TITANIC SURVIVAL PROJECT

END TERM REPORT

By:
Samriddhi Gupta
Section: K21ML
Roll.No: 16
Registration number:
12107731

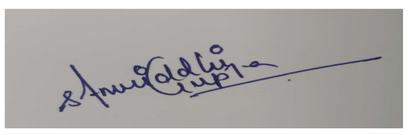
&

Shalini Guha
Section: K21ML
Roll.No: 15
Registration number:
12107495

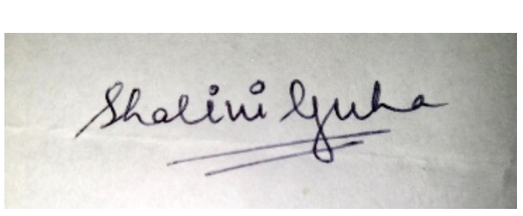
Department of Intelligent Systems
School of Computer Science Engineering
Lovely Professional University, Jalandhar
October-2023

Student Declaration

I hereby declare that I am the author of this report. The report contains all the original material. Every piece of information that was taken from another source has been properly credited. I understand that I shall accept full responsibility if any portion of the report is discovered to have been copied.



Signature:
Name: Samriddhi Gupta
Roll.No: 16



Signature:
Name: Shalini Guha
Roll.No: 15

Place: Lovely Professional University
Date: 31 October-2023

Acknowledgement

With outmost thanks, I acknowledge the support and assistance of my professor, Dr. Dhanpratap Singh, who has always motivated me to pursue this project. I could not have completed this assignment successfully without his constant direction and tenacious assistance. This project would not have been successful without the support of the computer science department of Lovely Professional University, Punjab, for which I am thankful. I am also grateful to my friends and family for their unwavering love and support over the years.

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Verified that "Samriddhi Gupta" and “Shalini Guha”, who completed the project work under my direction, is the legitimate author of this project report on "Titanic Survival Project."

Signature of the

Supervisor: Name: Dr.

Dhanpratap Singh

ID:25706

Department:

Computer Science Engineering

Objective:

The Titanic Survival Project seeks to develop a predictive model to determine passengers' survival on the RMS Titanic by conducting data analysis to understand influential factors and evaluating the model's performance. In sum, this project aims to uncover survival factors, build a robust model and offer valuable insights for similar scenarios, all within a concise framework.

This Project is a testament to the enduring interest in the Titanic's story and the growing capabilities of data analysis and machine learning in unraveling complex historical narratives. It stands as a collaborative effort that seeks to commemorate the past and provide insights that may inform future endeavors in data science and historical research.

History:

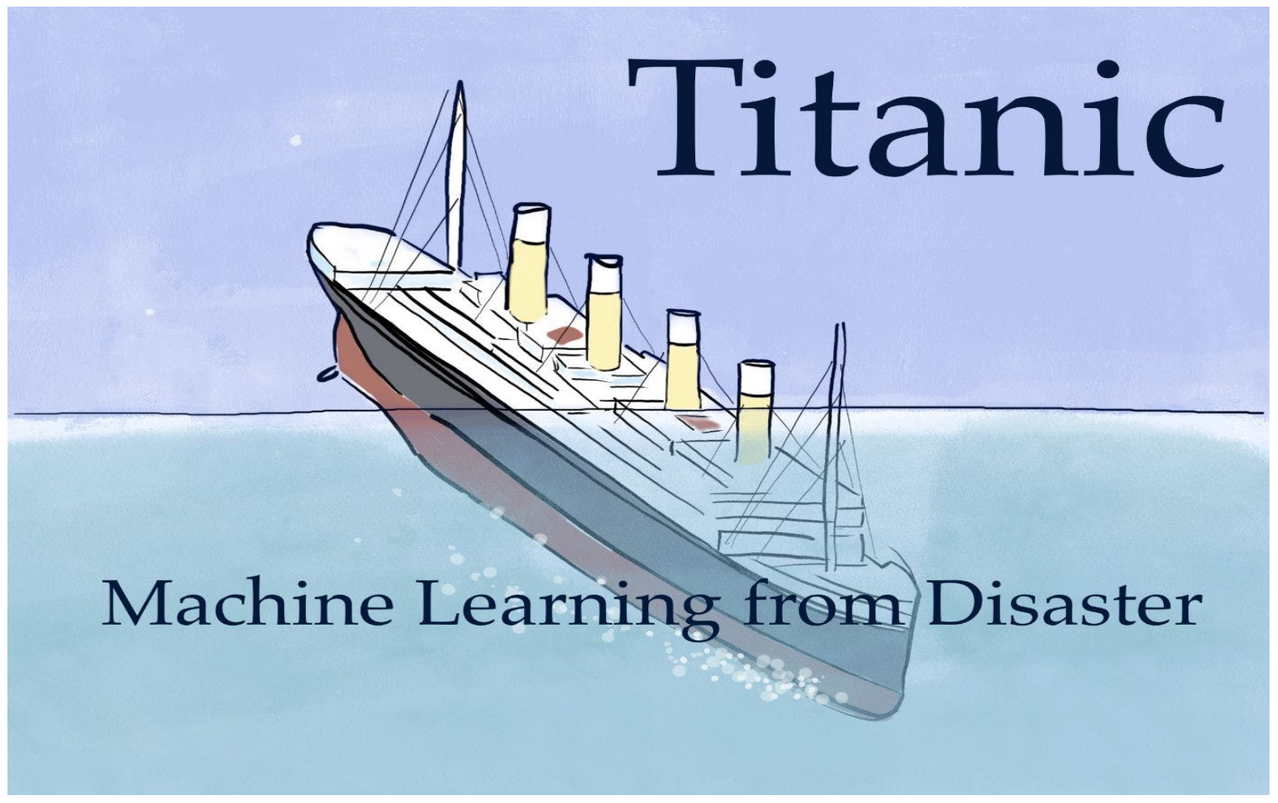
The titanic Survival Project traces its origins to the release of the Titanic dataset and the growing interest in data analysis and machine learning due to its historical significance and the challenges it posed on for predictive modeling.

