National College of Ireland

Configuration Manual

MSc Research Project
MSc Fintech

Allan Ng'ang'a
Student ID: X23329696

School of Computing
National College of Ireland

Supervisor: Faithful Onwuegbuche

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Allan Arthur Ng'ang'a		
Student ID:	X23329696		
Programme:	MSCFTD	**Year:**	2025
Module:	MSc (Research) Practicum		
Supervisor:	Faithful Onwuegbuche		
Submission Due Date:	15th September 2025		
Project Title:	AI-Driven Credit Scoring Using Alternative Data: Unlocking Financial Access In Emerging Markets		
Word Count:	5571	**Page Count** 19 pages	

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: Allan Arthur Ng'ang'a

Date: 15th September 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

Allan Ng'ang'a
Student ID: x23329696

1 Introduction

This configuration manual provides step-by-step instructions for replicating the AI-driven credit scoring pipeline developed for the MSc Fintech research project titled "**AI-Driven Credit Scoring Using Alternative Data: Unlocking Financial Access in Emerging Markets**."
The project uses the 2024 FinAccess Household Survey from Kenya, a nationally representative dataset, to train machine learning models that predict creditworthiness based on alternative data sources such as mobile money usage, savings behavior, and demographic characteristics.
The goal is to create a reproducible, explainable, and fairness-aware pipeline that can help policymakers, fintech companies, and regulators design more inclusive credit scoring systems for underserved populations.

2 System Requirements

Hardware:

- Processor: Intel i5 (8th Gen) or equivalent (minimum)
- RAM: 8 GB minimum (16 GB recommended)
- Storage: 10 GB free space
- GPU: Optional (for faster training in Google Colab)

Software:

- Operating System: Windows 10/11, macOS 12+, or Ubuntu 20.04+
- Python: Version 3.9+
- Jupyter Notebook or **Google Colab**

Python Libraries:
Install required dependencies:
pip install pandas numpy scikit-learn imbalanced-learn shap xgboost lightgbm matplotlib seaborn joblib

3 Section 3

Dataset:

- **Name:** FinAccess Household Survey 2024
- **Source:** FinAccess (Central Bank of Kenya, Kenya National Bureau of Statistics, FSD Kenya)
- **Format:** CSV file

Steps to Load Data in Colab:

```python
import pandas as pd

# Loading main dataset
data = pd.read_csv("2024_Finaccess Public data.csv")


# Dimensions and data types
print(data.shape)
print(data.dtypes)

# Preview data
data.head()
```

Preprocessing Steps:

1. Handle missing values:

```python
# Calculate percentage of missing values per column
missing_percent = data.isnull().mean() * 100

# Identify columns with more than 80% missing
cols_to_drop = missing_percent[missing_percent > 80].index

# Drop those columns
data = data.drop(columns=cols_to_drop)

# print dropped columns
print(" Dropped columns with more than 80% missing values:")
print(list(cols_to_drop))
```

2. Feature selection

PreProcessing & Feature Selection

```python
[ ] demographic_columns_to_keep = [
        'agegroup_of_selected_individual', # Using agegroup as age_of_respondent was dropped earlier
        'sex_of_selected_individual',
        'a20_education_completed', # Using a20_education_completed instead of education_level_of_respodent
        'a21_marital_status', # Using a21_marital_status instead of marital_status_of_respondent
        'a8_cluster_type', # Urban/Rural - highly relevant for emerging markets context
        'disability_status_indicator',
        'a19_relation_to_hh_head' # Can provide context about financial dependency
    ]
```

Target to be in binary (0 – 1)

```python
[ ] # Convert the target variable to numerical (1 for 'Yes', 0 for 'No', NaN for others)
    dataset_filtered['e1_have_you_applied_for_and_been_denied_a_loan_in_the_last_12_months_numeric'] = dataset_filtered['e1_have_you_applied

    # Mean target score by gender
```

4 Model Configuration Requirements

Models Implemented:

- Logistic Regression (class weights balanced)
- Random Forest

- XGBoost
- LightGBM

Pipeline Steps:

Pre-SMOTE

```
[ ]  from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.metrics import (
         accuracy_score, roc_auc_score, f1_score, precision_score, recall_score,
         confusion_matrix, classification_report
     )
     from sklearn.calibration import CalibrationDisplay
     from sklearn.linear_model import LogisticRegression
     from sklearn.ensemble import RandomForestClassifier
     from xgboost import XGBClassifier
     from lightgbm import LGBMClassifier
     import matplotlib.pyplot as plt
     import re
     import numpy as np
     import pandas as pd

     # Clean column names
     X.columns = ["".join(c if c.isalnum() else "_" for c in str(x)) for x in X.columns]
     X.columns = [re.sub(r'_+', '_', col).strip('_') for col in X.columns]

     # Split data
     X_train, X_test, y_train, y_test = train_test_split(
         X, y, test_size=0.25, random_state=42, stratify=y
     )

     # Scale data
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)

     # Define models
     models = {
         "Logistic Regression (Pre-SMOTE)": LogisticRegression(
             max_iter=1000, class_weight='balanced', random_state=42),
         "Random Forest (Pre-SMOTE)": RandomForestClassifier(
             n_estimators=200, class_weight='balanced_subsample', random_state=42),
         "XGBoost (Pre-SMOTE)": XGBClassifier(
             scale_pos_weight=(len(y_train) - sum(y_train)) / sum(y_train),
             eval_metric='logloss', random_state=42),
         "LightGBM (Pre-SMOTE)": LGBMClassifier(class_weight='balanced', random_state=42)
     }
```

Model Evaluation:

```python
# Evaluate models
results = []
for name, model in models.items():
    print(f"\n{'='*60}\n🔍 Evaluating: {name} \n{'='*60}")

    if name == "Logistic Regression":
        X_tr, X_te = X_train_scaled, X_test_scaled
    else:
        X_tr, X_te = X_train, X_test

    model.fit(X_tr, y_train)
    y_pred = model.predict(X_te)
    y_proba = model.predict_proba(X_te)[:, 1] if hasattr(model, "predict_proba") else y_pred

    # Add all consistent metrics
    results.append({
        'Model': name,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred, zero_division=0),
        'Recall': recall_score(y_test, y_pred, zero_division=0),
        'F1 Score': f1_score(y_test, y_pred, zero_division=0),
        'ROC-AUC': roc_auc_score(y_test, y_proba),
        'Confusion Matrix': confusion_matrix(y_test, y_pred)
    })

    # Print Confusion Matrix & Report
    print("\n📊 Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))

    print("\n📋 Classification Report:")
    print(classification_report(y_test, y_pred, digits=3))

    # Plot Calibration Curve
    if hasattr(model, "predict_proba"):
        CalibrationDisplay.from_estimator(model, X_te, y_test, n_bins=10, name=name)
        plt.title(f"Calibration Curve: {name} ")
        plt.grid(True)
        plt.show()

# Convert results into DataFrame
pre_df = pd.DataFrame(results).set_index('Model')

print("\n\n🌟 Pre-SMOTE Model Metrics:")
display(pre_df.round(4))
```

5. SHAPE Explainability

```python
# ✅ SHAP Explainability for Logistic Regression (Post-SMOTE)
import shap
import pandas as pd
import numpy as np

# Create SHAP explainer using scaled training data
final_logreg = post_models["XGBoost"]

# SHAP expects a DataFrame for feature names
X_train_shap = pd.DataFrame(X_resampled_unscaled, columns=X.columns) # Use unscaled data
X_test_shap = pd.DataFrame(X_test, columns=X.columns) # Use unscaled data

# Convert boolean columns to int
for col in X_train_shap.columns:
    if X_train_shap[col].dtype == bool:
        X_train_shap[col] = X_train_shap[col].astype(int)
for col in X_test_shap.columns:
    if X_test_shap[col].dtype == bool:
        X_test_shap[col] = X_test_shap[col].astype(int)


# Create SHAP explainer for linear model
explainer = shap.Explainer(final_logreg, X_train_shap)

# Compute SHAP values for the test set
shap_values = explainer(X_test_shap)

# Beeswarm plot - global feature impact
shap.plots.beeswarm(shap_values)

# bar plot - average absolute impact per feature
shap.plots.bar(shap_values)

# Local explanation (e.g., for first prediction)
shap.plots.waterfall(shap_values[0])
```

6.Fairness Audit

Fairness

```
[ ]  # Use the already trained Logistic Regression model
     fairness_model = post_models["XGBoost"]

     from sklearn.metrics import classification_report

     # Columns to audit
     group_columns = ['sex_of_selected_individual', 'agegroup_of_selected_individual', 'a8_cluster_type']

     # Get the original indices that correspond to the test set
     test_indices = X_test.index

     # Evaluate for each group
     for group_col in group_columns:
         print(f"\n\n🔍 Fairness Audit by: {group_col}")

         for group_value in dataset_cleaned[group_col].dropna().unique():
             # Get the indices from the original dataset for the current group
             group_indices_original = dataset_cleaned[dataset_cleaned[group_col] == group_value].index

             # Find the intersection of these indices with the test set indices
             group_test_indices = test_indices.intersection(group_indices_original)

             if len(group_test_indices) == 0:
                 print(f"⚠️ No samples for group: {group_value}")
                 continue

             # Get the positions of these indices in the X_test array
             group_positions_in_test = [X_test.index.get_loc(i) for i in group_test_indices]

             # Subset X and y using these positions
             X_group = X_test_scaled[group_positions_in_test]
             y_group = y_test.loc[group_test_indices]

             # Predict
             y_pred_group = fairness_model.predict(X_group)

             # Output report
             print(f"\n📊 Group: {group_value} ({len(group_test_indices)} samples)")
             print(classification_report(y_group, y_pred_group, digits=3))
```

Show hidden output

7. Saving & Loading the Pipeline

```
[ ]  from sklearn.pipeline import Pipeline
     from sklearn.preprocessing import StandardScaler
     from sklearn.linear_model import LogisticRegression
     import joblib

     # Create pipeline (scaler + logistic regression)
     final_pipeline = Pipeline([
         ('scaler', StandardScaler()),  # fits on unscaled X
         ('logreg', LogisticRegression(max_iter=1000, class_weight='balanced', random_state=42))
     ])


     final_pipeline.fit(X_resampled_unscaled, y_resampled_unscaled)
```

Load Model

```
joblib.dump(final_pipeline, "loan_denial_prediction_pipeline.pkl")
print("✅ Final pipeline saved as 'loan_denial_prediction_pipeline.pkl'")
```

```
✅ Final pipeline saved as 'loan_denial_prediction_pipeline.pkl'
```

References

AFI (2021) *Alternative Data for Credit Scoring: Regulatory Considerations and Use Cases*. Alliance for Financial Inclusion. Available at: https://www.afi-global.org

Barocas, S., Hardt, M. and Narayanan, A. (2020) *Fairness and Machine Learning: Limitations and Opportunities*. Cambridge: fairmlbook.org.

Bharath, R., Pranav, R. and Agarwal, S. (2023) 'AI-Driven Lending and Financial Inclusion: Emerging Models in Asia and Africa', *World Journal of Advanced Research and Reviews*, 20(2), pp. 108–122.

Björkegren, D. and Grissen, D. (2018) 'Behavior Revealed in Mobile Phone Usage Predicts Loan Repayment', *World Bank Economic Review*, 32(2), pp. 358–385. doi:10.1093/wber/lhx019.

CGAP (2020) *Needs-Based Segmentation of Underserved Populations in Kenya*. Available at: https://www.cgap.org

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002) 'SMOTE: Synthetic Minority Over-sampling Technique', *Journal of Artificial Intelligence Research*, 16, pp. 321–357.

Chen, T. and Guestrin, C. (2016) 'XGBoost: A Scalable Tree Boosting System', *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. doi:10.1145/2939672.2939785.

Chen, Y., Liu, X. and Jiang, X. (2022) 'Explainable Artificial Intelligence in Financial Credit Scoring', *IEEE Access*, 10, pp. 80325–80336.

European Commission (n.d.) *Ethics Guidelines for Trustworthy AI*. Available at: https://ec.europa.eu

FinRegLab (2022) *Alternative Data for Credit Underwriting: Empirical Research Report – Kenya Pilot*. Washington, D.C.: FinRegLab. Available at: https://finreglab.org

FSD Kenya (2019) *Digital Credit in Kenya: Facts and Figures from FinAccess 2019*. Nairobi: FSD Kenya.

Ghosh, S., Malik, S. and Bandyopadhyay, S. (2021) ‘Alternative Credit Scoring Models for Emerging Markets: A Machine Learning Approach’, *Journal of Risk and Financial Management*, 14(5), p. 206. doi:10.3390/jrfm14050206.

ICCR (2022) *Use of Alternative Data in Credit Risk Assessment: Opportunities and Challenges*. International Committee on Credit Reporting. Available at: https://www.worldbank.org

JUMO (2022) *Data-Driven Credit Models in Africa: Unlocking Access through Mobile Usage*. Available at: https://www.jumo.world

Kou, G., Yang, P., Xiao, F., Chen, Y. and Alsaadi, F.E. (2021) ‘Evaluation of Feature Selection Methods for Explainable Credit Scoring’, *Expert Systems with Applications*, 173, 114671. doi:10.1016/j.eswa.2021.114671.

Lundberg, S.M. and Lee, S.I. (2017) ‘A Unified Approach to Interpreting Model Predictions’, *Advances in Neural Information Processing Systems (NeurIPS)*, 30, pp. 4765–4774.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K. and Galstyan, A. (2021) ‘A Survey on Bias and Fairness in Machine Learning’, *ACM Computing Surveys*, 54(6), pp. 1–35. doi:10.1145/3457607.

Naik, M. (2021) ‘Fairness Audits and Bias Detection in Credit Decisioning Systems’, *AI Ethics Journal*, 2(1), pp. 33–47.

RiskSeal (2024) *Alternative Credit Scoring and Financial Inclusion: Industry Trends Report*. Available at: https://www.riskseal.com

Tala (2023) *Data for Good: AI Models for Microloans in Africa*. Available at: https://www.tala.co

World Bank (2023) *Fintech and Digital Financial Services: Global Perspectives and Regulatory Insights*. Washington, D.C.: The World Bank.

Xu, X., Wang, W. and Zhao, Y. (2022) ‘Mobile Financial Behavior Data and Credit Risk Evaluation: Empirical Evidence from China’, *Finance Research Letters*, 45, 102421. doi:10.1016/j.frl.2021.102421.