

Buenafe, Lorenz Angelo N.  
1915058  
CPE 019 - CPE32S9

▼ PART 1

Part 1: Import the Libraries and Data

```
from google.colab import drive
drive.mount("/content/drive")

Mounted at /content/drive

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

path = "/content/drive/MyDrive/CPE 019 Emerging Technologies 3/HOA 4.1/titanic_train.csv"
df = pd.read_csv(path)

df["Sex"] = df["Sex"].apply(lambda toLabel: 0 if toLabel == 'male' else 1)
df["Age"].fillna(df["Age"].mean(), inplace=True)

df.info()

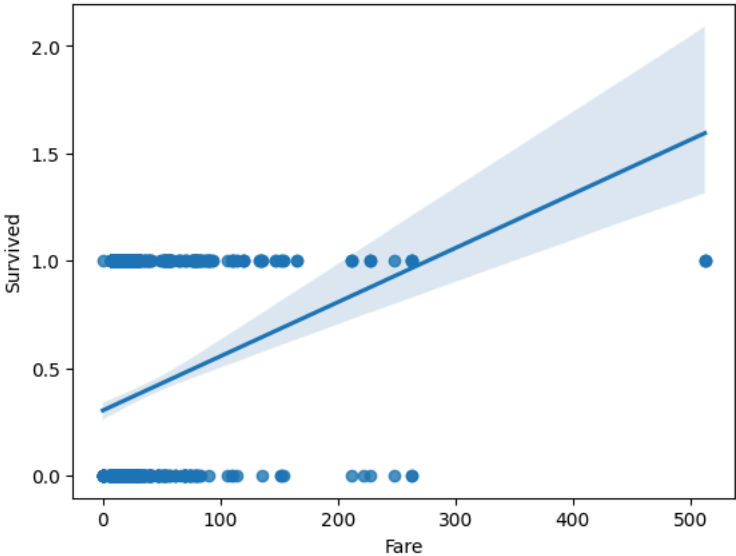
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  -
0   PassengerId  891 non-null    int64
1   Survived     891 non-null    int64
2   Pclass       891 non-null    int64
3   Name         891 non-null    object
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

df.describe()

      PassengerId  Survived  Pclass    Sex    Age    SibSp  Parch
count  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000  891.000000
mean    446.000000    0.383838    2.308642    0.352413    29.699118    0.523008    0.381594
std     257.353842    0.486592    0.836071    0.477990    13.002015    1.102743    0.806057
min       1.000000    0.000000    1.000000    0.000000     0.420000    0.000000    0.000000
25%    223.500000    0.000000    2.000000    0.000000    22.000000    0.000000    0.000000
50%    446.000000    0.000000    3.000000    0.000000    29.699118    0.000000    0.000000
75%    668.500000    1.000000    3.000000    1.000000    35.000000    1.000000    0.000000
max     891.000000    1.000000    3.000000    1.000000    80.000000    8.000000    6.000000
```

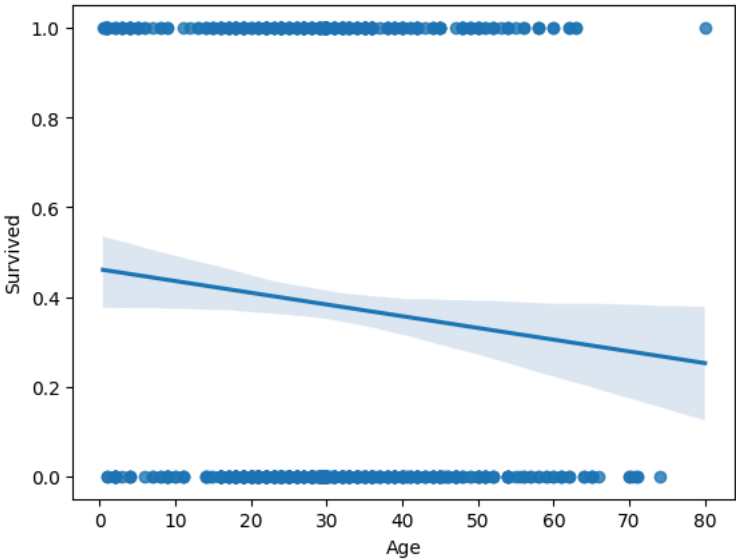
Part 2: Plot the Data

```
sns.regplot(x="Fare", y="Survived", data=df);
```



Part 3: Perform Simple Linear Regression on the SURVIVAL feature column

```
sns.regplot(x="Age", y="Survived", data=df);
```



▼ PART 2

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
path = "/content/drive/MyDrive/CPE 019 Emerging Technologies 3/HOA 4.1/titanic_train.csv"
training = pd.read_csv(path)
```

b) Verify the import and take a look at the data. Variable Description

- 1. PassengerID 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 children 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)

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

- Yes

```
#Code cell 3
#view the first few rows of the data
#
training.head(5)
```

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

Next steps: [View recommended plots](#)

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.

```
#code cell 4
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.

```
#code cell 5
#view the first few rows of the data again
training.head(5)
```

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

Next steps: [View recommended plots](#)

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.

```
#code cell 6
training["Age"].fillna(training["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.
training["Age"].head(10)
```

```
0    22.000000
1    38.000000
2    26.000000
3    35.000000
4    35.000000
```



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  
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.

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

b) Install Graphviz

o 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.

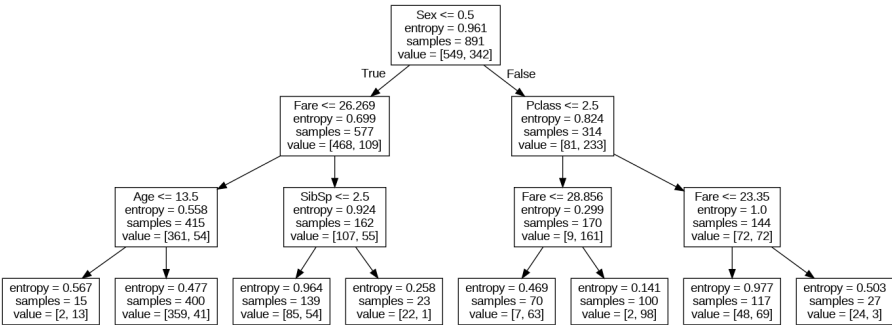
The dot file that was created above can be converted to a .png file with the graphiz dot renderer. This is a shell command, so use ! before it to run it from this notebook. The new titanic.png graphic file should appear in the directory that contains this notebook.

```
#code cell 13
#run the Graphviz dot command to convert the .dot file to .png
!dot -Tpng /content/drive/MyDrive/titanic.dot -o /content/drive/MyDrive/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 library
from IPython.display import Image
#display the decison tree graphic
Image("/content/drive/MyDrive/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 criterion 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.

What describes the group that had the most deaths by number? Which group had the most survivors?  
Because the male demograph was higher than female, male death rate was more compared to the female death rate.

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 In this step, you will import and prepare the data for analysis.

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.
path = "/content/drive/MyDrive/CPE 019 Emerging Technologies 3/HOA 4.1/titanic_train.csv"
testing = pd.read_csv(path)
testing.info(5)
```

```
<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
```

How many records are in the data set?  
The dataset has 891  
Which important variables(s) are missing values and how many are missing?  
fare values are 86 while only 1 was missing

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
testing["Sex"] = testing["Sex"].apply(lambda toLabel: 0 if toLabel == 'male' else 1)
training.head(5)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500		S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs)	1	38.0	1	0	PC 17599	71.2833	C 85	C

Next steps: [View recommended plots](#)

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)
```

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.

```
#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(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	1	38.000000	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	1	26.000000	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.000000	1	0	113803	53.1000

Next steps:

 View recommended plots

Step 2: Label the testing dataset

In this step, you will apply the learned model to 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
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 argument 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 argument
target_labels = pd.DataFrame({'Est_Survival':target_labels, 'Name':testing['Name']})
testing.head(20)
```



	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.000000	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	1	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.000000	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	0	35.000000	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	0	29.699118	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	0	54.000000	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	0	2.000000	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	1	27.000000	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	1	14.000000	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	1	4.000000	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	1	58.000000	0	0	113783	26.5500	C103	S
12	13	0	3	Saunderscock, Mr. William Henry	0	20.000000	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	0	39.000000	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	1	14.000000	0	0	350406	7.8542	NaN	S

Next steps:

 View recommended plots

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
path = "/content/drive/MyDrive/CPE 019 Emerging Technologies 3/HOA 4.1/titanic_train.csv"
all_data = pd.read_csv(path)
# 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 oservations. Store this in in the variable acc.
acc = np.sum(testing_results['Est_Survival'] == testing_results['Survived']) /float(len(testing_results))
# Print the result
print(acc)

0.8226711560044894
```

✓ Part 3: Evaluate the Decision Tree Model

The sklearn library includes a module that can be used to evaluate the accuracy 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 accuracy 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.

```
#code cell 22
#import the titanic_all.csv file into a dataframe called all_data. Specify the list of columns to import.
path = "/content/drive/MyDrive/CPE 019 Emerging Technologies 3/HOA 4.1/titanic_train.csv"
all_data = pd.read_csv(path, usecols=['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Fare'])
#View info for the new dataframe
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   Survived    891 non-null    int64
 1   Pclass      891 non-null    int64
 2   Sex         891 non-null    object
 3   Age         714 non-null    float64
 4   SibSp       891 non-null    int64
 5   Fare        891 non-null    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 41.9+ KB
```

How many records are in the data set?

891

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

177

Step 2: Prepare the data.

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

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

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
#display the first few rows of the data set
all_data["Age"].fillna(all_data["Age"].mean(), inplace=True)
```

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 variables 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 library
from sklearn.model_selection import train_test_split
#create the input and target variables as uppercase X and lowercase y. Reuse the columns variable.
X = all_data[list(columns)].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 strings using str()
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 = 0.8277153558052435 Testing score = 0.803921568627451
```

▼ Part 4 For Further Study (optional)

- 1. Remove observations with missing Age values.

Using a mean to replace missing age values may affect the accuracy of the model. One approach to this might be to remove all observations