DSF 35.0 Data Science

Random Forest Classification for Titanic Survival Prediction

Lutfiah Zahara
Math- Universitas Syiah Kuala

https://www.linkedin.com/in/lutfiah-zahara/

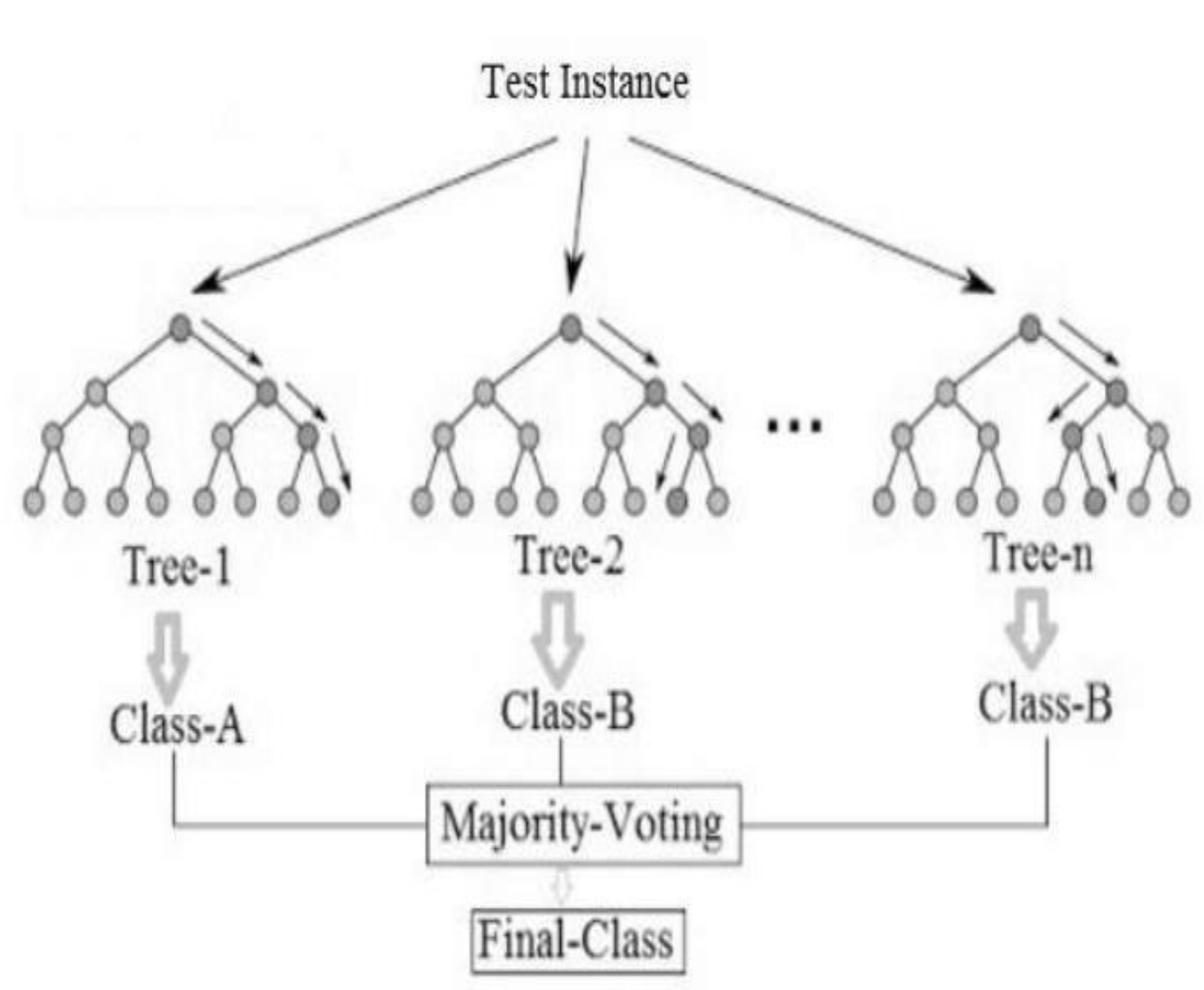
Titanic Dataset

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

This dataset contains information about the passengers of the Titanic that sank in 1912, and is often used to work on classification problems, namely predicting whether a passenger survived or not based on various features.

https://www.kaggle.com/datasets/yasserh/titanic-dataset

Random Forest



(Sumber: Krishnachandran, 2018)

Random Forest is a machine learning algorithm used for data classification. This algorithm is a development method of decision trees by forming several decision trees. In making a classification, all trees will make a prediction where the decision will be taken from the majority voting of each existing decision tree.

Random Forest Algorithm

1. The algorithm will select a random sample from the provided dataset.

2. Create a decision tree for each selected sample.

3. Each decision tree will produce a prediction. Each prediction result will be voted on using the most frequently appearing value.

4. The algorithm will select the most frequently selected prediction result as the final prediction.

For each decision tree in Random Forest, the formula used is:

$$f(x) = \sum_{i=1}^{n} w_i h_i(x)$$

Note:

$f(x)$: The output of each decision tree

n : Number of notes in a decision tree

w_i : The weight of each note in the decision tree

$h(x)$: Function that returns a value of 0 or 1, depending on whether x satisfies the conditions given by the node.

