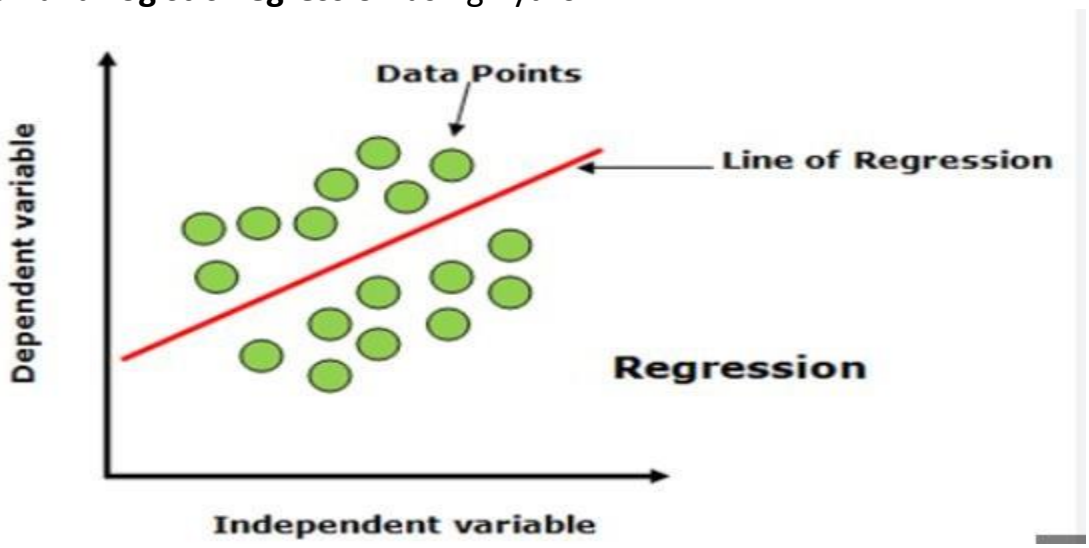
LAB No. 2

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.

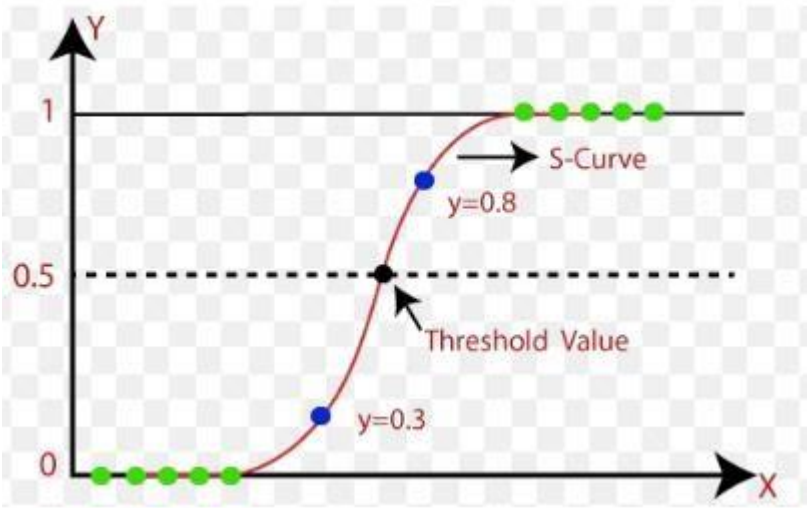
Introduction

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

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)**.

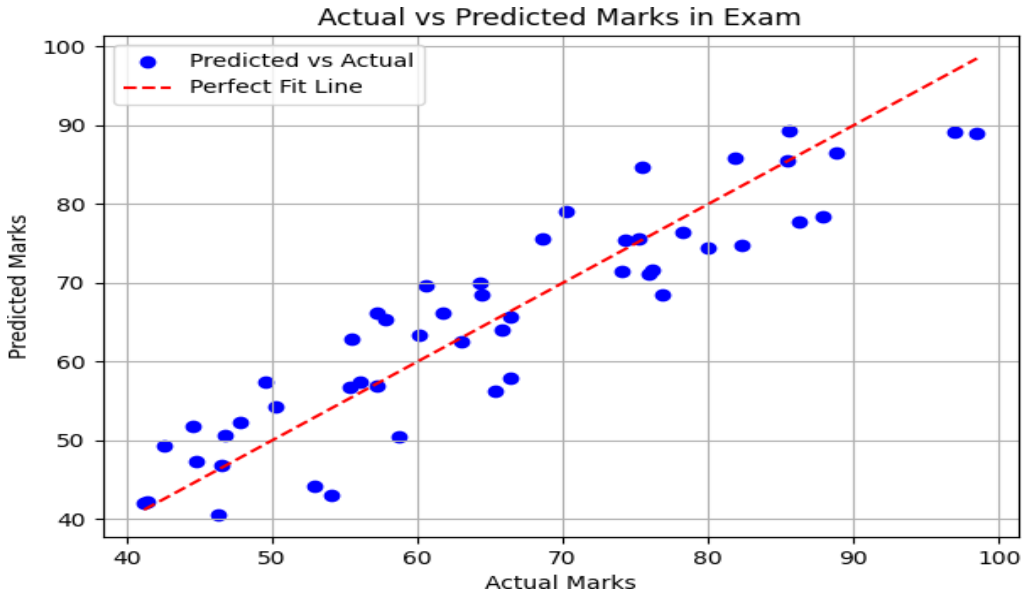
Code:

```
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)

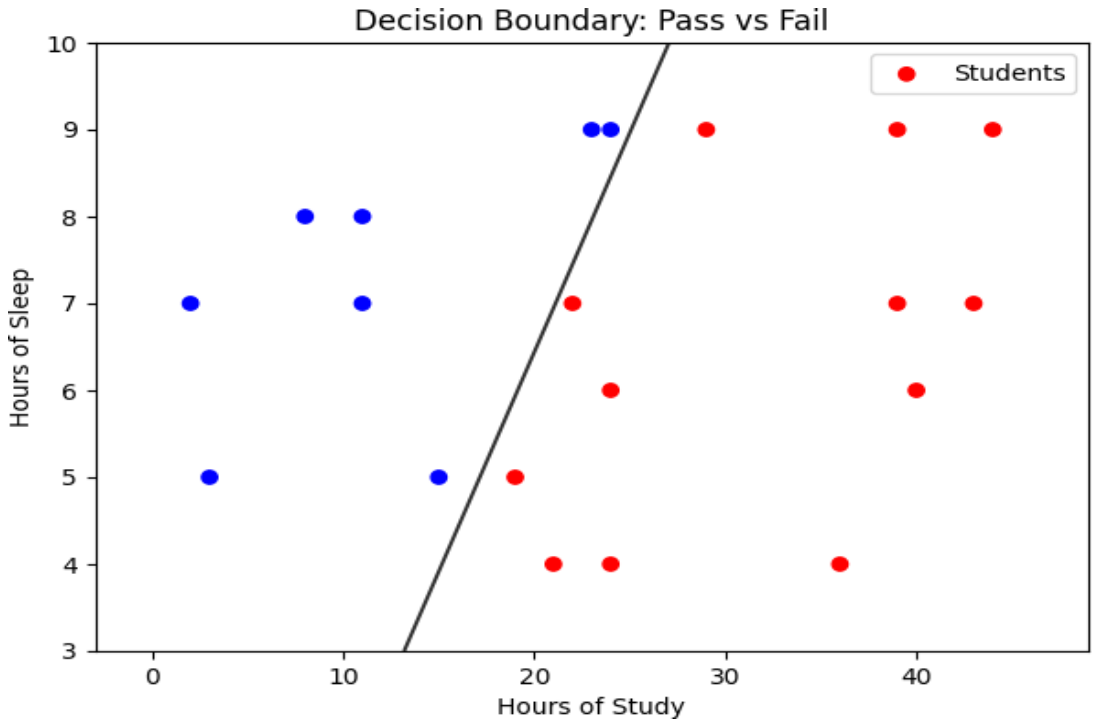
S Task:

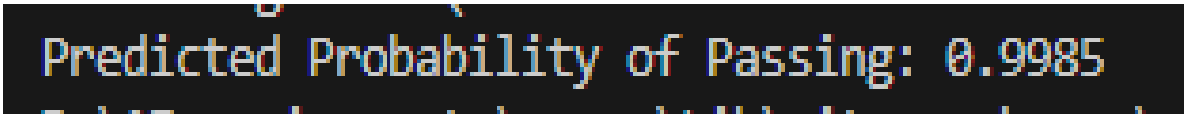
1. Fit a **logistic regression classifier**.
2. Predict the probability of passing if a student studies 30 hours and sleeps 6 hours.
3. Plot the **decision boundary** (pass vs fail).

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:





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.GGG8

