Deep Learning Assignment No.1

Que.1 : The sinking of the Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew. While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others. In this challenge, we ask you to build a predictive model that answers the question: “what sorts of people were more likely to survive?”

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

```
titanic_df = pd.read_csv("C:/Users/Lenovo/Documents/Data Set/titanic.csv")
```

```
titanic_df
```

```
     Unnamed: 0  PassengerId  Survived  Pclass  \
0             0            1         0       3
1             1            2         1       1
2             2            3         1       3
3             3            4         1       1
4             4            5         0       3
..          ...          ...       ...     ...
707         885          886         0       3
708         886          887         0       2
709         887          888         1       1
710         889          890         1       1
711         890          891         0       3

                                                  Name     Sex   Age
SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0
1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
```

```
1
2                              Heikkinen, Miss. Laina  female  26.0
0
3        Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                            Allen, Mr. William Henry    male  35.0
0
..                                                ...     ...   ...
...
707               Rice, Mrs. William (Margaret Norton)  female  39.0
0
708                              Montvila, Rev. Juozas    male  27.0
0
709                       Graham, Miss. Margaret Edith  female  19.0
0
710                              Behr, Mr. Karl Howell    male  26.0
0
711                                Dooley, Mr. Patrick    male  32.0
0

     Parch            Ticket     Fare Embarked
0        0         A/5 21171   7.2500        S
1        0          PC 17599  71.2833        C
2        0  STON/O2. 3101282   7.9250        S
3        0            113803  53.1000        S
4        0            373450   8.0500        S
..     ...               ...      ...      ...
707      5            382652  29.1250        Q
708      0            211536  13.0000        S
709      0            112053  30.0000        S
710      0            111369  30.0000        C
711      0            370376   7.7500        Q

[712 rows x 12 columns]
```

```
titanic_df.head()
```

```
   Unnamed: 0  PassengerId  Survived  Pclass  \
0           0            1         0       3
1           1            2         1       1
2           2            3         1       3
3           3            4         1       1
4           4            5         0       3

                                                Name     Sex   Age
SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0
1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
1
```

```
2                             Heikkinen, Miss. Laina  female  26.0
0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
1
4                           Allen, Mr. William Henry    male  35.0
0

   Parch            Ticket     Fare Embarked
0      0         A/5 21171   7.2500        S
1      0          PC 17599  71.2833        C
2      0  STON/O2. 3101282   7.9250        S
3      0            113803  53.1000        S
4      0            373450   8.0500        S
```

```
titanic_df.tail()
```

```
     Unnamed: 0  PassengerId  Survived  Pclass  \
707         885          886         0       3
708         886          887         0       2
709         887          888         1       1
710         889          890         1       1
711         890          891         0       3

                                     Name     Sex   Age  SibSp  Parch
Ticket  \
707  Rice, Mrs. William (Margaret Norton)  female  39.0      0      5
382652
708                 Montvila, Rev. Juozas    male  27.0      0      0
211536
709          Graham, Miss. Margaret Edith  female  19.0      0      0
112053
710                 Behr, Mr. Karl Howell    male  26.0      0      0
111369
711                   Dooley, Mr. Patrick    male  32.0      0      0
370376

       Fare Embarked
707  29.125        Q
708  13.000        S
709  30.000        S
710  30.000        C
711   7.750        Q
```

```
titanic_df.isnull().sum()
```

```
Unnamed: 0     0
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
```

```
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

```
titanic_df.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
707    False
708    False
709    False
710    False
711    False
Length: 712, dtype: bool
```

```
titanic_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 712 entries, 0 to 711
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   712 non-null    int64
 1   PassengerId  712 non-null    int64
 2   Survived     712 non-null    int64
 3   Pclass       712 non-null    int64
 4   Name         712 non-null    object
 5   Sex          712 non-null    object
 6   Age          712 non-null    float64
 7   SibSp        712 non-null    int64
 8   Parch        712 non-null    int64
 9   Ticket       712 non-null    object
 10  Fare         712 non-null    float64
 11  Embarked     712 non-null    object
dtypes: float64(2), int64(6), object(4)
memory usage: 66.9+ KB
```

```
titanic_df.describe()
```

```
       Unnamed: 0  PassengerId    Survived      Pclass         Age  \
count  712.000000   712.000000  712.000000  712.000000  712.000000
mean   447.589888   448.589888    0.404494    2.240169   29.642093
std    258.683191   258.683191    0.491139    0.836854   14.492933
```

```
min      0.000000    1.000000    0.000000    1.000000    0.420000
25%    221.750000  222.750000    0.000000    1.000000   20.000000
50%    444.000000  445.000000    0.000000    2.000000   28.000000
75%    676.250000  677.250000    1.000000    3.000000   38.000000
max    890.000000  891.000000    1.000000    3.000000   80.000000

            SibSp       Parch        Fare
count  712.000000  712.000000  712.000000
mean     0.514045    0.432584   34.567251
std      0.930692    0.854181   52.938648
min      0.000000    0.000000    0.000000
25%      0.000000    0.000000    8.050000
50%      0.000000    0.000000   15.645850
75%      1.000000    1.000000   33.000000
max      5.000000    6.000000  512.329200
```

```
titanic_df = titanic_df.drop(["Unnamed:
0","PassengerId","Name","Ticket"],axis = 1)
```

```
titanic_df
```

```
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked
0           0       3    male  22.0      1      0   7.2500        S
1           1       1  female  38.0      1      0  71.2833        C
2           1       3  female  26.0      0      0   7.9250        S
3           1       1  female  35.0      1      0  53.1000        S
4           0       3    male  35.0      0      0   8.0500        S
..        ...     ...     ...   ...    ...    ...      ...      ...
707         0       3  female  39.0      0      5  29.1250        Q
708         0       2    male  27.0      0      0  13.0000        S
709         1       1  female  19.0      0      0  30.0000        S
710         1       1    male  26.0      0      0  30.0000        C
711         0       3    male  32.0      0      0   7.7500        Q

[712 rows x 8 columns]
```

```
titanic_df.shape
```

```
(712, 8)
```

```
print(titanic_df.columns)
```

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

```
titanic_df.corr()
```

```
          Survived    Pclass       Age     SibSp     Parch      Fare
Survived  1.000000 -0.356462 -0.082446 -0.015523  0.095265  0.266100
Pclass   -0.356462  1.000000 -0.365902  0.065187  0.023666 -0.552893
Age      -0.082446 -0.365902  1.000000 -0.307351 -0.187896  0.093143
```

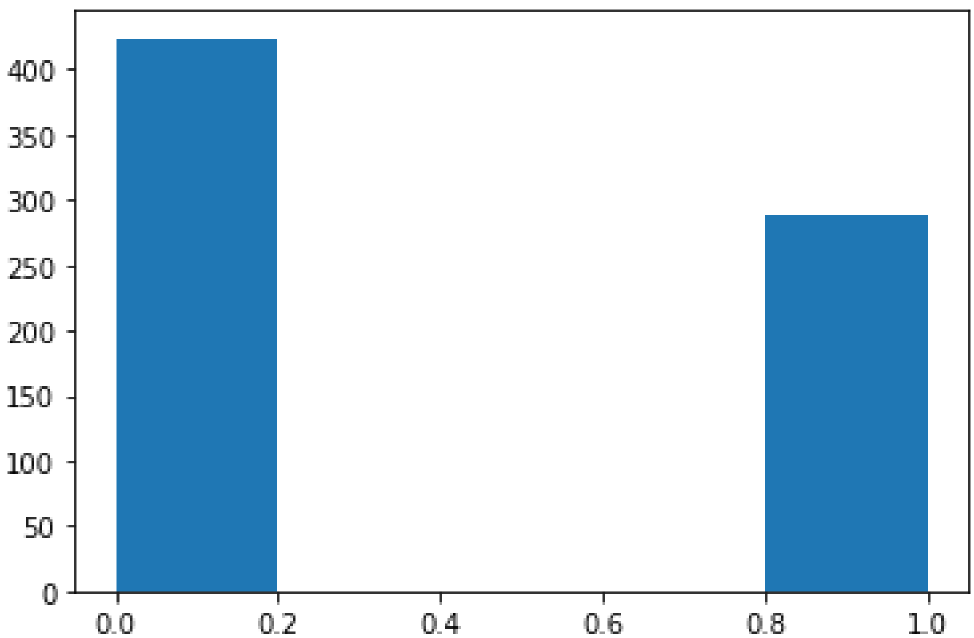
```
SibSp    -0.015523  0.065187 -0.307351  1.000000  0.383338  0.139860
Parch     0.095265  0.023666 -0.187896  0.383338  1.000000  0.206624
Fare      0.266100 -0.552893  0.093143  0.139860  0.206624  1.000000
```

Exploratory Data Analysis of each column

1. Histogram

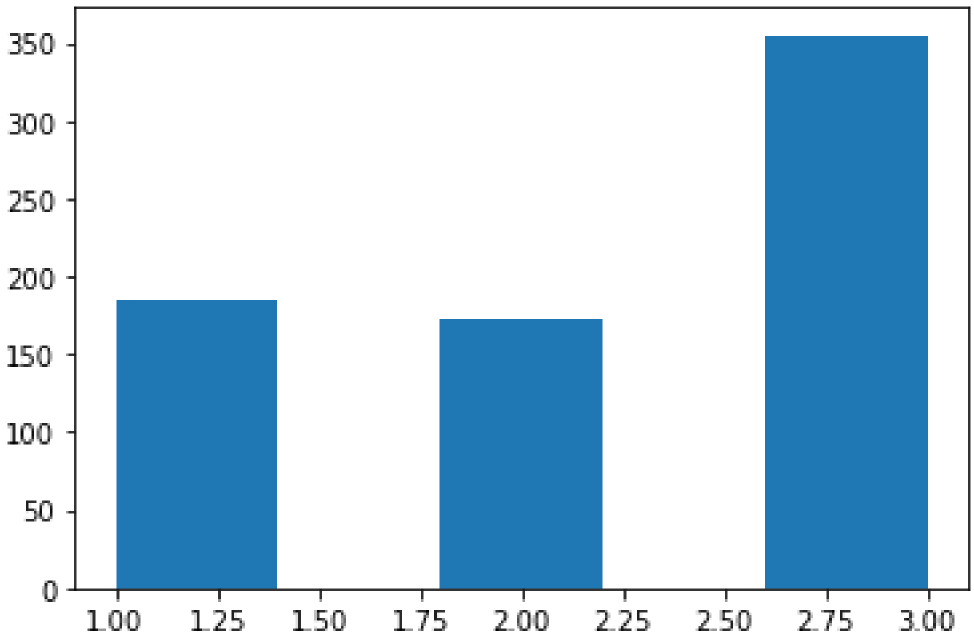
```
import matplotlib.pyplot as plt
plt.hist(titanic_df['Survived'],bins=5)
```

```
(array([424.,   0.,   0.,   0., 288.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. ]),
 <BarContainer object of 5 artists>)
```



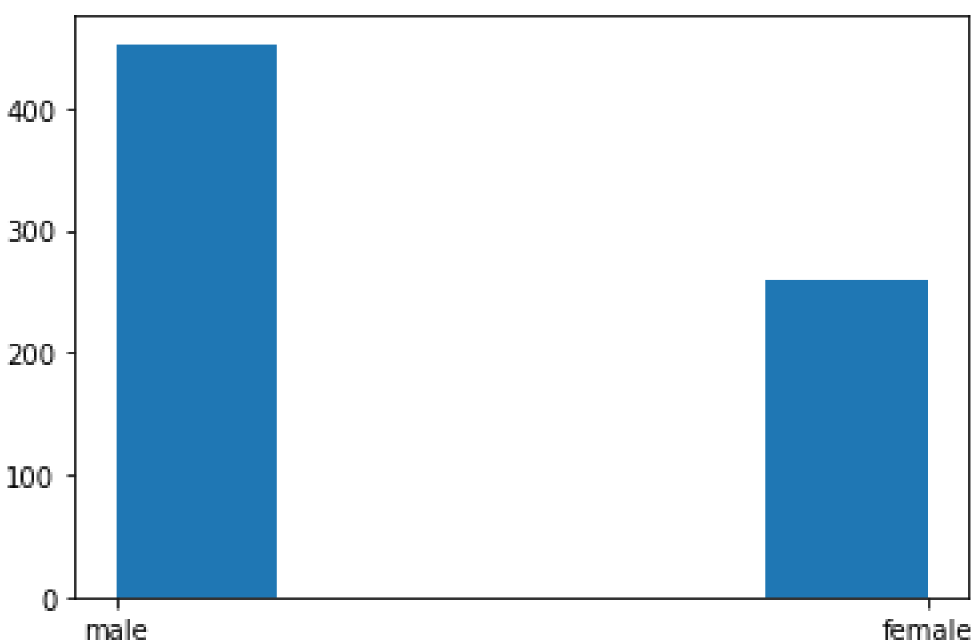

