UCK 358E: Project #1

21600 - Monday

Due on April 30, 2023

Asst. Assoc. Prof. Dr. Barış Başpınar

Hüseyin Tutan* **- 110190021**

*tutan19@itu.edu.tr

Contents

1 Analysis of the Given Code

First, let's make the code executable and add some basic comment lines.

```
#!/usr/bin/env python3

# @Author: Huseyin Tutan

# Import the libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


# Import the data

data = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")

# Code Section

y = data["Survived"]
features = ["Pclass", "Sex", "SibSp", "Parch"]
X = pd.get_dummies(data[features])
X_test = pd.get_dummies(test[features])

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)

model.fit(X, y)
predictions = model.predict(X_test)
output = pd.DataFrame({'PassengerId': test.PassengerId, 'Survived': predictions})
output.to_csv('submission1.csv', index=False)
```

1.1 Control of Overfitting or Underfitting

Firstly, I planned to check is there any suitability problems. Then, I did a quick search on kaggle.com discussion forum and found that post. "Overfitting and underfitting the Titanic" I adapted the code to my own code for check our code's fitting status and our code last section became like this.

```
# read in the data
train_data = pd.read_csv('train.csv')
test_data  = pd.read_csv('test.csv')
solution   = pd.read_csv('submission1.csv')

# select some features
features = ["Pclass", "Sex", "SibSp", "Parch"]
X       = pd.get_dummies(train_data[features])
y       = train_data["Survived"]
final_X_test  = pd.get_dummies(test_data[features])

# perform the classification and the fit
classifier = RandomForestClassifier(random_state=4)
classifier.fit(X, y)

predictions = classifier.predict(final_X_test)
K_splits = 11

# calculate the scores
CV_scores = cross_val_score(classifier, X, y, cv=K_splits)

# Print Section & Visualize

print("The mean accuracy score of the train data is %.5f" % classifier.score(X, y))
print("The mean  cross-validation score is %.5f \pm %0.2f" % (CV_scores.mean(), CV_score
print("The test (i.e. leaderboard) score is %.5f" % accuracy_score(solution['Survived'],

print("The individual cross-validation scores are: \n",CV_scores)
print("The minimum cross-validation score is %.3f" % min(CV_scores))
print("The maximum cross-validation score is %.3f" % max(CV_scores))

cv = StratifiedKFold(n_splits=K_splits)
visualizer = CVScores(classifier, cv=cv, scoring='f1_weighted',size=(1200, 400))
visualizer.fit(X, y)
visualizer.show()
```

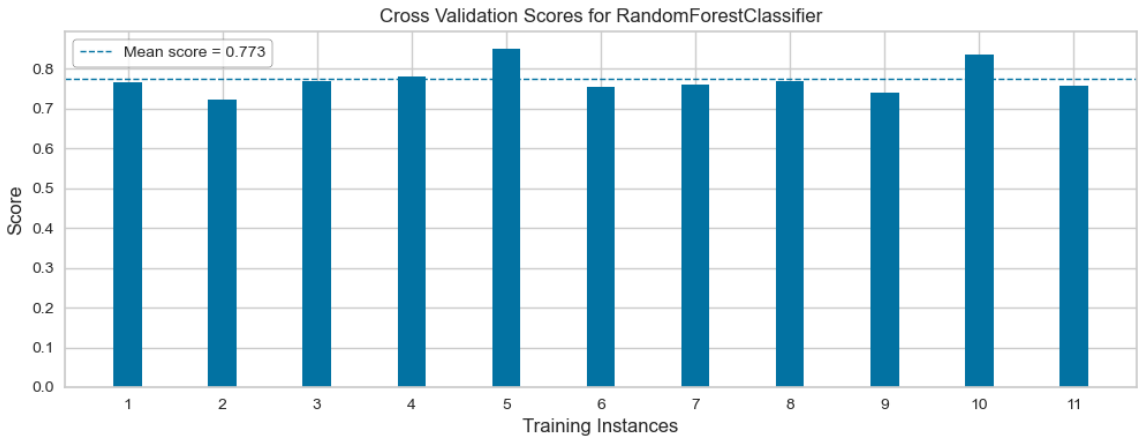
Also added these headers to code:

```
import colorama

from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from colorama import Fore
from sklearn.ensemble import RandomForestClassifier
from yellowbrick.model_selection import CVScores
from sklearn.model_selection import StratifiedKFold
```

Finally, the terminal output is as follows:

```
The mean accuracy score of the train data is 0.81706
The mean  cross-validation   score is 0.77890 \pm 0.07
The test (i.e. leaderboard)  score is 0.95215
The individual cross-validation scores are:
 [0.7654321  0.72839506 0.77777778 0.77777778 0.85185185 0.7654321
 0.7654321  0.77777778 0.75308642 0.83950617 0.7654321 ]
The minimum cross-validation score is 0.728
The maximum cross-validation score is 0.852
```



Direct Quote from web-site:
"We now find that the

$$CV(score) < LB(score)$$

That said, it is more usual to find that

$$CV(score) \approx LB(score)$$

up to the point of overfitting. Also note that the leaderboard score is lower when underfitting (0.66) than it was in the overfitting case above (0.76)."
So, we can say there is no over or underfitting issues in our code.

2 Data Preprocessing

In this section, after analyzing the code, I reported the different methods I tried in the preprocessing phase.

