machine-learning-titanic-Copy1

February 21, 2024

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/titanic/train.csv
/kaggle/input/titanic/test.csv
/kaggle/input/titanic/gender_submission.csv
```

0.1 Loading Necessary Libraries

[2]:
```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.svm import SVC
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
```

0.2 Loading The Dataset

[3]:
```
df_train=pd.read_csv("/kaggle/input/titanic/train.csv")
df_test=pd.read_csv("/kaggle/input/titanic/test.csv")
```

0.3 Viewing The Top 10 Rows

[4]:
```
print(df_train.head())
print(df_test.head())
```

```
   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
   PassengerId  Pclass                                          Name     Sex  \
0          892       3                              Kelly, Mr. James    male
1          893       3              Wilkes, Mrs. James (Ellen Needs)  female
2          894       2                     Myles, Mr. Thomas Francis    male
3          895       3                              Wirz, Mr. Albert    male
4          896       3  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female

    Age  SibSp  Parch   Ticket     Fare Cabin Embarked
0  34.5      0      0   330911   7.8292   NaN        Q
1  47.0      1      0   363272   7.0000   NaN        S
2  62.0      0      0   240276   9.6875   NaN        Q
3  27.0      0      0   315154   8.6625   NaN        S
4  22.0      1      1  3101298  12.2875   NaN        S
```

0.4 Checking Unique Values Of Columns

[5]:
```
print(df_train.columns.unique())
print(df_test.columns.unique())
```

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

0.5 Getting Information On The Data

[6]:
```
print(df_test.info())
print(df_train.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   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    object
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
None
<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
None
```

0.6 Checking For Null Values

[7]:
```
print(df_train.isnull().sum())
print(df_test.isnull().sum())
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

[8]:
```
## So There Are Null Values In Age,Cabin And Embarked, So We Have To Fill Them␣
↪Accordingly..
```

[9]:
```
for i in df_train.columns:
    if df_train[i].isnull().sum()>0:
        if df_train[i].dtype=="O":
            df_train[i]=df_train[i].fillna(df_train[i].mode()[0])
        else:
```

```
            df_train[i]=df_train[i].fillna(df_train[i].mean())
df_train.isnull().sum()
```

[9]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

[10]:
```
for i in df_test.columns:
    if df_test[i].isnull().sum()>0:
        if df_test[i].dtype=="O":
            df_test[i]=df_test[i].fillna(df_test[i].mode()[0])
        else:
            df_test[i]=df_test[i].fillna(df_test[i].mean())
df_test.isnull().sum()
```

[10]:
```
PassengerId    0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

0.7 Doing Encoding For The Object-Type Columns

[11]:
```
df_train.set_index("PassengerId",inplace=True)
df_train['Embarked']=LabelEncoder().fit_transform(df_train['Embarked'])
df_train.drop(["Name","Ticket"],axis=1,inplace=True)
df_train['Cabin']=LabelEncoder().fit_transform(df_train['Cabin'])
df_train['Sex']=LabelEncoder().fit_transform(df_train['Sex'])
```

[12]:
```
## So we have Filled The Missing Values,Done With Encoding , We Proceed To Our Next Step.
## That is Making Machine Learning Model
```

0.8 We Chose Random Forest As Our Machine Learning MOdel

0.9 Training The Data:

[13]:
```
## Splitting The DAta In X and Y set
ytrain=df_train["Survived"]
df_train.drop("Survived",axis=1,inplace=True)
xtrain=df_train
xtrain.head()
```

[13]:

PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
1	3	1	22.0	1	0	7.2500	47	2
2	1	0	38.0	1	0	71.2833	81	0
3	3	0	26.0	0	0	7.9250	47	2
4	1	0	35.0	1	0	53.1000	55	2
5	3	1	35.0	0	0	8.0500	47	2

[14]:
```
df_test.set_index("PassengerId",inplace=True)
df_test.drop(["Name","Ticket"],axis=1,inplace=True)
df_test['Cabin']=LabelEncoder().fit_transform(df_test['Cabin'])
df_test['Sex']=LabelEncoder().fit_transform(df_test['Sex'])
df_test['Embarked']=LabelEncoder().fit_transform(df_test['Embarked'])
xtest=df_test
```

0.10 Fitting Our MAchine Learning Model

[15]:
```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_jobs=-1,oob_score=True)
model.fit(xtrain,ytrain)
y_pred=model.predict(xtest)
y_pred
```

[15]:
```
array([0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 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, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0,
```

```
1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 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, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 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, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1])
```

[16]:
```
y_output=pd.read_csv("/kaggle/input/titanic/test.csv")
```

[17]:
```
output = pd.DataFrame({'PassengerId': y_output.PassengerId, 'Survived': y_pred})
```

[18]:
```
output
```

[18]:

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	1
4	896	1
..	…	…
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	1

[418 rows x 2 columns]

0.11 Importing Our Test Results

[19]:
```
output.to_csv('submission.csv', index=False)
```

[ ]:

[ ]:

[ ]: