수학으로 이해하는 기계학습

- 수학과 22학번 고홍규
- 과제 250406
- 머신러닝 모델실습

기본세팅

```
#구글 드라이브 연결
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd

#경고 메세지 무시
import warnings
warnings.filterwarnings('ignore')
```

Train 데이터 전처리

```
train = pd.read_csv("drive/My Drive/Colab Notebooks/2025MLwithmath/titanic/train.csv")
```

```
def preprocessing(df):
  '''
  - 결측치 처리(Age, Cabin, Embarked)
  - 범주형 데이터 처리(원핫인코딩:Sex, Embarked)
  - PassengerId, Name, Ticket 삭제
  '''
  #결측치 처리
  df['Age'] = df['Age'].fillna(df['Age'].mean())  #age 평균값
  df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])  #Embarked 최빈값
  df = df.drop('Cabin', axis=1) #Cabin 삭제

  #범주형 데이터 처리 (원핫 인코딩 후 원본 컬럼은 삭제함)
  df_encoding = pd.get_dummies(df[['Sex','Embarked']])
  df = pd.concat([df, df_encoding], axis=1)
  df = df.drop('Sex', axis=1) #Sex 삭제
  df = df.drop('Embarked', axis=1) #Embarked 삭제

  #승객번호, 이름, 티켓 삭제
  # df = df.drop('PassengerId', axis=1) #Name 삭제
```

```
    df = df.drop('Name', axis=1) #Name 삭제
    df = df.drop('Ticket', axis=1) #Ticket 삭제

    return df


# train 데이터 전처리
train = preprocessing(train)
train.head()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male
0	1	0	3	22.0	1	0	7.2500	False	True
1	2	1	1	38.0	1	0	71.2833	True	False
2	3	1	3	26.0	0	0	7.9250	True	False
3	4	1	1	35.0	1	0	53.1000	True	False
4	5	0	3	35.0	0	0	8.0500	False	True

다음 단계: train 변수로 코드 생성 추천 차트 보기 New interactive sheet

Test 데이터 전처리

```
# test데이터 불러옴
test = pd.read_csv("drive/My Drive/Colab Notebooks/2025MLwithmath/titanic/test.csv")


test.head(1)
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarl
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	

다음 단계: test 변수로 코드 생성 추천 차트 보기 New interactive sheet

```
train.shape, test.shape
```

```
((891, 12), (418, 11))
```

```
# Fare 결측치가 보임
test.isnull().sum()
```

	0
PassengerId	0
Pclass	0
Name	0
Sex	0
Age	86
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	327
Embarked	0

dtype: int64

```
# age와 cabin데이터 결측치 제거
test = preprocessing(test)
test.isnull().sum()
```

	0
PassengerId	0
Pclass	0
Age	0
SibSp	0
Parch	0
Fare	1
Sex_female	0
Sex_male	0
Embarked_C	0
Embarked_Q	0
Embarked_S	0

dtype: int64

```
# Fare에 결측치가 있는 행을 삭제 하면 절대 안됨
```

```
# Fare결측치를
# 중앙값으로 대체
test['Fare'] = test['Fare'].fillna(test['Fare'].median())
```

```
test.isnull().sum()
# 결측치 제거 끝
```

	0
PassengerId	0
Pclass	0
Age	0
SibSp	0
Parch	0
Fare	0
Sex_female	0
Sex_male	0
Embarked_C	0
Embarked_Q	0
Embarked_S	0

dtype: int64

머신러닝 기본 모델 세팅

```
from sklearn.ensemble import RandomForestClassifier
# 랜덤포레스트 모델 불러옴
```

```
# Survived의 Label 값인 target만듬
target = train['Survived']
target
```

	Survived
0	0
1	1
2	1
3	1
4	0
...	...
886	0
887	1
888	0
889	1
890	0

891 rows × 1 columns

dtype: int64

```
X = train.drop(['Survived', 'PassengerId'], axis=1) #사실상train데이터에서 'Survived'없는것
X_test = test.drop(['PassengerId'], axis=1)# 사실상 test데이터


# 랜덤포레스트 모델 쓰자
model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
# n_estimators=100 → 결정트리 100개를 사용하겠다
# max_depth=5 → 각 트리의 최대 깊이를 5로 제한 (과적합 방지)
# random_state=1 → 랜덤 결과 고정 (재현 가능성)
# model = RandomForestClassifier()
model.fit(X, target) #survived없는 train데이터를 feature로, surviviied데이터를 target으로
# 모델을 학습시킴
predictions = model.predict(X_test)
# 학습된 모델을 이용해 X_test를 Question.


predictions
# Answer값(=예측) 출력
```

```
array([0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
```

```
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0])
```

```
# 제출 파일 만들기
output = pd.DataFrame({'PassengerId': test.PassengerId, 'Survived': predictions})
output.to_csv("drive/My Drive/Colab Notebooks/2025MLwithmath/titanic/MYsubmission1.csv", index=Fal
print("Your submission was successfully saved!")
```

```
Your submission was successfully saved!
```

```
output# 제출 파일
```

	PassengerId	Survived
0	892	0
1	893	0
2	894	0
3	895	1
4	896	0
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

다음 단계: output 변수로 코드 생성 추천 차트 보기 New interactive sheet

```
# 이제 데이콘에 제출하자.
```

Splitting the Training Data

```
from sklearn.model_selection import train_test_split

X = train.drop(['Survived','PassengerId'], axis=1) #사실상train데이터에서 'Survived'없는것
target = train["Survived"] #train데이터에서 Survived만.
```

```
x_train, x_val, y_train, y_val = train_test_split(X, target, test_size = 0.2, random_state = 0)
# 전체 train데이터(X, target)를 80%:20% 비율로 나눠서
# test_size=0.2 전체 데이터 중 **20%를 검증용(val)**으로 사용
# random_state=0    매번 실행해도 같은 결과로 나누게끔 고정하는 숫자
#  (여기선 랜덤 고정=0 ->계속 같은결과로 나눠짐)
```

모델들

```
# Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

gaussian = GaussianNB()
gaussian.fit(x_train, y_train) #모델.fit = 학습(x_train, y_train 데이터 기반)

y_pred = gaussian.predict(x_val)
# 모델.predict = 학습된 모델을 이용해 x_val를 Question.
# Answer값을 y_pred에 저장


acc_gaussian = round(accuracy_score(y_pred, y_val) * 100, 2)
# Answer값(=예측)(=y_pred)과 target값(y_val)을 비교해서

print(acc_gaussian)# 모델 정확도 출력
```

```
79.89
```

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_pred = logreg.predict(x_val)
acc_logreg = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_logreg)
```

```
79.89
```

```
# Support Vector Machines
from sklearn.svm import SVC

svc = SVC()
svc.fit(x_train, y_train)
y_pred = svc.predict(x_val)
acc_svc = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_svc)
```

```
72.07
```

```
# Linear SVC
from sklearn.svm import LinearSVC

linear_svc = LinearSVC()
linear_svc.fit(x_train, y_train)
y_pred = linear_svc.predict(x_val)
acc_linear_svc = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_linear_svc)
```

```
79.89
```

```
# Perceptron
from sklearn.linear_model import Perceptron

perceptron = Perceptron()
perceptron.fit(x_train, y_train)
y_pred = perceptron.predict(x_val)
acc_perceptron = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_perceptron)
```

```
75.42
```

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier

decisiontree = DecisionTreeClassifier()
decisiontree.fit(x_train, y_train)
y_pred = decisiontree.predict(x_val)
acc_decisiontree = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_decisiontree)
```

```
77.09
```

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

randomforest = RandomForestClassifier()
randomforest.fit(x_train, y_train)
y_pred = randomforest.predict(x_val)
acc_randomforest = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_randomforest)
```

```
83.24
```

```
# KNN or k-Nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
y_pred = knn.predict(x_val)
```

