Part 1 Create a Decision Tree Classifier

Step 1: Create the dataframe

```
#a Import pandas and the csv file

#Code cell 1
#import pandas
import pandas as pd
#create a pandas dataframe called "training" from the titanic-train.csv file
training = "/content/titanic_train.csv"

tF = pd.read_csv(training)
```

```
#b) Verify the import and take a look at the data.

#Code cell 2
#verify the contents of the training dataframe using the pandas info() method.
#training.

tF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
#pd.set_option('display.max_rows', None)
#display(tF)
```

```
miss_age_values = tF["Age"].isnull().sum()
if miss_age_values == 0:
    print("No missing values in the 'Age' column.")
else:
    print(f"Remaining missing values in the 'Age' column: {miss_age_values}")
```

```
Remaining missing values in the 'Age' column: 177
```

```
#Code cell 3
#view the first few rows of the data

tF.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	E
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	

Next steps: View recommended plots

Most columns have 891 number of values except at "Cabin: 204" and "Embarked: 889". So there's some missing values at Cabin and Embarked.

Step 2: Prepare the Data for the Decision Tree Model.

```
#a) Replace string data with numeric labels
#code cell 4
print(tF["Sex"].unique())

# Replace string data with numeric labels in the "Sex" column
tF["Sex"] = tF["Sex"].apply(lambda to_label: 0 if to_label == 'male' else 1)
```

```
['male' 'female']
```

```
#b) Verify that the Gender variable has been changed.

#code cell 5
#view the first few rows of the data again

tF.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	C123	
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	NaN	

Next steps: View recommended plots

```
#c) Address Missing Values in the Dataset

#code cell 6
tF["Age"].fillna(tF["Age"].mean(), inplace=True)
```

```
#d) Verify that the values have been replaced.

#code cell 7
#verify that the missing values for the age variable have been eliminated.

missing_age_values = tF["Age"].isnull().sum()
if missing_age_values == 0:
    print("No missing values in the 'Age' column.")
else:
    print(f"Remaining missing values in the 'Age' column: {missing_age_values}")
```

```
 No missing values in the 'Age' column.
```

```
#display(tF)
```

Step 3: Train and Score the Decision Tree Model.

```
#a) Create an array object with the variable that will be the target for the model

#code cell 8
#create the array for the target values
y_target = tF["Survived"].values
```

```
#b) Create an array of the values that will be the input for the model.

#code cell 9
columns = ["Fare", "Pclass", "Sex", "Age", "SibSp"]
#create the variable to hold the features that the classifier will use
X_input = tF[list(columns)].values
```

```
#c) Create the learned model.
#code cell 10
#import the tree module from the sklearn library
from sklearn import tree

#create clf_train as a decision tree classifier object
clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)

#train the model using the fit() method of the decision tree object.
#Supply the method with the input variable X_input and the target variable y_target
clf_train = clf_train.fit(X_input, y_target)
```

```
#d) Evaluate the model

#code cell 11
clf_train.score(X_input,y_target)
```

```
0.8226711560044894
```

Step 4 Visualize the Tree

```
# a) Create the intermediate file output

#code cell 12
from six import StringIO
with open("/content/titanic.dot", 'w') as f:
  f = tree.export_graphviz(clf_train, out_file=f, feature_names=columns)
```

```
#b) install Graphviz
!apt-get install graphviz
```

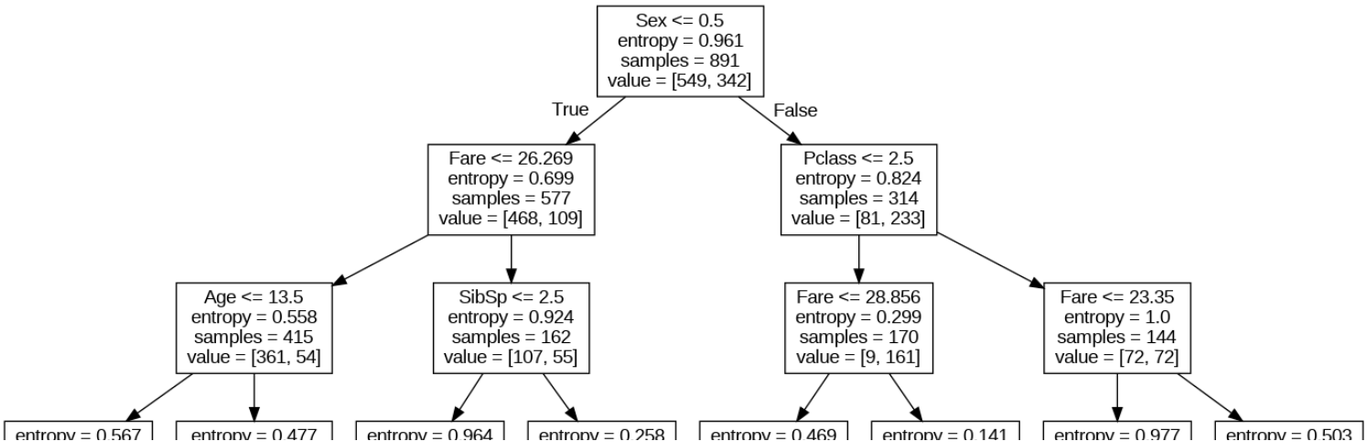
```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
graphviz is already the newest version (2.42.2-6).
0 upgraded, 0 newly installed, 0 to remove and 35 not upgraded.
```

```
#c) Convert the intermediate file to a graphic

#code cell 13
#run the Graphviz dot command to convert the .dot file to .png
!dot -Tpng /content/titanic.dot -o /content/titanic.png
```

```
#d) Display the image

#code cell 14
#import the Image module from the Ipython.display libary
from IPython.display import Image
#display the decison tree graphic
Image("/content/titanic.png")
```



