Elements OF AI/ML

Document File Containing the Working of Model

Name : Gaurav Srivastava

SAP : 500122467

ROLL : R2142230319

BATCH : 11

Working of the Flask App :

→ This is the home page of the flask app , i.e. , app.py which is linked with my Machine Learning Model , i.e. , Diabetic-Model.pkl.

→ Now the user enters real time data in the flask app for each features.

- First we enter that data that should predict Diabetes in the final predication.

  Pregnancies : 2

  Glucose : 120

  Blood Pressure : 80

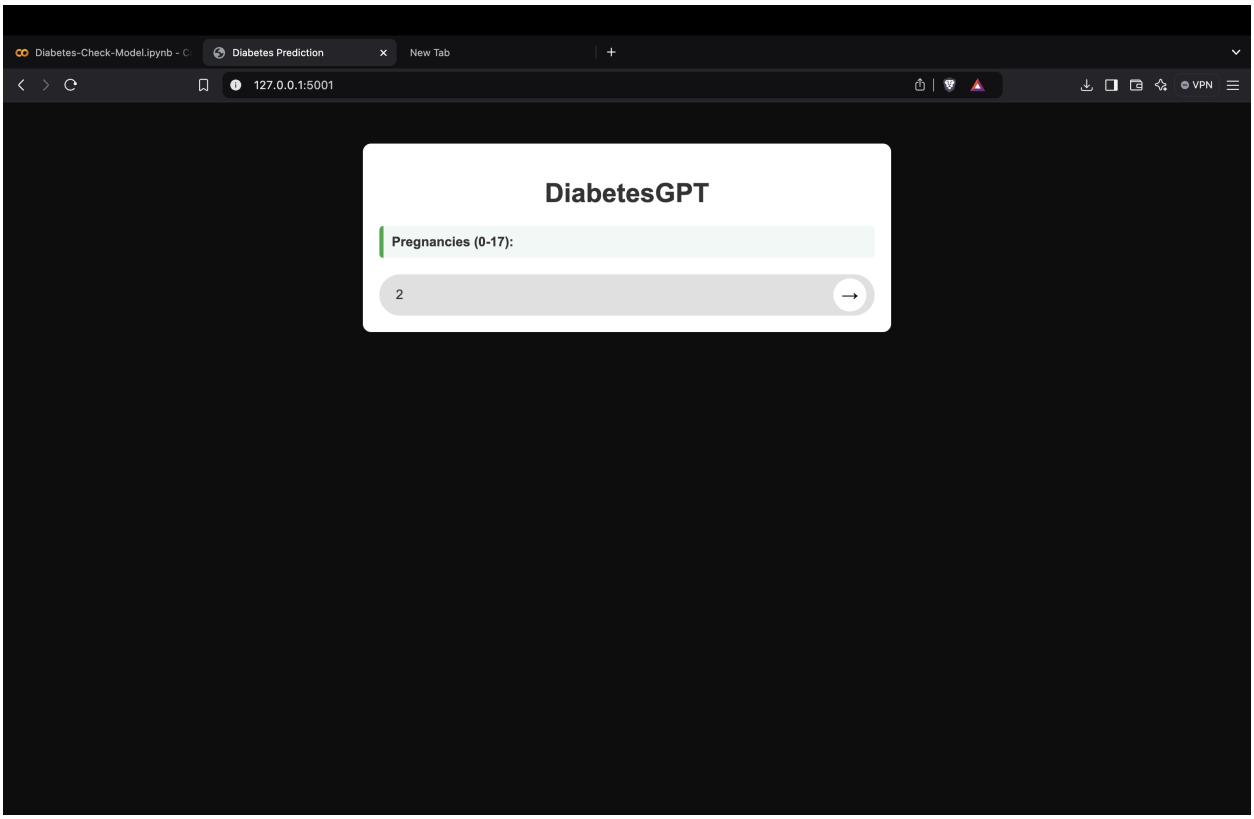
  Skin Thickness : 28.5

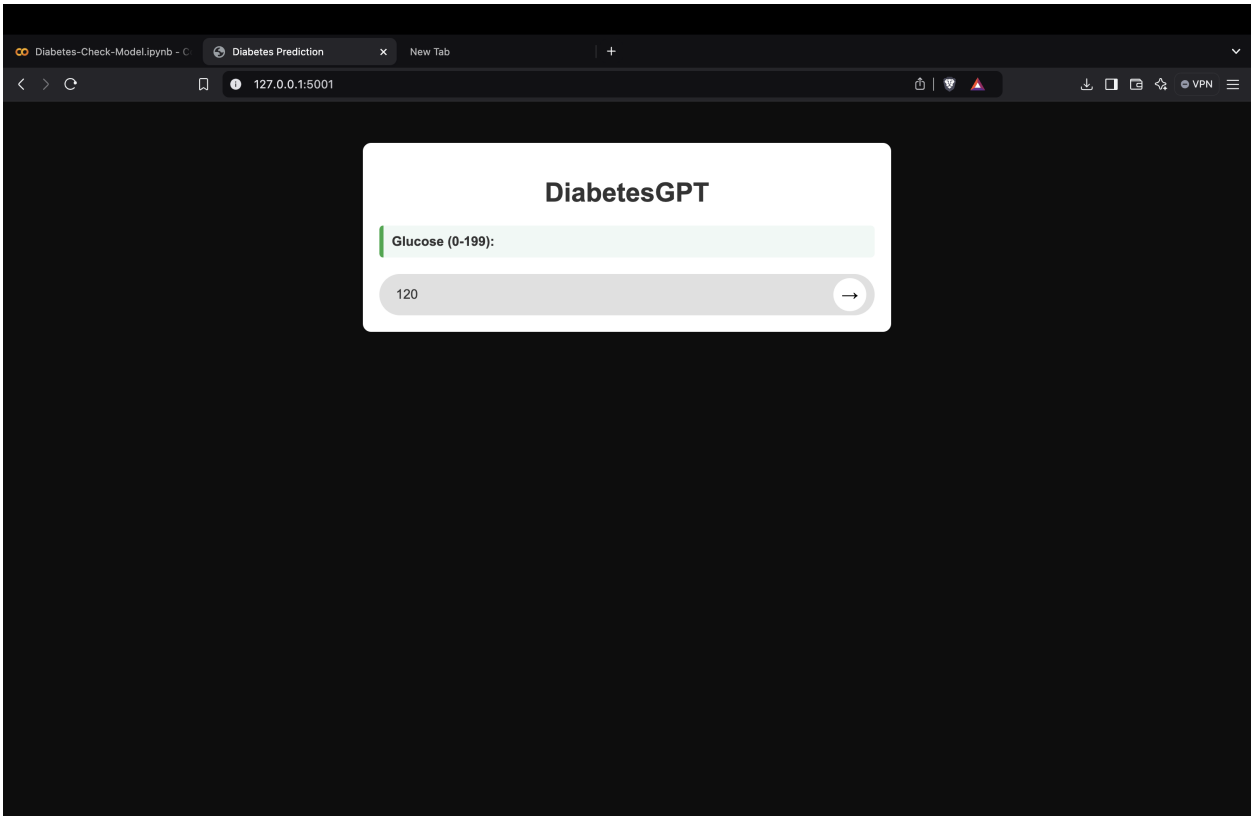
  Insulin : 80

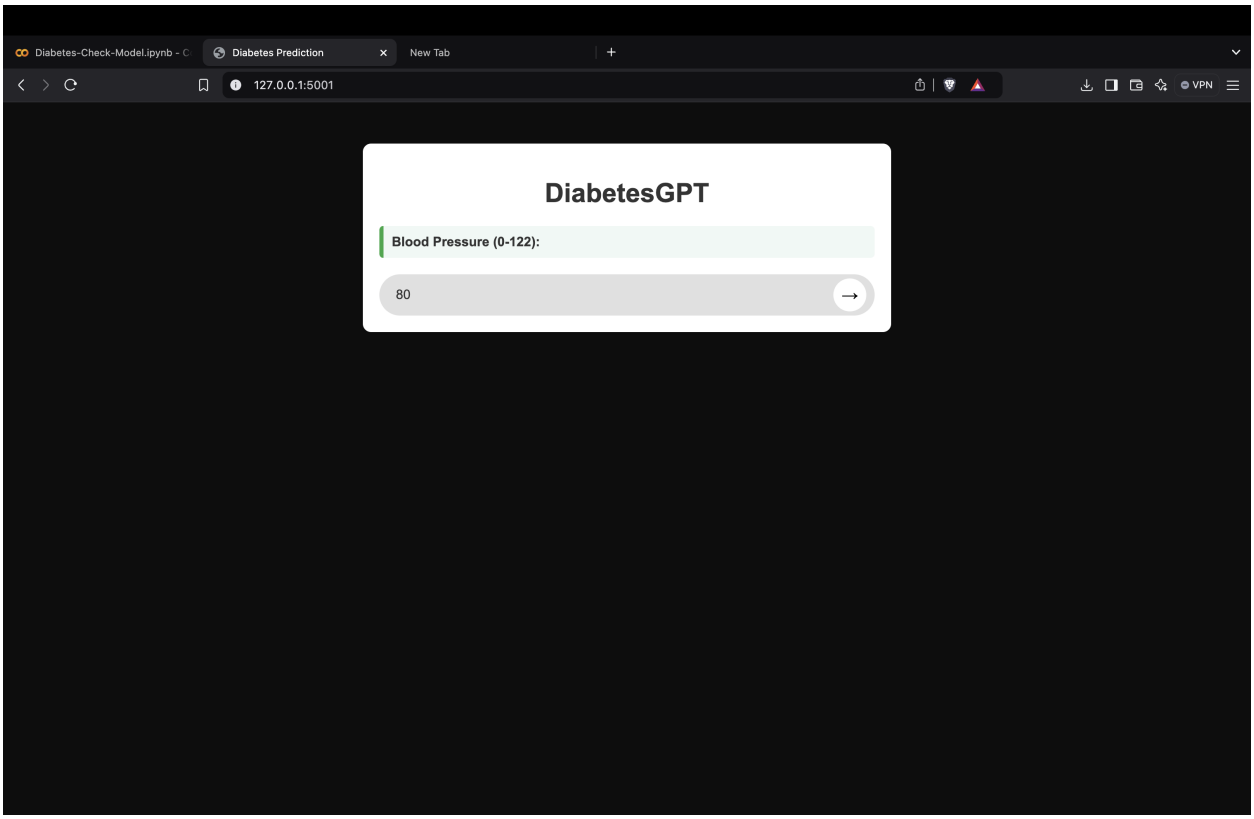
  BMI : 28

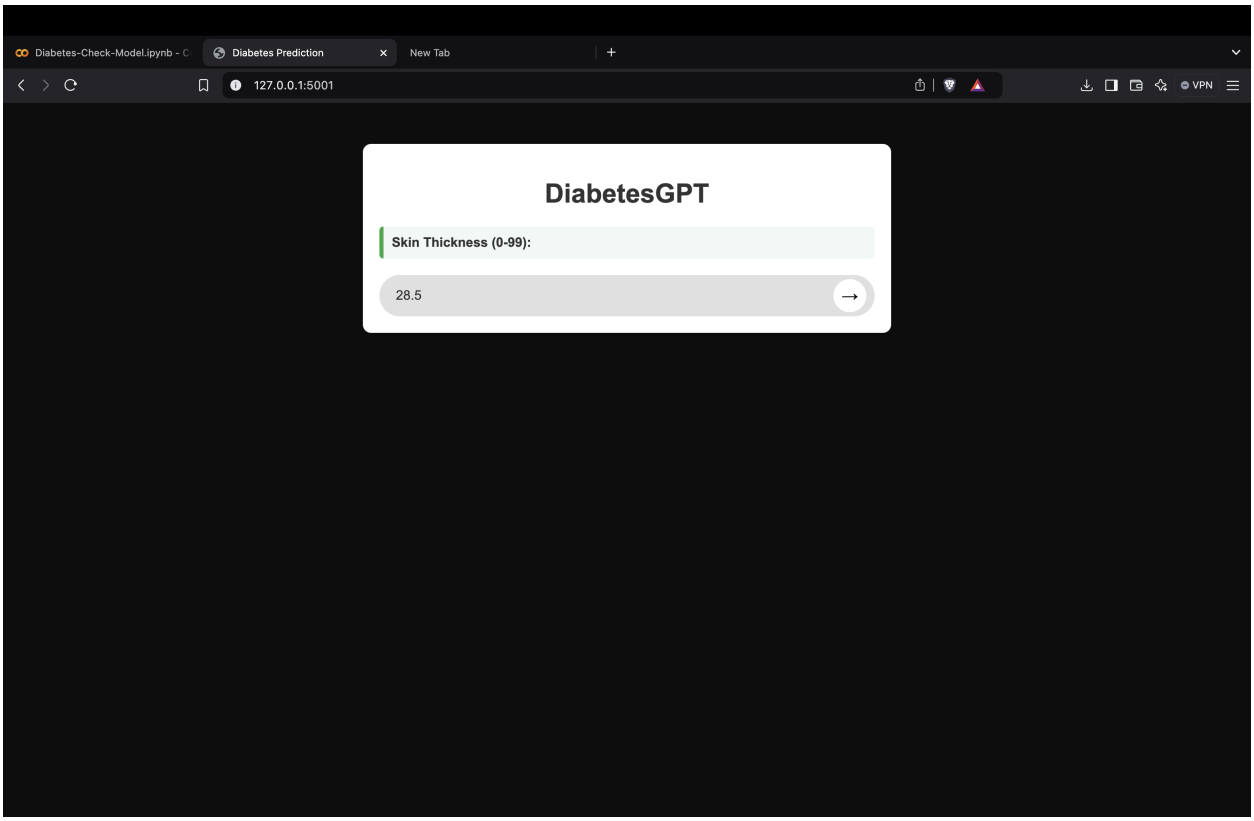