```
import matplotlib.pyplot as plt
plt.hist(titanic_df['Pclass'],bins=5)
```

```
(array([184.,   0., 173.,   0., 355.]),
 array([1. , 1.4, 1.8, 2.2, 2.6, 3. ]),
 <BarContainer object of 5 artists>)
```



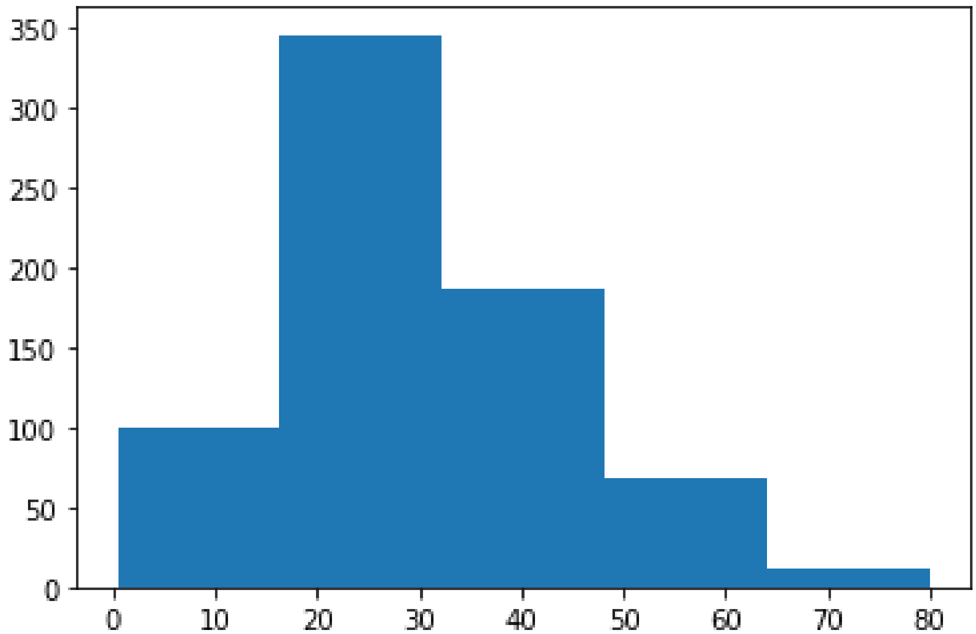

```
import matplotlib.pyplot as plt
plt.hist(titanic_df['Sex'],bins=5)
```

```
(array([453.,   0.,   0.,   0., 259.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. ]),
 <BarContainer object of 5 artists>)
```



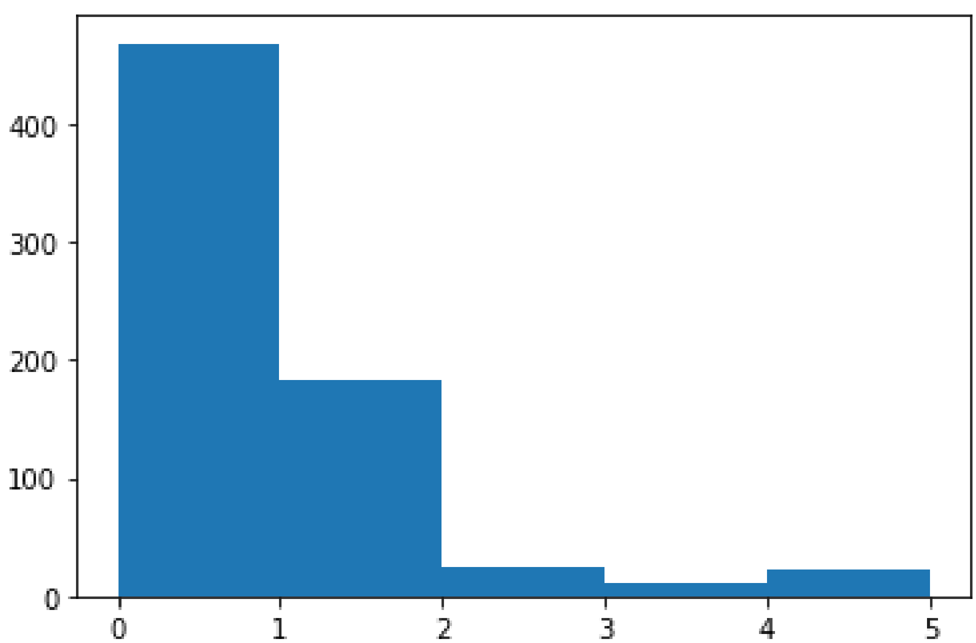

```
import matplotlib.pyplot as plt
plt.hist(titanic_df['Age'],bins=5)
```

```
(array([100., 346., 187.,  68.,  11.]),
 array([ 0.42 , 16.336, 32.252, 48.168, 64.084, 80.   ]),
 <BarContainer object of 5 artists>)
```



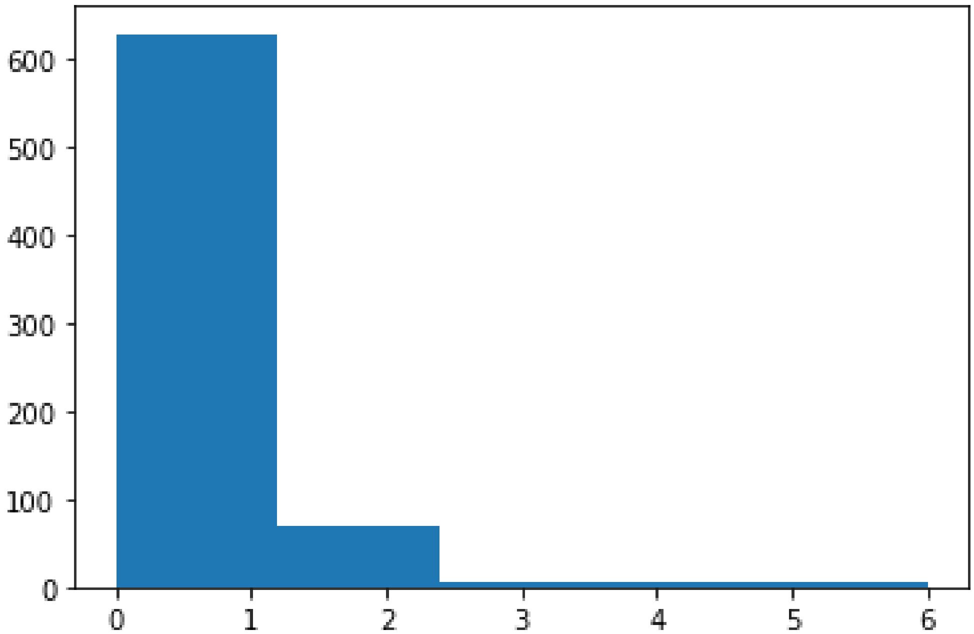

```
import matplotlib.pyplot as plt
plt.hist(titanic_df['SibSp'],bins=5)
```

```
(array([469., 183.,  25.,  12.,  23.]),
 array([0., 1., 2., 3., 4., 5.]),
 <BarContainer object of 5 artists>)
```



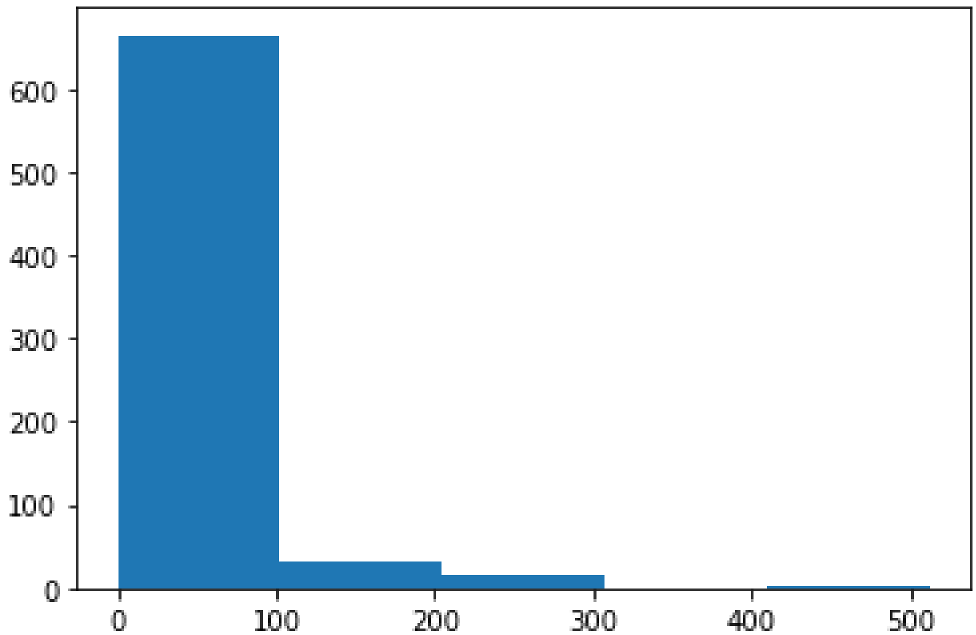
```python
import matplotlib.pyplot as plt
plt.hist(titanic_df['Parch'],bins=5)
```

```
(array([629.,  68.,   5.,   4.,   6.]),
 array([0. , 1.2, 2.4, 3.6, 4.8, 6. ]),
 <BarContainer object of 5 artists>)
```



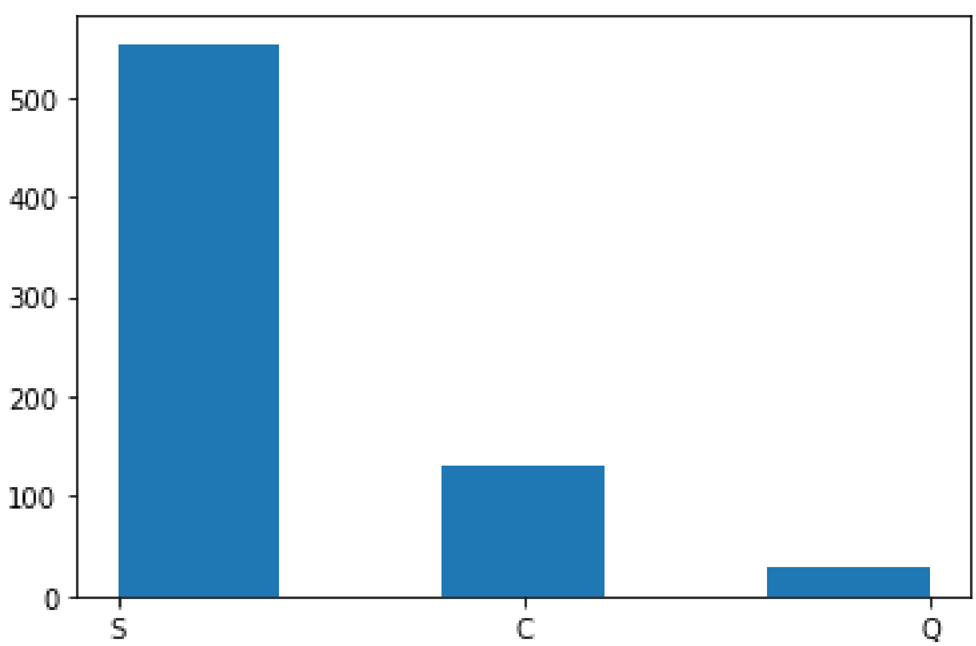