e) Interpret the Tree

What describes the group that had the most deaths by number? Which group had the most survivors?

Part 2: : Apply the Decision Tree Model

Step 1: Import and Prepare the Data

```
#a) Import the data.

#code cell 15
#import the file into the 'testing' dataframe.

test = "/content/titanic_test.csv"

tsf = pd.read_csv(test)
```

```
# Code Cell 16
# Display the number of records in the "testing" dataset
num_records = tsf.shape[0]
print("Number of records in the 'testing' dataset:", num_records)
```

```
Number of records in the 'testing' dataset: 418
```

```
tsf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

There's missing values at "Age", Fare, "Cabin". "Age" has 86 missing values. "Fare" has 1 missing values. "Cabin" has 327 missing values.

```
tsf.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Next steps: View recommended plots

```
#b) Use a lambda expression to replace the "male" and "female" values
#   with 0 for male and 1 for female.

#code cell 16
#replace the Gender labels in the testing dataframe
# Hint: look at code cell 4

print(tsf["Sex"].unique())

# Replace string data with numeric labels in the "Sex" column
tsf["Sex"] = tsf["Sex"].apply(lambda to_label: 0 if to_label == 'male' else 1)
```

```
[0 1]
```

```
tsf.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	0	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	1	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	0	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	0	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	1	22.0	1	1	3101298	12.2875	NaN	S

Next steps: View recommended plots

```
#code cell 17
#Use the fillna method of the testing dataframe column "Age"
#to replace missing values with the mean of the age values.
tsf["Age"].fillna(tsf["Age"].mean(), inplace=True)

missing_age_values1 = tsf["Age"].isnull().sum()
if missing_age_values1 == 0:
    print("No missing values in the 'Age' column.")
else:
    print(f"Remaining missing values in the 'Age' column: {missing_age_values1}")
```

```
No missing values in the 'Age' column.
```

```
#code cell 18
#verify the data preparation steps. Enter and run both the info and head
#methods from here, by entering and running one and then the other.

tsf.info()
tsf.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    int64  
 4   Age          418 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
 10  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 36.0+ KB
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	1	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	1	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	1	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	1	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	1	22.0	1	1	3101298	12.2875	NaN	S

Next steps: View recommended plots

Step 2: Label the testing dataset

```
#a) Create the array of input variables from the testing data set.

#code cell 19
#create the variable X_input to hold the features that the classifier will use

columns1 = ["Parch", "Pclass", "Sex", "Age", "SibSp"]
X_input1 = tsf[list(columns1)].values
```

```
#b) Apply the model to the testing data set.

# Code Cell 20 (modified)
# Apply the model to the testing data and store the result in a pandas dataframe.
# Use X_input as the argument for the predict() method of the clf_train classifier object
target_labels = clf_train.predict(X_input1)

# Convert the target array into a pandas dataframe using the pd.DataFrame() method and target as an ar
target_labels = pd.DataFrame({'Est_Survival': target_labels, 'Name': tsf['Name']})

# Display the first few rows of the data set
print(target_labels.head())

# Additional line to display the 'target_labels' DataFrame
target_labels.head()
```

```
   Est_Survival                                     Name
0             1                          Kelly, Mr. James
1             1          Wilkes, Mrs. James (Ellen Needs)
2             1                 Myles, Mr. Thomas Francis
3             1                          Wirz, Mr. Albert
4             1  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
```

	Est_Survival	Name
0	1	Kelly, Mr. James
1	1	Wilkes, Mrs. James (Ellen Needs)
2	1	Myles, Mr. Thomas Francis
3	1	Wirz, Mr. Albert
4	1	Hirvonen, Mrs. Alexander (Helga E Lindqvist)

Next steps: View recommended plots

```
#c) Evaluate the accuracy of the estimated labels

#code cell 21
#import the numpy library as np
import numpy as np
# Load data for all passengers in the variable all_data
all_data = pd.read_csv("/content/titanic_all.csv")

# Merging using the field Name as key, selects only the rows of the
# two datasets that refer to the same passenger
testing_results = pd.merge(target_labels, all_data[['Name','Survived']], on=['Name'])


# Compute the accuracy as a ratio of matching observations to total osbervations. Store this in in the
acc = np.sum(testing_results['Est_Survival'] == testing_results['Survived']) / float(len(testing_resul

# Print the results
print(f"Accuracy: {acc * 100:.2f}%")
```

```
Accuracy: 37.03%
```

Part 3: Evaluate the Decision Tree Model

Step 1: Import the data

```
#code cell 22
#import the titanic_all.csv file into a dataframe called all_data. Specify the list of columns to impo

all_data = pd.read_csv("/content/titanic_all.csv", usecols=['Survived','Pclass',
'Gender','Age','SibSp','Fare'])

#View info for the new dataframe

all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1308 entries, 0 to 1307
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  1308 non-null   int64
 1   Pclass    1308 non-null   int64
 2   Gender    1308 non-null   object
 3   Age       1045 non-null   float64
 4   SibSp     1308 non-null   int64
 5   Fare      1308 non-null   float64
dtypes: float64(2), int64(3), object(1)
memory usage: 61.4+ KB
```

There are 1308 records in the dataset. And only at "Age" that has missing values.

```
all_data.head()
```

	Survived	Pclass	Gender	Age	SibSp	Fare
0	1	1	female	29.0000	0	211.3375
1	1	1	male	0.9167	1	151.5500
2	0	1	female	2.0000	1	151.5500
3	0	1	male	30.0000	1	151.5500
4	0	1	female	25.0000	1	151.5500

Next steps: View recommended plots

```
#code cell 23
#Label the gender variable with 0 and 1
print(all_data["Gender"].unique())

# Replace string data with numeric labels in the "Sex" column
all_data["Gender"] = all_data["Gender"].apply(lambda to_label: 0 if to_label == 'male' else 1)

all_data.head()
```

```
[1]
```

	Survived	Pclass	Gender	Age	SibSp	Fare
0	1	1	1	29.0000	0	211.3375
1	1	1	1	0.9167	1	151.5500
2	0	1	1	2.0000	1	151.5500
3	0	1	1	30.0000	1	151.5500
4	0	1	1	25.0000	1	151.5500

Next steps: View recommended plots

```
#c) Replace the missing age values with the mean of the age of all members of the data set
#code cell 24
#replace missing Age values with the mean age
all_data["Age"].fillna(all_data["Age"].mean(), inplace=True)

missing_age_values2 = all_data["Age"].isnull().sum()
if missing_age_values2 == 0:
    print("No missing values in the 'Age' column.")
else:
    print(f"Remaining missing values in the 'Age' column: {missing_age_values2}")

#display the first few rows of the data set
all_data.info()
all_data.head()
```

```
No missing values in the 'Age' column.
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1308 entries, 0 to 1307
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Survived  1308 non-null   int64  
 1   Pclass    1308 non-null   int64  
 2   Gender    1308 non-null   int64  
 3   Age       1308 non-null   float64
 4   SibSp     1308 non-null   int64  
 5   Fare      1308 non-null   float64
dtypes: float64(2), int64(4)
memory usage: 61.4 KB
```

	Survived	Pclass	Gender	Age	SibSp	Fare
0	1	1	1	29.0000	0	211.3375
1	1	1	1	0.9167	1	151.5500
2	0	1	1	2.0000	1	151.5500
3	0	1	1	30.0000	1	151.5500
4	0	1	1	25.0000	1	151.5500

Next steps: View recommended plots

Step 2: Create the input and output variables for the training and testing data.

```
#a) Designate the input variables and output variables and generate the array
#code cell 25
#Import train_test_split() from the sklearn.model_selection libary
from sklearn.model_selection import train_test_split
#create the input and target variables as uppercase X and lowercase y. Reuse the (

columns2 = ['Survived','Pclass','Gender','Age','SibSp','Fare']

X = all_data[list(columns2)].values
y = all_data["Survived"].values

#generate the four testing and training data arrays with the train_test_split() me
X_train,X_test,y_train,y_test=train_test_split(X, y, test_size=0.40, random_state
```

```
#b) Train the model and fit it to the testing data

#code cell 26
#create the training decision tree object
clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)
```