Reto Kaggle Titanic

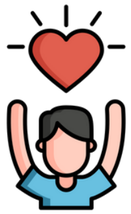
Alfredo Park
A01658259

Miguel Bustamante
A01781583

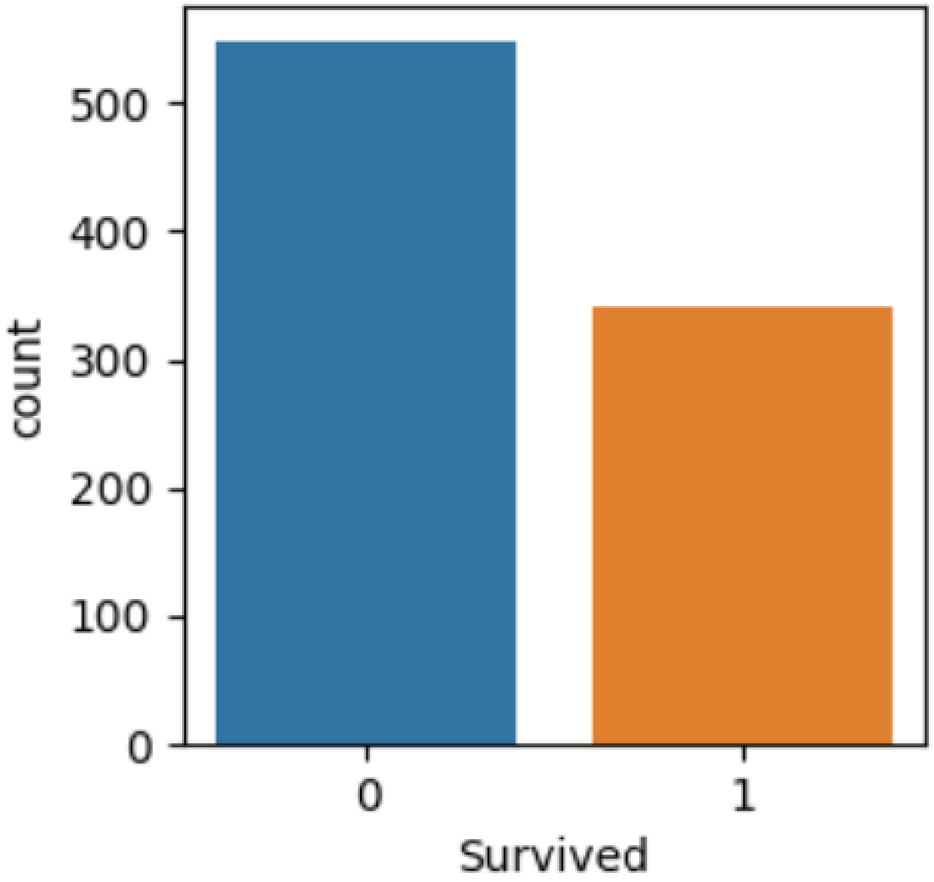
Fernando Arana
A01272933

Mauricio Juárez
A01660336

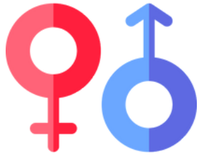
Distribution | Survived



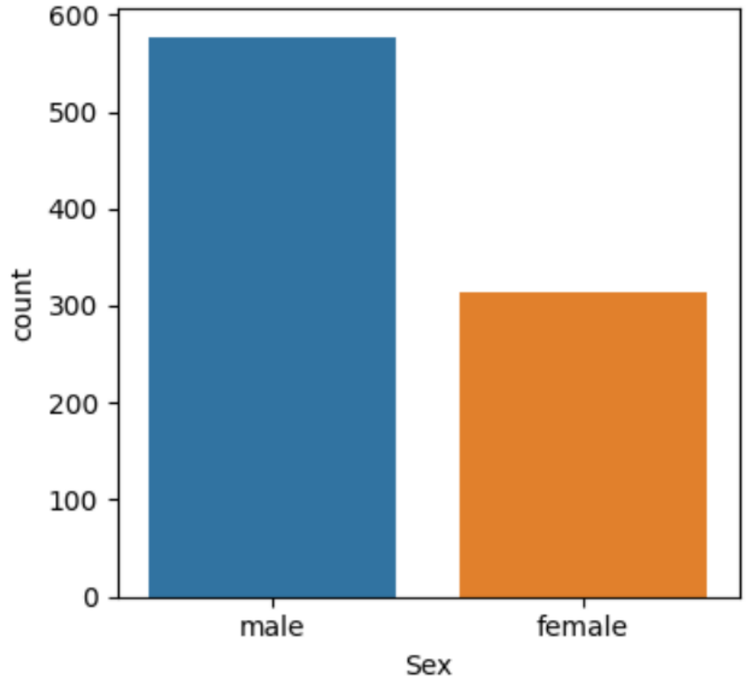
```
Porcentage of survived:  61.62 %
Porcentage of no-survived:  38.38 %
[]
```

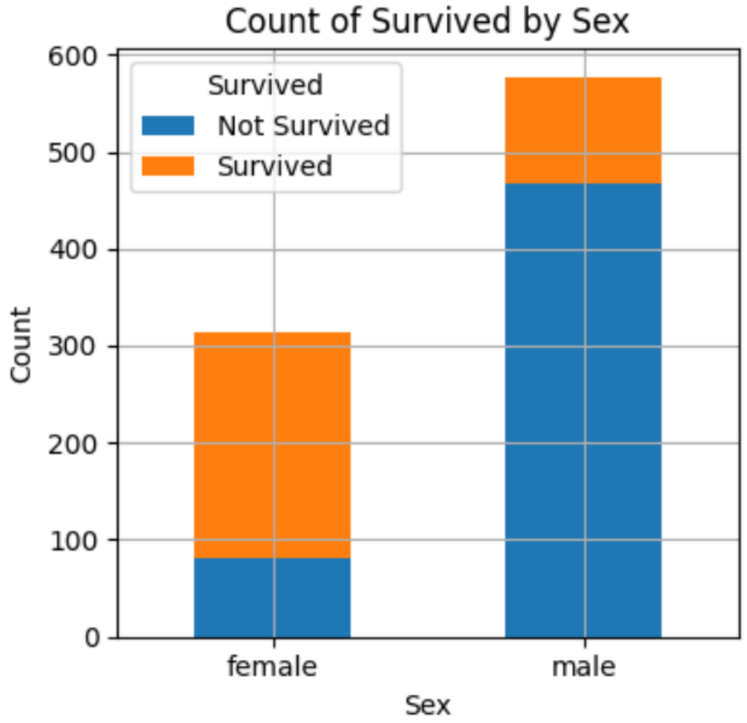


Distribution | Sex



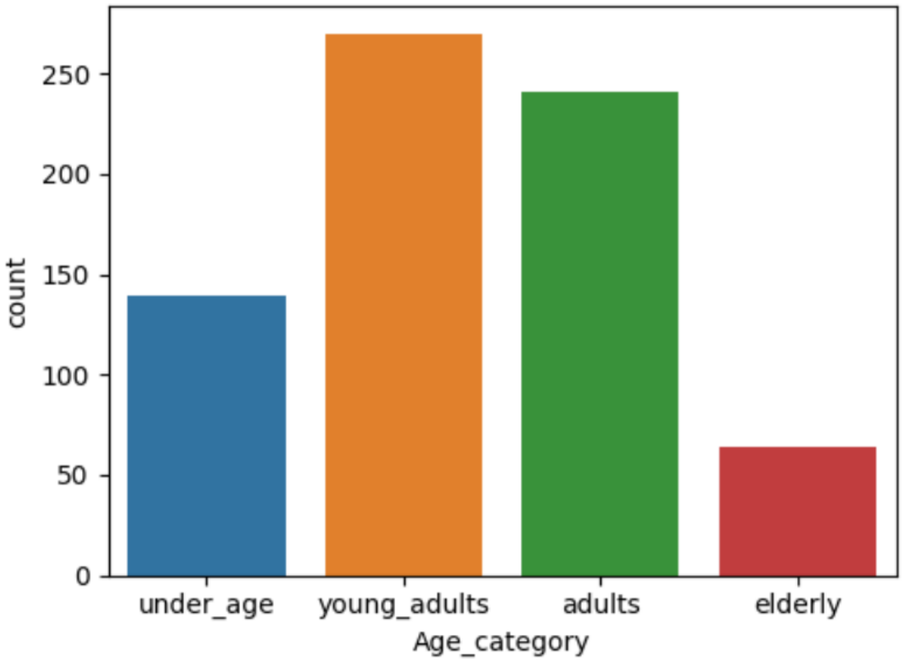
```
Porcentage of male ( 577 ):  64.76 %
Porcentage of female ( 314 ):  35.24 %


Porcentage of male surviving:  18.89 %
Porcentage of male not surviving:  81.11 %


Porcentage of female surviving:  74.2 %
Porcentage of female not surviving:  25.8 %
[]
```

