test1

February 3, 2026

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from holoviews.annotators import preprocess
from material.plugins.search.config import pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve,␣
↪roc_auc_score
```

[4]:
```
#
try:
    train_data = pd.read_csv('data/train.csv')
    test_data = pd.read_csv('data/testA.csv')
    sample_submit = pd.read_csv('data/sample_submit.csv')
    print("    ")
except FileNotFoundError as e:
    print(f"    : {e}")
    raise
```

[5]:
```
#
train_data = train_data.dropna(subset=['isDefault'])
```

[6]:
```
#
X = train_data.drop(['id','isDefault'], axis=1)
y = train_data['isDefault']
X_test = test_data.drop(['id'], axis=1)
```

[7]:
```
#
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object']).columns
```

[8]:
```
#
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
#
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

[9]:
```
#
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features),
    ]
)
```

[10]:
```
#
X_train, X_valid, y_train, y_valid = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)
```

[30]:
```
#      pipeline
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(
        n_estimators=100,
        n_jobs=-1,       #  CPU
        max_depth=20,    #
        class_weight='balanced',
        random_state=42
    )),
])
```

[31]:
```
#
model.fit(X_train, y_train)
```

[31]:
```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
SimpleImputer()),
                                                                  ('scaler',
StandardScaler())]),
```

```
                                                  Index(['loanAmnt', 'term',
'interestRate', 'installment', 'employmentTitle',
       'homeOwnership', 'annualIncome', 'verificationStatus', 'purpose',
       'postCode', 'regionCode', 'dti', 'delinquency_2years', 'ficoRangeLow',
       'fico…
      dtype='object')),
                                                 ('cat',
                                                  Pipeline(steps=[('imputer',
SimpleImputer(fill_value='missing',
 strategy='constant')),
                                                                  ('onehot',
OneHotEncoder(handle_unknown='ignore'))]),
                                                  Index(['grade', 'subGrade',
'employmentLength', 'issueDate',
       'earliesCreditLine'],
      dtype='object'))])),
                ('classifier',
                 RandomForestClassifier(class_weight='balanced', max_depth=20,
                                        n_jobs=-1, random_state=42))])
```

[38]:
```
#
y_pred = model.predict(X_valid)
print(f'Validation Accuracy: {accuracy_score(y_valid, y_pred)}')
```

```
Validation Accuracy: 0.65893125
```

[33]:
```
#
test_data_processed = model.named_steps['preprocessor'].transform(X_test)
test_predictions = model.named_steps['classifier'].predict(test_data_processed)
```

[39]:
```
#
print(f'Validation Accuracy: {accuracy_score(y_valid, y_pred):.4f}')
```

```
Validation Accuracy: 0.6589
```

[40]:
```
#
submission = sample_submit.copy()
submission['isDefault'] = test_predictions
submission.to_csv('submission.csv', index=False)
```

[42]:
```
#   1   0
print("    ")
print(y_train.value_counts())

#           1
print("\n     ")
print(pd.Series(test_predictions).value_counts())
```

```
isDefault
0    512066
1    127934
Name: count, dtype: int64


0    120207
1     79793
Name: count, dtype: int64
```