To form a Random Forest, the formula used is:

$$F(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$

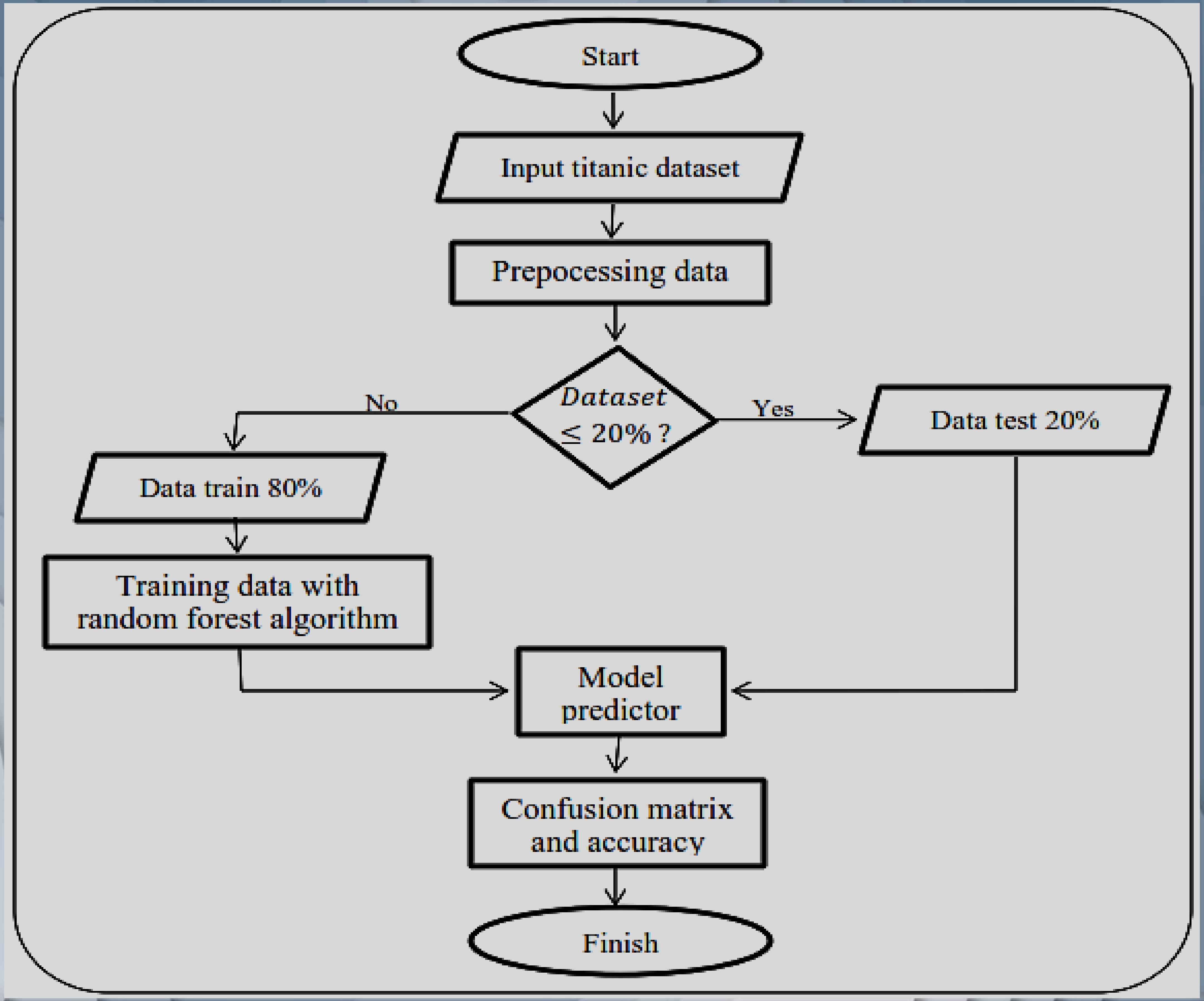
Note:

$\mathrm{F}(x)$: Output from Random Forest

N : Number of decision trees in Random Forest

$f_i(x)$: Output of the i-th decision tree

Flowchart


Start
Input titanic dataset
Prepocessing data
Dataset
≤ 20% ?
No
Yes
Data test 20%
Data train 80%
Training data with
random forest algorithm
Model
predictor
Confusion matrix
and accuracy
Finish

Tools

scikit learn

matplotlib

NumPy

pandas

Data Preprocessing

Check Empty Values

```
data_titanik.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
```

Before further analysis, we must check the data and preprocess the data if the data is still dirty. The image on the side is the result obtained after checking the empty values. Because there are still empty values in the age, cabin and embarked columns. So, we can fill the column using the median, mode, mean values and eliminate the column.

Data Preprocessing

ibimbing

1 Fill the empty age column value with median

```
data_titanik['Age'] = data_titanik['Age'].fillna(data_titanik['Age'].median())
data_titanik
```

2 Fill the empty embarked column value with mode

```
data_titanik['Embarked'] = data_titanik['Embarked'].fillna(data_titanik['Embarked'].mode()[0])
data_titanik
```

3 Remove cabin column values because many are empty

```
data_titanik = data_titanik.drop('Cabin', axis=1)
data_titanik
```

Check data results after preprocessing

```
data_titanik.isnull().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
```

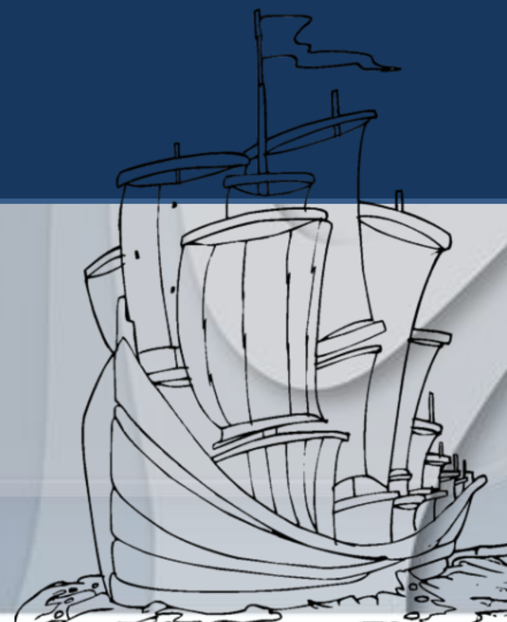
Data Preprocessing

Check Duplicate Data

```
# check the duplicates
duplicates = data_titanik[data_titanik.duplicated()]
print("Duplicate rows :")
print(duplicates)
```

```
Duplicate rows :
Empty DataFrame
Columns: [PassengerId, Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked]
Index: []
```

