Customer Risk Profiling - Project Report

Prepared using Banking.csv

Project: Risk analysis, EDA, feature engineering, and Random Forest modeling.

Project Summary:

Customer Risk Profiling - Project Summary

Found important columns mapping (examples):

client_id, name, age, location_id, joined_bank, banking_contact, nationality, occupation, fee_structure, loyalty_classification, estimated_income, superannuation_savings, amount_of_credit_cards, credit_card_balance, bank_loans, bank_deposits, checking_accounts, saving_accounts, foreign_currency_account, business_lending, properties_owned, risk_weighting, brid, genderid, iaid, risk_category, loan_to_income, credit_utilization, deposit_to_income, accounts_total

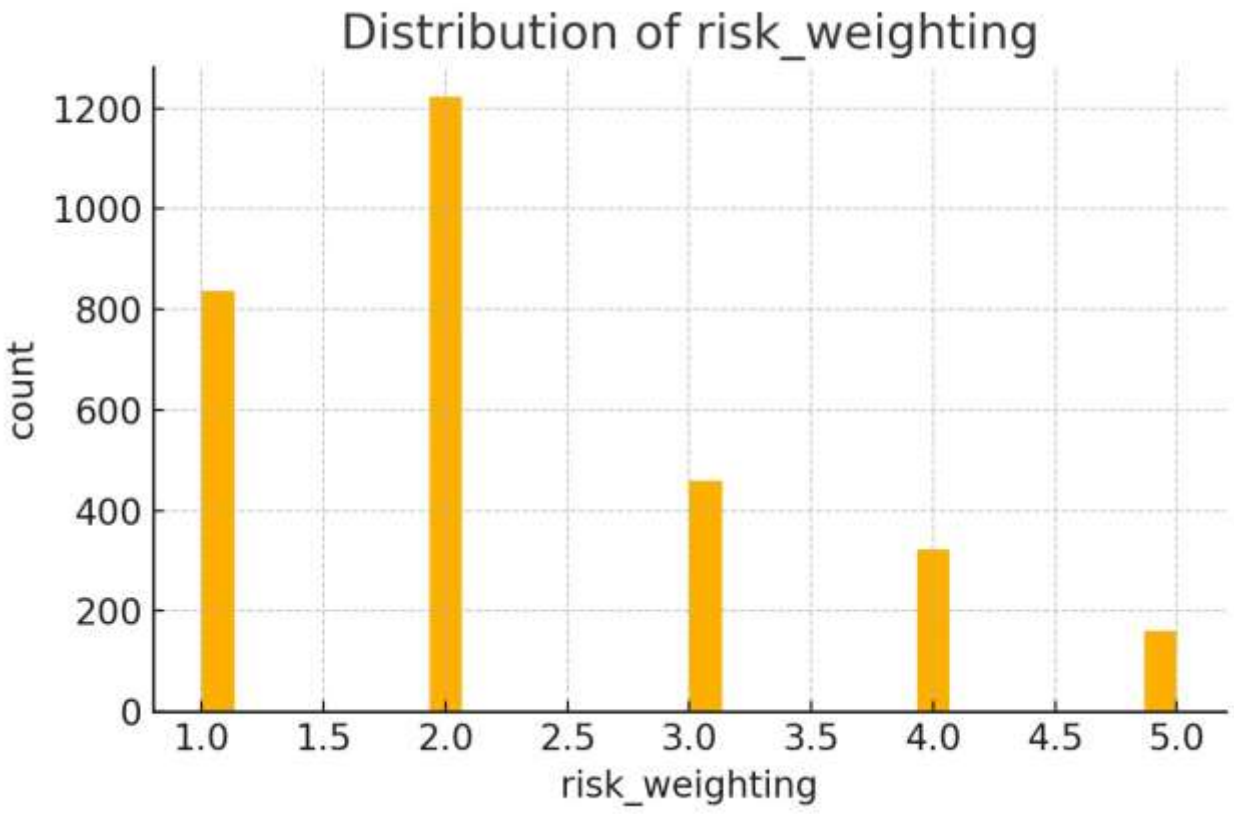
Risk category counts: Low 3000 Medium 0 High 0 Top correlations with risk (numeric): risk_weighting 1.000000 estimated_income 0.664726 superannuation_savings 0.499640 accounts_total 0.434713 bank_loans 0.421824 business_lending 0.417875 foreign_currency_account 0.401872 credit_card_balance 0.399694 checking_accounts 0.373076 credit_utilization 0.355927

