Research Project

* **EDA**: Started by looking into data distributions. Found the age distribution to be slightly right-skewed. Ticket class seemed to influence survival rates.
* **Data Processing**: Addressed missing values based on their nature. Created a family size feature which might be a better representative than having siblings/spouses and parents/children separately.
* **Training Model**: Used a Random Forest classifier as it's robust to outliers and can capture complex relationships.
* **Model Validation**: Accuracy was good, but the focus should be more on precision and recall, especially in the context of predicting survival.

Code:  
  
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import OneHotEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

#1. EDA (Exploratory Data Analysis)

# Load the dataset

data = pd.read\_csv('titanic\_passengers.csv')

# Get an overview of the data

print(data.head())

print(data.info())

#Visualizations

# Histogram for age distribution

sns.histplot(data['Age'].dropna(), kde=True)

plt.title('Age Distribution')

plt.show()

# Bar plot for the survival rate with respect to Pclass (Ticket class)

sns.countplot(x='Pclass', hue='Survived', data=data)

plt.title('Survival by Ticket Class')

plt.show()

#Interesting Find: Age seems to be slightly right skewed, and the higher class passengers (Pclass = 1) have a higher survival rate.

#2. Data Processing

#Handling Missing Data

# Check for missing values

print(data.isnull().sum())

# Fill Age missing values with median

data['Age'].fillna(data['Age'].median(), inplace=True)

# Drop Cabin column due to excessive missing values

data.drop(columns='Cabin', inplace=True)

# Fill Embarked with mode (most common value)

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

#Feature Engineering

# Create a family size feature

data['FamilySize'] = data['SibSp'] + data['Parch']

# Drop unnecessary columns

data.drop(columns=['Name', 'Ticket', 'PassengerId'], inplace=True)

#3. Train Model

# Convert categorical columns to one-hot encoding

data\_encoded = pd.get\_dummies(data, drop\_first=True)

X = data\_encoded.drop(columns='Survived')

y = data\_encoded['Survived']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

#A Random Forest is essentially a collection of decision trees. Thought it was very interesting.

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

#4. Model Validation

y\_pred = clf.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

Visualizations:

* A screenshot of a graph

  Description automatically generated

A screenshot of a computer screen

Description automatically generated  
  
  
**Technique Summary**: Introduced one-hot encoding for categorical variables, used RandomForest for initial modeling.