Diabetes Prediction Using Support Vector Machine (SVM)

By: Riya Sharma

Date: June 2024

1. Introduction

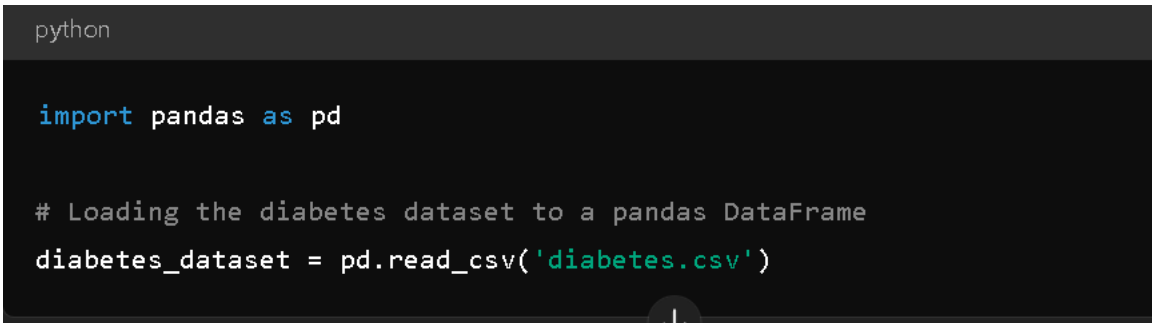
Diabetes is a chronic medical condition that affects how your body turns food into energy. Early detection and diagnosis are crucial for effective management. In this project, we use a Support Vector Machine (SVM) classifier to predict diabetes outcomes based on patient data.

2. Dataset

The dataset used in this project is sourced from Kaggle and originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset contains various medical features for patients, including whether they have diabetes or not.

2.1 Loading the Dataset

Explanation: This code snippet demonstrates how to load the dataset from a CSV file into a pandas Data Frame, a common data structure used in Python for data manipulation.



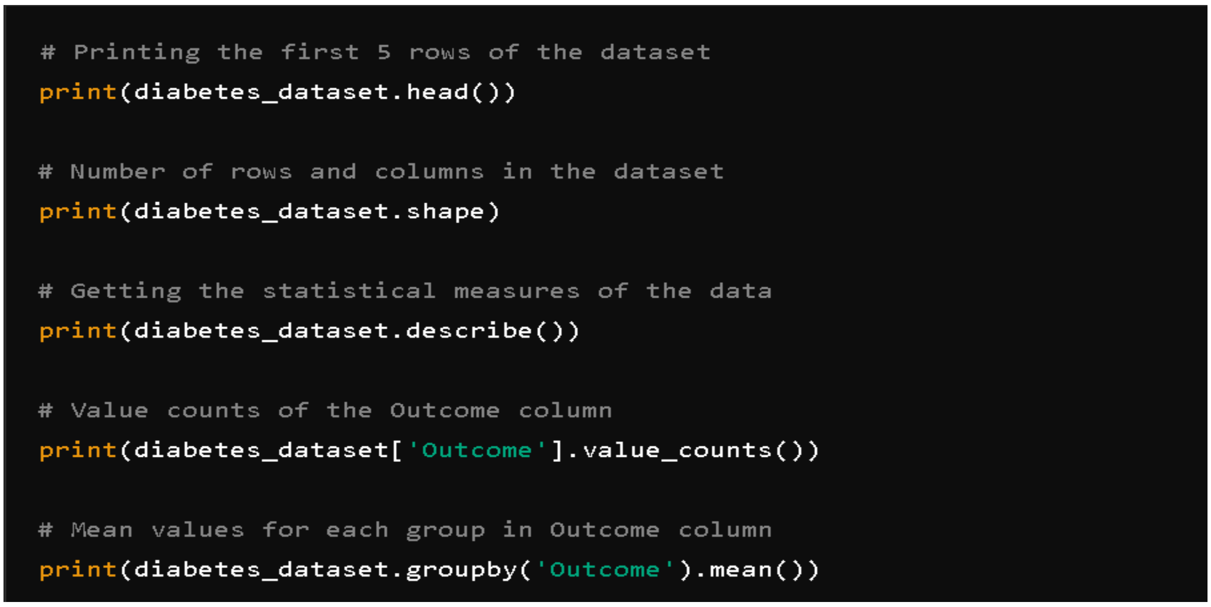
```python
import pandas as pd

# Loading the diabetes dataset to a pandas DataFrame
diabetes_dataset = pd.read_csv('diabetes.csv')
```

2.2 Data Overview

Explanation:

- First 5 rows: Provides a glimpse of the data, helping to understand its structure and contents.
- Shape: Shows the number of rows and columns, giving an idea of the dataset's size.

- Statistical measures: Offers summary statistics (mean, standard deviation, etc.) for each feature, helping to understand the data distribution.
- Value counts: Displays the distribution of the target variable (Outcome), indicating how many instances of each class are present.
- Group means: Shows the average values of features grouped by the outcome, providing insights into how features differ between classes.



```python
# Printing the first 5 rows of the dataset
print(diabetes_dataset.head())

# Number of rows and columns in the dataset
print(diabetes_dataset.shape)

# Getting the statistical measures of the data
print(diabetes_dataset.describe())

# Value counts of the Outcome column
print(diabetes_dataset['Outcome'].value_counts())

# Mean values for each group in Outcome column
print(diabetes_dataset.groupby('Outcome').mean())
```

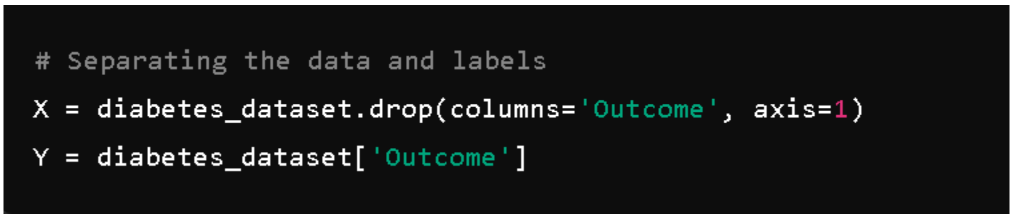
3. Data Preprocessing

Explanation:

Data preprocessing is essential for preparing the data for machine learning models. This section includes separating features from labels and standardizing the data to ensure that all features contribute equally to the model.

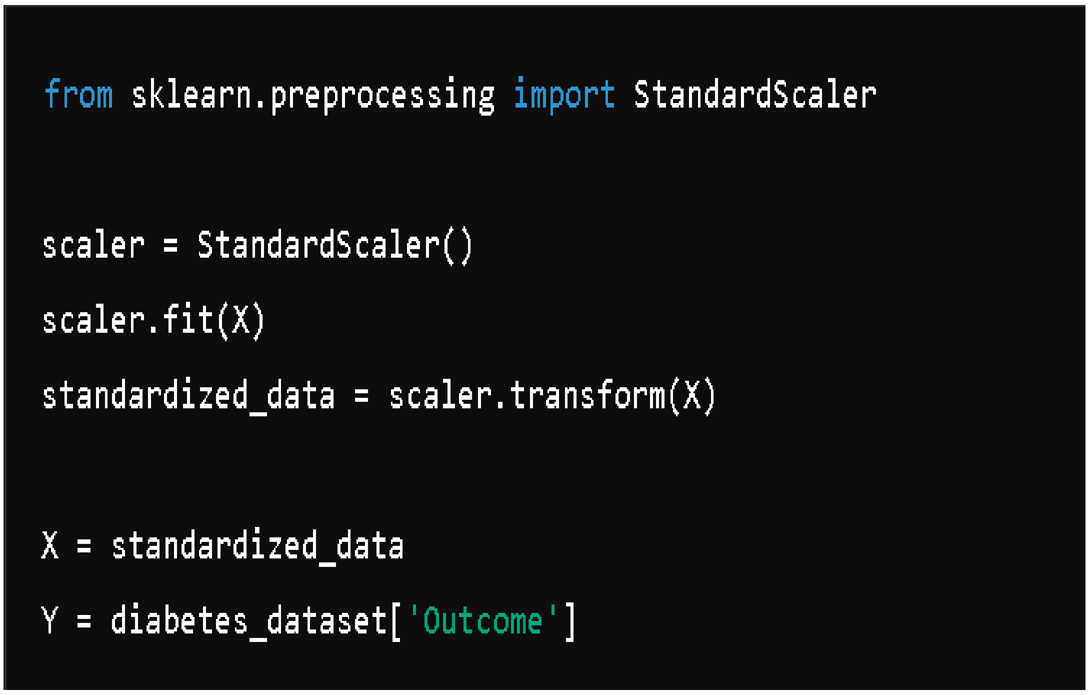
3.1 Separating Features and Labels

Explanation: Here, X contains the features (input variables), and Y contains the labels (the target variable). The Outcome column is dropped from X and kept as Y.



```
# Separating the data and labels
X = diabetes_dataset.drop(columns='Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

3.2 Standardizing the Data



```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X)
standardized_data = scaler.transform(X)

X = standardized_data
Y = diabetes_dataset['Outcome']
```

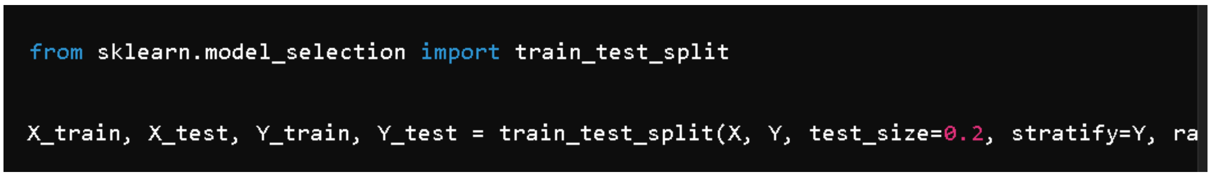
Explanation: Standardization is applied to normalize the feature values so that they have a mean of 0 and a standard deviation of 1. This ensures that features with different scales do not disproportionately affect the model.

4. Model Training and Evaluation

Explanation:

This section covers the process of splitting the data into training and testing sets, training the SVM classifier, and evaluating its performance.

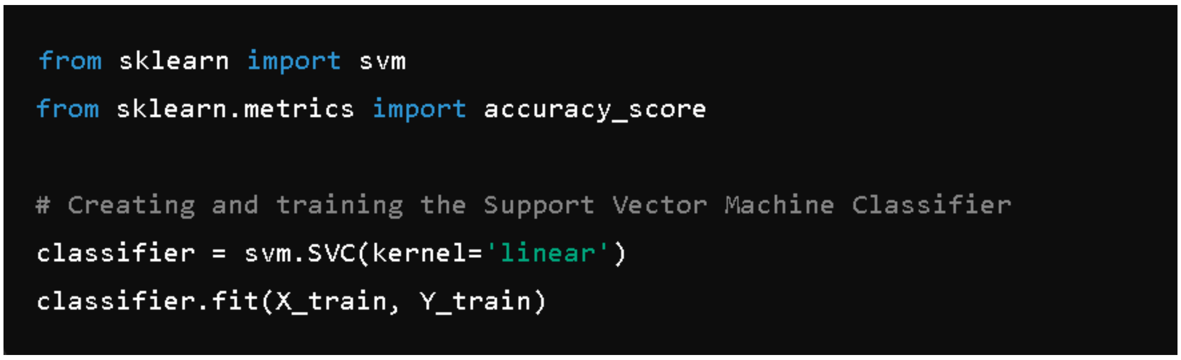
4.1 Splitting the Data



```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, ra
```

Explanation: The dataset is divided into training and testing sets. The test_size=0.2 parameter means 20% of the data is used for testing. stratify=Y ensures that the split maintains the proportion of classes in both training and testing sets.

4.2 Training the Model



```python
from sklearn import svm
from sklearn.metrics import accuracy_score

# Creating and training the Support Vector Machine Classifier
classifier = svm.SVC(kernel='linear')
classifier.fit(X_train, Y_train)
```

Explanation: An SVM classifier with a linear kernel is created and trained on the training data. SVM is a powerful classification algorithm that finds the optimal hyperplane to separate classes.

4.3 Evaluating the Model

```
# Accuracy score on the training data
X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
print('Accuracy score of the training data : ', training_data_accuracy)

# Accuracy score on the test data
X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score of the test data : ', test_data_accuracy)
```

Explanation: The accuracy of the model is evaluated on both training and test datasets. Accuracy measures how well the model's predictions match the true labels. This helps assess the model's performance and generalizability

5. Making Predictions

5.1 Input Data

Explanation:

This section demonstrates how to use the trained model to make predictions on new, unseen data.

```
input_data = (5, 166, 72, 19, 175, 25.8, 0.587, 51)
```

5.2 Preparing the Input Data

```python
import numpy as np

# Changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# Reshaping the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)

# Standardizing the input data
std_data = scaler.transform(input_data_reshaped)
```

Explanation: The input data is converted to a numpy array, reshaped to match the input format expected by the model, and standardized using the same scaler applied to the training data.

5.3 Making a Prediction

```python
# Making a prediction
prediction = classifier.predict(std_data)
print(prediction)

if (prediction[0] == 0):
    print('The person is not diabetic')
else:
    print('The person is diabetic')
```

Sure! Here's a detailed explanation for each section of the project report:

Diabetes Prediction Using Support Vector Machine (SVM)

1. Introduction

Explanation:

This section introduces the purpose and significance of the project. Diabetes is a serious health condition, and early diagnosis can significantly impact treatment and management. This project focuses on predicting the likelihood of diabetes in patients based on their medical data using a machine learning algorithm known as Support Vector Machine (SVM).

2. Dataset

Explanation:

The dataset used for this project contains various features related to patient health, such as glucose levels, blood pressure, and body mass index (BMI). It also includes a label indicating whether the patient has diabetes or not. This section covers how the dataset is loaded and provides an overview of its structure and contents.

2.1 Loading the Dataset

python

Copy code

```
import pandas as pd


# Loading the diabetes dataset to a pandas DataFrame

diabetes_dataset = pd.read_csv('diabetes.csv')
```

Explanation: This code snippet demonstrates how to load the dataset from a CSV file into a pandas DataFrame, a common data structure used in Python for data manipulation.

2.2 Data Overview

python

Copy code

```
# Printing the first 5 rows of the dataset

print(diabetes_dataset.head())


# Number of rows and columns in the dataset

print(diabetes_dataset.shape)
```

# Getting the statistical measures of the data

print(diabetes_dataset.describe())

# Value counts of the Outcome column

print(diabetes_dataset['Outcome'].value_counts())

# Mean values for each group in Outcome column

print(diabetes_dataset.groupby('Outcome').mean())

Explanation:

- **First 5 rows**: Provides a glimpse of the data, helping to understand its structure and contents.
- **Shape**: Shows the number of rows and columns, giving an idea of the dataset's size.
- **Statistical measures**: Offers summary statistics (mean, standard deviation, etc.) for each feature, helping to understand the data distribution.
- **Value counts**: Displays the distribution of the target variable (Outcome), indicating how many instances of each class are present.
- **Group means**: Shows the average values of features grouped by the outcome, providing insights into how features differ between classes.

3. Data Preprocessing

Explanation:

Data preprocessing is essential for preparing the data for machine learning models. This section includes separating features from labels and standardizing the data to ensure that all features contribute equally to the model.

3.1 Separating Features and Labels

python

Copy code

# Separating the data and labels

```
X = diabetes_dataset.drop(columns='Outcome', axis=1)
Y = diabetes_dataset['Outcome']
```

Explanation: Here, X contains the features (input variables), and Y contains the labels (the target variable). The Outcome column is dropped from X and kept as Y.

3.2 Standardizing the Data

python

Copy code

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X)
standardized_data = scaler.transform(X)

X = standardized_data
Y = diabetes_dataset['Outcome']
```

Explanation: Standardization is applied to normalize the feature values so that they have a mean of 0 and a standard deviation of 1. This ensures that features with different scales do not disproportionately affect the model.

4. Model Training and Evaluation

Explanation:

This section covers the process of splitting the data into training and testing sets, training the SVM classifier, and evaluating its performance.

4.1 Splitting the Data

python

Copy code

```
from sklearn.model_selection import train_test_split
```

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)

Explanation: The dataset is divided into training and testing sets. The test_size=0.2 parameter means 20% of the data is used for testing. stratify=Y ensures that the split maintains the proportion of classes in both training and testing sets.

4.2 Training the Model

python

Copy code

```
from sklearn import svm

from sklearn.metrics import accuracy_score


# Creating and training the Support Vector Machine Classifier

classifier = svm.SVC(kernel='linear')

classifier.fit(X_train, Y_train)
```

Explanation: An SVM classifier with a linear kernel is created and trained on the training data. SVM is a powerful classification algorithm that finds the optimal hyperplane to separate classes.

4.3 Evaluating the Model

python

Copy code

```
# Accuracy score on the training data

X_train_prediction = classifier.predict(X_train)

training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print('Accuracy score of the training data : ', training_data_accuracy)


# Accuracy score on the test data

X_test_prediction = classifier.predict(X_test)
```

test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

print('Accuracy score of the test data : ', test_data_accuracy)

Explanation: The accuracy of the model is evaluated on both training and test datasets. Accuracy measures how well the model's predictions match the true labels. This helps assess the model's performance and generalizability.

5. Making Predictions

Explanation:

This section demonstrates how to use the trained model to make predictions on new, unseen data.

5.1 Input Data

python

Copy code

input_data = (5, 166, 72, 19, 175, 25.8, 0.587, 51)

Explanation: An example of a new data instance is provided. This data represents the features of a patient for whom we want to predict diabetes.

5.2 Preparing the Input Data

python

Copy code

```
import numpy as np


# Changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)


# Reshaping the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1, -1)


# Standardizing the input data
```

```
std_data = scaler.transform(input_data_reshaped)
```

Explanation: The input data is converted to a numpy array, reshaped to match the input format expected by the model, and standardized using the same scaler applied to the training data.

5.3 Making a Prediction

Explanation: The standardized input data is used to make a prediction with the trained SVM model. The prediction result indicates whether the person is predicted to have diabetes or not.

6. Conclusion

The SVM classifier model has been successfully trained and evaluated on the diabetes dataset. The model provides an accuracy score for both training and test datasets, indicating its effectiveness in predicting diabetes. Additionally, we demonstrated how to make predictions for new data instances using the trained model.