Assignment 13 Solution

Submitted by: Yameen Ali

Go to Github repository

Exploratory Data Analysis (EDA) on TITANIC Dataset

Importing Dependencies

In [2]:
```
import seaborn as sns
import numpy as np
import pandas as pd
```

Load Dataset from Seaborn

In [3]:
```
df = sns.load_dataset('titanic')
```

In [4]:
```
df
```

Out[4]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns

Data Preprocessing

Columns

In [5]:
```
df.columns
```

Out[5]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

Checking Missing Values

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

Missing Values Percentage

In [7]:
```
total_values = len(df)
null = df.isnull().sum()
missing_percentage = (null/total_values)*100
missing_percentage
```

Out[7]:
```
survived        0.000000
pclass          0.000000
sex             0.000000
age            19.865320
sibsp           0.000000
parch           0.000000
fare            0.000000
embarked        0.224467
class           0.000000
who             0.000000
adult_male      0.000000
deck           77.216611
embark_town     0.224467
alive           0.000000
alone           0.000000
dtype: float64
```

Drop Columns have missing values to balance the dataset

In [8]:
```
df.drop('deck',axis=1 , inplace=True)
```

In [9]:
```
df.drop('embarked',axis=1, inplace=True)
```

In [10]:
```
df.drop('alive',axis=1, inplace=True)
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   class        891 non-null    category
 8   who          891 non-null    object
 9   adult_male   891 non-null    bool
 10  embark_town  889 non-null    object
 11  alone        891 non-null    bool
dtypes: bool(2), category(1), float64(2), int64(4), object(3)
memory usage: 65.5+ KB
```

mode of categorical data in embarked_town column and fill null values

In [12]:
```
mod = df['embark_town'].mode()[0]
df['embark_town'].fillna(mod, inplace=True)
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   class        891 non-null    category
 8   who          891 non-null    object
 9   adult_male   891 non-null    bool
 10  embark_town  891 non-null    object
 11  alone        891 non-null    bool
dtypes: bool(2), category(1), float64(2), int64(4), object(3)
memory usage: 65.5+ KB
```

mean of numeric data and fill missing values

In [14]:
```
mea = df['age'].mean()
```

In [15]:
```
df['age'].fillna(mea,inplace=True)
```

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          891 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   class        891 non-null    category
 8   who          891 non-null    object
 9   adult_male   891 non-null    bool
 10  embark_town  891 non-null    object
 11  alone        891 non-null    bool
dtypes: bool(2), category(1), float64(2), int64(4), object(3)
memory usage: 65.5+ KB
```

---

1. Basic Overview:

- a) How many passengers are in the Titanic dataset?
- b) What are the column names of this dataset?
- c) Write the values of Survived column?

a) How many passengers are in the Titanic dataset?

In [17]:
```
total = len(df)
total
```

Out[17]:
```
891
```

b) What are the column names of this dataset?

In [18]:
```
df.columns
```

Out[18]:
```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare', 'class',
       'who', 'adult_male', 'embark_town', 'alone'],
      dtype='object')
```

c) Write the values of Survived column?

In [19]:
```
df['survived']
```

Out[19]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: survived, Length: 891, dtype: int64
```

---

2. Demographics:

- a) How many males and females were aboard the Titanic?
- b) Which age group had the highest number of passengers?
- c) Calculate the mean and median age of passengers using NumPy?

a) How many males and females were aboard the Titanic?

In [20]:
```
gender_counts = df['sex'].value_counts()
```

In [21]:
```
print("Number of Males and Females abroad the titanic")
print(gender_counts)
```

```
Number of Males and Females abroad the titanic
male      577
female    314
Name: sex, dtype: int64
```

b) Which age group had the highest number of passengers?

In [114…
```
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80]
labels = ['0-10', '11-20', '21-30', '31-40', '41-50', '51-60', '61-70', '71-80']
df['age_group'] = pd.cut(df['age'], bins, labels=labels)
```

In [115…
```
# Find the age group with the highest number of passengers
age_group_counts = df['age_group'].value_counts()
highest_age_group = age_group_counts.idxmax()
print("\nAge group with the highest number of passengers:")
print(highest_age_group)
```

```
Age group with the highest number of passengers:
21-30
```

c) Calculate the mean and median age of passengers using NumPy

In [117…
```
mean_age = np.nanmean(df['age'])
median_age = np.nanmedian(df['age'])
print(f"Mean age of passengers: {mean_age}")
print(f"Median age of passengers: {median_age}")
```

```
Mean age of passengers: 29.69911764705882
Median age of passengers: 29.69911764705882
```

