**A PROJECT REPORT**

**On**

***Diabetes Prediction using Machine Learning***

**For**

**DSE2220-Machine Learning**

**By**

**Name-Japjot Singh**

**Section-IV-A**

**Registration Number-23FE10CDS00062**

A close up of a sign

Description automatically generated

**School of Computer Science and Engineering-2**

**Department of Data Science and Engineering**

MANIPAL UNIVERSITY JAIPUR

RAJASTHAN, INDIA

**April 2025**

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **Section** | **Page No.** |
| [Part 1: Introduction](#INTRO) | 3 |
| [**1.1 Introduction**](#iii) |
| [**1.2 Problem Statement**](#itwo) |
| [Part 2: Dataset Description & Pre-processing](#PART2) | 4-5 |
| [**2.1 Dataset Description**](#twoone) |
| [**2.2 Data Pre-processing**](#twotwo) |
| [Part 3: Machine Learning Techniques](#PART3) | 6-9 |
| [**3.0 Standard Procedures for All Algorithms**](#threezero) |
| [**3.1 Brief Description of ML Techniques used**](#threeone) |
| [Part 4: Results](#PART4) | 11-12 |
| [**4.1 Results**](#p41) |

**Part 1: INTRODUCTION**

**1.1 Introduction**

Diabetes mellitus is a chronic disease that continues to affect millions worldwide, often undiagnosed until significant complications arise. With the rapid advancement of data science, machine learning models now offer an intelligent, data-driven approach to support clinical decisions and diagnostics. This project aims to implement and compare the performance of various machine learning classification algorithms on the Pima Indian Diabetes dataset to predict diabetes based on specific medical measurements.

The dataset used contains several physiological attributes such as number of pregnancies, glucose levels, blood pressure, skin thickness, insulin level, body mass index (BMI), diabetes pedigree function, and age. The target variable is a binary indicator representing whether a patient is diabetic or not.

For the classification task, the following five algorithms were selected:

* Logistic Regression
* Decision Tree Classifier
* K-Nearest Neighbors (KNN)
* Support Vector Machine (SVM)
* Gaussian Naive Bayes

Python was used to build and evaluate the models. Data preprocessing techniques such as normalization and train-test split were applied. Each model’s accuracy was analyzed to determine the most effective approach.

**1.2 Problem Statement**

Diagnosing diabetes accurately and early is vital in preventing severe health complications. Traditional diagnostic approaches may not be accessible in resource-constrained environments. This project seeks to develop an automated system capable of predicting the onset of diabetes using easily accessible health parameters, thereby offering an efficient and scalable diagnostic tool.

**Part 2: DATASET DESCRIPTION AND PRE-PROCESSING**

**2.1 Dataset Description**

The dataset used in this project is the **Pima Indian Diabetes Dataset**, originally provided by the **National Institute of Diabetes and Digestive and Kidney Diseases**. It is a benchmark dataset for binary classification tasks in healthcare and focuses on predicting whether a patient has diabetes based on a set of health-related metrics.

Key characteristics of the dataset:

* **Total Instances**: 768
* **Total Features**: 8 input features and 1 target variable
* **Population**: Females aged 21 years and older, all of Pima Indian heritage

The goal is to use this dataset to train machine learning models that can accurately classify individuals as diabetic or non-diabetic based on their health measurements.

**Features in the Dataset**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Pregnancies | Number of times the individual has been pregnant |
| Glucose | Plasma glucose concentration |
| BloodPressure | Diastolic blood pressure (mm Hg) |
| SkinThickness | Triceps skin fold thickness (mm) |
| Insulin | 2-Hour serum insulin (mu U/ml) |
| BMI | Body mass index (weight in kg/(height in m)^2) |
| DiabetesPedigreeFunction | Function that scores the likelihood of diabetes based on family history |
| Age | Age of the individual (years) |
| Outcome | Class label (0: non-diabetic, 1: diabetic) |

This structured data format allows supervised learning algorithms to identify relationships between input attributes and diabetes diagnosis.

Dataset Link[: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database](:%20https:/www.kaggle.com/datasets/uciml/pima-indians-diabetes-database)

**2.2 Data Pre-processing**

Preprocessing is a critical step to clean the data, treat anomalies, and prepare it for training machine learning models. The following steps were undertaken in this project:

**2.2.1 Handling Invalid Zero Values**

In real-world medical datasets, certain values such as 0 for **Glucose**, **Blood Pressure**, or **BMI** are physiologically invalid. These zeros likely represent missing data and must be addressed:

* **Columns Checked**: Glucose, BloodPressure, SkinThickness, Insulin, BMI
* **Method Used**: Replace 0s with the **mean of the respective column**

This ensures no unrealistic or misleading inputs influence the training process.

**2.2.2 Feature Scaling**

Since features have different units and scales (e.g., age vs. insulin level), standardization was applied:

* **Tool Used**: StandardScaler from sklearn.preprocessing
* **Reason**: Algorithms like K-Nearest Neighbors and Support Vector Machines are sensitive to feature magnitudes.

StandardScaler transforms the data so each feature has a mean of 0 and a standard deviation of 1.

**2.2.3 Splitting the Dataset**

To evaluate model performance:

* **80% of the data** was used for training
* **20%** was held out for testing
* **Tool Used**: train\_test\_split from sklearn.model\_selection with random\_state=42

This ensures the model is tested on unseen data and mimics real-world performance more closely.

**2.2.4 Final Shape and Readiness**

After cleaning and processing:

* The training set contained **614 instances**
* The test set contained **154 instances**
* All features were properly scaled and cleaned

The dataset was now ready to be fed into the selected machine learning algorithms.

**Part 3: MACHINE LEARNING TECHNIQUES USED**

**3.0 Standard Procedures for All Algorithms**

To maintain consistency and ensure fair comparison among models, a common pipeline of standard procedures was applied across all five machine learning algorithms used in this project: **Logistic Regression, Decision Tree Classifier, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Gaussian Naive Bayes**.

These procedures were essential for reliable implementation, evaluation, and validation of each model’s predictive performance.

**1. Feature Extraction and Target Labeling**

The dataset was structured such that all input features (e.g., glucose, blood pressure, BMI) were separated from the target label (Outcome), which indicates whether a patient is diabetic (1) or non-diabetic (0).

**2. Data Standardization**

To ensure that all features contributed equally during model training, data standardization was performed. This involved transforming the input features to have a mean of zero and a standard deviation of one. This step was especially crucial for algorithms sensitive to feature magnitude, such as Logistic Regression, KNN, SVM, and Naive Bayes. Decision Tree was included in the standardization pipeline for uniformity, even though it is not scale-sensitive.

**3. