FOR AFAME TECHNOLOGIES

TITANIC
SURVIVAL PREDICTION

PROJECT REPORT

By - Rahul Badola

TABLE OF CONTENT

INTRODUCTION

The RMS Titanic's tragic sinking in 1912 stands as a stark reminder of the vulnerability of human life in the face of natural disasters. Despite its opulence and perceived invincibility, the ship's collision with an iceberg led to the loss of over 1,500 lives. Today, leveraging data science techniques, we delve into the Titanic dataset, examining factors such as passenger class, age, and gender to discern patterns in survival rates. This project report explores the human stories within the data, aiming to unravel the mysteries surrounding Titanic's fateful voyage and shed light on the factors that influenced survival aboard the iconic vessel.

PYTHON LIBRARIES USED

1. **pandas (pd)**: Pandas is a powerful library for data manipulation and analysis in Python. It provides data structures like DataFrames, which are ideal for handling structured data.
2. **numpy (np)**: NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.
3. **seaborn (sns)**: Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
4. **matplotlib.pyplot (plt)**: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Pyplot is a module within Matplotlib that provides a MATLAB-like interface for plotting.
5. **warnings**: The warnings module is used to handle warnings that occur during program execution. In this case, you've suppressed all warnings from being displayed with warnings.filterwarnings('ignore').
6. **imblearn.under_sampling.RandomUnderSampler**: Imbalanced-learn is a library used for dealing with imbalanced datasets in machine learning. RandomUnderSampler is a technique used to balance class distribution by randomly eliminating samples from the majority class(es).
7. **sklearn.preprocessing.LabelEncoder**: LabelEncoder is used to convert categorical labels (e.g., strings) into numerical labels for machine learning algorithms.
8. **sklearn.model_selection.train_test_split**: This function is used to split datasets into random train and test subsets. It's commonly used for model evaluation and validation.
9. **sklearn.ensemble.RandomForestClassifier**: RandomForestClassifier is an ensemble learning method used for classification tasks. It fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
10. **sklearn.metrics.accuracy_score**: accuracy_score is a function used to compute the accuracy classification score, which is the fraction of correct predictions among the total number of predictions made.

11. sklearn.metrics.classification_report: classification_report is a function used to build a text report showing the main classification metrics (precision, recall, F1-score, and support) for each class in a classification task.

DATA OVERVIEW

0 - Not Survived	1 - Survived
891 - Rows	11 - Columns

DATA CLEANING

remove the 'PassengerId' column from the dataset since it does not contribute to the prediction task. This code will drop the 'PassengerId' column from the DataFrame df in place, meaning the change will be applied directly to the DataFrame without the need to reassign it to a new variable.

Dtypes

Survived	int64
Pclass	int64
Name	object
Sex	object
Age	float64
SibSp	int64
Parch	int64
Ticket	object
Fare	float64
Cabin	object
Embarked	object

```
# Dropping the Column 'PassengerId' since this column is not useful for prediction
df.drop('PassengerId',axis=1,inplace=True)
```

DEALING WITH NULL VALUES

Null values

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

```
# Dropping 'Cabin' Column:
df.drop('Cabin',axis=1,inplace=True)
```

Since the Age contains values in float so we can fill null values with its mean.

```
# Filling Null values with its mean

df.Age.fillna(df.Age.mean(),inplace=True)
```

Null values(After Replacing)

```
Survived    0
Pclass      0
Name        0
Sex         0
Age         0
SibSp       0
Parch       0
Ticket      0
Fare        0
Embarked    0
dtype: int64
```

Since the Age contains values in float so we can fill null values with its mean.

```
# Dealing with the column name 'Embarked'

df.Embarked.fillna(df.Embarked.mode()[0],inplace=True)
```

Null Values in the column 'Embarked' has been replaced with its mode (most repeated value).

