코알라유니브_week4
"Random Forest로 아이리스 종 구분하기"

발표자 김채은

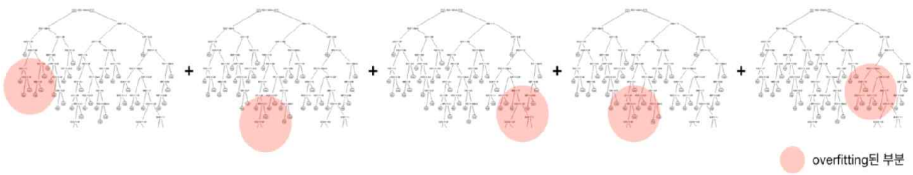
3주차. "Scikit-learn으로 Decision Tree구현"
4주차. "Scikit-learn으로 Random Forest구현"

■Stage1
-Ensemble 소개
-Random Forest 소개
-Decision Tree를 Ensemble 해보기

□Ensemble 소개
Ensemble(앙상블)이란?
여러개의 Machine결과를 연결하여 취합한 보다 강력한 모델

□Random Forest 소개
Random Forest란?
Ensemble 알고리즘의 일종으로, 여러 Decision Tree만들고 연결하여 결과 취합 후 평균

✎Why Random Forest? Decision Tree는 학습 데이터를 과하게 학습하여(overfitting), 실제 데이터에서 오차 발생시키는 단점 존재 → 여러개의 Decision Tree를 만들면 overfitting된 부분이 각기 다르기 때문에 모델들을 평균내면 overfitting 문제가 상당부분 해결



□Decision Tree를 Ensemble 해보기
1) 트리 여러개 만들기
<코드>

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier()
tree.fit(x_train, y_train)
print('training set accuracy: ')
tree.score(x_train, y_train)
```

```
tree1 = DecisionTreeClassifier()
tree1.fit(x_train, y_train)
print('training set accuracy: ')
tree1.score(x_train, y_train)

tree2 = DecisionTreeClassifier()
tree2.fit(x_train, y_train)
print('training set accuracy: ')
tree2.score(x_train, y_train)
```

결과>>> tree, tree1, tree2의 accuracy가 똑같음 → 데이터가 제한되어있는데, 한 번 학습을 했으므로, 여러개 만들어도 유사한 정확도를 가지는 모델로 생성됨 → 데이터 제한된 경우는 정석적인 데이터 분석 필요

<training데이터를 training과 validation으로 나누기>

Validation은 최종 Test하기 이전에 중간중간 예비 Test

✎비유
Test ≒ 수능
Validation ≒ 3, 6, 9월 모의고사

<코드>
```
x_valid = x_train[0:100]
y_valid = y_train[0:100]

x_train = x_train[100:]
y_train = y_train[100:]
```

```
tree = DecisionTreeClassifier()
tree.fit(x_train, y_train)
print('training set accuracy: ',tree.score(x_train, y_train))
print('validation set accuracy: ',tree.score(x_valid, y_valid))

tree1 = DecisionTreeClassifier()
tree1.fit(x_train, y_train)
print('training set accuracy: ',tree1.score(x_train, y_train))
print('validation set accuracy: ',tree1.score(x_valid, y_valid))

tree2 = DecisionTreeClassifier()
tree2.fit(x_train, y_train)
print('training set accuracy: ',tree2.score(x_train, y_train))
print('validation set accuracy: ',tree2.score(x_valid, y_valid))
```

찍어보면 validation set accuracy는 실행할 때마다 다른 값을 보임 -> train데이터처럼 과하게 학습되지 않았기 때문. 따라서 트리의 정확도는 training set accuracy보다 validaion set accuracy로 보는게 합당

□Decision Tree를 Ensemble해보기

<코드>

1. 각 모델 예측하기

```
prediction = tree.predict(test)
prediction1 = tree1.predict(test)
prediction2 = tree2.predict(test)
```

2. 예측값들 평균내기

```
ensemble = (prediction + prediction1 + prediction2) / 3
ensemble[ensemble > 0.5] = 1
ensemble[ensemble <= 0.5] = 0
```

3. 평균을 구함으로써 소수점으로 된 survived 데이터를 다시 1, 0으로 표시
   즉, 실수형을 정수형으로 형변환

```
ensemble = ensemble.astype(int)
```

■Stage2

-Iris 문제 분석

-Feature Engineering with Pandas / Visualizaion with Seaborn

-바이올린플롯 배우기

□Iris 문제 분석

꽃잎 및 꽃받침의 길이와 너비정보(SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm)로 → Iris 종(Setosa, Versicolor, Virginica) 구분하기

<코드>

df.info() #행, 열 개수와 각 변수들의 자료개수 및 형태(type)

df.describe() #변수별로 자료개수, 평균값, 표준편차, 다섯숫자요약(최대값, 최소값, 사분위수)

□Feature Engineering with Pandas / Visualizaion with Seaborn

특징들이 종을 얼마나 잘 구분해줄 수 있는지를 그래프로 봄

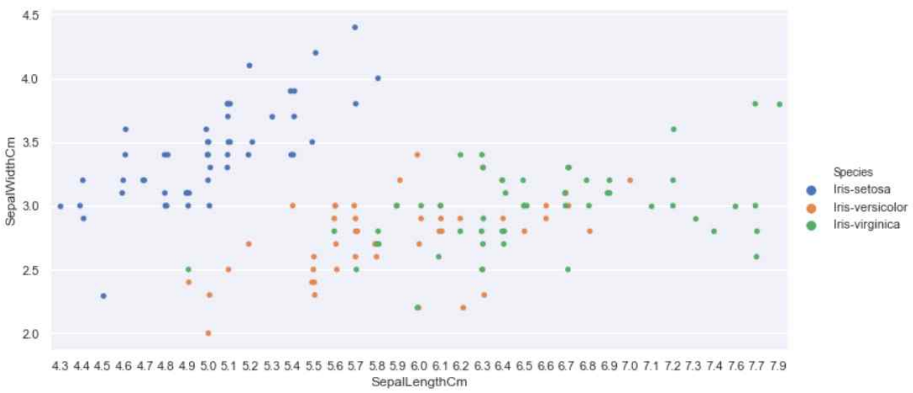
<코드>

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
```

✎꽃받침(Sepal)의 길이와 너비로 Plot그리기

<코드>

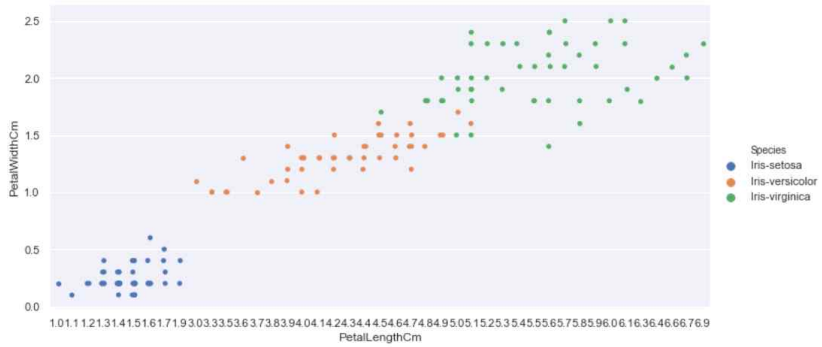
sns.catplot(data=df, x='SepalLengthCm', y='SepalWidthCm', hue = 'Species', aspect=2)



✎꽃잎(Petal)의 길이와 너비로 Plot 그리기

<코드>

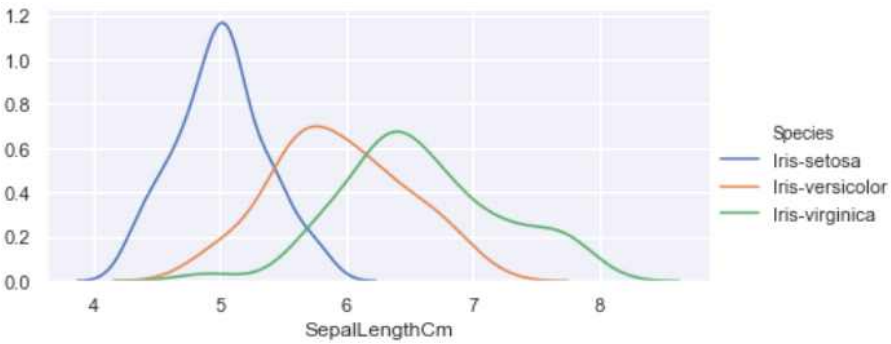
```
sns.catplot(data=df, x='PetalLengthCm', y='PetalWidthCm', hue = 'Species', aspect=2)
```



✎꽃받침 길이(SepalLengthCm)에 따른 밀집도

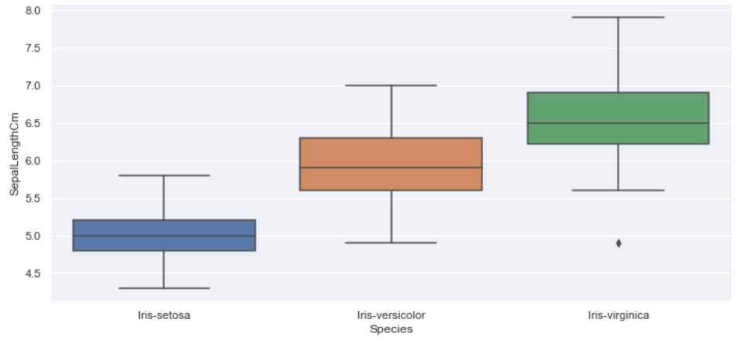
<코드>

```
facet = snsFacetGrid(df, hue='Species', aspect=2)
facet.map(sns.kdeplot, 'SepalLengthCm')
facet.add_legend()
plot.show()
```



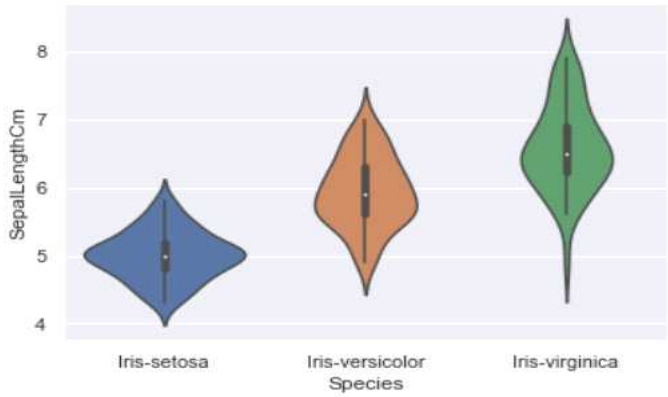
✎종(Species)에 따른 꽃받침 길이(SepalLengthCm) boxplot그려보기

```
sns.catplot(kind='box', data=df, x='Species', y='SepalLengthCm', aspect=2)
```



□바이올린플롯 배우기
violin plot그리기 데이터 범위와 밀집정도를 한번에 확인가능
kdeplot과 boxplot의 장점 합친 것

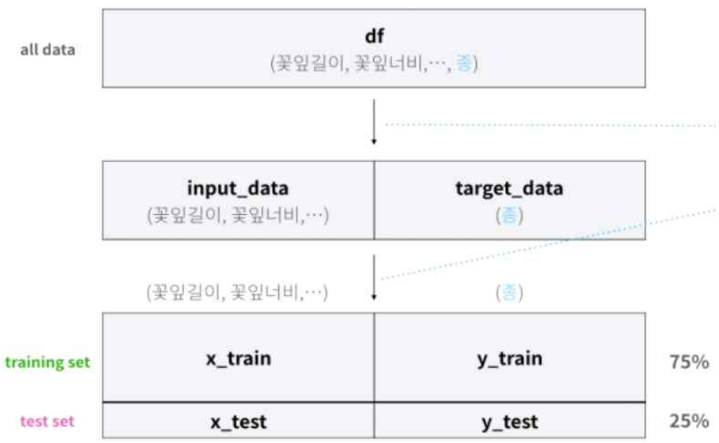
<코드>
sns.violinplot(data=df, x='Species', y=SepallLengthCm')



■Stage3
-Iris를 Scikit-learn으로 Random-Forest구현하기

□Scikit-learn으로 Random-Forest구현하기

✎데이터셋 구성 계획하기
1. Input data와 Target data 나누기
2. Training set과 Test set 나누기 ← 비율은 보통 8:2, 7:3



<코드>

1. Input data(X)와 Target data(Y) 나누기

```
input_data = df[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
target_data = df['Species']
```

2. Training data와 Test data 나누기

🖌train_test_split 함수 사용하기 ← 75% : 25% 비율로 random으로 고르게 나누어줌

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(input_data, target_data)
```

#First, Decision Tree

```
from sklearn.tree import DecisionTreeClassifier    #불러오기
tree = DecisionTreeClassifier    #Decision Tree알고리즘으로 초기화
tree.fit(x_train,y_train)    #학습시킴

print('training set accuracy: ', tree.score(x_train, y_train))
print('test set accuracy:', tree.score(x_test, y_test))    #정확도

prediction = tree.predict(x_test)    #예측하기
prediction
```

#Second, Random Forest

```
from sklearn.ensemble import RandomForestClassifier    #불러오기
forest = RandomForestClassifier(n_estimators=100) #RandomForest알고리즘으로 초기화
#Decision Tree의 개수
forest.fit(x_train, y_train)    #학습시킴

print('training set accuracy: ', forest.score(x_train, y_train))
print('test set accuracy:', forest.score(x_test, y_test))    #정확도

prediction_by_forest = forest.predict(x_test)    #예측하기
prediction_by_forest
```

■Stage4. 누가 내 우수고객이 될까?
(우수고객1, 아님0)

1. machine learning 위해서는 문자형 데이터를 숫자로 바꾸어주어야함
df.loc[조건, 열] = 원하는 값
df.loc[df['지역'] == '서울', '지역'] = 0
df.loc[df['지역'] == '경기', '지역'] = 1
df.loc[df['지역'] == '제주', '지역'] = 2

2. Input데이터와 Target데이터 나누어주기
x_train = df[['나이', '성별', '지역', '관심사']] #ID와 이름은 우수고객에 영향 없을 것임
y_train = df['우수고객분류']

3. 모델 만들기 및 정확도
from sklearn.ensemble import RandomForestClassifier #불러오기
forest = RandomForestClassifier(n_estimators=10) #RandomForest알고리즘으로 초기화
forest.fit(x_train, y_train) #학습시킴

print('training set accuracy: ', forest.score(x_train, y_train)) #정확도

4. 예측하기
df_test = pd.read_csv('data/new_customers.csv') #파일읽기
문자형 자료는 -> 숫자형으로 변환
x_test = df_test[['나이', '성별', '지역', '관심사']] #Input data(X)만들기

prediction = forest.predict(x_test) #예측하기
prediction