Ch3

April 9, 2025

1 Lending Club Data

- Lending Club 데이터
  - 2007~2017 3분기의 기간 동안의 대출 데이터 (Kaggle 제공, https://www.kaggle.com/wendykan/lending-club-loan-data)
  - 원 자료는 88만개 이상의 사례의 150개 특성변수에 대한 정보를 담고 있음.
  - 10만 건을 random sampling한 자료에 대해 데이터 정제, 특성 선택 및 변환 등의 전처리를 적용한 데이터를 이용하여 분석.

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[3]:
```
datasets = pd.read_csv('/content/drive/MyDrive/2025 Spring/ㅂㅣㄱㄷㅔㅇㅣㅌㅓㅇㅘ␣
↪ㄱㅡㅁㅇㅠㅇㅈㅏㄹㅛㅂㅜㄴㅅㅓㄱ/LoanData.csv')
datasets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86138 entries, 0 to 86137
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              86138 non-null  int64
 1   loan_amnt               86138 non-null  float64
 2   funded_amnt             86138 non-null  float64
 3   term                    86138 non-null  int64
 4   int_rate                86138 non-null  float64
 5   installment             86138 non-null  float64
 6   grade                   86138 non-null  int64
 7   sub_grade               86138 non-null  int64
 8   home_ownership          86138 non-null  int64
```

```
 9   verification_status    86138 non-null  int64
 10  purpose                86138 non-null  int64
 11  addr_state             86138 non-null  int64
 12  dti                    86138 non-null  float64
 13  earliest_cr_line       86138 non-null  int64
 14  open_acc               86138 non-null  float64
 15  revol_util             86094 non-null  float64
 16  initial_list_status    86138 non-null  int64
 17  last_pymnt_amnt        86138 non-null  float64
 18  application_type       86138 non-null  int64
 19  acc_open_past_24mths   86138 non-null  float64
 20  avg_cur_bal            86138 non-null  float64
 21  bc_open_to_buy         85142 non-null  float64
 22  bc_util                85089 non-null  float64
 23  mo_sin_old_rev_tl_op   86138 non-null  float64
 24  mo_sin_rcnt_rev_tl_op  86138 non-null  float64
 25  mort_acc               86138 non-null  float64
 26  num_actv_rev_tl        86138 non-null  float64
 27  charged_off            86138 non-null  int64
 28  log_annual_inc         86138 non-null  float64
 29  fico_score             86138 non-null  float64
dtypes: float64(18), int64(12)
memory usage: 19.7 MB
```

[4]:
```
datasetX, datasetY = datasets.drop('charged_off', axis=1),␣
↪datasets['charged_off']
```

[5]:
```
X_train, X_test, Y_train, Y_test = train_test_split(datasetX, datasetY,␣
↪test_size=0.2, random_state=123)
X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size=1/
↪8, random_state=456)
```

[6]:
```
print( X_train.shape, X_val.shape, X_test.shape )
```

```
(60296, 29) (8614, 29) (17228, 29)
```

