Titanic - Machine Learning from Disaster

In [147…
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
```

In [148…
```
# Import dataset
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
test_id = test["PassengerId"]
```

In [149…
```
# Data description
train.head()
```

Out[149]:

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

In [150…
```
test.head()
```

Out[150]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [151…
```
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
```

Comment: We can see that columns [Age], [Cabin], and [Embarked] have some missing values because the length of dataframe is 891.

We will select some certain columns to evaluate data

In [152…
```
# Remove unimportant columns in dataframe
def clean_data(df):

    # Drop columns "Name", "Ticket" and "Cabin"
    df = df.drop(columns=["Ticket", "Cabin", "Name"])

    # Replace missing values in "Fare" and "Age" column
    cols = ["Fare","Age"]
    for col in cols:
        df[col].fillna(round(df[col].mean(),3), inplace = True)

    # Assign missing values in column "Embarked" to "U" value
    df.Embarked.fillna("U", inplace=True)
    return df
```

```
train = clean_data(train)
test = clean_data(test)
```

In [153…
```
# Check the quantity of missing values
train['Embarked'].value_counts()["U"]
```

Out[153]:
```
2
```

In [154…
```
# Convert column "Sex" and "Embarked" to values
le = preprocessing.LabelEncoder()
cols = ["Sex","Embarked"]
for col in cols:
    train[col] = le.fit_transform(train[col])
    test[col] = le.transform(test[col])
    print(le.classes_)
train.head()
```

```
['female' 'male']
['C' 'Q' 'S' 'U']
```

Out[154]:

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

In [155…
```
# Extract "Survived" column as y, and the others as X
y = train["Survived"]
X = train.drop("Survived", axis = 1)

# Split training set into train and test set
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size = 0.5, random_state = 42)

# Using logistic regression model to fit into model with 0 randomness and limit 1000 iteration values
mdl = LogisticRegression(random_state=0, max_iter=1000)
mdl.fit(X_train,y_train)
```

Out[155]:
```
LogisticRegression(max_iter=1000, random_state=0)
```

In [156…
```
# Find predicted values based on test set
y_pred = mdl.predict(X_test)

# Double check by calculating accuracy_score, precision, recall, F1 score, and AUC for reflection
print("Accuracy:",accuracy_score(y_test,y_pred))
print("Precision:",precision_score(y_test, y_pred))
print("Recall:",recall_score(y_test,y_pred))
print("F1 Score:",f1_score(y_test,y_pred))
print("AUC:",roc_auc_score(y_test,y_pred))
```

```
Accuracy: 0.8071748878923767
Precision: 0.7888198757763976
Recall: 0.7094972067039106
F1 Score: 0.7470588235294118
AUC: 0.7910781913669367
```

Comment: According to these 5 values, we can see that in general, logistic regression model is kind of performing well in predicting the survival chance of passengers based on given dataset.

In [157…
```
# Using test set to predict chance of surviving for passengers.
titanic_pred = mdl.predict(test)

# Display the output
df = pd.DataFrame({"PassengerId":test_id.values,
                   "Survived":titanic_pred})
df
```

Out[157]:

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

418 rows × 2 columns