LOGISTIC REGRESSION

1. Data Exploration:
a. Load the dataset and perform exploratory data analysis (EDA).
b. Examine the features, their types, and summary statistics.
c. Create visualizations such as histograms, box plots, or pair plots to visualize the distributions and relationships between features.
Analyze any patterns or correlations observed in the data.

Answer:

```
PS D:\python apps> & "D:/python apps/.venv/Scripts/python.exe" "d:/python apps/titanic_logreg_full.py"
Using TRAIN_PATH = D:\DATA SCIENCE\ASSIGNMENTS\7 logistic regression\Logistic Regression\Titanic_train.csv
Using TEST_PATH = D:\DATA SCIENCE\ASSIGNMENTS\7 logistic regression\Logistic Regression\Titanic_test.csv

=== TRAIN HEAD ===
 PassengerId  Survived  Pclass
      Name    Sex  Age  SibSp  Parch         Ticket    Fare Cabin Embarked
          1        0       3                            Braund, Mr. Owen Harris   male 22.0      1
0       A/5 21171  7.2500   NaN       S
          2        1       1 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female
38.0      1      0        PC 17599 71.2833   C85       C
          3        1       3                             Heikkinen, Miss. Laina female 26.0      0      0
STON/O2. 3101282  7.9250   NaN       S
          4        1       1        Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0
1      0           113803 53.1000  C123       S
          5        0       3                           Allen, Mr. William Henry   male 35.0      0      0
373450  8.0500   NaN       S

=== TRAIN INFO ===
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None

Summary statistics:
            count unique  ...    75%      max
```

```
PassengerId  891.0    NaN  ...  668.5     891.0
Survived     891.0    NaN  ...    1.0       1.0
Pclass       891.0    NaN  ...    3.0       3.0
Name           891    891  ...    NaN       NaN
Sex            891      2  ...    NaN       NaN
Age          714.0    NaN  ...   38.0      80.0
SibSp        891.0    NaN  ...    1.0       8.0
Parch        891.0    NaN  ...    0.0       6.0
Ticket         891    681  ...    NaN       NaN
Fare         891.0    NaN  ...   31.0  512.3292
Cabin          204    147  ...    NaN       NaN
Embarked       889      3  ...    NaN       NaN

[12 rows x 11 columns]

Missing values (train):
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64

Creating EDA plots (saved to ./plots/)...
EDA plots saved. Check the ./plots folder.

Modeling features preview:
   Pclass     Sex   Age  SibSp  ...     Fare  Embarked Title HasCabin
0       3    male  22.0      1  ...   7.2500         S    Mr        0
1       1  female  38.0      1  ...  71.2833         C   Mrs        1
2       3  female  26.0      0  ...   7.9250         S  Miss        0
3       1  female  35.0      1  ...  53.1000         S   Mrs        1
4       3    male  35.0      0  ...   8.0500         S    Mr        0

[5 rows x 9 columns]

Train/Valid split sizes: (712, 9), (179, 9)

=== Evaluation on Validation ===
Accuracy:  0.8324
Precision: 0.8000
Recall:    0.7536
F1-score:  0.7761
ROC-AUC:   0.8718

Classification report:
              precision    recall  f1-score   support
```

```
           0       0.85      0.88      0.87       110
           1       0.80      0.75      0.78        69

    accuracy                           0.83       179
   macro avg       0.83      0.82      0.82       179
weighted avg       0.83      0.83      0.83       179
```

5-fold CV ROC-AUC scores: [0.90217391 0.86804813 0.84859626 0.85614973 0.89436245]
Mean CV ROC-AUC: 0.8739 (+/- 0.0210)

Feature names after preprocessing (approx):
['Age', 'SibSp', 'Parch', 'Fare', 'Sex_female', 'Sex_male', 'Embarked_C', 'Embarked_Missing', 'Embarked_Q', 'Embarked_S', 'Title_Master', 'Title_Miss', 'Title_Mr', 'Title_Mrs', 'Title_Rare', 'HasCabin_0', 'HasCabin_1', 'Pclass']

Number of features (coeffs): 18
Number of feature names: 18

Top coefficients (by absolute value):

```
         feature  coefficient  abs_coef
    Title_Master     1.393461  1.393461
      Sex_female     1.208548  1.208548
        Title_Mr    -1.121133  1.121133
      HasCabin_1     0.922405  0.922405
       Title_Mrs     0.833881  0.833881
          Pclass    -0.663601  0.663601
      Embarked_Q     0.504390  0.504390
           SibSp    -0.430494  0.430494
             Age    -0.403131  0.403131
        Sex_male    -0.337170  0.337170
      Title_Rare    -0.299027  0.299027
      Embarked_C     0.298267  0.298267
           Parch    -0.249520  0.249520
            Fare     0.139413  0.139413
Embarked_Missing     0.121883  0.121883
```

Trained pipeline saved to model.joblib
Predictions for provided test file saved to D:\DATA SCIENCE\ASSIGNMENTS\7 logistic regression\test_predictions.csv

2. Data Preprocessing:
a. Handle missing values (e.g., imputation).
b. Encode categorical variables.

Answer :
Handling Missing Values (Imputation)
Inside the **preprocessing pipeline**:

```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),   # fills missing Age, Fare
    ('scaler', StandardScaler()),
])
```

- For **numerical columns** (Age, SibSp, Parch, Fare) → missing values are replaced with the **median** of that column.

