National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Aishwarya Rani Gopal
Student ID: 21226105

School of Computing
National College of Ireland

Supervisor: Mayank Jain

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aishwarya Rani Gopal
Student ID:	21226105
Programme:	Data Analytics
Year:	2023
Module:	MSc Research Project
Supervisor:	Mayank Jain
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	820
Page Count:	7

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:	Aishwarya Rani Gopal
Date:	29th January 2024

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

Aishwarya Rani Gopal
21226105

1 Introduction

The main objective of this manual is to explain the hardware and software setups required to duplicate the research in the future. This paper describes the steps necessary to execute the code along with the applications and packages required.

2 System Configuration

This section discusses the hardware and software configurations of the project.

2.1 Hardware Configuration

The Hardware configuration of the project is mention in Figure 1.

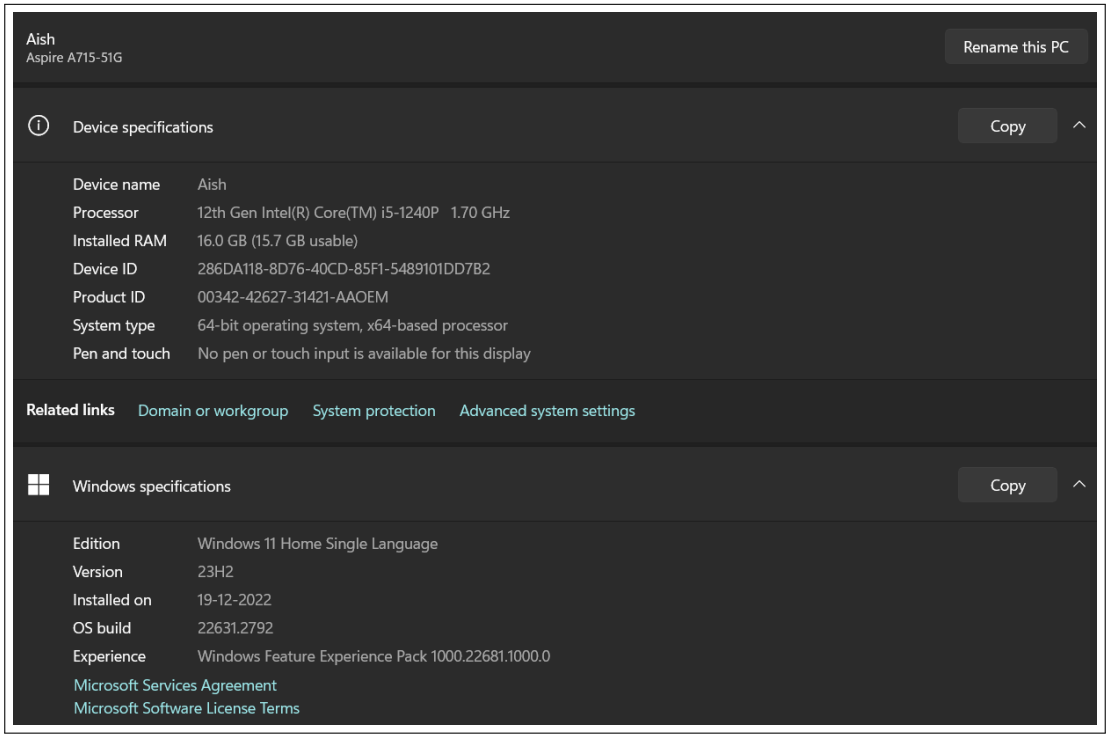

Figure 1: Hardware Specification

2.2 Software Configuration

Google Colab[1] is easy to use. Using a Google account sign-in, Google Colab can be accessed. Required files can be imported into Google Drive and can be accessed using Google Colab. It integrates smoothly with various Google services and supports GPU and TPU acceleration, making it an ideal platform for machine learning and data analysis projects (Figure 2). Other Software like Microsoft Excel is used for initial data exploration. Lifetime of the VM is 12hrs.

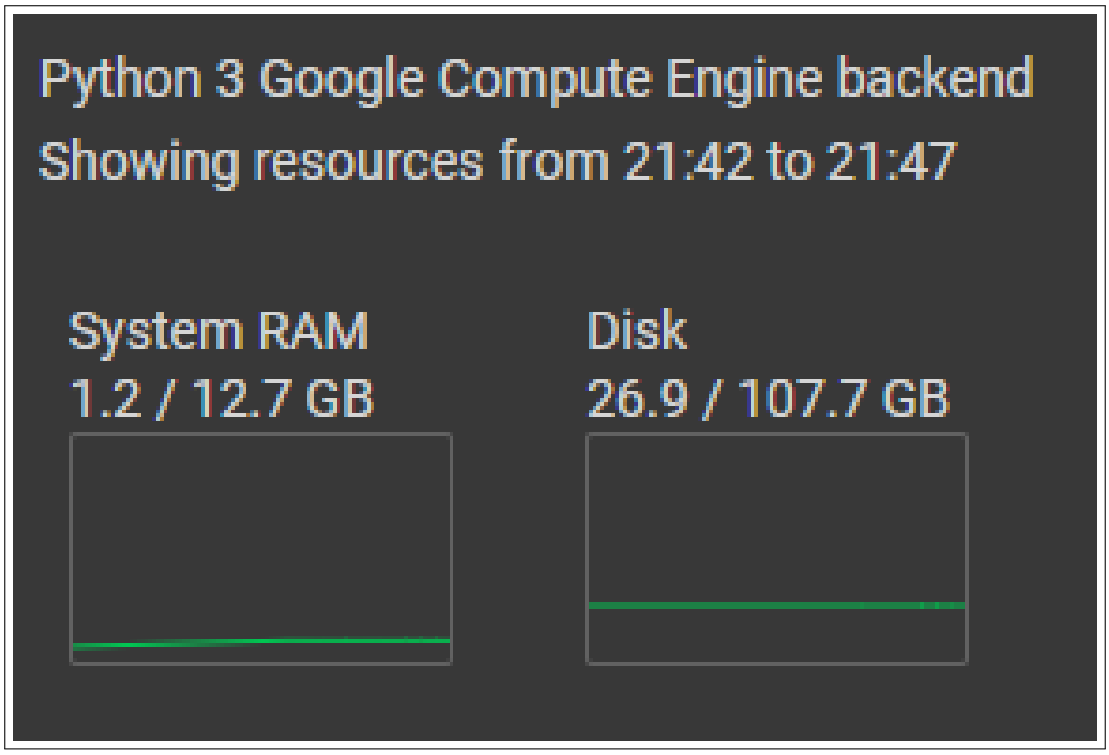

Figure 2: Software Specification

3 Python Libraries

Three Colab notebooks have been used for this project as shown in Figure 3.

- Data Visualization - DataVisualize_Irish.ipynb is used to perform Exhaustive Exploratory Data Analysis
- Modelling for Irish Dataset - ModelBuilding_Irish.ipynb is used in Data Preparation and Model Building for Irish Dataset.
- Model Building for OSHA Dataset - ModelBuilding_OSHA.ipynb is used in Data Preparation and Model Building for OSHA Dataset.