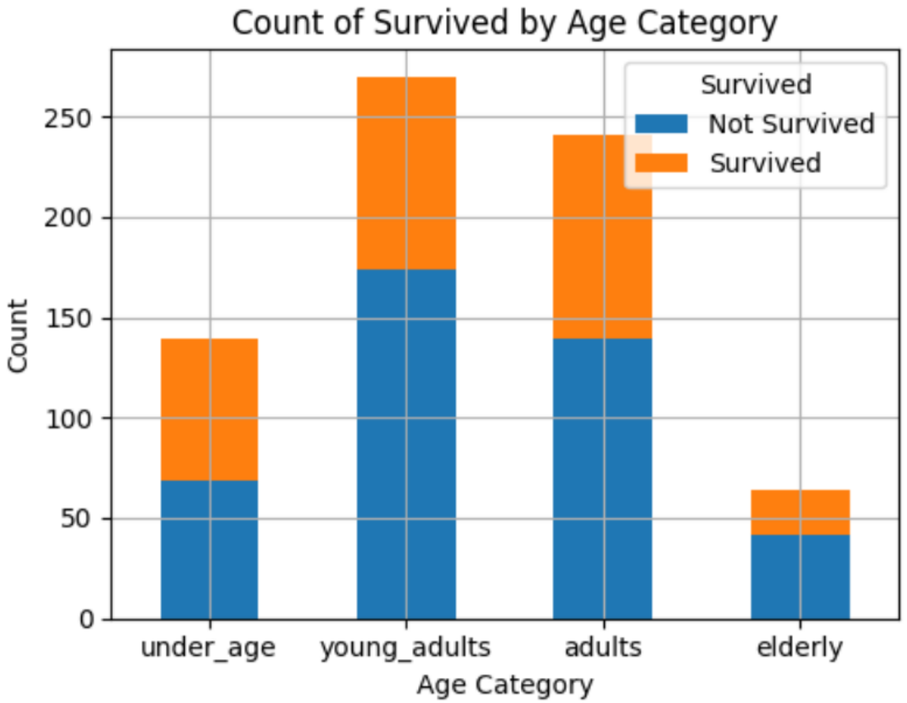




Distribution | Age

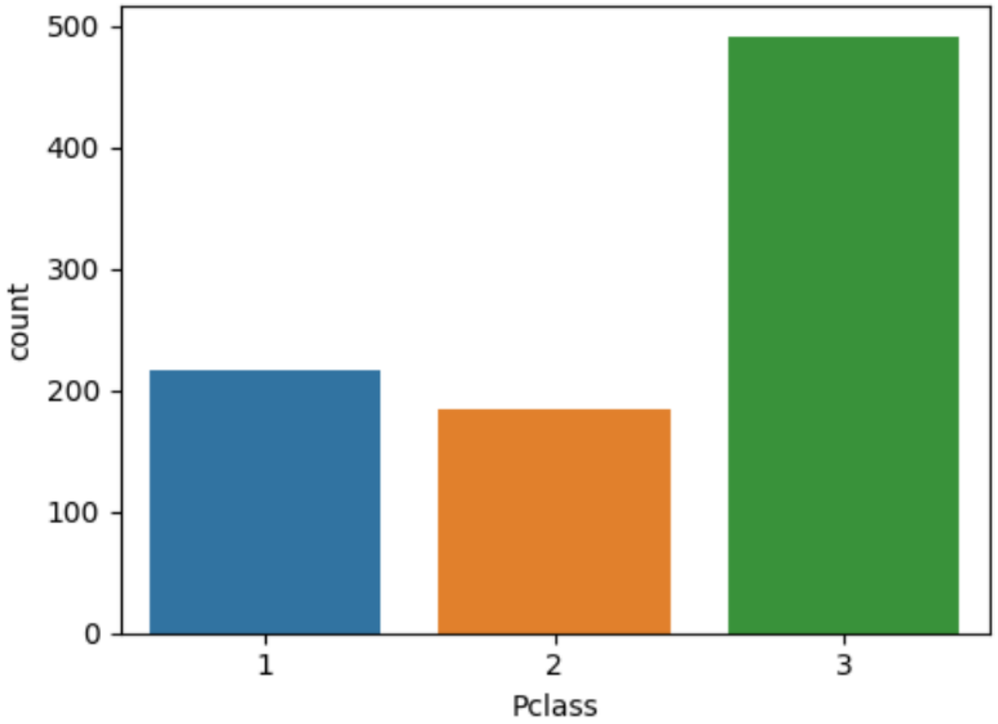
Under age:	0 - 18 years old	Not-Survived: 49.64 %	Survived: 50.36 %
Young adults:	19 - 30 years old	Not-Survived: 64.44 %	Survived: 35.56 %
Adults:	30 - 50 years old	Not-Survived: 57.68 %	Survived: 42.32 %
Elderly:	50+ years old	Not-Survived: 65.62 %	Survived: 34.38 %

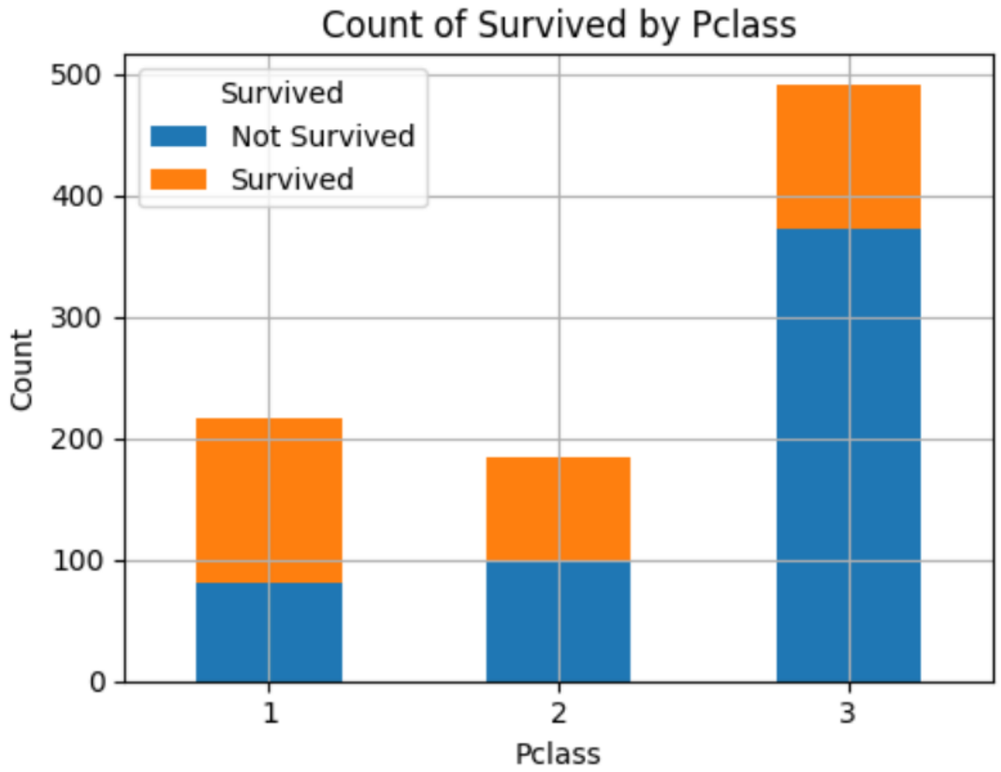




Distribution | Pclass

```
Porcentage of people in 1st class ( 184 ):  20.65 %     Not-Survived:  37.04 %     Survived: 62.96 %
Porcentage of people 2nd class ( 216 ):  24.24 %        Not-Survived:  52.72 %     Survived: 47.28 %
Porcentage of people 3rd class ( 491 ):  55.11 %        Not-Survived:  75.76 %     Survived: 24.24 %
```



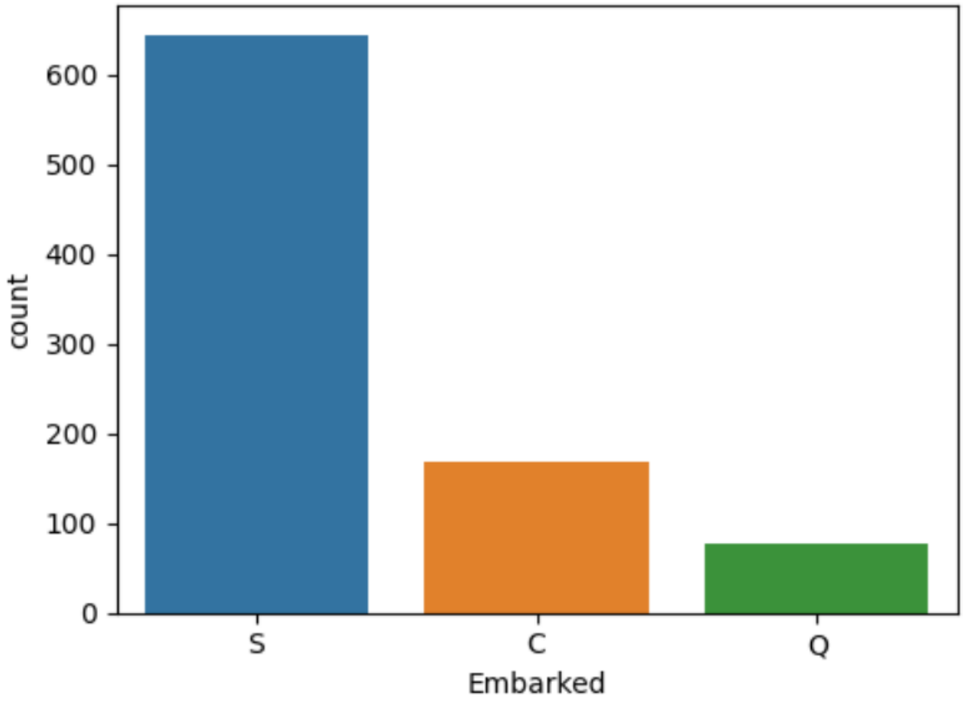


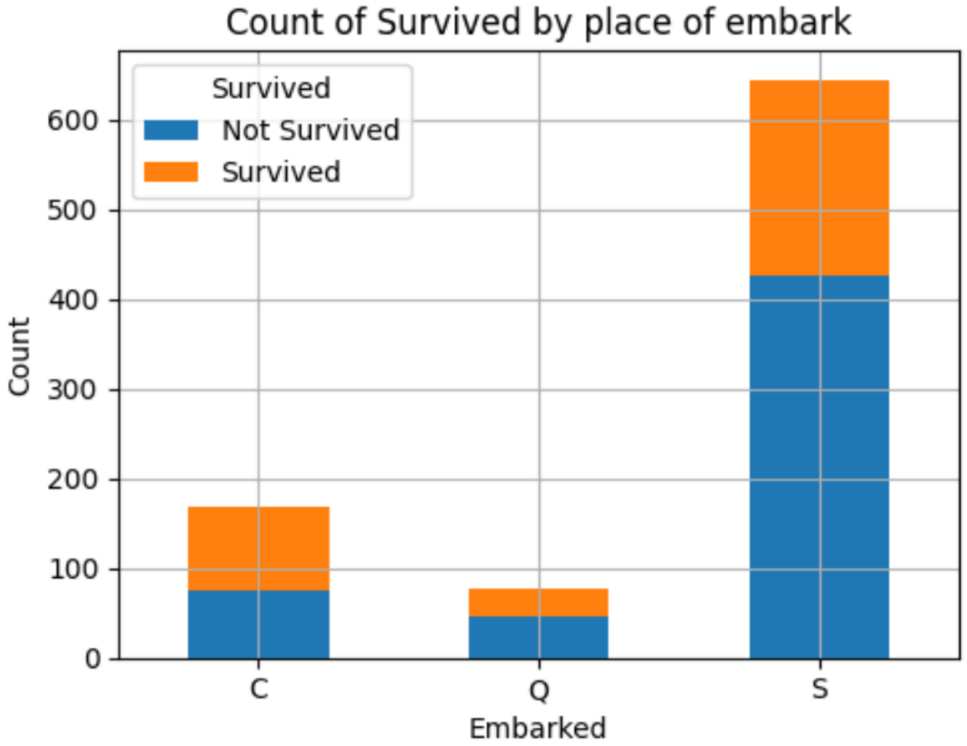
Distribution | Embarked

```
Porcentage of embark in C = Cherbourg ( 168 ):  18.86 %
Porcentage of embark in Q = Queenstown ( 77 ):  8.64 %
Porcentage of embark in S = Southampton ( 644 ):  72.28 %
```

