Titanic- data analysis

Team 1

Delgado Rios, Leonardo - A00827915
Hure, Marin Joseph - A01762723
Sánchez Pérez, Raúl Andrés - A01367635
Barredez Rios, Carlos Andres - A01653183

Tecnológico de Monterrey

Can we determine the key factors influencing survival?

Date: April 15, 1912

Event: Sinking of the Titanic with significant loss of life.

Dataset: Passenger details including age, fare, class, gender, etc.

Objective: To uncover the underlying social and economic dynamics of the tragedy

Data Science:

A discipline integrating mathematics, statistics, and computer science to interpret, visualize, and extract insights from large volumes of data.

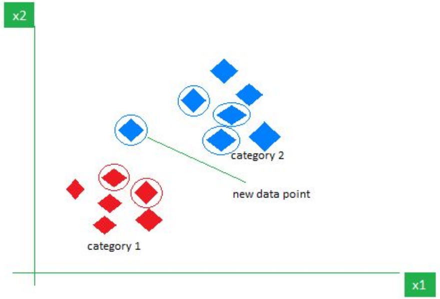
Machine Learning (ML)

A branch of artificial intelligence focused on developing algorithms that allow computers to learn from and make predictions or decisions based on data.

ML Algorithms

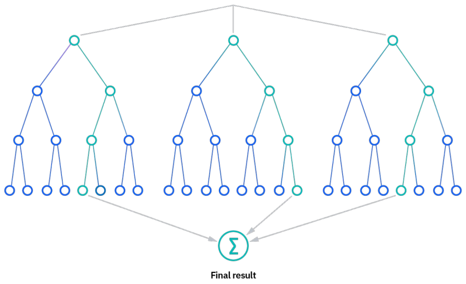
K-Nearest Neighbors

Definition: Based on the majority votes of its K closest neighbors.
Advantages: No training phase required and intuitively simple.
Disadvantages: Computationally expensive and sensitive to irrelevant features.



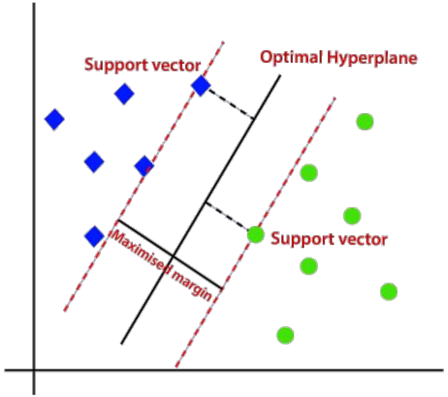
Random Forest

Definition: Builds multiple decision trees during training and outputs the class that is the mode of the classes of the individual trees.
Advantages: Handles large datasets with higher dimensionality, provides estimates of feature importance, and prevents overfitting.
Disadvantages: Can be slow in generating predictions due to the number of trees.



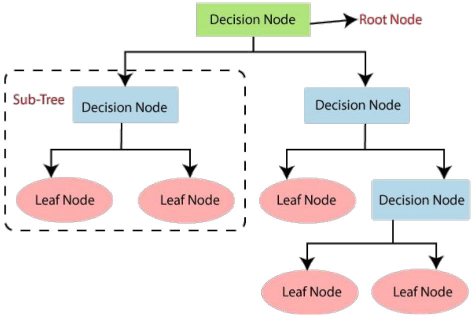
Support Vector Machines

Definition: Identifying the optimal hyperplane that best divides a dataset into classes.
Advantages: Effective in high-dimensional spaces.
Disadvantages: Not suitable for very large datasets.



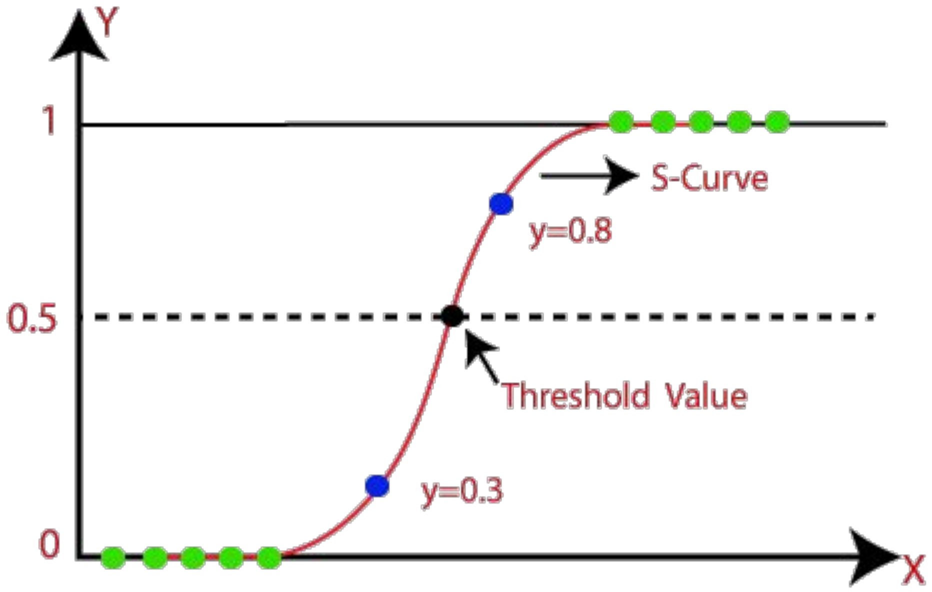
Decision Tree

Definition: A flowchart-like structure where each internal node represents a feature(or attribute), each leaf node represents a class label, and branches represent conjunctions of features leading to class labels.
Advantages: Simple to understand and interpret, requires little data preparation.
Disadvantages: Can easily become complex and prone to overfitting.



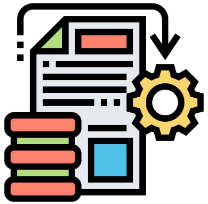
Logistic Regression

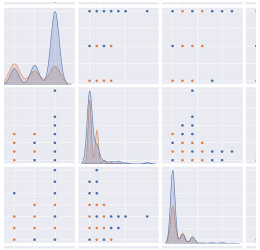
Definition: A statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome.
Advantages: Works well for linearly separable data.
Disadvantages: Assumes a linear boundary between classes and might not be suitable for non-linear data.