The dataset includes information on all the 2,224 passengers, containing the demographic information such as name, age, sex, passenger class and occupation. Once the dataset was collected to explore and identify the patterns, it was found that certain groups of passengers were more likely to survive than others. For example, women and children were more likely to survive than men, and passengers in higher passenger classes were more likely to survive than those in lower passenger classes.

Later, the team then used the data to train a Random Forest classifier, a machine learning algorithm used for making predictions by creating a large number of decision trees and then averaging the predictions of the trees. Once the model was trained, significantly higher accuracy was achieved than the accuracy of a random guess.

The project's results and findings were disseminated within the data science community, contributing to the collective knowledge of predictive modeling and historical analysis. It served as an educational resource for aspiring data scientists and researchers interested in similar exploratory projects.

The Titanic Survival Project is a testament to the enduring interest in the Titanic's story and the growing capabilities of data analysis and machine learning in unraveling complex historical narratives.



https://github.com/inashellshelley/Titanic-SurvivalMLProject

Dataset Description

Titanic dataset is one of the most used data sets. It contains information about the passengers onboard the Titanic when the ship went down in 1912. It includes things like age, gender, passenger class and fare paid. It also includes information about whether the passengers survived the sinking or not. Titanic data sets are used for many different purposes in data analysis. For example, it can be used to predict survival based on the characteristics of the passengers or to explore patterns within the data. There are two files in the Titanic dataset. One is for training the model and the other is for testing the model.

The dataset contains a mix of numerical and categorical features. The numerical features include Age, SibSp, Parch, Fare, and Cabin. The categorical features include Survived, Pclass, Sex, and Embarked.The dataset also contains some missing values. The Age feature is missing for about 20% of the passengers. The Cabin feature is missing for about 75% of the passengers. The Embarked feature is missing for 2 passengers.

The dataset consists of 12 columns, each providing essential_information about the passengers.

```
titanicdata.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
```

The dataset contains the following features:

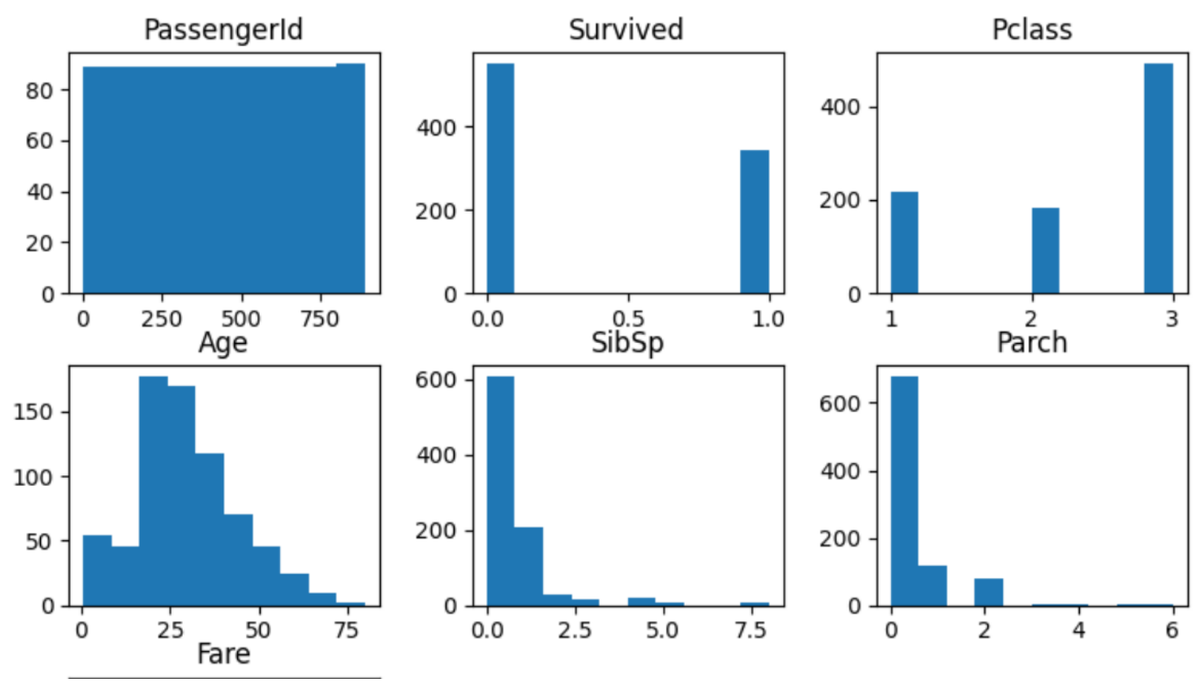
PassengerId: A unique identifier for each passenger.
Survived: Whether the passenger survived the shipwreck.
Pclass: The passenger class (1st, 2nd, or 3rd).
Name: The passenger's name.
Sex: The passenger's gender.
Age: The passenger's age in years.
SibSp: The number of siblings or spouses traveling with the passenger.
Parch: The number of parents or children traveling with the passenger.
Ticket: The passenger's ticket number.
Fare: The fare paid for the passenger's ticket.
Cabin: The passenger's cabin number.
Embarked: The port where the passenger embarked on the Titanic (Southampton, Cherbourg, or Queenstown).

