International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 3, May-Jun-2025, ISSN (Online): 2395-566X

Diabetes Prediction using SVM Machine Learning Algorithm

Deepak Tomar[1], Kismat Chhillar[2]
[1]System Analyst, Computer Center Bundelkhand University Jhansi, India.
[2]Assistant Professor, Dept. of Maths & Computer Applications Bundelkhand University Jhansi, India

Abstract- Diabetes mellitus has emerged as a major global health concern, necessitating early detection and effective predictive mechanisms to support timely medical intervention. Machine learning techniques have increasingly been employed in healthcare analytics to improve diagnostic accuracy and assist clinicians in decision making. Among these techniques, the Support Vector Machine (SVM) algorithm has demonstrated strong performance in classification problems involving medical datasets. This study explores the application of SVM for predicting the likelihood of diabetes using patient health indicators such as body mass index, blood glucose level, age, and family medical history. By analyzing patterns within clinical data, the model classifies individuals into diabetic and non-diabetic categories. The predictive capability of SVM allows the identification of individuals who may be at risk of developing diabetes, thereby enabling preventive healthcare measures. Empirical findings from related studies indicate that SVM-based models can achieve high predictive accuracy, making them a reliable approach for diabetes prediction and early risk assessment in medical decision support systems.

Keywords – Diabetes Prediction, Support Vector Machine (SVM), Machine Learning, Medical Data Analysis, Healthcare Analytics, Disease Risk Assessment.

I. INTRODUCTION

Traditional Diabetes mellitus is a chronic metabolic disorder that has become one of the most significant public health challenges worldwide. It is characterized by elevated blood glucose levels resulting from defects in insulin secretion, insulin action, or both. The growing prevalence of diabetes has been associated with lifestyle changes, aging populations, and increasing rates of obesity. According to global health reports, millions of individuals remain undiagnosed, which increases the risk of severe complications such as cardiovascular disease, kidney failure, nerve damage, and vision impairment. Early identification of individuals at risk of diabetes is therefore essential for effective prevention and timely medical intervention.

In the healthcare realm, early detection and prevention of disease plays pivotal roles in managing the chronic conditions and improving patient outcomes. Diabetes which is a widespread disorder, if left untreated poses significant health risks. Taking this into consideration, the integration of machine learning (ML) techniques holds promise for revolutionizing practices of healthcare, particularly in predictive analytics. In this paper, diabetes prediction system is implemented using Support Vector Machine (SVM). SVM is a powerful ML algorithm for identifying persons at risk of diabetes based on some health metrics. Through practical steps, we explored SVM methodology, from data preprocessing to model training and model evaluation

Traditional diagnostic methods rely on laboratory tests and clinical evaluation, which may detect the disease only after it has progressed to a certain stage. With the rapid advancement of data-driven technologies, machine learning techniques have emerged as valuable tools in medical diagnosis and disease prediction. These techniques are capable of analyzing large volumes of healthcare data and identifying complex patterns that may not be easily recognized through conventional statistical approaches. By leveraging patient health indicators such as age, body mass index, glucose level, and hereditary factors, machine learning models can assist healthcare professionals in predicting the likelihood of diabetes at an early stage.

Among various machine learning algorithms, Support Vector Machine (SVM) has gained considerable attention for classification tasks in healthcare applications. SVM is particularly effective in handling high dimensional data and identifying optimal decision boundaries that separate different classes. In the context of diabetes prediction, SVM can analyse patient datasets and classify individuals into diabetic or non-diabetic categories based on underlying patterns in medical attributes. Its ability to achieve high accuracy and robustness makes it a suitable technique for predictive modeling in medical diagnostics.

The integration of SVM based predictive models into healthcare systems can contribute to improved screening processes and proactive disease management. By identifying individuals who are at a higher risk of developing diabetes, healthcare providers can implement preventive strategies such as lifestyle modification, regular monitoring, and early treatment. Consequently, the use of machine learning approaches like SVM not only enhances diagnostic efficiency but also supports the development of intelligent decision support systems that can improve overall healthcare outcomes.

The rest of the paper is organized as follows. Section 2 covers related research and literature review. Section 3 elaborates about SVM and the steps followed for research.

Section 4 is about practical implementation of SVM for diabetes prediction. Section 5 discusses about Results and discussion. Finally, the paper concludes with section 6. In section 6, conclusion and future work is discussed.

II. LITERATURE REVIEW

(Asfaw, 2019) examined the timely prediction of diabetes with regard to Several risk factors related to this disease using machine learning techniques. For effective prediction of Diabetes mellitus an investigation was conducted using six popular ML algorithms, namely naive bayes (NB), Random Forest (RF), support vector machine (SVM), logistics regression (LR), K-Nearest Neighbor (KNN) and C4.5 decision -making tree, on data of adult populations to predict diabetes. A technique that achieved the highest performance in terms of accuracy was considered the best choice. It was observed that SVM has achieved a better accuracy of 96.4 % to predict diabetes mellitus. (Mujumdar & Vaidehi, 2019) applied various machine learning algorithms to the data file and the classification was done. Logistics Regression amongst various algorithms gave the highest accuracy of 96%. Adaboost classifier through application of pipeline gave the best model with an accuracy of 98.8%. Machine learning algorithm accuracy was compared with two different data sets. Obviously, the model improved the accuracy and the prediction of diabetes.