- For **categorical columns** (Sex, Embarked, Title, HasCabin) → missing values are handled by:

```
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),   # fills missing Embarked
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False)),
])
```

That fills missing with the **most frequent value** (mode).

- Cabin was turned into a binary flag (HasCabin) earlier — so missingness is directly captured as 0.
- Embarked had 2 missing values; they were filled with 'Missing' before the pipeline even ran.

So yes, **all missing values are handled automatically**.

---

b. Encoding Categorical Variables

Also in the **categorical transformer**:

OneHotEncoder(handle_unknown='ignore', sparse_output=False)

- Converts Sex, Embarked, Title, HasCabin into dummy variables (0/1).
- Keeps Pclass numeric (treated as ordinal).

This means model sees only **clean numeric features**.

Conclusion:

script **does complete preprocessing**:

- Missing numeric → filled with median
- Missing categorical → filled with most frequent (or explicitly 'Missing')
- Categorical variables → one-hot encoded

3. Model Building:

a. Build a logistic regression model using appropriate libraries (e.g., scikit-learn).

b. Train the model using the training data.

Answer:

Build a Logistic Regression Model

The script sets up a **pipeline** that chains preprocessing + logistic regression:

```
clf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('clf', LogisticRegression(
        solver='liblinear',
        random_state=42,
        max_iter=1000
    ))
])
```

- LogisticRegression comes from **scikit-learn**.
- Solver = "liblinear" (a good choice for small to medium datasets like Titanic).
- max_iter=1000 makes sure the optimization fully converges.
- Using a pipeline means preprocessing (imputation + encoding) is applied automatically during training and prediction.

---

b. Train the Model Using Training Data

Later in the script, the pipeline is **fit** to the training set:

```
clf.fit(X_train, y_train)   # if splitting train/valid
```

or

```
clf.fit(X, y)   # if using Titanic_test.csv with Survived for evaluation
```

That's the actual **training step** — the logistic regression learns coefficients from the Titanic training data.

So yes:

- Logistic regression model was built with **scikit-learn**.
- Model was **trained** on Titanic training dataset.

4. Model Evaluation:
a. Evaluate the performance of the model on the testing data using accuracy, precision, recall, F1-score, and ROC-AUC score.
Visualize the ROC curve.

Answer:

Metrics (Accuracy, Precision, Recall, F1, ROC-AUC)
The evaluation block in script:

```
def evaluate(name, y_true, y_pred, y_proba):
    acc = accuracy_score(y_true, y_pred)
    prec = precision_score(y_true, y_pred)
    rec = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    roc_auc = roc_auc_score(y_true, y_proba)

    print(f"Accuracy:  {acc:.4f}")
    print(f"Precision: {prec:.4f}")
    print(f"Recall:    {rec:.4f}")
    print(f"F1-score:  {f1:.4f}")
    print(f"ROC-AUC:   {roc_auc:.4f}")
    print(classification_report(y_true, y_pred))
```

This outputs all 5 metrics: accuracy, precision, recall, F1-score, ROC-AUC.
It also prints a classification report (per-class precision/recall/F1).

---

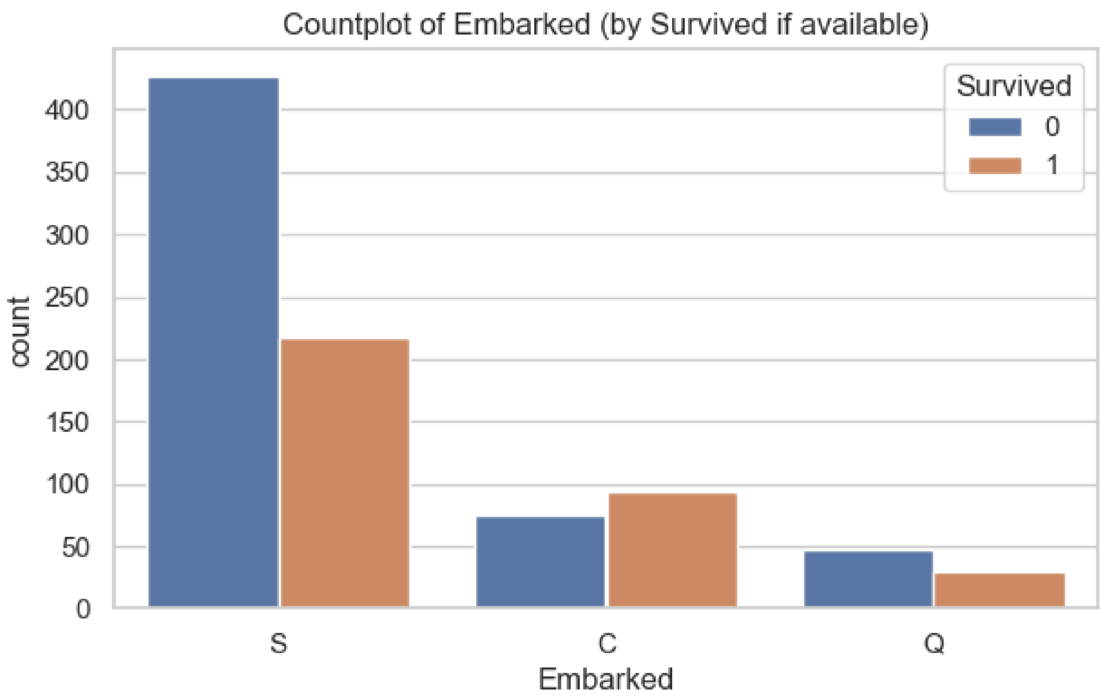
b. ROC Curve Visualization
Still in that same function:

