National College of Ireland

Configuration Manual

MSc Research Project
MSc Data Analytics

Varun Sai Yandapalli
Student ID: 23325836

School of Computing
National College of Ireland

Supervisor: Mr. Eric Gyamfi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Varun Sai Yandapalli
Student ID:	23325836
Programme:	MSc Data Anlalysis_B **Year:** 2024-25
Module:	MSc Research Project
Lecturer:	Mr. Eric Gyamfi
Submission Due Date:	15th September 2025
Project Title:	Stacked Machine Learning for Explainable Fraud detection using LIME and LLM.
Word Count:	800 **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:	Varun Sai Yandapalli
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

Varun Sai Yandapalli
Student ID: 23325836

1. Introduction

Banking systems today handle millions of transactions every day, making them prime targets for fraudulent activities. Detecting such frauds in real-time is critical to ensure financial security, maintain customer trust, and comply with regulatory standards. Traditional machine learning models can detect anomalies, but understanding *why* a transaction is marked fraudulent remains a challenge.

This project focuses on developing a robust fraud detection system using a combination of traditional machine learning models and **Large Language Models (LLMs)** for enhanced interpretability. The machine learning models are trained to classify transactions as legitimate or fraudulent, while LLMs are used to provide natural language summaries and explanations for the model's decisions.

The system is built on a real-world transactional dataset containing detailed customer, account, and transaction metadata. By integrating explainable AI techniques and LLMs, the solution not only identifies fraud but also makes the decision-making process transparent to both analysts and stakeholders.

This manual provides the technical configuration, dependencies, and step-by-step setup needed to reproduce and deploy the complete fraud detection pipeline.

2. Environmental Setup

2.1 Hardware Requirements

- 16GB RAM (min.)
- 500 GB HardDrive.
- Intel. Core i5 (min.)

2.2 Software Requirements

- Windows 11 or later
- Python 3.10 or later

2.3 Programming Prerequisites

- Python 3.10 or later
- Kaggle or Jupyter Notebook or Google colab

Kaggle has been selected to execute the code because it is cloud, which means secure accessibility in any location at any time and all the required packages are preinstalled. The work was performed on a Kaggle Notebook that had a Tesla T4 GPU providing enough resources in both RAM and GPU to allow the uninterrupted training of the model.

IDE: Kaggle Notebook

Programming Language: Python

Computation: Tesla T4 GPU

Visualization Library: Matplotlib, Plotly, Seaborn

ML Library: scikit-learn, XGBoost, mlxtend, LIME, imbalanced-learn

LIME: Explainable AI

LLM: Perplexity AI API (Sonar Pro Model)

3. Libraries Required

Library	Purpose
pandas	to load, clean, explore, and manipulate data
numpy	Numerical computation
seaborn, matplotlib, plotly	Data visualization
scikit-learn	ML algorithms, preprocessing, metrics
imbalanced-learn	SMOTE for class balancing
xgboost	Gradient boosting model
mlxtend	Stacking ensemble
lime	Local model interpretability
python-dotenv	Load API keys from .env
requests	API calls to LLM

4. Dataset

Source: https://www.kaggle.com/datasets/marusagar/bank-transaction-fraud-detection

- Contains detailed customer, account, and transaction metadata
- Includes both legitimate and fraudulent transaction labels

5. Project Flow

5.1 Importing all libraries : All the required libraries and modules for the project are imported using the code below.

```
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_scor
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from mlxtend.classifier import StackingClassifier
```

5.2 Data Loading : The dataset is loaded via kaggle's dataset using pandas which achieved a seamless integration.

```
df=pd.read_csv('/kaggle/input/bank-transaction-fraud-detection/Bank_Transaction_Fraud_Detection.csv')
```

```
df.head()
```