```python
import matplotlib.pyplot as plt
plt.hist(titanic_df['Fare'],bins=5)
```

```
(array([664.,  30.,  15.,   0.,   3.]),
 array([  0.     , 102.46584, 204.93168, 307.39752, 409.86336,
512.3292 ]),
 <BarContainer object of 5 artists>)
```




```
import matplotlib.pyplot as plt
plt.hist(titanic_df['Embarked'],bins=5)
```

```
(array([554.,   0., 130.,   0.,  28.]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. ]),
 <BarContainer object of 5 artists>)
```

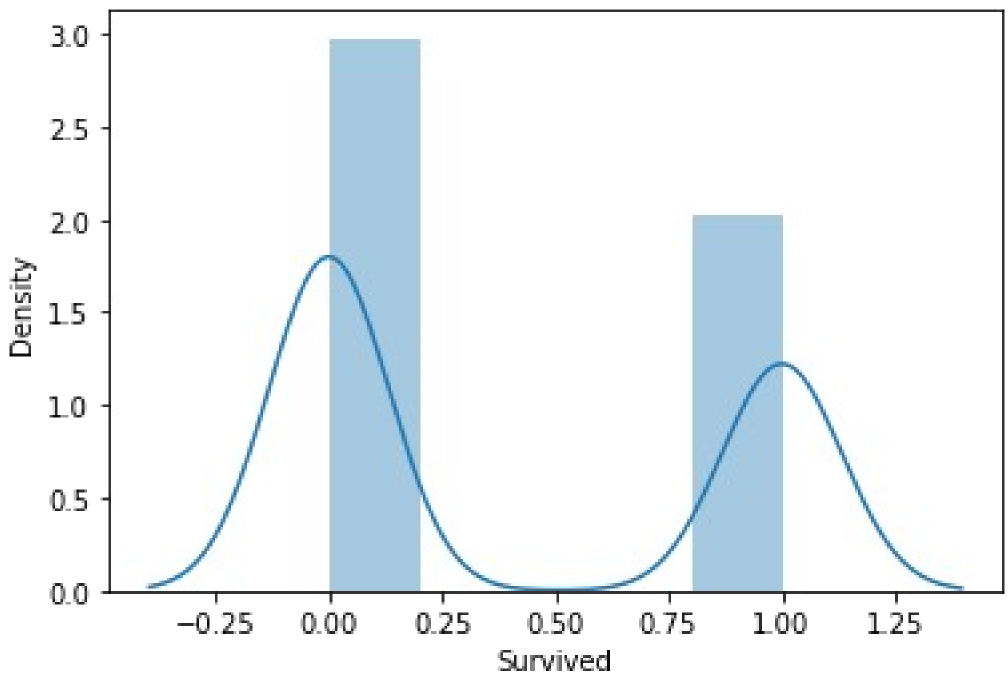


2.Distplot

```
import seaborn as sns
sns.distplot(titanic_df['Survived'])
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

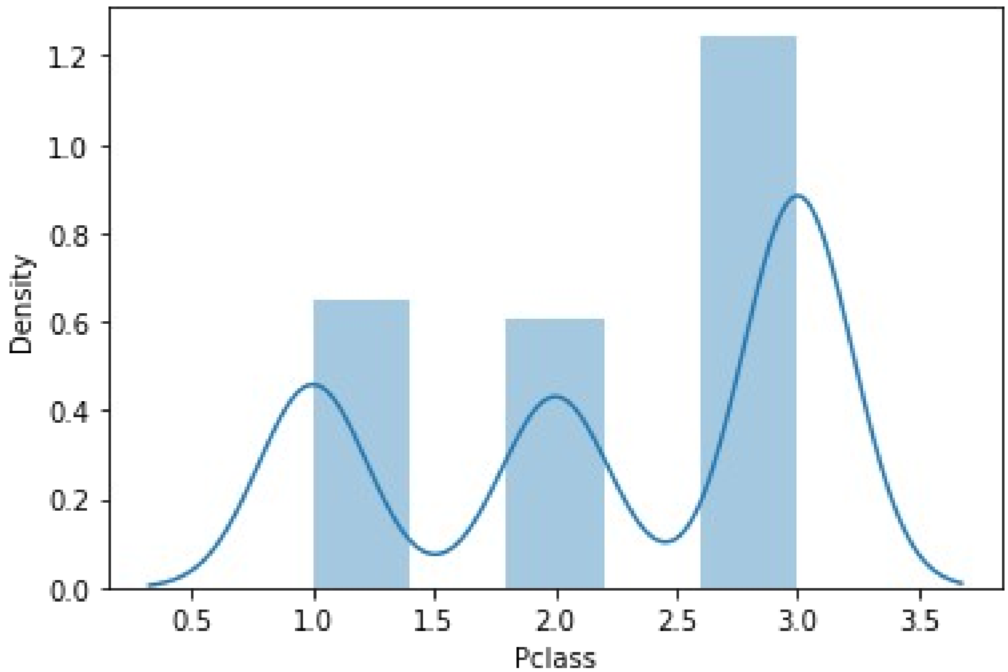
```
<AxesSubplot:xlabel='Survived', ylabel='Density'>
```




```
import seaborn as sns
sns.distplot(titanic_df['Pclass'])
```

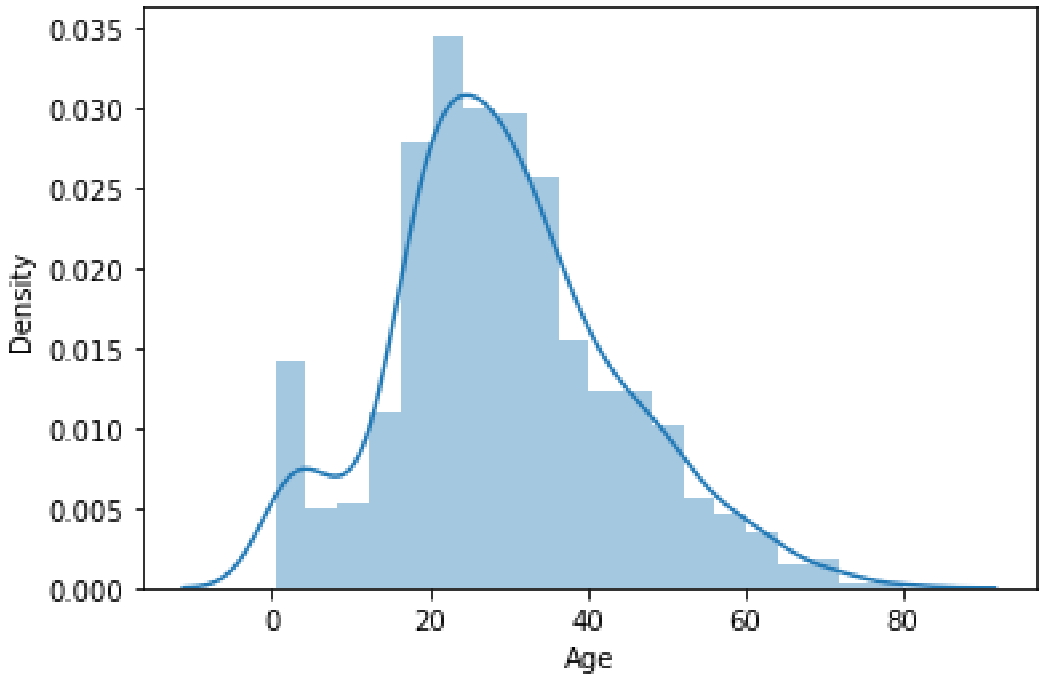
```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='Pclass', ylabel='Density'>
```




```
import seaborn as sns
sns.distplot(titanic_df['Age'])
```

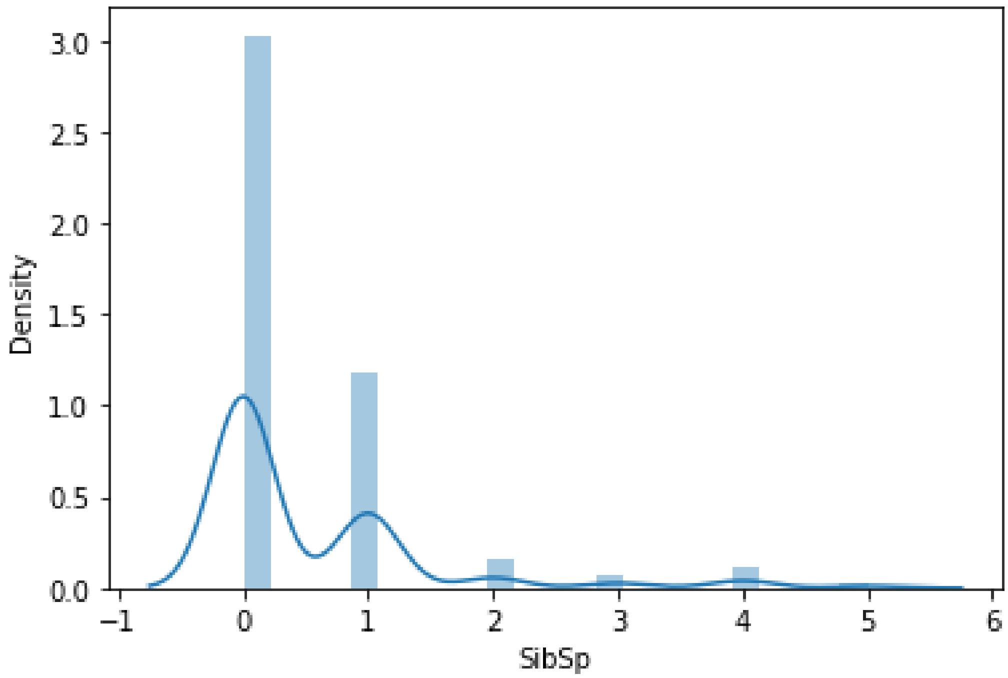
```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='Age', ylabel='Density'>
```




```
import seaborn as sns
sns.distplot(titanic_df['SibSp'])
```

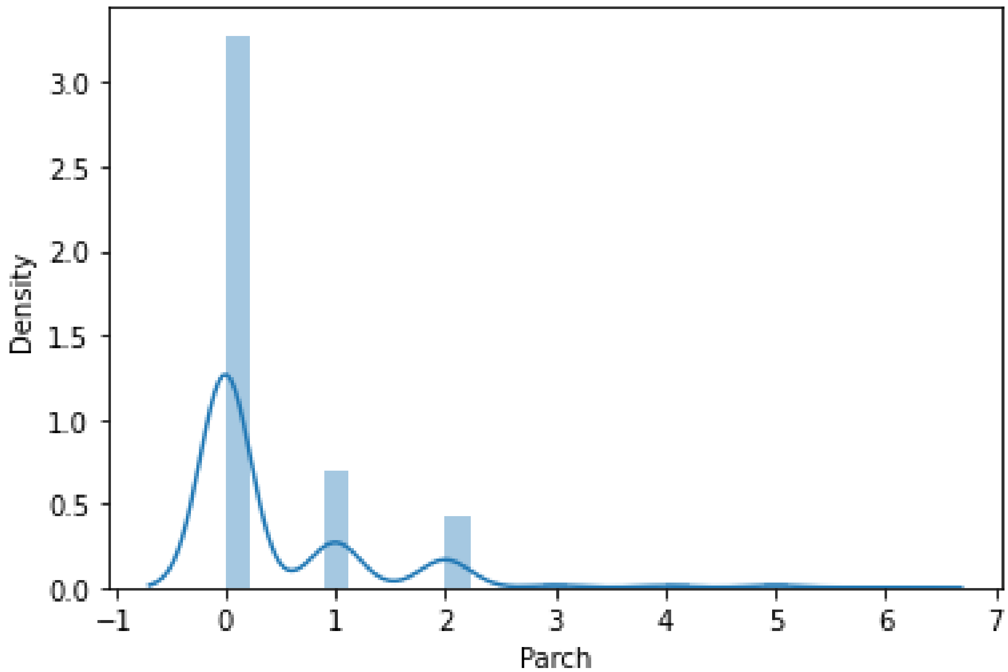
```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='SibSp', ylabel='Density'>
```



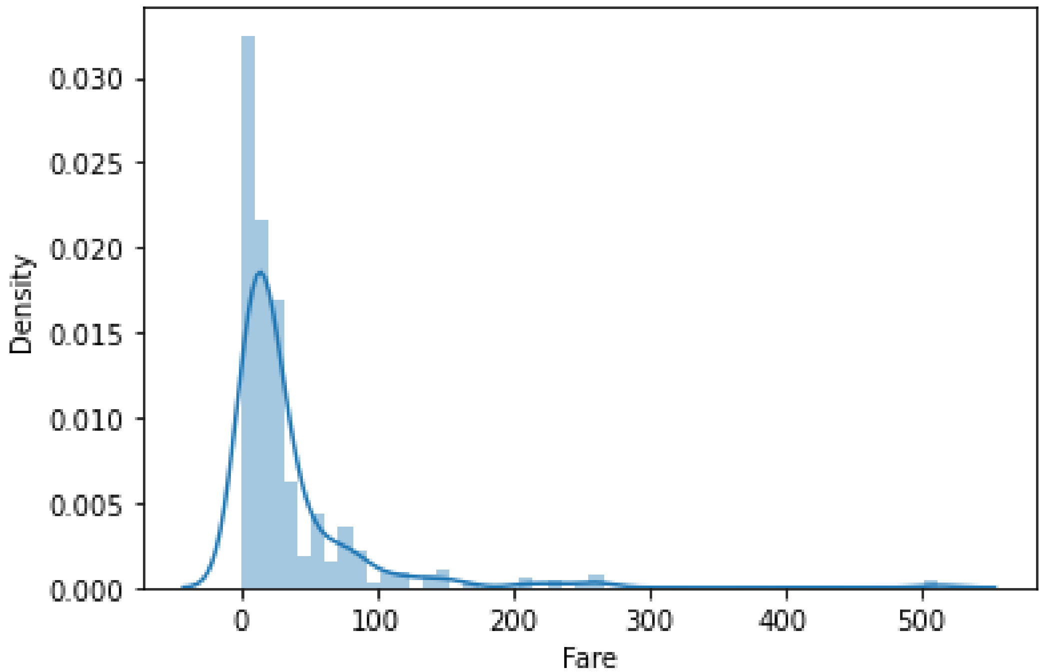

```python
import seaborn as sns
sns.distplot(titanic_df['Parch'])
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='Parch', ylabel='Density'>
```




```python
import seaborn as sns
sns.distplot(titanic_df['Fare'])
```

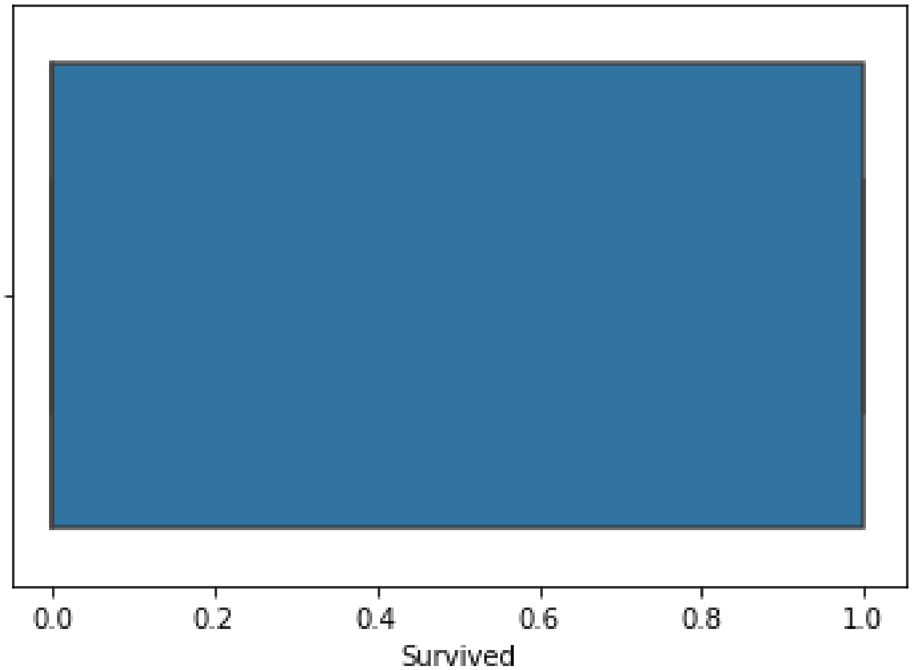
```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\
distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='Fare', ylabel='Density'>
```



3.Box Plot

```
import seaborn as sns
sns.boxplot(titanic_df['Survived'])
```

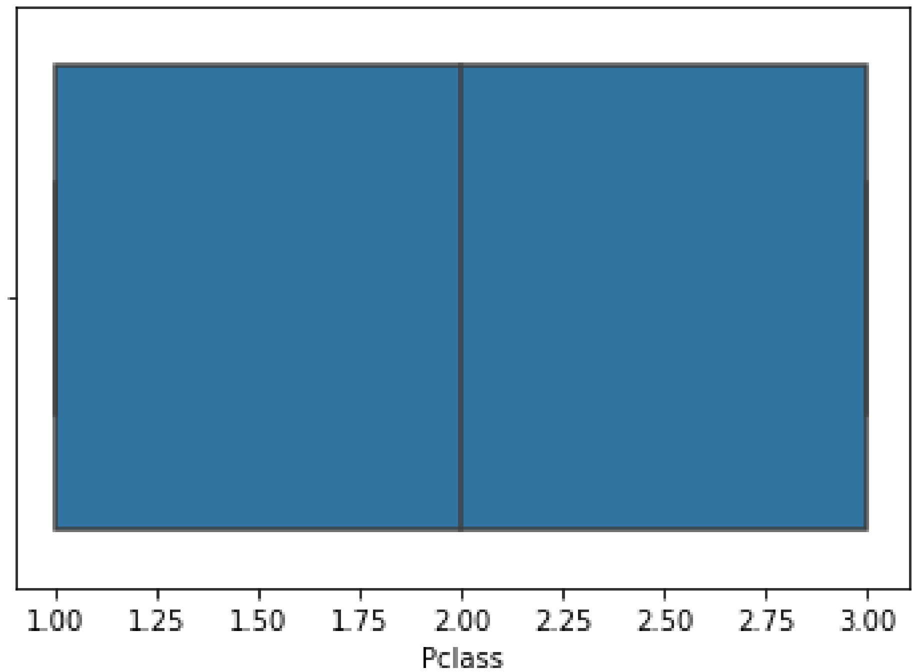
```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(

<AxesSubplot:xlabel='Survived'>
```




```python
import seaborn as sns
sns.boxplot(titanic_df['Pclass'])
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

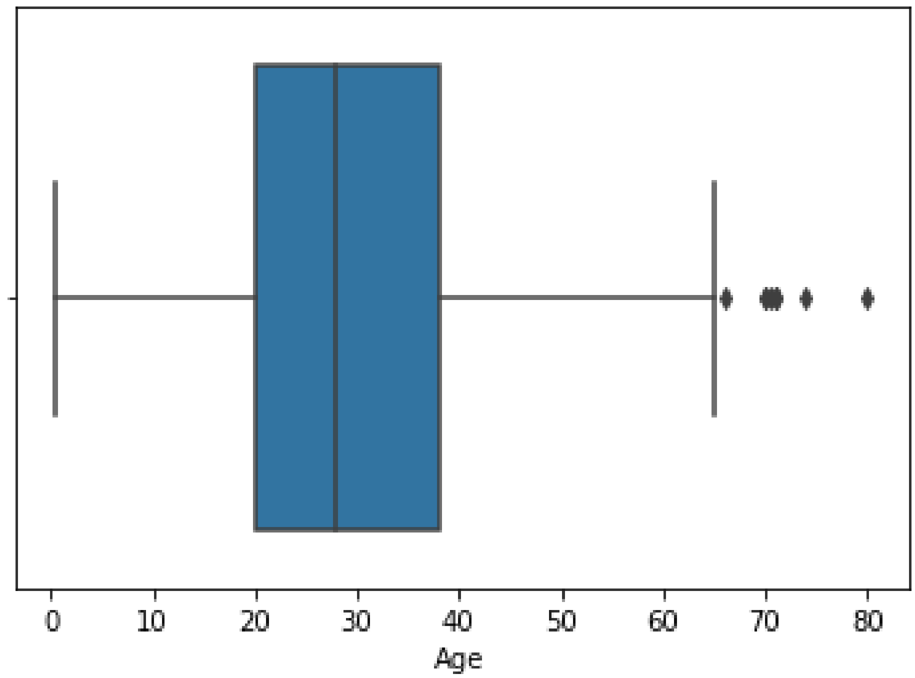
```
<AxesSubplot:xlabel='Pclass'>
```




```
import seaborn as sns
sns.boxplot(titanic_df['Age'])
```

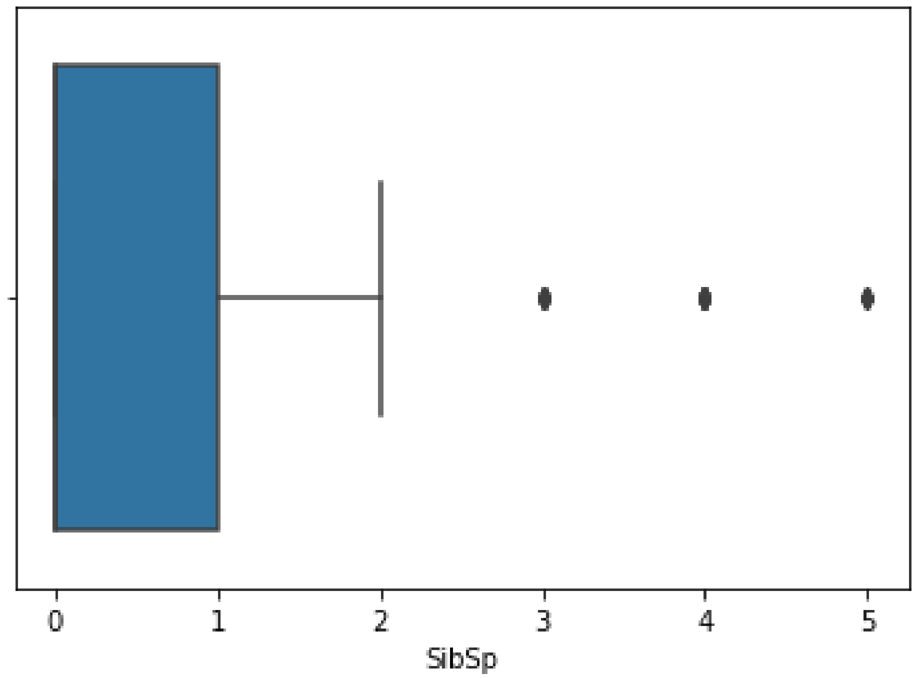
```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='Age'>
```