[1] `https://colab.research.google.com/`

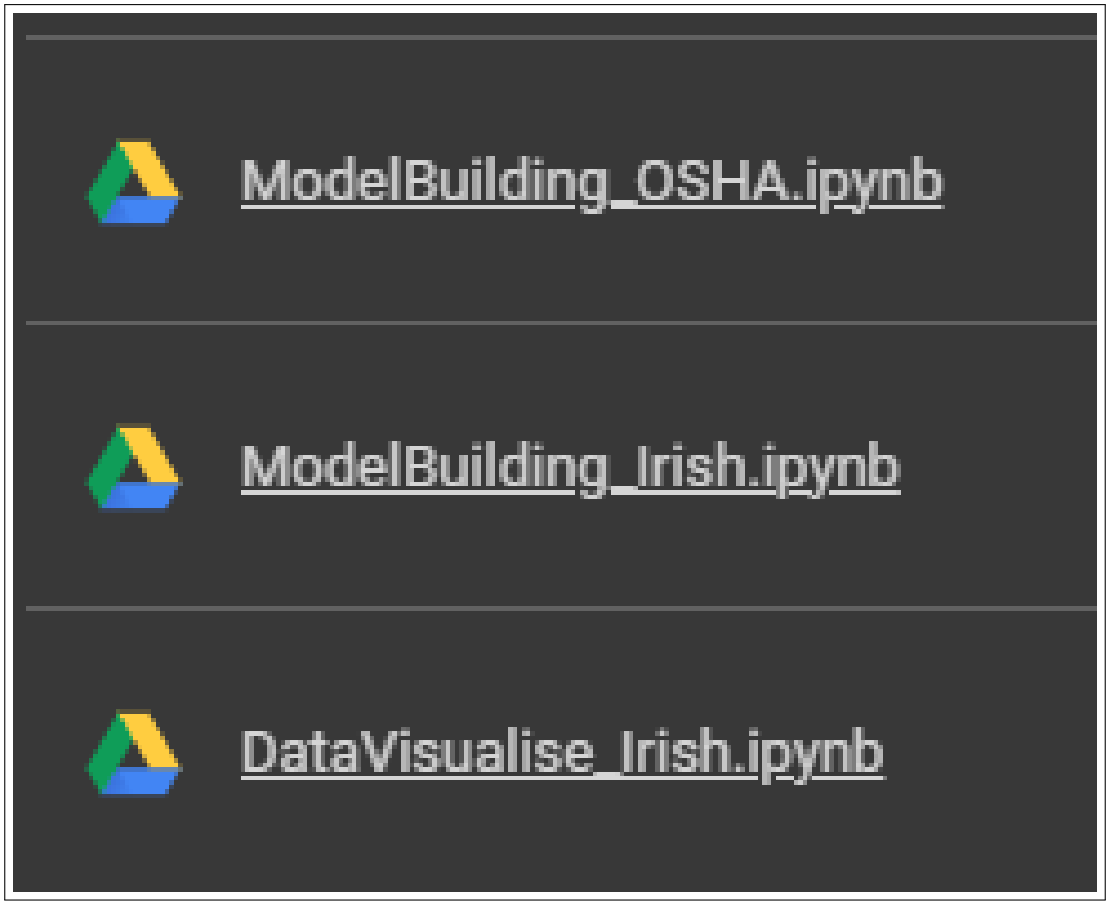

Figure 3: Colab Files

3.1 Installing Libraries

The Figure 4 shows the code snippet to installation of all the required libraries used in the research.

```
!pip install pandas numpy matplotlib seaborn joblib scikit-learn imbalanced-learn xgboost tensorflow scipy wordcloud
```

Figure 4: Libraries installation

3.2 Importing Libraries

The Figure 5 shows the code snippet to Import the downloaded libraries used in the research.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
from sklearn.model_selection import train_test_split, RandomizedSearchCV, StratifiedKFold
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, MinMaxScaler, label_binarize, StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, precision_recall_curve, auc, roc_auc_score, roc_curve
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import make_pipeline
from scipy.stats import chi2_contingency, f_oneway
from wordcloud import WordCloud
from itertools import cycle
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.utils.class_weight import compute_class_weight
```

Figure 5: Importing Libraries

4 Project Development

4.1 Data Preparation

This research uses two datasets. Irish Dataset which is the primary data is taken from the Irish Government website [2]. The secondary dataset is taken from Kaggle which contains the Occupational Safety and Health Administration(OSHA)[3] details.

These two files are loaded in Google Drive which is imported into Google Colab by mounting the drive as shown in Figure 6.

