# 

2:



# Name:Radhika Raut.



1.

import pandas as pd

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.linear\_model import LogisticRegression #Load the dataset

df = pd.read\_csv('titanic.csv')

# Preprocessing steps: Handle missing data, encode categorical variables, and drop unnecessary columns # Handle missing values for 'Age' and 'Fare'

df['Age'] = df['Age'].fillna(df['Age'].median())

df['Fare'] = df['Fare'].fillna(df['Fare'].median()) # You might want to handle missing Fare values as well # Drop unnecessary columns if they exist

columns\_to\_drop = ['Cabin', 'Ticket', 'Name', 'PassengerId']

df.drop(columns=[col for col in columns\_to\_drop if col in df.columns], inplace=True) # Feature scaling

from sklearn.preprocessing import StandardScaler scaler = StandardScaler()

df[['Age', 'Fare']] = scaler.fit\_transform(df[['Age', 'Fare']]) if {'Age', 'Fare'}.issubset(df.columns) else df[['Age', 'Fare']]

2.



and



# Split into features (X) and target (y)

X = df.drop('Survived', axis=1) if 'Survived' in df.columns else None y = df['Survived'] if 'Survived' in df.columns else None

#Split data into training (80%) and testing (20%) sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) print(f"Training samples: {X\_train.shape[0]}, Testing samples: {X\_test.shape[0]}")



3.



#Initialize the Logistic Regression model log\_model = LogisticRegression(max\_iter=500) #Train the model on the training data log\_model.fit(X\_train, y\_train)

print("Model training completed.")

4.



#Predict on the test data

y\_pred = log\_model.predict(X\_test) #Show some predictions print(f"Predicted values: {y\_pred[:10]}") print(f"Actual values: {y\_test[:10].values}")

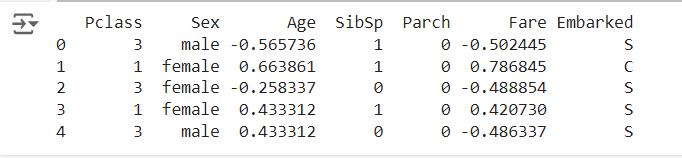


Exp 3:

Name:Radhika Raut



1.



import pandas as pd

from sklearn.preprocessing import StandardScaler # Load the Titanic dataset

df = pd.read\_csv('titanic.csv')

# Preprocessing steps: Handle missing data

# Fill missing values for 'Age' with the median and reassign to avoid chained assignment warning df['Age'] = df['Age'].fillna(df['Age'].median())

# Fill missing values for 'Fare' with the median and reassign to avoid chained assignment warning df['Fare'] = df['Fare'].fillna(df['Fare'].median())

# Drop columns with irrelevant information

# Ensure that only existing columns are dropped columns\_to\_drop = ['Cabin', 'Ticket', 'Name', 'PassengerId']

df.drop(columns=[col for col in columns\_to\_drop if col in df.columns], inplace=True) # Scale numerical features like Age and Fare

scaler = StandardScaler()

df[['Age', 'Fare']] = scaler.fit\_transform(df[['Age', 'Fare']]) # Define features (X) and target (y)

X = df.drop('Survived', axis=1) if 'Survived' in df.columns else None y = df['Survived'] if 'Survived' in df.columns else None

# Print the feature data print(X.head())

2.

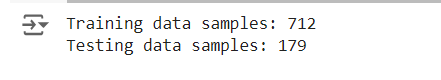


Split Data into Training

and



Testing Sets



from sklearn.model\_selection import train\_test\_split

#Split the data into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) #Print the number of samples in each set

print(f"Training data samples: {X\_train.shape[0]}") print(f"Testing data samples: {X\_test.shape[0]}")

3.

Check the Training

and



Testing Sets

#Check the shape of the training and testing sets print(f"X\_train shape: {X\_train.shape}") print(f"X\_test shape: {X\_test.shape}") print(f"y\_train shape: {y\_train.shape}") print(f"y\_test shape: {y\_test.shape}")

# 

4:

Name:Radhika Raut



# Import necessary libraries import pandas as pd

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, roc\_curve, auc import matplotlib.pyplot as plt

# Step 1: Prepare the Dataset # Load the Titanic dataset

df = pd.read\_csv('titanic.csv')

# Handle missing values for 'Age' and 'Embarked' df['Age'] = df['Age'].fillna(df['Age'].median())

df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

# Drop columns that are not needed

df.drop(columns=['Cabin', 'Ticket', 'Name', 'PassengerId'], inplace=True)

# Convert categorical variables into dummy variables

df = pd.get\_dummies(df, columns=['Sex', 'Embarked'], drop\_first=True)

# Scale numerical variables scaler = StandardScaler()

df[['Age', 'Fare']] = scaler.fit\_transform(df[['Age', 'Fare']])

# Define features (X) and target (y) X = df.drop('Survived', axis=1)

y = df['Survived']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

2.



Train a Classification Model

# Train a Logistic Regression model

model = LogisticRegression(max\_iter=500) model.fit(X\_train, y\_train)



## Make Predictions on the Test Data

3.



# Make predictions on the test data y\_pred = model.predict(X\_test)

4.



Evaluate the Model’s Performance

# Evaluate the accuracy of the model accuracy = accuracy\_score(y\_test, y\_pred) print(f"Accuracy: {accuracy:.4f}")

# Confusion matrix



Name:Radhika Raut



1. Load and Preprocess the Dataset

import pandas as pd

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler from sklearn.linear\_model import LogisticRegression from sklearn.ensemble import RandomForestClassifier from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report, roc\_auc\_score

# Load the Titanic dataset

df = pd.read\_csv('titanic.csv')

# Handle missing values for 'Age' and 'Embarked' df['Age'] = df['Age'].fillna(df['Age'].median())

df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])

# Drop columns that are not needed

df.drop(columns=['Cabin', 'Ticket', 'Name', 'PassengerId'], inplace=True)

# Convert categorical variables into dummy variables

df = pd.get\_dummies(df, columns=['Sex', 'Embarked'], drop\_first=True)

# Scale numerical variables scaler = StandardScaler()

df[['Age', 'Fare']] = scaler.fit\_transform(df[['Age', 'Fare']])

# Define features (X) and target (y) X = df.drop('Survived', axis=1)

y = df['Survived']

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

2.



Train Multiple Models

# Train Logistic Regression

log\_reg = LogisticRegression(max\_iter=500) log\_reg.fit(X\_train, y\_train)

# Train Random Forest

rf = RandomForestClassifier(n\_estimators=100, random\_state=42) rf.fit(X\_train, y\_train)

# Train Support Vector Machine

svm = SVC(probability=True, random\_state=42) svm.fit(X\_train, y\_train)

3.



Make Predictions on

Test

Data

# Predictions for Logistic Regression y\_pred\_log\_reg = log\_reg.predict(X\_test) # Predictions for Random Forest y\_pred\_rf = rf.predict(X\_test)

# Predictions for SVM y\_pred\_svm = svm.predict(X\_test)