신용카드 이상거래 탐지 머신러닝모델 제작

조하진

CONTENTS

01

데이터 선정

- **데이터 선정과 문제정의**

Credit card fraud detection 데이터

Target = “Class”

이상거래 = 1 정상거래 = 0

분류 문제

```
df.columns
```

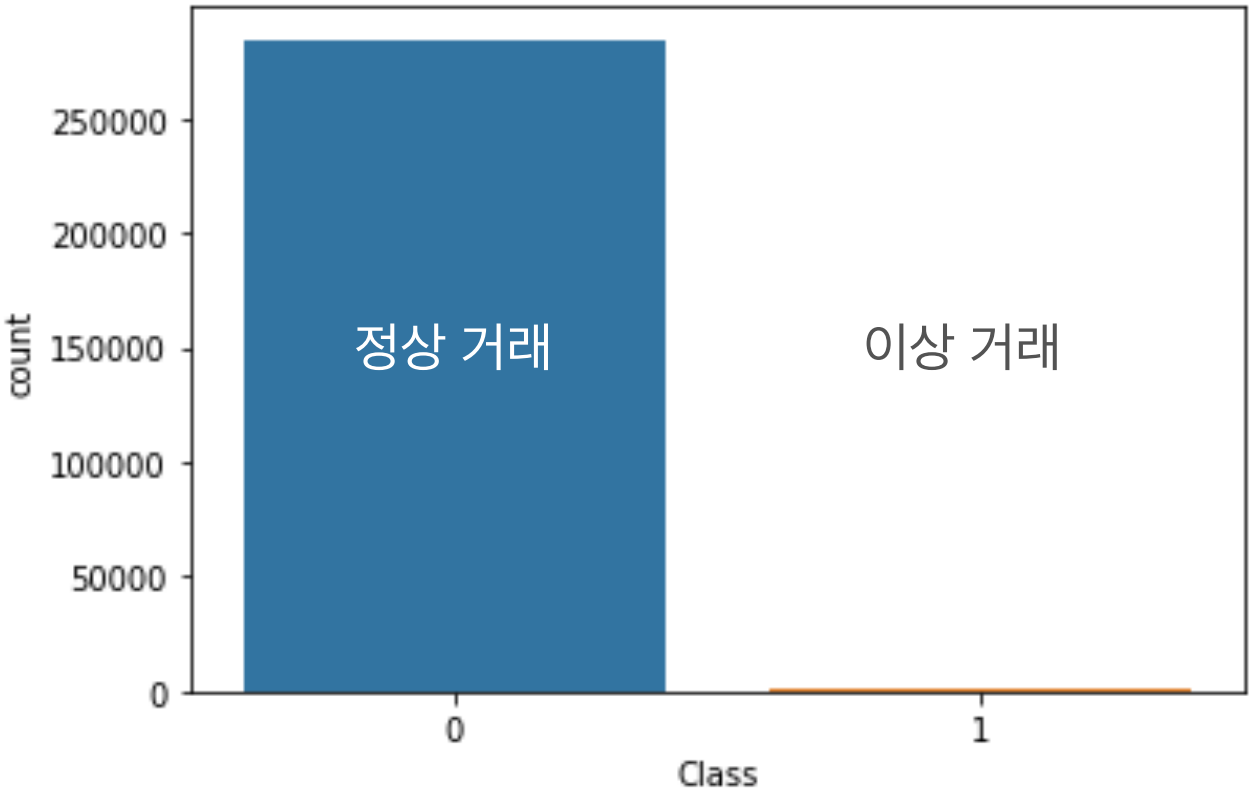
```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
```

```
df.shape
```

```
(284807, 31)
```

▪ 데이터 선정과 문제정의

타겟 분포



불균형 데이터

02

목표 및 평가지표

▪ 목표 및 평가지표

목표 - 사기탐지

Confusion Matrix		예측값	
		정상	사기
실제값	정상	True Negative(정상)	False Positive(오탐지)
	사기	False Negative (미탐지)	True Positive(정탐지)

재현율

$$\frac{\text{True Positive(정탐지)}}{\text{False Negative (미탐지)} + \text{True Positive(정탐지)}}$$

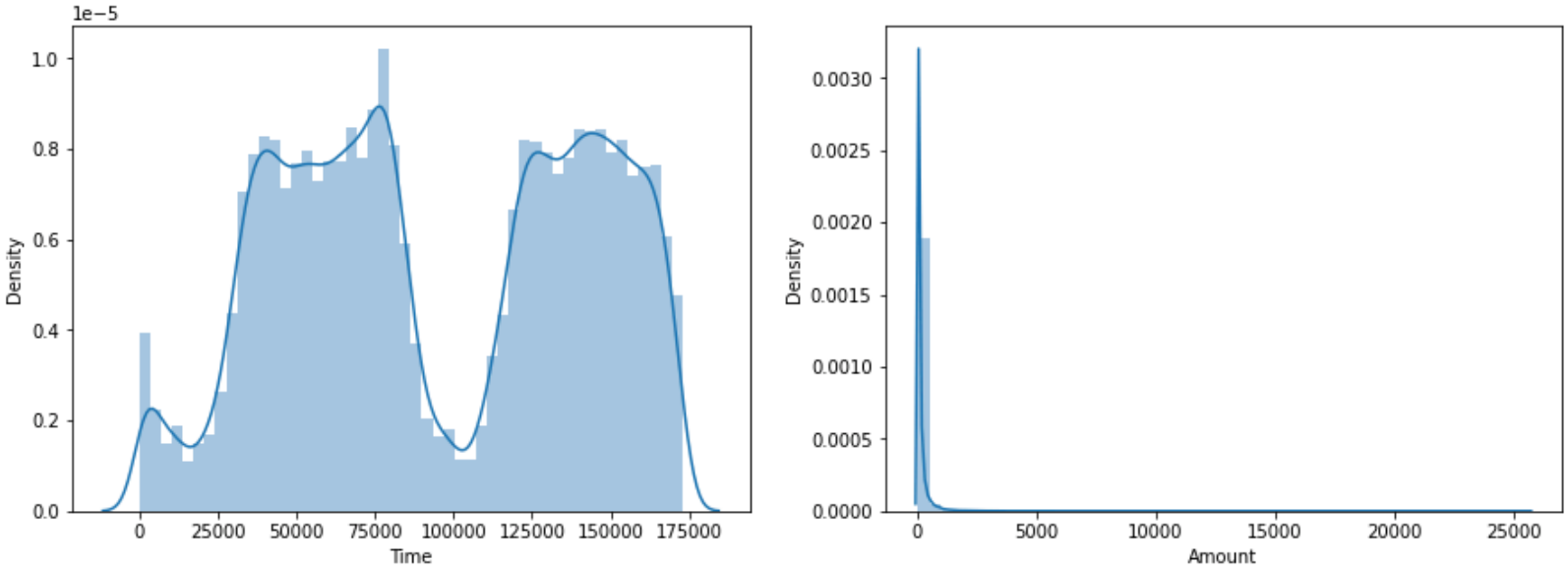
03

EDA와 데이터 전처리

EDA와 데이터 전처리

결측치 없음 전반적으로 정제된 데이터

거래시간과 거래금액 분포도

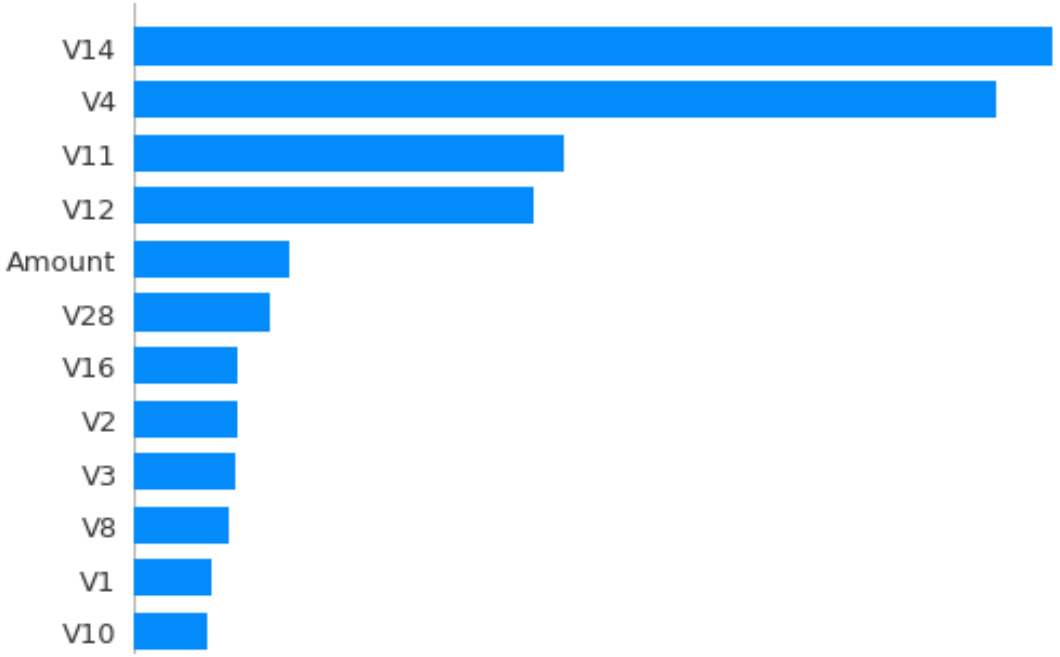


스케일링 필요

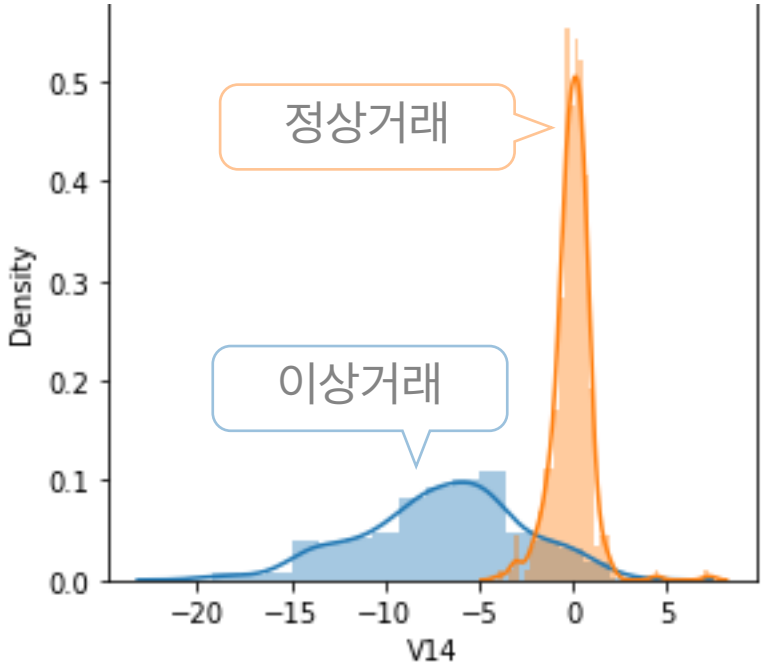
▪ EDA와 데이터 전처리

특성 중요도

타겟값에 영향을 끼치는 정도

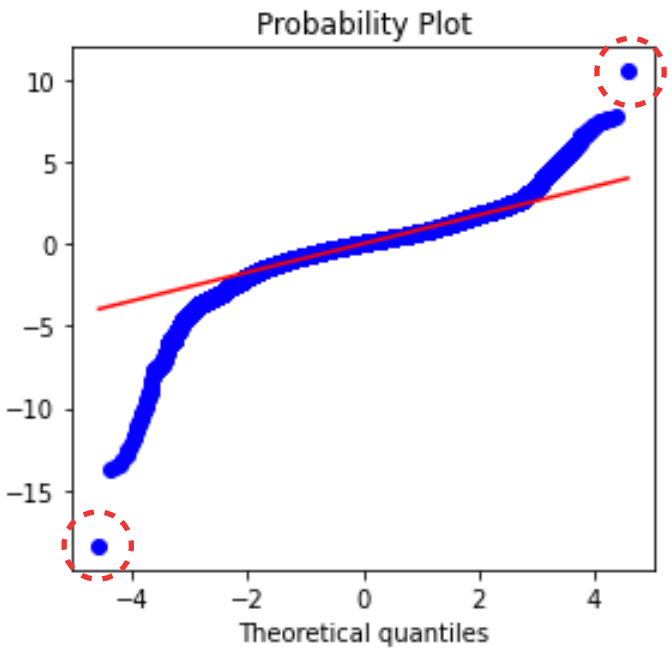


1 순위 특성의 타겟 분포도

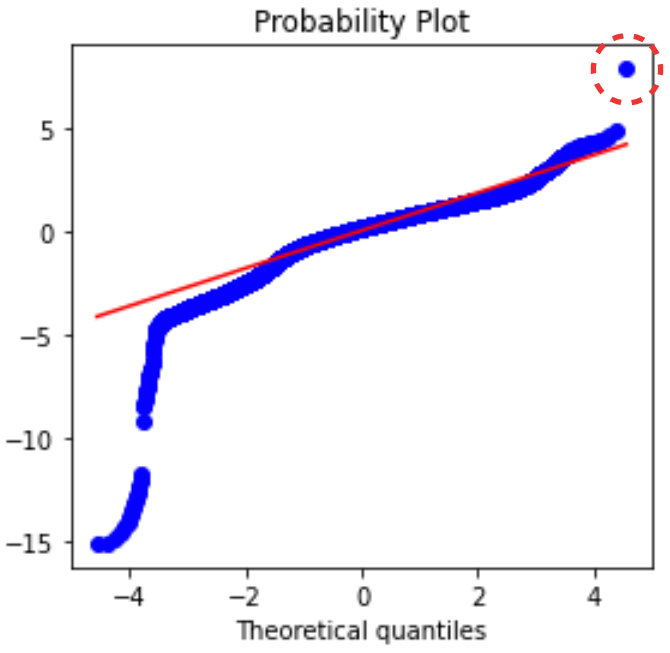


▪ EDA와 데이터 전처리

이상치 제거 : 중요 특성 위주로



V14



