1. Select the features you intend to use as independent variables and identify your target (dependent) variable. Split the data into training and testing sets. Create a logistic regression classifier and fit the model.

# Drop irrelevant columns (would likely not affect the result)

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

# Encode categorical variables (Sex, Embarked (txt->int))

df['Sex'] = df['Sex'].map({'male': 1, 'female': 0})

df['Embarked'] = df['Embarked'].map({'C': 1, 'Q': 2, 'S': 3})

# Handle missing values (fill Age & Fare with median, Embarked with mode)

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

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

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

# Define features (independent variables) and target (dependent variable)

X = df.drop(columns=['Survived']) # Target feature droped out from training

y = df['Survived'] # Target variable

# Normalize features

scaler = StandardScaler()

X[['Age', 'Fare', 'Embarked', 'Sex']] = scaler.fit\_transform(X[['Age', 'Fare', 'Embarked', 'Sex']])

# Train-Test Split & Logistic Regression Model (80/20)

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

# Train logistic regression model

log\_reg = LogisticRegression(max\_iter=200)

log\_reg.fit(X\_train, y\_train)

1. Utilize your model to make predictions on the testing data, calculate evaluation metrics such as accuracy and recall, and print the results.

y\_pred = log\_reg.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

print(f"\nModel Performance:")

print(f"Accuracy: {accuracy:.4f}")

print(f"Recall: {recall:.4f}")

1. Display the theta parameter values.

theta\_values = pd.DataFrame(log\_reg.coef\_.flatten(), index=X.columns, columns=['Coefficient'])

print(theta\_values)

1. Create a DataFrame with 3 records (for 3 persons), use your model to make predictions, and print the predicted results using text descriptions such as 'survived' and 'not survived'.

sample\_data = pd.DataFrame({

'Pclass': [1, 3, 2],

'Sex': [0, 1, 0],

'Age': [25, 40, 3],

'SibSp': [0, 1, 0],

'Parch': [0, 2, 1],

'Fare': [71, 7.5, 12],

'Embarked': [1, 3, 2]

})

# Apply same normalization

sample\_data[['Age', 'Fare', 'Embarked', 'Sex']] = scaler.transform(sample\_data[['Age', 'Fare', 'Embarked', 'Sex']])

# Make predictions

sample\_predictions = log\_reg.predict(sample\_data)

# Convert predictions to text format (1/0)

sample\_results = ["Survived" if pred == 1 else "Not Survived" for pred in sample\_predictions]

sample\_data['Prediction'] = sample\_results # Embed prediction to sample data -> result

print("\nPredictions for 3 Sample Passengers:")

print(sample\_data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked', 'Prediction']]) # List other features pred data (post norm)

1. Alter the training/testing split fraction and the maximum iteration of the logistic regression model, observe and print the different outcomes.

split\_variants = [0.1, 0.3, 0.5]

iteration\_variants = [100, 500, 1000]

print("\n🛠 Effect of Different Splits & Iterations on Model Performance:")

for split in split\_variants:

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

for max\_iter in iteration\_variants:

log\_reg = LogisticRegression(max\_iter=max\_iter)

log\_reg.fit(X\_train, y\_train)

y\_pred = log\_reg.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

rec = recall\_score(y\_test, y\_pred)

print(f"Test Size: {split}, Max Iter: {max\_iter} => Accuracy: {acc:.4f}, Recall: {rec:.4f}")

**RESULT:**

Model Performance:

Accuracy: 0.8045

Recall: 0.7297

Theta (Coefficients) for Features:

Coefficient

Pclass -0.941624

Sex -1.284521

Age -0.394204

SibSp -0.299545

Parch -0.122847

Fare 0.123909

Embarked -0.174083

Predictions for 3 Sample Passengers:

Pclass Sex Age SibSp Parch Fare Embarked Prediction

0 1 -1.355574 -0.335187 0 0 0.781141 -1.942303 Survived

1 3 0.737695 0.817561 1 2 -0.497411 0.585954 Not Survived

2 2 -1.355574 -2.025883 0 1 -0.406805 -0.678175 Survived

🛠 Effect of Different Splits & Iterations on Model Performance:

Test Size: 0.1, Max Iter: 100 => Accuracy: 0.8444, Recall: 0.8333

Test Size: 0.1, Max Iter: 500 => Accuracy: 0.8444, Recall: 0.8333

Test Size: 0.1, Max Iter: 1000 => Accuracy: 0.8444, Recall: 0.8333

Test Size: 0.3, Max Iter: 100 => Accuracy: 0.8134, Recall: 0.7297

Test Size: 0.3, Max Iter: 500 => Accuracy: 0.8134, Recall: 0.7297

Test Size: 0.3, Max Iter: 1000 => Accuracy: 0.8134, Recall: 0.7297

Test Size: 0.5, Max Iter: 100 => Accuracy: 0.8027, Recall: 0.7095

Test Size: 0.5, Max Iter: 500 => Accuracy: 0.8027, Recall: 0.7095

Test Size: 0.5, Max Iter: 1000 => Accuracy: 0.8027, Recall: 0.7095