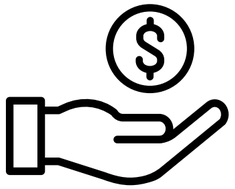
```
Not-Survived:  44.64 %
Not-Survived:  61.04 %
Not-Survived:  66.3 %
```

```
Survived: 55.36 %
Survived: 38.96 %
Survived: 33.7 %
```

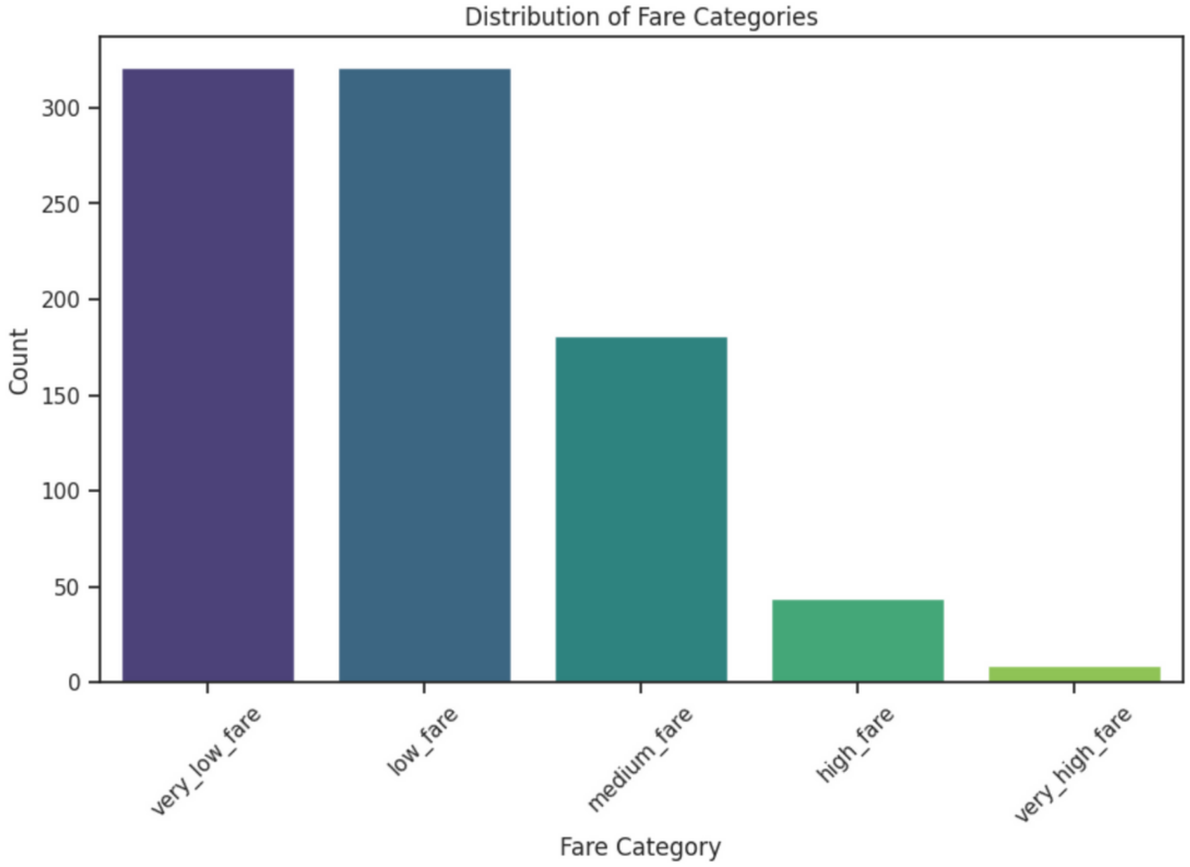




Distribution | Fare



```
Very Low Fare paid ($0 - $10 dlls):      321 people.
Low Fare paid ($11 - $30 dlls):          321 people.
Medium Fare paid ($31 - $100 dlls):      181 people.
High Fare paid ($101 - $250 dlls):       44 people.
Very High Fare paid ($251 - $550 dlls):  9 people.
```



Distribution | Feature Engineering

Total Family

$$TotalFamily = \frac{Siblings}{Spouse} + \frac{Parents}{Children} = \frac{Siblings + Parents}{Spouse + Children}$$

if Total_Family == 0

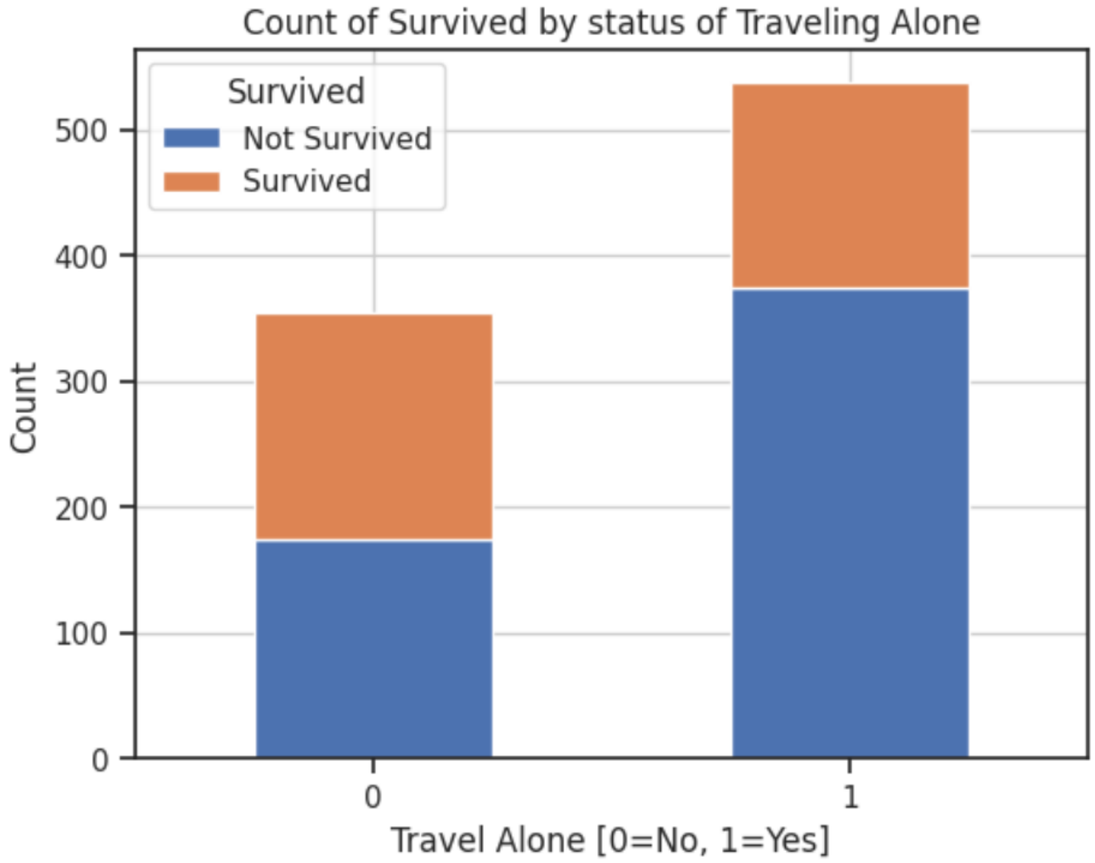
then travels alone

else if Total_Family > 0

then travels with company

```
Count of people Traveling Alone:  354
Count of people Not-Traveling Alone:  537
```

Travel Alone?



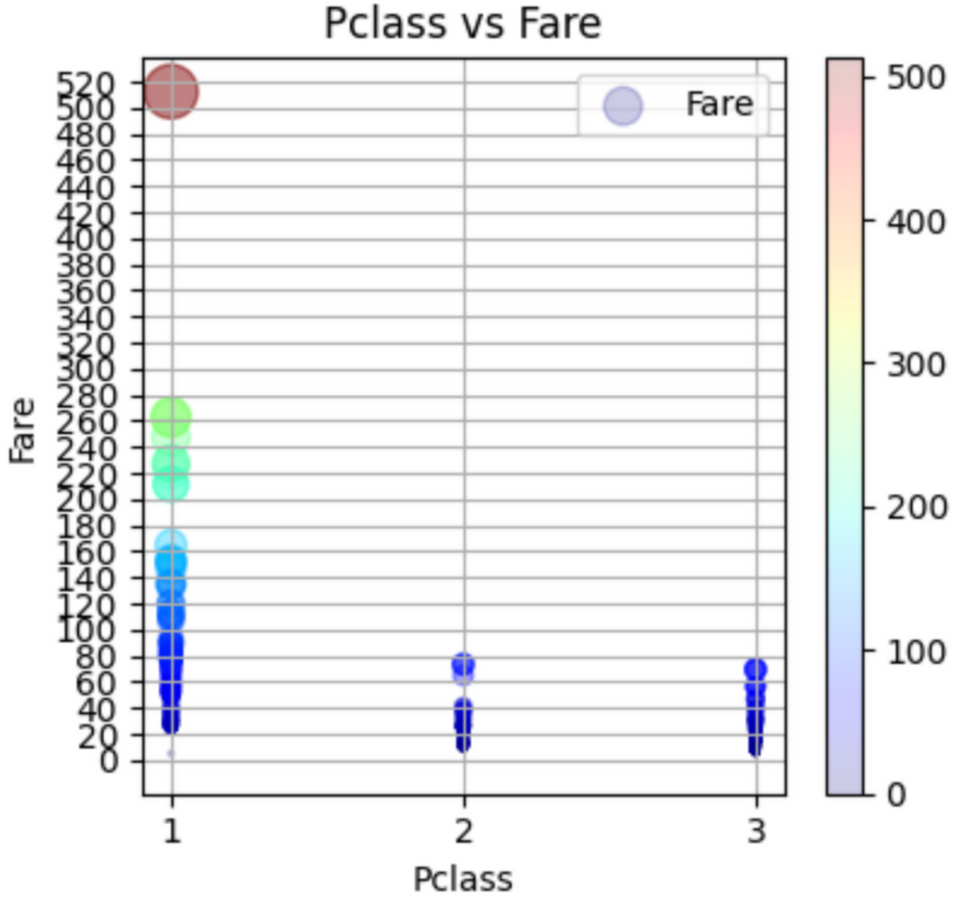
Distribution

Analyzing Pclass vs Fare

This graph reveals some interesting insights. It's noticeable that the fare prices for 2nd and 3rd class tickets are similar.