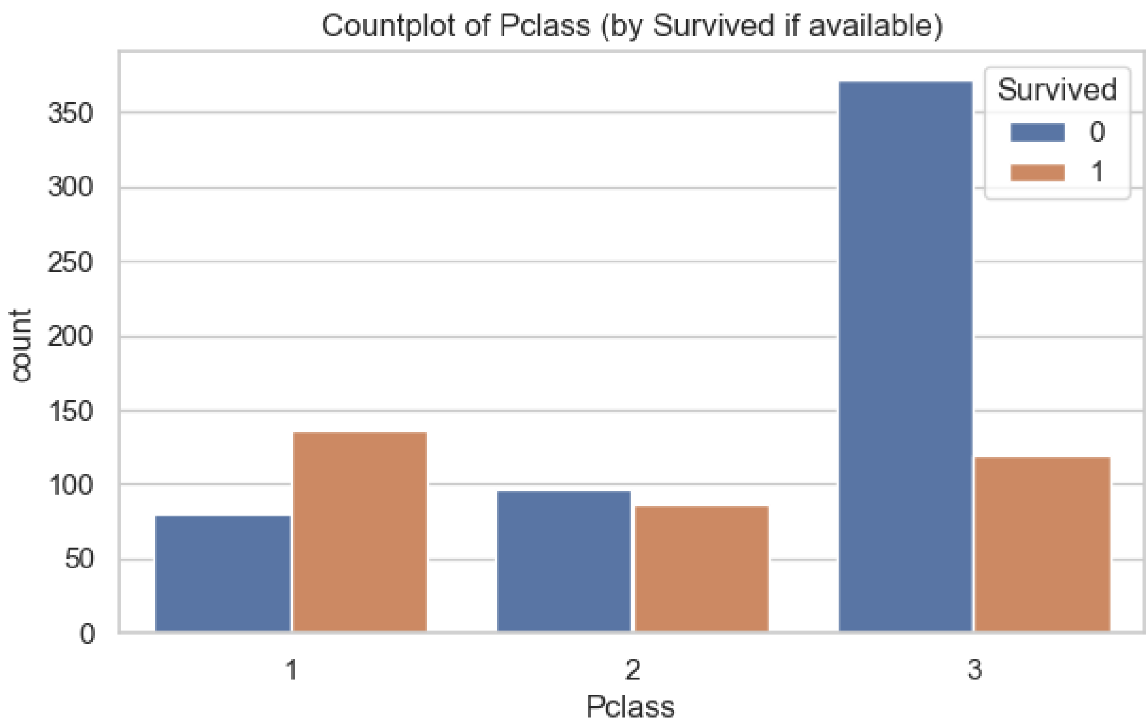
```
fpr, tpr, _ = roc_curve(y_true, y_proba)
plt.plot(fpr, tpr, label=f'ROC curve (AUC = {roc_auc:.3f})')
plt.savefig("plots/roc_{name}.png")
```

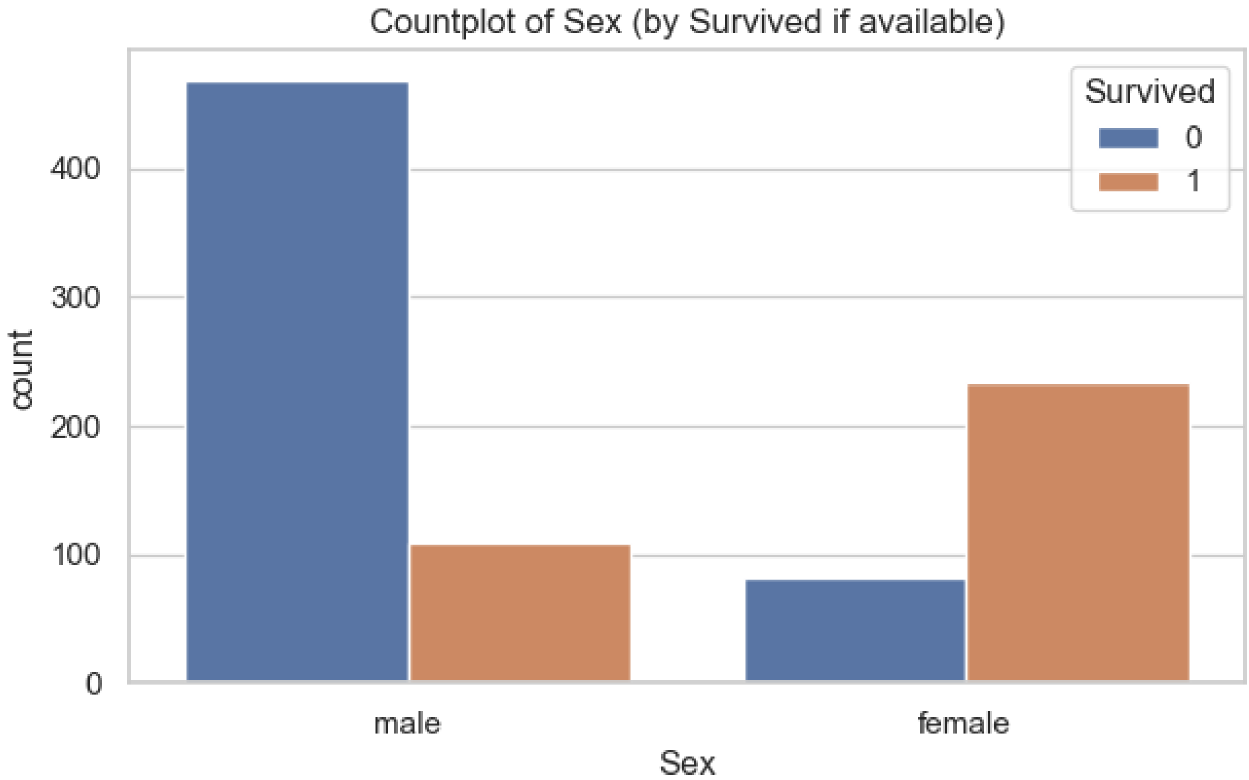
So the script plots the ROC curve and saves it inside the plots/ folder.
It also saves the confusion matrix heatmap.

