Ashpak Sheikh

DECISION TREE CLASSIFIER

BATCH : DSWDMOD 020421

Business problem: -

Divide the diabetes data into train and test datasets and build a Random Forest and Decision Tree model with Outcome as the output variable.

About data: -

We have been given data about diabetes patients with their details about age ,weight, number of times pregnant etc

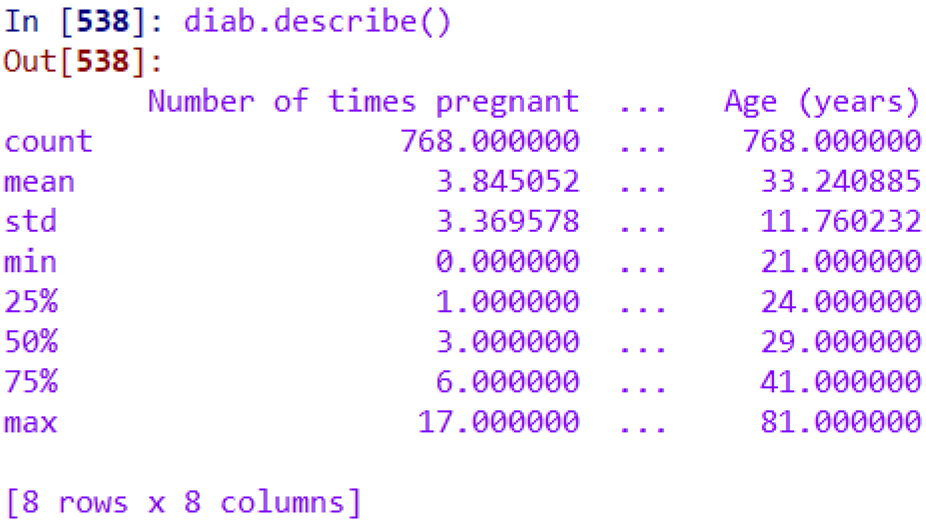
Analysis with Python: -

importing required libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


#loading data set

diab=pd.read_csv("D:/DataScience/Class/assignment working/DC/Diabetes.csv")
```

checking description

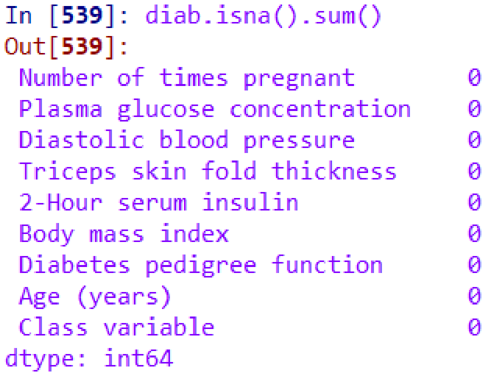
```
diab.describe()
```



```
In [538]: diab.describe()
Out[538]:
        Number of times pregnant  ...   Age (years)
count                 768.000000  ...    768.000000
mean                    3.845052  ...     33.240885
std                     3.369578  ...     11.760232
min                     0.000000  ...     21.000000
25%                     1.000000  ...     24.000000
50%                     3.000000  ...     29.000000
75%                     6.000000  ...     41.000000
max                    17.000000  ...     81.000000

[8 rows x 8 columns]
```

checking missing data

```
diab.isna().sum()
```



```
In [539]: diab.isna().sum()
Out[539]:
 Number of times pregnant        0
 Plasma glucose concentration    0
 Diastolic blood pressure        0
 Triceps skin fold thickness     0
 2-Hour serum insulin            0
 Body mass index                 0
 Diabetes pedigree function      0
 Age (years)                     0
 Class variable                  0
dtype: int64
```

creating dummies

```
diab=pd.get_dummies(diab,columns=[" Class variable"],drop_first=True)
```

seperating target and predictors

```
target=diab[" Class variable_YES"]

predictors=diab.drop(" Class variable_YES",axis=1)
```

splitting data

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train ,y_test =train_test_split(predictors,target)
```

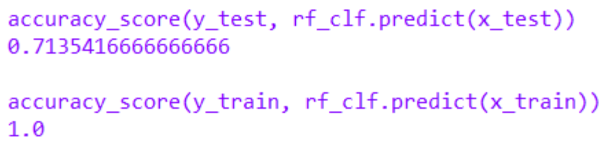
classification with default data

```
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(n_estimators=500, n_jobs=-1, random_state=42)

rf_clf.fit(x_train, y_train)
```

checking accuracy

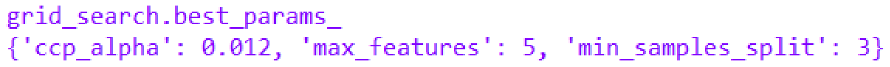
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
confusion_matrix(y_test, rf_clf.predict(x_test))

accuracy_score(y_test, rf_clf.predict(x_test))

accuracy_score(y_train, rf_clf.predict(x_train))
```



```
accuracy_score(y_test, rf_clf.predict(x_test))
0.7135416666666666

accuracy_score(y_train, rf_clf.predict(x_train))
1.0
```

```
#model is over fitting so cross validating


#GridSearchCV

#cross validating hyperparameters


from sklearn.model_selection import GridSearchCV

rf_clf_grid = RandomForestClassifier(n_estimators=500, n_jobs=-1, random_state=42)

param_grid = {"max_features": [4, 5, 6, 7, 8, 9, 10], "min_samples_split": [2, 3,4,5, 10],"ccp_alpha":[0.012]}

grid_search = GridSearchCV(rf_clf_grid, param_grid, n_jobs = -1, cv = 5, scoring = 'accuracy')


grid_search.fit(x_train, y_train)

grid_search.best_params_

cv_rf_clf_grid = grid_search.best_estimator_
```



```
grid_search.best_params_
{'ccp_alpha': 0.012, 'max_features': 5, 'min_samples_split': 3}
```

checking accuracy

```
from sklearn.metrics import accuracy_score, confusion_matrix


confusion_matrix(y_test, cv_rf_clf_grid.predict(x_test))

accuracy_score(y_test, cv_rf_clf_grid.predict(x_test))

accuracy_score(y_train, cv_rf_clf_grid.predict(x_train))
accuracy_score(y_test, cv_rf_clf_grid.predict(x_test))
0.7447916666666666

accuracy_score(y_train, cv_rf_clf_grid.predict(x_train))
0.8038194444444444
```

Cross validation for decision tree

```
from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

base_learn=DecisionTreeClassifier()

paramiter_grid={"max_features" : [2,3,4,5,6,7,8,9],"min_samples_split":[2,3,4,5,6]}

grid_search=GridSearchCV(base_learn,paramiter_grid,scoring="accuracy" ,cv=5 ,n_jobs=-1 )

grid_search.fit(x_train,y_train)


grid_search.best_params_
```

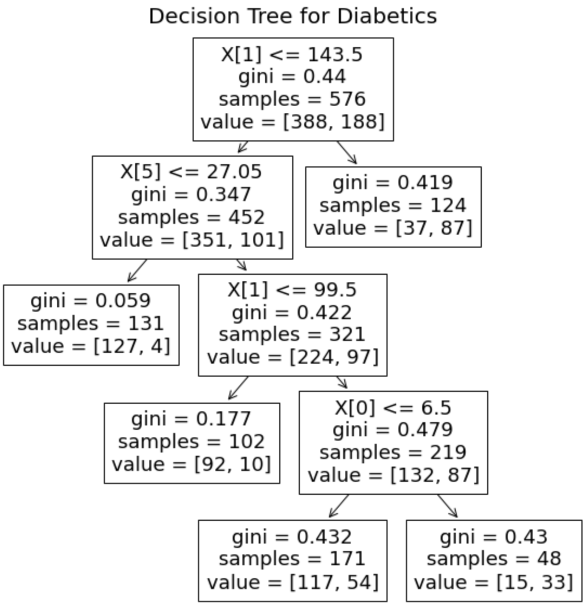
```
grid_search.best_params_
{'max_features': 5, 'min_samples_split': 6}
```

Decision tree on best parameters from grid search

```
from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier(max_features= 8, min_samples_split=4,ccp_alpha=0.011)

dt.fit(x_train, y_train)

test_pred=dt.predict(x_test)

np.mean(y_test==test_pred)

np.mean(y_train==dt.predict(x_train))
```

```
np.mean(y_test==test_pred)
0.703125

np.mean(y_train==dt.predict(x_train))
0.7916666666666666
```

```
plt.figure(figsize=(10,10))

tree.plot_tree(dt)

plt.title("Decision Tree for Diabetics",fontsize=20)
```



Summary and inference: -

- As we can see in this case both the classifier are giving almost same accuracy
- Though we can still prefer random forest over Decision Tree
- Final accuracy is not that good so we can try Ensemble techniques for better accuracy