National College of Ireland

Configuration Manual

Final MSc Research Project
Data Analytics

Sanjay Rajendra Raut
Student ID: x22196901

School of Computing
National College of Ireland

Supervisor: Dr. Abid Yaqoob

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sanjay Rajendra Raut
Student ID:	22196901
Programme:	Master's in data Analytics
Year:	2024
Module:	Msc In Research Project
Supervisor:	Dr. Abid Yaqoob
Submission Due Date:	29/01/2025
Project Title:	Personalized Health and Nutrition Recommendations Using Machine Learning
Word Count:	713
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: ……Sanjay Raut……

Date: ……29/01/2025……

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

Sanjay Rajendra Raut
Student ID: x22196901

1 Introduction

This configuration manual allows one to learn how to configure and perform the scripts required to analyze dietary data. This document covers the constraints of the computer system that is needed to execute the workflow, library dependencies, how to get the datasets, how to execute the workflow, and some basic troubleshooting.

Thus, this manual covers the gap between the notebook and the analysis objectives and provides an algorithm on how to repeat the results. Below are elaborated all the software and hardware requirements as well as steps to prepare the dataset.

2 Hardware Requirements

- **Processor**: Minimum Intel Core i5 or AMD equivalent
- **Memory (RAM)**: At least 8 GB (16 GB recommended for larger datasets)
- **Operating System**: Windows 10, macOS, or Linux (64-bit recommended)
- **Storage**: 20 GB of free disk space
- **GPU (optional)**: NVIDIA GPU with at least 4 GB VRAM for faster computation in clustering tasks

3 Software Requirements

Programming Language: Python 3.8 or later
IDE/Environment:

- Jupyter Notebook (via Anaconda Navigator or standalone)
- VS Code (Visual Studio Code)
- PyCharm

Additional Tools:

- A browser for opening Jupyter Notebook
- Git for version control (optional but recommended for collaborative work)

4 Library Package Requirements

The following Python libraries are required to execute the notebook. Use the pip command to install them:
General Libraries

- pandas (Data manipulation and analysis)
- numpy (Numerical computations)
- seaborn (Data visualization)

Machine Learning and Preprocessing

- scikit-learn
- IterativeImputer
- StandardScaler
- KMeans
- PCA (Principal Component Analysis)
- RandomForestClassifier
- LabelEncoder
- train_test_split
- classification_report
- accuracy_score
- GridSearchCV
- catboost
- CatBoostClassifier
- imbalanced-learn
- SMOTE (Synthetic Minority Oversampling Technique)

Deep Learning

- tensorflow
- Sequential
- Dense

Other Utilities

- networkx (Graph-based visualizations)
- matplotlib (Graph plotting)
- joblib (Saving and loading models)

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
import networkx as nx
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import LabelEncoder
from catboost import CatBoostClassifier
from imblearn.over_sampling import SMOTE
from catboost import CatBoostClassifier
from sklearn.model_selection import GridSearchCV
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import joblib
```

Figure 1. Packages Used

5 Dataset Description

A dataset consists of a total of 1,800 observations and 97 independent variables that outlines the various aspects of an individual's diet. It includes some major category wise variables like survey_id, Age in years, Sex, exercise frequency and so on with many more variables

that will also illustrate the key picture. This data set will provide enhanced capability of exploring patterns in age, gender, exercise, and other activities affecting health. It can be rightly aligned to facilitate data preprocessing, EDA and techniques in enhanced modeling. Because of its broad coverage, the dataset is particularly useful in research that deals with the potential relation between diet and lifestyle on the one hand, and health outcomes on the other hand.

The dataset utilized in this study was sourced from the GitHub repository linked to the research paper "Dietary Patterns and the Gut Microbiome" (DOI: 10.1016/j.ajcn.2022.02.001). The dataset discussed here was originally compiled by American Gut Project for gathering information from diverse demographic, lifestyle, and dietary variables. Using this well curated and known dataset, this study takes advantage of secondary data to analyze dietary patterns and their interactions with individual attributes.

6 Dataset Preparation and Pre-processing

The dataset preparation includes reading into Metadata.csv and performing an overview scan for null values and corrupt values. Empty values are managed using IterativeImputer for precision, whereas duplicate records and ambiguous labeling are avoided.

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
```

```
# Find columns with missing values
missing_columns = data.columns[data.isnull().any()].tolist()
missing_columns
```

```
['sample_id_microbiome',
 'SAMPLE_NAME',
 'shannon',
 'observed_otus',
 'chao1',
 'simpson',
 'faith_pd']
```

```
iter_imputer = IterativeImputer(max_iter=10, random_state=0)
```

Figure 2: Imputer Code For Missing Value

StandardScaler evaluates correlation and LabelEncoder standardizes scales for analysis and compatibility. The lack of balance in data is handled by oversampling, using SMOTE, and the feature set is split into train and test data using train_test_split to provide accurate assessment of the model.

