Name: Kelly Joseph Calvadores
Course and Section: CPE 019 - CPE32S3
Date of Submission: January 31, 2024
Instructor: Engr. Roman M. Richard

Part A

Do the following objectives:
Part 1: Import the Libraries and Data
Part 2: Plot the Data
Part 3: Perform Simple Linear Regression on the SURVIVAL feature column (you can check the internet on how you can perform simple linear regression)

Part 1: Import the Libraries and Data

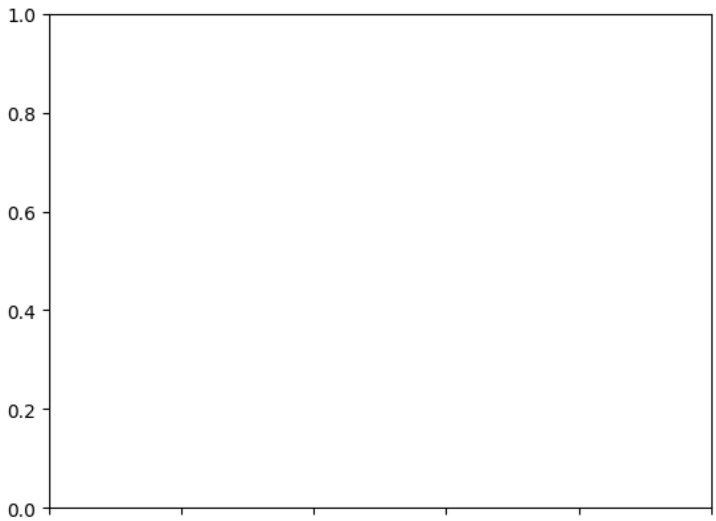
+ Code + Text

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns


TitanicTest = "titanic_test.csv"
TestFrame = pd.read_csv(TitanicTest)


#NOTE: there are 2 titanic train but it is still same content
#Error: ParserError: Error tokenizing data. C error: Expected 2 fields in line 6, saw 3 (Don't know how to solve)
TitanicTrain = "titanictrain.csv"
TrainFrame = pd.read_csv(TitanicTrain)
```

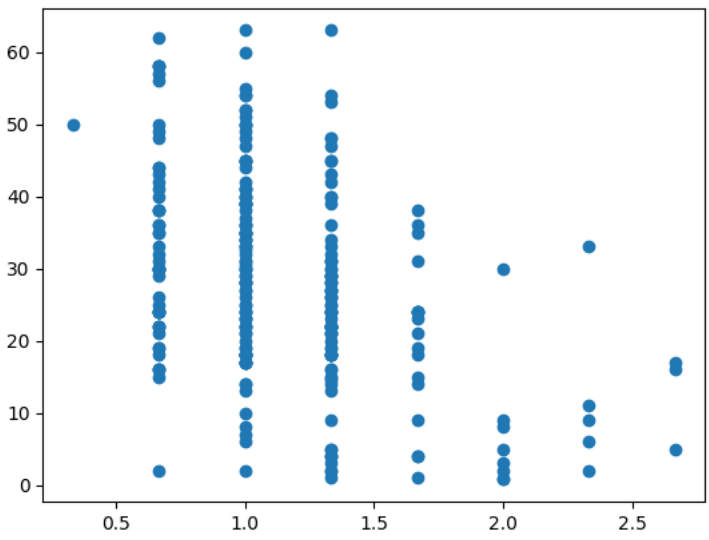
Part 2: Plot the Data

```
#Train the Gender/Sex in TrainFrame
MenTrain = TrainFrame[(TrainFrame.Sex == 'male')]
WomenTrain = TrainFrame[(TrainFrame.Sex == 'female')]

MenTrainMean = MenTrain[["Survived", "Pclass", "SibSp"]].mean(axis = 1)
WomenTrainMean = WomenTrain[["Survived", "Pclass", "SibSp"]].mean(axis = 1)