  Diabetic Pedigree Function : 0.45

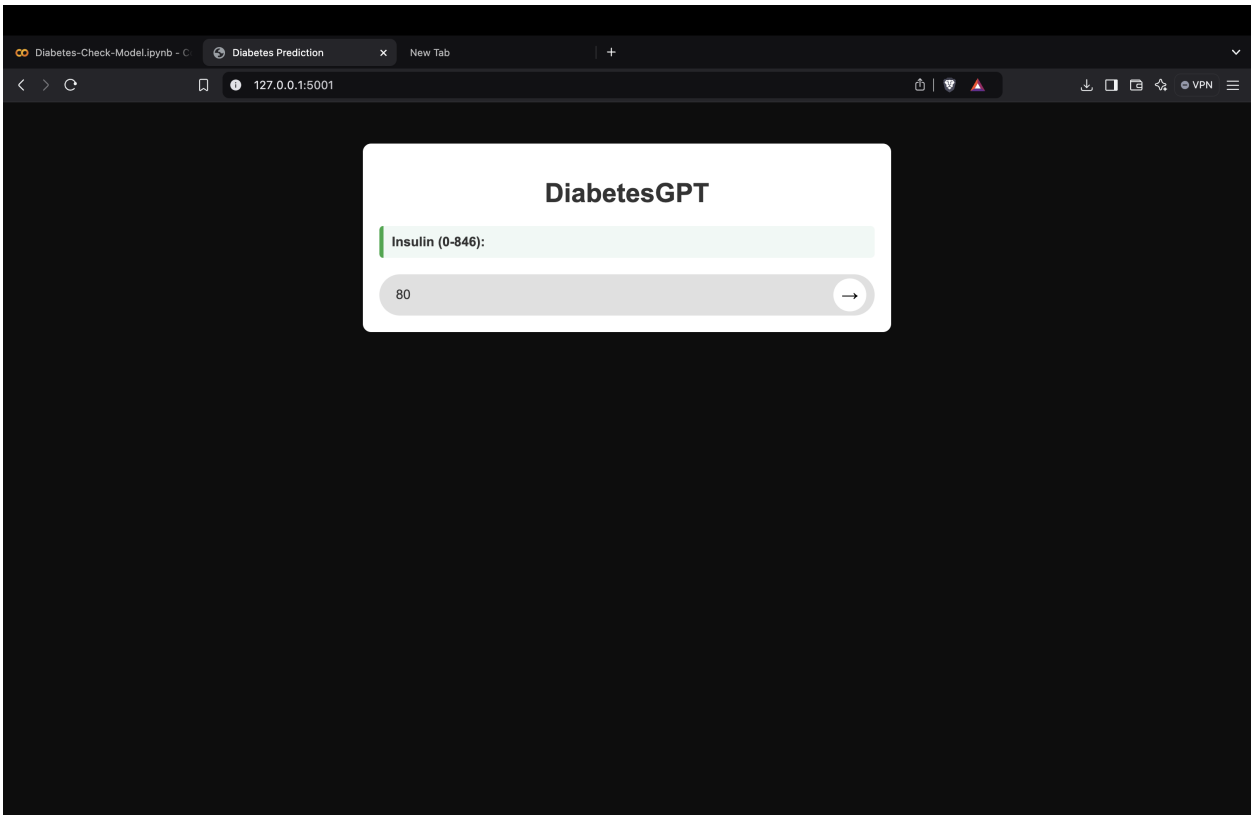
  Age : 45

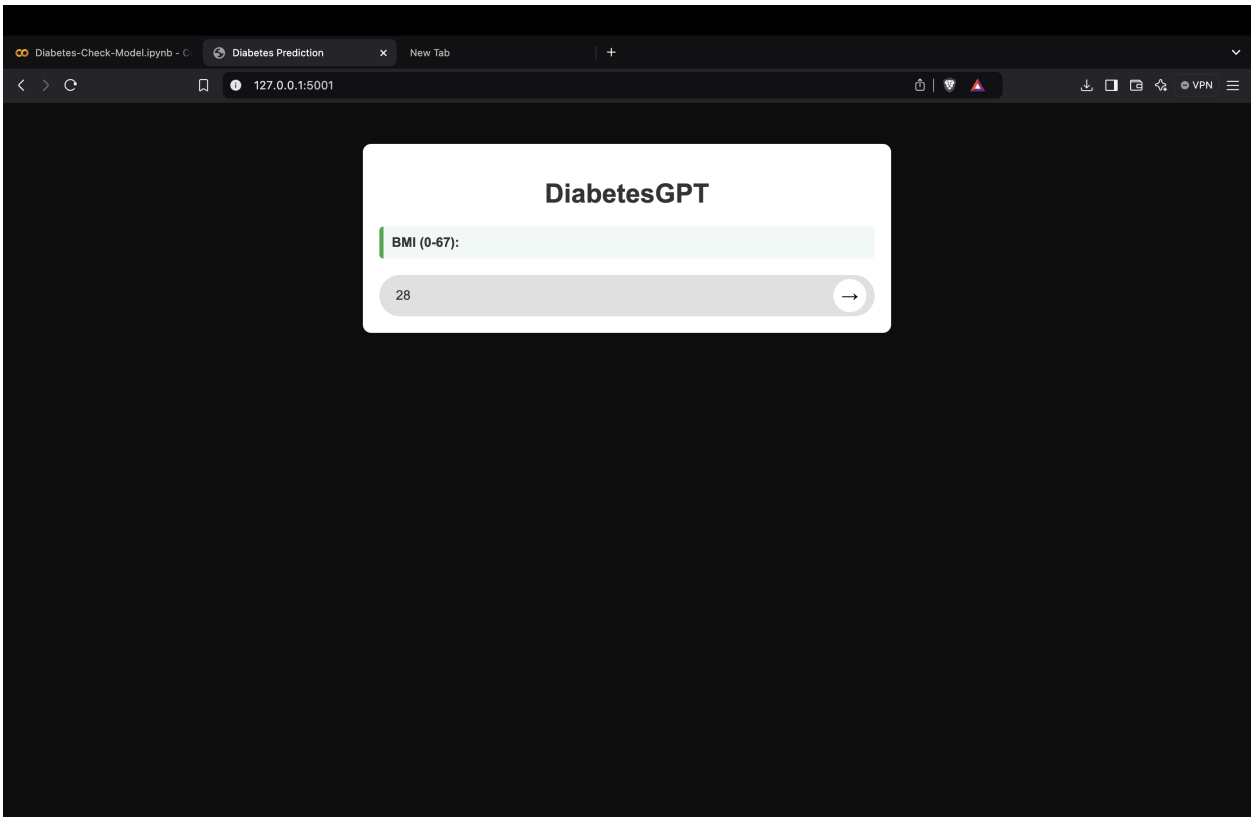

Diabetes-Check-Model.ipynb - C
Diabetes Prediction
New Tab
127.0.0.1:5001
VPN
DiabetesGPT
Pregnancies (0-17):
2