(Viloriaa, Herazo-Beltran, Cabrera, & Pineda, 2020) obtained an effective Diagnostic dYG Classifier based on the patient's age, BMI and CG. DM diagnosis by a doctor is complicated, because multiple factors are involved in a disease, and the diagnosis is also subject to human error. Normal blood test lack enough information for correct diagnosis of the disease. A support vector machine (SVM) was implemented for prediction of the diagnosis of DM. An SVM was obtained with 99.2% accuracy with Colombian patients and an accuracy of 65.6% was obtained with a data set of patients having different ethnic background. (Reza, Hafsha, Amin, Yasmin, & Ruhi, 2023) an improved non-linear kernel was introduced in predicting Type 2 diabetes using the PIMA dataset. The evaluation revealed significant improvements in prediction accuracy as compared to existing kernel functions. The proposed approach also demonstrated significant improvement in various evaluation matrices which emphasized the method's effectiveness. While this study demonstrates promising results.

(Larabi-Marie-Sainte, Aburahmah, Almohaini, & Saba, 2019) tried different classifier types and build new models for Increasing the diabetes prediction accuracy. All Deep Learning (DL) and machine learning classifiers (ML) that were Used in the last six years has been reviewed for their accuracy and frequency of use. The accuracy of the ML techniques was 68% - 74%. For DL Algorithms, the highest accuracy achieved by scientists were 95%. (Sonar & JayaMalini, 2019) Different algorithms were used for prediction namely SVM, Decision tree, Naive Bayes and ANN. SVM algorithm works well even with unstructured and semi structured data like trees, text and images. The drawback of the SVM algorithm is that several key parameters are needed to be set correctly. All the four algorithms were evaluated for prediction. Decision Tree models gave precision of 85%, 77.3% for Support Vector Machine and for Naive Bayes 77%. Outcomes showed a significant accuracy of the methods. (Alanazi & A. Mezher, 2020) proposed a model which combined two machine learning algorithms namely Support Vector Machine and Random Forest for diabetes prediction. For this a real dataset was collected from Security Force Primary Health Care. The proposed model achieved an accuracy of 98% and ROC 99%. The results showed that Random Forest algorithm gave better accuracy score as compared to Support Vector Machine.

(Thaiyalnayaki, 2021) used 9 attributes for analysis. MLP deep learning classifier gave a classification accuracy of 77 %. The comparison of MLP deep learning classifier and SVM classifier was performed where the SVM classifier Correctly Classified 500 Instances with a classification accuracy of 65 % and Incorrectly Classified Instances with 34.8958 %. It was concluded that Deep learning perceptron classifier performs well with diabetes dataset and can be used for further automatic identification and detection analysis. (Soni & Varma, 2020) Machine Learning Classification and ensemble techniques were used on a dataset to predict diabetes. Machine learning techniques that were used are namely K-Nearest Neighbor (KNN), Decision Tree (DT), Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF) and Gradient Boosting (GB). The accuracy was different for every model. The results showed that Random Forest achieved highest accuracy as compared to other machine learning techniques.

(Joshi & Chawan, 2018) aimed to develop a system which could perform an early prediction of diabetes with a higher accuracy by combining the results of various machine learning techniques. The supervised machine learning methods that were used are namely SVM and Logistic regression. Also proposed an effective technique for earlier detection of the diabetes disease. (Arora, Singh, Nayef Al-Dabagh, & Maitra, 2022) proposed a novel architecture for predicting diabetes patients using support vector machine (SVM) and K-means clustering technique. The features extracted from K-means were then classified using an SVM classifier. Pima Indians Diabetes dataset was used. Accuracy of 98.7% was achieved. On this dataset, the combined method performed better as compared to the conventional SVM-based classification. Also compared the accuracy, precision, recall, and F1- score of different machine learning techniques used. (Mohan & Jain, 2020) Support Vector Machine [SVM] was applied for diabetes prediction. The performance of SVM algorithm was analyzed for different available kernels. The best kernel was selected and used for diabetes prediction. The work was implemented in python programming language and it was concluded that SVM performance was as good as other ML algorithms.

(Pethunachiyar, 2020) SVM with different kernel functions was applied for diabetes prediction. SVM with linear kernel performed best with the highest accuracy value for the classification of diabetes. SVM with different kernel functions was applied for diabetes prediction. SVM with linear kernel performed best with the highest accuracy value for the classification of diabetes. The SVM with Radial Kernel Produced 99% accuracy, SVM with Linear Kernel function produced 100%, and SVM with Polynomial kernel produced 90% for the considered data set. (Raj, Sanjay, M, & Sampath, 2020) analysis was carried out using two data mining classification algorithms such as Support Vector Machine and Naive Bayes. The analysis aimed for diabetes prediction using health record and compared the accuracy of SVM and Naive Bayes to find a better algorithm for prediction of diabetes. It was concluded that the Support vector machine algorithm was found to be more efficient in comparison to Naive Bayes algorithm for predicting diabetics over sample datasets.

