Titanic - Machine Learning from Disaster

December 21, 2023

Initially importing usefull liraries

```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
```

Importing training data set in a variable

```
[2]: titanic_train = pd.read_csv(r'C:\Users\Bikash shah\Desktop\titanic\train.csv')
```

```
[3]: titanic_train
```

```
[3]:      PassengerId  Survived  Pclass  \
     0              1         0       3
     1              2         1       1
     2              3         1       3
     3              4         1       1
     4              5         0       3
     ..             …       …     …
     886          887         0       2
     887          888         1       1
     888          889         0       3
     889          890         1       1
     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
     ..                                                   …     …   …    …
     886                              Montvila, Rev. Juozas    male  27.0      0
     887                       Graham, Miss. Margaret Edith  female  19.0      0
     888           Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1
     889                              Behr, Mr. Karl Howell    male  26.0      0
     890                                Dooley, Mr. Patrick    male  32.0      0

          Parch            Ticket     Fare Cabin Embarked
```

```
0        0          A/5 21171   7.2500   NaN        S
1        0           PC 17599  71.2833   C85        C
2        0   STON/O2. 3101282   7.9250   NaN        S
3        0             113803  53.1000  C123        S
4        0             373450   8.0500   NaN        S
..      …              …      …   …    …
886      0             211536  13.0000   NaN        S
887      0             112053  30.0000   B42        S
888      2         W./C. 6607  23.4500   NaN        S
889      0             111369  30.0000  C148        C
890      0             370376   7.7500   NaN        Q

[891 rows x 12 columns]
```

```
[4]: titanic_train.head()
```

```
[4]:    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
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 Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

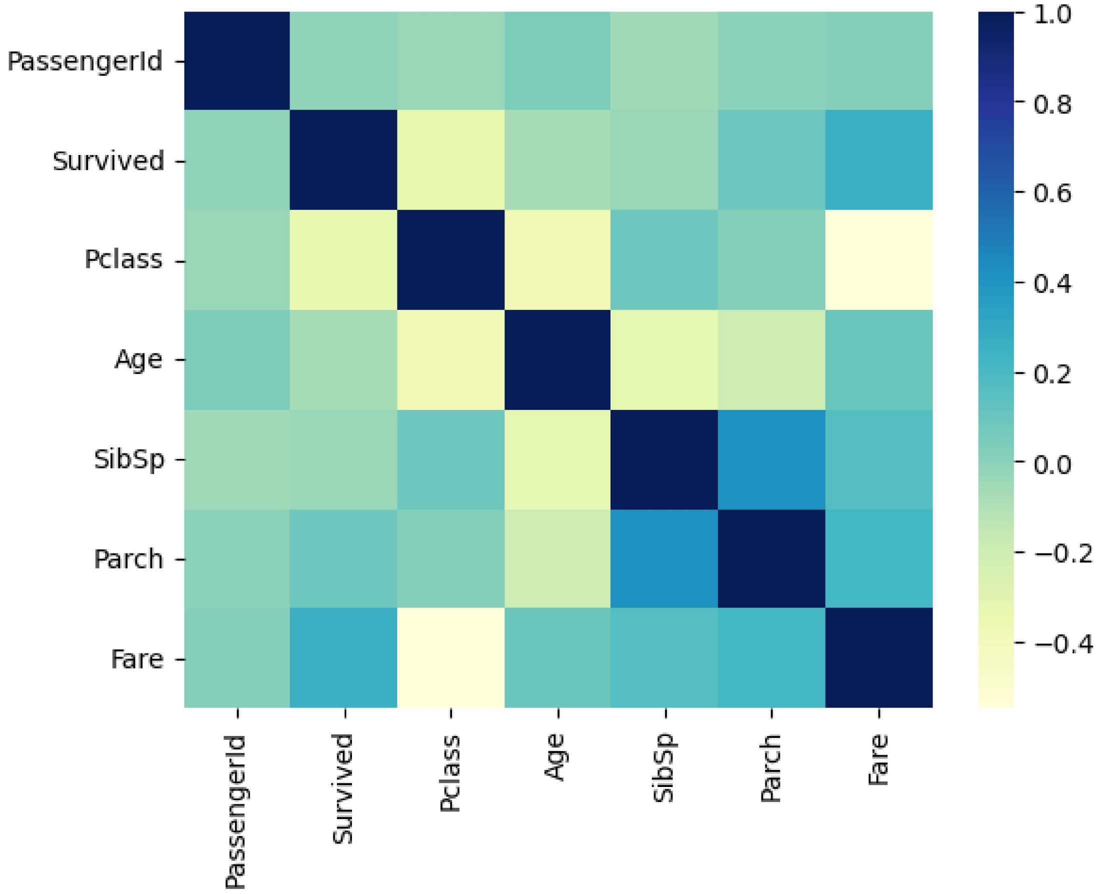
Looking for a correlation among the variables at the datasets

