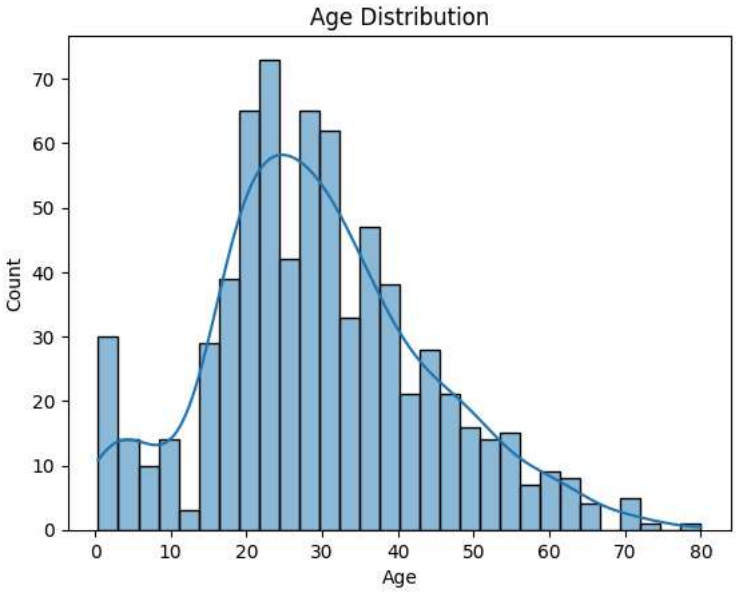
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_text
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
df = pd.read_csv("train.csv")


df.info()
df.describe()
```

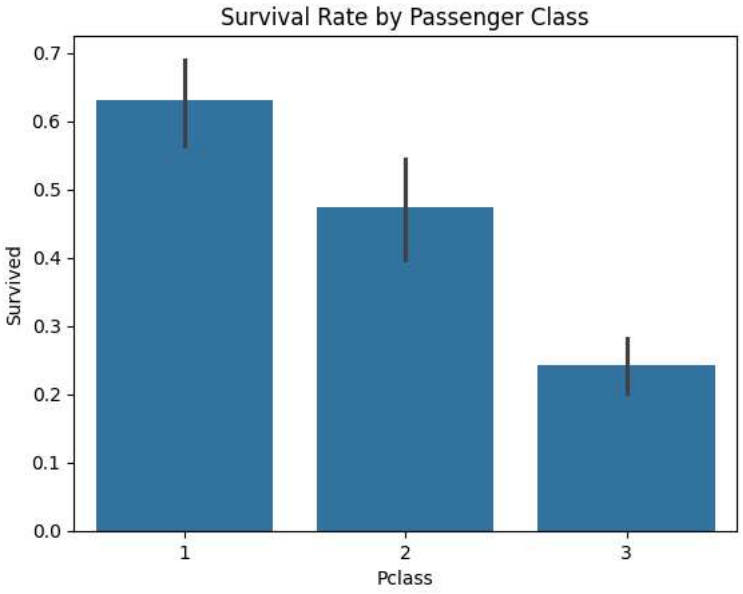
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    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
```

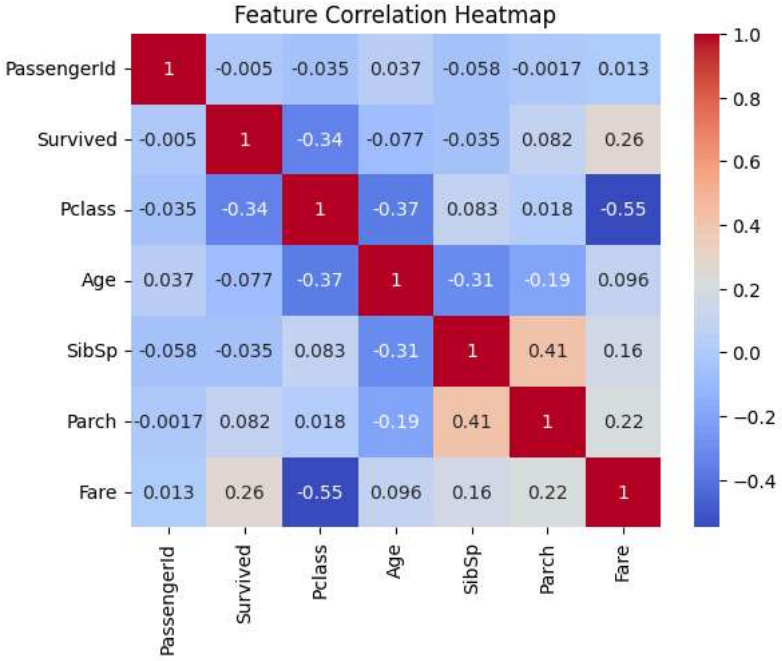
| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|
| **count** | 891.000000 | 891.000000 | 891.000000 | 714.000000 | 891.000000 | 891.000000 | 891.000000 |
| **mean** | 446.000000 | 0.383838 | 2.308642 | 29.699118 | 0.523008 | 0.381594 | 32.204208 |
| **std** | 257.353842 | 0.486592 | 0.836071 | 14.526497 | 1.102743 | 0.806057 | 49.693429 |
| **min** | 1.000000 | 0.000000 | 1.000000 | 0.420000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 223.500000 | 0.000000 | 2.000000 | 20.125000 | 0.000000 | 0.000000 | 7.910400 |
| **50%** | 446.000000 | 0.000000 | 3.000000 | 28.000000 | 0.000000 | 0.000000 | 14.454200 |
| **75%** | 668.500000 | 1.000000 | 3.000000 | 38.000000 | 1.000000 | 0.000000 | 31.000000 |
| **max** | 891.000000 | 1.000000 | 3.000000 | 80.000000 | 8.000000 | 6.000000 | 512.329200 |

```
sns.histplot(df['Age'].dropna(), bins=30, kde=True)
plt.title("Age Distribution")
plt.show()

# Compare survival rate across Pclass
sns.barplot(x="Pclass", y="Survived", data=df)
plt.title("Survival Rate by Passenger Class")
plt.show()

# Display correlation heatmap
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```

[+]


Age Distribution
Count
Age
0
10
20
30
40
50
60
70
80


Survival Rate by Passenger Class
Survived
Pclass
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1
2
3


Feature Correlation Heatmap
PassengerId
Survived
Pclass
Age
SibSp
Parch
Fare
1 -0.005 -0.035 0.037 -0.058 -0.0017 0.013
-0.005 1 -0.34 -0.077 -0.035 0.082 0.26
-0.035 -0.34 1 -0.37 0.083 0.018 -0.55
0.037 -0.077 -0.37 1 -0.31 -0.19 0.096
-0.058 -0.035 0.083 -0.31 1 0.41 0.16
-0.0017 0.082 0.018 -0.19 0.41 1 0.22
0.013 0.26 -0.55 0.096 0.16 0.22 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

