

project1

July 12, 2025

1 Practical Machine Learning and Data Exploration

1.1 Project 1 - Titanic-Dataset

1.1.1 Student: Sara Ismail

```
[1]: import pandas as pd
import numpy as np
```

```
[4]: # Load dataset
df = pd.read_csv(r"C:\Users\some0\Desktop\Master\Practical_Machine_Learning_and_Data_Exploration/
↳Titanic-Dataset.csv")
# Preview first 5 rows
df.head()
```

```
[4]: PassengerId  Survived  Pclass  \
0             1         0         3
1             2         1         1
2             3         1         3
3             4         1         1
4             5         0         3
```

```
                                Name    Sex  Age  SibSp  \
0                Braund, Mr. Owen Harris  male  22.0    1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0    1
2                Heikkinen, Miss. Laina  female  26.0    0
3  Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0    1
4                Allen, Mr. William Henry  male  35.0    0
```

```
    Parch    Ticket   Fare Cabin Embarked
0      0  A/5 21171   7.2500   NaN        S
1      0  PC 17599  71.2833   C85        C
2      0 STON/O2. 3101282   7.9250   NaN        S
3      0   113803  53.1000  C123        S
4      0   373450   8.0500   NaN        S
```

2 Identify Data Types of Features

```
[5]: df.dtypes
```

```
[5]: PassengerId      int64
Survived            int64
Pclass             int64
Name               object
Sex               object
Age              float64
SibSp             int64
Parch            int64
Ticket            object
Fare             float64
Cabin            object
Embarked          object
dtype: object
```

```
[6]: df.describe()
```

```
[6]:
```

	PassengerId	Survived	Pclass	Age	SibSp \
count	891.000000	891.000000	891.000000	714.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008
std	257.353842	0.486592	0.836071	14.526497	1.102743
min	1.000000	0.000000	1.000000	0.420000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000
50%	446.000000	0.000000	3.000000	28.000000	0.000000
75%	668.500000	1.000000	3.000000	38.000000	1.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000

	Parch	Fare
count	891.000000	891.000000
mean	0.381594	32.204208
std	0.806057	49.693429
min	0.000000	0.000000
25%	0.000000	7.910400
50%	0.000000	14.454200
75%	0.000000	31.000000
max	6.000000	512.329200

3 Handling missing values

3.1 Count of Missing Values

```
[8]: df.isnull().sum()
```

```
[8]: 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
```

3.2 Dropping Columns With >30% Missing Data

```
[9]: missing_percentage = df.isnull().mean() * 100
      threshold = 30
      # Filter columns
      columns_to_drop = missing_percentage[missing_percentage > threshold].index

      # Drop columns
      df = df.drop(columns=columns_to_drop)

      # Results
      print(f"Columns dropped: {list(columns_to_drop)}")
      print(f"-----")
      print(f"Remaining columns: {df.columns}")
```

```
Columns dropped: ['Cabin']
```

```
-----
```

```
Remaining columns: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex',
                          'Age', 'SibSp',
                          'Parch', 'Ticket', 'Fare', 'Embarked'],
                          dtype='object')
```

3.3 Pre-processing on dataset

```
[10]: from sklearn.preprocessing import LabelEncoder
      label_encoder = LabelEncoder()

      for column in df.select_dtypes(include=['object']).columns:
          df[column] = label_encoder.fit_transform(df[column])

      df.head()
```

```
[10]:
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	\
0	1	0	3	108	1	22.0	1	0	523	
1	2	1	1	190	0	38.0	1	0	596	
2	3	1	3	353	0	26.0	0	0	669	
3	4	1	1	272	0	35.0	1	0	49	
4	5	0	3	15	1	35.0	0	0	472	

	Fare	Embarked
0	7.2500	2
1	71.2833	0
2	7.9250	2
3	53.1000	2
4	8.0500	2

3.4 Handling Nulls Based on Data Type

```
[12]: # Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent value)
    df[column] = df[column].fillna(mean_value)

# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent value)
    df[column] = df[column].fillna(mode_value)

#Recount missing values
df.isnull().sum()
```

```
[12]: PassengerId    0
Survived           0
Pclass             0
Name               0
Sex                0
Age                0
SibSp              0
Parch              0
Ticket             0
Fare               0
Embarked           0
dtype: int64
```

4 Handling Outliers

4.1 Detect Outliers