```
[5]: import seaborn as sns

     sns.heatmap(titanic_train.corr(), cmap='YlGnBu')
     plt.show()
```

```
C:\Users\Bikash shah\AppData\Local\Temp\ipykernel_8012\1610899979.py:3:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
```

```
sns.heatmap(titanic_train.corr(), cmap='YlGnBu')
```



Spliting this training data sets into two data set which is training and testing with the help of StratifiedShuffleSplit in the same ratio of survived, pclass and sex variables.

[6]:
```
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2)
for train_indicates, test_indicates in split.split(titanic_train,␣
 ↪titanic_train[['Survived', 'Pclass', 'Sex']]):
    strat_train_set = titanic_train.loc[train_indicates]
    strat_test_set = titanic_train.loc[test_indicates]
```

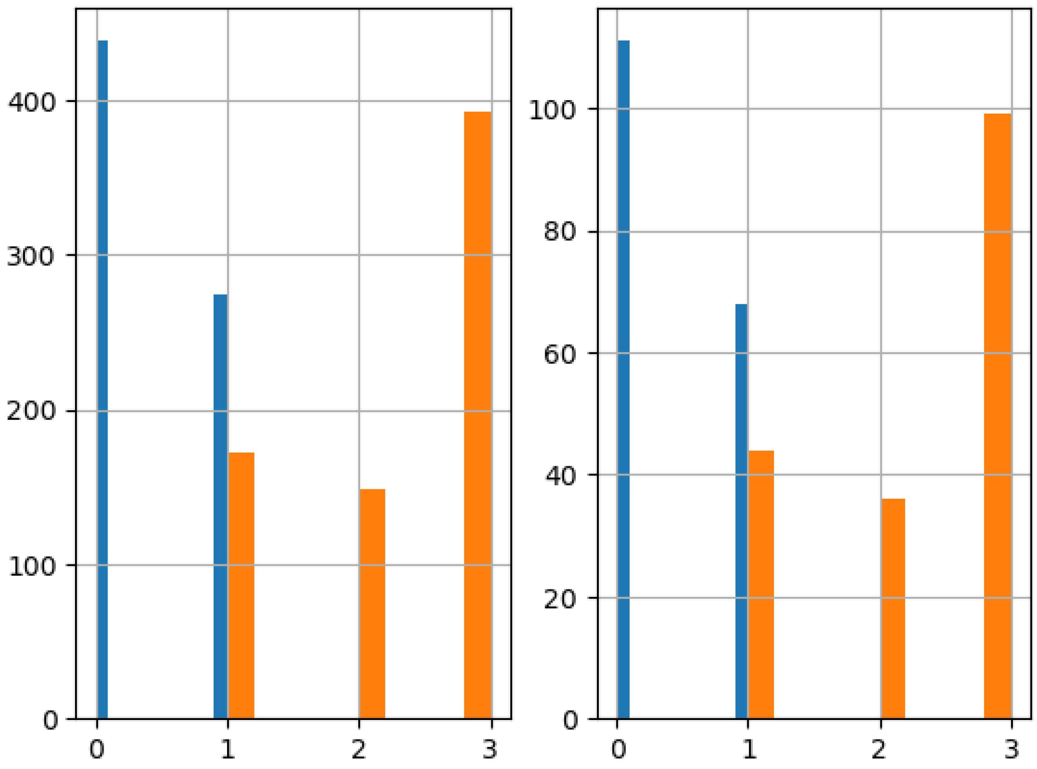
Looking in histogram is the data set distributed rationally in both training and testing data sets.

[7]:
```
plt.subplot(1,2,1)
strat_train_set['Survived'].hist()
strat_train_set['Pclass'].hist()

plt.subplot(1,2,2)
```

```
strat_test_set['Survived'].hist()
strat_test_set['Pclass'].hist()

plt.show()
```



We will look for the information about having null value or not.

[8]:
```
strat_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 444 to 171
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  712 non-null    int64
 1   Survived     712 non-null    int64
 2   Pclass       712 non-null    int64
 3   Name         712 non-null    object
 4   Sex          712 non-null    object
 5   Age          570 non-null    float64
 6   SibSp        712 non-null    int64
 7   Parch        712 non-null    int64
 8   Ticket       712 non-null    object
 9   Fare         712 non-null    float64
 10  Cabin        161 non-null    object
 11  Embarked     710 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 72.3+ KB
```

1 Creating a pipeline for cleaning the data set.

We will create a class for adding mean value at Age variable for the null value.

[9]:

```
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.impute import SimpleImputer

class AgeImputer(BaseEstimator, TransformerMixin):

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        imputer = SimpleImputer(strategy = 'mean')
        X['Age'] = imputer.fit_transform(X[['Age']])
        return X
```

We will create a class for changing the categorical data into binary data which 0 and 1.

[10]:

```
from sklearn.preprocessing import OneHotEncoder

class FeatureEncoder(BaseEstimator, TransformerMixin):

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        encoder = OneHotEncoder()
        matrix = encoder.fit_transform(X[['Embarked']]).toarray()

        colume_names = ['C', 'S', 'Q', 'N']

        for i in range(len(matrix.T)):
            X[colume_names[i]] = matrix.T[i]

        matrix = encoder.fit_transform(X[['Sex']]).toarray()

        colume_names = ['Female', 'Male']

        for i in range(len(matrix.T)):
            X[colume_names[i]] = matrix.T[i]
        return X
```

We will create a class for removing the variable which we are not going to need.

[11]:
```
class FeatureDropper(BaseEstimator, TransformerMixin):

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X.drop(['Embarked', 'Name', 'Ticket', 'Cabin', 'Sex', 'N'],␣
 ↪axis=1, errors = 'ignore')
```

We have used pipeline to create a pipeline to make data set into numeric data set which will help us predicting more accurately.

[12]:
```
from sklearn.pipeline import Pipeline

pipeline = Pipeline([('ageimputer', AgeImputer()),
                    ('featureencoder', FeatureEncoder()),
                    ('featuredropper', FeatureDropper())])
```

Changing the training data set into numeric data set.

[13]:
```
strat_train_set = pipeline.fit_transform(strat_train_set)
```

[14]:
```
strat_train_set.head()
```

[14]:
```
     PassengerId  Survived  Pclass     Age  SibSp  Parch     Fare    C    S  \
444          445         1       3  29.581      0      0   8.1125  0.0  0.0
592          593         0       3  47.000      0      0   7.2500  0.0  0.0
580          581         1       2  25.000      1      1  30.0000  0.0  0.0
46            47         0       3  29.581      1      0  15.5000  0.0  1.0
852          853         0       3   9.000      1      1  15.2458  1.0  0.0

       Q  Female  Male
444  1.0     0.0   1.0
592  1.0     0.0   1.0
580  1.0     1.0   0.0
46   0.0     0.0   1.0
852  0.0     1.0   0.0
```

[15]:
```
strat_train_set.describe()
```

[15]:
```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   712.000000  712.000000  712.000000  712.000000  712.000000
mean    445.676966    0.384831    2.308989   29.581000    0.518258
std     259.409965    0.486897    0.835249   13.088319    1.086785
min       1.000000    0.000000    1.000000    0.420000    0.000000
```

```
25%     220.500000    0.000000    2.000000   22.000000    0.000000
50%     443.500000    0.000000    3.000000   29.581000    0.000000
75%     675.250000    1.000000    3.000000   35.000000    1.000000
max     890.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare           C           S           Q      Female  \
count  712.000000  712.000000  712.000000  712.000000  712.000000  712.000000
mean     0.356742   31.632689    0.181180    0.089888    0.726124    0.351124
std      0.764349   47.914117    0.385438    0.286222    0.446260    0.477657
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      0.000000    7.895800    0.000000    0.000000    0.000000    0.000000
50%      0.000000   14.454200    0.000000    0.000000    1.000000    0.000000
75%      0.000000   30.178100    0.000000    0.000000    1.000000    1.000000
max      5.000000  512.329200    1.000000    1.000000    1.000000    1.000000

             Male
count  712.000000
mean     0.648876
std      0.477657
min      0.000000
25%      0.000000
50%      1.000000
75%      1.000000
max      1.000000
```

[16]:
```
strat_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 444 to 171
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  712 non-null    int64
 1   Survived     712 non-null    int64
 2   Pclass       712 non-null    int64
 3   Age          712 non-null    float64
 4   SibSp        712 non-null    int64
 5   Parch        712 non-null    int64
 6   Fare         712 non-null    float64
 7   C            712 non-null    float64
 8   S            712 non-null    float64
 9   Q            712 non-null    float64
 10  Female       712 non-null    float64
 11  Male         712 non-null    float64
dtypes: float64(7), int64(5)
memory usage: 72.3 KB
```

Changing data set into data points for training data set.

[17]:

```
from sklearn.preprocessing import StandardScaler

X = strat_train_set.drop(['Survived'], axis=1)
y = strat_train_set['Survived']

scaler = StandardScaler()
X_data = scaler.fit_transform(X)
y_data = y.to_numpy()
```

Applying RandomForestClassifier model algorithm for predicting the testing set.

[18]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

clf = RandomForestClassifier()

param_grid = [{'n_estimators': [10, 100, 200, 500], 'max_depth': [None, 5, 10],␣
 ↪'min_samples_split':[2,3,4]}]

grid_search = GridSearchCV(clf, param_grid, cv=3, scoring='accuracy',␣
 ↪return_train_score=True)
grid_search.fit(X_data, y_data)
```

[18]:

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid=[{'max_depth': [None, 5, 10],
                          'min_samples_split': [2, 3, 4],
                          'n_estimators': [10, 100, 200, 500]}],
             return_train_score=True, scoring='accuracy')
```

[19]:

```
final_clf = grid_search.best_estimator_
```

[20]:

```
final_clf
```

[20]:

```
RandomForestClassifier(min_samples_split=4, n_estimators=500)
```

As we cans see that max depth the tree went up to is 10, mimimum split is 3, and estimator is 500. Now we are going to do same thing for testing data set.

[21]:

```
strat_test_set = pipeline.fit_transform(strat_test_set)
```

[22]:

```
X_test = strat_test_set.drop(['Survived'], axis=1)
y_test = strat_test_set['Survived']

scaler = StandardScaler()
X_data_test = scaler.fit_transform(X_test)
y_data_test = y_test.to_numpy()
```

[23]:

```
final_clf.score(X_data_test,y_data_test)
```

```
[23]: 0.7988826815642458
```

```
[24]: final_data = pipeline.fit_transform(titanic_train)
```

```
[25]: final_data.head()
```

```
[25]:    PassengerId  Survived  Pclass   Age  SibSp  Parch     Fare    C    S    Q  \
      0            1         0       3  22.0      1      0   7.2500  0.0  0.0  1.0
      1            2         1       1  38.0      1      0  71.2833  1.0  0.0  0.0
      2            3         1       3  26.0      0      0   7.9250  0.0  0.0  1.0
      3            4         1       1  35.0      1      0  53.1000  0.0  0.0  1.0
      4            5         0       3  35.0      0      0   8.0500  0.0  0.0  1.0

         Female  Male
      0     0.0   1.0
      1     1.0   0.0
      2     1.0   0.0
      3     1.0   0.0
      4     0.0   1.0
```

Now we are going to use the real whole training data for in the algorithm.

```
[26]: X_final = final_data.drop(['Survived'], axis=1)
      y_final = final_data['Survived']

      scaler = StandardScaler()
      X_data_final = scaler.fit_transform(X_final)
      y_data_final = y_final.to_numpy()
```

```
[27]: prod_clf = RandomForestClassifier()

      param_grid = [{'n_estimators': [10, 100, 200, 500], 'max_depth': [None, 5, 10],␣
        ↪'min_samples_split':[2,3,4]}]

      grid_search = GridSearchCV(prod_clf, param_grid, cv=3, scoring='accuracy',␣
        ↪return_train_score=True)
      grid_search.fit(X_data_final, y_data_final)
```

```
[27]: GridSearchCV(cv=3, estimator=RandomForestClassifier(),
                   param_grid=[{'max_depth': [None, 5, 10],
                                'min_samples_split': [2, 3, 4],
                                'n_estimators': [10, 100, 200, 500]}],
                   return_train_score=True, scoring='accuracy')
```

```
[28]: prod_final_clf = grid_search.best_estimator_
```

```
[29]: prod_final_clf
```

```
[29]: RandomForestClassifier(max_depth=5, min_samples_split=3, n_estimators=200)
```

With the help of model we have trained for training data set, we are going to predict the testing data by repeting the same process.

