Configuration Manual

MSc Research Project
MSc. Data Analytics

Tej Patel Yeliyuru Ramu
Student ID: x23216077

School of Computing
National College of Ireland

Supervisor: Cristina Hava Muntean

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Tej Patel Yeliyuru Ramu

Student ID: X23216077

Programme: Data Analytics **Year:** 2024

Module: MSc Research Project

Lecturer: Cristina Hava Muntean

Submission Due Date: 12/12/2024

Project Title: Advancing Chronic Disease Analytics by Predicting Cardiovascular Disease Risk Based on Demographic and Health Factors in the US

Word Count: 1117 **Page Count:** 6

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: Tej Patel Yeliyuru Ramu

Date: 12/12/2024

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

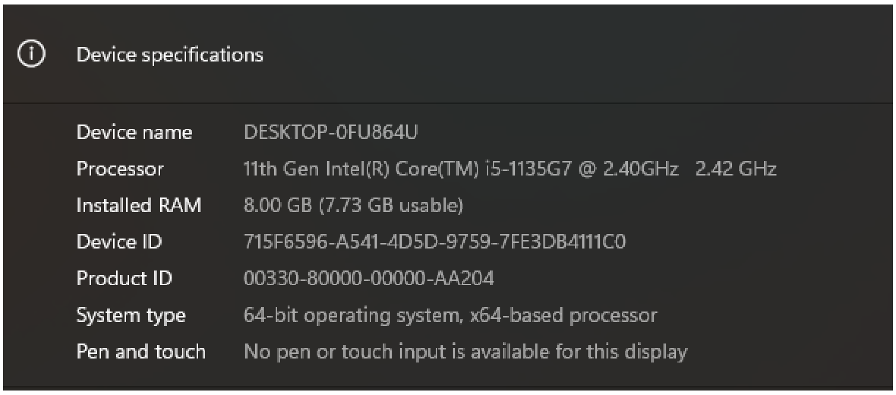
Tej Patel Yeliyuru Ramu
Student ID: x23216077

1 Introduction

This configuration manual presents the technical configuration and steps required to construct machine learning models that can effectively estimate the risk of CVD. In this paper, detailed steps are provided to build the computational environment including hardware, software, and required libraries. Further, it describes how the data has been collected, focusing mainly on the use of the US. The pre-processed version of the Chronic Diseases Indicator Dataset, followed by a discussion on the methods applied to clean and prepare the data, is presented. The machine learning model applied such as Logistic Regression, Random Forest, Gradient Boosting, KNN and SVM together with the metrics to establish their performances is shown. One can easily reproduce these experiments, extend the research and investigate possible improvements in the models applied to CVD risk prediction by following this guide.

2 Environment Setup

The environment setup developed for this research is focused on the attainment of the compromise between best models and its computational efficiency and ease of use. The hardware used for the study of this project is shown in Figure 1. An average hardware setup was used, Intel Core i5 from the 8th generation supported by 8GB of RAM, the installed operating system was Windows 11, 64-bit. This hardware configuration was more than enough for the task demands of data processing and machine learning during the research. The software environment was implemented based on JupyterLab, a very powerful platform that makes Python work seamlessly, version 3.9, for running the code and supporting development processes throughout the duration of the project. The key libraries used in this project are pandas for data manipulation, NumPy and scikit-learn for the construction of the models, matplotlib, seaborn for the visualization and many more.



Device specifications

Device name	DESKTOP-0FU864U
Processor	11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
Installed RAM	8.00 GB (7.73 GB usable)
Device ID	715F6596-A541-4D5D-9759-7FE3DB4111C0
Product ID	00330-80000-00000-AA204
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Figure 1: System Configuration used in the research

3 Hardware Requirements

Processor: Minimum Intel Core i5 or equivalent
RAM: Minimum 8 GB
Storage: Minimum 1 GB of free space for application and datasets

