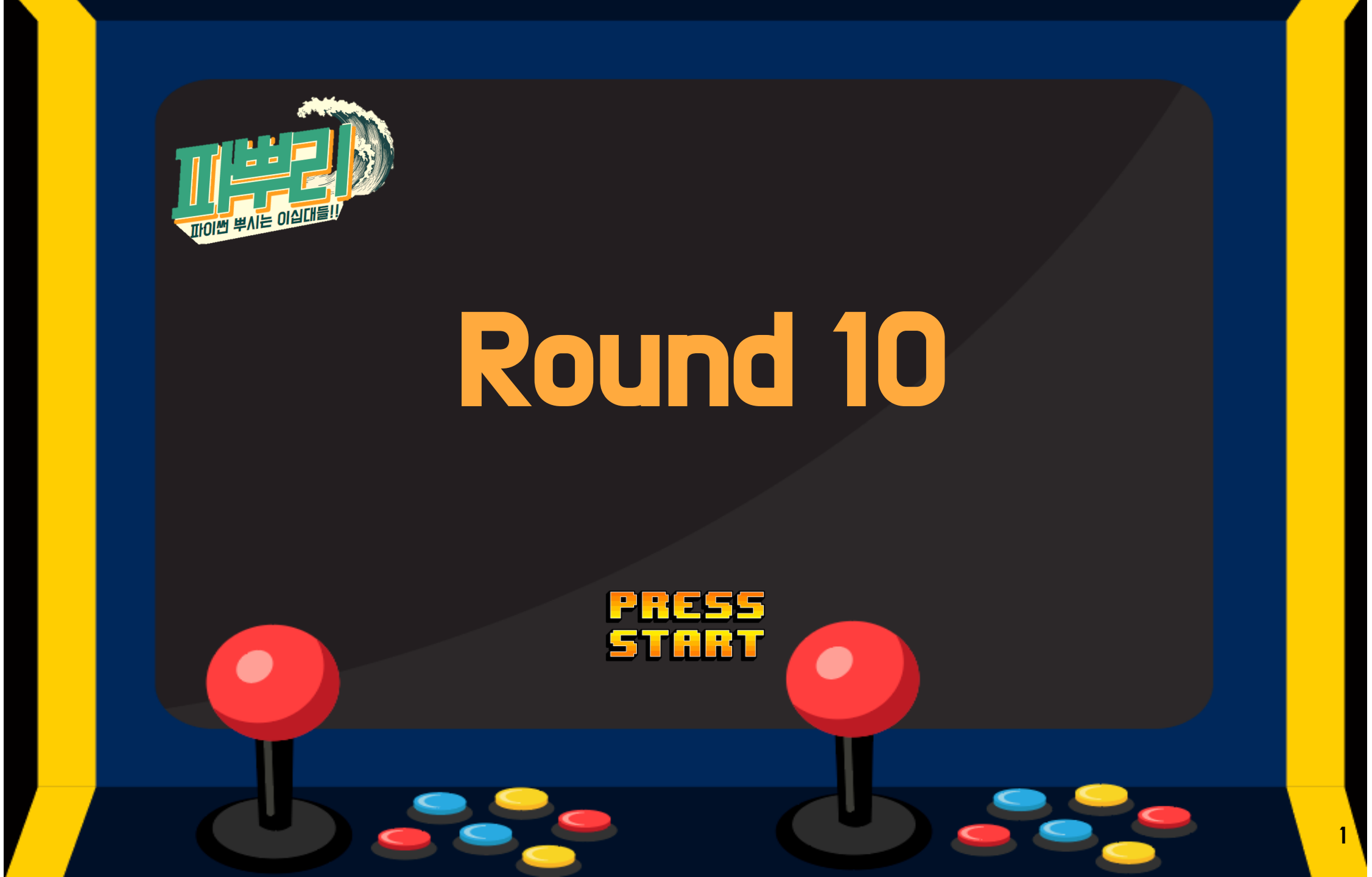

파뿌리
파이썬 뿌시는 이십대들!!
Round 10
PRESS
START

파뿌리
파이썬 뿌시는 이십대들!!

《 Round 10 》

- 분류분석 개요
- 분류분석 실습

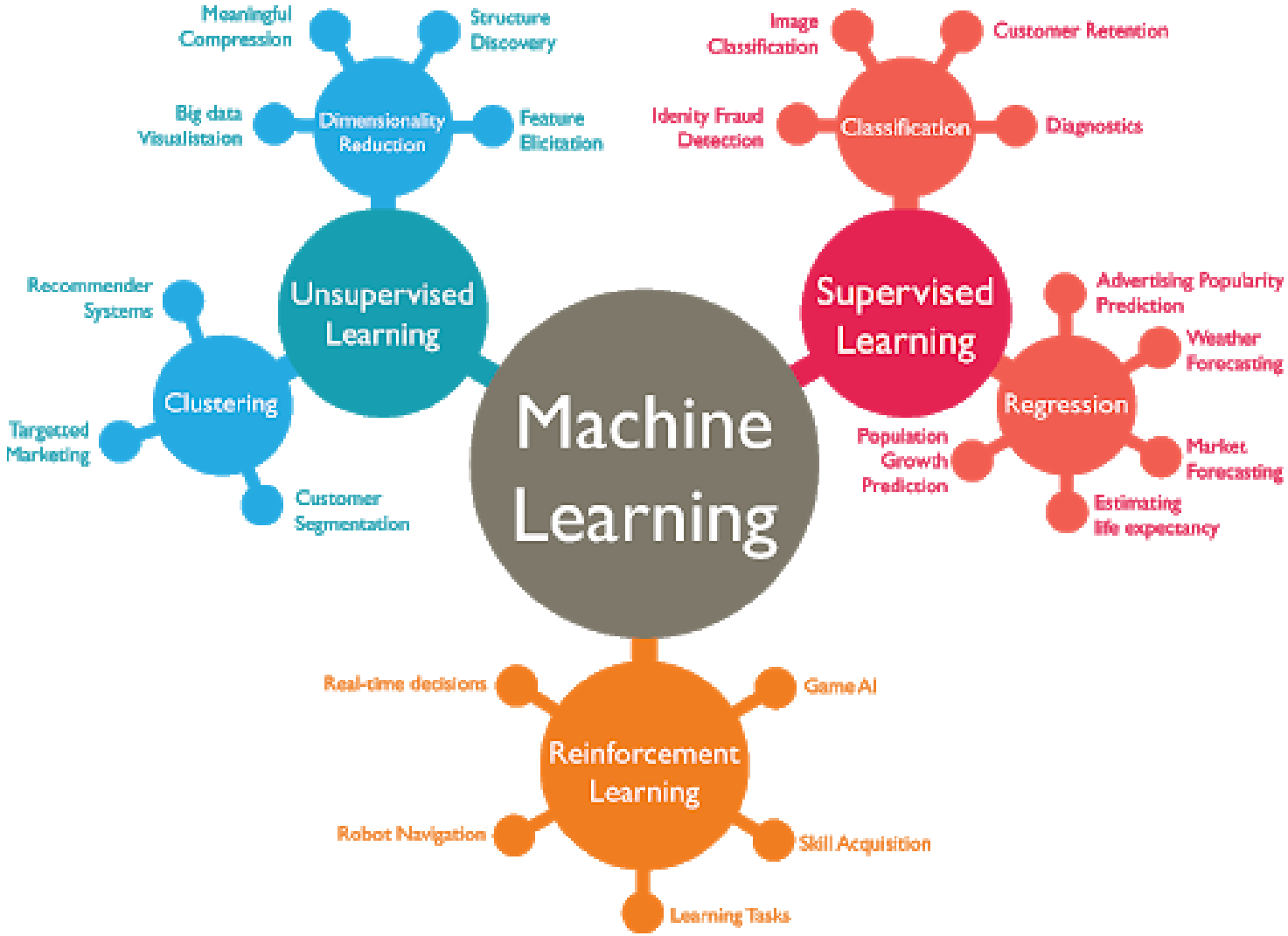
New Assignment

파뿌리
파이썬 뿌시는 이십대들!!

《 Round 10 》

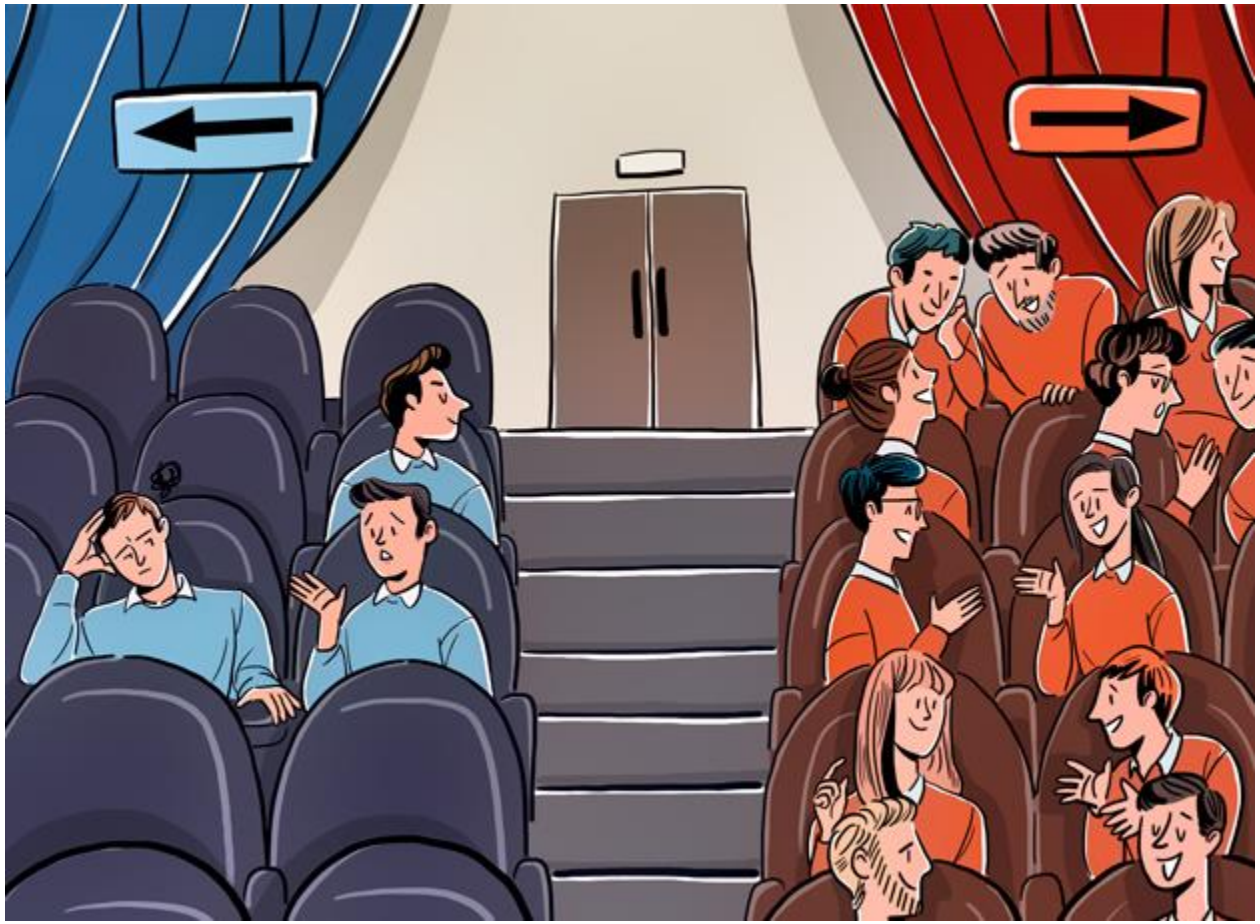
- 분류분석 개요 《
- 분류분석 실습

Machine Learning?