V12

특성 값 분포도

- **EDA와 데이터 전처리**

다운샘플링

```
df_normal = df[df.Class==0] # 클래스가 0인것 - 정상
df_fraud = df[df.Class==1] # 클래스가 1인것 - 사기
df_normal_downsampled = df_normal.sample(frac=1,random_state=0)[:df_fraud.shape[0]]
df = pd.concat([df_normal_downsampled, df_fraud]) # 샘플링한 데이터를 다시 합침
```

업샘플링

```
from imblearn.over_sampling import SMOTE
```

```
smote = SMOTE(random_state=0)
```

```
X_train, y_train = smote.fit_sample(X_train, y_train)
```

베이스 라인

```
baseline = train['Class'].value_counts(normalize = True)[0]
print('베이스라인 모델 : ',baseline *100 ,'%')
```

```
베이스라인 모델 :  50.0 %
```

- **EDA와 데이터 전처리**

다운샘플링 후 로지스틱 결과

```
modeling(lr, X_train_down, X_val_down, y_train_down, y_val_down) # 로지스틱
```

```
훈련 정확도 :  0.9503012048192772
검증 정확도 :  0.9768263512475587
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	45493
1	0.06	0.83	0.11	76
accuracy			0.98	45569
macro avg	0.53	0.90	0.55	45569
weighted avg	1.00	0.98	0.99	45569

- **EDA와 데이터 전처리**

업샘플링 후 로지스틱 결과