There is no duplicate data in the data

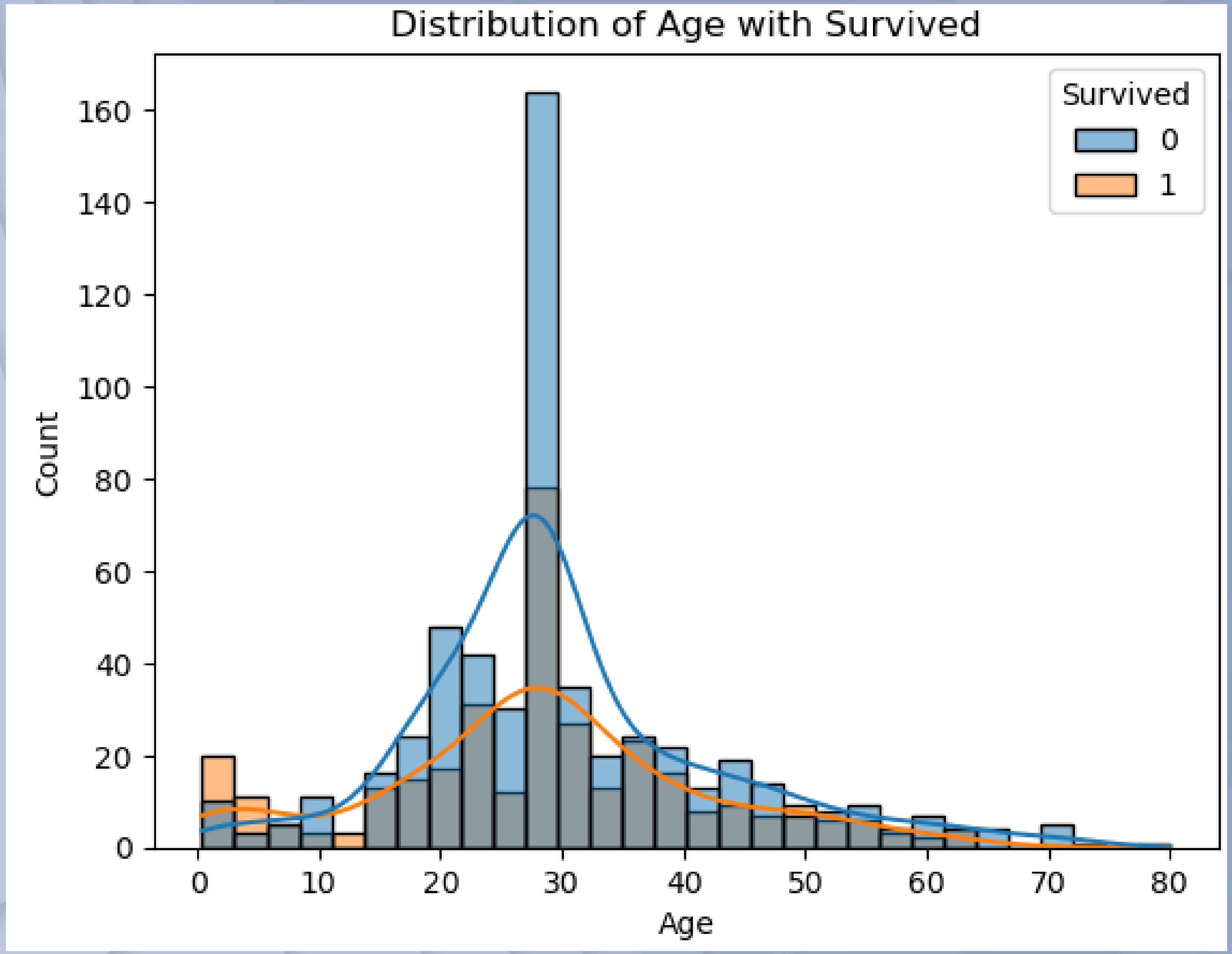


Exploratory Data Analysis

(EDA)

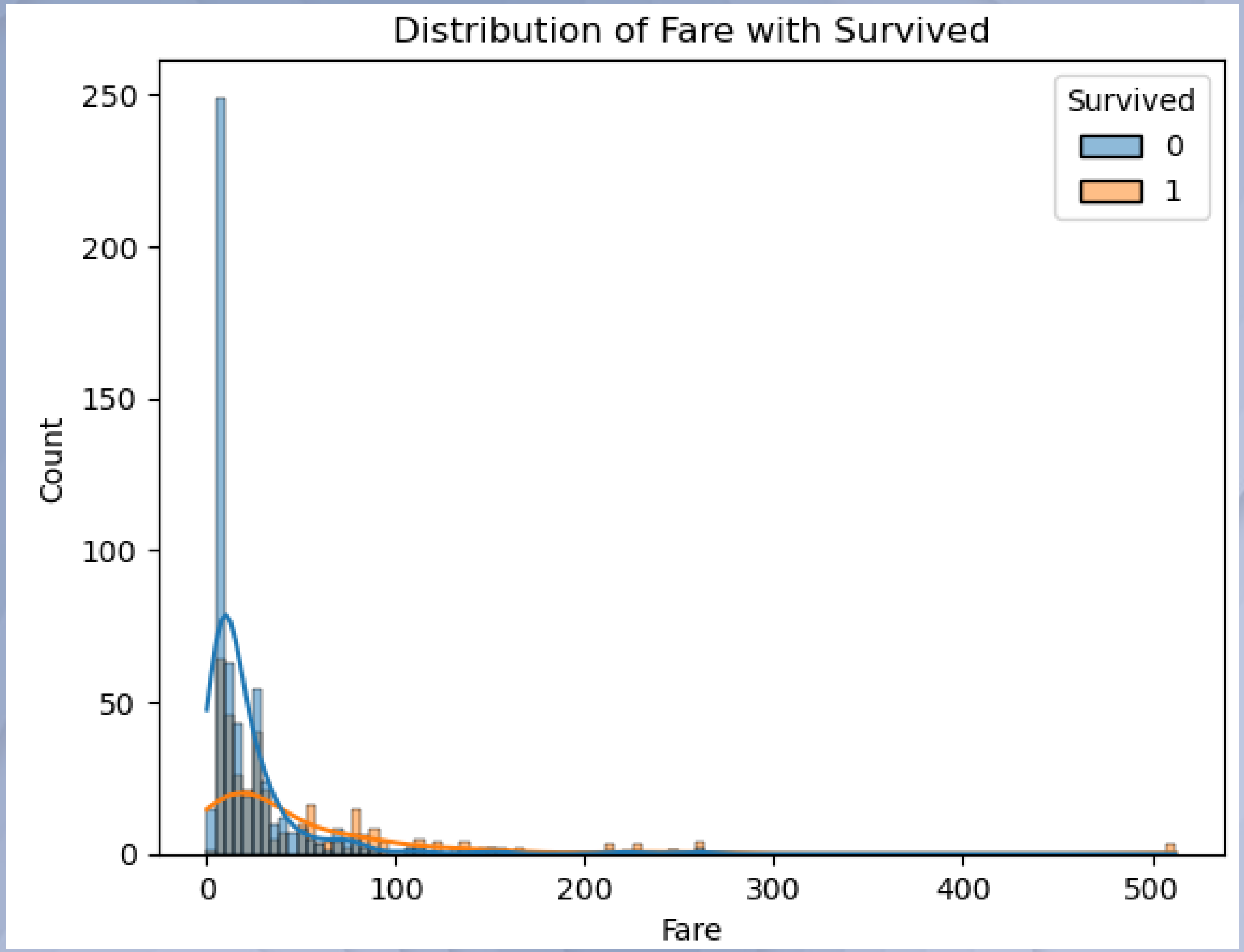
EDA (Exploratory Data Analysis)

