Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies in CpE 2
2nd Semester	AY 2023-2024

ACTIVITY NO. 4	Decision Tree Classification
Name	Cuevas, Christian Jay L.
Section	CPE32S3
Date Performed:	
Date Submitted:	
Instructor:	Engr. Roman M. Richard

PART 1 - Simple Linear Regression

Here in this part, we want to apply simple linear regression to this dataset.

```
1 import numpy as np
2 import matplotlib.pyplot as plt  # To visualize
3 import pandas as pd  # To read data
4 from sklearn.linear_model import LinearRegression
```

```
1 data = pd.read_csv('/content/drive/MyDrive/Hands-On Activity 4 EMTECH 2/titanic_train.csv')  # loa
```

```
1 data.head()
```

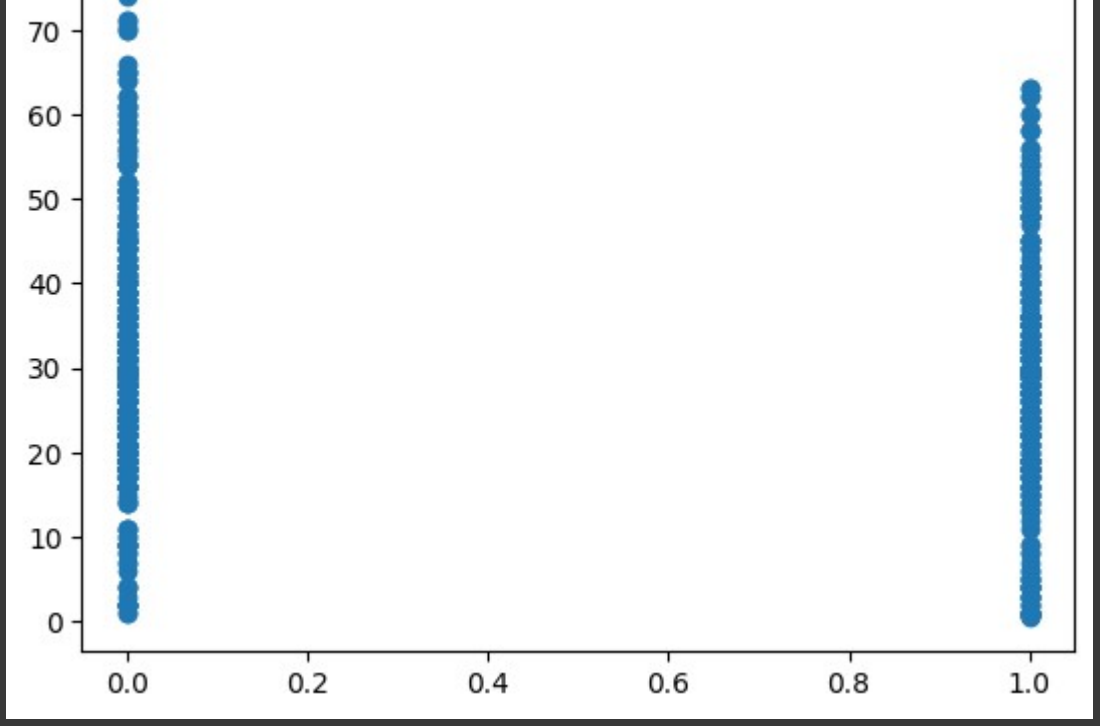
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Eml
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs	female	38.0	1	0	PC 17599	71.2833	C85	

Next steps: View recommended plots

```
1 data["Age"].fillna(data["Age"].mean(), inplace=True)
```

```
1 plt.scatter(data["Survived"], data["Age"])
```

