Project 1 - Titanic - Part I - [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
Jupyter Notebook:
- _Make sure you select __'Anyone with the Lin

__This jupyter notebook page is located at__:h
com/drive/1onjY0Zg6mqDNVjsMvdy2vSeoTIZXpfG0?us

We will click on the link above to visit to yo
```

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/1onjY0Zg6mqDNVjsMvdy2vSeoTIZXpfG0?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.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)
```

Answer the Questions Below

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.

- Survival - Defines who survived with 0 representing not surviving and 1 as survived
- pclass - The ticket class of a passenger with 1 represent 1st class, 2 representing 2nd class and 3 representing 3rd class
- Sex - Represents the gender of passenger
- Age - Represents the age in years of passenger
- sibsp - amount of siblings / spouses aboard the Titanic
- parch - amount of parents / children aboard the Titanic also Some children travelled only with a nanny, therefore parch=0 for them.
- ticket - Ticket number
- fare - Passenger fare
- cabin - Cabin number
- embarked - Port of Embarkation represented through C = Cherbourg, Q = Queenstown, S = Southampton
- Passengerid = the id of passenger which is an identifier for wach passenger
- Name - The passnengers name

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()

From the head of the train and test data we can see that the out of the first 5 passengers 4 of them embarked from southampton. Also that the cabin values are not displayed correctly when using the head function on the train data. The disparity between different cabin class fare was also an interesting observation.

```
print(train_data.head())
```

```
   PassengerId  Survived  Pclass  ...     Fare Cabin  Embarked
0            1         0       3  ...   7.2500   NaN         S
1            2         1       1  ...  71.2833   C85         C
2            3         1       3  ...   7.9250   NaN         S
```

```
3            4         1       1  ...  53.1000  C123         S
4            5         0       3  ...   8.0500   NaN         S

[5 rows x 12 columns]
```

```
print(test_data.head())
```

```
   PassengerId  Pclass  ... Cabin Embarked
0          892       3  ...   NaN        Q
1          893       3  ...   NaN        S
2          894       2  ...   NaN        Q
3          895       3  ...   NaN        S
4          896       3  ...   NaN        S

[5 rows x 11 columns]
```

4.2.b. info()

```
train_data.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
```

```
test_data.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
```

We can see from looking at the info from the test and training sets the different data types that are used under each column and the total amount of each data type udner dtypes. We can also the total memory usage of the test and training data. With the training data having a memory usage of 83.7+ KB.

4.2.c. describe()

```
train_data.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
test_data.describe()
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare
count	418.000000	418.000000	332.000000	418.000000	418.000000	417.000000

We can se that the average fare cost fro the training data was around 32.2 pounds while the test data shows an average fare price of 35.63 pounds. ALso that 38% survived the crash in the training data while in the test data it was 39% survival rate. Also that the average age for the training data was less than 30 and the same for the test set.

50%	1100.500000	3.000000	27.000000	0.000000	0.000000	14.454200

4.2.d. value_counts()

max	1309.000000	3.000000	76.000000	8.000000	9.000000	512.329200

```
test_data.value_counts()
```

```
PassengerId  Pclass  Name                                              Sex     Age
1306         1       Oliva y Ocana, Dona. Fermina                      female  39.0
1034         1       Ryerson, Mr. Arthur Larned                        male    61.0
992          1       Stengel, Mrs. Charles Emil Henry (Annie May Morris) female  43.0
1001         2       Swane, Mr. George                                 male    18.5
1004         1       Evans, Miss. Edith Corse                          female  36.0

1198         1       Allison, Mr. Hudson Joshua Creighton              male    30.0
1200         1       Hays, Mr. Charles Melville                        male    55.0
1206         1       White, Mrs. John Stuart (Ella Holmes)             female  55.0
1208         1       Spencer, Mr. William Augustus                     male    57.0
904          1       Snyder, Mrs. John Pillsbury (Nelle Stevenson)     female  23.0
Length: 87, dtype: int64
```

```
train_data.value_counts()
```

```
PassengerId  Survived  Pclass  Name                                                 Se
890          1         1       Behr, Mr. Karl Howell                                ma
337          0         1       Pears, Mr. Thomas Clinton                            ma
332          0         1       Partner, Mr. Austen                                  ma
330          1         1       Hippach, Miss. Jean Gertrude                         fe
328          1         2       Ball, Mrs. (Ada E Hall)                              fe

584          0         1       Ross, Mr. John Hugo                                  ma
582          1         1       Thayer, Mrs. John Borland (Marian Longstreth Morris) fe
578          1         1       Silvey, Mrs. William Baird (Alice Munger)            fe
573          1         1       Flynn, Mr. John Irwin ("Irving")                     ma
2            1         1       Cumings, Mrs. John Bradley (Florence Briggs Thayer)  fe
Length: 183, dtype: int64
```

We can that the test data and training data have difference in columns displayed after running value counts with the discrepancies being the survived column. We can also see that their was a higher rate of sibling in the test data in comparison to the training data with the test data having a total value of 6 SibSP counts while in the training data it was 4 for toal value.

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?

