
X = data[["Hours_Study", "Hours_Sleep"]]
y = data["Pass"]
model = LogisticRegression()
model.fit(X, y)
new_student = np.array([[30, 6]])
prob_pass = model.predict_proba(new_student)[0][1]
print(f"Predicted Probability of Passing: {prob_pass:.4f}")
plt.figure(figsize=(7,5))
plt.scatter(data["Hours_Study"], data["Hours_Sleep"], c=data["Pass"], cmap="bwr",
label="Students")
x_min, x_max = X["Hours_Study"].min()-5, X["Hours_Study"].max()+5
y_min, y_max = X["Hours_Sleep"].min()-1, X["Hours_Sleep"].max()+1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 200))
 grid = np.c_[xx.ravel(), yy.ravel()]
probs = model.predict_proba(grid)[:, 1].reshape(xx.shape)
# Contour plot (boundary at probability = 0.5)
plt.contour(xx, yy, probs, levels=[0.5], cmap="Greys", vmin=0, vmax=0.6)
plt.xlabel("Hours of Study")
plt.ylabel("Hours of Sleep")
plt.title("Decision Boundary: Pass vs Fail")
plt.legend()
plt.show()
```

output:

```
📊 Model Evaluation Results:
Accuracy:  1.0000
Precision: 1.0000
Recall:    1.0000
🧍 Patient Info → BMI=28, Age=45, Glucose=150
Predicted: Diabetic
Probability of being diabetic: 0.9998
```

Q5. Comparison – Linear vs Logistic Regression

Dataset columns:

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

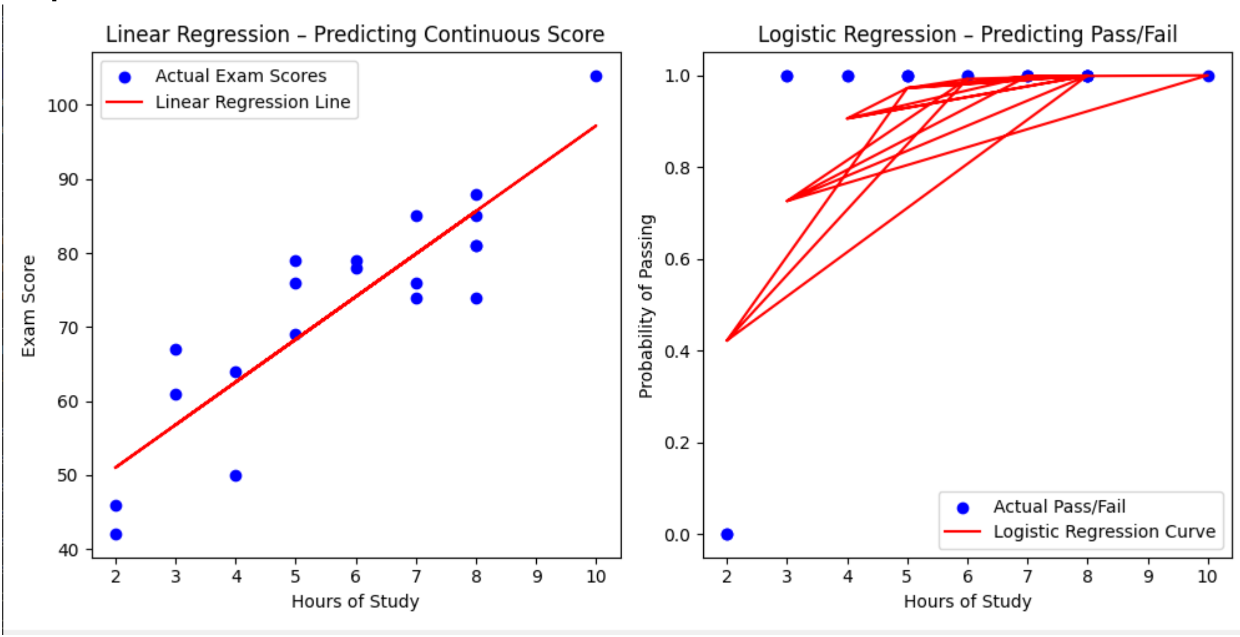
👉 Task:

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.

Code:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression, LogisticRegression
import matplotlib.pyplot as plt
data = pd.read_excel("linear_vs_logistic.xlsx")
X = data[["Hours_Study"]]
y_score = data["Exam_Score"]
y_pass = data["Pass"]
lin_model = LinearRegression()
lin_model.fit(X, y_score)
score_pred = lin_model.predict(X)
log_model = LogisticRegression()
log_model.fit(X, y_pass)
pass_pred_prob = log_model.predict_proba(X)[:, 1]
pass_pred = log_model.predict(X)
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.scatter(X, y_score, color='blue', label='Actual Exam Scores')
plt.plot(X, score_pred, color='red', label='Linear Regression Line')
plt.xlabel("Hours of Study")
plt.ylabel("Exam Score")
plt.title("Linear Regression – Predicting Continuous Score")
plt.legend()
plt.subplot(1,2,2)
plt.scatter(X, y_pass, color='blue', label='Actual Pass/Fail')
plt.plot(X, pass_pred_prob, color='red', label='Logistic Regression Curve')
plt.xlabel("Hours of Study")
plt.ylabel("Probability of Passing")
plt.title("Logistic Regression – Predicting Pass/Fail")
plt.legend()
plt.tight_layout()
plt.show()
print("📊 Comparison Results:")
print(f"Linear Regression predicts continuous scores (e.g., {score_pred[:5].round(2)})")
print(f"Logistic Regression predicts probabilities (e.g., {pass_pred_prob[:5].round(2)})")
print("\n❌ Linear regression is unsuitable for classification because:")
print("   - It can predict values below 0 or above 1, which are invalid probabilities.")
print("   - It assumes a linear relationship, while classification requires an S-shaped
(sigmoid) curve.")
print("   - Logistic regression outputs probabilities that can be thresholded (e.g., >0.5 →
Pass).")
```

output:



📌 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.