Task – Titanic dataset

1- : import the Libraries:

Import pandas as pd

Import numpy as np

Import matplotlib.pyplot as plt

Import seaborn as sns

Import string

2- : Getting the data:

data=pd.read_csv("train.csv") I supposed the data is saved on my desktop.

Then, I want to see the list of features and description the data:

data.describe(include="all") . all : to see all the value included nan values.

```
       PassengerId    Survived      Pclass  ...        Fare        Cabin  Embarked
count   891.000000  891.000000  891.000000  ...  891.000000          204       889
unique         NaN         NaN         NaN  ...         NaN          147         3
top            NaN         NaN         NaN  ...         NaN  C23 C25 C27         S
freq           NaN         NaN         NaN  ...         NaN            4       644
mean    446.000000    0.383838    2.308642  ...   32.204208          NaN       NaN
std     257.353842    0.486592    0.836071  ...   49.693429          NaN       NaN
min       1.000000    0.000000    1.000000  ...    0.000000          NaN       NaN
25%     223.500000    0.000000    2.000000  ...    7.910400          NaN       NaN
50%     446.000000    0.000000    3.000000  ...   14.454200          NaN       NaN
75%     668.500000    1.000000    3.000000  ...   31.000000          NaN       NaN
max     891.000000    1.000000    3.000000  ...  512.329200          NaN       NaN

[11 rows x 12 columns]
```

data.columns.values

```
['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
 'Ticket' 'Fare' 'Cabin' 'Embarked']
```

The training data has 891 passengers , 11 features and 1 target columns .

- Then, I will check the missing data in our dataset:
  data.isnull().sum().sort_values(ascending=False)

if we take a look to the list .177 values are missing from Age column, it's about 20% missing data , also, the Cabin feature has 687 values are missing from his data, it's approximately 77% of its values,

and finally, the Embarked feature is missing about 0.22%.

So, we Should Visualize the data to see the effect each column but before let's see more details about our training data.

```
Cabin          687
Age            177
Embarked         2
Fare             0
Ticket           0
Parch            0
SibSp            0
Sex              0
Name             0
Pclass           0
Survived         0
PassengerId      0
```

:

data[“Sex”].value_counts() # The number of males and females
data[“Sex”].value_counts(normalize=True)*100 # the rate of male and females in all dataset.

Output : male : 577   , 64.758698 %

Female : 314   , 35.241302 %

3- Analysis the data:

I will calculate the percentage of gender according to each feature:

- **The percentage of gender in Survived column:**
  - Passengers who are not survived:

data["Sex"][data["Survived"]==0].value_counts(normalize=True)*100

output: male :85.245902 %

female: 14.754098 %

- Passengers who are survived:

data["Sex"][data["Survived"]==1].value_counts(normalize=True)*100

output: male: 68.128655 %

female: 31.871345 %

- **The percentage of gender in Pclass feature:**

- Class 1: data["Sex"][data["Pclass"]==1].value_counts(normalize=True)*100

Output: male: 56.481481 %

Female: 43.518519 %

- Class 2: data["Sex"][data["Pclass"]==2].value_counts(normalize=True)*100

Output: male:58.695652 %

Female: 41.304348 %

- Class 3: data["Sex"][data["Pclass"]==3].value_counts(normalize=True)*100

Output: male: 70.672098 %

Female: 29.327902 %

- **The percentage of gender in SibSp feature:**

in this feature I will use the for loop to calculate the percentage of gender according to how many siblings and spouse. With unique and sorted methods.

```
for i in sorted(data.SibSp.unique()):
    data["Sex"][data["SibSp"]==i].value_counts(normalize=True)*100
```

```
The Percentage of Gender with number of Sibling and Spouse = 0
male      71.381579
female    28.618421
Name: Sex, dtype: float64
The Percentage of Gender with number of Sibling and Spouse = 1
female    50.717703
male      49.282297
Name: Sex, dtype: float64
The Percentage of Gender with number of Sibling and Spouse = 2
male      53.571429
female    46.428571
Name: Sex, dtype: float64
The Percentage of Gender with number of Sibling and Spouse = 3
female    68.75
male      31.25
Name: Sex, dtype: float64
The Percentage of Gender with number of Sibling and Spouse = 4
male      66.666667
female    33.333333
Name: Sex, dtype: float64
The Percentage of Gender with number of Sibling and Spouse = 5
male      80.0
female    20.0
Name: Sex, dtype: float64
The Percentage of Gender with number of Sibling and Spouse = 8
male      57.142857
female    42.857143
Name: Sex, dtype: float64
```

