

Configuration Manual

MSc Practicum part-2
MSc In Cyber Security

Arun Joy
Student ID: 23201592

School of Computing
National College of Ireland

Supervisor: Mr Vikas Sahni

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Arun Joy
Student ID: 23201592
Programme: MSc In Cyber Security **Year:** 2024-2025
Module: MSc Practicum part-2
Lecturer: Mr Vikas Sahni
Submission Due Date: 12/12/2024
Project Title: Insider Threat Detection using Ensemble and Sequential Models
Word Count: 606 **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: ARUN JOY
Date: 12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Arun Joy
23201592

1 Introduction

This document gives a configuration manual for a Google Colab machine learning project. This project builds an insider threat detection model using ensemble and sequential models. The study covers the examination of two datasets, email.csv and psychometric.csv, using classification methods. The dataset utilised here is CERT from Kaggle¹. This manual includes dataset preparation, feature selection, classification and assessment.

2 Hardware Requirements

The following pieces of hardware were used to build this project:

- Processor: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
- RAM: 16.0 GB
- Operating System: Microsoft Windows 11 Home
- Storage: 256 GB SSD
- Internet Connection: To access Google Colab, a steady internet connection is required.

3 Software Requirements

The following pieces of software tools were used to build this project:

- Google Colab²: Cloud-based Jupyter notebook environment.
- Python 3.10.12
- Libraries Used:
 - pandas 2.2.2
 - numpy 1.26.4
 - scikit-learn 1.5.2
 - tensorflow 2.17.1
 - matplotlib 3.8.0
 - seaborn 0.13.2

Google Colab is a cloud-based Jupyter notebook environment that offers an easily accessible platform for the tasks involving coding, data analysis and machine learning. It provides compatibility with contemporary libraries and frameworks and supports Python 3.10.12.

¹ <https://www.kaggle.com/code/alabiobin/insider-threat-detection/>

² <https://colab.google/>

4 Setting Up Google Colab Environment

Open the Google Colab and open New Notebook from the file. Click on the Connect button in the top-right corner to connect to a runtime environment.

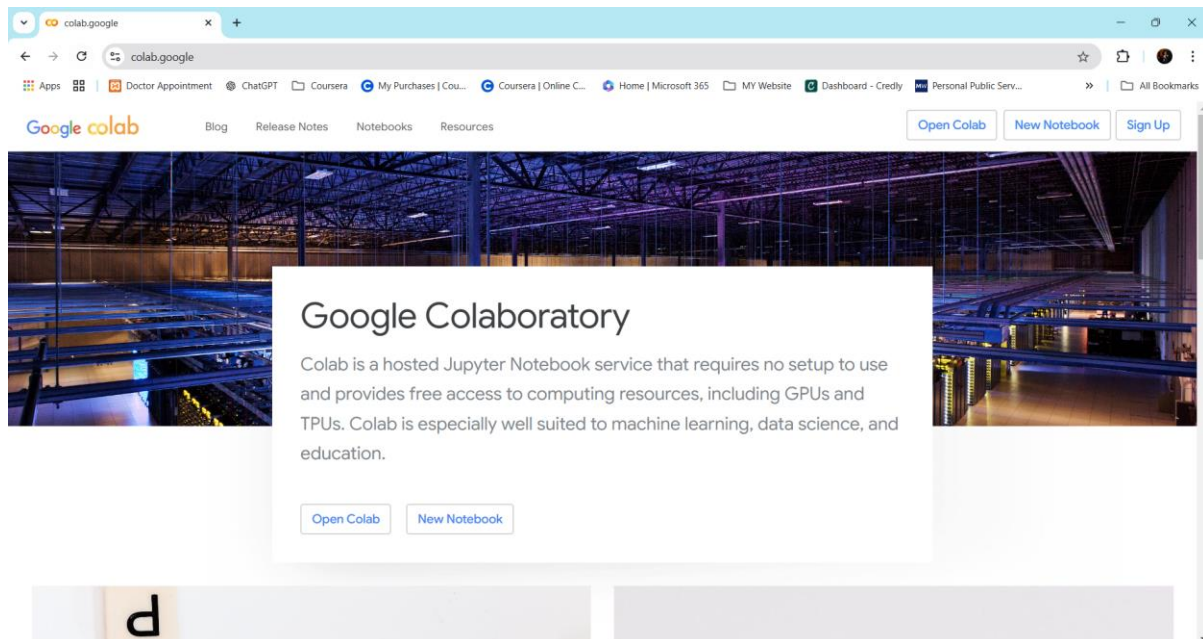


Figure 1: Google Colab

5 Dataset Preparation

Upload the files email.csv and psychometric.csv from the local machine. The email.csv contains email metadata and content. The Columns include: date: Timestamp of email, user: Email sender, attachments: Number of attachments, to, cc, bcc: Recipient details, content: Email body. The psychometric.csv can be used for the future additions. Import the required libraries and load the dataset as shown in Figure 2.

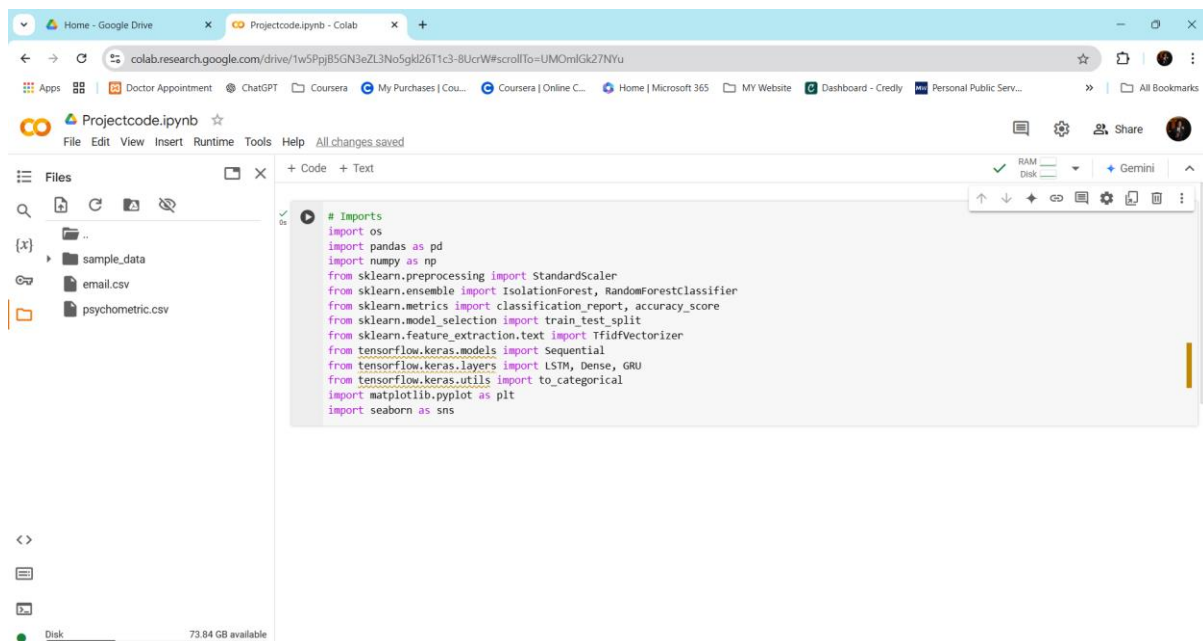
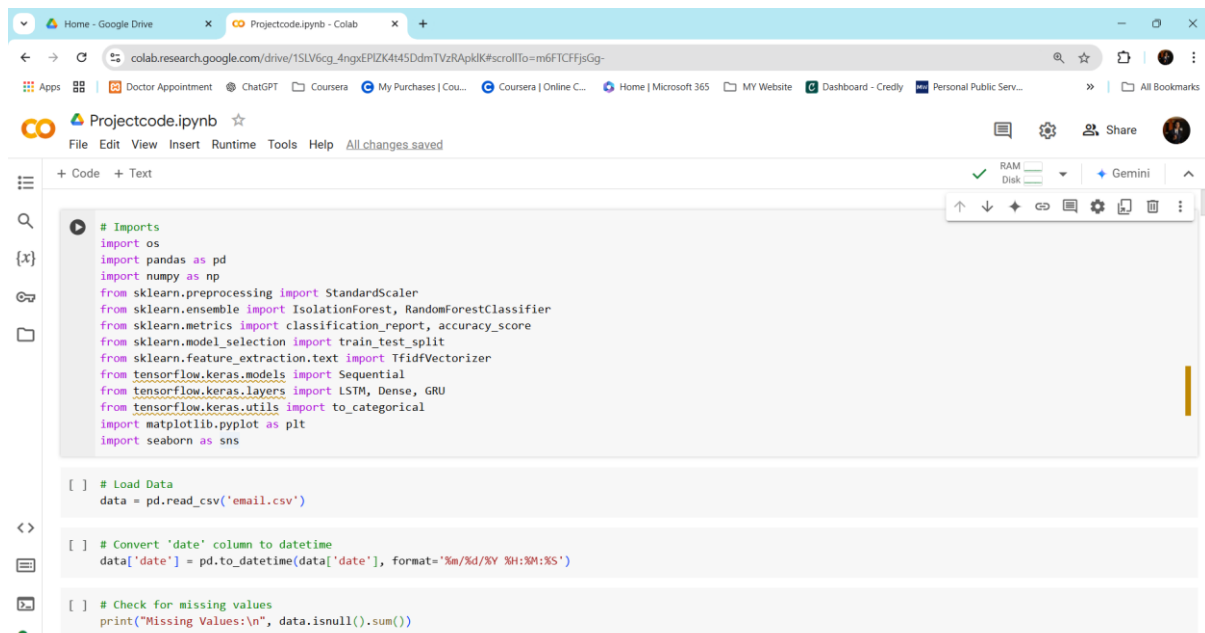


Figure 2: Upload datasets and import libraries

6 Code Configuration

Loading the dataset and the preprocessing of data including the conversion of data column to datetime format, fill missing values in cc and bcc with empty strings, exploratory data analysis, frequency of emails by the user and the number of attachments analysis are shown in Figure 3, Figure 4, Figure 5 and Figure 6. The feature engineering creates total number of recipients across to, cc, bcc and extracts the hour and day from the date column. The visualization script generates the histogram of email activity by hour and bar chart of the top 10 users by email activity as shown in Figure 7. The anomaly detection that includes the Isolation Forest to identifies anomalies in numeric features like size, attachments and num_recipients as shown in Figure 8. The Feature Extraction with TF-IDF Vectorization that processes the content column to extract 1,000 most significant features as shown in Figure 9.



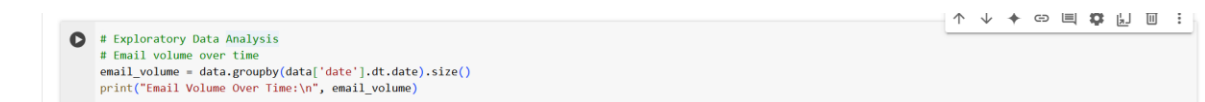
```
# Imports
import os
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import IsolationForest, RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, GRU
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import seaborn as sns

[ ] # Load Data
data = pd.read_csv('email.csv')

[ ] # Convert 'date' column to datetime
data['date'] = pd.to_datetime(data['date'], format='%m/%d/%Y %H:%M:%S')

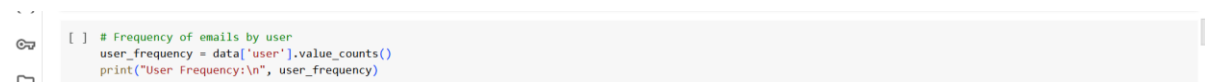
[ ] # Check for missing values
print("Missing Values:\n", data.isnull().sum())
```

Figure 3: Load data, convert 'date' column to datetime and check for missing values



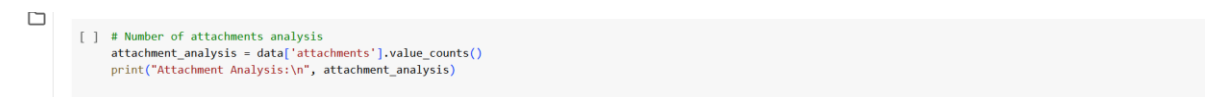
```
# Exploratory Data Analysis
# Email volume over time
email_volume = data.groupby(data['date'].dt.date).size()
print("Email Volume Over Time:\n", email_volume)
```

Figure 4: Exploratory Data Analysis



```
[ ] # Frequency of emails by user
user_frequency = data['user'].value_counts()
print("User Frequency:\n", user_frequency)
```

Figure 5: Frequency of emails by user



```
[ ] # Number of attachments analysis
attachment_analysis = data['attachments'].value_counts()
print("Attachment Analysis:\n", attachment_analysis)
```

Figure 6: Number of attachments analysis

```

[ ] # Data Preprocessing and Feature Engineering
data['cc'] = data['cc'].fillna('')
data['bcc'] = data['bcc'].fillna('')
data['num_recipients'] = data['to'].str.count(';') + data['cc'].str.count(';') + data['bcc'].str.count(';') + 1
data['hour'] = data['date'].dt.hour
data['day_of_week'] = data['date'].dt.dayofweek

[ ] # Visualizations
plt.figure(figsize=(12, 6))
sns.histplot(data['hour'], bins=24, kde=False)
plt.title('Email Activity by Hour')
plt.xlabel('Hour of Day')
plt.ylabel('Email Count')
plt.show()

top_users = data['user'].value_counts().head(10)
plt.figure(figsize=(12, 6))
sns.barplot(x=top_users.index, y=top_users.values)
plt.title('Top 10 Users by Email Activity')
plt.xlabel('User')
plt.ylabel('Email Count')
plt.show()

```

Figure 7: Data Preprocessing and Feature Engineering and Visualization

```

[ ] # Prepare data for anomaly detection
features = ['size', 'attachments', 'num_recipients', 'hour', 'day_of_week']
X = data[features]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

[ ] # Isolation Forest model for anomaly detection
model = IsolationForest(n_estimators=100, contamination=0.01, random_state=42)
data['anomaly'] = model.fit_predict(X_scaled)

[ ] # Analyze anomalies
anomalies = data[data['anomaly'] == -1]
print(f"Number of Anomalies Detected: {len(anomalies)}")

```

Figure 8: Prepare data for anomaly detection, Isolation Forest and analyze anomalies

```

[ ] # Sample and further preprocess the data
data_sample = data.sample(frac=0.1, random_state=42)
data_sample['cc'] = data_sample['cc'].fillna('')
data_sample['bcc'] = data_sample['bcc'].fillna('')
data_sample['num_recipients'] = data_sample['to'].str.count(';') + data_sample['cc'].str.count(';') + data_sample['bcc'].str.count(';') + 1
data_sample['hour'] = data_sample['date'].dt.hour
data_sample['day_of_week'] = data_sample['date'].dt.dayofweek

[ ] # TF-IDF Vectorization on email content
tfidf = TfidfVectorizer(max_features=1000)
content_tfidf = tfidf.fit_transform(data_sample['content']).toarray()
content_tfidf_df = pd.DataFrame(content_tfidf, columns=tfidf.get_feature_names_out())

[ ] # Combine numeric features with TF-IDF features
X_numeric = data_sample[features]
X = pd.concat([X_numeric.reset_index(drop=True), content_tfidf_df.reset_index(drop=True)], axis=1)
scaler = StandardScaler()
X_numeric_scaled = scaler.fit_transform(X_numeric)
X_scaled = pd.concat([pd.DataFrame(X_numeric_scaled, columns=features).reset_index(drop=True), content_tfidf_df.reset_index(drop=True)], axis=1)

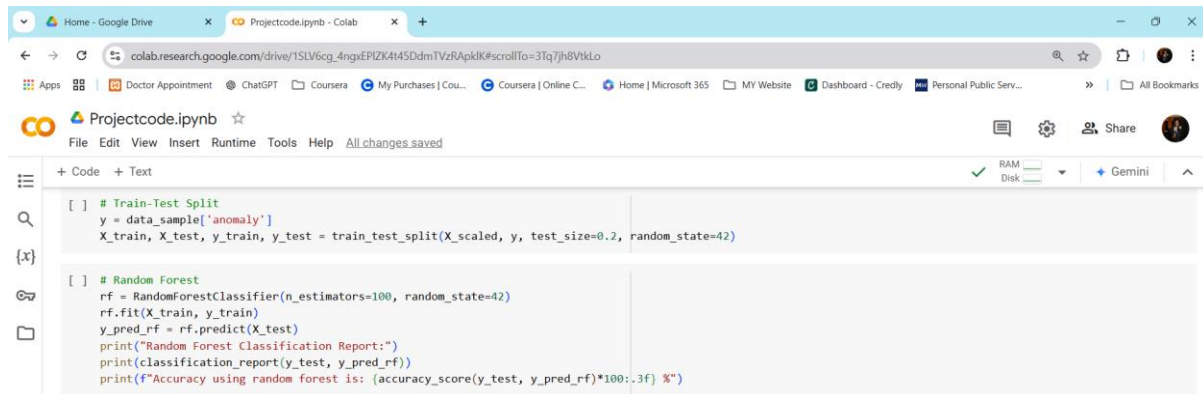
[ ] # Isolation Forest on the combined data
model = IsolationForest(n_estimators=100, contamination=0.01, random_state=42)
data_sample['anomaly'] = model.fit_predict(X_scaled)
data_sample['anomaly'] = data_sample['anomaly'].map({1: 0, -1: 1})