However, when it comes to 1st class tickets, we observe a wider range of fare prices. Some 1st class tickets were sold at prices comparable to those of 2nd and 3rd classes.

On the other hand, certain 1st class tickets were sold at significantly higher prices. This suggests the presence of outliers in the Fare attribute.



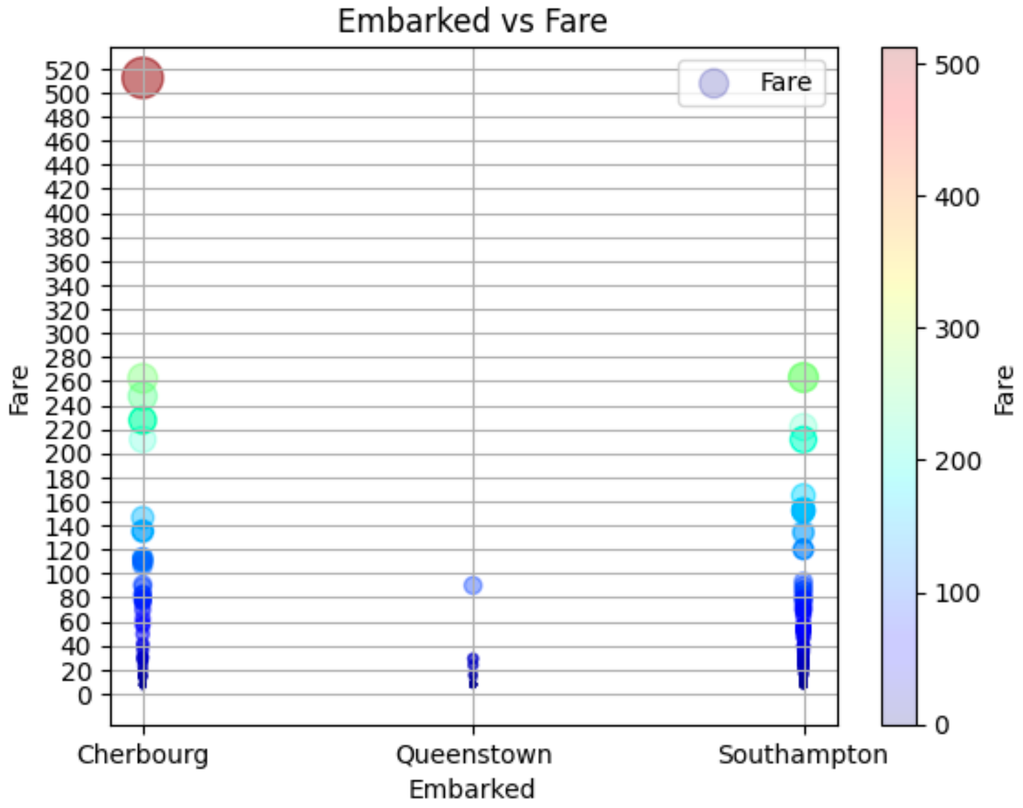
Distribution

Analyzing Port of embarkation vs Fare

This graph illustrates fare prices based on the port of embarkation.

The data shows that fares from Cherbourg were notably higher, followed by Southampton, while Queenstown had the lowest fares.

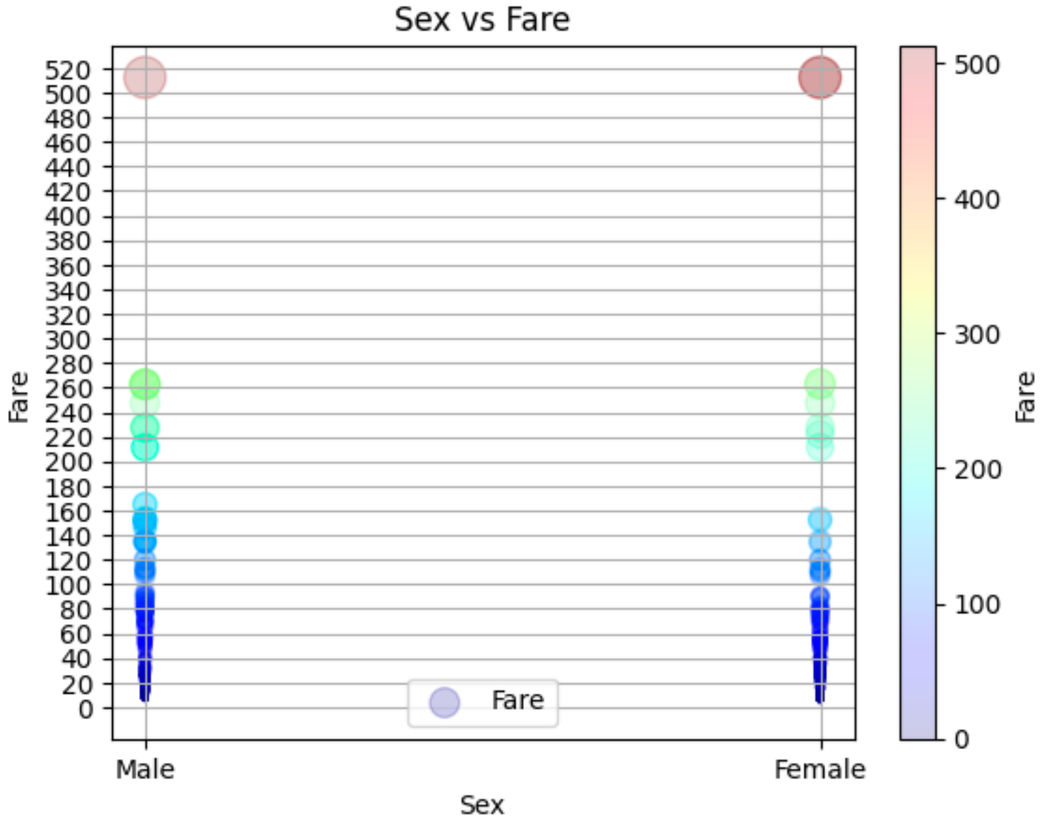
This suggests a trend where 1st class tickets originated from Cherbourg, 2nd class from Southampton, and 3rd class from Queenstown."



Distribution

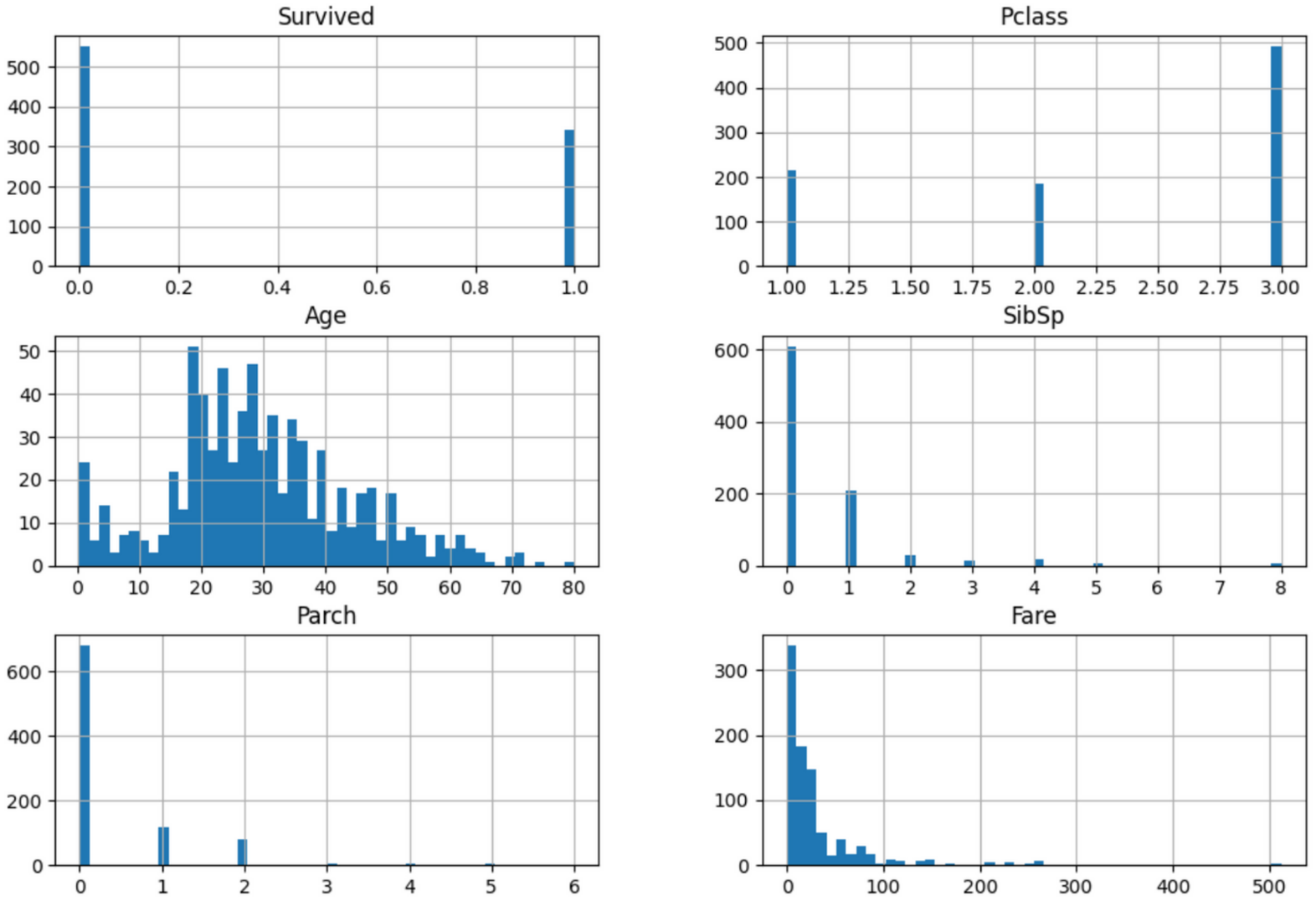
Analyzing Sex vs Fare

This graph illustrates fare prices based on the sex. We found no outliers or any insights worth nothing except that prices were sold evenly for men and women.



Distribution

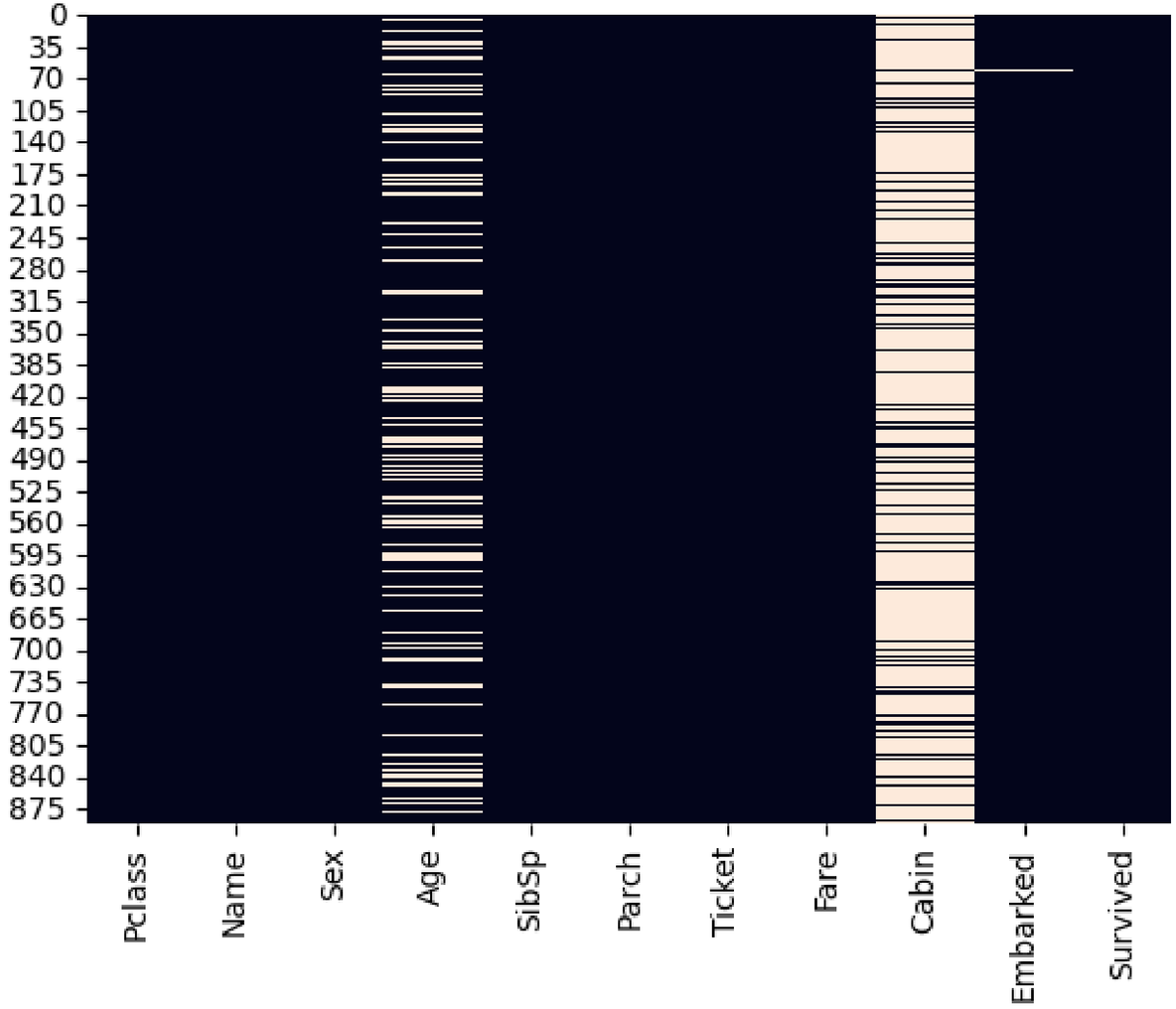
Numeric Features (6 / 11)



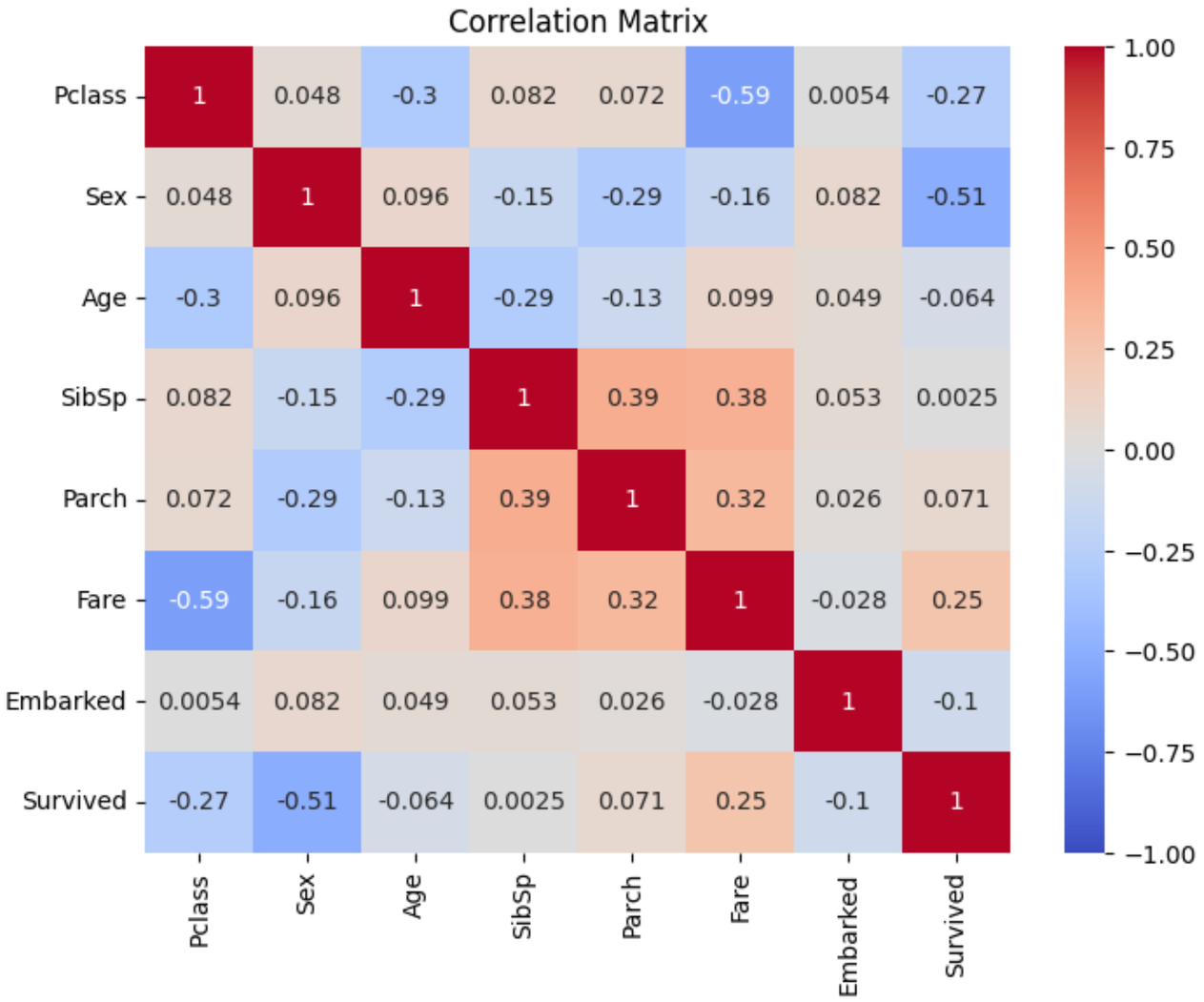
1. Exploration and Data Processing

Missing Values

Pclass has 0 missing values
Name has 0 missing values
Sex has 0 missing values
Age has 177 missing values
SibSp has 0 missing values
Parch has 0 missing values
Ticket has 0 missing values
Fare has 0 missing values
Cabin has 687 missing values
Embarked has 2 missing values
Survived has 0 missing values



Correlation Analysis

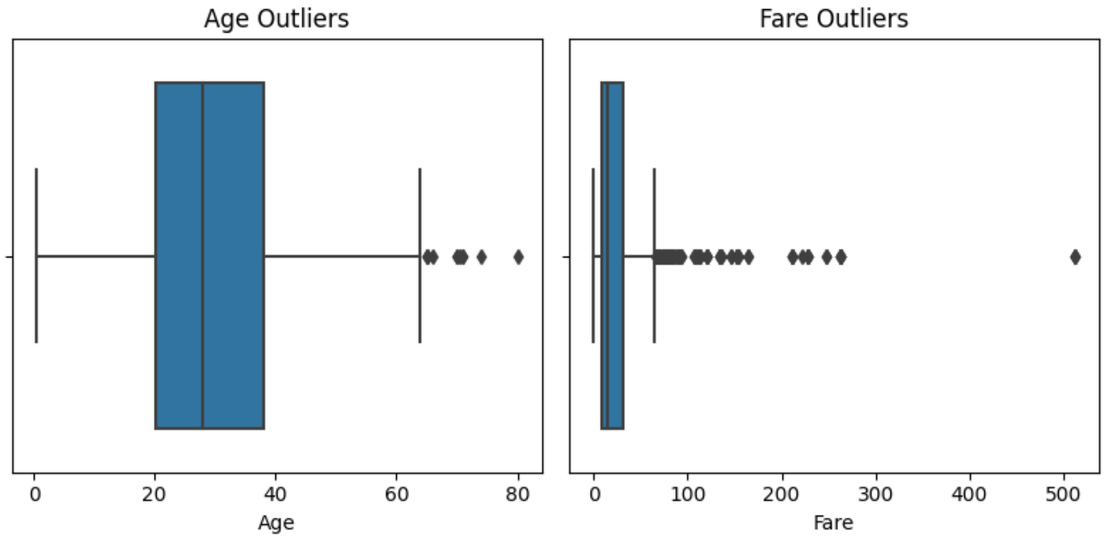


The values that are most related to the target variable are Fare, Parch and SibSp.