```
[17]: def replace_outliers_with_nan(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Replace outliers with NaN
    return column.where((column >= lower_bound) & (column <= upper_bound), np.
↪nan)

# Replace outliers with NaN for numeric feature
for col in df.select_dtypes(include='number').columns:
    df[col] = replace_outliers_with_nan(df[col])

# Recount missing values
null_counts = df.isnull().sum()
print("Sum of null values in each column:")
print(null_counts)
```

Sum of null values in each column:

PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	130
SibSp	46
Parch	213
Ticket	0
Fare	146
Embarked	0

dtype: int64

5 Re-handling missing values

```
[20]: ## Dropping Columns With >30% Missing Data

missing_percentage = df.isnull().mean() * 100
threshold = 30
# Filter columns
columns_to_drop = missing_percentage[missing_percentage > threshold].index
```

```

# Drop columns
df = df.drop(columns=columns_to_drop)

# Results
print(f"Columns dropped: {list(columns_to_drop)}")
print(f"-----")
print(f"Remaining columns: {df.columns}")

#-----#

## Handling Nulls Based on Data Type

# Fill missing values in columns with mode
for column in df.select_dtypes(include=['float64']).columns:
    mean_value = df[column].mean() # Get the mode (most frequent value)
    df[column] = df[column].fillna(mean_value)

# Fill missing values in categorical columns with mode
for column in df.select_dtypes(include=['object']).columns:
    mode_value = df[column].mode()[0] # Get the mode (most frequent value)
    df[column] = df[column].fillna(mode_value)

#Recount missing values
print(f"-----")
print(f"Count of Missing Values")
df.isnull().sum()

```

Columns dropped: []

Remaining columns: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Embarked'], dtype='object')

Count of Missing Values

```

[20]: PassengerId    0
      Survived      0
      Pclass        0
      Name          0
      Sex           0
      Age           0
      SibSp         0
      Parch         0
      Ticket        0
      Fare          0
      Embarked      0

```

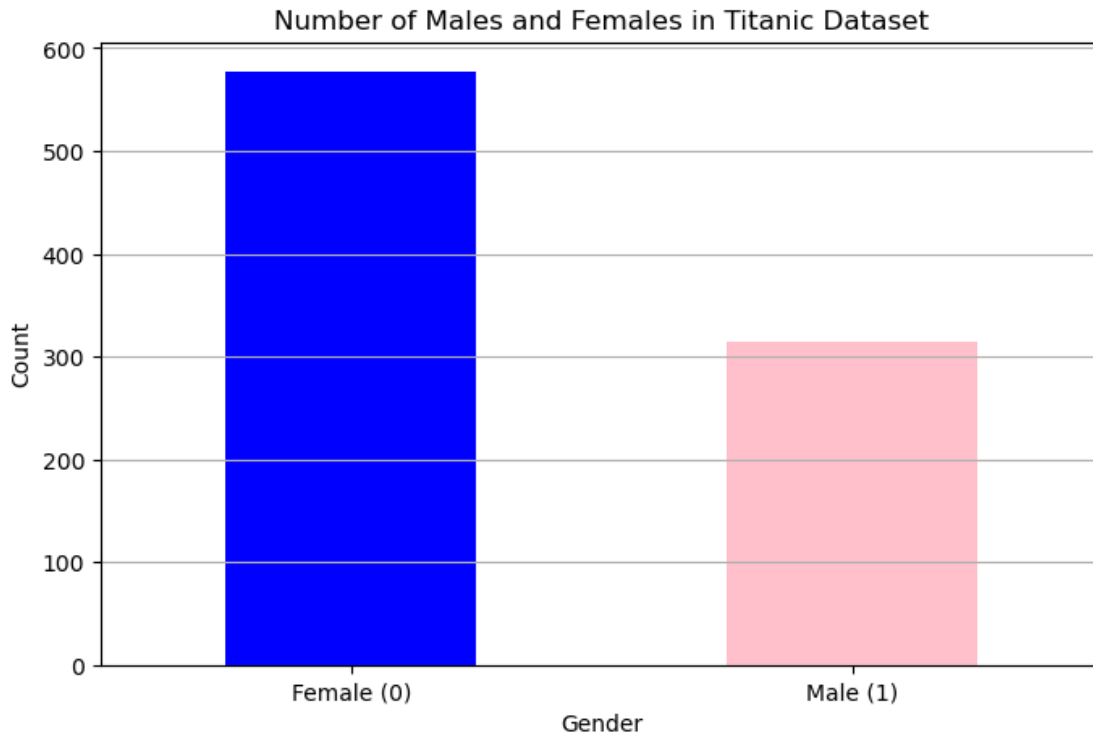
dtype: int64

6 Data Visualization

```
[ ]: from matplotlib import pyplot as plt
import seaborn as sns
```

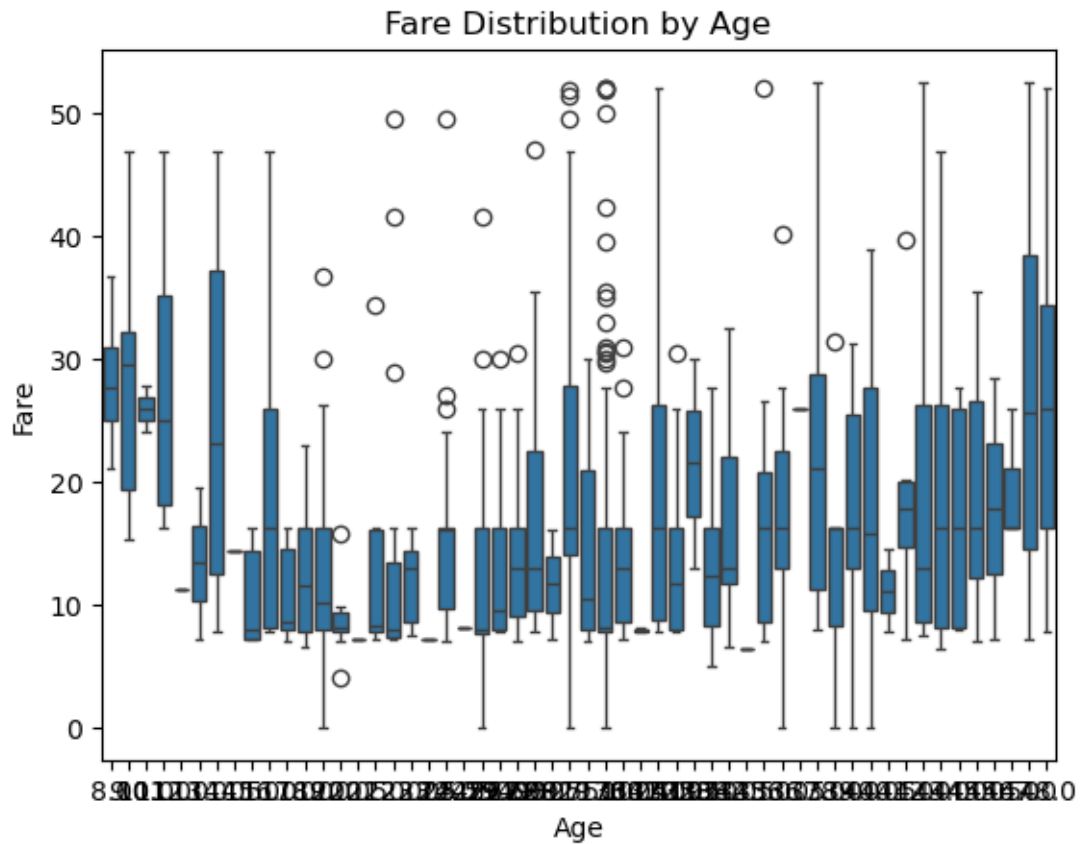
6.1 Number of Males and Females in Titanic Dataset