Balancing of the Dataset

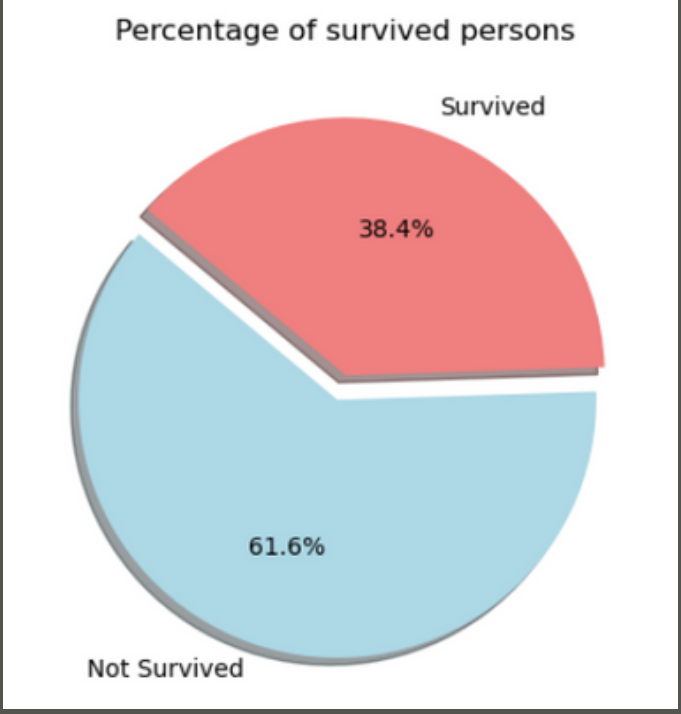
```
# Checking the balancing of the Dataset
df.Survived.value_counts() # Data is imb
```

Data is imbalanced, which results in biased results, so, we have to use under-sampeling technique to balance the Dataset.

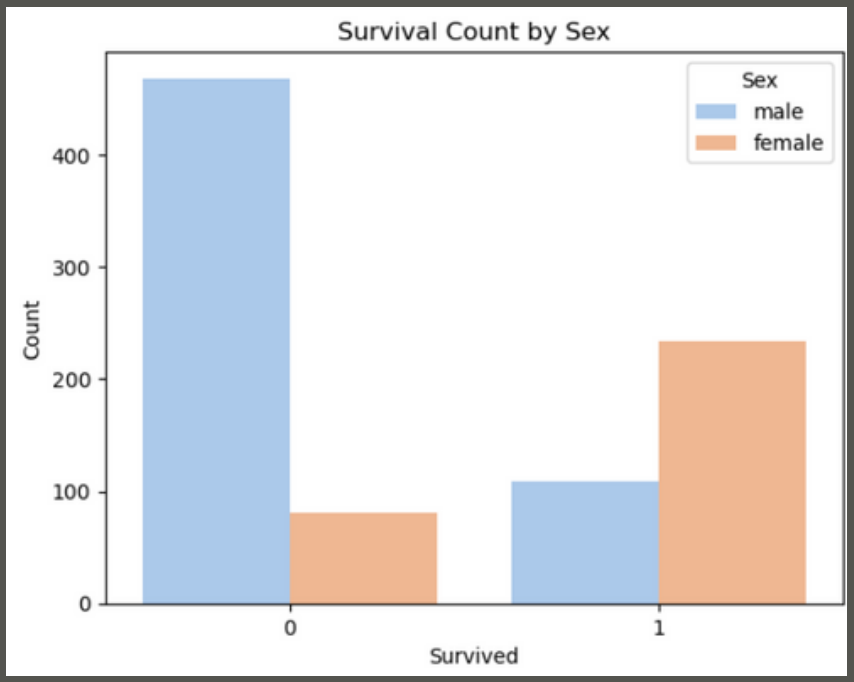
Not Survived - 549

Survived - 342

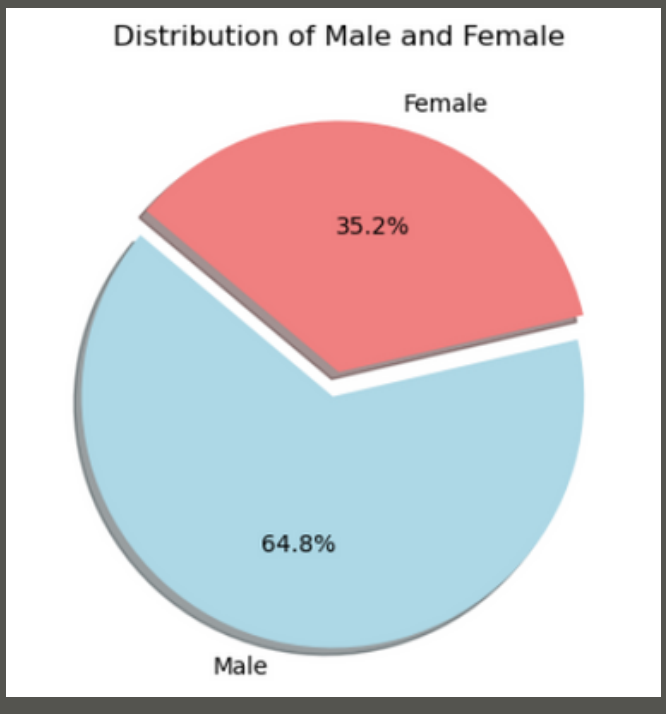
EDA - (EXPLORATORY DATA ANALYSIS)



