작성일자 : 2023/11/20

과제-1 : Gaussian Naive Bayes

학번 : 202210829
이름 : 김진석
학과 : 휴먼지능정보공학과

목차

(1) Fraud_data.csv 활용

1.1 접근

1.1.1 배경

가우시안 나이브 베이즈(Gaussian Naive Bayes) 분류 모델을 통해 사기 여부를 예측하는 과제이다. 데이터 셋의 각 특성이 사기 발생에 미치는 영향을 분석하고, 이를 통해 각 특성의 중요성과 사기 발생 가능성에 대한 확률을 구하고 분류하는 문제이다.

1.1.2 데이터 구성

제공된 데이터 셋은 "History(신용 이력)", "CoApplicant(공동 신청자)", "Accommodation(거주 여부)", 그리고 "Fraud(사기)"라는 네 가지 열을 포함하고 있다. 각 행은 특정한 사례를 나타내며 해당 사례가 사기 여부를 포함한 정보를 담고 있다. "History" 열은 사례의 신용 이력을, "CoApplicant" 열은 공동 신청자의 존재 여부를, "Accommodation" 열은 거주 여부를 나타내며, "Fraud" 열은 사기 발생 여부를 나타낸다.

1.1.3 구현 설계

방법1

먼저, 가우시안 나이브 베이즈(Gaussian Naive Bayes) 모델은 각 특성의 값이 정규 분포를 따른다고 가정하는 모델이다. 따라서 이 모델은 입력 데이터의 각 특성이 정규 분포를 따른다고 가정하고 작동한다. 그러기 때문에 교수님이 주신 코드의 replace()를 활용하여 Encoding을 해주고, 각 열에 해당하는 값을 변수에 저장하고, 각 변수들을 하나의 description feature로 만들어 X 변수에 저장하여 X_train 값을 만들어 주고, target feature인 “Fraud” 열은 Y 변수에 저장하여 Y_train 값을 만들어 준다. 마지막으로 모델을 불러와 학습을 시키고 예측을 진행한다.

방법2

먼저, 가우시안 나이브 베이즈(Gaussian Naive Bayes) 모델은 각 특성의 값이 정규 분포를 따른다고 가정하는 모델이다. 하지만 주어진 데이터가 정규 분포를 따른다고 장담할 수 없기 때문에 정규성 검정을 진행한다. 또한 데이터의 표본 즉 n이 30 이하이므로 샤피로 검정이나 비모수 검정 콜모고로프-스미르노프 검정을 활용하여 p-value 값에 따른 정규성 검정을 진행해야 한다. 그리고 각 열들이 결과에 어떤 영향을 미치는지 이해하기 위해 상관 계수를 계산함으로써 각 열들 간의 연관성을 확인하고 상관 계수에 따른 각 열에 가중치를 주어 더 정확한 예측을 위한 전처리를 한다. 그런 다음 앞서 전처리한 데이터에 맞는 모델을 불러와 학습과 예측을 진행한다.

1.2 구현 및 결과

1.2.1 방법1

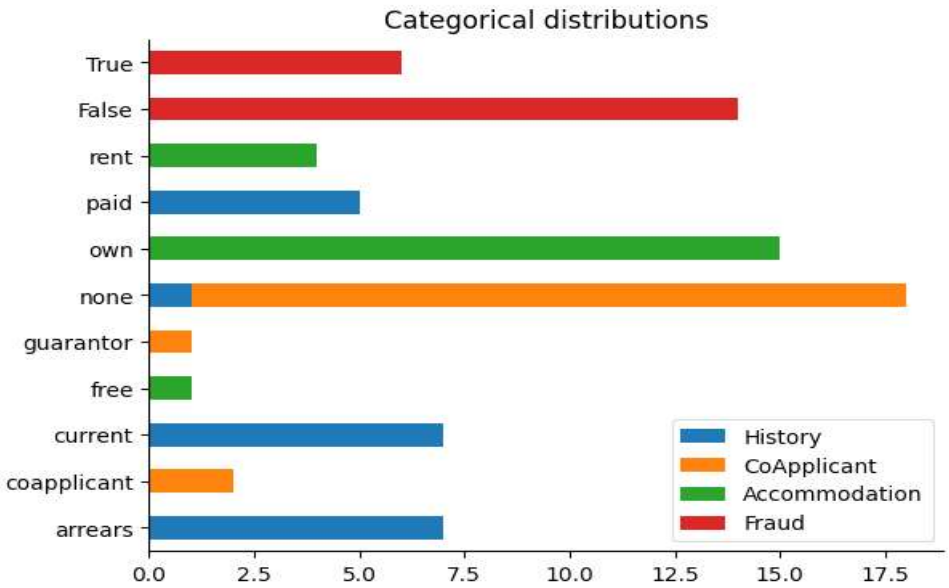


그림1 각 열의 데이터 분포 시각화

각 데이터의 인사이트를 얻고자 각 열의 분포를 나타내는 시각화를 해보았다.

	ID	History	CoApplicant	Accommodation	Fraud
0	1	1	0	1	1
1	2	2	0	1	0
2	3	2	0	1	0
3	4	2	1	0	1
4	5	0	0	1	0
5	6	0	0	1	1
6	7	1	0	1	0
7	8	0	0	1	0
8	9	1	0	0	0
9	10	0	0	1	1
10	11	1	1	1	0
11	12	1	0	1	1
12	13	1	0	0	1
13	14	2	0	1	0
14	15	0	0	1	0
15	16	1	0	1	0
16	17	0	1	0	0
17	18	0	0	-1	0
18	19	0	0	1	0
19	20	2	0	1	0

그림2 replace()를 적용한 df 결과

replace()에 True False 부분이 string type으로 적용되어 있어 bool type으로 바꿔주어 “Fraud” 열에도 Encoding 된 값이 적용되게 하였다.

```
history = res['History']
coapplicant = res['CoApplicant']
accommodation = res['Accommodation']
```

그림3 각 열에 해당하는 변수에 replace()를 적용한 값을 저장하는 부분

보통 이렇게 변수로 각 열을 저장하면 리스트나 concat 함수를 활용하여 다시 합치고 전치행렬 T를 적용해 다시 저장하지만, 나는 iloc을 활용하였다.

```
X = res.iloc[:, 1:4].values

X
```

```
array([[ 1,  0,  1],
       [ 2,  0,  1],
       [ 2,  0,  1],
       [ 2,  1,  0],
       [ 0,  0,  1],
       [ 0,  0,  1],
       [ 1,  0,  1],
       [ 0,  0,  1],
       [ 1,  0,  0],
       [ 0,  0,  1],
       [ 1,  1,  1],
       [ 1,  0,  1],
       [ 1,  0,  0],
       [ 2,  0,  1],
       [ 0,  0,  1],
       [ 1,  0,  1],
       [ 0,  1,  0],
       [ 0,  0, -1],
       [ 0,  0,  1],
       [ 2,  0,  1]])
```

그림4 iloc을 이용해 X 값을 분리하는 부분

```
[20]  Y = res['Fraud']
      Y
```

```
0     1
1     0
2     0
3     1
4     0
5     1
6     0
7     0
8     0
9     1
10    0
11    1
12    1
13    0
14    0
15    0
16    0
17    0
18    0
19    0
Name: Fraud, dtype: int64
```

그림5 Y값 저장하는 부분

```
[56]  test_case = pd.read_csv('/content/drive/MyDrive/case.csv')
```

```
[57]  test_case
```

	ID	History	CoApplicant	Accommodation
0	1	paid	none	rent
1	2	paid	guarantor	rent
2	3	arrears	guarantor	rent
3	4	arrears	guarantor	own
4	5	arrears	coapplicant	own

test_case를 csv 파일로 작성하여 불러와 위와 같은 전처리를 하여 X_test를 만든다.

```
test_X = test_case_res.iloc[:, 1:4].values

test_X
```

```
array([[2, 0, 0],
       [2, 1, 0],
       [0, 1, 0],
       [0, 1, 1],
       [0, 1, 1]])
```

```
model = GaussianNB()
model.fit(X, Y)
predicted = model.predict(test_X)
pred_prob = model.predict_proba(test_X)
```

X_test 값 추출하는 부분

각각의 case에 대한 결과

	ID	History	CoApplicant	Accommodation	Fraud	pred_prob_False	pred_prob_Ture
0	1	paid	none	rent	0	0.781854	0.218146
1	2	paid	guarantor	rent	0	0.681318	0.318682
2	3	arrears	guarantor	rent	0	0.614370	0.385630
3	4	arrears	guarantor	own	0	0.582578	0.417422
4	5	arrears	coapplicant	own	0	0.582578	0.417422

1.2.2 방법2

데이터가 과연 정규성을 만족하는지 각각의 정규 분포와 비모수 검정을 진행한 후 상관 계수를 계산하여 상관 계수 값에 따른 가중치를 주어 학습과 예측을 해보겠다.

먼저 데이터의 표본이 30이하이므로 비모수 검정을 진행해야 하기 때문에 주어진 데이터를 숫자형으로 바꿔주기 위해 one-hot encoding을 해준다.

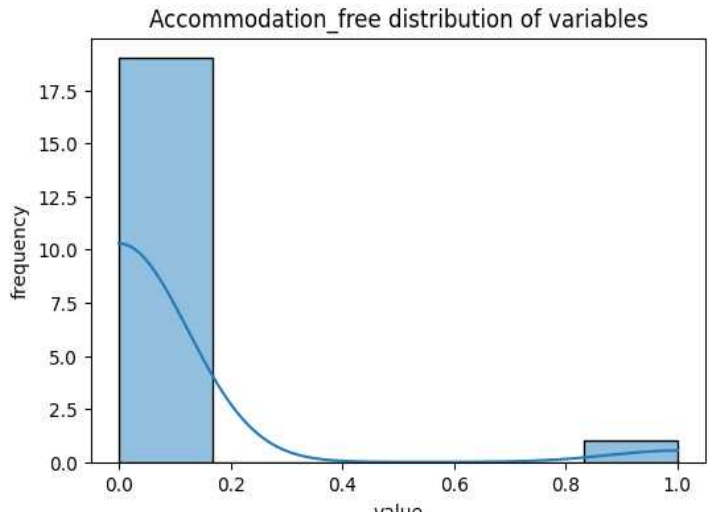
판다스의 dummies 함수를 이용하여 진행하였다.

	ID	Fraud	History_arrears	History_current	History_none	History_paid	CoApplicant_coapplicant	CoApplicant_guarantor	CoApplicant_none	Accommodation_free	Accommodation_own	Accommodation_rent
0	1	True	0	1	0	0	0	0	1	0	1	0
1	2	False	0	0	0	1	0	0	1	0	1	0
2	3	False	0	0	0	1	0	0	1	0	1	0
3	4	True	0	0	0	1	0	1	0	0	0	1
4	5	False	1	0	0	0	0	0	1	0	1	0
5	6	True	1	0	0	0	0	0	1	0	1	0
6	7	False	0	1	0	0	0	0	1	0	1	0
7	8	False	1	0	0	0	0	0	1	0	1	0
8	9	False	0	1	0	0	0	0	1	0	0	1
9	10	True	0	0	1	0	0	0	1	0	1	0
10	11	False	0	1	0	0	1	0	0	0	1	0
11	12	True	0	1	0	0	0	0	1	0	1	0
12	13	True	0	1	0	0	0	0	1	0	0	1
13	14	False	0	0	0	1	0	0	1	0	1	0
14	15	False	1	0	0	0	0	0	1	0	1	0
15	16	False	0	1	0	0	0	0	1	0	1	0
16	17	False	1	0	0	0	1	0	0	0	0	1
17	18	False	1	0	0	0	0	0	1	1	0	0
18	19	False	1	0	0	0	0	0	1	0	1	0
19	20	False	0	0	0	1	0	0	1	0	1	0

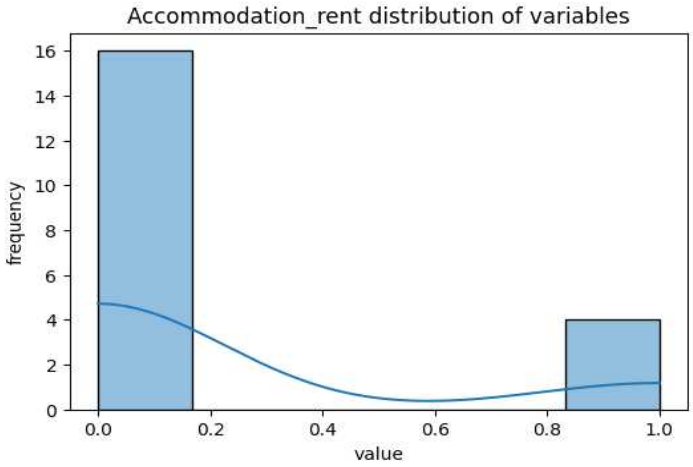
원-핫 인코딩을 진행한 결과

```
Fraud 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
History_arrears 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
History_current 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
History_none 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
History_paid 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
CoApplicant_coapplicant 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
CoApplicant_guarantor 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
CoApplicant_none 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
Accommodation_free 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
Accommodation_own 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
Accommodation_rent 변수의 콜모고로프-스미르노프 검정 결과:
데이터가 정규 분포를 따르지 않습니다.
```

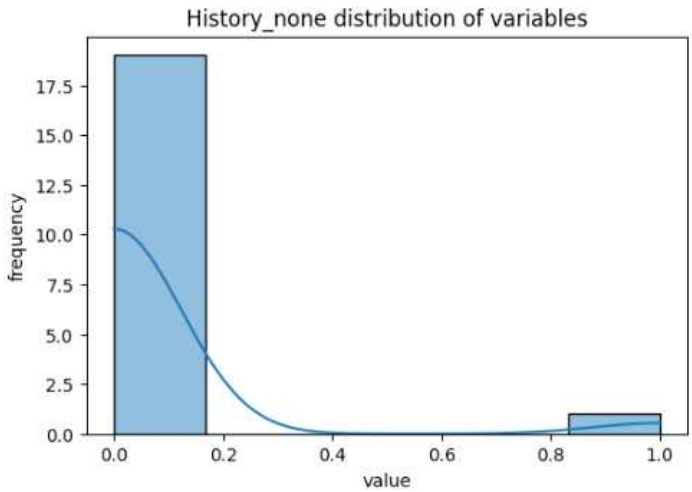
p-value 값을 0.05로 두었을 때 검정 결과이다. 각각의 열의 정규분포를 시각화 해 보자.



정규 분포 그림 1



정규 분포 그림 2



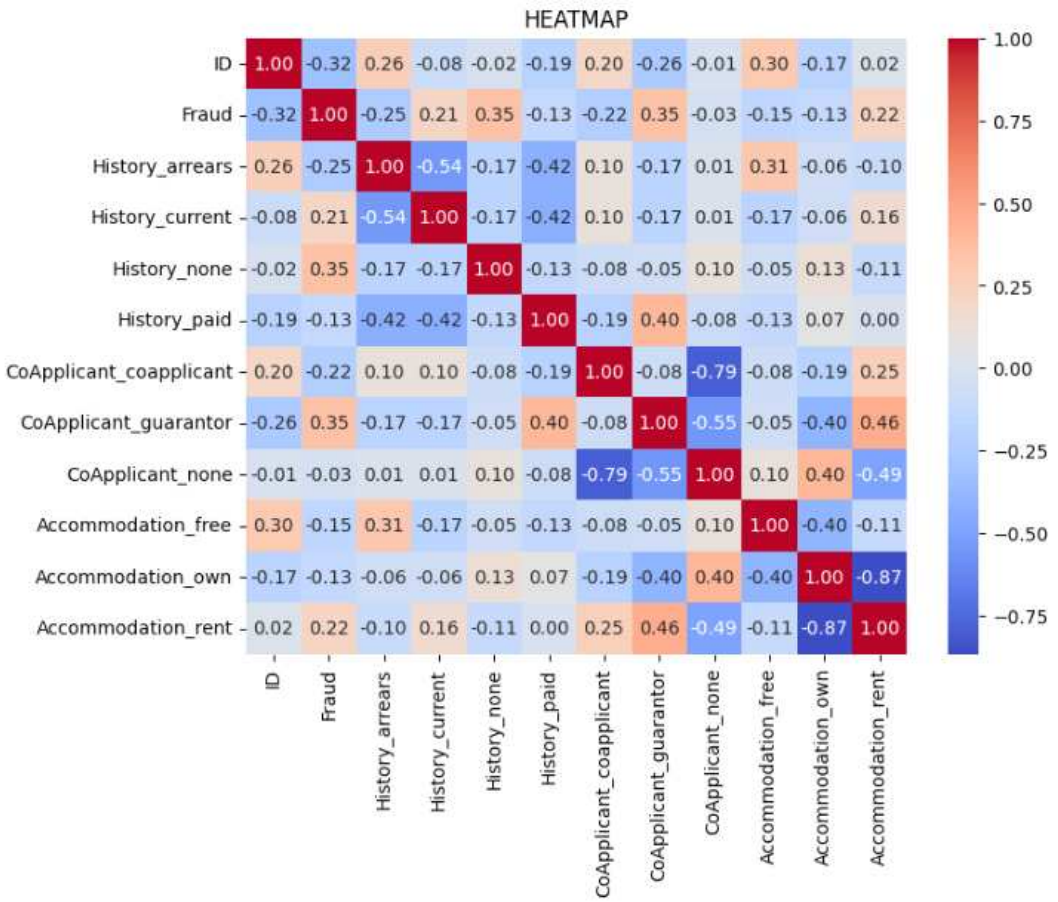
정규 분포 그림 3

일부 변수의 그림이지만 정규 분포를 따르지 않는 것으로 보인다.

그럼 변수 간의 영향도와 가중치 할당을 위해 상관 계수를 계산해 보겠다.

	ID	Fraud	History_arrears	History_current	History_none	History_paid	CoApplicant_coapplicant	CoApplicant_guarantor	CoApplicant_none	Accommodation_free	Accommodation_own	Accommodation_rent
ID	1.000000	-0.321672	0.263604	-0.081808	-0.019893	-0.190238	0.202326	-0.258607	-0.012142	0.298393	-0.170213	0.021678
Fraud	-0.321672	1.000000	-0.251630	0.205879	0.350438	-0.125988	-0.218218	0.350438	-0.030557	-0.150188	-0.125988	0.218218
History_arrears	0.263604	-0.251630	1.000000	-0.538462	-0.168345	-0.423659	0.104828	-0.168345	0.014679	0.312641	-0.060523	-0.104828
History_current	-0.081808	0.205879	-0.538462	1.000000	-0.168345	-0.423659	0.104828	-0.168345	0.014679	-0.168345	-0.060523	0.157243
History_none	-0.019893	0.350438	-0.168345	-0.168345	1.000000	-0.132453	-0.076472	-0.052632	0.096374	-0.052632	0.132453	-0.114708
History_paid	-0.190238	-0.125988	-0.423659	-0.423659	-0.132453	1.000000	-0.192450	0.397360	-0.080845	-0.132453	0.066667	0.000000
CoApplicant_coapplicant	0.202326	-0.218218	0.104828	0.104828	-0.076472	-0.192450	1.000000	-0.076472	-0.793492	-0.076472	-0.192450	0.250000
CoApplicant_guarantor	-0.258607	0.350438	-0.168345	-0.168345	-0.052632	0.397360	-0.076472	1.000000	-0.546119	-0.052632	-0.397360	0.458831
CoApplicant_none	-0.012142	-0.030557	0.014679	0.014679	0.096374	-0.080845	-0.793492	-0.546119	1.000000	0.096374	0.404226	-0.490098
Accommodation_free	0.298393	-0.150188	0.312641	-0.168345	-0.052632	-0.132453	-0.076472	-0.052632	0.096374	1.000000	-0.397360	-0.114708
Accommodation_own	-0.170213	-0.125988	-0.060523	-0.060523	0.132453	0.066667	-0.192450	-0.397360	0.404226	-0.397360	1.000000	-0.866025
Accommodation_rent	0.021678	0.218218	-0.104828	0.157243	-0.114708	0.000000	0.250000	0.458831	-0.490098	-0.114708	-0.866025	1.000000

corr 함수를 활용하여 상관 계수 계산



HEATMAP을 통한 시각화

결과를 보면 Fraud 값에 가장 영향을 주는 것은 0.35의 양의 상관 관계를 가진 History_none과 CoApplicant_guarantor 변수이다.

상관 계수에 따라 각 열에 가중치를 주고 학습을 진행하였다.
그리고 test case 넣어 보니 결과가 False, False, False, True, True가 나왔다.

```
] X_predicted

array([False, False, False,  True,  True])
```

```
] print(pred_prob)

[[1.00000000e+00 0.00000000e+00]
 [9.99979716e-01 2.02843230e-05]
 [9.99999138e-01 8.62029691e-07]
 [0.00000000e+00 1.00000000e+00]
 [0.00000000e+00 1.00000000e+00]]
```

결과 값이 방법1과 다르게 나왔다. 왜 그런지는 모르겠지만, 데이터가 너무 적어서 생긴 문제일 수 있고, 진행과정 중 실수가 있었을 수 있다고 생각한다. 이런 과정을 겪고 나니 교수님이 올리신 replace() 함수가 적절하게 가중치를 잘 적용한 encoding 이라고 생각한다.

(2) Social_Network_Ads.csv 활용

2.1 접근

2.1.1 배경

이 데이터셋은 다양한 특성을 가진 유저 정보를 포함하고 있으며, 이를 활용하여 특정 상품이나 서비스를 구매하는 데 영향을 미치는 요소들을 분석하고 예측하는 데 사용될 수 있습니다. 성별, 나이, 소득과 같은 특성들이 구매 여부와 관련이 있는지에 대한 분석을 통해 마케팅 전략이나 제품 개발 등에 유용한 인사이트를 도출할 수 있을 것으로 예상됩니다.

2.1.2 데이터 구성

User ID: 각 유저를 식별하는 고유한 ID 값으로, 유저 개인을 식별하기 위해 사용됩니다.

Gender: 유저의 성별을 나타냅니다. 여성과 남성의 성별 정보가 있으며, 이는 유저 특성 분석에 활용될 수 있습니다.

Age: 유저의 나이 정보입니다. 나이는 구매 패턴이나 선호도와 관련하여 중요한 요소로 분석될 수 있습니다.

Estimated Salary: 유저가 추정하는 연봉이나 소득 정보입니다. 이는 유저의 구매력을 추정하는 데 사용될 수 있습니다.

Purchased: 유저가 특정 제품 또는 서비스를 구매했는지 여부를 나타내는 값으로, 1은 구매를 의미하고 0은 미구매를 의미합니다.

2.1.3 구현 설계

데이터를 description feature로 만들어 X 변수에 저장하여 값을 만들어 주고, target feature인 “Purchased” 열은 Y 변수에 저장하여 값을 만들어 준다. 그리고 8 : 2로 X_train, Y_train과 X_test, Y_test 비율을 train_test_split 함수로 나눈다. 그리고 다양한 전처리(Label Encoding, Feature Scaling)을 진행하여 데이터 학습이 잘 될 수 있도록 전처리를 한다. 그리고 GaussianNB 모델을 사용하여 학습하고 예측을 진행한다.

다음은 train_test_split 대신 k-fold를 활용하여 진행해 보겠다. 그리고 각각 AUC 값과 정밀도, 재현율, F1-score 값, ROC 커브를 시각화를 진행하겠다.

2.2 구현 및 결과

2.2.1 Hold-Out

```
[133] # Importing the libraries
      import numpy as np
      import matplotlib.pyplot as plt
      import pandas as pd

      from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
      from sklearn.naive_bayes import GaussianNB
      from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

      from sklearn.metrics import roc_curve
      from sklearn.metrics import roc_auc_score
```

필요한 모듈 불러오기

```
# Importing the dataset
dataset = pd.read_csv('/content/drive/MyDrive/Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

hold out 방법인 train_test_split 방법을 사용하여 train, test 데이터 분리

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
```

```
# Feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
# Training the Naive Bayes model on the Training set
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

label Encoding과 Standard Scaler를 통해 값을 정규화한뒤 GaussianNB 모델을 통해 학습을 진행한다.

```
[139] # Predicting the Test set results
      y_pred = classifier.predict(X_test)
      ac = accuracy_score(y_test,y_pred)

