Data preprocessing

In [1]:

```
#first we will clean the data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
train=pd.read_csv(r'C:\Users\Pranav\Desktop\Prachi\titanic-dataset\titanic\train.csv')
test=pd.read_csv(r'C:\Users\Pranav\Desktop\Prachi\titanic-dataset\titanic\test.csv')
gender=pd.read_csv(r'C:\Users\Pranav\Desktop\Prachi\titanic-dataset\titanic\gender_submission.csv')
```

In [3]:

```
train.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
```

In [4]:

```
row=train.shape[0]
r_null=train.isnull().sum()
r_null
```

Out[4]:

```
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
```

In [5]:

```
r_null/row*100
```

Out[5]:

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

In [6]:

```
train=train.drop(columns=['Cabin'],axis=1) #to drop null values columns #axis=1 is for columns 0 is for rows
```

In [7]:

```
train['Age']=train['Age'].fillna(train['Age'].mean())  #filling the null values by mean value of age column
```

In [8]:

```
train['Embarked']=train['Embarked'].fillna(train['Embarked'].mode()[0])# using mode function to fill categorical values
```

In [9]:

```
train.isnull().sum() # dataset is fully cleaned with no null values
```

Out[9]:

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

In [10]:

```
#now cleaning test data
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    object 
 4   Sex          418 non-null    object 
 5   Age          332 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    object 
 9   Fare         417 non-null    float64
 10  Cabin        91 non-null     object 
 11  Embarked     418 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 39.3+ KB
```

In [11]:

```
row=test.shape[0]
r_null=test.isnull().sum()
r_null
```

Out[11]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [12]:

```
test=test.drop(columns=['Cabin'],axis=1) #to drop null values columns #axis=1 is for columns 0 is for rows
```

In [13]:

```
test['Age']=test['Age'].fillna(test['Age'].mean())  #filling the null values by mean value of age column
```

In [14]:

```
test['Fare']=train['Fare'].fillna(train['Fare'].mean())
```

In [15]:

```
r_null/row*100 # test data is cleaned
```

Out[15]:

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            20.574163
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.239234
Cabin          78.229665
Embarked        0.000000
dtype: float64
```

In [16]:

```
gender.info() # gender has no null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Survived     418 non-null    int64
dtypes: int64(2)
memory usage: 6.7 KB
```

In [2]:

```
train.info()
```

In [17]:

```
#Now the data preprocessing starts- We are converting categorical data into numerical data by labelencoding
from sklearn.preprocessing import LabelEncoder #sklearn is a library, label encoder is a function of sklearn
le=LabelEncoder()
object_list=train.select_dtypes(include=['object']).columns
for i in object_list:
    train[i]=le.fit_transform(train[i])#converting string data into numerical
```

In [18]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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    int32  
 4   Sex          891 non-null    int32  
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    int32  
 9   Fare         891 non-null    float64
 10  Embarked     891 non-null    int32  
dtypes: float64(2), int32(4), int64(5)
memory usage: 62.8 KB
```

In [19]:

```
#for test dataset
from sklearn.preprocessing import LabelEncoder #sklearn is a library, label encoder is a function of sklearn
le=LabelEncoder()
object_list=test.select_dtypes(include=['object']).columns
for i in object_list:
    test[i]=le.fit_transform(test[i])
```

In [20]:

```
test.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   Survived     418 non-null    int64  
 2   Pclass       418 non-null    int64  
 3   Name         418 non-null    int32  
 4   Sex          418 non-null    int32  
 5   Age          418 non-null    float64
 6   SibSp        418 non-null    int64  
 7   Parch        418 non-null    int64  
 8   Ticket       418 non-null    int32  
 9   Fare         418 non-null    float64
 10  Embarked     418 non-null    int32  
dtypes: float64(2), int32(4), int64(5)
memory usage: 29.5 KB
```

In [21]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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    int32  
 4   Sex          891 non-null    int32  
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    int32  
 9   Fare         891 non-null    float64
 10  Embarked     891 non-null    int32  
dtypes: float64(2), int32(4), int64(5)
memory usage: 62.8 KB
```

In [25]:

```
#slicing data(we use slicing method for the target variable "survived"
#column separated from the train data as well as test data)
x_train=train.iloc[:,2:]     # select all rows but exclude column 1 "survived"
y_train=train.iloc[:,:2] #[rows , columns][: means all rows included and :2 means 0:2 2
#columns which is 0 and 1 excluding 2nd column]
x_test=test.iloc[:,2:]
y_test=test.iloc[:,:2]
```

In [26]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train) #will transform column according to model
x_test=sc.transform(x_test) #will transform model according to column
```