```
import seaborn as sns
sns.boxplot(titanic_df['SibSp'])
```

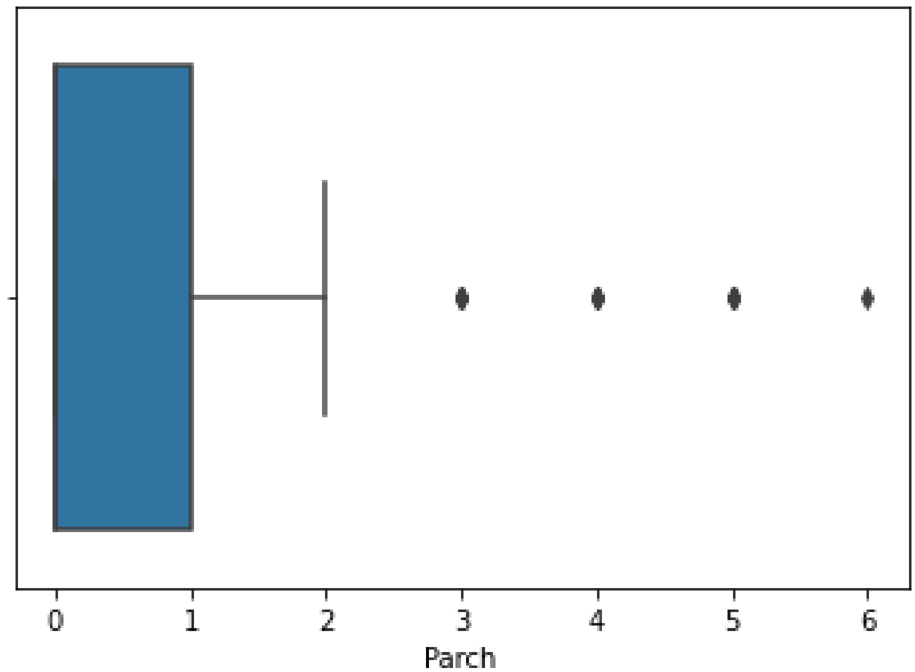
```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(

<AxesSubplot:xlabel='SibSp'>
```




```
import seaborn as sns
sns.boxplot(titanic_df['Parch'])
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

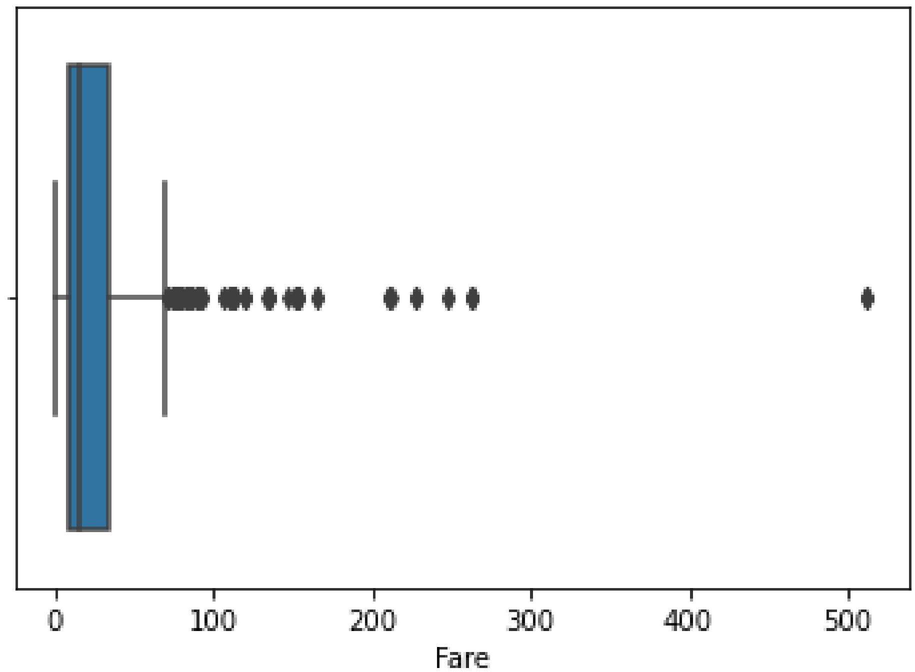
```
<AxesSubplot:xlabel='Parch'>
```




```
import seaborn as sns
sns.boxplot(titanic_df['Fare'])
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(
```

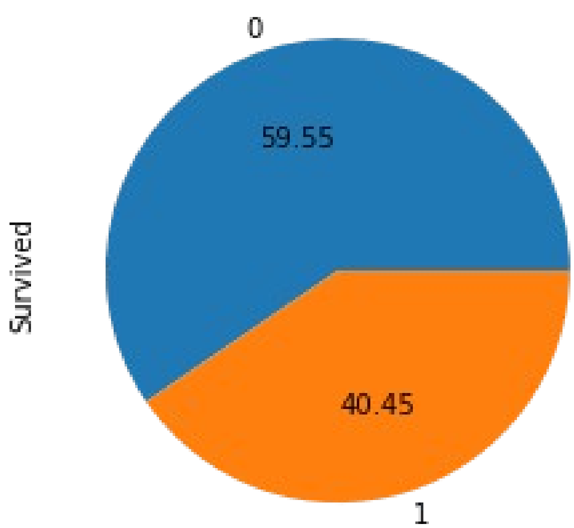
```
<AxesSubplot:xlabel='Fare'>
```



4.Pie Chart

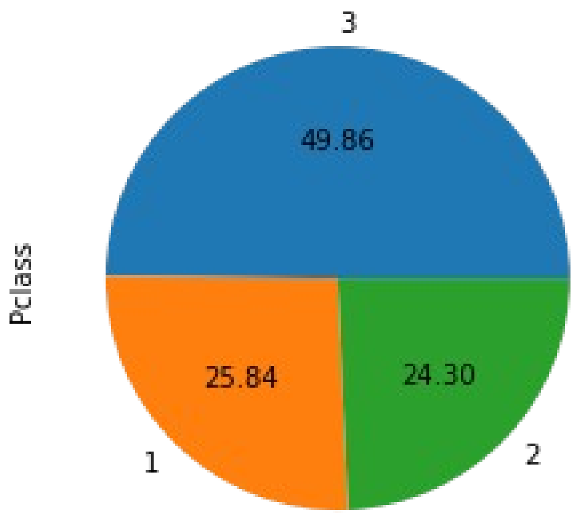
```
titanic_df['Survived'].value_counts().plot(kind='pie',autopct='%.2f')
```

```
<AxesSubplot:ylabel='Survived'>
```



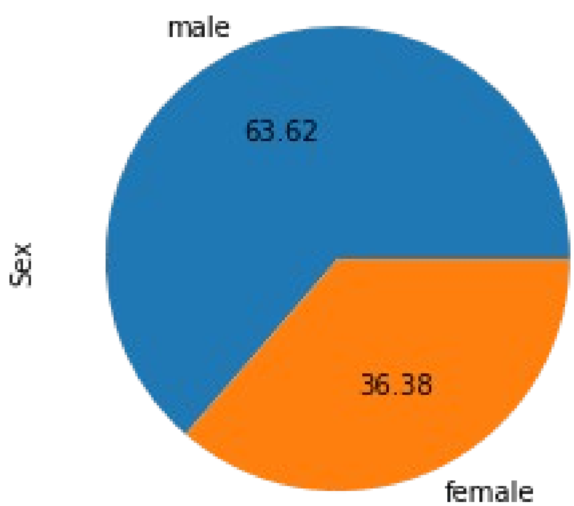

```
titanic_df['Pclass'].value_counts().plot(kind='pie',autopct='%.2f')
```

```
<AxesSubplot:ylabel='Pclass'>
```



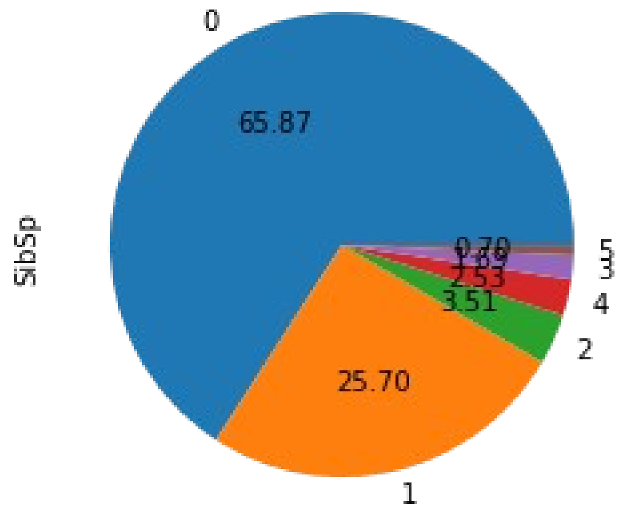

```
titanic_df['Sex'].value_counts().plot(kind='pie',autopct='%.2f')
```

<AxesSubplot:ylabel='Sex'>



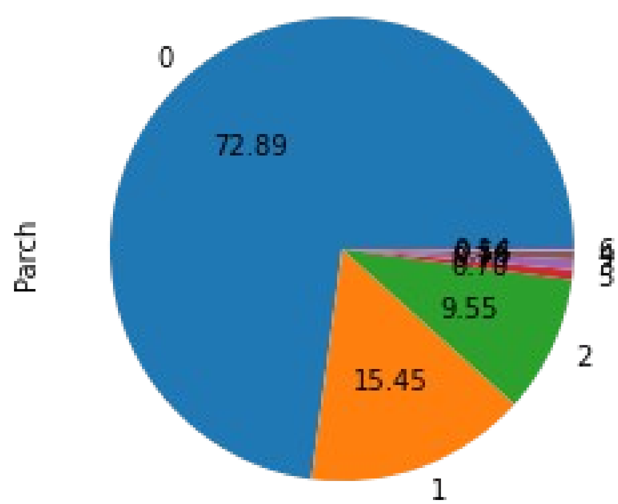

```
titanic_df['SibSp'].value_counts().plot(kind='pie',autopct='%.2f')
```

<AxesSubplot:ylabel='SibSp'>



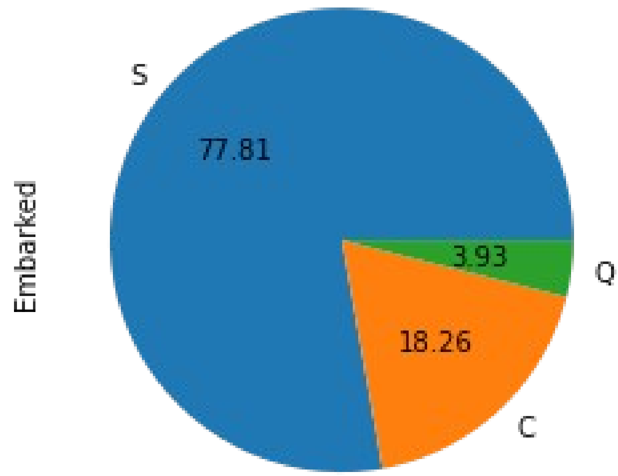

```
titanic_df['Parch'].value_counts().plot(kind='pie',autopct='%.2f')
```

<AxesSubplot:ylabel='Parch'>




```
titanic_df['Embarked'].value_counts().plot(kind='pie',autopct='%.2f')
```

<AxesSubplot:ylabel='Embarked'>



Define a function to remove outliers using the IQR method

```
def remove_outliers(titanic_df1, column):
    Q1 = titanic_df1[column].quantile(0.25)
    Q3 = titanic_df1[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return titanic_df1[(titanic_df1[column] >= lower_bound) &
(titanic_df1[column] <= upper_bound)]

# Call the function on the desired columns of the dataset
df = remove_outliers(titanic_df, 'Fare')

df
```

```
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked
0           0       3    male  22.0      1      0   7.2500        S
2           1       3  female  26.0      0      0   7.9250        S
3           1       1  female  35.0      1      0  53.1000        S
4           0       3    male  35.0      0      0   8.0500        S
5           0       1    male  54.0      0      0  51.8625        S
..        ...     ...     ...   ...    ...    ...      ...      ...
707         0       3  female  39.0      0      5  29.1250        Q
708         0       2    male  27.0      0      0  13.0000        S
709         1       1  female  19.0      0      0  30.0000        S
710         1       1    male  26.0      0      0  30.0000        C
711         0       3    male  32.0      0      0   7.7500        Q

[617 rows x 8 columns]
```

```
df.shape
```

```
(617, 8)
```

```
# Count the occurrences of each unique value in the column
value_counts = df['SibSp'].value_counts()
```

```
value_counts
```

```
0    426
1    141
4     18
2     18
3      9
5      5
Name: SibSp, dtype: int64
```

```
# Count the occurrences of each unique value in the column
value_counts = df['Parch'].value_counts()
```

```
value_counts
```

```
0    464
1     90
2     49
5      5
3      5
4      3
6      1
Name: Parch, dtype: int64
```

