

QUESTION 1

In [51]:

```
## Importing packages
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import metrics
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

A. Load the titanic.csv file into a Pandas data frame. Are there any missing values in the data frame? How many missing values occur for each of the columns?

In [52]:

```
### Loading the dataset
titan = pd.read_csv('/public/bmort/python/titanic.csv')
titan.head()
```

Out[52]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0



In [53]:

```
titan.describe(include='all')
```

Out[53]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000
unique	NaN	NaN	NaN	891	2	NaN	NaN
top	NaN	NaN	NaN	Braund, Mr. Owen Harris	male	NaN	NaN
freq	NaN	NaN	NaN	1	577	NaN	NaN
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000

In [54]:

```
## finding missing values  
titan.isna().sum()
```

Out[54]:

```
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
```

From the output above, we can see that there are 177, 686 and 2 missing values in age,cabin and embarked columns respectively.

In [55]:

```
##we fill in the null spaces with median values
imp_value = titan['Age'].median()
imp_value
```

Out[55]:

28.0

In [56]:

```
# fillna puts a value in for missing values
titan['Age'] =titan['Age'].fillna(imp_value)
```

In [57]:

```
titan.isna().sum()
```

Out[57]:

```
PassengerId      0
Survived          0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin            687
Embarked         2
dtype: int64
```

Thus there no missing values for age

B. What percent of the passengers survived?

In [58]:

```
## Percentage of surviving passengers
sur_pass = titan[titan['Survived']==1].count()[1]
sur_pass
```

Out[58]:

342

In [59]:

```
per_surv = round(sur_pass/len(titan),3)*100
per_surv
print(f'About {per_surv}% of the passengers survived')
```

About 38.4% of the passengers survived

C. What was the maximum fare that was paid to purchase a ticket by a passenger?

In [60]:

```
## Maximum fare paid by a passenger
max_fare = titan['Fare'].max()
print(f'The maximum fare paid by a passenger is {max_fare}')
```

The maximum fare paid by a passenger is 512.3292

D. How many unique places did the passengers embark from?

In [61]:

```
## the unique places the passengers embark from
uni_places = titan['Embarked'].unique()
print(f"The passengers embarked from {len(uni_places)} places namely {(uni_places[0:4])}, with three places known and one unknown place")
```

The passengers embarked from 4 places namely ['S' 'C' 'Q' nan], with three places known and one unknown place

E. Using Scikit-Learn, normalize the values in the Age, SibSp, Parch, and Fare columns so that the range for each column is [0, 1].

In [62]:

```
## Using the MinMaxScaler model to tranform the Age, SibSp, Parch and Fare data
# create a scaler object
scaler = MinMaxScaler()
# fit and transform the data
scaled_data = pd.DataFrame(scaler.fit_transform(titan[['Age', 'SibSp', 'Parch', 'Fare']]),
                           columns=titan[['Age', 'SibSp', 'Parch', 'Fare']].columns)

scaled_data
```

Out[62]:

	Age	SibSp	Parch	Fare
0	0.271174	0.125	0.000000	0.014151
1	0.472229	0.125	0.000000	0.139136
2	0.321438	0.000	0.000000	0.015469
3	0.434531	0.125	0.000000	0.103644
4	0.434531	0.000	0.000000	0.015713
...
886	0.334004	0.000	0.000000	0.025374
887	0.233476	0.000	0.000000	0.058556
888	0.346569	0.125	0.333333	0.045771
889	0.321438	0.000	0.000000	0.058556
890	0.396833	0.000	0.000000	0.015127


891 rows × 4 columns

In [63]:

