ReciHealth

An AI-Driven Nutritional and Health Assistant for Personalized Wellness

15.09.2024

K Nithinram

Overview

Maintaining a healthy diet is challenging in today's fast-paced world of irregular food compositions and busy lifestyles. This article presents ReciHealth, an AI-driven nutritional and health assistant for personalized wellness, aiming to tackle this problem by personalizing dietary guidance for local foods and cuisines. Users input their daily food choices, recipes, ingredients, routines, and health goals. ReciHealth analyzes these data, considering factors like age, gender, and dietary restrictions, to provide a detailed nutritional breakdown and predict potential health outcomes. The platform utilizes machine learning algorithms empowering users to make informed choices, Predict and suggest improvements to existing recipes to improve their health impact, and connect users with valuable partners. Users can access expert advice from doctors and fitness centers, indulge in occasional treats through restaurant and food delivery partnerships, or have groceries delivered directly from partnered stores. This integrated approach positions ReciHealth as a powerful tool for individuals seeking to optimize their health and achieve their fitness goals.

1. Prototype Selection

Problem Statement

A healthy diet in a relentless cadence of today's lifestyle can be a constant battle. People often leave little time to prioritize their nutritional needs. Consumers facing obstacles in deciding on healthier food products hindered ability to make informed choices and achieve optimal well-being, and the lack of focus on healthy eating habits leads to a variety of challenges:

- Irregular Food Compositions.
- Uncertainty about Nutrition Intake.
- Balancing Indulgence with Health.
- Resto-Fitness Disconnect.
- Limited Collaboration.

Fortunately, these challenges presented by the complexities of healthy eating, lack of transparency in food content, difficulty tracking intake, and a disconnect between fitness and food choices create the opportunity for innovative solutions. Utilizing artificial intelligence can empower informed decision-making, and promote a balanced lifestyle.

Market/Customer/Business Need Assessment

There has been a transformative shift in the health and wellness industry, consumer awareness, and the rising prevalence of lifestyle-related diseases. Consumers have become more proactive in managing their health, and there is a growing demand for integrated solutions.

- **Global Health and Wellness Market:** The health and wellness market is experiencing significant growth, driven by increasing consumer awareness of health issues, the rise of lifestyle-related diseases, and the growing adoption of healthy living practices. According to a report by Grand View Research, the global health and wellness market size was valued at USD 4.4 trillion in 2022 and is expected to expand at a compound annual growth rate (CAGR) of 5.5 % from 2023 to 2030.
- **Digital Health Market:** The digital health market, encompassing telemedicine, health apps, and wearable devices, is projected to grow substantially. This growth is fueled by advancements in technology, increasing smartphone penetration, and the need for remote healthcare solutions. According to MarketsandMarkets, the global digital health market is expected to reach USD 639.4 billion by 2026, growing at a CAGR of 28.5 %.

- **Nutrition and Fitness Apps Market:** The market for nutrition and fitness apps is booming as more consumers seek to manage their health proactively. Statista reports that revenue in the fitness apps segment is projected to reach USD 15.96 billion in 2024, with an annual growth rate of 9.37 % from 2024 to 2028.

Target Specifications and Characterization

- **User profile customization and dynamic recommendations:** The platform manages a wide range of users of different dynamics in terms of age, health, and nutrition factors, and machine learning integration supports getting user dynamic predictions and suggestions on customized dietary and fitness regulations and tracking specifically to different user characteristics.
- **Beneficial Partner Integration with customer engagements**: For Enhanced Food Safety and Quality, Holistic Health Solutions, Product Convenience and Accessibility, and Professional Guidance.

Customer characterization:

- Health-conscious individuals (Adults aged 18-60), Fitness Enthusiasts (Younger adults aged 18-35), Busy Professionals (Working adults aged 25-50), and Individuals with Specific Health Goals or Restrictions (Adults aged 30-60) are the targeted groups of users for this platform.

Iterative and continuous improvements and strategic partnerships can cultivate a better-informed health-conscious society, enhance food safety and quality in the market, a customer-positive market, and foster a healthy community among users and the food industry.

External Search

Market Research and Trends:

- Statista: Offered comprehensive statistics and reports on health and wellness industry trends, consumer behavior, and market analysis.
- Global Wellness Institute: Provided valuable insights into the global wellness economy, including industry reports and market research.
- ZION Market Research: India Digital Health Industry Prospective, provided a comprehensive analysis of the Indian digital health industry.
- Indian Brand Equity Foundation (IBEF): Provided insight into the comprehensive analysis of the Indian health industry.

Food and Nutritional Information and Health Care datasets:

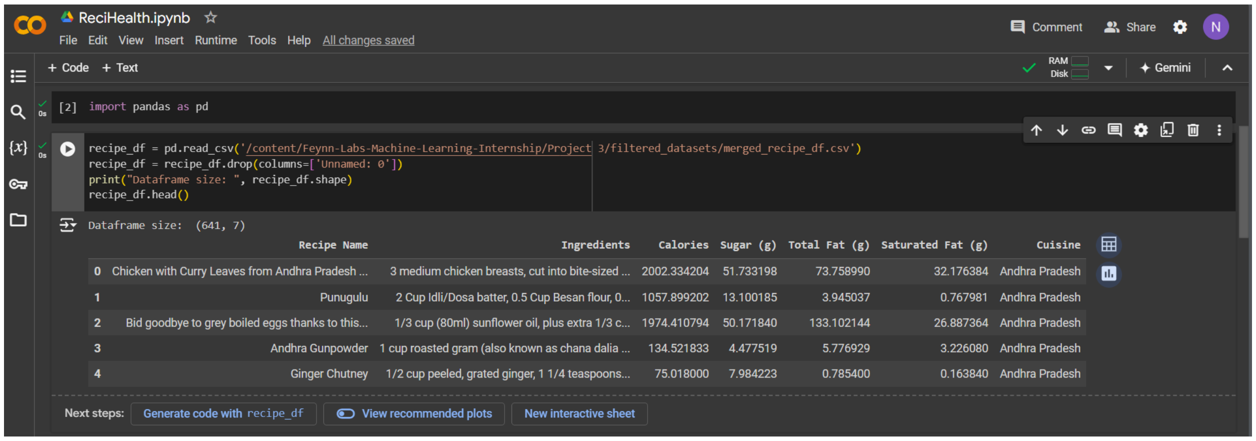
- EDAMAM: One of the leading providers of nutrition data and analytics through free and subscription-based API service, Provided a wide range of Indian cuisine-based food recipes and nutritional information for model training.
- Diabetes prediction dataset: A Comprehensive Dataset from Kaggle for Predicting Diabetes with Medical & Demographic Data.
- Obesity based on eating habits & physical cond.: A Comprehensive Dataset from Kaggle for Predicting Obesity with ages between 14 and 61 and diverse eating habits and physical condition.

Research Journals and papers:

- Diet Recommendation System Using Machine Learning: Provided insights on a diet recommendation system assessing health parameters for customized diet plans using machine learning.
- Machine Learning in Nutrition Research: Provided insights on an extensive study in the utilization of machine learning algorithms in characterizing personalized nutrition to tackle health complications like obesity, diabetes, and cardiovascular diseases.

Datasets Description:

Recipe Dataset:


ReciHealth.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Gemini
Next steps: Generate code with recipe_df View recommended plots New interactive sheet

```
[2] import pandas as pd
```

```
recipe_df = pd.read_csv('/content/Feynn-Labs-Machine-Learning-Internship/Project 3/filtered_datasets/merged_recipe_df.csv')
recipe_df = recipe_df.drop(columns=['Unnamed: 0'])
print("Dataframe size: ", recipe_df.shape)
recipe_df.head()
```

```
Dataframe size:  (641, 7)
```

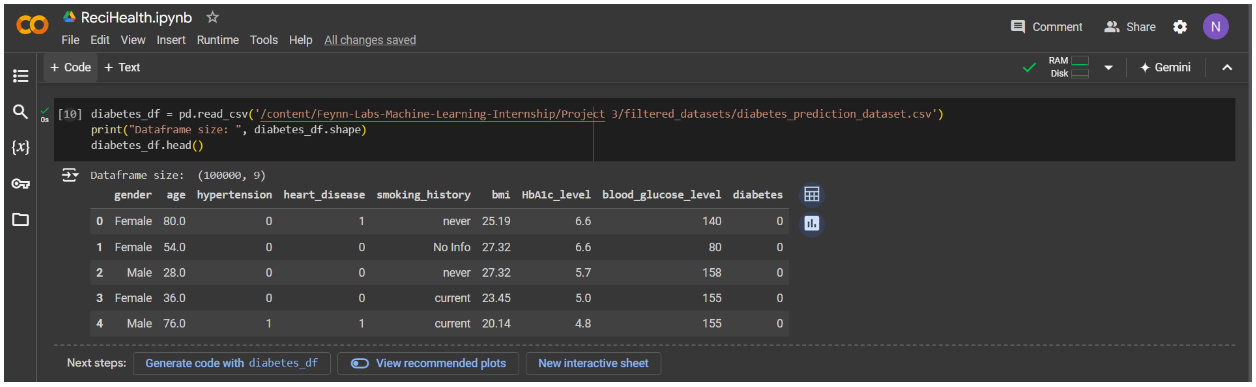
	Recipe Name	Ingredients	Calories	Sugar (g)	Total Fat (g)	Saturated Fat (g)	Cuisine
0	Chicken with Curry Leaves from Andhra Pradesh ...	3 medium chicken breasts, cut into bite-sized ...	2002.334204	51.733198	73.758990	32.176384	Andhra Pradesh
1	Punugulu	2 Cup Idli/Dosa batter, 0.5 Cup Besan flour, 0...	1057.899202	13.100185	3.945037	0.767981	Andhra Pradesh
2	Bid goodbye to grey boiled eggs thanks to this...	1/3 cup (80ml) sunflower oil, plus extra 1/3 c...	1974.410794	50.171840	133.102144	26.887364	Andhra Pradesh
3	Andhra Gunpowder	1 cup roasted gram (also known as chana dalia ...	134.521833	4.477519	5.776929	3.226080	Andhra Pradesh
4	Ginger Chutney	1/2 cup peeled, grated ginger, 1 1/4 teaspoons...	75.018000	7.984223	0.785400	0.163840	Andhra Pradesh

The recipe dataset consists of attributes such as:

- Recipe name
- Ingredients (Respective of each recipe)
- Respective of each recipe their Calories, Sugar (g), Total Fat (g), and Saturated Fat (g) values.