```
modeling(lr, X_train, X_val, y_train, y_val) # 로지스틱
```

```
훈련 정확도 :  0.9591732984523391
검증 정확도 :  0.9808857092696629
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	45492
1	0.07	0.89	0.14	76
accuracy			0.98	45568
macro avg	0.54	0.94	0.56	45568
weighted avg	1.00	0.98	0.99	45568

04

머신러닝 적용

▪ 머신러닝 적용

하이퍼 파라미터 튜닝

```
logistic 최적 하이퍼파라미터 :  {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
logistic score :  0.9592722243457845
```

```
modeling(lr_tuning,X_train, X_val, y_train, y_val)
```

```
훈련 정확도 :  0.9593079493492789
검증 정확도 :  0.9806004213483146
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	45492
1	0.07	0.89	0.13	76
accuracy			0.98	45568
macro avg	0.54	0.94	0.56	45568
weighted avg	1.00	0.98	0.99	45568

- **머신러닝 적용**

하이퍼 파라미터 튜닝

```
XGboost 최적 하이퍼파라미터 :  {'sub_sample': 0.5, 'max_depth': 5, 'learning_rate': 0.1}
XGboost log loss :  0.19755496654395355
```

```
modeling(xg_tuning,X_train_down, X_val_down, y_train_down, y_val_down)
```

```
훈련 정확도 :  1.0
검증 정확도 :  0.9714279444359104
```

	precision	recall	f1-score	support
0	1.00	0.97	0.99	45493
1	0.05	0.84	0.09	76
accuracy			0.97	45569
macro avg	0.52	0.91	0.54	45569
weighted avg	1.00	0.97	0.98	45569

- **머신러닝 적용**

테스트 셋 교차검증 결과

```
lr_tuned = LogisticRegression(C=100,  solver='lbfgs', random_state=2, max_iter=500)
modeling_cv(lr_tuned,X_train, X_test, y_train, y_test) # 로지스틱 테스트셋 결과
```

```
훈련 교차검증 :  [0.95957736 0.95950866 0.95867053 0.95917891 0.95941193]
테스트 교차검증 :  [0.99894672 0.99973666 0.99912219 0.99903441 0.99912219]
훈련 교차검증 평균 : 0.9592694780558837
테스트 교차검증 평균 :  0.9991924342218809
훈련 정확도 :  0.9593079493492789
테스트 정확도 :  0.9792314039430488
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	56865
1	0.07	0.91	0.13	96
accuracy			0.98	56961
macro avg	0.53	0.94	0.56	56961
weighted avg	1.00	0.98	0.99	56961

- **머신러닝 적용**

임계값 조정을 통한 오차행렬

임곗값: 0.3
오차 행렬
[[43547 1945]
 [6   70]]

	precision	recall	f1-score
0	1.00	0.96	0.98
1	0.03	0.92	0.07

임곗값: 0.7
오차 행렬
[[45050   442]
 [8   68]]

	precision	recall	f1-score
0	1.00	0.99	1.00
1	0.13	0.89	0.23

임곗값: 0.4
오차 행렬
[[44230 1262]
 [7   69]]

	precision	recall	f1-score
0	1.00	0.97	0.99
1	0.05	0.91	0.10

임곗값: 0.8
오차 행렬
[[45173   319]
 [8   68]]

	precision	recall	f1-score
0	1.00	0.99	1.00
1	0.18	0.89	0.29

- **머신러닝 적용**

최적의 임계값 찾기

```
recall_predict = Binarizer(threshold=0.1).fit_transform(y_pred_threshold.reshape(-1,1))
matrix_score(y_test, recall_predict)
```

```
오차 행렬
[[49183  7682]
 [    3    93]]
              precision    recall  f1-score   support

           0       1.00      0.86      0.93     56865
           1       0.01      0.97      0.02        96

    accuracy                           0.87     56961
   macro avg       0.51      0.92      0.48     56961
weighted avg       1.00      0.87      0.93     56961
```

