# CST8502 - Lab 5
# Python Exercise

**Student Name: Alvin Litani Liauw**
**Student Number: 041118874**

**For every step, include screenshot of the code and the results in this document (screenshot from colab/jupyter notebook). Also, in your words, explain your code and results. If there is no explanation, no marks will be given. No need to write long paragraphs, but one or 2 lines per step.**

**1**

```
# Read CSV file and put it inside dataframe
df = pd.read_csv('train.csv')

# Show number of attributes
print(f'Number of attributes: {df.shape[1]}')

# Show names of attributes
print(f'Name of attributes: {df.columns}')

# Show number of instances
print(f'Number of instances: {df.shape[0]}')

# Show columns and top 5 rows
df.head()
```

```
Number of attributes: 12
Name of attributes: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
Number of instances: 891
```

| | 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 |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | S |
| **3** | 4 | 1 | 1 | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| **4** | 5 | 0 | 3 | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

**2**

```
# Check for duplicates after dropping PassengerId
df = df.drop(columns='PassengerId')
print(f'Number of duplicated rows: {df.duplicated().sum()}')

# Show only columns with missing data
missing_data = df.isnull().sum()
print(f'{missing_data[missing_data > 0]}')

# Check columns that have more than 50% missing data
missing_proportion = df.isnull().mean()
columns_to_drop = missing_proportion[missing_proportion > 0.5]
```

```
Number of attributes: 7
Name of attributes: Index(['Survived', 'Pclass', 'Sex', 'Fare', 'Embarked', 'AgeGroup',
       'Relatives'],
      dtype='object')
```

**3**

```
# Drop those columns together with "Name" and "Ticket" columns
df = df.drop(columns=['Name','Ticket'] + list(columns_to_drop.index))
```

```
Number of duplicated rows: 0
Age         177
Cabin       687
Embarked      2
dtype: int64
```

**4**

```
# Bin Age column into AgeGroup
df['AgeGroup'] = df['Age'].apply(lambda x:
                        'NK' if pd.isnull(x) else
                        'Child' if x < 16 else
                        'Youth' if x < 30 else
                        'Adult' if x < 65 else
                        'Senior')


# Define the bins and labels
```

```
bins = [-float('inf'), 0, 3, float('inf')]
labels = ['None', 'Few', 'Many']

# Create new 'Relatives' column based on number of relatives
df['Relatives'] = pd.cut(df['Parch'] + df['SibSp'] , bins=bins, labels=labels)

df['Fare'] = df['Fare'].apply(lambda x:
                              'Free' if x == 0 else
                              'Low' if x < 50 else
                              'Average' if x < 100 else
                              'High')

# Drop SibSp, Parch, Age columns
df = df.drop(columns=['SibSp','Parch','Age'])
print(f'Number of attributes: {df.shape[1]}')
print(f'Name of attributes: {df.columns}')
```

```
Number of attributes: 7
Name of attributes: Index(['Survived', 'Pclass', 'Sex', 'Fare', 'Embarked', 'AgeGroup',
       'Relatives'],
      dtype='object')
```

**5**

```
# Do one-hot encoding
data_encoded = pd.get_dummies(df,drop_first = True)
data_encoded.head()
```

| | Survived | Pclass | Sex_male | Fare_Free | Fare_High | Fare_Low | Embarked_Q | Embarked_S | AgeGroup_Child | AgeGroup_NK | AgeGroup_Senior | AgeGroup_Youth | Relatives |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | True | False | False | True | False | True | False | False | False | True | |
| **1** | 1 | 1 | False | False | False | False | False | False | False | False | False | False | |
| **2** | 1 | 3 | False | False | False | True | False | True | False | False | False | True | |
| **3** | 1 | 1 | False | False | False | False | False | True | False | False | False | False | |
| **4** | 0 | 3 | True | False | False | True | False | True | False | False | False | False | |

**6**

```
# Split data into training and testing data
label = data_encoded['Survived']
attributes = data_encoded.drop(columns='Survived')
```

**attributes_train, attributes_test, label_train, label_test = train_test_split(attributes, label, test_size=0.3)**

**attributes_train.info()**
**label_train.info()**

```
<class 'pandas.core.frame.DataFrame'>
Index: 623 entries, 846 to 848
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Pclass            623 non-null    int64
 1   Sex_male          623 non-null    bool
 2   Fare_Free         623 non-null    bool
 3   Fare_High         623 non-null    bool
 4   Fare_Low          623 non-null    bool
 5   Embarked_Q        623 non-null    bool
 6   Embarked_S        623 non-null    bool
 7   AgeGroup_Child    623 non-null    bool
 8   AgeGroup_NK       623 non-null    bool
 9   AgeGroup_Senior   623 non-null    bool
 10  AgeGroup_Youth    623 non-null    bool
 11  Relatives_Few     623 non-null    bool
 12  Relatives_Many    623 non-null    bool
dtypes: bool(12), int64(1)
memory usage: 17.0 KB
<class 'pandas.core.series.Series'>
Index: 623 entries, 846 to 848
Series name: Survived
Non-Null Count  Dtype
--------------  -----
623 non-null    int64
dtypes: int64(1)
memory usage: 9.7 KB
```

