Project Titanic - Part II - [Your Name]

The goal is to predict whether or not a passenger survived based on attributes such as their age, sex, passenger class, where they embarked and so on.

1) Upload this jupyter notebook page to your colab

2) Get the Shareable link for your page and update the URL below for your Jupyter Notebook:

- *Make sure you select* ***'Anyone with the Link'*** *option*

This jupyter notebook page is located at: https://colab.research.google.com/drive/1UhO5nMZ4EU-nNHTXasGzenCRN-tkByq6?usp=sharing

We will click on the link above to visit to your Jupyter Notebook page.

3) Get the Data:

Download the data (train.csv and test.csv files) from Kaggle and then upload them using the first code block below.

- To download the files, login to Kaggle and go to the Titanic challenge

Keep the following code block as it is. Use it to upload the donwloaded csv files and to save them into your colab:

```
from google.colab import files
import pandas as pd
import io
import os

train_data_dict = files.upload() #uploads as a disctionary and creates a file
os.remove('train.csv') #remove the file created during upload that is in the root folder
train_data = pd.read_csv(io.StringIO(train_data_dict['train.csv'].decode('utf-8')),sep=',') #

test_data_dict = files.upload() #uploads as a disctionary and creates a file
os.remove('test.csv') #remove the file created during upload that is in the root folder
test_data = pd.read_csv(io.StringIO(test_data_dict['test.csv'].decode('utf-8')),sep=',') #get
```

```
titanic_dir_path = os.path.join("datasets", "titanic")
os.makedirs(titanic_dir_path, exist_ok=True) #create the folder
train_csv_path = os.path.join(titanic_dir_path, "train.csv") #create the path for the csv fil
test_csv_path = os.path.join(titanic_dir_path, "test.csv") #create the path for the csv file

train_data.to_csv(train_csv_path, index=False) #save the data to csv file
test_data.to_csv(test_csv_path, index=False) #save the data to csv file
```

Choose Files train.csv

- **train.csv**(application/vnd.ms-excel) - 61194 bytes, last modified: 10/9/2021 - 100% done

```
Saving train.csv to train.csv
```

Choose Files test.csv

- **test.csv**(application/vnd.ms-excel) - 28629 bytes, last modified: 10/9/2021 - 100% done

```
Saving test.csv to test (3).csv
```

Once you upload the data, they will be saved into the `datasets/titanic` directory. After uploading, you don't need to upload them again. You can start run your code starting the below code block.

```
import pandas as pd
import os

titanic_dir_path = os.path.join("datasets", "titanic")
train_csv_path = os.path.join(titanic_dir_path, "train.csv") #create the path for the csv fil
test_csv_path = os.path.join(titanic_dir_path, "test.csv") #create the path for the csv file

train_data = pd.read_csv(train_csv_path)
test_data = pd.read_csv(test_csv_path)
```

The questions under section 4 were in the Part I of the project. For this assignment, you need to answer the questions under section 5.

You can skip to 4.5

Discover, Visualize, Prepare Data:

4.1) Which attributes do we have, and what are they meaning? List the attributes and then briefly explain. To get the description of the attributes, you can do a little research on the web. No code is needed to answer this question.

4.2) Show your results and explain the insights you got by studying the data with each of the following methods on both the train and test data (Note: I am not looking for a long list of insights, 2-3 insights per method execution would be fine):

4.2.a. head()

4.2.b. info()

4.2.c. describe()

4.2.d. value_counts()

4.3) Prepare a DataFrame that contains the following numeric fields: Survided, Sex, Age, SibSp, Parch, Fare. Plot these numeric fields on a histogram. Did you notice anything new using the histogram?

4.4) Use groupby of Pandas to answer the following questions.

For the following examples, use group by and plot for example:

> dataFrame.groupby('attribute1')['attribute2'].median()

> dataFrame.groupby('attribute1')['attribute2'].median().plot(kind='bar')

4.4) a) Find the average survival rate based on passenger class and plot the results. What is the insight you gain?

4.4) b) Find the average survival rate based on sex and plot the results. What is the insight?

4.4) c) Find the median age by Pclass and Sex.

4.4) d) Find out the median fare based on passenger class and embarked place.

