In [110…
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [111…
```python
# loading the Dataset
titanic_df = pd.read_csv(r'C:\Users\admin\Desktop\2213557-MA 336\Titanic-Dataset.csv')
```

In [82]:
```python
# Explore the dataset
titanic_df.head()
```

Out[82]:

	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

In [83]:
```python
# Check for missing values
titanic_df.isnull().sum()
```

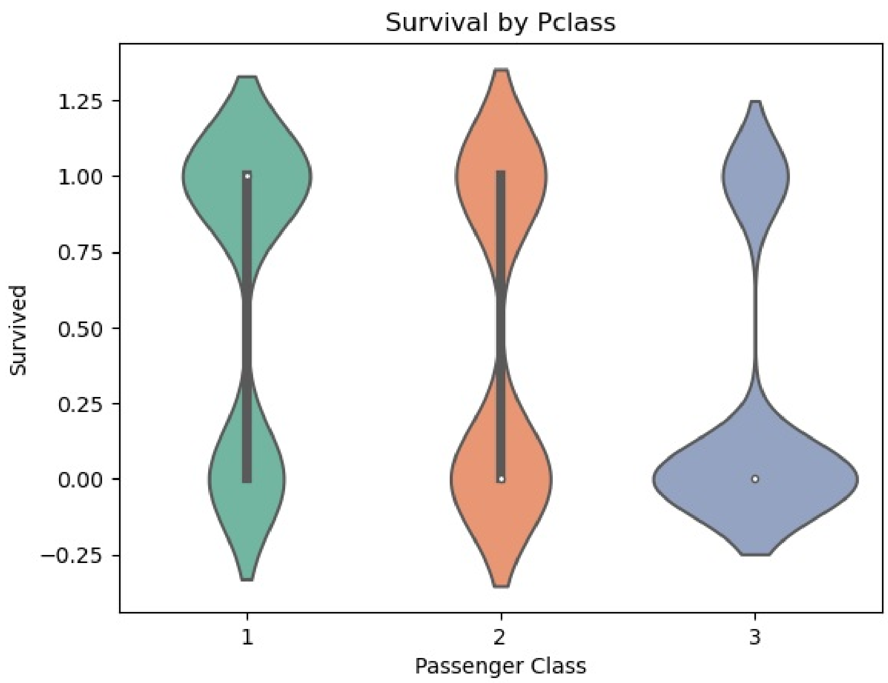
Out[83]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [84]:
```python
# Handle missing values
# For simplicity,  drop the 'Cabin' column and filling missing values in 'Age' with the mean
titanic_df.drop('Cabin', axis=1, inplace=True)
titanic_df['Age'].fillna(titanic_df['Age'].mean(), inplace=True)
```

In [85]:
```python
# Encoding categorical variables like 'Sex' and 'Embarked'
label_encoder = LabelEncoder()
titanic_df['Sex'] = label_encoder.fit_transform(titanic_df['Sex'])
titanic_df['Embarked'] = label_encoder.fit_transform(titanic_df['Embarked'].astype(str))
```