	Customer_ID	Customer_Name	Gender	Age	State	City	Bank_Branch	Account_Type	Transaction_ID	Transaction_Date	...	Merchant_Category	Account_Balance	Transaction_Device	T
0	d5f6ec07-d69e-4f47-b9b4-7c58ff17c19e	Osha Tella	Male	60	Kerala	Thiruvananthapuram	Thiruvananthapuram Branch	Savings	4fa3208f-9e23-42dc-b330-844829d0c12c	23-01-2025	...	Restaurant	74557.27	Voice Assistant	
1	7c14ad51-781a-4db9-b7bd-67439c175262	Hredhaan Khosla	Female	51	Maharashtra	Nashik	Nashik Branch	Business	c9de0c06-2c4c-40a9-97ed-3c7b8f97c79c	11-01-2025	...	Restaurant	74622.66	POS Mobile Device	
2	3a73a0e5-d4da-45aa-85f3-528413900a35	Ekani Nazareth	Male	20	Bihar	Bhagalpur	Bhagalpur Branch	Savings	e41c55f9-c016-4ff3-872b-cae72467c75c	25-01-2025	...	Groceries	66817.99	ATM	
3	7902f4ef-9050-4a79-857d-9c2ea3181940	Yamini Ramachandran	Female	57	Tamil Nadu	Chennai	Chennai Branch	Business	7f7ee11b-ff2c-45a3-802a-49bc47c02ecb	19-01-2025	...	Entertainment	58177.08	POS Mobile App	
4	3a4bba70-d9a9-4c5f-8b92-1735fd8c19e9	Kritika Rege	Female	43	Punjab	Amritsar	Amritsar Branch	Savings	f8e6ac6f-81a1-4985-bf12-f60967d852ef	30-01-2025	...	Entertainment	16108.56	Virtual Card	

5 rows × 24 columns

5.3 Data Cleaning: There are no null values and outliers. I just dropped few columns which.h are not required for model building

```
df.drop(['Customer_ID','Merchant_ID','Transaction_ID','Transaction_Currency'],axis=1,inplace=True)# unwanted columns
```

5.4 Data Visualization : There are few plots and graphs created to understand the distribution and nature of the dataset using plotly,seaborn and matplotlib.

5.5 Data Preprocessing :- Label encoding or to_categorical (converting categorical to numbers, handling imbalanced data,and data scaling.

```
X = df.drop("Is_Fraud", axis=1)
y = df["Is_Fraud"]
```

```
col = X.select_dtypes(exclude=['float64','int64']).columns.tolist()
#label encoding(converting categorical data to number)
le = LabelEncoder()
X[col] = X[col].apply(le.fit_transform).astype('int')
```

5.6 Feature importance: Plotting the feature importance graph using random forest and using only top 15 feature for model training

```
rf = RandomForestClassifier()
rf.fit(sm_x,sm_y)
importances = rf.feature_importances_
feature_names = X.columns  # Or use list of column names if it's a NumPy array

# Create DataFrame for easy sorting and plotting
feat_imp_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feat_imp_df = feat_imp_df.sort_values(by='Importance', ascending=False)

# Plot
plt.figure(figsize=(10, 6))
plt.barh(feat_imp_df['Feature'], feat_imp_df['Importance'], color='skyblue')
plt.xlabel('Importance Score')
plt.title('Feature Importance from Random Forest')
plt.gca().invert_yaxis()  # Most important at the top
plt.grid(True, axis='x')
plt.tight_layout()
plt.show()
```

5.7 Splitting data : For the ML Training the dataset is split in 90-10 ration for testing and training.

```
#split data in train and test with ratio of 90/10
X_train, X_test, y_train, y_test = train_test_split(newx,sm_y,test_size=0.1, stratify=sm_y, random_state=1)
```

5.8 Model Building

The codes attached below are the training part of applicable models for this project.

All the models used for this project are mentioned below:

5.8.1 Logistic regression

```
lr = LogisticRegression( C= 0.1, l1_ratio= 0, penalty= 'l2', solver='lbfgs',random_state=1)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
```

5.8.2 Decision Tree

```
dt = DecisionTreeClassifier(random_state=1)
dt.fit(X_train,y_train)
y_pred = dt.predict(X_test)
```

5.8.3 Random Forest

```
rfmodel = RandomForestClassifier(max_depth=5)
rfmodel.fit(X_train,y_train)
y_pred = rfmodel.predict(X_test)
```

5.8.4 Ada boost

```
ada = AdaBoostClassifier(random_state=1)
ada.fit(X_train, y_train)
y_pred = ada.predict(X_test)
```

5.8.5 Xgboost

```
model = XGBClassifier(max_depth=5,random_state=1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

5.8.6 Stacking:

In this code , I have used stacking technique to combine the advantages of different models which helps in gaining the opinion of every. Stacking uses a metamodel and combines the outputs of other models which helps in increase in overall accuracy.

```
# Define base models
model1 =  AdaBoostClassifier(random_state=1)
model2 =  RandomForestClassifier(random_state=1)
model3 = LogisticRegression(random_state=1)

# Define meta-model
meta_model = XGBClassifier(random_state=1)

# Create Stacking Classifier
stac_model = StackingClassifier(
    classifiers=[model1, model2, model3],
    meta_classifier=meta_model
)

# Fit the model
stac_model.fit(X_train, y_train)

# Predict
y_pred = stac_model.predict(X_test.values)
```

5.8.7 Model Evaluation :

Confusion Matrix, Classification Report: There are evaluated codes and outputs for every model under their training code cells.

5.8.8 Explainable ai :

This code takes one transaction as input from the test data and runs it through Lime, and displays the reason why model predicted Fraud or Not Fraud for this specific transaction.

```
from lime import lime_tabular

feature_names = [i for i in X.columns]

# Ensure X_train is a NumPy array
explainer = lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=feature_names,
    mode='classification'
)

# Pick a sample from test set to explain
sample_idx = 1

# Ensure X_test is also a NumPy array when passing a single row
explanation = explainer.explain_instance(
    data_row=X_test.iloc[sample_idx].values,
    predict_fn=stac_model.predict_proba    # returns class probabilities
)
# Visualize explanation
explanation.show_in_notebook(show_table=True)
```

6. LLM:

This code takes one prediction of a single model and explains that prediction by telling which features contributed most to making the decision, prints the results as a table, which we pass to llm to get human readable reasons for prediction.

```
# Create LIME tabular explainer
explainer = lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=X.columns.tolist(),
    class_names=[0,1],  # optional: update for your class labels
    mode='classification'
)

# Pick a test instance index to explain


# Explain that instance
explanation = explainer.explain_instance(
    data_row=X_test.iloc[sample_idx].values,
    predict_fn=stac_model.predict_proba  # If you're using stacking, use stac_model.predict_proba
)

# Convert explanation to DataFrame (tabular format)
exp_list = explanation.as_list()
exp_df = pd.DataFrame(exp_list, columns=['Feature', 'Importance'])

# Show explanation as DataFrame (tabular)
print("LIME Explanation for Instance Index", sample_idx)
print(exp_df)

# OPTIONAL: Save explanation to CSV
exp_df.to_csv("lime_explanation_instance_{}.csv".format(sample_idx), index=False)

# Show in notebook (if used in Jupyter)
explanation.show_in_notebook(show_table=True)
```

The code will pass prediction done using this model over to the Perplexity AI API (using the Sonar Pro model) and request it to create a short, human-readable explanation as to why the transaction was identified as either being a fraud or not being a fraud. It identifies the features of the transaction, the result that happened and the LIME explanation and prints the reasoning that has been generated by the AI.

```
import requests
api_key = "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"
url = "https://api.perplexity.ai/chat/completions"
headers = {
    "Authorization": f"Bearer {api_key}",
    "Content-Type": "application/json"
}
payload = {
    "model": "sonar-pro",   # or "gpt-4.1", "claude-4-sonnet", etc.
    "messages": [
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": f"This is a credit card fraud detection project,this are the feature of {X_test.iloc[sample_idx]},an
    ],
    "temperature": 0.8
}
response = requests.post(url, headers=headers, json=payload)
print(response.json()["choices"][0]["message"]["content"])
```

Final Output:

This is the snapshot of the explanation produced by the AI .

The output for your credit card fraud detection project shows feature values for a specific record and a list of features with their importance scores from an explainable AI method (such as SHAP). The model classified this record as **not fraud (0)**.

Summary of the output:
- The explainable AI results rank features (such as State, City, Transaction_Time, etc.) by their influence on the prediction. Features like **State > 0.67** and **City > 0.50** have large negative importance values, meaning they contributed strongly to predicting this record as *not fraud*.
- Features with positive values (e.g., **0.40 < Transaction_Time <= 0.80**) provided small support towards predicting fraud, bu t their influence was outweighed by the negative contributions.

Reason for the output using SHAP AI prediction:
- **SHAP (SHapley Additive Explanations)** breaks down the prediction into contributions from each feature.
- For this record, features such as **State** and **City** had SHAP values of approximately **-0.48** and **-0.42**, indicating they pushed the model strongly toward "not fraud".
- Only a few features (like certain ranges of Transaction_Time and Merchant_Category) contributed slightly towards predicting f raud, but not enough to change the final decision.
- The overall sum of SHAP values led to a model output below the fraud threshold, so the record was classified as **not fraud** [3].
- This approach increases transparency and trust, as stakeholders can see which features were most influential in the model's d ecision[3].

In summary, the model decided "not fraud" for this record because the most influential features (especially State and City) had values that strongly suggest normal, non-fraudulent behavior according to the model's learned patterns. SHAP provided a local e xplanation showing how each feature value contributed to the prediction[3].