So yes — Step 4: Model Evaluation is fully implemented.
You should now have:

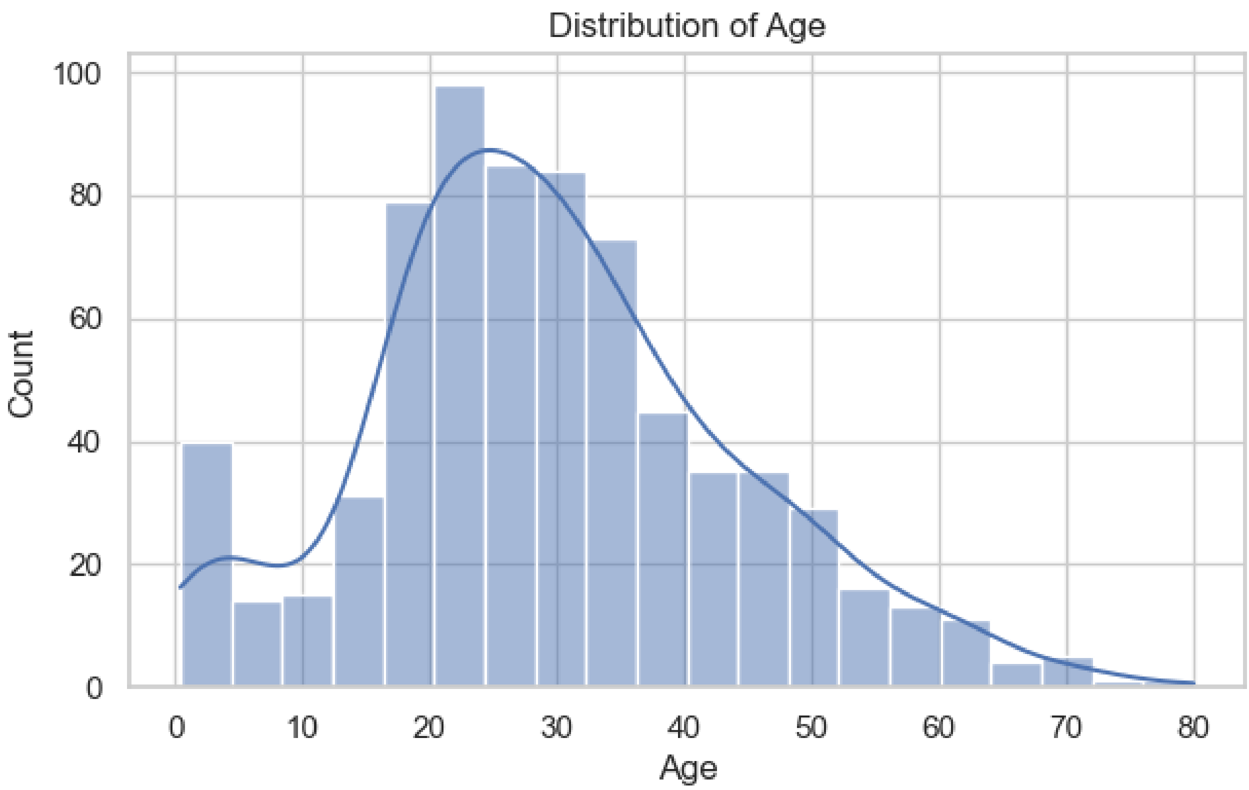
- **Printed metrics in terminal.**
- **Saved plots:**
  - **plots/roc_*.png → ROC curve**
  - **plots/confusion_matrix_*.png → confusion matrix**


Countplot of Embarked (by Survived if available)
Survived
0
1
count
400
350
300
250
200
150
100
50
0
S
C
Q
Embarked