[7]:
```
X_train.shape
```

[7]:
```
(60296, 29)
```

[8]:
```
Y_train.value_counts()
```

[8]:
```
charged_off
0    48829
1    11467
Name: count, dtype: int64
```

2 XGBoost

- xgboost.XGBClassifier(분류)와 xgboost.XGBRegressor(회귀) 클래스
  - Implementation of the scikit-learn API for XGBoost regressor/classifier ## XGBoost 파라미터
- General 파라미터
  - booster
    * 'gbtree'(default), 'gblinear', 'dart' 중 하나를 선택.
  - verbosity
    * 메세지 출력 범위 설정, 0(silent), 1(warning), 2(info), 3(debug)
  - n_jobs
    * xgboost를 실행하는 데 사용되는 병렬 thread의 수.
    * default는 모두 사용하는 것.
- Booster 파라미터 ('gbtree' 기준)
  - n_estimators
    * 몇 개의 estimator를 포함하는지를 입력.
    * default는 100.
  - learning_rate
    * 학습률. default는 0.3이고 0~1의 사이로 입력.
    * 과적합을 방지하기 위해 각 estimator의 가중치를 줄여주는 역할을 함.
    * 작을수록 모델이 견고해지고 과적합 방지에 좋지만, 더 많은 estimators가 요구되며 학습시간은 길어짐.
  - min_child_weight
    * 트리 노드 분할 시 자식노드에 속한 자료들의 weight의 합에 대한 최소값.
  - gamma
    * pruning 관련 하이퍼파라미터
    * 양의 실수 값으로 설정. default는 0임.
    * 클 수록 과적합을 방지하나, 너무 큰 경우 언더피팅이 될 수 있음.
  - reg_lambda
    * L2 규제 하이퍼파라미터
    * 커질수록 보수적인 모델이 되어 과적합을 방지하나, 너무 큰 경우 언더피팅이 될 수 있음.
  - reg_alpha
    * L1 규제 하이퍼파라미터
    * 커질수록 보수적인 모델이 되어 과적합을 방지하나, 너무 큰 경우 언더피팅이 될 수 있음.
    * 특성변수가 sparse하거나 매우 많을 때 적용 권장.
  - max_depth
    * estimator로 사용되는 각 트리의 최대깊이. default는 6임.
    * 최대 깊이의 트리는 -1로 설정.
    * 데이터의 복잡도에 따라 적정한 깊이가 설정되어야 함.
    * 너무 작으면 언더피팅이 될 수 있고, 너무 크면 오버피팅의 가능성이 높아지며 학습시간이 길어짐.
  - subsample
    * 각 트리 단위로 적용되는 Row sampling 비율로 과적합을 방지하고 학습시간을 줄여줌.
    * 0~1 사이의 값을 입력. default는 1임.
  - colsample_bytree
    * 각 트리 단위로 적용되는 Column sampling 비율로 과적합을 방지하고 학습시간을

줄여줌.
    * 0~1 사이의 값을 입력. default는 1임.
  - tree_method
    * 'auto'(default), 'exact', 'approx', 'hist', 'gpu_hist' 중 하나를 선택.
    * 'auto'는 데이터의 크기가 작은 경우에는 'exact', 큰 경우에는 'approx'를 적용하는 것임.
  - sketch_eps
    * tree_method가 'approx'인 경우에만 적용됨.
    * 버킷의 수는 1/sketch_eps로 결정되며, default는 0.03임.
  - scale_pos_weight
    * 범주의 비중이 불균형인 이진 분류 문제에서 cost-sensitive training을 하도록 함.
    * sum(negative instances) / sum(positive instances)로 입력.
- Learning Task 파라미터
  - objective
    * 손실함수 지정. 이를 최소로 하도록 학습함.
    * 자주 활용되는 손실함수
      · reg:squarederror : 회귀용. 오차제곱 손실.
      · reg:squaredlogerror : 회귀용. 오차로그제곱 손실.
      · binary:logistic : 이진 분류용. 예측 확률 반환.
      · multi:softmax : 다중 분류용. 예측 클래스 반환.
      · multi:softprob : softmax와 동일한데, 예측 확률 반환.
  - base_score
    * 초기편향치로, defatul는 0.5임.
  - eval_metric
    * 평가지표 지정. 각 스텝마다 완성된 모델을 이 지표를 통해 평가함.
    * 자주 활용되는 평가지표
      · rmse : root mean square error (회귀의 default)
      · mae : mean absolute error
      · logloss : negative log-likelihood
      · mlogloss : multiclass logloss
      · error : binary classification error rate로 임계치는 0.5 기준 (분류의 default) (error@t : 임계치 t를 적용한 error rate)
      · merror : multiclass classification error rate
      · auc : area under the curve
  - early_stopping_rounds
    * eval_metric의 지표가 early_stopping_rounds 횟수 동안 개선되지 않으면, n_estimators에 도달하기 전에 멈추도록 함.
    * 양의 정수값으로 입력. default는 0.
    * fit() 메서드에서 검증용 데이터를 eval_set에 지정해 주어야 함.
    * 훈련이 끝난 인스턴스의 best_ntree_limit 속성을 이용하여 최적의 tree 갯수를 확인할 수 있으며, predict() 메서드로 예측 시에는 ntree_limit=best_ntree_limit가 적용됨.

XGBoost 모델 학습과 예측

- 주요 메서드
  - fit(X_train, y_train) 메서드로 모델을 학습
    * eval_set : [(X_eval, y_eval)] 형식으로 검증데이터 지정할 수 있음.

- predict(X_test) 메서드로 학습된 모델을 이용한 예측
    - ntree_limit : default는 0(모든 tree를 사용하는 것)인데, early_stopping이 적용된 경우 best_ntree_limit이 적용됨.
- evals_result() 메서드로 검증 데이터에 대한 평가결과를 확인. 단, fit() 메서드에서 eval_set에 검증용 데이터가 지정되어 있어야 함.

- 주요 속성
  - feature_importances_ : 각 특성변수 별 특성중요도

[9]:
```
! pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.11/dist-
packages (3.0.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from xgboost) (2.0.2)
Requirement already satisfied: nvidia-nccl-cu12 in
/usr/local/lib/python3.11/dist-packages (from xgboost) (2.26.2.post1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages
(from xgboost) (1.14.1)
```

[10]:
```
from xgboost import XGBClassifier
model= XGBClassifier( booster = 'gbtree', objective='binary:logistic',
                      learning_rate=0.1,
                      scale_pos_weight =float(Y_train.value_counts()[0]) /␣
↪Y_train.value_counts()[1],
                      verbosity=1 )
```

[11]:
```
params = {
          'n_estimators':[100, 500],
          'max_depth':[3, 6, 9],
          'min_child_weight':[0, 0.1, 0.3, 0.5],
          'gamma' : [0, 0.1, 1, 5],
          'colsample_bytree':[0.6, 0.8],
          }
```

[12]:
```
from sklearn.model_selection import RandomizedSearchCV
grid_xgb = RandomizedSearchCV( model,
                       param_distributions = params,
                       n_iter = 25,
                       cv = 3,
                       scoring ='accuracy',
                       refit = True)
grid_xgb.fit( X_train, Y_train )
```

[12]:
```
RandomizedSearchCV(cv=3,
                   estimator=XGBClassifier(base_score=None, booster='gbtree',
                                           callbacks=None,
                                           colsample_bylevel=None,
                                           colsample_bynode=None,
```

```
                                        colsample_bytree=None, device=None,
                                        early_stopping_rounds=None,
                                        enable_categorical=False,
                                        eval_metric=None, feature_types=None,
                                        feature_weights=None, gamma=None,
                                        grow_policy=None,
                                        importance_type=None,
                                        interaction_const...
                                        max_delta_step=None, max_depth=None,
                                        max_leaves=None,
                                        min_child_weight=None, missing=nan,
                                        monotone_constraints=None,
                                        multi_strategy=None,
                                        n_estimators=None, n_jobs=None,
                                        num_parallel_tree=None, ...),
                   n_iter=25,
                   param_distributions={'colsample_bytree': [0.6, 0.8],
                                        'gamma': [0, 0.1, 1, 5],
                                        'max_depth': [3, 6, 9],
                                        'min_child_weight': [0, 0.1, 0.3, 0.5],
                                        'n_estimators': [100, 500]},
                   scoring='accuracy')
```

[13]:
```
grid_xgb.best_params_
```

[13]:
```
{'n_estimators': 500,
 'min_child_weight': 0.1,
 'max_depth': 9,
 'gamma': 0.1,
 'colsample_bytree': 0.6}
```

[14]:
```
grid_xgb.best_score_
```

[14]:
```
np.float64(0.8815343863446312)
```

[15]:
```
gridresult = pd.DataFrame(grid_xgb.cv_results_).iloc[:,[4, 5, 6, 7, 8, 13, 14,
 ↪15]]
gridresult.sort_values(['rank_test_score'])[ :10 ]
```

[15]:
```
    param_n_estimators  param_min_child_weight  param_max_depth  param_gamma  \
0                  500                     0.1                9          0.1
16                 500                     0.3                9          0.1
11                 500                     0.5                9          0.0
8                  500                     0.1                6          0.1
7                  500                     0.5                6          0.0
2                  500                     0.3                9          1.0
21                 100                     0.0                9          0.1
```

```
13                   100                      0.3                  9           1.0
19                   100                      0.3                  9           1.0
1                    500                      0.1                  6           1.0

    param_colsample_bytree  mean_test_score  std_test_score  rank_test_score
0                      0.6         0.881534        0.001998                1
16                     0.8         0.881319        0.001716                2
11                     0.6         0.881170        0.002068                3
8                      0.8         0.861798        0.001995                4
7                      0.6         0.860339        0.003105                5
2                      0.8         0.858813        0.002188                6
21                     0.6         0.854684        0.002704                7
13                     0.6         0.854352        0.002033                8
19                     0.8         0.852047        0.001997                9
1                      0.8         0.846690        0.000854               10
```

[16]:

```python
finalmodel = XGBClassifier( booster = 'gbtree', objective = 'binary:logistic',
                      verbosity = 0,
                      colsample_bytree = 0.6, gamma = 0.1, max_depth = 9,␣
 ↪min_child_weight = 0.1,
                      n_estimators = 10000, learning_rate = 0.01,
                      scale_pos_weight =float(Y_train.value_counts()[0]) /␣
 ↪Y_train.value_counts()[1],
                      eval_metric = 'logloss',
                      early_stopping_rounds = 1000 )
finalmodel.fit( X_train, Y_train,
              eval_set = [ (X_train, Y_train), (X_val, Y_val) ],
)
```

```
[0]     validation_0-logloss:0.68703    validation_1-logloss:0.68718
[1]     validation_0-logloss:0.68503    validation_1-logloss:0.68546
[2]     validation_0-logloss:0.67901    validation_1-logloss:0.67961
[3]     validation_0-logloss:0.67701    validation_1-logloss:0.67787
[4]     validation_0-logloss:0.67116    validation_1-logloss:0.67219
[5]     validation_0-logloss:0.66922    validation_1-logloss:0.67048
[6]     validation_0-logloss:0.66350    validation_1-logloss:0.66493
[7]     validation_0-logloss:0.65790    validation_1-logloss:0.65948
[8]     validation_0-logloss:0.65239    validation_1-logloss:0.65415
[9]     validation_0-logloss:0.64700    validation_1-logloss:0.64890
[10]    validation_0-logloss:0.64169    validation_1-logloss:0.64375
...
[4730]  validation_0-logloss:0.03710    validation_1-logloss:0.24906
[4731]  validation_0-logloss:0.03709    validation_1-logloss:0.24905
[4732]  validation_0-logloss:0.03708    validation_1-logloss:0.24906
[4733]  validation_0-logloss:0.03706    validation_1-logloss:0.24907
[4734]  validation_0-logloss:0.03705    validation_1-logloss:0.24907
[4735]  validation_0-logloss:0.03704    validation_1-logloss:0.24908
[4736]  validation_0-logloss:0.03702    validation_1-logloss:0.24910
```

```
[4737]  validation_0-logloss:0.03701    validation_1-logloss:0.24910
[4738]  validation_0-logloss:0.03700    validation_1-logloss:0.24911
[4739]  validation_0-logloss:0.03697    validation_1-logloss:0.24911
```

[16]:
```
XGBClassifier(base_score=None, booster='gbtree', callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.6, device=None, early_stopping_rounds=1000,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, feature_weights=None, gamma=0.1,
              grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.01, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=9, max_leaves=None,
              min_child_weight=0.1, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=10000, n_jobs=None,
              num_parallel_tree=None, ...)
```

[17]:
```
result = finalmodel.evals_result()
result.keys()
```

[17]:
```
dict_keys(['validation_0', 'validation_1'])
```

[18]:
```
result['validation_0'].keys()
```

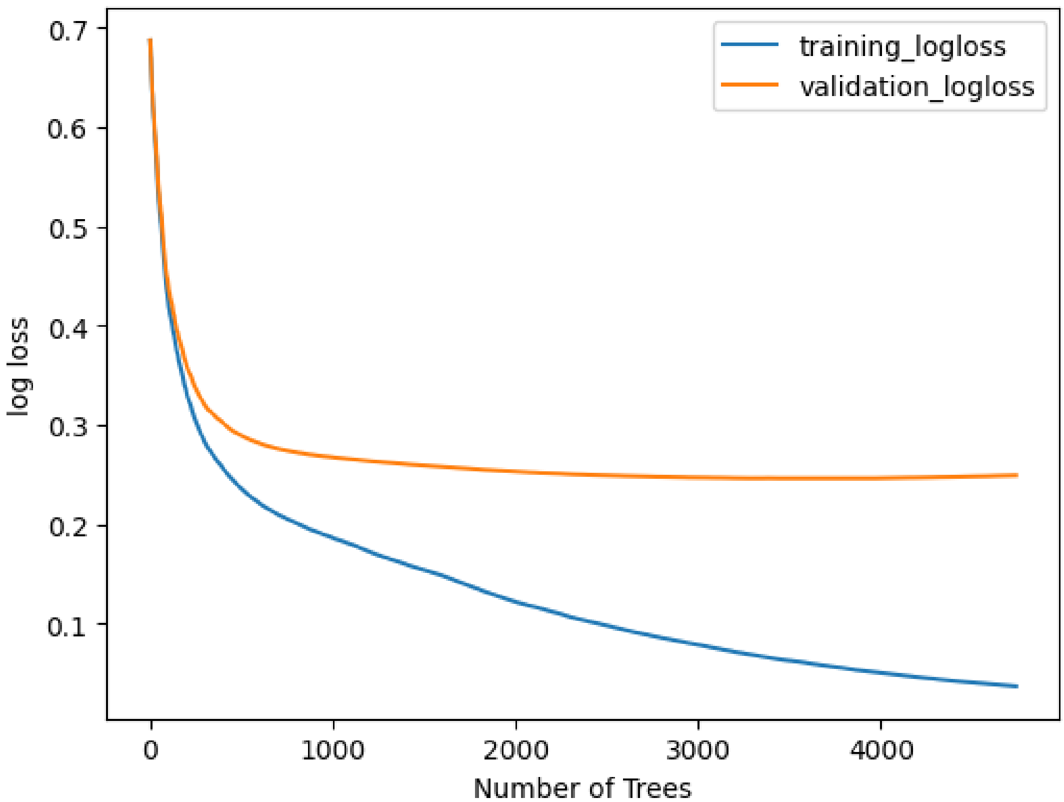
[18]:
```
odict_keys(['logloss'])
```

[19]:
```
result['validation_0']['logloss'][:10]
```

[19]:
```
[0.6870319234424502,
 0.6850344511464433,
 0.6790085127281861,
 0.677014981426523,
 0.6711560831954724,
 0.6692185087048661,
 0.6634964771905866,
 0.6578983055714098,
 0.6523943465073054,
 0.6469965264046953]
```

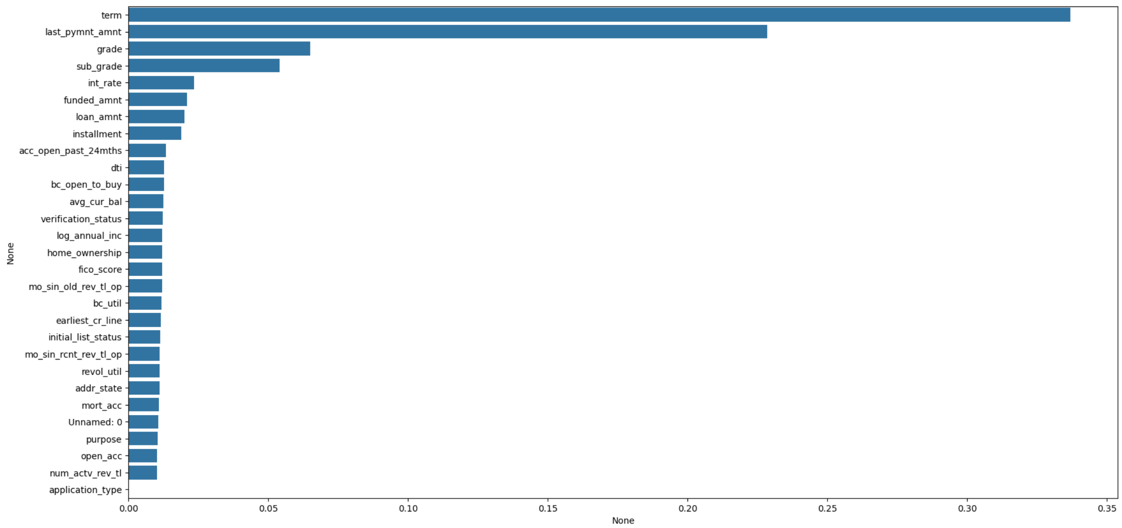
[20]:
```
plt.plot(result['validation_0']['logloss'], label = 'training_logloss')
plt.plot(result['validation_1']['logloss'], label = 'validation_logloss')
plt.xlabel('Number of Trees')
plt.ylabel('log loss')
plt.legend()
```

[20]:
```
<matplotlib.legend.Legend at 0x7f5a31c16390>
```



[21]:
```
imp_values = pd.Series( finalmodel.feature_importances_, index = X_train.columns␣
↪)
imp_values = imp_values.sort_values( ascending = False )
plt.figure( figsize = (20, 10))
sns.barplot( x = imp_values, y = imp_values.index )
```

[21]:
```
<Axes: xlabel='None', ylabel='None'>
```



[22]:
```
finalmodel.predict(  X_train.iloc[0:1, :] )
```

[22]:
```
array([0])
```

[23]:
```
finalmodel.predict_proba(  X_train.iloc[0:1, :] )
```

[23]:
```
array([[9.9996459e-01, 3.5393325e-05]], dtype=float32)
```

[24]:
```
Y_pred = finalmodel.predict( X_test )
```

[25]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix ( Y_test, Y_pred )
```

[25]:
```
array([[12800,  1275],
       [  853,  2300]])
```

[26]:
```
accuracy_score ( Y_test, Y_pred )
```

[26]:
```
0.87648014859531
```

[27]:
```
!pip install shap
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages
(0.47.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from shap) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages
(from shap) (1.14.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-
packages (from shap) (1.6.1)
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(from shap) (2.2.2)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.11/dist-
packages (from shap) (4.67.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.11/dist-
packages (from shap) (24.2)
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.11/dist-
packages (from shap) (0.0.8)
Requirement already satisfied: numba>=0.54 in /usr/local/lib/python3.11/dist-
packages (from shap) (0.61.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.11/dist-
packages (from shap) (3.1.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.11/dist-packages (from shap) (4.13.1)
Requirement already satisfied: llvmlite<0.45,>=0.44.0dev0 in
/usr/local/lib/python3.11/dist-packages (from numba>=0.54->shap) (0.44.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->shap) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->shap) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas->shap) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn->shap) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->shap) (1.17.0)
```

[28]:
```
import shap
idx=12
print( X_train.iloc[idx, :] )
```

```
Unnamed: 0              60561.00000
loan_amnt               12175.00000
funded_amnt             12175.00000
term                       36.00000
int_rate                   16.99000
installment               434.02000
grade                       3.00000
sub_grade                  17.00000
home_ownership              3.00000
verification_status         1.00000
purpose                     2.00000
addr_state                 17.00000
dti                         5.38000
earliest_cr_line          495.00000
open_acc                    6.00000
```

```
revol_util                    78.10000
initial_list_status            1.00000
last_pymnt_amnt              433.61000
application_type               0.00000
acc_open_past_24mths           2.00000
avg_cur_bal                 1133.00000
bc_open_to_buy               271.00000
bc_util                       92.90000
mo_sin_old_rev_tl_op          82.00000
mo_sin_rcnt_rev_tl_op         17.00000
mort_acc                       0.00000
num_actv_rev_tl                6.00000
log_annual_inc                 4.68125
fico_score                   682.00000
Name: 51478, dtype: float64
```

[29]:
```
print( Y_train.iloc[idx] )
```

```
0
```

[30]:
```
finalmodel.predict( X_train.iloc[idx:(idx+1), :])
```

[30]:
```
array([0])
```

[31]:
```
finalmodel.predict_proba( X_train.iloc[idx:(idx+1), :])
```

[31]:
```
array([[0.74943763, 0.25056237]], dtype=float32)
```

[32]:
```
explainer = shap.TreeExplainer( finalmodel )