```
## Replacing the transformed columns into the main data
titan[['Age', 'SibSp', 'Parch', 'Fare']] = scaled_data[['Age', 'SibSp', 'Parch', 'Fare']]
titan.head()
```

Out[63]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	3	Braund, Mr. Owen Harris	male	0.271174	0.125	0.0	A/5 2117
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	0.472229	0.125	0.0	PC 17596
2	3	1	3	Heikkinen, Miss. Laina	female	0.321438	0.000	0.0	STON/O2 3101282
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	0.434531	0.125	0.0	113803
4	5	0	3	Allen, Mr. William Henry	male	0.434531	0.000	0.0	373450



F. Label encode the values in the Pclass, Sex, and Embarked columns.

In [64]:

```
# Label_encoder object knows how to understand word labels.
label_encoder = LabelEncoder()

# Encode labels in column 'Pclass'.
titan['Pclass'] = label_encoder.fit_transform(titan['Pclass'])

titan['Pclass'].unique()
```

Out[64]:

```
array([2, 0, 1])
```

In [65]:

```
# Label_encoder object knows how to understand word labels.  
label_encoder = LabelEncoder()  
  
# Encode labels in column 'Sex'.  
titan['Sex'] = label_encoder.fit_transform(titan['Sex'])  
  
titan['Sex'].unique()
```

Out[65]:

```
array([1, 0])
```

In [66]:

```
# Label_encoder object knows how to understand word labels.  
label_encoder = LabelEncoder()  
  
# Encode labels in column 'Embarked'.  
titan['Embarked'] = label_encoder.fit_transform(titan['Embarked'])  
  
titan['Embarked'].unique()
```

Out[66]:

```
array([2, 0, 1, 3])
```


In [67]:

```
titan.head()
```

Out[67]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0	2	Braund, Mr. Owen Harris	1	0.271174	0.125	0.0	A/5 21171
1	2	1	0	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	0.472229	0.125	0.0	PC 17599
2	3	1	2	Heikkinen, Miss. Laina	0	0.321438	0.000	0.0	STON/O2. 3101282
3	4	1	0	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	0.434531	0.125	0.0	113803
4	5	0	2	Allen, Mr. William Henry	1	0.434531	0.000	0.0	373450

G. Partition the titanic data set so that a random sample of 80% of the data will be used for training and 20% will be used for testing your machine learning model.

In [68]:

```
X = titan[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']].to_numpy()
```

In [69]:

```
y = titan['Survived'].to_numpy()
```

In [70]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, train_size = 0.80)
```

H. Using the Pclass, Sex, Age, SibSp, Parch, Fare, and Embarked features and ScikitLearn, generate a support vector machine (SVM) machine learning model with a linear basis function kernel to predict if a passenger survives.

In [71]:

```
# Creating an SVM classifier with a linear kernel
titan_clf = svm.SVC(kernel='linear')

##training the model
titan_clf.fit(X_train,y_train)
```

Out[71]:

```
SVC(kernel='linear')
```

I. Perform k-fold cross validation (with 5 splits) on the model with the training set. What is the average and standard deviation of the accuracy of the model?

In [72]:

```
## Cross validation procedure
cv = KFold(n_splits=5)

## Assessing the model and determining the scores
score = cross_val_score(titan_clf, X_train,y_train,cv= cv, scoring='accuracy' )
list(score)
```

Out[72]:

```
[0.7482517482517482,
 0.8041958041958042,
 0.8098591549295775,
 0.7816901408450704,
 0.7887323943661971]
```

In [73]:

```
## Computing average and standard deviation
avg_acc = score.mean()
sd_acc = score.std()
print(f'The average and standard deviation of the accuracy of the model are {round(avg_acc,3)} and {round(sd_acc, 3)}')
```

The average and standard deviation of the accuracy of the model are 0.787 and 0.022

J. Use the trained model to predict the survival outcomes of the passengers in the /public/bmort/python/test.csv data set. Provide your answer as a Python list of 0s and 1s.