**7**

**# Train the model on the training data**
**model = DecisionTreeClassifier(max_depth=3)**
**model.fit(attributes_train, label_train)**

**8**

**label_pred = model.predict(attributes_test)**
**print(f'Number of survivors in testing set: {label_pred.sum()} out of {label_pred.size}')**

```
Number of survivors in testing set: 99 out of 268
```

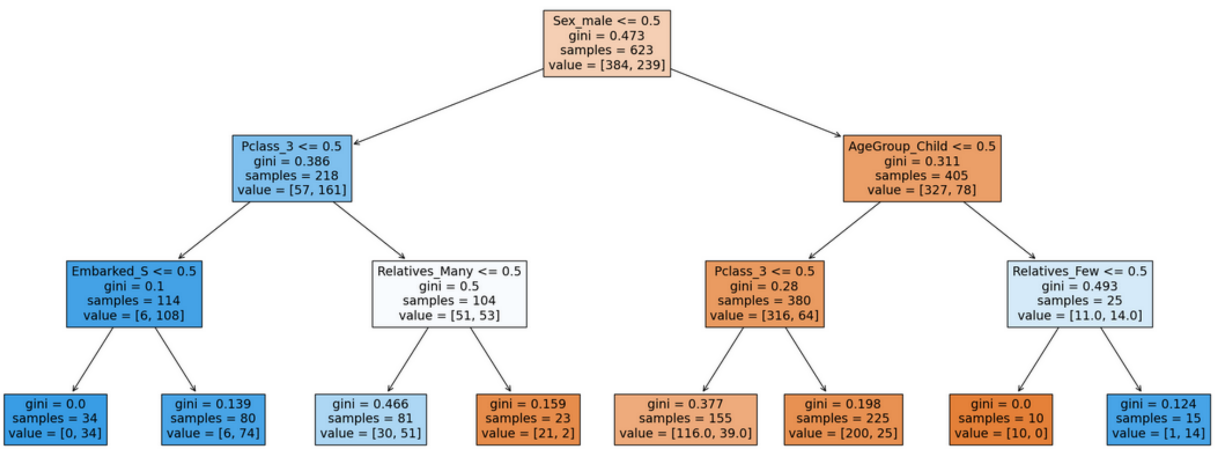
**9**

```
# Show accuracy and confusion matrix
accuracy = accuracy_score(label_test, label_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

ConfusionMatrix = confusion_matrix(label_test,label_pred)
print(ConfusionMatrix)

# Create the tree diagram
fig = plt.figure(figsize=(25,10))
tree.plot_tree(model, feature_names=attributes_train.columns, filled=True)
plt.show()
```

```
Accuracy: 82.46%
[[144  21]
 [ 26  77]]
```




**10**

```
# Split data into training and testing data
label = data_encoded['Survived']
attributes = data_encoded.drop(columns='Survived')

attributes_train, attributes_test, label_train, label_test = train_test_split(attributes, label,
test_size=0.3)
```

11

```
# Scale the training data
scaler = StandardScaler()
attributes_train = scaler.fit_transform(attributes_train)
attributes_test = scaler.transform(attributes_test)
```

12

```
# Do one-hot encoding
data_encoded = pd.get_dummies(df,drop_first = True)
print(data_encoded)
```

13

```
# Initialize the k-NN classifier with a chosen k value (e.g., k=5)
knn = KNeighborsClassifier(n_neighbors=5)

# Train the k-NN model on the training data
knn.fit(attributes_train, label_train)

# Predict on the test set
label_pred = knn.predict(attributes_test)

# Show accuracy and confusion matrix for testing set
accuracy = accuracy_score(label_test, label_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

ConfusionMatrix = confusion_matrix(label_test,label_pred)
print(ConfusionMatrix)
```

```
Accuracy: 76.87%
[[135  33]
 [ 29  71]]
```