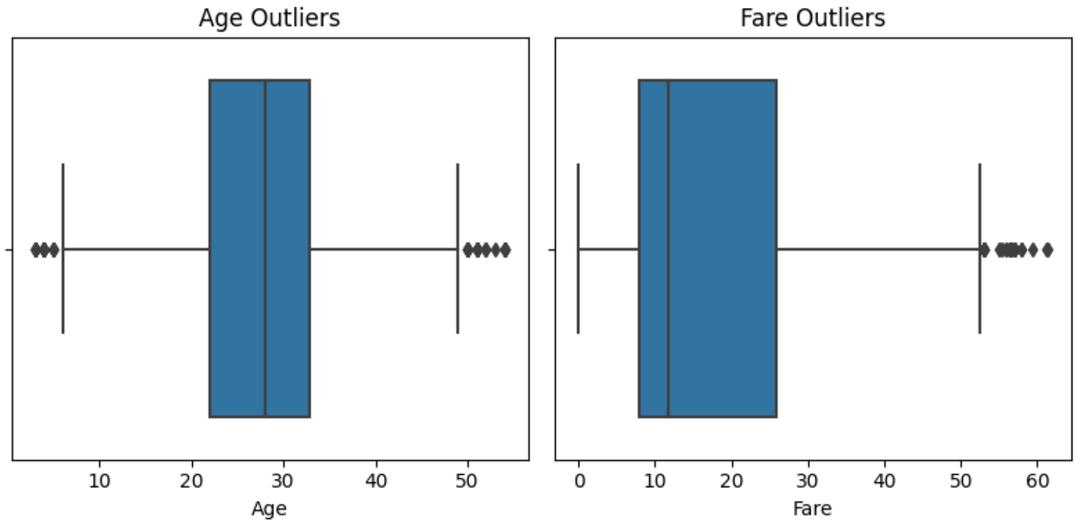
Data Transformation

The obtained results by using the IQR Method to reduce outliers indicate that there was some data that was not useful for training the ML models, therefore it was removed using an interquartil range of 90% to 10% for high limit and low limit respectively.

Before

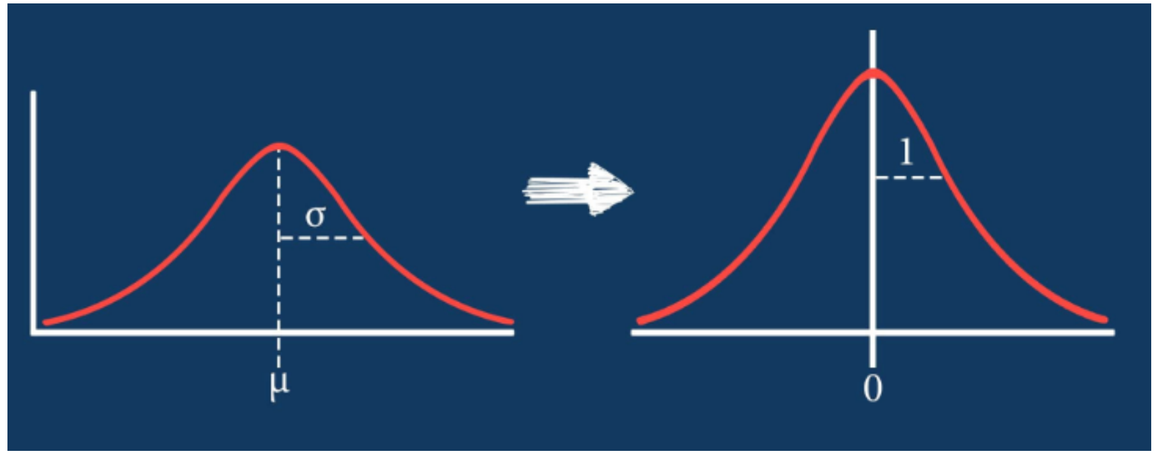


After



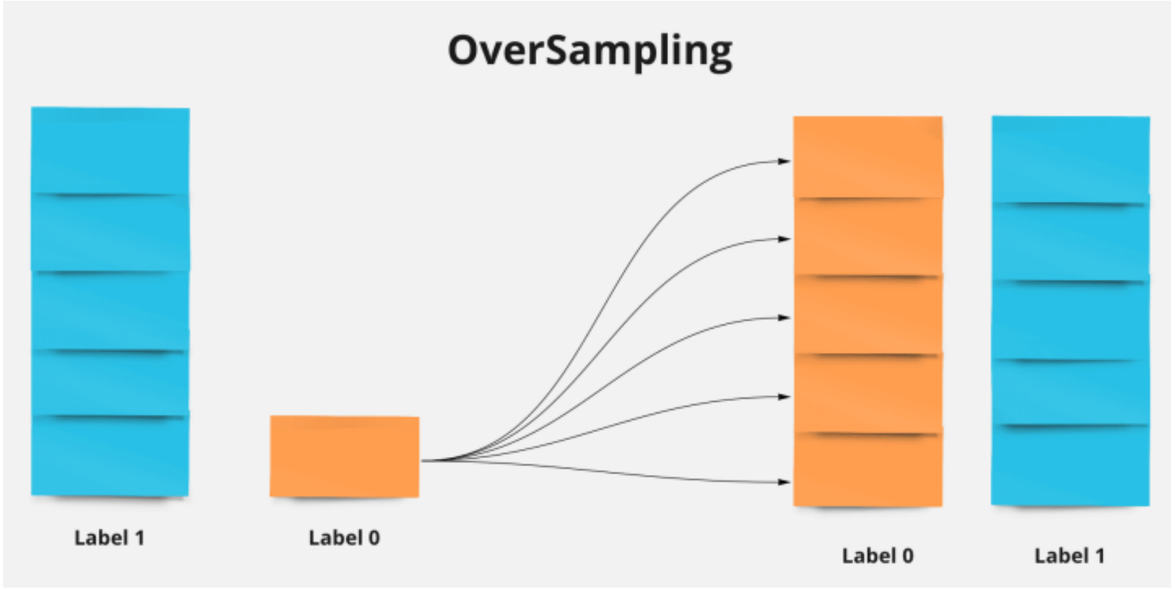
Data Scaling

To ensure optimal performance of our machine learning algorithms, especially those that are distance-based such as k-nearest neighbors and support vector machines, we apply a scaling technique.



Handling imbalanced datasets

Our dataset is imbalanced. Techniques that solve this exist, but sometimes they can be detrimental and it's worth exploring both to determine the most effective approach.



2. Classification

Machine Learning Models

1. Logistic Regression
2. Random Forest Classifier
3. Support Vector Machine
4. K Nearest Neighbors

Assessment Metrics

Unbalanced | Preprocessed

Métrica / Clasificador	Random Forest	Logistic Regression	Support Vector Machine	KNN
Accuracy	0.98 | 0.94	0.80 | 0.81	0.84 | 0.84	0.83 | 0.83
Precision	0 | 0.98 | 0.94 1 | 0.98 | 0.95	0 | 0.82 | 0.83 1 | 0.76 | 0.77	0 | 0.84 | 0.84 1 | 0.83 | 0.82	0 | 0.82 | 0.83 1 | 0.84 | 0.84
Recall	0 | 0.99 | 0.97 1 | 0.96 | 0.90	0 | 0.86 | 0.87 1 | 0.70 | 0.71	0 | 0.91 | 0.90 1 | 0.72 | 0.73	0 | 0.92 | 0.92 1 | 0.68 | 0.69
F1 Score	0 | 0.98 | 0.94 1 | 0.97 | 0.95	0 | 0.84 | 0.85 1 | 0.73 | 0.74	0 | 0.87 | 0.87 1 | 0.77 | 0.77	0 | 0.87 | 0.87 1 | 0.75 | 0.76
Kaggle	0.75598 | 0.7344	0.7655 | 0.77511	0.77511 | 0.7829	0.75837 | 0.72248

Assessment Metrics | Unbalanced Dataset

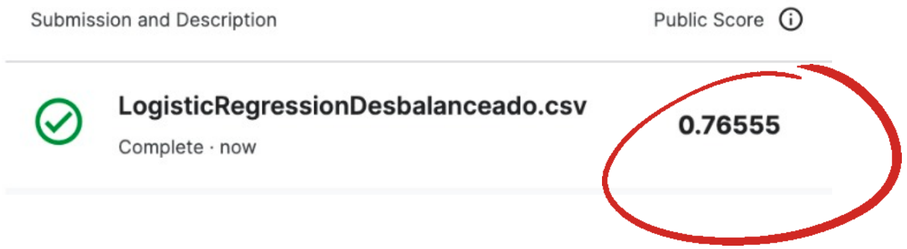
LogisticRegression

```
LogisticRegression()
```

Colab

	precision	recall	f1-score	support
0	0.82	0.86	0.84	549
1	0.76	0.70	0.73	342
accuracy			0.80	891

Kaggle



KNeighborsClassifier

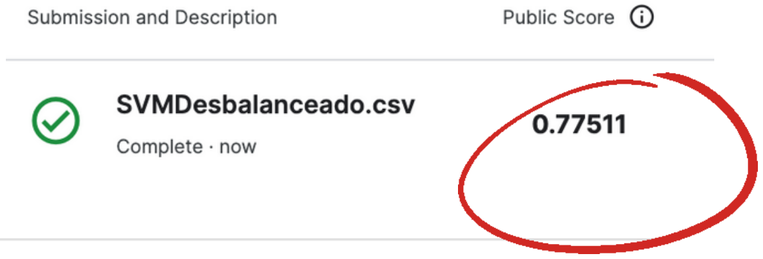
```
KNeighborsClassifier(n_neighbors=3)
```

	precision	recall	f1-score	support
0	0.82	0.92	0.87	549
1	0.84	0.68	0.75	342
accuracy			0.83	891

SVC

```
SVC(random_state=42)
```

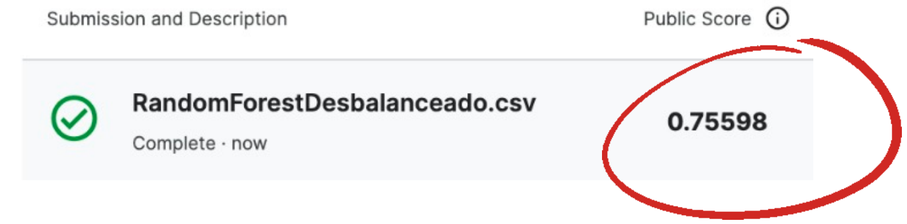
	precision	recall	f1-score	support
0	0.84	0.91	0.87	549
1	0.83	0.72	0.77	342
accuracy			0.84	891



RandomForestClassifier

```
RandomForestClassifier(random_state=15)
```

	precision	recall	f1-score	support
0	0.98	0.99	0.98	549
1	0.98	0.96	0.97	342
accuracy			0.98	891



Assessment Metrics

Balanced Dataset and preprocessed

Colab

LogisticRegression

```
LogisticRegression()
```

	precision	recall	f1-score	support
0	0.83	0.87	0.85	549
1	0.77	0.71	0.74	342
accuracy			0.81	891

RandomForestClassifier

```
RandomForestClassifier(random_state=15)
```

	precision	recall	f1-score	support
0	0.94	0.97	0.95	549
1	0.95	0.90	0.92	342
accuracy			0.94	891