Feature Engineering

```
# one-hot encode the 'gender' column
one_hot_encoded = pd.get_dummies(df[['Sex']])

# concatenate the original dataframe with the one-hot encoded
dataframe
df_encoded = pd.concat([df, one_hot_encoded], axis=1)

# display the resulting dataframe
print(df_encoded)
```

```
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked  \
0           0       3    male  22.0      1      0   7.2500        S
2           1       3  female  26.0      0      0   7.9250        S
3           1       1  female  35.0      1      0  53.1000        S
4           0       3    male  35.0      0      0   8.0500        S
5           0       1    male  54.0      0      0  51.8625        S
..        ...     ...     ...   ...    ...    ...      ...      ...
707         0       3  female  39.0      0      5  29.1250        Q
```

```
708          0       2    male  27.0      0      0  13.0000        S
709          1       1  female  19.0      0      0  30.0000        S
710          1       1    male  26.0      0      0  30.0000        C
711          0       3    male  32.0      0      0   7.7500        Q

     Sex_female  Sex_male
0             0         1
2             1         0
3             1         0
4             0         1
5             0         1
..          ...       ...
707           1         0
708           0         1
709           1         0
710           0         1
711           0         1

[617 rows x 10 columns]
```

```
df1 = df_encoded
```

```
df1
```

```
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked  \
0           0       3    male  22.0      1      0   7.2500        S
2           1       3  female  26.0      0      0   7.9250        S
3           1       1  female  35.0      1      0  53.1000        S
4           0       3    male  35.0      0      0   8.0500        S
5           0       1    male  54.0      0      0  51.8625        S
..        ...     ...     ...   ...    ...    ...      ...      ...
707         0       3  female  39.0      0      5  29.1250        Q
708         0       2    male  27.0      0      0  13.0000        S
709         1       1  female  19.0      0      0  30.0000        S
710         1       1    male  26.0      0      0  30.0000        C
711         0       3    male  32.0      0      0   7.7500        Q

     Sex_female  Sex_male
0             0         1
2             1         0
3             1         0
4             0         1
5             0         1
..          ...       ...
707           1         0
708           0         1
709           1         0
710           0         1
711           0         1
```

```
[617 rows x 10 columns]
```

Apply ordinal encoding to the Embarked column

```python
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder

# Define the categories and their order
categories = ['C', 'Q', 'S']

# Initialize the OrdinalEncoder object
encoder = OrdinalEncoder(categories=[categories])

# Fit and transform the 'Embarked' column using the encoder
df1['Embarked_encoded'] = encoder.fit_transform(df1[['Embarked']])

# Display the resulting DataFrame
print(df1[['Embarked', 'Embarked_encoded']].head())
```

```
  Embarked  Embarked_encoded
0        S               2.0
2        S               2.0
3        S               2.0
4        S               2.0
5        S               2.0
```

```python
df1
```

```
     Survived  Pclass     Sex   Age  SibSp  Parch     Fare Embarked  \
0           0       3    male  22.0      1      0   7.2500        S
2           1       3  female  26.0      0      0   7.9250        S
3           1       1  female  35.0      1      0  53.1000        S
4           0       3    male  35.0      0      0   8.0500        S
5           0       1    male  54.0      0      0  51.8625        S
..        ...     ...     ...   ...    ...    ...      ...      ...
707         0       3  female  39.0      0      5  29.1250        Q
708         0       2    male  27.0      0      0  13.0000        S
709         1       1  female  19.0      0      0  30.0000        S
710         1       1    male  26.0      0      0  30.0000        C
711         0       3    male  32.0      0      0   7.7500        Q

     Sex_female  Sex_male  Embarked_encoded
0             0         1               2.0
2             1         0               2.0
3             1         0               2.0
4             0         1               2.0
5             0         1               2.0
..          ...       ...               ...
707           1         0               1.0
```

```
708           0         1               2.0
709           1         0               2.0
710           0         1               0.0
711           0         1               1.0

[617 rows x 11 columns]
```

```
df1.shape
```

```
(617, 11)
```

```
df2 = df1.drop(["Sex","Embarked"],axis = 1)
```

```
df2
```

```
     Survived  Pclass   Age  SibSp  Parch     Fare  Sex_female  
Sex_male  \
0           0       3  22.0      1      0   7.2500           0         
1
2           1       3  26.0      0      0   7.9250           1         
0
3           1       1  35.0      1      0  53.1000           1         
0
4           0       3  35.0      0      0   8.0500           0         
1
5           0       1  54.0      0      0  51.8625           0         
1
..        ...     ...   ...    ...    ...      ...         ...        .
..
707         0       3  39.0      0      5  29.1250           1         
0
708         0       2  27.0      0      0  13.0000           0         
1
709         1       1  19.0      0      0  30.0000           1         
0
710         1       1  26.0      0      0  30.0000           0         
1
711         0       3  32.0      0      0   7.7500           0         
1

     Embarked_encoded
0                 2.0
2                 2.0
3                 2.0
4                 2.0
5                 2.0
..                ...
707               1.0
708               2.0
709               2.0
710               0.0
```

```
711                1.0

[617 rows x 9 columns]
```

```python
df2.shape
```

```
(617, 9)
```

```python
X = df2.drop(["Survived"],axis = 1) # Independent Variable
```

```python
X
```

```
     Pclass   Age  SibSp  Parch     Fare  Sex_female  Sex_male  \
0         3  22.0      1      0   7.2500           0         1
2         3  26.0      0      0   7.9250           1         0
3         1  35.0      1      0  53.1000           1         0
4         3  35.0      0      0   8.0500           0         1
5         1  54.0      0      0  51.8625           0         1
..      ...   ...    ...    ...      ...         ...       ...
707       3  39.0      0      5  29.1250           1         0
708       2  27.0      0      0  13.0000           0         1
709       1  19.0      0      0  30.0000           1         0
710       1  26.0      0      0  30.0000           0         1
711       3  32.0      0      0   7.7500           0         1

     Embarked_encoded
0                 2.0
2                 2.0
3                 2.0
4                 2.0
5                 2.0
..                ...
707               1.0
708               2.0
709               2.0
710               0.0
711               1.0

[617 rows x 8 columns]
```

```python
y = df2[["Survived"]] # Dependent Variable
```

```python
y
```

```
     Survived
0           0
2           1
3           1
4           0
5           0
..        ...
707         0
```

```
708         0
709         1
710         1
711         0

[617 rows x 1 columns]
```

Apply the Standard Scaler

```
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)

X

array([[ 0.77887597, -0.48335538,  0.53751292, ..., -0.7002415 ,
         0.7002415 ,  0.45533969],
       [ 0.77887597, -0.20393369, -0.51199806, ...,  1.42807873,
        -1.42807873,  0.45533969],
       [-1.92093568,  0.42476514,  0.53751292, ...,  1.42807873,
        -1.42807873,  0.45533969],
       ...,
       [-1.92093568, -0.69292166, -0.51199806, ...,  1.42807873,
        -1.42807873,  0.45533969],
       [-1.92093568, -0.20393369, -0.51199806, ..., -0.7002415 ,
         0.7002415 , -2.38248441],
       [ 0.77887597,  0.21519886, -0.51199806, ..., -0.7002415 ,
         0.7002415 , -0.96357236]])

y

     Survived
0           0
2           1
3           1
4           0
5           0
..        ...
707         0
708         0
709         1
710         1
711         0

[617 rows x 1 columns]
```

Used the Deep learning model (It is ANN with binary classification problem)

