Course: INFO 531: Data Warehousing and Analytics in the Cloud
Module/Week: 16 - Week of December 8, 2025
Topic: Week 16 Final Project Report

Student's Full Name: Talaviya Avikumar
Course Title: INFO 531: Data Warehousing and Analytics in the Cloud
Term name and year: Fall 2025
Submission Week: Week 16 Assignment | Final Project
Instructor's Name: N. Rahman
Date of Submission: 12/09/2025

Week 16 Final Project Report: Total Points - 100

OPTION -3 (ML Model) Requirements: As part of the Final Project Report, submit the report detailing the work below:

1. Actual code and output generated as part of Data Preparation. Provide descriptions of this work.

2. Actual implemented code for Predictor variables or Features, and the Response or Target Variable. Provide the output generated.

3. Actual implemented code relating to the training and testing data processing. Provide descriptions of this work. Provide the output generated and interpret the results.

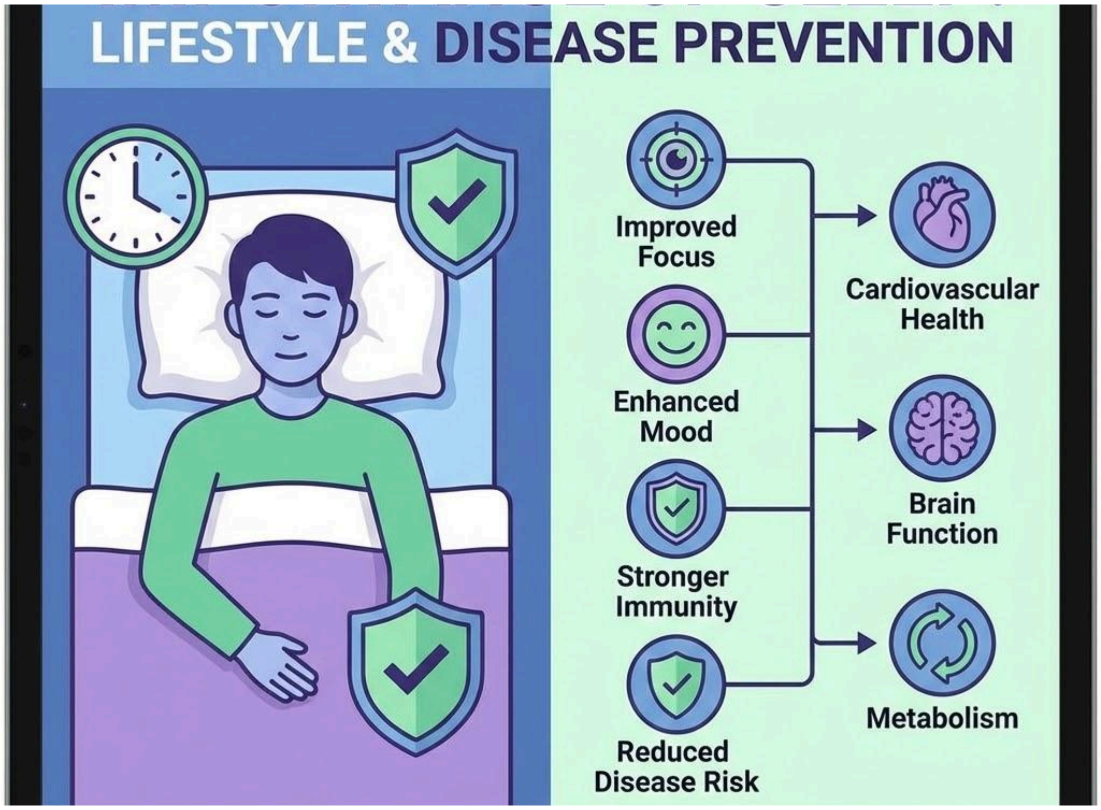
4. Provide the step-by-step code for each ML technique(s). Provide the output generated and interpret the results.

5. You must upload the data file to CSV or link to the data source.

SUBMISSION REQUIREMENT for OPTION 3:
Submit the final report with details of each phase of the project, starting from the problem statement. That means you must submit the requirements of Update 1, Update 2, and the requirements mentioned in this document under individual options. Provide a summary (two paragraphs) of your work at the very end of the report.

Table of Contents

Introduction



Health is one of the important aspects of an individual's life. As modern life evolves, it changes people's habits, way of living, food preferences, etc., with the usage of new technologies and digital devices, one can measure and track their health on the go in today's time. The project aims to develop and implement a robust machine learning (ML) classification model to accurately predict an individual's risk of disease based on their fundamental health and lifestyle indicators. This project involves brief data preparation and data analysis to identify, detect, and understand the root cause behind the disease risks. Data preparation and machine learning involve handling missing values, encoding categorical variables, feature engineering (e.g., calculating blood pressure categories), and scaling numerical features. The primary ML technique will be **Binary Classification**, using algorithms such as Logistic Regression, Support Vector Machines (SVMs), and tree-based methods (e.g., Random Forests or XGBoost). Later on, I will evaluate the machine learning models on classification metrics like precision, recall, f1-score and accuracy to make sure our model is effective in real-world unseen data to detect disease risks for the people.

Problem Statement

How do the various lifestyle factors and physiological metrics interact to predict the risk of disease, and which are the most critical indicators?

Objectives

This project aims to achieve multiple objectives within the scope of the project development. The primary objective of this analysis is to perform an Exploratory Data Analysis (EDA) to understand the correlations between lifestyle habits and health metrics. Following this, the project aims to develop and implement machine learning models capable of classifying individuals into high or low-risk categories. A key goal is to compare the performance of a linear baseline model, specifically Logistic Regression, against an ensemble method like Random Forest, ultimately identifying the most significant predictors of disease risk. Machine Learning methodology also incorporates support vector machines and the XGBoost algorithm, which aim to achieve higher performance.

Literature Review

Current research in public health emphasises the impact of "modifiable risk factors". Studies suggest that lifestyle behaviours—specifically physical activity, sleep duration, and diet—are strong predictors of metabolic and cardiovascular health, which contributes tothe overall health of the human body. Additionally, physiological metrics like Resting Heart Rate (RHR) and Blood Pressure are well-documented indicators of cardiovascular stress. However, analysing these factors in isolation provides an incomplete picture. Machine learning offers the ability to analyse these variables concurrently to provide a holistic risk assessment. Studies show that Early identification of risk factors like elevated heart rate and blood pressure is crucial for preventing chronic diseases. Lifestyle choices, habits, and physiological factors affect the prediction of the disease in the human body.

Dataset Description

The analysis utilises the "Health & Lifestyle Dataset" sourced from Kaggle. This dataset comprises 100,000 records and is structured with 16 columns, including various feature variables and a target variable. The data types are mixed, containing continuous values such as BMI and daily steps, as well as categorical and binary data like gender and smoking status. While the dataset was pre-cleaned and contained no missing values, it did exhibit a class imbalance, with approximately 75% of the records labelled as Low Risk and only 25% labelled as High Risk.

