HOA_4_Regression_TITANIC

February 22, 2024

|

Activity 4.1

|

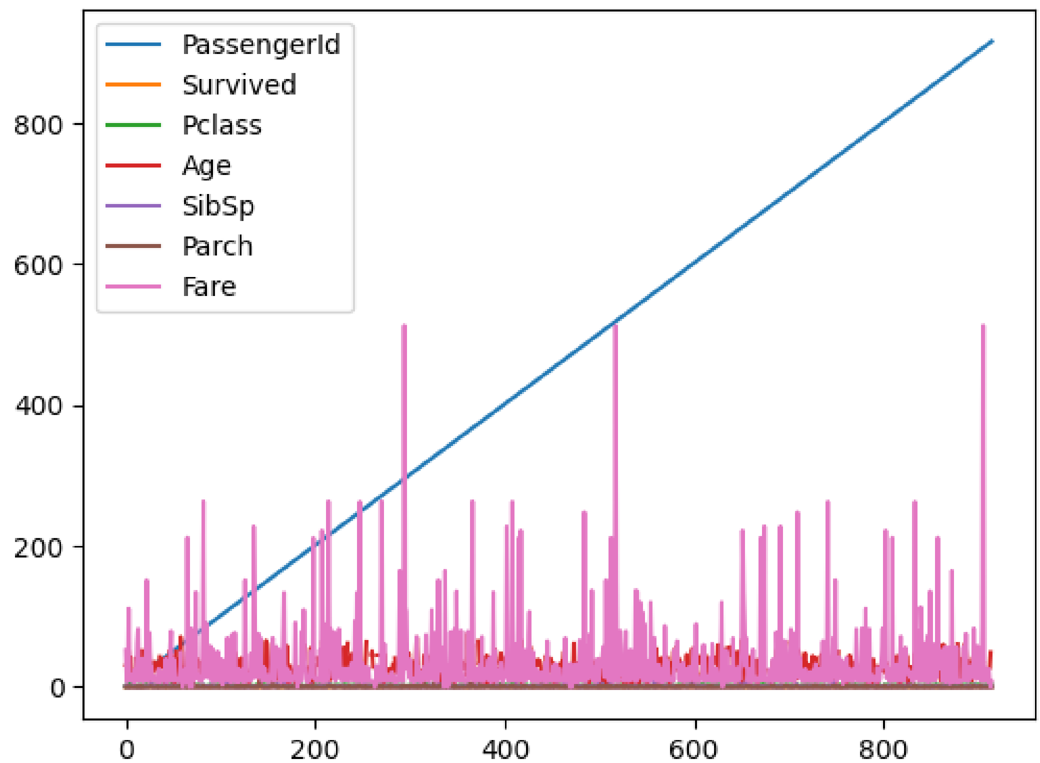
Advanced Data Analytics and Machine Learning

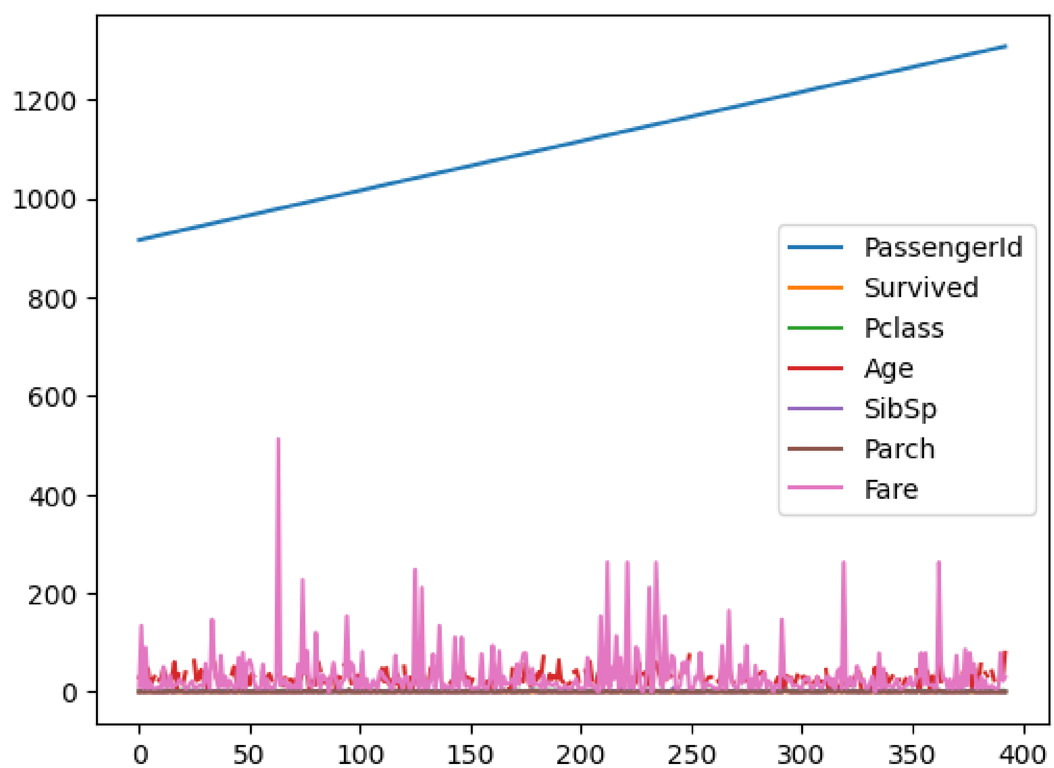
Name: Luigi T. Francisco Course and Section: CPE32S3 Date Submitted: 02/23/2024 Instructor: Engr. Roman Richard Date Performed: 02/23/2024 Date Submitted 02/23/2024

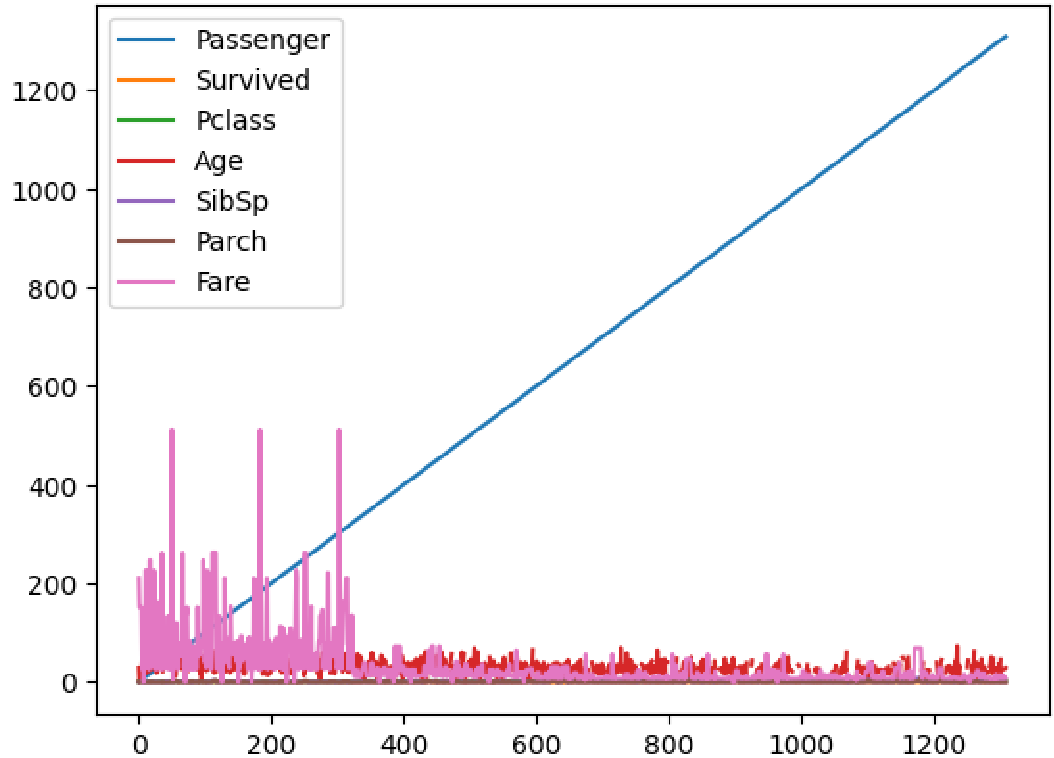
[85]:
```
#PART 1
import numpy as np
import matplotlib.pyplot as plt  # To visualize
import pandas as pd  # To read data
from sklearn.linear_model import LinearRegression
titanicTrain = pd.read_csv("titanic_train.csv")
titanicTest = pd.read_csv("titanic_test.csv")
titanicAll = pd.read_csv("titanic_all.csv")
```

[86]:
```
import matplotlib.pyplot as plt
titanicTrain.plot()
titanicTest.plot()
titanicAll.plot()
```

[86]:
```
<Axes: >
```


PassengerId
Survived
Pclass
Age
SibSp
Parch
Fare
0
200
400
600
800
0
200
400
600
800


PassengerId
Survived
Pclass
Age
SibSp
Parch
Fare
0
200
400
600
800
1000
1200
0
50
100
150
200
250
300
350
400



[87]:
```
titanicTrain.info()
titanicTest.info()
titanicAll.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 915 entries, 0 to 914
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  915 non-null    int64
 1   Survived     915 non-null    int64
 2   Pclass       915 non-null    int64
 3   Name         915 non-null    object
 4   Gender       915 non-null    object
 5   Age          738 non-null    float64
 6   SibSp        915 non-null    int64
 7   Parch        915 non-null    int64
 8   Ticket       915 non-null    object
 9   Fare         915 non-null    float64
 10  Cabin        202 non-null    object
```

