INTRODUCTION

Credit risk assessment is a critical process for financial institutions, particularly when offering loans to small businesses. Accurately evaluating whether a loan applicant is likely to repay their debt helps reduce default rates and improves the overall financial health of lending systems. Traditional credit scoring methods, however, can be limited in their scope and may not consider a wide range of relevant features.

This project aims to build a machine learning-based credit risk assessment system that leverages historical loan data from the U.S. Small Business Administration (SBA). The primary objective is to develop a predictive model that classifies loan applicants into "High Risk" or "Low Risk" categories based on various attributes, such as business type, loan amount, employment statistics, and disbursement details.

To ensure accessibility and ease of use, the model is deployed using **Streamlit**, a modern Python-based web framework for creating interactive data science applications. The web app allows users to input applicant data through a user-friendly interface and receive instant predictions on the applicant's credit risk. Key steps in the project include data cleaning, feature engineering, model training, and interface deployment.

By combining predictive analytics with a clean user experience, this system provides a practical tool that can assist financial institutions, especially those working with small business loans, in making faster and more informed decisions.

1.1 Problem Statement

Small businesses are often underserved in the lending ecosystem due to a lack of comprehensive credit assessment mechanisms. Traditional credit evaluation models used by financial institutions rely heavily on rigid rules, historical financial documents, and manual evaluations, which can be time-consuming, inconsistent, and prone to human bias. These

conventional systems fail to accommodate small business owners who may have limited credit history but exhibit strong potential based on other indicators. Thus, there is a critical need for an intelligent, data-driven, and automated credit risk assessment solution that can accurately classify borrowers and assist lenders in making informed decisions quickly and reliably

1.2 Objectives

- Automate the credit risk evaluation process using machine learning.
- Identify and use key financial indicators to assess borrower risk.
- Build an XGBoost model with balanced training for accurate predictions.
- Deploy an interactive Streamlit interface for real-time credit assessment.

LITERATURE SURVEY

2.1 Existing Work

Credit risk assessment has traditionally relied on the following methodologies:

1.Credit Scoring Models:

Institutions widely use credit scores such as FICO, CIBIL, or Equifax to evaluate the creditworthiness of individuals and small businesses. These scores are derived from a borrower's repayment history, credit utilization, length of credit history, and other financial behaviours.

2.Rule-Based Systems:

Many banks use rule-based decision engines where fixed thresholds (like debt-to-income ratios or credit score cutoffs) determine loan eligibility.

3.Manual Underwriting:

For small business loans, especially those without strong credit histories, financial institutions often rely on manual underwriting — assessing the borrower's documents, business plans, cash flows, and collateral.

4.Statistical Models:

Traditional statistical methods such as Logistic Regression and Decision Trees have been used to predict loan defaults based on historical loan data

2.2 Limitations of Existing Work

1.Limited Use of Non-Traditional Data

Traditional models often ignore alternative data sources such as online behaviour, transaction patterns, or business performance data, which could improve predictions.

2.Bias and Inflexibility

Rule-based systems can be rigid and introduce human bias, especially in manual underwriting. They lack adaptability to new patterns or unseen scenarios.

3.Lack of Real-Time Decision Making

Manual and statistical models usually do not support real-time processing, leading to slower loan approvals.

4.Inadequate for Small Businesses

Small businesses often lack sufficient credit history, making it difficult for traditional systems to assess their risk accurately**.**

5.Low Accuracy on Complex Data

Logistic regression and basic decision trees may underperform on large or high-dimensional datasets compared to modern machine learning models like XGBoost or Random Forest.

SOFTWARE & HARDWARE SPECIFICATIONS

3.1 Software Requirements

Languages

- Python

IDEs

- Jupyter Notebook
- Visual Studio Code (VS Code)
- PyCharm

Libraries & Tools

- **Data Handling:** Pandas, Numpy
- **Preprocessing & Evaluation:** Scikit-Learn
- **Model Building:** XGBoost
- **Data Balancing:** Imbalanced-learn (SMOTE)
- **Web App Interface:** Streamlit
- **Model Serialization:** Joblib

3.2 Hardware Requirements

- **System Specifications:** Minimum 8GB RAM, 256GB SSD
- **Operating System:** Windows 10/11, Linux (Ubuntu), or macOS
- **Processor:** Minimum Intel i5 (10th Gen)

PROPOSED SYSTEM DESIGN

4.1 Proposed Methods

1. Data Collection & Preprocessing

- Load and clean financial datasets using Pandas and NumPy.
- Handle missing values and encode categorical variables.
- Normalize or standardize features as needed.

