Data Analysis on Titanic Dataset

Art and the Artist

Introduction to the Dataset - Dhamini

Transformation - Saheema

Predictive Model - Ronald and Priyangka

Forecasting Model - Arjun

Classifiers - Yasharth and Dhamini

Summarizing all together - Dhamini

Introduction to the Dataset

Transformation

```
import copy
import pandas as pd
data=pd.read_csv("/content/titanic.csv")
data.head(80)
x_min=min(data["Age"])
y_max=max(data["Age"])
age_transformed=copy.copy(data["Age"])
for i in range(len(data["Age"])):
  age_transformed[i]=((data["Age"])[i]-x_min)/(y_max-x_min)
print(data["Age"])
print(age_transformed)
```

Predictive model

```python
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import warnings
```

```python
[ ] warnings.filterwarnings('ignore')
```

```python
[ ] from google.colab import files
    uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the curr

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

```python
[ ] df = pd.read_csv('train.csv')
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
[ ] df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```
[ ] df.shape
```

```
(891, 12)
```

```
//hypothesis
//female survived more
// passenger with first class survived.

pd.crosstab(df['Sex'],df['Survived'])
```

Survived	0	1
Sex		
female	81	233
male	468	109

```
// passenger with first class survived.
pd.crosstab(df['Pclass'],df['Survived'])
```

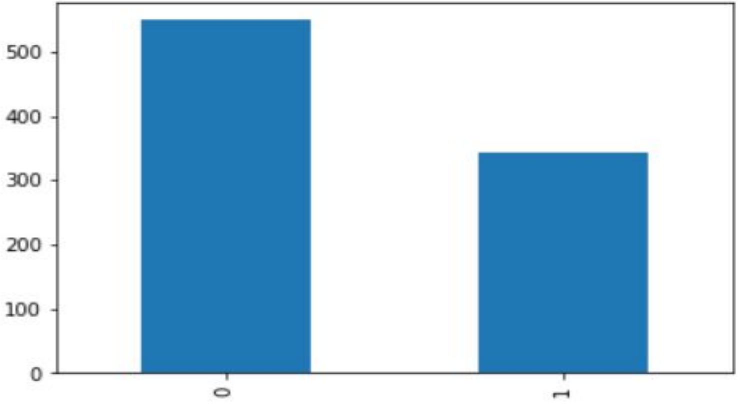
Survived	0	1
Pclass		
1	80	136
2	97	87
3	372	119

```
df.dtypes
```

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

```
df['Survived'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc35aa3c3c8>
```



```
[ ] df.isnull().sum()
    //
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
[ ]
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0
std	257.353842	0.486592	0.836071	14.526497	1.102743	0
min	1.000000	0.000000	1.000000	0.420000	0.000000	0
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0
max	891.000000	1.000000	3.000000	80.000000	8.000000	6

```
[ ] df['Age'].value_counts()
```

```
young_Adult        358
adults             153
children           139
Senior citizens     64
Name: Age, dtype: int64
```

```
[]
def fill_age(df):
  bins=[0,18,35,50,80]
  group=['children','young_Adults','Adults','Senior citizens']
  df['Age']=pd.cut(df['Age'],bins, labels=group)
  mode=df['Age'].mode()[0]
  df['Age'].fillna(mode,inplace=True)
  return df
[]
df['Embarked'].mode()[0]
```

```
'S'
```

[]

```
def fill_Embarked(df):
  df.Embarked.fillna('S',inplace=TRUE)
  return(df)
```