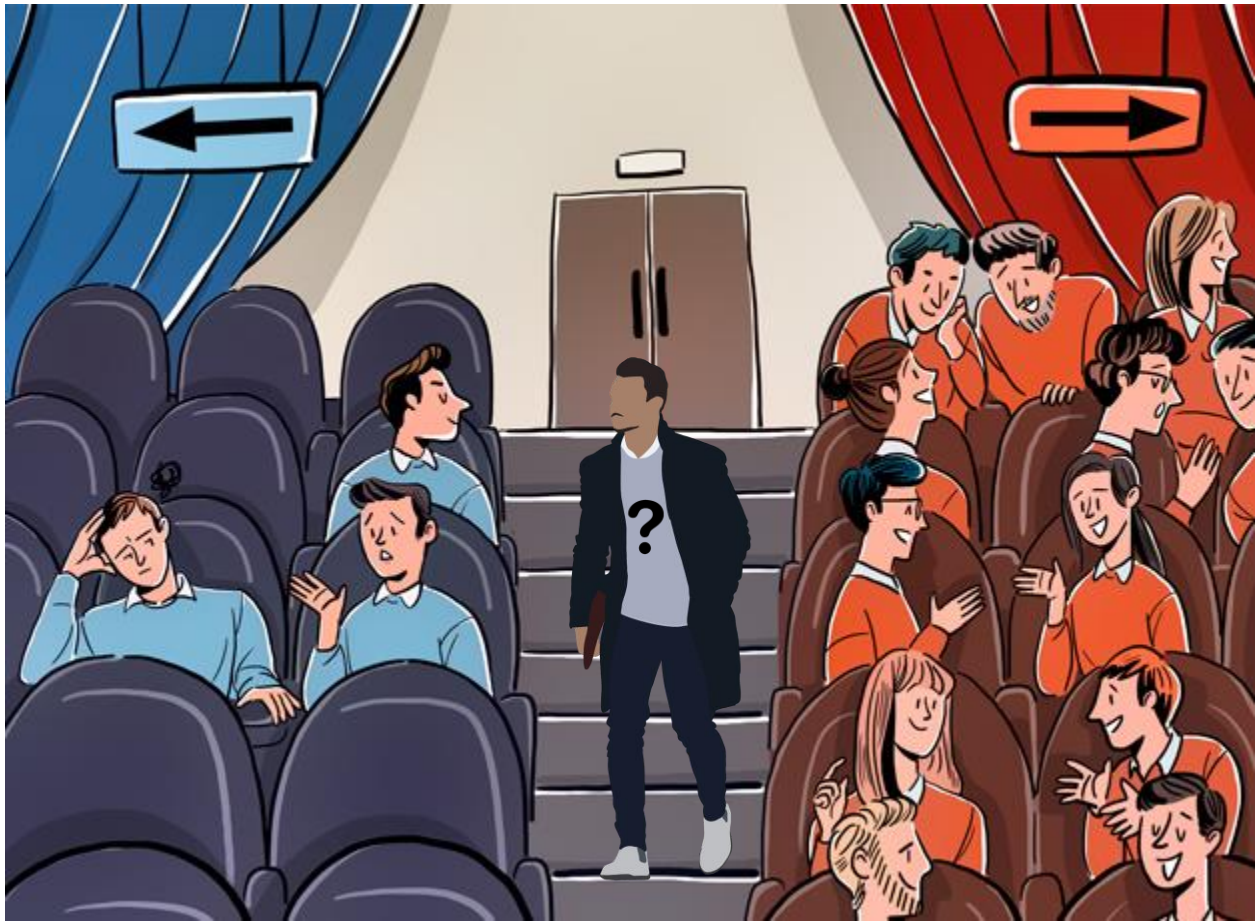
파뿌리
파이썬 뿌시는 이십대들!!

Classification?

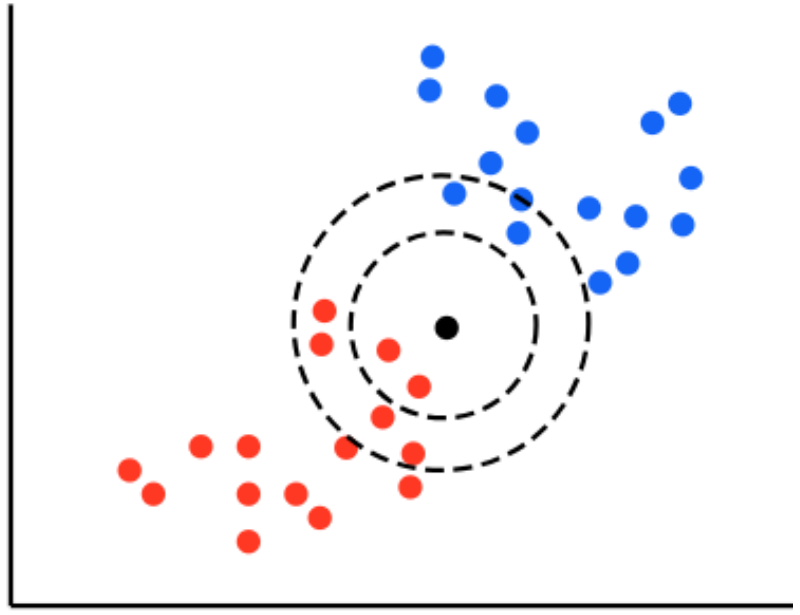


파뿌리
파이썬 뿌시는 이십대들!!

Classification?

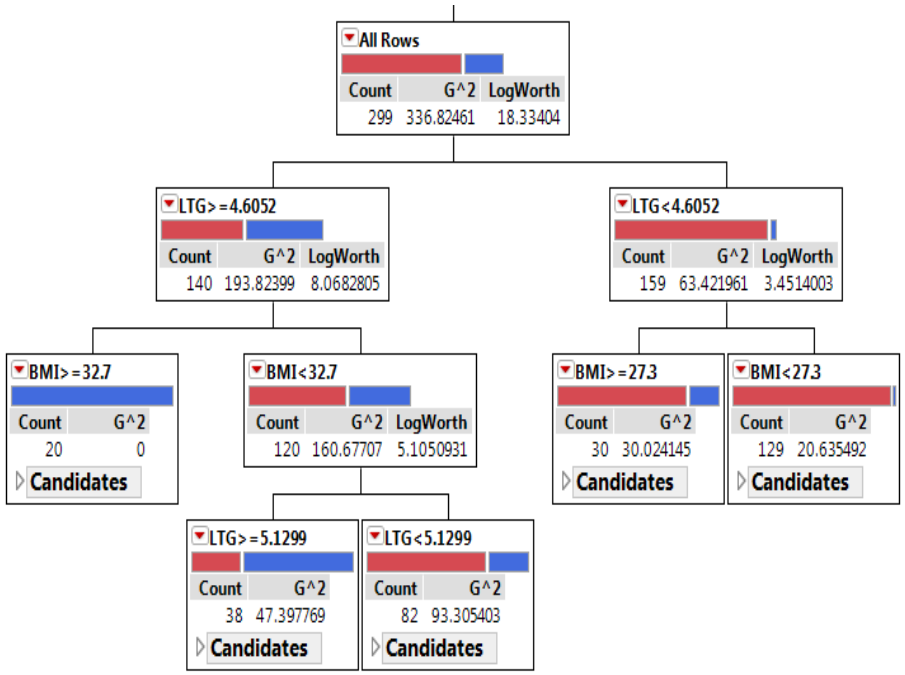


K Nearest Neighbours(KNN)



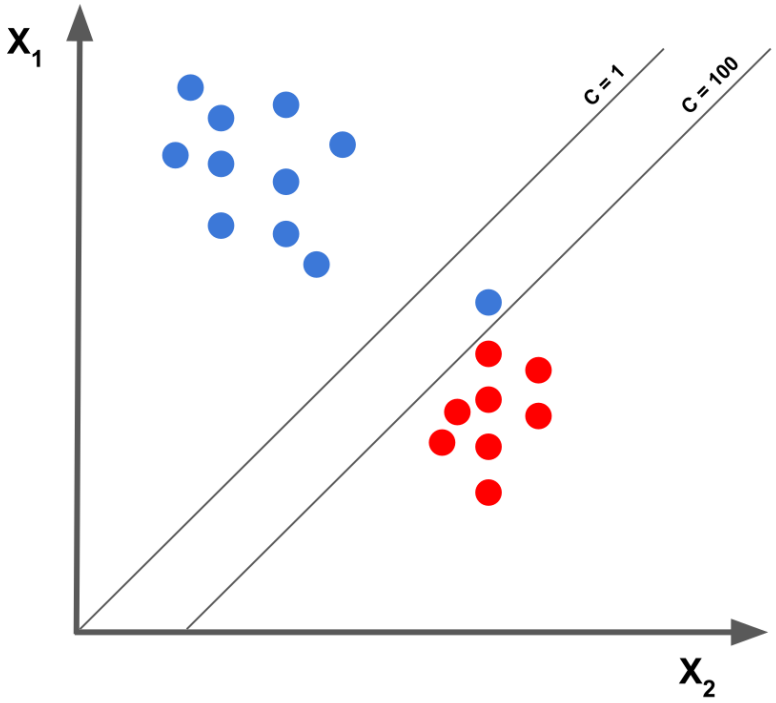
- 거리 기반의 분류 방법
- 어떠한 관측치와 가장 가까운 K개의 이웃으로 분류를 결정
- 과반수의 이웃을 따르기 때문에 K는 홀수 일 때 유리함.
- 이상치에 강하다는 특징.

Decision Tree



- 불순도를 기반으로 한 분류 방법
- 중요성이 높은 변수부터 불순도가 최소가 되는 지점을 분기
- DT 자체가 설명력을 확실히 가져간다는 것이 장점
- LGBM, XGBoost 등 DT를 기반으로 한 모델의 성능적인 강점도 많이 두드러지는 추세.

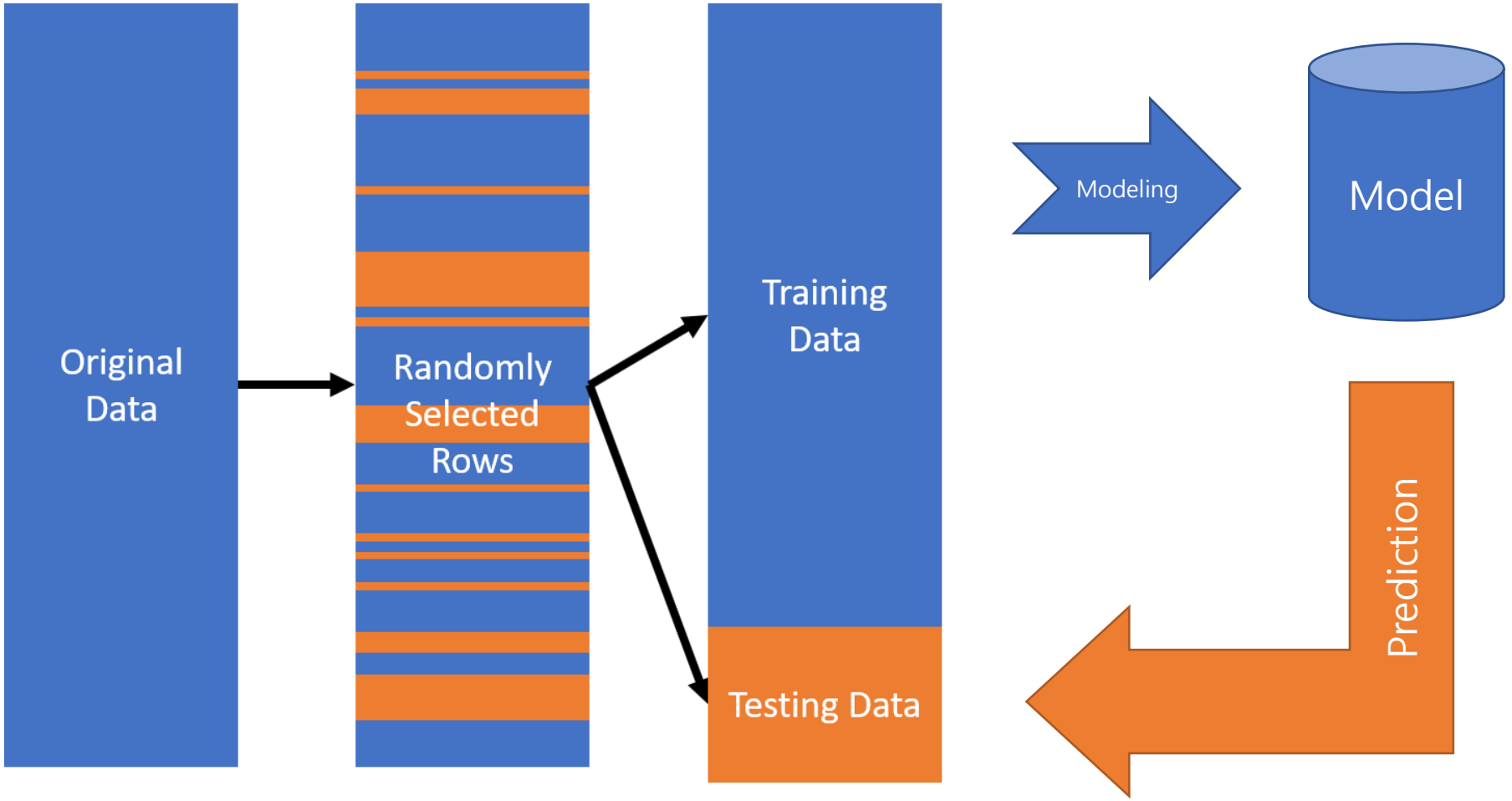
Support Vector Machine



- 두 집단 사이의 마진을 기준으로 한 분류법
- 서포트 벡터 간의 마진이 최대가 되는 선을 기준으로 분류
- 파라미터 C를 통해 오차허용을 유연하게 정의해줄 수 있음
- 대체적으로 빠르고 높은 성능

Learning library

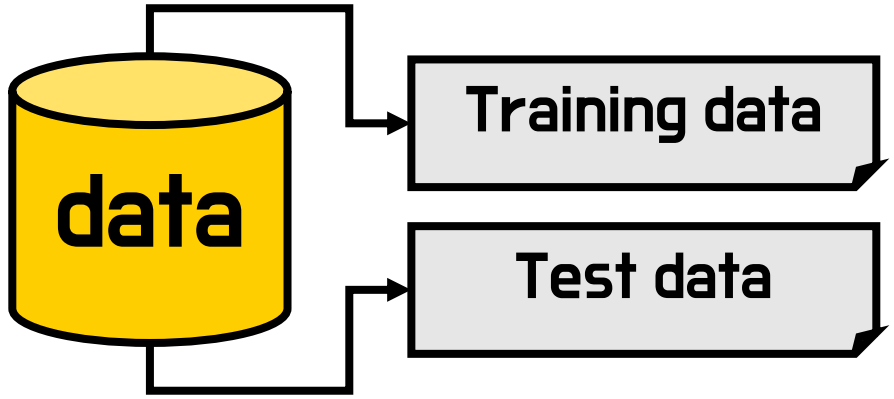
Data splitting



Data splitting

```
## dataset을 training data와 test data로 분할
# train_test_split(독립변수, 종속변수, test data 사이즈(%), 랜덤 추출 시드값)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=10)

print('train data 개수: ', len(X_train))
print('test data 개수: ', len(X_test),"\n")
```



파뿌리
파이썬 뿌시는 이십대들!!

《 Round 9 》

- 분류분석 개요
- 분류분석 실습 《

지표 해석 - Confusion Matrix

		Predicted		
		Yes	No	
Actual	Yes	2 (True +ve)	1 (False -ve)	2/(2+1)=⅔ **Recall (Sensitivity)**
	No	2 (False +ve)	3 (True -ve)	3/(3+2)=⅗ **(Specificity)**
		2/(2+2)=50% **(Precision)**		**Accuracy=** (2+3)/(2+1+2+3)= 5/8

source code :

https://github.com/koptimizer/Python_Breakers/blob/master/season2/code/claSample.ipynb

지표 해석 - Precision & Recall

		Predicted		
		Yes	No	
Actual	Yes	2 (True +ve)	1 (False -ve)	2/(2+1)=⅔ Recall (Sensitivity)
	No	2 (False +ve)	3 (True -ve)	3/(3+2)=⅗ (Specificity)
		2/(2+2)=50% (Precision)		Accuracy= (2+3)/(2+1+2+3)= 5/8

- Precision(정밀도) : Yes라고 예측한 사람 중에서 실제 Yes의 비율
- Recall(재현율, 민감도) :

지표 해석 - Accuracy & F1-score

		Predicted		
		Yes	No	
Actual	Yes	2 (True +ve)	1 (False -ve)	2/(2+1)=⅔ Recall (Sensitivity)
	No	2 (False +ve)	3 (True -ve)	3/(3+2)=⅗ (Specificity)
		2/(2+2)=50% (Precision)		Accuracy= (2+3)/(2+1+2+3)= 5/8

- Accuracy(정확도) : 전체 사람 중에서 정확하게 분류한 비율
- F1-score : Precision과 Recall의 조화평균 → 훨씬 정확

KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, Y_train)
Y_hat = knn.predict(X_test)

from sklearn import metrics
knn_confusion_matrix = metrics.confusion_matrix(Y_test, Y_hat)
print(knn_confusion_matrix)
```

```
[[13  0  0]
 [ 0 13  1]
 [ 0  4 14]]
```

```
knn_report = metrics.classification_report(Y_test, Y_hat)
print(knn_report)
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       0.76      0.93      0.84        14
   virginica       0.93      0.78      0.85        18

    accuracy                           0.89        45
   macro avg       0.90      0.90      0.90        45
weighted avg       0.90      0.89      0.89        45
```

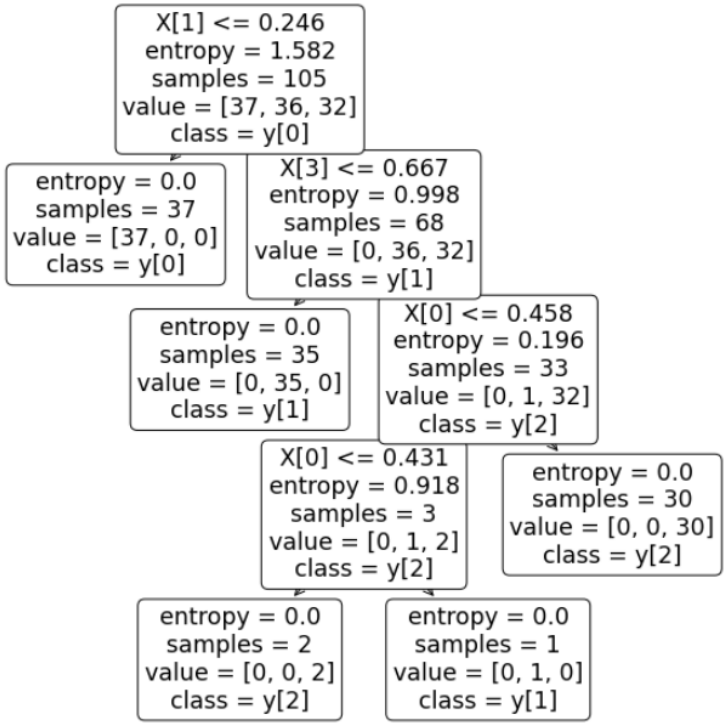
DT

```
from sklearn import tree
dt = tree.DecisionTreeClassifier(criterion='entropy', max_depth=5)
dt.fit(X_train, Y_train)
y_hat = dt.predict(X_test)

from sklearn import metrics
dt_confusion_matrix = metrics.confusion_matrix(Y_test, Y_hat)
print(dt_confusion_matrix)
```

```
[[13  0  0]
 [ 0 13  1]
 [ 0  4 14]]
```

```
fig, ax = plt.subplots(figsize=(10, 10))
tree.plot_tree(dt, impurity=True, class_names = True, rounded = True)
plt.show()
```



SVM

```
from sklearn import svm
svm = svm.SVC(kernel = 'linear')
svm.fit(X_train, Y_train)
y_hat = svm.predict(X_test)

from sklearn import metrics
svm_confusion_matrix = metrics.confusion_matrix(Y_test, Y_hat)
print(svm_confusion_matrix)
```

```
[[13  0  0]
 [ 0 13  1]
 [ 0  4 14]]
```

```
svm_report = metrics.classification_report(Y_test, Y_hat)
print(svm_report)
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       0.76      0.93      0.84        14
   virginica       0.93      0.78      0.85        18

    accuracy                           0.89        45
   macro avg       0.90      0.90      0.90        45
weighted avg       0.90      0.89      0.89        45
```

파뿌리
파이썬 뿌시는 이십대들!!

NEXT STAGE