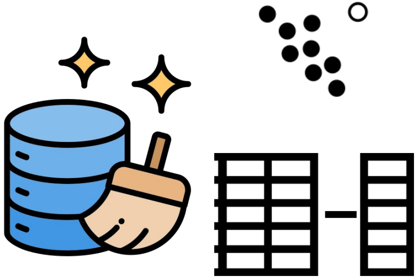
Results

Data Pre-processing

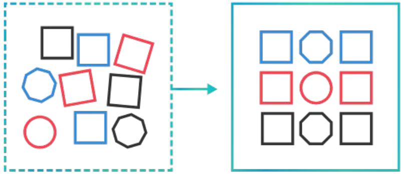




Data Cleaning

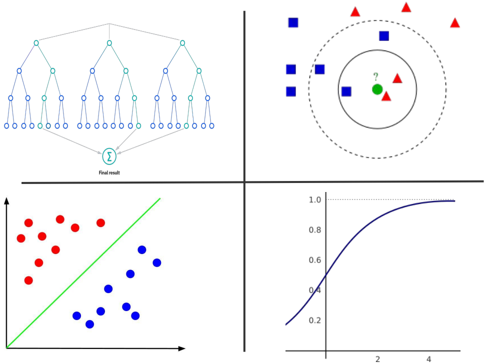


Data Transformation

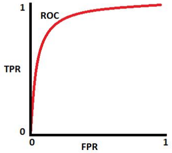


Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_Q	Embarked_S
0	0	1	0	0	1
1	0	0	1	0	0