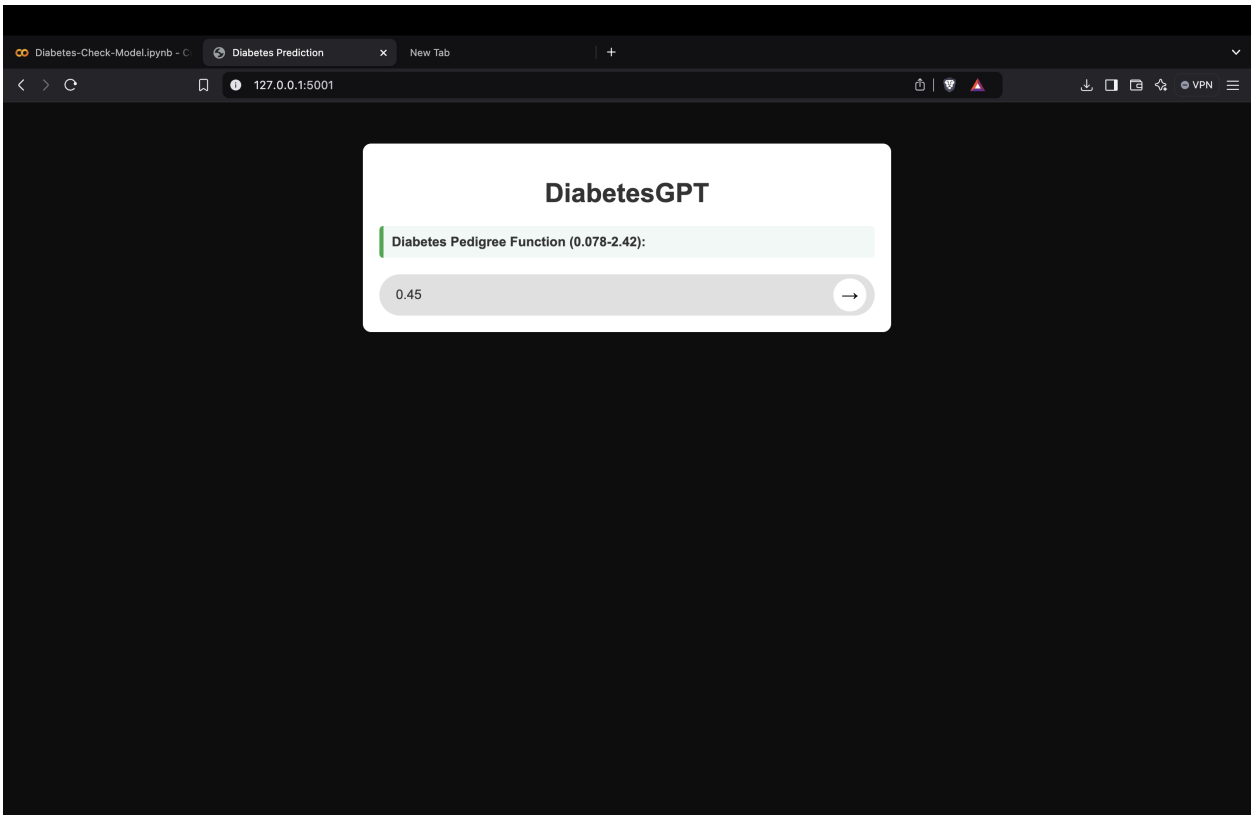

Diabetes-Check-Model.ipynb - C
Diabetes Prediction
New Tab
127.0.0.1:5001
VPN
DiabetesGPT
Glucose (0-199):
120