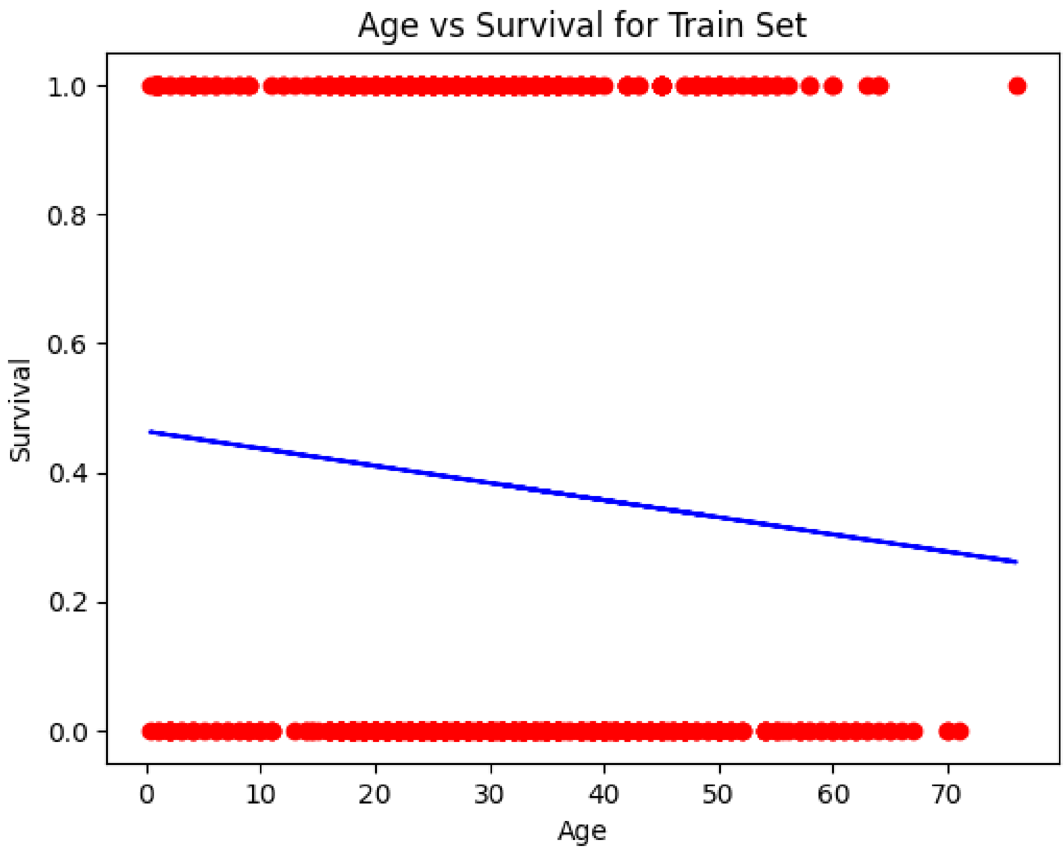
```
 11  Embarked     914 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 85.9+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 393 entries, 0 to 392
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  393 non-null    int64
 1   Survived     393 non-null    int64
 2   Pclass       393 non-null    int64
 3   Name         393 non-null    object
 4   Gender       393 non-null    object
 5   Age          307 non-null    float64
 6   SibSp        393 non-null    int64
 7   Parch        393 non-null    int64
 8   Ticket       393 non-null    object
 9   Fare         393 non-null    float64
 10  Cabin        93 non-null     object
 11  Embarked     392 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 37.0+ KB
<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
```

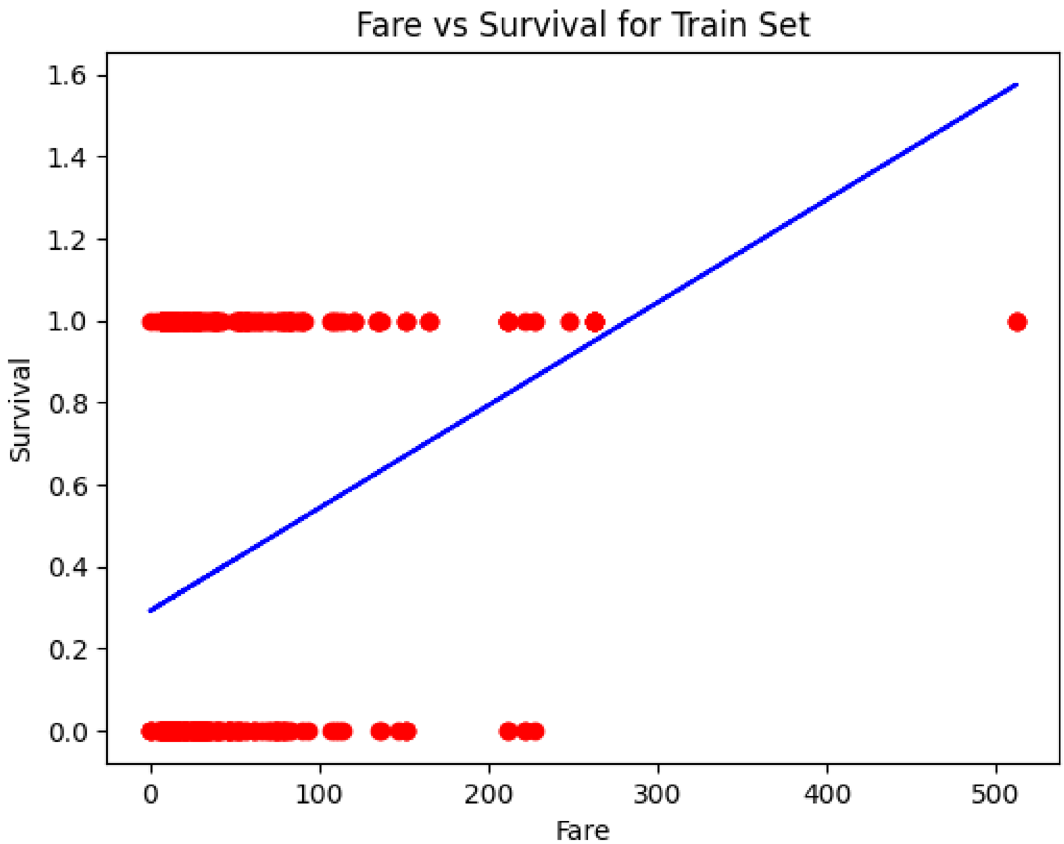
[88]:
```
#Lets apply the simple regression model but lets drop all the na entries for age
titanicTrain.dropna(subset=['Age'], inplace=True)
titanicTest.dropna(subset=['Age'], inplace=True)
titanicAll.dropna(subset=['Age'], inplace=True)
```

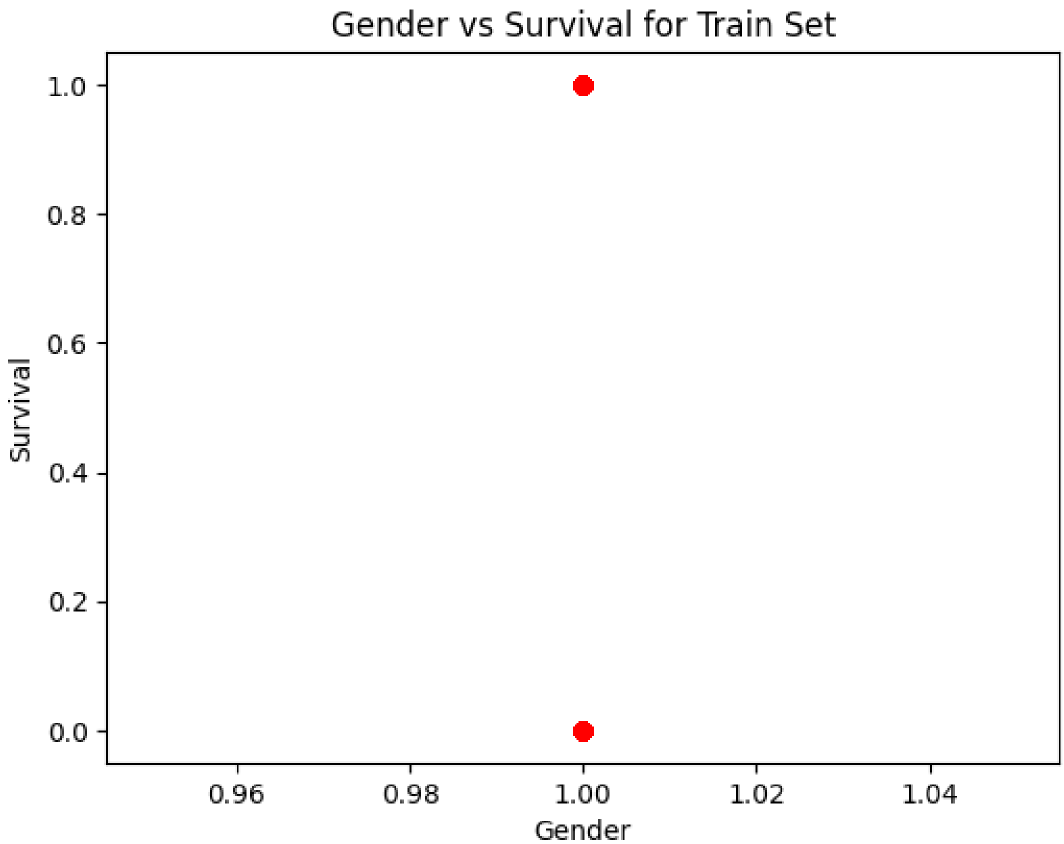
[138]:

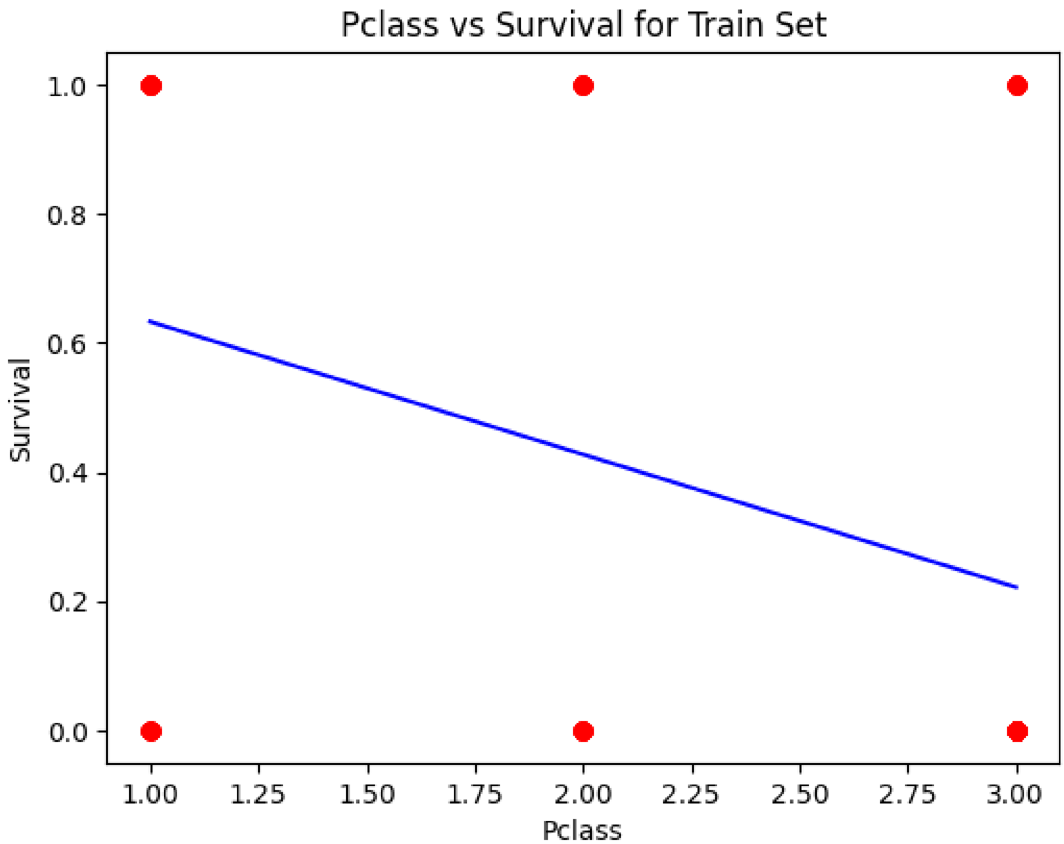
```python
#lets now perform some test
length1 = len(titanicTrain.index)
length2 = len(titanicTest.index)
length3 = len(titanicAll.index)

titanicTrain["Gender"] = titanicTrain["Gender"].apply(lambda toLabel:0 if␣
↪toLabel == 'male' else 1)
titanicTest["Gender"] = titanicTest["Gender"].apply(lambda toLabel:0 if toLabel␣
↪== 'male' else 1)
titanicAll["Gender"] = titanicAll["Gender"].apply(lambda toLabel:0 if toLabel␣
↪== 'male' else 1)

x_collect=[titanicTrain.Age.values,titanicTrain.Fare.values,titanicTrain.Gender.
↪values,titanicTrain.Pclass.values]
x_collect1=[titanicTest.Age.values,titanicTest.Fare.values,titanicTest.Gender.
↪values,titanicTest.Pclass.values]
x_collect2=[titanicAll.Age.values,titanicAll.Fare.values,titanicAll.Gender.
↪values,titanicAll.Pclass.values]
y_targets=[titanicTrain.Survived.values,titanicTest.Survived.values,titanicAll.
↪Survived.values]

x_description=["Age","Fare","Gender","Pclass"]



#x_inReg= titanicTrain.Age.values
#x_inReg1= x_inReg.reshape(length1, -1)

def doRegression(x_collector:list,y_outReg,x_description:list,dataLabel:
↪str,lengthSaid):
  count = 0
  for x_input in x_collector:
    x_inReg1= x_collector[count].reshape(lengthSaid, -1)
    y_outReger = y_outReg.reshape(lengthSaid, -1)
    regressor = LinearRegression()
    regressor.fit(x_inReg1, y_outReger)
    y_pred = regressor.predict(x_inReg1)
    plt.scatter(x_inReg1 ,y_outReger, color = 'red')
    plt.plot(x_inReg1, regressor.predict(x_inReg1), color = 'blue')
    plt.title(x_description[count]+' vs Survival for '+dataLabel)
    plt.xlabel(x_description[count])
    plt.ylabel('Survival')
    plt.show()
    count+=1
doRegression(x_collect,y_targets[0],x_description,"Train Set",length1)
doRegression(x_collect1,y_targets[1],x_description,"Testing Set",length2)
```

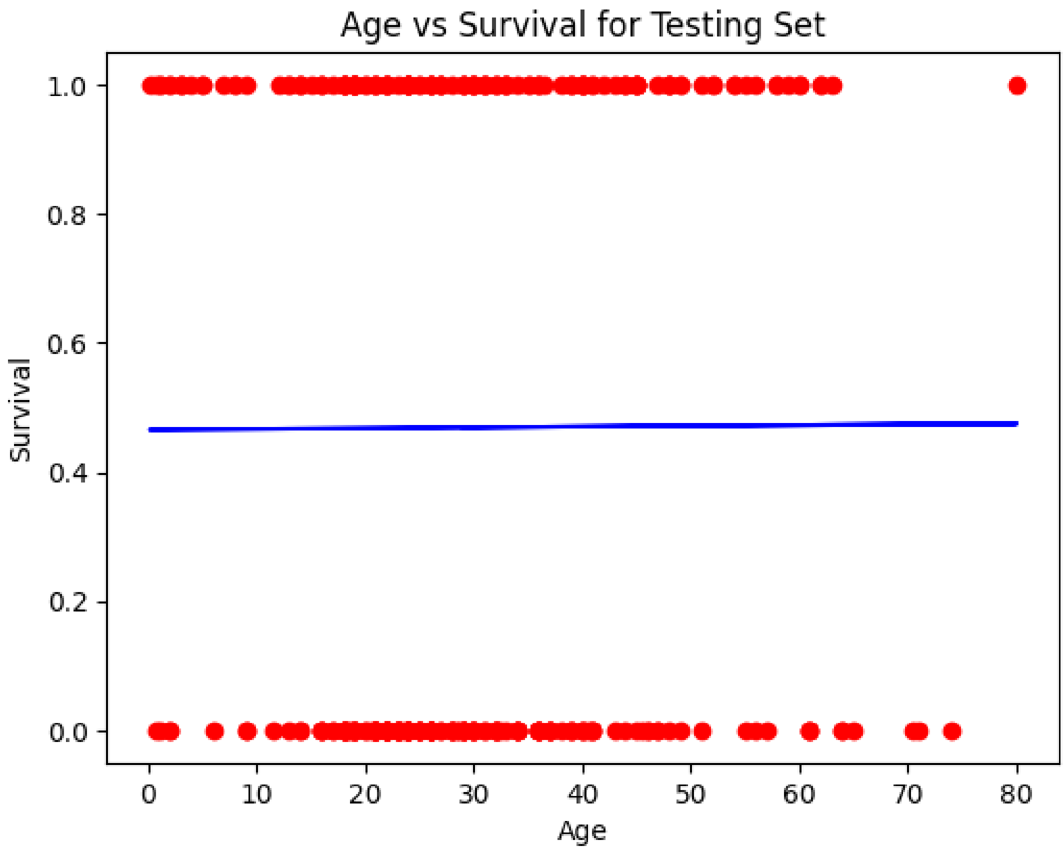
```
doRegression(x_collect2,y_targets[2],x_description,"All Set",length3)
```



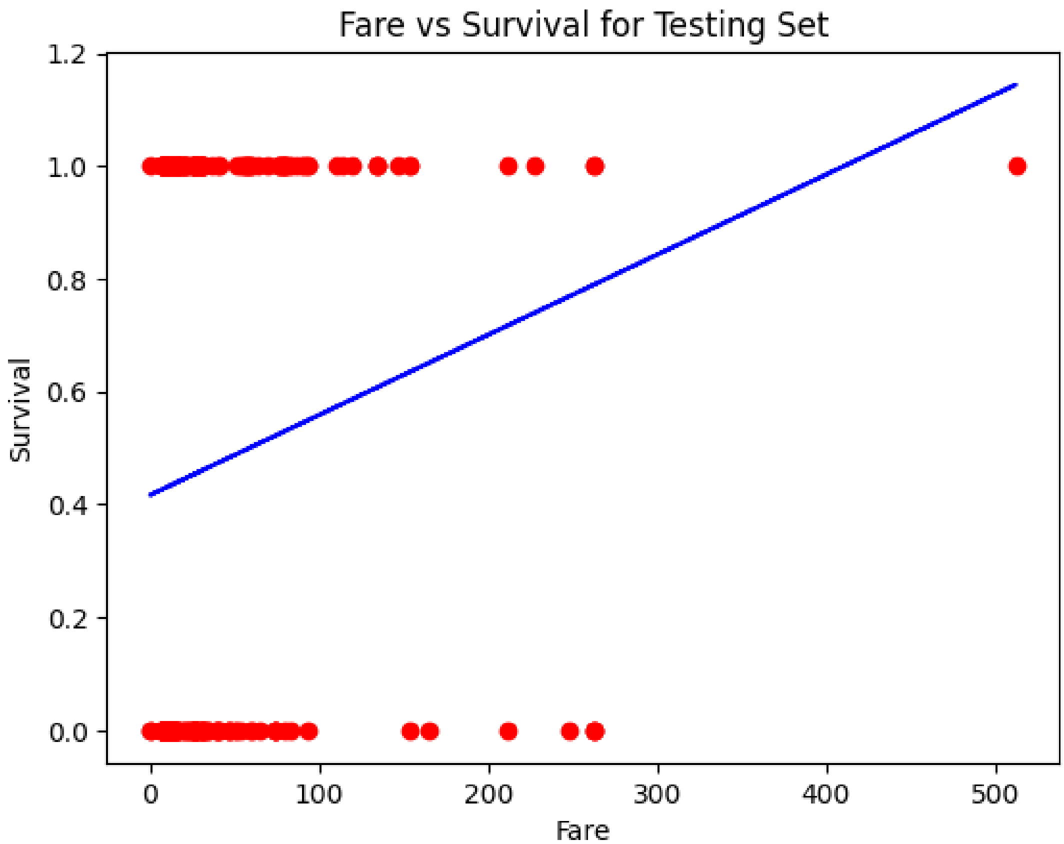

Fare vs Survival for Train Set
Survival
Fare
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500

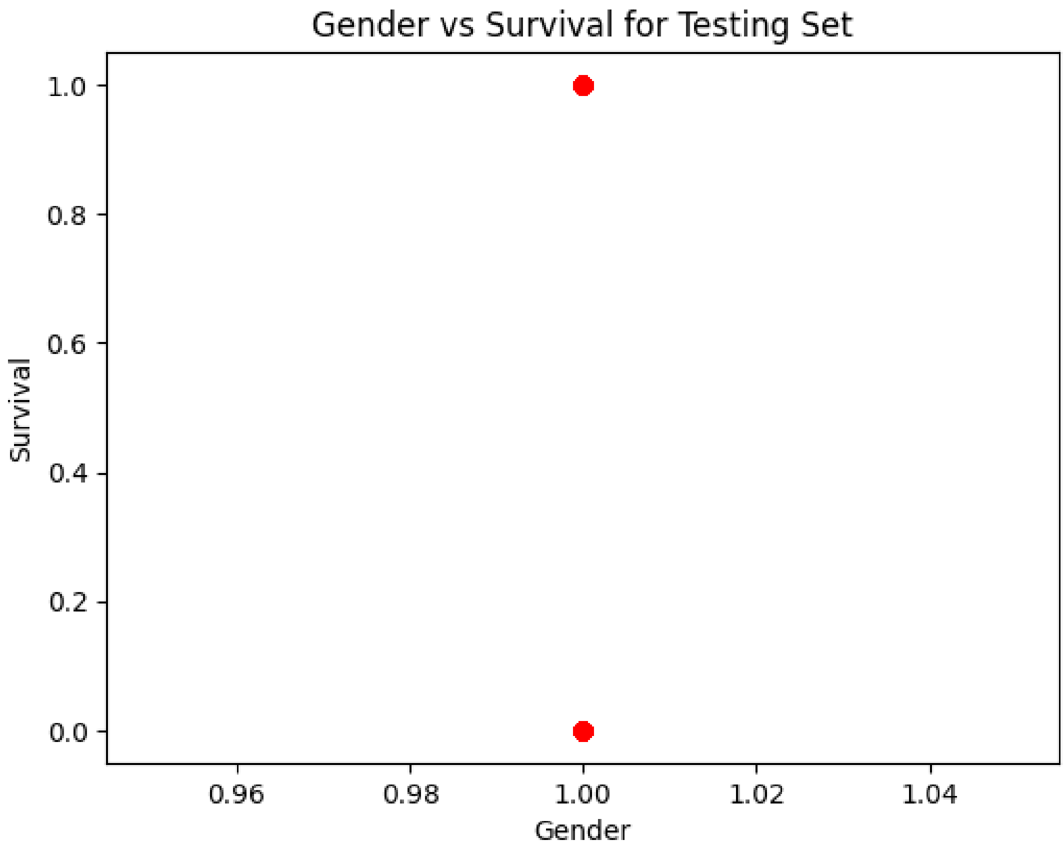

Gender vs Survival for Train Set
1.0
0.8
0.6
0.4
0.2
0.0
Survival
0.96
0.98
1.00
1.02
1.04
Gender

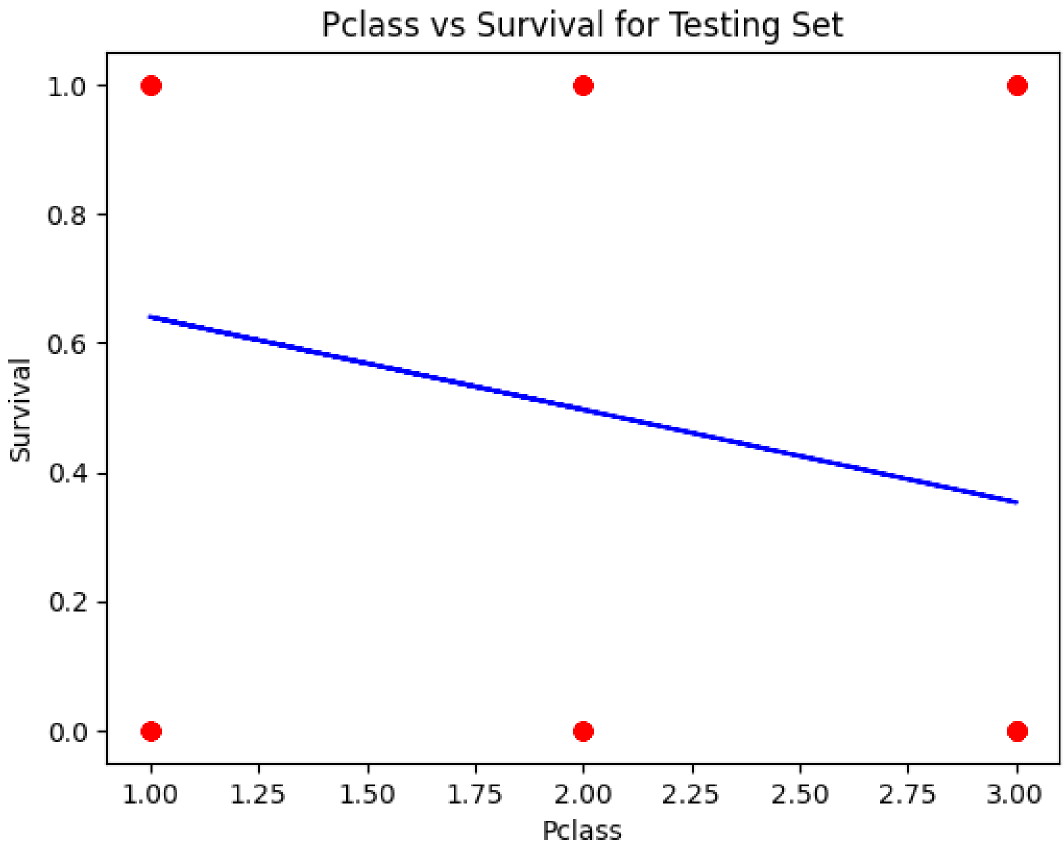

Pclass vs Survival for Train Set
1.0
0.8
0.6
0.4
0.2
0.0
Survival
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass

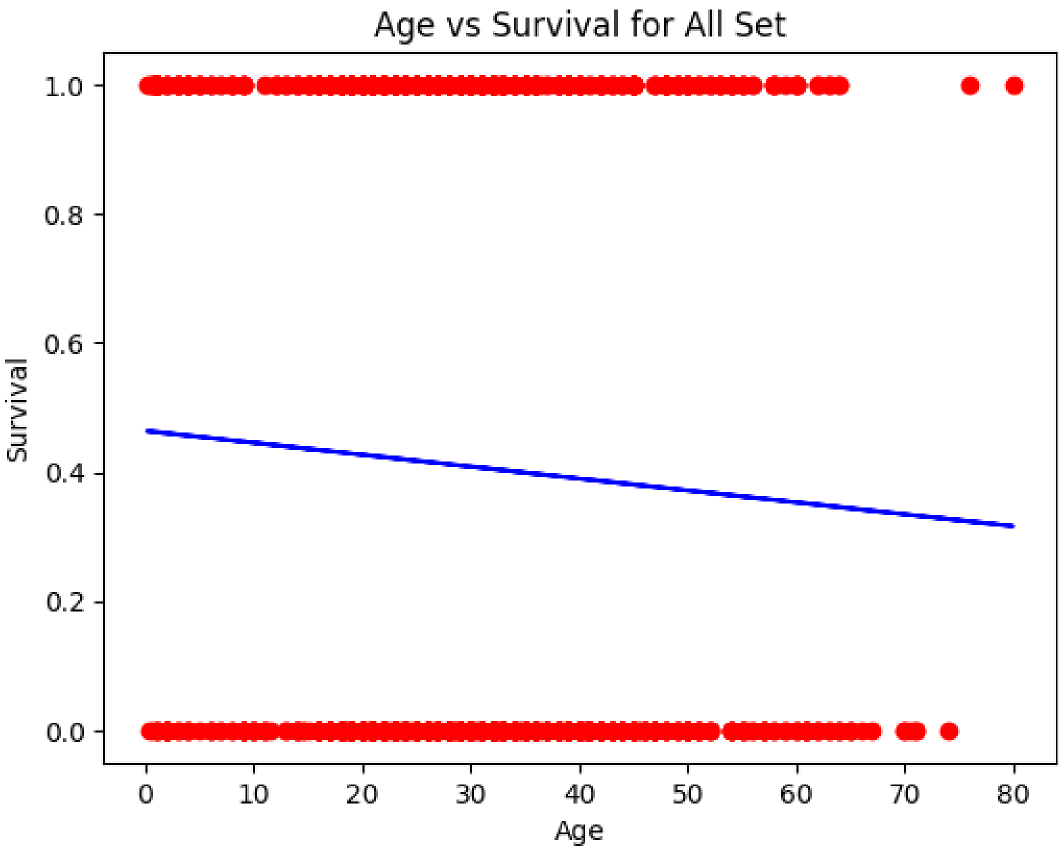


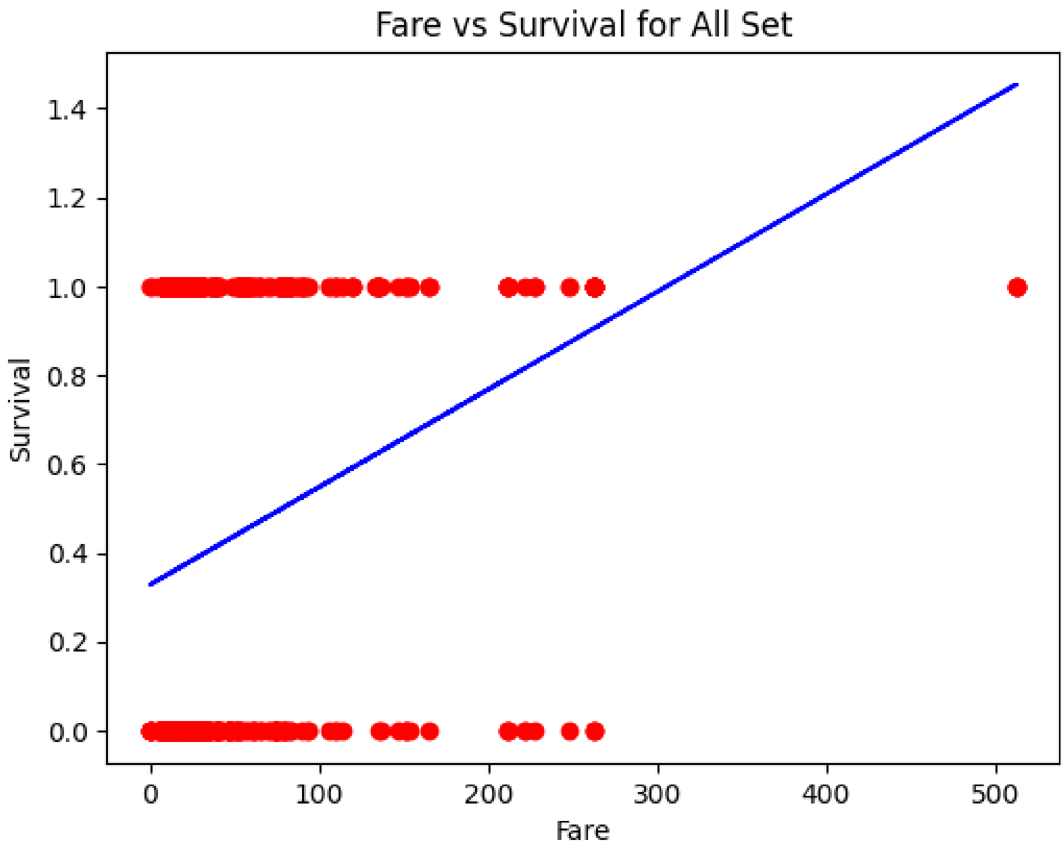
Age vs Survival for Testing Set
1.0
0.8
0.6
0.4
0.2
0.0
Survival
0
10
20
30
40
50
60
70
80
Age

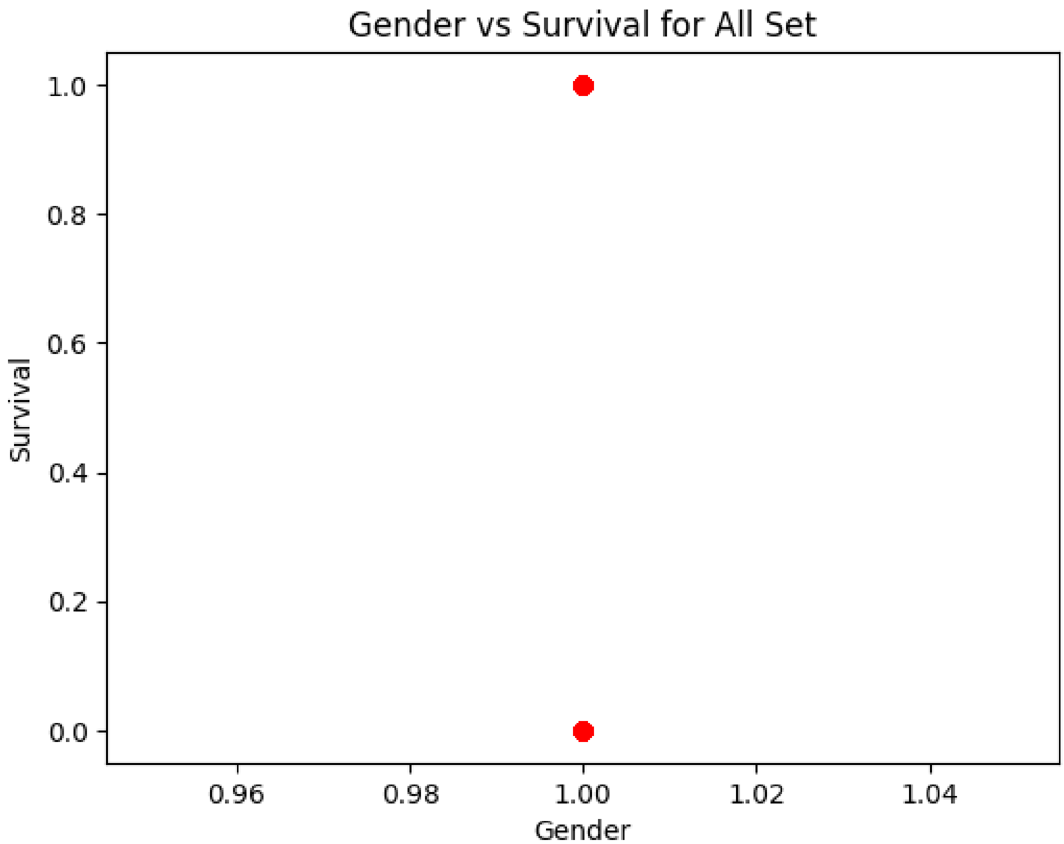

Fare vs Survival for Testing Set
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Survival
0
100
200
300
400
500
Fare

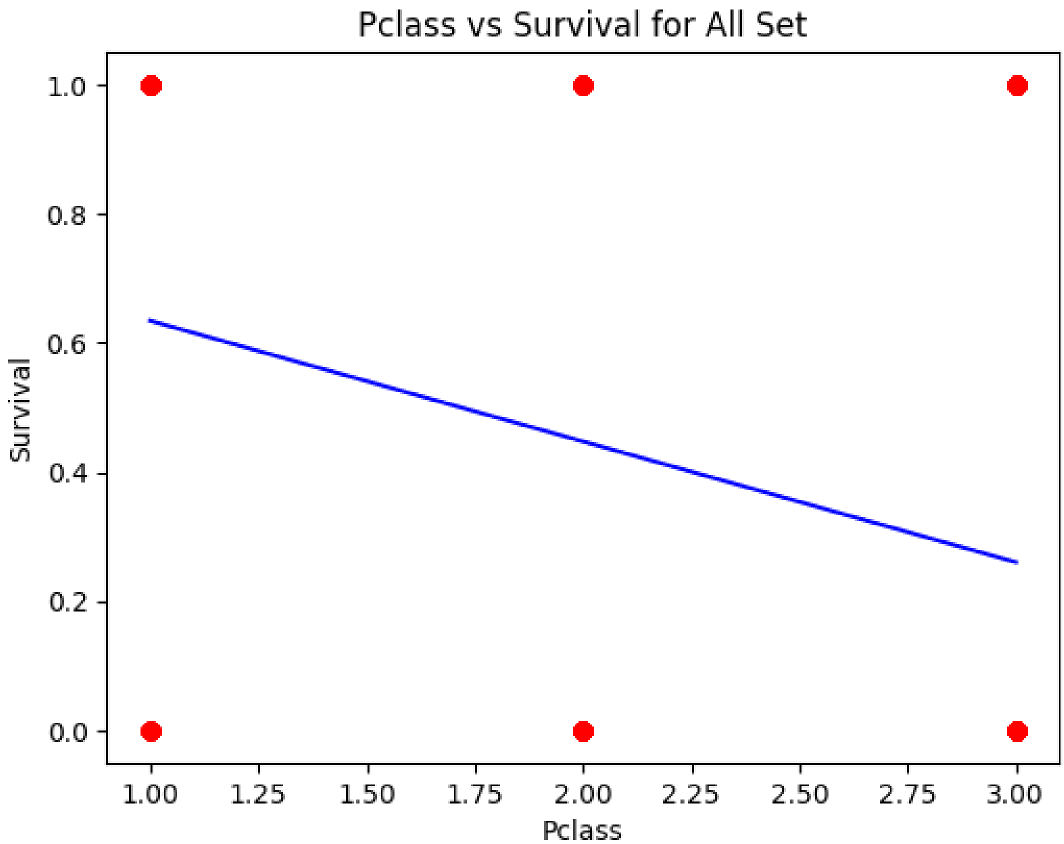

Gender vs Survival for Testing Set
1.0
0.8
0.6
0.4
0.2
0.0
Survival
0.96
0.98
1.00
1.02
1.04
Gender


Pclass vs Survival for Testing Set
1.0
0.8
0.6
0.4
0.2
0.0
Survival
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
Pclass


Age vs Survival for All Set
1.0
0.8
0.6
0.4
0.2
0.0
Survival
0
10
20
30
40
50
60
70
80
Age


Fare vs Survival for All Set
Survival
Fare
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500


Gender vs Survival for All Set
1.0
0.8
0.6
0.4
0.2
0.0
Survival
0.96
0.98
1.00
1.02
1.04
Gender



[137]:
```
titanicAll.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   int64
 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(6), object(4)
```

```
memory usage: 106.1+ KB
```

Conclusion for part 1

In plotting the data, It seems if i plotted every variable against each other all at once its hard to read.

Meanwhile, A simple regression is fascinating to do with numerous variable the thing though is that it doesn't seem to able to represent things very well if the values are only 1,0 which i gave to female and male against survival thats also 0 and 1.

Part 2 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?

How does gender, age, ticket class, and fare paid affect passenger survival in titanic?

[15]:
```
#Step 1 Create the data frame
# Lets import pandas and the csv file


# create a data frame for the training data set

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

[16]:
```
# Lets verify our work data frame !
training.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 915 entries, 0 to 914
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  915 non-null    int64
 1   Survived     915 non-null    int64
 2   Pclass       915 non-null    int64
 3   Name         915 non-null    object
 4   Gender       915 non-null    object
 5   Age          738 non-null    float64
 6   SibSp        915 non-null    int64
 7   Parch        915 non-null    int64
 8   Ticket       915 non-null    object
 9   Fare         915 non-null    float64
 10  Cabin        202 non-null    object
 11  Embarked     914 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 85.9+ KB
```

Are there any missing values in the data set?

Yes, In the Column of Age and Cabin

[102]:
```
# Lets check out the first 5 rows
training.head()
```

[102]:
```
   PassengerId  Survived  Pclass                         Name  Gender  \
0            1         0       1       Davidson, Mr. Thornton       0
1            2         0       3             Asim, Mr. Adola       0
2            3         0       3           Nankoff, Mr. Minko       0
3            4         0       1     Thayer, Mr. John Borland       0
4            5         0       3  Strandberg, Miss. Ida Sofia       1

         Age  SibSp  Parch             Ticket      Fare Cabin Embarked
0  31.000000      1      0         F.C. 12750   52.0000   B71        S
1  35.000000      0      0  SOTON/O.Q. 3101310    7.0500   NaN        S
2  29.970867      0      0             349218    7.8958   NaN        S
3  49.000000      1      1              17421  110.8833   C68        C
4  22.000000      0      0               7553    9.8375   NaN        S
```

[18]:
```
# STEP 2 Lets prepare the data for the decision Tree Model!!!

#scikit-learn(idk maybe science kit learn?) it states it can only process
#numeric data, thus maybe we have to convert things to numbers?

# this code seems to change the gender values with lambda
# Lambda seems to mean "if it is such thing then change it to this"
#Notably Gender = Sex in this scenario
training["Gender"] = training["Gender"].apply(lambda toLabel:0 if toLabel ==␣
 ↪'male' else 1)
```

[19]:
```
training.head()
```

[19]:
```
   PassengerId  Survived  Pclass                         Name  Gender   Age  \
0            1         0       1       Davidson, Mr. Thornton       0  31.0
1            2         0       3             Asim, Mr. Adola       0  35.0
2            3         0       3           Nankoff, Mr. Minko       0   NaN
3            4         0       1     Thayer, Mr. John Borland       0  49.0
4            5         0       3  Strandberg, Miss. Ida Sofia       1  22.0

   SibSp  Parch             Ticket      Fare Cabin Embarked
0      1      0         F.C. 12750   52.0000   B71        S
1      0      0  SOTON/O.Q. 3101310    7.0500   NaN        S
2      0      0             349218    7.8958   NaN        S
3      1      1              17421  110.8833   C68        C
4      0      0               7553    9.8375   NaN        S
```

Yep, it changed

[20]:
```
# Adress Missing Values of Age in the Data Set

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

[21]:
```
training.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 915 entries, 0 to 914
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  915 non-null    int64
 1   Survived     915 non-null    int64
 2   Pclass       915 non-null    int64
 3   Name         915 non-null    object
 4   Gender       915 non-null    int64
 5   Age          915 non-null    float64
 6   SibSp        915 non-null    int64
 7   Parch        915 non-null    int64
 8   Ticket       915 non-null    object
 9   Fare         915 non-null    float64
 10  Cabin        202 non-null    object
 11  Embarked     914 non-null    object
dtypes: float64(2), int64(6), object(4)
memory usage: 85.9+ KB
```

Verified that age have 891 entries thus those values were correctly replaced.

[22]:
```
training["Age"].mean()
```

[22]:
```
29.970867208672082
```

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

29.699 replaced the missing ages

[23]:
```
training.head(50) # for verification
```

[23]:
```
   PassengerId  Survived  Pclass  \
0            1         0       1
1            2         0       3
2            3         0       3
3            4         0       1
4            5         0       3
5            6         0       2
6            7         0       3
7            8         0       3
8            9         0       3
9           10         0       3
```

```
10           11         0       3
11           12         0       2
12           13         1       1
13           14         1       1
14           15         0       3
15           16         1       3
16           17         0       2
17           18         1       1
18           19         0       3
19           20         1       3
20           21         0       3
21           22         1       3
22           23         1       1
23           24         0       3
24           25         0       3
25           26         1       1
26           27         1       2
27           28         0       3
28           29         0       3
29           30         1       3
30           31         0       3
31           32         0       2
32           33         0       3
33           34         0       3
34           35         0       3
35           36         1       2
36           37         0       3
37           38         0       3
38           39         0       3
39           40         0       3
40           41         0       2
41           42         0       3
42           43         0       3
43           44         1       2
44           45         0       3
45           46         0       3
46           47         0       3
47           48         1       3
48           49         0       1
49           50         0       3

                                          Name  Gender        Age  \
0                      Davidson, Mr. Thornton       0  31.000000
1                            Asim, Mr. Adola       0  35.000000
2                          Nankoff, Mr. Minko       0  29.970867
3                    Thayer, Mr. John Borland       0  49.000000
4                Strandberg, Miss. Ida Sofia       1  22.000000
```

5	Bowenur, Mr. Solomon	0	42.000000
6	Bowen, Mr. David John "Dai"	0	21.000000
7	Assam, Mr. Ali	0	23.000000
8	Thomas, Mr. John	0	29.970867
9	Moran, Mr. Daniel J	0	29.970867
10	Lahoud, Mr. Sarkis	0	29.970867
11	Maybery, Mr. Frank Hubert	0	40.000000
12	Greenfield, Mr. William Bertram	0	23.000000
13	Snyder, Mrs. John Pillsbury (Nelle Stevenson)	1	23.000000
14	Mahon, Mr. John	0	29.970867
15	Nakid, Mr. Sahid	0	20.000000
16	Stokes, Mr. Philip Joseph	0	25.000000
17	Lines, Mrs. Ernest H (Elizabeth Lindsey James)	1	51.000000
18	Flynn, Mr. John	0	29.970867
19	O'Leary, Miss. Hanora "Norah"	1	29.970867
20	Zabour, Miss. Hileni	1	14.500000
21	Chip, Mr. Chang	0	32.000000
22	Cleaver, Miss. Alice	1	22.000000
23	Canavan, Miss. Mary	1	21.000000
24	Moen, Mr. Sigurd Hansen	0	25.000000
25	Harper, Mr. Henry Sleeper	0	48.000000
26	Quick, Miss. Winifred Vera	1	8.000000
27	Rice, Master. George Hugh	0	8.000000
28	Berglund, Mr. Karl Ivar Sven	0	22.000000
29	Murphy, Miss. Katherine "Kate"	1	29.970867
30	Brocklebank, Mr. William Alfred	0	35.000000
31	Sedgwick, Mr. Charles Frederick Waddington	0	25.000000
32	Jardin, Mr. Jose Neto	0	29.970867
33	Willey, Mr. Edward	0	29.970867
34	Goldsmith, Mr. Frank John	0	33.000000
35	Phillips, Miss. Kate Florence ("Mrs Kate Louis…	1	19.000000
36	Holm, Mr. John Fredrik Alexander	0	43.000000
37	Thomson, Mr. Alexander Morrison	0	29.970867
38	Salonen, Mr. Johan Werner	0	39.000000
39	Olsen, Mr. Henry Margido	0	28.000000
40	Malachard, Mr. Noel	0	29.970867
41	Mangan, Miss. Mary	1	30.500000
42	Panula, Master. Eino Viljami	0	1.000000
43	Louch, Mrs. Charles Alexander (Alice Adelaide …	1	42.000000
44	Braund, Mr. Lewis Richard	0	29.000000
45	Sadlier, Mr. Matthew	0	29.970867
46	Rouse, Mr. Richard Henry	0	50.000000
47	Bing, Mr. Lee	0	32.000000
48	Guggenheim, Mr. Benjamin	0	46.000000
49	Vander Planke, Miss. Augusta Maria	1	18.000000

SibSp Parch Ticket Fare Cabin Embarked

0	1	0	F.C. 12750	52.0000	B71	S
1	0	0	SOTON/O.Q. 3101310	7.0500	NaN	S
2	0	0	349218	7.8958	NaN	S
3	1	1	17421	110.8833	C68	C
4	0	0	7553	9.8375	NaN	S
5	0	0	211535	13.0000	NaN	S
6	0	0	54636	16.1000	NaN	S
7	0	0	SOTON/O.Q. 3101309	7.0500	NaN	S
8	0	0	2681	6.4375	NaN	C
9	1	0	371110	24.1500	NaN	Q
10	0	0	2624	7.2250	NaN	C
11	0	0	239059	16.0000	NaN	S
12	0	1	PC 17759	63.3583	D10 D12	C
13	1	0	21228	82.2667	B45	S
14	0	0	AQ/4 3130	7.7500	NaN	Q
15	1	1	2653	15.7417	NaN	C
16	0	0	F.C.C. 13540	10.5000	NaN	S
17	0	1	PC 17592	39.4000	D28	S
18	0	0	368323	6.9500	NaN	Q
19	0	0	330919	7.8292	NaN	Q
20	1	0	2665	14.4542	NaN	C
21	0	0	1601	56.4958	NaN	S
22	0	0	113781	151.5500	NaN	S
23	0	0	364846	7.7500	NaN	Q
24	0	0	348123	7.6500	F G73	S
25	1	0	PC 17572	76.7292	D33	C
26	1	1	26360	26.0000	NaN	S
27	4	1	382652	29.1250	NaN	Q
28	0	0	PP 4348	9.3500	NaN	S
29	1	0	367230	15.5000	NaN	Q
30	0	0	364512	8.0500	NaN	S
31	0	0	244361	13.0000	NaN	S
32	0	0	SOTON/O.Q. 3101305	7.0500	NaN	S
33	0	0	S.O./P.P. 751	7.5500	NaN	S
34	1	1	363291	20.5250	NaN	S
35	0	0	250655	26.0000	NaN	S
36	0	0	C 7075	6.4500	NaN	S
37	0	0	32302	8.0500	NaN	S
38	0	0	3101296	7.9250	NaN	S
39	0	0	C 4001	22.5250	NaN	S
40	0	0	237735	15.0458	D	C
41	0	0	364850	7.7500	NaN	Q
42	4	1	3101295	39.6875	NaN	S
43	1	0	SC/AH 3085	26.0000	NaN	S
44	1	0	3460	7.0458	NaN	S
45	0	0	367655	7.7292	NaN	Q
46	0	0	A/5 3594	8.0500	NaN	S

```
47       0       0                  1601    56.4958       NaN          S
48       0       0              PC 17593    79.2000   B82 B84          C
49       2       0                345764    18.0000       NaN          S
```

[24]:
```
#STEP 3 Lets Train and Score the Decision Tree Model

# creating our arrays this one is for the target variable!
y_target = training["Survived"].values
```

[25]:
```
# Create an array for the factors? Im not sure about the SibSp though.

columns=["Fare","Pclass","Gender","Age","SibSp"]
# the x variable for our y
x_input = training[list(columns)].values
```

[26]:
```
# Now we create the learned model
from sklearn import tree

# first we create the classifier object
clf_train = tree.DecisionTreeClassifier(criterion="entropy",max_depth=3)

#what is fit method  anyway ? well it said use fit() method of the decision␣
↪tree object
# so this fit variable gets input and output

clf_train = clf_train.fit(x_input, y_target)
```

[27]:
```
# EVALUATE THE MODEL ? How do it do that? we use score method of the object
clf_train.score(x_input,y_target)
```

[27]: 0.8163934426229508

So this means the model is somehow 82% correct of the time well that's what it said on the paper.

[28]:
```
# Step 6 Visualize the Tree
from six import StringIO
with open ("titanic.dot",'w') as f:
    f = tree.export_graphviz(clf_train,out_file=f,feature_names=columns)
```

Now we install graphiz so how do i do thaT?

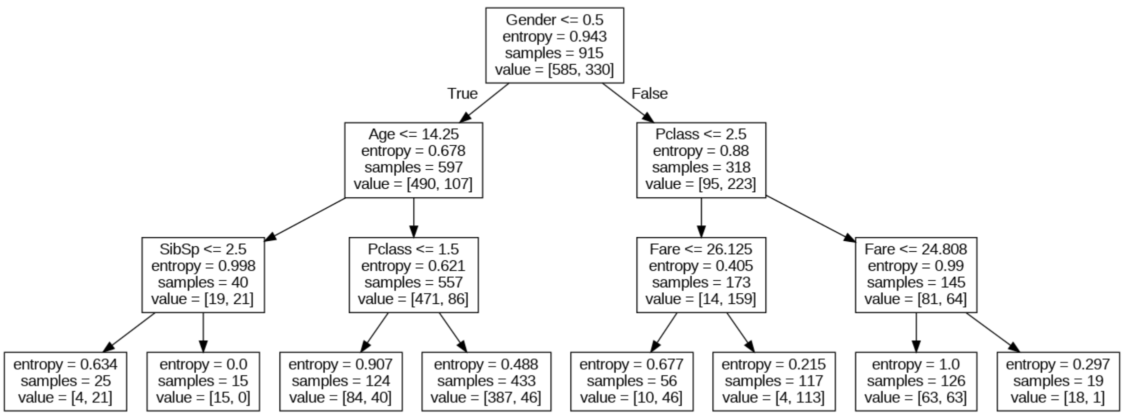
[29]:
```
!pip install graphviz
```

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

[30]:
```
# so i somehow installed it

!dot -Tpng titanic.dot -o titanic.png
```

[31]:
```
#image module import
from IPython.display import Image

Image("titanic.png")
```

[31]:



The group that had the most death was those that are men,paid lower fare above the age of 13.5 had the most death at 359 people.

The group that had the most survivors are women, Passenger class that are lower than 3 paying a fare greater than 28.856 with survivors of 98 people.

[32]:
```
# Lets apply the decision tree model

testing = pd.read_csv("titanic_test.csv")
```

[33]:
```
testing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 393 entries, 0 to 392
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  393 non-null    int64
 1   Survived     393 non-null    int64
 2   Pclass       393 non-null    int64
 3   Name         393 non-null    object
 4   Gender       393 non-null    object
 5   Age          307 non-null    float64
 6   SibSp        393 non-null    int64
 7   Parch        393 non-null    int64
```

```
 8   Ticket       393 non-null    object
 9   Fare         393 non-null    float64
 10  Cabin        93 non-null     object
 11  Embarked     392 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 37.0+ KB
```

How many records are in the data set? **418 records** Which important variables(s) are missing values and how many are missing? **The important Variables that have missing values are Age and Fare which have missing values of 86 and 1 respectively.**

[34]:
```
#replace the sex to either 0 or 1
testing["Gender"] = testing["Gender"].apply(lambda toLabel:0 if toLabel ==␣
↪'male' else 1)
#verify
testing.head()
```

[34]:
```
   PassengerId  Survived  Pclass  \
0          916         0       2
1          917         1       1
2          918         0       3
3          919         1       1
4          920         1       3

                                                Name  Gender   Age  SibSp  \
0                        Coleridge, Mr. Reginald Charles       0  29.0      0
1  Spedden, Mrs. Frederic Oakley (Margaretta Corn…       1  40.0      1
2                                  Windelov, Mr. Einar       0  21.0      0
3                               Minahan, Miss. Daisy E       1  33.0      1
4                       Wilkes, Mrs. James (Ellen Needs)       1  47.0      1

   Parch             Ticket    Fare Cabin Embarked
0      0        W./C. 14263   10.50   NaN        S
1      1              16966  134.50   E34        C
2      0  SOTON/OQ 3101317    7.25   NaN        S
3      0              19928   90.00   C78        Q
4      0             363272    7.00   NaN        S
```

[35]:
```
testing["Age"].fillna(testing["Age"].mean(), inplace=True)
testing["Fare"].fillna(testing["Fare"].mean(), inplace=True)
testing.info()
testing["Age"].mean()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 393 entries, 0 to 392
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
```

```
 0   PassengerId  393 non-null    int64
 1   Survived     393 non-null    int64
 2   Pclass       393 non-null    int64
 3   Name         393 non-null    object
 4   Gender       393 non-null    int64
 5   Age          393 non-null    float64
 6   SibSp        393 non-null    int64
 7   Parch        393 non-null    int64
 8   Ticket       393 non-null    object
 9   Fare         393 non-null    float64
 10  Cabin        93 non-null     object
 11  Embarked     392 non-null    object
dtypes: float64(2), int64(6), object(4)
memory usage: 37.0+ KB
```