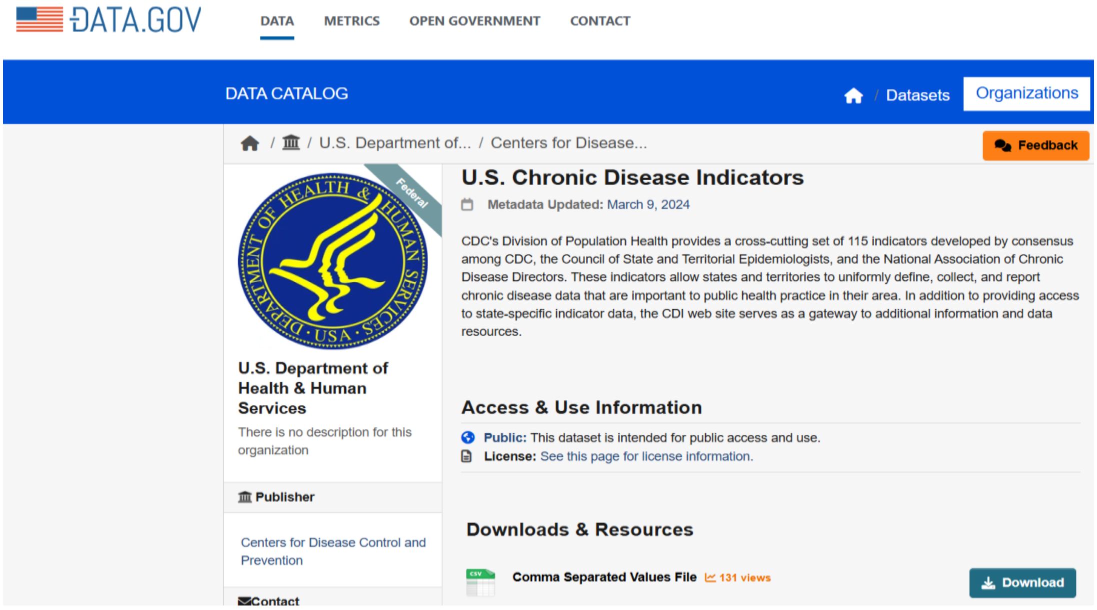
4 Software Requirements

Operating System
Windows: Windows 10 or later (for development environment)
macOS: macOS 10.14 or later (for development environment)
Linux: Ubuntu 20.04 or later (for development environment)
Development Environment:
JupyterLab (for writing and executing Python code).
Google Colab (for cloud-based computation and additional processing power)
Python Version:
Python 3.9 (compatible with all required libraries).
Anaconda: For managing Python environments and packages.

5 Data Collection and Preparation

The dataset utilized for this research is the US Chronic Disease Indicators (CDI), sourced from the US Centres for Disease Control and Prevention (CDC) shown in the below Figure 2, covering health data from 2015 to 2022. The dataset contains 310,481 rows and 34 features including demographic and health indicators such as age, sex, race, and health conditions. The link to the dataset in provided below for your reference.

U.S. Chronic Disease Indicators Dataset - https://catalog.data.gov/dataset/u-s-chronic-disease-indicators Services & U.S. Department of Health & Human Services, (2024)



U.S. Chronic Disease Indicators

Metadata Updated: March 9, 2024

CDC's Division of Population Health provides a cross-cutting set of 115 indicators developed by consensus among CDC, the Council of State and Territorial Epidemiologists, and the National Association of Chronic Disease Directors. These indicators allow states and territories to uniformly define, collect, and report chronic disease data that are important to public health practice in their area. In addition to providing access to state-specific indicator data, the CDI web site serves as a gateway to additional information and data resources.

U.S. Department of Health & Human Services

There is no description for this organization

Centers for Disease Control and Prevention

Access & Use Information

Public: This dataset is intended for public access and use.

License: See this page for license information.

Downloads & Resources

Comma Separated Values File 131 views

Figure 2: Dataset Location

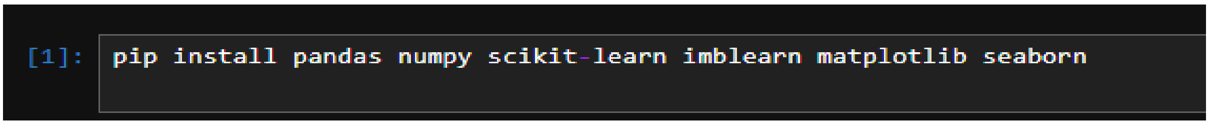
Data Cleaning Process:

Missing Values: Non informative columns (Response, StratificationCategory2) were dropped. Missing values in critical columns like DataValue were imputed using the **median** to maintain data integrity.
Duplicate Records: Identical rows were removed to ensure data uniqueness.
Cardiovascular Data Filtering: The dataset was filtered to retain only rows related to cardiovascular diseases, reducing the data to 23,393 rows and 18 columns.
Feature Selection: Key demographic factors such as age, sex, race, region and relevant factors were kept for analysis, removing irrelevant features to improve model efficiency.

6 Library and Package Installation

To set up the environment, make sure that the following Python libraries are installed. These are essential for data handling, visualization, model building, and evaluation, Index (2024)
pandas: Data manipulation and analysis.
numpy: Numerical operations on arrays and matrices.
scikit-learn: Machine learning models, preprocessing, and evaluation metrics.
imblearn: Handling imbalanced datasets using SMOTE.
matplotlib: Basic plotting and data visualization.
seaborn: Advanced visualization, especially for heatmaps and pair plots.

Installation

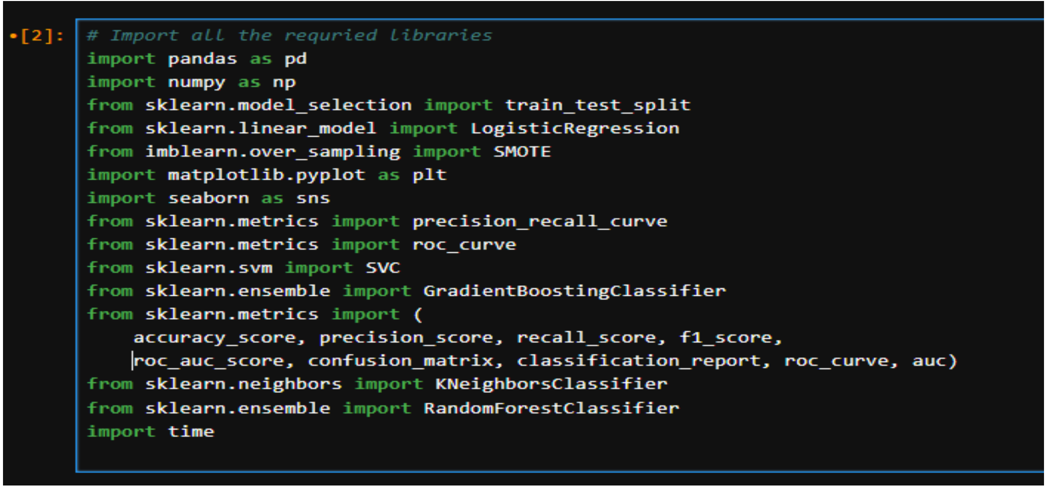


```
[1]: pip install pandas numpy scikit-learn imblearn matplotlib seaborn
```

Figure 3: Installed Commands

Figure 3 shows all the command installed required libraries for the project, ensuring the environment is ready for the machine learning tasks.

Libraries imported



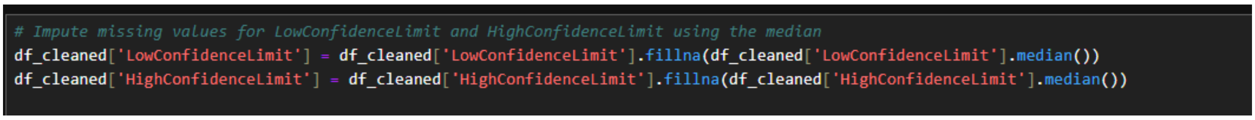
```
•[2]: # Import all the requried libraries
      import pandas as pd
      import numpy as np
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from imblearn.over_sampling import SMOTE
      import matplotlib.pyplot as plt
      import seaborn as sns
      from sklearn.metrics import precision_recall_curve
      from sklearn.metrics import roc_curve
      from sklearn.svm import SVC
      from sklearn.ensemble import GradientBoostingClassifier
      from sklearn.metrics import (
          accuracy_score, precision_score, recall_score, f1_score,
          roc_auc_score, confusion_matrix, classification_report, roc_curve, auc)
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.ensemble import RandomForestClassifier
      import time
```

Figure 4: All the required libraries imported

Figure 4 shows all the libraries required and imported to start with the research

7 Data Cleaning

Handling Missing Values:
As shown in figure 5, columns with more than 50% missing data were dropped (Response, DataValueFootnote) For columns with remaining missing values, such as LowConfidenceLimit and HighConfidenceLimit, median imputation was applied to fill missing entries. Also, Regions were mapped to East or West based on state locations.



```
# Impute missing values for LowConfidenceLimit and HighConfidenceLimit using the median
df_cleaned['LowConfidenceLimit'] = df_cleaned['LowConfidenceLimit'].fillna(df_cleaned['LowConfidenceLimit'].median())
df_cleaned['HighConfidenceLimit'] = df_cleaned['HighConfidenceLimit'].fillna(df_cleaned['HighConfidenceLimit'].median())
```

Figure 5: Input median for missing values

8 Model Implementation

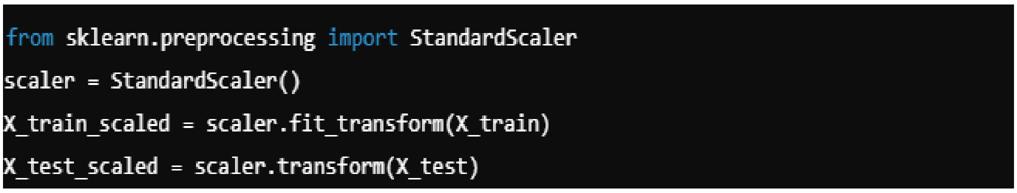
Machine learning model used for this research are listed below

Logistic Regression
Random Forest
Gradient Boosting
Support Vector Machine (SVM)
K-Nearest Neighbors (KNN)

9 Experiments

The dataset was split into different training and test sets such as 75-25, 70-30 and 80-20 split, which means 75% of the data was used for training, and 25% was used for testing and so on. This confirms that the model can generalize well on the unseen data.

Scaling

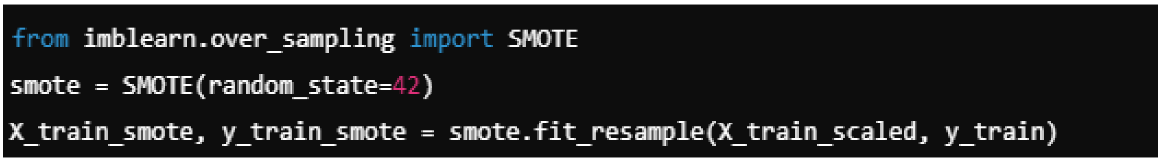


```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Figure 6: Standardization of features

Standardization of features was done using StandardScaler from scikit-learn as shown in the above figure 6

Handling Class Imbalance

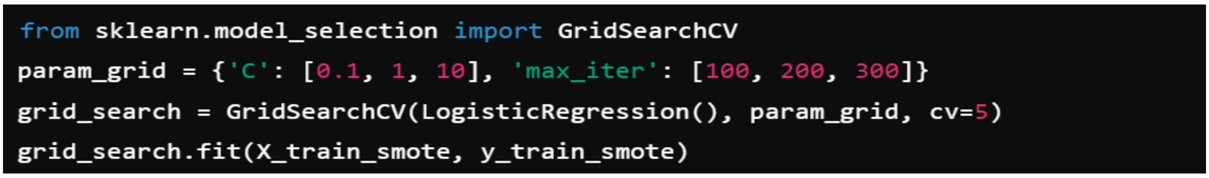


```
from imblearn.over_sampling import SMOTE
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled, y_train)
```

Figure 7: Application of SMOTE Technique

SMOTE (Synthetic Minority Over-sampling Technique) was applied to address class imbalance in the dataset as shown in the above figure 7

Hyperparameter Tuning



```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10], 'max_iter': [100, 200, 300]}
grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5)
grid_search.fit(X_train_smote, y_train_smote)
```

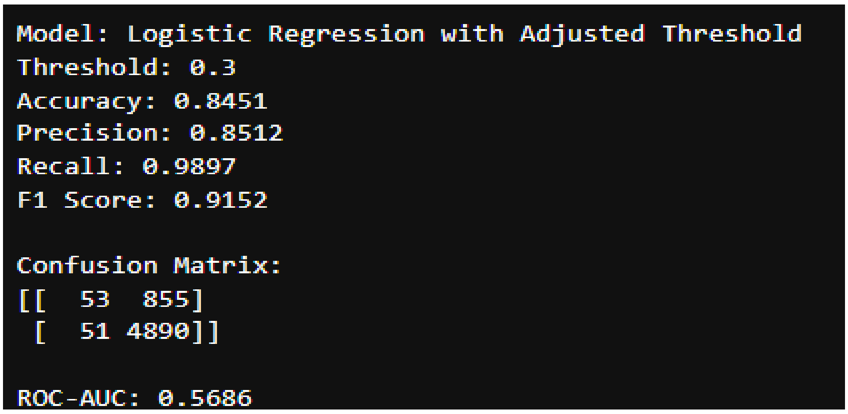
Figure 8: Hyperparameter Tuning of the models

GridSearchCV and RandomizedSearchCV were used to fine-tune the hyperparameters of the models as shown in the above figure 8

10 Results and Statistical Inference

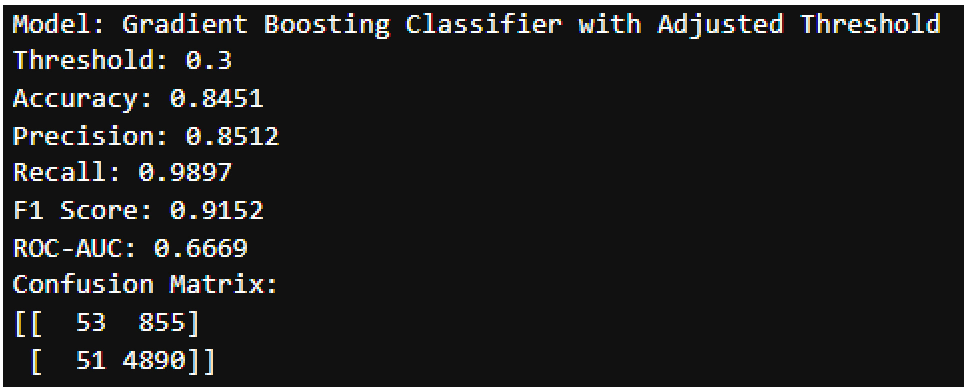
Models were evaluated based on Accuracy, Precision, Recall, F1-Score, and ROC-AUC to measure their classification performance.

The Logistic Regression, Random Forest and Gradient Boosting models performed the best, achieving an accuracy of 84.5% and F1-scores close to 0.91, indicating the best classification performance. The Output of these codes are shown in the Figures 9, 10 & 11



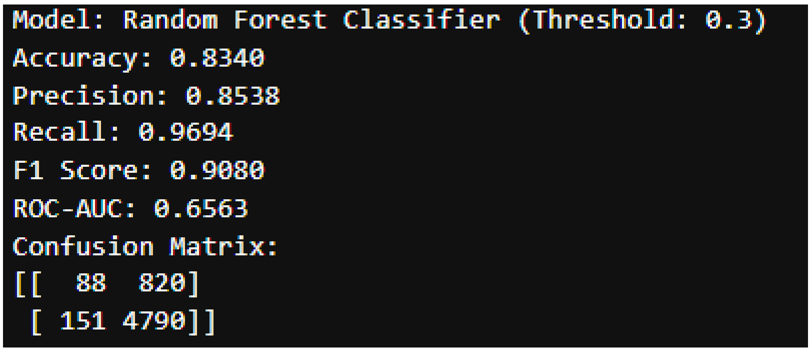
```
Model: Logistic Regression with Adjusted Threshold
Threshold: 0.3
Accuracy: 0.8451
Precision: 0.8512
Recall: 0.9897
F1 Score: 0.9152

Confusion Matrix:
[[  53  855]
 [  51 4890]]

ROC-AUC: 0.5686
```

Figure 9: Output of the Logistic Regression model with Adjusted Threshold



```
Model: Gradient Boosting Classifier with Adjusted Threshold
Threshold: 0.3
Accuracy: 0.8451
Precision: 0.8512
Recall: 0.9897
F1 Score: 0.9152
ROC-AUC: 0.6669
Confusion Matrix:
[[  53  855]
 [  51 4890]]
```

Figure 10: Output of the Gradient Boosting Classifier model with Adjusted Threshold



```
Model: Random Forest Classifier (Threshold: 0.3)
Accuracy: 0.8340
Precision: 0.8538
Recall: 0.9694
F1 Score: 0.9080
ROC-AUC: 0.6563
Confusion Matrix:
[[  88  820]
 [ 151 4790]]
```

Figure 11: Output of the Random Forest Classifier model with Adjusted Threshold

11 Conclusion

In Conclusion, this configuration manual provides the setup guide on how one is able to reproduce the setup, data preparation, and running machine learning models presented in this work. After this, the reader will be easily able to reproduce the experiments and extend the research by trying other models or datasets. The extra code snippets and further statistical analysis can be found in the given Python Notebooks.

References

Index, 2024. *Top 10 Python Libraries For Data Visualization.* [Online]
Available at: https://www.index.dev/blog/top-10-python-libraries-for-data-visualization
[Accessed 9 November 2024].

Services, Department of Health & Human Services, 2024. *U.S. Chronic Disease Indicators.* [Online]
Available at: https://catalog.data.gov/dataset/u-s-chronic-disease-indicators
[Accessed 4 September 2024].