3.Travel Class & Fare:

- a) How many passengers were in each class (1st, 2nd, 3rd)?
- b) What's the average fare for each class? Is there a correlation between fare and class?
- c) Calculate the standard deviation of fares using NumPy?

a) How many passengers were in each class (1st, 2nd, 3rd)

In [25]:
```
df['class'].value_counts()
```

Out[25]:
```
Third     491
First     216
Second    184
Name: class, dtype: int64
```

b) What's the average fare for each class? Is there a correlation between fare and class?

In [26]:
```
df.groupby('class')['fare'].mean()
```

Out[26]:
```
class
First     84.154687
Second    20.662183
Third     13.675550
Name: fare, dtype: float64
```

In [27]:
```
# Map class categories to numerical values
class_mapping = {'First': 1, 'Second': 2, 'Third': 3}
df['class_num'] = df['class'].map(class_mapping)

# Drop rows with NaN values in 'fare' or 'class_num'
df_filtered = df[['fare', 'class_num']].dropna()

# Calculate correlation
correlation = df_filtered['fare'].corr(df_filtered['class_num'])
print("\nCorrelation between fare and class (1st, 2nd, 3rd):")
print(correlation)
```

```
Correlation between fare and class (1st, 2nd, 3rd):
-0.5494996199439078
```

c) Calculate the standard deviation of fares using NumPy?

In [28]:
```
std_fare = np.nanstd(df['fare'])
```

In [29]:
```
std_fare = np.nanstd(df['fare'])
print("\nStandard deviation of fares:")
print(std_fare)
```

```
Standard deviation of fares:
49.6655344447741
```

---

4. Survival Analysis:

- a) What's the overall survival rate of passengers?
- b) Does gender influence the chances of survival?
- c) Which class had the highest survival rate?

a) What's the overall survival rate of passengers?

In [30]:
```
df['survived'].mean()
```

Out[30]:
```
0.3838383838383838
```

b) Does gender influence the chances of survival?

In [31]:
```
df.groupby('sex')['survived'].mean()
```

Out[31]:
```
sex
female    0.742038
male      0.188908
Name: survived, dtype: float64
```

c) Which class had the highest survival rate?

In [32]:
```
survival_by_class = df.groupby('class')['survived'].mean()
maximum_survival = survival_by_class.idxmax() # check maximum survival
print("\nSurvival rate by class:")
print(survival_by_class)
print("\nClass with the highest survival rate:")
print(maximum_survival)
```

```
Survival rate by class:
class
First     0.629630
Second    0.472826
Third     0.242363
Name: survived, dtype: float64

Class with the highest survival rate:
First
```

---

5. Embarkation:

- a) From which port did most passengers embark?
- b) Was there a relationship between embarkation port and survival rate?

a) From which port did most passengers embark?

In [33]:
```
port = df['embark_town'].value_counts()
max_embark = port.idxmax() # check maximum survival
print("\nSurvival rate by class:")
print(port)
print("\nClass with the highest survival rate:")
print(max_embark)
```

```
Survival rate by class:
Southampton    646
Cherbourg      168
Queenstown      77
Name: embark_town, dtype: int64

Class with the highest survival rate:
Southampton
```

b) Was there a relationship between embarkation port and survival rate?

In [34]:
```
df.groupby('embark_town')['survived'].mean()
```

Out[34]:
```
embark_town
Cherbourg      0.553571
Queenstown     0.389610
Southampton    0.339009
Name: survived, dtype: float64
```

---

6. Cabin & Accommodation:

- a) How many passengers had cabins? And how many did not?
- b) Is there any correlation between having a cabin and survival?

a) How many passengers had cabins? And how many did not?

In [80]:
```
# create a dummy 'cabin' column with some NaN values
np.random.seed(0)
df['cabin'] = np.where(np.random.rand(len(df)) > 0.5, 'C123',np.nan)

cabin_info = df['cabin'].notna().value_counts()
print(f"Paseengers with and without cabins: {cabin_info}")

df['has_cabin'] = df['cabin'].notna()
```

```
Paseengers with and without cabins: True    891
Name: cabin, dtype: int64
```

survival rate based on whether a passenger had a cabin or not

In [77]:
```
survival_rate = df.groupby('has_cabin')['survived'].mean()
print(survival_rate)
```

```
has_cabin
True    0.383838
Name: survived, dtype: float64
```

In [81]:
```
# Convert 'has_cabin' to numeric (True -> 1, False -> 0)
df['has_cabin_num'] = df['has_cabin'].astype(int)
```

```
cabin_survival_correlation = df['has_cabin_num'].corr(df['survived'])
print("\nCorrelation between having a cabin and survival:")
print(cabin_survival_correlation)
```

```
Correlation between having a cabin and survival:
nan
```

---

7. Family:

- a) How many passengers traveled alone, and how many traveled with family?
- b) Does traveling with family or alone influence the chances of survival?

a) How many passengers traveled alone, and how many traveled with family?

In [85]:
```
df
```

Out[85]:

	survived	pclass	sex	age	sibsp	parch	fare	class	who	adult_male	embark_town	alone	age_group	class_num	ca
0	0	3	male	22.000000	1	0	7.2500	Third	man	True	Southampton	False	21-30	3	C
1	1	1	female	38.000000	1	0	71.2833	First	woman	False	Cherbourg	False	31-40	1	C
2	1	3	female	26.000000	0	0	7.9250	Third	woman	False	Southampton	True	21-30	3	C
3	1	1	female	35.000000	1	0	53.1000	First	woman	False	Southampton	False	31-40	1	C
4	0	3	male	35.000000	0	0	8.0500	Third	man	True	Southampton	True	31-40	3	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.000000	0	0	13.0000	Second	man	True	Southampton	True	21-30	2	C
887	1	1	female	19.000000	0	0	30.0000	First	woman	False	Southampton	True	11-20	1	C
888	0	3	female	29.699118	1	2	23.4500	Third	woman	False	Southampton	False	21-30	3	
889	1	1	male	26.000000	0	0	30.0000	First	man	True	Cherbourg	True	21-30	1	C
890	0	3	male	32.000000	0	0	7.7500	Third	man	True	Queenstown	True	31-40	3	

891 rows × 18 columns

In [138…
```
df['family_size'] = df['sibsp'] + df['parch']
```

In [139…
```
travel_alone = df['family_size'] == 0
travel_with_family = df['family_size'] > 0
```

In [140…
```
travel_alone.sum()
```

Out[140]:
```
537
```

In [141…
```
travel_with_family.sum()
```

Out[141]:
```
354
```

In [142…
```
df['family_size'].sum()
```

Out[142]:
```
806
```

b) Does traveling with family or alone influence the chances of survival?

In [143…
```
survival_alone = df[travel_alone]['survived'].mean()
```

In [147…
```
survival_with_family = df[travel_with_family]['survived'].mean()
```

In [148…
```
print("Survival rate for passengers who traveled alone:", survival_alone)
print("Survival rate for passengers who traveled with family:", survival_with_family)
```

```
Survival rate for passengers who traveled alone: 0.30353817504655495
Survival rate for passengers who traveled with family: 0.5056497175141242
```

---

8. Feature Engineering:

- a) Can you create a new feature "IsChild" which is True for passengers under 10 and False otherwise?

- b) How does being a child relate to the chances of survival?

a) Can you create a new feature "IsChild" which is True for passengers under 10 and False otherwise?

In [161…
```
df['IsChild'] = df['age'] < 10
print(df[['age','IsChild']].head(10))
```

```
         age  IsChild
0  22.000000    False
1  38.000000    False
2  26.000000    False
3  35.000000    False
4  35.000000    False
5  29.699118    False
6  54.000000    False
7   2.000000     True
8  27.000000    False
9  14.000000    False
```

In [164…
```
df['IsChild'] = (df['age'] < 10 ).astype(int)
print(df[['age','IsChild']].head(10))
# 0 for False and 1 for True
```

```
         age  IsChild
0  22.000000        0
1  38.000000        0
2  26.000000        0
3  35.000000        0
4  35.000000        0
5  29.699118        0
6  54.000000        0
7   2.000000        1
8  27.000000        0
9  14.000000        0
```

b) How does being a child relate to the chances of survival?

In [165…
```
df.groupby('IsChild')['survived'].mean()
```

Out[165]:
```
IsChild
0    0.366707
1    0.612903
Name: survived, dtype: float64
```

This indicates that children had a higher survival rate compared to adults, suggesting that being a child did have a positive influence on the chances of survival.
