머신러닝을 활용한 대출 승인 예측 실습

프로젝트 개요 및 데이터 이해

프로젝트 소개

✅ **목표**: 주어진 대출 신청 데이터로부터 **대출 승인 여부**(1: 승인, 0: 거절)를 예측하는 머신러닝 모델을 개발합니다.

✅ 활용 목적

- 금융기관의 대출 심사 자동화
- 신용 위험 분석 및 사전 리스크 대응

✅ 사용 모델

- 로지스틱 회귀, SVM, KNN

프로젝트 개요 및 데이터 이해

데이터셋 개요

- 총 14개의 변수로 구성된 CSV 파일
- 각 행(row)은 1명의 대출 신청자 정보를 나타냄
- 타겟 변수: 대출 승인 여부

프로젝트 개요 및 데이터 이해

데이터 컬럼 설명

컬럼명	설명	데이터 예시
나이	신청자의 나이 (연령)	22, 25
성별	남성/여성	male, female
교육 수준	최종 학력 수준	High School, Bachelor
연 소득	연간 소득 (단위: 원)	71948, 12438
근무 경력(년)	직장 경력 (년 단위)	0, 3
주택 소유 형태	주거 형태	RENT, OWN, MORTGAGE
대출 금액	신청한 대출 금액	35000, 5500
대출 목적	대출 사용 목적	PERSONAL, EDUCATION, MEDICAL
대출 이자율	적용될 이자율 (%)	16.02, 12.87
소득 대비 대출금 비율	대출금 / 소득 비율	0.49, 0.08
신용 이력 길이(년)	신용 기록 보유 기간	3, 2
신용 점수	개인 신용 점수 (일반적으로 300~850 범위)	561, 635
과거 대출 연체 여부	이전 대출의 연체 여부	Yes, No
대출 승인 여부	대출이 실제로 승인되었는지 여부 (1: 승인, 0: 거절)	0, 1

데이터 탐색 및 시각화

라이브러리 설치 및 임포트

```
# 📦 Step 0: 라이브러리 설치 및 임포트
!pip install feature-engine

# ✅ 데이터 분석 및 시각화를 위한 라이브러리
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import numpy as np
import pandas as pd

# ✅ 머신러닝 관련 라이브러리
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve

# ✅ 기타 유틸
import xgboost as xgb
import warnings
warnings.filterwarnings("ignore")
```

데이터 탐색 및 시각화

데이터 로딩 및 기본 정보 확인

- 데이터셋을 불러오고 df.info()로 컬럼명, null 여부, 데이터 타입 등 기초 정보 확인
- 총 14개의 변수, 0개의 결측치
- 대부분의 변수는 수치형 (float64, int64)
- 몇몇 변수는 문자열 형태 (object) → 범주형 처리 대상
- 범주형 변수는 라벨 인코딩 또는 원-핫 인코딩 필요
- 수치형 변수는 이상치 제거, 정규화 필요

```
# 📊 Step 1: 데이터 로드 및 기본 정보 확인
df = pd.read_csv("./loan_data.csv")
df.info()

# 📌 나이 컬럼 타입 변환
df['나이'] = df['나이'].astype('int')

# 🔍 범주형 / 수치형 컬럼 구분
cat_cols = [var for var in df.columns if df[var].dtypes == 'object']
num_cols = [var for var in df.columns if df[var].dtypes != 'object']
print(f'Categorical columns: {cat_cols}')
print(f'Numerical columns: {num_cols}')
```

데이터 탐색 및 시각화

범주형 변수 시각화

- 변수별 막대그래프
- 각 카테고리 비율(%)과 "5% 기준선" 시각화
- 분포가 불균형한 컬럼은 모델 학습 시 주의 필요
- '대출 목적'처럼 클래스가 많은 경우 → 희소 클래스 처리 검토

```python
# 📊 Step 2: 범주형 변수 분포 시각화

def plot_categorical_column(dataframe, column):
    plt.figure(figsize=(7, 7))
    ax = sns.countplot(x=dataframe[column])
    total_count = len(dataframe[column])
    threshold = 0.05 * total_count
    category_counts = dataframe[column].value_counts(normalize=True) * 100

    # 기준선 추가
    ax.axhline(threshold, color='red', linestyle='--', label=f'5% 기준선')

    # 각 막대 위에 비율 표기
    for p in ax.patches:
        height = p.get_height()
        percentage = (height / total_count) * 100
        ax.text(p.get_x() + p.get_width() / 2., height + 0.02 * total_count, f'{percentage:.2f}%', ha="center")

    plt.title(f'{column} 변수 분포')
    plt.legend()
    plt.tight_layout()
    plt.show()

# 모든 범주형 변수 시각화
for col in cat_cols:
    plot_categorical_column(df, col)
```