(Assegie & Nair, 2020) the performance of different machine learning models, namely Linear Support Vector Machine (LSVM), Gaussian Naïve Bayes (GNB), and Random Forest (RF) was evaluated. The analysis showed that, the linear support vector machine (LSVM) performed well on prediction of diabetes with an accuracy of 78.39%. The Gaussian Naïve Bayes performed with an accuracy of 74.15% and the Random Forest (RF) performed the least with an average accuracy of 72.72%. (Xue, Min, & Ma, 2020) used supervised machine-learning algorithms like Naive Bayes classifier, Support Vector Machine (SVM), and LightGBM for diabetes prediction. It was concluded that the performance of support vector machine was best as compared to other algorithms used.

III. PROPOSED METHODOLOGY

1. Importing Libraries

In this section, we import the necessary libraries required for our diabetes prediction project. We import NumPy and Pandas for data manipulation, StandardScaler for feature scaling, train_test_split for splitting our dataset into training and testing sets, SVM from scikit-learn for building our predictive model, and accuracy_score for evaluating the performance of our model. These libraries provide us with the tools and functionalities needed to preprocess our data, train our model, and assess its accuracy effectively. The steps of research methodology are shown in figure 1 ahead.

2. Load the Dataset

In this section, we load the diabetes dataset into a Pandas DataFrame named diabetes_dataset. This dataset contains information about various health metrics such as glucose levels, blood pressure, BMI, and other attributes, along with a target variable indicating diabetes diagnosis. By loading the dataset into a DataFrame, we can easily explore its structure, visualize the data, and prepare it for model training. Let's proceed with loading the dataset and begin our analysis

3. Data Preprocessing

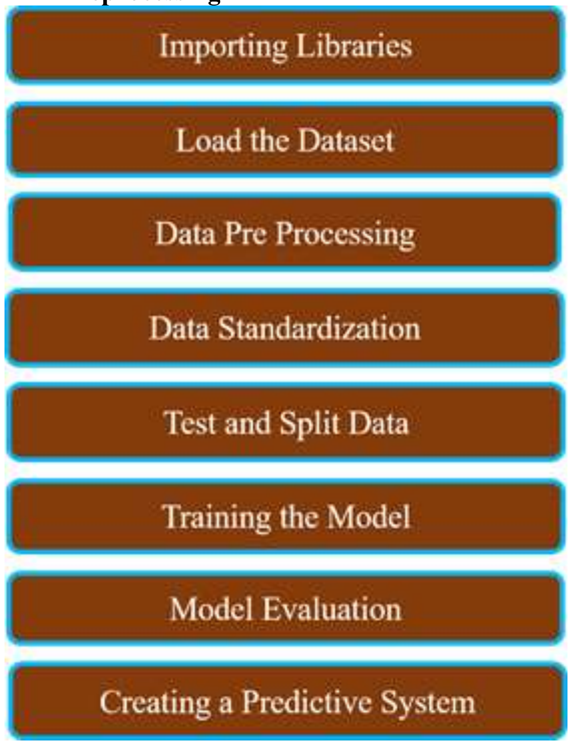


Fig. 1. Research Methodology Steps

Before diving into model training, it's essential to preprocess our dataset to ensure its suitability for analysis. Let's walk through the key preprocessing steps. We use the head() function to display the first 5 rows of the dataset, providing a glimpse into its structure and contents. The shape attribute provides the dimensions of the dataset, indicating the number of rows and columns. By invoking the describe() function, we obtain statistical measures such as mean, standard deviation, minimum, maximum, and quartiles for each feature in the dataset. This helps us understand the distribution and characteristics of the data. We use value_counts() to count the occurrences of each class in the target variable 'Outcome'. This reveals the balance between positive and negative cases of diabetes in the dataset. Here, we split the dataset into features (X) and labels (Y). Features comprise all columns except ‘Outcome’, while labels contain only the ‘Outcome’ column. This separation is essential for model training and evaluation. In our dataset, the label ‘0’ corresponds to individuals classified as non-diabetic, while the label ‘1’ represents those identified as diabetic.

Data Standardization
In this step, we standardize our feature data to ensure that all features have a mean of 0 and a standard deviation of 1. Here, we create an instance of the StandardScaler class from scikit-learn, which will be used to standardize our feature data. We use the fit() method to compute the mean and standard deviation for each feature in our dataset. This step calculates the parameters required for standardization. Next, we apply the transformation to our feature data using the transform() method. This process standardizes each feature by subtracting the mean and dividing by the standard deviation. We print the standardized feature data to observe the transformation applied to our dataset. Finally, we update our feature data (X) with the standardized version, ensuring that all features are now standardized. We assign the labels (Y) to the ‘Outcome’ column of the original dataset. Lastly, we print the standardized features (X) and labels (Y) to confirm that our data is ready for model training. By standardizing our feature data, we ensure that all features contribute equally to the model training process, leading to more reliable predictions. Let’s proceed with model training using our standardized dataset.