```
import pandas as pd
import numpy as np

df_numerics = train_data.select_dtypes(include=np.number)
df_numerics
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
0	1	0	3	22.0	1	0	7.2500
1	2	1	1	38.0	1	0	71.2833
2	3	1	3	26.0	0	0	7.9250
3	4	1	1	35.0	1	0	53.1000
4	5	0	3	35.0	0	0	8.0500
...	...	...	...	...	...	...	...
886	887	0	2	27.0	0	0	13.0000
887	888	1	1	19.0	0	0	30.0000
888	889	0	3	NaN	1	2	23.4500
889	890	1	1	26.0	0	0	30.0000
890	891	0	3	32.0	0	0	7.7500

891 rows × 7 columns

Double-click (or enter) to edit

Double-click (or enter) to edit

4.4) Use groupby of Pandas to explain the following questions. Study the examples listed on the following webapage about groupby and plot functions. Note that **PDFs of these webpages are attached** to the assignment:

- https://towardsdatascience.com/pandas-groupby-explained-453692519d0
- https://medium.com/@sciencelee/making-plots-with-the-pandas-groupby-ac492941af28

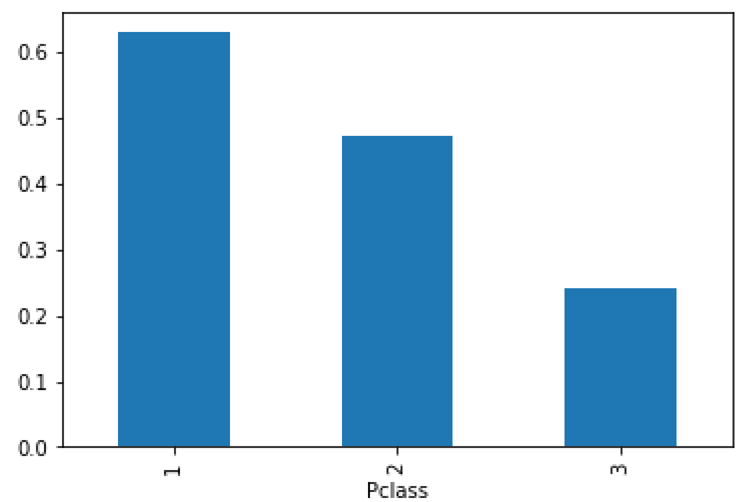
For the following examples, use group by and plot for example:

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

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

```
survived_by_class = train_data.groupby('Pclass')['Survived'].mean().plot(kind='bar')
survived_by_class
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d1b48efd0>
```



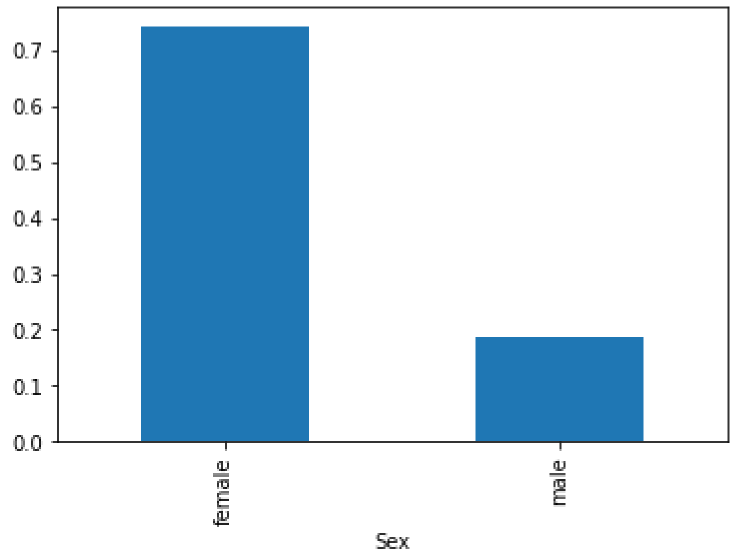
We can that the people in the 1st class cabin had an average survial rate of above 60% with the 2nd class cabins having around 45% and the 3rd class cabin having about 23% survival rate. This shows that the passengers that spent more had a higher rate of survival after the crash.