- Cuisine

Diabetes Prediction dataset:



```
[10] diabetes_df = pd.read_csv('/content/Feynn-Labs-Machine-Learning-Internship/Project 3/filtered_datasets/diabetes_prediction_dataset.csv')
     print("Dataframe size: ", diabetes_df.shape)
     diabetes_df.head()
```

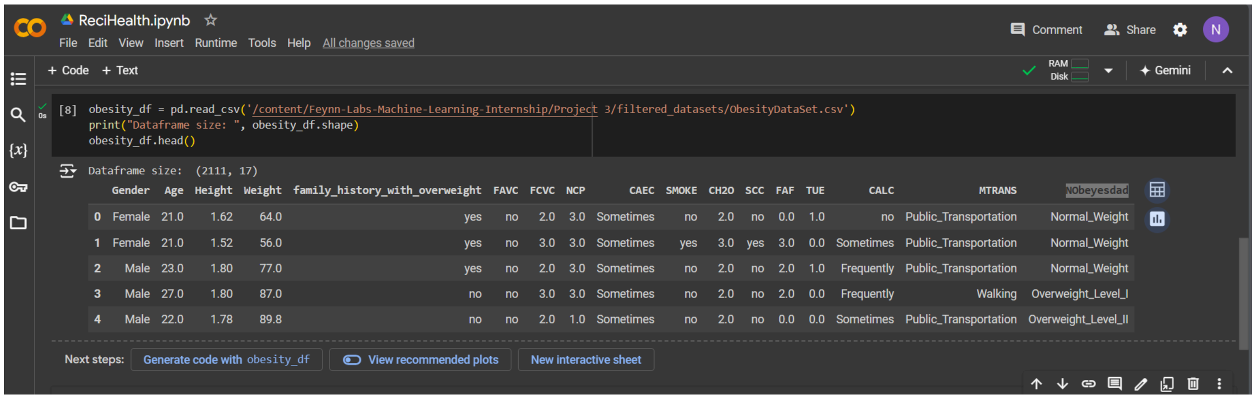
```
Dataframe size:  (100000, 9)
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0

The dataset consists of attributes such as:

- age
- gender
- body mass index (BMI)
- hypertension, heart disease
- smoking history
- HbA1c level
- blood glucose level
- diabetes (whether is or not diabetic)

Obesity Prediction dataset:



```
[8] obesity_df = pd.read_csv('/content/Feynn-Labs-Machine-Learning-Internship/Project 3/filtered_datasets/ObesityDataSet.csv')
    print("Dataframe size: ", obesity_df.shape)
    obesity_df.head()
```

```
Dataframe size:  (2111, 17)
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRANS	NObeyesdad
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_Transportation	Normal_Weight
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transportation	Normal_Weight
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_Transportation	Normal_Weight
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	Walking	Overweight_Level_I
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_Transportation	Overweight_Level_II

Attributes related to eating habits are:

- Frequent consumption of high-caloric food (FAVC)
- Frequency of consumption of vegetables (FCVC)
- Number of main meals (NCP)
- Consumption of food between meals (CAEC)

- Consumption of water daily (CH20)
- Consumption of alcohol (CALC).

Attributes related to the physical condition are:

- Calorie consumption monitoring (SCC)
- Physical activity frequency (FAF)
- Time using technology devices (TUE)
- Transportation used (MTRANS)

Other variables were:

- Gender, Age, Height, and Weight
- NObeyesdad (type of obesity)

Benchmarking

There are a few popular algorithm-based platforms such as **MyFitnessPal, Cronometer, Lose It!, and HealthifyMe** that offer calorie tracking, macronutrient tracking, exercise logging, weight loss-focused platforms with meal planning, and community features. Benchmarking involves comparing against factors such as User Satisfaction, Accuracy, Engagement, Partnerships, Project Processes, and Performance Metrics to industry best standards and practices, there is a need for continuous research of methodologies and machine learning for the best results on consecutive technique developments.

Applicable Standards and Regulations

- Compliance with food labeling, health claims, and nutritional information laws **(FSSAI, FDA, EFSA).**
- Complete adherence towards health and safety regulations, and Digital health regulations **(National Health Policy 2017, MDR, 21st Century Cures Ac**t).
- Clear guidelines for data processing and user permission, compliance towards respectful usage and protection of sensitive personal data, and regulations to enhance data privacy and protection laws **(Information Technology (IT) Act 2000, PDPB, GDPR, and HIPAA).**
- Consumer protection, transparency, accuracy, and protection against unfair business practices **(Consumer Protection Act 2019, FTC).**
- Environmental Regulations. Sustainable practices and responsible consumption which applies to integrating the platform's operations and partnerships **(MoEFCC, EPA).**
- Handling Local business collaboration law for joint ventures.

Applicable Constraints

- Requirements and developments for up-to-date and robust technical infrastructures for handling large/complex data and machine learning.
- Improvising machine learning models and clever data management for better handling of consumer dynamics for accurate results.
- Continuous data collection and storage solutions for both user data management and food/nutrition datasets.
- Initial difficulties in aggregating business partners in terms of restaurants and food product sales partners.

Business Opportunities

By leveraging AI and machine learning algorithms, ReciHealth aims to provide tailored recipe recommendations based on users' dietary needs, fitness goals, and health conditions, opening ways for monetization. A **subscription-based model** will allow users to access premium features such as personalized meal plans, detailed nutrition tracking, and expert consultations, providing a steady revenue stream. Additionally, the platform hopes to partner with **restaurants, food delivery services, and grocery stores**, earning **affiliate commissions** from orders or referrals made through the app. Another direct monetization path is **in-app purchases**, where users can buy specific diet plans, fitness programs, or recipe packs tailored to their health goals. Advertising within the free version of the platform, targeting health-conscious users with relevant products and services, creates an additional revenue stream. These direct monetization avenues position ReciHealth for sustainable growth while offering valuable health solutions to users.

Concept Generation

The conceptualization of ReciHealth was driven by the need to address the growing demand in personalized health and nutrition solutions, a personalized dietary and nutrition assistant, Diverse Dietary Needs, and Balancing Indulgence with Health by uniquely focusing on recipe modification to accommodate diverse dietary requirements, and leveraging machine learning algorithms to provide accessible, in-depth nutritional analysis and tailored health recommendations.

Concept Development

ReciHealth concept development aims to create a comprehensive and user-centric platform that leverages machine learning algorithms to offer personalized recipe and nutrition recommendations. The idea is to design a platform that not only suggests recipes based on individual dietary needs and health conditions—such as diabetes and obesity—but also incorporates detailed nutritional insights like calories, sugar, and fats to support users in making informed food choices. The development process involves analyzing user preferences, health data, and nutritional requirements to refine the recommendation algorithms and ensure that the platform delivers accurate, relevant suggestions. By integrating machine learning models and leveraging a diverse dataset of recipes, ReciHealth aims to offer a tailored experience that enhances user health and well-being.

1.1 **Feasibility**

The chosen prototype for development is **ReciHealth**, an AI-driven personalized recipe and nutrition recommendation service. The feasibility of this product is high for the following reasons:

- **Current Technology**: ReciHealth can be developed using existing machine learning algorithms, models, and techniques for recommendation engines and nutritional analysis, all of which are well-researched fields.
- **Timeframe**: Within 2-3 years, it will be feasible to build a functional version of the app using Python, machine learning libraries, and databases for storing recipes and user health data.
- **Data Availability**: There is an abundance of publicly available recipe and nutritional datasets, and APIs such as **Nutritionix, Edamam,** or **Spoonacular**, which can be leveraged to accelerate the development process.

These diverse revenue streams make ReciHealth well-positioned for long-term profitability in the expanding wellness and digital health sector.

1.2 **Viability**

ReciHealth holds long-term potential for staying relevant over 20-30 years due to:

- **Growing Health Awareness**: The increasing public focus on personalized health and fitness will make such a product viable for decades.
- **AI Advancements**: As AI technology advances, ReciHealth can evolve to offer more precise and personalized recommendations.
- **Sustainability**: With continuous data collection and health insights, the platform can remain updated and relevant, creating long-term user engagement.

1.3 Monetization

ReciHealth is directly monetizable in multiple ways:

- **Subscription-Based Model**: Users can pay for premium features like detailed diet tracking, advanced analytics, and personalized meal plans.
- **Partner Commissions**: ReciHealth can earn commissions through partnerships with restaurants, food delivery services, and fitness centers by recommending their services to users.
- **Advertisements**: Relevant advertisements for health-related products or services can be integrated into the free version of the app for an additional revenue stream.

2. Prototype Development

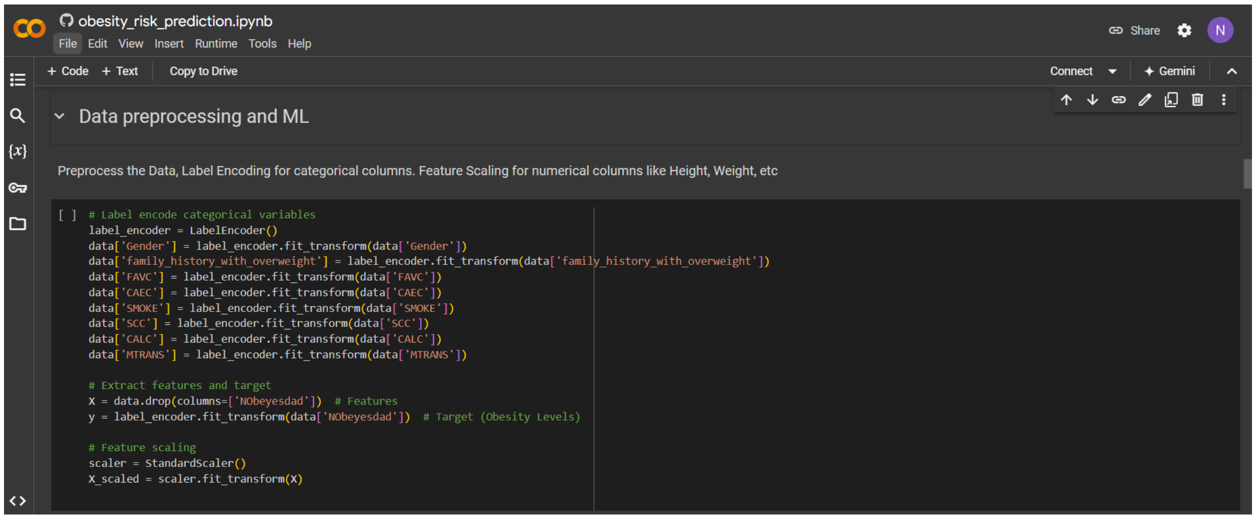
The current prototype is developed in 2 stages:

- Obesity-Based Recipe Suggestions
- Diabetic-Based Recipe Suggestions

This approach is chosen due to the critical importance of consumer health care, the main health concerns closely related to food and nutrition are diabetes and obesity where macronutrients such as Sugar, Calories, Total Fat, and Saturated Fats have a direct cause on health if not managed properly, the easiest way to manage is through a supervised food and nutrition intake.

2.1 Small Scale Code Implementation

2.1.1 Obesity Based Recipe Suggestions:


obesity_risk_prediction.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text Copy to Drive
Connect
Gemini

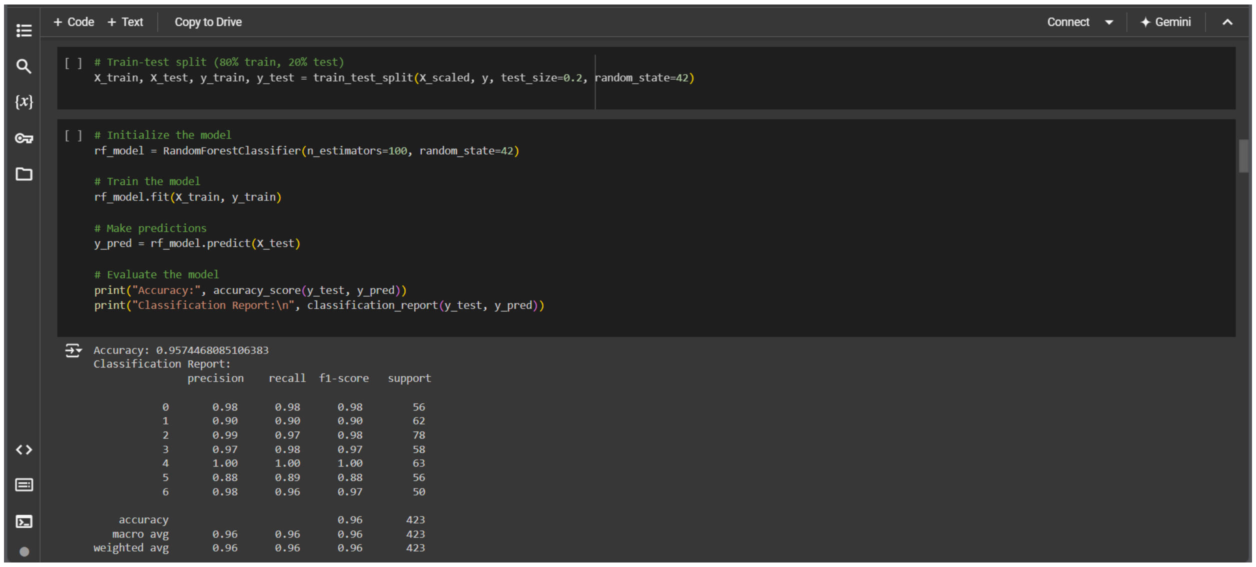
Data preprocessing and ML

Preprocess the Data, Label Encoding for categorical columns. Feature Scaling for numerical columns like Height, Weight, etc

```
# Label encode categorical variables
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])
data['family_history_with_overweight'] = label_encoder.fit_transform(data['family_history_with_overweight'])
data['FAVC'] = label_encoder.fit_transform(data['FAVC'])
data['CAEC'] = label_encoder.fit_transform(data['CAEC'])
data['SMOKE'] = label_encoder.fit_transform(data['SMOKE'])
data['SCC'] = label_encoder.fit_transform(data['SCC'])
data['CALC'] = label_encoder.fit_transform(data['CALC'])
data['MTRANS'] = label_encoder.fit_transform(data['MTRANS'])

# Extract features and target
X = data.drop(columns=['NObeyesdad'])  # Features
y = label_encoder.fit_transform(data['NObeyesdad'])  # Target (Obesity Levels)

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

The prototype development for the Obesity-Based Recipe Suggestions model involved several key stages to ensure the model's effectiveness and accuracy.

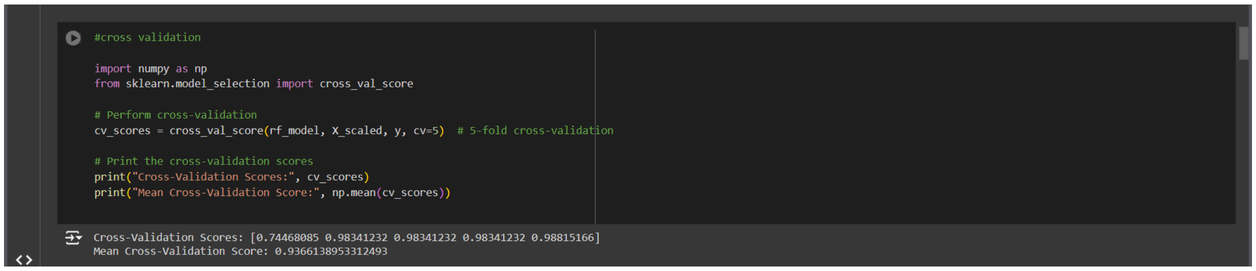
Initially, categorical variables such as **'Gender,' 'family_history_with_overweight,' 'FAVC,' 'CAEC,' 'SMOKE,' 'SCC,' 'CALC,' and 'MTRANS'** were label-encoded to convert them into a numerical format suitable for machine learning. Subsequently, the features were scaled using **`StandardScaler`** to standardize the data.



```
[ ] # Train-test split (80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

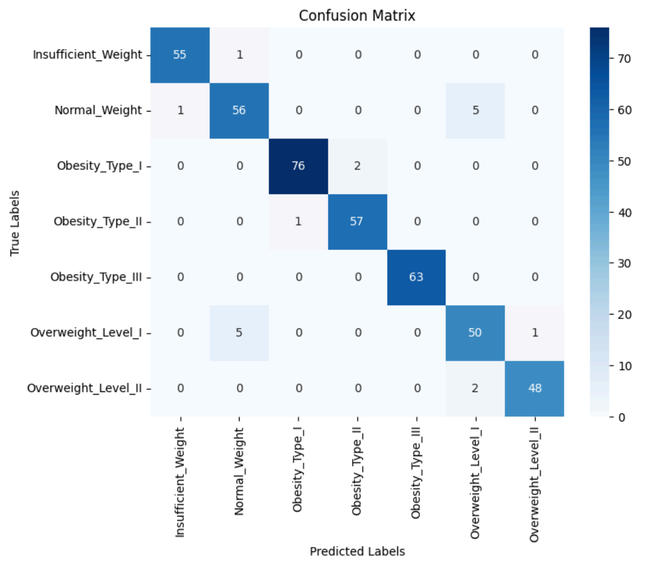
```
[ ] # Initialize the model
    rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

    # Train the model
    rf_model.fit(X_train, y_train)

    # Make predictions
    y_pred = rf_model.predict(X_test)

    # Evaluate the model
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9574468085106383
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        56
           1       0.90      0.90      0.90        62
           2       0.99      0.97      0.98        78
           3       0.97      0.98      0.97        58
           4       1.00      1.00      1.00        63
           5       0.88      0.89      0.88        56
           6       0.98      0.96      0.97        50

    accuracy                           0.96       423
   macro avg       0.96      0.96      0.96       423
weighted avg       0.96      0.96      0.96       423
```

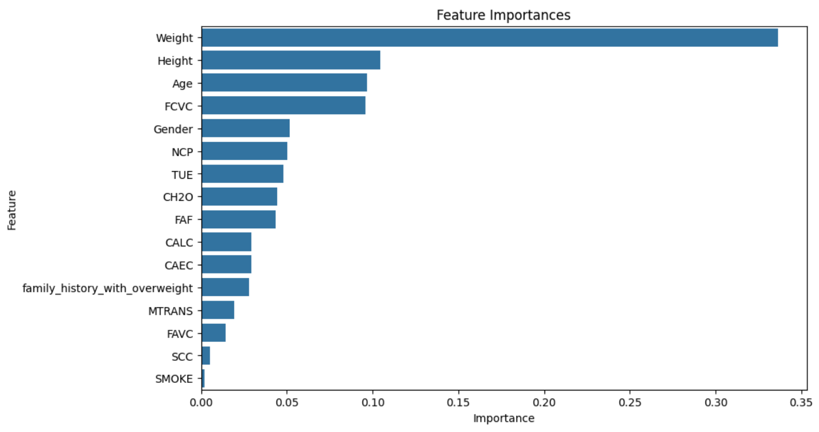


```
#cross validation

import numpy as np
from sklearn.model_selection import cross_val_score

# Perform cross-validation
cv_scores = cross_val_score(rf_model, X_scaled, y, cv=5)  # 5-fold cross-validation

# Print the cross-validation scores
print("Cross-Validation Scores:", cv_scores)
print("Mean Cross-Validation Score:", np.mean(cv_scores))
```

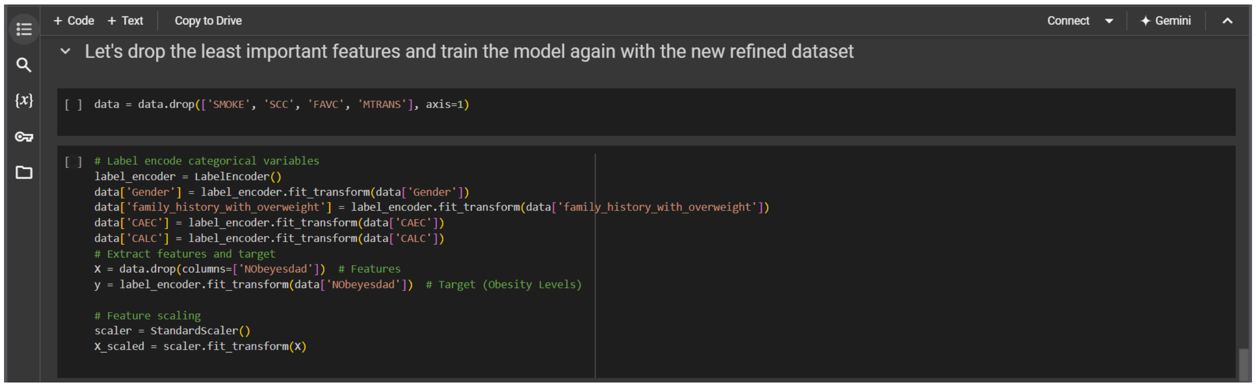
```
Cross-Validation Scores: [0.74468085 0.98341232 0.98341232 0.98341232 0.98815166]
Mean Cross-Validation Score: 0.9366138953312493
```



The dataset was then split into **training and testing sets (80% train, 20% test)**, and a **RandomForestClassifier with 100 estimators** was employed for model training. The initial model achieved an impressive accuracy of **95.7%** and demonstrated strong performance in classifying different obesity levels based on the classification report.



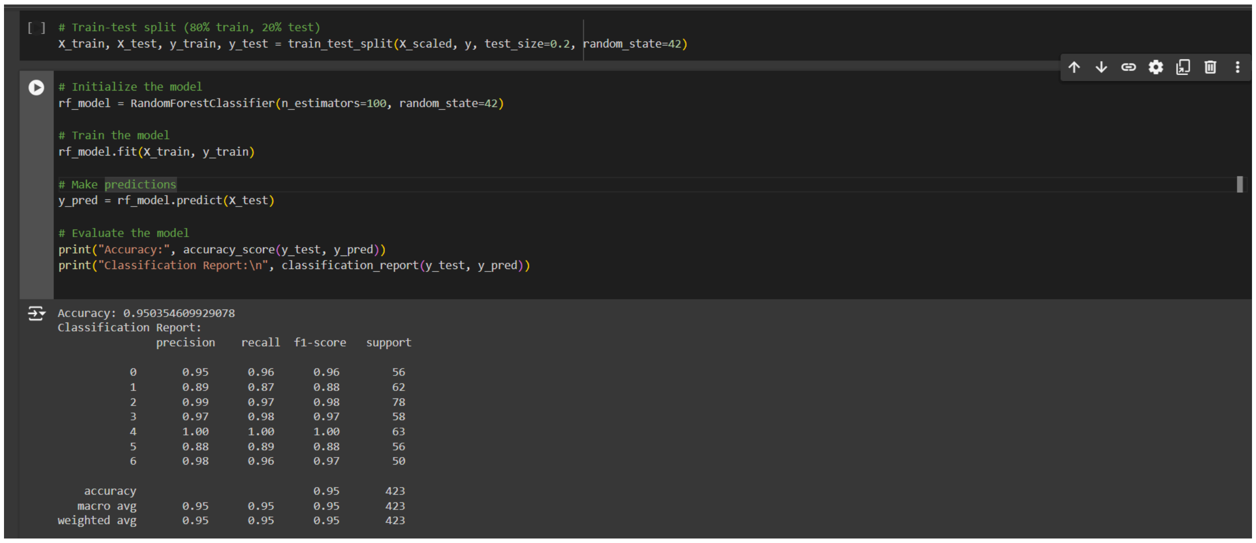
Feature importance analysis was conducted, revealing key features such as **'Weight,' 'Height,' and 'Age' as highly significant.** Based on this analysis, less important features like **'SMOKE,' 'SCC,' 'FAVC,' and 'MTRANS' were removed.**



Let's drop the least important features and train the model again with the new refined dataset

```
[ ] data = data.drop(['SMOKE', 'SCC', 'FAVC', 'MTRANS'], axis=1)
```

```
[ ] # Label encode categorical variables
    label_encoder = LabelEncoder()
    data['Gender'] = label_encoder.fit_transform(data['Gender'])
    data['family_history_with_overweight'] = label_encoder.fit_transform(data['family_history_with_overweight'])
    data['CAEC'] = label_encoder.fit_transform(data['CAEC'])
    data['CALC'] = label_encoder.fit_transform(data['CALC'])
    # Extract features and target
    X = data.drop(columns=['NObeyesdad'])  # Features
    y = label_encoder.fit_transform(data['NObeyesdad'])  # Target (Obesity Levels)

    # Feature scaling
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
```