임계값 0.1일 때 **재현율 97%** but **f1-score 2%**

- **머신러닝 적용**

최적의 임계값 찾기

```
custom_predict = Binarizer(threshold=0.999).fit_transform(y_pred_threshold.reshape(-1,1))
matrix_score(y_test, custom_predict)
```

```
오차 행렬
[[56843    22]
 [   16    80]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56865
           1       0.78      0.83      0.81        96

    accuracy                           1.00     56961
   macro avg       0.89      0.92      0.90     56961
weighted avg       1.00      1.00      1.00     56961
```

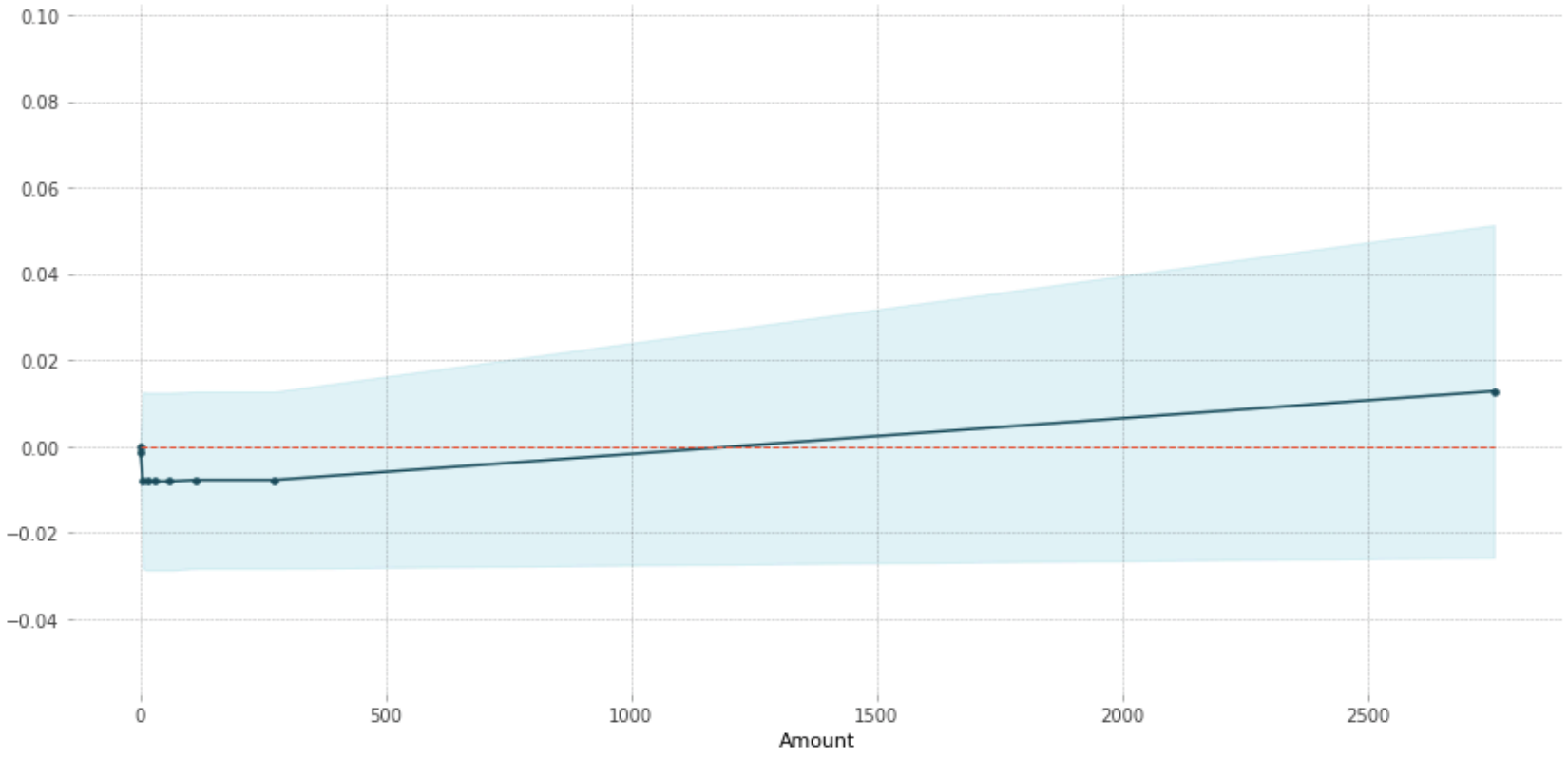
임계값 : 0.999 일때 **재현율 83% f1-score 81%**

05

머신러닝 모델 시각화

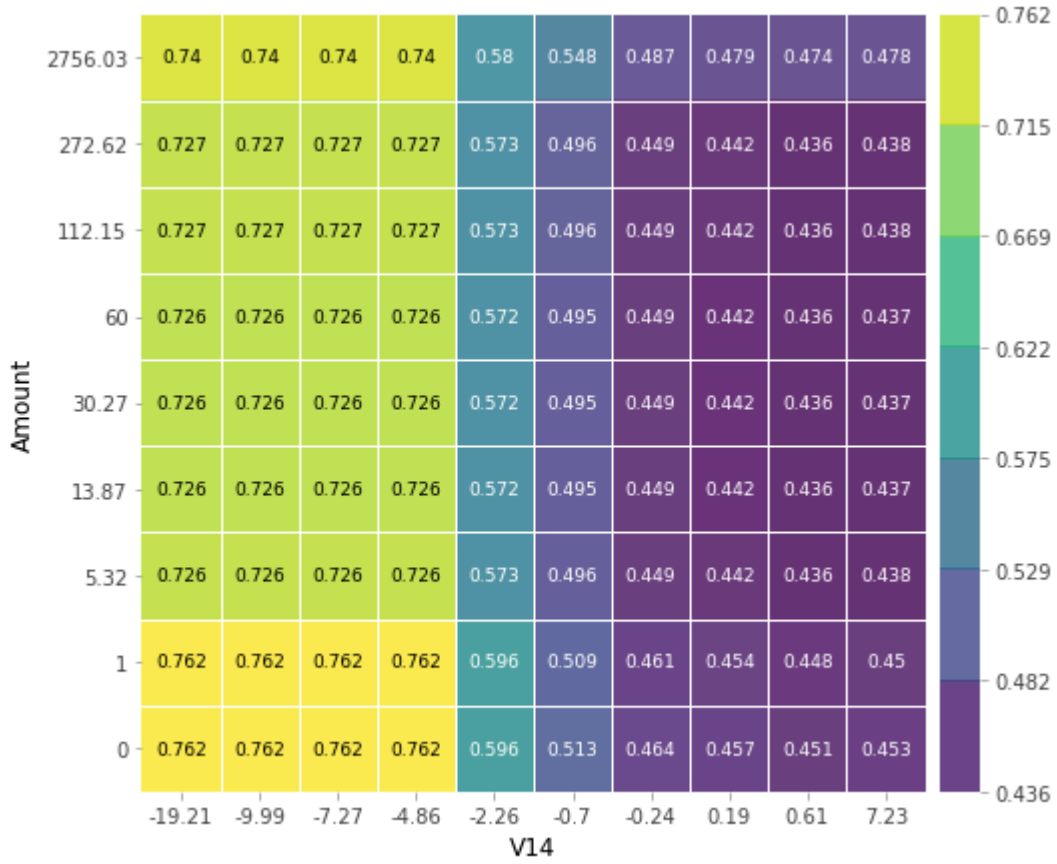
▪ 모델 해석

PDP for feature“Amount”