	0	1	2	3	4
PassengerId	892	893	894	895	896
Pclass	3	3	2	3	3
Name	Kelly, Mr. James	Wilkes, Mrs. James (Ellen Needs)	Myles, Mr. Thomas Francis	Wirz, Mr. Albert	Hirvonen, Mrs. Alexander (Helga E Lindqvist)
Sex	male	female	male	male	female
Age	34.500000	47.000000	62.000000	27.000000	22.000000
SibSp	0	1	0	0	1
Parch	0	0	0	0	1
Ticket	330911	363272	240276	315154	3101298
Fare	7.829200	7.000000	9.687500	8.662500	12.287500
Cabin	nan	nan	nan	nan	nan
Embarked	Q	S	Q	S	S

Project Description:

Titanic Survival Prediction

```
titanicdata.hist(bins=10,figsize=(9,7),grid=False);
```



The output displayed in the code shows the count of missing values for each column in the "titanicdata" DataFrame. It lists the column names, such as "PassengerId," "Survived," "Age," "Cabin," and "Embarked," along with the corresponding number of missing values in each column.

Data Preprocessing: This code is part of the data preprocessing phase of a machine learning project. Detecting and handling missing values is a crucial step in preparing data for modeling.

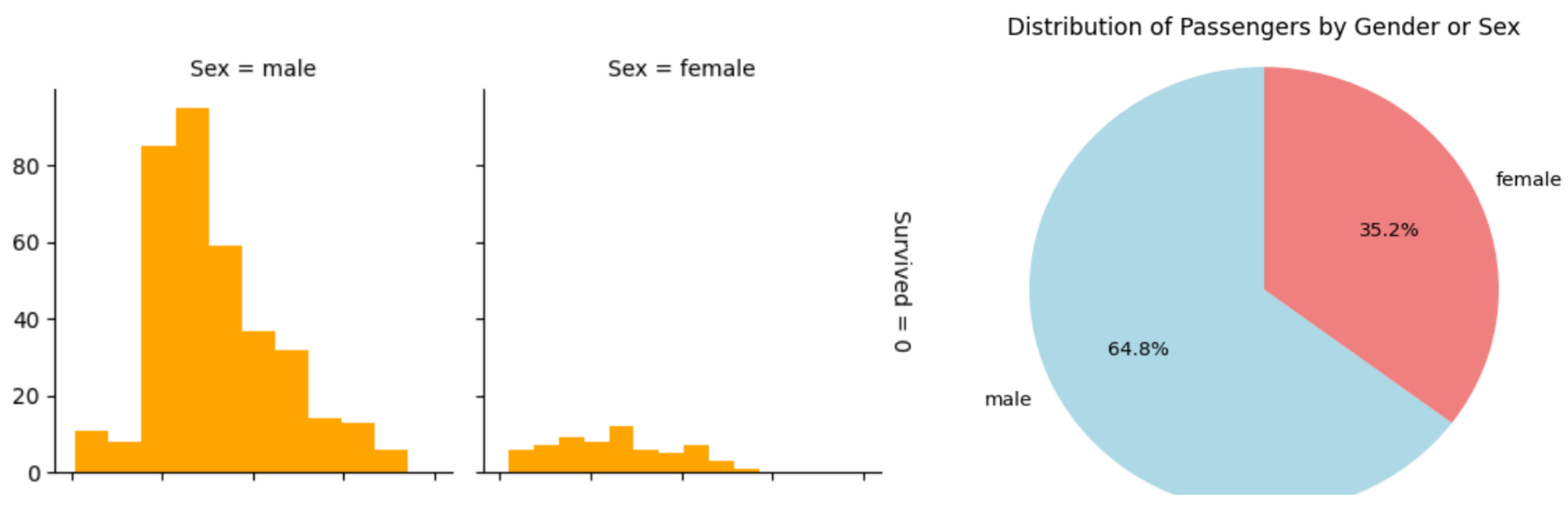
Exploratory Data Analysis (EDA): The code provides a summary of missing values in the dataset, which is a common aspect of EDA. EDA involves analyzing and visualizing the dataset to gain insights into its structure and quality.

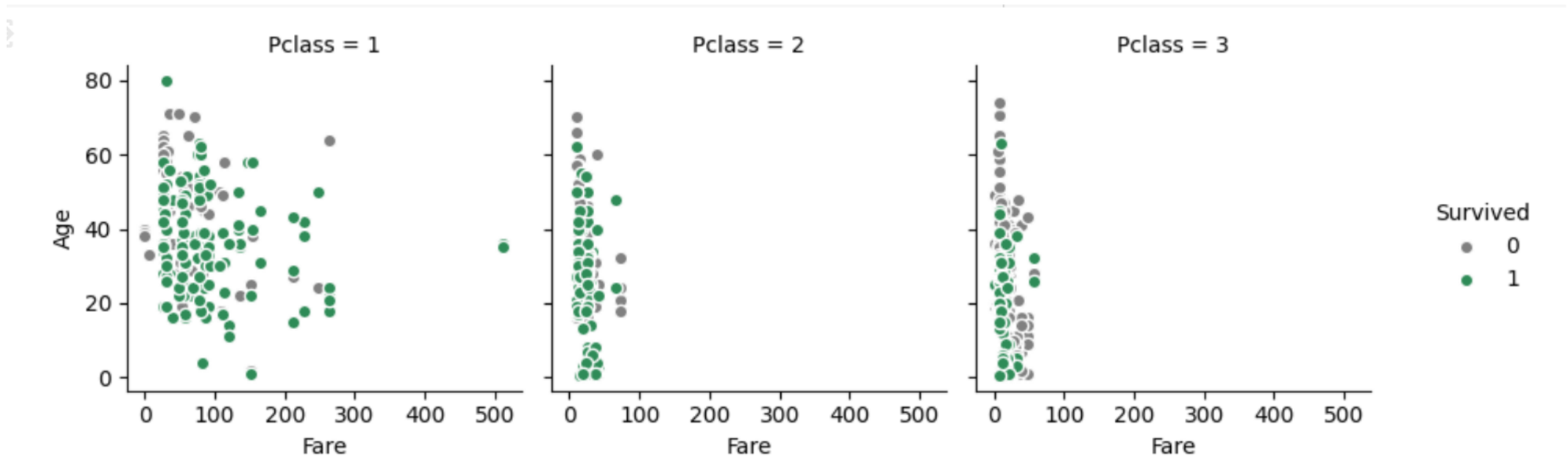
Explanation of Missing Values Analysis:

The code first identifies columns that have missing values using the titanicdata.isnull().any() condition.