KNeighborsClassifier

```
KNeighborsClassifier(n_neighbors=3)
```

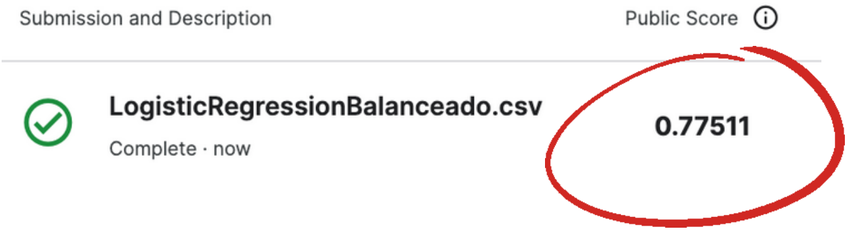
	precision	recall	f1-score	support
0	0.83	0.92	0.87	549
1	0.84	0.69	0.76	342
accuracy			0.83	891

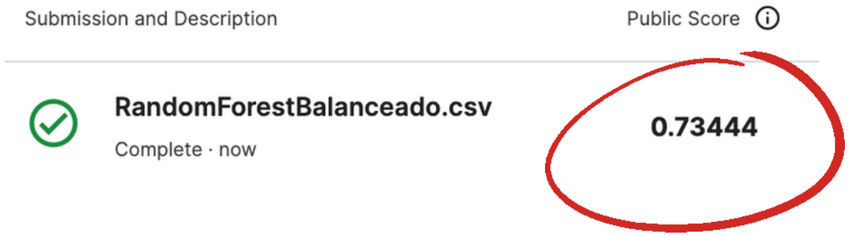
SVC

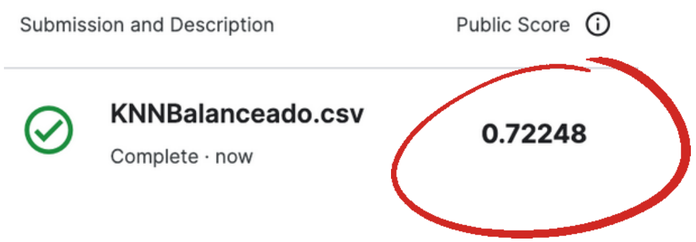
```
SVC(random_state=42)
```

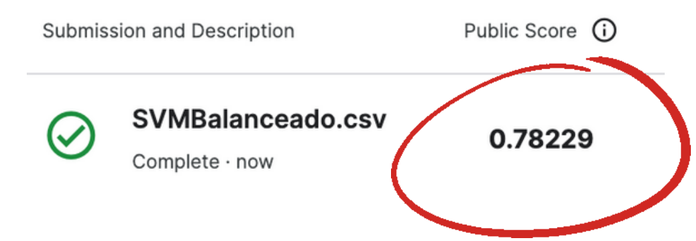
	precision	recall	f1-score	support
0	0.84	0.90	0.87	549
1	0.82	0.73	0.77	342
accuracy			0.84	891

Kaggle









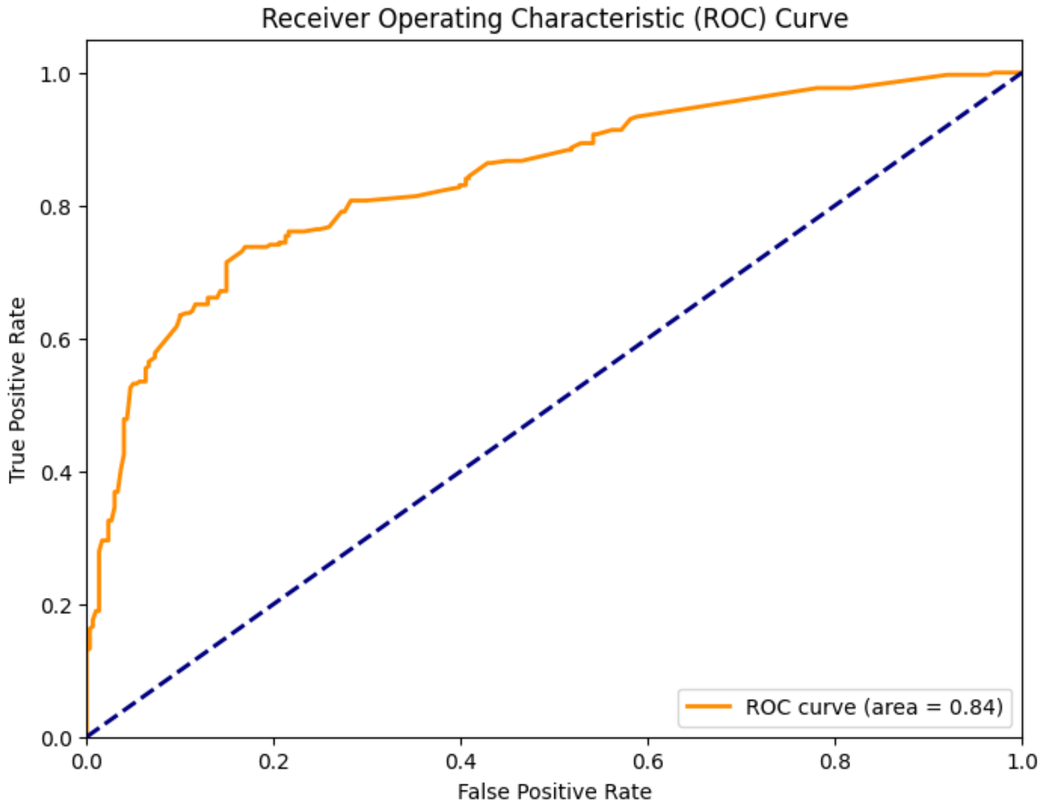
2. Classification

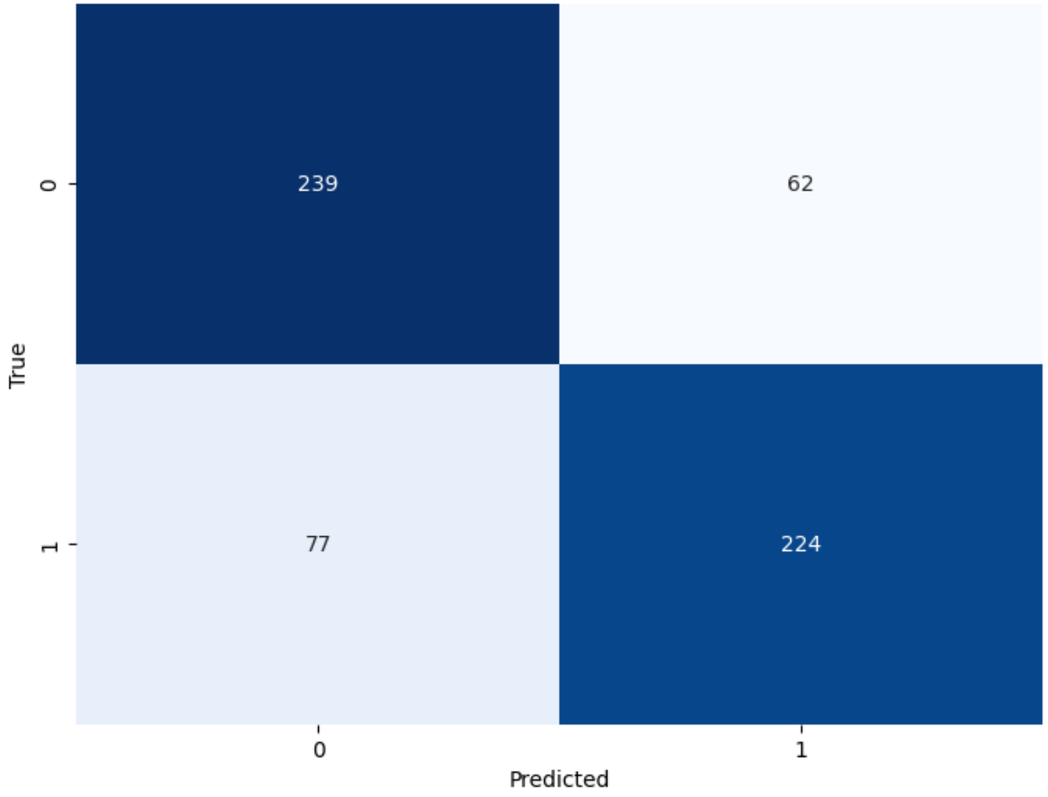
Assessment Metrics | K - Cross Validation

K-Cross Validation p/modelo	Resultados
Logistic Regression	Cross-validation scores: [*0.75609756*, *0.58536585*, *0.775*, *0.725*, *0.825*, *0.625*, *0.775*, *0.775*, *0.825*, *0.875*, *0.725*, *0.75*, *0.825*, *0.825*, *0.775*] Logistic Regression Mean Accuracy: **0.76**
K Nearest Neighbors	KNN Cross-validation scores [*0.70731707*, *0.6097561*, *0.775*, *0.775*, *0.825*, *0.775*, *0.75*, *0.775*, *0.8*, *0.75*, *0.775*, *0.825*, *0.8*, *0.8*, *0.725*] KNN Mean Accuracy: **0.76**
Support Vector Machine	Cross validation scores [*0.70731707*, *0.65853659*, *0.75*, *0.8*, *0.825*, *0.7*, *0.85*, *0.8*, *0.775*, *0.85*, *0.75*, *0.825*, *0.825*, *0.825*, *0.775*] SVM Mean Accuracy: **0.78**
Random Forest Classifier	Cross validation scores Random Forest: [*0.70731707*, *0.65853659*, *0.825*, *0.85*, *0.775*, *0.75*, *0.775*, *0.725*, *0.85*, *0.85*, *0.825*, *0.9*, *0.775*, *0.85*, *0.8*] Random Forest Mean accuracy **0.79**