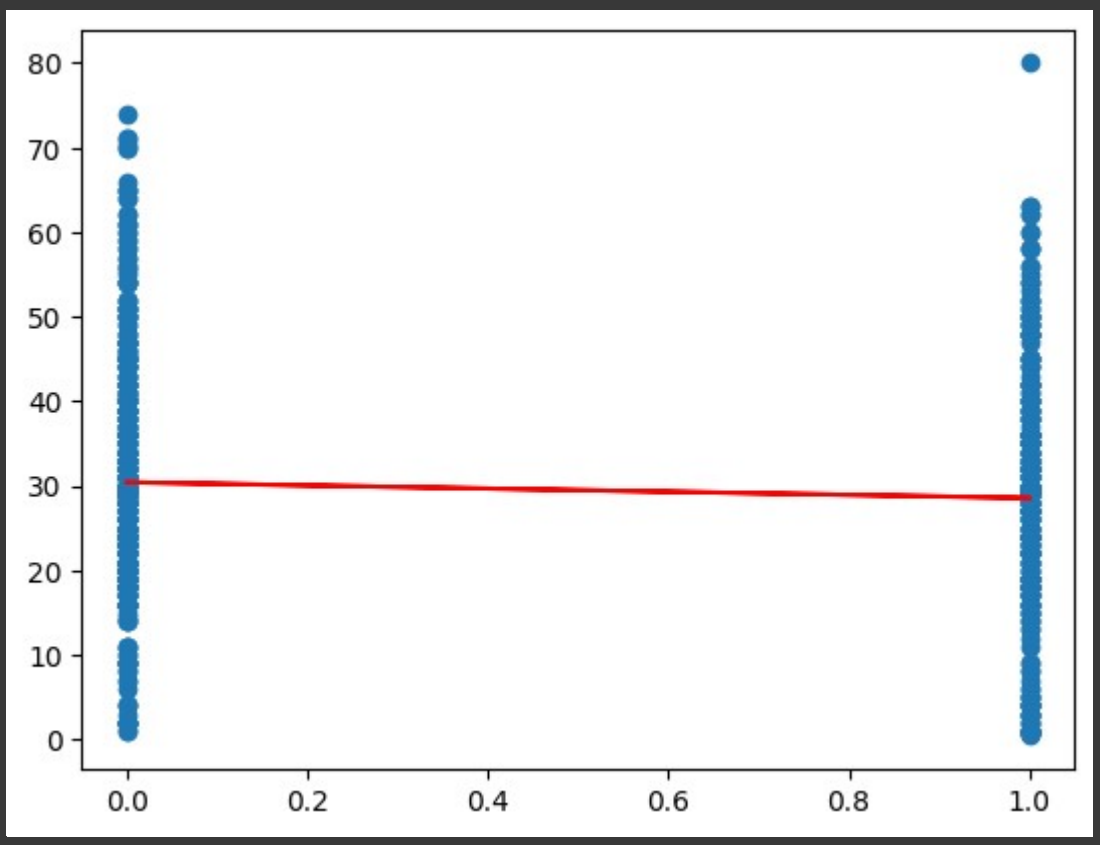
```
<matplotlib.collections.PathCollection at 0x7b06faec6b00>
```




```
model = LinearRegression()
```

```
X = data["Survived"].values.reshape(-1, 1)  # values converts it into a numpy array
Y = data["Age"].values.reshape(-1, 1)
model.fit(X, Y)
Y_pred = model.predict(X)
```

```
plt.scatter(X, Y)
plt.plot(X, Y_pred, color='red')
plt.show()
```



OBSERVATION

OBSERVATION:

I performed a simple linear regression and observed that it can't be used in the data of "Survived" column since the survived column values are only 0 and 1. This means that a simple linear regression can't predict or create any assumptions in this data. This proved that there are data where we can't apply simple linear regression

PART 2 - Lab - Decision Tree Classification

Scenario / Background

In this lab you will create a decision tree classifier that will work with a data set which contains the details about the more than 1300 hundred passengers who were onboard the passenger liner Titanic on its infamous maiden voyage.

Part 1: Create a Decision Tree Classifier

In this part of the lab, you will create a decision tree classifier that will learn from a labelled dataset.