4.5 includes the code (solutions) needed to start section 5.

4.5) Work on missing values on the whole data set. Examples:

- https://towardsdatascience.com/machine-learning-with-the-titanic-dataset-7f6909e58280 PDF is attached to the assignment.

4.5) a) Perform the followings:

- 1) Create a new 'all_data' frame by appending test data to train data.
- 2) Using pandas methods see and show that some indexes repeat. Find a way to Use re-organize the index so that they are unique and do not have an extra 'index' column.
- 3) Then check the data using the info() method and list which columns have missing data (other than 'Survived')

```
all_data = train_data.append(test_data) # important
all_data.reset_index(inplace = True, drop = True) # reset the inde inplace and drop the creat
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   object 
 5   Age          1046 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object 
 11  Embarked     1307 non-null   object 
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

4.5) b) Fill missing values of 'Age' field with the median age of the passenger class and sex that you found for the question above. Use the apply method with lambda function.

```
# Fill in missing age information based on the median age for its class and sex.
all_data['Age'] = all_data.groupby(['Pclass','Sex'])['Age'].apply(lambda x : x.fillna(x.media
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        295 non-null    object
 11  Embarked     1307 non-null   object
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

4.5) d) Fill missing values of 'Cabin' field with the 'NA' value.

```
all_data.Cabin = all_data.Cabin.fillna('NA')
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64
 3   Name         1309 non-null   object
 4   Sex          1309 non-null   object
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   int64
 8   Ticket       1309 non-null   object
 9   Fare         1308 non-null   float64
 10  Cabin        1309 non-null   object
 11  Embarked     1307 non-null   object
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

4.5) e) Fill missing values of 'Embarked' field with the most frequently seen 'Embarked' value.

```
# Fill in missing Embarked based on the most frequent Embarked
all_data['Embarked'].fillna(all_data['Embarked'].mode()[0], inplace = True)
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   object 
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1308 non-null   float64
 10  Cabin        1309 non-null   object 
 11  Embarked     1309 non-null   object 
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

4.5) c) Fill missing values of 'Fare' field with the median fare of the passenger class and embarked location that you found for the question above. Use the apply method with lambda function.

```
# Fill in missing fare based on its class and embarked place
all_data['Fare'] = all_data.groupby(['Pclass','Embarked'])['Fare'].apply(lambda x : x.fillna(
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  1309 non-null   int64  
 1   Survived     891 non-null    float64
 2   Pclass       1309 non-null   int64  
 3   Name         1309 non-null   object 
 4   Sex          1309 non-null   object 
 5   Age          1309 non-null   float64
 6   SibSp        1309 non-null   int64  
 7   Parch        1309 non-null   int64  
 8   Ticket       1309 non-null   object 
 9   Fare         1309 non-null   float64
 10  Cabin        1309 non-null   object 
 11  Embarked     1309 non-null   object 
dtypes: float64(3), int64(4), object(5)
memory usage: 122.8+ KB
```

Next Questions are **for Part II**.

Feature Engineering

5) 1) Create a new feature 'Family_Size'

- Create a new feature 'Family_Size' using other features (and also adding the person him/herself to the family size).
- Then plot a bar chart to show how many of each 'Family_Size' value exists.
- Finally plot a bar chart to show the relationship between 'Family_Size' and the 'Survival'

```
all_data["Family_Size"] = all_data["SibSp"] + all_data["Parch"] + 1
```