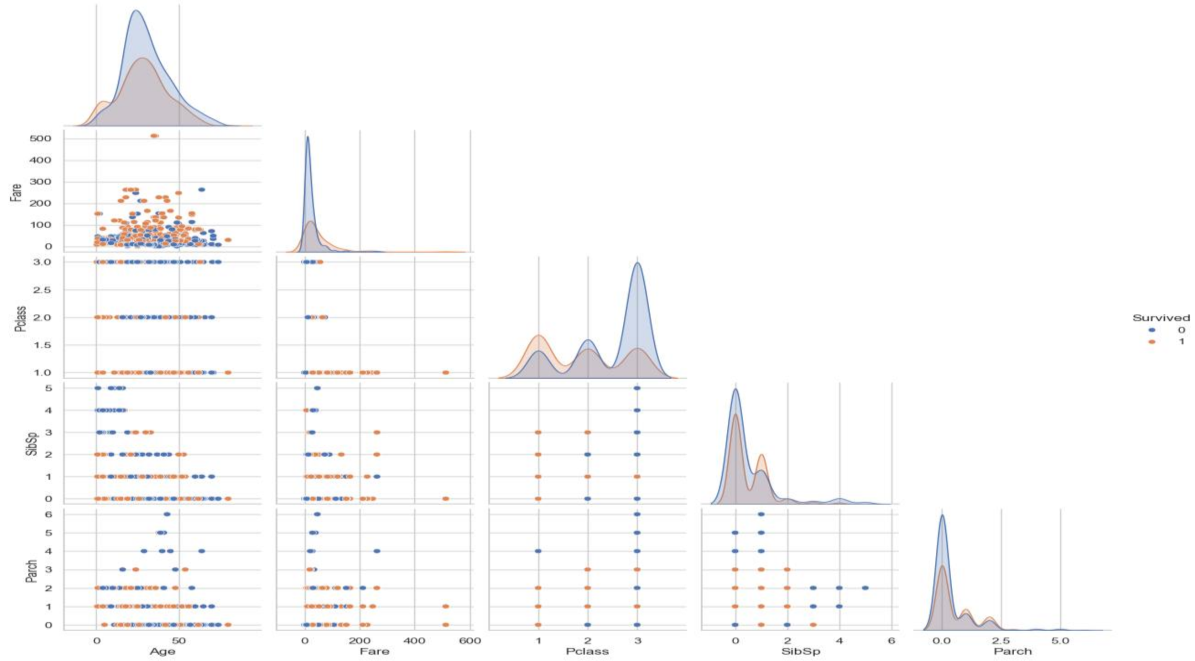

Countplot of Pclass (by Survived if available)
Survived
0
1
count
350
300
250
200
150
100
50
0
1
2
3
Pclass

