Credit Risk Prediction using Machine Learning

Dataset : Credit Risk Dataset (Kaggle)

Tools : Python, Scikit-learn, Logistic Regression, XGBoost, Random Forest, SQLite, Pandas

Github :[view project here]

Irkham Nur Hidayat

WHY THIS ANALYSIS MATTERS

This analysis helps financial institutions minimize risk by identifying potential loan defaulters before approval. By using machine learning models, decisions can be made faster and more accurately, reducing human bias and improving credit risk management strategies.

KEY OBJECTIVE

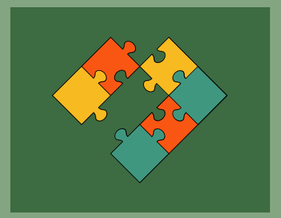
1. preprocess and analyze the credit risk dataset effectively
2. To develop multiple machine learning models for predicting default risk.
3. To compare model performance to determine the most accurate classifier.
4. To assist financial institutions in making informed lending decisions.

DATASET DESCRIPTION

Column	Description
person_age	Age
person_income	Annual income
person_home_ownership	Home ownership
person_emp_length	Employment length (in years)
loan_intent	Loan intent
loan_grade	Loan grade (A,B,C,D,E)

Column	Description
loan_amnt	Loan amount
loan_int_rate	Interest rate
loan_status	Loan status (0 is non default, 1 is deadult)
loan_percent_income	Percent income
cb_person_default_on_file	Historical default
cb_person_cred_hist_length	Credit history length

PREPOCESSING



HANDLING MISSING VALUES

- Drop rows with missing person_emp_length
- Fill missing loan_int_rate with median value per loan_grade

ENCODING CATEGORICAL VARIABLES

Convert categorical variables (e.g. loan_intent, home_ownership) to dummy variables using One-Hot Encoding

FEATURE SCALING

Standardize numerical features such as:

- person_age
- person_income
- loan_amnt
- loan_percent_income
- etc.

HANDLING CLASS IMBALANCE

Apply upsampling to the minority class (loan default = 1) using resample() from sklearn

LOGISTIC REGRESSION

- A simple linear statistical model
- Suitable as a baseline for binary classification (default or not)
- Easy to interpret: coefficients show variable influence
- Accuracy : 0.81%

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pickle

# Train the PD model
model_log = LogisticRegression()
model_log.fit(X_train, y_train)

# Predict
y_pred = model_log.predict(X_test)

# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

RANDOM FOREST

- An ensemble model based on decision trees
- Combines many trees → more stable and accurate predictions
- Robust to overfitting and handles nonlinear patterns well
- Accuracy : 0.91%

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Buat model Random Forest
model_rf = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=42)

# Latih model
model_rf.fit(X_train, y_train)

# Prediksi
y_pred_rf = model_rf.predict(X_test)

# Evaluasi akurasi
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f"Random Forest Accuracy: {accuracy_rf:.2f}")
```

XGBoost

- A fast and efficient gradient boosting model
- Improves previous model weaknesses step-by-step
- Delivers high performance in many data science competitions
- Accuracy : 0.89%

```python
import xgboost as xgb

# Train XGBoost model
model_xg = xgb.XGBClassifier(n_estimators=100, max_depth=3, learning_rate=0.1, use_label_encoder=False)
model_xg.fit(X_train, y_train)

# Predict
y_pred = model_xg.predict(X_test)

# Evaluate
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```