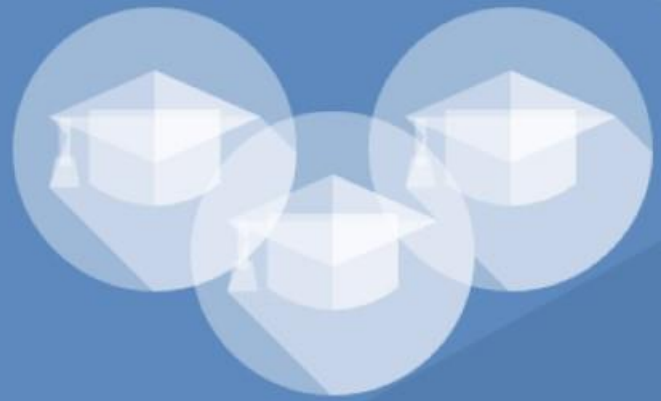
소득예측

20181108256 김현수



INDEX 목차

PART.1

개요

누군가가 1년에 5만 달러 이상을 벌고 있는지 아닌지를
정확하게 예측하는 것을 목표로 함

PART.2

사용 라이브러리

Numpy
행렬이나 일반적으로 대규모 다차원 배열을 쉽게 처리할 수 있도록 지원하는 멋진 라이브러리

Pandas
데이터 조작 및 분석을 위한 Python 프로그래밍 언어용으로 작성된 멋진 라이브러리

Seaborn
파이썬 데이터를 시각화해주는 멋진 라이브러리

PART.2

사용 라이브러리

matplotlib
다양한 데이터를 많은 방법으로 도식화 할 수 있도록 하는 멋진 파이썬 라이브러리

Scikit-Learn
파이썬 프로그래밍 언어용 기계 학습 라이브러리

PART.3

데이터 소개

	id	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
0	0	40	Private	168538	HS-grad	9	Married-civ-spouse	Sales	Husband	White	Male	0	0	60	United-States	>50K
1	1	17	Private	101626	9th	5	Never-married	Machine-op-inspct	Own-child	White	Male	0	0	20	United-States	<=50K
2	2	18	Private	353358	Some-college	10	Never-married	Other-service	Own-child	White	Male	0	0	16	United-States	<=50K
3	3	21	Private	151158	Some-college	10	Never-married	Prof-specialty	Own-child	White	Female	0	0	25	United-States	<=50K
4	4	24	Private	122234	Some-college	10	Never-married	Adm-clerical	Not-in-family	Black	Female	0	0	20	?	<=50K

- age: 나이
- workclass: 고용형태
- fnlwgt: 사람 대표성을 나타내는 가중치(final weight의 약자)
- education: 교육 수준
- education_num: 교육 수준 수치
- marital_status: 결혼 상태
- occupation: 업종
- relationship: 가족 관계
- race: 인종
- sex: 성별
- capital_gain: 양도 소득
- capital_loss 양도 손실
- hours_per_week: 주당 근무 시간
- native_country: 국적
- income: 수익 (예측해야 하는 값) (>50K: 1 <=50K: 0)

PART.3

데이터 전처리

```
# 데이터 읽어 드리기
dataset = pd.read_csv("../input/adult.csv")
```

```
# >50K인 데이터와 <=50K인 데이터 갯수
sns.countplot(dataset['income'],label="Count")
sns.plt.show()
```

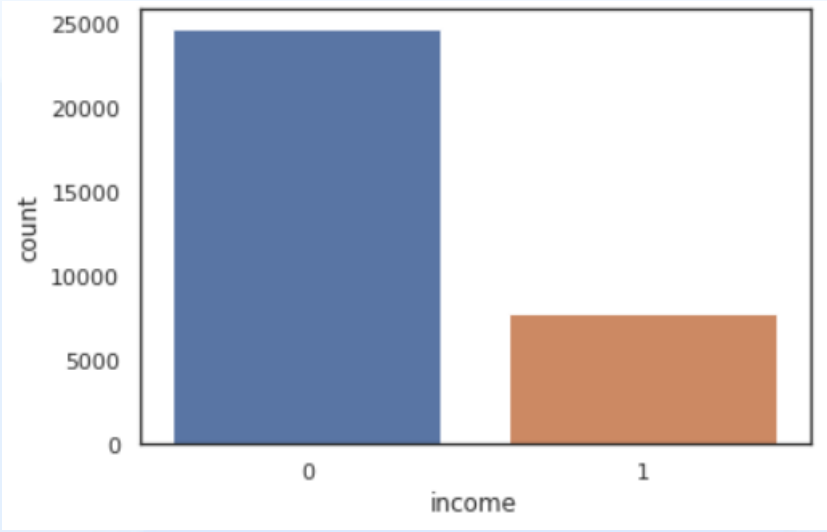
```
# 예측하려는 항목을 포매팅
dataset['income']=dataset['income'].map({'<=50K': 0, '>50K': 1, '<=50K.': 0, '>50K.': 1})
```

PART.3

데이터 전처리

```
# >50K인 데이터와 <=50K인 데이터 갯수
sns.countplot(dataset['income'],label="Count")
sns.plt.show()
```

countplot: 각 카테고리 값별로 데이터가 얼마나 있는지 표시할 수 있다.

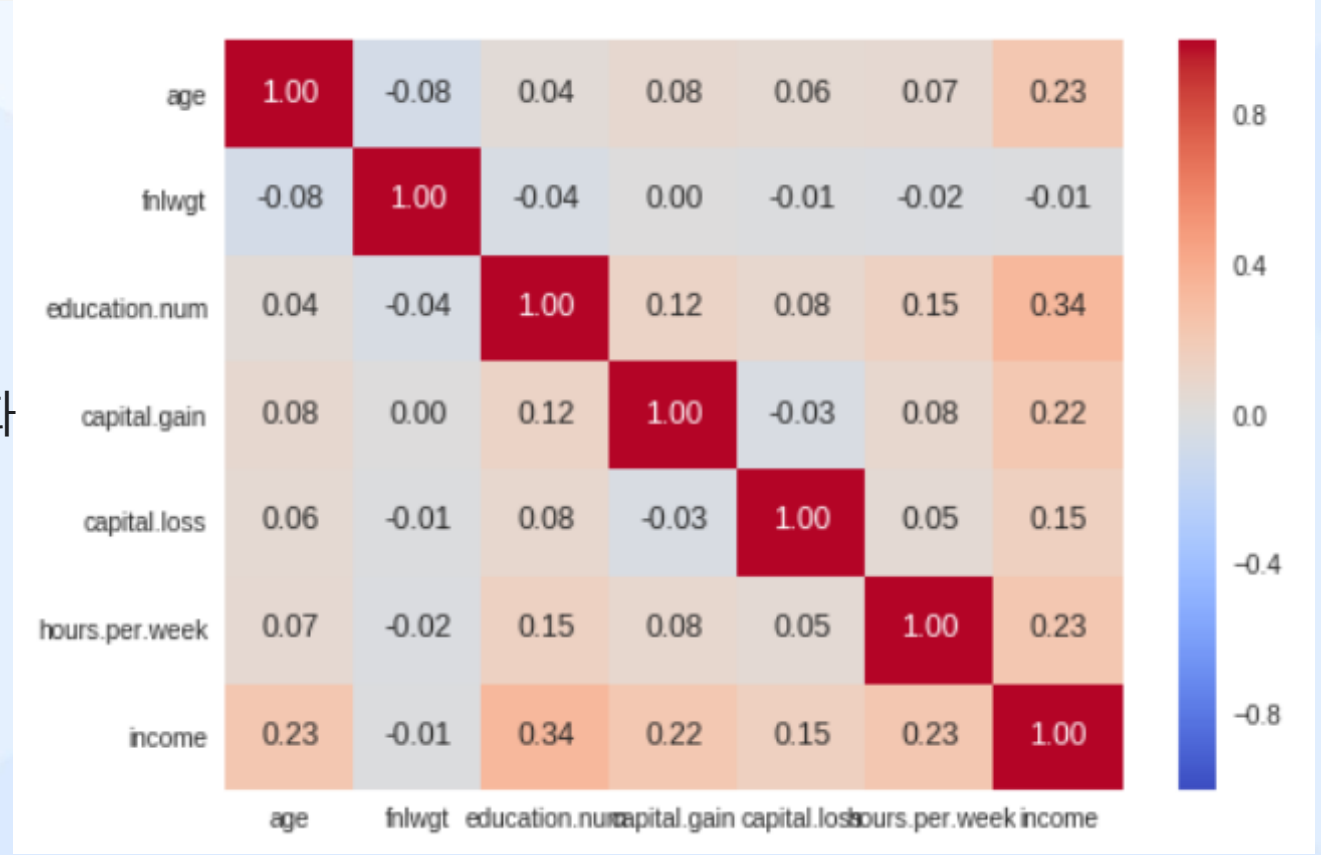


PART.3

데이터 전처리

```
# 값 들간의 상관 관계
g = sns.heatmap(dataset[numeric_features].corr(),annot=True, fmt = ".2f", cmap = "coolwarm")
sns.plt.show()
```

Heatmap: 데이터들의 배열 을 색상으로 표현해주는 그래프이다
heatmap을 사용하면 두 개의 카테고리 값에 대한 값 변화를 한눈에 알기 쉽다.
대용량 데이터도 heatmap을 이용해 시각화 한다면 이미지 몇장으로 표현이 가능하다.



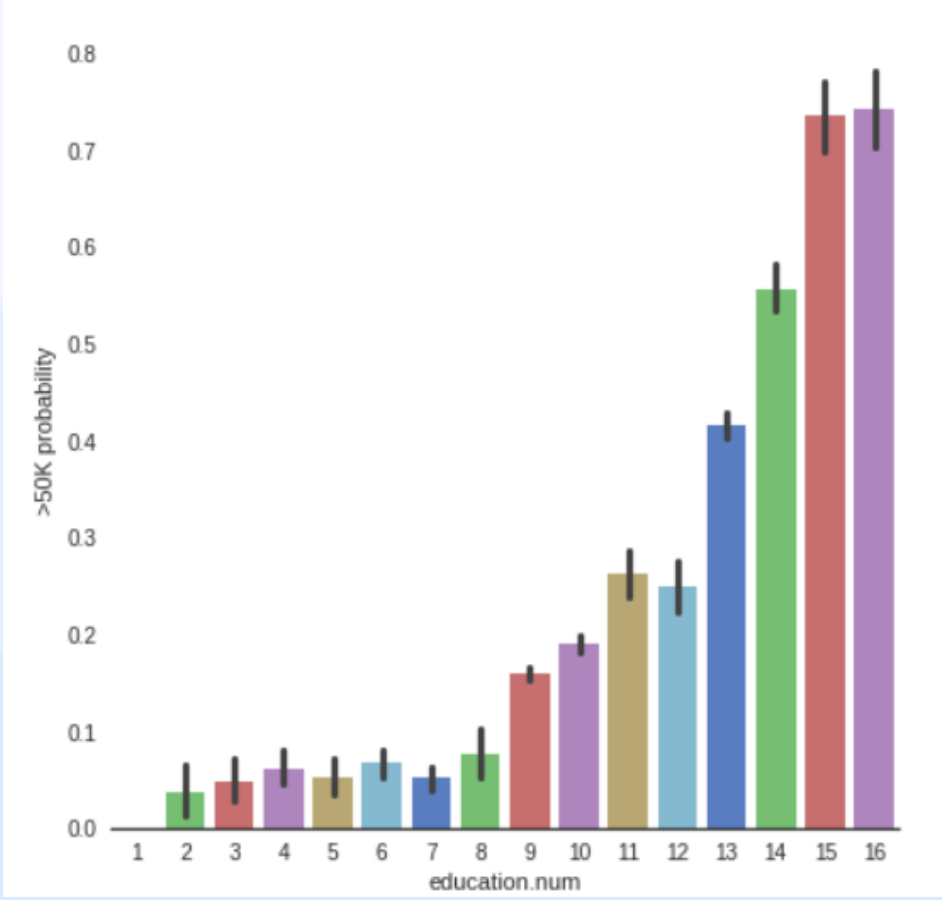
PART.3

데이터 전처리

```
# 교육 수준수치와 소득 비교
g = sns.factorplot(x="education.num",y="income",data=dataset,kind="bar",size = 6,palette = "muted")
g.despine(left=True)
g = g.set_ylabels(">50K probability")
```

•sns.factorplot(x,y,hue) : y평균값 그래프. hue로 지정된 필드의 종류만큼 라인이 그려짐.

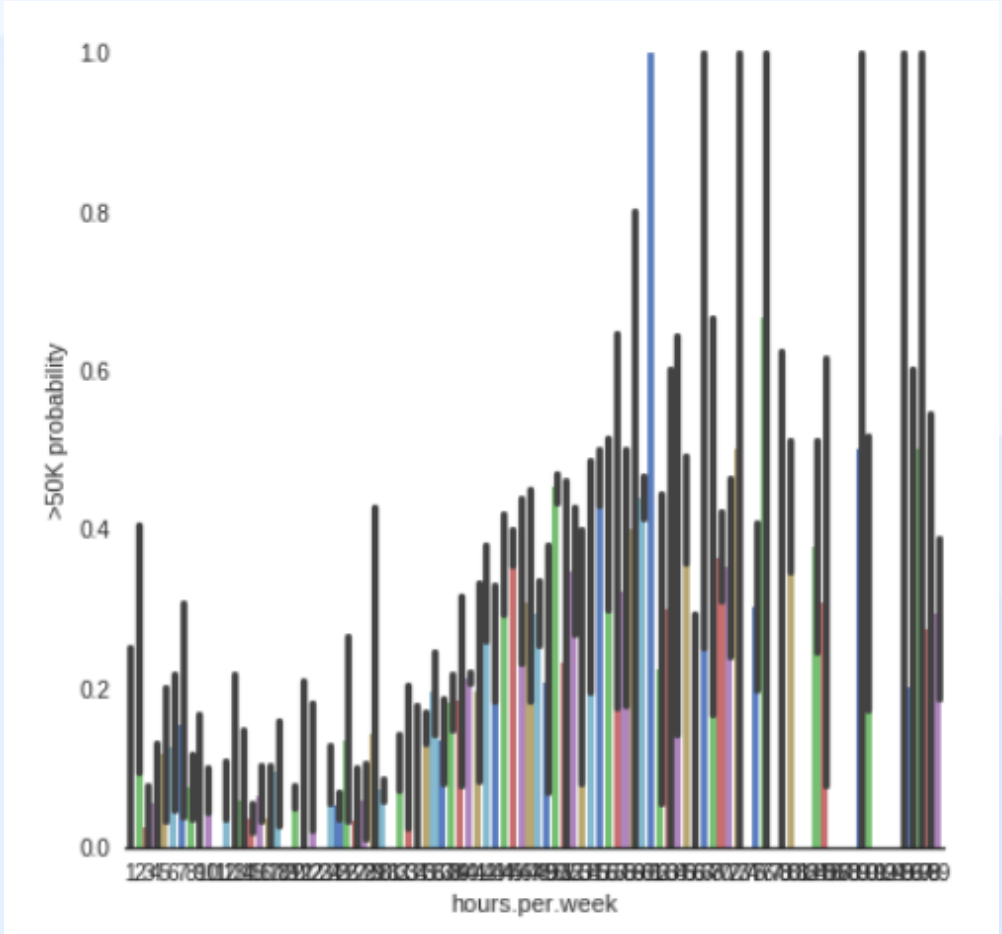
plot을 지정한 후, 그 뒤에 sns.despine() 이라고 지정하면 테두리를 제거한다.