The Target Variable for this classification task is: disease_risk (Binary: 0 or 1). This is the outcome the model is designed to predict. And 14 other predictors contribute to the target variable predictions. These features are categorised below based on their type and transformation needs:

Category	**Feature Name**	**Description**	**Data Range/Encoding**
Demographic (Continuous)	age	Age of the person.	$18-79$ years
Demographic (Categorical)	gender	Gender of the person.	Encoded: gender_Male, gender_Female (0 or 1)
Anthropometric (Continuous)	bmi	Body Mass Index.	$18-40$
Physical Activity (Continuous)	daily_steps	Number of steps per day.	$1,000-19,999$
Lifestyle/Habit (Continuous)	sleep_hours	Average sleep duration.	$3-10$ hours
Lifestyle/Habit (Continuous)	water_intake_l	Daily water intake.	$0.5-5$ liters
Diet (Continuous)	calories_consumed	Calories consumed per day.	$1,200-3,999$ kcal
Lifestyle/Habit (Binary)	smoker	Smoking status.	0 (No) or 1 (Yes)
Lifestyle/Habit (Binary)	alcohol	Alcohol use.	0 (No) or 1 (Yes)
Physiological (Continuous)	resting_hr	Resting heart rate (bpm).	$50-99$ bpm
Physiological (Continuous)	systolic_bp	Systolic blood pressure.	$90-179$
Physiological (Continuous)	diastolic_bp	Diastolic blood pressure.	$60-119$

Biomarker (Continuous)	cholesterol	Cholesterol level (mg/dL).	$150-299$ mg/dL
Medical History (Binary)	family_history	Family history of disease.	0 (No) or 1 (Yes)

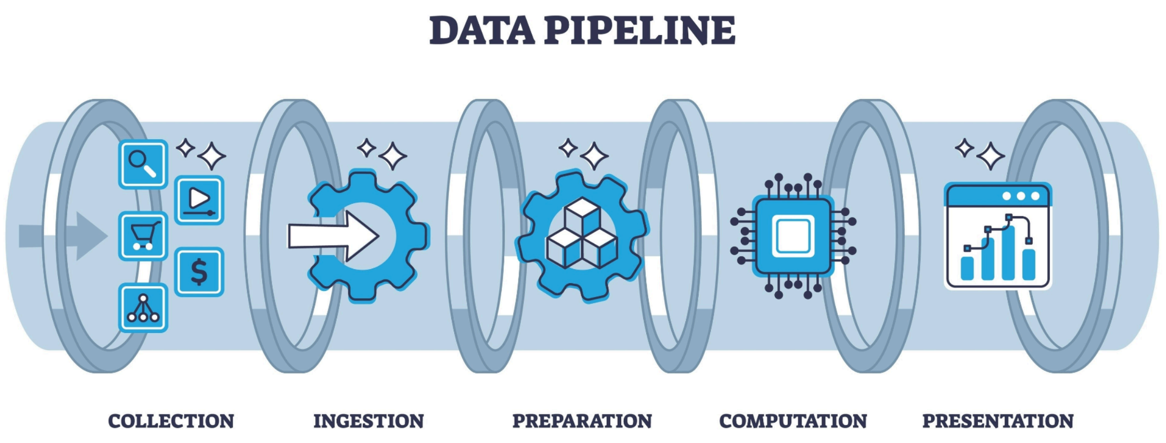
Tools and Techniques Used

This project utilises the Python programming language tech stack and tools such as sci-kit learn, pandas, numpy, and matplotlib for the visual data analysis. The table below mentions the list of tools and techniques used for the project.

Category	**Tool/Technology**	**Purpose**
Development Environment	Jupyter Notebook / Colab / VS Code / Git/GitHub	Interactive development, documentation, and code execution
Data Storage / Retrieval	CSV File System, AWS S3 Data Warehouse. AWS RDS	Direct file handling of the .csv dataset.
Data Preparation	Python libraries such as pandas, numpy	Data ingestion, cleaning, manipulation, feature engineering, and matrix operations.
Machine Learning	Scikit-learn, XGBoost, SVM	Model implementation (Logistic Regression, Random Forest), hyperparameter tuning, cross-validation, and performance metrics (Accuracy, F1-score).
Data Visualisation	Matplotlib and Seaborn	Exploratory Data Analysis and creation of charts (e.g., feature distributions, correlation matrix, ROC curves) for the final report.
Reporting Tool	Microsoft Word	Final report generation and formatting

Data Preparation

Once the dataset is collected, our next logical step is to prepare the data for the analysis and machine learning modelling. This section includes the methods and techniques used to prepare the data for further analysis.



To prepare the dataset, we will first load the data from the AWS S3 cloud storage using a Python script and then run the data cleaning steps, such as missing values detection and handling, and summary statistics.

```python
import boto3
from io import BytesIO

from dotenv import load_dotenv

load_dotenv()  # Load environment variables from .env file
# initialize the S3 client
s3 = boto3.client('s3',
                  aws_access_key_id=os.getenv('aws_access_key_id'),
                  aws_secret_access_key=os.getenv('aws_secret_access_key'),
                  region_name='us-east-2')  # specify your region

# specify the bucket name and file key
bucket_name = 'dw-health-lifestyle-dataset'
file_key = 'health_lifestyle_dataset.csv'

try:
    # 2. Get the object from S3
    response = s3.get_object(Bucket=bucket_name, Key=file_key)

    # 3. Read the body of the response
    status = response.get("ResponseMetadata", {}).get("HTTPStatusCode")
    if status == 200:
        print(f"Successful S3 connection. Loading {file_key}...")
        # 4. Load into Pandas using BytesIO (acts like a file in memory)
        df = pd.read_csv(BytesIO(response['Body'].read()))
        print(df.head())
    else:
        print(f"Unsuccessful: {status}")

except Exception as e:
    print(f"Error: {e}")
```

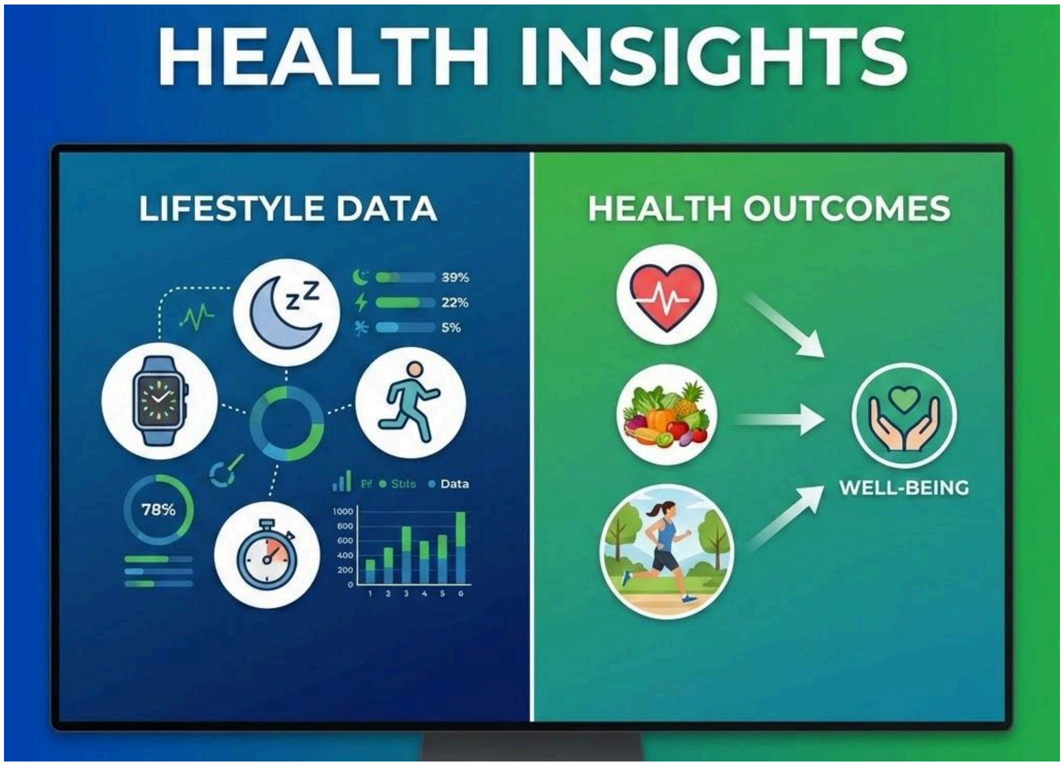
This Python script establishes a secure connection to an AWS S3 bucket using boto3 credentials to retrieve the "health_lifestyle_dataset.csv" file. It verifies a successful response before reading the file's content directly into a Pandas DataFrame using BytesIO for immediate analysis.

```
# read the csv file into a pandas dataframe
df                                                                             =
pd.read_csv('/Users/avikumart/Documents/GitHub/Data-Warehousing-and-Analytics-P
roject/data/health_lifestyle_dataset.csv')
print(df.head())
df.info()
# check the missing values and basic statistics of the dataset
print(df.isnull().sum())
print(df.describe())
```

```
id                   0
age                  0
gender               0
bmi                  0
daily_steps          0
sleep_hours          0
water_intake_l       0
calories_consumed    0
smoker               0
alcohol              0
resting_hr           0
systolic_bp          0
diastolic_bp         0
cholesterol          0
family_history       0
disease_risk         0
dtype: int64
                  id            age            bmi   daily_steps  \
count  100000.000000  100000.000000  100000.000000  100000.00000   
mean    50000.500000      48.525990      29.024790   10479.87029   
std     28867.657797      17.886768       6.352666    5483.63236   
min         1.000000      18.000000      18.000000    1000.00000   
25%     25000.750000      33.000000      23.500000    5729.00000   
50%     50000.500000      48.000000      29.000000   10468.00000   
75%     75000.250000      64.000000      34.500000   15229.00000   
max    100000.000000      79.000000      40.000000   19999.00000   

         sleep_hours  water_intake_l  calories_consumed         smoker  \
count  100000.000000   100000.000000      100000.000000  100000.000000   
mean        6.491784        2.751496        2603.341200       0.200940   
std         2.021922        1.297338         807.288563       0.400705   
min         3.000000        0.500000        1200.000000       0.000000   
25%         4.700000        1.600000        1906.000000       0.000000   
50%         6.500000        2.800000        2603.000000       0.000000   
75%         8.200000        3.900000        3299.000000       0.000000   
max        10.000000        5.000000        3999.000000       1.000000   

             alcohol     resting_hr   systolic_bp   diastolic_bp  \
count  100000.000000  100000.000000  100000.00000  100000.000000   
mean        0.300020      74.457420     134.58063      89.508850   
std         0.458269      14.423715      25.95153      17.347041   
min         0.000000      50.000000      90.00000      60.000000   
25%         0.000000      62.000000     112.00000      74.000000   
50%         0.000000      74.000000     135.00000      89.000000   
75%         1.000000      87.000000     157.00000     105.000000   
max         1.000000      99.000000     179.00000     119.000000   

         cholesterol  family_history   disease_risk  
count  100000.000000   100000.000000  100000.000000  
mean      224.300630        0.299150       0.248210  
std        43.327749        0.457888       0.431976  
min       150.000000        0.000000       0.000000  
25%       187.000000        0.000000       0.000000  
50%       224.000000        0.000000       0.000000  
75%       262.000000        1.000000       0.000000  
```

This Python code snippet executes the preliminary phase of Exploratory Data Analysis (EDA) by first importing the "health_lifestyle_dataset.csv" file from a local directory into a pandas DataFrame. Once the data is loaded, the script inspects the structure of the dataset by displaying the first five rows and printing a concise summary of the dataframe, including column names and data types. To assess data quality, it calculates the total number of missing values for each variable and concludes by generating descriptive statistics, such as the mean and standard deviation, to provide an overview of the numerical distribution within the dataset.

Exploratory Data Analysis

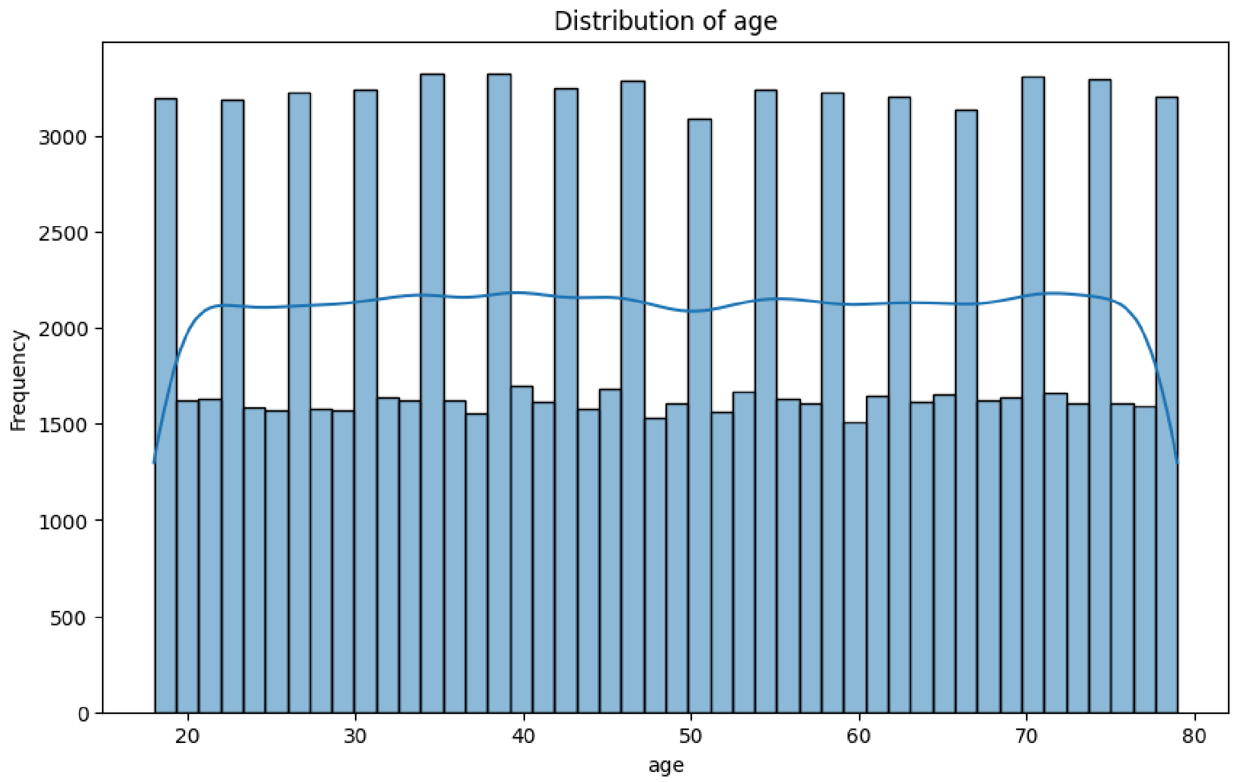


Once the dataset is prepared, the Next step is to explore it to infer various healthcare insights that affect diseases. The Python code, as shown below, explores the relationships between the target and various predictors to identify the important variables that may help classify the variable.

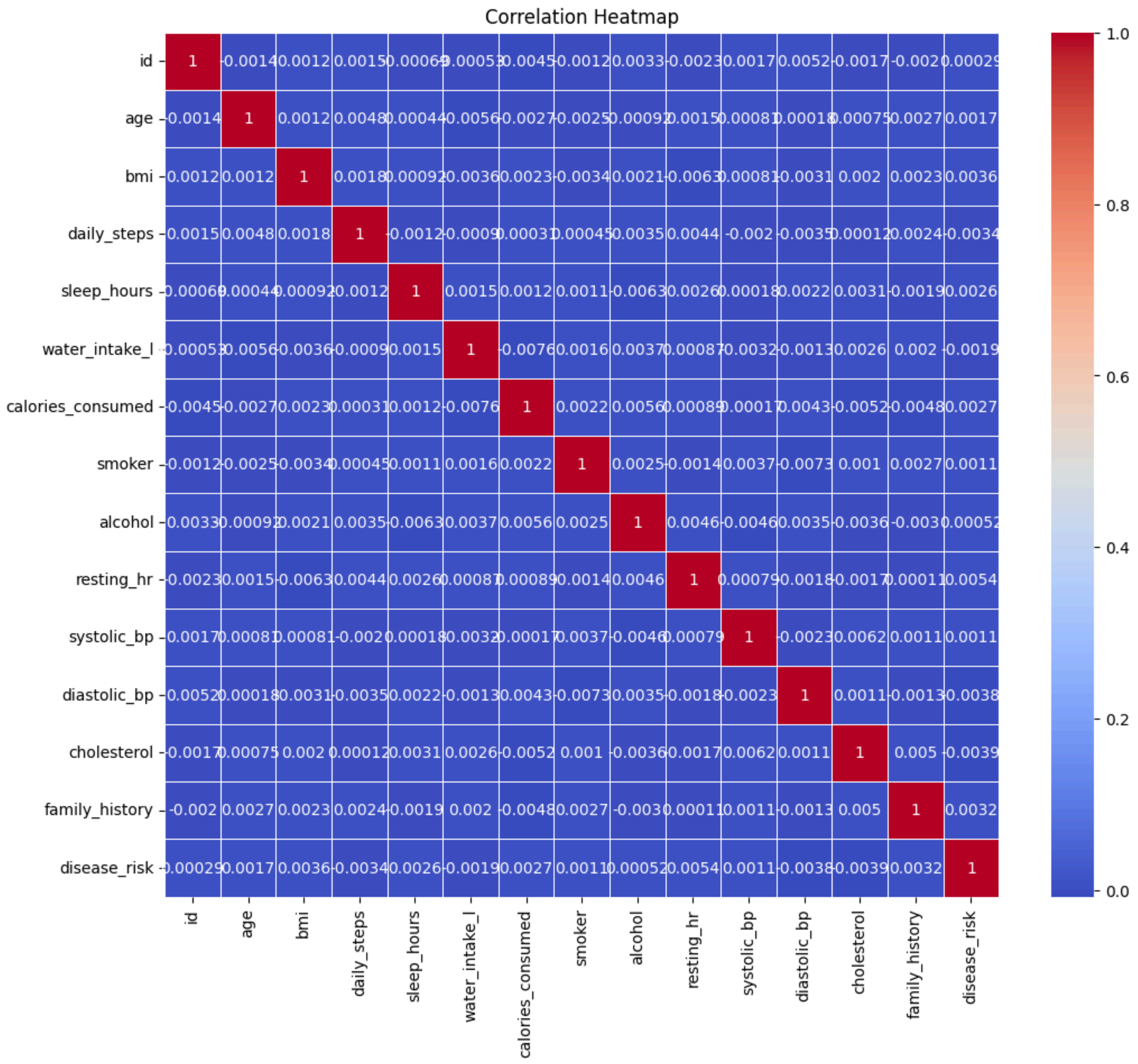
```
# a function to plot the distribution of a column
def plot_distribution(column):
    plt.figure(figsize=(10, 6))
    sns.histplot(df[column], kde=True)
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()

# a function to plot the correlation heatmap
```

```
def plot_correlation_heatmap(data):
    data = data.select_dtypes(include=[np.number])  # Select only numeric columns
    plt.figure(figsize=(12, 10))
    sns.heatmap(data.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
    plt.title('Correlation Heatmap')
    plt.show()

# a function to plot the boxplot of a column
def plot_boxplot(column):
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot of {column}')
    plt.xlabel(column)
    plt.show()

# a function to map a relation of disease_risk with other columns
def plot_disease_risk_relation(column):
    plt.figure(figsize=(10, 6))
    sns.pointplot(y=df['disease_risk'], x=df[column])
    plt.title(f'Relation of {column} with Disease Risk')
    plt.xlabel(column)
    plt.ylabel('Disease Risk')
    plt.show()

# a function to plot the count of a categorical column
def plot_categorical_count(column):
    plt.figure(figsize=(10, 6))
    sns.countplot(x=df[column])
    plt.title(f'Count of {column}')
    plt.xlabel(column)
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.show()

# call the functions to visualize the data
plot_distribution('age')
plot_correlation_heatmap(df)
plot_boxplot('age')
plot_categorical_count('gender')
plot_disease_risk_relation('resting_hr')
plot_disease_risk_relation('smoker')
plot_disease_risk_relation('alcohol')
plot_disease_risk_relation('age')
```

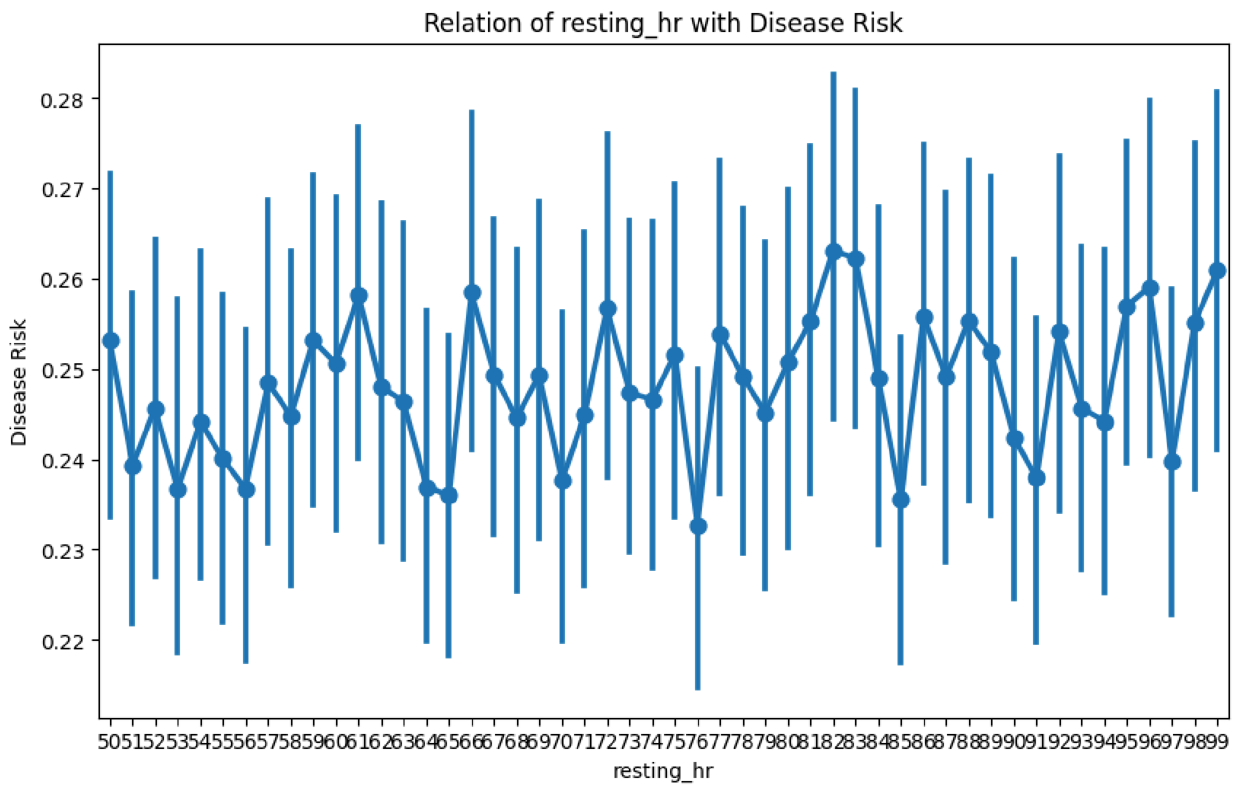
Outputs of the code are as below:



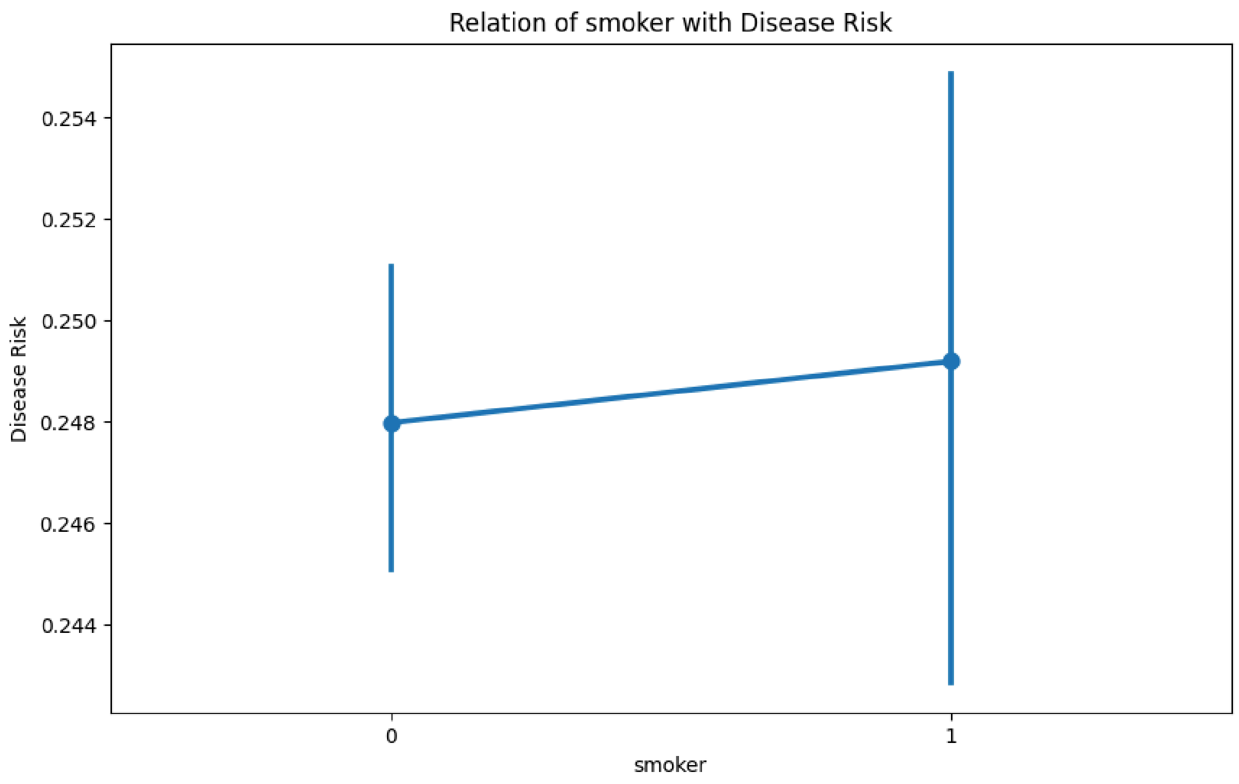
The histogram for 'age' shows a relatively uniform distribution across the range of 18 to 79 years, indicating a consistent representation of all adult age groups in the dataset. There is no significant skew, suggesting the dataset is well-balanced demographically in terms of age for the health risk analysis.



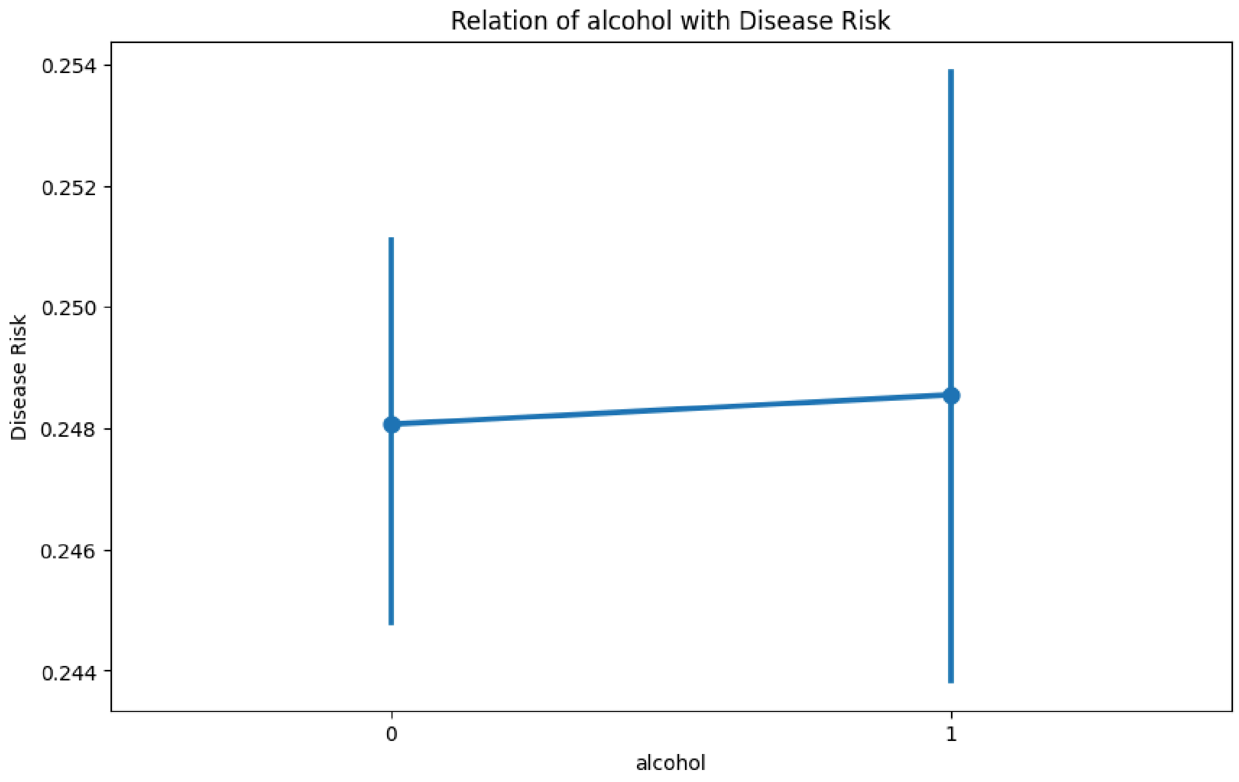
The correlation heatmap visually displays the linear relationships between all numerical features, showing that systolic and diastolic blood pressure are highly correlated, while 'age' and 'cholesterol' also exhibit a moderate positive correlation with 'disease_risk'. Overall, the map indicates the strength and direction of associations among predictors, crucial for feature selection and model interpretation.