[]

```
df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

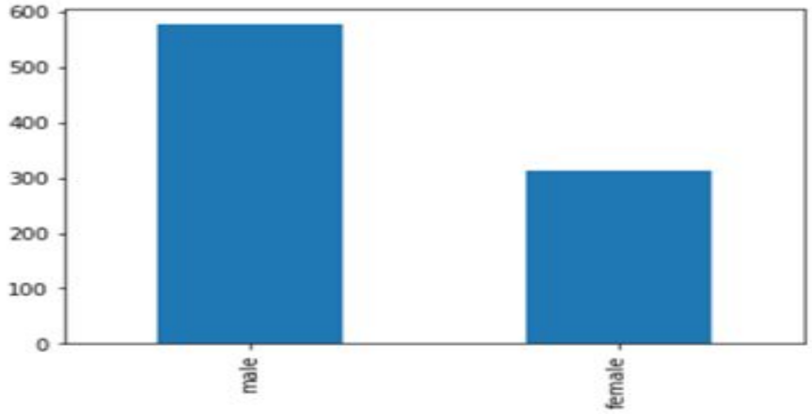
[]

```
df['Pclass'].value_counts()
```

```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

```
[]
df['Sex'].value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f21ec270128>
```




```
df['Sex'].value_counts()
```

```
male      577
female    314
Name: Sex, dtype: int64
```

```
[]
df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

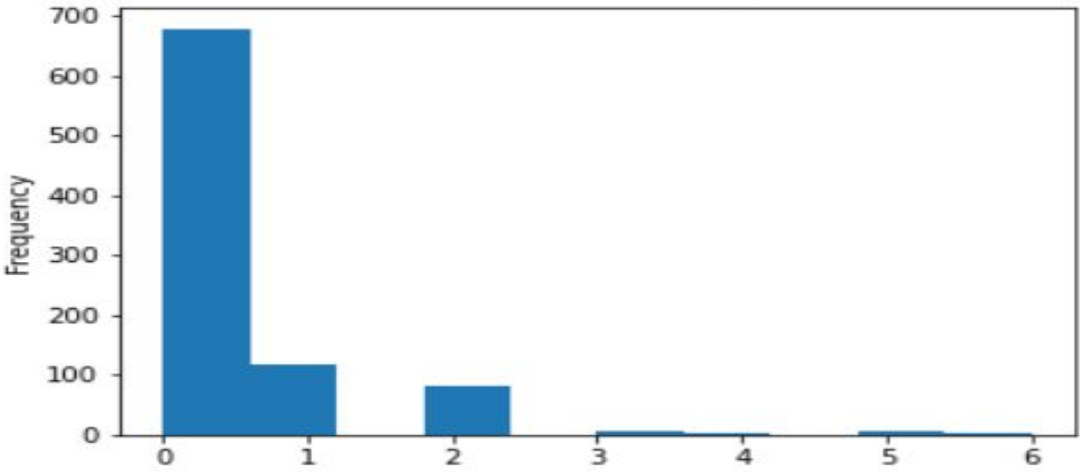
[]

```
df['SibSp'].value_counts()
```

```
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```

```
df['Parch'].plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f21ebd77668>



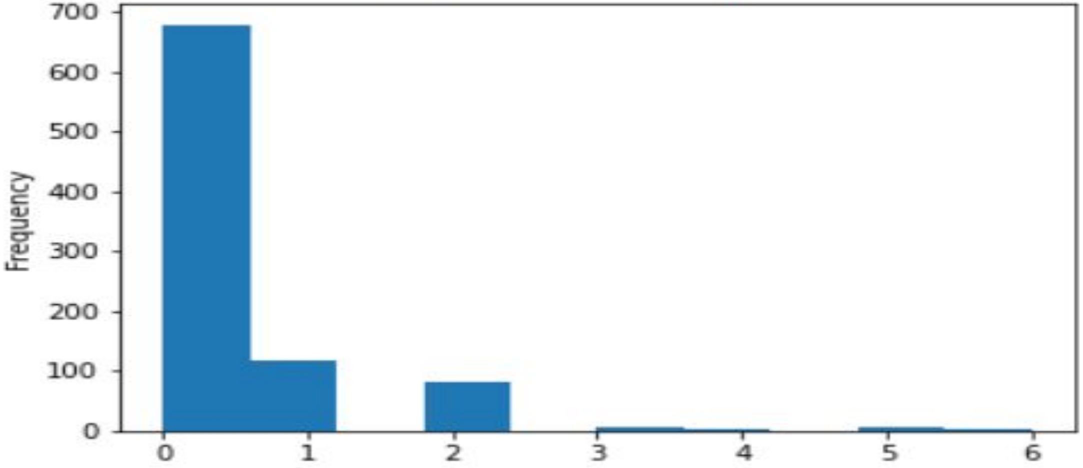
[]

```
df['SibSp'].value_counts()
```

```
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```

```
df['Parch'].plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f21ebd77668>



[]

```python
df['Cabin'].value_counts()
```

```
B96 B98        4
G6             4
C23 C25 C27    4
D              3
E101           3
              ..
A6             1
B71            1
D11            1
B69            1
C106           1
Name: Cabin, Length: 147, dtype: int64
```

[]

```python
df['Embarked'].value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

[]
```
df.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

[]
```
df['Ticket'].value_counts()
```

```
1601                7
347082              7
CA. 2343            7
CA 2144             6
347088              6
                   ..
248747              1
SO/C 14885          1
SC/PARIS 2131       1
PC 17595            1
7267                1
Name: Ticket, Length: 681, dtype: int64
```

```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

[]

```
train,test=train_test_split(df,test_size=0.2,random_state=12)
```

[]

```
train.shape,test.shape
```

```
((712, 12), (179, 12))
```

```
[]
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	young_Adults	0	2	347742	11.1333	NaN	S
150	151	0	2	Bateman, Rev. Robert James	male	Senior citizens	0	0	S.O.P. 1166	12.5250	NaN	S
221	222	0	2	Bracken, Mr. James H	male	young_Adults	0	0	220367	13.0000	NaN	S
365	366	0	3	Adahl, Mr. Mauritz Nils Martin	male	young_Adults	0	0	C 7076	7.2500	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	young_Adults	8	2	CA. 2343	69.5500	NaN	S

```
[]
def x_and_y (df):
    x=df.drop(['Survived','PassengerId','Cabin','Name','Ticket',],axis=1)
    y=df['Survived']
    return x,y
```

```
[]
x_train,y_train=x_and_y(train)
x_test,y_test=x_and_y(test)
```

```
[]
y_train.isnull().sum()
```

```
0
```

```
[]
log_model=LogisticRegression()
log_model.fit(x_train,y_train)
prediction=log_model.predict(x_test)
score=accuracy_score(y_test,prediction)
print(score*100)
```

```
78.77094972067039
```

Forecasting Model

- Forecasting models are used to forecast future data as a function of past data.

- If there us no X value we go for forecasting model.

- X values are not used for forecasting model.

Forecasting Model

Time Series Data :

A time series is a sequence of observations over a certain period. The simplest example of a time series that all of us come across on a day to day basis is the change in temperature throughout the day or week or month or year.

Forecasting Model

Types of Time Series models:

- Linear: Auto-regressive models (AR), with moving average (ARMA), with integral terms (ARIMA), with local regression (ARMAX)

- Non-linear: Recurrent Neural Networks (RNN)

Forecasting Model

Auto-regressive Model : An AR model is one in which Y_t depends on its own past values. AR(p)

Moving Average Model : A MA model is one when Y_t depends only on random error term which follow a white noise process.

Forecasting Model

Auto-regressive Moving Average Model (ARMA) :

There are situations where the time-series may be represented as a mox of both AR and MA model which is refered as ARMA (p,q) model

The model depends on p of its own past values and q past values of white noise disturbances.

Forecasting Model

Non Linear model : Recurrent Models

A Recurrent Model takes an input and one output, The output of the model is now fed back to the model as a new input

Eg: Recurrent Neural Networks (RNN)

Classifiers

KNN

Summarize