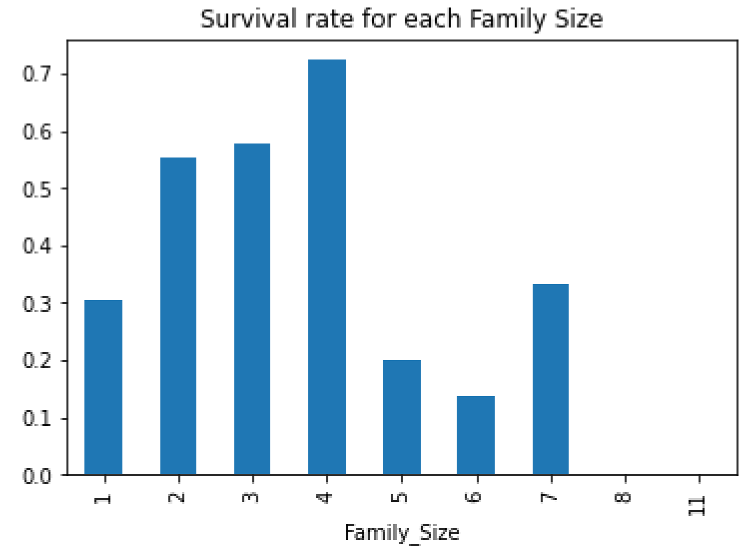
```
all_data["Family_Size"].value_counts()
```

```
1     790
2     235
3     159
4      43
6      25
5      22
7      16
11     11
8       8
Name: Family_Size, dtype: int64
```

```
all_data.Family_Size.value_counts().plot(kind='bar', title = "Family Size on Ship")
```

```
all_data[["Family_Size", "Survived"]].groupby("Family_Size")["Survived"].mean().plot(kind="ba
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff441c12c10>
```



5) 2) Create a new feature 'Fare_Category'

- Use qcut method of Pandas for creating 'Fare_Category' field from Fare so that we have 5 categories of Fare. Note that: 1) With qcut We decompose a distribution so that there are (approximately) the same number of cases ineach category. 2) qcut returns categorical data and we need to convert it to string using astype(str). Otherwise one-hot-encoder question below might have issues.
- Use value_counts() method to show the results.
- Plot a bar chart to show the relationship between 'Fare_Category' and the 'Survival'

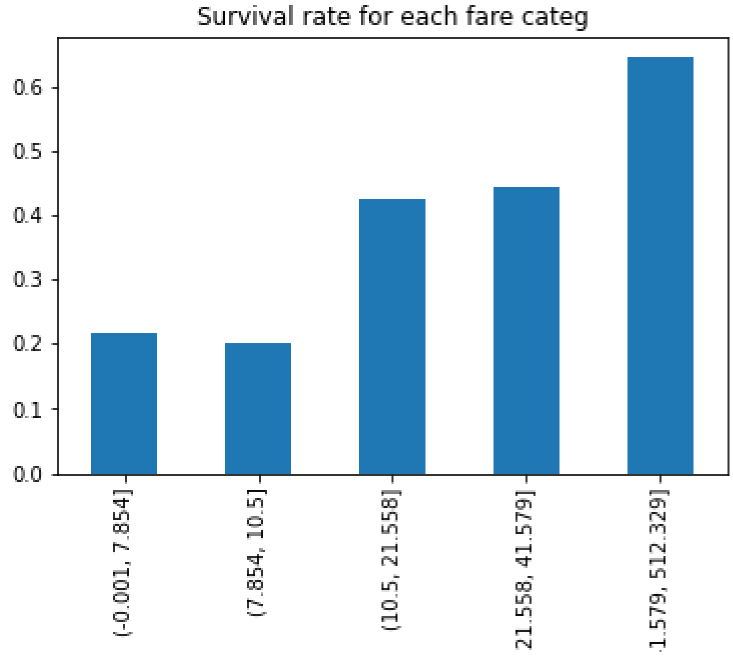
```
all_data["Fare_Category"] = pd.qcut(all_data["Fare"], q = 5)
```

```
all_data["Fare_Category"].value_counts()
```

```
(-0.001, 7.854]      275
(21.558, 41.579]     265
(41.579, 512.329]    259
(7.854, 10.5]        255
(10.5, 21.558]       255
Name: Fare_Category, dtype: int64
```

```
all_data[["Fare_Category", "Survived"]].groupby("Fare_Category")["Survived"].mean().plot(kind
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff4315ce890>
```



5) 3) Create a new feature 'Age_Category'

- Use cut method of Pandas for creating 'Age_Category' field from Age so that we have 5 categories of Age. Note that: 1) With cut, the bins are formed based on the values of the variable, regardless of how many cases fall into a category. 2) cut returns categorical data and we need to convert it to string using astype(str). Otherwise one-hot-encoder question below might have issues.
- Use value_counts() method to show the results.
- Plot a bar chart to show the relationship between 'Age_Category' and the 'Survival'

```
all_data['Age_Category'] = pd.cut(all_data['Age'], 5)
```

```
all_data["Age_Category"]= all_data["Age_Category"].astype("str")
```

```
(16.136, 32.102]    748
(32.102, 48.068]    308
(0.0902, 16.136]    134
(48.068, 64.034]    106
(64.034, 80.0]       13
Name: Age_Category, dtype: int64
```

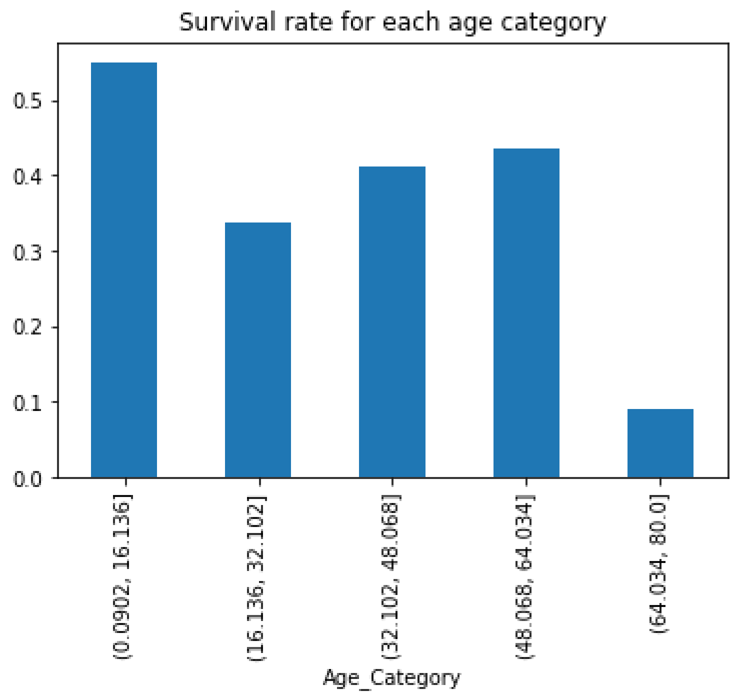
```
all_data["Age_Category"].value_counts()
```

```
(16.136, 32.102]    748
(32.102, 48.068]    308
(0.0902, 16.136]    134
(48.068, 64.034]    106
```

```
(64.034, 80.0]       13
Name: Age_Category, dtype: int64
```

```
all_data[["Age_Category", "Survived"]].groupby("Age_Category")["Survived"].mean().plot(kind="
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff431382d90>
```



6) Encoders

6) 1) Using LabelEncoder, create the 'Sex_Numeric' based on the values of the 'Sex' attribute.

```
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()

all_data["Sex_Numeric"]  = labelencoder.fit_transform(all_data["Sex"])
all_data
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599
2	3	1.0	3	Heikkinen, Miss	female	26.0	0	0	STON/O2.

6) 2) Use OneHotEncoder to create new attributes for the 'Embarked' attribute.

Note: You can benefit from the following article for One-Hot-Encoding questions:

- https://towardsdatascience.com/machine-learning-with-the-titanic-dataset-7f6909e58280

(Lily May

```
from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder(sparse = False,
                    handle_unknown="error",
                    drop="first")

enc_df = pd.DataFrame(enc.fit_transform(all_data[["Embarked"]]))

enc_df
```

6) 3) Use OneHotEncoder to create new attributes for the 'Fare_Category' attribute.

```
enc = OneHotEncoder(sparse = False,
                    handle_unknown="error",
                    drop="first")

enc_df1 = pd.DataFrame(enc.fit_transform(all_data[["Fare_Category"]]))

enc_df1
```

	0	1	2	3
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	1.0
2	1.0	0.0	0.0	0.0
3	0.0	0.0	0.0	1.0
4	1.0	0.0	0.0	0.0
...	...	...	...	...
1304	1.0	0.0	0.0	0.0
1305	0.0	0.0	0.0	1.0
1306	0.0	0.0	0.0	0.0
1307	1.0	0.0	0.0	0.0
1308	0.0	0.0	1.0	0.0

1309 rows × 4 columns

6) 4) Use OneHotEncoder to create new attributes for the 'Age_Category' attribute.