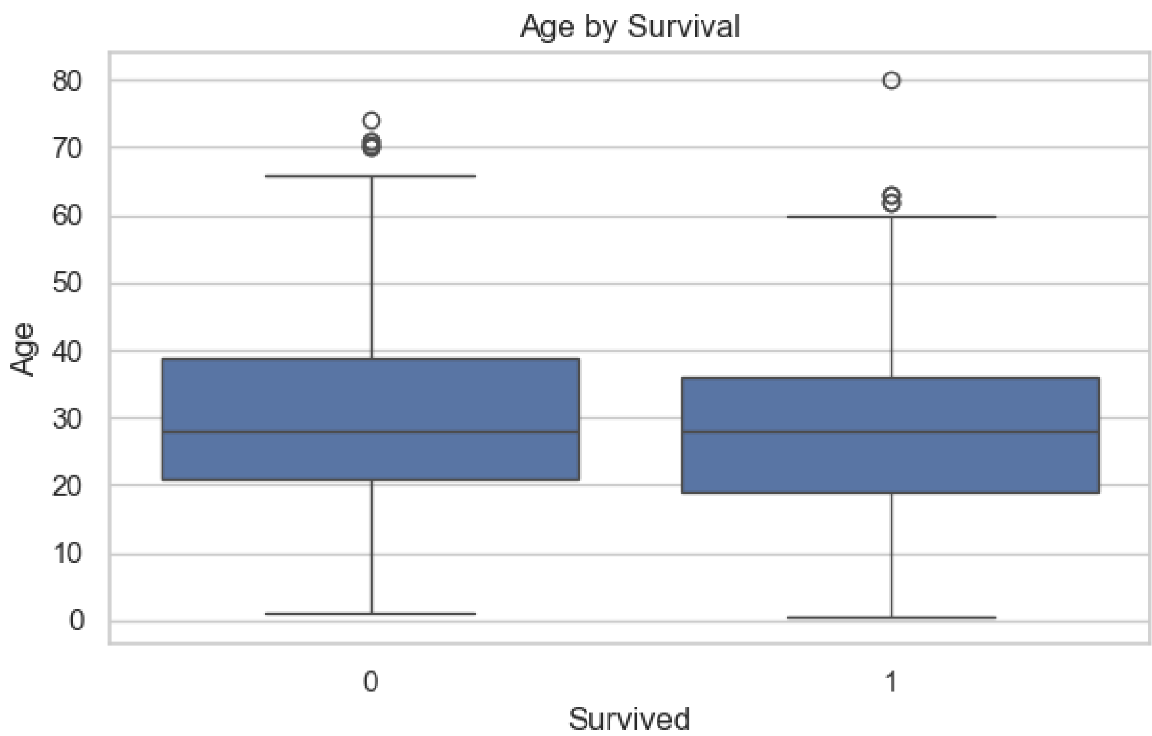


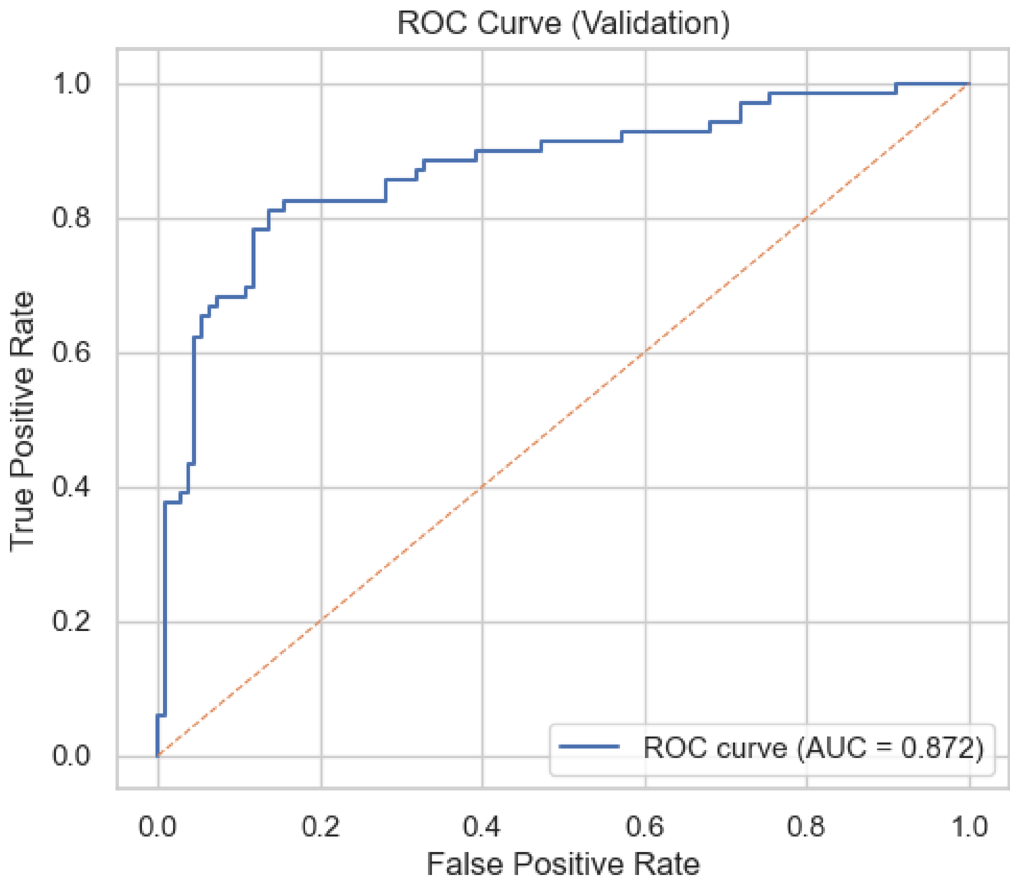
Countplot of Sex (by Survived if available)
Survived
0
1
count
0
100
200
300
400
male
female
Sex



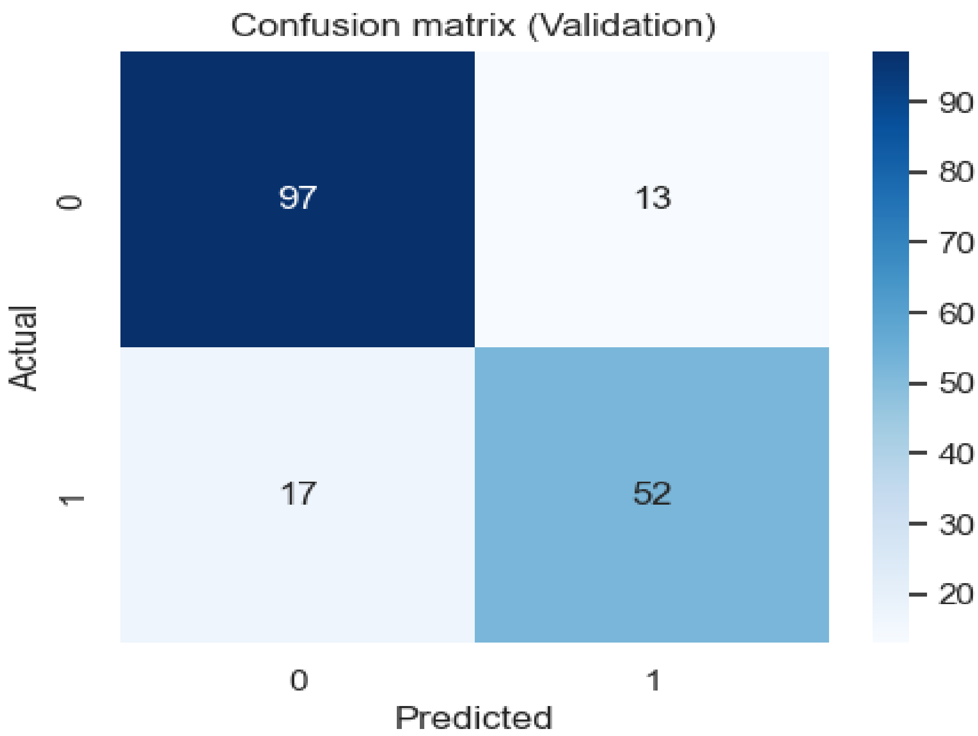
Distribution of Age
Count
0
20
40
60
80
100
0
10
20
30
40
50
60
70
80
Age