데이터 탐색 및 시각화

수치형 변수 분포 시각화

- 분포의 왜도(skewness) 파악
- 정규화 필요 여부 판단
- 예: 연 소득, 신용 점수는 치우쳐 있음

- 이상치(outliers) 존재 여부 시각적 확인

```
# 📊 Step 3: 수치형 변수 분포 및 타겟 변수 시각화
# 히스토그램
df[num_cols].hist(bins=30, figsize=(12,10))
plt.show()

# 타겟 분포 파이차트
label_prop = df['대출 승인 여부'].value_counts()
plt.pie(label_prop.values, labels=['Rejected (0)', 'Approved (1)'], autopct='%.2f')
plt.title('Target label proportions')
plt.show()

# 박스플롯 (이상치 탐지)
for col in num_cols:
    sns.boxplot(df[col])
    plt.title(f'Boxplot of {col}')
    plt.show()
```

데이터 전처리

정규화

구분	목적	적용 대상	방법
StandardScaler	평균 0, 표준편차 1로 스케일링	왜도 있는 수치형 변수들 (ex. 연 소득, 대출 금액 등)	ss = StandardScaler()
MinMaxScaler	0~1 사이로 압축	이자율 등 비율 변수	mms = MinMaxScaler()

```
# 🧹 Step 4: 데이터 정규화 (표준화 + 정규화)
skewed_cols = ['나이', '연 소득', '근무 경력(년)',
               '대출 금액', '소득 대비 대출금 비율',
               '신용 이력 길이(년)', '신용 점수']
norm_cols= ['대출 이자율']

mms = MinMaxScaler()
ss = StandardScaler()

df[skewed_cols] = ss.fit_transform(df[skewed_cols])
df[norm_cols] = mms.fit_transform(df[norm_cols])
```

데이터 전처리

범주형 변수 인코딩

머신러닝 모델은 문자열 처리를 못하기 때문에 → 숫자형으로 변환 필요

컬럼명	원래 값	매핑
성별	male / female	0 / 1
교육 수준	High School / Master 등	0 ~ 4
대출 목적	PERSONAL, MEDICAL 등	0 ~ 5
과거 대출 연체 여부	No / Yes	0 / 1

```
# 🔢 Step 5: 범주형 변수 수치화 (Label Encoding)
df['교육 수준'].replace({
    'High School': 0, 'Associate': 1, 'Bachelor': 2, 'Master': 3, 'Doctorate': 4
}, inplace=True)

gender_mapping = {'male': 0, 'female': 1}
home_ownership_mapping = {'RENT': 0, 'OWN': 1, 'MORTGAGE': 2, 'OTHER': 3}
loan_intent_mapping = {'PERSONAL': 0, 'EDUCATION': 1, 'MEDICAL': 2, 'VENTURE': 3, 'HOMEIMPROVEMENT': 4, 'DEBTCONSOLIDATION': 5}
previous_loan_defaults_mapping = {'No': 0, 'Yes': 1}

df['성별'] = df['성별'].map(gender_mapping)
df['주택 소유 형태'] = df['주택 소유 형태'].map(home_ownership_mapping)
df['대출 목적'] = df['대출 목적'].map(loan_intent_mapping)
df['과거 대출 연체 여부'] = df['과거 대출 연체 여부'].map(previous_loan_defaults_mapping)
```