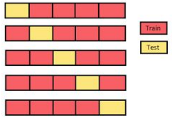
Algorithm Selection



Model Evaluation



MSE
MAE



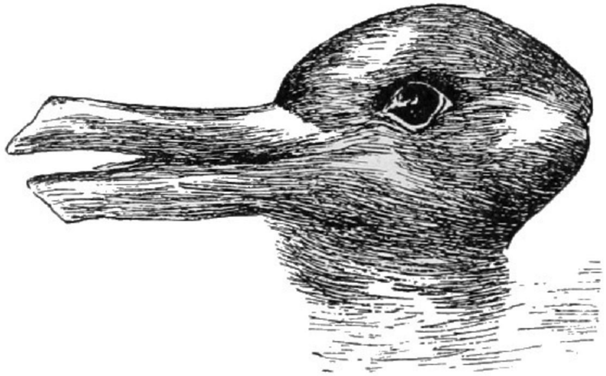
Final Predictions

Submission and Description	Public Score
SVCPredictions.csv Complete · now	0.78229
LogisticRegressionPredictions.csv Complete · 13s ago	0.77033
DecisionTreePredictions.csv Complete · 25s ago	0.76315
kNNPredictions.csv Complete · 44s ago	0.73205
RandomForestPredictions.csv Complete · 1m ago	0.71291

```
<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   Name         891 non-null    object
 2   Sex          891 non-null    object
 3   Age          714 non-null    float64
 4   Pclass       891 non-null    int64
 5   SibSp        891 non-null    int64
 6   Parch        891 non-null    int64
 7   Ticket       891 non-null    object
 8   Fare         891 non-null    float64
 9   Cabin        204 non-null    object
 10  Embarked     889 non-null    object
 11  Survived     891 non-null    int64
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Importance of Data Pre-Processing

- ❖ Understanding the structure of our data.
- ❖ Find human or computer errors.
- ❖ Improves accuracy and reliability of our final model.

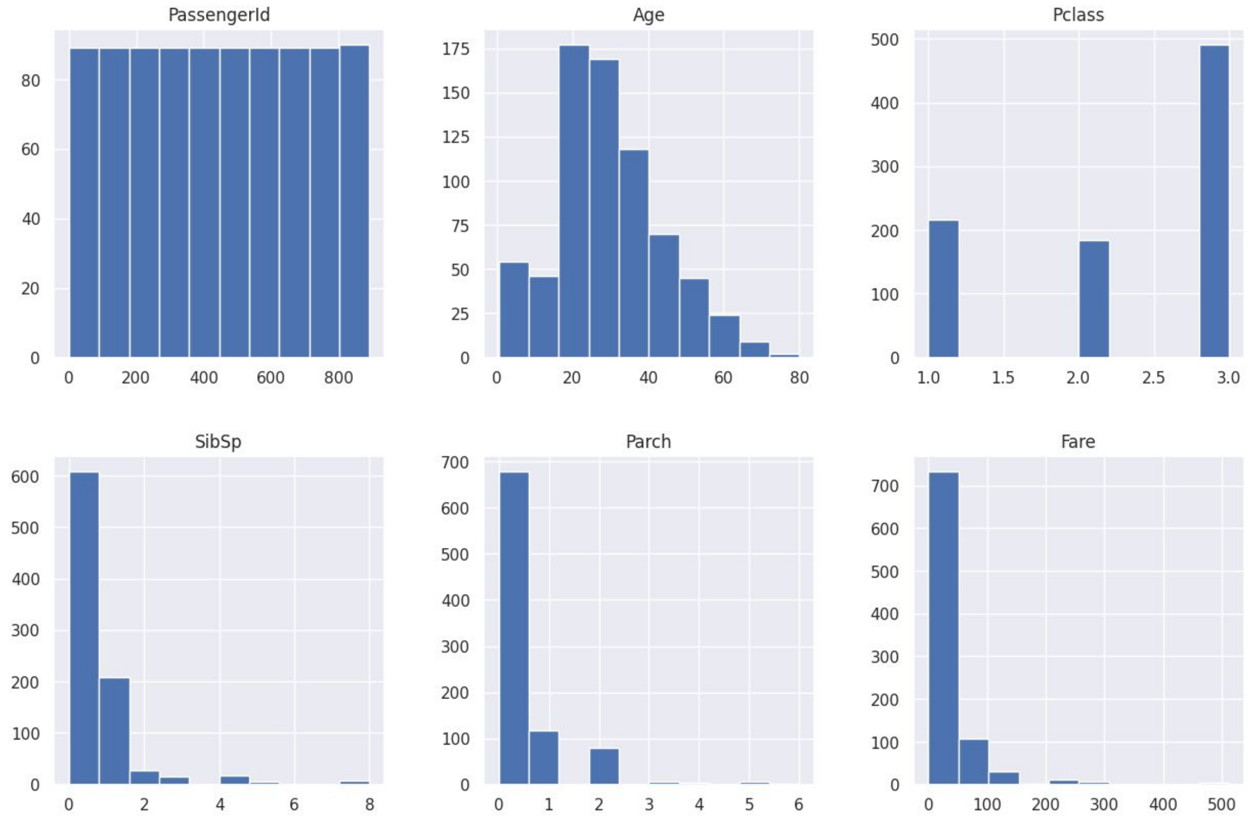


	count	mean	std	min	25%	50%	75%	max
PassengerId	891.0	446.000000	257.353842	1.00	223.5000	446.0000	668.5	891.0000
Age	714.0	29.699118	14.526497	0.42	20.1250	28.0000	38.0	80.0000
Pclass	891.0	2.308642	0.836071	1.00	2.0000	3.0000	3.0	3.0000
SibSp	891.0	0.523008	1.102743	0.00	0.0000	0.0000	1.0	8.0000
Parch	891.0	0.381594	0.806057	0.00	0.0000	0.0000	0.0	6.0000
Fare	891.0	32.204208	49.693429	0.00	7.9104	14.4542	31.0	512.3292
Survived	891.0	0.383838	0.486592	0.00	0.0000	0.0000	1.0	1.0000

Describing our data can help us:

- Identify data types
- Detect missing data
- Detect the presence of outliers
- Analyze distribution
- Assess quality and consistency
- Identify data characteristics

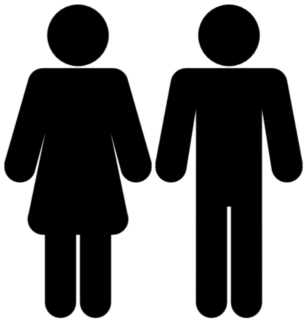
Data distribution of dataset



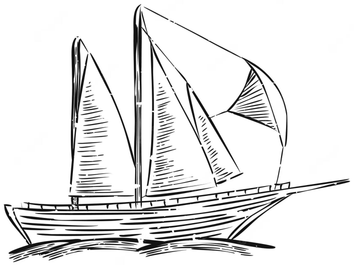
Data Pre-processing (Cont.)

Feature value counts

Class 1: **216**
Class 2: **184**
Class 3: **491**



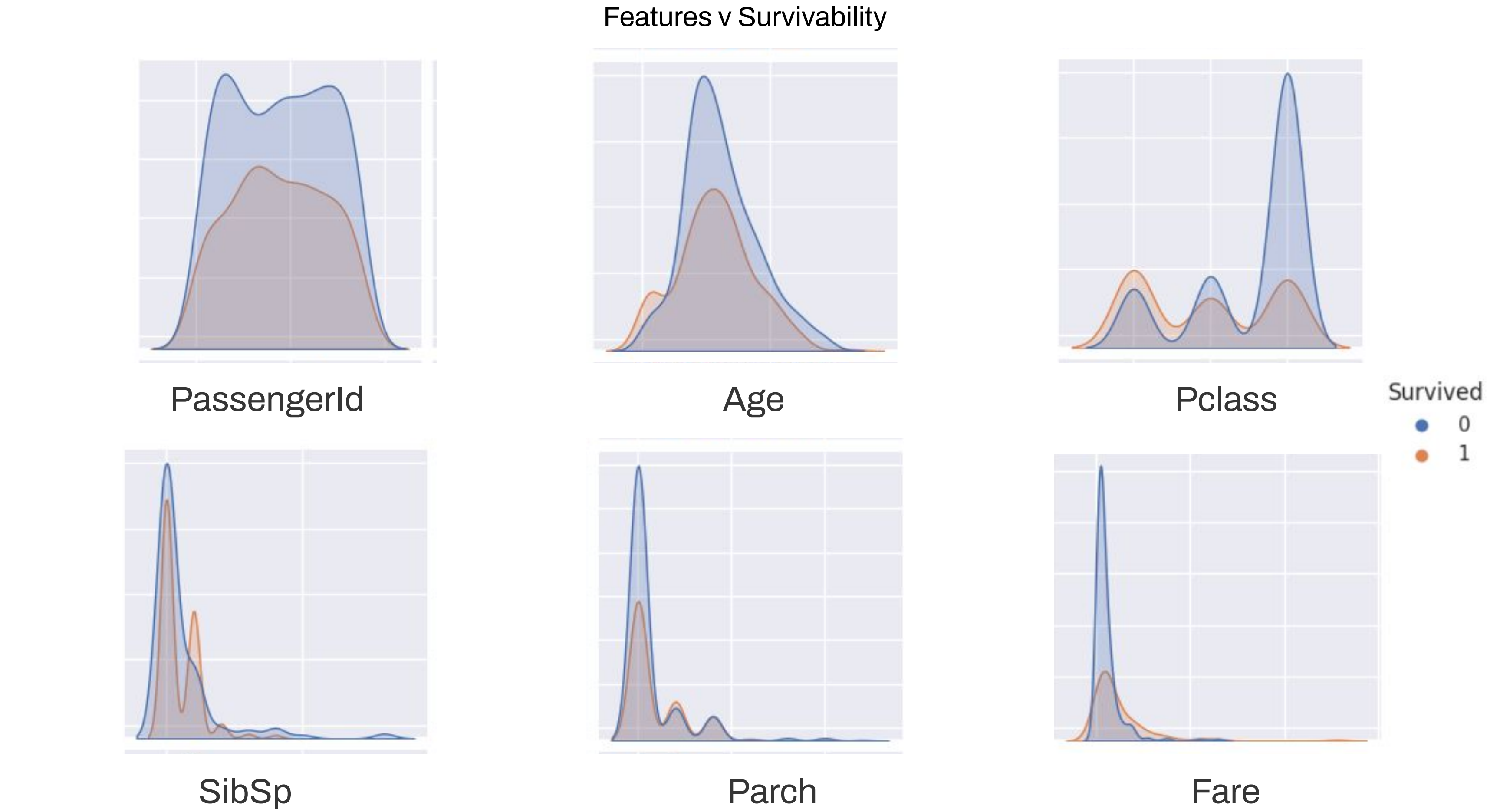
Male: **577**
Female: **314**



Port S: **644**
Port C: **168**
Port Q: **77**

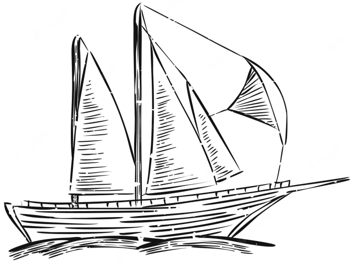
Childs: **83**
Adults: **808**

Data Pre-processing (Cont.)

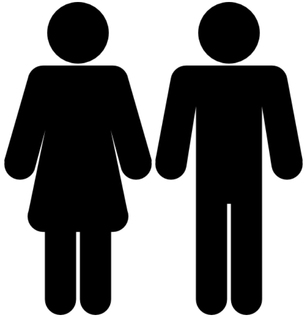


Data Pre-processing (Cont.)

Class 1: **62.96 %**
Class 2: **47.28 %**
Class 3: **24.24 %**



Port S: **55.35 %**
Port C: **38.96 %**
Port Q: **33.69 %**



Male: **18.89 %**
Female: **74.20 %**

isChild: **59.03 %**

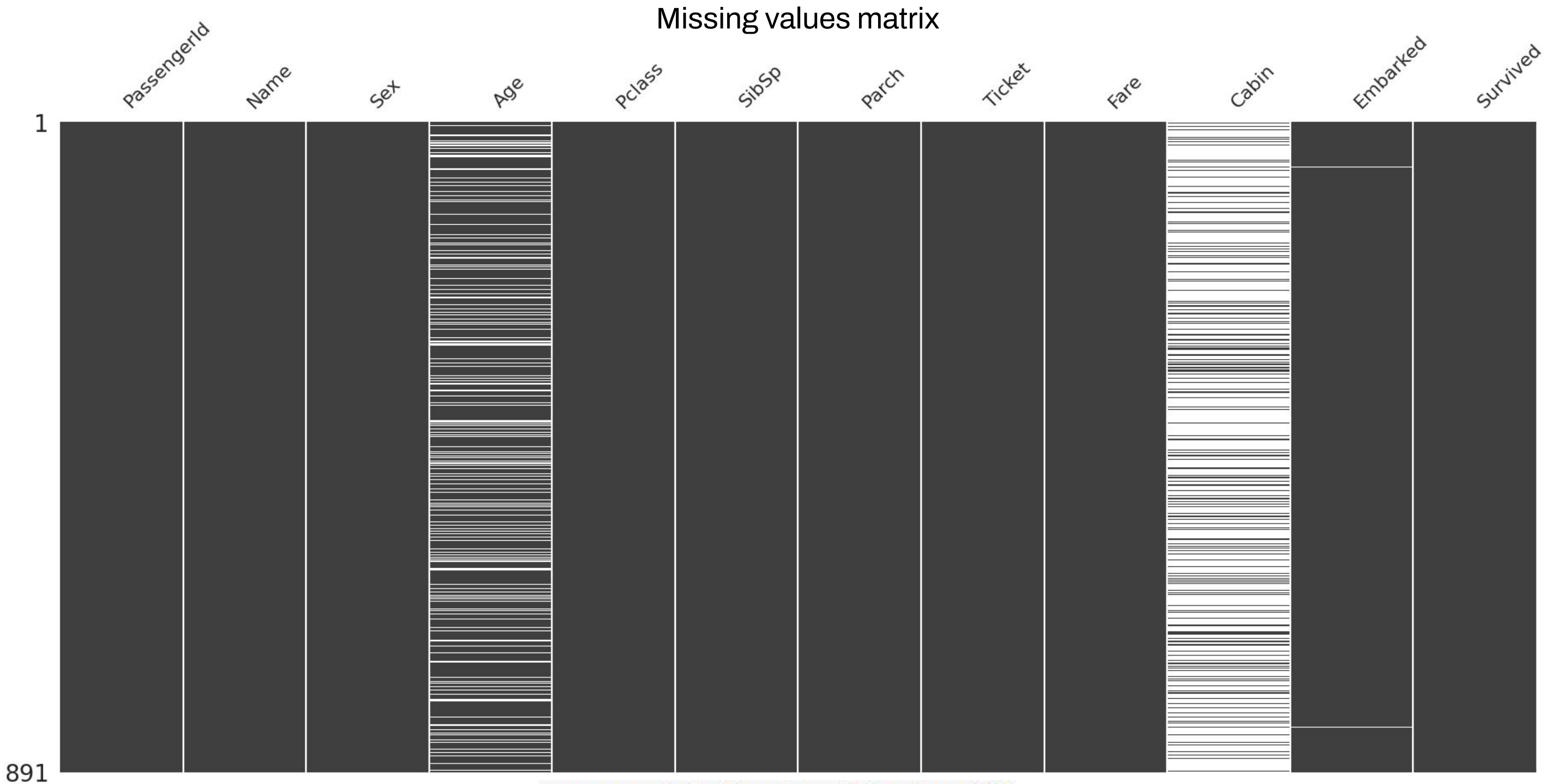
Overall Survival
Did: **38.38 %**
Did not: **61.62 %**

```
PassengerId      0
Name             0
Sex              0
Age            177
Pclass           0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
Survived         0
dtype: int64
```

```
train_titanic_df.isnull().sum()
```

Importance of Data Cleaning

- **Ensures Accuracy:**
  - Data cleaning ensures that the data is free from errors and inaccuracies, reflecting the real-world
- **Boosts Model Performance:**
  - Clean data leads to better machine learning model performance
- **Improves Quality:**
  - It enhances data quality for reliable analysis




```
msno.matrix(train_titanic_df)
```

Age & Embarked

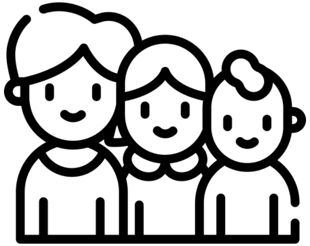
- ❖ We spotted some missing values on [Age] and [Embarked], we decided to fill these using median (numerical) , and mode (categorical) respectively

```
train_titanic_df['Age'].fillna(train_titanic_df['Age'].median(), inplace=True)
train_titanic_df['Embarked'].fillna(train_titanic_df['Embarked'].mode()[0], inplace=True)
```

We chose these methods because they showed the best performance

Thresholds Pclass:
(0.50, 4.50)
Potentially Outliers:
0

Thresholds Age:
(-6.69, 64.81)
Potentially Outliers:
11

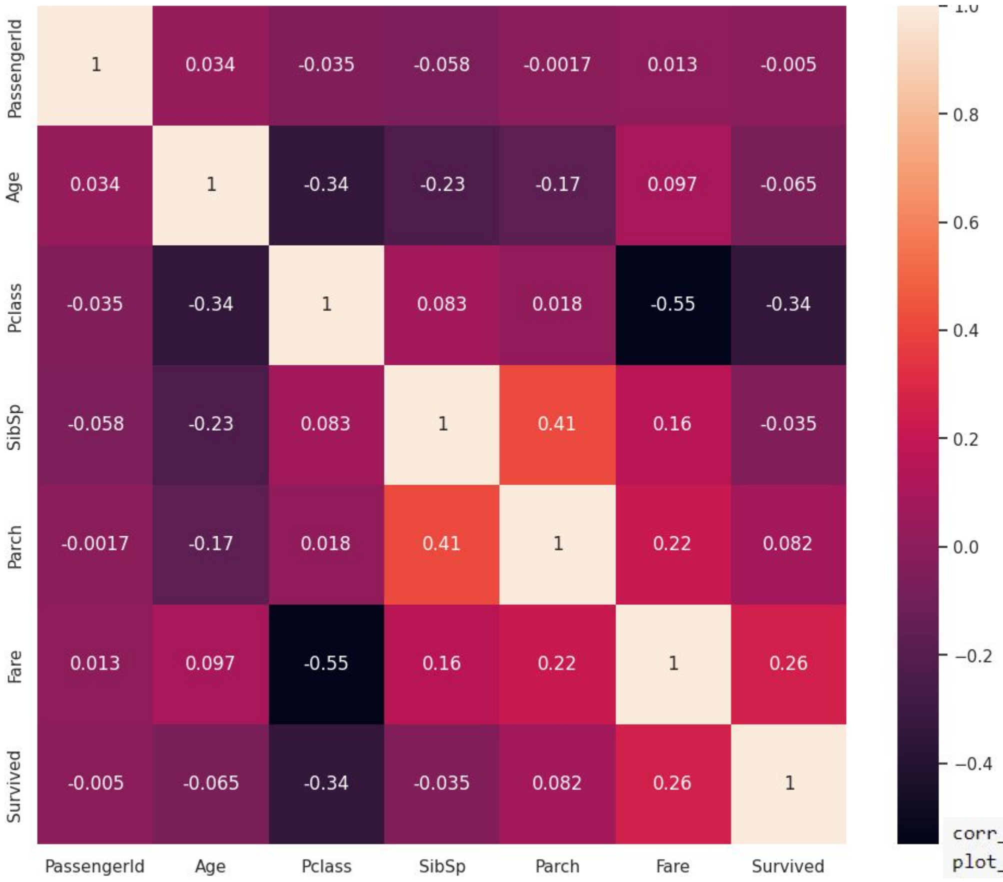


Thresholds SibSp:
(-1.50, 2.50)
Potentially Outliers:
46

Thresholds Parch:
(0.00, 0.00)
Potentially Outliers:
213

Thresholds Fare:
(-26.72, 65.63)
Potentially Outliers:
116

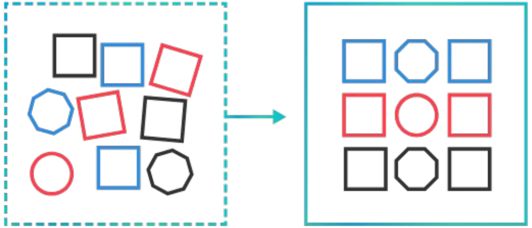
Correlation Matrix
Heatmap

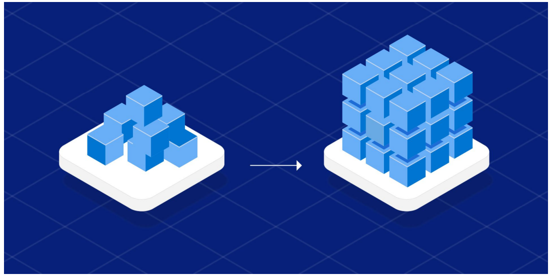



```
corr_matrix = train_titanic_df.corr()
plot_correlation_matrix(corr_matrix)
```

Importance of Data Transformation

- **Model Compatibility:**
  - Data transformation makes data compatible with machine learning models that require numerical input
- **Enhances Predictions:**
  - Effective encoding boosts model accuracy by including crucial categorical data
- **Enables Wider Applicability:**
  - Data transformation makes the data compatible with a wider range of algorithms and techniques, expanding the potential for analysis





Sex Feature & Drop

- ❖ We can identify a male or a female by defining isFemale to 1 (True), 'Sex' is dropped in pair with other features

```
# Transforming sex categories into numerical here only to appear in plotting charts.
train_titanic_df["isFemale"] = train_titanic_df.apply(isFemale, axis = 1)

# Drop the features that we are not going to use and review the transformation of Sex.
train_titanic_df = train_titanic_df.drop(["PassengerId", "Name", "Sex", "Ticket", "Cabin"], axis = 1)
train_titanic_df["isFemale"].value_counts()
```

```
0    577
1    314
Name: isFemale, dtype: int64
```

Embarked and Pclass feature

❖ Both Embarked and Pclass were categorical data, in order to be applied to our ML algorithms these needed to become numerical, we used “pd.get_dummies()”

❖ This function uses **One-hot encoding** a technique that creates new binary columns for each category. Each binary column represents the presence or absence of a specific category for each data point