```
[30]: titanic_test = pd.read_csv(r'C:\Users\Bikash shah\Desktop\titanic\test.csv')
```

```
[31]: titanic_test
```

```
[31]:      PassengerId  Pclass                                          Name  \
      0            892       3                              Kelly, Mr. James
      1            893       3              Wilkes, Mrs. James (Ellen Needs)
      2            894       2                     Myles, Mr. Thomas Francis
      3            895       3                              Wirz, Mr. Albert
      4            896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)
      ..           …     …                                           …
      413         1305       3                            Spector, Mr. Woolf
      414         1306       1                  Oliva y Ocana, Dona. Fermina
      415         1307       3                  Saether, Mr. Simon Sivertsen
      416         1308       3                           Ware, Mr. Frederick
      417         1309       3                      Peter, Master. Michael J

              Sex   Age  SibSp  Parch              Ticket      Fare Cabin Embarked
      0      male  34.5      0      0              330911    7.8292   NaN        Q
      1    female  47.0      1      0              363272    7.0000   NaN        S
      2      male  62.0      0      0              240276    9.6875   NaN        Q
      3      male  27.0      0      0              315154    8.6625   NaN        S
      4    female  22.0      1      1             3101298   12.2875   NaN        S
      ..      …   …    …    …                 …       …   …       …
      413    male   NaN      0      0           A.5. 3236    8.0500   NaN        S
      414  female  39.0      0      0            PC 17758  108.9000  C105        C
      415    male  38.5      0      0  SOTON/O.Q. 3101262    7.2500   NaN        S
      416    male   NaN      0      0              359309    8.0500   NaN        S
      417    male   NaN      1      1                2668   22.3583   NaN        C

      [418 rows x 11 columns]
```

```
[32]: final_test_data = pipeline.fit_transform(titanic_test)
```

```
[33]: final_test_data
```

```
[33]:      PassengerId  Pclass       Age  SibSp  Parch     Fare    C    S    Q  \
      0            892       3  34.50000      0      0   7.8292  0.0  1.0  0.0
      1            893       3  47.00000      1      0   7.0000  0.0  0.0  1.0
      2            894       2  62.00000      0      0   9.6875  0.0  1.0  0.0
      3            895       3  27.00000      0      0   8.6625  0.0  0.0  1.0
      4            896       3  22.00000      1      1  12.2875  0.0  0.0  1.0
      ..           …     …       …    …    …      …  …  …  …
```

```
413         1305       3  30.27259      0       0    8.0500  0.0  0.0  1.0
414         1306       1  39.00000      0       0  108.9000  1.0  0.0  0.0
415         1307       3  38.50000      0       0    7.2500  0.0  0.0  1.0
416         1308       3  30.27259      0       0    8.0500  0.0  0.0  1.0
417         1309       3  30.27259      1       1   22.3583  1.0  0.0  0.0

     Female  Male
0       0.0   1.0
1       1.0   0.0
2       0.0   1.0
3       0.0   1.0
4       1.0   0.0
..      …    …
413     0.0   1.0
414     1.0   0.0
415     0.0   1.0
416     0.0   1.0
417     0.0   1.0

[418 rows x 11 columns]
```

[34]:

```
X_final_test = final_test_data
X_final_test = X_final_test.fillna(method='ffill')

scaler = StandardScaler()
X_data_final_test = scaler.fit_transform(X_final_test)
```

[35]:

```
X_data_final_test
```

[35]:

```
array([[-1.72791209,  0.87348191,  0.3349926 , …, -1.35067551,
        -0.75592895,  0.75592895],
       [-1.71962474,  0.87348191,  1.32553003, …,  0.74037028,
         1.32287566, -1.32287566],
       [-1.71133739, -0.31581919,  2.51417495, …, -1.35067551,
        -0.75592895,  0.75592895],
       …,
       [ 1.71133739,  0.87348191,  0.65196458, …,  0.74037028,
        -0.75592895,  0.75592895],
       [ 1.71962474,  0.87348191,  0.        , …,  0.74037028,
        -0.75592895,  0.75592895],
       [ 1.72791209,  0.87348191,  0.        , …, -1.35067551,
        -0.75592895,  0.75592895]])
```

Now we are going to predict the final test variable with the help of final classifier.

[36]:

```
predictions = prod_final_clf.predict(X_data_final_test)
```

[37]:

```
predictions
```

[37]:
```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0],
      dtype=int64)
```

[38]:
```
print(len(predictions))
```

```
418
```

We will write this predicted data into CSV file.

[100]:
```
final_df = pd.DataFrame(titanic_test['PassengerId'])
final_df['Survived'] = predictions
final_df.to_csv(r'C:\Users\Bikash shah\Desktop\titanic\prediction.csv',␣
 ↪index=False)
```

[ ]:
```

```