5주. Decision Tree, RF, SVM			
학번	3 2 1 7 0 5 7 8	이름	김산

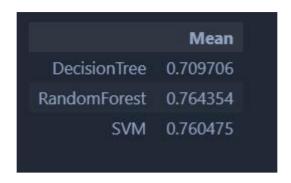
PimalndiansDiabetes dataset을 가지고 Classification 을 하고자 한다. (마지막의 diabetes 컬럼이 class label 임)

- Q1 (4점) scikit-learn에서 제공하는 DecisionTree, RandonForest, support vector machine 알고리즘를 이용하여 **PimalndiansDiabetes dataset**에 대한 분류 모델을 생성하고 accuracy를 비교하시오.
- 10-fold cross validation을 실시하여 mean accuracy를 비교한다
- 각 알고리즘의 hyper parameter 의 값은 default value를 이용한다.

Source code:

```
from sklearn.tree import DecisionTreeClassifier, export graphviz
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from sklearn.model selection import KFold
from sklearn.model selection import cross val score
import pandas as pd
df = pd.read csv("./Data/PimaIndiansDiabetes.csv")
df X = df.loc[:, df.columns != "diabetes"]
df y = df["diabetes"]
classifiers = ['DecisionTree', 'RandomForest', 'SVM']
models =
[DecisionTreeClassifier(),RandomForestClassifier(),svm.SVC()]
kfold = KFold(n splits=10, shuffle=True, random state=1234)
mean = []
for i in models:
   model = i
   cv result =
cross val score(model,df X,df y,cv=kfold,scoring="accuracy")
   mean.append(cv_result.mean())
accuracy df=pd.DataFrame({'Mean':mean},index=classifiers)
accuracy df
```

실행화면 캡쳐:



Q2. (3점) 다음의 조건에 따라 support vector machine 알고리즘를 이용하여 **PimalndiansDiabetes dataset**에 대한 분류 모델을 생성하고 accuracy를 비교하시오.

- hyper parameter 중 kernel 에 대해 linear, poly, rbf, sigmoid, precomputed를 각 각 테스트하여 어떤 kernel 이 가장 높은 accuracy를 도출하는지 확인하시오.
- 10-fold cross validation을 실시하여 mean accuracy를 비교한다

Source code:

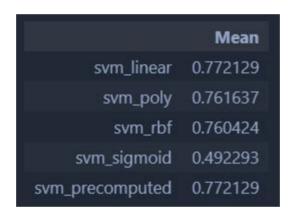
```
from sklearn import svm
from sklearn.model selection import KFold
from sklearn.model selection import cross val score
import pandas as pd
import numpy as np
#데이터프레임 read
df = pd.read_csv("./Data/PimaIndiansDiabetes.csv")
df X = df.loc[:, df.columns != "diabetes"]
df y = df["diabetes"]
#linear, poly, rbf, sigmoid 커널 수행
classifiers = ['svm_linear', 'svm_poly', 'svm_rbf', 'svm_sigmoid',
'svm precomputed']
models = [svm.SVC(kernel='linear'), svm.SVC(kernel='poly'),
svm.SVC(kernel='rbf'), svm.SVC(kernel='sigmoid')]
kfold = KFold(n_splits=10)
mean = []
for i in models:
   model = i
   cv result =
```

```
ross_val_score(model,df_X,df_y,cv=kfold,scoring="accuracy")
mean.append(cv_result.mean())

# precomputed 커널 수행
gram_test = np.dot(df_X, df_X.T)
model = svm.SVC(C=1.0, kernel='precomputed')
cv_result = cross_val_score(model, gram_test, df_y,
cv=kfold,scoring="accuracy")
mean.append(cv_result.mean())

accuracy_df=pd.DataFrame({'Mean':mean},index=classifiers)
accuracy_df
```

실행화면 캡쳐:



Q3. (3점) 다음의 조건에 따라 Random Forest 알고리즘를 이용하여 **PimalndiansDiabetes dataset**에 대한 분류 모델을 생성하고 accuracy를 비교하시오.

-다음의 hyper parameter를 테스트 하시오

- . n_estimators : 100, 200, 300, 400, 500
- . max_features : 1, 2, 3, 4, 5
- 어떤 조합이 가장 높은 accuracy를 도출하는지 확인하시오.
- 10-fold cross validation을 실시하여 mean accuracy를 비교한다

Source code:

```
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import KFold

from sklearn.model_selection import cross_val_score
```

```
import pandas as pd
df = pd.read csv("./Data/PimaIndiansDiabetes.csv")
df X = df.loc[:, df.columns != "diabetes"]
df y = df["diabetes"]
n_{estimators} = [100, 200, 300, 400, 500]
max_features = [1,2,3,4,5]
index = []
models = []
for n in n_estimators:
   for feature in max features:
      index.append(str(n)+ ' ' + str(feature))
      models.append(RandomForestClassifier(n_estimators
max_features= feature))
kfold = KFold(n_splits=10, shuffle=True, random_state=1234)
mean = []
for i in range(len(models)):
   cv_result = cross_val_score(
      models[i], df_X, df_y, cv=kfold, scoring="accuracy")
   mean.append(cv_result.mean())
```

실행화면 캡쳐:

300과 1의 조합이 가장 높은 accuracy를 도출하는 것을 확인할 수 있었습니다 .

	Mean
100 1	0.761705
100 2	0.770813
100 3	0.756494
100 4	0.768267
100 5	0.769549
200 1	0.764354
200 2	0.766934
200 3	0.765602
200 4	0.770813
200 5	0.769532
300 1	0.772146
300 2	0.761740
300 3	0.756528
300 4	0.764286
300 5	0.769549
400 1	0.770882
400 2	0.766951
400 3	0.761740
400 4	0.762987
400 5	0.765602
500 1	0.763055
500 2	0.768233
500 3	0.772078
500 4	0.769549
500 5	0.769532