- **The percentage of gender in Parch feature:**

The same way to calculate the genders according to number the parents and children.

```
for i in sorted (data.Parch.unique()):
        data["Sex"][data["Parch"]==i].value_counts(normalize=True)*100
```

```
The Percentage of Gender with number of Parent and Child= 0
male      71.386431
female    28.613569
Name: Sex, dtype: float64
The Percentage of Gender with number of Parent and Child= 1
female    50.847458
male      49.152542
Name: Sex, dtype: float64
The Percentage of Gender with number of Parent and Child= 2
female    61.25
male      38.75
Name: Sex, dtype: float64
The Percentage of Gender with number of Parent and Child= 3
female    80.0
male      20.0
Name: Sex, dtype: float64
The Percentage of Gender with number of Parent and Child= 4
male      50.0
female    50.0
Name: Sex, dtype: float64
The Percentage of Gender with number of Parent and Child= 5
female    80.0
male      20.0
Name: Sex, dtype: float64
The Percentage of Gender with number of Parent and Child= 6
female    100.0
```

- **The percentage of gender in Age feature:**

In Age feature we have 177 missing data we have to fill in. and to plot age feature, I divided the data in age column to 2 categories.(0-18) and (18-80).

```
mean = data["Age"].mean()

std = data["Age"].std()

is_null = data["Age"].isnull().sum()

np.random.seed(42)

# compute random numbers between the mean, std and is_null

range_age=np.random.randint(mean - std, mean + std, size = is_null)

data=data.fillna({"Age":np.random.randint(np.min(range_age),np.max(range_age))})
```

category=pd.cut(data["Age"],[0,18,80],labels=["child(0-18)","Adult(18-80)"])

data.insert(5,"Age Group",category)

Now, we have new column called “Age Group” with object type.

To come back to calculate the percentage of gender according to the age:

- The percentage of gender in age (0-18):

data["Sex"][data["Age Group"]=="child(0-18)"].value_counts(normalize=True)*100

output: male: 51.079137 %

female: 48.920863 %

- The percentage of gender in age (18-80):

data["Sex"][data["Age Group"]=="Adult(18-80)"].value_counts(normalize=True)*100)

output: male:67.287234 %

female: 32.712766 %

alse, we can the number of both gender in each category:

res=data.groupby(["age"])["Sex"].count()

(0-18) 139 passengers

(18-80) 752 passengers

- **The percentage of gender in Embarked feature:**

To fill in the missing data in Embarked feature, I should calculate the number of passengers who embarked from each port:

Southampton ///data[data["Embarked"]=="S"].shape[0]) #644

Cherbourg /// data[data["Embarked"]=="C"].shape[0]) # 168

Queenstown ///data[data["Embarked"]=="Q"].shape[0]) #77

I found the number of passengers who embarked from Southampton greater than others so I will the 2 missing data by S.

data=data.fillna({"Embarked":"S"})

to come back to the percentage of gender of passengers who embarked from each port:

- Southampton port:

data["Sex"][data["Embarked"]=="S"].value_counts(normalize=True)*100 output: male: 68.266254 %

female:31.733746 %

- Cherbourg port:

data["Sex"][data["Embarked"]=="C"].value_counts(normalize=True)*100

output: male: 56.547619 %

female:43.452381 %

- Queenstown port:

data["Sex"][data["Embarked"]=="Q"].value_counts(normalize=True)*100

output: male: 53.246753 %

female: 46.753247 %

- **The percentage of gender in Fare feature:**

As Age feature I should divide the data to many categories to plot it .So I will use cut method in pandas package and insert a new column called "Fare Group".

```
fare=pd.cut(data["Fare"],[-1,100,200,300,400,513],labels=["0-100","100-200","200-300","300-400","400-513"])
```

```
data.insert(5,"Fare Group",fare)
```

then, I will calculate the percentage in each category:

data["Sex"][data["Fare Group"]=="0-100"].value_counts(normalize=True)*100

male 66.587112 %

female 33.412888 %

data["Sex"][data["Fare Group"]=="100-200"].value_counts(normalize=True)*100

female 66.666667 %

male 33.333333 %

data["Sex"][data["Fare Group"]=="200-300"].value_counts(normalize=True)*100

female 64.705882 %

male 35.294118 %

data["Sex"][data["Fare Group"]=="300-400"].value_counts(normalize=True)*100

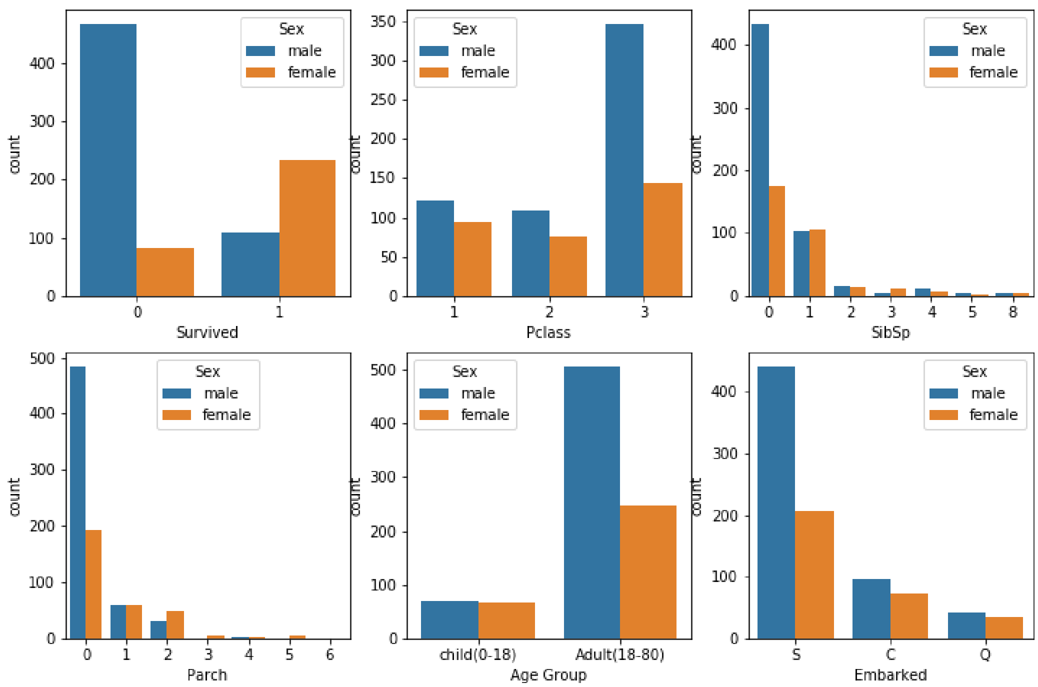
0 %

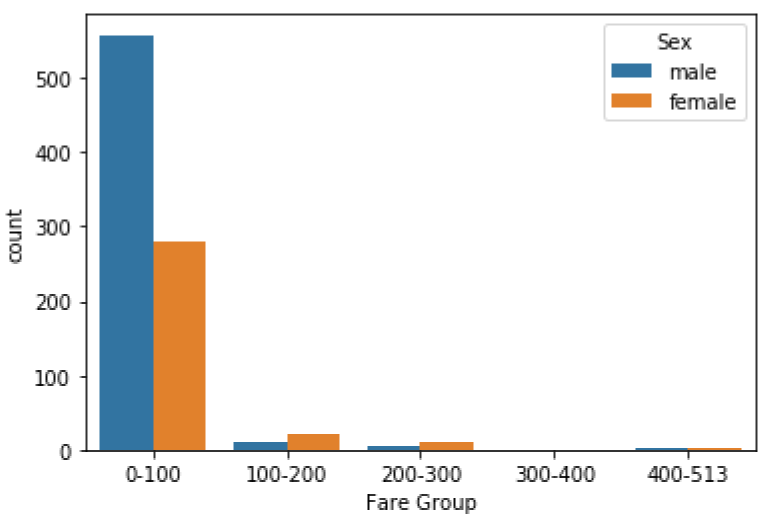
data["Sex"][data["Fare Group"]=="400-513"].value_counts(normalize=True)*100

male 66.666667 %

female 33.333333 %

4- Visualize and analysis the data :