```

Figure 9: Sample and further preprocess the data and TF-IDF Vectorization

7 Model Training and Testing

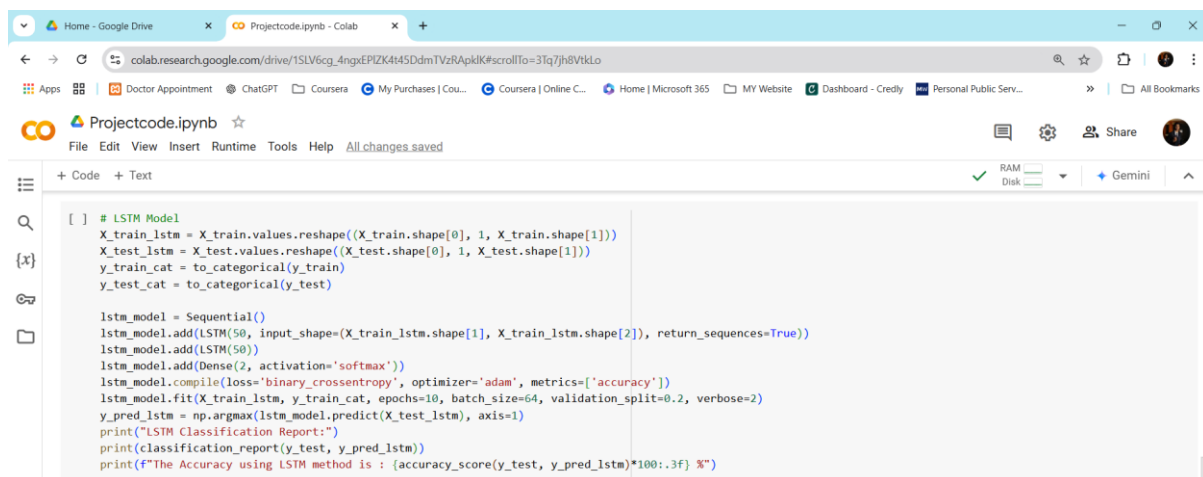
The data is split into training (80%) and testing (20%) sets. The models include Random Forest, LSTM, GRU, Gradient Boosting, Multi-Layer Perceptron (MLP) and Stacking Ensemble. Each model outputs have a classification Report including Precision, Recall, F1-Score and accuracy as shown in Figure 10, Figure 11, Figure 12, Figure 13 Figure 14 and Figure 15.



```
[ ] # Train-Test Split
y = data_sample['anomaly']
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

[ ] # Random Forest
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))
print(f"Accuracy using random forest is: {accuracy_score(y_test, y_pred_rf)*100:.3f} %")
```

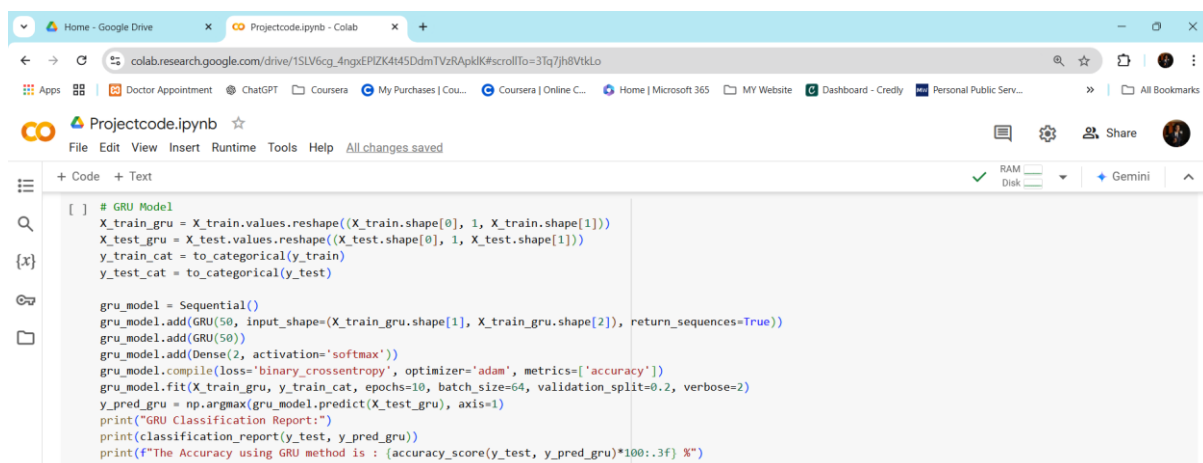
Figure 10: Train-Test Split and Random Forest



```
[ ] # LSTM Model
X_train_lstm = X_train.values.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test_lstm = X_test.values.reshape((X_test.shape[0], 1, X_test.shape[1]))
y_train_cat = to_categorical(y_train)
y_test_cat = to_categorical(y_test)

lstm_model = Sequential()
lstm_model.add(LSTM(50, input_shape=(X_train_lstm.shape[1], X_train_lstm.shape[2]), return_sequences=True))
lstm_model.add(LSTM(50))
lstm_model.add(Dense(2, activation='softmax'))
lstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
lstm_model.fit(X_train_lstm, y_train_cat, epochs=10, batch_size=64, validation_split=0.2, verbose=2)
y_pred_lstm = np.argmax(lstm_model.predict(X_test_lstm), axis=1)
print("LSTM Classification Report:")
print(classification_report(y_test, y_pred_lstm))
print(f"The Accuracy using LSTM method is : {accuracy_score(y_test, y_pred_lstm)*100:.3f} %")
```

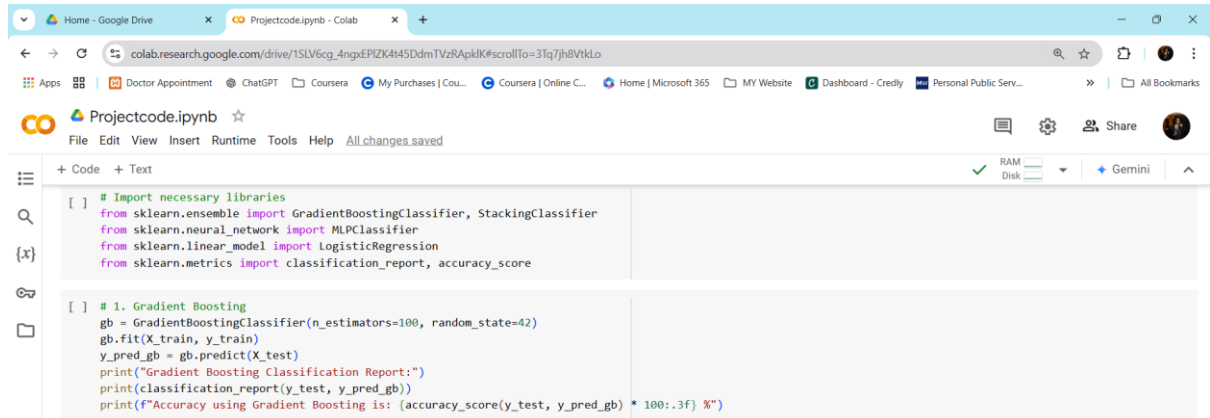
Figure 11: LSTM Model



```
[ ] # GRU Model
X_train_gru = X_train.values.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test_gru = X_test.values.reshape((X_test.shape[0], 1, X_test.shape[1]))
y_train_cat = to_categorical(y_train)
y_test_cat = to_categorical(y_test)

gru_model = Sequential()
gru_model.add(GRU(50, input_shape=(X_train_gru.shape[1], X_train_gru.shape[2]), return_sequences=True))
gru_model.add(GRU(50))
gru_model.add(Dense(2, activation='softmax'))
gru_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
gru_model.fit(X_train_gru, y_train_cat, epochs=10, batch_size=64, validation_split=0.2, verbose=2)
y_pred_gru = np.argmax(gru_model.predict(X_test_gru), axis=1)
print("GRU Classification Report:")
print(classification_report(y_test, y_pred_gru))
print(f"The Accuracy using GRU method is : {accuracy_score(y_test, y_pred_gru)*100:.3f} %")
```

Figure 12: GRU Model

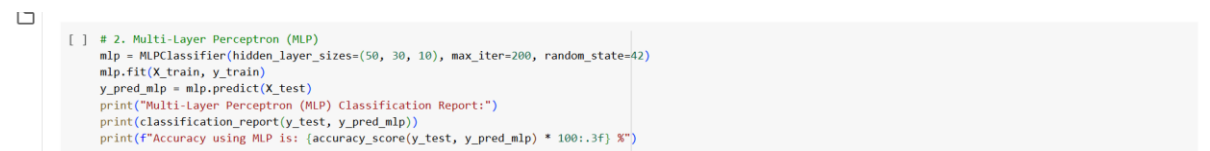


The screenshot shows a Google Colab notebook titled 'Projectcode.ipynb'. The code is as follows:

```
[ ] # Import necessary libraries
from sklearn.ensemble import GradientBoostingClassifier, StackingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

[ ] # 1. Gradient Boosting
gb = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb.fit(X_train, y_train)
y_pred_gb = gb.predict(X_test)
print("Gradient Boosting Classification Report:")
print(classification_report(y_test, y_pred_gb))
print(f"Accuracy using Gradient Boosting is: {accuracy_score(y_test, y_pred_gb) * 100:.3f} %")
```

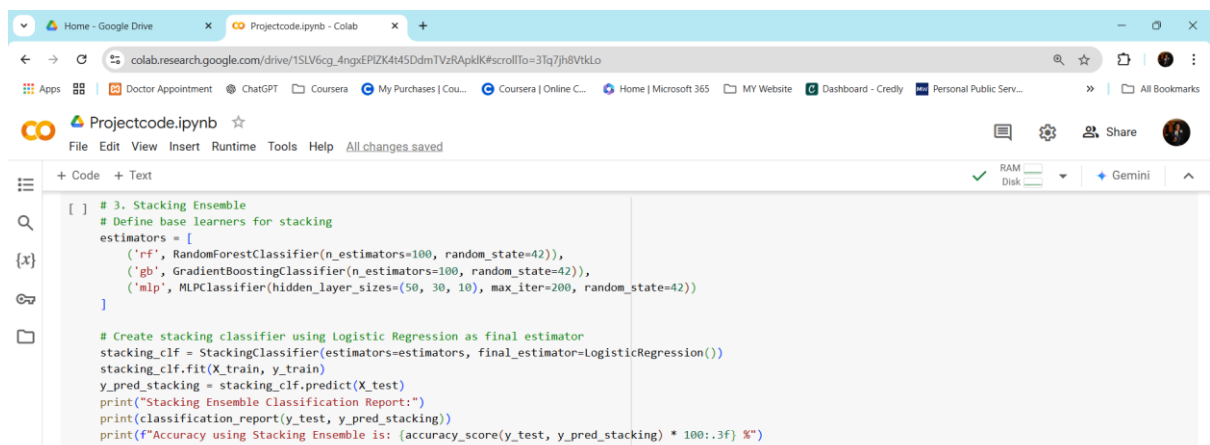
Figure 13: Gradient Boosting



The screenshot shows a Google Colab notebook titled 'Projectcode.ipynb'. The code is as follows:

```
[ ] # 2. Multi-Layer Perceptron (MLP)
mlp = MLPClassifier(hidden_layer_sizes=(50, 30, 10), max_iter=200, random_state=42)
mlp.fit(X_train, y_train)
y_pred_mlp = mlp.predict(X_test)
print("Multi-Layer Perceptron (MLP) Classification Report:")
print(classification_report(y_test, y_pred_mlp))
print(f"Accuracy using MLP is: {accuracy_score(y_test, y_pred_mlp) * 100:.3f} %")
```

Figure 14: Multi-Layer Perceptron (MLP)



The screenshot shows a Google Colab notebook titled 'Projectcode.ipynb'. The code is as follows:

```
[ ] # 3. Stacking Ensemble
# Define base learners for stacking
estimators = [
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
    ('gb', GradientBoostingClassifier(n_estimators=100, random_state=42)),
    ('mlp', MLPClassifier(hidden_layer_sizes=(50, 30, 10), max_iter=200, random_state=42))
]

# Create stacking classifier using Logistic Regression as final estimator
stacking_clf = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression())
stacking_clf.fit(X_train, y_train)
y_pred_stacking = stacking_clf.predict(X_test)
print("Stacking Ensemble Classification Report:")
print(classification_report(y_test, y_pred_stacking))
print(f"Accuracy using Stacking Ensemble is: {accuracy_score(y_test, y_pred_stacking) * 100:.3f} %")
```

Figure 15: Stacking Ensemble