#Men(TitanicTrain)
plt.scatter(MenTrainMean, MenTrain["Age"])
plt.show()
```

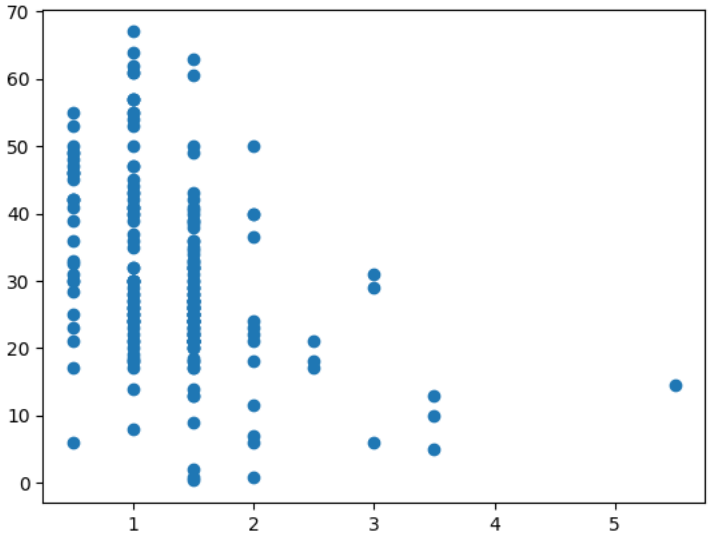
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-118-327f81377eed> in <cell line: 2>()
      1 #Men(TitanicTrain)
----> 2 plt.scatter(MenTrainMean, MenTrain["Age"])
      3 plt.show()

                                 2 frames
/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py in scatter(self, x,
y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors,
plotnonfinite, **kwargs)
   4582         y = np.ma.ravel(y)
   4583         if x.size != y.size:
-> 4584             raise ValueError("x and y must be the same size")
   4585
   4586         if s is None:

ValueError: x and y must be the same size
```



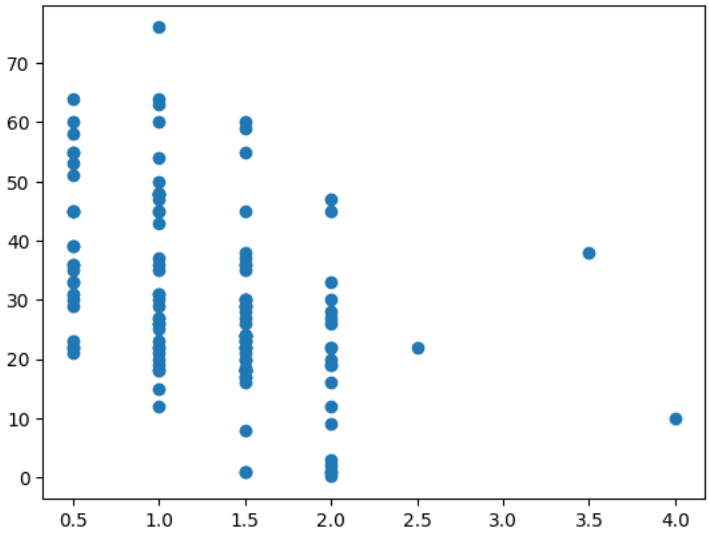

```
#Women(TitanicTrain)
plt.scatter(WomenTrainMean, WomenTrain["Age"])
plt.show()
```




```
#Train the Gender/Sex in TestFrame
MenTest = TestFrame[(TestFrame.Sex == 'male')]
WomenTest = TestFrame[(TestFrame.Sex == 'female')]

MenTestMean = MenTest[["Pclass", "SibSp"]].mean(axis = 1)
WomenTestMean = WomenTest[["Pclass", "SibSp"]].mean(axis = 1)


#Men(TitanicTest)
plt.scatter(MenTestMean, MenTest["Age"])
plt.show()
```

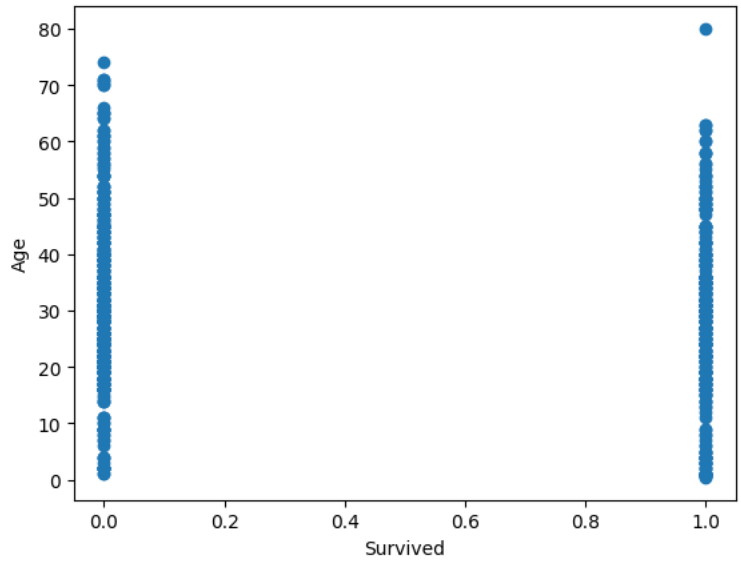



```
#Women(TitanicTest)
plt.scatter(WomenTestMean, WomenTest["Age"])
plt.show()
```

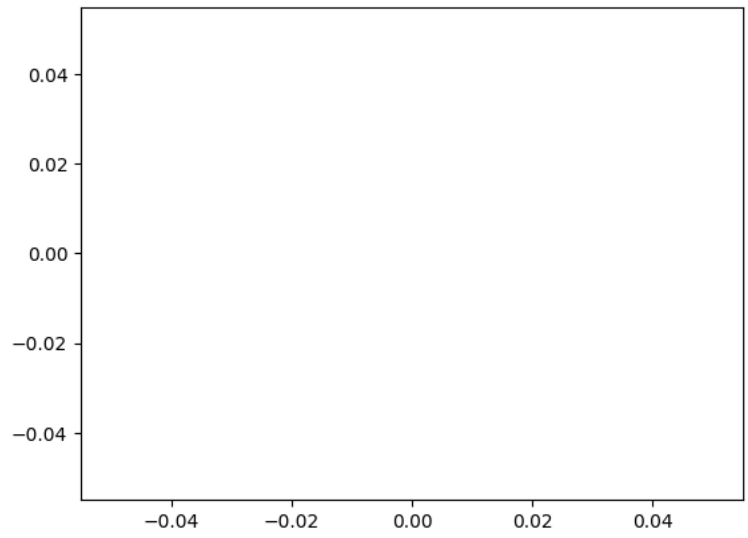


Part 3: Perform Simple Linear Regression on the SURVIVAL feature column (you can check the internet on how you can perform simple linear regression)

```
#Use TrainFrame for Referrence
x = TrainFrame['Survived']
y = TrainFrame['Age']
plt.scatter(x, y)
plt.xlabel('Survived')
plt.ylabel('Age')
plt.show()
```




```
# Convertion of data into arrays
x = np.array(x)
y = np.array(y)


# Calculate
b1, b0 = np.polyfit(x, y, 1)

#Regression line
plt.plot(x, b1*x + b0)
plt.show()
```



Part B

Part 1: Create a Decision Tree Classifier

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: My question will be, Does the survivability depends on passenger ticket class?

Step 1: Create the dataframe

a) Import pandas and the csv file

```
import pandas as pd

training = pd.read_csv("titanic_train.csv")
```

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

```
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?

- There are several missing values in the columns of age and class

```
training.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
				Futrelle, Mrs. Jacques						

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

a) Replace string data with numeric labels

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

b) Verify that the Gender variable has been changed.

```
training.head(10)
```

	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	7.250(
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.283:
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.925(
				Futrelle, Mrs. Jacques						

c) Address Missing Values in the Dataset

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

d) Verify that the values have been replaced.

```
training.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	0	22.000000	1	0	A/5 21171	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.000000	1	0	PC 17599	7
2	3	1	3	Heikkinen, Miss. Laina	1	26.000000	0	0	STON/O2. 3101282	
				Futrelle, Mrs. Jacques						

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

- The value that was used to replace the missing ages is the value with the float parameter such as in PassengerId 6 where his age is NaN.

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.

```
y_target = training["Survived"].values
```

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

```
columns = ["Fare", "Pclass", "Sex", "Age", "SibSp"]

X_input = training[list(columns)].values
```

c) Create the learned model

```
#import the tree module from the sklearn library
from sklearn import tree

clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)
clf_train = clf_train.fit(X_input, y_target)
```

d) Evaluate the model

```
(clf_train.score(X_input,y_target))*100
```

```
82.26711560044893
```

Step 6: Visualize the Tree

a) Create the intermediate file output

```
from six import StringIO
with open("titanic_train.csv", 'w') as f:
  f = tree.export_graphviz(clf_train, out_file=f, feature_names=columns)
```

b) Install Graphviz

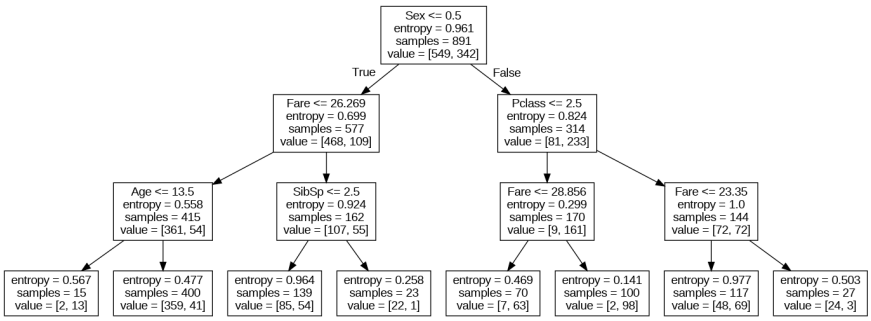
```
#This cannot be done in notebook due to this code is used in terminal
#apt-get install graphviz
```

c) Convert the intermediate file to a graphic

```
!dot -Tpng titanic_train.csv -o titanic_train.png
```

d) Display the image

```
from IPython.display import Image

Image("titanic_train.png")
```



e) Interpret the tree

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

- The group that had the most death by number is with the ticket class with only Fare, and the group that has the most survivors is the one with the Pclass.

Part 2: Apply the Decision Tree Model

Step 1: Import and Prepare the Data

a) Import the data.