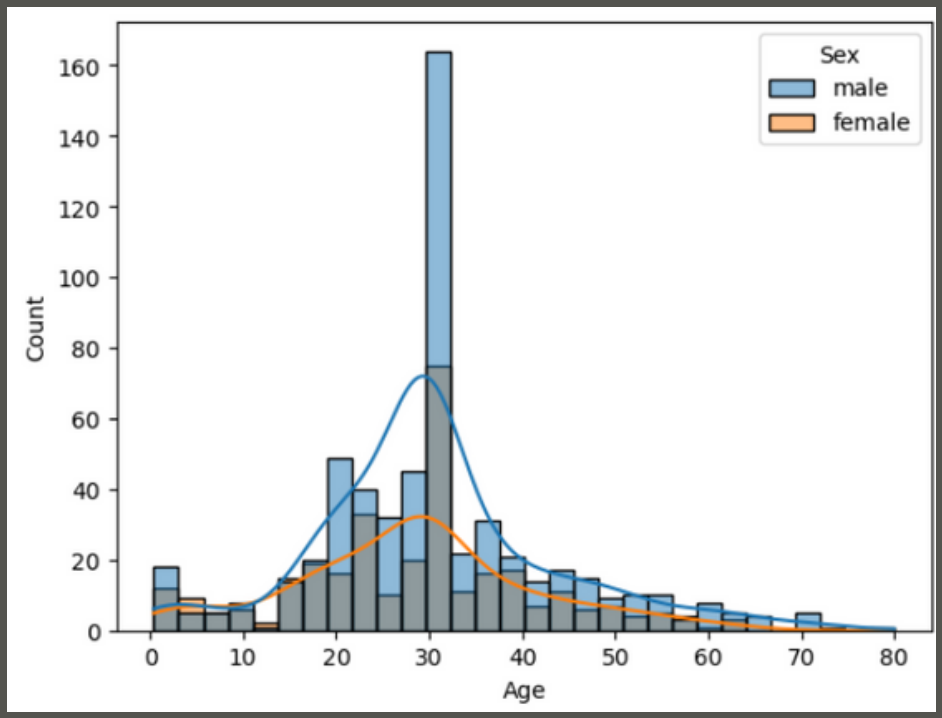
This pie chart visualizes the percentage of survived persons from the Titanic dataset. It shows that **38.4%** of individuals survived the disaster, while the majority, comprising **61.6%**, did not survive. This stark contrast highlights the tragic outcome of the Titanic's sinking and underscores the importance of understanding factors that influenced survival rates.



This bar plot visualizes the survival count by sex from the Titanic dataset. It reveals that a larger number of males did not survive the disaster, whereas a greater proportion of females survived. This discrepancy suggests a potential correlation between sex and survival on the Titanic, highlighting the importance of considering gender dynamics in analyzing historical events like this.

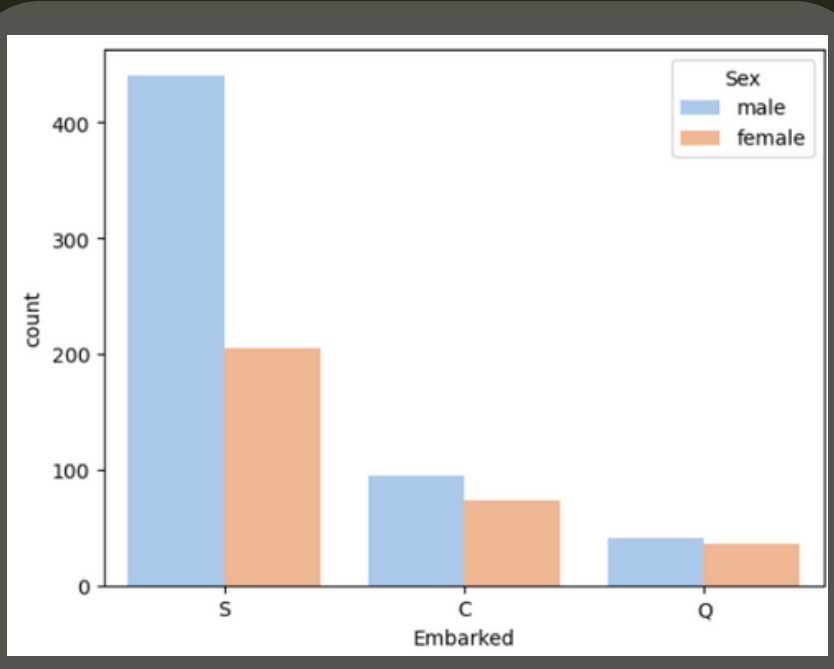


This pie chart illustrates the distribution of males and females in the Titanic dataset. It indicates that approximately **64.8%** of the individuals in the dataset are male, while around **35.2%** are female. This gender distribution provides valuable insight into the composition of passengers aboard the Titanic and underscores the importance of considering gender dynamics in analyzing historical events.