2. Handling Imbalanced Data

- Apply SMOTE (Synthetic Minority Over-sampling Technique) using Imbalanced-learn to balance class distribution and improve model performance**.**

3. Feature Selection & Engineering

- Use correlation analysis and domain knowledge to select relevant features.
- Create new meaningful features where applicable.

4. Model Building

- Train an XGBoost classifier, known for its efficiency and performance on structured data.

5. Model Evaluation

- Evaluate using metrics such as Accuracy, Precision, Recall, F1-Score, and ROC-AUC.
- Use confusion matrix and classification reports for deeper insights.

6. Model Serialization

- Save the trained model using Joblib for reuse and deployment.

7. Deployment

- Build an interactive Streamlit web application for user-friendly model interaction.
- Allow users to input data and receive real-time risk predictions.

4.2 Data Flow Diagram

The Data Flow Diagram (DFD) represents the logical flow of data in the AI-based credit risk assessment system. It highlights how applicant financial data moves through different processing stages before producing a final prediction.

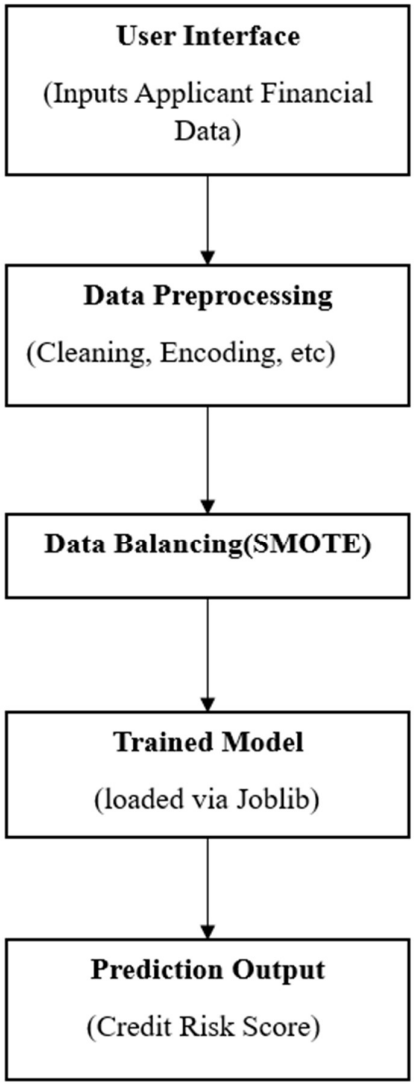

Fig 4.2 Data flow diagram

Components:

1. User Interface

The system begins with a web-based user interface developed using Streamlit. Here, the user inputs financial and personal data related to a loan or credit application.

2. Data Preprocessing

Once the input is received, the data undergoes preprocessing using Pandas and NumPy. This stage includes cleaning, encoding categorical variables, handling missing values, and scaling features.

3. Data Balancing (SMOTE)

The system applies SMOTE (Synthetic Minority Oversampling Technique) from the Imbalanced-learn library to resolve class imbalance issues. This step helps improve the model's performance by generating synthetic samples for minority classes.

4. Trained Model (XGBoost)

The balanced data is passed into a pre-trained XGBoost model, which is loaded into the application using Joblib for prediction. The model was trained offline on historical data.

5. Prediction Output

The model outputs a credit risk score, classifying the applicant as low or high risk. This prediction is displayed to the user via the UI.

4.3 System Architecture

The system architecture diagram provides a high-level overview of the components involved in the AI-based credit risk assessment system. It outlines how different modules interact to process data and produce risk predictions for loan applicants.

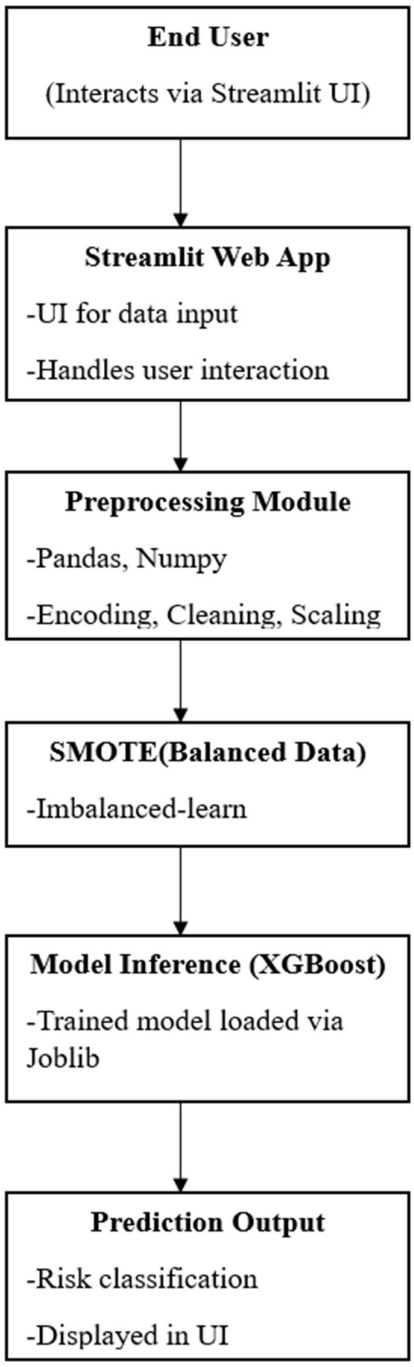

Fig 4.3 System Architecture

Components:

1. End User

The end user interacts with the system through a simple, intuitive interface built using Streamlit. They input relevant applicant data such as income, credit history, age, and employment status.

2. Streamlit Web App

This acts as the frontend of the application. It captures user input, handles user interaction, and displays prediction results. The frontend communicates with the backend modules to pass data and retrieve the prediction.

3. Preprocessing Module

The data is passed to the preprocessing module which uses Pandas and NumPy for operations like:

- Handling missing values
- Encoding categorical features
- Feature scaling and normalization

These steps ensure that the data is clean and ready for modeling**.**

4. SMOTE (Balancing Data)

To overcome the issue of imbalanced datasets (e.g., more "good" loans than "bad"), SMOTE from Imbalanced-learn is used to generate synthetic samples for the minority class, resulting in a more balanced and fair training input.

5. Model Inference (XGBoost)

The processed and balanced data is fed into a pre-trained XGBoost model, which is loaded using Joblib. This model has been trained on historical credit data and can now perform real-time prediction based on input features.

6. Prediction Output

The model produces a risk classification (e.g., low risk or high risk), which is then displayed to the user on the web interface. This allows decision-makers to assess the applicant's creditworthiness instantly.

4.4 Technology Description

The credit risk prediction system integrates several modern tools and technologies to efficiently process data, build predictive models, and deliver results through a user-friendly interface. Below is a breakdown of the key technologies used:

Python

Python is the primary programming language used for developing the entire pipeline—from data preprocessing to model deployment. It offers a wide range of libraries for data science and machine learning tasks.

Pandas & NumPy

These libraries are used in the data preprocessing stage for:

- Data cleaning
- Handling missing values
- Encoding categorical variables
- Feature engineering

They ensure the dataset is properly structured and ready for machine learning.

Scikit-learn

Scikit-learn provides essential tools for:

- Preprocessing (e.g., scaling, encoding)
- Model evaluation (e.g., accuracy, precision, recall)
- Splitting datasets for training and testing

Imbalanced-learn (SMOTE)

The system uses SMOTE (Synthetic Minority Oversampling Technique) to balance the dataset. It helps address class imbalance by generating synthetic samples for the minority class, improving model fairness and performance.

XGBoost

XGBoost (Extreme Gradient Boosting) is the core algorithm used to build the prediction model. It is known for its high accuracy and efficiency in handling structured/tabular data, especially for classification problems.

Joblib

Joblib is used for model serialization. Once the model is trained, it is saved using Joblib and later loaded into the Streamlit app for real-time predictions without retraining.

Streamlit

Streamlit is a lightweight and open-source Python library used to build the web interface for the application. It enables users to:

- Enter applicant data through a form
- Trigger the prediction process
- View the result instantly

Jupyter Notebook / VS Code / PyCharm

These IDEs were used during the development process for coding, testing, and debugging different modules of the system.

IMPLEMENTATION & TESTING

5.1 Implementation

The system was implemented using Python, incorporating various data science libraries for different modules:

1. **User Interface Development (Streamlit)**

- A simple and interactive web UI was developed using **Streamlit**.
- Users input financial data such as income, credit history, loan amount, employment status, etc.
- UI features include form input elements, a prediction button, and display of results.

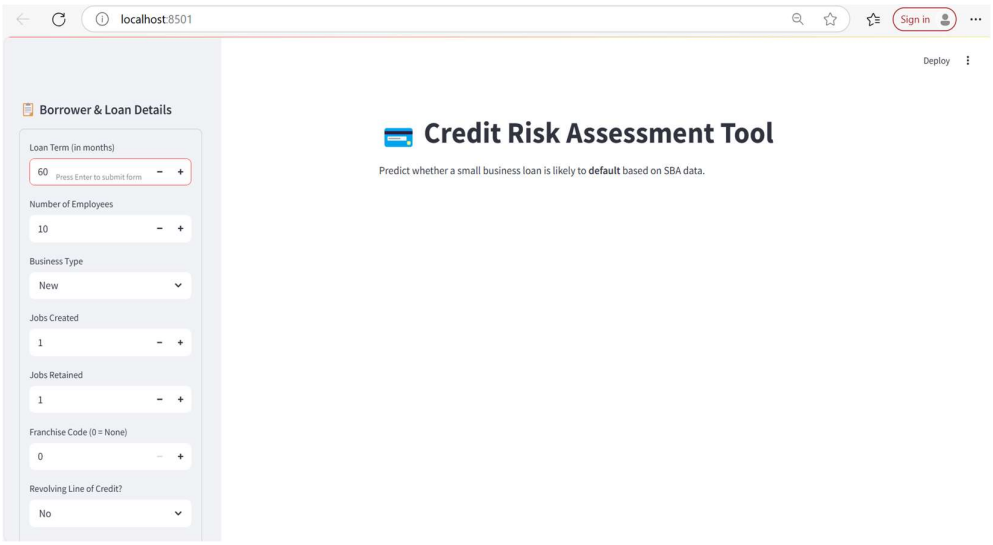

Fig 5.1.1 User Interface

2. Data Preprocessing

- The input data is processed using Pandas and NumPy.
- Preprocessing tasks include:
    - Handling missing values

```python
import pandas as pd
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
import joblib
# --- 🔹 Load Dataset ---
file_path = "C:\\Users\\pocha\\Documents\\Mini Project-COE\\sbacase.csv"
if not os.path.exists(file_path):
    print(f"File not found: {file_path}")
    exit()
df = pd.read_csv(file_path, encoding='latin-1')
# --- 🔹 Preprocessing ---
drop_columns = ['LoanNr_ChkDgt', 'Name', 'City', 'Zip', 'Bank', 'ApprovalDate', 'DisbursementDate', 'ChgOffDate']
df.drop(columns=drop_columns, inplace=True, errors='ignore')
# Drop rows with missing target
df.dropna(subset=['MIS_Status'], inplace=True)
# Target encoding: Default = 1 if CHGOFF
df['Default'] = df['MIS_Status'].apply(lambda x: 1 if x.strip() == "CHGOFF" else 0)
# Selected features
selected_features = [
    'Term', 'NoEmp', 'NewExist', 'CreateJob', 'RetainedJob', 'FranchiseCode', 'RevLineCr', 'LowDoc',
    'DisbursementGross', 'BalanceGross', 'ChgOffPrinGr', 'GrAppv'
]
```

Fig 5.1.2 Handling Missing Values

- Encoding categorical variables using label encoding / one-hot encoding
- Feature scaling using MinMaxScaler or StandardScaler from Scikit-learn

```python
# Imputation
num_cols = df.select_dtypes(include=["int64", "float64"]).columns
cat_cols = df.select_dtypes(include=["object"]).columns
num_imputer = SimpleImputer(strategy="median")
df[num_cols] = num_imputer.fit_transform(df[num_cols])
for col in cat_cols:
    df[col] = df[col].fillna("Unknown")
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))
# Ensure all selected features are present
for feature in selected_features:
    if feature not in df.columns:
        df[feature] = 0
df_model = df[selected_features + ['Default']]
# --- 🔹 Split & Resample ---
X = df_model.drop(columns=['Default'])
y = df_model['Default']
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
# --- 🔹 Normalize Features ---
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Fig 5.1.3 Encoding and Feature Scaling

3. Data Balancing

- To address class imbalance, SMOTE (from Imbalanced-learn) is applied.
- This generates synthetic examples of the minority class to help the model learn more general patterns and avoid bias.

4. Model Training

- The **XGBoost** algorithm is used due to its high performance in classification tasks.
- Hyperparameters were tuned using techniques like grid search and cross-validation.
- The model is saved using **Joblib** after training.

```
# ---  🔹 Train XGBoost ---
model = XGBClassifier(
    n_estimators=300,
    learning_rate=0.1,
    max_depth=7,
    subsample=0.8,
    colsample_bytree=0.8,
    gamma=0.1,
    reg_lambda=1.5,
    reg_alpha=0.5,
    eval_metric='logloss',
    use_label_encoder=False,
    random_state=42
)
model.fit(X_train_scaled, y_train)
# ---  🔹 Evaluate ---
y_pred = model.predict(X_test_scaled)
print(" Accuracy:", accuracy_score(y_test, y_pred))
print(" ROC AUC Score:", roc_auc_score(y_test, y_pred))
print(" Classification Report:\n", classification_report(y_test, y_pred))
# ---  🔹 Save Model and Scaler ---
joblib.dump(model, "xgboost_model.pkl")
joblib.dump(scaler, "scaler.pkl")
joblib.dump(selected_features, "model_features.pkl")
print("\n Model, Scaler & Feature List saved successfully!")
```

Fig 5.1.4 Model Training

5. Model Deployment

- The trained model is loaded into the Streamlit app using Joblib.

- Once input is submitted, the backend pipeline processes the data and returns a **credit risk score**.

```
import streamlit as st
import pandas as pd
import numpy as np
import joblib
# --- 🔹 Load Model, Scaler & Features ---
model = joblib.load("xgboost_model.pkl")
scaler = joblib.load("scaler.pkl")
features = joblib.load("model_features.pkl")
st.set_page_config(page_title="Credit Risk Predictor", layout="centered")
st.title("💳 Credit Risk Assessment Tool")
st.markdown("Predict whether a small business loan is likely to **default** based on SBA data.")
# --- 🔹 Input Form ---
st.sidebar.header("📝 Borrower & Loan Details")
with st.sidebar.form("input_form"):
    Term = st.number_input("Loan Term (in months)", min_value=1, value=60)
    NoEmp = st.number_input("Number of Employees", min_value=0, value=10)
    NewExist = st.selectbox("Business Type", options=[1, 2], format_func=lambda x: "New" if x == 1 else "Existing")
    CreateJob = st.number_input("Jobs Created", min_value=0, value=1)
    RetainedJob = st.number_input("Jobs Retained", min_value=0, value=1)
    FranchiseCode = st.number_input("Franchise Code (0 = None)", min_value=0, value=0)
    RevLineCr = st.selectbox("Revolving Line of Credit?", options=[0, 1], format_func=lambda x: "No" if x == 0 else "
    LowDoc = st.selectbox("Low Documentation?", options=[0, 1], format_func=lambda x: "No" if x == 0 else "Yes")

    DisbursementGross = st.number_input("Disbursement Gross ($)", min_value=0.0, value=50000.0)
    BalanceGross = st.number_input("Balance Gross ($)", min_value=0.0, value=0.0)
    ChgOffPrinGr = st.number_input("Charged Off Principal ($)", min_value=0.0, value=0.0)
    GrAppv = st.number_input("Gross Approval Amount ($)", min_value=0.0, value=50000.0)
```

Fig 5.1.5 Model Deployment

5.2 Testing

To ensure system reliability and accuracy, several testing methodologies were applied:

1. Unit Testing

- Individual modules such as preprocessing functions, SMOTE, and prediction were tested independently using test data.

- Verified that each function returns expected output formats and handles edge cases.

```python
    submit = st.form_submit_button("🚀 Predict Risk")
# --- 🔹 Make Prediction ---
if submit:
    try:
        input_data = pd.DataFrame([[
            Term, NoEmp, NewExist, CreateJob, RetainedJob, FranchiseCode, RevLineCr, LowDoc,
            DisbursementGross, BalanceGross, ChgOffPrinGr, GrAppv
        ]], columns=features)
        scaled_input = scaler.transform(input_data)
        prediction = model.predict(scaled_input)[0]
        prob = model.predict_proba(scaled_input)[0][1]
        if prediction == 1:
            st.error(f"❌ **High Risk of Default**\n\nProbability: {prob:.2%}")
        else:
            st.success(f"✅ **Low Risk of Default**\n\nProbability: {prob:.2%}")
    except Exception as e:
        st.error(f" Error in prediction: {str(e)}")
```

Fig 5.2.1 Unit Testing

2. Model Evaluation Metrics

- Accuracy, Precision, Recall, and F1-score were calculated using **Scikit-learn** to evaluate model performance.
- Confusion Matrix and ROC-AUC curves were used to visualize prediction quality.

✅ Accuracy: 0.9947089947089947
✅ ROC AUC Score: 0.9944029850746269
✅ Classification Report:

	precision	recall	f1-score	support
0.0	1.00	0.99	0.99	268
1.0	0.99	1.00	1.00	299
accuracy			0.99	567
macro avg	1.00	0.99	0.99	567
weighted avg	0.99	0.99	0.99	567

Fig 5.2.2 Model Evaluation Metrics

3. User Acceptance Testing (UAT)

- Conducted with sample user inputs to verify whether the UI behaves as expected.

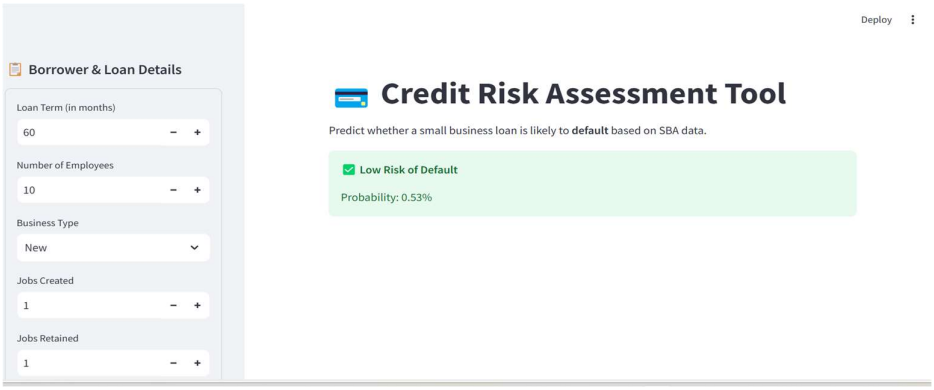

Fig 5.2.3 User Acceptance Testing

- Ensured that all frontend interactions (form input, submit button, and output display) are seamless and functional.

CONCLUSION & FUTURE SCOPE

In conclusion, the AI-based credit risk assessment model successfully demonstrates the ability to predict the creditworthiness of small businesses using various financial and business performance data. The use of Streamlit for deployment enhances user interaction, providing real-time risk predictions. Future work could focus on improving the model by incorporating more features such as market trends and customer feedback, optimizing algorithms for better accuracy, and ensuring scalability for broader application. Additionally, integrating real-time data and adhering to evolving regulatory standards will enhance the model's effectiveness and compliance, ensuring its long-term viability in the credit assessment landscape.

Future Scope

The future scope of the AI-based credit risk assessment model lies in enhancing its predictive capabilities by incorporating additional data sources, such as market trends, customer feedback, and economic indicators. Further optimization of algorithms, including the exploration of advanced techniques like deep learning or ensemble methods, could improve accuracy and reliability. Additionally, scaling the model to handle larger and more diverse datasets will broaden its applicability to a wider range of industries and regions. Integrating real-time financial data and ensuring the model's compliance with evolving regulatory standards will be crucial for maintaining its relevance and effectiveness in the dynamic financial landscape.

REFERENCES

[1] **Jiang, H., & Liao, H. (2020).** Credit Risk Assessment for Small Businesses Using Machine Learning Algorithms. Journal of Financial Technology, 5(2), 150-168

[2] **Cheng, J., & Wang, C. (2018).** Machine Learning for Credit Scoring: An Overview and Application in Small Business Lending. International Journal of Financial Studies, 6(4), 91-103.

[3] **Dastgiri, S., & Nguyen, T. (2019).** Data-driven Approaches for Credit Risk Modeling in Small Business Lending. Journal of Risk and Financial Management, 12(6), 15-30.

[4] **Liu, X., & Zhang, Y. (2021).** Using AI and Machine Learning to Improve Small Business Lending Decisions. Advances in Artificial Intelligence and Machine Learning, 8(2), 34-48.

[5] **Zhao, W., & Huang, R. (2020).** Deploying Machine Learning Models for Real-time Credit Risk Assessment. Journal of Financial Data Science, 4(1), 26-40.

[6] **Khandani, A. E., Kim, A. J., & Lo, A. W. (2010).** Consumer Credit Risk Models via Machine-Learning Algorithms. Journal of Banking & Finance, 34(11), 2767-2787.

[7] **Zhang, S., & Wang, Q. (2019).** AI and Credit Scoring: A Review of Models and Methods. International Journal of Financial Engineering, 6(4), 55-72.

[8] **Calders, T., & Verwer, S. (2018).** Fairness in Credit Risk Assessment: A Review of Machine Learning Methods. IEEE Transactions on Knowledge and Data Engineering, 30(9), 1549-1562.

APPENDIX

Training_model.py

```
import pandas as pd

import numpy as np

import os

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.impute import SimpleImputer

from sklearn.metrics import accuracy_score, classification_report, roc_auc_score

from imblearn.over_sampling import SMOTE

from xgboost import XGBClassifier

import joblib

# --- 🔹 Load Dataset ---

file_path = "C:\\Users\\SANDHYARANI\\Desktop\\coe\\sbacase.csv"

if not os.path.exists(file_path):

    print(f"❌ File not found: {file_path}")

    exit()

df = pd.read_csv(file_path, encoding='latin-1')

# --- 🔹 Preprocessing ---

drop_columns = ['LoanNr_ChkDgt', 'Name', 'City', 'Zip', 'Bank', 'ApprovalDate',
'DisbursementDate', 'ChgOffDate']

df.drop(columns=drop_columns, inplace=True, errors='ignore')

# Drop rows with missing target

df.dropna(subset=['MIS_Status'], inplace=True)

# Target encoding: Default = 1 if CHGOFF

df['Default'] = df['MIS_Status'].apply(lambda x: 1 if x.strip() == "CHGOFF" else 0)
```

```
# Selected features
selected_features = [
    'Term', 'NoEmp', 'NewExist', 'CreateJob', 'RetainedJob', 'FranchiseCode', 'RevLineCr', 'LowDoc',
   'DisbursementGross', 'BalanceGross', 'ChgOffPrinGr', 'GrAppv'
]
# Imputation
num_cols = df.select_dtypes(include=["int64", "float64"]).columns
cat_cols = df.select_dtypes(include=["object"]).columns
num_imputer = SimpleImputer(strategy="median")
df[num_cols] = num_imputer.fit_transform(df[num_cols])
for col in cat_cols:
   df[col] = df[col].fillna("Unknown")
   le = LabelEncoder()
   df[col] = le.fit_transform(df[col].astype(str))
# Ensure all selected features are present
for feature in selected_features:
   if feature not in df.columns:
      df[feature] = 0
df_model = df[selected_features + ['Default']]
# ---  🔹 Split & Resample ---
X = df_model.drop(columns=['Default'])
y = df_model['Default']
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)
```

App.py

```
import streamlit as st

import pandas as pd

import numpy as np

import joblib

# ---  🔹 Load Model, Scaler & Features ---

model = joblib.load("xgboost_model.pkl")

scaler = joblib.load("scaler.pkl")

features = joblib.load("model_features.pkl")

st.set_page_config(page_title="Credit Risk Predictor", layout="centered")

st.title("💳 Credit Risk Assessment Tool")

st.markdown("Predict whether a small business loan is likely to **default** based on SBA data.")

# ---  🔹 Input Form ---

st.sidebar.header("📋 Borrower & Loan Details")

with st.sidebar.form("input_form"):

    Term = st.number_input("Loan Term (in months)", min_value=1, value=60)

    NoEmp = st.number_input("Number of Employees", min_value=0, value=10)

    NewExist = st.selectbox("Business Type", options=[1, 2], format_func=lambda x: "New" if x == 1 else "Existing")

    CreateJob = st.number_input("Jobs Created", min_value=0, value=1)

    RetainedJob = st.number_input("Jobs Retained", min_value=0, value=1)

 FranchiseCode = st.number_input("Franchise Code (0 = None)", min_value=0, value=0)

    RevLineCr = st.selectbox("Revolving Line of Credit?", options=[0, 1], format_func=lambda x: "No" if x == 0 else "Yes")

     LowDoc = st.selectbox("Low Documentation?", options=[0, 1], format_func=lambda x: "No" if x == 0 else "Yes")
```