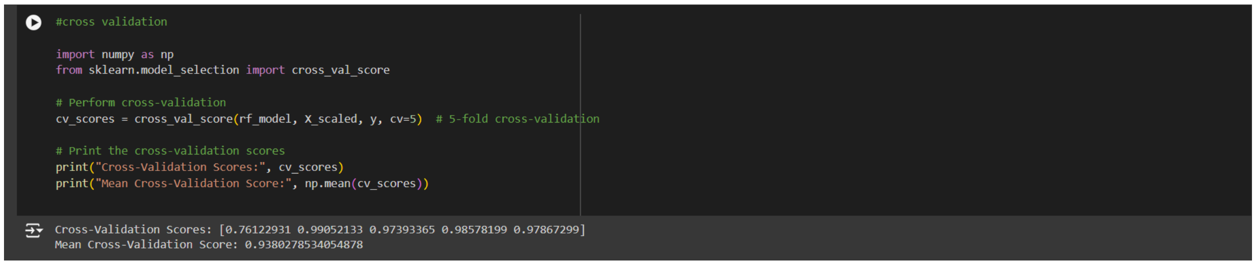


```
[ ] # Train-test split (80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
# Initialize the model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Make predictions
y_pred = rf_model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.950354609929078
Classification Report:
               precision    recall  f1-score   support

           0       0.95      0.96      0.96        56
           1       0.89      0.87      0.88        62
           2       0.99      0.97      0.98        78
           3       0.97      0.98      0.97        58
           4       1.00      1.00      1.00        63
           5       0.88      0.89      0.88        56
           6       0.98      0.96      0.97        50

    accuracy                           0.95       423
   macro avg       0.95      0.95      0.95       423
weighted avg       0.95      0.95      0.95       423
```

In the Post-feature removal, the cross-validation scores showed some improvement: the mean cross-validation score increased from approximately 0.937 to 0.938. The new scores were [0.76122931, 0.99052133, 0.97393365, 0.98578199, 0.97867299], indicating that the model's performance became more robust and consistent after removing less significant features.



```
#cross validation

import numpy as np
from sklearn.model_selection import cross_val_score

# Perform cross-validation
cv_scores = cross_val_score(rf_model, X_scaled, y, cv=5)  # 5-fold cross-validation

# Print the cross-validation scores
print("Cross-Validation Scores:", cv_scores)
print("Mean Cross-Validation Score:", np.mean(cv_scores))
```

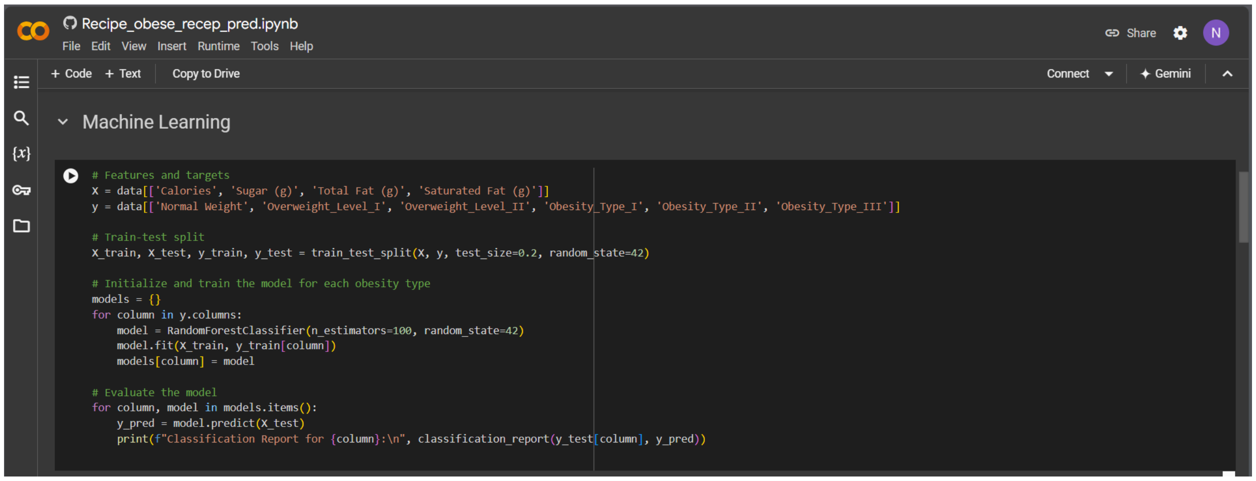
```
Cross-Validation Scores: [0.76122931 0.99052133 0.97393365 0.98578199 0.97867299]
Mean Cross-Validation Score: 0.9380278534054878
```

The recipe dataset was updated to classify recipes based on nutritional limits defined for various obesity risk types (Insufficient Weight, Normal Weight, Overweight Levels I & II, Obesity Types I, II, & III). The key nutrients considered were **Sugar (g)**, **Total Fat (g)**, and **Saturated Fat (g)**, with specific maximum values for each obesity category.

The thresholds for these nutrients were set as follows:

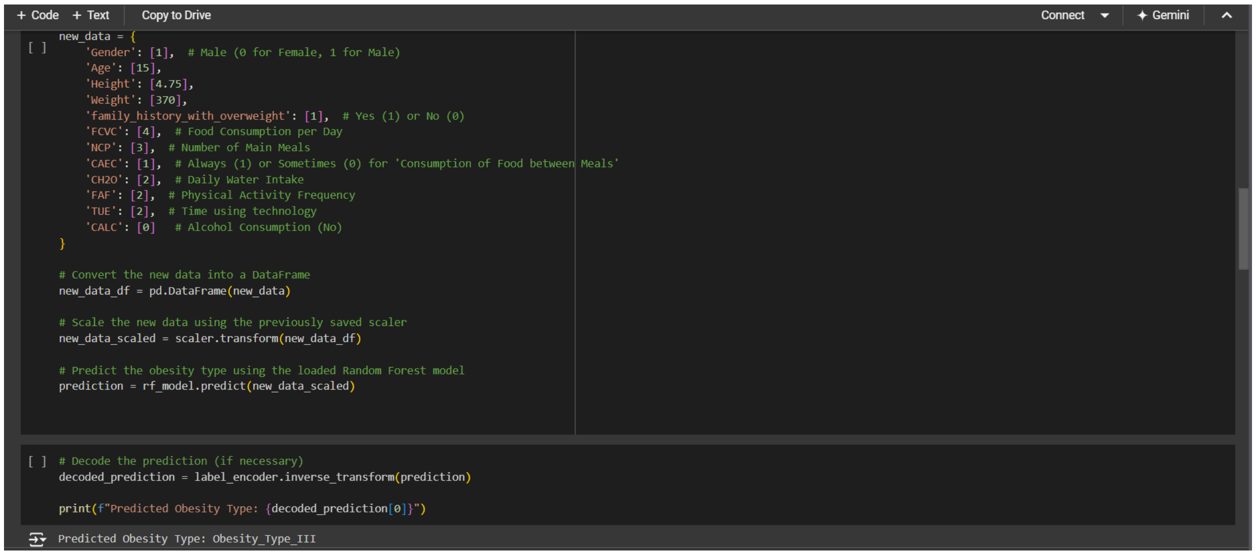
- Insufficient Weight= Max sugar=30g, max total fat=30g, max saturated fat=15g.
- Normal Weight= Max sugar=25g, max total fat=25g, max saturated fat=10g.
- Overweight Level I = Max sugar=20g, max total fat=20g, max saturated fat=8g.
- Overweight Level II = Max sugar=15g, max total fat=15g, max saturated fat=7g.
- Obesity Type I = Max sugar=12g, max total fat=12g, max saturated fat=6g.
- Obesity Type II = Max sugar=10g, max total fat=10g, max saturated fat=5g.
- Obesity Type III = Max sugar=8g, max total fat=8g, max saturated fat=4g.

While this is set to an average accepted value according to nutritional guidelines, it is preferable to have a medical professional set the optimal levels.


Recipe_obese_recep_pred.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text Copy to Drive
Connect
Gemini
Machine Learning

```
# Features and targets
X = data[['Calories', 'Sugar (g)', 'Total Fat (g)', 'Saturated Fat (g)']]
y = data[['Normal Weight', 'Overweight_Level_I', 'Overweight_Level_II', 'Obesity_Type_I', 'Obesity_Type_II', 'Obesity_Type_III']]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the model for each obesity type
models = {}
for column in y.columns:
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train[column])
    models[column] = model

# Evaluate the model
for column, model in models.items():
    y_pred = model.predict(X_test)
    print(f"Classification Report for {column}:\n", classification_report(y_test[column], y_pred))
```

The updated recipe dataset was trained using **RandomForestClassifier** for various obesity risk types, including Normal Weight, Overweight Levels I and II, and Obesity Types I, II, and III. The features used to train the model include **Calories**, **Sugar (g)**, **Total Fat (g)**, and **Saturated Fat (g)** from the recipe dataset. Each model is trained to predict whether a recipe fits the nutritional criteria for a specific obesity type, and the evaluation shows high accuracy and strong F1-scores for most categories.

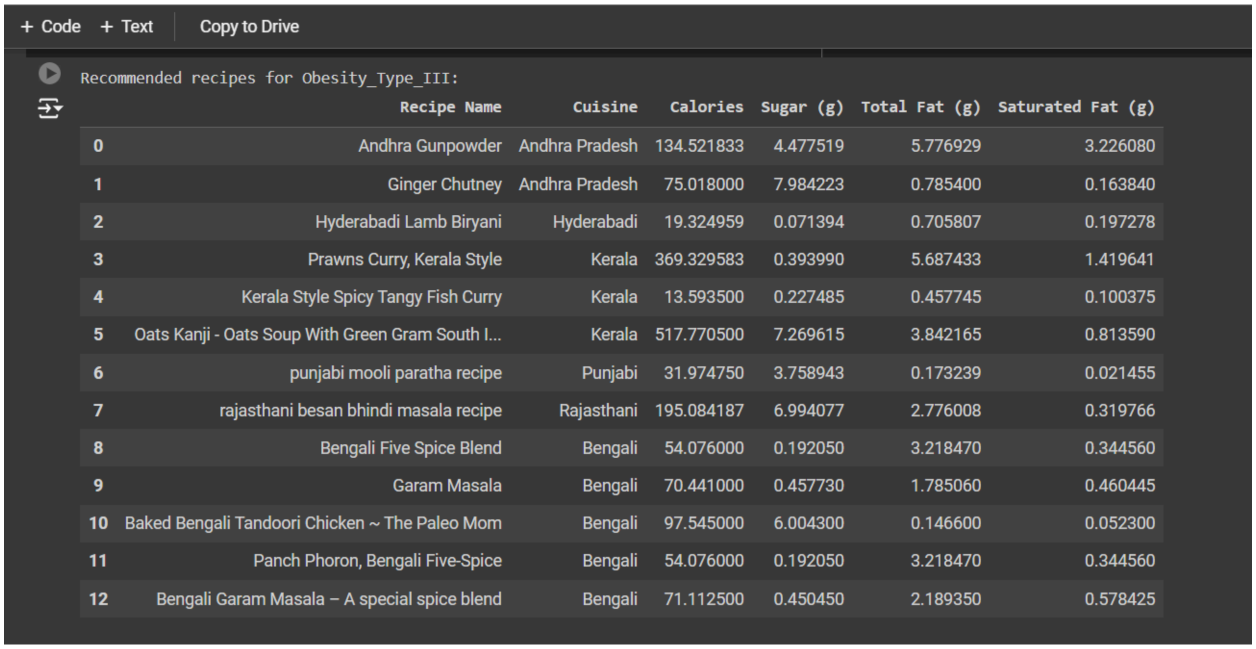


```
new_data = {
    'Gender': [1],  # Male (0 for Female, 1 for Male)
    'Age': [15],
    'Height': [4.75],
    'Weight': [370],
    'family_history_with_overweight': [1],  # Yes (1) or No (0)
    'FCVC': [4],  # Food Consumption per Day
    'NCP': [3],  # Number of Main Meals
    'CAEC': [1],  # Always (1) or Sometimes (0) for 'Consumption of Food between Meals'
    'CH2O': [2],  # Daily Water Intake
    'FAF': [2],  # Physical Activity Frequency
    'TUE': [2],  # Time using technology
    'CALC': [0]   # Alcohol Consumption (No)
}

# Convert the new data into a DataFrame
new_data_df = pd.DataFrame(new_data)

# Scale the new data using the previously saved scaler
new_data_scaled = scaler.transform(new_data_df)

# Predict the obesity type using the loaded Random Forest model
prediction = rf_model.predict(new_data_scaled)
```

```
# Decode the prediction (if necessary)
decoded_prediction = label_encoder.inverse_transform(prediction)

print(f"Predicted Obesity Type: {decoded_prediction[0]}")
```

```
Predicted Obesity Type: Obesity_Type_III
```

Once trained, the model is used to predict the obesity type of a new user based on their personal characteristics, such as age, weight, and eating habits. In this example, a user is predicted to fall under **Obesity_Type_III**, which requires stricter control over sugar, total fat, and saturated fat intake.