데이터 전처리

이상치 제거

- 방법: IQR 기반 우측 극단값 제거
- capping_method='iqr', tail='right'
- 상자그림에서 박스를 벗어나는 오른쪽 극단값을 컷팅
- 대출 승인 여부(타겟)는 제거 대상에서 제외

```python
# 🧹 Step 6: 이상치 제거 (Feature-engine 사용)
from feature_engine.outliers import OutlierTrimmer
trimmer = OutlierTrimmer(
    capping_method='iqr',
    tail='right',
    variables=list(df.columns.drop('대출 승인 여부'))
)
df2 = trimmer.fit_transform(df)

# 상관관계 히트맵
plt.figure(figsize=(15, 8))
sns.heatmap(df2.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

데이터 전처리

학습/검증 데이터셋 분리

- 모델의 일반화 성능을 평가하기 위해 데이터를
학습용(train)과 검증용(test)으로 분리

```
# 🧠 Step 7: 모델 학습 데이터 구성
threshold = 0.1
correlation_matrix = df2.corr()
high_corr_features = correlation_matrix.index[abs(correlation_matrix["대출 승인 여부"]) > threshold].tolist()
high_corr_features.remove("대출 승인 여부")

X_selected = df[high_corr_features]
Y = df["대출 승인 여부"]

X_train, X_test, y_train, y_test = train_test_split(X_selected, Y, test_size=0.2, random_state=42)
```

모델 학습 및 평가

로지스틱 회귀

- 이진 분류에서 가장 기본적인 모델
- 결과는 확률로 출력되며 0.5 기준으로 분류

서포트 벡터 머신 (SVM)

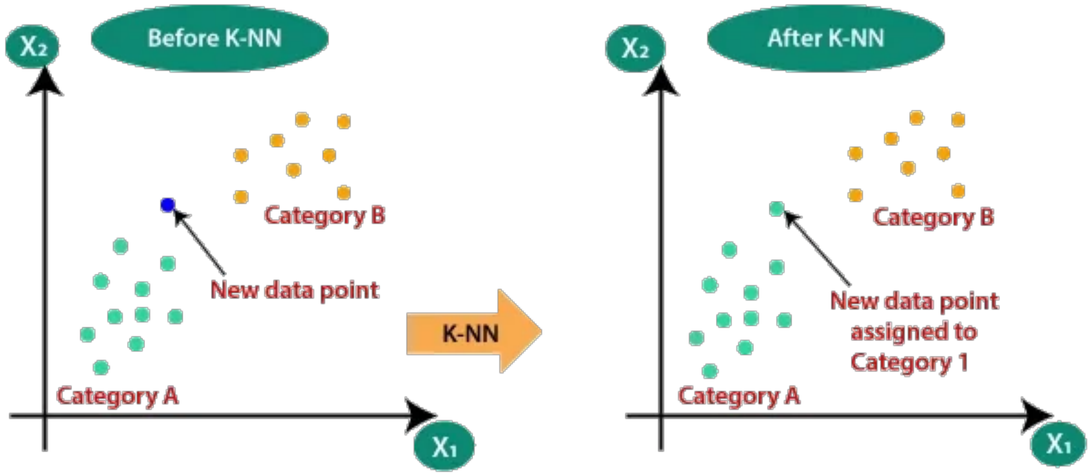
- 고차원에서의 최적의 분리 경계를 찾는 강력한 모델
- 커널을 이용해 선형/비선형 분류 가능

```
# 🤖 Step 8: 모델 학습 및 평가
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train, y_train)
Y_pred = model.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, Y_pred))

model2 = SVC()
model2.fit(X_train, y_train)
Y_pred2 = model2.predict(X_test)
print("SVM Accuracy:", accuracy_score(y_test, Y_pred2))

# Confusion Matrix 시각화
def plot_confusion(cm, title):
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,
                xticklabels=["Predicted 0", "Predicted 1"],
                yticklabels=["Actual 0", "Actual 1"])
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title(title)
    plt.show()

plot_confusion(confusion_matrix(y_test, Y_pred2), "SVM Confusion Matrix")
```

모델 학습 및 평가

K-최근접 이웃 (K-Nearest Neighbor, KNN)

- 새로운 데이터를 예측할 때 가장 가까운 K개의 이웃을 기준으로 결정




```
# 🧠 Step 9: KNN 모델 학습 및 평가
k = 3
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

print(f'KNN Accuracy: {accuracy_score(y_test, y_pred_knn) * 100:.2f}%')
print("\nClassification Report:\n", classification_report(y_test, y_pred_knn))
plot_confusion(confusion_matrix(y_test, y_pred_knn), "KNN Confusion Matrix")
```