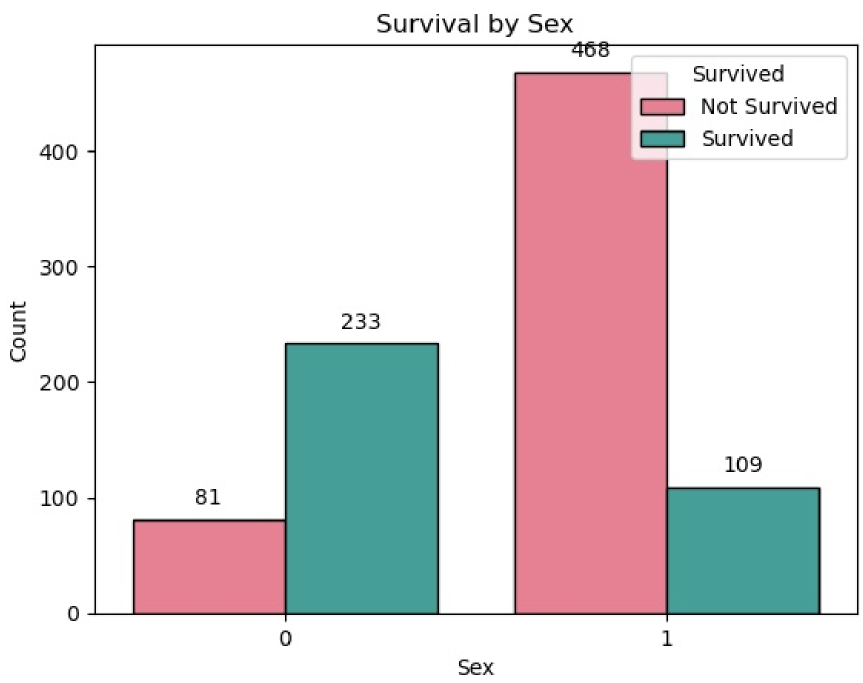
EXPLORATORY DATA ANALYSIS

In [113…
```python
sns.violinplot(x='Pclass', y='Survived', data=titanic_df, palette='Set2')
plt.title('Survival by Pclass')
plt.xlabel('Passenger Class')
plt.ylabel('Survived')
plt.show()
```



The plot visually represents how the survival status is distributed across different passenger classes ('Pclass'). Here, we can clearly see that passengers with Pclass 3 have higher survival rate comapare to 1st class and 2nd class.

In [87]:
```
# Stacked bar plot for 'Sex' with hue='Survived'
ax = sns.countplot(x='Sex', hue='Survived', data=titanic_df, palette='husl', edgecolor='black')

# Add annotations to the bars
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()), ha='center', va='center'

plt.title('Survival by Sex')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.legend(title='Survived', loc='upper right', labels=['Not Survived', 'Survived'])
plt.show()
```



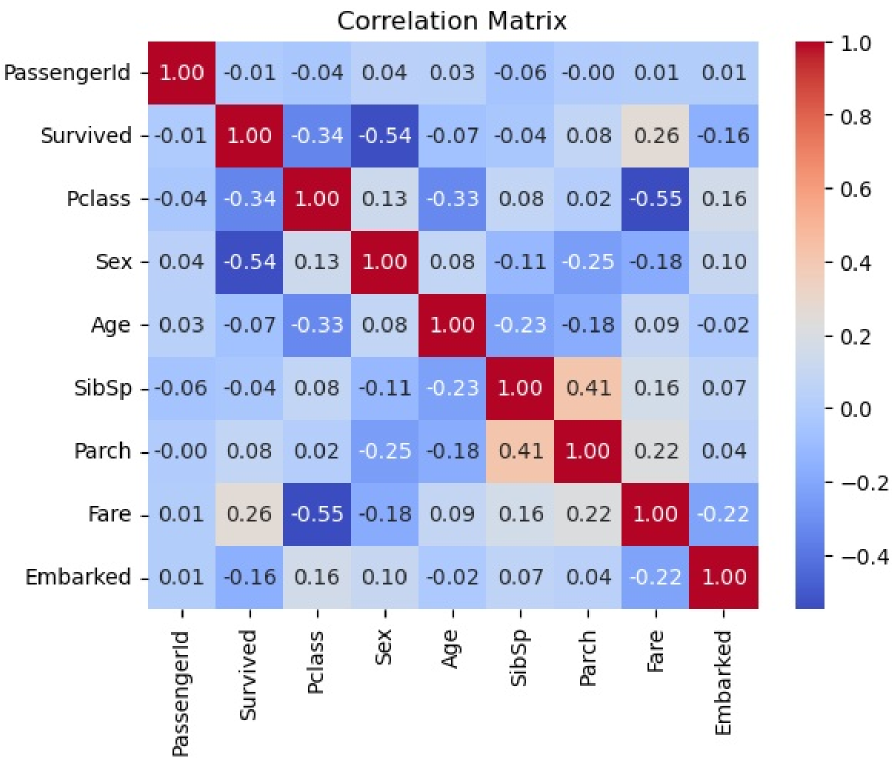
The stacked bar plot illustrates the distribution of survival outcomes based on gender ('Sex') in the Titanic dataset. The analysis reveals distinct patterns between male and female passengers. For male passengers, the count of those who did not survive is 81, while 233 male passengers survived. On the other hand, among female passengers, 468 did not survive, and 109 successfully survived. This visual representation underscores the significant impact of gender on survival outcomes, showcasing a notable difference in survival rates between male and female passengers aboard the Titanic.