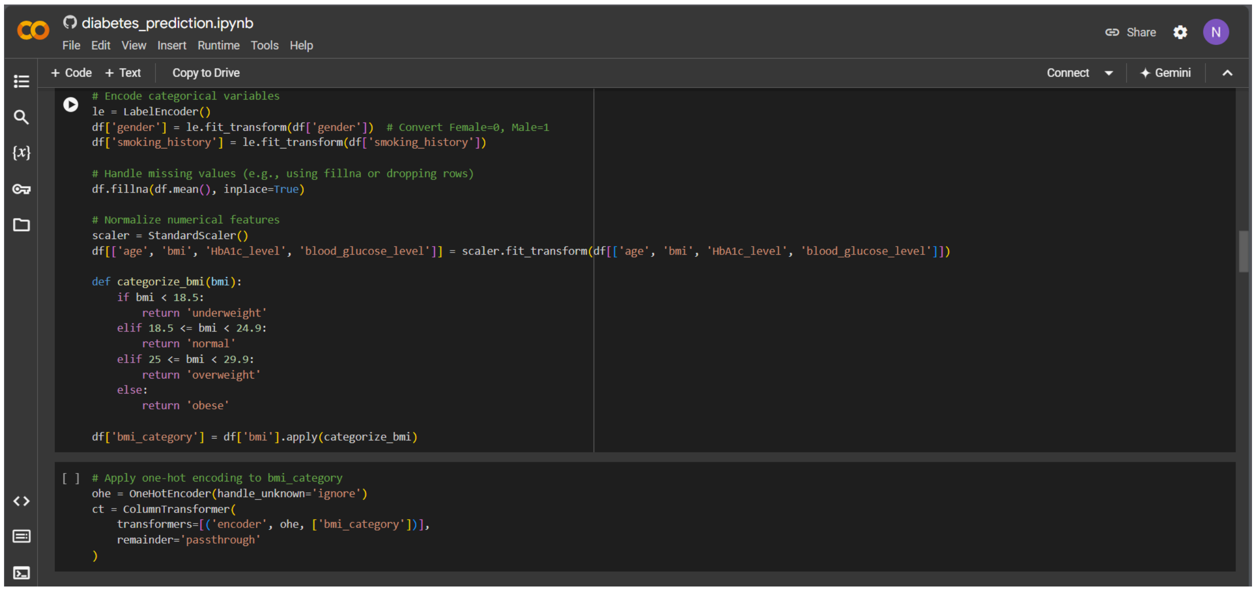


```
Recommended recipes for Obesity_Type_III:
```

	Recipe Name	Cuisine	Calories	Sugar (g)	Total Fat (g)	Saturated Fat (g)
0	Andhra Gunpowder	Andhra Pradesh	134.521833	4.477519	5.776929	3.226080
1	Ginger Chutney	Andhra Pradesh	75.018000	7.984223	0.785400	0.163840
2	Hyderabadi Lamb Biryani	Hyderabadi	19.324959	0.071394	0.705807	0.197278
3	Prawns Curry, Kerala Style	Kerala	369.329583	0.393990	5.687433	1.419641
4	Kerala Style Spicy Tangy Fish Curry	Kerala	13.593500	0.227485	0.457745	0.100375
5	Oats Kanji - Oats Soup With Green Gram South I...	Kerala	517.770500	7.269615	3.842165	0.813590
6	punjabi mooli paratha recipe	Punjabi	31.974750	3.758943	0.173239	0.021455
7	rajasthani besan bhindi masala recipe	Rajasthani	195.084187	6.994077	2.776008	0.319766
8	Bengali Five Spice Blend	Bengali	54.076000	0.192050	3.218470	0.344560
9	Garam Masala	Bengali	70.441000	0.457730	1.785060	0.460445
10	Baked Bengali Tandoori Chicken ~ The Paleo Mom	Bengali	97.545000	6.004300	0.146600	0.052300
11	Panch Phoron, Bengali Five-Spice	Bengali	54.076000	0.192050	3.218470	0.344560
12	Bengali Garam Masala – A special spice blend	Bengali	71.112500	0.450450	2.189350	0.578425

Using the user's obesity type, the model then evaluates a set of recipes to identify those that meet the nutritional limits for **Obesity_Type_III**. Recipes that fit the criteria are recommended to the user, with details including the **Recipe Name**, **Cuisine**, and key nutritional information like **Calories**, **Sugar (g)**, **Total Fat (g)**, and **Saturated Fat (g)**. The result is a personalized list of healthy recipes, curated specifically for the user's health needs.

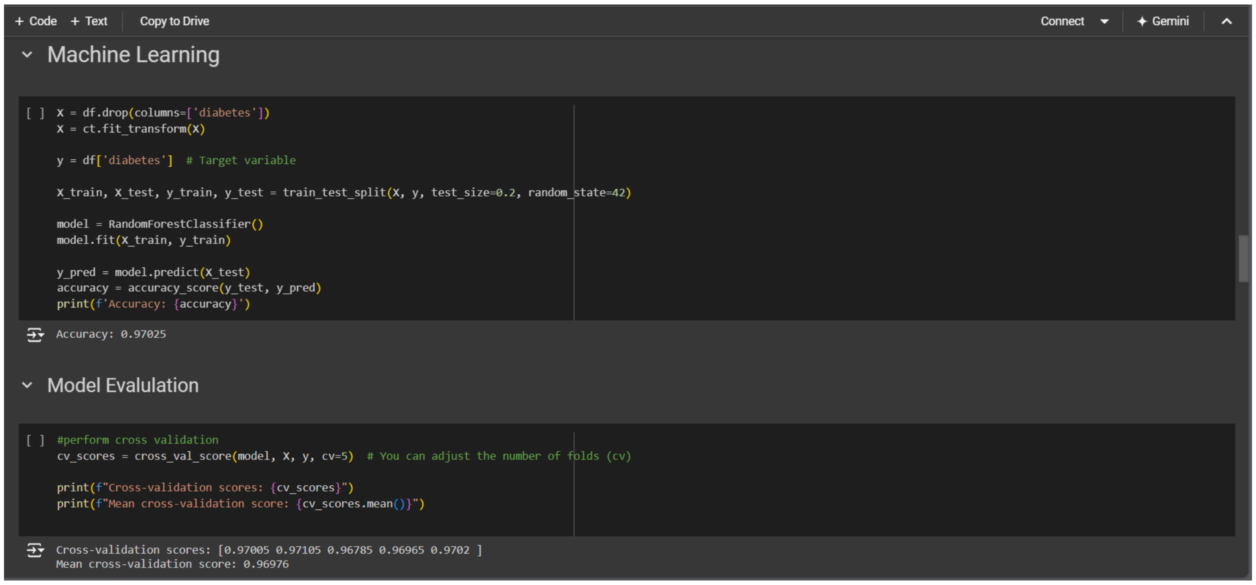
2.1.2 Diabetic-Based Recipe Suggestions


diabetes_prediction.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text Copy to Drive
Connect
Gemini

```python
# Encode categorical variables
le = LabelEncoder()
df['gender'] = le.fit_transform(df['gender'])  # Convert Female=0, Male=1
df['smoking_history'] = le.fit_transform(df['smoking_history'])

# Handle missing values (e.g., using fillna or dropping rows)
df.fillna(df.mean(), inplace=True)

# Normalize numerical features
scaler = StandardScaler()
df[['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']] = scaler.fit_transform(df[['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']])

def categorize_bmi(bmi):
    if bmi < 18.5:
        return 'underweight'
    elif 18.5 <= bmi < 24.9:
        return 'normal'
    elif 25 <= bmi < 29.9:
        return 'overweight'
    else:
        return 'obese'

df['bmi_category'] = df['bmi'].apply(categorize_bmi)
```

```python
# Apply one-hot encoding to bmi_category
ohe = OneHotEncoder(handle_unknown='ignore')
ct = ColumnTransformer(
    transformers=[('encoder', ohe, ['bmi_category'])],
    remainder='passthrough'
)
```

The dataset diabetes contained columns like **gender**, **age**, **hypertension**, **heart disease**, **smoking history**, **BMI**, **HbA1c level**, and **blood glucose level**. The categorical variables, such as **gender** and **smoking history**, are encoded using **LabelEncoder** while missing values are handled by filling them with the column's mean. The numerical features, including **age**, **BMI**, **HbA1c level**, and **blood glucose level**, are normalized using **StandardScaler**. The BMI is categorized into **underweight**, **normal**, **overweight**, and **obese**, and **one-hot encoding** is applied to the **bmi_category**.



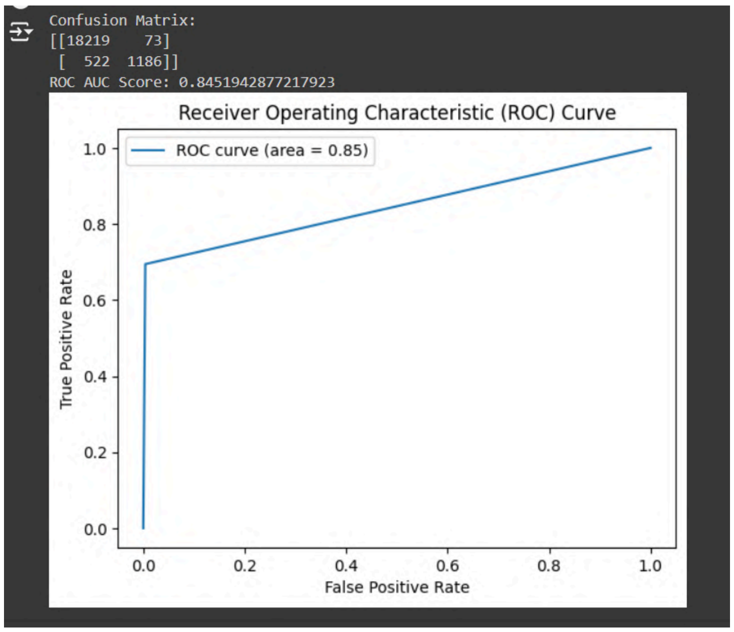
```python
X = df.drop(columns=['diabetes'])
X = ct.fit_transform(X)

y = df['diabetes']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.97025
```

```python
#perform cross validation
cv_scores = cross_val_score(model, X, y, cv=5)  # You can adjust the number of folds (cv)

print(f"Cross-validation scores: {cv_scores}")
print(f"Mean cross-validation score: {cv_scores.mean()}")
```

```
Cross-validation scores: [0.97005 0.97105 0.96785 0.96965 0.9702 ]
Mean cross-validation score: 0.96976
```

RandomForestClassifier model is trained on the processed dataset. After splitting the data into training and test sets (80% train, 20% test), the model achieves an accuracy of **0.97025**. Cross-validation (5-fold) is performed to ensure model generalizability, yielding a mean score of **0.96976**.



```
Confusion Matrix:
[[18219    73]
 [  522  1186]]
ROC AUC Score: 0.8451942877217923
```

The **confusion matrix** shows the model's performance in terms of true positives, true negatives, false positives, and false negatives, with a strong overall performance. The **ROC AUC score** of **0.845** indicates good discriminatory ability in predicting diabetic and non-diabetic patients. Additionally, the ROC curve visualizes the model's performance at different threshold settings.