PART.3

데이터 전처리

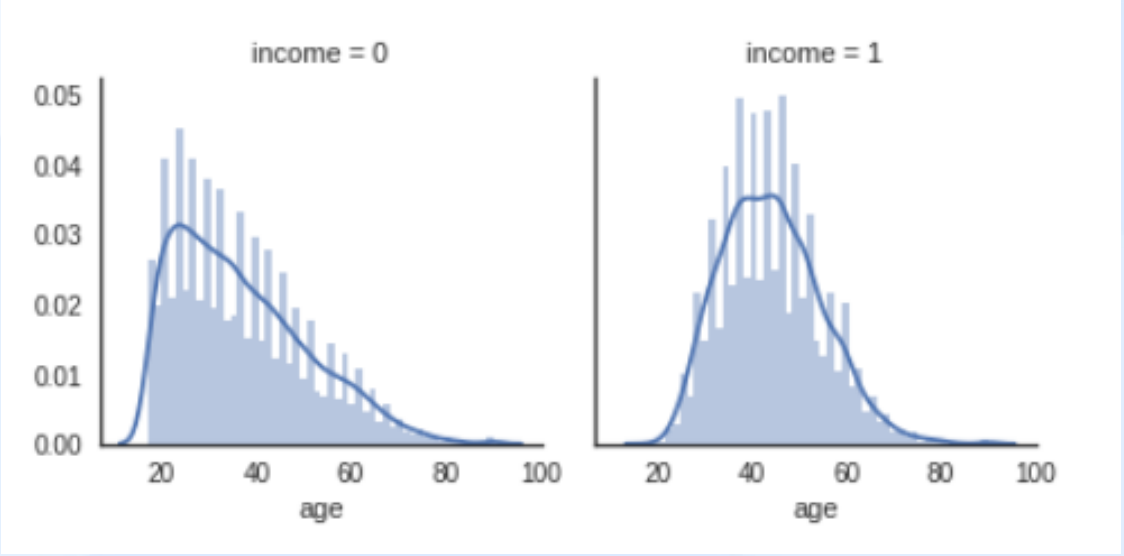
```
# 주당 시간과 소득 비교
g   = sns.factorplot(x="hours.per.week",y="income",data=dataset,kind="bar",size = 6,palette = "muted")
g.despine(left=True)
g = g.set_ylabels(">50K probability")
```



PART.3

데이터 전처리

```
# 나이와 소득 비교
g = sns.FacetGrid(dataset, col='income')
g = g.map(sns.distplot, "age")
sns.plt.show()
```



PART.3

데이터 전처리

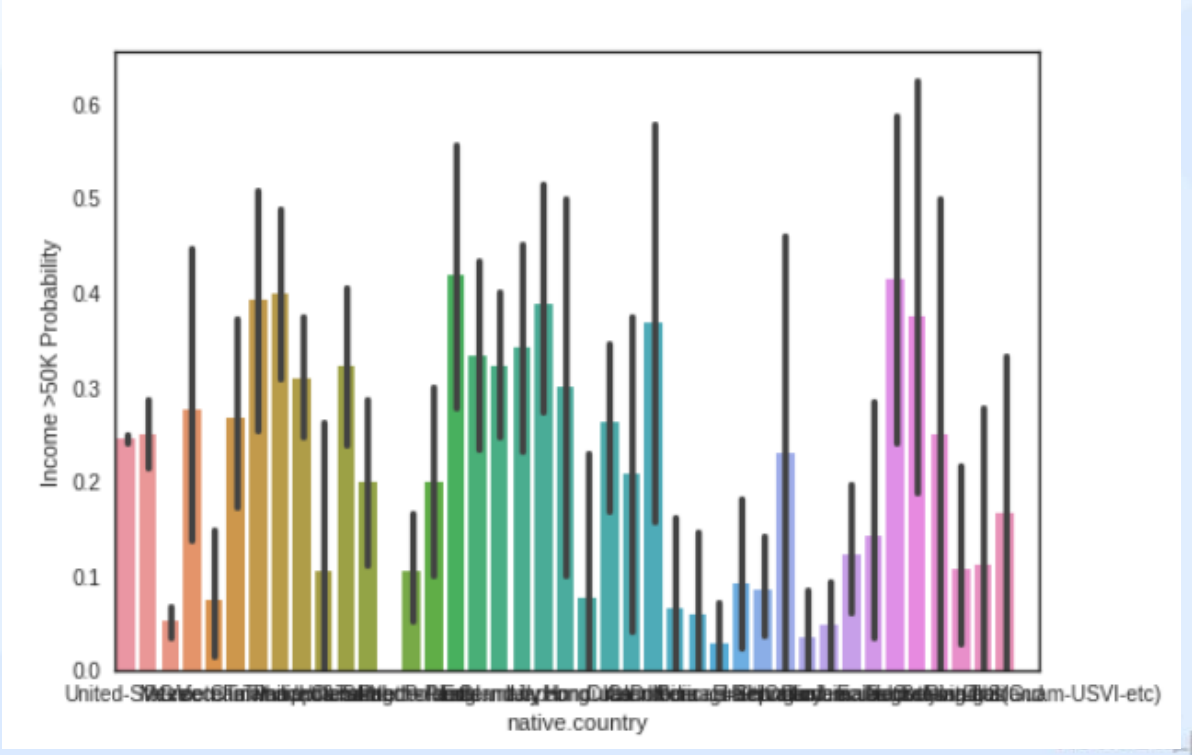
```
# 누락된 항목 채우기
dataset["workclass"] = dataset["workclass"].fillna("X")
dataset["occupation"] = dataset["occupation"].fillna("X")
dataset["native.country"] = dataset["native.country"].fillna("United-States")
```

PART.3

데이터 전처리

```
# 지역 소득비교
g = sns.barplot(x="native.country",y="income",data=dataset)
g = g.set_ylabel("Income >50K Probability")
sns.plt.show()
```