      ac
```

```
0.9125
```

```
[145] # Making the Confusion Matrix
      cm = confusion_matrix(y_test, y_pred)
      lr_probs = classifier.predict_proba(X_test)
      lr_probs
```

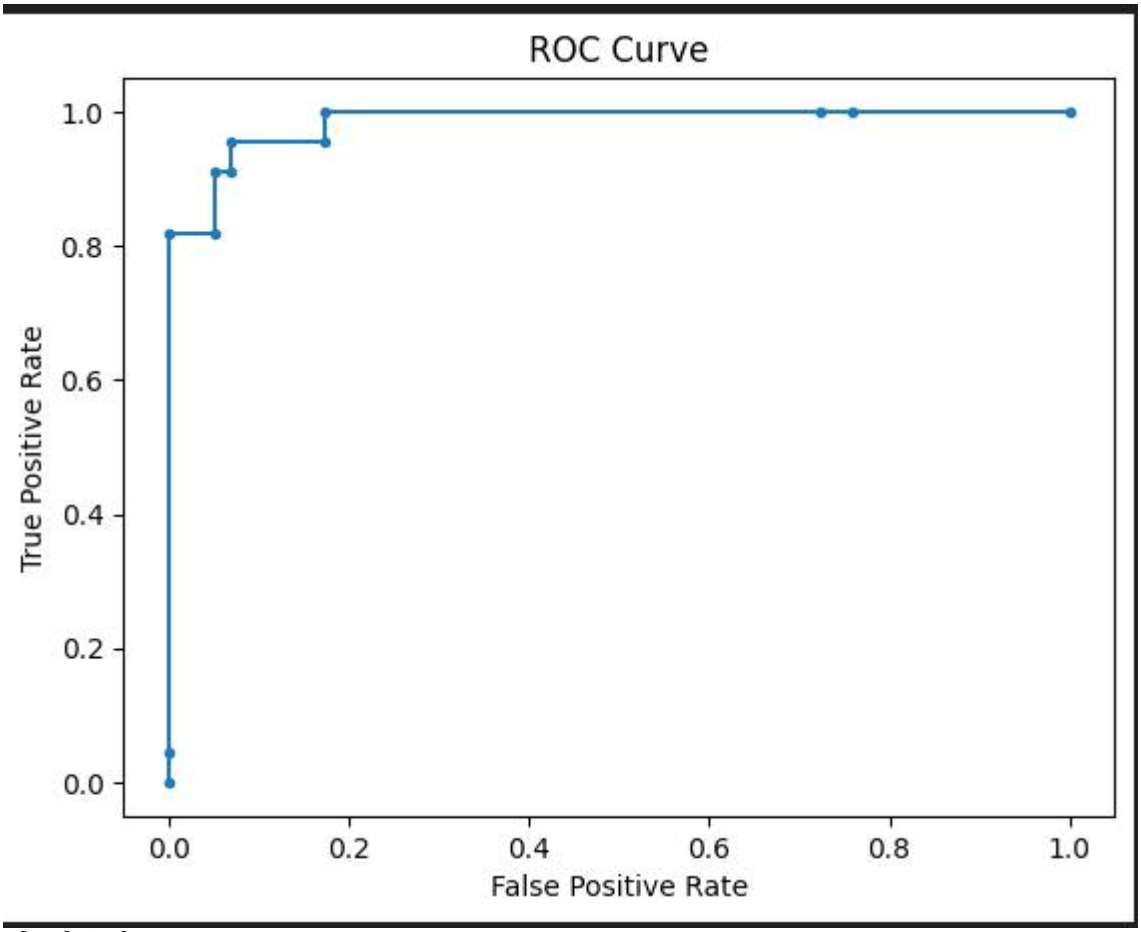
```
[0.75520429, 0.24479571],
[0.13099759, 0.86900241],
[0.97992931, 0.02007069],
[0.00455902, 0.99544098],
[0.92054458, 0.07945542],
[0.92955737, 0.07044263],
[0.84925986, 0.15074014],
[0.864263  , 0.135737  ],
[0.96926322, 0.03073678],
[0.78265882, 0.21734118],
[0.10438626, 0.89561374],
[0.86207101, 0.13792899],
[0.97996064, 0.02003936],
[0.99180918, 0.00819082],
[0.97349249, 0.02650751],
[0.91339539, 0.08660461],
[0.94062349, 0.05937651],
[0.38449172, 0.61550828],
[0.92822786, 0.07177214],
```

테스트 결과와 예측 정확도, 분류 확률을 출력해보았다.

```
Precision: 0.8571428571428571
Recall (Sensitivity): 0.8181818181818182
Specificity: 0.9482758620689655
AUC 스코어: 0.9843260188087775
```

Hold-out 방법의 measure들을 확인하면 AUC 스코어가 높게 나온걸 볼 수 있다.

ROC Curve를 그려보면



잘 나온다.

2.2.2 K-Fold

먼저 sklearn.model selection을 이용하여 KFold를 불러와 사용하였고, 총 2가지 인공지능 모델(GaussianNB, RandomForestClassifier)을 사용하였다.