```
train_titanic_df = pd.get_dummies(train_titanic_df, columns=["Pclass","Embarked"])
```

Pclass_1	Pclass_2	Pclass_3	Embarked_C	Embarked_Q	Embarked_S
0	0	1	0	0	1
1	0	0	1	0	0

Standardization

❖ For features **[Age, SibSp, Parch, Fare]**, we use a Standard Scaler. This process ensures that the mean of the standardized data is close to 0.

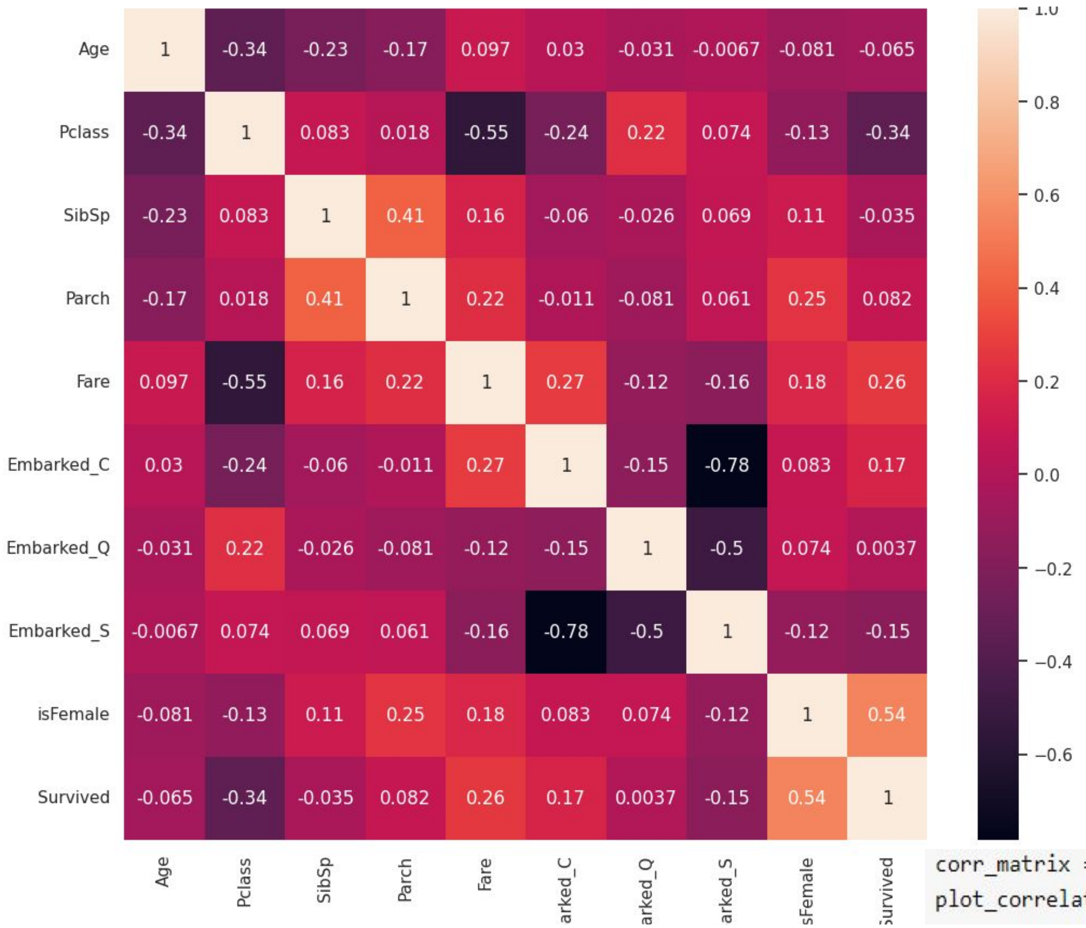
❖ Standardization is useful for algorithms that are sensitive to the scale of input features, as it brings all features to a common scale it prevents some features from dominating others

$$z = \frac{x - \mu}{\sigma}$$

$\mu =$ Mean
$\sigma =$ Standard Deviation

```
scaler = StandardScaler()
features = ["Age", "SibSp", "Parch", "Fare"]
train_titanic_df[features] = scaler.fit_transform(train_titanic_df[features])
```

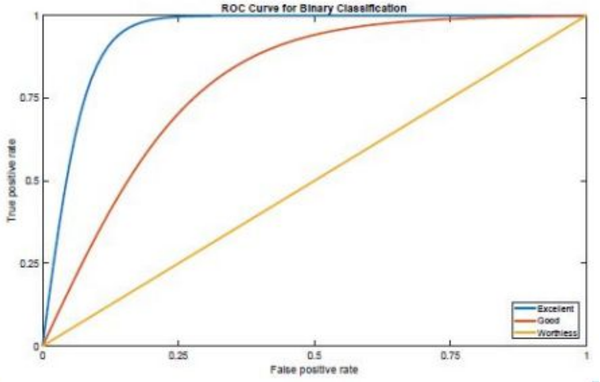
Correlation Matrix Heatmap




```
corr_matrix = train_titanic_df.corr()
plot_correlation_matrix(corr_matrix)
```

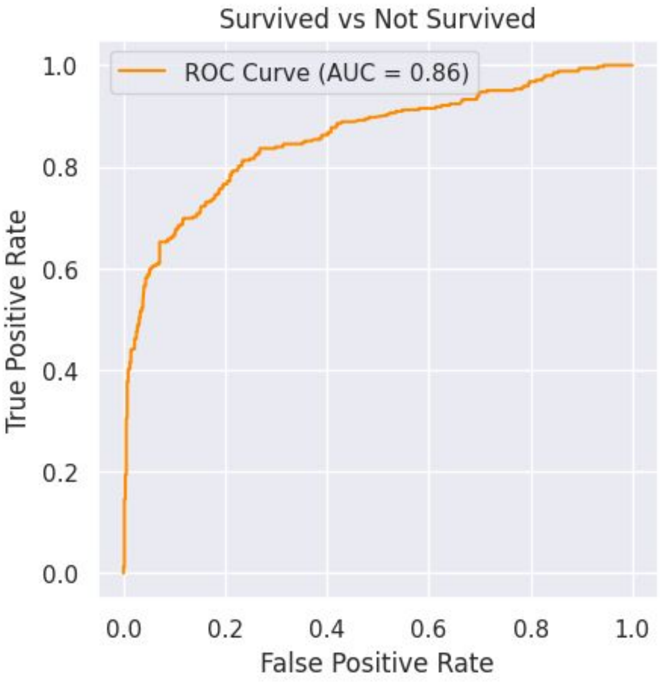
How we are measuring performance?

- Accuracy
- Precision
- Confusion Matrix
- ROC Curve and AUC

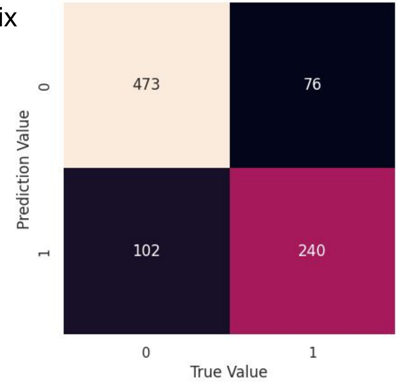


		Actual	
		Class1	Class 2
Predicted	Class 1	True Positive (T_p)	False Positive (F_p)
	Class 2	False Negative (F_n)	True Negative (T_n)

Logistic Regression Model



Confusion Matrix
TRAIN



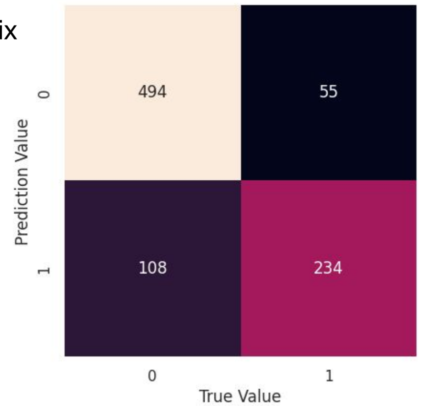

```
The selected parameters are {}
The precision on the training set using CV : 0.789
Binary Precision_score :  0.759493670886076
Weighted Precision_score :  0.7983827265500354
Average_precision_score :  0.6474561291350099
Binary f1_score :  0.729483282674772
Weighted f1_score :  0.7985881049786554
              precision    recall  f1-score   support

           0       0.86      0.82      0.84       575
           1       0.70      0.76      0.73       316

    accuracy                           0.80       891
   macro avg       0.78      0.79      0.79       891
weighted avg       0.80      0.80      0.80       891
```

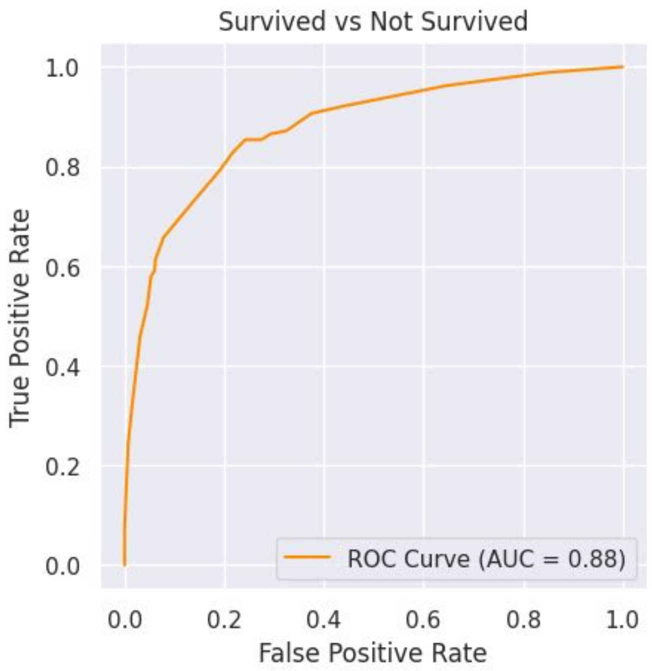
Cross-Validation
TEST

```
----------------- MEAN -----------------
Mean_test_score : 0.7890151277383717
Std_test_score : 0.019488362104751616
```

k-Nearest Neighbours Model

Confusion Matrix
TRAIN



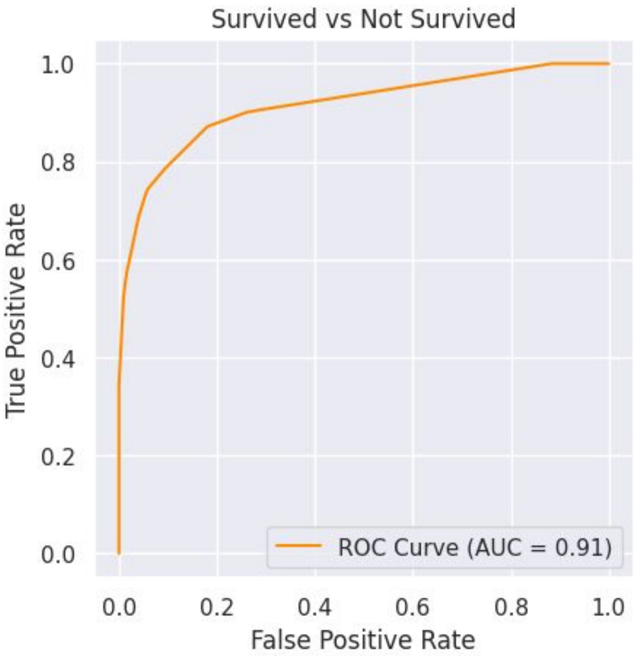



```
The selected parameters are {'n_neighbors': 25}
The precision on the training set using CV : 0.804
Binary Precision_score :  0.8096885813148789
Weighted Precision_score :  0.816410550457384
Average_precision_score :  0.6752095715854594
Binary f1_score :  0.7416798732171157
Weighted f1_score :  0.8135884851536233
              precision    recall  f1-score   support

           0       0.90      0.82      0.86       602
           1       0.68      0.81      0.74       289

    accuracy                           0.82       891
   macro avg       0.79      0.82      0.80       891
weighted avg       0.83      0.82      0.82       891
```

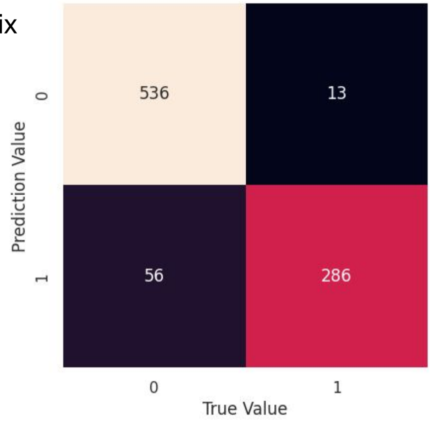
Cross-Validation
TEST

```
----------------- MEAN -----------------
Mean_test_score : 0.7914456091896303
Std_test_score : 0.024318084318998345
```

Decision Tree Model



Confusion Matrix
TRAIN



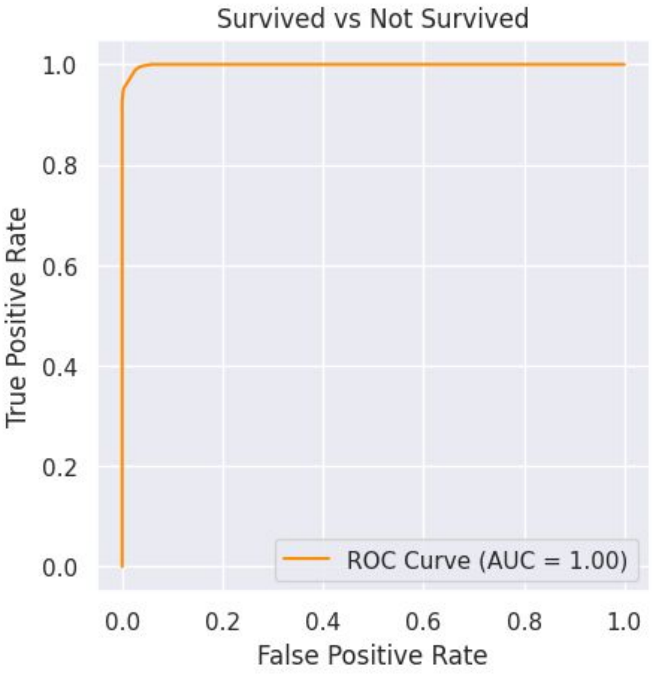

