Assignment 2 – Data cleaning, preparation, and transformation On Titanic Dataset

Section: ZBB
Group 3:

- Eunice Chua
- Tareq Haboukh
- Fides Audrielle Urgel
- Liezel Anne Viray

The purpose of this assignment is practice data preparation in Python This assignment provides you with an opportunity to demonstrate the achievement of the following course learning outcomes:

- Understand key data preparation steps.
- Be able to manipulate data in Python.
- Know most essential Python libraries and functions.

Instructions

1. Download Titanic dataset

Work in Python

In [24]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

file_path = "Titanic.csv"
Titanic = pd.read_csv(file_path)
df = pd.DataFrame(Titanic)
```

2. Apply following data preparation rules, note that additional documentation on Python is available at

https://docs.python.org

https://matplotlib.org

https://pandas.pydata.org/pandas-docs/stable

# Drop the columns where all elements are missing values:

In [25]:

```
df.dropna(axis=1, how='all')
df.head(2)
```

Out[25]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

# Drop the columns where any of the elements are missing values:

In [26]:

```
df.dropna(axis=1, how='any')
df.head(2)
```

Out[26]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

# Keep only the rows which contain 2 missing values maximum:

In [27]:

```
df.dropna(axis=1, thresh=2)
df.head(2)
```

Out[27]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

# Fill all missing values with the mean of the particular column:

In [28]:

```
# We choose to amend missing values with the mean of Age
df['Age'] = df['Age'].fillna(df['Age'].mean())
df.head(2)
```

Out[28]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

# Fill any missing value in column 'A' with the column median:

In [29]:

```
# df['Age'] = df['Age'].fillna(df['Age'].median())
df['Age'].fillna(df['Age'].median())
df.head(2)
```

Out[29]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C

3. Look at the statistical summary of the prepared dataset:

In [30]:

```
print(df.describe().round(2))
```

```
       PassengerId  Survived  Pclass    Age  SibSp  Parch    Fare
count       891.00    891.00  891.00 891.00 891.00 891.00  891.00
mean        446.00      0.38    2.31  29.70   0.52   0.38   32.20
std         257.35      0.49    0.84  13.00   1.10   0.81   49.69
min           1.00      0.00    1.00   0.42   0.00   0.00    0.00
25%         223.50      0.00    2.00  22.00   0.00   0.00    7.91
50%         446.00      0.00    3.00  29.70   0.00   0.00   14.45
75%         668.50      1.00    3.00  35.00   1.00   0.00   31.00
max         891.00      1.00    3.00  80.00   8.00   6.00  512.33
```

4. Look at the statistical summary for categorical variables:

In [31]:

```
categorical = df.dtypes[df.dtypes == "object"].index
print(categorical)
df[categorical].describe()
```

```
Index(['Name', 'Sex', 'Ticket', 'Cabin', 'Embarked'], dtype='object')
```

Out[31]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

5. Identify categorical variables and explain why they cannot be used in the raw form in the analysis.

Categorical variables: Name, Sex, Ticket, Cabin and Embarked
These cannot be used in the raw form because it cannot be directly interpreted in machine learning models. It cannot fit into a regression equation without treating it.

6. Transform one of the categorical variables into numerical using label encoder method.

In [32]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
EmbarkedEncoded = le.fit_transform(df['Embarked'])
df['EmbarkedEncoded'] = EmbarkedEncoded
# df.drop('Embarked', axis=1, inplace=True)
df.head(2)
```

Out[32]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	EmbarkedEncoded
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	0

7. Transform gender into numerical variable using dummy coding.

In [33]:

```
df_new = pd.get_dummies(df, columns=["Sex"])
# df_new = df_new.drop('Sex_male', axis=1, inplace=True)
df_new.head(2)
```

Out[33]:

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	EmbarkedEncoded	Sex_female	Sex_male
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN	S	2	0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85	C	0	1	0

Additional data cleaning:

- PassengerId, Ticket, Embarked, and Sex_female should be dropped.
  The first is just an arbitrary integer value, the second one is distinct for every passenger, Embarked is encoded, and only 1 feature need for Sex

In [34]:
```
df_new.drop(['Ticket', 'PassengerId','Embarked', 'Sex_female'], axis=1, inplace=True)
```

- Title (Mr., Mrs., Miss,etc) should be extracted from Name, and those should be further converted into 0 if the title is common (Mr., Miss.) and 1 if it isn't (Dr., Rev., Capt.).

In [35]:
```
df_new['Title'] = df_new['Name'].apply(lambda x: x.split(',')[1].strip().split(' ')[0])
df_new['Title'] = [0 if x in ['Mr.', 'Miss.', 'Mrs.'] else 1 for x in df_new['Title']]
df_new = df_new.rename(columns={'Title': 'TitleUnusual'})
```

- Finally, Name should be dropped.