- **모델 해석**

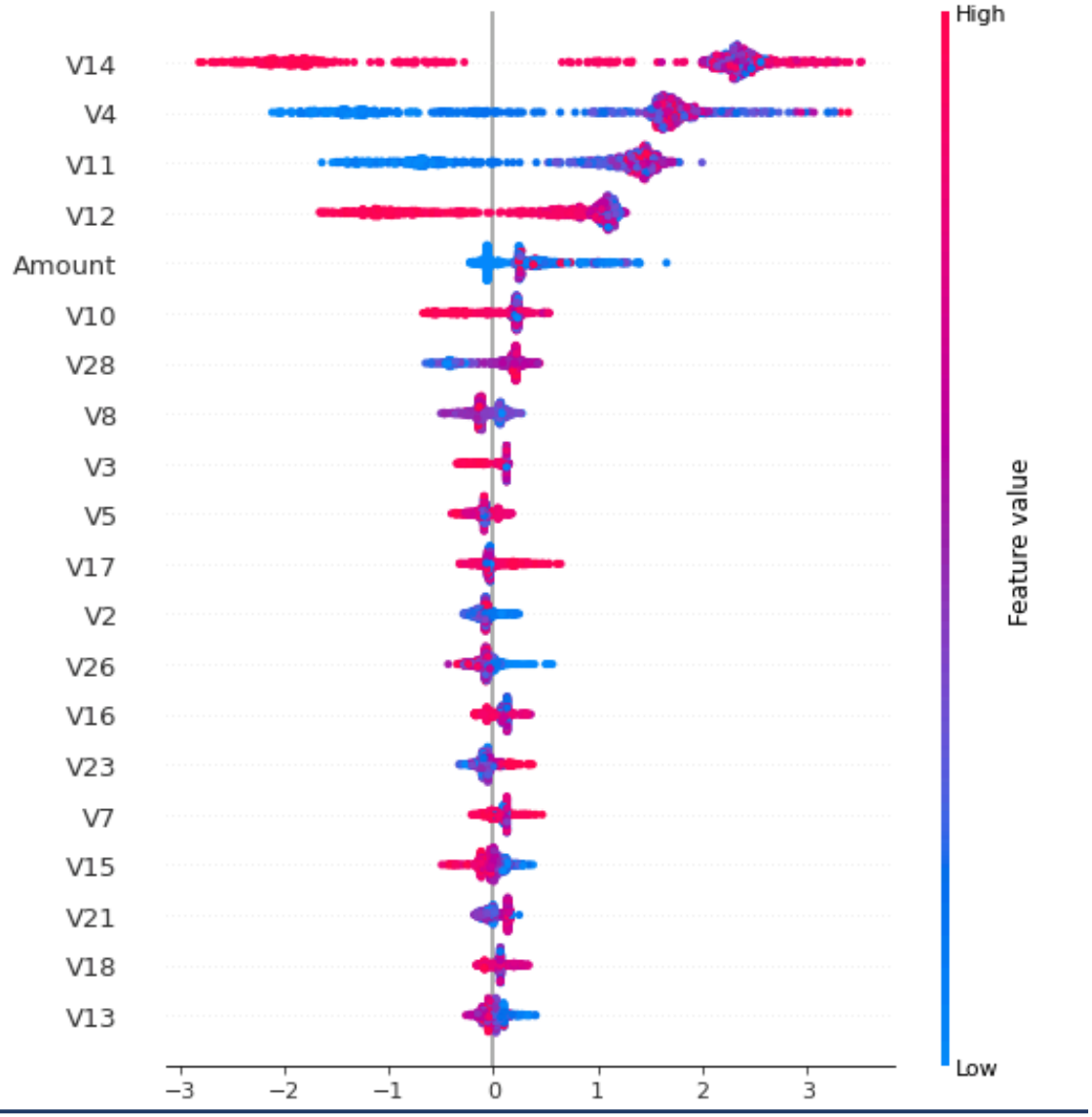
PDP interactor for feature“Amount” and “V14”



- **모델 해석**

Shap value를 활용한

특성 중요도



감사합니다.