```
from google.colab import drive
drive.mount(/content/drive)

# Load the dataset
data = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/hsa-incidents.csv")
```

```
from google.colab import drive
drive.mount(/content/drive)

# Load the OSHA dataset
data_path = '/content/drive/MyDrive/Colab Notebooks/OSHA HSE DATA_ALL ABSTRACTS 15-17_FINAL.csv'
data = pd.read_csv(data_path)
```

Figure 6: Mounting and reading the files

[2] https://data.gov.ie/dataset/workplace-incidents-2017-2021?package_type=dataset
[3] https://www.kaggle.com/datasets/ruqaiyaship/osha-accident-and-injury-data-1517

4.2 Data Handling

- After removing the missing values from The Irish Dataset, all the categorical variables were encoded into binary using one-hot encoding shown in Figure 7.
- A Decision Tree classifier was used for feature selection as shown in Figure 8.
- After which Feature scaling was performed using MinMax scaling as illustrated in Figure 9.
- Resampling techniques such as SMOTE and RUS have been employed to balance the class distribution, especially the minority class as shown in 10.

```
# 4. One-Hot Encoding
object_cols = [col for col in X_train.columns if X_train[col].dtype == 'object']
OH_encoder = OneHotEncoder(handle_unknown='ignore', sparse=False)
OH_cols_train = pd.DataFrame(OH_encoder.fit_transform(X_train[object_cols]))
OH_cols_val = pd.DataFrame(OH_encoder.transform(X_val[object_cols]))
OH_cols_test = pd.DataFrame(OH_encoder.transform(X_test[object_cols]))

OH_cols_train.index, OH_cols_val.index, OH_cols_test.index = X_train.index, X_val.index, X_test.index

num_X_train = X_train.drop(object_cols, axis=1)
num_X_val = X_val.drop(object_cols, axis=1)
num_X_test = X_test.drop(object_cols, axis=1)

X_train_OH = pd.concat([num_X_train, OH_cols_train], axis=1)
X_val_OH = pd.concat([num_X_val, OH_cols_val], axis=1)
X_test_OH = pd.concat([num_X_test, OH_cols_test], axis=1)
```

Figure 7: Code Snippet for One-Hot encoding

```
# 5. Feature Selection
feature_selector = DecisionTreeClassifier()
feature_selector.fit(X_train_OH, y_train)
feature_importances = feature_selector.feature_importances_
features_df = pd.DataFrame({'Feature': X_train_OH.columns, 'Importance': feature_importances})
top_features = features_df.sort_values(by='Importance', ascending=False)

selected_features = top_features['Feature'][:20].tolist()
X_train_selected = X_train_OH[selected_features]
X_val_selected = X_val_OH[selected_features]
X_test_selected = X_test_OH[selected_features]
```

Figure 8: Code Snippet for Feature Selection

```
# 8. Feature Scaling
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train_selected)
X_val_scaled = scaler.transform(X_val_selected)
X_test_scaled = scaler.transform(X_test_selected)
```

Figure 9: Code Snippet for Feature Scaling

```
# 9. Resampling
smote = SMOTE()
X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled, y_train_encoded)

rus = RandomUnderSampler()
X_train_rus, y_train_rus = rus.fit_resample(X_train_scaled, y_train_encoded)
```

Figure 10: Code Snippet for Resampling

5 Model Building

This research performs Six experiments with five Machine Learning Algorithms such as SVM, Adaboost, XG Boost, Naive Bayes and Multi- Layer Perceptron (MLP). Each of these algorithms with the same parameters are Experimented with Feature Selected and Class balanced datasets both in Irish Dataset and OSHA dataset.

Each Model is trained with Dataset from Feature selection, Feature Selection and SMOTE and Feature Selection and RUS as shown in Figure 11 for both Irish Dataset and OSHA dataset. The Same technique is applied to dataset with all features and all features and RUS.

Modelling with Feature selected Data

```
[ ] # Binarize the output
    n_classes = len(np.unique(y_train_encoded))
    y_train_bin = label_binarize(y_train_encoded, classes=[*range(n_classes)])
    y_val_bin = label_binarize(y_val_encoded, classes=[*range(n_classes)])

    # Initialize models
    models = {
        'SVM': SVC(class_weight='balanced', probability=True, random_state=42),
        'AdaBoost': AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200, algorithm="SAMME.R", learning_rate=0.5, random_state=42),
        'XGBoost': xgb.XGBClassifier(n_estimators=200, learning_rate=0.05, max_depth=3, random_state=42),
        'Naive Bayes': GaussianNB(),
        'Neural Network': MLPClassifier(hidden_layer_sizes=(100,), max_iter=200, alpha=1e-4, solver='sgd', verbose=10, tol=1e-4, random_state=42, learning_rate_init=.1)
    }

    # Train models, evaluate and plot Precision-Recall curve
    results_fs = pd.DataFrame(columns=['Model', 'Accuracy', 'Classification Report', 'Confusion Matrix'])
    for name, model in models.items():
        print(f"Training {name}...")
        model.fit(X_train_selected, y_train_encoded)
        y_pred = model.predict(X_val_selected)
        accuracy = accuracy_score(y_val_encoded, y_pred)
        class_report = classification_report(y_val_encoded, y_pred)
        conf_matrix = confusion_matrix(y_val_encoded, y_pred)
        results_fs = results_fs.append({'Model': name, 'Accuracy': accuracy, 'Classification Report': class_report, 'Confusion Matrix': conf_matrix}, ignore_index=True)
        # Save the model
        joblib.dump(model, f'/content/drive/MyDrive/{name}_model.pkl')
        print(f"{name} trained and saved.\n")
```

Modeling with Feature selected SMOTE data

+ Code + Text

```
# Binarize the output
n_classes = len(np.unique(y_train_smote))
y_train_bin = label_binarize(y_train_smote, classes=[*range(n_classes)])
y_val_bin = label_binarize(y_val_encoded, classes=[*range(n_classes)])

# Initialize models
models = {
    'SVM': SVC(class_weight='balanced', probability=True, random_state=42),
    'AdaBoost': AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200, algorithm="SAMME.R", learning_rate=0.5, random_state=42),
    'XGBoost': xgb.XGBClassifier(n_estimators=200, learning_rate=0.05, max_depth=3, random_state=42),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(hidden_layer_sizes=(100,), max_iter=200, alpha=1e-4, solver='sgd', verbose=10, tol=1e-4, random_state=42, learning_rate_init=.1)
}

# Train models, evaluate and plot Precision-Recall curve
results_smote = pd.DataFrame(columns=['Model', 'Accuracy', 'Classification Report', 'Confusion Matrix', 'AUC'])
for name, model in models.items():
    print(f"Training {name}...")
    model.fit(X_train_smote, y_train_smote)
    y_pred = model.predict(X_val_scaled)
    accuracy = accuracy_score(y_val_encoded, y_pred)
    class_report = classification_report(y_val_encoded, y_pred)
    conf_matrix = confusion_matrix(y_val_encoded, y_pred)
```

Modeling with Feature selected RUS data

```
# Binarize the output
n_classes = len(np.unique(y_train_rus))
y_train_bin = label_binarize(y_train_rus, classes=[*range(n_classes)])
y_val_bin = label_binarize(y_val_encoded, classes=[*range(n_classes)])

# Initialize models
models = {
    'SVM': SVC(class_weight='balanced', probability=True, random_state=42),
    'AdaBoost': AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200, algorithm="SAMME.R", learning_rate=0.5, random_state=42),
    'XGBoost': xgb.XGBClassifier(n_estimators=200, learning_rate=0.05, max_depth=3, random_state=42),
    'Naive Bayes': GaussianNB(),
    'Neural Network': MLPClassifier(hidden_layer_sizes=(100,), max_iter=200, alpha=1e-4, solver='sgd', verbose=10, tol=1e-4, random_state=42, learning_rate_init=.1)
}

# Train models, evaluate and plot Precision-Recall curve
results_rus = pd.DataFrame(columns=['Model', 'Accuracy', 'Classification Report', 'Confusion Matrix', 'AUC'])
for name, model in models.items():
    print(f"Training {name}...")
    model.fit(X_train_rus, y_train_rus)
    y_pred = model.predict(X_val_scaled)
    accuracy = accuracy_score(y_val_encoded, y_pred)
    class_report = classification_report(y_val_encoded, y_pred)
    conf_matrix = confusion_matrix(y_val_encoded, y_pred)
```

Figure 11: Code Snippet for Modeling