In [74]:

```
testing_data = pd.read_csv('/public/bmort/python/test.csv')
```

In [75]:

```
testing_data.shape
```

Out[75]:

```
(418, 11)
```

In [76]:

```
testing_data.isna().sum()
```

Out[76]:

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp           0
Parch           0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [77]:

```
##we fill in the null spaces with median values
imp_value1 = testing_data['Age'].median()
imp_value1
```

Out[77]:

```
27.0
```

In [78]:

```
# fillna puts a value in for missing values
testing_data['Age'] =testing_data['Age'].fillna(imp_value1)
```

In [79]:

```
fare_value = testing_data['Fare'].median()
fare_value
```

Out[79]:

```
14.4542
```

In [80]:

```
# fillna puts a value in for missing values
testing_data['Fare'] =testing_data['Fare'].fillna(fare_value)
```

In [81]:

```
testing_data.isna().sum()
```

Out[81]:

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             0
SibSp           0
Parch           0
Ticket          0
Fare            0
Cabin          327
Embarked         0
dtype: int64
```

In [82]:

```
## Normalizing our testing data
## Using the MinMaxScaler model to tranform the Age, SibSp, Parch and Fare data
# create a scaler object
scaler1 = MinMaxScaler()
# fit and transform the data
scaled_data1 = pd.DataFrame(scaler1.fit_transform(testing_data[['Age', 'SibSp', 'Parch', 'Fare']]),
                           columns=testing_data[['Age', 'SibSp', 'Parch', 'Fare']].columns)

scaled_data1
```

Out[82]:

	Age	SibSp	Parch	Fare
0	0.452723	0.000	0.000000	0.015282
1	0.617566	0.125	0.000000	0.013663
2	0.815377	0.000	0.000000	0.018909
3	0.353818	0.000	0.000000	0.016908
4	0.287881	0.125	0.111111	0.023984
...
413	0.353818	0.000	0.000000	0.015713
414	0.512066	0.000	0.000000	0.212559
415	0.505473	0.000	0.000000	0.014151
416	0.353818	0.000	0.000000	0.015713
417	0.353818	0.125	0.111111	0.043640

418 rows × 4 columns

In [83]:

```
## Replacing the transformed columns into the main data
testing_data[['Age', 'SibSp', 'Parch', 'Fare']] = scaled_data1[['Age', 'SibSp', 'Parch', 'Fare']]
testing_data
```

Out[83]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	892	3	Kelly, Mr. James	male	0.452723	0.000	0.000000	330911
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	0.617566	0.125	0.000000	363272
2	894	2	Myles, Mr. Thomas Francis	male	0.815377	0.000	0.000000	240276
3	895	3	Wirz, Mr. Albert	male	0.353818	0.000	0.000000	315154
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	0.287881	0.125	0.111111	3101298
...
413	1305	3	Spector, Mr. Woolf	male	0.353818	0.000	0.000000	A.5. 3236
414	1306	1	Oliva y Ocana, Dona. Fermina	female	0.512066	0.000	0.000000	PC 17758
415	1307	3	Saether, Mr. Simon Sivertsen	male	0.505473	0.000	0.000000	SOTON/O.Q. 3101262
416	1308	3	Ware, Mr. Frederick	male	0.353818	0.000	0.000000	359309
417	1309	3	Peter, Master. Michael J	male	0.353818	0.125	0.111111	2668

418 rows × 11 columns



In [84]:

```
# Label_encoder object knows how to understand word labels.
label_encoder = LabelEncoder()

# Encode labels in column 'Pclass'.
testing_data['Pclass'] = label_encoder.fit_transform(testing_data['Pclass'])

testing_data['Pclass'].unique()
```

Out[84]:

```
array([2, 1, 0])
```

In [85]:

```
# Label_encoder object knows how to understand word labels.
label_encoder = LabelEncoder()

# Encode labels in column 'Sex'.
testing_data['Sex'] = label_encoder.fit_transform(testing_data['Sex'])

testing_data['Sex'].unique()
```

Out[85]:

```
array([1, 0])
```

In [86]:

```
# Label_encoder object knows how to understand word labels.
label_encoder = LabelEncoder()

# Encode labels in column 'Embarked'.
testing_data['Embarked'] = label_encoder.fit_transform(testing_data['Embarked'])

testing_data['Embarked'].unique()
```

Out[86]:

```
array([1, 2, 0])
```

In [87]:

testing_data

Out[87]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	892	2	Kelly, Mr. James	1	0.452723	0.000	0.000000	330911	0.0
1	893	2	Wilkes, Mrs. James (Ellen Needs)	0	0.617566	0.125	0.000000	363272	0.0
2	894	1	Myles, Mr. Thomas Francis	1	0.815377	0.000	0.000000	240276	0.0
3	895	2	Wirz, Mr. Albert	1	0.353818	0.000	0.000000	315154	0.0
4	896	2	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	0	0.287881	0.125	0.111111	3101298	0.0
...
413	1305	2	Spector, Mr. Woolf	1	0.353818	0.000	0.000000	A.5. 3236	0.0
414	1306	0	Oliva y Ocana, Dona. Fermina	0	0.512066	0.000	0.000000	PC 17758	0.2
415	1307	2	Saether, Mr. Simon Sivertsen	1	0.505473	0.000	0.000000	SOTON/O.Q. 3101262	0.0
416	1308	2	Ware, Mr. Frederick	1	0.353818	0.000	0.000000	359309	0.0
417	1309	2	Peter, Master. Michael J	1	0.353818	0.125	0.111111	2668	0.0

418 rows × 11 columns



In [88]:

```
X_new = testing_data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']].to_numpy()
```


In [89]:

```
## Using the model to predict the testing data  
final = list(titan_clf.predict(X_new))  
final
```

Out[89]:

[0,
1,
0,
0,
1,
0,
1,
0,
1,
0,
0,
0,
1,
0,
1,
1,
0,
0,
1,
1,
0,
0,
1,
1,
0,
0,
1,
0,
0,
0,
0,
0,
1,
1,
0,
0,
1,
1,
0,
0,
0,
0,
0,
1,
1,
0,
0,
0,
1,
1,
0,
0,
1,
1,
0,
0,
0,
0,
1,
1,
0,
0,
0,
0,
1,
0,
0,

0,
0,
1,
0,
1,
1,
0,
0,
1,
1,
0,
1,
0,
1,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
1,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
1,
0,
1,
0,
1,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
1,
1,
0,
0,
1,
0,
1,
1,
0,

1,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
1,
0,
1,
1,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
0,
1,
0,
0,
1,
1,
1,
0,
1,
1,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
0,
1,
1,
0,
0,
0,
0,
0,
1,
1,
0,
1,
1,
0,
0,
1,

0,
1,
0,
1,
0,
1,
0,
0,
0,
0,
0,
0,
0,
0,
0,
1,
0,
1,
1,
1,
0,
0,
0,
1,
0,
1,
0,
0,
0,
0,
0,
1,
1,
0,
1,
1,
0,
1,
0,
1,
0,
1,
0,
1,
0,
1,
1,
1,
0,
1,
0,
0,
0,
0,
1,
0,
0,
0,
0,
0,
0,
1,
1,
1,
1,
0,
0,
0,

[illegible]

[illegible]


```
0,
1,
1,
0,
0,
1,
0,
0,
1,
1,
1,
0,
0,
0,
0,
1,
1,
0,
1,
0,
0,
0,
0,
1,
0,
0,
0,
1,
0,
0,
1,
0,
0,
0,
0,
1,
0,
0,
0,
0,
1,
1,
1,
1,
1,
0,
1,
0,
0,
0]
```

In [90]:

```
## Computing the accuracy of the model
acc = titan_clf.score(X,y)
print(f'The accuracy of the model is {round(acc,4)*100}%')
```

The accuracy of the model is 78.68%