Train and Test Split
In this section, we split our standardized feature data
(X) and corresponding labels (Y) into training and testing sets to assess the performance of our predictive model. Let’s dissect the code. We use the train_test_split() function to divide our dataset into training and testing sets. The parameters include:
X: The feature data
Y: The corresponding labels
test_size: The proportion of the dataset to include in the testing set (here, 20%)
stratify: Ensures that the class distribution is preserved in both the training and testing sets, which is crucial for imbalanced datasets like ours (where 'Outcome' contains both diabetic and non-diabetic instances)
random_state: A seed for the random number generator, ensuring reproducibility of results.

We print the shapes of the original feature data (X), the training set (X_train), and the testing set (X_test) to verify that the dataset has been successfully split. By splitting our dataset into training and testing sets, we can train our model on the training data and evaluate its performance on unseen data. This helps us assess the generalization ability of our model and detect any potential overfitting issues.

Training the Model
In this section, we train our Support Vector Machine (SVM) classifier using the training data. Let’s break down the code. Here, we create an instance of the Support Vector Machine classifier from scikit-learn. We specify the linear kernel, indicating that we want to use a linear decision boundary for classification. We train the SVM classifier using the fit() method, which takes the training features (X_train) and corresponding labels (Y_train) as input. This step involves learning the parameters of the SVM model from the training data, such as the optimal hyperplane that separates the classes. By training the SVM classifier on our training dataset, we aim to learn patterns and relationships in the data that can help classify new instances accurately. Once trained, the model can be used to make predictions on unseen data.

Model Evaluation
We evaluate the performance of our trained Support Vector Machine (SVM) classifier using the accuracy score. Let’s delve into the code. We make predictions on the training dataset using the trained classifier (classifier.predict(X_train)) and calculate the accuracy score by comparing the predicted labels (X_train_prediction) with the actual labels (Y_train) using the accuracy_score() function. We print the accuracy score of the model on the training data to assess how well the classifier performs on data it has already seen.

Similarly, we make predictions on the testing dataset using the trained classifier (classifier.predict(X_test)) and calculate the accuracy score by comparing the predicted labels (X_test_prediction) with the actual labels (Y_test). Finally, after evaluating our trained Support Vector Machine (SVM) classifier, we obtained the following accuracy scores:

Accuracy score of the training data: 0.7866 Accuracy score of the test data: 0.7727
These scores indicate the percentage of correctly classified instances in the training and testing datasets, respectively. With an accuracy score of approximately 78.66% on the training data and 77.27% on the test data, our SVM classifier demonstrates reasonably good performance in predicting instances of diabetes.

Creating a Predictive System
To make predictions using our trained Support Vector Machine (SVM) classifier, we follow these steps:
- Define the input data

- Convert the input data to a NumPy array
- Reshape the array as we are predicting for one instance
- Standardize the input data
- Make predictions
- Print the prediction

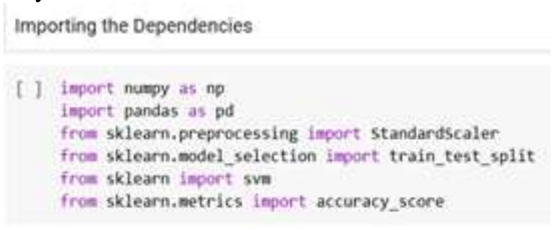
When we apply the input data and follow the above steps, we obtain the prediction that the person is diabetic. This predictive system enables us to classify individuals based on their health metrics and predict whether they are likely to have diabetes. Such systems can be invaluable in healthcare settings for early detection and intervention. The result of our predictive system shows that the person is diabetic, with a confidence score of 1.

IV. RESULTS AND DISCUSSION

Key points about using SVM for diabetes prediction:

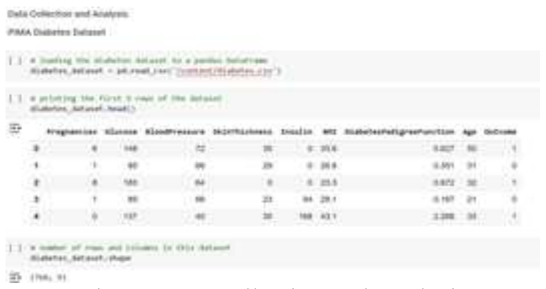
- Accuracy
- Data Input
- Classification
- Kernel Functions

SVMs can achieve high accuracy for diabetes prediction as compared to other machine learning algorithms that makes it a valuable tool for early detection. The SVM model took different medical parameters as input like blood pressure, glucose levels, BMI, number of pregnancies, family history of diabetes and age. The SVM classified individuals as either diabetic or non-diabetic on the basis of input data. SVMs utilize kernel functions like Radial Basis Function (RBF) for transforming data into a higher dimensional space which enables better separation between classes. The main benefits of using SVM for prediction of diabetes is early detection of diabetes, personalized risk assessment and data handling flexibility.