[35]: 29.565689576547232

[36]:
```
testing.head(30)
```

[36]:
```
    PassengerId  Survived  Pclass  \
0           916         0       2
1           917         1       1
2           918         0       3
3           919         1       1
4           920         1       3
5           921         0       3
6           922         0       3
7           923         0       3
8           924         0       3
9           925         0       2
10          926         0       3
11          927         0       1
12          928         1       3
13          929         1       3
14          930         0       3
15          931         1       3
16          932         0       2
17          933         1       3
18          934         0       3
19          935         0       3
20          936         0       3
21          937         0       3
22          938         0       3
23          939         1       3
24          940         0       3
25          941         1       3
26          942         0       3
```

27	943	1	2
28	944	1	3
29	945	0	3

	Name	Gender	Age \
0	Coleridge, Mr. Reginald Charles	0	29.00000
1	Spedden, Mrs. Frederic Oakley (Margaretta Corn…	1	40.00000
2	Windelov, Mr. Einar	0	21.00000
3	Minahan, Miss. Daisy E	1	33.00000
4	Wilkes, Mrs. James (Ellen Needs)	1	47.00000
5	Abbott, Mr. Rossmore Edward	0	16.00000
6	Karlsson, Mr. Nils August	0	22.00000
7	Connaghton, Mr. Michael	0	31.00000
8	Foley, Mr. William	0	29.56569
9	Leyson, Mr. Robert William Norman	0	24.00000
10	Henriksson, Miss. Jenny Lovisa	1	28.00000
11	Brandeis, Mr. Emil	0	48.00000
12	de Mulder, Mr. Theodore	0	30.00000
13	Turja, Miss. Anna Sofia	1	18.00000
14	Ford, Mr. Edward Watson	0	18.00000
15	Osman, Mrs. Mara	1	31.00000
16	Lingane, Mr. John	0	61.00000
17	Touma, Master. Georges Youssef	0	7.00000
18	Van Impe, Mr. Jean Baptiste	0	36.00000
19	Johnston, Mr. Andrew G	0	29.56569
20	Chronopoulos, Mr. Demetrios	0	18.00000
21	Badt, Mr. Mohamed	0	40.00000
22	Rasmussen, Mrs. (Lena Jacobsen Solvang)	1	29.56569
23	de Messemaeker, Mrs. Guillaume Joseph (Emma)	1	36.00000
24	Samaan, Mr. Elias	0	29.56569
25	Turkula, Mrs. (Hedwig)	1	63.00000
26	Augustsson, Mr. Albert	0	23.00000
27	Davis, Miss. Mary	1	28.00000
28	Hansen, Mrs. Claus Peter (Jennie L Howard)	1	45.00000
29	Jussila, Miss. Mari Aina	1	21.00000

	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	0	W./C. 14263	10.5000	NaN	S
1	1	1	16966	134.5000	E34	C
2	0	0	SOTON/OQ 3101317	7.2500	NaN	S
3	1	0	19928	90.0000	C78	Q
4	1	0	363272	7.0000	NaN	S
5	1	1	C.A. 2673	20.2500	NaN	S
6	0	0	350060	7.5208	NaN	S
7	0	0	335097	7.7500	NaN	Q
8	0	0	365235	7.7500	NaN	Q
9	0	0	C.A. 29566	10.5000	NaN	S

```
10      0     0          347086    7.7750   NaN        S
11      0     0        PC 17591   50.4958   B10        C
12      0     0          345774    9.5000   NaN        S
13      0     0            4138    9.8417   NaN        S
14      2     2      W./C. 6608   34.3750   NaN        S
15      0     0          349244    8.6833   NaN        S
16      0     0          235509   12.3500   NaN        Q
17      1     1            2650   15.2458   NaN        C
18      1     1          345773   24.1500   NaN        S
19      1     2      W./C. 6607   23.4500   NaN        S
20      1     0            2680   14.4542   NaN        C
21      0     0            2623    7.2250   NaN        C
22      0     0           65305    8.1125   NaN        S
23      1     0          345572   17.4000   NaN        S
24      2     0            2662   21.6792   NaN        C
25      0     0            4134    9.5875   NaN        S
26      0     0          347468    7.8542   NaN        S
27      0     0          237668   13.0000   NaN        S
28      1     0          350026   14.1083   NaN        S
29      1     0            4137    9.8250   NaN        S
```

Values replaced properly

[37]:
```
#Label the testing data set
x_input = testing[list(columns)].values
```

[38]:
```
#testing["Cabin"].fillna(testing["Cabin"].mean(), inplace=True)
target_labels = clf_train.predict(x_input)
target_labels = pd.DataFrame({'Est_Survival':target_labels, 'Name':
 ↪testing['Name']})
```

[39]:
```
import numpy as np
all_data = pd.read_csv("titanic_all.csv")
testing_results = pd.merge(target_labels, all_data[['Name','Survived']],␣
 ↪on=['Name'])
```

[40]:
```
acc = np.sum(testing_results['Est_Survival'] == testing_results['Survived']) /
 ↪float(len(testing_results))
```

[41]:
```
print(acc)
```

```
0.7588832487309645
```

[42]:
```
all_data = pd.read_csv("titanic_all.csv",␣
 ↪usecols=['Survived','Pclass','Gender','Age','SibSp','Fare'])
```

[43]:
```
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
```

So the model is 76.8% accurate

How many records are in the data set? **1308 records** Which important variables(s) are missing values and how many are missing? **Age and its missing 263 values**

[44]:
```
all_data["Gender"] = all_data["Gender"].apply(lambda toLabel:0 if toLabel ==␣
↪'male' else 1)
all_data["Age"].fillna(all_data["Age"].mean(), inplace=True)
all_data.head()
```

[44]:

	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

[46]:
```
 #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.
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)
```

[49]:
```
# Now Lets train the model to fit in the testing data

clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)
clf_Train = clf_train.fit(X_train,y_train)
```

[52]:
```
#Compare the models by scoring each

train_score = str(clf_train.score(X_train,y_train))
test_score = str(clf_train.score(X_test,y_test))

print('Training score = '+ train_score +' Testing score ='+ test_score)
```

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

Part 4 For Further Study

[53]:
```
trainAgeDropped = pd.read_csv("titanic_train.csv")
trainAgeDropped.dropna(subset=['Age'], inplace=True)
```

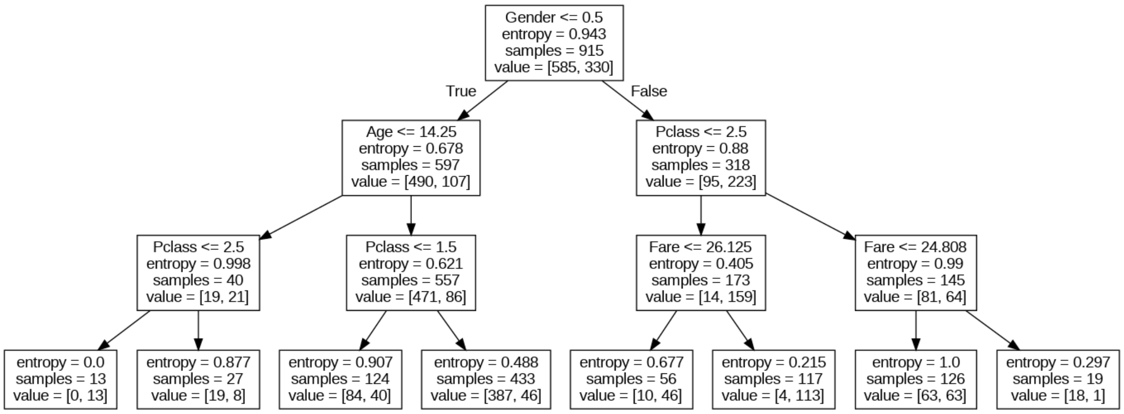
[55]:
```
trainAgeDropped.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 738 entries, 0 to 913
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  738 non-null    int64
 1   Survived     738 non-null    int64
 2   Pclass       738 non-null    int64
 3   Name         738 non-null    object
 4   Gender       738 non-null    object
 5   Age          738 non-null    float64
 6   SibSp        738 non-null    int64
 7   Parch        738 non-null    int64
 8   Ticket       738 non-null    object
 9   Fare         738 non-null    float64
 10  Cabin        187 non-null    object
 11  Embarked     737 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 75.0+ KB
```

[57]:
```
y_target1 = training["Survived"].values

columns=["Fare","Pclass","Gender","Age"] # REMOVED SPOUSE SIBLINGS INPUT
# the x variable for our y
x_input1 = training[list(columns)].values
clf_train1 = tree.DecisionTreeClassifier(criterion="entropy",max_depth=3)



clf_train1 = clf_train.fit(x_input1, y_target1)

clf_train1.score(x_input1,y_target1)
```

```
with open ("titanic2.dot",'w') as f:
    f = tree.export_graphviz(clf_train1,out_file=f,feature_names=columns)
!dot -Tpng titanic2.dot -o titanic2.png
Image("titanic2.png")
```

[57]:



Conclusion for This Actity

I've learned how to plot data, manipulate data frames, perform simple linear regression, clean data, train a decisiontree classifier, train such model, and apply it, score those models, and etc.

In short, I learned a lot, and this activity allowed me to be creative in how i approach on cleaning data set, selecting variables, and perform data analysis.