```
# Part 2: Handling Missing Values
# Extract titles from names
df['Title'] = df['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)

title_age_map = df.groupby('Title')['Age'].median()
df['Age'] = df.apply(lambda row: title_age_map[row['Title']] if pd.isnull(row['Age']) else row['Age'], axis=1)

# Fill missing Fare values
df['Fare'] = df.groupby('Pclass')['Fare'].transform(lambda x: x.fillna(x.median()))

# Fill missing Embarked values
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)

# Drop Cabin column
df.drop(columns=['Cabin'], inplace=True)
```

```
<ipython-input-13-cfa57c79762e>:12: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assigr
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


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

```
# Part 3: Feature Engineering
# Create FamilySize
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1

# Convert Age into categories
df['AgeCategory'] = pd.cut(df['Age'], bins=[0, 12, 18, 60, 100], labels=['Child', 'Teen', 'Adult', 'Senior'])

# Convert FamilySize into categories
df['FamilyCategory'] = pd.cut(df['FamilySize'], bins=[0, 1, 4, 20], labels=['Single', 'Small Family', 'Large Family'])


df = pd.get_dummies(df, columns=['Sex', 'Embarked', 'Pclass', 'FamilyCategory', 'AgeCategory'], drop_first=True)


X = df[['Sex_male', 'Age', 'Pclass_2', 'Pclass_3', 'Fare', 'FamilySize']]
y = df['Survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Decision Tree
model = DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=42)
model.fit(X_train, y_train)

# Evaluate Model
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Export Decision Tree
print(export_text(model, feature_names=list(X.columns)))
```

```
Model Accuracy: 81.56%
|--- Sex_male <= 0.50
|   |--- Pclass_3 <= 0.50
|   |   |--- Age <= 2.50
|   |   |   |--- Pclass_2 <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- Pclass_2 >  0.50
|   |   |   |   |--- class: 1
|   |   |--- Age >  2.50
|   |   |   |--- Age <= 27.50
|   |   |   |   |--- class: 1
|   |   |   |--- Age >  27.50
|   |   |   |   |--- class: 1
|   |--- Pclass_3 >  0.50
|   |   |--- Fare <= 23.35
|   |   |   |--- Fare <= 7.74
|   |   |   |   |--- class: 1
|   |   |   |--- Fare >  7.74
|   |   |   |   |--- class: 1
|   |   |--- Fare >  23.35
|   |   |   |--- Age <= 5.50
|   |   |   |   |--- class: 0
|   |   |   |--- Age >  5.50
|   |   |   |   |--- class: 0
```

```
|--- Sex_male >  0.50
|   |--- Age <= 6.50
|   |   |--- FamilySize <= 4.50
|   |   |   |--- class: 1
|   |   |--- FamilySize >  4.50
|   |   |   |--- FamilySize <= 6.50
|   |   |   |   |--- class: 0
|   |   |   |--- FamilySize >  6.50
|   |   |   |   |--- class: 0
|   |--- Age >  6.50
|   |   |--- Fare <= 26.27
|   |   |   |--- Age <= 13.50
|   |   |   |   |--- class: 1
|   |   |   |--- Age >  13.50
|   |   |   |   |--- class: 0
|   |   |--- Fare >  26.27
|   |   |   |--- FamilySize <= 5.00
|   |   |   |   |--- class: 0
|   |   |   |--- FamilySize >  5.00
|   |   |   |   |--- class: 0
```

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print(f"Random Forest Accuracy: {rf_accuracy * 100:.2f}%")
```

```
Random Forest Accuracy: 82.12%
```