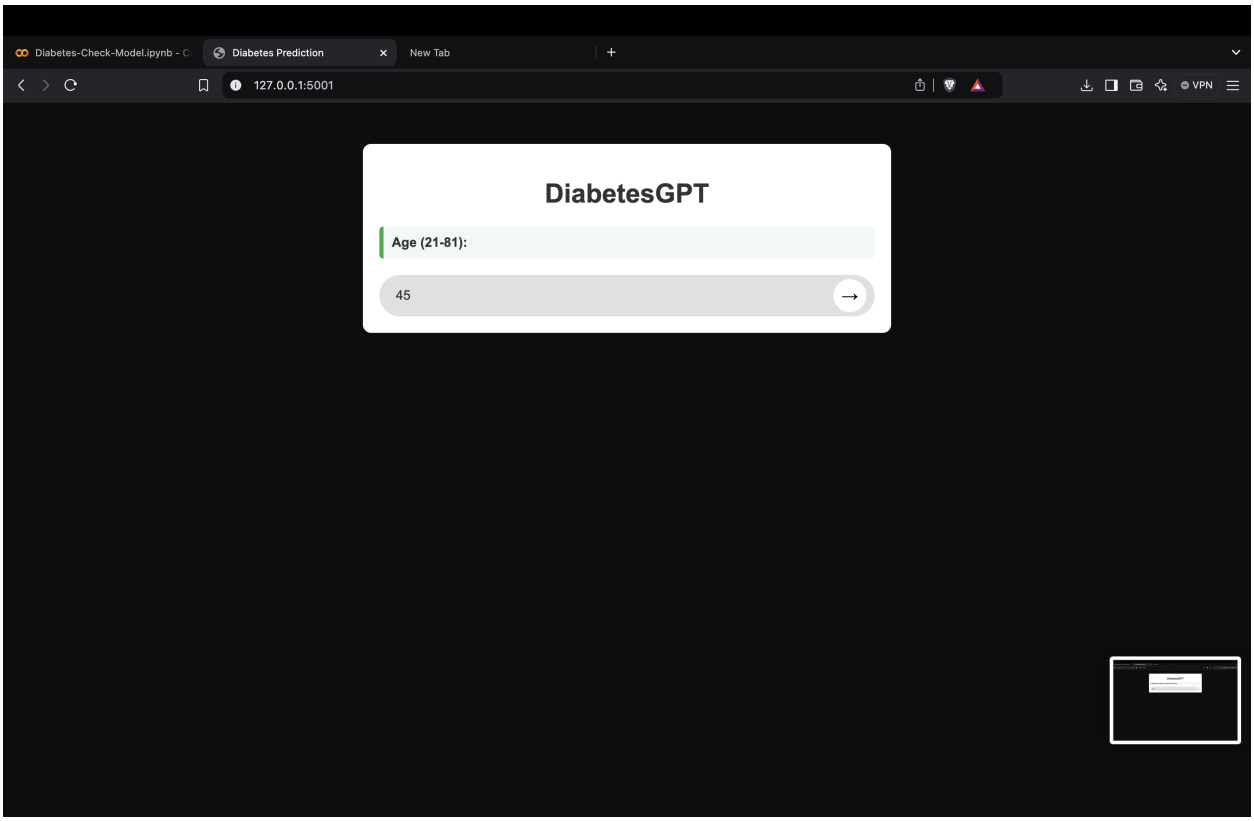

Diabetes-Check-Model.ipynb - C
Diabetes Prediction
New Tab
127.0.0.1:5001
VPN
DiabetesGPT
Blood Pressure (0-122):
80

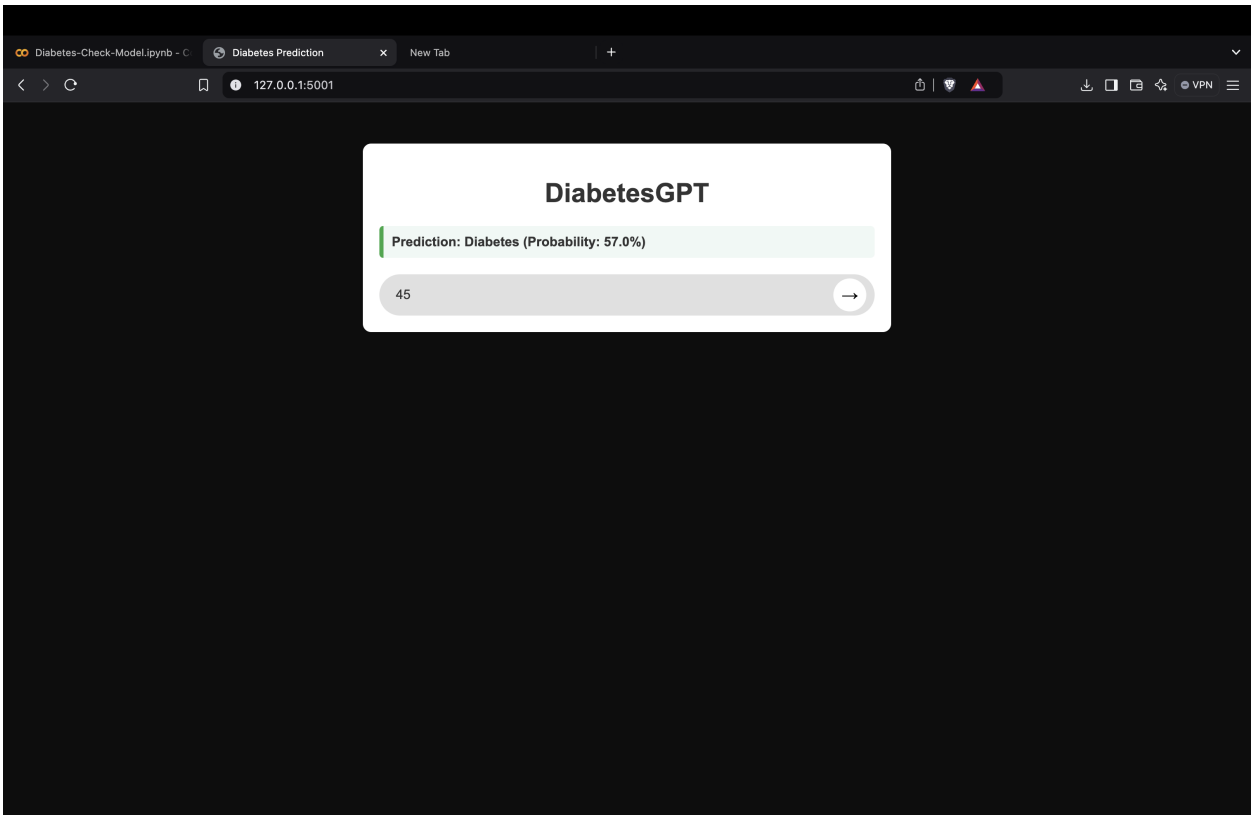

Diabetes-Check-Model.ipynb - C
Diabetes Prediction
New Tab
127.0.0.1:5001
VPN
DiabetesGPT
Skin Thickness (0-99):
28.5


Diabetes-Check-Model.ipynb - C
Diabetes Prediction
New Tab
127.0.0.1:5001
VPN
DiabetesGPT
Insulin (0-846):
80


Diabetes-Check-Model.ipynb - C
Diabetes Prediction
New Tab
127.0.0.1:5001
VPN
DiabetesGPT
BMI (0-67):
28