```
testing = pd.read_csv("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
```

```
testing.head(10)
```

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

How many records are in the data set?

- There are atleast thousand records in this data set.

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

- The variables that are missing values is age and cabin. There are several values that are missing in the column age and cabin.

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

```
testing["Sex"] = testing["Sex"].apply(lambda toLabel: 0 if toLabel == 'male' else 1)
testing.head(10)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	892	3	Kelly, Mr. James	0	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	1	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	0	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	0	27.0	0	0	315154	8.6625	NaN	
			Hirvonen, Mrs.								

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

```
testing["Age"].fillna(testing["Age"].mean(), inplace = True)
testing.head(10)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	892	3	Kelly, Mr. James	0	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	1	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	0	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	0	27.0	0	0	315154	8.6625	NaN	
			Hirvonen, Mrs.								

d) Verify that the values have been replaced.

```
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    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
```

```
testing.head(12)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	892	3	Kelly, Mr. James	0	34.50000	0	0	330911	7.8292	Nal
1	893	3	Wilkes, Mrs. James (Ellen Needs)	1	47.00000	1	0	363272	7.0000	Nal
2	894	2	Myles, Mr. Thomas Francis	0	62.00000	0	0	240276	9.6875	Nal
3	895	3	Wirz, Mr. Albert	0	27.00000	0	0	315154	8.6625	Nal
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	1	22.00000	1	1	3101298	12.2875	Nal

Step 2: Label the testing dataset

Replace the columns with the Nan values into 0

```
testing["Fare"] = testing["Fare"].apply(lambda toLabel: 0 if pd.isnull(toLabel) else toLabel)
testing["Cabin"] = testing["Cabin"].apply(lambda toLabel: 0 if pd.isnull(toLabel) else toLabel)

testing.head(10)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Em
0	892	3	Kelly, Mr. James	0	34.5	0	0	330911	7.8292	0	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	1	47.0	1	0	363272	7.0000	0	
2	894	2	Myles, Mr. Thomas Francis	0	62.0	0	0	240276	9.6875	0	
3	895	3	Wirz, Mr. Albert	0	27.0	0	0	315154	8.6625	0	
			Hirvonen, Mrs.								

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

```
X_input = testing[list(columns)].values
```

b) Apply the model to the testing data set.

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

target_labels.head(10)
```

	Est_Survival	Name
0	0	Kelly, Mr. James
1	1	Wilkes, Mrs. James (Ellen Needs)
2	0	Myles, Mr. Thomas Francis
3	0	Wirz, Mr. Albert
4	1	Hirvonen, Mrs. Alexander (Helga E Lindqvist)
5	0	Svensson, Mr. Johan Cervin
6	1	Connolly, Miss. Kate
7	0	Caldwell, Mr. Albert Francis
8	1	Abrahim, Mrs. Joseph (Sophie Halaut Easu)
9	0	Davies, Mr. John Samuel

c) Evaluate the accuracy of the estimated labels

```
import numpy as np
all_data = pd.read_csv("titanic_all.csv")
testing_results = pd.merge(target_labels, all_data[['Name', 'Survived']], on=['Name'])

acc = np.sum(testing_results['Est_Survival'] == testing_results['Survived']) / float(len(testing_results))
print(acc)
```

```
0.7682619647355163
```

Part 3: Evaluate the Decision Tree Model

Step 1: Import the data

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

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
```

```
all_data.head(10)
```

	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
5	1	1	male	48.0000	0	26.5500
6	1	1	female	63.0000	1	77.9583
7	0	1	male	39.0000	0	0.0000
8	1	1	female	53.0000	2	51.4792
9	0	1	male	71.0000	0	49.5042

How many records are in the data set?

- There are 1309 records in the data set

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

- The important variables that are missing values are column Age and Column Cabin. In Age, there are several values that are missing while in cabin there are a bit values are missing.

Step 2: Prepare the data.

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

```
all_data["Gender"] = all_data["Gender"].apply(lambda toLabel: 0 if toLabel == 'male' else 1)
all_data.head(10)
```

	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
5	1	1	0	48.0000	0	26.5500
6	1	1	1	63.0000	1	77.9583
7	0	1	0	39.0000	0	0.0000
8	1	1	1	53.0000	2	51.4792
9	0	1	0	71.0000	0	49.5042

c) Replace the missing age values with the mean of the age of all members of the data set.

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