The dataset contains the names and demographic details for each passenger. In addition, details of the passengers' trip are included. From this data, we can be build a decision tree that illustrates the factors that contributed to survivability, or lack of it, for the voyage. The datasets contain the following variables:

Variable	Description
1. Passenger ID	Unique identifier for each passenger
2. Survival	Did the passenger survive? (0 = No; 1 = Yes)
3. Pclass	Passenger ticket class. (1 = 1st; 2 = 2nd; 3 = 3rd)
4. Name	Name of the passenger. (last name, first name)
5. Gender	Male or Female
6. Age	Age in years. Mostly integers with float values for children under one year.
7. SibSp	Number of siblings or spouse onboard.
8. Parch	Number of parents or child onboard.
9. Ticket	Ticket number
10. Fare	Amount paid for fare in pre-1970 British Pounds
11. Cabin	Cabin number
12. Embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

With the data above, what kinds of questions can we ask about the factors that contributed to passengers surviving or perishing in the Titanic disaster?

ANSWER:

The questions we can ask relating on how they survive the Titanic disaster while taking into account the factors are the following:

- What is the range of age of the people who survived?
- What is the distribution of the passengers ticket class and cabin, and their survival rate?
- Is the survival rate of passengers with Siblings, Spouse, Parents or Children onboard higher than those travelling alone?
- Is there any relationship between the survival rate and the cabin number?
- Are there any difference between the survival rate of people that embarked on different ports?
- Is there any relationship between the ticket fare and the survival rate?

Step 1: Create the dataframe

a) Import pandas and the csv file

First, import pandas and create a dataframe from the Titanic training data set, which is stored in the titanic-train.csv file. Use the pd.read_csv() method.

```
#Code cell 1
#import pandas
import pandas as pd
#create a pandas dataframe called "training" from the titanic-train.csv file
training = pd.read_csv("/content/drive/MyDrive/Hands-On Activity 4 EMTECH 2/titanic_train.csv")
```

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

training.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
```

Are there missing values in the data set?

ANSWERS:

There are missing values in the dataset, specifically in the column of "Age", "Cabin", and "Embarked". The total count of the data is 891. The count for Age is 714, Cabin is 204, and Embarked is 889. There are 177 missing data in Age, 687 for Cabin and 2 in Embarked.

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

a) Replace string data with numeric labels

We will use scikit-learn to create the decision trees. The decision tree model we will be using can only handle numeric data. The values for the Gender variable must be transformed into numeric representations. 0 will be used to represent "male" and 1 will represent "female."

In this code, a lambda expression is used with the apply() dataframe method. This lambda expression represents a function that uses a conditional statement to replace the text values in the columns with the appropriate numeric value. The lambda statement can be interpreted as "if the parameter toLabel equals 'male', return 0, if the value is something else, return 1." The apply() method will execute this function on the values in every row of the "Gender" column of the dataframe.

```
1 training.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Eml
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	

Next steps: View recommended plots

OBSERVATION:

The value for the "Sex" column is either male or female. We want to change that into 0 or 1 respectively so that we can process the dataset much easier.

```
1 #code cell 4
2 training["Sex"] = training["Sex"].apply(lambda toLabel: 0 if toLabel == 'male' else 1)
```

b) Verify that the Gender variable has been changed.

The output should show values of 0 or 1 for the Gender variable in the dataset.

```
1 training.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embar
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	

Next steps: View recommended plots

OBSERVATION:

The value for male and female has been changed in to 0 and 1.

c) Address Missing Values in the Dataset

The output of the info() method above indicated that about 180 observations are missing the age value. The age value is important to our analysis. We must address these missing values in some way. While not ideal, we can replace these missing age values with the mean of the ages for the entire dataset.

This is done by using the fillna() method on the "Age" column in the dataset. The fillna() method will change the original dataframe by using the inplace = True argument.

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

```
1 training.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    int64  
 5   Age          891 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(6), object(4)
memory usage: 83.7+ KB
```

OBSERVATION:

The value earlier of the "Age" column is 714, now it has been modified to 891. This happened because we filled in the missing values with the mean of the Age column. The inplace = True means that it directly modified the dataframe without creating a new one.

---

d) Verify that the values have been replaced.

```
1 #code cell 7
2 #verify that the missing values for the age variable have been eliminated
3 training["Age"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 891 entries, 0 to 890
Series name: Age
Non-Null Count  Dtype
--------------  -----
891 non-null    float64
dtypes: float64(1)
memory usage: 7.1 KB
```

OBSERVATION:

Here we can see that from the original 714 entries in the "Age" column, it became 891 which means that all of the missing entries have been filled up.

What is the value that was used to replace the missing ages?