Age
Fare
Pclass
SibSp
Parch
Survived
0
1

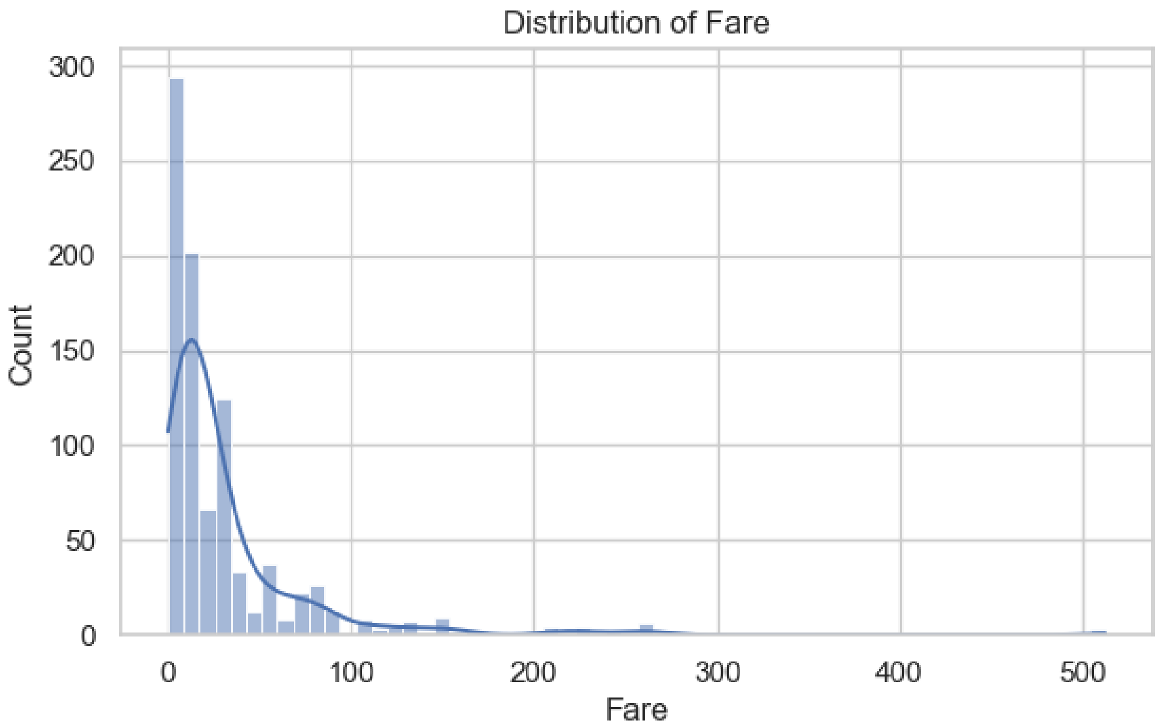

Age by Survival
Age
Survived
0
1

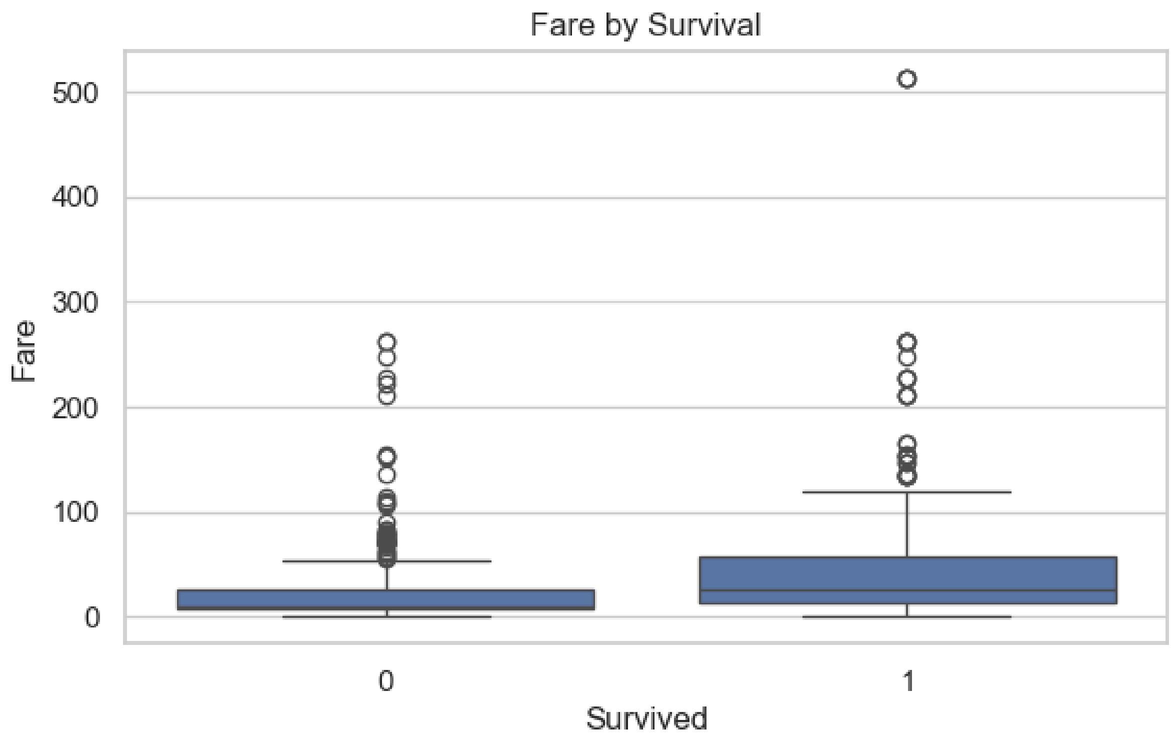


ROC Curve (Validation)
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC curve (AUC = 0.872)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate



Confusion matrix (Validation)
0
1
Actual
97
13
17
52
90
80
70
60
50
40
30
20
0
1
Predicted


Distribution of Fare
Count
300
250
200
150
100
50
0
0
100
200
300
400
500
Fare


Fare by Survival
Fare
500
400
300
200
100
0
0
1
Survived

5. Interpretation:
a. Interpret the coefficients of the logistic regression model.

Answer:
Interpreting the Coefficients
In logistic regression:

- **Positive coefficient → increases survival probability (holding all else constant).**
- **Negative coefficient → decreases survival probability.**
- **Magnitude = strength of influence.**

For example:

- **If Sex_female has a strong positive coefficient → being female strongly increases the odds of survival.**
- **If Pclass=3 (3rd class) has a strong negative coefficient → being in 3rd class decreases odds of survival.**

The model you trained extracted features like:

- **Sex_male, Sex_female**
- **Embarked_C, Embarked_Q, Embarked_S**
- **Title_Mr, Title_Mrs, Title_Miss, etc.**
- **Pclass (numeric 1–3)**
- **HasCabin (0/1)**
- **Numeric standardized versions of Age, Fare, SibSp, Parch.**

b. Discuss the significance of features in predicting the target variable (survival probability in this case).