7 Applying K-Means Clustering

In this research, the K-Means Clustering algorithm was employed to analyze dietary data and identify two primary groups.
1) Healthy
2)Unhealthy

```
# Applying K-Means clustering (2 clusters for Healthy, Unhealthy)
kmeans = KMeans(n_clusters=2, random_state=42)
clusters = kmeans.fit_predict(dietary_data_imputed)
```

```
# Adding cluster labels to the dataset
data['Diet_Cluster'] = clusters
```

```
from sklearn.ensemble import RandomForestClassifier
dietary_data = data[dietary_columns]
```

```
# Normalize the data for the RandomForestClassifier
scaler = StandardScaler()
dietary_data_scaled = scaler.fit_transform(dietary_data)
```

```
# Train a Random Forest Classifier to estimate feature importance
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(dietary_data_scaled, clusters)
```

```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Figure 3. K-Means Cluster Code

8 Model Building

Applied six Machine Learning models namely; Random Forest, Gradient Boosting, SVM, Neural Networks, CatBoost, and LightGBM. The Random Forest model, after hyperparameter optimization and SMOTE method.

Random Forest

```
# Training a Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100,random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# Making predictions on the test set
y_pred = rf_model.predict(X_test)
```

```
# Evaluating the model
classification_report_output = classification_report(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
```

```
print(classification_report_output)
```

```
              precision    recall  f1-score   support

           0       0.73      0.75      0.74       244
           1       0.45      0.42      0.43       116

    accuracy                           0.64       360
   macro avg       0.59      0.59      0.59       360
weighted avg       0.64      0.64      0.64       360
```

```
accuracy
```

```
0.6444444444444445
```

Figure 4: Random Forest Code

Hyperparameter On Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

```
# Defining parameter grid for Random Forest
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

```
# Performing Grid Search for Random Forest
grid_search_rf = GridSearchCV(
    RandomForestClassifier(random_state=42),
    param_grid_rf,
    cv=5,
    scoring='accuracy',
    n_jobs=-1
)
grid_search_rf.fit(X_train, y_train)
```

```
# Best parameters and performance
best_params_rf = grid_search_rf.best_params_
best_score_rf = grid_search_rf.best_score_
# Evaluating the best model on the test set
best_rf_model = grid_search_rf.best_estimator_
y_pred_best_rf = best_rf_model.predict(X_test)
```

```
classification_report_best_rf = classification_report(y_test, y_pred_best_rf)
accuracy_best_rf = accuracy_score(y_test, y_pred_best_rf)
print(best_params_rf)best_score_rf
```

```
print("Classification Report of Random Forest")
print(classification_report_best_rf)
```

```
Classification Report of Random Forest
              precision    recall  f1-score   support

           0       0.73      0.96      0.83       244
           1       0.76      0.24      0.37       116

    accuracy                           0.73       360
   macro avg       0.74      0.60      0.60       360
weighted avg       0.74      0.73      0.68       360
```

```
print(accuracy_best_rf)
```

```
0.7305555555555555
```

Figure 5: Hyperparameter on Random Forest

Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
# Training a Gradient Boosting Classifier
gb_model = GradientBoostingClassifier(random_state=42)
gb_model.fit(X_train, y_train)
```

```
GradientBoostingClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# Making predictions on the updated test set using Gradient Boosting
y_pred_gb = gb_model.predict(X_test)
```

```
# Evaluating the Gradient Boosting model
classification_report_gb = classification_report(y_test, y_pred_gb)
accuracy_gb = accuracy_score(y_test, y_pred_gb)
```

```
print("Classification Report For Gradient Boost" ,classification_report_gb)
```

```
Classification Report For Gradient Boost               precision    recall  f1-score   support

           0       0.73      0.95      0.83       244
           1       0.73      0.26      0.38       116

    accuracy                           0.73       360
   macro avg       0.73      0.61      0.60       360
weighted avg       0.73      0.73      0.68       360
```

```
print("Accuracy For Gradient Boost", accuracy_gb)
```

```
Accuracy For Gradient Boost 0.7305555555555555
```

Figure 6: Gradient Boosting

XGBoost

```
from xgboost import XGBClassifier
```

```
# Training an XGBoost Classifier
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_state=42)
xgb_model.fit(X_train, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='mlogloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=None,
              n_jobs=None, num_parallel_tree=None, random_state=42, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# Making predictions on the test set using XGBoost
y_pred_xgb = xgb_model.predict(X_test)
```

```
# Evaluating the XGBoost model
classification_report_xgb = classification_report(y_test, y_pred_xgb)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
```

```
print("Classification Report For XGBoost" ,classification_report_xgb)
```

```
Classification Report For XGBoost               precision    recall  f1-score   support

           0       0.73      0.87      0.80       244
           1       0.55      0.34      0.42       116

    accuracy                           0.70       360
   macro avg       0.64      0.60      0.61       360
weighted avg       0.67      0.70      0.67       360
```