```
acc_knn = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_knn)
```

```
72.63
```

```
# Stochastic Gradient Descent
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier()
sgd.fit(x_train, y_train)
y_pred = sgd.predict(x_val)
acc_sgd = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_sgd)
```

```
80.45
```

```
# Gradient Boosting Classifier
from sklearn.ensemble import GradientBoostingClassifier

gbk = GradientBoostingClassifier()
gbk.fit(x_train, y_train)
y_pred = gbk.predict(x_val)
acc_gbk = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_gbk)
```

```
83.8
```

```
# XGBoost Classifier
from xgboost import XGBClassifier

xgb = XGBClassifier()
xgb.fit(x_train, y_train)
y_pred = xgb.predict(x_val)
acc_xgb = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_xgb)
```

```
83.8
```

```
# LightGBM Classifier
from lightgbm import LGBMClassifier

lgbc = LGBMClassifier()
lgbc.fit(x_train, y_train)
y_pred = lgbc.predict(x_val)
acc_lgbc = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_lgbc)
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
84.92
```

```
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest', 'Naive Bayes', 'Perceptron', 'Linear SVC',
              'Decision Tree', 'Stochastic Gradient Descent', 'Gradient Boosting Classifier',
              'XGBClassifier', 'LGBMClassifier'],
```

```
    'Score': [acc_svc, acc_knn, acc_logreg,
              acc_randomforest, acc_gaussian, acc_perceptron,acc_linear_svc, acc_decisiontree,
              acc_sgd, acc_gbk, acc_xgb, acc_lgbc]})
models.sort_values(by='Score', ascending=False)

#그동안의 코드에서 모델별 accuracy를 각각 acc_svc, acc_knn, acc_logreg, .. 이런 변수에 저장해왔음
# 그것들을 걍 출력한 거임. 표로
```

	Model	Score
11	LGBMClassifier	84.92
9	Gradient Boosting Classifier	83.80
10	XGBClassifier	83.80
3	Random Forest	83.24
8	Stochastic Gradient Descent	80.45
6	Linear SVC	79.89
2	Logistic Regression	79.89
4	Naive Bayes	79.89
7	Decision Tree	77.09
5	Perceptron	75.42
1	KNN	72.63
0	Support Vector Machines	72.07

```
# Gradient Boosting Classifier로 학습시킨거 모델 제출하자.

# gbk = GradientBoostingClassifier()
# gbk.fit(x_train, y_train)
#위에서 이미 학습을 시킴. 그래서 이 코드들은  생략
predictions = gbk.predict(test.drop('PassengerId', axis=1))
# 왜 X_test가 아닌 test.drop('PassengerId', axis=1)으로 Q?
# 왜냐. test['PassengerId'] 데이터가 전처리에서 문제.그래서 일단 빼고 Q.하고
# 나중에 제출할때 추가하자.



output = pd.DataFrame({ 'PassengerId' : test['PassengerId'], 'Survived': predictions })
# 방금한 말임
output.to_csv("drive/My Drive/Colab Notebooks/2025MLwithmath/titanic/MYsubmission2.csv", index=Fal
```

Train 전체 학습

```
# 위랑 비슷한 방식인데, val 없고 바로 test데이터로 예측하자.
target = train['Survived']
X = train.drop(['Survived', 'PassengerId'], axis=1)
```

```
X = train.drop(['Survived', 'PassengerId'], axis=1)
X_test = test.drop(['PassengerId'], axis=1)


# LightGBM Classifier
lgbc = LGBMClassifier()
lgbc.fit(X, target)
y_pred = lgbc.predict(X_test)
```

```
[LightGBM] [Info] Number of positive: 342, number of negative: 549
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.00046
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 227
[LightGBM] [Info] Number of data points in the train set: 891, number of used features: 10
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.383838 -> initscore=-0.473288
[LightGBM] [Info] Start training from score -0.473288
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
output = pd.DataFrame({ 'PassengerId' : test['PassengerId'], 'Survived': y_pred })
# 방금한 말임
output.to_csv('drive/My Drive/Colab Notebooks/2025MLwithmath/titanic/fulldata_lgb_submissic
```