

EXPERIMENT 12

Development of ensemble model for an application

NAME: Rahul Goel

REG NO: RA1911030010094

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. Baggeddecisiontrees:

```
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', 'class']

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, random_state=seed)
results = model_selection.cross_val_score(model, X, Y,
cv=kfold) print(results)

```

`print(results.mean())` **OUTPUT:**

```

# BAGGED DECISION TREES

In [12]: import pandas
from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

In [5]: url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)

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

In [9]: model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X, Y, cv=kfold)

In [10]: print(results)

[0.66233766 0.84415584 0.75324675 0.64935065 0.83116883 0.83116883
 0.83116883 0.84415584 0.68421053 0.78947368]

In [11]: print(results.mean())

0.7720437457279563

```

2. RandomForest:

```

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', 'class']

dataframe = pandas.read_csv(url, names=names)

array = dataframe.values
X = array[:,0:8]
Y = array[:,8]

```

```

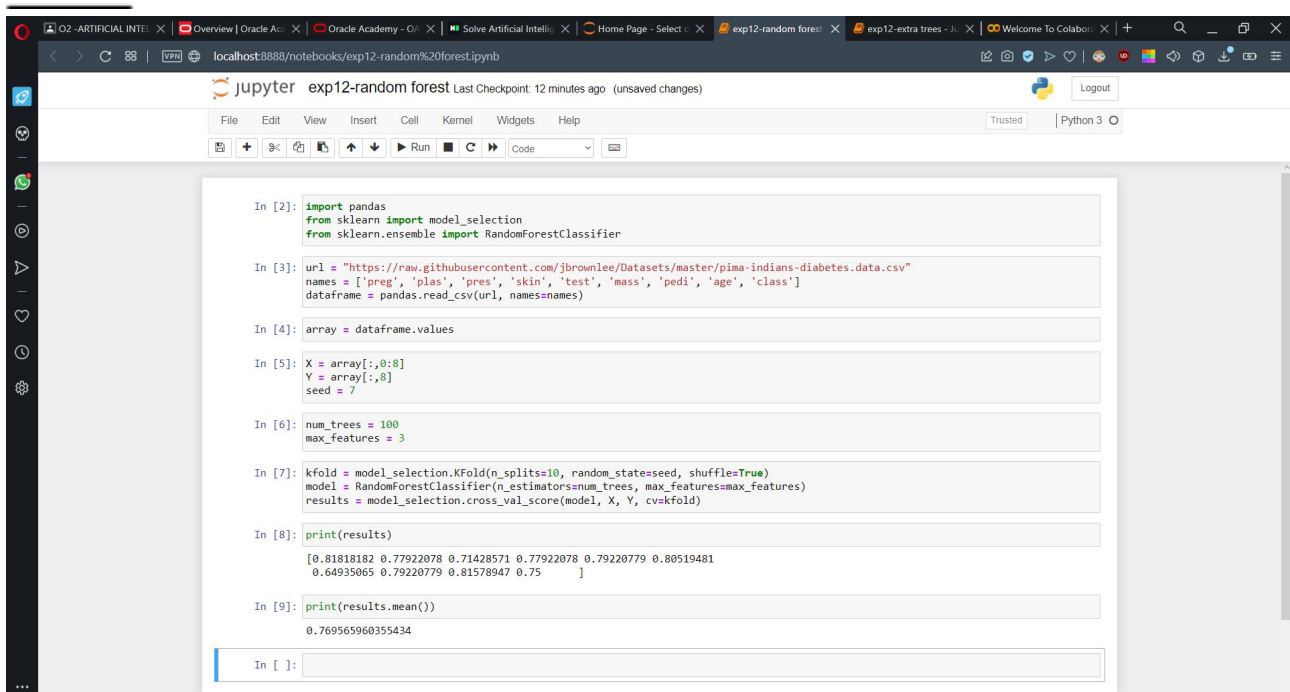
seed = 7
num_trees = 100
max_features = 3
kfold = model_selection.KFold(n_splits=10,
random_state=seed, shuffle=True)
model = RandomForestClassifier(n_estimators=num_trees,
max_features= max_features)

results1 = model_selection.cross_val_score(model, X, Y,
cv=kfold)

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

```

OUTPUT:



```

In [2]: import pandas
from sklearn import model_selection
from sklearn.ensemble import RandomForestClassifier

In [3]: url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)

In [4]: array = dataframe.values

In [5]: X = array[:,0:8]
Y = array[:,8]
seed = 7

In [6]: num_trees = 100
max_features = 3

In [7]: kfold = model_selection.KFold(n_splits=10, random_state=seed, shuffle=True)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, X, Y, cv=kfold)

In [8]: print(results)
[0.81818182 0.77922078 0.71428571 0.77922078 0.79220779 0.80519481
 0.64935065 0.79220779 0.81578947 0.75      ]

In [9]: print(results.mean())
0.769565960355434

In [ ]:

```

3. Extratrees:

```

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', 'class']

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_features)

results2 = model_selection.cross_val_score(model, X, Y,
cv=kfold)

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

```

OUTPUT:

```

In [1]: import pandas
from sklearn import model_selection
from sklearn.ensemble import ExtraTreesClassifier

In [2]: url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)

In [3]: array = dataframe.values

In [4]: X = array[:,0:8]
Y = array[:,8]
seed = 7
num_trees = 100
max_features = 7

In [6]: kfold = model_selection.KFold(n_splits=10, random_state=seed, shuffle=True)
model = ExtraTreesClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, X, Y, cv=kfold)

In [7]: print(results)

[0.79220779 0.79220779 0.7012987 0.77922078 0.79220779 0.77922078
0.67532468 0.79220779 0.78947368 0.73684211]

In [8]: print(results.mean())

0.7630211893369789

In [ ]:

```

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.