```
print("Accuracy For XGBoost", accuracy_xgb)
```

```
Accuracy For XGBoost 0.6972222222222222
```

Figure 7: XGBoost

SVM

```
from sklearn.svm import SVC
```

```
# Training an SVM Classifier
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
```

```
SVC(kernel='linear', random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# Making predictions on the test set using SVM
y_pred_svm = svm_model.predict(X_test)
```

```
# Evaluating the SVM model
classification_report_svm = classification_report(y_test, y_pred_svm)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
```

```
print("Classification Report For SVM" ,classification_report_svm)
```

```
Classification Report For SVM               precision    recall  f1-score   support

           0       0.72      0.99      0.83       244
           1       0.91      0.18      0.30       116

    accuracy                           0.73       360
   macro avg       0.82      0.59      0.57       360
weighted avg       0.78      0.73      0.66       360
```

```
print("Accuracy For SVM", accuracy_svm)
```

```
Accuracy For SVM 0.7305555555555555
```

Figure 8: SVM

Catboost

```
from catboost import CatBoostClassifier
```

```
# Train a CatBoost Classifier
catboost_model = CatBoostClassifier(
    iterations=500,
    learning_rate=0.1,
    depth=6,
    random_state=42,
    verbose=False
)
```

```
# Fit the model on the training data
catboost_model.fit(X_train, y_train)
```

```
<catboost.core.CatBoostClassifier at 0x16ab2c12b70>
```

```
# Make predictions on the test set
y_pred_catboost = catboost_model.predict(X_test)
```

```
# Evaluate the CatBoost model
classification_report_catboost = classification_report(y_test, y_pred_catboost)
```

```
accuracy_catboost = accuracy_score(y_test, y_pred_catboost)
```

```
print("Classification Report of Catboost")
print(classification_report_catboost)
```

```
Classification Report of Catboost
              precision    recall  f1-score   support

           0       0.72      0.87      0.79       244
           1       0.52      0.30      0.38       116

    accuracy                           0.69       360
   macro avg       0.62      0.59      0.59       360
weighted avg       0.66      0.69      0.66       360
```

```
print("Accuracy For Catboost", accuracy_catboost)
```

```
Accuracy For Catboost 0.6861111111111111
```

Figure 9: Catboost

Lightgbm

```
from lightgbm import LGBMClassifier
```

```
# Train a LightGBM Classifier
lgbm_model = LGBMClassifier(
    boosting_type='gbdt',
    num_leaves=31,
    learning_rate=0.1,
    n_estimators=100,
    random_state=42
)
```

```
# Fit the model on the training data
lgbm_model.fit(X_train, y_train)
```

```
[LightGBM] [Info] Number of positive: 462, number of negative: 978
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.003275 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 83
[LightGBM] [Info] Number of data points in the train set: 1440, number of used features: 9
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.320833 -> initscore=-0.749945
[LightGBM] [Info] Start training from score -0.749945
LGBMClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
# Make predictions on the test set
y_pred_lgbm = lgbm_model.predict(X_test)
```

```
# Evaluate the LightGBM model
classification_report_lgbm = classification_report(y_test, y_pred_lgbm)
```

```
accuracy_lgbm = accuracy_score(y_test, y_pred_lgbm)
```

```
print("Classification Report For LightBGM")
print(classification_report_lgbm)
```

```
Classification Report For LightBGM
              precision    recall  f1-score   support

           0       0.73      0.90      0.80       244
           1       0.58      0.28      0.38       116

    accuracy                           0.70       360
   macro avg       0.65      0.59      0.59       360
weighted avg       0.68      0.70      0.67       360
```

```
print("Accuracy", accuracy_lgbm)
```

```
Accuracy 0.7027777777777777
```

Figure 10: Lightbgm

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Build the model
model = Sequential([
    Dense(64, activation='relu', input_dim=X_train.shape[1]),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(len(y.unique()), activation='softmax')  # Output layer
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)

# Evaluate on test set
loss, accuracy = model.evaluate(X_test, y_test)
print("Neural Network Accuracy:", accuracy)
```

Figure 11: Neural Network

Saving the best Random Forest model from GridSearchCV

```
import joblib

# Save the best Random Forest model from GridSearchCV
model_filename = 'best_rf_model.pkl'
joblib.dump(best_rf_model, model_filename)

print(f"Model saved as '{model_filename}'")
```

```
Model saved as 'best_rf_model.pkl'
```

```
# Load the saved model
loaded_model = joblib.load('best_rf_model.pkl')

# Example prediction
y_pred_example = loaded_model.predict(X_test)
print("Example Predictions:", y_pred_example)
```

```
Example Predictions: [0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
 1 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1
 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
```

Figure 12: Saving Final Model

9 Troubleshooting

Library Not Found: Missing libraries need to be installed using the pip install and follow this syntax command**.**

<library_name>command.

File Path Errors: Make sure that the dataset file is located in right path**.**

Runtime Errors: Debug by commenting out a portion of code, identifying a particular cell which is causing issues, or use of the print statement.