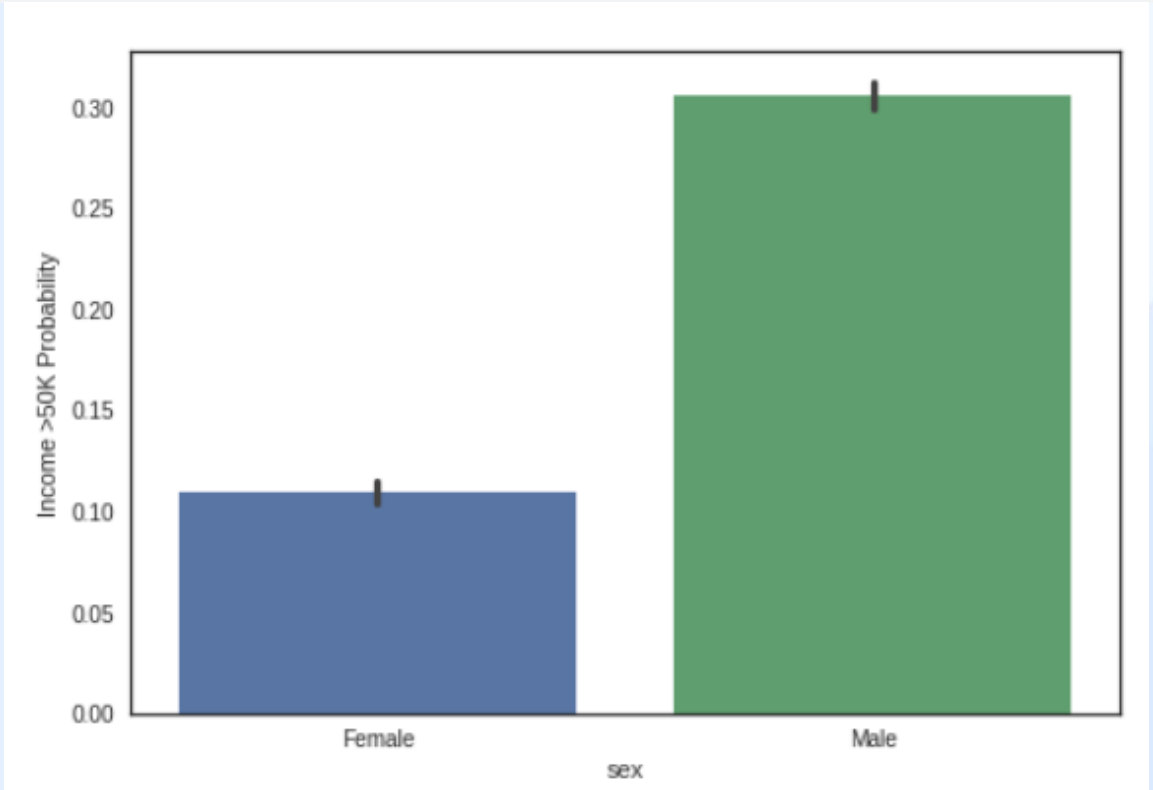
barplot: 막대 그래프 생성
x인자에는 x축 y인자에는 y축을 설정



PART.3

데이터 전처리

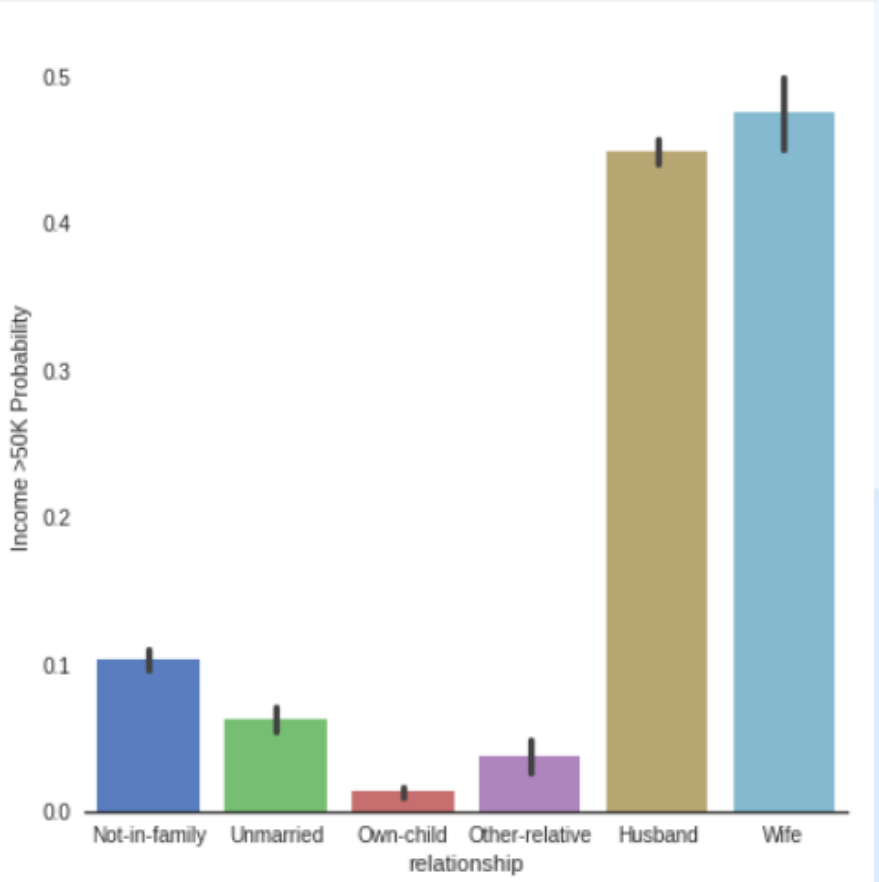
```
# 성별 소득 비교
g = sns.barplot(x="sex",y="income",data=dataset)
g = g.set_ylabel("Income >50K Probability")
sns.plt.show()
```



PART.3

데이터 전처리

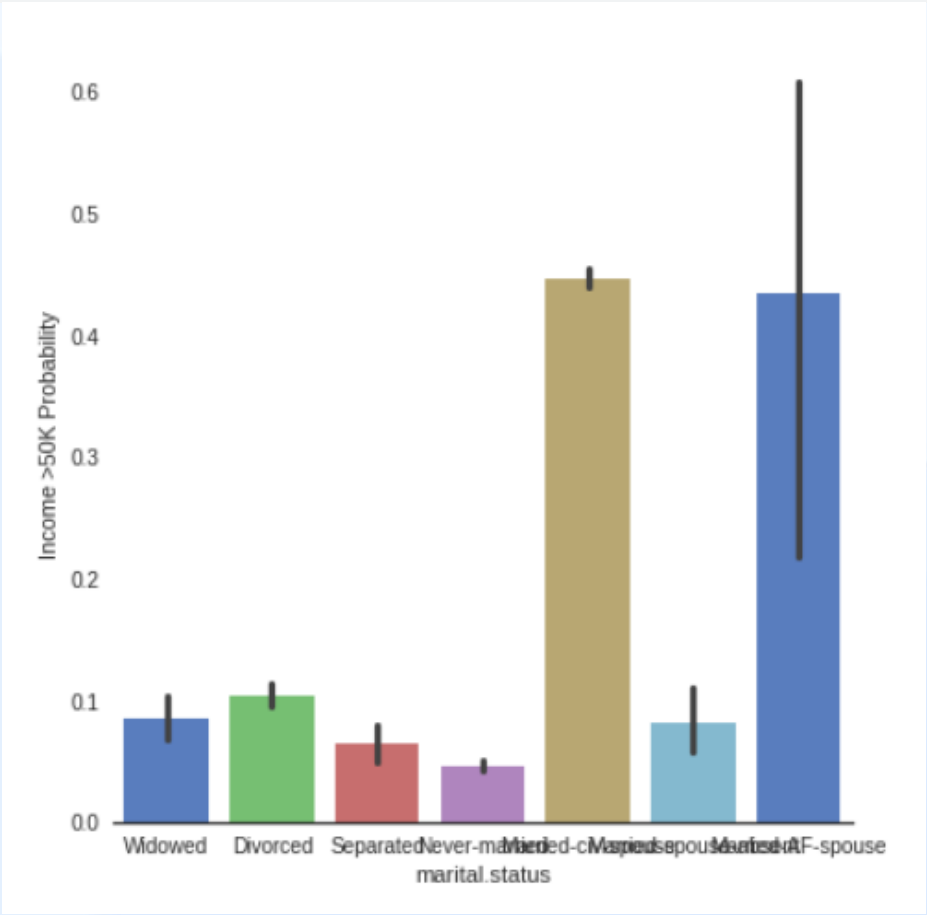
```
# 가족관계별 소득 비교
g = sns.factorplot(x="relationship",y="income",data=dataset,kind="bar", size = 6 ,palette = "muted")
g.despine(left=True)
g = g.set_ylabels("Income >50K Probability")
sns.plt.show()
```