```
kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# 교차 검증 수행
results = cross_val_score(model, X, y, cv=kfold)

# 결과 출력
print("평균 정확도:", results.mean())
print("분산 정확도:", results.var())
```

```
평균 정확도: 0.8875
분산 정확도: 0.0019062500000000017
```

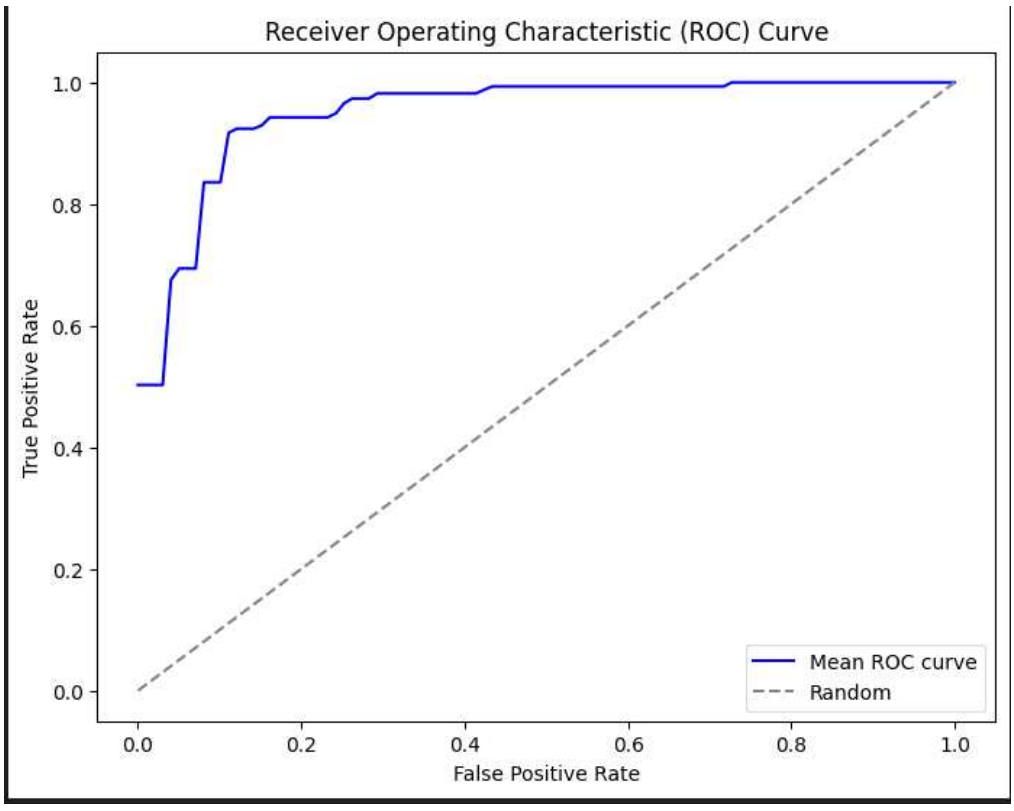
cross_val_score함수를 통해 편리하게 평균과 분산을 구했다.

이 코드에서 model 부분만 RandomForestClassifier 바꿔서 진행하니 결과가 비슷하게 나왔다.

```
model = RandomForestClassifier()

kfold = KFold(n_splits=10, shuffle=True, random_state=42)

# 교차 검증 수행
results = cross_val_score(model, X, y, cv=kfold)

# 결과 출력
print("예측 정확도 평균:", results.mean())
print("예측 정확도 분산:", results.var())
```

```
예측 정확도 평균: 0.89
예측 정확도 분산: 0.001275
```

다음은 K-Fold로 진행했을 때의 ROC curve 이다.



Hold-Out과 K-Fold의 결과론적으로 큰 차이는 없었지만 다시 한번 실행했을 때 결과 값이 바뀌는 정도 즉 일반화 부분이 가장 크게 차이가 난거 같았다. K-Fold가 더 일반화가 잘 되어 평균적으로 좋은 성능을 내는거 같다.

참고 문헌

colab : https://colab.research.google.com/?hl=ko
python으로 시작하는 빅데이터분석 및 인공지능 - 조준모
chatGPT.ai - 시각화 코드 생성 부분 활용