2.1 Missing Values

Firstly, I analyzed is there any missing values in our data with this code .

```
print(data.isna().sum())
```

Terminal output looks like below:

```
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
```

As seen in terminal, we have some missing values at Ages, Cabin and Embarked data. Afterwards, I completed the missing data with the following codes and the specified methods. I filled the age part with the mean method

```
data['Age'].fillna(data['Age'].mean(), inplace=True)
```

I filled the embarked part with the mod method

```
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
```

In cabin column there are so many missing values so I decided to do not complete it and do not use it for training my model.

2.2 Feature Engineering

I did research about some useful patterns for Titanic Challange. Below are some links that I used:
"Titanic - Advanced Feature Engineering Tutorial"
"Kaggle-Competition-Titanic Survival"
The attributes SibSp and Parch represent the count of siblings, spouses, parents, and children that were accompanying the passengers during the voyage. We can merge these attributes to create a new feature.

```
data['Age'].fillna(data['Age'].mean(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
test['Age'].fillna(test['Age'].mean(), inplace=True)
test['Embarked'].fillna(test['Embarked'].mode()[0], inplace=True)
```

Here is Kaggle.com scoreboard progress after this implementation.

3 Different Machine Learning Models

3.1 GradientBoostingClassifier

Added this code section then calculated the score on Kaggle.com

```
from sklearn.ensemble import GradientBoostingClassifier

model = GradientBoostingClassifier(n_estimators=100, max_depth=5, random_state=1)
```

Score: 0.76315

3.2 Support Vector Machine

```
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.svm import SVC

model = make_pipeline(StandardScaler(), SVC(kernel='rbf', C=1, gamma=0.1, random_state=1
```

Score: 0.78468

3.3 Naive Bayes

```
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
```

Score: 0.76076

3.4 Logistic Regression

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
```

Score: 0.75358

4 Hyperparameter Tuning for the Best Scored ML Method

Added n_jobs = -1 parameter to RandomForest model. It's mean "using all processors".
I did not found more useful parameters for RandomForest. Then, I Wrote a code for best parameter selection.

```
Parameter Selector

param_grid = {
    'n_estimators': [100, 300, 500],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'n_jobs': [-1],
    'random_state':[100, 500, 5000]
}

model = RandomForestClassifier(random_state=42)

grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5)

grid_search.fit(X, y)

print("Best parameters: ", grid_search.best_params_)
print("Best score: ", grid_search.best_score_))
```

After, these parameter tunings we reach the **0.78947** score.

5 Final Python Code

```
#!/usr/bin/env python3

# @Author: Huseyin Tutan 110190021

# IMPORT THE LIBRARIES

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import colorama

from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from colorama import Fore
from sklearn.ensemble import RandomForestClassifier
from yellowbrick.model_selection import CVScores
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV


# IMPORT THE DATA
data = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")

# MISS VALUES

data['Age'].fillna(data['Age'].mean(), inplace=True)
test['Age'].fillna(test['Age'].mean(), inplace=True)

data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
test['Embarked'].fillna(test['Embarked'].mode()[0], inplace=True)


```

```
# CODE SECTION

data2 = [data, test]
for dataset in data2:
    dataset['relatives'] = dataset['SibSp'] + dataset['Parch']
    dataset.loc[dataset['relatives'] > 0, 'travelled_alone'] = 'No'
    dataset.loc[dataset['relatives'] == 0, 'travelled_alone'] = 'Yes'

y = data["Survived"]
features = ["Pclass", "Sex", "Age", "Embarked", "travelled_alone"]
X = pd.get_dummies(data[features])
X_test = pd.get_dummies(test[features])

# PARAMETER SELECTOR

# param_grid = {
#     'n_estimators': [100, 300, 500],
#     'max_depth': [3, 4, 5],
#     'min_samples_split': [2, 5, 10],
#     'min_samples_leaf': [1, 2, 4],
#     'n_jobs': [-1],
#     'random_state':[0, 100, 500, 5000]
# }

# model = RandomForestClassifier(random_state=42)

# grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5)

# grid_search.fit(X, y)

# print("Best parameters: ", grid_search.best_params_)
# print("Best score: ", grid_search.best_score_)

model = RandomForestClassifier(n_estimators=100,
                               max_depth=4,
                               min_samples_split=10,
                               min_samples_leaf=2,
                               random_state=5000,
                               n_jobs=-1)
# model = GradientBoostingClassifier(n_estimators=100, max_depth=5, random_state=1)
# model = make_pipeline(StandardScaler(), SVC(kernel='rbf', C=1, gamma=0.1, random_state
# model = GaussianNB()
# model = LogisticRegression()
```

```
83
84
85 model.fit(X, y)
86 predictions = model.predict(X_test)
87 output = pd.DataFrame(
88     {'PassengerId': test.PassengerId, 'Survived': predictions})
89 output.to_csv('submission1.csv', index=False)
90
91 # COMPUTE CROSS-VALIDATION SCORE
92
93 scores = cross_val_score(model, X, y, cv=5)
94 print("Cross-validation scores:", scores)
95 print("Mean score:", scores.mean())
96
97 # COMPUTE ACCURACY SCORE
98
99 train_predictions = model.predict(X)
100 accuracy = accuracy_score(y, train_predictions)
101 print("Train accuracy:", accuracy)
```