PART.3

데이터 전처리

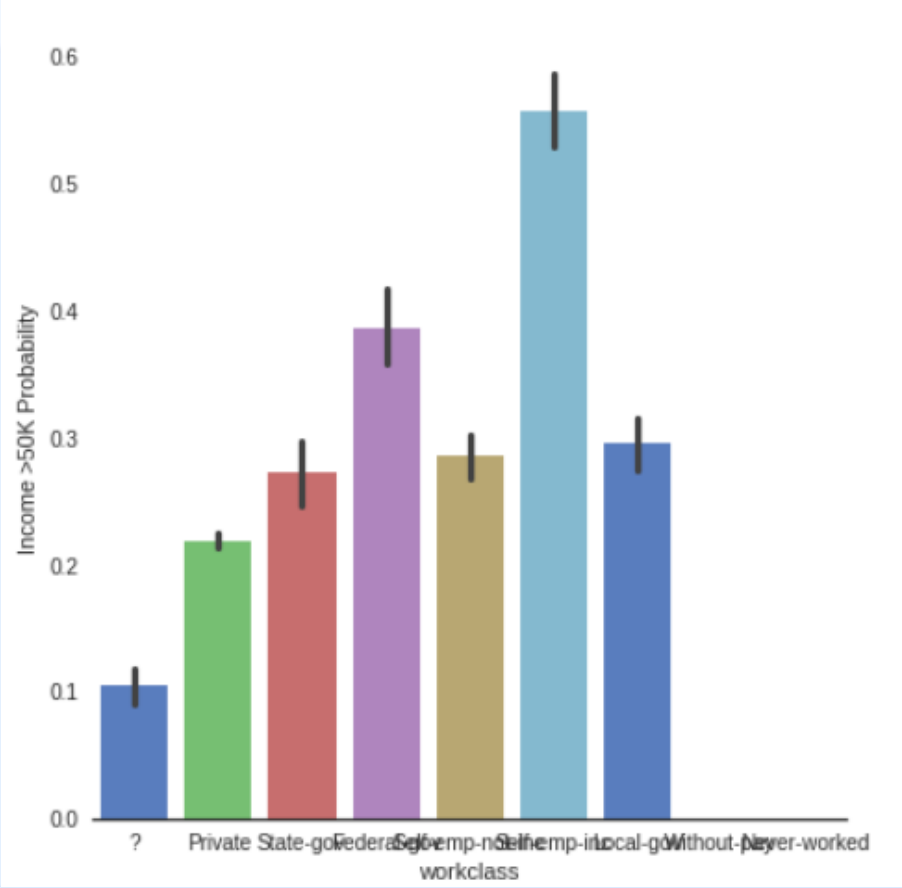
```
# 결혼 여부별 소득 비교
g = sns.factorplot(x="marital.status",y="income",data=dataset,kind="bar", size = 6 ,palette = "muted")
g.despine(left=True)
g = g.set_ylabels("Income >ㄴ50K Probability")
sns.plt.show()
```



PART.3

데이터 전처리

```
# 고용형태 별 소득 비교
g = sns.factorplot(x="workclass",y="income",data=dataset,kind="bar", size = 6 ,palette = "muted")
g.despine(left=True)
g = g.set_ylabels("Income >50K Probability")
sns.plt.show()
```



PART.3

데이터 전처리

```
# 성별을 0/1로 변환
dataset["sex"] = dataset["sex"].map({"Male": 0, "Female":1})
```

```
# 결혼 여부 컬럼 만들기 결혼1 미혼0
dataset["marital.status"] = dataset["marital.status"].replace(['Never-married','Divorced','Separated','Widowed'], 'Single')
dataset["marital.status"] = dataset["marital.status"].replace(['Married-civ-spouse','Married-spouse-absent','Married-AF-spouse'], 'Married')
dataset["marital.status"] = dataset["marital.status"].map({"Married":1, "Single":0})
dataset["marital.status"] = dataset["marital.status"].astype(int)
```

PART.3

데이터 전처리

```
# 사용하지 않는 데이터 삭제
dataset.drop(labels=["workclass","education","occupation","relationship","race","native.country"], axis = 1, inplace = True)
print('Dataset with Dropped Labels')
print(dataset.head())
```

```
Dataset with Dropped Labels
   age  fnlwgt  education.num  marital.status  sex  capital.gain  capital.loss  hours.per.week  income
0   90   77053              9               0    1             0          4356              40       0
1   82  132870              9               0    1             0          4356              18       0
2   66  186061             10               0    1             0          4356              40       0
3   54  140359              4               0    1             0          3900              40       0
4   41  264663             10               0    1             0          3900              40       0
```

PART.4

데이터 학습

```
# 학습 데이터와 검증 데이터 분할
array = dataset.values
X = array[:,0:8]
Y = array[:,8]
print('Split Data: X')
print(X)
print('Split Data: Y')
print(Y)
validation_size = 0.20
seed = 7
num_folds = 10
scoring = 'accuracy'
X_train, X_validation, Y_train, Y_validation = train_test_split(X,Y,test_size=validation_size,random_state=seed)
```

```
Split Data: X
[[    90  77053      9 ...,      0   4356     40]
 [    82 132870      9 ...,      0   4356     18]
 [    66 186061     10 ...,      0   4356     40]
 ...,
 [    40 154374      9 ...,      0      0     40]
 [    58 151910      9 ...,      0      0     40]
 [    22 201490      9 ...,      0      0     20]]
Split Data: Y
[0 0 0 ..., 1 0 0]
```

PART.4

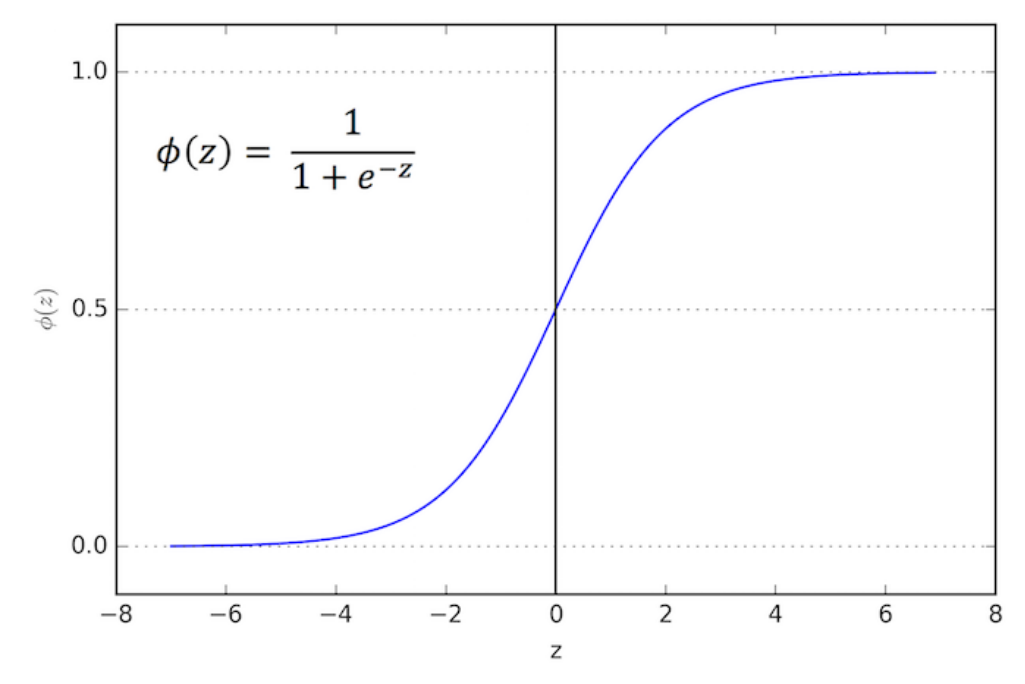
데이터 학습

```
#랜덤 포레스트와 5가지의 알고리즘 (LR, LDA, KNN, CART, GNB)
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('RF', RandomForestClassifier(n_estimators=num_trees, max_features=max_features)))
```

PART.4

데이터 학습

LogisticRegression

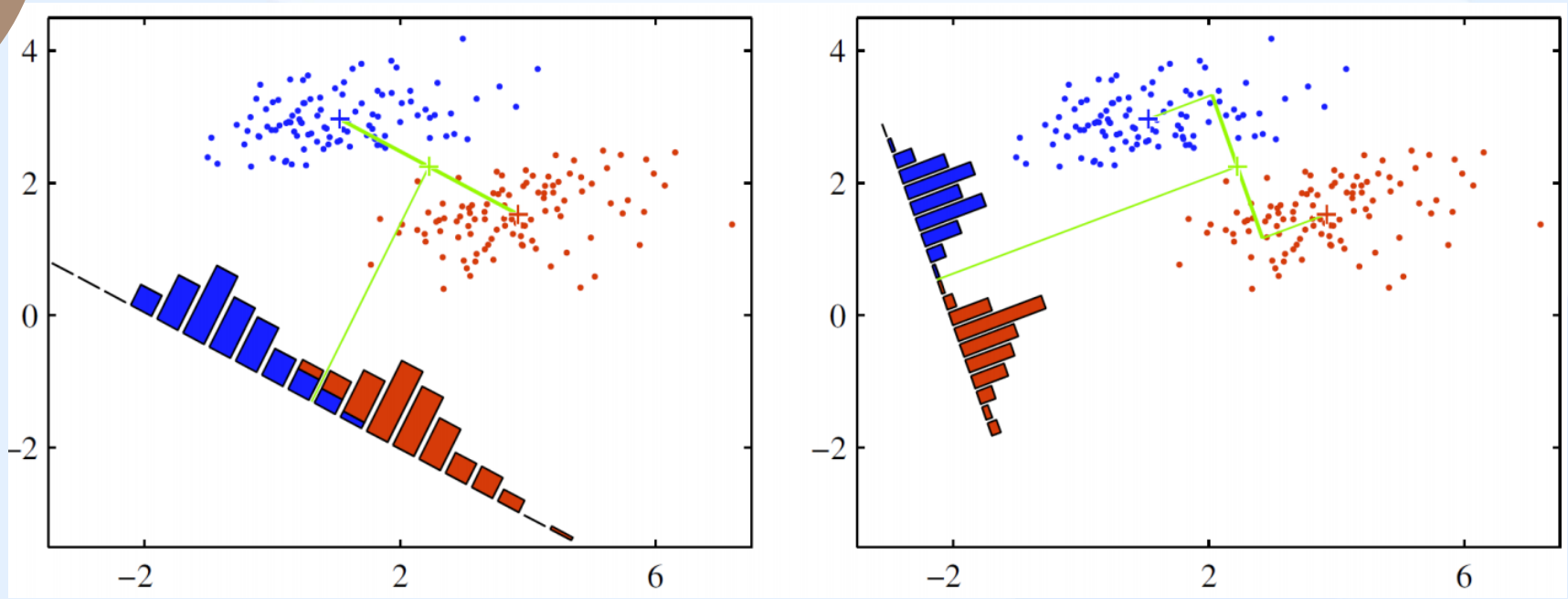


Logistic Regression은 데이터가 특정 카테고리에 속할지를0과 1사이의 연속적인 확률로 예측하는 회귀 알고리즘

데이터 학습

PART.4

LinearDiscriminantAnalysis

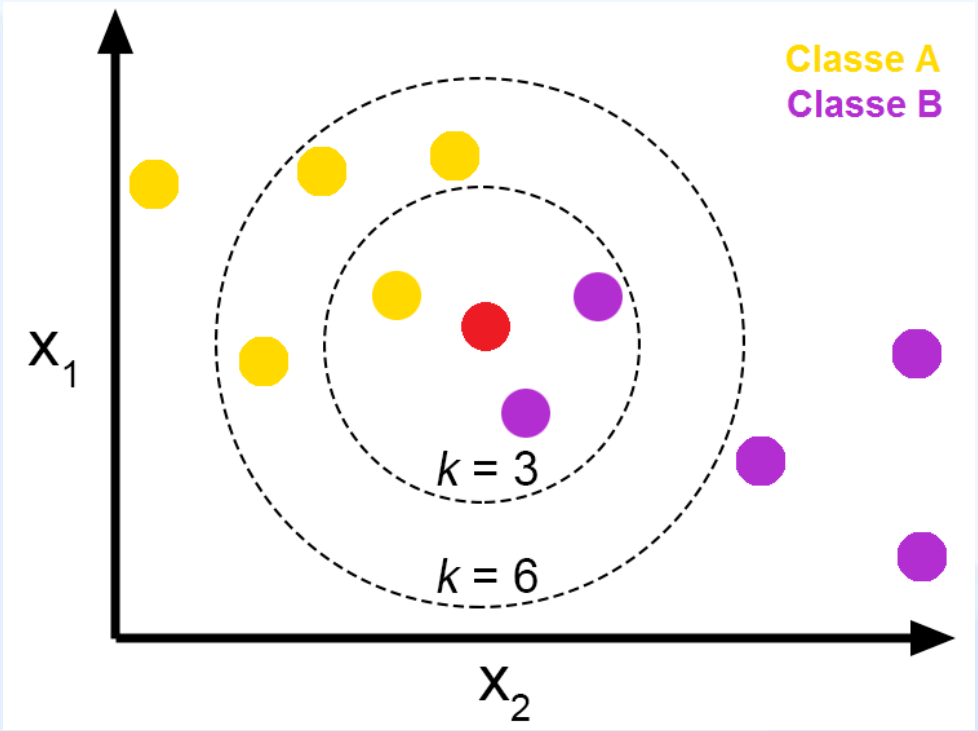


데이터 분포를 학습해 **결정경계(Decision boundary)**를 만들어 데이터를 **분류(classification)**하는 모델

PART.4

데이터 학습

KNeighborsClassifier

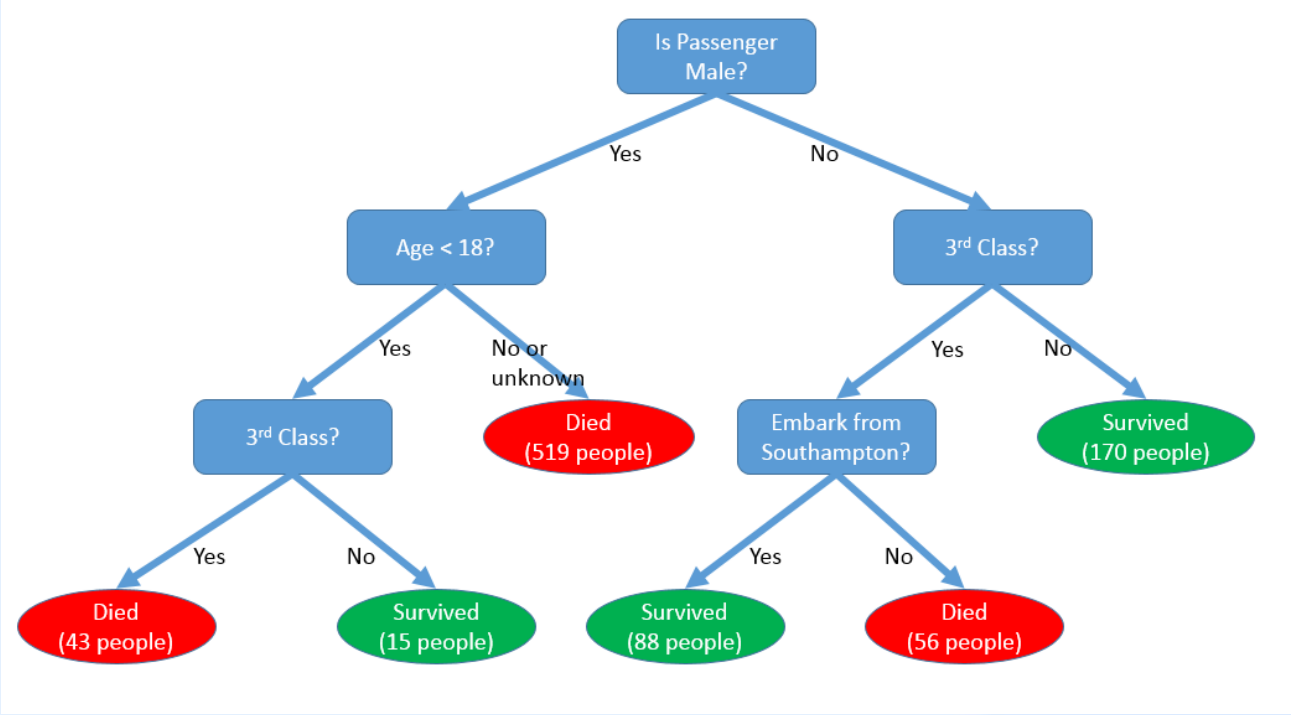


어떤 데이터가 주어지면 그 주변(이웃)의 데이터를 살펴본 뒤 더 많은 데이터가 포함되어 있는 범주로 분류하는 방식

PART.4

데이터 학습

DecisionTreeClassifier