```
enc = OneHotEncoder(sparse = False,
                    handle_unknown="error",
                    drop="first")

enc_df2 = pd.DataFrame(enc.fit_transform(all_data[["Age_Category"]]))

enc_df2
```

	0	1	2	3
0	1.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0
3	0.0	1.0	0.0	0.0
4	0.0	1.0	0.0	0.0
...	...	...	...	...
1304	1.0	0.0	0.0	0.0
1305	0.0	1.0	0.0	0.0
1306	0.0	1.0	0.0	0.0
1307	1.0	0.0	0.0	0.0

1309 rows × 4 columns

6) 5) Use OneHotEncoder to create new attributes for the 'PClass' attribute.

```
enc = OneHotEncoder(sparse = False,
                    handle_unknown="error",
                    drop="first")

enc_df3 = pd.DataFrame(enc.fit_transform(all_data[["Pclass"]]))

enc_df3
```

	0	1
0	0.0	1.0
3	0.0	0.0
...	...	...
1305	0.0	0.0
1308	0.0	1.0

6) 6) Convert 'Sex_Numeric' and 'Family_Size' fields to 'float16'.

```
all_data["Sex_Numeric"]= all_data["Sex_Numeric"].astype("float16")
```

```
all_data["Family_Size"]=all_data["Sex_Numeric"].astype("float16")
```

7) 1) Create the correlation matrix for the all_data data frame and show the values for 'Survived' column in an descending order.

```
matrix = all_data.corr()
matrix.sort_values(by = "Survived", ascending=False )
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
Survived	-0.005007	1.000000	-0.338481	-0.058635	-0.035322	0.081629	0.257307
Fare	0.031029	0.257307	-0.558740	0.198711	0.160388	0.221668	1.000000
Parch	0.008942	0.081629	0.018322	-0.134239	0.373587	1.000000	0.221668
PassengerId	1.000000	-0.005007	-0.038354	0.020478	-0.055224	0.008942	0.031029
SibSp	-0.055224	-0.035322	0.060832	-0.204025	1.000000	0.373587	0.160388
Age	0.020478	-0.058635	-0.451983	1.000000	-0.204025	-0.134239	0.198711
Pclass	-0.038354	-0.338481	1.000000	-0.451983	0.060832	0.018322	-0.558740
Family_Size	0.013406	-0.543351	0.124617	0.074529	-0.109609	-0.213125	-0.185744
Sex_Numeric	0.013406	-0.543351	0.124617	0.074529	-0.109609	-0.213125	-0.185744

7) 2) Based on the correlation matrix results, identify some of the features as unimportant and drop them and assign the remaining DataFrame to the variable named 'important_data'. When you drop features, leave at least 10 columns besides 'Survivided' in the 'important_data' DataFrame. After that, check the correlation to 'Survived' as you did before.

```
important_data = all_data.drop(columns=["Cabin", "Embarked", "Parch"])
```

```
important_data
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Ticket	Far
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	A/5 21171	7.250
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	PC 17599	71.283
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	STON/O2. 3101282	7.925
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	113803	53.100

8) 1) Create X_train, Y_train and X_test DataFrames. Note that X_train should have 891 instances and the rest should go to X_test. Drop the 'Survived' from X_test. Check the X_train, X_test and Y_train.

```
X_train = train_data
```

```
Y_train = train_data["Survived"]
```

```
X_test = important_data.drop(labels=["Survived"], axis = 1)
```

```
X_train
```

	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,						

```
Y_train
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

```
X_test
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
			Braund,						
1	2	1	Bradley	female	38.0	1	0	PC 17599	71.2833
			Mrs.						

8) 2) Use StandardScaler of Scikit Learn to scale the 'Fare' feature of both X_train and X_test.

```
from sklearn.preprocessing import StandardScaler

scaler_sd = StandardScaler()

X_train = scaler_sd.fit_transform(train_data[["Fare"]])

X_test =  scaler_sd.fit_transform(train_data[["Fare"]])
```

9) Run all of your code and get your output

10) Print the latest status of your notebook to a pdf file

- The pdf file **must include the link of your jupyter notebook page** (see step 2 above)

11) **Submit the PDF** file on Canvas

0s completed at 10:02 PM