```
The selected parameters are {'max_depth': 10}
The precision on the training set using CV : 0.818
Binary Precision_score :  0.9565217391304348
Weighted Precision_score :  0.9250258163301641
Average_precision_score :  0.8627490259831985
Binary f1_score :  0.8923556942277692
Weighted f1_score :  0.9214206764430246
              precision    recall  f1-score   support

           0       0.98      0.91      0.94       592
           1       0.84      0.96      0.89       299

    accuracy                           0.92       891
   macro avg       0.91      0.93      0.92       891
weighted avg       0.93      0.92      0.92       891
```

Cross-Validation
TEST

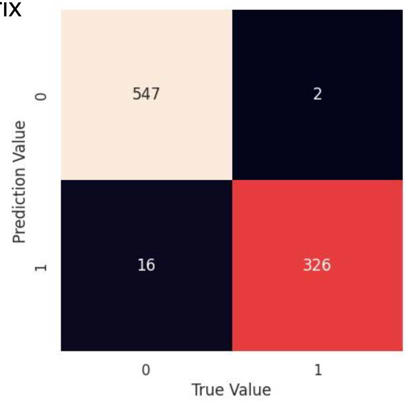
```
---------------- MEAN ----------------
Mean_test_score : 0.7971655891030067
Std_test_score : 0.027895771286105236
```

Model Evaluation (Cont.)

Random Forest Model



Confusion Matrix
TRAIN

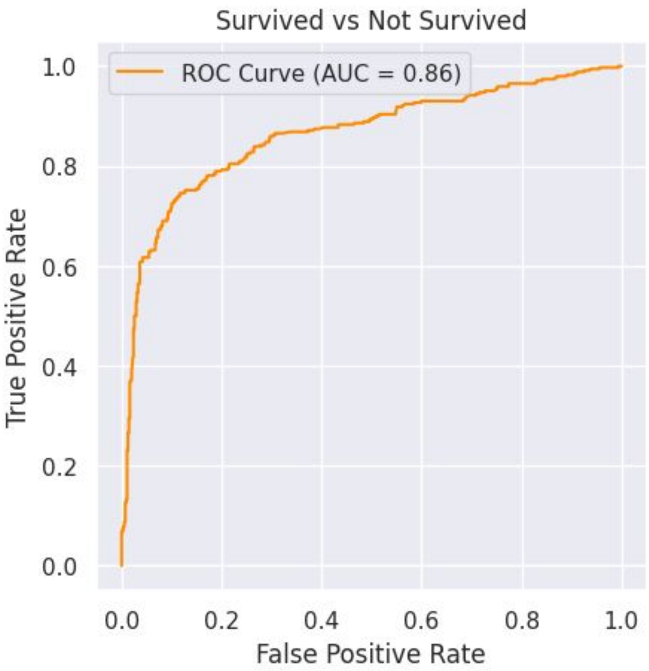



```
The selected parameters are {'bootstrap': False, 'max_features': 6, 'n_estimators': 4}
The precision on the training set using CV : 0.817
Binary Precision_score :  0.9939024390243902
Weighted Precision_score :  0.9801487123542315
Average_precision_score :  0.9653614305946356
Binary f1_score :  0.973134328358209
Weighted f1_score :  0.9797140849057524
              precision    recall  f1-score   support

           0       1.00      0.97      0.98       563
           1       0.95      0.99      0.97       328

    accuracy                           0.98       891
   macro avg       0.97      0.98      0.98       891
weighted avg       0.98      0.98      0.98       891
```

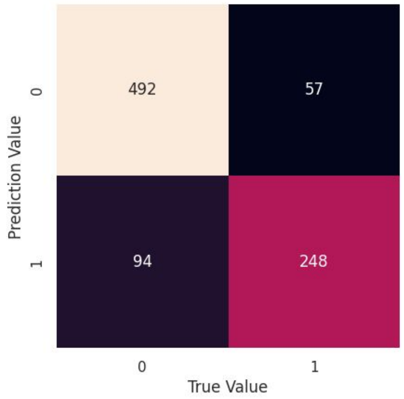
Cross-Validation
TEST

```
----------------- MEAN -----------------
Mean_test_score : 0.7948303098780155
Std_test_score : 0.026129778217375162
```

Support Vector Machine Model



Confusion Matrix
TRAIN




```
The selected parameters are {'degree': 1, 'probability': True}
The precision on the training set using CV : 0.822
Binary Precision_score :  0.8131147540983606
Weighted Precision_score :  0.8294280577836779
Average_precision_score :  0.6951265119801213
Binary f1_score :  0.7666151468315301
Weighted f1_score :  0.8284440108827827
              precision    recall  f1-score   support

           0       0.90      0.84      0.87       586
           1       0.73      0.81      0.77       305

    accuracy                           0.83       891
   macro avg       0.81      0.83      0.82       891
weighted avg       0.84      0.83      0.83       891
```

Cross-Validation
TEST

```
----------------- MEAN -----------------
Mean_test_score : 0.8215491808423827
Std_test_score : 0.012995405474709435
```

Which model is the chosen one?

- SVC gave the best mean score in the cross validation for the test dataset

- SVC showed the best predictions on Kaggle

Why?

- SVC was better at generalization, this meaning it performed better on unseen data

- SVC had a more robust feature representation making it less sensitive to noise or outliers

Submission and Description	Public Score
SVCPredictions.csv Complete · now	0.78229
LogisticRegressionPredictions.csv Complete · 13s ago	0.77033
DecisionTreePredictions.csv Complete · 25s ago	0.76315
kNNPredictions.csv Complete · 44s ago	0.73205
RandomForestPredictions.csv Complete · 1m ago	0.71291

Final thoughts

Project Resources

- ❖ GitHub Repository
- ❖ Google Drive Repository
- ❖ Google Colaboratory

END