shap_values = explainer.shap_values( X_train )
```

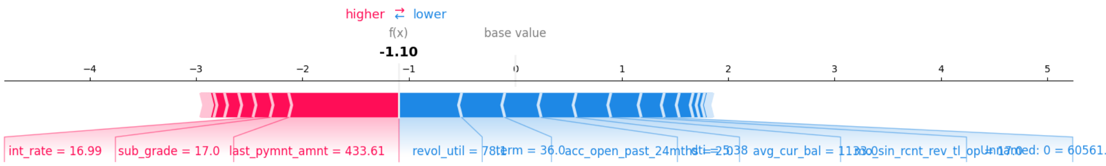
[33]:
```
shap_values
```

[33]:
```
array([[-2.4230785e-03, -8.8768058e-02, -4.1600831e-02, ...,
        -1.2775280e-01,  5.6166384e-02,  1.6114617e-02],
       [ 1.4133357e-01,  2.4022374e-01,  9.8373510e-02, ...,
         7.4724652e-02, -3.5402644e-02, -1.5340754e-01],
       [-4.4971026e-02, -2.3305659e-04, -6.9840974e-03, ...,
         1.3030774e-02,  8.7066591e-02,  1.9344002e-01],
       ...,
       [-2.1238716e-01,  1.2418623e-01,  6.1614510e-02, ...,
        -1.8524936e-01, -1.3783604e-01,  7.4659444e-02],
       [-5.1938850e-02, -2.8449601e-01, -9.8605856e-02, ...,
         2.1632536e-01, -9.9154197e-02, -4.3514479e-02],
       [ 3.1928816e-03, -5.4380249e-02, -2.7232980e-02, ...,
        -1.4158332e-01, -3.2802559e-02,  4.6506036e-02]], dtype=float32)
```

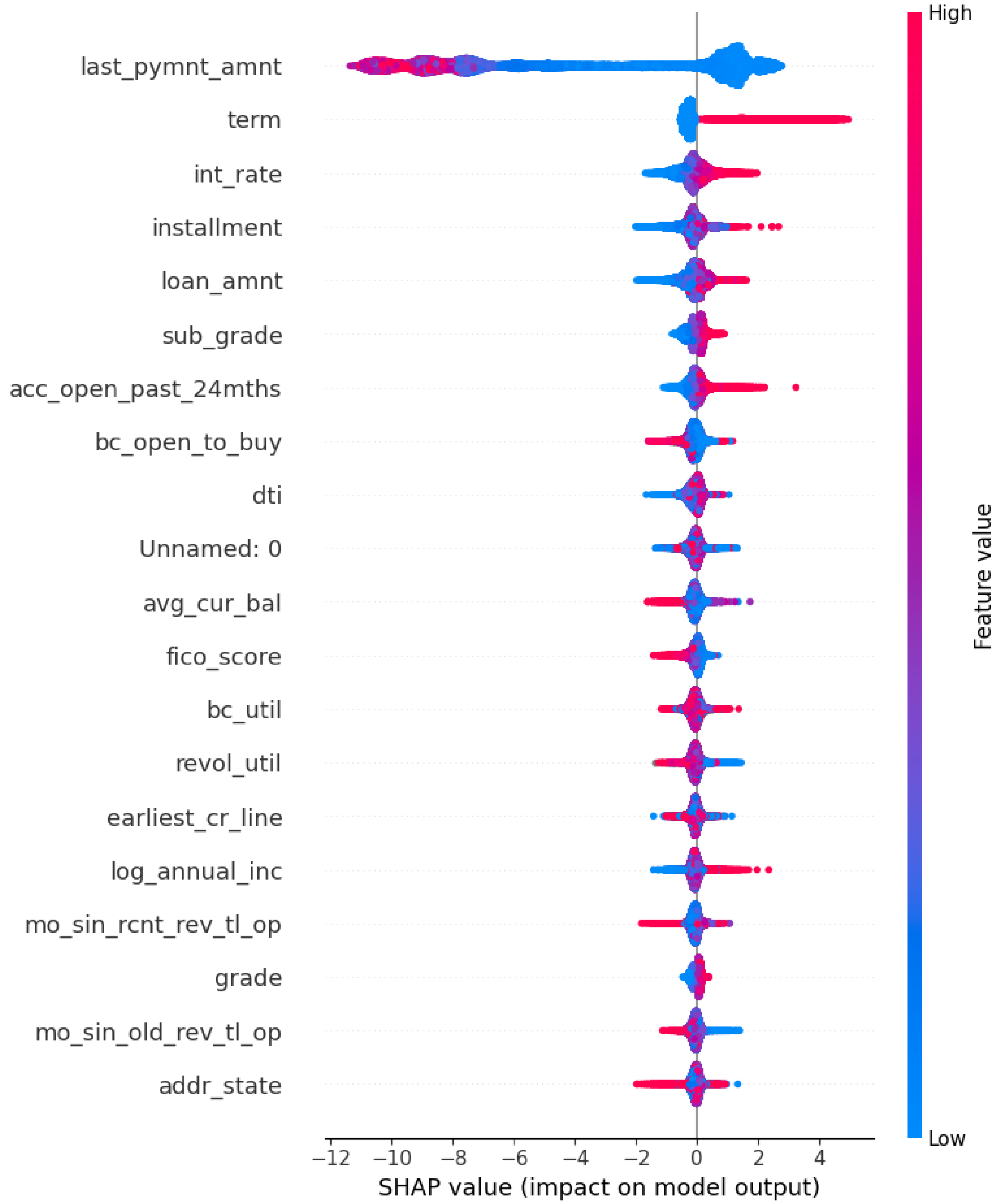
[34]:
```
explainer.expected_value
```

[34]: np.float32(-0.003537062)

[35]:
```
shap.force_plot( explainer.expected_value, shap_values[idx, :], X_train .
↪iloc[idx,:],matplotlib=True  )
```



[36]:
```
shap.summary_plot( shap_values, X_train  )
```


last_pymnt_amnt
term
int_rate
installment
loan_amnt
sub_grade
acc_open_past_24mths
bc_open_to_buy
dti
Unnamed: 0
avg_cur_bal
fico_score
bc_util
revol_util
earliest_cr_line
log_annual_inc
mo_sin_rcnt_rev_tl_op
grade
mo_sin_old_rev_tl_op
addr_state
−12 −10 −8 −6 −4 −2 0 2 4
SHAP value (impact on model output)
High
Feature value
Low