This histogram with density plot overlays depicts the distribution of ages among male and female passengers in the Titanic dataset. The histogram bars represent the count of individuals within different age groups, while the density curves provide an estimation of the probability density function for each gender.

From the visualization, it can be observed that the age distribution is skewed towards younger individuals, with a notable peak in the early 20s. Additionally, the density curves reveal subtle differences between the age distributions of males and females, suggesting potential variations in age demographics between the two genders aboard the Titanic.



This bar plot illustrates the count of passengers based on their port of embarkation (S = Southampton, C = Cherbourg, Q = Queenstown) and their respective genders.
It's evident that the majority of passengers embarked from Southampton, with a higher proportion of males compared to females. Cherbourg had the second highest number of passengers, with a relatively

balanced distribution between males and females. Queenstown had the fewest passengers, with more males than females.

PREDICTION

STEP - 1

```
# Using Label Encoder to Encode 'Embarked'

le=LabelEncoder()
df.Embarked=le.fit_transform(df.Embarked)
df.Sex=le.fit_transform(df.Sex)
```

Here, we instantiate a LabelEncoder object and then use the `fit_transform` method to encode the 'Embarked' and 'Sex' columns of the DataFrame `df`. This process assigns numerical labels to each unique category in these columns, making them suitable for machine learning algorithms that require numerical input.

STEP - 2

```
# Creating Dependent And Independent Variable

x=df.drop(['Survived','Name'],axis=1)
y=df.Survived
```

Here, x is assigned the DataFrame df after dropping the columns 'Survived' and 'Name' along the columns axis (axis=1), making it the independent variable matrix. y is assigned the 'Survived' column from the DataFrame df, representing the dependent variable. This separation is crucial for training machine learning models, where x contains the features used for prediction and y contains the target variable to be predicted.

STEP - 3

```
us=RandomUnderSampler()
x_resample,y_resample=us.fit_resample(x,y)
```

Here, a RandomUnderSampler object `us` is instantiated, which will randomly undersample the majority class (0 for 'Not Survived') to balance the class distribution. The `fit_resample` method is then applied to the independent variable matrix `x` and the dependent variable `y` to perform the undersampling. The resulting `x_resample` and `y_resample` contain the balanced dataset, where both classes are represented equally, thus mitigating bias in the classification results.

STEP - 4

```
x_resample.reset_index(drop=True)
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	26.000000	0	0	7.8958	2
1	2	0	24.000000	0	0	13.0000	2
2	3	1	45.000000	0	0	6.9750	2
3	3	1	29.699118	0	0	7.2292	0
4	3	1	29.699118	0	0	7.8958	2
...	...	...	...	...	...	...	...
679	3	0	15.000000	0	0	7.2250	0
680	1	0	56.000000	0	1	83.1583	0
681	2	0	25.000000	0	1	26.0000	2
682	1	0	19.000000	0	0	30.0000	2
683	1	1	26.000000	0	0	30.0000	0

684 rows × 7 columns

The code `x_resample.reset_index(drop=True)` resets the index of the DataFrame `x_resample` after undersampling, dropping the old index. This ensures that the index is reset to consecutive integers starting from 0, maintaining the integrity of the data after undersampling.

The resulting DataFrame shows 684 rows and 7 columns, indicating that the index has been successfully reset, and the DataFrame is ready for further analysis or modeling tasks.

STEP - 5

```
# splitting Training and testing Data

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=46)
```

Here, the `train_test_split` function from scikit-learn is used to split the independent variable matrix `x` and the dependent variable `y` into training and testing sets. The parameter `test_size=0.20` specifies that 20% of the data will be used for testing, while the remaining 80% will be used for training. Additionally, `random_state=46` sets the random seed for reproducibility, ensuring that the same random split is obtained each time the code is run.

STEP - 6

```
# assigning the varaible to Algorithm(Random forest classifier)
rfc=RandomForestClassifier(n_estimators=100, max_depth=5)
```

Here, the RandomForestClassifier from scikit-learn is instantiated with parameters `n_estimators=100` and `max_depth=5`. This creates a random forest classifier with 100 decision trees (estimators) and a maximum depth of 5 for each tree. The `rfc` variable now holds this classifier object, which can be used for training and making predictions on the dataset.

STEP - 7