```
1 training["Age"].mean()
```

```
29.69911764705882
```

```
1 training.head(6)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.000000	1	0	PC 17599	71.2833	C85
				Heikkinen,					STON/		

Next steps: View recommended plots

ANSWER:

The value that was used to replace this value is the mean of the column "Age" which is 29.699118. All of the NA value in the column was filled with that specific value which you can see in the index 5 of the table above. The age of Moran, Mr. James is 29.699118

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.

The purpose of the model is to classify passengers as survivors or victims. The dataset identifies survivors and victims. The model will learn which input variable values are most likely to belong to victims and survivors, and then use that information to classify passengers from a unique test data set.

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

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

Only some of the features of the data are useful for creating the classifier tree. We create a list of the columns from the data that we want the classifier to use as the input variables and then create an array using the column name from that variable. The variable X_input holds the values for all the features that the model will use to learn how to make the classifications. After the model is trained, we will use this variable to assign these labels to the test data set.

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

c) Create the learned model.

Import the decision tree module from the sklearn machine learning library. Create the classifier object clf_train. Then, use the fit() method of the classifier object, with the X_input and y_target variables as parameters, to train the 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

Use the score() method of the decision tree object to display the percentage accuracy of the assignments made by the classifier. It takes the input and target variables as arguments.

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

```
0.8226711560044894
```

This score value indicates that classifications made by the model should be correct approximately 82% of the time.

Step 6: Visualize the Tree

a) Create the intermediate file output

Import the sklearn.externals.six StringIO module which is used to output the characteristics of the decision tree to a file. We will create a Graphviz dot file which will allow us to export the results of the classifier into a format that can be converted into a graphic.

```
!pip install six
```

```
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (1.16.0)
```

```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (0.20.1)
```

```
#code cell 12
from six import StringIO

with open("titanic.dot", 'w') as f:
  f = tree.export_graphviz(clf_train, out_file=f, feature_names=columns)
```

b) Install Graphviz

To visualize the decision tree, Graphviz needs to be installed from a terminal. The installation requires that a prompt be answered, which can't be done from a notebook code cell. Use the apt-get install graphviz

command from the terminal command line to install this software.

c) Convert the intermediate file to a graphic

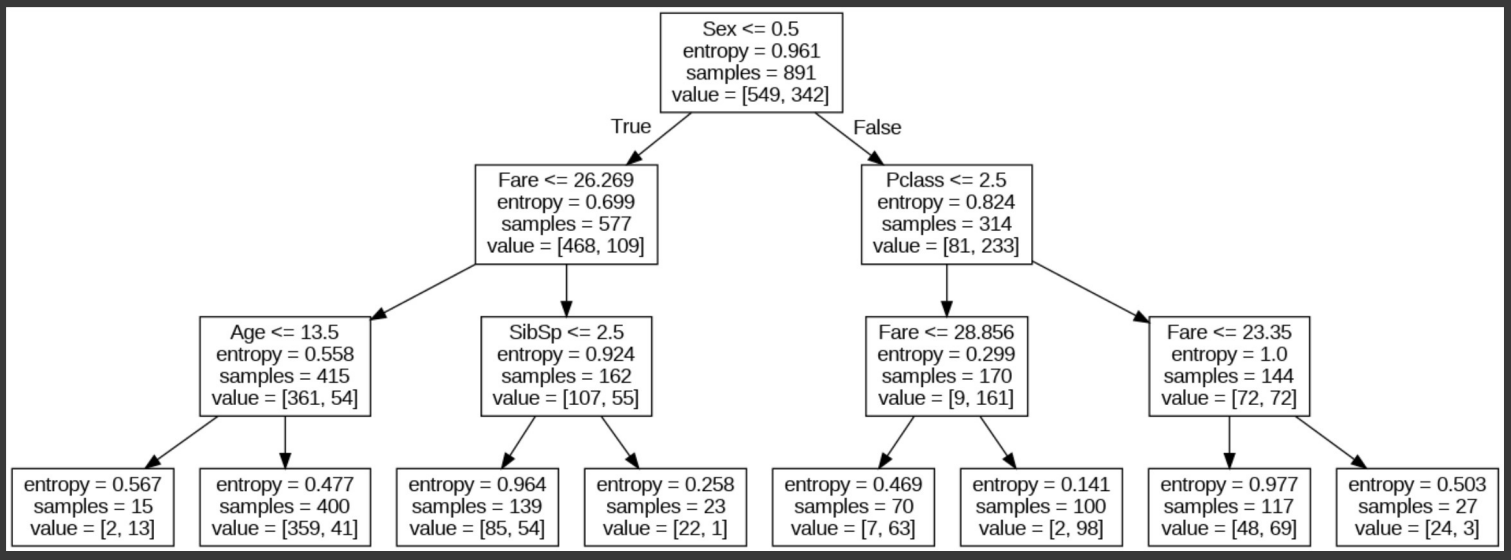
The dot file that was created above can be converted to a .png file with the graphiz dot renderer. This is a shell command, so use ! before it to run it from this noteblook. The new titanic.png graphic file should appear in the directory that contains this notebook.

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

d) Display the image

Now we will import the Image module from the IPython.display library. This will allow us to open and display an external graphics file on the notebook page. The Image function is used to display the file, with the .png file name as argument.

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



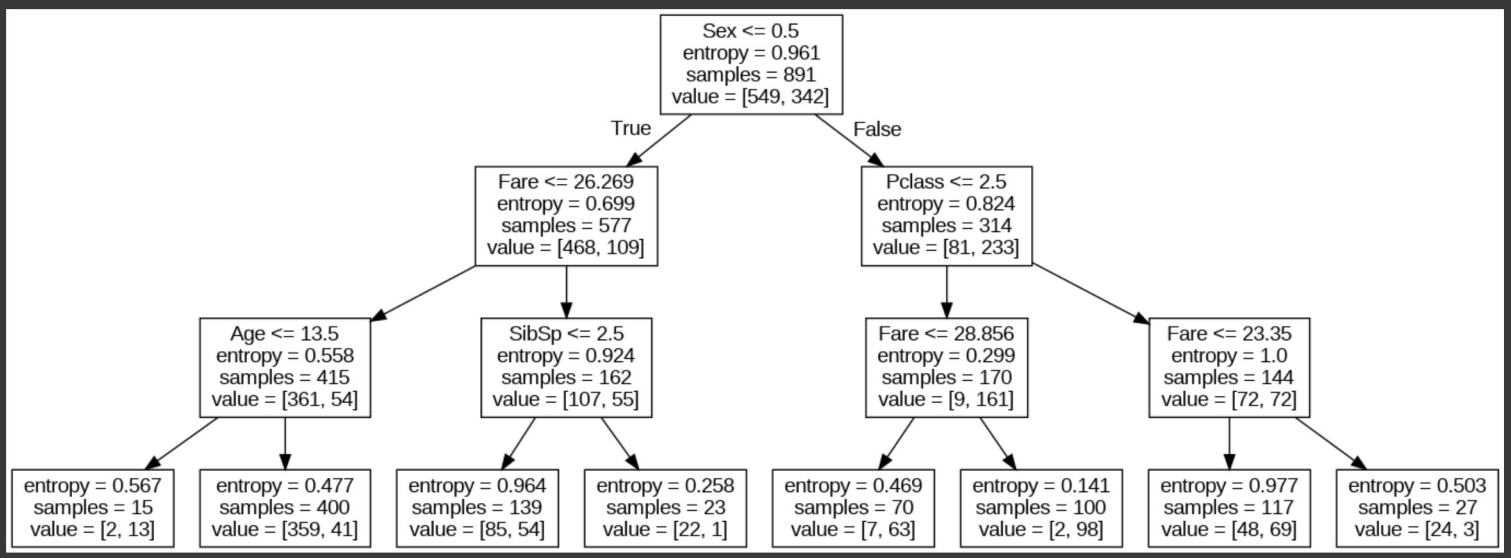
e) Interpret the tree

From the tree, we can see several things. First, at the root of the tree is the Gender variable, indicating that it is the single most important factor in making the classification. The branches to the left are for Gender = 0 or male. Each root and intermediate node contains the decision factor, the entropy, and the number of passengers who fit the critierion at that point in the tree. For example, the root node indicates that there are 891 observations that make up the learning data set. At the next level, we can see that 577 people were male, and 314 were female. In the third level, at the far right, we can see that 415 people were male and paid a fare of less than 26.2686. Finally, the leaf nodes for that intermediate node indicate that 15 of these passengers were below the age of 13.5, and the other 400 were older than that age.

Finally, the elements in the value array indicate survival. The first value is the number of people who died, and the second is the number of survivors for each criterion. The root node tells us that out of our sample, 549 people died and 342 survived.

Entropy is a measure of noise in the decision. Noise can be viewed as uncertainty. For example, in nodes in which the decision results in equal values in the survival value array, the entropy is at its highest possible value, which is 1.0. This means that the model was unable to definitively make the classification decision based on the input variables. For values of very low entropy, the decision was much more clear cut, and the difference in the number of survivors and victims is much higher.

```
1 Image("titanic.png")
```



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

ANSWER:

The group that has the most deaths is the group of male with age higher than 13.5. The total deaths of that group is 359 and the survivors are only 41. The group which has the most survivors is the group of female which paid a fare higher than 28.856.

Part 2: Apply the Decision Tree Model

In this part of the lab, we will use the results of the learned decision tree model to label an unlabelled dataset of Titanic passengers. The decision tree will evaluate the features of each observation and label the observation as survived (label = 1) or died (label = 0).

Step 1: Import and Prepare the Data

a) Import the data.

Name the dataframe "testing" and import the file titanic-test.csv.

```
#code cell 15
#import the file into the 'testing' dataframe.
testing = pd.read_csv("/content/drive/MyDrive/Hands-On Activity 4 EMTECH 2/titanic_test.csv")
```

```
testing.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
```

How many records are in the data set?

ANSWER:

There are a total of 418 entries in the dataset.

Which important variables(s) are missing values and how many are missing?

ANSWER:

There are 86 entries missing in the column of "Age", 1 entry missing in the column of "Fare" and 327 entries missing in the column of "Cabin".

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

```
#code cell 16
testing["Sex"] = testing["Sex"].apply(lambda toLabel: 0 if toLabel ==
'male' else 1)
```

c) Replace the missing age values with the mean of the ages.

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

OBSERVATION:

I observed here that there is also a missing float value in the "Fare" column so I also used fillna method to fill the missing value with the mean of the "Fare" column. This will ensure that all the floating value that we have are all with values and it will avoid any error later in the code.

d) Verify that the values have been replaced.

Check that the missing values have been filled and that the Gender labels are 0 and 1.

```
#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.
testing.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
			Wilkes,								

1	893	3	Mrs. James (Ellen Needs)	1	47.0	1	0	363272	7.0000	NaN	S
			Myles, Mr.								

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
columns = ["Fare", "Pclass", "Sex", "Age", "SibSp"]
X_input = testing[list(columns)].values
```

b) Apply the model to the testing data set.

Use the predict() method of the clf_train object that was trained to label the observations in the testing data set with the most likely survival classification. Provide the array of input variables from the testing data set as the parameter for this method.

```
#code cell 20
#apply the model to the testing data and store the result in a pandas dataframe.
#Use X_input as the argurment for the predict() method of the clf_train classifier object

target_labels = clf_train.predict(X_input)

#convert the target array into a pandas dataframe using the pd.DataFrame() method and target as ar

target_labels = pd.DataFrame({'Est_Survival':target_labels, 'Name':testing['Name']})

#display the first few rows of the data set
```

c) Evaluate the accuracy of the estimated labels

The ground truth for the survival of each passenger can be found in another file called all_data.csv. To select only the passengers contained in the testing dataset, we merge the target_labels dataframe and the all_data dataframe on the field Name. We then compare the estimated label with the ground truth dataframe and compute the accuracy of the learned model.

```
#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/drive/MyDrive/Hands-On Activity 4 EMTECH 2/titanic_all.csv")
```

```
7
8 # Merging using the field Name as key, selects only the rows of the two datasets that refer to the
9 testing_results = pd.merge(target_labels, all_data[['Name','Survived']], on=['Name'])
10
11 # Compute the accuracy as a ratio of matching observations to total osbervations. Store this in in
12 acc = np.sum(testing_results['Est_Survival'] == testing_results['Survived']) / float(len(testing_r
13 # Print the result
14
```

```
1 print(acc)
```

```
0.7682619647355163
```

Part 3: Evaluate the Decision Tree Model

The sklearn library includes a module that can be used to evaluate the accurracy of the decision tree model. The train_test_split() method will divide the observations in whole data set into two randomly selected arrays of observations that makeup the testing and training datasets. After fitting the model to the training data, the trained model can be scored and the prediction accurracy compared for both the training and test datasets. It is desirable for the two scores to be close, but the accuracy for the test dataset is normally lower that for the training data set.

Step 1: Import the data

This time we will import the data from a csv file, but we will specify the columns that we want to have appear in the dataframe. We will do this by passing an array-like list of column names to the read_csv() method usecols parameter. Use the following columns: 'Survived', 'Fare', 'Pclass', 'Gender', 'Age', and 'SibSP'. Each should be in quotes and the list should be square brackets. Name this dataframe all_data.

```
1 #code cell 22
2 #import the titanic_all.csv file into a dataframe called all_data. Specify the list of columns to
3 all_data = pd.read_csv("/content/drive/MyDrive/Hands-On Activity 4 EMTECH 2/titanic_all.csv", usec
4 'Gender','Age','SibSp','Fare'])
5 #View info for the new dataframe
```

```
1 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

```
1 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
```

How many records are in the data set?

ANSWER:

There are 1,308 records in this dataset which is the combined number for the titanic_train and titanic_test.

Which important variables(s) are missing values and how many are missing?

ANSWER:

There are 263 values missing in the Age variable/column.

Step 2: Prepare the data.

a) Remove the "male" and "female" strings and replace them with 0 and 1 respectively.

```
1 #code cell 23
2 #Label the gender variable with 0 and 1
3 all_data["Gender"] = all_data["Gender"].apply(lambda toLabel: 0 if toLabel == 'male' else 1)
```

```
1 all_data.head()
```

	Survived	Pclass	Gender	Age	SibSp	Fare
0	1	1	1	29.0000	0	211.3375
1	1	1	0	0.9167	1	151.5500
2	0	1	1	2.0000	1	151.5500
3	0	1	0	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)
#display the first few rows of the data see
```

```
all_data["Age"].mean()
```

```
29.851834162679427
```

```
all_data.head(20)
```

	Survived	Pclass	Gender	Age	SibSp	Fare
0	1	1	1	29.000000	0	211.3375
1	1	1	0	0.916700	1	151.5500
2	0	1	1	2.000000	1	151.5500
3	0	1	0	30.000000	1	151.5500
4	0	1	1	25.000000	1	151.5500
5	1	1	0	48.000000	0	26.5500
6	1	1	1	63.000000	1	77.9583
7	0	1	0	39.000000	0	0.0000
8	1	1	1	53.000000	2	51.4792
9	0	1	0	71.000000	0	49.5042
10	0	1	0	47.000000	1	227.5250
11	1	1	1	18.000000	1	227.5250
12	1	1	1	24.000000	0	69.3000
13	1	1	1	26.000000	0	78.8500
14	1	1	0	80.000000	0	30.0000
15	0	1	0	29.851834	0	25.9250
16	0	1	0	24.000000	0	247.5208
17	1	1	1	50.000000	0	247.5208
18	1	1	1	32.000000	0	76.2917
19	0	1	0	36.000000	0	75.2417

Next steps: View recommended plots

OBSERVATION:

You can see here that the age for the peron in index 15 is 29.851834

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

The sklearn library includes modules that help with model selection. We will import from sklearn.model_selection the train_test_split() method. This method will automatically split the entire dataset, returning in total four numpy arrays, two for the features (test and validation) and two for the labels (test and validation). One parameter of the method specifies the proportion of observations to use for testing and training. Another parameter specifies a seed value that will be used to randomize assignment of the observation to testing or training. This is used so that another user can replicate your work by receiving the same assignments of observations to datasets. The syntax of the method is: `train_test_split(input_X, target_y, test_size=0.4, random_state=0)`

40% of the data will be used for testing. The random seed is set to 0.

The method returns four values. These values are the input varibles for training and testing data and the target variables for the training and testing data in that order

a) Designate the input variables and output variables and generate the arrays.

```
#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 columns variable.
columns1 = ["Fare", "Pclass", "Gender", "Age", "SibSp"]
X = all_data[list(columns1)].values
y = all_data["Survived"].values
#generate the four testing and training data arrays with the train_test_split() method
X_train,X_test,y_train,y_test=train_test_split(X, y, test_size=0.40, random_state=0)
```

b) Train the model and fit it to the testing data.

Now the model can be fit again. The model will be trained using only the training datat, as selected by the train_test_split function

```
#code cell 26
#create the training decision tree object
clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)
#fit the training model using the input and target variables
clf_train = clf_train.fit(X_train, y_train)
```

c) Compare models by scoring each.

Use the score() method of each decision tree object to generate scores.

```
#code cell 27
#score the model on the two datasets and store the scores in variables. Convert the scores to stri
train_score = str(clf_train.score(X_train,y_train))
test_score = str(clf_train.score(X_test,y_test))
#output the values in a test string
print('Training score = '+ train_score+' Testing score = '+test_score)
```

```
Training score = 0.8201530612244898 Testing score = 0.8053435114503816
```

OBSERVATION:

I observed that the Training score and Testing score values are close to each other. The value of Training score is higher than the testing score, this means that this model can accurately identify the survival rate of a person based on a given data.

We have now compared the scores for the trained model on both test and validation data. As expected, the test accuracy score is close, but lower than the score for the training data. This is because normally, the model tends to overfit the training data, therefore the test score is a better evaluation of how the model is able to generalize outside of the training data

Part 4 For Further Study (optional)

1. Remove observations with missing Age values.

```
#import the data again to a new variable
import numpy as np
all_data1 = pd.read_csv("/content/drive/MyDrive/Hands-On Activity 4 EMTECH 2/titanic_all.csv")
```

```
all_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1308 entries, 0 to 1307
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Passenger  1308 non-null   int64  
 1   Survived   1308 non-null   int64  
 2   Pclass     1308 non-null   int64  
 3   Name       1308 non-null   object 
 4   Gender     1308 non-null   object 
 5   Age        1045 non-null   float64
 6   SibSp      1308 non-null   int64  
 7   Parch      1308 non-null   int64  
 8   Ticket     1308 non-null   object 
 9   Fare       1308 non-null   float64
 10  Cabin      295 non-null    object 
 11  Embarked   1306 non-null   object 
dtypes: float64(2), int64(5), object(5)
memory usage: 122.8+ KB
```

```
#Drop Nan values
all_data1.dropna(subset = ["Age"], inplace = True)
```

```
#All the data with NaN values for Age has been dropped that's why the count became 1045
all_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1045 entries, 0 to 1307
Data columns (total 12 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Passenger  1045 non-null   int64  
 1   Survived   1045 non-null   int64  
 2   Pclass     1045 non-null   int64  
 3   Name       1045 non-null   object 
 4   Gender     1045 non-null   object 
 5   Age        1045 non-null   float64
 6   SibSp      1045 non-null   int64  
 7   Parch      1045 non-null   int64  
 8   Ticket     1045 non-null   object 
 9   Fare       1045 non-null   float64
 10  Cabin      272 non-null    object 
 11  Embarked   1043 non-null   object 
dtypes: float64(2), int64(5), object(5)
memory usage: 106.1+ KB
```

CONCLUSION/OBSERVATION:

This is a very good introduction in using machine learning to analyze and find relationship in a data. It shows the power of machine learning and how it can be used in other application with large datasets. I learned a lot of new methods or functions that can be used with pandas to visualize an image, and I also learned the decision tree model. I also concluded that linear regression analysis is not applicable to all data and there are data where it is hard to infer any useful information when using linear regression analysis.