The original recipe dataset was updated to classify recipes based on nutritional limits defined for various diabetic risk types concerning criteria based on nutritional guidelines.

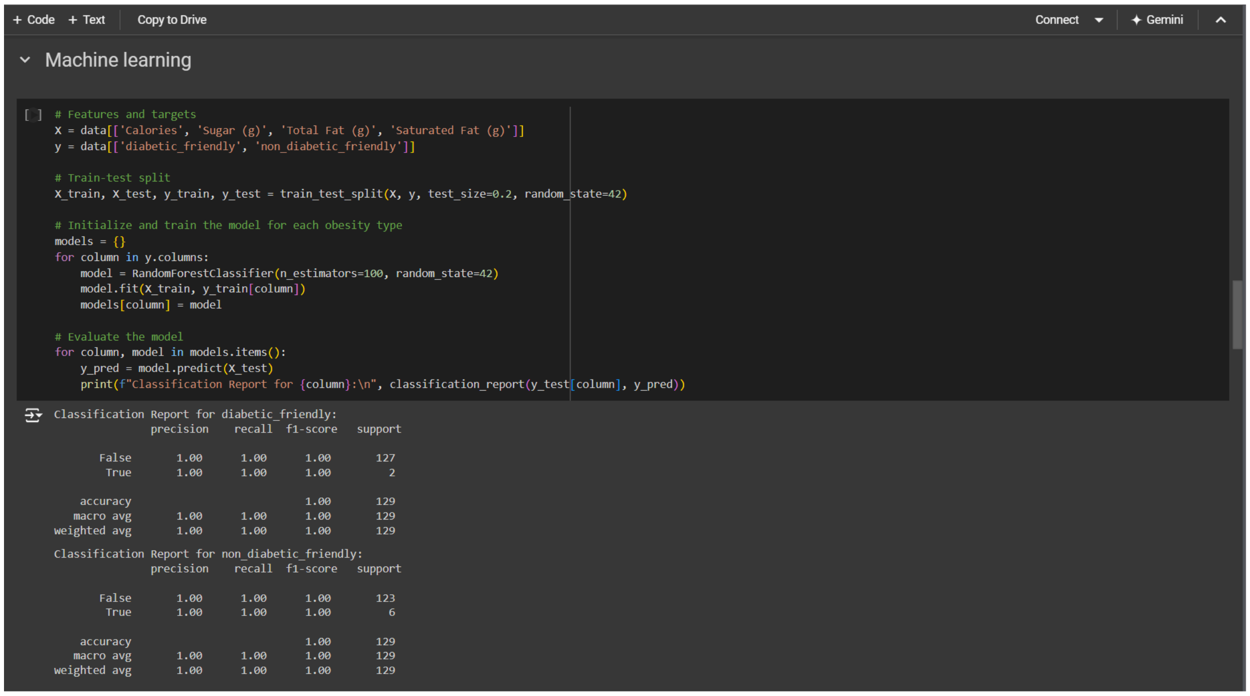
The thresholds for the nutrients were set as follows:

- Diabetic-Friendly: Recipes must have less than 5g of sugar, fewer than 500 calories, and less than 20g of total fat.
- Non-Diabetic-Friendly: Recipes can have up to 10g of sugar, fewer than 800 calories, and less than 30g of total fat.

The criteria script applies these rules to the dataset, creating two new columns—`diabetic_friendly` and `non_diabetic_friendly`—indicating whether each recipe meets the corresponding criteria.

RandomForestClassifier is used for each target, trained separately to classify recipes as either diabetic-friendly or not, the features used are nutritional values like `Calories`, `Sugar (g)`, `Total Fat (g)`, and `Saturated Fat (g)`. The target variables are the newly created columns `diabetic_friendly` and `non_diabetic_friendly`.

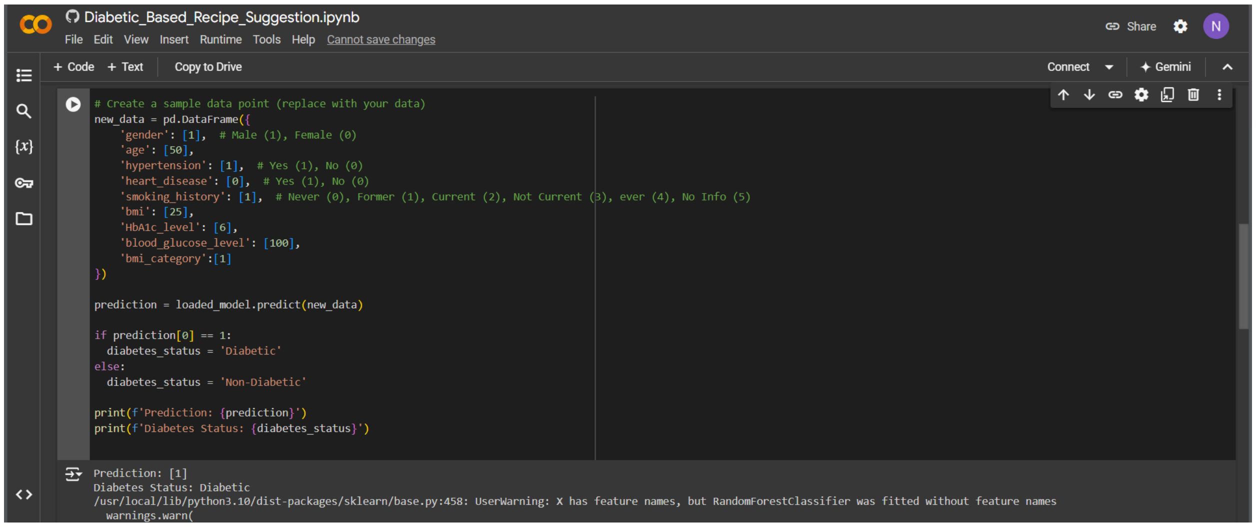
The dataset is split into training and testing sets using an 80/20 split. Each model is trained on the training data and evaluated on the test data.



```python
# Features and targets
X = data[['Calories', 'Sugar (g)', 'Total Fat (g)', 'Saturated Fat (g)']]
y = data[['diabetic_friendly', 'non_diabetic_friendly']]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the model for each obesity type
models = {}
for column in y.columns:
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train[column])
    models[column] = model

# Evaluate the model
for column, model in models.items():
    y_pred = model.predict(X_test)
    print(f"Classification Report for {column}:\n", classification_report(y_test[column], y_pred))
```

```
Classification Report for diabetic_friendly:
               precision    recall  f1-score   support

       False       1.00      1.00      1.00       127
        True       1.00      1.00      1.00         2

    accuracy                           1.00       129
   macro avg       1.00      1.00      1.00       129
weighted avg       1.00      1.00      1.00       129

Classification Report for non_diabetic_friendly:
               precision    recall  f1-score   support

       False       1.00      1.00      1.00       123
        True       1.00      1.00      1.00         6

    accuracy                           1.00       129
   macro avg       1.00      1.00      1.00       129
weighted avg       1.00      1.00      1.00       129
```

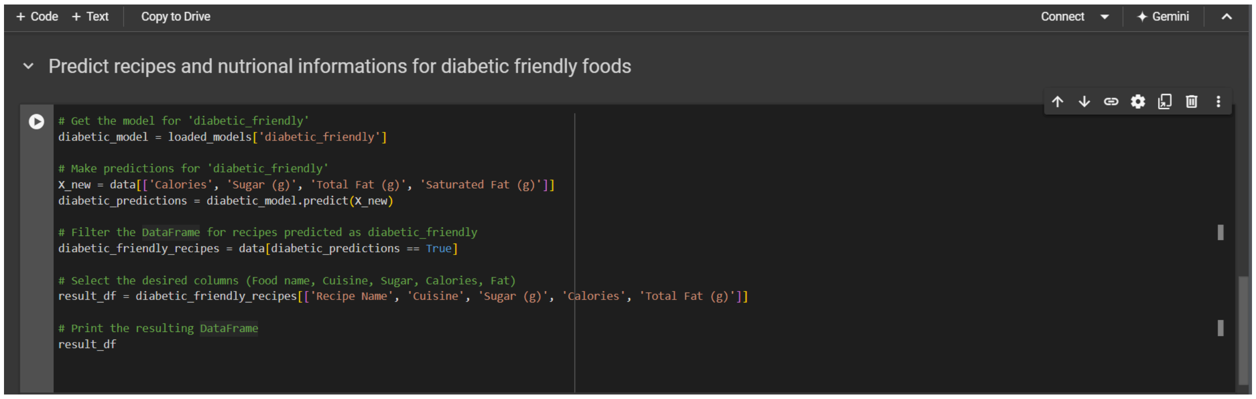
The performance metrics such as precision, recall, and F1-score for both models achieved perfect classification results with the best accuracy. This suggests that the criteria used to separate recipes are distinct enough to allow the Random Forest models to classify the recipes with high precision.



```python
# Create a sample data point (replace with your data)
new_data = pd.DataFrame({
    'gender': [1],  # Male (1), Female (0)
    'age': [50],
    'hypertension': [1],  # Yes (1), No (0)
    'heart_disease': [0],  # Yes (1), No (0)
    'smoking_history': [1],  # Never (0), Former (1), Current (2), Not Current (3), ever (4), No Info (5)
    'bmi': [25],
    'HbA1c_level': [6],
    'blood_glucose_level': [100],
    'bmi_category':[1]
})

prediction = loaded_model.predict(new_data)

if prediction[0] == 1:
  diabetes_status = 'Diabetic'
else:
  diabetes_status = 'Non-Diabetic'

print(f'Prediction: {prediction}')
print(f'Diabetes Status: {diabetes_status}')
```

```
Prediction: [1]
Diabetes Status: Diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:458: UserWarning: X has feature names, but RandomForestClassifier was fitted without feature names
  warnings.warn(
```