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

```
survived_by_sex = train_data.groupby('Sex')['Survived'].mean().plot(kind='bar')
```

```
survived_by_sex
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6d1b068450>
```



The insight is that about 74% of the males survived while for the males it was about 17% survived after the crash.

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

```
average_age = train_data.groupby(['Pclass','Sex'])['Age'].mean()
average_age
```

```
Pclass  Sex
1       female    34.611765
        male      41.281386
2       female    28.722973
        male      30.740707
3       female    21.750000
        male      26.507589
Name: Age, dtype: float64
```

- Shows that for cabin class 1 the median age was 34 years for female and 41 for male
- Shows that for cabin class 2 the median age was 28 years for female and 30 for male
- Shows that for cabin class 3 the median age was 21 years for female and 26 for male

---

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

```
average_fare = train_data.groupby(['Pclass','Embarked'])['Fare'].mean()
average_fare
```

```
Pclass  Embarked
1       C           104.718529
        Q            90.000000
        S            70.364862
2       C            25.358335
        Q            12.350000
        S            20.327439
3       C            11.214083
        Q            11.183393
        S            14.644083
Name: Fare, dtype: float64
```

Double-click (or enter) to edit

4.5) We will work on missing values on the whole data set. You can benefit from the following article for some of the questions below:

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

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')

Double-click (or enter) to edit

```
concat_df = [train_data, test_data]
all_data = pd.concat(concat_df)
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. Laina	female	26.0	0	0	STON/O2. 3101282
3	4	1.0	1	Futrelle, Mrs. Jacques	female	35.0	1	0	113803

```
all_data.Cabin.duplicated()
```

```
0      False
1      False
2       True
3      False
4       True
       ...
413     True
414    False
415     True
416     True
417     True
Name: Cabin, Length: 1309, dtype: bool
```

```
all_data.Fare.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
413     True
414     True
415     True
416     True
417     True
Name: Fare, Length: 1309, dtype: bool
```

```
all_data.Embarked.duplicated()
```

```
0      False
1      False
```

```
2       True
3       True
4       True
       ...
413     True
414     True
415     True
416     True
417     True
Name: Embarked, Length: 1309, dtype: bool
```

```
all_data.Cabin.duplicated()
```

```
0      False
1      False
2       True
3      False
4       True
       ...
413     True
414    False
415     True
416     True
417     True
Name: Cabin, Length: 1309, dtype: bool
```

```
all_data.isnull()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	False	False	False	False	False	False	False	False	False	False	Tru
1	False	False	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	False	Tru
3	False	False	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	False	Tru
...	...	...	...	...	...	...	...	...	...	...	
413	False	True	False	False	False	True	False	False	False	False	Tru
414	False	True	False	False	False	False	False	False	False	False	Fals
415	False	True	False	False	False	False	False	False	False	False	Tru
416	False	True	False	False	False	True	False	False	False	False	Tru
417	False	True	False	False	False	True	False	False	False	False	Tru

1309 rows × 12 columns

```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1309 entries, 0 to 417
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: 132.9+ KB
```

Double-click (or enter) to edit

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.

```
all_data['Age']=train_data.groupby(['Pclass','Sex'])['Age'].apply(lambda·x:·x.fillna(x.median
```

```
all_data.isnull()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi

Double-click (or enter) to edit

Double-click (or enter) to edit

3	False	False	False	False	False	False	False	False	False	False	Fals

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.

414	False	True	False	False	False	False	False	False	False	False	Fals

```
all_data['Fare']=train_data.groupby(['Pclass','Embarked'])['Fare'].apply(lambda x: x.fillna(x
all_data
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
0	1	0.0	3	Braund, Mr. Owen Harris	male	7.2500	1	0	A/5 21171
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	71.2833	1	0	PC 17599
2	3	1.0	3	Heikkinen, Miss. Laina	female	7.9250	0	0	STON/O2. 3101282
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	53.1000	1	0	113803
				Allen, Mr					

Double-click (or enter) to edit

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

```
all_data['Cabin']=
```

Double-click (or enter) to edit

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

```
all_data·=·all_data.fillna(all_data['Embarked'].value_counts().index[0])
```

5) Run all of your code and get your output

6) 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)

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

Next Questions **will be in Part II**. In case you want to head start, you can start working on the following questions:

Feature Engineering

8) 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'

Double-click (or enter) to edit

8) 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'

Double-click (or enter) to edit

8) 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'

Double-click (or enter) to edit

9) Encoders

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

Double-click (or enter) to edit

9) 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

Double-click (or enter) to edit

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

Double-click (or enter) to edit

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

Double-click (or enter) to edit

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

Double-click (or enter) to edit

✓ 0s completed at 11:55 PM