- 코 알 라 스 터 디 -

Week4_데이터로 꽃의 종류를 구분해보자 + 우수고객 분석

1 Stage1 - Random Forest를 배워보자

- Ensemble(앙상블)?

여러 머신러닝 모델을 연결하여 더 강력한 모델을 만드는 기법
다른 모양으로 형성된 모델 여러개를 연결하여 마치 여러명의 전문가에게 의견을 듣는 구조를 이용하는 것

- **Random Forest**

여러개의 Decision Tree를 만들고 연결하여 결과를 취합한 후 평균내어 성능을 높인 모델
(의사결정나무의 과적합-*일부 데이터에 너무 특화된 학습으로 인해 실제 예측이 빗나가는 상황*-문제를 해결)

1 Stage1 - Random Forest를 배워보자

- **Decision Tree를 Ensemble 해보기**

week4_Stage1

In [41]:
```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier()
tree.fit(x_train, y_train)

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

```
training set accuracy:  0.9292929292929293
```

In [42]:
```
tree1 = DecisionTreeClassifier()
tree1.fit(x_train, y_train)

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

```
training set accuracy:  0.9292929292929293
```

In [43]:
```
tree2 = DecisionTreeClassifier()
tree2.fit(x_train, y_train)

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

```
training set accuracy:  0.9292929292929293
```

In [44]:
```
tree3 = DecisionTreeClassifier()
tree3.fit(x_train, y_train)

print('training set accuracy: ', tree3.score(x_train, y_train))
```

```
training set accuracy:  0.9292929292929293
```

Tree학습

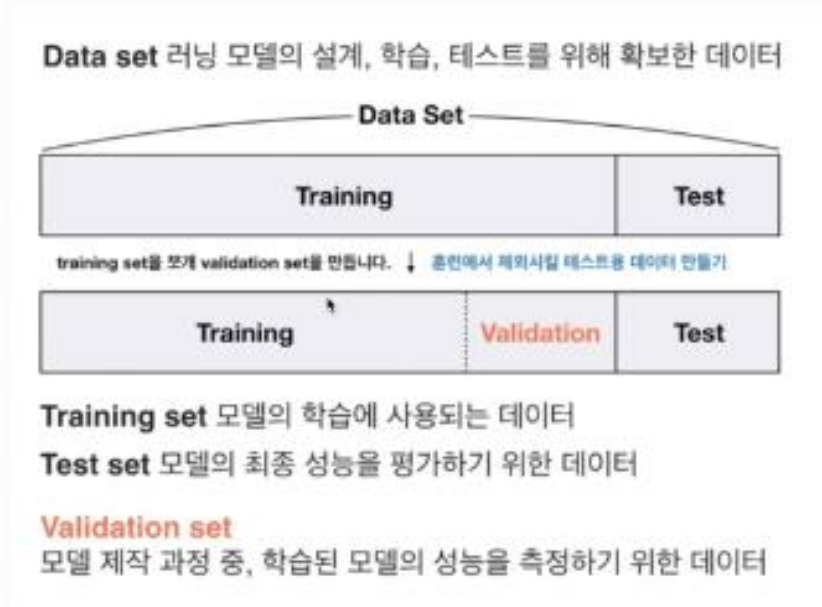
성능 검사

Tree 3개 만들기

서로 다른 Tree.
데이터 특성 상 비슷한 accuracy가 나옴.

1 Stage1 - Random Forest를 배워보자

- **Decision Tree를 Ensemble 해보기**



-> 모델 최종 선택 후 성능 점검 : test set

- **Training Set 나누고 성능 점검**

```
In [45]: x_valid = x_train[0:100]
         y_valid = y_train[0:100]

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

```
In [49]: 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))

         tree3 = DecisionTreeClassifier()
         tree3.fit(x_train, y_train)

         print('training set accuracy: ', tree3.score(x_train, y_train))
         print('validation set accuracy: ', tree3.score(x_valid, y_valid))
```

```
training set accuracy:  0.9279393173198482
validation set accuracy:  0.79
training set accuracy:  0.9279393173198482
validation set accuracy:  0.79
training set accuracy:  0.9279393173198482
validation set accuracy:  0.79
```

->트리의 성능은 79%라고 말하는 게 더 정확

2 Stage2 – Iris 문제 Feature Engineering

- **아이리스 Feature Engineering**

아이리스 꽃잎과 꽃받침의 너비, 길이 정보를 바탕으로 setosa 종인지 versicolor 종인지 혹은 virginica 종인지 구분하는 문제

1. Iris 데이터 불러오기

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
In [6]: df.describe()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

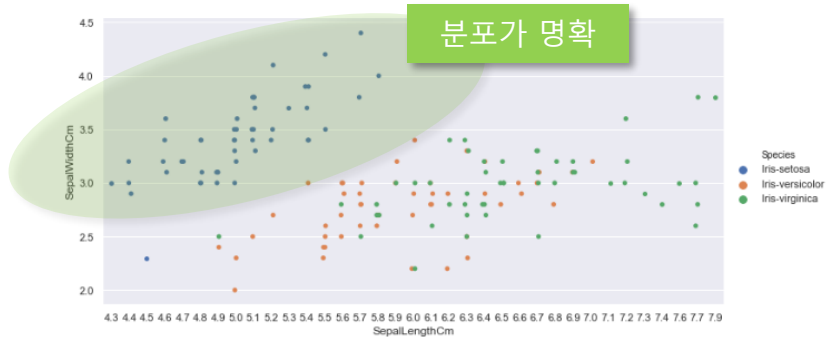
2. 꽃받침 길이, 너비에 따른 종 분포도 보기

```
In [7]: import matplotlib.pyplot as plt
        %matplotlib inline

        import seaborn as sns
        sns.set()
```

```
In [8]: sns.catplot(data=df, x='SepalLengthCm', y='SepalWidthCm', hue='Species', aspec
```

```
Out[8]: <seaborn.axisgrid.FacetGrid at 0x2154964b108>
```



2 Stage2 – Iris 문제 Feature Engineering

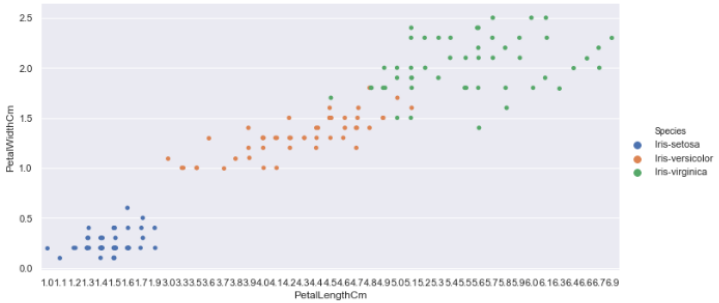
- **아이리스 Feature Engineering**

아이리스 꽃잎과 꽃받침의 너비, 길이 정보를 바탕으로 setosa 종인지 versicolor 종인지 혹은 virginica 종인지 구분하는 문제

3. (미션1) 꽃잎의 길이, 너비에 따른 종 분포도 보기

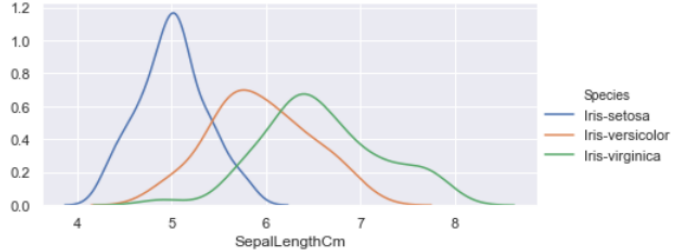
```
In [9]: sns.catplot(data=df, x='PetalLengthCm', y='PetalWidthCm', hue='Species', aspec
```

```
Out[9]: <seaborn.axisgrid.FacetGrid at 0x2154b9b9248>
```



4. kdeplot으로 꽃받침 길이별 밀집 수준 파악

```
In [11]: facet = sns.FacetGrid(df, hue='Species', aspect=2)
facet.map(sns.kdeplot, 'SepalLengthCm')
facet.add_legend()
plt.show()
```



꽃받침의 길이가 짧을수록 setosa 가능성 높음
길수록 virginica 가능성 높음

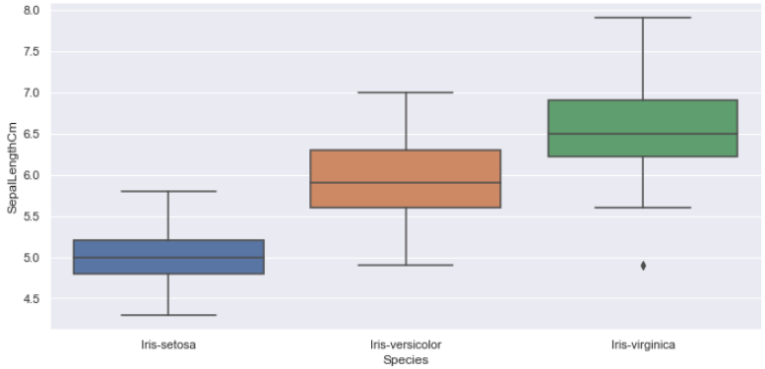
2 Stage2 – Iris 문제 Feature Engineering

- **아이리스 Feature Engineering**

아이리스 꽃잎과 꽃받침의 너비, 길이 정보를 바탕으로 setosa 종인지 versicolor 종인지 혹은 virginica 종인지 구분하는 문제

5. box plot으로 종별 꽃받침 길이 범위 파악

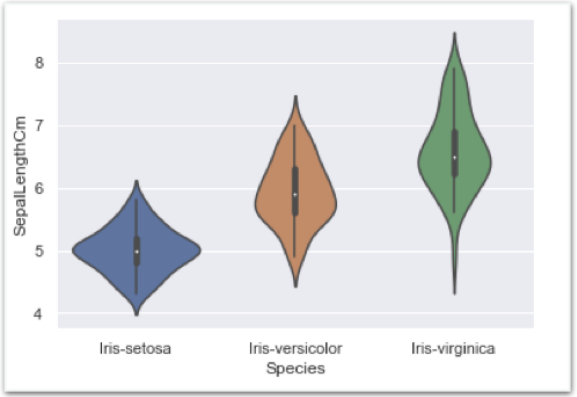
```
In [12]: sns.catplot(kind='box',data=df, x='Species', y='SepalLengthCm', aspect=2)
Out[12]: <seaborn.axisgrid.FacetGrid at 0x2154c54b688>
```



Box plot : 밀집도 수준 알기 어려움
Kdeplot : 밀집도 파악용

-> kdeplot + box plot -=> violin plot

6. 값의 범위(최소, 최대, 이상값)와 밀집 정도까지 시각적으로 한번에 알 수 있는 violin plot



3 Stage3 – Scikit-learn으로 Random Forest 구현하기

- **Scikit-learn으로 Random Forest 구현하기**

데이터 수가 적기 때문에, training/test 두 부분으로 나눌 것

In [3]:
```
input_data = df[['SepalLengthCm','SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
target_data = df['Species']
```

In [4]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(input_data, target_data)
x_train
```

Out[4]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
69	5.6	2.5	3.9	1.1
130	7.4	2.8	6.1	1.9
38	4.4	3.0	1.3	0.2
36	5.5	3.5	1.3	0.2
115	6.4	3.2	5.3	2.3
...	...	...	...	...
40	5.0	3.5	1.3	0.3
136	6.3	3.4	5.6	2.4
85	6.0	3.4	4.5	1.6
63	6.1	2.9	4.7	1.4
73	6.1	2.8	4.7	1.2

112 rows × 4 columns

랜덤으로 섞인 데이터가 4가지로 구성되는 것

Train_test_split 이용

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier()
tree.fit(x_train, y_train)

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

```
training set accuracy:  1.0
test set accuracy : 0.9736842105263158
```

X_test와 매칭

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

```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor'], dtype=object)
```

3 Stage3 – Scikit-learn으로 Random Forest 구현하기

- 사이킷런의 랜덤포레스트

```
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier(n_estimators=10)
forest.fit(x_train, y_train)

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

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

몇 개의 의사결정 나무를 앙상블할까?
디폴트 : 100개

```
training set accuracy:  0.9910714285714286
test set accuracy : 0.9473684210526315
```

더 나은 성능

```
array(['Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor'], dtype=object)
```

4

Stage4 – 고객분석 : 누가 내 우수고객이 될까?

1. 숫자로 바꾸기 (성별, 지역, 관심사)

```
df.loc[ df['성별'] == 'male', '성별'] = 0
df.loc[ df['성별'] == 'female', '성별'] = 1
```

```
df
```

	ID	우수고객분류	이름	나이	성별	지역	관심사
0	cdg	1	최도근	10	0	서울	축구
1	leekun	0	이헌영	42	0	서울	축구
2	son7	0	손흥민	38	1	경기	esports
3	parklangers	1	박지성	12	0	서울	축구
4	bluedragon	0	이청용	22	1	경기	esports
5	tuna	1	지동원	11	0	서울	야구
6	key	1	기성용	19	1	경기	축구
7	koojc	0	구자철	28	0	제주	esports

```
df.loc[ df['지역'] == '서울', '지역'] = 0
df.loc[ df['지역'] == '제주', '지역'] = 1
df.loc[ df['지역'] == '경기', '지역'] = 2

df.loc[ df['관심사'] == '축구', '관심사'] = 0
df.loc[ df['관심사'] == 'esports', '관심사'] = 1
df.loc[ df['관심사'] == '야구', '관심사'] = 2
```

2. input, target 데이터 설정

```
#input
x_train = df[['나이', '성별', '지역', '관심사']]
#target
y_train = df['우수고객분류']
```

이름과 ID는 우수고객에 영향이 없을 것이므로 제외

3. 랜덤 포레스트

```
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier(n_estimators = 10)
forest.fit(x_train, y_train)

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

```
training set accuracy: 1.0
```

4 Stage4 – 고객분석 : 누가 내 우수고객이 될까?

1. Test data 확인 (우수고객 여부 부재)

```
df_test = pd.read_csv('data/new_customers.csv')
df_test
```

	ID	이름	나이	성별	지역	관심사
0	tseter1	이강인	10	male	서울	축구
1	tester2	이승우	25	female	경기	야구

2. 숫자로 바꾸기 + 'df' -> 'df_test'

```
df_test.loc[ df_test['성별'] == 'male', '성별'] = 0
df_test.loc[ df_test['성별'] == 'female', '성별'] = 1

df_test.loc[ df_test['지역'] == '서울', '지역'] = 0
df_test.loc[ df_test['지역'] == '제주', '지역'] = 1
df_test.loc[ df_test['지역'] == '경기', '지역'] = 2

df_test.loc[ df_test['관심사'] == '축구', '관심사'] = 0
df_test.loc[ df_test['관심사'] == 'esports', '관심사'] = 1
df_test.loc[ df_test['관심사'] == '야구', '관심사'] = 2
```

```
df_test
```

3. Prediction

```
x_test = df_test[['나이', '성별', '지역', '관심사']]

prediction = forest.predict(x_test)
prediction
```

```
array([1, 0], dtype=int64)
```

```
x_test
```

	나이	성별	지역	관심사
0	10	0	0	0
1	25	1	2	2

```
forest.predict([[25, 1, 2, 2], [25, 2, 1, 0]])
```

```
array([0, 0], dtype=int64)
```

각 정보를 list로 넣어도 가능

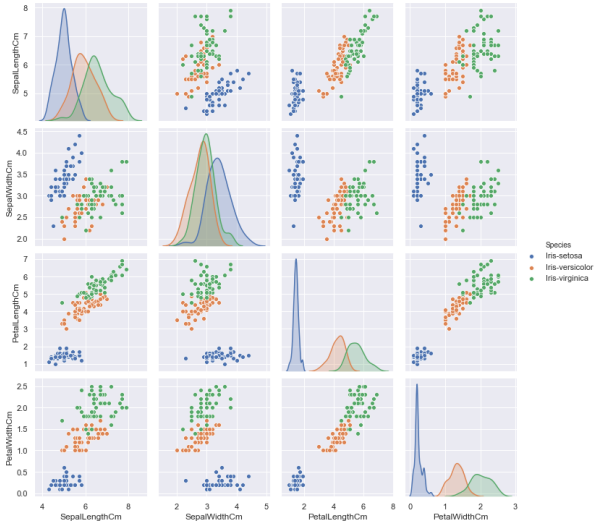
5 Challenge 1 – pair plot 그리기

```
import pandas as pd

df = pd.read_csv('data/iris.csv')
df
```

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

```
sns.pairplot(data=df,hue='Species',
        x_vars=['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
        y_vars=['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']
```



5 Challenge 2 – Randomforest 적용하기

Decision Tree로 학습한 것 -> Randomforest로

```
from sklearn.ensemble import RandomForestClassifier

forest = RandomForestClassifier()
forest.fit(x_train, y_train)

forest.score(x_train, y_train)
```

```
0.9279393173198482
```

```
forest.score(x_valid, y_valid)
```

```
0.81
```

데이터 특성 때문에
딱히 randomforest에서
성능이 확 좋아지진 않는 상황

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

submit = pd.DataFrame({
    'PassengerId' : df_test['PassengerId'],
    'Survived' : prediction
})

submit.to_csv('submit_csv', index=False)
```