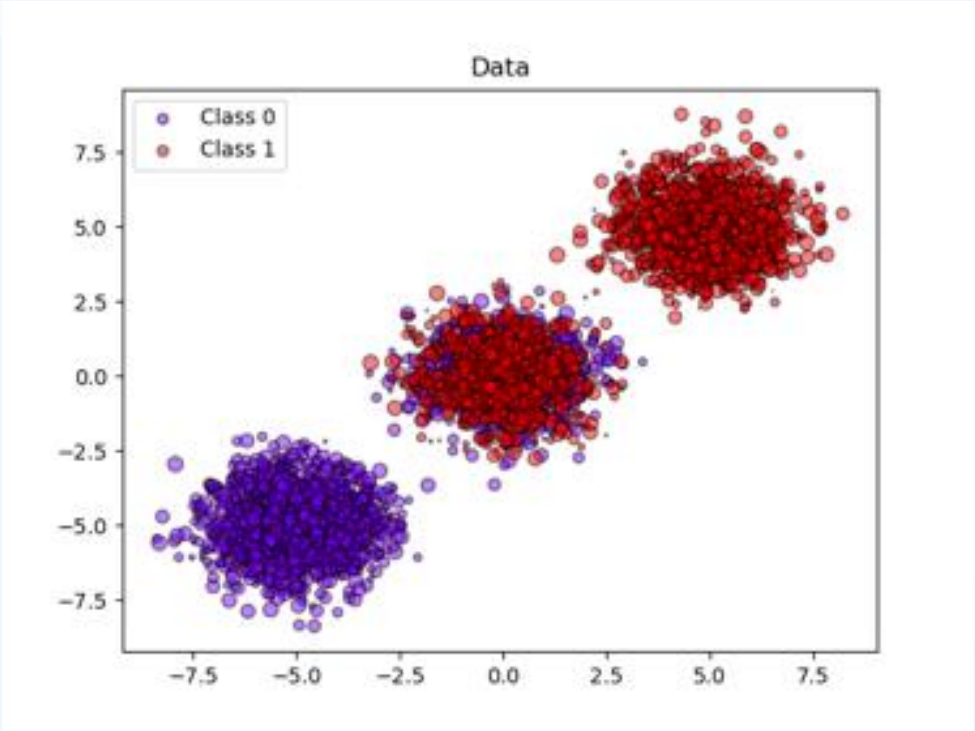


데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 트리 기반의 분류 규칙을 만드는 알고리즘

PART.4

데이터 학습

DecisionTreeClassifier

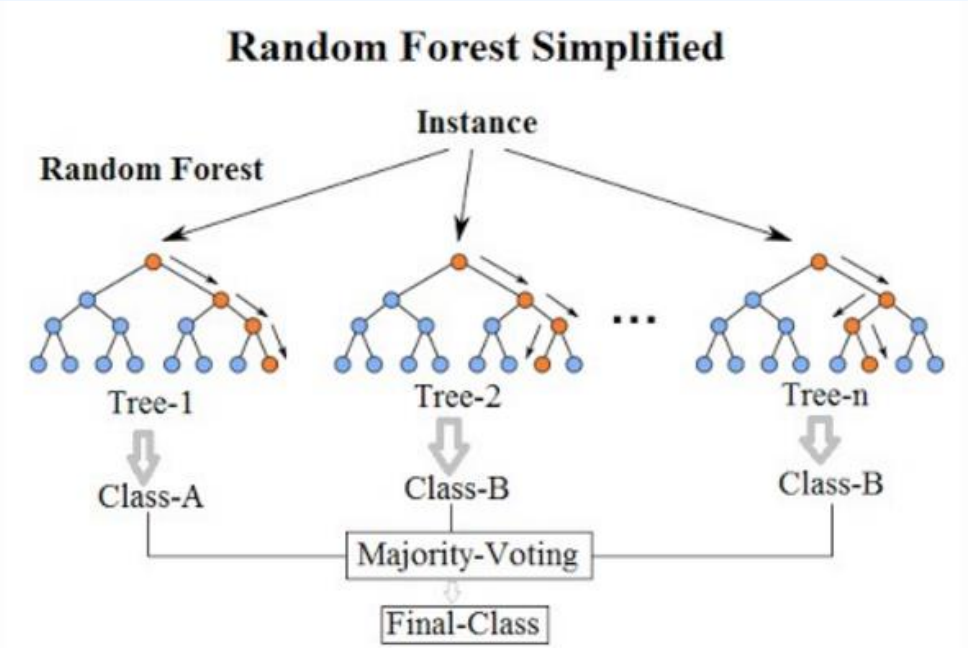


나이브 베이즈란 각 특성을 개별로 취급해 파라미터를 학습하고 그 특성에서 클래스별 통계를 단순하게 취합시킨다
GaussianNB는 연속적인 어떤 데이터를 다룬다

PART.4

데이터 학습

RandomForestClassifier



랜덤 포레스트는 여러 개의 결정 트리 분류기가 배깅을 기반으로 각자의 데이터를 샘플링 하여 학습을 수행한 후에 최종적으로 보팅을 통해 예측 결정을 하게 됩니다.
사이킷런에서는 RandomForestClassifier 클래스를 통해서 랜덤 포레스트 기반의 분류를 지원