In [109…
```
import seaborn as sns
import matplotlib.pyplot as plt

# Define age groups
age_bins = [0, 18, 35, 50, float('inf')]  # 'inf' represents positive infinity
```

```
age_labels = ['0-18', '19-35', '36-50', '51+']

# Create a new column 'AgeGroup' in the DataFrame
titanic_df['AgeGroup'] = pd.cut(titanic_df['Age'], bins=age_bins, labels=age_labels, include_lowest=True)

# Calculate survival rate by age group
survival_rate_by_age = titanic_df.groupby('AgeGroup')['Survived'].mean().reset_index()

# Line chart for survival rate by age group
sns.lineplot(x='AgeGroup', y='Survived', data=survival_rate_by_age, marker='o', color='blue', linewidth=2)
plt.title('Survival Rate by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Survival Rate')
plt.show()
```

The line chart vividly illustrates the disparity in survival rates across distinct age groups. Notably, the age group 0-18 emerges with the highest survival rate.

In [89]:
```
# Correlation matrix
correlation_matrix = titanic_df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



Feature Selection

In [90]:
```
# Select features for the model
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']

# Define the target variable
target = 'Survived'

# Create feature matrix (X) and target vector (y)
X = titanic_df[features]
y = titanic_df[target]
```

Split the Data into Training and Testing Sets

In [91]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Build and Train the Model

In [92]:
```
# Create and train the RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

Out[92]:
```
RandomForestClassifier(random_state=42)
```

Make predictions

In [93]:
```
# Predictions on the test set
y_pred = model.predict(X_test)
```

Evaluate the Model

In [94]:
```
# Check accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Classification Report and Confusion Matrix
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

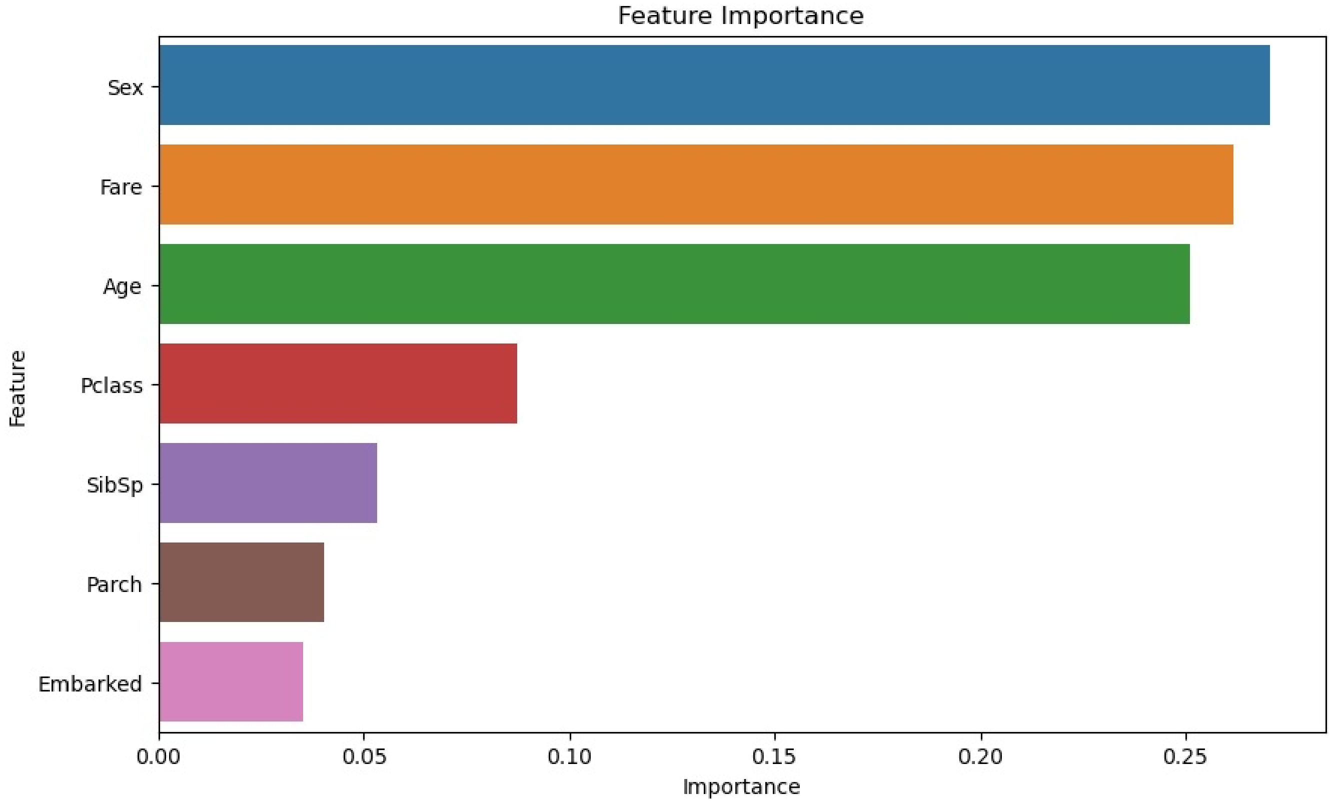
```
Accuracy: 0.82
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       105
           1       0.81      0.73      0.77        74