Top feature importances (RF): age 0.0 estimated_income 0.0 superannuation_savings 0.0 amount_of_credit_cards 0.0 credit_card_balance 0.0 bank_loans 0.0 bank_deposits 0.0 checking_accounts 0.0 saving_accounts 0.0 foreign_currency_account 0.0

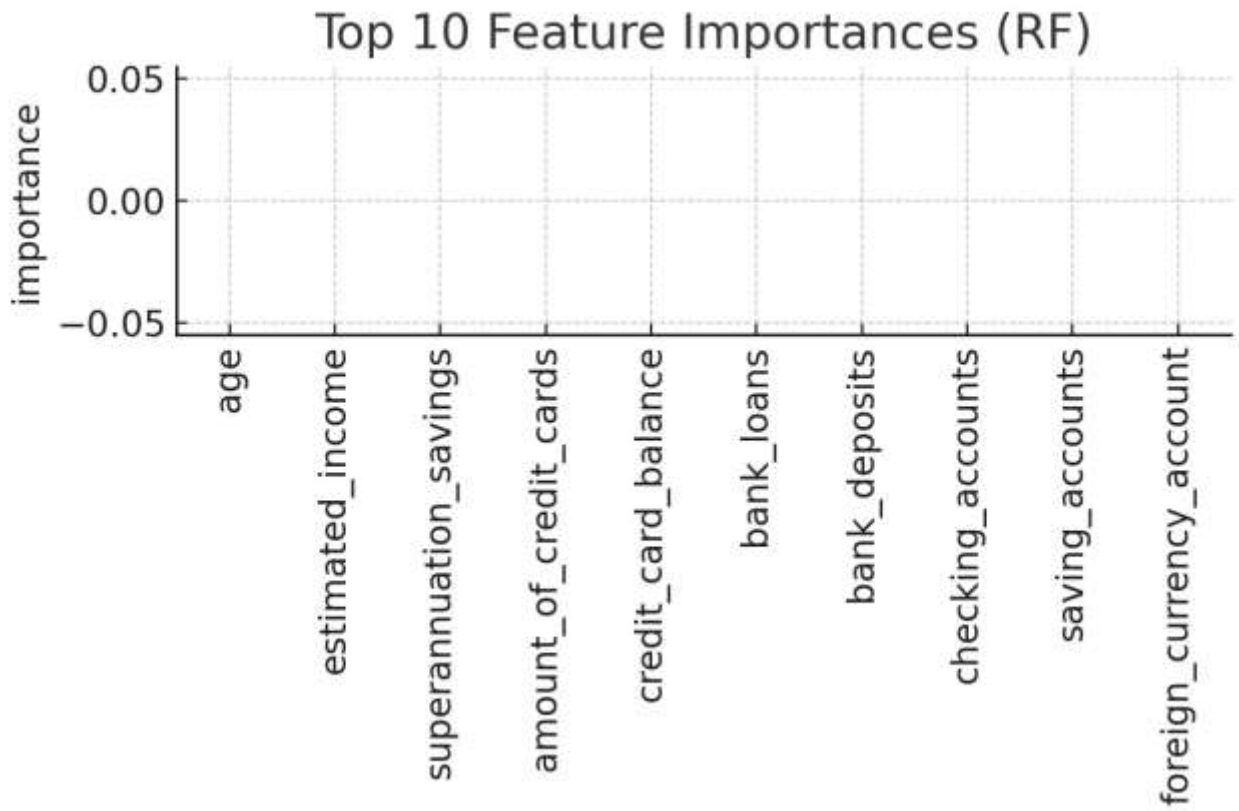
Model ROC AUC: Not available

Confusion matrix: [[600]]

Risk Distribution



Top Feature Importances



Analysis Code (main script)

```
# Customer Risk Profiling - Analysis Script (cleaned version) # Steps: load data,
normalize columns, create risk categories, engineer features, # train Random
Forest to predict high risk, save summary and plots.

import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer from
sklearn.preprocessing import StandardScaler from
sklearn.model_selection import train_test_split from
sklearn.ensemble import RandomForestClassifier

# Load data
df = pd.read_csv('/mnt/data/Banking.csv')

# Normalize column names
df.columns = [c.strip().lower().replace(' ', '_').replace('-', '_') for c in df.columns]

# Identify risk column
possible_risk_cols = [c for c in df.columns if 'risk' in c]
risk_col = 'risk_weighting' if 'risk_weighting' in df.columns else (possible_risk_cols[0] if
possible_risk_cols else None) if risk_col is None:     raise ValueError('No risk-like column
found.')

# Create risk category
df['risk_category'] = pd.cut(df[risk_col], bins=[-np.inf,5,10,np.inf], labels=['Low','Medium','High'])

# Helper function to find columns
def find_col_like(df, keywords):
for k in keywords:
        for c in df.columns:
if k in c:
return c     return None

age_col = find_col_like(df, ['age'])
income_col = find_col_like(df, ['estimated_income','income','estimatedincome'])
super_col = find_col_like(df, ['superannuation'])
amt_cards_col = find_col_like(df, ['amount_of_credit_cards','amount_of_credit'])
credit_bal_col = find_col_like(df, ['credit_card_balance','credit_card'])
bank_loans_col = find_col_like(df, ['bank_loans','bankloan']) bank_deposits_col
= find_col_like(df, ['bank_deposits','bankdeposits']) checking_col =
find_col_like(df, ['checking_accounts','checking']) saving_col =
find_col_like(df, ['saving_accounts','saving'])
foreign_col = find_col_like(df, ['foreign_currency_account','foreign_currency'])

# Feature engineering eps = 1e-6
if bank_loans_col and income_col:
    df['loan_to_income'] = df[bank_loans_col] / (df[income_col] + eps)
if credit_bal_col and amt_cards_col:
    df['credit_utilization'] = df[credit_bal_col] / (df[amt_cards_col] + eps)
if bank_deposits_col and income_col:
    df['deposit_to_income'] = df[bank_deposits_col] / (df[income_col] + eps)
if checking_col and saving_col and foreign_col:     df['accounts_total'] =
df[checking_col] + df[saving_col] + df[foreign_col]

# Prepare target
df['is_high_risk'] = (df['risk_category'] == 'High').astype(int)

# Select features for model
features = [age_col, income_col, super_col, amt_cards_col, credit_bal_col,
bank_loans_col, bank_deposits_col, checking_col, saving_col, foreign_col,
'loan_to_income','credit_utilization','deposit_to_income','accounts_total'] features = [f
for f in features if f and f in df.columns]
```

```
# Impute and scale
imputer = SimpleImputer(strategy='median')
X = imputer.fit_transform(df[features]) y
= df['is_high_risk'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y if
len(np.unique(y))>1 else None) scaler = StandardScaler() X_train = scaler.fit_transform(X_train) X_test
= scaler.transform(X_test)

# Train model
rf = RandomForestClassifier(n_estimators=200, random_state=42, class_weight='balanced' if len(np.unique(y))>1
else None)
rf.fit(X_train, y_train)

# Save feature importances and evaluation as needed
importances = rf.feature_importances_
print('Feature importances:', sorted(zip(features, importances), key=lambda x: -x[1])) - Customer Risk Profiling
Project
```