Assessment Metrics

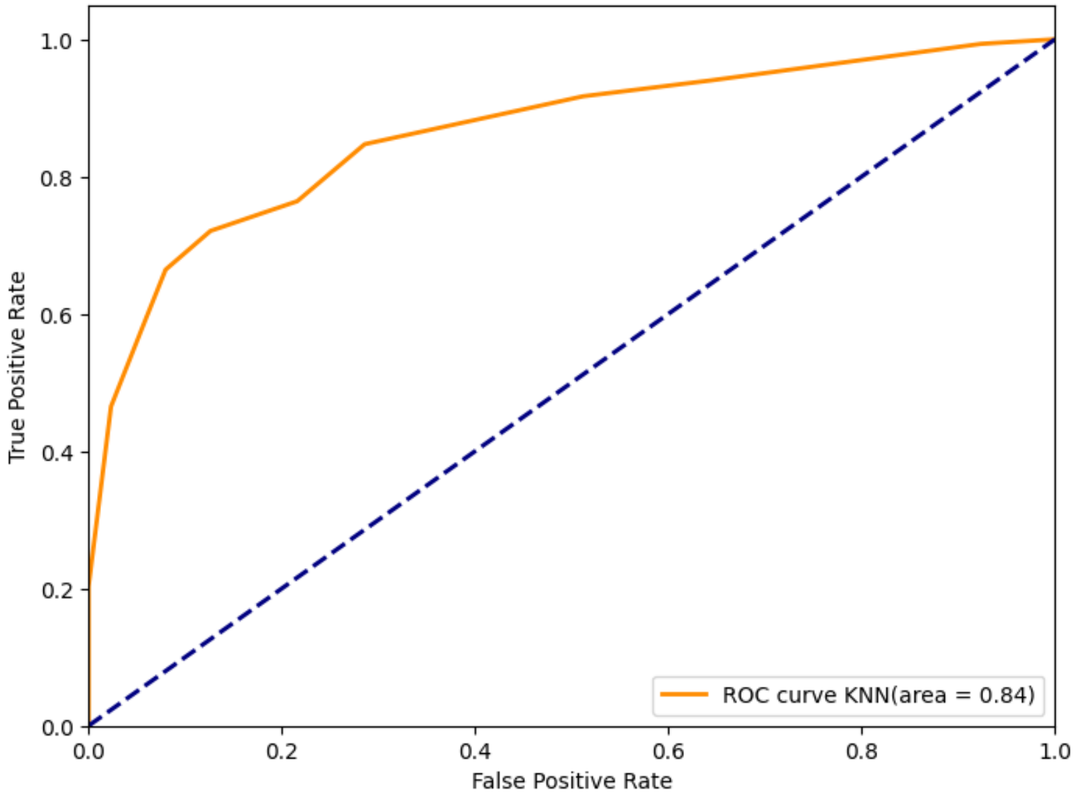
```
▼ LogisticRegression
LogisticRegression()
```

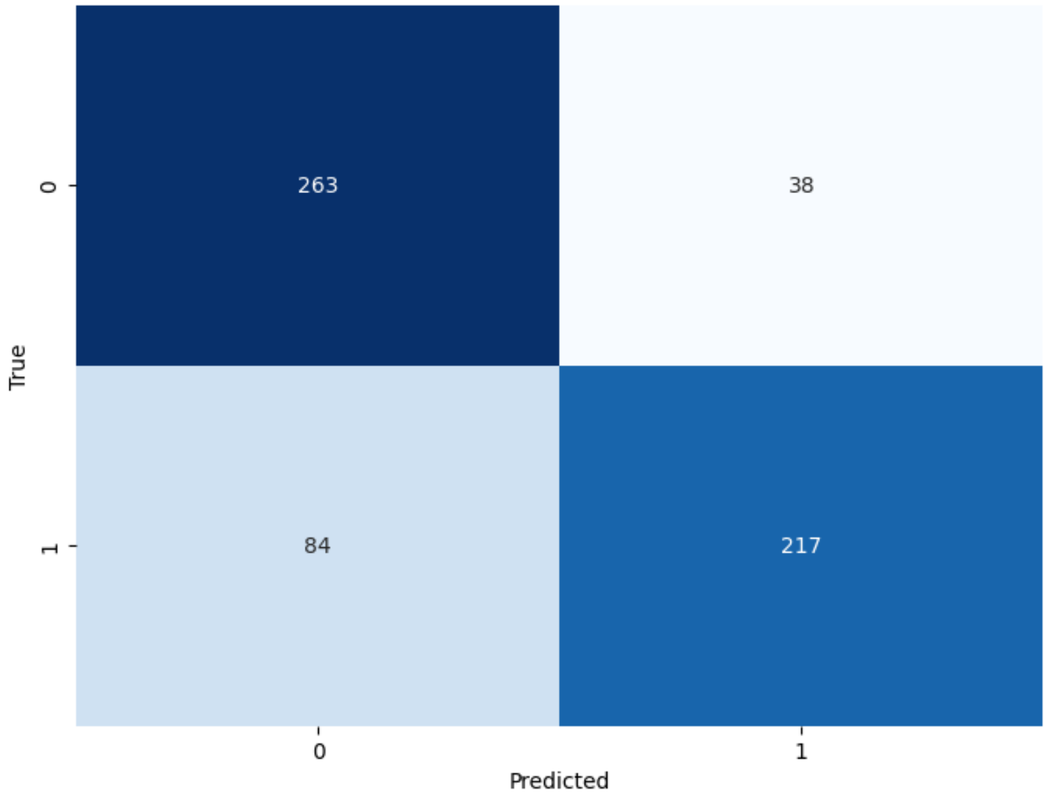




Assessment Metrics

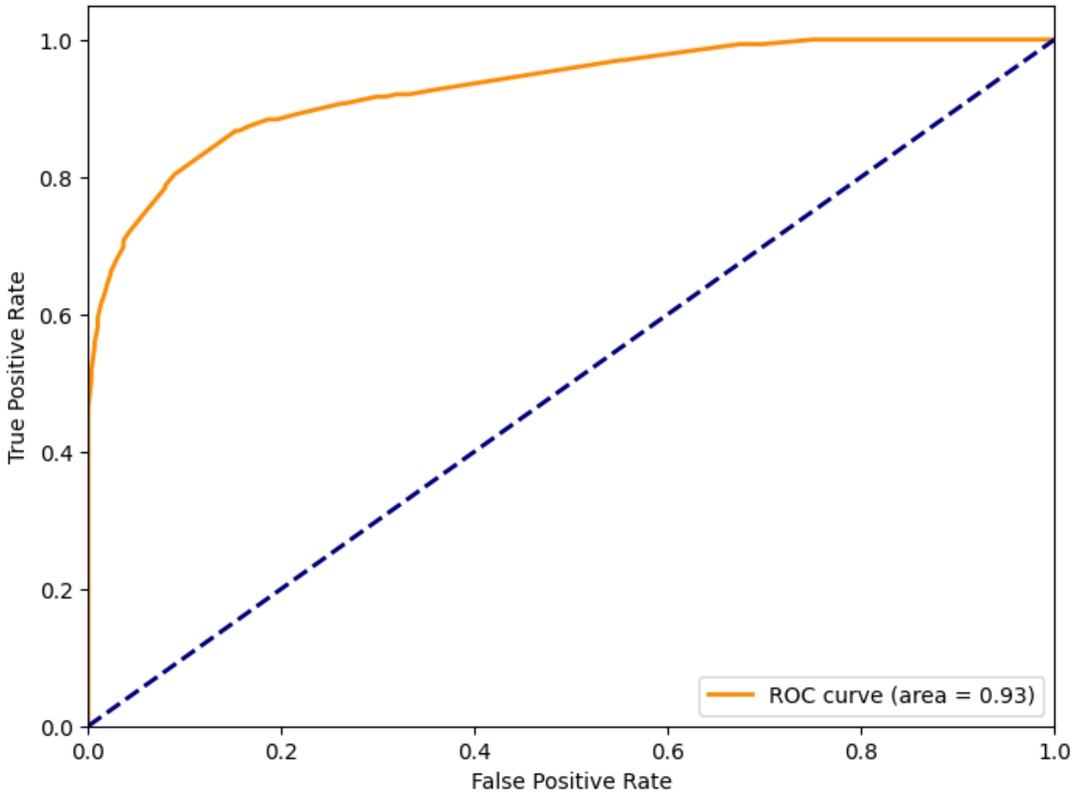
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

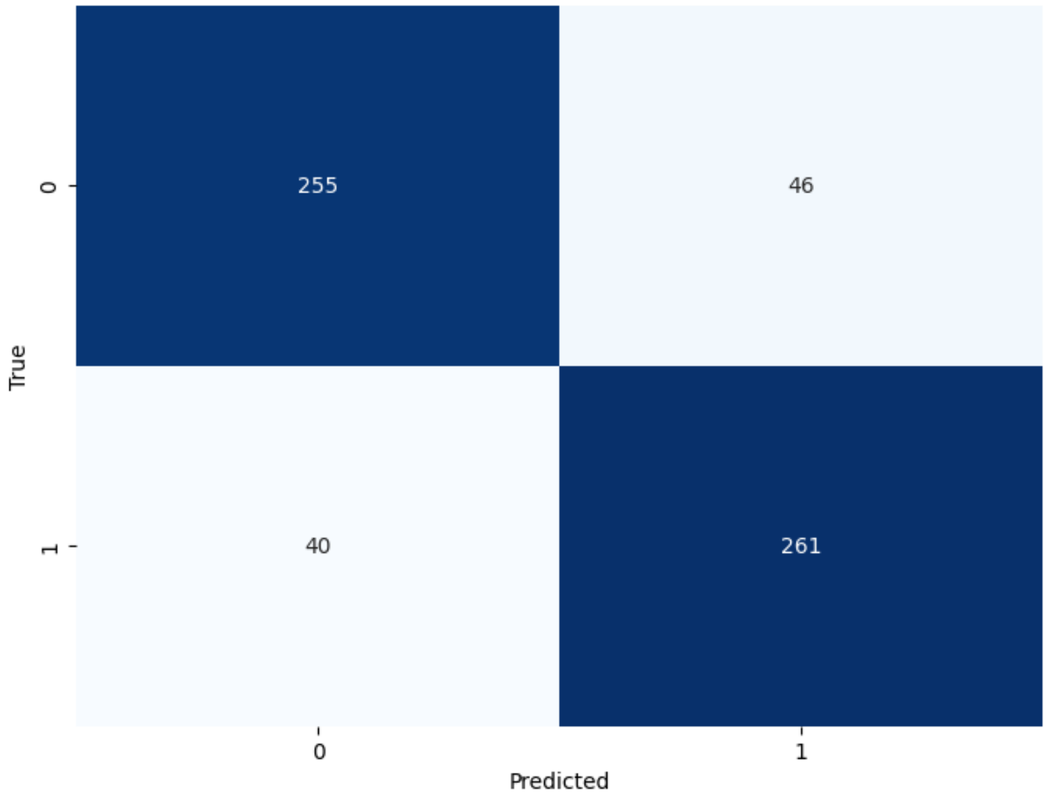




Assessment Metrics

```
▼ RandomForestClassifier
RandomForestClassifier(random_state=15)
```





Assessment Metrics

```
▼ SVC
SVC(random_state=42)
```