It then calculates and displays the number of missing values for each of these columns using titanicdata.isnull().sum().

The output shows that "Age" has 177 missing values, "Cabin" has 687 missing values, and "Embarked" has 2 missing values.





Visualizations for gender-based survival analysis, passenger counts by boarding location, and survival counts by passenger class. These visualizations can help in understanding how different factors may have influenced the survival of passengers on the Titanic.

Data Visualization: The code generates various data visualizations to explore and understand patterns and relationships in the Titanic dataset. These visualizations can help identify factors that may have influenced passenger survival.

Data Presentation: The code uses various plot types, including scatter plots, bar plots, and count plots, to present information in a visually appealing and informative way.

- FacetGrid with Scatter Plot:

g = sea.FacetGrid(titanicdata, hue="Survived", col="Sex", margin_titles=True, palette="Set1", hue_kws=dict(marker=["^", "v"])): This line creates a facet grid, where each facet represents a combination of "Sex" and "Survived" with different marker symbols for each "Survived" class (0 and 1).

g.map(plt.scatter, "Fare", "Age", edgecolor="w").add_legend(): This line maps a scatter plot of "Fare" (x-axis) against "Age" (y-axis) for each facet. The "edgecolor" parameter specifies the color of marker edges. The add_legend() function adds a legend to differentiate between the "Survived" classes.

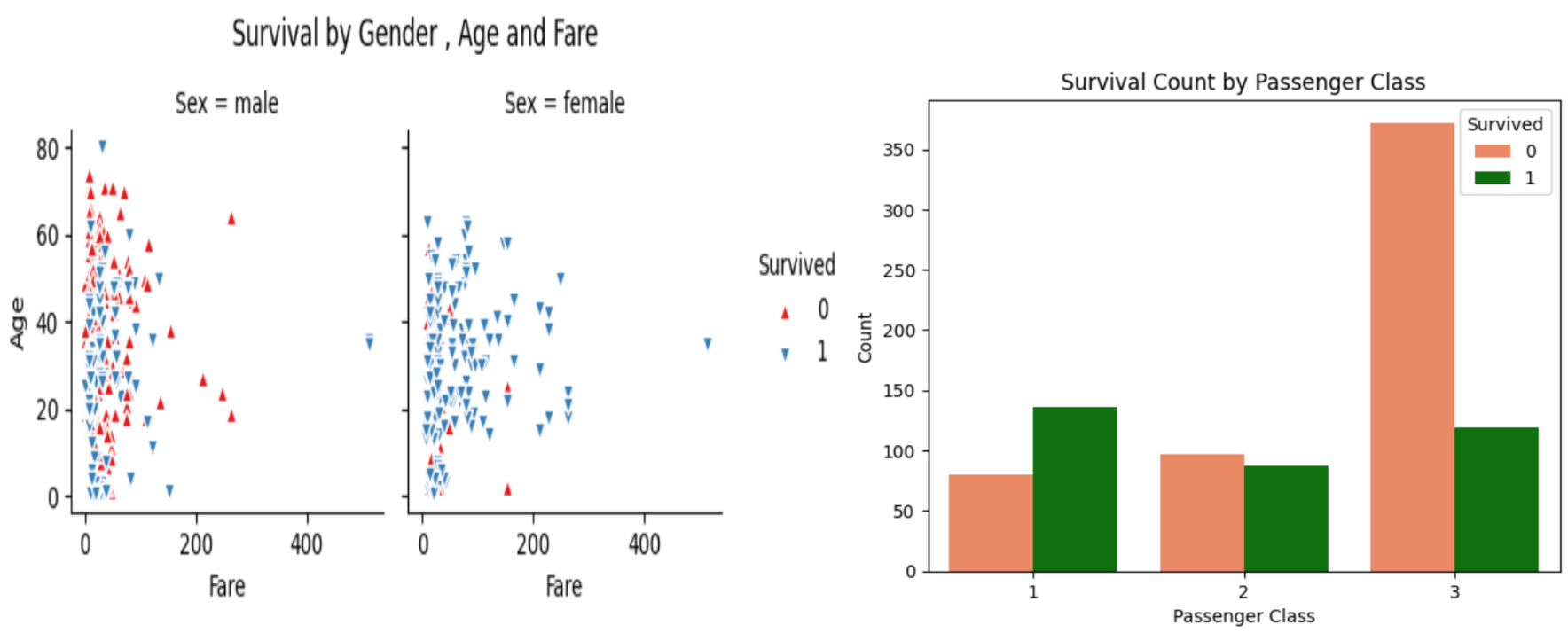
plt.subplots_adjust(top=0.8): This adjusts the layout of the subplots to make room for the main title.

g.fig.suptitle('Survival by Gender, Age, and Fare'): This sets the main title for the facet grid visualization.

- Bar Plot for Embarked:

titanicdata.Embarked.value_counts().plot(kind='bar', alpha=0.55): This line plots a bar chart to show the count of passengers per boarding location ("Embarked"). The "alpha" parameter controls the transparency of the bars.

plt.title("Passengers per boarding location"): Sets the title for the bar chart.



