Department of Computer Science and Engineering (AI)

Health Risk Classification

Predict risk category (low/medium/high) based on BMI, exercise, and eating habits.

A Project Report

Submitted By:

Rakhi Garhwal

202401100300194

OF

B.Tech CSE(AI)

Introduction

In today's fast-paced world, many people struggle to maintain a healthy lifestyle. Poor dietary choices, lack of physical activity, and increasing consumption of junk food have led to a rise in lifestyle-related diseases such as obesity, heart disease, and diabetes. Early detection and prevention of health risks are more important than ever.

This project aims to develop a **Health Risk Classification system** using a machine learning model in Python. The goal is to predict an individual's health risk level — categorized as **Low**, **Medium**, or **High** — based on three main lifestyle factors:

- **Body Mass Index (BMI)**: A measure of body fat based on height and weight.
- **Exercise Hours Per Week**: Represents how physically active a person is.
- **Junk Food Frequency Per Week**: Shows dietary habits and unhealthy food intake.

Using these inputs, the machine learning model can learn patterns from historical data and make intelligent predictions about new individuals' health risks. This type of system can be integrated into fitness apps, used by nutritionists, or serve as a self-assessment tool for the general public.

With the help of data analysis and the Random Forest Classifier algorithm, this project shows how artificial intelligence can be used to promote wellness, encourage healthier choices, and possibly prevent future health issues.

Methodology

To predict health risk levels effectively, a machine learning-based approach was followed using Python and scikit-learn. The overall process involved the following steps:

---

1. Dataset Preparation
The dataset included details like BMI, weekly exercise hours, and junk food consumption frequency. Each record was labeled with a health risk level — Low, Medium, or High. This dataset served as the input for training and testing the machine learning model.

---

2. Data Cleaning and Preprocessing
Any rows with missing or null values were removed to maintain the accuracy of the model. Since the target column `risk_level` contained text labels, it was converted into numeric format using **Label Encoding** so the algorithm could understand it.

---

3. Feature Selection
Three main features were selected as input for prediction:

- **BMI**
- **Exercise Hours per Week**
- **Junk Food Frequency**
  These features were chosen because they directly affect an individual's health and are easy to collect in real-life scenarios.

---

4. Model Selection and Training

We used the **Random Forest Classifier**, which is known for its high accuracy and ability to handle classification tasks well. The data was split into training and testing sets using an 80:20 ratio. The model was then trained on the training set.

---

5. Evaluation

After training, the model's accuracy was tested using the test set. We used an **accuracy score** and a **classification report** to evaluate how well the model predicted each risk level. The model showed good performance and was able to predict with reasonable accuracy.

---

6. Prediction

Finally, the trained model could predict the health risk of a new person by using their BMI, exercise routine, and junk food habits. This makes the model useful for health awareness tools or early risk detection.

Code

```
# Import necessary libraries
import pandas as pd
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

# Step 1: Load the dataset
df = pd.read_csv("/health_risk.csv")  # Replace with your actual CSV file
name

# Step 2: Display the first few rows
print("Sample Data:")
print(df.head())

# Step 3: Check for missing values (optional)
print("\nMissing values:\n", df.isnull().sum())
df = df.dropna()  # Drop any rows with missing values (or fill if needed)

# Step 4: Encode the target column (risk_level)
le_risk = LabelEncoder()
df['risk_level_encoded'] = le_risk.fit_transform(df['risk_level'])

# Step 5: Define features (X) and target (y)
X = df[['bmi', 'exercise_hours', 'junk_food_freq']]
y = df['risk_level_encoded']

# Step 6: Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
```

```
)

# Step 7: Train the model using Random Forest
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 8: Make predictions
y_pred = model.predict(X_test)

# Step 9: Evaluate the model
print("\nClassification Report:")
print(classification_report(y_test, y_pred,
target_names=le_risk.classes_))

print("Accuracy Score:", accuracy_score(y_test, y_pred))

# Step 10 (Optional): Predict risk for a new person
# Example: A person with BMI=29, exercises 5 hours/week, eats junk 2
times/week
new_input = pd.DataFrame({
    'bmi': [29],
    'exercise_hours': [5],
    'junk_food_freq': [2]
})

predicted_risk = model.predict(new_input)
print("\nPredicted Risk Level for new input:",
le_risk.inverse_transform(predicted_risk)[0])
```

Output/Result

Commands + Code + Text

```
Sample Data:
         bmi  exercise_hours  junk_food_freq risk_level
0  28.730279              13               1       high
1  31.301442              12               4     medium
2  32.549043               9               0     medium
3  30.463670               2               1     medium
4  28.431755               2               1        low

Missing values:
 bmi              0
exercise_hours    0
junk_food_freq    0
risk_level        0
dtype: int64

Classification Report:
              precision    recall  f1-score   support

        high       0.20      0.20      0.20         5
         low       0.14      0.20      0.17         5
      medium       0.62      0.50      0.56        10

    accuracy                           0.35        20
   macro avg       0.32      0.30      0.31        20
weighted avg       0.40      0.35      0.37        20

Accuracy Score: 0.35
```

References

- **Dataset**: Provided by instructor / created manually
- **Libraries Used**:
    - `pandas` – Data handling
    - `scikit-learn` – ML model and evaluation
- **Tools**: Jupyter Notebook / Google Colab
- **Model Used**: Random Forest Classifier
- **Python Version**: 3.x