6주. Decision Tree, RF, SVM			
학번	32153180	이름	이상민

```
# 과제에 필요한 package
import pandas as pd
import numpy as np

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
from sklearn import svm
```

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

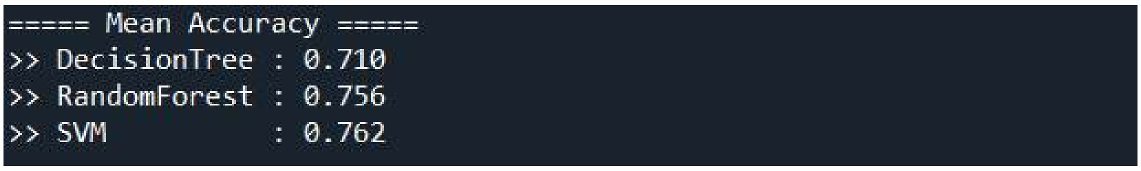
Q1 (4점) scikit-learn에서 제공하는 DecisionTree, RandonForest, support vector machine 알고리즘를 이용하여 PimaIndiansDiabetes dataset에 대한 분류 모델을 생성하고 accuracy를 비교하시오.
- 10-fold cross validation을 실시하여 mean accuracy를 비교한다
- 각 알고리즘의 hyper parameter 의 값은 default value를 이용한다.

Source code :

```
data = pd.read_csv('C:/Users/sangmin/Desktop/학교생활/4-2/딥러닝클라우드
/dataset/PimaIndiansDiabetes.csv')
data_X = data.loc[:, data.columns != "diabetes"]
data_y = data["diabetes"]
arr_X = np.array(data_X)
arr_y = np.array(data_y)

model_dt = DecisionTreeClassifier(random_state=100)
model_rf = RandomForestClassifier(random_state=100)
model_svm = svm.SVC(random_state=100)

kf = KFold(n_splits=10, random_state=100, shuffle=True)

acc_dt = np.zeros(10)
acc_rf = np.zeros(10)
acc_svm = np.zeros(10)
```

```
i = 0
for train_index, test_index in kf.split(arr_X):
    train_X, test_X = arr_X[train_index], arr_X[test_index]
    train_y, test_y = arr_y[train_index], arr_y[test_index]

    model_dt.fit(train_X, train_y)
    model_rf.fit(train_X, train_y)
    model_svm.fit(train_X, train_y)

    pred_dt = model_dt.predict(test_X)
    pred_rf = model_rf.predict(test_X)
    pred_svm = model_svm.predict(test_X)

    acc_dt[i] = accuracy_score(test_y, pred_dt)
    acc_rf[i] = accuracy_score(test_y, pred_rf)
    acc_svm[i] = accuracy_score(test_y, pred_svm)

    i += 1

print('===== Mean Accuracy =====')
print('>> DecisionTree : {:.3f}'.format(np.mean(acc_dt)))
print('>> RandomForest : {:.3f}'.format(np.mean(acc_rf)))
print('>> SVM          : {:.3f}'.format(np.mean(acc_svm)))
```

실행화면 캡처:



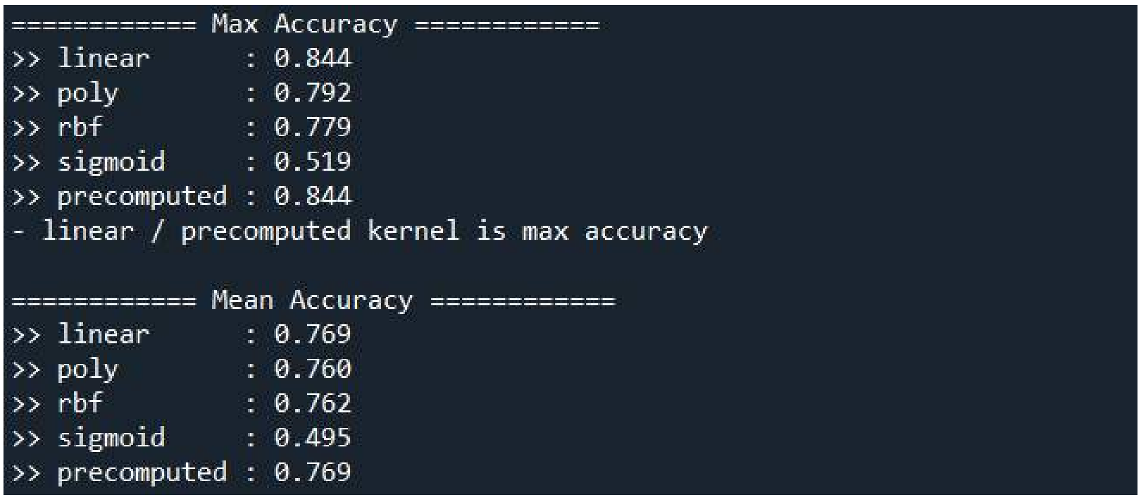
```
===== Mean Accuracy =====
>> DecisionTree : 0.710
>> RandomForest : 0.756
>> SVM          : 0.762
```

Q2. (3점) 다음의 조건에 따라 support vector machine 알고리즘를 이용하여 **PimaIndiansDiabetes dataset**에 대한 분류 모델을 생성하고 accuracy를 비교하시오.
- hyper parameter 중 kernel 에 대해 linear, poly, rbf, sigmoid, precomputed를 각각 테스트하여 어떤 kernel 이 가장 높은 accuracy를 도출하는지 확인하시오.
- 10-fold cross validation을 실시하여 mean accuracy를 비교한다

Source code :

```
model_lin = svm.SVC(random_state=100, kernel="linear")
model_poly = svm.SVC(random_state=100, kernel="poly")
model_rbf = svm.SVC(random_state=100, kernel="rbf")
model_sig = svm.SVC(random_state=100, kernel="sigmoid")
model_pre = svm.SVC(random_state=100, kernel="precomputed")

kf = KFold(n_splits=10, random_state=100, shuffle=True)

acc_lin = np.zeros(10)
acc_poly = np.zeros(10)
acc_rbf = np.zeros(10)
acc_sig = np.zeros(10)
acc_pre = np.zeros(10)

max_acc = 0
i = 0
for train_index, test_index in kf.split(arr_X):
    train_X, test_X = arr_X[train_index], arr_X[test_index]
    train_y, test_y = arr_y[train_index], arr_y[test_index]

    gram_train = np.dot(train_X, train_X.T)

    model_lin.fit(train_X, train_y)
    model_poly.fit(train_X, train_y)
    model_rbf.fit(train_X, train_y)
    model_sig.fit(train_X, train_y)
    model_pre.fit(gram_train, train_y)

    gram_test = np.dot(test_X, train_X.T)

    pred_lin = model_lin.predict(test_X)
    pred_poly = model_poly.predict(test_X)
    pred_rbf = model_rbf.predict(test_X)
    pred_sig = model_sig.predict(test_X)
    pred_pre = model_pre.predict(gram_test)

    acc_lin[i] = accuracy_score(test_y, pred_lin)
    acc_poly[i] = accuracy_score(test_y, pred_poly)
    acc_rbf[i] = accuracy_score(test_y, pred_rbf)
    acc_sig[i] = accuracy_score(test_y, pred_sig)
    acc_pre[i] = accuracy_score(test_y, pred_pre)

    accuracy = max(max_acc, acc_lin[i], acc_poly[i], acc_rbf[i],
                   acc_sig[i], acc_pre[i])

    if max_acc < accuracy:
        max_acc = accuracy
        max_idx = i

    i += 1

print('============ Max Accuracy ============')
print('>> linear      : {:.3f}'.format(acc_lin[max_idx]))
print('>> poly        : {:.3f}'.format(acc_poly[max_idx]))
print('>> rbf         : {:.3f}'.format(acc_rbf[max_idx]))
print('>> sigmoid     : {:.3f}'.format(acc_sig[max_idx]))
print('>> precomputed : {:.3f}'.format(acc_pre[max_idx]))
print('- linear / precomputed kernel is max accuracy')

print('\n============ Mean Accuracy ============')
print('>> linear      : {:.3f}'.format(np.mean(acc_lin)))
print('>> poly        : {:.3f}'.format(np.mean(acc_poly)))
print('>> rbf         : {:.3f}'.format(np.mean(acc_rbf)))
print('>> sigmoid     : {:.3f}'.format(np.mean(acc_sig)))
print('>> precomputed : {:.3f}'.format(np.mean(acc_pre)))
```

실행화면 캡쳐:



```
============ Max Accuracy ============
>> linear      : 0.844
>> poly        : 0.792
>> rbf         : 0.779
>> sigmoid     : 0.519
>> precomputed : 0.844
- linear / precomputed kernel is max accuracy

============ Mean Accuracy ============
>> linear      : 0.769
>> poly        : 0.760
>> rbf         : 0.762
>> sigmoid     : 0.495
>> precomputed : 0.769
```

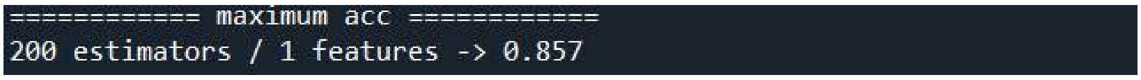
Q3. (3점) 다음의 조건에 따라 Random Forest 알고리즘를 이용하여 **PimaIndiansDiabetes dataset**에 대한 분류 모델을 생성하고 accuracy를 비교하시오.
-다음의 hyper parameter를 테스트 하시오
 . n_estimators : 100, 200, 300, 400, 500
 . max_features : 1, 2, 3, 4, 5
 어떤 조합이 가장 높은 accuracy를 도출하는지 확인하시오.
- 10-fold cross validation을 실시하여 mean accuracy를 비교한다

Source code :

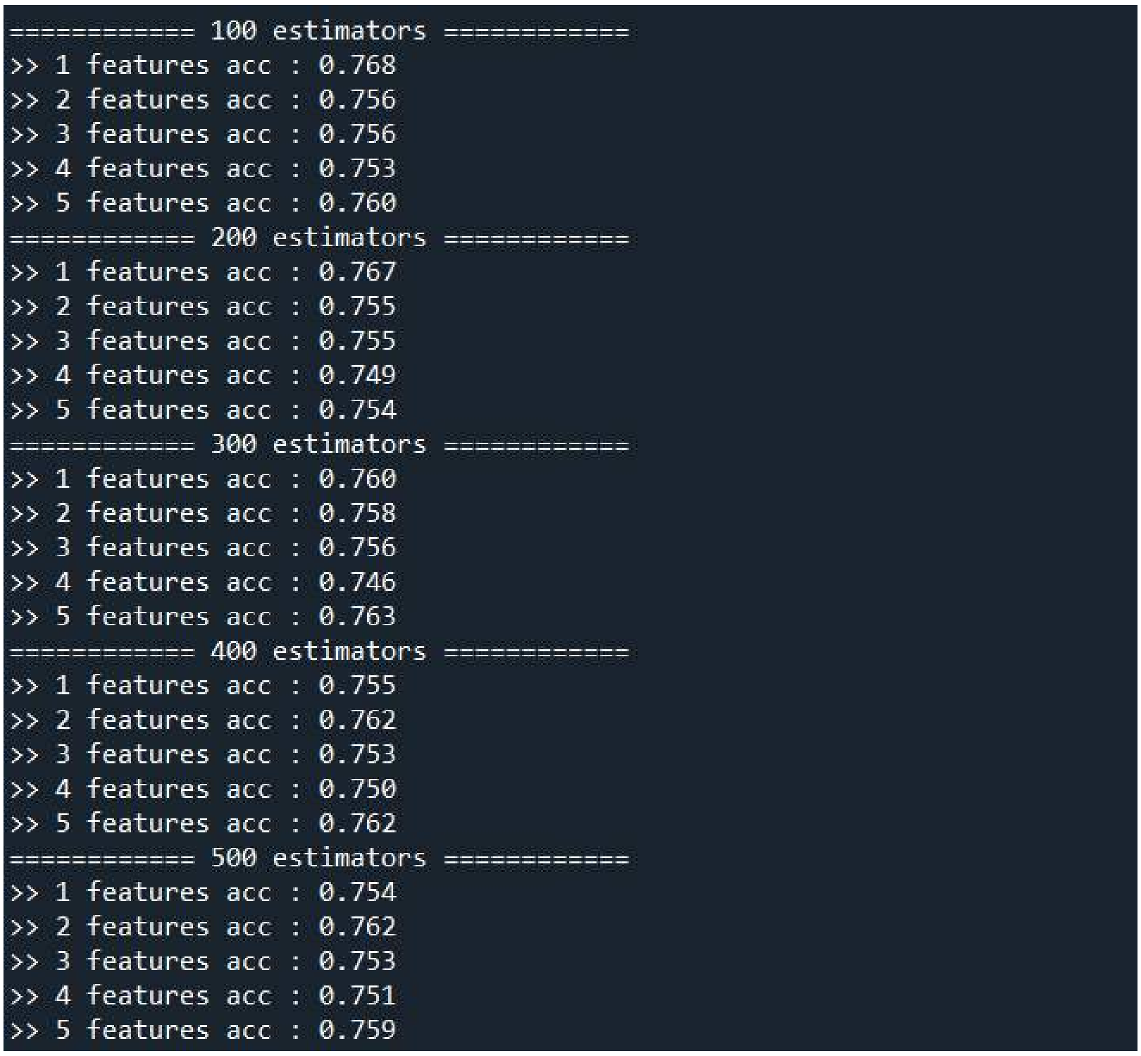
```
estimators = [100, 200, 300, 400, 500]
features = [1, 2, 3, 4, 5]
model = [[] for _ in range(5)]

kf = KFold(n_splits=10, random_state=100, shuffle=True)

idx = 0
for i in estimators:
    for j in features:
        model[idx].append(RandomForestClassifier(random_state=100,
n_estimators=i, max_features=j))
    idx += 1

acc = [[np.zeros(10) for _ in range(5)] for _ in range(5)]
mean_acc = [[] for _ in range(5)]

max_acc = 0
max_i, max_j, max_idx = 0, 0, 0
```

```
for i in range(len(model)):
    for j in range(len(model[i])):
        idx = 0
        for train_index, test_index in kf.split(arr_X):
            train_X, test_X = arr_X[train_index], arr_X[test_index]
            train_y, test_y = arr_y[train_index], arr_y[test_index]

            model[i][j].fit(train_X, train_y)
            pred = model[i][j].predict(test_X)
            acc[i][j][idx] = accuracy_score(test_y, pred)

            if max_acc < acc[i][j][idx]:
                max_acc = acc[i][j][idx]
                max_i, max_j, max_idx = i, j, idx

            idx += 1
        mean_acc[i].append(np.mean(acc[i][j]))

print('============ maximum acc ============')
print('{} estimators / {} features -> {:.3f}'\
      .format(estimators[max_i],                    features[max_j],
acc[max_i][max_j][max_idx]))
print('\n')

for i in range(len(mean_acc)):
    print('============                   {}                  estimators
============'.format(estimators[i]));
    for j in range(len(mean_acc[i])):
        print('>>   {}   features   acc   :   {:.3f}'.format(features[j],
mean_acc[i][j]))
```

실행화면 캡쳐:



```
============ maximum acc ============
200 estimators / 1 features -> 0.857
```



```
============ 100 estimators ============
>> 1 features acc : 0.768
>> 2 features acc : 0.756
>> 3 features acc : 0.756
>> 4 features acc : 0.753
>> 5 features acc : 0.760
============ 200 estimators ============
>> 1 features acc : 0.767
>> 2 features acc : 0.755
>> 3 features acc : 0.755
>> 4 features acc : 0.749
>> 5 features acc : 0.754
============ 300 estimators ============
>> 1 features acc : 0.760
>> 2 features acc : 0.758
>> 3 features acc : 0.756
>> 4 features acc : 0.746
>> 5 features acc : 0.763
============ 400 estimators ============
>> 1 features acc : 0.755
>> 2 features acc : 0.762
>> 3 features acc : 0.753
>> 4 features acc : 0.750
>> 5 features acc : 0.762
============ 500 estimators ============
>> 1 features acc : 0.754
>> 2 features acc : 0.762
>> 3 features acc : 0.753
>> 4 features acc : 0.751
>> 5 features acc : 0.759
```