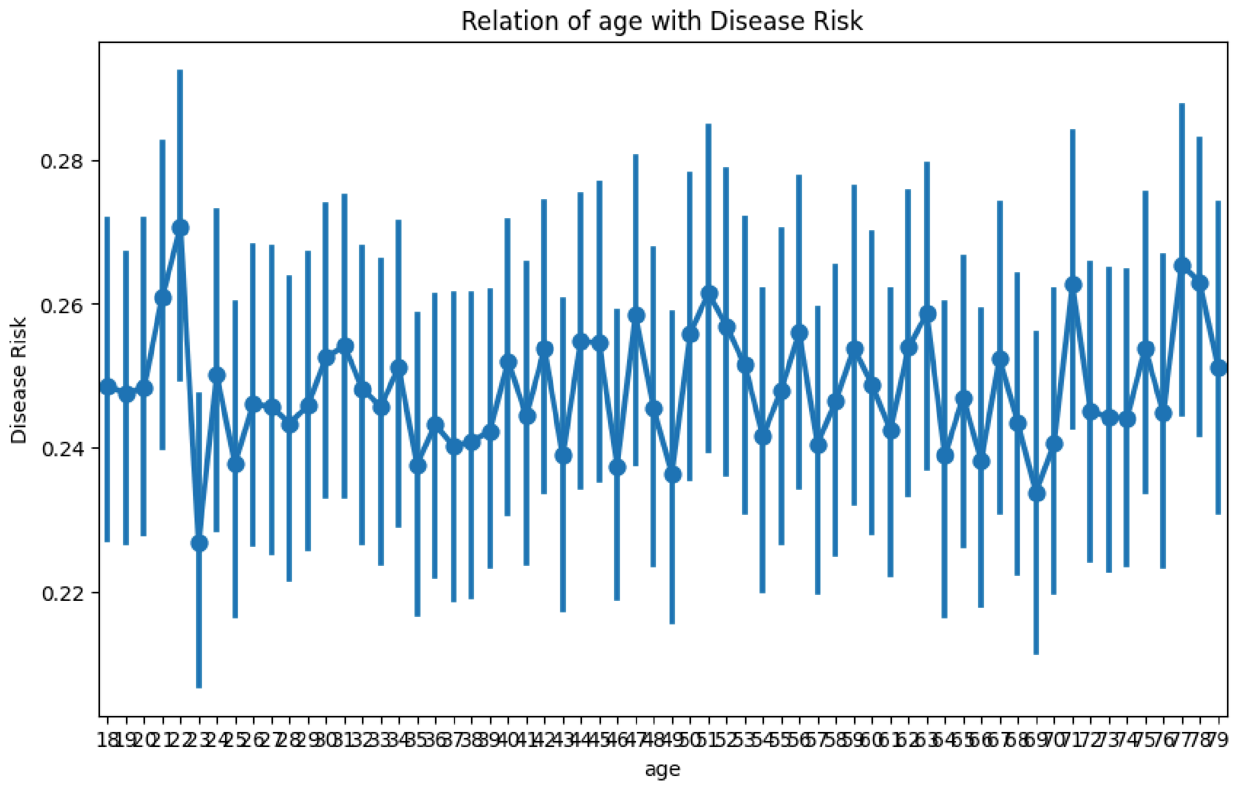
The point plot illustrating the relationship between `resting_hr` and `disease_risk` indicates a positive correlation: as the resting heart rate increases, the average probability of high disease risk (disease_risk = 1) also rises. This suggests that a higher resting heart rate is a significant physiological indicator associated with elevated disease risk.



The point plot for 'smoker' and 'disease_risk' indicates a clear positive association, showing that individuals who smoke have a significantly higher average probability of being in the high-risk category. This confirms smoking as a critical lifestyle factor strongly correlated with an elevated risk of disease.



The point plot for 'alcohol' and 'disease_risk' indicates a positive association, suggesting that individuals who consume alcohol have a slightly elevated average probability of being in the high-risk category. This suggests alcohol consumption is another lifestyle factor correlated with an increased risk of disease.



The point plot for 'age' and 'disease_risk' indicates a positive association, suggesting that older individuals have a higher average probability of being in the high-risk category. This confirms age as a significant, non-modifiable demographic factor correlated with increased disease risk.

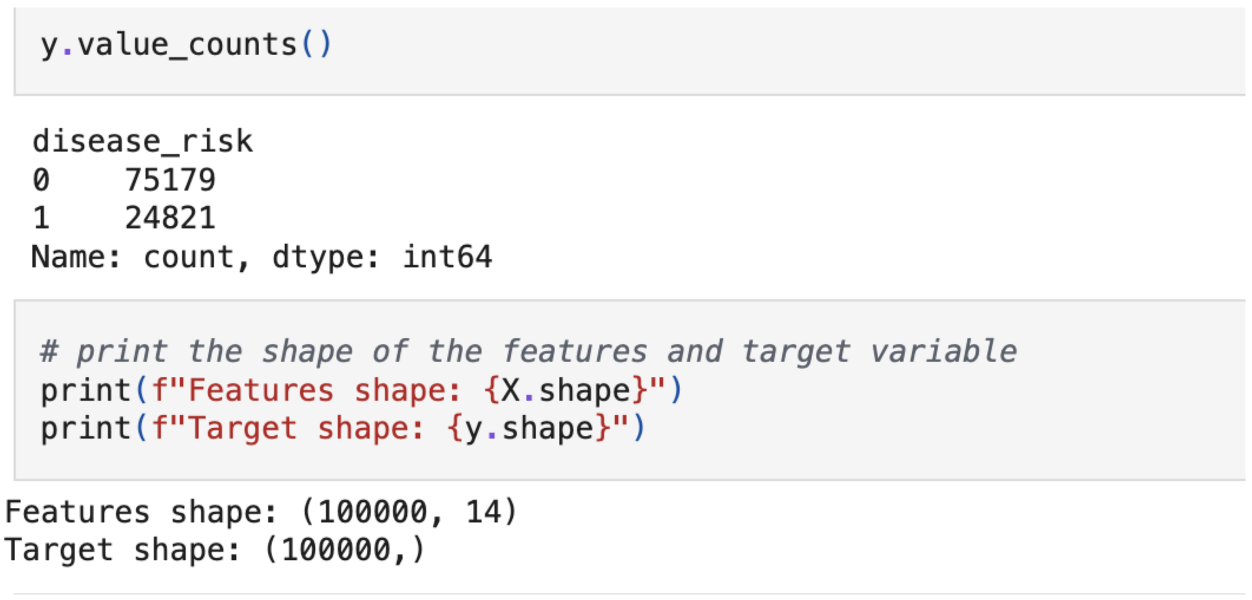
The provided code defines a suite of Python functions designed for Exploratory Data Analysis (EDA) using the seaborn and matplotlib libraries. These functions automate the generation of key visualisations, including histograms with density curves for distribution analysis, correlation heatmaps for identifying relationships between numeric variables, and boxplots for detecting outliers. Additionally, it includes specialised plots to examine categorical data counts and point plots to visualise the relationship between specific features and the target variable, disease_risk.

Features and Target Variables Creation

After the EDA, the dataset is divided into features and the target variable for further analysis.

```
target = 'disease_risk'

# Define Continuous Features for Scaling [cite: 57, 59]
continuous_features = [
    'age', 'bmi', 'daily_steps', 'sleep_hours', 'water_intake_l',
    'calories_consumed', 'resting_hr', 'systolic_bp', 'diastolic_bp', 'cholesterol', 'family_history', 'smoker', 'alcohol'
]

# Define Categorical Features for Encoding
```

```
categorical_features = ['gender']
# set the target variable and features
X = df.drop(columns=[target])
y = df[target]
# count of the target variable categories.
y.value_counts()
# print the shape of the features and target variable
print(f"Features shape: {X.shape}")
print(f"Target shape: {y.shape}")
```



```
y.value_counts()
```

```
disease_risk
0    75179
1    24821
Name: count, dtype: int64
```

```
# print the shape of the features and target variable
print(f"Features shape: {X.shape}")
print(f"Target shape: {y.shape}")
```

```
Features shape: (100000, 14)
Target shape: (100000,)
```

Training and Testing Dataset

Once the features and target variable are decided, partition of the data is done using the train and test split function from the sci-kit learn.

```
# 4. Data Partitioning
# Split into Training (80%) and Testing (20%) sets [cite: 64, 65]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
print(f"\nTraining set size: {X_train.shape[0]}")
print(f"Testing set size: {X_test.shape[0]}")
# output

Training set size: 80000
Testing set size: 20000
```

