E23CSEU0055_Lab01_CSET301(P)

Name - Arihant Gupta - E23CSEU0055

Batch - EB02

Date - 31.07.2025

Ipynb File - cset301-aiml-sem5/jul29/one.ipynb at main · GuptajiRocks/cset301-aiml-sem5

1. Load the Dataset

a.
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
import matplotlib.pyplot as plt

df =
pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv")
```

2. Identify and Handle Missing Values

a.
```
# Code
print(df.isnull().sum())

#Output
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
```

Checking if any null values are present. We can see columns “Age”, “Cabin”, “Embarked” are having null values. Age is the only numeric value, so we will impute the NaN values with the mean of the age column.

```
df["Age"] = df["Age"].fillna(df["Age"].mean())

#Age NaN values have been imputed with mean now.
```

Since the column Cabin has very high NaN values, is categorical in nature with every entry unique and not affecting the result in any way whatsoever. We will drop the cabin column and create a new DataFrame without it.

b.
```
# Code
df_n = df.drop("Cabin", axis=1)
print(df_n.isnull().sum())

#Output
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       2
dtype: int64
```

3. Deal with Duplicate Data

a.
```
# Code
print(df.duplicated().sum())

#Output
0
```

4. Converting Categorical Column to Numerical

a. We can see Age has no null values, Cabin has been dropped. Now comes Embarked column. Embarked acts as the class label and has three unique values namely “E”, “C” and “S”. To fill in the NaN value, I will impute a value “Z” as it would not mean anything but will remove the NaN. We’ll apply Label Encoding to the column and convert it to numerical value for further analysis.

b.
```
#Code
print(df_n[df_n["Embarked"].isnull()])

#Output
PassengerId  Survived  Pclass                                      Name  \
61            62         1       1                       Icard, Miss. Amelie
829          830         1       1  Stone, Mrs. George Nelson (Martha Evelyn)
```

```
        Sex   Age  SibSp  Parch  Ticket  Fare Embarked
61   female  38.0      0      0  113572  80.0      NaN
829  female  62.0      0      0  113572  80.0      NaN
```

c.
```
# Fillna with "Z"
df_n["Embarked"] = df_n["Embarked"].fillna("Z")

le = LabelEncoder()
df_n["Embarked"] = le.fit_transform(df_n["Embarked"])
print(df_n["Embarked"].unique())

#Output
[2 0 1 3]
```

The unique numerical values of Embarked column suggest that the categorical data has been converted to numerical form.

5. Normalize appropriate numerical features.

a.
```
stdC = StandardScaler()
df_n["Age"] = stdC.fit_transform(df_n[["Age"]])
```

Age has been normalized using Z-score normalization - (StandardScaler() from Scikit-Learn)

6. Apply sorting and filtering logic

a. I will sort the values based on Age.

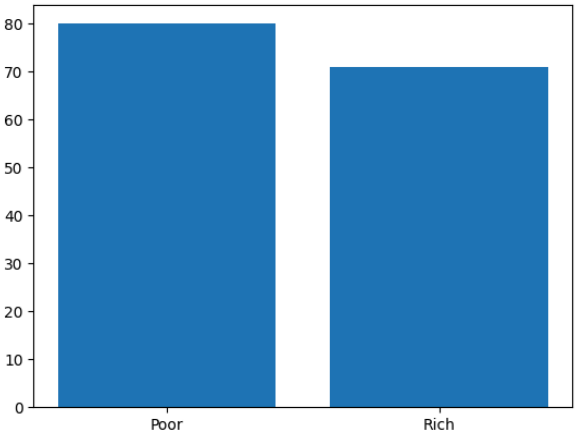
i.
```
# Sorting
print(df_n[["Name", "Sex","Age"]].sort_values(by="Age"))

#Output
Name     Sex       Age
803       Thomas, Master. Assad Alexander    male -2.253155
755             Hamalainen, Master. Viljo    male -2.233917
644                Baclini, Miss. Eugenie  female -2.227761
469          Baclini, Miss. Helene Barbara  female -2.227761
831       Richards, Master. George Sibley    male -2.221604
..                                     ...     ...       ...
116                 Connors, Mr. Patrick    male  3.139805
96              Goldschmidt, Mr. George B    male  3.178283
493               Artagaveytia, Mr. Ramon    male  3.178283
851                   Svensson, Mr. Johan    male  3.409146
630  Barkworth, Mr. Algernon Henry Wilson    male  3.870872

[891 rows x 3 columns]
```

7. Engineer new column based on Logic.
    a. Filtering out the people who have Fare more than the average fare. And creating a new column of Status, wherein people who have fare higher than average are rich, and others are poor.
    b.

```
#Code
mufare = df_n["Fare"].mean()
df_n["Status"] = df["Fare"].apply(lambda x : "Rich" if x > mufare else "Poor")
print(df_n[["Fare", "Status"]])

#Output
Fare Status
0     7.2500   Poor
1    71.2833   Rich
2     7.9250   Poor
3    53.1000   Rich
4     8.0500   Poor
..       ...    ...
886  13.0000   Poor
887  30.0000   Poor
888  23.4500   Poor
889  30.0000   Poor
890   7.7500   Poor

[891 rows x 2 columns]
```

8. Visualization
    a. Creating a Bar chart comparing the Age and Status of members in the data.
    b. We are comparing status with Age that is before normalization to get a numerical understanding.
    c.

```
#Code
print(df["Age"])
plt.bar(df_n["Status"], height=df["Age"])
plt.show()

```



We can see that Poor people have Age range from 0 to 80, and Rich People have Age range from 0 to 70.