```
[39]: plt.figure(figsize=(8, 5))
gender_counts = df['Sex'].value_counts()
gender_counts.plot(kind='bar', color=['blue', 'pink'])
plt.xticks([0, 1], ['Female (0)', 'Male (1)'], rotation=0)
plt.title('Number of Males and Females in Titanic Dataset')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()
```



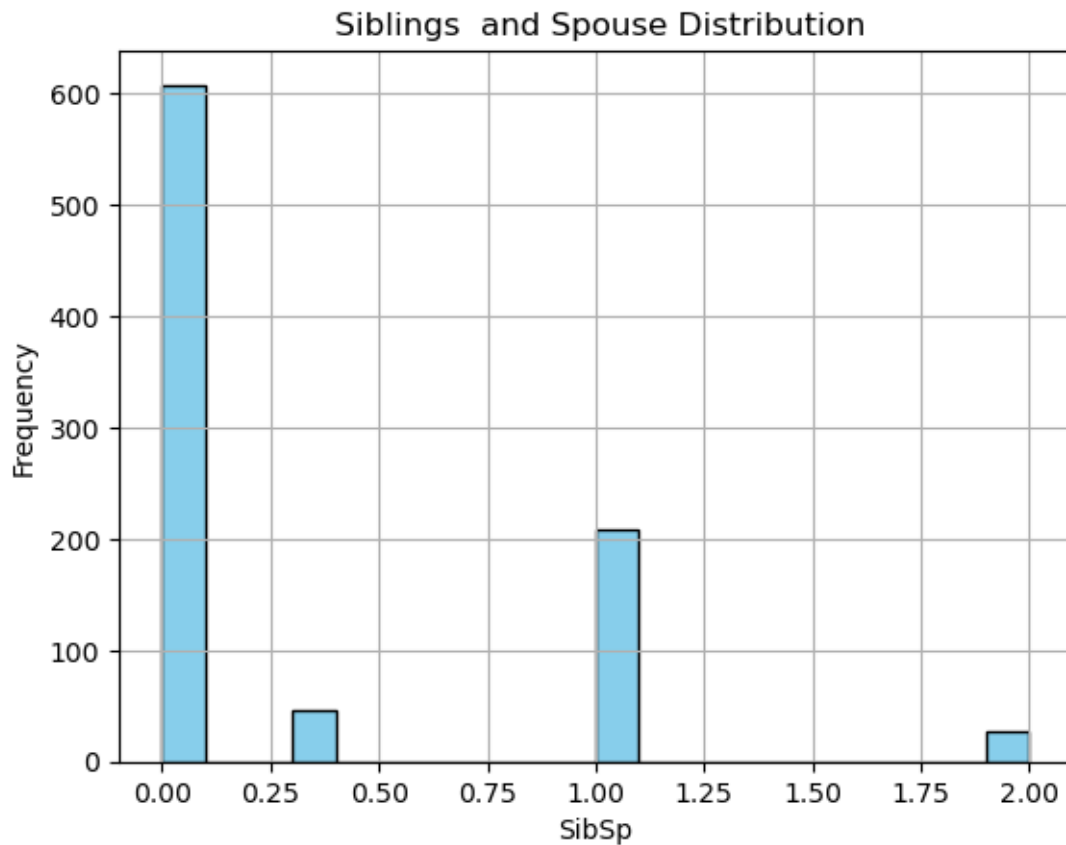
6.2 Fare Distribution by Age

```
[45]: sns.boxplot(x='Age', y='Fare', data=df)
plt.title("Fare Distribution by Age")
plt.show()
```



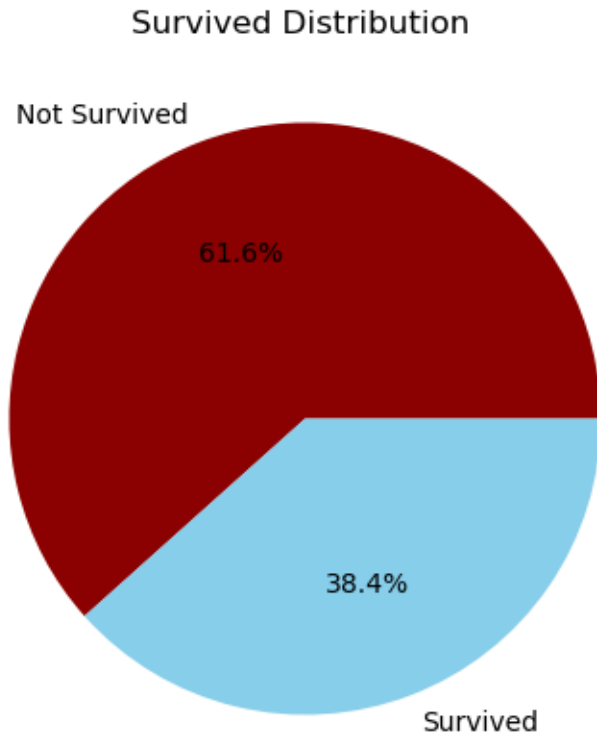
6.3 Siblings - Spouse

```
[44]: df['SibSp'].hist(bins=20, color='skyblue', edgecolor='black')
plt.title(" Siblings and Spouse Distribution")
plt.xlabel("SibSp")
plt.ylabel("Frequency")
plt.show()
```



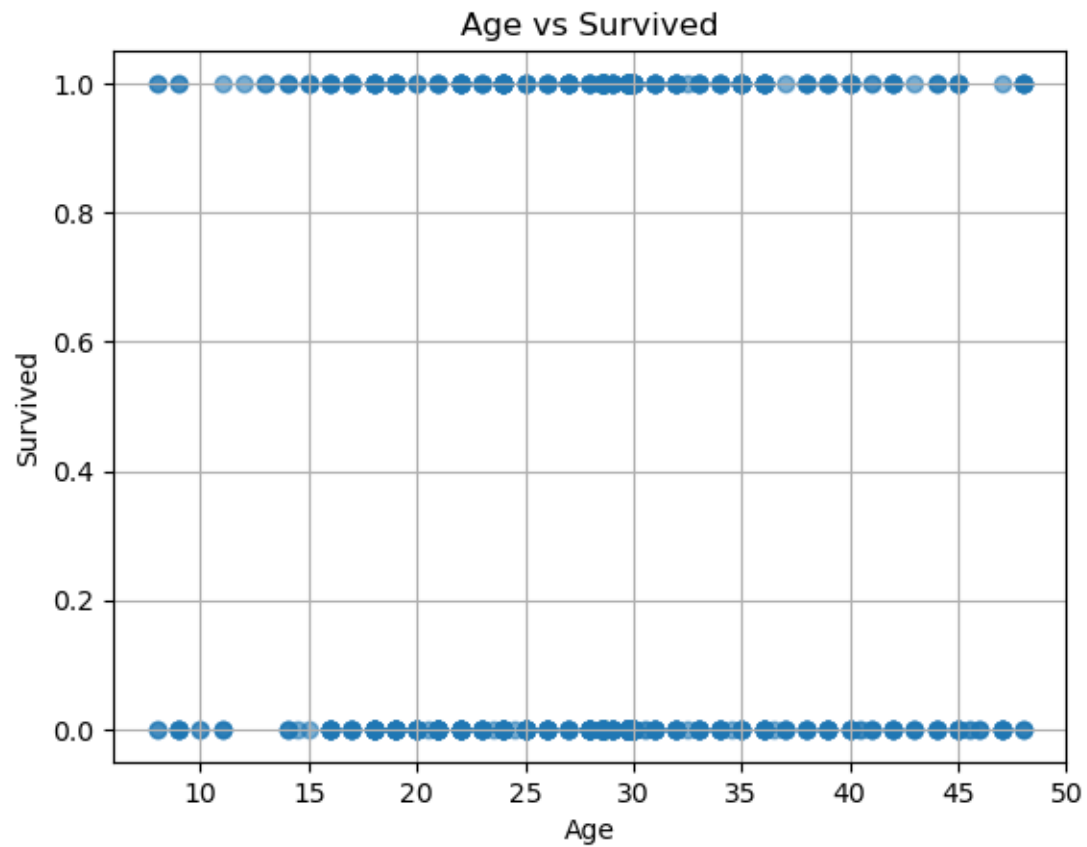
6.4 Survived Distribution

```
[36]: survived_counts = df['Survived'].value_counts()
plt.figure(figsize=(8, 5))
plt.pie(survived_counts, labels=['Not Survived', 'Survived'], autopct='%1.
    ↪1f%', colors=['darkred', 'skyblue'])
plt.title('Survived Distribution')
plt.show()
```



6.5 Age vs Survived

```
[38]: plt.scatter(df['Age'], df['Survived'], alpha=0.6)
plt.xlabel("Age")
plt.ylabel("Survived")
plt.title("Age vs Survived")
plt.grid(True)
plt.show()
```



6.6 Passenger Class Distribution

```
[40]: sns.countplot(x='Pclass', data=df)
plt.title("Distribution of Pclass")
plt.show()
```