PART.4

데이터 학습

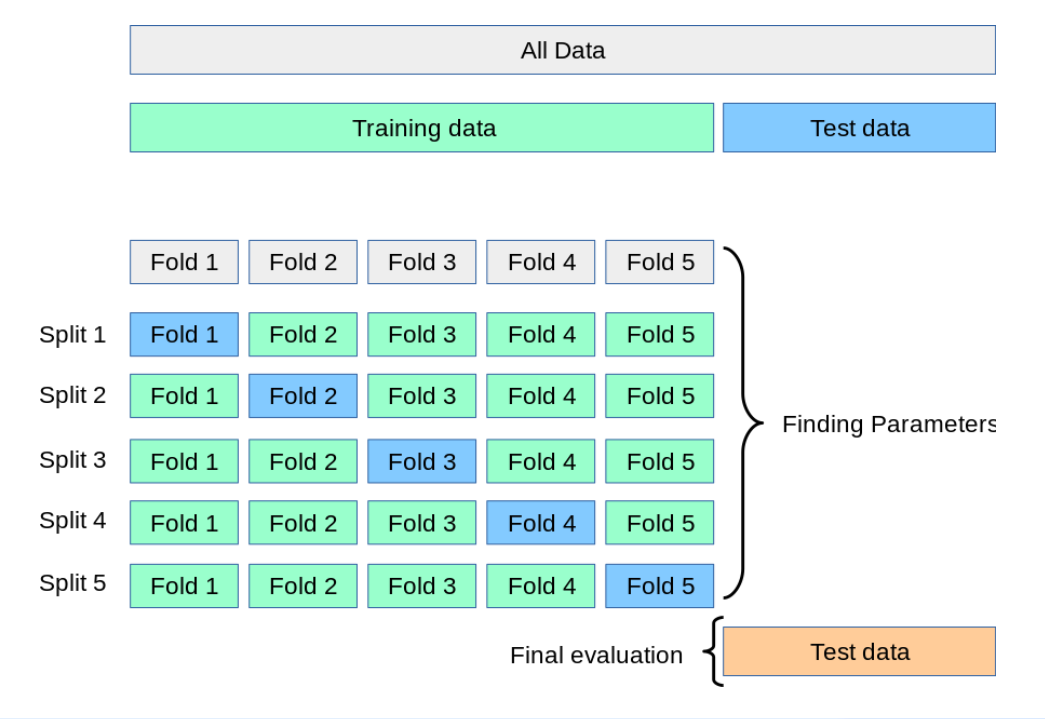
```
# 각 모델 평가
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, random_state=seed)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.796529 (0.004215)
LDA: 0.829507 (0.004318)
KNN: 0.774455 (0.005765)
CART: 0.808623 (0.005509)
NB: 0.794303 (0.003642)
RF: 0.840641 (0.004920)
```

PART.4

데이터 학습

KFold



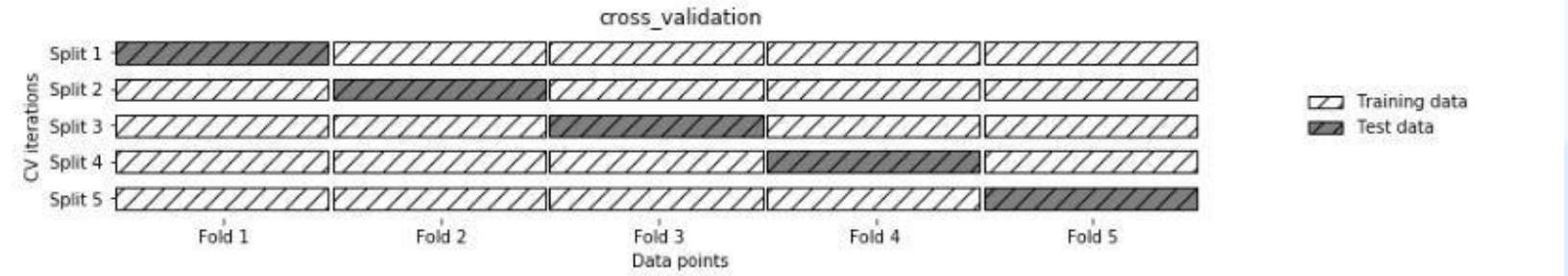
가장 보편적으로 사용되는 교차 검증 기법
먼저 K개의 데이터 폴드 세트를 만들어서 K번만큼 각 폴드 세트에 학습과 검증 평가를 1반복적으로 수행하는 방법

데이터 학습

PART.4

cross_val_score

```
import mglearn
mglearn.plots.plot_cross_validation()
```



교차검증을 위한 사이킷런함수
cross_val_score에 들어가는 매개변수는 (모델명, 훈련데이터, 타깃, CV)

데이터 학습

PART.4

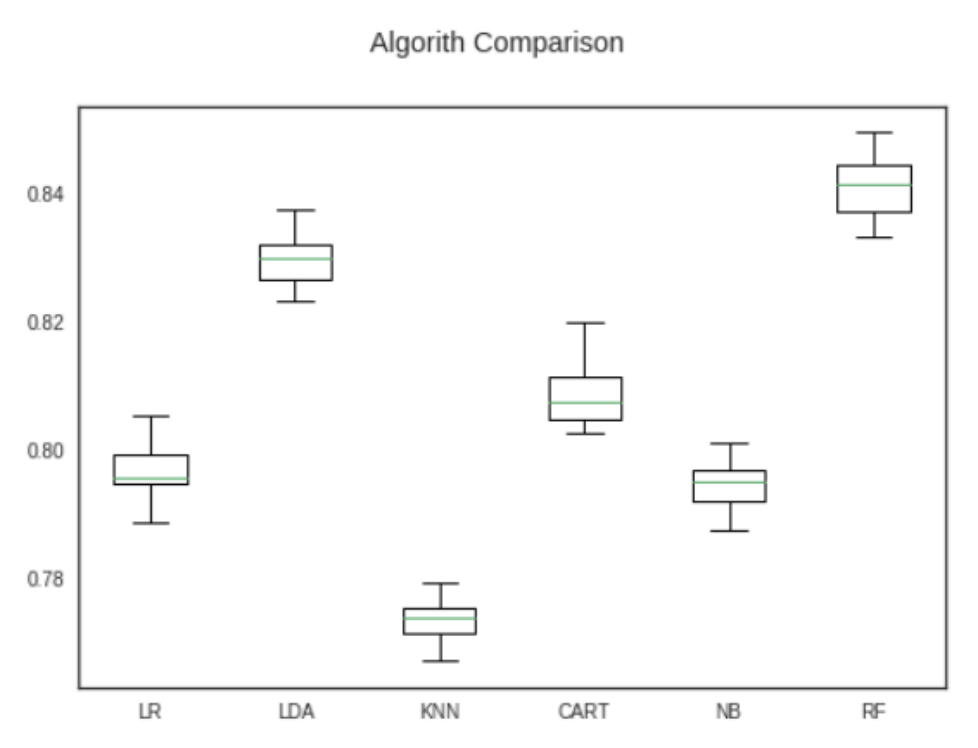
```
fig = plt.figure()
fig.suptitle('Algorith Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

figure()은 Figure인스턴스를 생성하는데 Figure인스턴스의 역할은 이미지 전체의 영역을 확보하는 것이다.
인수로 아래와 같은 항목을 지정할 수 있다.

plt.figure()에서는 이미지 영역의 확보뿐이므로, 그래프에는 아무것도 그려지지 않는다.

plt.figure()으로 그래프를 그리기 위해서는 subplot를 추가할 필요가 있다. subplot의 추가는, add_subplot 메소드를 사용한다.

111의 의미는, 1행째의 1열의 첫 번째라는 의미



PART.4

데이터 학습

```
random_forest = RandomForestClassifier(n_estimators=250,max_features=5)
random_forest.fit(X_train, Y_train)
predictions = random_forest.predict(X_validation)
print("Accuracy: %s%%" % (100*accuracy_score(Y_validation, predictions)))
```

```
Accuracy: 84.32366037156457%
```

Accuracy: 실제 데이터에서 예측 데이터가 얼마나 같은지를 판단하는 단순한 지표이다

마무리

PART.5

결론

여러가지의 알고리즘 중 랜덤 포레스트가 제일 정확했고 약84%의 정확성을 보였다

소감

처음 해보는 머신러닝이라 어려운것도 많았지만 Know-how가 아닌 Know-where를 명심하여 분석/예측을 해 뿌듯했다

THANK YOU

감사합니다.