Age Column Analysis



Children aged ≤ 10 years have a better chance of survival, perhaps because they are prioritized for rescue using lifeboats.

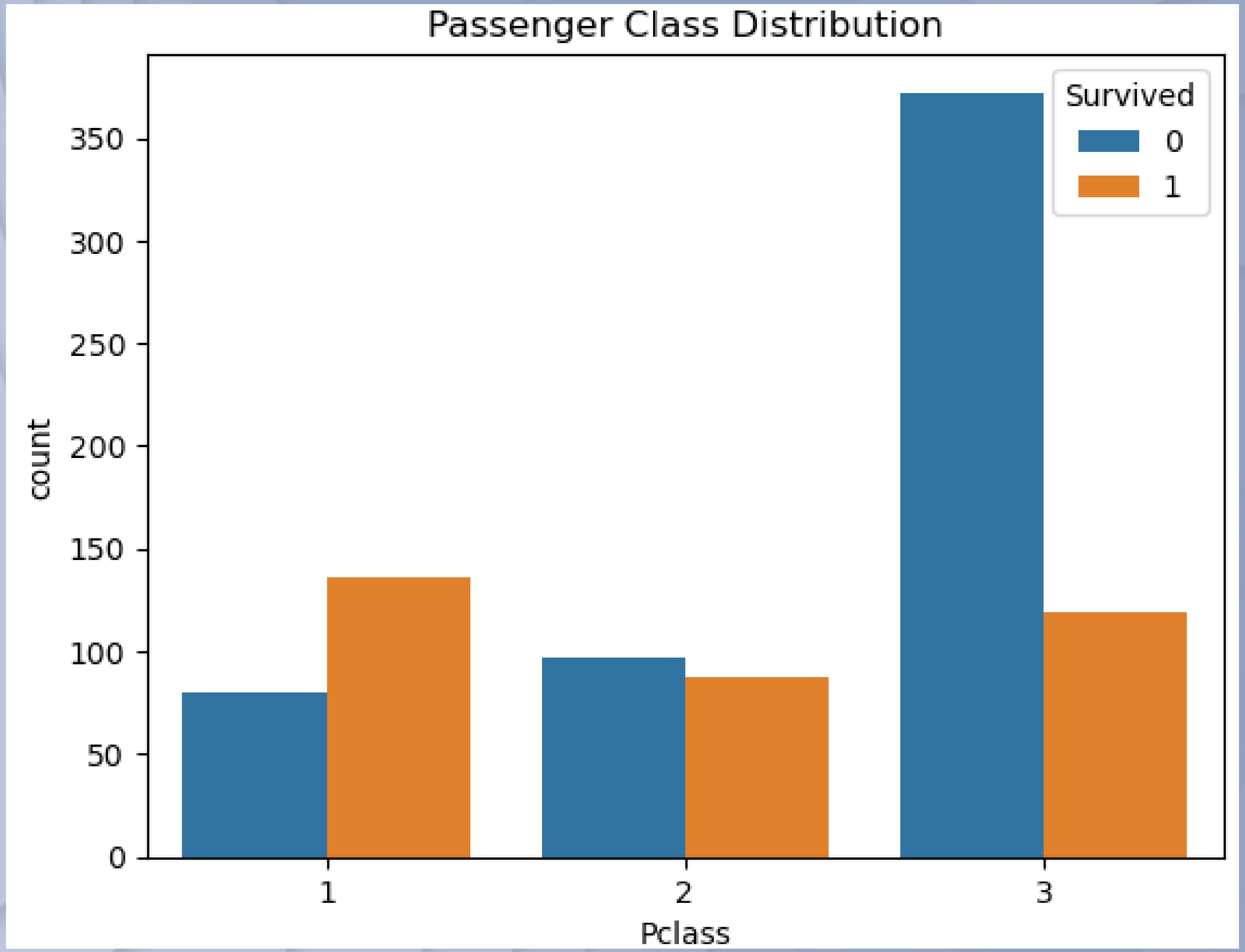
Fare Column Analysis



Passengers with high costs have a better chance of survival than those with low costs.

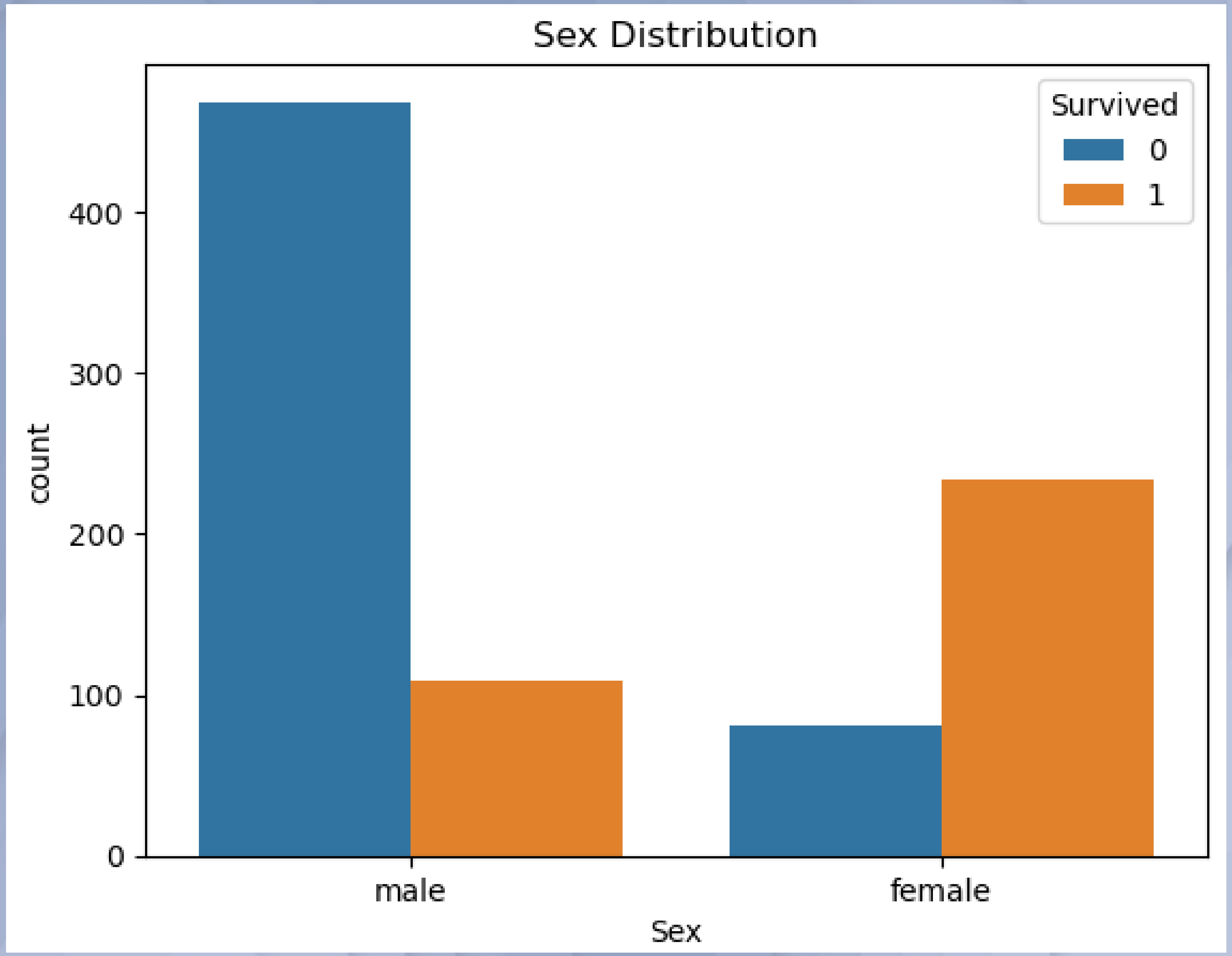
EDA (Exploratory Data Analysis)

Class Column Analysis



Passengers in class 1 have a greater chance of survival compared to other classes.

