Configuration Manual

MSc Research Project
Masters of Science in Data Analytics

Atharva Vyawahare
Student ID: 22212523

School of Computing
National College of Ireland

Supervisor: Vikas Tomer

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student	Atharva Vyawahare **Name:**
Student ID:	22212523
Programme:	Masters of Science in Data Analytics **Year:** 2024
Module:	MSc Research Project
Lecturer:	Vikas Tomar
Submission Due Date:	16/09/2024
Project Title:	Exploring the Economic and social Aspects of Youth Smoking: A Multi-dimensional Analysis

Word Count: 649 **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: Atharva Vyawahare

Date: 12/08/2024

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

Atharva Vyawahare
Student ID: x22212523

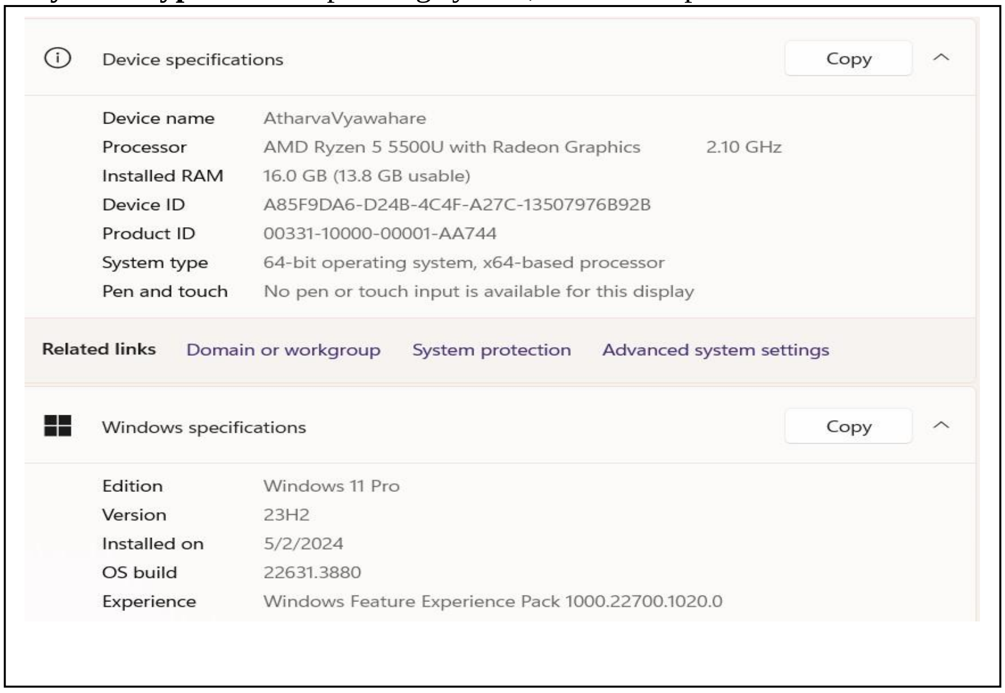
1. Introduction

This configuration manual provides a well detail instruction for setting up, configuring, and executing the analysis on youth tobacco use.he document covers the necessary hardware and software requirements, package installations, dataset details, model preparation, and the steps required to replicate the project's results, in terms of designed to work on the system configuration provided.

2. Hardware Requirements

Local machine: The project was developed and tested on the following hardware configuration (Figure 1).

- **Processor:** AMD Ryzen 5 5500U with Radeon Graphics, 2.10 GHz
- **RAM:** 16.0 GB
- **System Type:** 64-bit operating system, x64-based processor



Device specifications

Device name	AtharvaVyawahare
Processor	AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz
Installed RAM	16.0 GB (13.8 GB usable)
Device ID	A85F9DA6-D24B-4C4F-A27C-13507976B92B
Product ID	00331-10000-00001-AA744
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications

Edition	Windows 11 Pro
Version	23H2
Installed on	5/2/2024
OS build	22631.3880
Experience	Windows Feature Experience Pack 1000.22700.1020.0

Figure 1 Hardware Requirements

This configuration was sufficient to handle the dataset processing, model training, and evaluation tasks required for the project

3 Software Requirement

The project was implemented using the following software:

- Operating System: Windows 10 64-bit was used for a stable and compatible environment for running Python-based applications.
- Programming Language: Python 3.8 was selected for its robust libraries and its use in data analysis and machine learning.
- Integrated Development Environment (IDE): Jupyter Notebook (Figure 2) was utilized for its interactive coding environment, for step-by-step analysis and visualization.