Let me explain what i see in the figure:

1- The number of males who are died in the accident greater than females's so, females have a much higher chance of survival than Males so the Survived feature is essential in our Predictions.
2-the number of males with lower socioeconomic class greater than females'. It's 70.67% for males compared with 29.32% for females in third class so the socioeconomic is important.
3- Without Siblings and Spouse, the number of Males also greater than Females' so that 71.381% for males and 28.618% for females which don't have siblings and Spouse with them in the ship.
4-Without Parents or Children also Males are more numerous than Females so that 71.38 for males and 28.61% for females.
5-The Adult Males are more numerous than Adults Females the rate 66.63% Males / 33.56% Females.
6-It's clear Finally the Males who start the trip from Southampton more numerous than the females who start from this port. it's 68.47% for Males / 31.52% for Females.
7-Acording to the price of the ticket: actually, it will be good for classification Genders because almost there is a difference between the rate of males and females in each category about 60% to 30%.

5- Remove columns:

i will drop the Cabin and ticket because i have about 77% values missing

from Cabin and each ticket has a different number so it will not help us in classification.

also, i will drop the PassengerId because it's a serial number so it does not contribute to Gender Prediction.

```
data=data.drop(["PassengerId","Cabin","Ticket","Age Group","Fare Group"],axis=1)
```

6- Preparing the string in our data:

We have a Name column in our training data, string type so I will try to prepare the data before converting it to numerical values.

- i will remove the punctuation without using nltk library and without removing stopwords because these are names of people.
- convert the male and female to numerical values because the machine learning algorithms deal just with numerical values.

- Convert Embarked and Name data to numerical values by using LabelEncoder method.

7- Training the data:

When I put data.info(), I found some columns with object and int32 so I convert all data to be same type (int64)

data["Age"]=data["Age"].apply(np.int64)

data["Name"]=data["Name"].apply(np.int64)

data["Fare"]=data["Fare"].apply(np.int64)

data["Embarked"]=data["Embarked"].apply(np.int64)

```
Survived    891 non-null int64
Pclass      891 non-null int64
Name        891 non-null int64
Sex         891 non-null int64
Age         891 non-null int64
SibSp       891 non-null int64
Parch       891 non-null int64
Fare        891 non-null int64
Embarked    891 non-null int64
dtypes: int64(9)
memory usage: 62.8 KB
None
```

In training data, I will use 5 classification machine learning algorithms and compare between them, then implement K cross Validation:

First: we can predict the same training data but it will not give us a real accuracy because the algorithm already was training with these data so i will split the data to training and testing data.

x_train,x_test,y_train,y_test =train_test_split(X,Y,test_size=0.3,random_state=42)

- Naive Bayes algorithm :

```
Accuracy_Naive_bayes: 71.64179104477611
Classification_report_Naive_bayes:
               precision    recall  f1-score   support

           0       0.59      0.63      0.61        95
           1       0.79      0.76      0.78       173

    accuracy                           0.72       268
   macro avg       0.69      0.70      0.69       268
weighted avg       0.72      0.72      0.72       268

Confusing Matrix_Naive bayes :
 [[ 60  35]
 [ 41 132]]
```

- Support Vector Machine:

```
Accuracy_SVM : 79.1044776119403
Classification_report_SVM:
              precision    recall  f1-score   support

           0       0.78      0.70      0.74       113
           1       0.80      0.86      0.83       155

    accuracy                           0.79       268
   macro avg       0.79      0.78      0.78       268
weighted avg       0.79      0.79      0.79       268

Confusing Matrix_SVM :
 [[ 79  34]
 [ 22 133]]
```

- Logistic Regression algorithm:

```
Accuracy_LogisticRegression : 79.1044776119403
Classification_report_LogisticRegression:
              precision    recall  f1-score   support

           0       0.77      0.70      0.74       111
           1       0.80      0.85      0.83       157

    accuracy                           0.79       268
   macro avg       0.79      0.78      0.78       268
weighted avg       0.79      0.79      0.79       268

Confusing Matrix_LogisticRegression :
 [[ 78  33]
 [ 23 134]]
```

- Decision Tree algorithm :

```
Accuracy_DecisionTree : 70.8955223880597
Classification_report_DecisionTree:
              precision    recall  f1-score   support

           0       0.57      0.62      0.60        93
           1       0.79      0.75      0.77       175

    accuracy                           0.71       268
   macro avg       0.68      0.69      0.68       268
weighted avg       0.72      0.71      0.71       268

Confusing Matrix_DecisionTree :
 [[ 58  35]
 [ 43 132]]
```

- Random Forests algorithm:

```
Accuracy_RandomForest : 80.59701492537313
Classification_report_RandomForest:
              precision    recall  f1-score   support

           0       0.75      0.74      0.75       103
           1       0.84      0.85      0.84       165

    accuracy                           0.81       268
   macro avg       0.80      0.79      0.79       268
weighted avg       0.81      0.81      0.81       268

Confusing Matrix_RandomForest :
 [[ 76  27]
 [ 25 140]]
```

At the beginning, it’s clear that the Random Forest algorithm gave me the greatest accuracy 80.59 % with True Female =76 and True Male=140

```
                        Algorithm
The Accuracy
80.597015          Random Forests
79.104478                     SVM
79.104478     Logistic Regression
71.641791             Naive bayes
70.895522           Decision Tree
```

I will use K-Fold Cross Validation to see the performance of Random Forests algorithm:

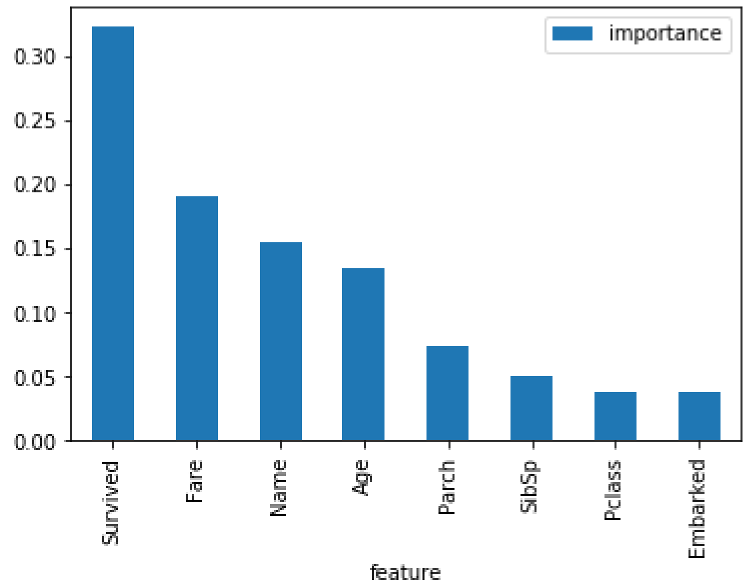
I will suppose we have K=15, so the result will be an array has 15 different values.

```
The Accuracy : [0.74418605 0.81395349 0.79069767 0.80952381 0.83333333 0.90243902
 0.80487805 0.73170732 0.80487805 0.7804878  0.75609756 0.75609756
 0.73170732 0.82926829 0.87804878]
Mean :  0.7978202738838022
Standard_Deviation:  0.04870312868019548
```

This figure gave us more realistic imagine about how Random Forest worked so finally, my model has average 79.78% with standard deviation approximately 4 % which indicates how precise the estimates are. That’s mean in this model the accuracy can differ-4 to +4 % as I expect.

Finally, If we take a look about the features which had higher effect in Random Forest Algorithm.

feature	importance
Survived	0.323
Fare	0.190
Name	0.154
Age	0.135
Parch	0.074
SibSp	0.050
Pclass	0.037
Embarked	0.037



As I can see that the features [Embarked , Pclass] have less effect so maybe by removing them we will get a higher accuracy .