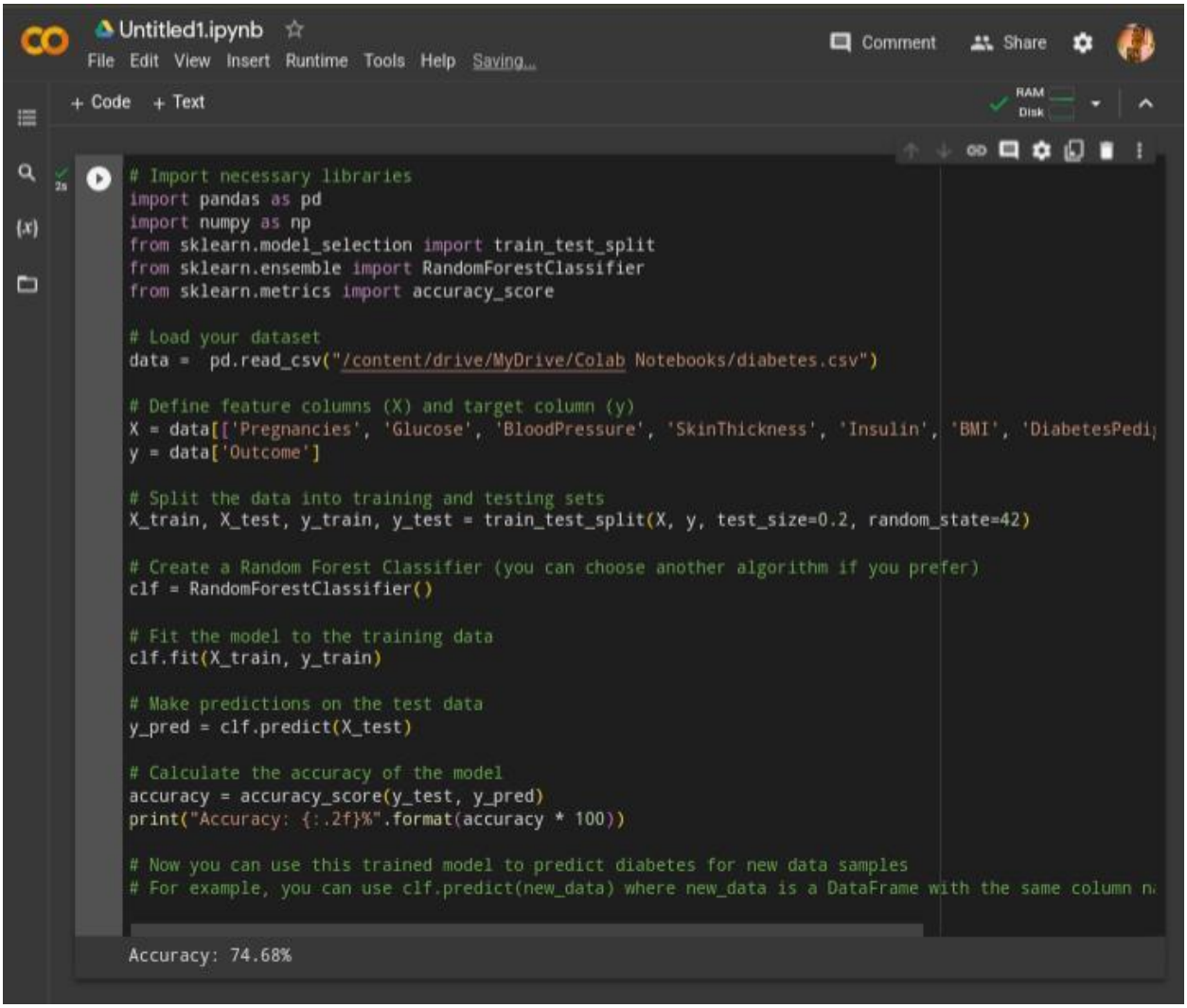
AI based diabetes prediction system

Overview

This Python code is designed to create an AI-based diabetes prediction system using a dataset with the following columns: 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', and 'Outcome'. It employs the Random Forest Classifier for prediction.



Prerequisites

Before using this code, ensure that you have the following libraries installed:

- pandas

- numpy

- scikit-learn

1. Load Dataset

- The code begins by loading your dataset from a CSV file. Replace `"your_dataset.csv"` with the path to your dataset file.

```
# Load your dataset
Dataset_path = "/content/drive/MyDrive/Colab Notebooks/diabetes.csv"
Data = pd.read_csv(dataset_path)
```

2. Define Feature and Target Columns

- Specify the feature columns (X) and the target column (y) based on your dataset's column names.

```
# Define feature columns (X) and target column (y)
X = data['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]
Y = data['Outcome']
```

3. Split Data

- The data is split into training and testing sets using `train_test_split`. Adjust the `test_size` and `random_state` parameters as needed.

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4. Model Creation

- A Random Forest Classifier is used for building the prediction model. You can replace it with another machine learning algorithm of your choice.

```
# Create a Random Forest Classifier (you can choose another algorithm if you prefer)
Clf = RandomForestClassifier()
```

5. Model Training

- The model is trained using the training data with `clf.fit(X_train, y_train)`.

```
# Fit the model to the training data
Clf.fit(X_train, y_train)
```

6. Make Predictions

- Predictions are made on the test data with `clf.predict(X_test)`.

 # Make predictions on the test data

Y_pred = clf.predict(X_test)

7. Evaluate Model Performance

- The accuracy of the model is calculated and displayed. You can add more evaluation metrics as needed.

8.Prediction for New Data

- You can use the trained model to predict diabetes for new data samples by calling `clf.predict(new_data)` where `new_data` is a DataFrame with the same column names.

 # Calculate the accuracy of the model

Accuracy = accuracy_score(y_test, y_pred)

*Print("Accuracy: {:.2f}%".format(accuracy * 100))*

 # Create a simple bar plot to visualize accuracy

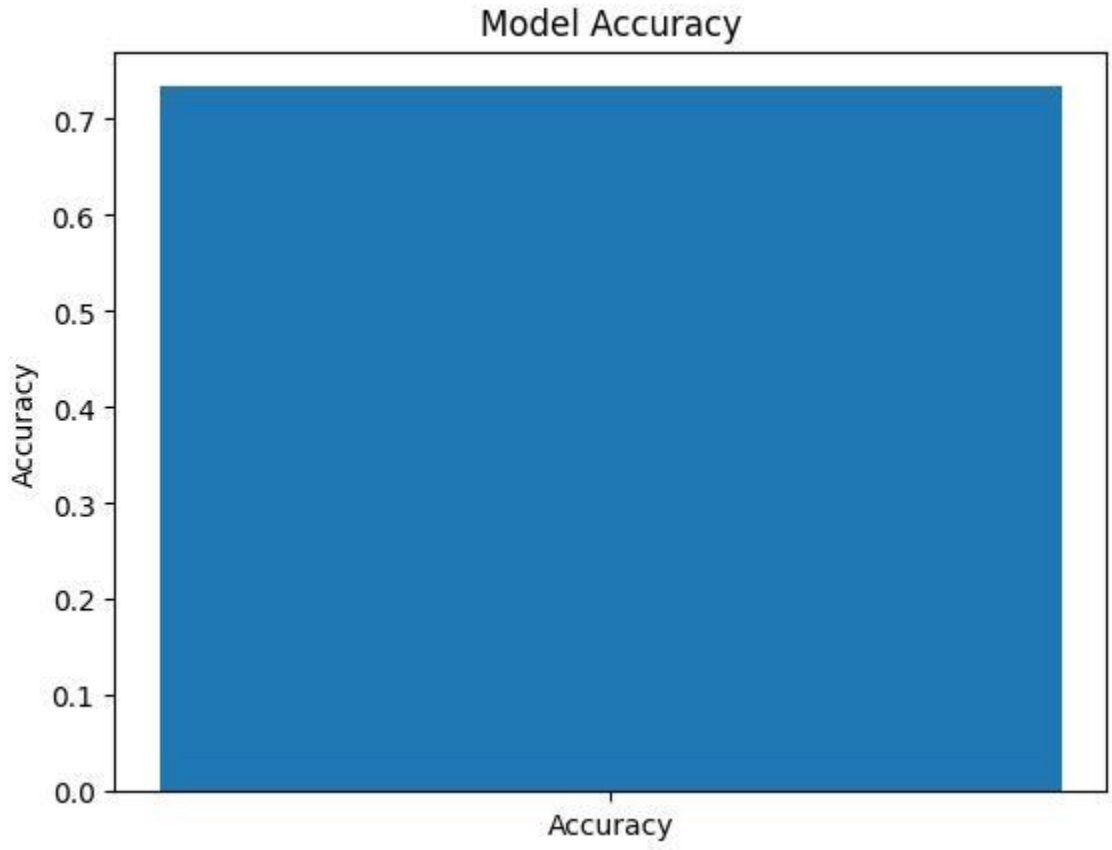
Plt.bar(['Accuracy'], [accuracy])

Plt.title('Model Accuracy')

Plt.ylabel('Accuracy')

Plt.show()

Output :



Accuracy :73.38%

Conclusion:

This code is a simplified example. In real-world applications, more comprehensive data preprocessing and model optimization would be necessary for building a robust diabetes prediction system.