Q5. Comparison – Linear vs Logistic Regression

Dataset columns:

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

Task:

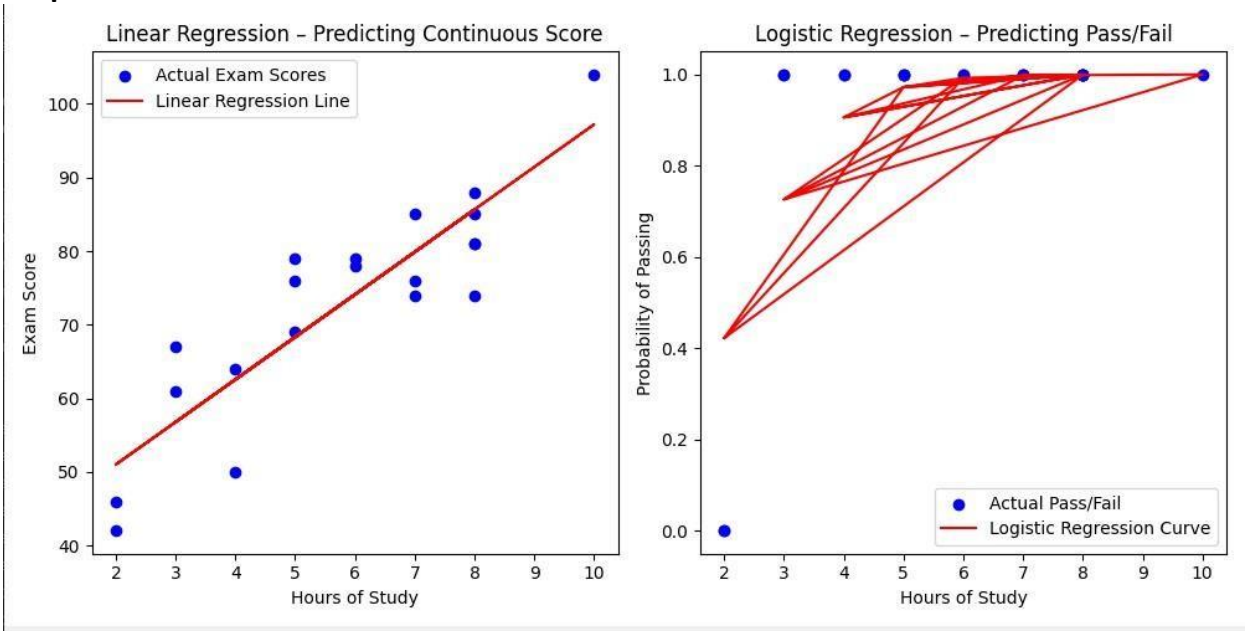
1. Use **Linear Regression** to predict exam scores.
2. Use **Logistic Regression** to predict pass/fail.
3. Compare results — explain why linear regression is unsuitable for classification.

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.

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency 10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines