Project Report: Diabetes Prediction Web Application

Introduction

In today's world, where health concerns are increasingly prevalent, our project aims to provide a user-friendly web application that predicts diabetes risk based on individual health data. By leveraging machine learning, we can offer insights that empower users to take charge of their health.

Technologies Used

Python: The backbone of our application, facilitating all backend processes.
Flask: A lightweight framework that helps us create a smooth web experience.
Pandas & NumPy: Essential tools for data manipulation and analysis, making it easy to handle our dataset.
Scikit-learn: The powerhouse for our machine learning model, specifically the Random Forest Classifier.
Pickle: A handy module for saving and loading our trained model.

Dataset

Location: Our dataset is stored at `AD_Project/AD_Project/Diabaties/diabetes_prediction_dataset.csv`.
Content: This dataset contains various health metrics, which we use to predict whether an individual is likely to develop diabetes.

Application Structure

1.Training the Model:
- We first load the dataset and check for any missing values, ensuring our data is clean.
- The target variable (the outcome we want to predict) is identified, and features are prepared.

- A model is trained using a Random Forest algorithm, which is known for its accuracy and robustness.
- The trained model and the features used are saved for future use.

2.Loading the Model:
- When the application starts, it tries to load the previously trained model.
- If the model isn't found, it automatically retrains it, ensuring we always have a current model ready to make predictions.

3.User Interface:
- The home page welcomes users and allows them to input their health data easily.
- Users can submit their information via a web form or through an API, making it versatile for different needs.
- Predictions come back quickly, along with probabilities to give users a clearer picture of their risk.

4.Model Retraining:
- Users can request model retraining through a dedicated route, ensuring that the application stays up-to-date with the latest data.

Code Highlights

Key Functions

-train_model(): This function handles everything from loading the dataset to training the model and saving it for later use.

-load_model(): Responsible for fetching the model from storage or initiating training if it's not available.

-predict(): This is where the magic happens! It processes user inputs and returns predictions.

-retrain(): Allows for refreshing the model with new data when needed.

Example Code Snippet

```
'''python
def train_model():
  Load and preprocess the dataset
   df = pd.read_csv("AD_Project/AD_Project/Diabaties/diabetes_prediction_dataset.csv")
  ... (rest of the training logic)
   rf_model.fit(X_train, y_train)
   Save the model
   with open('models/random_forest.pkl', 'wb') as file:
      pickle.dump(rf_model, file)
```

Models Used in the Diabetes Prediction Project

1. Logistic Regression:Logistic Regression is a popular statistical method used to predict outcomes that can fall into two categories (like yes/no or 1/0). It works by estimating the probability that a given input point belongs to a particular category using a logistic function.

How We Used It:

- We employed the LogisticRegression class from the sklearn library.
- To ensure the model converges properly during training, we set the maximum number of iterations to 1000.

2. Decision Tree Classifier:A Decision Tree Classifier is like a flowchart where each node represents a decision based on the input features. It splits the data into branches to make predictions, which makes it easy to understand and interpret.

How We Used It:

- We used the DecisionTreeClassifier from sklearn, relying on its default settings to create our model.

Both models were trained on the same set of data, allowing us to compare their performance. After training, we evaluated how well each model predicted diabetes outcomes based on a separate test dataset.

Example Code Snippet

```
# Train Logistic Regression model
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)

# Train Decision Tree model
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)
```

Results

Our application successfully predicts diabetes based on user inputs, providing valuable insights into their health risks. During training, we also print the model's accuracy and a classification report, giving transparency to our users.

Conclusion

This project is a testament to how technology can aid in health management. By creating an accessible tool for diabetes prediction, we aim to empower individuals to make informed health decisions.

Future Directions

- We plan to enhance the model's accuracy by exploring different algorithms and tuning parameters.
- Improving the user interface will make the application even more engaging and user-friendly.
- Adding features for data visualization could provide users with deeper insights into their health data.