Richard Nadar
RA1911030010109
5-4-2022

ARTIFICIAL INTELLIGENCE (18CSC305J) LAB

EXPERIMENT 12: Development of ensemble model for an application

AIM: To develop a model an ensemble model for an application.

Language: Python

Theory:

Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote.

Problem Formulation and Algorithm:

- Import the required libraries.
- Import the data file in the program.
- Clean the data.
- Find the required features on which the model predicts.
- Split the data into two parts say X and Y.
- Train the model using the data X and find the mean absolute error with the predicted value and Y.
- We use random forest and decision trees algorithms to predict our model.
- A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging.

CODE:

1. Bagged decision trees:

```
import pandas
from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier


url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-
indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age','cl
ass']
dataframe = pandas.read_csv(url, names=names)


array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed, shuffle=True
)
cart = DecisionTreeClassifier()
num_trees = 100


model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, ran
dom_state=seed)
results = model_selection.cross_val_score(model, X, Y, cv=kfold)
print(results)


print(results.mean())
```

2. Random Forest:

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


url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-
indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age','cl
ass']
dataframe = pandas.read_csv(url, names=names)


array = dataframe.values
```

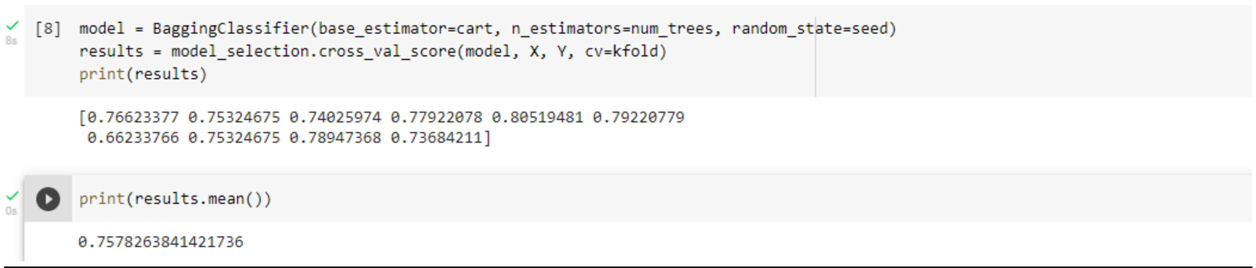
```
X = array[:,0:8]
Y = array[:,8]
seed = 7
num_trees = 100
max_features = 3


kfold = model_selection.KFold(n_splits=10, random_state=seed, shuffle=True
)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_fe
atures)
results1 = model_selection.cross_val_score(model, X, Y, cv=kfold)

print(results1)
print(results1.mean())
```

3. Extra trees:

```
import pandas
from sklearn import model_selection
from sklearn.ensemble import ExtraTreesClassifier


url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-
indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age','cl
ass']
dataframe = pandas.read_csv(url, names=names)


array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
seed = 7
num_trees = 100
max_features = 7


kfold = model_selection.KFold(n_splits=10, random_state=seed, shuffle=True
)
model = ExtraTreesClassifier(n_estimators=num_trees, max_features=max_feat
ures)
results2 = model_selection.cross_val_score(model, X, Y, cv=kfold)


print(results2)
print(results2.mean())
```

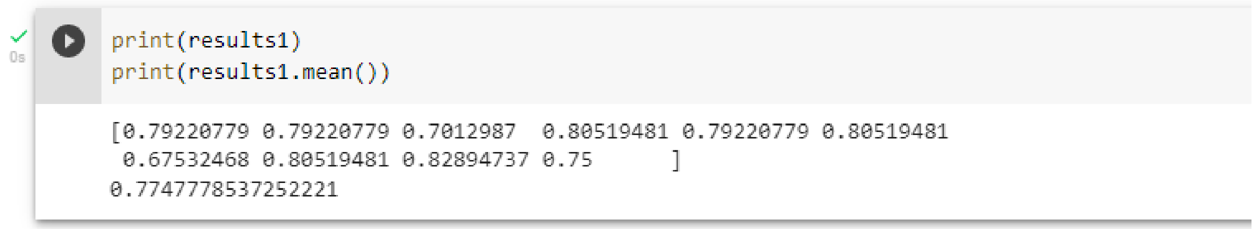
OUTPUT:

1. Bagged decision trees:



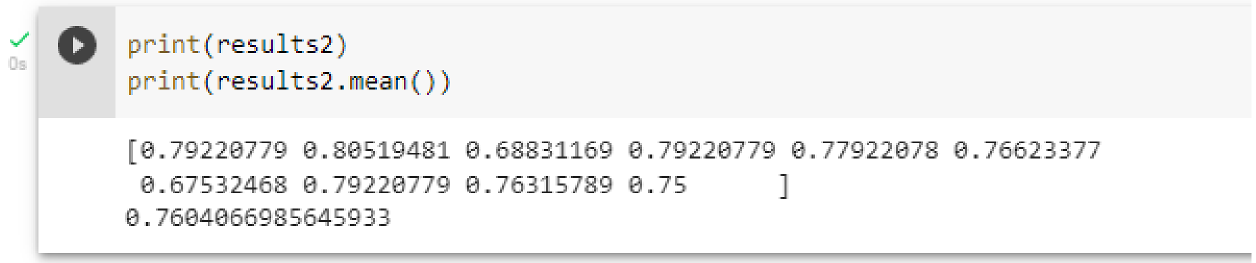
```
[8] model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
    results = model_selection.cross_val_score(model, X, Y, cv=kfold)
    print(results)

    [0.76623377 0.75324675 0.74025974 0.77922078 0.80519481 0.79220779
     0.66233766 0.75324675 0.78947368 0.73684211]

    print(results.mean())

    0.7578263841421736
```

2. Random Forest:



```
print(results1)
print(results1.mean())

[0.79220779 0.79220779 0.7012987  0.80519481 0.79220779 0.80519481
 0.67532468 0.80519481 0.82894737 0.75      ]
0.7747778537252221
```

3. Extra trees:



```
print(results2)
print(results2.mean())

[0.79220779 0.80519481 0.68831169 0.79220779 0.77922078 0.76623377
 0.67532468 0.79220779 0.76315789 0.75      ]
0.7604066985645933
```

Verification:

From both the algorithms we can see the difference in the mean absolute error. The predictions made are nearby the data which are input, so we can verify the data is correctly predicted as it lies along the dataset provided.

RESULT: Hence, successfully implemented the problem and verified the output and document result.