    accuracy                           0.82       179
   macro avg       0.81      0.80      0.81       179
weighted avg       0.82      0.82      0.81       179

[[92 13]
 [20 54]]
```

The Random Forest model, deployed to predict survival outcomes in the dataset, demonstrated an overall accuracy of 82%. This signifies that the model correctly classified approximately 82% of the instances, reflecting a reasonable level of predictive performance. A closer examination of precision, recall, and F1-scores for each class reveals a slightly better performance for predicting instances labeled as Not Survived (class 0) compared to Survived (class 1). Specifically, the model achieved a precision of 0.82 and a recall of 0.88 for Not Survived instances, indicating a high degree of accuracy in identifying passengers who did not survive. Conversely, the model exhibited a precision of 0.81 and a recall of 0.73 for Survived instances, indicating that while it identified a substantial portion of survivors, there were instances of false negatives. The confusion matrix further highlights this balance, with 92 instances correctly predicted as Not Survived, 54 correctly predicted as Survived, 13 instances incorrectly predicted as Survived, and 20 instances incorrectly predicted as Not Survived.

Feature Importance

In [95]:
```
# Feature Importance
feature_importance = model.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importance})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Plot feature importance
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance')
```

```
plt.show()
```



The feature importance plot highlights the key factors that influenced the Random Forest model's predictions. 'Sex', 'Fare', and 'Age' emerge as particularly impactful features, aligning with historical expectations regarding the Titanic disaster

Decision tree clasifier - Hyperparameter Tuning and Evaluation using GridSearchCV

In [105…

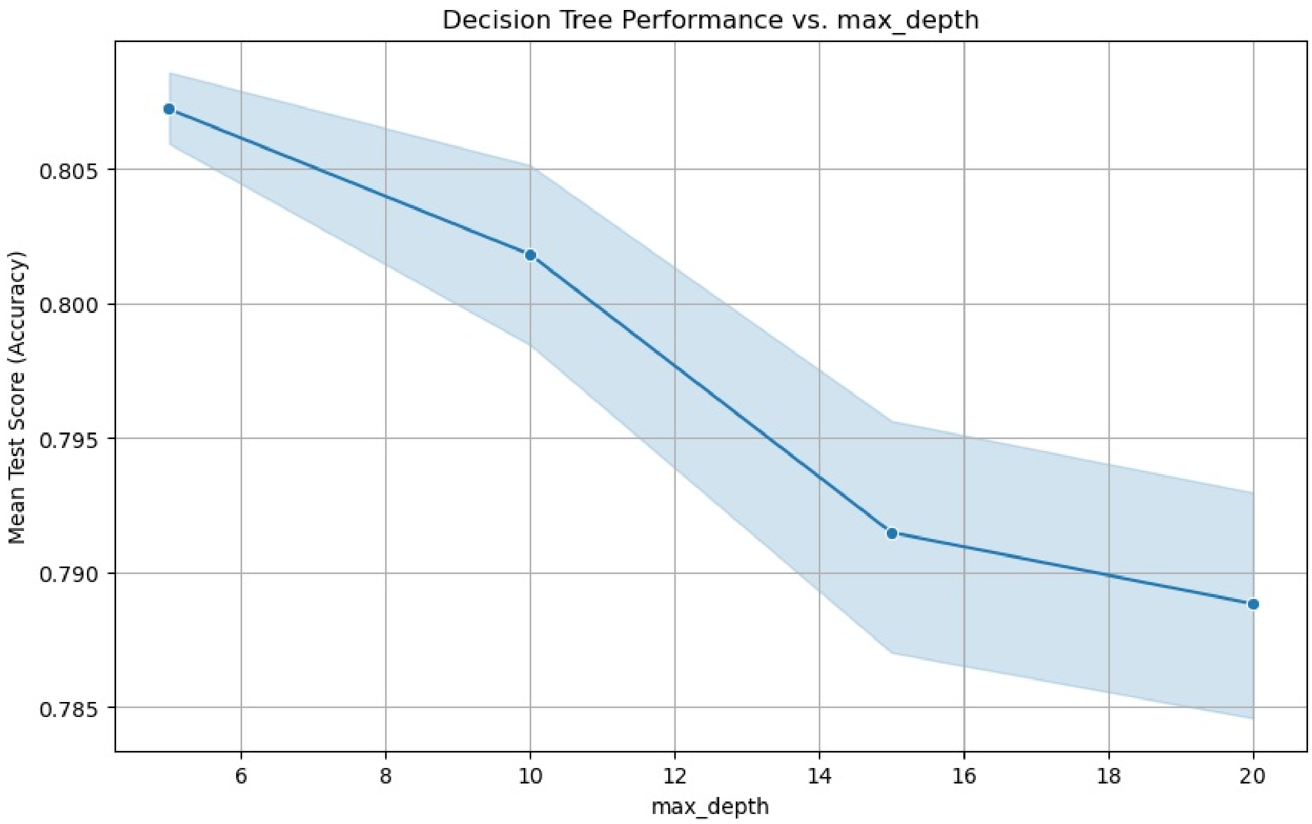
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

# Create a DecisionTreeClassifier
dt_model = DecisionTreeClassifier(random_state=42)

# Define the parameter grid to search
param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Use GridSearchCV to find the best parameters
grid_search = GridSearchCV(dt_model, param_grid, cv=5, scoring='accuracy', verbose=1)
grid_search.fit(X_train, y_train)

# Display the best parameters
print("Best Parameters:", grid_search.best_params_)

# Get the best model
best_dt_model = grid_search.best_estimator_

# Predictions on the test set using the best model
y_best_dt_pred = best_dt_model.predict(X_test)

# Evaluate the best model
accuracy_best_dt = accuracy_score(y_test, y_best_dt_pred)
print(f'Best Decision Tree Accuracy: {accuracy_best_dt:.2f}')

# Display classification report and confusion matrix for the best model
print(classification_report(y_test, y_best_dt_pred))
print(confusion_matrix(y_test, y_best_dt_pred))
```

```
Fitting 5 folds for each of 180 candidates, totalling 900 fits
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10, 'spl
itter': 'best'}
Best Decision Tree Accuracy: 0.83
              precision    recall  f1-score   support

           0       0.82      0.91      0.86       105
           1       0.85      0.72      0.78        74

    accuracy                           0.83       179
   macro avg       0.84      0.82      0.82       179
weighted avg       0.83      0.83      0.83       179

[[96  9]
 [21 53]]
```

Visualization of the Decision Tree Model

In [104…

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Extract the results from the grid search
results = pd.DataFrame(grid_search.cv_results_)

# Plot the performance against different values of max_depth
plt.figure(figsize=(10, 6))
sns.lineplot(x='param_max_depth', y='mean_test_score', data=results, marker='o')

plt.title('Decision Tree Performance vs. max_depth')
plt.xlabel('max_depth')
plt.ylabel('Mean Test Score (Accuracy)')
plt.grid(True)
plt.show()
```



The model, with the best hyperparameters, is performing well with an accuracy of 83%. Class 0 (Not Survived) has higher precision, recall, and F1-score compared to class 1 (Survived). The confusion matrix provides a detailed breakdown of correct and incorrect predictions across both classes. Overall, this Decision Tree model, tuned with the specified hyperparameters, demonstrates a strong performance in predicting survival outcomes on the given dataset.

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js