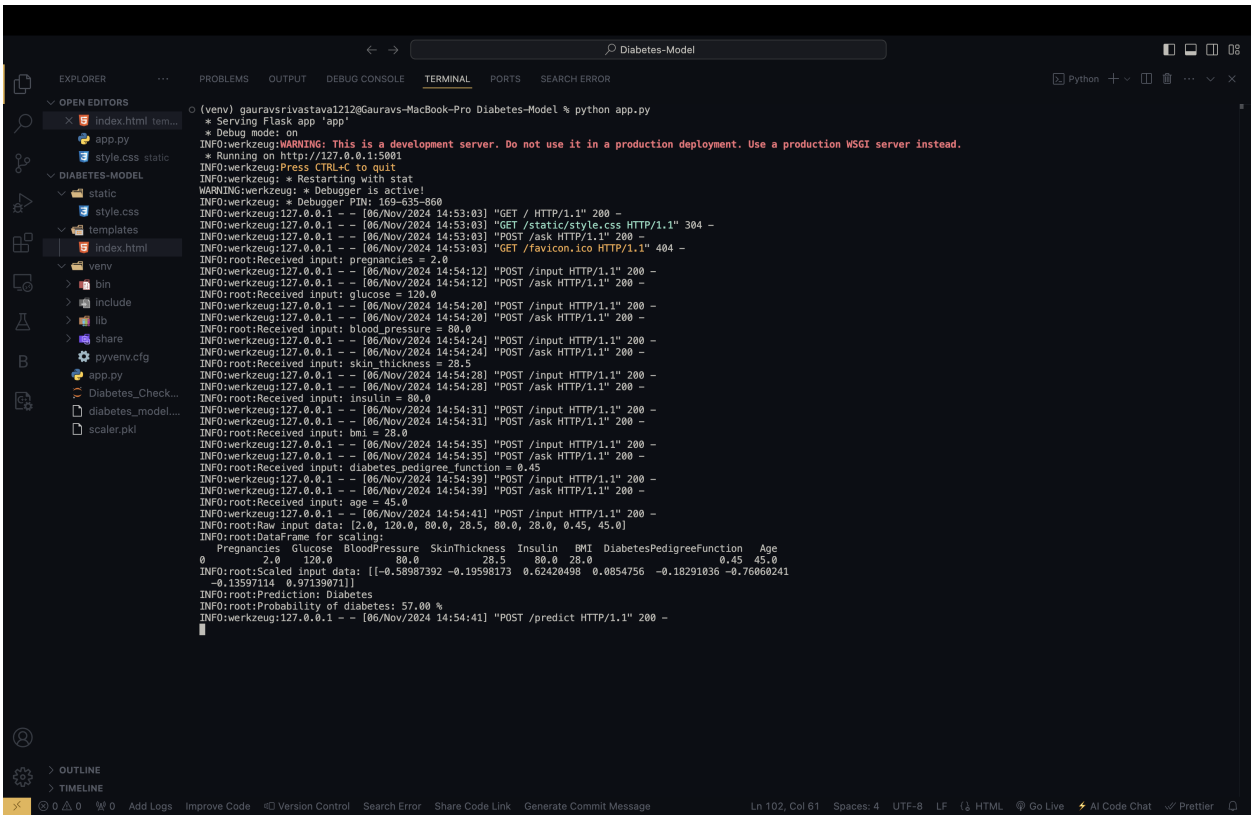

Diabetes-Check-Model.ipynb - C
Diabetes Prediction
New Tab
127.0.0.1:5001
VPN
DiabetesGPT
Diabetes Pedigree Function (0.078-2.42):
0.45
→


Diabetes-Check-Model.ipynb - C
Diabetes Prediction
New Tab
127.0.0.1:5001
VPN
DiabetesGPT
Age (21-81):
45
→



Hence it predicts : `57.0 %` ,i.e. , chances of this person getting diabetic !

At the backend , we also do console.log to see that user entered data is successfully getting extracted for the model prediction or not :



Now we look into our model working :

1. Data Import and Preprocessing:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_repo

# Load the dataset
data = pd.read_csv('diabetes.csv')

# Split features and target
X = data.drop('Outcome', axis=1)
```

```
y = data['Outcome']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_s

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

This segment imports necessary libraries, loads the dataset, splits it into features (X) and target (y), then further splits into training and testing sets. It also scales the features using StandardScaler.

2. Model Training:

```
# Train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state
rf_model.fit(X_train_scaled, y_train)
```

Here, a Random Forest Classifier is initialized and trained on the scaled training data.

3. Model Evaluation:

```
# Make predictions on the test set
y_pred = rf_model.predict(X_test_scaled)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

This section uses the trained model to make predictions on the test set, calculates the accuracy, and prints a detailed classification report.

4. Feature Importance:

```
# Get feature importance
feature_importance = rf_model.feature_importances_
feature_names = X.columns

# Sort features by importance
sorted_idx = np.argsort(feature_importance)
sorted_features = feature_names[sorted_idx]

# Plot feature importance
plt.figure(figsize=(10, 6))
plt.barh(range(len(feature_importance)), feature_importance[sort
plt.yticks(range(len(feature_importance)), sorted_features)
plt.xlabel('Importance')
plt.title('Feature Importance')
plt.show()
```

This part extracts and visualizes the importance of each feature in the Random Forest model's decision-making process.

5. Model Persistence:

```
import joblib

# Save the model
joblib.dump(rf_model, 'Diabetic-Model.pkl')

# Save the scaler
joblib.dump(scaler, 'scaler.pkl')
```

Finally, this segment saves both the trained model and the scaler used for feature scaling. These saved files can be later used in the Flask application for making predictions on new data.