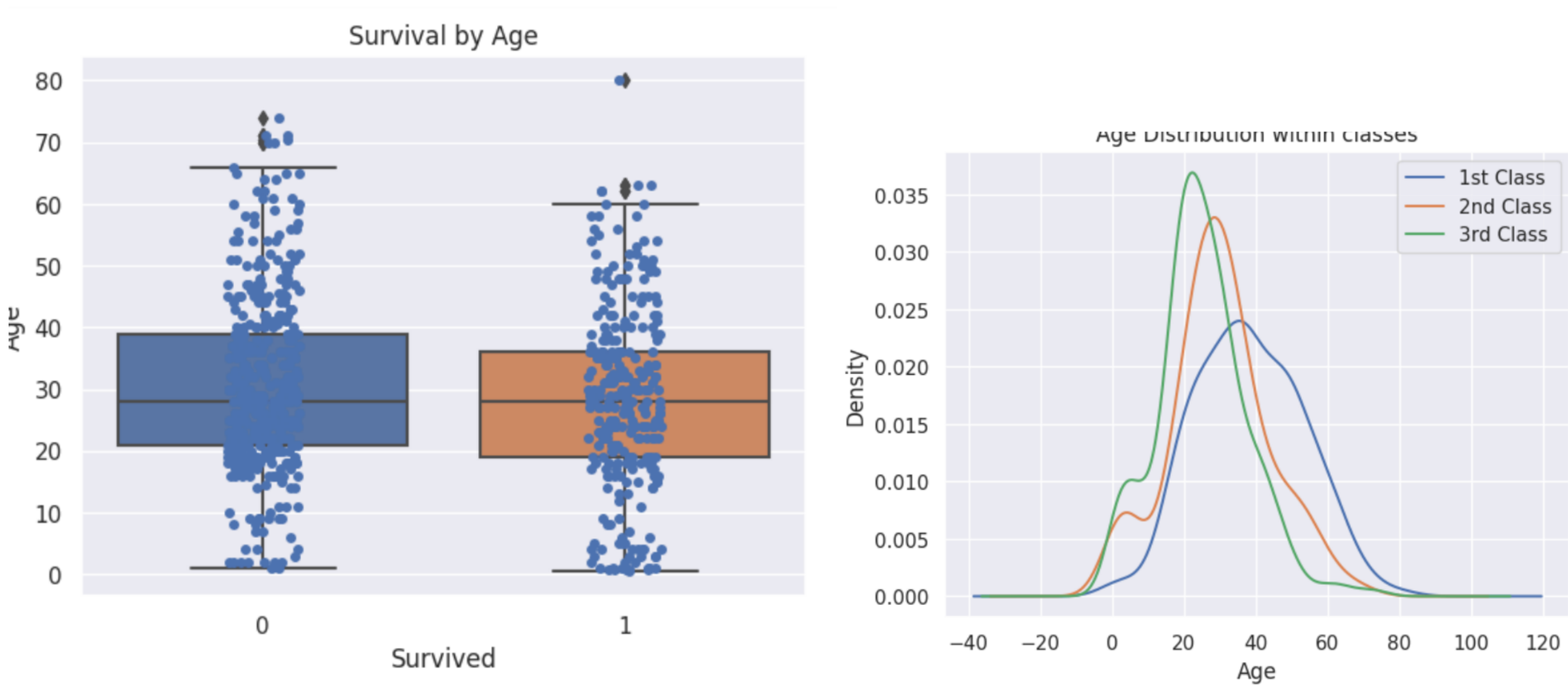
- Count Plot for Passenger Class and Survival:

sea.countplot(x='Pclass', hue='Survived', data=titanicdata, palette=['coral','green']): This line creates a count plot to display the count of passengers in each passenger class ("Pclass") and their survival status ("Survived"). The "hue" parameter colors the bars based on survival status using the specified color

palette.
plt.xlabel('Passenger Class'): Sets the x-axis label
plt.ylabel('Count'): Sets the y-axis label.
plt.title('Survival Count by Passenger Class'): Sets the title for the count plot.
plt.show(): This line displays all the visualizations created in the previous steps.

- Encoding Categorical Variables:

labelEnc = LabelEncoder(): This line creates an instance of the LabelEncoder class from the scikit-learn library. LabelEncoder is used to encode categorical variables into numerical values.
cat_vars = ['Embarked', 'Sex', "Title", "FsizeD", "NlengthD", 'Deck']: This line defines a list of categorical variables in the dataset that need to be encoded.
for col in cat_vars:: This loop iterates through each categorical variable in the list.
titanicdata[col] = labelEnc.fit_transform(titanicdata[col]): Within the loop, this line encodes the categorical variable specified by col in the "titanicdata" DataFrame into numerical values using LabelEncoder. It replaces the original categorical values with the encoded values.
titanic_test[col] = labelEnc.fit_transform(titanic_test[col]): Similar to the previous line, this encodes the same categorical variable in the test dataset ("titanic_test").
Display the First Rows of the DataFrame:
titanicdata.head(): This line displays the first few rows of the "titanicdata" DataFrame after the categorical variables have been encoded.
Styling the DataFrame:
styled_table = titanicdata.head().style.set_table_styles([...]): This line creates a styled representation of the first few rows of the "titanicdata" DataFrame. The styling defines the appearance of the table, including background colors for headers and cells.

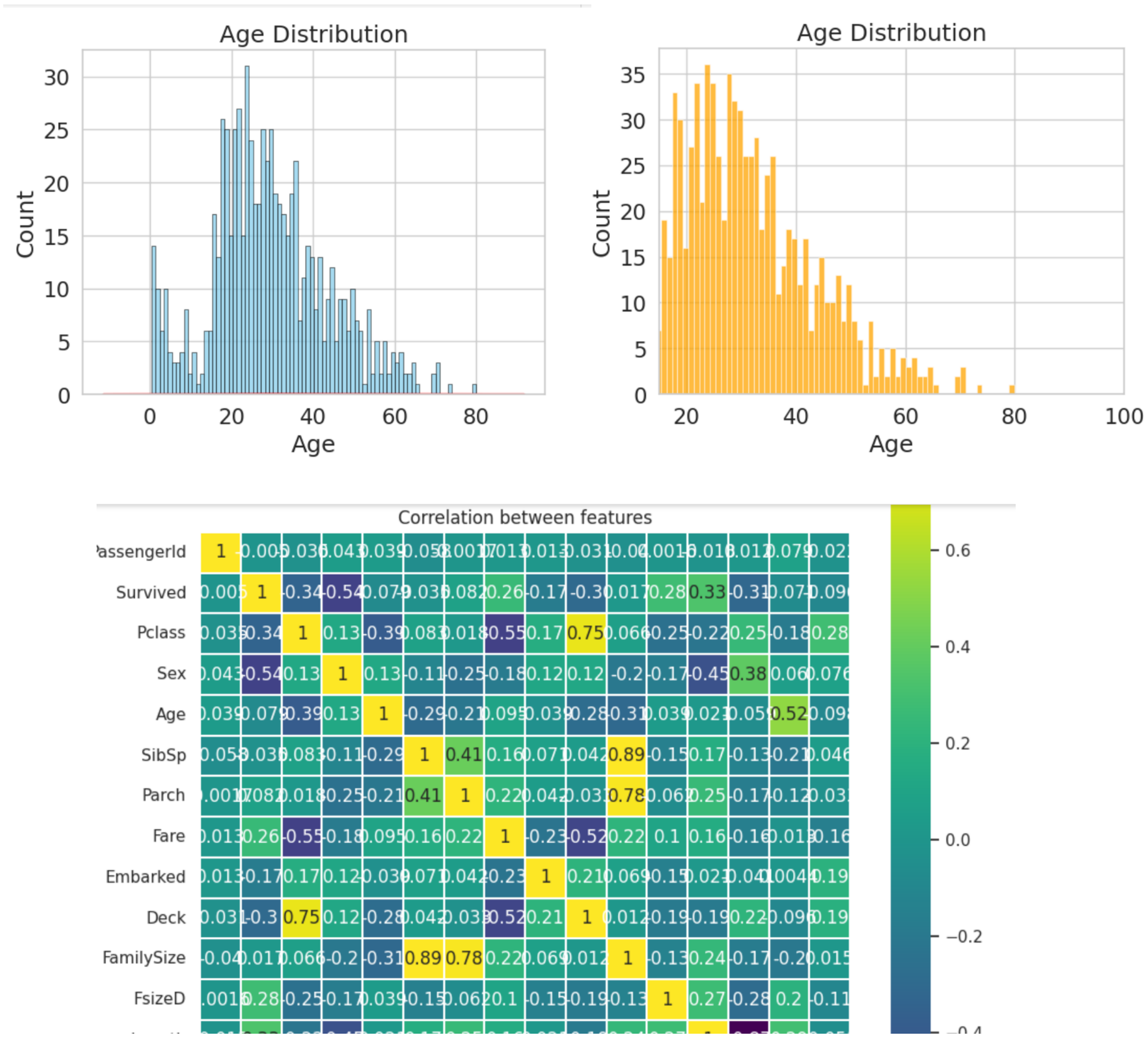


	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000
mean	1100.500000	2.265550	30.272590	0.447368	0.392344	35.627188
std	120.810458	0.841838	14.181209	0.896760	0.981429	55.907576
min	892.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	996.250000	1.000000	21.000000	0.000000	0.000000	7.895800
50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200
75%	1204.750000	3.000000	39.000000	1.000000	0.000000	31.500000
max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

Logistic Regression: Logistic regression is used for binary classification tasks, and this code uses it to predict the "Survived" class (0 or 1) based on the given features.
Cross-Validation: Cross-validation is a technique for assessing the model's performance and generalization to new data. It helps prevent overfitting and provides a more accurate evaluation of the model.
Evaluation Metric (F1 Score): The F1 score is a commonly used metric for binary classification models. It combines precision and recall to provide a balanced measure of a model's performance.

Working:

Feature Selection:
age_df = df[['Age', 'Embarked', 'Fare', 'Parch', 'SibSp', 'TicketNumber', 'Title', 'Pclass', 'FamilySize', 'FsizeD', 'NameLength', 'NlengthD', 'Deck']]: This line selects a subset of features from the original DataFrame (specified by df). These features are used as input variables for predicting missing ages.
Data Splitting:
train = age_df.loc[(df.Age.notnull())]: This line creates a subset of the "age_df" DataFrame that contains only rows where the "Age" column is not null. This subset is used as the training data for building the regression model.
test = age_df.loc[(df.Age.isnull())]: This line creates a subset of the "age_df" DataFrame that contains only rows where the "Age" column is null. This subset represents the data for which missing ages need to be predicted.
Splitting Input (X) and Target (y) Variables:
y = train.values[:, 0]: This line extracts the target variable ("Age") from the training data.
X = train.values[:, 1::]: This line extracts the input variables (features) from the training data, excluding the "Age" column.
Random Forest Regression Model:
rt = RandomForestRegressor(n_estimators=2000, n_jobs=-1): This line creates a random forest regression model with 2000 decision trees and uses all available CPU cores (n_jobs=-1) for parallel processing.
Model Training:
rt.fit(X, y): This line trains the random forest regression model using the training data. The model learns to predict missing ages based on the selected features.
Age Prediction:
predictedAges = rt.predict(test.values[:, 1::]): This line uses the trained model to predict missing ages for the test data (where "Age" is null). It uses the same set of features for prediction.
Filling Missing Ages:
df.loc[(df.Age.isnull()), 'Age'] = predictedAges: This line replaces the missing "Age" values in the original DataFrame (specified by df) with the predicted ages.
Return the Updated DataFrame:
return df: The function returns the updated DataFrame with missing ages filled.



The logistic regression model is trained and evaluated using cross-validation. Cross-validation helps estimate the model's performance on unseen data by splitting the dataset into multiple training and testing sets.

The F1 score is computed for each fold of the cross-validation, and the mean F1 score is printed, which provides an estimate of how well the logistic regression model performs in terms of precision and recall for predicting survival on the Titanic dataset.

PREDICTION OF SURVIVAL

Using Logistic Regression Model

```
[60] from sklearn.linear_model import LogisticRegre
     from sklearn.model_selection import cross_val_

     predictors = ["Pclass", "Sex", "Fare", "Embark

     lr = LogisticRegression(random_state=1)

     cv = ShuffleSplit(n_splits=10, test_size=0.3,

     scores = cross_val_score(lr, titanicdata[predi

     print(scores.mean())
```

```
0.7450401762716468
```

Using Random Forest algorithm Essential Features prediction

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold, cross_val_p
import numpy as np
import pandas as pd

predictors = ["Pclass", "Sex", "Age", "Fare", "Nlength

rf = RandomForestClassifier(random_state=1, n_estimato

kf = KFold(n_splits=5, shuffle=True, random_state=1)
cv = ShuffleSplit(n_splits=10, test_size=0.3, random_s

predictions = cross_val_predict(rf, titanicdata[predic
predictions = pd.Series(predictions)

scores = cross_val_score(rf, titanicdata[predictors],

print(scores.mean())
```

```
0.7413230613452748
```

Calculate Feature Importances:

importances = rf.feature_importances_: This line retrieves the feature importances computed by a Random Forest model and stores them in the "importances" variable.

std = np.std([rf.feature_importances_ for tree in rf.estimators_], axis=0): This line calculates the standard deviation of feature importances across all decision trees in the Random Forest ensemble and stores it in the "std" variable.

indices = np.argsort(importances)[::-1]: This line sorts the indices of features in descending order of importance based on their importance scores.

Create the Figure and Title:

plt.figure(figsize=(10, 6)): This line specifies the size of the figure (the plot area) to control its dimensions.

plt.title("Feature Importances By Random Forest Model"): Sets the title for the bar chart.

Define Colors:

colors = ['royalblue', 'limegreen', 'orange', 'red', 'purple', 'gold', 'pink', 'beige']: This line defines a list of colors that will be used for the bars in the bar chart. You can customize the colors as needed.

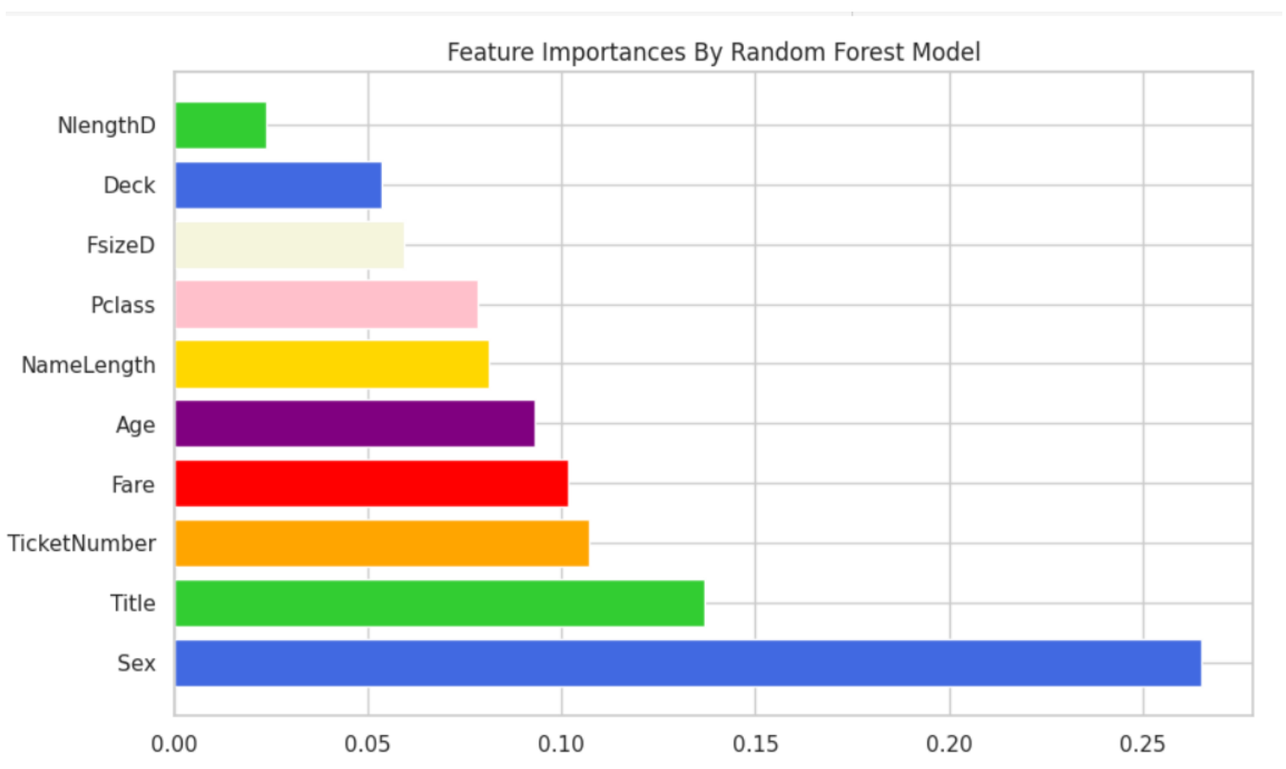
Create the Bar Chart:

plt.barh(range(len(sorted_important_features), importances[indices], color=colors): This line creates a horizontal bar chart. It uses the sorted feature importances, with colors specified for each feature.

plt.yticks(range(len(sorted_important_features)), sorted_important_features): Sets the y-axis labels using the names of the sorted features.

Show the Plot:

plt.show(): This line displays the bar chart with the feature importances.



AdaBoost Classifier: The code demonstrates the use of an AdaBoost classifier, which is an ensemble learning method that combines multiple weak learners to create a strong classifier.

Create an AdaBoost Classifier:

adb = AdaBoostClassifier(): This line creates an instance of the AdaBoost classifier. The default hyperparameters are used in this case.

Fit the Model:

adb.fit(titanicdata[predictors], titanicdata["Survived"]): This line fits the AdaBoost classifier to the training data. It uses the specified features (predictors) and the "Survived" target variable for training.

Define Cross-Validation Strategy:

cv = ShuffleSplit(n_splits=10, test_size=0.3, random_state=50): This code creates a ShuffleSplit cross-validation strategy with 10 splits and a 30% test size. The random seed is

set for reproducibility.

Cross-Validation and Model Evaluation:

scores = cross_val_score(adb, titanicdata[predictors], titanicdata["Survived"], scoring='f1', cv=cv): This line calculates the F1 scores using cross-validation. The F1 score is a common metric for classification models.

print(scores.mean()): The mean F1 score is printed, providing an estimate of the AdaBoost model's performance in predicting survival on the Titanic dataset.

Create a Voting Classifier:

eclf1 = VotingClassifier(estimators=[('lr', lr), ('rf', rf), ('adb', adb)], voting='soft'): This code creates a Voting Classifier named eclf1. It combines three base classifiers: Logistic Regression (lr), Random Forest (rf), and AdaBoost (adb) using soft voting. Soft voting involves averaging the predicted probabilities from the base classifiers.

Fit the Model:

eclf1 = eclf1.fit(titanicdata[predictors], titanicdata["Survived"]): This line fits the ensemble model to the training data using the specified features and target variable.

Make Predictions:

predictions = eclf1.predict(titanicdata[predictors]): This line generates predictions for the training data using the ensemble model.

test_predictions = eclf1.predict(titanic_test[predictors]): This line generates predictions for the test data using the ensemble model.

Post-Processing:

test_predictions = test_predictions.astype(int): This line converts the test predictions to integers.

Create a Submission File:

submission.to_csv("titanic_submission.csv", index=False): This line saves the submission DataFrame to a CSV file, which can be used for submitting predictions to a competition or evaluation.

PassengerI	Survived
892	0
893	0
894	0
895	0
896	0
897	0
898	0
899	0
900	1
901	0
902	0
903	0
904	1

strat_test_set

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Deck	FamilySize	FsizeD	NameLength	NlengthD	Title	TicketNumber
174	175	0	1	Smith, Mr. James Clinch	1	1.909196	0	0	17764	-0.030371	A7	0	0	1	1	23	2	2	17764.0
349	350	0	3	Dimic, Mr. Jovan	1	0.891820	0	0	315088	-0.474005	NaN	2	8	1	1	16	3	2	315088.0
454	455	0	3	Peduzzi, Mr. Joseph	1	-0.035354	0	0	A/5 2817	-0.486337	NaN	2	8	1	1	19	3	2	2817.0
656	657	0	3	Radeff, Mr. Alexander	1	-0.304686	0	0	349223	-0.489442	NaN	2	8	1	1	21	2	2	349223.0
90	91	0	3	Christmann, Mr. Emil	1	-0.052885	0	0	343276	-0.486337	NaN	2	8	1	1	20	3	2	343276.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
500	501	0	3	Calic, Mr. Petar	1	-0.924922	0	0	315086	-0.474005	NaN	2	8	1	1	16	3	2	315086.0
121	122	0	3	Moore, Mr. Leonard Charles	1	-0.560356	0	0	A4. 54510	-0.486337	NaN	2	8	1	1	26	2	2	54510.0
596	597	1	2	Leitch, Miss. Jessie Wills	0	0.209652	0	0	248727	0.016023	NaN	2	8	1	1	26	2	1	248727.0
595	596	0	3	Van Impe, Mr. Jean Baptiste	1	0.455802	1	1	345773	-0.162169	NaN	2	8	3	2	27	2	2	345773.0
				Hansen, Mr.															

Conclusion:

The Titanic Survival Project, explored through a series of code snippets and topics, provides valuable insights into the world of data science and machine learning. This project encompasses various aspects of data analysis, preprocessing, model building, and evaluation, reflecting the essential steps in a typical data science workflow.

One of the primary tasks in this project was data preprocessing. The code showcased how to handle missing values, encode categorical variables, and engineer new features. These preprocessing steps are vital for ensuring the quality and readiness of data for machine learning model training. The careful selection of features and their encoding for the predictive models demonstrated the importance of feature engineering in enhancing model performance.

The project introduced several machine learning algorithms, including Random Forest, AdaBoost, and Logistic Regression. Each algorithm was employed to build predictive models for determining passenger survival on the Titanic. Cross-validation techniques were used to evaluate model performance and assess their generalization capabilities. The choice of appropriate evaluation metrics, such as the F1 score, underscored the importance of selecting metrics that align with the specific problem and dataset.

Furthermore, the project delved into ensemble learning by using a Voting Classifier to combine the predictions of multiple base models, enhancing the overall predictive power. This demonstrated how ensemble methods can often yield improved results by leveraging the strengths of individual classifiers.

The project's conclusion highlights the synergy of data preprocessing, machine learning, and ensemble modeling in making predictions and decisions based on the Titanic dataset. These skills and techniques are transferable to broader data science and machine learning projects, providing a solid foundation for tackling real-world problems.

In essence, the Titanic Survival Project is not just about predicting survival on a historical shipwreck but serves as a comprehensive learning journey. It illustrates the iterative and interactive nature of data analysis and model development, emphasizing the need for domain expertise, careful data handling, and robust model evaluation. This project underscores the critical role of data science and machine learning in extracting meaningful insights and making informed decisions from data, which can be applied to a wide range of domains and challenges.

References:-

sk-learn
Titanic Dataset
https://youtu.be/6P3HSOcCYPc?si=JZMPNY46wJJUKkIU