```
# fitting the training data into Algo
rfc.fit(x_train,y_train)
```

Here, the fit method of the RandomForestClassifier rfc is called with the training data x_train and y_train as arguments. This step trains the random forest classifier on the training data, allowing it to learn the patterns and relationships between the features and the target variable.

STEP - 8

```
# Model prediction

y_pred=rfc.predict(x_test)
```

Here, the predict method of the trained Random Forest Classifier rfc is used to predict the target variable based on the features in the testing dataset x_test. The predicted values are stored in the variable y_pred, which can be used for evaluating the model's performance and comparing it against the actual values in y_test.

STEP - 9

```
# Checking the Accuracy
accuracy_score(y_test,y_pred)
```

Here, the accuracy_score function from scikit-learn is used to compute the accuracy of the model by comparing the predicted values y_pred with the actual values y_test. This function returns the accuracy score, which represents the proportion of correctly classified instances out of the total number of instances in the testing dataset.

Classification Report

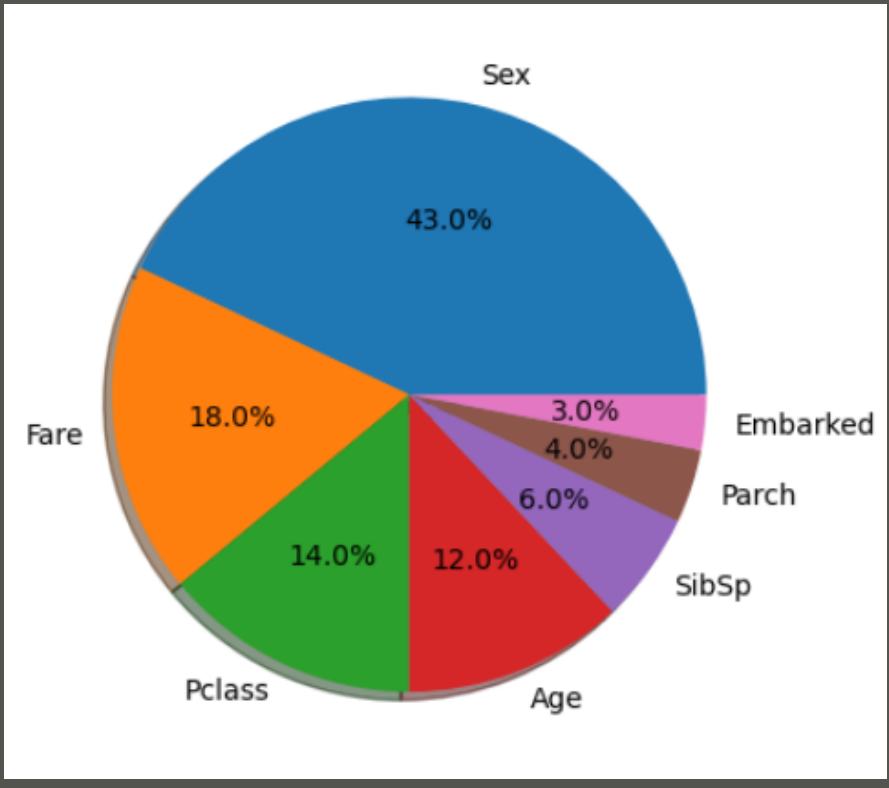
```
# Checking the Classification Report
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.82	0.93	0.87	104
1	0.89	0.72	0.79	75
accuracy			0.84	179
macro avg	0.85	0.83	0.83	179
weighted avg	0.85	0.84	0.84	179

Feature Imp.

	column	contribtion_in_%
0	Sex	0.43
1	Fare	0.18
2	Pclass	0.14
3	Age	0.12
4	SibSp	0.06
5	Parch	0.04
6	Embarked	0.03

Feature Imp.



CONCLUSION

In conclusion, this project aimed to predict the survival of passengers aboard the Titanic using machine learning techniques.

The dataset was first explored and preprocessed to handle missing values and encode categorical variables. Various visualizations were employed to understand the data distribution and relationships between different features.

Next, a Random Forest Classifier model was trained on the preprocessed data. The model was evaluated using metrics such as accuracy, precision, recall, and F1-score.

The classification report revealed that the model performed reasonably well, with an overall accuracy of 84%. It achieved high precision and recall for predicting both survival and non-survival cases, indicating a good balance between correctly identifying survivors and non-survivors.

Overall, this project demonstrates the application of machine learning algorithms in predicting survival outcomes based on historical data, offering insights into factors influencing survival rates on the Titanic. Further improvements and optimizations could be explored to enhance the model's performance and robustness.

THANK YOU
END OF THE PROJECT