Importing the Dependencies

```
[ ] import numpy as np
    import pandas as pd
    from sklearn.preprocessing import StandardScaler
    from sklearn.model_selection import train_test_split
    from sklearn import svm
    from sklearn.metrics import accuracy_score
```

Fig. 2. Importing Libraries


Data Collection and Analysis

PIMA Diabetes Dataset

```
[ ] # loading the diabetes dataset to a pandas DataFrame
    diabetes_dataset = pd.read_csv('/content/diabetes.csv')

[ ] # printing the first 5 rows of the dataset
    diabetes_dataset.head()

   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72             35        0  33.6                     0.627   50        1
1            1       85             66             29        0  26.6                     0.351   31        0
2            8      183             64              0        0  23.3                     0.672   32        1
3            1       89             66             23       94  28.1                     0.167   21        0
4            0      137             40             35      168  43.1                     2.288   33        1

[ ] # number of rows and Columns in this dataset
    diabetes_dataset.shape

(768, 9)
```

Fig. 3. Data Collection and Analysis

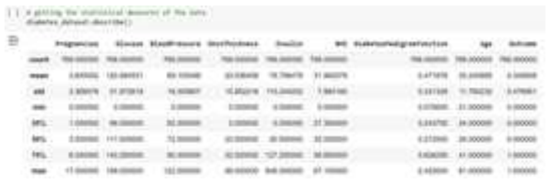

```
[ ] # getting the statistical measures of the data
    diabetes_dataset.describe()

       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin         BMI  DiabetesPedigreeFunction         Age     Outcome
count   768.000000  768.000000     768.000000     768.000000  768.000000  768.000000                768.000000  768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479   31.992578                  0.471876   33.240885    0.348958
std       3.369578   31.972618      19.355807      15.952218  115.244002    7.884160                  0.331329   11.760232    0.476951
min       0.000000    0.000000       0.000000       0.000000    0.000000    0.000000                  0.078000   21.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000   27.300000                  0.243750   24.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000   32.000000                  0.372500   29.000000    0.000000
75%       6.000000  140.250000      80.000000      32.000000  127.250000   36.600000                  0.626250   41.000000    1.000000
max      17.000000  199.000000     122.000000      99.000000  846.000000   67.100000                  2.420000   81.000000    1.000000
```

Fig. 4. Description of Data

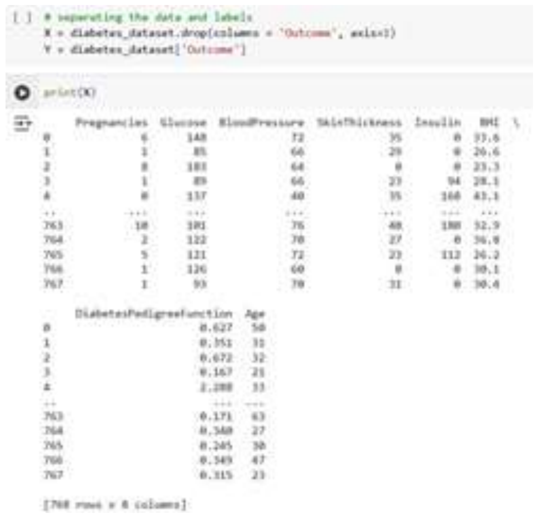

```
[ ] # separating the data and labels
    X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
    Y = diabetes_dataset['Outcome']

print(X)

     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

Fig. 5. Separating the Data and Labels

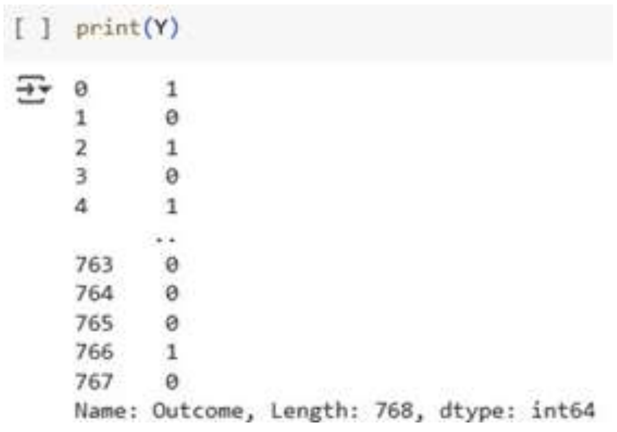
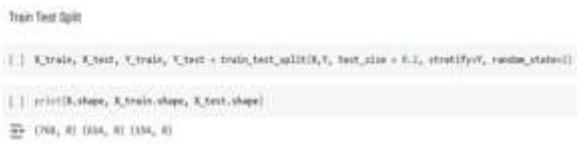
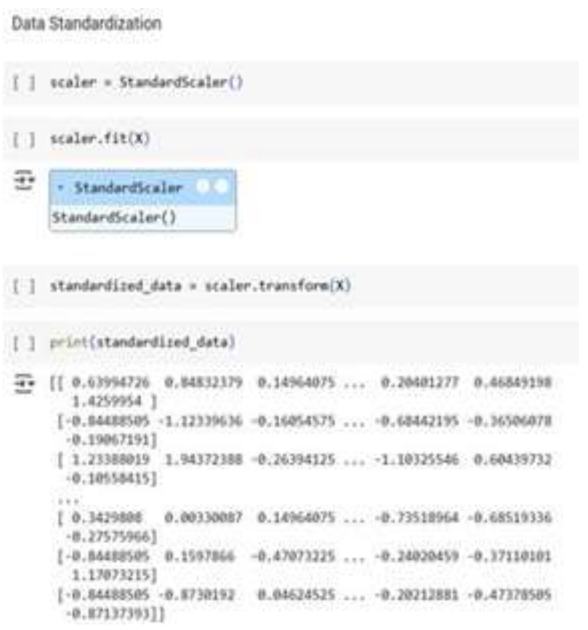

```
[ ] print(Y)

0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Fig. 6. Print the Outcomes


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)

[ ] print(X.shape, X_train.shape, X_test.shape)

(768, 8) (614, 8) (154, 8)
```

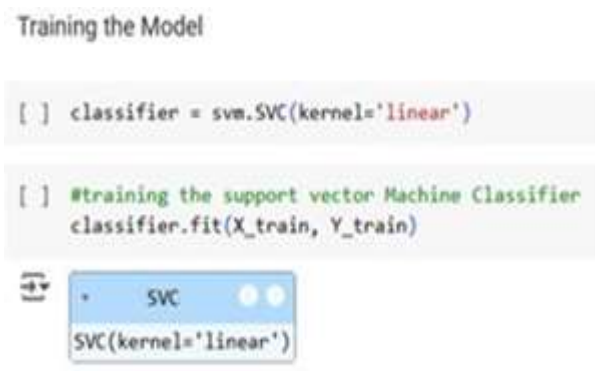
Fig. 7. Train Test Split



Data Standardization

```
[ ] scaler = StandardScaler()

[ ] scaler.fit(X)

    ▾ StandardScaler
    StandardScaler()

[ ] standardized_data = scaler.transform(X)

[ ] print(standardized_data)

    [[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
       1.4259954 ]
     [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
      -0.19067191]
     [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
      -0.10558415]
     ...
     [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
      -0.27575966]
     [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
       1.17073215]
     [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
      -0.87137393]]
```

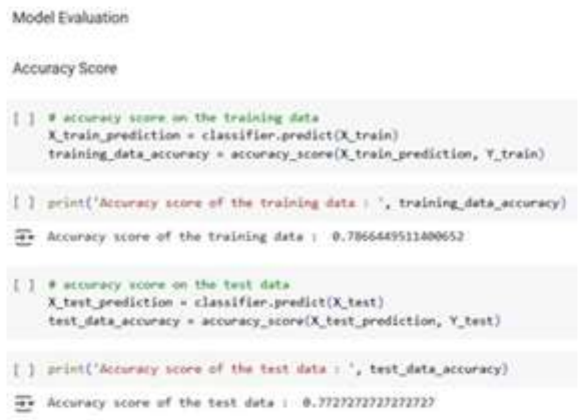
Fig. 8. Data Standardization



Training the Model

```
[ ] classifier = svm.SVC(kernel='linear')

[ ] #training the support vector Machine Classifier
    classifier.fit(X_train, Y_train)

    ▾ SVC
    SVC(kernel='linear')
```

Fig. 9. Training the Model

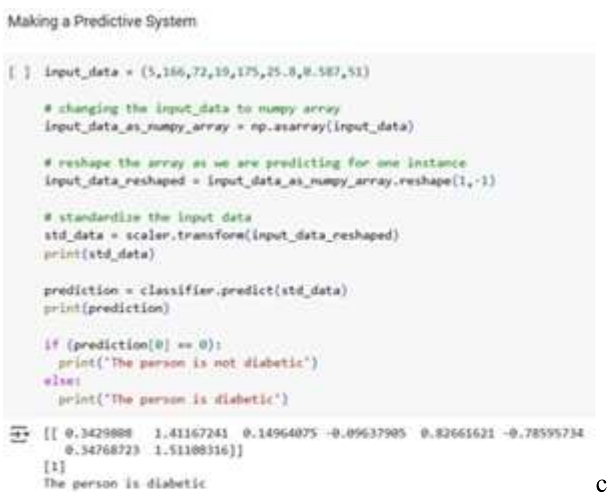


Model Evaluation

Accuracy Score

```
[ ] # 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 of the training data :  0.7866449511400652

[ ] # 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)

    Accuracy score of the test data :  0.7727272727272727
```

Fig. 10. Model Evaluation by Accuracy Score

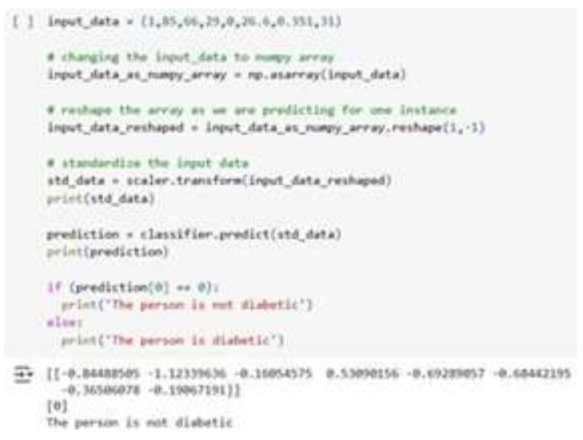


Making a Predictive System

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

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

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

    # standardize the input data
    std_data = scaler.transform(input_data_reshaped)
    print(std_data)

    prediction = classifier.predict(std_data)
    print(prediction)

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

    [[ 0.3429808   1.41167241  0.14964075 -0.09637905  0.82661621 -0.78595734
       0.34768723  1.51108316]]
    [1]
    The person is diabetic
```

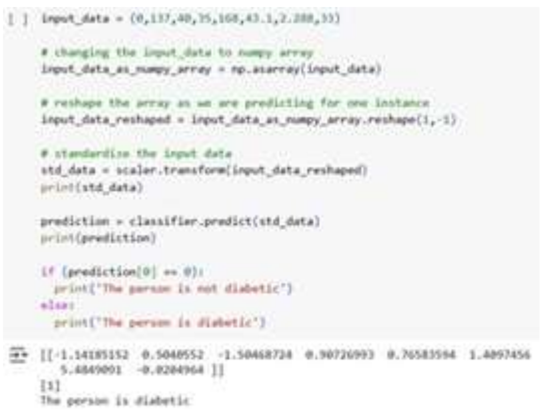
c

Fig. 11. Making a Predictive System



```
[ ] input_data = (1,85,66,29,0,26.6,0.351,31)

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

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

    # standardize the input data
    std_data = scaler.transform(input_data_reshaped)
    print(std_data)

    prediction = classifier.predict(std_data)
    print(prediction)

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

    [[-0.84488505 -1.12339636 -0.16054575  0.53090156 -0.69289057 -0.68442195
      -0.36506078 -0.19067191]]
    [0]
    The person is not diabetic
```

Fig. 12. Predicting if a person is Non-Diabetic



```
[ ] input_data = (0,137,40,35,168,43.1,2.288,33)

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

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

    # standardize the input data
    std_data = scaler.transform(input_data_reshaped)
    print(std_data)

    prediction = classifier.predict(std_data)
    print(prediction)

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

    [[-1.14185152  0.5040552  -1.50468724  0.90726993  0.76583594  1.4097456
       5.4849091  -0.0204964 ]]
    [1]
    The person is diabetic
```

Fig. 13. Predicting if a person is Diabetic

V. CONCLUSION

In this research, we developed a diabetes prediction system using the Support Vector Machine (SVM) algorithm. We began by importing the necessary libraries and loading the diabetes dataset. Through data preprocessing steps such as standardization and train-test split, we ensured our dataset was ready for model training. Next, we trained an SVM classifier and evaluated its performance using accuracy scores on both training and testing datasets. This allowed us to assess the model's ability to generalize to unseen data and make reliable predictions. With our trained model in hand, we implemented a predictive system capable of classifying individuals as diabetic or non-diabetic based on their health metrics. By providing the system's output and standardized feature values, we offered readers insights into interpreting the results and understanding the model's decision-making process. In conclusion, we highlighted the potential of machine learning in healthcare, particularly in early detection and intervention efforts for chronic conditions like diabetes. By leveraging data-driven approaches, we aim to contribute to improved patient outcomes and quality of life in the realm of healthcare.

VIII. FUTURE SCOPE

The application of Support Vector Machine (SVM) in diabetes prediction presents several opportunities for further research and technological advancement. One important future direction involves the integration of larger and more diverse healthcare datasets to improve the robustness and generalizability of predictive models. Incorporating data from electronic health records, wearable health monitoring devices, and genomic information can enhance the predictive capability of SVM models by capturing a wider range of patient health indicators. Such data driven approaches can help identify subtle risk factors and improve the early detection of diabetes across different demographic populations. Another promising area of future research is the development of hybrid and ensemble models that combine

SVM with other machine learning and deep learning techniques. Integrating SVM with algorithms such as Random Forest, Neural Networks, or Gradient Boosting can potentially enhance prediction accuracy and model stability. These hybrid approaches may help overcome the limitations of single model systems and enable more reliable predictions in complex medical datasets where multiple variables interact in nonlinear ways.