14

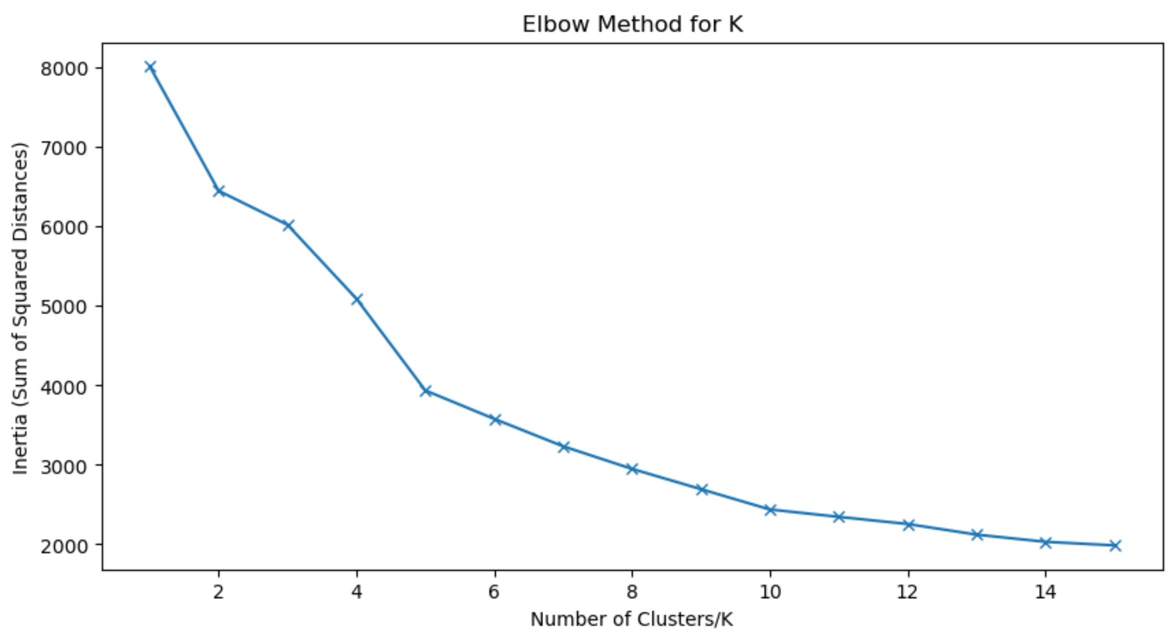
```
# Standardize the data
scaler = StandardScaler()
df_scaled = scaler.fit_transform(data_encoded)
```

```
# Sum of squared distances for each k
inertia = []

# Range of k values to test
k_values = range(1, 16)

# Perform K-Means for each k and store the inertia
for k in k_values:
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(df_scaled)
    inertia.append(kmeans.inertia_)

# Plot the elbow graph
plt.figure(figsize=(10, 5))
plt.plot(k_values, inertia, marker='x')
plt.xlabel('Number of Clusters/K')
plt.ylabel('Inertia (Sum of Squared Distances)')
plt.title('Elbow Method for K')
plt.show()
print('Depending on the random seed, k is usually either 7 or 8.')
```




```
Depending on the random seed, k is usually either 7 or 8.
```

15

```
# Perform Local Outlier Factor/LOF with 10% outlier assumption
lof = LocalOutlierFactor(contamination=0.1)
lof_outliers = lof.fit_predict(df_scaled)
lof_outliers = np.where(lof_outliers == -1)[0]  # LOF labels outliers as -1

# Perform Isolation Forest/ISF with 10% outlier assumption
isf = IsolationForest(contamination=0.1)
isf_outliers = isf.fit_predict(df_scaled)
isf_outliers = np.where(isf_outliers == -1)[0]  # ISF labels outliers as -1

# Find common outliers
common_outliers = np.intersect1d(lof_outliers, isf_outliers)

# Print common outliers
print("Number of common outliers:", len(common_outliers))
print("Common outliers:\n", df.iloc[common_outliers])
```

```
Number of common outliers: 16
Common outliers:
      Survived Pclass     Sex   Age      Fare Embarked  Relatives
16          0      3    male   2.0   29.1250        Q          5
116         0      3    male  70.5    7.7500        Q          0
122         0      2    male  32.5   30.0708        C          1
171         0      3    male   4.0   29.1250        Q          5
188         0      3    male  40.0   15.5000        Q          2
245         0      1    male  44.0   90.0000        Q          2
278         0      3    male   7.0   29.1250        Q          5
280         0      3    male  65.0    7.7500        Q          0
301         1      3    male  28.0   23.2500        Q          2
303         1      2  female  28.0   12.3500        Q          0
322         1      2  female  30.0   12.3500        Q          0
361         0      2    male  29.0   27.7208        C          1
412         1      1  female  33.0   90.0000        Q          1
626         0      2    male  57.0   12.3500        Q          0
787         0      3    male   8.0   29.1250        Q          5
885         0      3  female  39.0   29.1250        Q          5
```