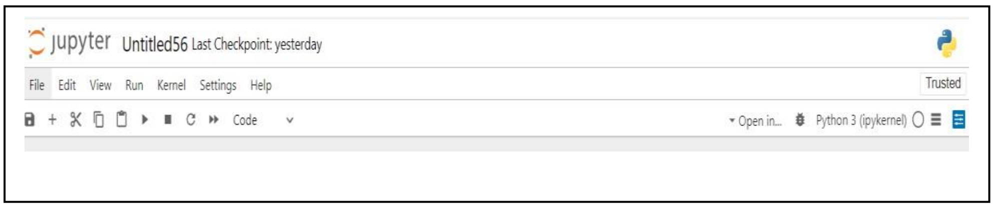
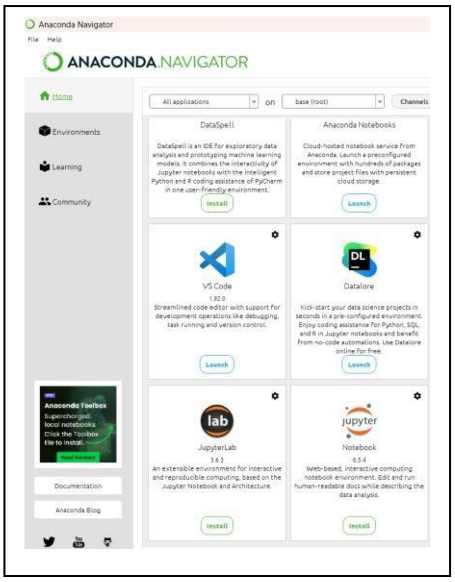


Figure 2 Jupyter Notebook interface

- Navigator: Anaconda Navigator was employed to manage packages, environments, and the Jupyter Notebook interface seamlessly (Figure 3).



DataSpell

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

Anaconda Notebooks

Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

VS Code

1.92.0

Streamlined code editor with support for development operations like debugging, task running and version control.

Datalore

Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.

JupyterLab

3.6.3

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook

6.5.4

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Figure 3 Anaconda Navigator

4. Package Requirements

The project utilizes several Python libraries for data processing, visualization, and machine learning. These packages can be installed via pip:

- pandas**:** For data manipulation and analysis.
- numpy**:** For numerical computations.
- scikit-learn: For machine learning model development.
- xgboost**:** For implementing the XGBoost algorithm.
- matplotlib**:** For creating visualizations.
- seaborn**:** For statistical data visualization.
- shap**:** For model interpretation using SHAP values.
- lime**:** For model interpretation using LIME.
- geopandas**:** For geospatial data analysis and visualization.

5. Data preparation

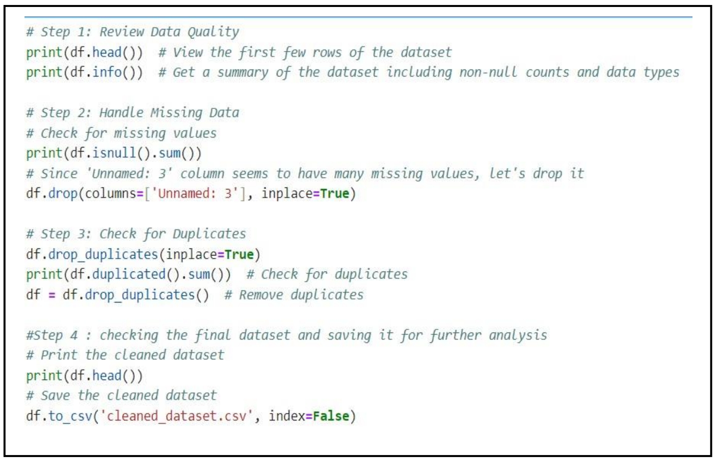
The analysis was conducted using three main datasets:

1. Youth Tobacco Survey (YTS): Data on youth tobacco use behaviors. YTS DATASET LINK

2. Behavioral Risk Factor Surveillance System (BRFSS): Health-related risk behaviors data.Behavioral Risk Factor Data Link

3. Annual Social and Economic Supplements (ASEC): Household income and socioeconomic factors data. Annual Social and Economic Supplements Data Link

These datasets were merged to create dataset for analysis. After merging, the dataset had few data cleaning steps.
Review Data Quality (Figure4):



```
# Step 1: Review Data Quality
print(df.head())  # View the first few rows of the dataset
print(df.info())  # Get a summary of the dataset including non-null counts and data types

# Step 2: Handle Missing Data
# Check for missing values
print(df.isnull().sum())
# Since 'Unnamed: 3' column seems to have many missing values, let's drop it
df.drop(columns=['Unnamed: 3'], inplace=True)

# Step 3: Check for Duplicates
df.drop_duplicates(inplace=True)
print(df.duplicated().sum())  # Check for duplicates
df = df.drop_duplicates()  # Remove duplicates

#Step 4 : checking the final dataset and saving it for further analysis
# Print the cleaned dataset
print(df.head())
# Save the cleaned dataset
df.to_csv('cleaned_dataset.csv', index=False)
```

Figure 4 Data Preparation

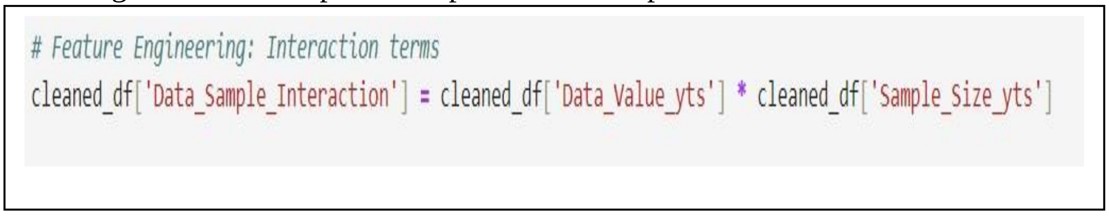
The first few rows and a summary of the dataset were reviewed to understand its structure, data types, and non-null counts.
Handle Missing Data:
Columns with significant missing data, such as 'Unnamed: 3', were identified and removed to maintain data integrity.

Check for Duplicates:
Duplicate rows were checked and removed to ensure that each data point was unique and contributed to the analysis.
Final Dataset Preparation:

The cleaned dataset was reviewed, saved in CSV format as cleaned_dataset.csv, and used for further analysis.
These steps ensured that the dataset was reliable and ready for subsequent machine learning model development and analysis. The cleaned dataset provided a robust foundation for the exploration of factors influencing youth tobacco use.

6. Feature Engineering

Creating Interaction Terms **:** New features were generated (Figure 5) by combining existing variables to capture complex relationships within the data.



```
# Feature Engineering: Interaction terms
cleaned_df['Data_Sample_Interaction'] = cleaned_df['Data_Value_yts'] * cleaned_df['Sample_Size_yts']
```

Figure 5 Feature Engineering

7. Model Implementation

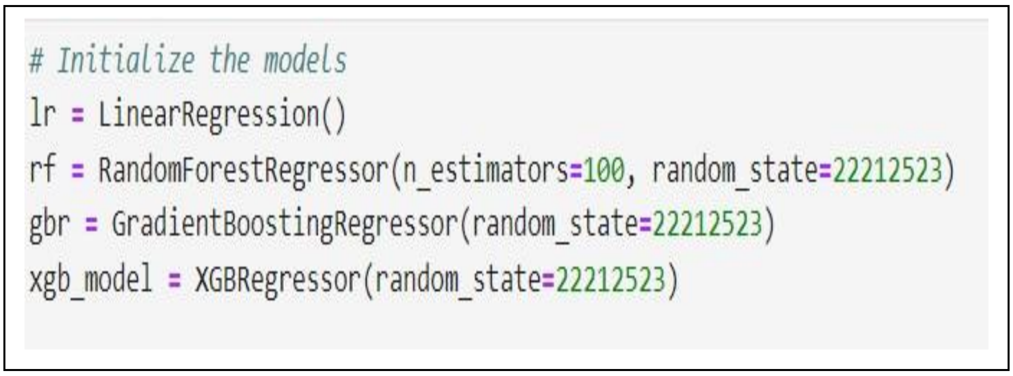
Four machine learning models(Figure 6) were implemented using Python's robust machine learning and libraries:

Linear Regression:
Library Used: Implemented using scikit-learn's `LinearRegression` class.

Random Forest Regressor:
Library Used: Implemented using scikit-learn's `RandomForestRegressor`.

Gradient Boosting Regressor:
Library Used: Implemented using scikit-learn's `GradientBoostingRegressor`.

XGBoost
Library Used: Implemented using the XGBoost library, specifically the `XGBRegressor` class.

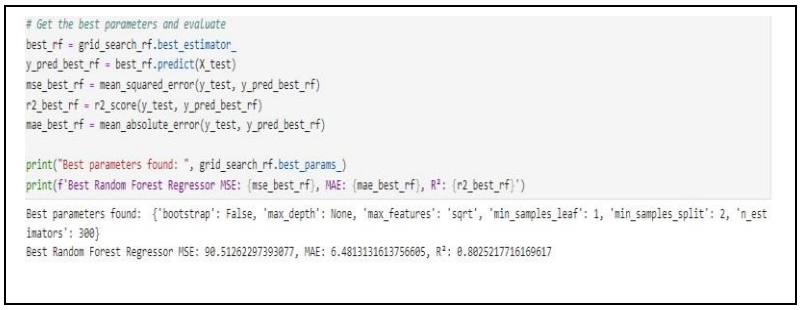


```
# Initialize the models
lr = LinearRegression()
rf = RandomForestRegressor(n_estimators=100, random_state=22212523)
gbr = GradientBoostingRegressor(random_state=22212523)
xgb_model = XGBRegressor(random_state=22212523)
```

Figure 6 Model Implementation

8. Model Tuning and Evaluation

Hyperparameter Tuning: GridSearchCV was employed for Random Forest regressor (Figure7), and RandomizedSearchCV for XGBoost to optimize model performance (Figure 8).

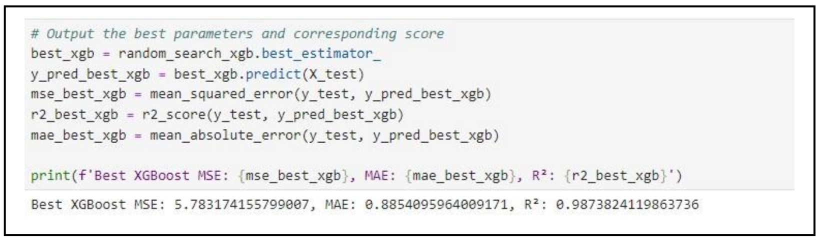


```
# Get the best parameters and evaluate
best_rf = grid_search_rf.best_estimator_
y_pred_best_rf = best_rf.predict(X_test)
mse_best_rf = mean_squared_error(y_test, y_pred_best_rf)
r2_best_rf = r2_score(y_test, y_pred_best_rf)
mae_best_rf = mean_absolute_error(y_test, y_pred_best_rf)

print("Best parameters found: ", grid_search_rf.best_params_)
print(f'Best Random Forest Regressor MSE: {mse_best_rf}, MAE: {mae_best_rf}, R²: {r2_best_rf}')
```

```
Best parameters found:  {'bootstrap': False, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_est
imators': 300}
Best Random Forest Regressor MSE: 90.51262297393077, MAE: 6.4813131613756605, R²: 0.8025217716169617
```

Figure 7 GridSearchCV



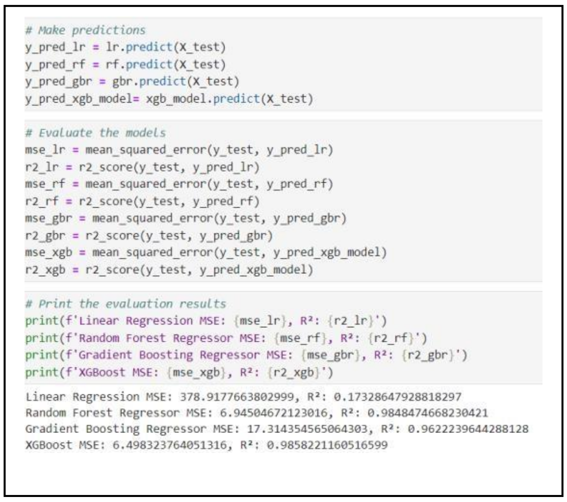
```
# Output the best parameters and corresponding score
best_xgb = random_search_xgb.best_estimator_
y_pred_best_xgb = best_xgb.predict(X_test)
mse_best_xgb = mean_squared_error(y_test, y_pred_best_xgb)
r2_best_xgb = r2_score(y_test, y_pred_best_xgb)
mae_best_xgb = mean_absolute_error(y_test, y_pred_best_xgb)

print(f'Best XGBoost MSE: {mse_best_xgb}, MAE: {mae_best_xgb}, R²: {r2_best_xgb}')
```

```
Best XGBoost MSE: 5.783174155799007, MAE: 0.8854095964009171, R²: 0.9873824119863736
```

Figure 8 RandomizedSearchCV

Model Evaluation: Evaluate the models using MSE, MAE, and R²(Figure 9) metrics, and interpret the results using SHAP(figure 10) and LIME (Figure 11)



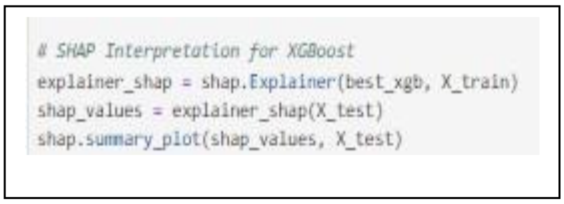
```
# Make predictions
y_pred_lr = lr.predict(X_test)
y_pred_rf = rf.predict(X_test)
y_pred_gbr = gbr.predict(X_test)
y_pred_xgb_model= xgb_model.predict(X_test)
```

```
# Evaluate the models
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
mse_gbr = mean_squared_error(y_test, y_pred_gbr)
r2_gbr = r2_score(y_test, y_pred_gbr)
mse_xgb = mean_squared_error(y_test, y_pred_xgb_model)
r2_xgb = r2_score(y_test, y_pred_xgb_model)
```

```
# Print the evaluation results
print(f'Linear Regression MSE: {mse_lr}, R²: {r2_lr}')
print(f'Random Forest Regressor MSE: {mse_rf}, R²: {r2_rf}')
print(f'Gradient Boosting Regressor MSE: {mse_gbr}, R²: {r2_gbr}')
print(f'XGBoost MSE: {mse_xgb}, R²: {r2_xgb}')
```

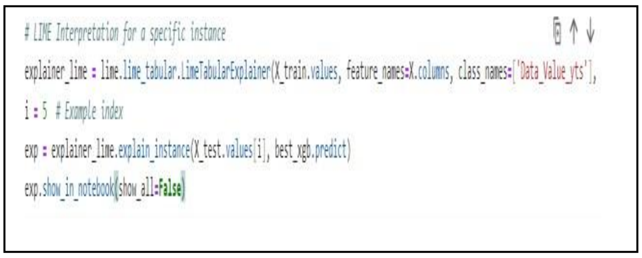
```
Linear Regression MSE: 378.9177663802999, R²: 0.17328647928818297
Random Forest Regressor MSE: 6.94504672123016, R²: 0.9848474668230421
Gradient Boosting Regressor MSE: 17.314354565064303, R²: 0.9622239644288128
XGBoost MSE: 6.498323764051316, R²: 0.9858221160516599
```

Figure 9 Evolution of MSE, MAE, and R²



```
# SHAP Interpretation for XGBoost
explainer_shap = shap.Explainer(best_xgb, X_train)
shap_values = explainer_shap(X_test)
shap.summary_plot(shap_values, X_test)
```

Figure 10 SHAP Analysis



```
# LIME Interpretation for a specific instance
explainer_lime = lime.lime_tabular.LimeTabularExplainer(X_train.values, feature_names=X.columns, class_names=['Data_Value_yts'],
i = 5  # Example index
exp = explainer_lime.explain_instance(X_test.values[i], best_xgb.predict)
exp.show_in_notebook(show_all=False)
```

Figure 11 Lime Analysis