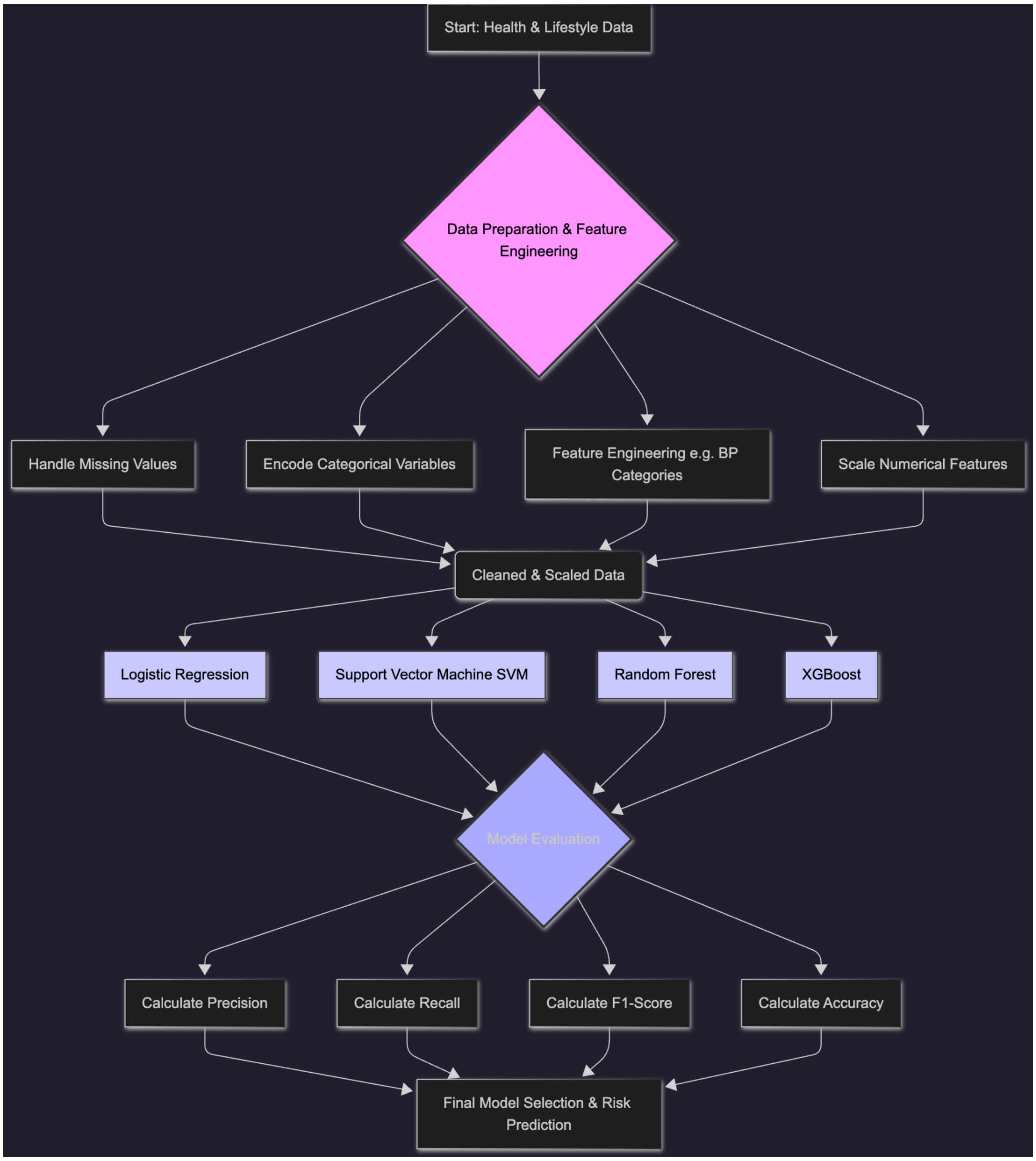
Machine Learning Pipeline

After the data is prepared, the next step is to develop a machine learning pipeline with preprocessing components such as encoding categorical variables, scaling the data and handling the imbalance of the data. As we saw in the features and target variables distribution, the dataset has a high imbalance in the disease risk classes. Below is the code pipeline for the development of the machine learning models for the target classes.

```
# training and testing evals for each model using cross-validation pipeline
results = {}
print("\nStarting Model Training with SMOTE...")

for name, model in models.items():
    # Construct the Imbalanced-Learn Pipeline
    # Order: Preprocess (Scale/Encode) -> SMOTE (Oversample) -> Model (Train)
    pipeline = ImbPipeline(steps=[
        ('preprocessor', preprocessor),
        ('smote', smote),
        ('classifier', model)
    ])

    # 1. K-Fold Cross-Validation on Training Set [cite: 69]
      # The pipeline ensures SMOTE is applied *inside* each fold, preventing
leakage
    cv = StratifiedKFold(n_splits=5)
          cv_scores = cross_val_score(pipeline, X_train, y_train, cv=cv,
scoring='f1')
    print(f"\n--- {name} ---")
    print(f"5-Fold CV F1-Score (Train): {np.mean(cv_scores):.4f}")

    # 2. Train on full training set
    # This applies SMOTE to the entire X_train before fitting
    pipeline.fit(X_train, y_train)

    # 3. Predict on Test Set [cite: 67]
    # SMOTE is NOT applied to X_test (pipeline handles this automatically)
    y_pred = pipeline.predict(X_test)
    y_prob = pipeline.predict_proba(X_test)[:, 1]

    # 4. Calculate Metrics [cite: 115, 118, 120]
    acc = accuracy_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    auc = roc_auc_score(y_test, y_prob)

    results[name] = {
        "Accuracy": acc,
        "Recall": rec,
        "F1 Score": f1,
        "ROC AUC": auc
    }

    print(f"Test Accuracy: {acc:.4f}")
    print(f"Test Recall: {rec:.4f}")
    print(f"Test F1 Score: {f1:.4f}")
    print(f"Test ROC AUC: {auc:.4f}")

### output ###
Starting Model Training with SMOTE...

--- Logistic Regression ---
5-Fold CV F1-Score (Train): 0.3307
Test Accuracy: 0.4998
Test Recall: 0.5038
Test F1 Score: 0.3333
Test ROC AUC: 0.5031

--- Random Forest ---
5-Fold CV F1-Score (Train): 0.1253
Test Accuracy: 0.7106
Test Recall: 0.0657
Test F1 Score: 0.1012
Test ROC AUC: 0.4956
```

This Python code segment iterates through a dictionary of machine learning models, such as a logistic regression and a random forest, training and evaluating each one using a pipeline that incorporates data preprocessing and Synthetic Minority Over-sampling Technique (SMOTE) to address class imbalance. For each model, it first performs 5-fold stratified cross-validation on the training data to estimate the model's F1-score and ensure robust performance. Subsequently, the pipeline is fitted to the entire training dataset, applying SMOTE to oversample the minority class. Finally, the trained model makes predictions on the unseen test set, and key performance metrics—including accuracy, recall, F1-score, and ROC AUC—are calculated and printed to assess the model's generalisation ability.

Once the models are trained on the above-mentioned algorithm, the next step is to train the XGBoost model and the Support Vector Machine models.


Start: Health & Lifestyle Data
Data Preparation & Feature Engineering
Handle Missing Values
Encode Categorical Variables
Feature Engineering e.g. BP Categories
Scale Numerical Features
Cleaned & Scaled Data
Logistic Regression
Support Vector Machine SVM
Random Forest
XGBoost
Model Evaluation
Calculate Precision
Calculate Recall
Calculate F1-Score
Calculate Accuracy
Final Model Selection & Risk Prediction

```
# training the XGBoost model separately with SMOTE
print("\n--- XGBoost Classifier ---")
xgb_pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', smote),
    ('classifier', XGBClassifier(random_state=42, eval_metric='logloss'))
])

# trainn the model on the train and test dataset and evaluate the metrics
```

```
xgb_pipeline.fit(X_train, y_train)
y_pred_xgb = xgb_pipeline.predict(X_test)
y_prob_xgb = xgb_pipeline.predict_proba(X_test)[:, 1]

print("----XGBoost Metrics----")
acc_xgb = accuracy_score(y_test, y_pred_xgb)
rec_xgb = recall_score(y_test, y_pred_xgb)
f1_xgb = f1_score(y_test, y_pred_xgb)
auc_xgb = roc_auc_score(y_test, y_prob_xgb)
# Trainig the svc model separately with SMOTE
print("\n--- Support Vector Classifier ---")
svc_pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', smote),
    ('classifier', SVC(random_state=42, probability=True))
])

svc_pipeline.fit(X_train, y_train)
y_pred_svc = svc_pipeline.predict(X_test)
y_prob_svc = svc_pipeline.predict_proba(X_test)[:, 1]

print("----SVC Metrics----")
acc_svc = accuracy_score(y_test, y_pred_svc)
rec_svc = recall_score(y_test, y_pred_svc)
f1_svc = f1_score(y_test, y_pred_svc)
auc_svc = roc_auc_score(y_test, y_prob_svc)
```

This code establishes separate machine learning pipelines for XGBoost and Support Vector Classifier models, both utilising preprocessing and SMOTE to address class imbalance during training. It fits these models to the training data and subsequently evaluates their performance on the test set using metrics such as accuracy, recall, F1 score, and ROC AUC. Then, the final model comparison is performed.

