uflxelqzu

June 4, 2023

1 PREDICTION OF THE WINNER OF A NBA GAME

We start by loading the libraries we will use in this exercise.

[42]:
```
#!/usr/bin/python3

from sys import argv, exit

import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
```

We load inputs and labels.

[43]:
```
inputs = np.load('inputs.npy')

inputs.shape
```

[43]:
```
(500, 60)
```

[44]:
```
labels = np.load('labels.npy')
labels.shape
```

[44]:
```
(500, 1)
```

We display inputs to see what they look like.

[45]:
```
print(inputs)
```

```
[[ 3.18523049  3.59592157 -3.17813934 …  3.12399465  4.52835403
   4.12871939]
 [-4.77291805  3.28182075 -0.34157396 …  0.1979122  -0.9100539
  -3.83402969]
 [-0.525279    4.20192714 -1.68276517 …  3.40499336  2.82181875
  -4.33408767]
 …
 [ 3.81445009  0.29948405  3.47920236 …  1.16673845  3.99212831
  -1.22771693]
 [ 0.21380738 -3.71846423  3.34507569 …  3.26112159 -3.22809925
  -0.15891529]
 [ 3.31576987  2.36659572 -0.02202948 …  2.51556703  4.31546946
   1.83445597]]
```

We separate inputs and labels into "train" and "test" in order to train our model with the train part and test the model with the test part. The size selected is 20% in the test.

```
[46]: X_train, X_test, y_train, y_test = train_test_split(inputs, labels, test_size=0.
       ↪2, random_state=42)
      X_train.shape
```

```
[46]: (400, 60)
```

```
[47]: logistic_params = {'C': [0.1, 1, 10], 'solver': ['liblinear', 'saga']}
      logistic_clf = LogisticRegression(random_state=0)
      logistic_grid_search = GridSearchCV(logistic_clf, logistic_params)
      logistic_grid_search.fit(X_train, y_train.ravel())

      logistic_best_params = logistic_grid_search.best_params_
      logistic_best_score = logistic_grid_search.best_score_

      print("Logistic Regression:")
      print("Best Parameters:", logistic_best_params)
      print("Best Score:", logistic_best_score)
```

```
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
```

```
Logistic Regression:
Best Parameters: {'C': 10, 'solver': 'saga'}
Best Score: 0.9099999999999999
```

```
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/linear_model/_sag.py:350:
ConvergenceWarning: The max_iter was reached which means the coef_ did not
converge
  warnings.warn(
```

With the logistic regression model we get a score of 0.9 which is good.

[48]:
```python
svm_params = {'C': [0.1, 1, 10]}
svm_clf = LinearSVC(verbose=0)
svm_grid_search = GridSearchCV(svm_clf, svm_params)
svm_grid_search.fit(X_train, y_train.ravel())

svm_best_params = svm_grid_search.best_params_
svm_best_score = svm_grid_search.best_score_

print("Linear SVM:")
print("Best Parameters:", svm_best_params)
print("Best Score:", svm_best_score)
```

```
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

```
Linear SVM:
```

```
Best Parameters: {'C': 0.1}
Best Score: 0.89
```

```
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
/usr/local/lib/python3.11/site-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

With the LinearSVC model we get a score of 0.89 which is 0.01 point lower than previous method.

[49]:
```
knn_params = {'n_neighbors': [3, 5, 7]}
knn_clf = KNeighborsClassifier()
knn_grid_search = GridSearchCV(knn_clf, knn_params)
knn_grid_search.fit(X_train, y_train.ravel())

knn_best_params = knn_grid_search.best_params_
knn_best_score = knn_grid_search.best_score_

print("k-Nearest Neighbors:")
print("Best Parameters:", knn_best_params)
print("Best Score:", knn_best_score)
```

```
k-Nearest Neighbors:
Best Parameters: {'n_neighbors': 7}
Best Score: 0.7849999999999999
```

Finally, we used the KneighborsClassifier, ending with a score of 0.78 and therefore has the lowest one of the bunch.

The best model here is the Logistic regression method that came first with a score of 0.9.

Note that using GridSearchCV on the training data, you can access the best parameters and the best score using the best_params_ and best_score_ attributes of the GridSearchCV object, respectively. Doing so will give you the best parameters and the best score for each model. You can change hyperparameters and their respective values in the params dictionaries to explore different combinations and see the impact it can have on performances for each model. In this case, LogisicRegression has toped our tests. Don't hesitate to play around with this yourself.