In [27]:

```
x_train
```

Out[27]:

```
array([[ 0.82737724, -1.31021659,  0.73769513, ...,  0.91896631,
        -0.50244517,  0.58595414],
       [-1.56610693, -0.99141018, -1.35557354, ...,  1.28262456,
         0.78684529, -1.9423032 ],
       [ 0.82737724, -0.35768524, -1.35557354, ...,  1.64628282,
        -0.48885426,  0.58595414],
       ...,
       [ 0.82737724, -0.12441226, -1.35557354, ...,  1.67617254,
        -0.17626324,  0.58595414],
       [-1.56610693, -1.41518943,  0.73769513, ..., -1.64656796,
        -0.04438104, -1.9423032 ],
       [ 0.82737724, -0.87477369,  0.73769513, ...,  0.63501397,
        -0.49237783, -0.67817453]])
```

In [28]:

```
x_test
```

Out[28]:

```
array([[ 0.82737724, -0.92920405,  0.73769513, ..., -0.92921469,
        -0.50244517, -0.67817453],
       [ 0.82737724, -0.16329109, -1.35557354, ..., -0.58548291,
         0.78684529,  0.58595414],
       [-0.36936484, -0.68426742,  0.73769513, ..., -1.32276267,
        -0.48885426, -0.67817453],
       ...,
       [ 0.82737724, -0.43933078,  0.73769513, ...,  0.03721958,
        -0.48633742,  0.58595414],
       [ 0.82737724, -0.23716087,  0.73769513, ..., -0.59046453,
         0.00595568,  0.58595414],
       [ 0.82737724, -0.55596728,  0.73769513, ..., -1.16335083,
        -0.38667072, -1.9423032 ]])
```

In [29]:

```
y_train
```

Out[29]:

	PassengerId	Survived
0	1	0
1	2	1
2	3	1
3	4	1
4	5	0
...	...	...
886	887	0
887	888	1
888	889	0
889	890	1
890	891	0

891 rows × 2 columns

In [57]:

```
y_test
```

Out[57]:

	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	3	206	1	34.50000	0	0	152	7.2500	1
1	3	403	0	47.00000	1	0	221	71.2833	2
2	2	269	1	62.00000	0	0	73	7.9250	1
3	3	408	1	27.00000	0	0	147	53.1000	2
4	3	178	0	22.00000	1	1	138	8.0500	2
...	...	...	...	...	...	...	...	...	...
413	3	353	1	30.27259	0	0	267	0.0000	2
414	1	283	0	39.00000	0	0	324	7.9250	0
415	3	332	1	38.50000	0	0	346	8.0500	2
416	3	384	1	30.27259	0	0	220	32.5000	2
417	3	302	1	30.27259	1	1	105	13.0000	0

418 rows × 9 columns

In [30]:

```
y_train=y_train.drop(columns=['PassengerId'],axis=1)
```

In [31]:

```
y_test=y_test.drop(columns=['PassengerId'],axis=1)
```

In [33]:

```
y_train  #total rows in y_train should match y_test
```

Out[33]:

	Survived
0	0
1	1
2	1
3	1
4	0
...	...
886	0
887	1
888	0
889	1
890	0

891 rows × 1 columns

In [34]:

```
#from sklearn.model_selection import train_test_split            #use when data is 100%
#x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,range_state+0)
```

In [35]:

```
#preparing model to check the dependency of survival on all the columns
from sklearn.linear_model import Ridge, Lasso, LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor

list_algo = [LinearRegression(), Ridge(), Lasso(), KNeighborsRegressor(), DecisionTreeRegressor()]

for algo in list_algo:
    model=algo
    model.fit(x_train, y_train)
    y_pred=model.predict(x_test)
    print(f'The score of {algo} is {algo.score(x_test,y_test)*100}%')
```

```
The score of LinearRegression() is 66.99864409093249%
The score of Ridge() is 66.96051313044707%
The score of Lasso() is -0.17636684303348193%
The score of KNeighborsRegressor() is 43.51127819548872%
The score of DecisionTreeRegressor() is -19.924812030075188%
```

In [ ]:

In [ ]:

In [ ]: