National College of Ireland

Configuration Manual

MSc Research Project: Enhancing Credit Risk Assessment using Interpretable Machine Learning Techniques
Programme Name: MSc in Artificial Intelligence for Business

Mohsin Sadaqat Ali
Student ID: X23397071

School of Computing
National College of Ireland

Supervisor: Mr Syed Muslim Jameel

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:……Mohsin Sadaqat Ali.……

Student ID: ……X23397071……

Programme: ……Msc Artificial Intelligence for Business **Year:** ……2024-25…

Module: ……Practicum II……

Lecturer: ……Mr.Mohit Garg……

Submission Due Date: ……11-Aug-2025……

Project Title: ……Enhancing Credit Risk Assessment using Interpretable Machine Learning Techniques

Word Count: ……553…… **Page Count:** ……8……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Mohsin Sadaqat Ali……

Date: ……10-Aug-2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Mohsin Sadaqat Ali
Student ID: X23397071

1 System Configuration

I have implemented this project using **Jupyter Notebook** hosted on **Google Colab**, providing an isolated cloud-based environment preconfigured with popular data science packages.

- **Platform:** Google Colaboratory
- **OS:** Linux (Google Cloud-backed virtual machine)
- **Python Version:** 3.10+
- **RAM:** ~12 GB
- **Hardware Accelerator:** My laptop having 8 gb RAM and 1 TB Hardisk

2 Software Requirement

Library	Version	Purpose
pandas	>=1.3	Data wrangling and manipulation
numpy	>=1.21	Numerical operations
matplotlib	>=3.4	Plotting graphs and charts
seaborn	>=0.11	Enhanced visualization
sklearn	>=1.0	Preprocessing, modeling, and evaluation metrics
xgboost	>=1.4	Gradient Boosted Trees for prediction
lightgbm	>=3.2	Efficient gradient boosting algorithm

3 Code Artefact

The Jupyter Notebook is logically structured and consists of the following stages:

1. **Data Collection and Cleaning:**
    - Credit risk dataset is loaded from a CSV file.
    - Missing values are handled.
    - Categorical variables are encoded using techniques like label encoding and one-hot encoding.

```
# Install necessary libraries for data processing, modeling, and visualization
!pip install imblearn
!pip install xgboost
!pip install catboost
!pip install lightgbm
!pip install scorecardpy
```

Show hidden output

```
# Import required libraries for data manipulation, visualization, statistical modeling, and machine learning algorithms
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels
import re
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
import lightgbm as lgb
from lightgbm import LGBMClassifier
from sklearn.metrics import mean_squared_error
```

```
# Set pandas display options to show all columns and full column width
pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', None)
```

```
# Load the training dataset from a CSV file and display the first few rows
train = pd.read_csv('./AmExpressCodeLab/train.csv')
train.head()
```

	customer_id	name	age	gender	owns_car	owns_house	no_of_children	net_yearly_income	no_of_days_employed	occupation_type	total_family_members	migrant_worker	yearly_debt_payments	credit_limit
0	CST_115179	ita Bose	46	F	N	Y	0.0	107934.04	612.0	Unknown	1.0	1.0	33070.28	18690.93
1	CST_121920	Alper Jonathan	29	M	N	Y	0.0	109862.62	2771.0	Laborers	2.0	0.0	15329.53	37745.19
2	CST_109330	Umesh Desai	37	M	N	Y	0.0	230153.17	204.0	Laborers	2.0	0.0	48416.60	41598.36
3	CST_128288	Rie	39	F	N	Y	0.0	122325.82	11941.0	Core staff	2.0	0.0	22574.36	32627.76
4	CST_151355	McCool	46	M	Y	Y	0.0	387286.00	1459.0	Core staff	1.0	0.0	38282.95	52950.64

2. **Exploratory Data Analysis (EDA):**
   - Visualizations for correlation heatmaps and distribution of credit status.
   - Trend identification among features like loan amount, income, credit history, etc.

```
[ ] # Print the shape (number of rows and columns) of the sampled DataFrame
    print(df.shape)
```

```
(30000, 19)
```

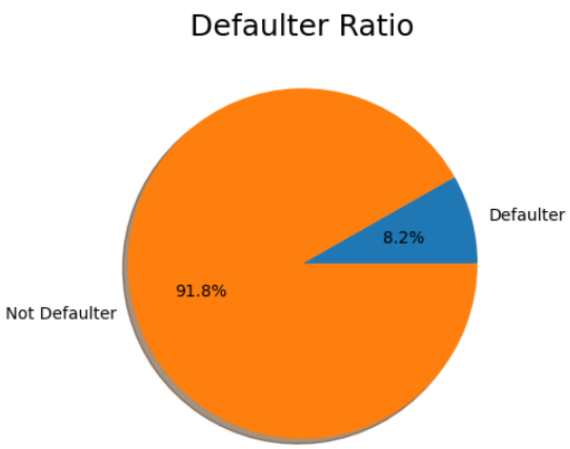
```
[ ] # Display concise information about the DataFrame, including column data types and non-null values
    df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30000 entries, 34126 to 33325
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   customer_id             30000 non-null  object
 1   name                    30000 non-null  object
 2   age                     30000 non-null  int64
 3   gender                  30000 non-null  object
 4   owns_car                29631 non-null  object
 5   owns_house              30000 non-null  object
 6   no_of_children          29498 non-null  float64
 7   net_yearly_income       30000 non-null  float64
 8   no_of_days_employed     29685 non-null  float64
 9   occupation_type         30000 non-null  object
 10  total_family_members    29944 non-null  float64
 11  migrant_worker          29947 non-null  float64
 12  yearly_debt_payments    29938 non-null  float64
 13  credit_limit            30000 non-null  float64
 14  credit_limit_used(%)    30000 non-null  int64
 15  credit_score            29994 non-null  float64
 16  prev_defaults           30000 non-null  int64
 17  default_in_last_6months 30000 non-null  int64
 18  credit_card_default     30000 non-null  int64
dtypes: float64(8), int64(5), object(6)
memory usage: 4.6+ MB
```

3. **Feature Engineering:**
   - Creation of new attributes from existing ones (e.g., debt-to-income ratio).
   - Encoding techniques for nominal and ordinal variables.

```
[ ] # Visualize the distribution of credit card default using a pie chart
    total_default_values = df.credit_card_default.value_counts(ascending=True).values

    plt.figure()
    plt.pie(total_default_values, labels=['Defaulter', 'Not Defaulter'], autopct='%1.1f%%', shadow=True)
    plt.title('Defaulter Ratio', fontsize=18)
    plt.show()
```



Encoding

```
# Encode the 'gender' column using binary encoding (1 for 'F', 0 for 'M')
df["gender"] = df["gender"].apply(lambda x : 1 if x == "F" else 0)
# The following lines are commented out, indicating binary encoding was considered but not applied to 'owns_car' and 'owns_house'
#df["owns_car"] = df["owns_car"].apply(lambda x : 1 if x == "Y" else 0)
#df["owns_house"] = df["owns_house"].apply(lambda x : 1 if x == "Y" else 0)
```

```
# Apply one-hot encoding to the 'occupation_type' column and concatenate the resulting dummy variables to the DataFrame
# Drop the original 'occupation_type' column and display the first few rows of the transformed DataFrame
occupation_type_encoded_df = pd.get_dummies(df["occupation_type"], prefix= "occupation")
df = pd.concat([df, occupation_type_encoded_df], axis=1)
df.drop(columns=["occupation_type"], inplace=True)
df.head()
```

	gender	no_of_days_employed	credit_limit_used(%)	credit_score	default_in_last_6months	credit_card_default	occupation_Accountants	occupation_Cleaning staff	occupation_Cooking staff	occupation_Core staff	occupat
34126	1	4181.0	23	807.0	0	0	0	0	0	0	
44609	1	365250.0	60	858.0	0	0	0	0	0	0	
44264	1	365241.0	81	667.0	0	1	0	0	0	0	
33556	1	1882.0	32	690.0	0	0	0	0	0	0	
28050	0	365244.0	65	721.0	0	0	0	0	0	0	

Data Imbalancing Handling

```
# Apply the Synthetic Minority Over-sampling Technique (SMOTE) to the training data to address class imbalance
# This creates synthetic samples of the minority class to balance the dataset
sm = SMOTE(random_state=42)
X_train,y_train = sm.fit_resample(X_train,y_train)
```

```
# Print the dimensions (shape) of the training feature and target sets after applying SMOTE
print("Dimension of X_train_sm Shape:", X_train.shape)
print("Dimension of y_train_sm Shape:", y_train.shape)
```

```
Dimension of X_train_sm Shape: (38552, 24)
Dimension of y_train_sm Shape: (38552,)
```

```
# Convert the SMOTE-resampled training data back into a pandas DataFrame and Series
# Display the shape of the resulting DataFrames and the first few rows of the training features
X_train = pd.DataFrame(data=X_train, columns=cols)
print("X_train.shape:",X_train.shape)
y_train = pd.Series(y_train)
print("y_train.shape:",y_train.shape)
X_train.head()
```

```
X_train.shape: (38552, 24)
y_train.shape: (38552,)
```

4. **Model Training:**
   - Algorithms implemented:
     - Logistic Regression
     - Random Forest
     - XGBoost
     - LightGBM
   - Data split using train_test_split (typically 70/30 or 80/20).

5. **Evaluation Metrics:**
   - **Accuracy**
   - **Precision**
   - **Recall**
   - **F1 Score**
   - **Confusion Matrix**
   - **ROC-AUC Score**

```
[ ] # Initialize and train a Logistic Regression model on the scaled and balanced training data
    logreg = LogisticRegression()
    logreg.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
[ ] # Evaluate the trained Logistic Regression model using the custom evaluation function
    evaluation(logreg)
```

```
==================================================
[[18241  1035]
 [  617 18659]]
              precision    recall  f1-score   support

           0       0.97      0.95      0.96     19276
           1       0.95      0.97      0.96     19276

    accuracy                           0.96     38552
   macro avg       0.96      0.96      0.96     38552
weighted avg       0.96      0.96      0.96     38552

Accuracy of TRAIN data: 95.71487860551981
F1_Score of TRAIN data: 95.7143747883949
==================================================
[[7799  462]
 [  20  719]]
```

Random Forest

```
[ ] # Initialize and train a Random Forest Classifier model on the scaled and balanced training data
    regr_rfr = RandomForestClassifier(random_state=42, oob_score=True)
    regr_rfr.fit(X_train, y_train)
```

```
RandomForestClassifier(oob_score=True, random_state=42)
```

```
[ ] # Evaluate the trained Random Forest Classifier model using the custom evaluation function
    evaluation(regr_rfr)
```

```
==================================================
[[19276     0]
 [    0 19276]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     19276
           1       1.00      1.00      1.00     19276

    accuracy                           1.00     38552
   macro avg       1.00      1.00      1.00     38552
weighted avg       1.00      1.00      1.00     38552

Accuracy of TRAIN data: 100.0
F1_Score of TRAIN data: 100.0
==================================================
[[8016  245]
 [  70  669]]
```

6. **Results and Interpretations:**
   - Visual comparison of model performance.
   - Feature importance plots for tree-based models.

```
# Plot the calculated error rates against the corresponding k values to visualize the optimal k
plt.figure(figsize=[10,6])
plt.plot(range(1,40), error_rate, color='blue', linestyle='dashed',
         marker='o', markerfacecolor='red', markersize=10)
plt.title('Error rate vs K')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

```
Text(0, 0.5, 'Error Rate')
```