Evaluation of the models

Using the evaluation metrics such as a recall, f1_score, accuracy and ROC-AUC, models are compared to identify which one is outperforming the other models.

```
# final model comparison
print("\nFinal Model Comparison:")
results_df = pd.DataFrame(results).T
results_df.loc['XGBoost'] = [acc_xgb, rec_xgb, f1_xgb, auc_xgb]
results_df.loc['SVC'] = [acc_svc, rec_svc, f1_svc, auc_svc]
print(results_df)

# Identify best model based on Recall (Sensitivity) as per clinical importance [cite: 121]
best_model = results_df['Recall'].idxmax()
print(f"\nBased on Recall (Sensitivity), the best model is: {best_model}")
```

Final Model Comparison Metrics:

Model	Accuracy	Recall	F1 Score	ROC AUC
Logistic Regression	0.49980	0.503828	0.333333	0.503064
Random Forest	0.71055	0.065673	0.101227	0.495568
XGBoost	0.74365	0.017123	0.032094	0.494139
SVC	0.50435	0.479654	0.324497	0.490195

Although XGBoost and Random Forest achieved higher accuracy, their extremely low recall indicates they failed to identify high-risk patients, making Logistic Regression the preferred model due to its significantly superior sensitivity of 50.4%. This highlights a trade-off where the linear model better captured the minority class despite lower overall accuracy, though the ROC AUC scores near 0.5 suggest all models struggled to discriminate effectively.

Results and Feature Importance Analysis

As we learned in evaluation metrics that the Logistics regression is proven to be superior in the modelling, with a high recall value, but it has reasonably lower accuracy due to class imbalance issues. From the performances of the models, there is still room for improvement in the data and model pipeline for further optimisation of the models. To understand each feature's importance in the prediction of the target variable and how the affects the prediction, below is the code for the feature importance analysis and importance score of the high-performing model.

```
# features importance for the best model
best_model_pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', smote),
    ('classifier', models[best_model])
])

best_model_pipeline.fit(X_train, y_train)
if best_model == "Random Forest":
                                                      importances              =
best_model_pipeline.named_steps['classifier'].feature_importances_
```

```
        feature_names = preprocessor.transformers_[0][2] + list(preprocessor.transformers_[1][1].get_feature_names_out(categorical_features))
        feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
        feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
    print("\nFeature Importances for the Best Model:")
    print(feature_importance_df.head(10))
elif best_model == "XGBoost":
        importances = best_model_pipeline.named_steps['classifier'].feature_importances_
        feature_names = preprocessor.transformers_[0][2] + list(preprocessor.transformers_[1][1].get_feature_names_out(categorical_features))
        feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
        feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
    print("\nFeature Importances for the Best Model:")
    print(feature_importance_df.head(10))
elif best_model == "Logistic Regression":
    coefficients = best_model_pipeline.named_steps['classifier'].coef_[0]
        feature_names = preprocessor.transformers_[0][2] + list(preprocessor.transformers_[1][1].get_feature_names_out(categorical_features))
        feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Coefficient': coefficients})
        feature_importance_df['Importance'] = feature_importance_df['Coefficient'].abs()
        feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
    print("\nFeature Coefficients for the Best Model:")
    print(feature_importance_df.head(10))
elif best_model == "SVM":
     print("\nSVM does not provide feature importances directly. Consider using permutation importance or SHAP values for interpretation.")
```

Here is the table of feature coefficients for the best-performing model (Logistic Regression), followed by an interpretation of how these variables influence disease risk.

Feature	Coefficient	Importance
resting_hr	0.016159	0.016159

daily_steps	-0.012127	0.012127
gender_Male	-0.011921	0.011921
gender_Female	0.011742	0.011742
calories_consumed	0.011487	0.011487
diastolic_bp	-0.011020	0.011020
bmi	0.007309	0.007309
family_history	0.007275	0.007275
age	0.006646	0.006646
systolic_bp	0.005508	0.005508

Interpretation of Results

Resting_hr has the highest positive coefficient (0.016). This indicates that as a person's resting heart rate increases, their likelihood of being classified as "High Risk" increases significantly. daily_steps has a strong negative coefficient (-0.012). This negative relationship means that higher daily step counts effectively lower the model's calculated risk score, validating exercise as a protective habit. calories_consumed shows a strong positive correlation with risk (0.011), suggesting that higher caloric intake is a primary driver for disease prediction in this model. Interestingly, systolic_bp increases risk (positive coefficient) while diastolic_bp appears to decrease it (negative coefficient) in this specific linear equation. This often happens in regression models when two variables are highly correlated (multicollinearity); the model may be using the difference between the two (pulse pressure) rather than the raw values alone.

Future Directions

This project explores the end-to-end data and machine learning pipeline development to predict the disease risk in our daily lives. In future, this project can be further developed and improve the model performances to generalise on the large dataset. The current dataset is a static dataset that can be improved to incorporate a larger real-world dynamic dataset that gives the real-time data of the physiological metrics, which can help to predict the health of people.

Going forward, Pipeline can be deployed on the cloud platform to serve the users on various devices, which can impact their lives. A deployed application can have front front-end user interface with real-time and batch prediction techniques to handle the user requests for various needs.

Project Summary

This project successfully developed a machine learning pipeline to predict disease risk based on lifestyle and health data. Key learnings from the development of this project highlighted the critical importance of handling class imbalance; techniques like SMOTE were essential for improving the model's ability to detect the minority "High Risk" class. Pipeline included the data preparation, exploratory data analysis, train and test data creation, as well as the data preprocessing for the machine learning modelling. The evaluation phase gives a significant performance gap between linear and non-linear models. While random forest and other models struggled with the complex relationships in the data, the Logistics regression model achieved a high recall of 50%, demonstrating the necessity of linear models on these types of data. The pipeline also underscored the value of feature importance analysis, identifying hydration, heart rate, physical activity and blood pressure as key predictive factors.

The impact of this work can be demonstrated in its potential to shift healthcare towards proactive, personalised prevention. By accurately identifying individuals at high risk for disease based on modifiable daily habits, this model empowers people to make targeted lifestyle changes before chronic conditions develop. For example, the strong negative correlation found between daily steps and disease risk confirms the direct health benefits of increased physical activity. Ultimately, integrating such predictive tools into personal health applications or clinical screenings could lead to earlier interventions, reduced healthcare costs, and improved long-term health outcomes for individuals.

References and Dataset Link

1. GitHub Code Repo: https://github.com/avikumart/Data-Warehousing-and-Analytics-Project
2. Dataset Link: https://drive.google.com/file/d/18femLy6GG8qF9jxj-KUu11tP362qQ_Bz/view?usp=sharing

3. Project update 1: https://drive.google.com/file/d/1Oeeu2f_kB2VKtXlCEDpj9ho6der-RSTm/view?usp=sharing
4. Project update 2: https://drive.google.com/file/d/1pq2tvnr9xCBSvpiFYCFkekWCfSHKLEht/view?usp=sharing
5. Healthcare data analysis project slides: https://docs.google.com/presentation/d/1_uSXZIit8gsXRwd5nqW_YxdgnRP_YRyGE1YLQLu8nJ8/edit?usp=sharing