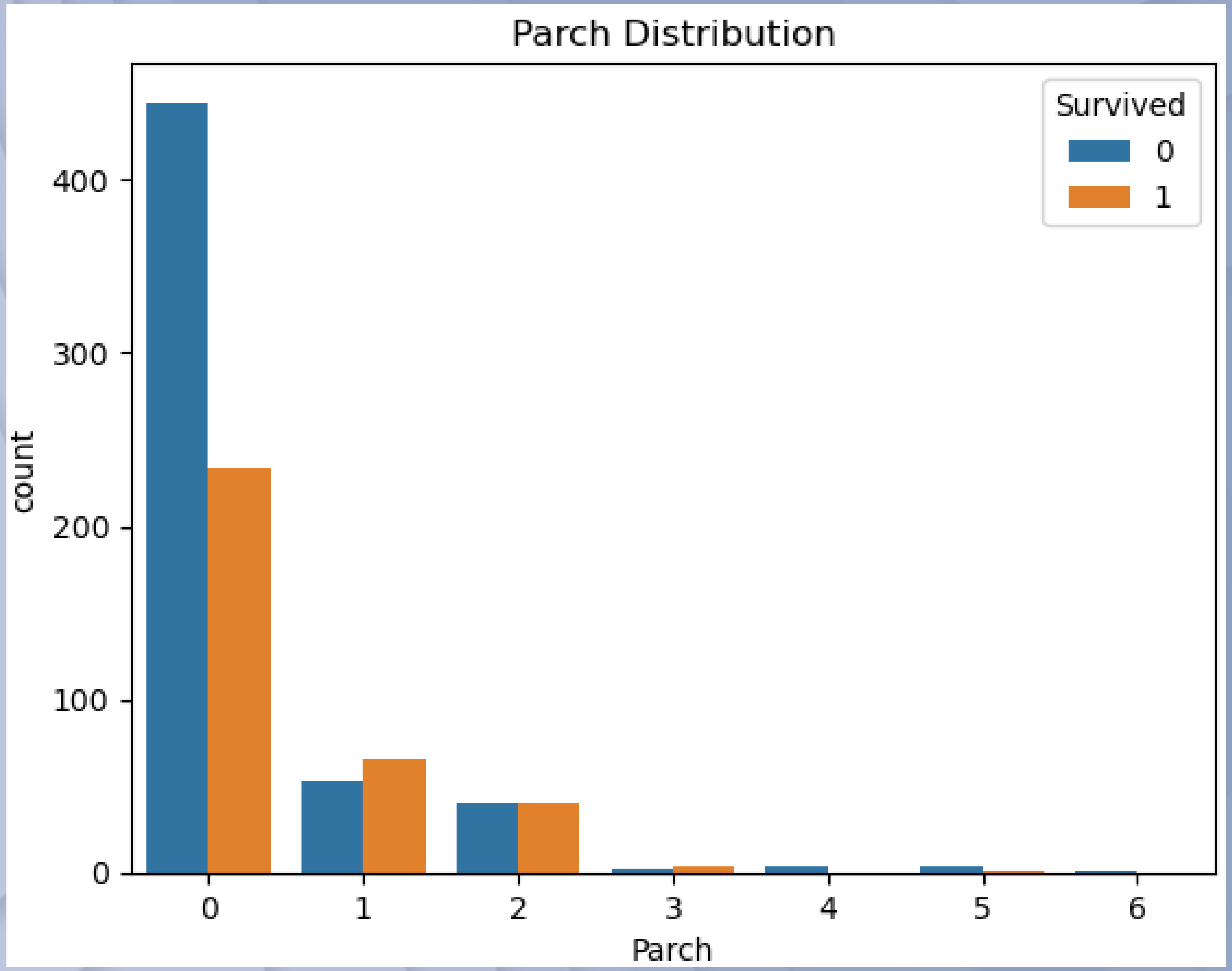
Sex Column Analysis



Female passengers have a better chance of survival than male passengers, perhaps because females are given priority to exit first.

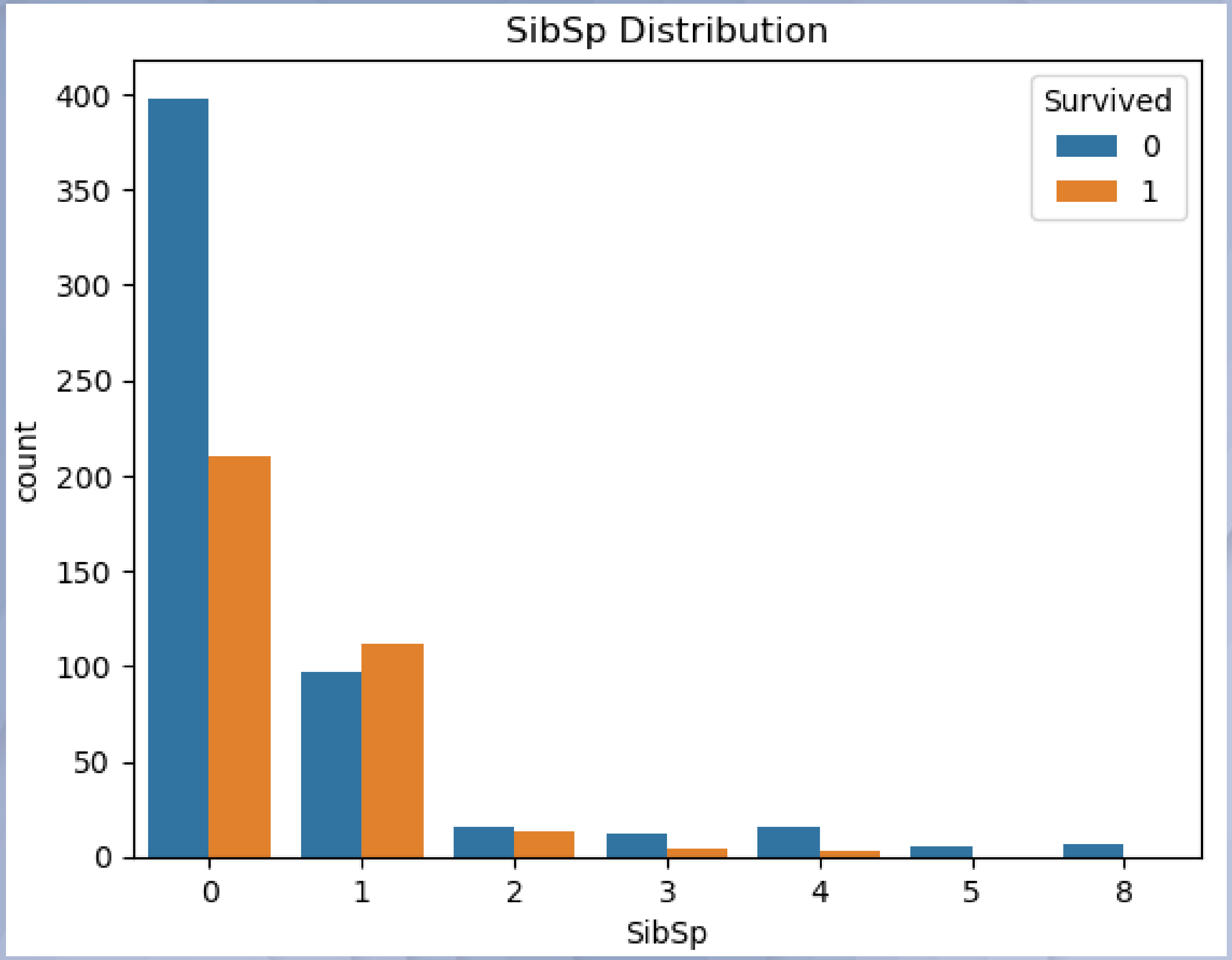
EDA (Exploratory Data Analysis)

Parch Column Analysis



Passengers with small families (Parch=1 or Parch=2) are more likely to survive than those alone (Parch=0) or with large families.