Answer:

Significance of Features (Titanic Survival)

From many Kaggle/Titanic logistic regression runs, the usual pattern is:

1. **Sex**
   - female has the strongest positive impact (the "women and children first" rule).
   - male is the opposite → strong negative coefficient.
2. **Pclass (Ticket Class)**
   - 1st class passengers had much higher survival chances (positive influence).
   - 3rd class passengers much lower (negative).
3. **Title (from Name)**
   - Mr is usually strongly negative (most adult males did not survive).
   - Miss and Mrs positive (many women survived).
   - Rare titles vary — some officers or aristocrats fared better.
4. **Fare**
   - Higher fare = higher survival probability (proxy for wealth → better cabins, closer to lifeboats).
5. **Age**
   - Younger passengers slightly more likely to survive (but less strong than sex/class).
6. **HasCabin**
   - Having a recorded cabin number (1) often correlates with wealth/class → positive effect.

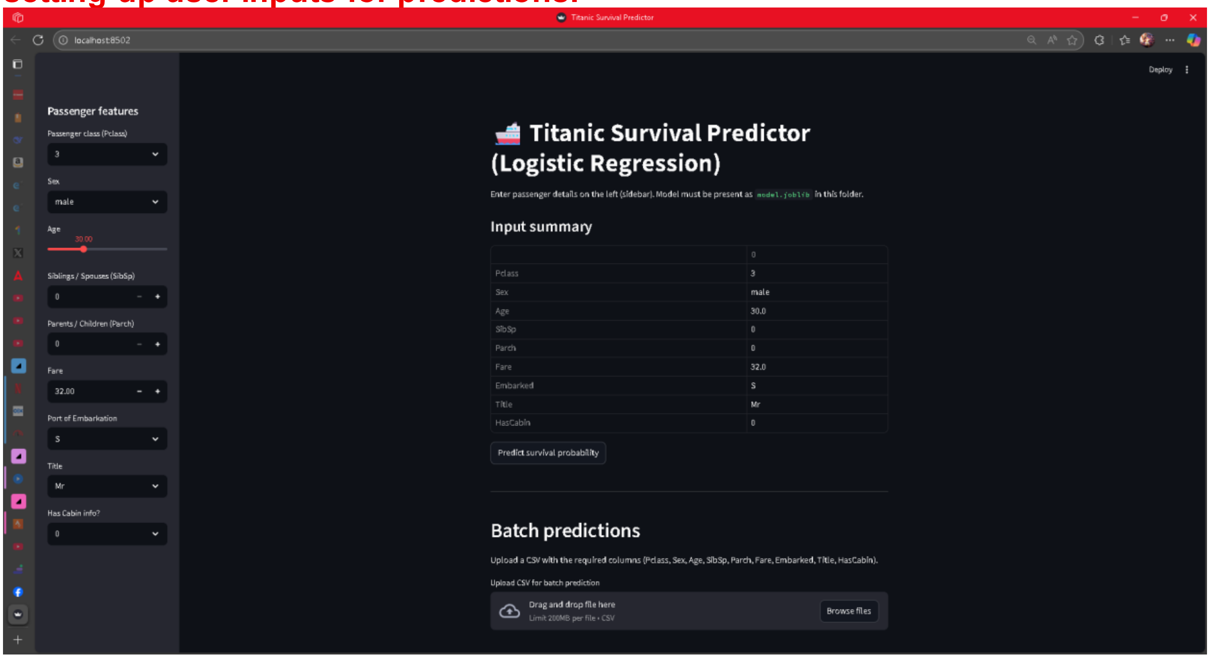
7. **Embarked**
    - Passengers from port C (Cherbourg) had higher survival odds than S (Southampton).

A Narrative Wrap-Up
So, logistic regression "thinks" survival probability rises if you were:
- **A female passenger,**
- **Traveling in 1st class,**
- **Paid a higher fare,**
- **With a known cabin,**
- **Holding a title like Miss/Mrs,**
- **Embarked from Cherbourg.**

And survival plummets if you were:
- **A male in 3rd class,**
- **With "Mr" in your name,**
- **Paid a very low fare,**
- **Had no cabin recorded.**

Which matches history pretty well: the Titanic's evacuation did prioritize women, children, and wealthy first-class passengers.

6. Deployment with Streamlit:
In this task, you will deploy logistic regression model using Streamlit. The deployment can be done locally or online via Streamlit Share. task includes creating a Streamlit app in Python that involves loading trained model and setting up user inputs for predictions.



(optional)For online deployment, use Streamlit Community Cloud, which supports deployment from GitHub repositories.
Detailed deployment instructions are available in the Streamlit Documentation.
https://docs.streamlit.io/streamlit-community-cloud/deploy-your-app
Interview Questions:
1. What is the difference between precision and recall?
2. What is cross-validation, and why is it important in binary classification?