The deployment of SVM based diabetes prediction models in real world healthcare environments also offers significant future potential. These models can be incorporated into clinical decision support systems, hospital information systems, and mobile health applications to assist healthcare professionals and patients in monitoring diabetes risk. Real time predictive systems integrated with telemedicine platforms could provide continuous health assessment and early warning signals for individuals who may develop diabetes, thereby improving preventive care and reducing healthcare costs. Future work can also focus on improving the interpretability and transparency of SVM based prediction models. In healthcare applications, it is important that predictive systems provide understandable explanations for their decisions so that clinicians can trust and effectively use the results. Research on explainable artificial intelligence techniques can help reveal the key medical factors influencing diabetes predictions. Such advancements will enhance the acceptance of machine learning based diagnostic tools and support their wider adoption in medical practice.

REFERENCES

1. Alanazi, A. S., & A. Mezher, M. (2020). Using Machine Learning Algorithms For Prediction of Diabetes Mellitus. 2020 International Conference on Computing and Information Technology (ICCIT-1441). 2 ICCIT- 1441, pp. 55-57. Tabuk, Saudi Arabia: IEEE. doi:10.1109/ICCIT- 144147971.2020.9213708
2. Arora, N., Singh, A., Nayef Al-Dabagh, M. Z., & Maitra, S. K. (2022, August 24). A Novel Architecture for Diabetes Patients' Prediction Using. Mathematical Problems in Engineering, 4815521. doi:https://doi.org/10.1155/2022/4815521
3. Asfaw, T. A. (2019). PREDICTION OF DIABETES MELLITUS USING
MACHINE LEARNING TECHNIQUES. International Journal of Computer Engineering and Technology IJCET, 25-32.
4. Assegie, T. A., & Nair, P. S. (2020, January). The Performance Of Different Machine Learning Models on Diabetes Prediction. International journal of scientific & technology research (IJSTR), 9(1), 2491-2494.
5. Joshi, T. N., & Chawan, P. M. (2018). Logistic Regression and SVM Based Diabetes Prediction System. International Journal For Technological Research In Engineering, 5(11), 4347-4350.
6. Larabi-Marie-Sainte, S., Aburahmah, L., Almohaini, R., & Saba, T. (2019). Current Techniques for Diabetes Prediction: Review. Applied Sciences, 9(21), 4604. doi:https://doi.org/10.3390/app9214604
7. Mohan, N., & Jain, V. (2020). Performance Analysis of Support Vector Machine in Diabetes Prediction. Fourth International
8. Conference on Electronics, Communication and Aerospace Technology (ICECA-2020) (pp. 1-3). Coimbatore, India: IEEE. doi:10.1109/ICECA49313.2020.9297411
9. Mujumdar, A., & Vaidehi, D. (2019). Diabetes Prediction using Machine Learning Algorithms. Procedia Computer Science(165), 292-299. doi:10.1016/j.procs.2020.01.047
10. Pethunachiyar, G. (2020). Classification of diabetes patients using kernel based support vector machines. 2020 International Conference on Computer Communication

and Informatics (ICCCI -2020) (pp. 1-4). Coimbatore, India: IEEE. doi:10.1109/ICCCI48352.2020.9104185

11. Raj, R., Sanjay, D., M, D., & Sampath, D. (2020). Comparison of Support Vector Machine and Naïve Bayes Classifiers for Predicting Diabetes. 2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE) (pp. 41-45). Bangalore, India: IEEE. doi:https://doi.org/10.1109/ICATIECE45860.2019.9063792
12. Reza, M., Hafsha, U., Amin, R., Yasmin, R., & Ruhi, S. (2023). Improving SVM performance for type II diabetes prediction with an. Computer Methods and Programs in Biomedicine Update, 4, 100118. doi:https://doi.org/10.1016/j.cmpbup.2023.100118
13. Sonar, P., & JayaMalini, P. (2019). Diabetes prediction using different machine learning approaches. 3rd International Conference on Computing Methodologies and Communication (ICCMC) (pp. 367-371). Erode, India: IEEE. doi:https://doi.org/10.1109/ICCMC.2019.8819841
14. Soni, M., & Varma, D. (2020, September). Diabetes Prediction using Machine Learning Techniques. International Journal of Engineering Research & Technology (IJERT), 9(9), 921-925.
15. Thaiyalnayaki, K. (2021, January). Classification of Diabetes Using Deep Learning and SVM Techniques. International Journal of Current Research and Review (IJCRR), 13(1), 146-149.
16. Viloriaa, A., Herazo-Beltran, Y., Cabrera, D., & Pineda, O. B. (2020, April 6-9). Diabetes Diagnostic Prediction Using Vector Support Machines. Procedia Computer Science(170), 376-381. doi:10.1016/j.procs.2020.03.065
17. Xue, J., Min, F., & Ma, F. (2020). Research on Diabetes Prediction Method Based on Machine Learning. Journal of Physics: Conference Series. 1684, p. 012062. Shanghai, China: IOP Science. doi:10.1088/1742-6596/1684/1/012062