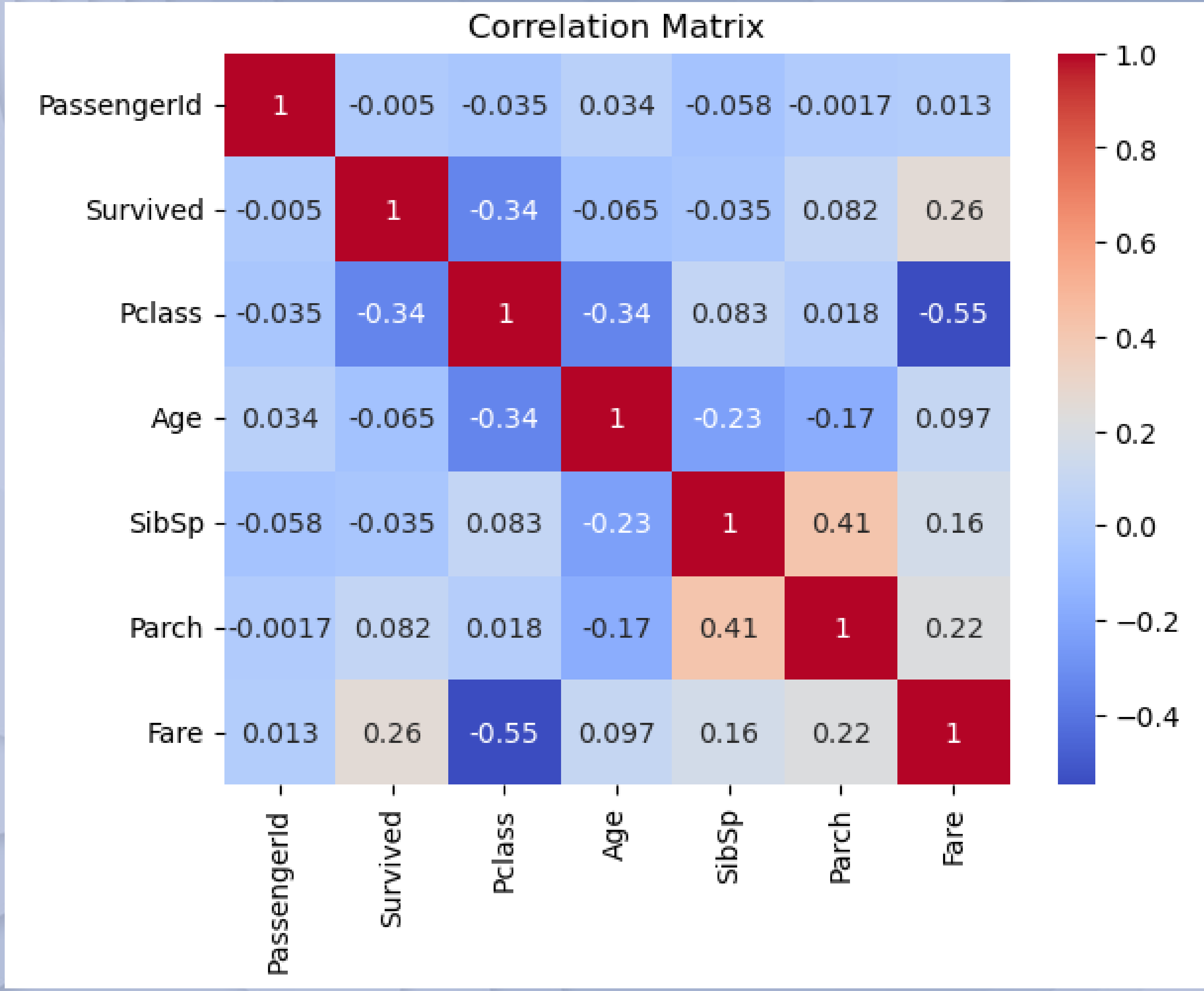
SibSp Column Analysis



Passengers with SibSp=1 have a higher chance of survival than those alone (SibSp=0). The chance decreases if SibSp is greater.

EDA (Exploratory Data Analysis)

Correlation Matrix



- ❑ Pclass and Fare have a negative correlation, the lower the Pclass the higher the Fare.
- ❑ The most influential feature on Survived is Pclass.

Converting Categorical Data to Numeric

```
le = LabelEncoder()
data_titanik['Sex'] = le.fit_transform(data_titanik['Sex'])
data_titanik['Embarked'] = le.fit_transform(data_titanik['Embarked'])
```

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

Future Selection

```
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
x = data_titanik[features]
y = data_titanik['Survived']
```

X

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	2
1	1	0	38.0	1	0	71.2833	0
2	3	0	26.0	0	0	7.9250	2
3	1	0	35.0	1	0	53.1000	2
4	3	1	35.0	0	0	8.0500	2
...	...	...	...	...	...	...	...

Y

```
y
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

Separating data for training and testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

- ✓ The train_test_split(x, y, test_size=0.2, random_state=42) function splits the x (features) and y (labels/targets) datasets into training and testing data.
- ✓ test_size=0.2: Specifies that 20% of the data will be used for testing, while 80% is used for training.

Modelling and Evaluation

Building a classification model using Random Forest

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(x_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

- n_estimators determines the number of trees to be created in the Random Forest. When we set n_estimators=100, this means that we will create 100 different decision trees.
- random_state is a number used to set the random "seed". Using random_state ensures that the results we get from running the model are always consistent. For example, every time we run the code with random_state=42, the results will be the same.

Evaluation

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

➢ Prediction of test data using the random forest model that has been built