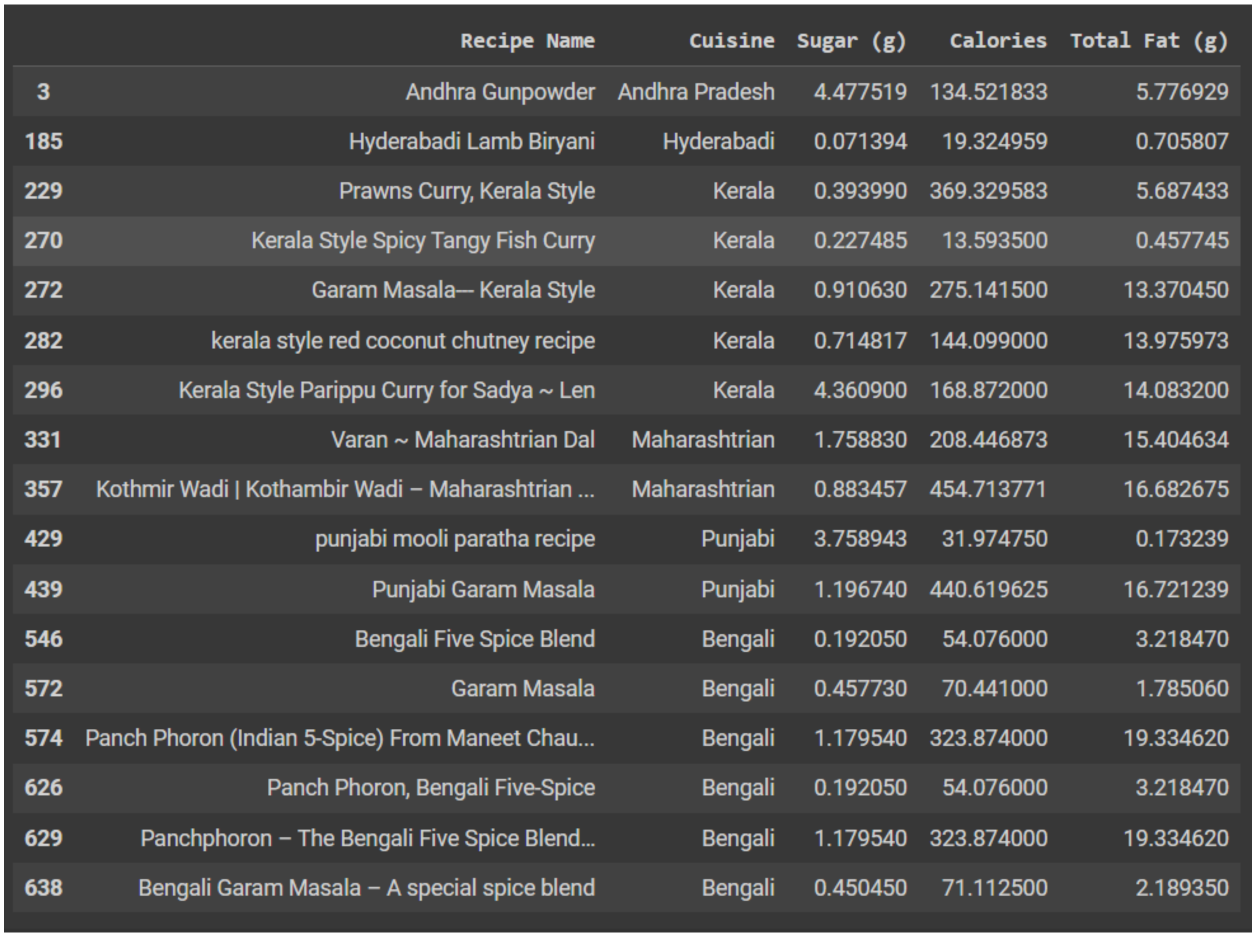
The pre-trained diabetes predicting model is loaded. Initially, a sample data is given input, representing a 50-year-old male with certain health conditions (hypertension but no heart disease) and various biometric factors, such as BMI, HbA1c level, and blood glucose level. The trained diabetes prediction model is used to predict whether this individual is diabetic, returning the result as "Diabetic."



Predict recipes and nutrional informations for diabetic friendly foods

```
# Get the model for 'diabetic_friendly'
diabetic_model = loaded_models['diabetic_friendly']

# Make predictions for 'diabetic_friendly'
X_new = data[['Calories', 'Sugar (g)', 'Total Fat (g)', 'Saturated Fat (g)']]
diabetic_predictions = diabetic_model.predict(X_new)

# Filter the DataFrame for recipes predicted as diabetic_friendly
diabetic_friendly_recipes = data[diabetic_predictions == True]

# Select the desired columns (Food name, Cuisine, Sugar, Calories, Fat)
result_df = diabetic_friendly_recipes[['Recipe Name', 'Cuisine', 'Sugar (g)', 'Calories', 'Total Fat (g)']]

# Print the resulting DataFrame
result_df
```

The model for classifying diabetic-friendly recipes is loaded, and predictions are made based on specific nutritional information, such as `Calories`, `Sugar (g)`, `Total Fat (g)`, and `Saturated Fat (g)`. Recipes predicted to be diabetic-friendly are filtered from the dataset, and the results are presented in a data frame that includes the recipe name, cuisine type, sugar content, calorie count, and total fat content.



	Recipe Name	Cuisine	Sugar (g)	Calories	Total Fat (g)
3	Andhra Gunpowder	Andhra Pradesh	4.477519	134.521833	5.776929
185	Hyderabadi Lamb Biryani	Hyderabadi	0.071394	19.324959	0.705807
229	Prawns Curry, Kerala Style	Kerala	0.393990	369.329583	5.687433
270	Kerala Style Spicy Tangy Fish Curry	Kerala	0.227485	13.593500	0.457745
272	Garam Masala— Kerala Style	Kerala	0.910630	275.141500	13.370450
282	kerala style red coconut chutney recipe	Kerala	0.714817	144.099000	13.975973
296	Kerala Style Parippu Curry for Sadya ~ Len	Kerala	4.360900	168.872000	14.083200
331	Varan ~ Maharashtrian Dal	Maharashtrian	1.758830	208.446873	15.404634
357	Kothmir Wadi | Kothambir Wadi – Maharashtrian ...	Maharashtrian	0.883457	454.713771	16.682675
429	punjabi mooli paratha recipe	Punjabi	3.758943	31.974750	0.173239
439	Punjabi Garam Masala	Punjabi	1.196740	440.619625	16.721239
546	Bengali Five Spice Blend	Bengali	0.192050	54.076000	3.218470
572	Garam Masala	Bengali	0.457730	70.441000	1.785060
574	Panch Phoron (Indian 5-Spice) From Maneet Chau...	Bengali	1.179540	323.874000	19.334620
626	Panch Phoron, Bengali Five-Spice	Bengali	0.192050	54.076000	3.218470
629	Panchphoron – The Bengali Five Spice Blend...	Bengali	1.179540	323.874000	19.334620
638	Bengali Garam Masala – A special spice blend	Bengali	0.450450	71.112500	2.189350

Final Product Prototype

The ReciHealth is designed to offer personalized recipes and nutrition recommendations, focusing on two primary health concerns: obesity and diabetes.

1. **Obesity-Based Recipe Suggestions:** The prototype uses a RandomForestClassifier to categorize recipes according to nutritional limits tailored to various obesity risk types. The classification model is trained on key features such as Weight, Height, and Age, and evaluates recipes based on Sugar, Total Fat, and Saturated Fat thresholds specific to categories ranging from Insufficient Weight to Obesity Type III. The model's high accuracy and robust performance ensure that users receive recommendations that adhere to their dietary needs, promoting better management of obesity-related health risks.
2. **Diabetic-Based Recipe Suggestions:** For managing diabetes, we use RandomForestClassifier to classify recipes as either diabetic-friendly or non-diabetic-friendly. This model considers nutritional guidelines, including sugar content, calorie count, and total fat, to categorize recipes. With a high accuracy rate and effective classification, users with diabetes receive tailored food recommendations that align with their dietary restrictions, supporting better blood sugar management and overall health.

The models process user-specific health data—such as age, weight, and biometric factors for obesity, or health conditions and biomarkers for diabetes—to generate personalized recipe suggestions. The end result is a curated list of recipes that meet individual health criteria, complete with detailed nutritional information, enhancing dietary management and supporting healthier lifestyle choices.

The Prototype Implementations can be accessed at:

- ReciHealth_Obesity_Based_Recipe_Suggestion.ipynb
- ReciHealth_Diabetic_Based_Recipe_Suggestion.ipynb

3. Business Modelling

3.1 Revenue Streams

Subscription-Based Model:

- **Premium Features:** Users can access advanced features such as personalized meal plans, detailed nutrition tracking, and expert consultations.

- **Benefits:** Provides a steady and recurring revenue stream. The subscription can be tiered (e.g., Basic, Standard, Premium) to cater to different user needs and budget levels.
- **Pricing Strategy:** Offer monthly, quarterly, and annual plans to increase user commitment and provide flexibility.

Affiliate Partnerships:

- **Partners:** Collaborate with restaurants, food delivery services, and grocery stores.
- **Revenue Generation:** Earn affiliate commissions from orders or referrals made through the app. Implement a tracking system to ensure accurate commission reporting.
- **Benefits:** Provides a revenue stream based on users' external purchases, extending the app's value proposition and enhancing its ecosystem.

In-App Purchases:

- **Offerings:** Users can buy specific diet plans, fitness programs, or curated recipe packs tailored to their health goals.
- **Benefits:** Generates additional revenue from users who seek more customized or advanced content beyond what is available through the subscription model.
- **Pricing Strategy:** Offer individual purchases or bundles to cater to varying user needs and preferences.

Advertising:

- **Targeted Ads:** Integrate relevant ads for health-related products or services within the platform's free version.
- **Benefits:** Generates revenue from advertisers while keeping the app accessible to users who may not be ready to commit to a subscription.
- **Ad Strategy:** Focus on health-conscious and relevant advertisements to maintain user engagement and app relevance.
- **Marketing and Outreach:** Promote the app through digital marketing, influencer partnerships, and targeted campaigns.

3.2 Sustainable Growth Strategy

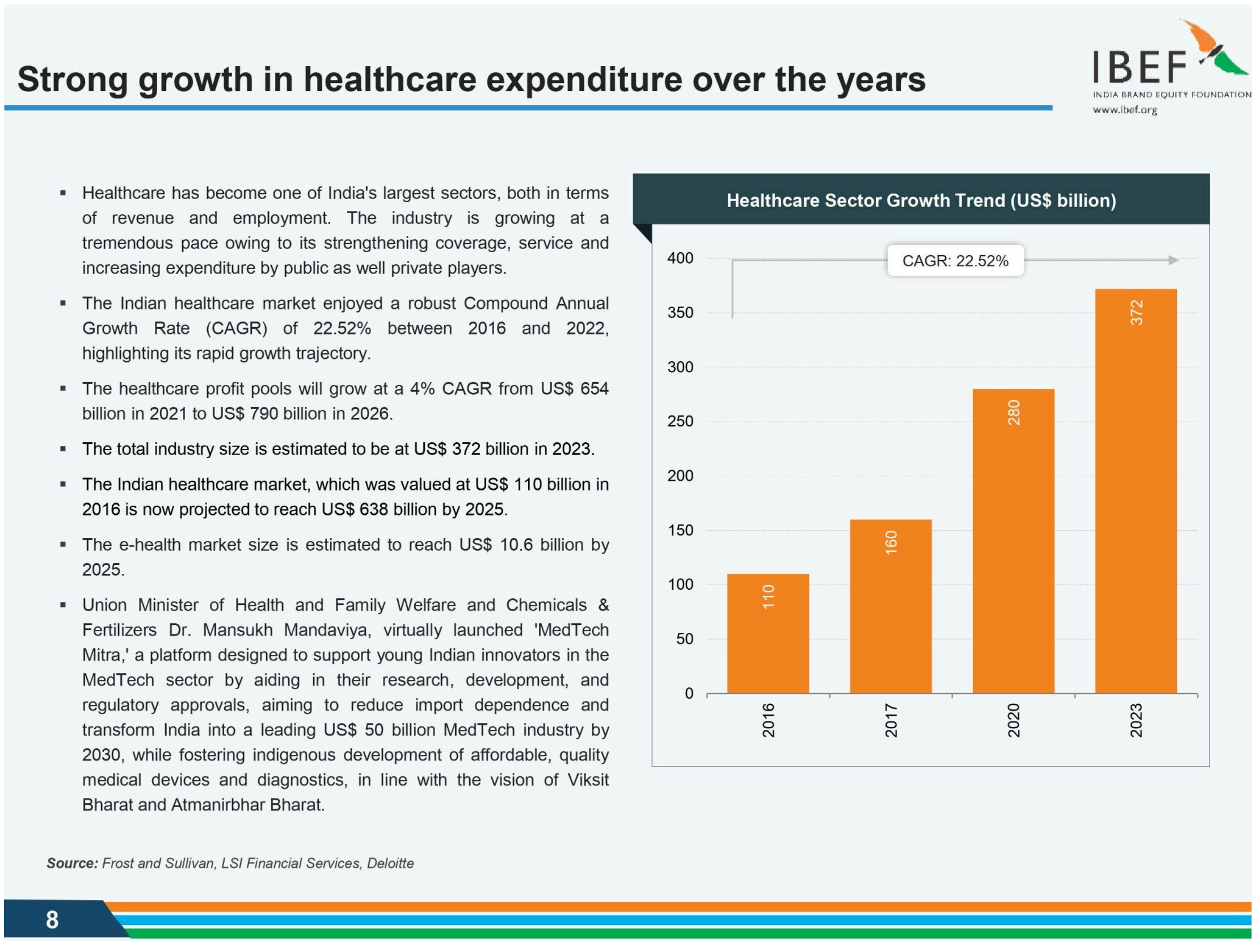
- **User Acquisition and Retention:** Continuously improve the AI algorithms and expand partnerships to enhance user experience and attract new users.
- **Engagement:** Regular updates and feature additions to keep users engaged and subscribed.