In [36]:
```
df_new.drop('Name', axis=1, inplace=True)
```

- Cabin should be replaced with Cabin_Known — 0 if the value is NaN otherwise 1.

In [37]:
```
df_new['Cabin_Known'] = [0 if str(x) == 'nan' else 1 for x in df_new['Cabin']]
df_new.drop('Cabin', axis=1, inplace=True)
```

In [38]:
```
# Dummy columns should be created from Embarked and first dummy column should be dropped to avoid collinearity issues.
# emb_dummies = pd.get_dummies(df_new['Embarked'], drop_first=True, prefix='Embarked')
# df_new = pd.concat([df_new, emb_dummies], axis=1)
# df_new.drop('Embarked', axis=1, inplace=True)
```

In [39]:
```
df_new.head()
```

Out[39]:

	Survived	Pclass	Age	SibSp	Parch	Fare	EmbarkedEncoded	Sex_male	TitleUnusual	Cabin_Known
0	0	3	22.0	1	0	7.2500	2	1	0	0
1	1	1	38.0	1	0	71.2833	0	0	0	1
2	1	3	26.0	0	0	7.9250	2	0	0	0
3	1	1	35.0	1	0	53.1000	2	0	0	1
4	0	3	35.0	0	0	8.0500	2	1	0	0

8. The target variable is Survived. You need to define the best subset of features, containing the most useful features and explain your reasons.

In [40]:
```
features = df_new.columns
features
```

Out[40]:
```
Index(['Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare',
       'EmbarkedEncoded', 'Sex_male', 'TitleUnusual', 'Cabin_Known'],
      dtype='object')
```

In [41]:
```
# Low Variance Filter
def low_variance_filter(df_new, threshold):
    variance = df_new.var()
    features = df_new.columns
    selectedVariables = []
    for i in range(0, len(df_new.columns)):
        if variance[i] >= threshold:
            selectedVariables.append(features[i])
    return selectedVariables
```

In [42]:
```
feature_selection_varianceThreshold = low_variance_filter(df_new, 0.2)
df_new[feature_selection_varianceThreshold].shape
```

Out[42]:
```
(891, 8)
```

In [43]:
```
from sklearn.feature_selection import VarianceThreshold

selector = VarianceThreshold(threshold=0.2)
```

```
selected_features = selector.fit_transform(df_new)
selector.get_params()
```

Out[43]:

```
{'threshold': 0.2}
```

Using Variance threshold filter method, the following Selected features are selected while removed those that are lower than set threshold. 0.20 is the threshold we've used. The assumption here is that those 8 features have higher variance possibly contain useful information for prediction.

In [44]:

```
from itertools import compress
features_selected_VarianceThreshold = list(compress(df_new.columns, selector.get_support()))

print("Selected Features:")
for i in range(len(features_selected_VarianceThreshold)):
    print(features_selected_VarianceThreshold[i])
```

```
Selected Features:
Survived
Pclass
Age
SibSp
Parch
Fare
EmbarkedEncoded
Sex_male
```

9. Fit the regression model and provide the performance measures.

In [45]:

```
from sklearn.model_selection import train_test_split

# Create data and target variables
df_new = df_new.filter(features_selected_VarianceThreshold)
y = df_new.loc[:, df_new.columns == 'Survived']
x = df_new.loc[:, df_new.columns != 'Survived']
# x = df_new.loc[:, features_selected_VarianceThreshold.columns != 'Survived']


# Split data into train and test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

In [46]:

```
# Perform Logistic Regression
from sklearn.linear_model import LogisticRegression

lr_model = LogisticRegression(solver='lbfgs', max_iter=1000)
lr_model.fit(x_train, y_train.values.ravel())
```

Out[46]:

```
LogisticRegression(max_iter=1000)
```

In [47]:

```
# Import Performance measures
from sklearn.metrics import classification_report, accuracy_score, mean_squared_error, confusion_matrix

y_pred = lr_model.predict(x_test)
```

In [48]:

```
# Model Classification Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       163
           1       0.79      0.70      0.74       105

    accuracy                           0.81       268
   macro avg       0.81      0.79      0.80       268
weighted avg       0.81      0.81      0.81       268
```

In [49]:

```
# Model Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of the logistic regression model on test set: %3f" %accuracy)
```

```
Accuracy of the logistic regression model on test set: 0.809701
```

In [50]:

```
# Calculate Mean Squared Error
mse = mean_squared_error(y_test,y_pred)
print(mse.round(3))
```

```
0.19
```

In [51]:

```
# Confusion Matrix
confusion_matrix(y_test, y_pred)
```

Out[51]:

```
array([[144,  19],
       [ 32,  73]], dtype=int64)
```

Submission Instructions:

Please pay attention to the following tips.

- You can google to learn more about how label encoder and dummy coding work, and what is the difference.

- You should submit a PDF file containing the answers to each question.
- On top of the first page, provide the name of all your group members.
- Please submit your work before the due date, February 27th, by the end of the day.