Keras Tuner- Decide Number of Hidden Layers And Neuron In Neural Network

```
pip install -U keras-tuner

Requirement already satisfied: keras-tuner in c:\users\lenovo\
anaconda3\lib\site-packages (1.2.1)
Requirement already satisfied: ipython in c:\users\lenovo\anaconda3\
lib\site-packages (from keras-tuner) (8.2.0)
Requirement already satisfied: requests in c:\users\lenovo\anaconda3\
lib\site-packages (from keras-tuner) (2.27.1)
Requirement already satisfied: tensorflow>=2.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from keras-tuner) (2.11.0)
Requirement already satisfied: packaging in c:\users\lenovo\anaconda3\
lib\site-packages (from keras-tuner) (21.3)
Requirement already satisfied: kt-legacy in c:\users\lenovo\anaconda3\
lib\site-packages (from keras-tuner) (1.0.4)
Requirement already satisfied: tensorflow-intel==2.11.0 in c:\users\
lenovo\anaconda3\lib\site-packages (from tensorflow>=2.0->keras-tuner)
(2.11.0)
Requirement already satisfied: numpy>=1.20 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (1.21.5)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
c:\users\lenovo\anaconda3\lib\site-packages (from tensorflow-
intel==2.11.0->tensorflow>=2.0->keras-tuner) (0.29.0)
Requirement already satisfied: tensorboard<2.12,>=2.11 in c:\users\
lenovo\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (2.11.2)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (1.12.1)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (3.3.0)
Requirement already satisfied: setuptools in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (61.2.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (0.4.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\lenovo\
```

```
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (1.6.3)
Requirement already satisfied: libclang>=13.0.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (15.0.6.1)
Requirement already satisfied: tensorflow-estimator<2.12,>=2.11.0 in
c:\users\lenovo\anaconda3\lib\site-packages (from tensorflow-
intel==2.11.0->tensorflow>=2.0->keras-tuner) (2.11.0)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in c:\users\
lenovo\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (3.19.1)
Requirement already satisfied: h5py>=2.9.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (3.6.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (1.42.0)
Requirement already satisfied: keras<2.12,>=2.11.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (2.11.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (23.1.4)
Requirement already satisfied: six>=1.12.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (2.2.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\
lenovo\anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (4.1.1)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorflow-intel==2.11.0-
>tensorflow>=2.0->keras-tuner) (1.4.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorflow-
intel==2.11.0->tensorflow>=2.0->keras-tuner) (0.37.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0
in c:\users\lenovo\anaconda3\lib\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow>=2.0-
>keras-tuner) (0.6.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\users\
lenovo\anaconda3\lib\site-packages (from tensorboard<2.12,>=2.11-
>tensorflow-intel==2.11.0->tensorflow>=2.0->keras-tuner) (1.33.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\
users\lenovo\anaconda3\lib\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow>=2.0-
>keras-tuner) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in c:\users\lenovo\
```

```
anaconda3\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow>=2.0->keras-tuner) (3.3.4)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\
users\lenovo\anaconda3\lib\site-packages (from
tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow>=2.0-
>keras-tuner) (1.8.1)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\lenovo\
anaconda3\lib\site-packages (from tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow>=2.0->keras-tuner) (2.0.3)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\lenovo\
anaconda3\lib\site-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow>=2.0-
>keras-tuner) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in c:\users\
lenovo\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow>=2.0-
>keras-tuner) (4.2.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\
lenovo\anaconda3\lib\site-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.12,>=2.11->tensorflow-intel==2.11.0->tensorflow>=2.0-
>keras-tuner) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\
lenovo\anaconda3\lib\site-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow>=2.0->keras-tuner) (1.3.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\
lenovo\anaconda3\lib\site-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow>=2.0->keras-tuner) (0.4.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\
lenovo\anaconda3\lib\site-packages (from requests->keras-tuner)
(1.26.9)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\
lenovo\anaconda3\lib\site-packages (from requests->keras-tuner)
(2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\lenovo\
anaconda3\lib\site-packages (from requests->keras-tuner) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\lenovo\
anaconda3\lib\site-packages (from requests->keras-tuner) (3.3)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<0.5,>=0.4.1->tensorboard<2.12,>=2.11->tensorflow-
intel==2.11.0->tensorflow>=2.0->keras-tuner) (3.2.2)
Requirement already satisfied: colorama in c:\users\lenovo\anaconda3\
lib\site-packages (from ipython->keras-tuner) (0.4.4)
Requirement already satisfied: traitlets>=5 in c:\users\lenovo\
anaconda3\lib\site-packages (from ipython->keras-tuner) (5.1.1)
Requirement already satisfied: decorator in c:\users\lenovo\anaconda3\
lib\site-packages (from ipython->keras-tuner) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\lenovo\
```

```
anaconda3\lib\site-packages (from ipython->keras-tuner) (0.18.1)
Requirement already satisfied: backcall in c:\users\lenovo\anaconda3\
lib\site-packages (from ipython->keras-tuner) (0.2.0)
Requirement already satisfied: matplotlib-inline in c:\users\lenovo\
anaconda3\lib\site-packages (from ipython->keras-tuner) (0.1.2)
Requirement already satisfied: prompt-toolkit!=3.0.0,!
=3.0.1,<3.1.0,>=2.0.0 in c:\users\lenovo\anaconda3\lib\site-packages
(from ipython->keras-tuner) (3.0.20)
Requirement already satisfied: stack-data in c:\users\lenovo\
anaconda3\lib\site-packages (from ipython->keras-tuner) (0.2.0)
Requirement already satisfied: pygments>=2.4.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from ipython->keras-tuner) (2.11.2)
Requirement already satisfied: pickleshare in c:\users\lenovo\
anaconda3\lib\site-packages (from ipython->keras-tuner) (0.7.5)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\lenovo\
anaconda3\lib\site-packages (from jedi>=0.16->ipython->keras-tuner)
(0.8.3)
Requirement already satisfied: wcwidth in c:\users\lenovo\anaconda3\
lib\site-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0-
>ipython->keras-tuner) (0.2.5)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\
lenovo\anaconda3\lib\site-packages (from packaging->keras-tuner)
(3.0.4)
Requirement already satisfied: pure-eval in c:\users\lenovo\anaconda3\
lib\site-packages (from stack-data->ipython->keras-tuner) (0.2.2)
Requirement already satisfied: asttokens in c:\users\lenovo\anaconda3\
lib\site-packages (from stack-data->ipython->keras-tuner) (2.0.5)
Requirement already satisfied: executing in c:\users\lenovo\anaconda3\
lib\site-packages (from stack-data->ipython->keras-tuner) (0.8.3)
Note: you may need to restart the kernel to use updated packages.
```

```python
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```python
import tensorflow as tf
from tensorflow import keras
from kerastuner.tuners import RandomSearch
from kerastuner import HyperParameters
from tensorflow.keras import regularizers
from tensorflow.keras.callbacks import EarlyStopping

# Define your model using the Keras API:

def build_model(hp):
    model = keras.Sequential()
    model.add(keras.layers.Dense(units=hp.Int('units', min_value=32,
max_value=512, step=32), activation='relu',
kernel_initializer=hp.Choice('kernel_initializer',
values=['glorot_uniform', 'he_normal']), input_shape=(8,),
```

```
kernel_regularizer=regularizers.l2(hp.Choice('l2_regularization',
values=[0.001, 0.0001, 0.00001]))))
    model.add(keras.layers.Dense(units=hp.Int('units', min_value=32,
max_value=512, step=32), activation='relu',
kernel_initializer=hp.Choice('kernel_initializer',
values=['glorot_uniform', 'he_normal']),
kernel_regularizer=regularizers.l2(hp.Choice('l2_regularization',
values=[0.001, 0.0001, 0.00001]))))
    model.add(keras.layers.Dense(1, activation='sigmoid'))

model.compile(optimizer=keras.optimizers.Adam(hp.Choice('learning_rate
', values=[1e-2, 1e-3, 1e-4])), loss='binary_crossentropy',
metrics=['accuracy'])
    early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=5)
    return model, early_stop

# Instantiate a tuner object and define the search space:

tuner = RandomSearch(build_model, objective='val_accuracy',
max_trials=5, executions_per_trial=3, directory='my_dir',
project_name='helloworld')

# Search for the best hyperparameters:

tuner.search(x=X_train, y=y_train, epochs=10, validation_data=(X_val,
y_val))

# Retrieve the best hyperparameters and retrain the model:

best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
model, early_stop = tuner.hypermodel.build(best_hps)
history = model.fit(X_train, y_train, epochs=100,
validation_data=(X_val, y_val), callbacks=[early_stop])
```

```
INFO:tensorflow:Reloading Tuner from my_dir\helloworld\tuner0.json

C:\Users\Lenovo\AppData\Local\Temp\ipykernel_9960\353560542.py:3:
DeprecationWarning: `import kerastuner` is deprecated, please use
`import keras_tuner`.
  from kerastuner.tuners import RandomSearch

INFO:tensorflow:Oracle triggered exit
Epoch 1/100
16/16 [==============================] - 5s 40ms/step - loss: 0.9884 -
accuracy: 0.7525 - val_loss: 0.8862 - val_accuracy: 0.7742
Epoch 2/100
16/16 [==============================] - 0s 9ms/step - loss: 0.8174 -
accuracy: 0.8012 - val_loss: 0.8016 - val_accuracy: 0.7984
Epoch 3/100
```

```
16/16 [==============================] - 0s 9ms/step - loss: 0.7303 -
accuracy: 0.8195 - val_loss: 0.7278 - val_accuracy: 0.7984
Epoch 4/100
16/16 [==============================] - 0s 8ms/step - loss: 0.6630 -
accuracy: 0.8296 - val_loss: 0.7001 - val_accuracy: 0.7903
Epoch 5/100
16/16 [==============================] - 0s 7ms/step - loss: 0.6260 -
accuracy: 0.8357 - val_loss: 0.6729 - val_accuracy: 0.7984
Epoch 6/100
16/16 [==============================] - 0s 8ms/step - loss: 0.5914 -
accuracy: 0.8377 - val_loss: 0.6384 - val_accuracy: 0.7903
Epoch 7/100
16/16 [==============================] - 0s 10ms/step - loss: 0.5681 -
accuracy: 0.8377 - val_loss: 0.6230 - val_accuracy: 0.7984
Epoch 8/100
16/16 [==============================] - 0s 9ms/step - loss: 0.5429 -
accuracy: 0.8418 - val_loss: 0.6070 - val_accuracy: 0.8145
Epoch 9/100
16/16 [==============================] - 0s 8ms/step - loss: 0.5306 -
accuracy: 0.8438 - val_loss: 0.5870 - val_accuracy: 0.7823
Epoch 10/100
16/16 [==============================] - 0s 7ms/step - loss: 0.5127 -
accuracy: 0.8377 - val_loss: 0.5888 - val_accuracy: 0.7984
Epoch 11/100
16/16 [==============================] - 0s 9ms/step - loss: 0.4924 -
accuracy: 0.8418 - val_loss: 0.5798 - val_accuracy: 0.7984
Epoch 12/100
16/16 [==============================] - 0s 11ms/step - loss: 0.4817 -
accuracy: 0.8337 - val_loss: 0.5919 - val_accuracy: 0.7903
Epoch 13/100
16/16 [==============================] - 0s 11ms/step - loss: 0.4958 -
accuracy: 0.8276 - val_loss: 0.5653 - val_accuracy: 0.7903
Epoch 14/100
16/16 [==============================] - 0s 10ms/step - loss: 0.4646 -
accuracy: 0.8600 - val_loss: 0.5583 - val_accuracy: 0.8065
Epoch 15/100
16/16 [==============================] - 0s 10ms/step - loss: 0.4604 -
accuracy: 0.8519 - val_loss: 0.5609 - val_accuracy: 0.8226
Epoch 16/100
16/16 [==============================] - 0s 11ms/step - loss: 0.4656 -
accuracy: 0.8357 - val_loss: 0.5535 - val_accuracy: 0.7984
Epoch 17/100
16/16 [==============================] - 0s 9ms/step - loss: 0.4581 -
accuracy: 0.8377 - val_loss: 0.5464 - val_accuracy: 0.8065
Epoch 18/100
16/16 [==============================] - 0s 7ms/step - loss: 0.4442 -
accuracy: 0.8499 - val_loss: 0.5490 - val_accuracy: 0.7984
Epoch 19/100
16/16 [==============================] - 0s 11ms/step - loss: 0.4371 -
accuracy: 0.8418 - val_loss: 0.5425 - val_accuracy: 0.8065
```

```
Epoch 20/100
16/16 [==============================] - 0s 13ms/step - loss: 0.4255 -
accuracy: 0.8458 - val_loss: 0.5490 - val_accuracy: 0.7903
Epoch 21/100
16/16 [==============================] - 0s 13ms/step - loss: 0.4247 -
accuracy: 0.8600 - val_loss: 0.5429 - val_accuracy: 0.7903
Epoch 22/100
16/16 [==============================] - 0s 11ms/step - loss: 0.4209 -
accuracy: 0.8458 - val_loss: 0.5418 - val_accuracy: 0.7984
Epoch 23/100
16/16 [==============================] - 0s 10ms/step - loss: 0.4165 -
accuracy: 0.8560 - val_loss: 0.5497 - val_accuracy: 0.7903
Epoch 24/100
16/16 [==============================] - 0s 17ms/step - loss: 0.4203 -
accuracy: 0.8621 - val_loss: 0.5305 - val_accuracy: 0.8226
Epoch 25/100
16/16 [==============================] - 0s 9ms/step - loss: 0.4114 -
accuracy: 0.8580 - val_loss: 0.5573 - val_accuracy: 0.7742
Epoch 26/100
16/16 [==============================] - 0s 9ms/step - loss: 0.4066 -
accuracy: 0.8458 - val_loss: 0.5308 - val_accuracy: 0.8145
Epoch 27/100
16/16 [==============================] - 0s 11ms/step - loss: 0.4210 -
accuracy: 0.8418 - val_loss: 0.5984 - val_accuracy: 0.7419
Epoch 28/100
16/16 [==============================] - 0s 13ms/step - loss: 0.4156 -
accuracy: 0.8479 - val_loss: 0.5313 - val_accuracy: 0.7903
Epoch 29/100
16/16 [==============================] - 0s 8ms/step - loss: 0.4050 -
accuracy: 0.8499 - val_loss: 0.5290 - val_accuracy: 0.8065
Epoch 30/100
16/16 [==============================] - 0s 8ms/step - loss: 0.4087 -
accuracy: 0.8600 - val_loss: 0.5469 - val_accuracy: 0.7742
Epoch 31/100
16/16 [==============================] - 0s 8ms/step - loss: 0.4050 -
accuracy: 0.8641 - val_loss: 0.5308 - val_accuracy: 0.8145
Epoch 32/100
16/16 [==============================] - 0s 9ms/step - loss: 0.3985 -
accuracy: 0.8621 - val_loss: 0.5566 - val_accuracy: 0.7661
Epoch 33/100
16/16 [==============================] - 0s 8ms/step - loss: 0.3924 -
accuracy: 0.8661 - val_loss: 0.5276 - val_accuracy: 0.8065
Epoch 34/100
16/16 [==============================] - 0s 8ms/step - loss: 0.3996 -
accuracy: 0.8641 - val_loss: 0.5454 - val_accuracy: 0.7903
Epoch 35/100
16/16 [==============================] - 0s 7ms/step - loss: 0.3906 -
accuracy: 0.8641 - val_loss: 0.5707 - val_accuracy: 0.7661
Epoch 36/100
16/16 [==============================] - 0s 7ms/step - loss: 0.3956 -
```

```
accuracy: 0.8540 - val_loss: 0.5400 - val_accuracy: 0.8065
Epoch 37/100
16/16 [==============================] - 0s 9ms/step - loss: 0.3877 -
accuracy: 0.8540 - val_loss: 0.5388 - val_accuracy: 0.7903
Epoch 38/100
16/16 [==============================] - 0s 9ms/step - loss: 0.3920 -
accuracy: 0.8621 - val_loss: 0.5587 - val_accuracy: 0.7823
```

```
# Evaluate the model on the test set and print the test accuracy:
test_loss, test_acc = model.evaluate(X_val, y_val)
print('Test accuracy:', test_acc)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.5587 -
accuracy: 0.7823
Test accuracy: 0.7822580933570862
```

```
# Evaluate the model on the test set and print the test accuracy:
train_loss, train_acc = model.evaluate(X_train, y_train)
print('Train accuracy:', train_acc)
```

```
16/16 [==============================] - 0s 2ms/step - loss: 0.3873 -
accuracy: 0.8580
Train accuracy: 0.8580121994018555
```

Que.3 Create a model to perform binary classification between horse and human images using convolutional neural networks. Dataset available in Tensorflow datasets

```
# install kaggle
!pip install -q kaggle
```

```
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>

Saving kaggle.json to kaggle.json

{'kaggle.json':
b'{"username":"saurabhmahadevpalve","key":"d8edaa6801661ee546bca299e87
7cd0e"}'}
```

```
# Create a kaggle folder
! mkdir ~/.kaggle
```

```
# Copy the kaggle .json to folder created
! cp kaggle.json ~/.kaggle/
```

```
# permission for the json to act(read and write)
! chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d sanikamal/horses-or-humans-dataset
```

```
Downloading horses-or-humans-dataset.zip to /content
100% 307M/307M [00:15<00:00, 24.3MB/s]
100% 307M/307M [00:15<00:00, 21.3MB/s]
```

```
import zipfile
zip_ref = zipfile.ZipFile('/content/horses-or-humans-dataset.zip',
'r')
zip_ref.extractall('/content')
zip_ref.close()
```

```
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import
Dense,Conv2D,MaxPooling2D,Flatten,BatchNormalization,Dropout
```

```
# generators
train_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/horse-or-human/train',
    labels='inferred',
    label_mode = 'int',
```

```python
    batch_size=32,
    image_size=(256,256)
)

validation_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/horse-or-human/validation',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)
```

```
Found 1027 files belonging to 2 classes.
Found 256 files belonging to 2 classes.
```

```python
# Normalize
def process(image,label):
    image = tf.cast(image/255. ,tf.float32)
    return image,label

train_ds = train_ds.map(process)
validation_ds = validation_ds.map(process)

# create CNN model

model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu
',input_shape=(256,256,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(64,kernel_size=(3,3),padding='valid',activation='relu
'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activation='rel
u'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))
```

```
model.summary()
```

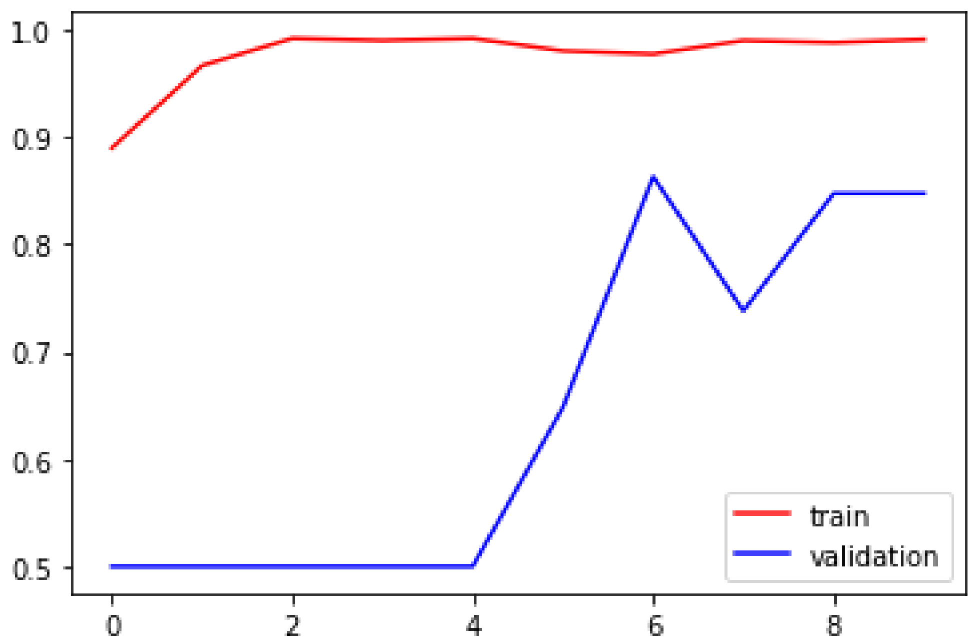
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 batch_normalization (BatchN  (None, 254, 254, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 batch_normalization_1 (Batc  (None, 125, 125, 64)     256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 batch_normalization_2 (Batc  (None, 60, 60, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 128)               14745728

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 14,848,193
Trainable params: 14,847,745
Non-trainable params: 448
_________________________________________________________________
```

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['ac
curacy'])
```

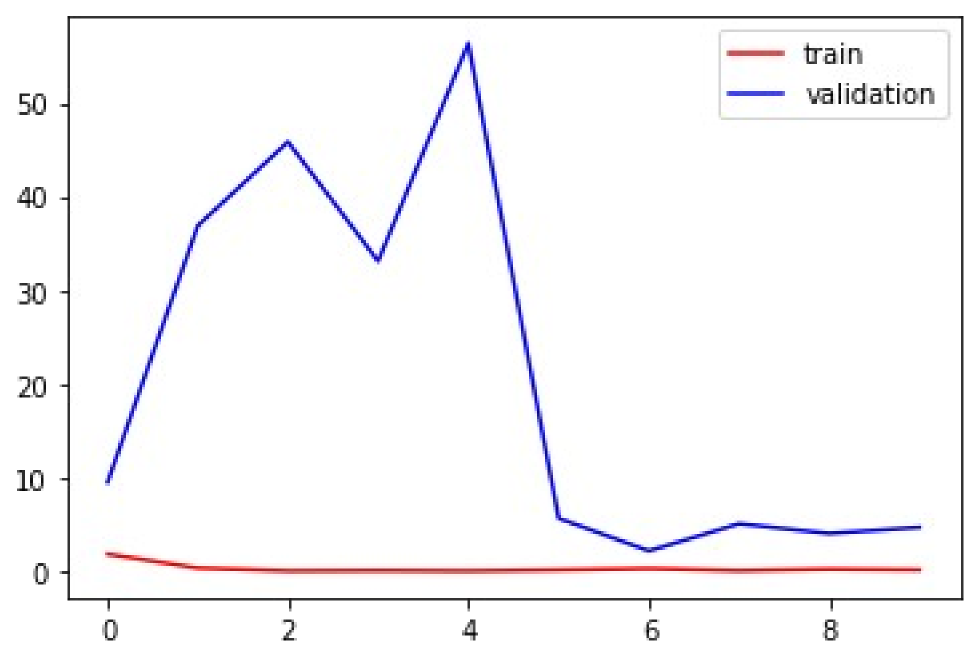
```
history = model.fit(train_ds,epochs=10,validation_data=validation_ds)
```

```
Epoch 1/10
33/33 [==============================] - 20s 162ms/step - loss: 1.8630
- accuracy: 0.8900 - val_loss: 9.5355 - val_accuracy: 0.5000
Epoch 2/10
33/33 [==============================] - 8s 207ms/step - loss: 0.3739
- accuracy: 0.9669 - val_loss: 36.8866 - val_accuracy: 0.5000
Epoch 3/10
33/33 [==============================] - 6s 157ms/step - loss: 0.0694
- accuracy: 0.9922 - val_loss: 45.8369 - val_accuracy: 0.5000
Epoch 4/10
33/33 [==============================] - 7s 173ms/step - loss: 0.1098
- accuracy: 0.9903 - val_loss: 33.1074 - val_accuracy: 0.5000
Epoch 5/10
33/33 [==============================] - 8s 202ms/step - loss: 0.0668
- accuracy: 0.9922 - val_loss: 56.3426 - val_accuracy: 0.5000
Epoch 6/10
33/33 [==============================] - 7s 177ms/step - loss: 0.1677
- accuracy: 0.9805 - val_loss: 5.6891 - val_accuracy: 0.6484
Epoch 7/10
33/33 [==============================] - 7s 179ms/step - loss: 0.3189
- accuracy: 0.9776 - val_loss: 2.2067 - val_accuracy: 0.8633
Epoch 8/10
33/33 [==============================] - 7s 153ms/step - loss: 0.0968
- accuracy: 0.9903 - val_loss: 5.0993 - val_accuracy: 0.7383
Epoch 9/10
33/33 [==============================] - 7s 193ms/step - loss: 0.2526
- accuracy: 0.9883 - val_loss: 4.0692 - val_accuracy: 0.8477
Epoch 10/10
33/33 [==============================] - 7s 179ms/step - loss: 0.1730
- accuracy: 0.9912 - val_loss: 4.7185 - val_accuracy: 0.8477
```

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validatio
n')
plt.legend()
plt.show()
```




```
plt.plot(history.history['loss'],color='red',label='train')
plt.plot(history.history['val_loss'],color='blue',label='validation')
plt.legend()
plt.show()
```



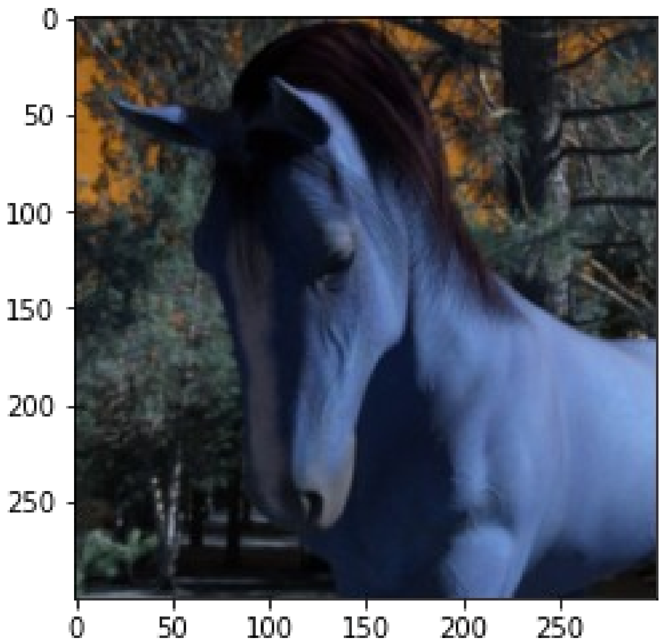

```
# ways to reduce overfitting

# Add more data
# Data Augmentation -> next video
# L1/L2 Regularizer
```

```
# Dropout
# Batch Norm
# Reduce complexity

import cv2

test_img = cv2.imread('/content/horse-or-human/train/horses/horse49-
4.png')

test_img

array([[[110, 107, 102],
        [ 94,  90,  83],
        [ 67,  66,  62],
        ...,
        [ 38,  45,  47],
        [ 48,  53,  54],
        [ 52,  59,  56]],

       [[ 79,  78,  77],
        [ 90,  88,  84],
        [ 80,  74,  66],
        ...,
        [ 37,  43,  45],
        [ 48,  53,  53],
        [ 54,  60,  58]],

       [[ 60,  63,  63],
        [ 69,  66,  63],
        [ 78,  68,  60],
        ...,
        [ 34,  40,  43],
        [ 46,  51,  50],
        [ 51,  56,  54]],

       ...,

       [[ 57,  80,  71],
        [ 59,  85,  77],
        [ 66,  94,  85],
        ...,
        [ 23,  20,  18],
        [ 23,  20,  18],
        [ 22,  19,  18]],

       [[ 63,  87,  77],
        [ 64,  90,  81],
        [ 64,  89,  82],
        ...,
        [ 24,  21,  19],
        [ 23,  21,  19],
```

```
        [ 23,  20,  19]],

       [[ 85, 105, 106],
        [ 89, 110, 110],
        [ 88, 108, 110],
        ...,
        [ 22,  19,  18],
        [ 21,  19,  17],
        [ 21,  19,  17]]], dtype=uint8)
```

```
plt.imshow(test_img)
```

```
<matplotlib.image.AxesImage at 0x7f9911830700>
```




```
test_img.shape
```

```
(300, 300, 3)
```

```
test_img = cv2.resize(test_img,(256,256))
```

```
test_input = test_img.reshape((1,256,256,3))
```

```
model.predict(test_input)
```

```
1/1 [==============================] - 0s 22ms/step
```

```
array([[1.]], dtype=float32)
```

Ans : When we insert the horse image then model will predict as 1 and human image then model will predict as 0.