```
y_pred = model.predict(x_test)
y_pred
```

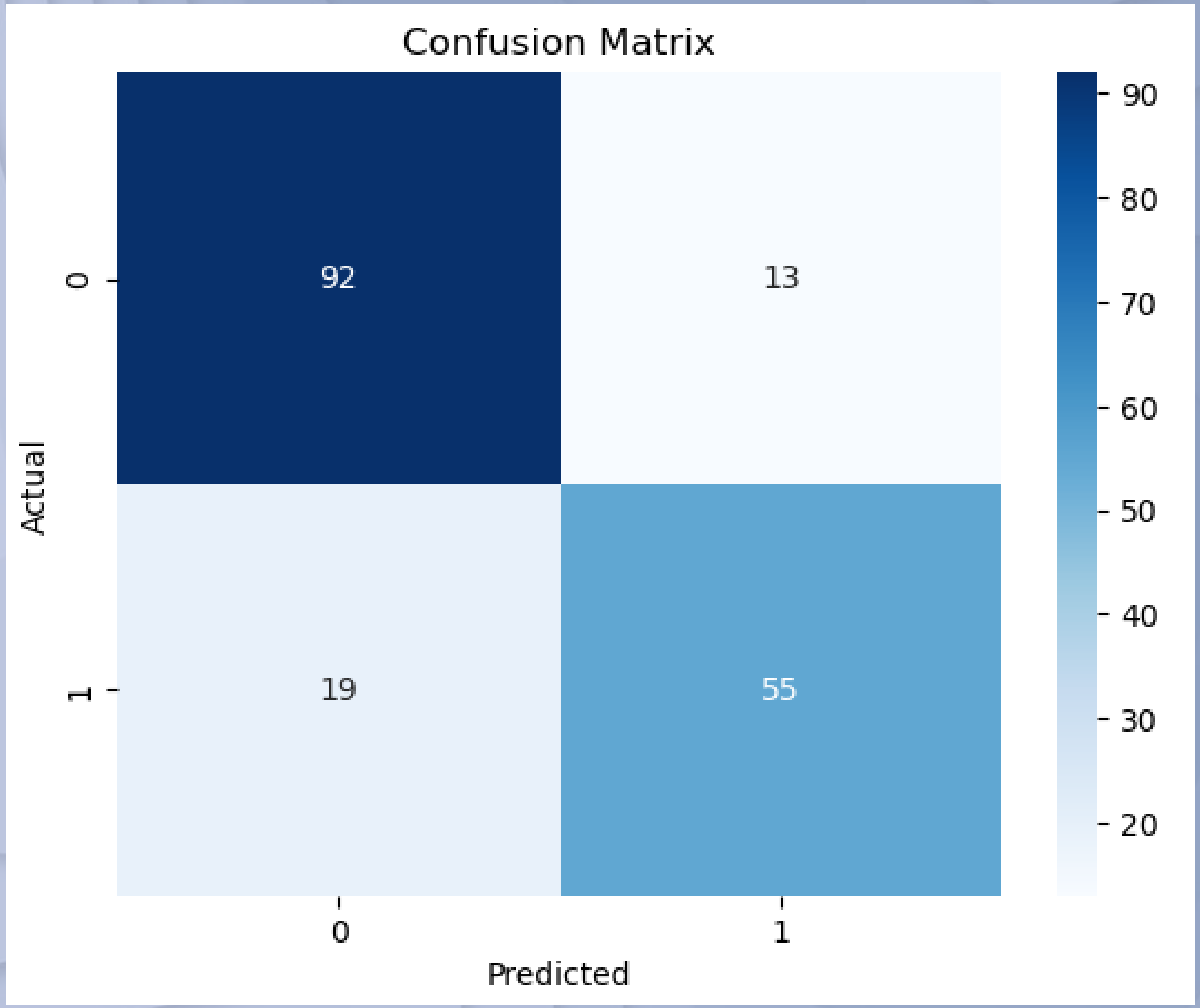
```
array([0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 1], dtype=int64)
```

```
data_titanik_comparison = pd.DataFrame({
    'Actual': y_test,
    'Predicted': y_pred
})
data_titanik_comparison
```

	Actual	Predicted
709	1	0
439	0	0
840	0	0
720	1	1
39	1	0
...	...	...
433	0	0
773	0	0
25	1	0
84	1	1
10	1	1

179 rows × 2 columns

Confusion Matrix & Accuracy



$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{92 + 55}{92 + 55 + 13 + 19} = 0{,}82$$

PREDICTION/ ACTUAL	POSITIVE	NEGATIVE
POSITIVE	True Positive	False Positive
NEGATIVE	True Negative	False Negative

- True Negative (TN) is 92, meaning the model successfully predicted 92 negative cases correctly.
- False Positive (FP) is 13, meaning the model incorrectly predicted 13 negative cases as positive.
- False Negative (FN) is 19, meaning the model incorrectly predicted 19 positive cases as negative.
- True Positive (TP) is 55, meaning the model successfully predicted 55 positive cases correctly.

Feature Importances

```
feature_importance = model.feature_importances_

feature_names = x_train.columns

data_titanik_feature_importance = pd.DataFrame({
    'Feature': feature_names,
    'Importance': feature_importance
})

data_titanik_feature_importance = data_titanik_feature_importance.sort_values(by='Importance', ascending=False)
data_titanik_feature_importance
```

	Feature	Importance
1	Sex	0.271410
5	Fare	0.265010
2	Age	0.249995
0	Pclass	0.086957
3	SibSp	0.053685
4	Parch	0.039897
6	Embarked	0.033044

- The most influential features in the Random Forest model are Sex (0.271), Fare (0.265), and Age (0.250). These features show significant relationships with safety prediction.

- Other features such as Pclass, SibSp, Parch and Embarked have lower influences.

Conclusion

- Women have a higher chance of survival, while higher ticket prices and younger age also correlate with higher survival rates.
- The Random Forest model shows that Sex, Fare, and Age are the most influential features in predicting the safety of Titanic passengers. Other features such as Pclass, SibSp, and Embarked have a smaller influence.
- From the confusion matrix, the model accuracy is calculated at 82.1%, indicating good performance in classification. However, there are still errors especially in False Negatives (19) which can be reduced with further optimization.

ibimbing

THANK'S

More Information!!!

https://www.linkedin.com/in/lutfiah-zahara/

lutfiah.zahara25@gmail.com