4. Financial Modelling

4.1 Identifying the Market

ReciHealth will be launched into the Indian health and digital wellness market, specifically targeting sectors like personalized nutrition, health-conscious consumers, and fitness enthusiasts. The app will cater to individuals with specific dietary needs (e.g., diabetes, obesity, hypertension), fitness goals, and a growing interest in healthy eating habits.

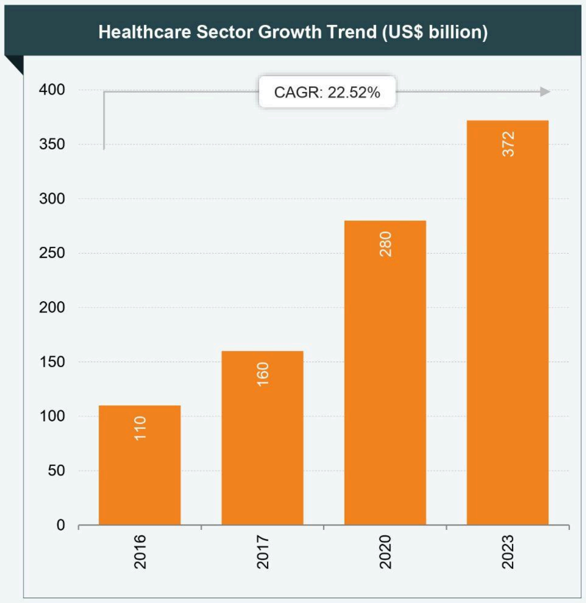
Indian Brand Equity Foundation (IBEF): Indian Healthcare Industry Analysis



Strong growth in healthcare expenditure over the years

IBEF INDIA BRAND EQUITY FOUNDATION www.ibef.org

- Healthcare has become one of India's largest sectors, both in terms of revenue and employment. The industry is growing at a tremendous pace owing to its strengthening coverage, service and increasing expenditure by public as well private players.
- The Indian healthcare market enjoyed a robust Compound Annual Growth Rate (CAGR) of 22.52% between 2016 and 2022, highlighting its rapid growth trajectory.
- The healthcare profit pools will grow at a 4% CAGR from US$ 654 billion in 2021 to US$ 790 billion in 2026.
- The total industry size is estimated to be at US$ 372 billion in 2023.
- The Indian healthcare market, which was valued at US$ 110 billion in 2016 is now projected to reach US$ 638 billion by 2025.
- The e-health market size is estimated to reach US$ 10.6 billion by 2025.
- Union Minister of Health and Family Welfare and Chemicals & Fertilizers Dr. Mansukh Mandaviya, virtually launched 'MedTech Mitra,' a platform designed to support young Indian innovators in the MedTech sector by aiding in their research, development, and regulatory approvals, aiming to reduce import dependence and transform India into a leading US$ 50 billion MedTech industry by 2030, while fostering indigenous development of affordable, quality medical devices and diagnostics, in line with the vision of Viksit Bharat and Atmanirbhar Bharat.

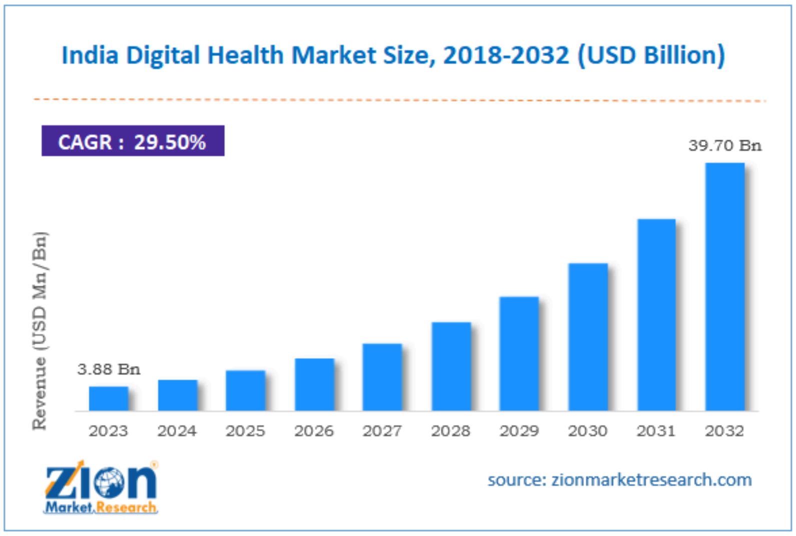


Source: *Frost and Sullivan, LSI Financial Services, Deloitte*

8

ZION Market Research: India Digital Health Industry

Indian digital health market size was worth around USD 3.88 billion in 2023 and is predicted to grow to around USD 39.70 billion by 2032 with a compound annual growth rate (CAGR) of roughly 29.5% between 2024 and 2032. Increasing rates of chronic diseases such as diabetes, respiratory conditions, and cardiovascular diseases are seen as putting pressure on the global healthcare system.



ZION Market Research - India Digital Health Market: Segmentation

Indian digital health industry is segmented based on **technology, component, application, end-user, and region.**

Based on the technology, the Indian digital health market is bifurcated into tele-healthcare, mHealth, digital health systems, and healthcare analytics. The tele-healthcare segment is expected to dominate the market over the forecast period. This growth rate is explained by improving internet connectivity, an increase in smartphone usage, increased technological readiness, a growing scarcity of healthcare professionals, rising medical costs, the accessibility of telehealth applications, and an increase in patient & physician adoption of these technologies. The ever-expanding telehealth applications and swift technical advancements contribute to the segment's growth. The main drivers of the segment's growth include increasing healthcare IT spending, government backing, and legislation encouraging healthcare digitization.

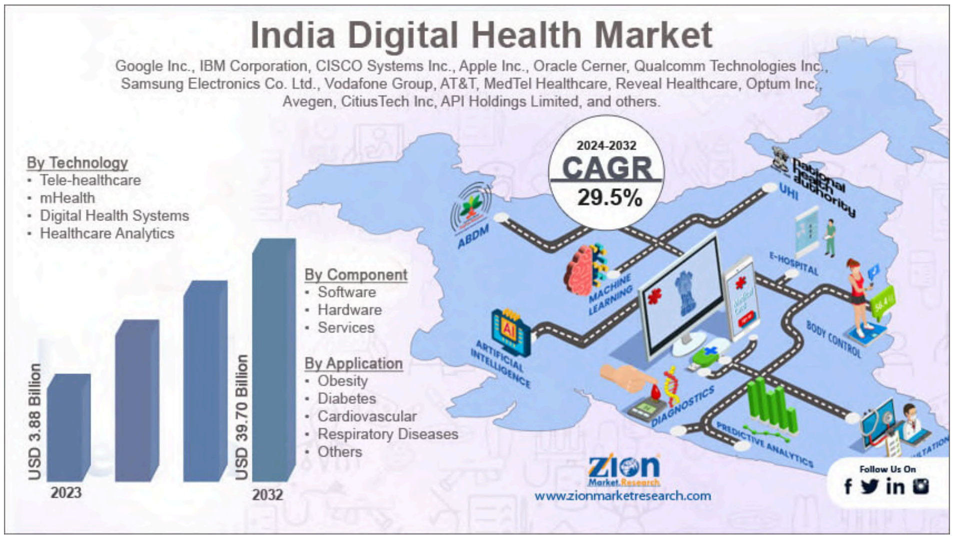
Based on the components, the Indian digital health industry is segmented into software, hardware, and services. The services segment is expected to capture the largest market share over the forecast period. Revenue has increased dramatically, especially since the

COVID-19 epidemic has boosted the usage of telemedicine services. Partnerships between insurers and healthcare providers, telehealth platform subscription models, and more consultations are the main factors driving revenue development. Revenue has also increased thanks to electronic health records (EHRs), hospital management systems (HMS), and other health IT technologies. These technologies are being purchased by healthcare facilities to increase patient care, operational effectiveness, and regulatory compliance.

Based on the application, the Indian digital health market is segmented into obesity, diabetes, cardiovascular, respiratory diseases, and others. The diabetes segment is expected to garner a significant revenue share over the forecast period. Diabetes is the largest segment in the digital health space due to its prevalence and related problems. The requirements of people with diabetes can be creatively met by digital health technologies. Digital health tools allow patients to take an active role in controlling their diabetes. These tools range from smartphone applications for glucose testing to wearable gadgets that track physical activity and provide real-time health data. Additionally, these technologies enable remote patient monitoring, which in turn helps healthcare providers manage diabetes more effectively by enabling timely data delivery, educated decision-making, and prompt treatments.

Based on the end-use, the Indian digital health industry is segmented into patients, providers, payers, and others. The patient segment is expected to capture a significant market share over the forecast period because of the movement in healthcare toward patient-centered care and people's increased understanding of health management. By giving patients, the means to manage their care, access health records, and engage in remote monitoring, digital health technologies have completely changed the healthcare industry.

Based on region (State Analysis), **Maharashtra, Tamil Nadu, Karnataka, Punjab, Rajasthan,** and **Uttar Pradesh** were found as the key players. Among them, **Maharashtra** is expected to dominate the market over the forecast period. Maharashtra is expected to dominate the Indian digital health market over the forecast period. Pune and Mumbai are two major centers for the uptake and development of digital health. The usage of telemedicine, mHealth apps, and online health services is made easier by the high level of smartphone accessibility and digital literacy among metropolitan populations. Moreover, Maharashtra has a far more developed healthcare infrastructure than other states, which facilitates the use of telemedicine platforms, hospitals, and electronic health records (EHRs). In addition, telemedicine has become more and more common in Maharashtra since it makes getting medical care easier and does not require physical travel to clinics or hospitals, as demonstrated by the COVID-19 epidemic. Thus, driving the market growth.



4.2 Financial Equation for the Market

Assumptions:

- **Product Cost:** ₹500 per month for a premium subscription.
- **Cost of Business Operations (Fixed Cost):** ₹2,00,000 per month (for running the app, marketing, and employee salaries).
- **Projected Sales:** Forecast that ReciHealth will have 10,000 premium users in the first month and expect a monthly growth rate of 10%.

4.2.1 Financial Equation:

Revenue = Product Price × Number of Premium Users - Fixed Costs

Let **x** represent the number of premium users.

Revenue Equation: $R(x) = 500x - 2,00,000$

- **Month 1 Revenue (10,000 users):**
  $R(10,000) = 500(10,000) - 2,00,000 = ₹48,00,000$
- **Month 2 (10% growth in users):**
  $R(11,000) = 500(11,000) - 2,00,000 = ₹53,00,000$

Thus, the profit increases with user growth, and the expected 10% increase in users results in an additional ₹5,00,000 in profit by the second month. For small-scale businesses or startups like **ReciHealth**, a subscription-based model offers strong potential for early profitability. With 10,000 users, the first-month profit can be expected to reach around **₹48,00,000**, and a 10% growth in users increases it to **₹53,00,000** in the second month. This shows how quickly a scalable service can generate profit with controlled operational costs.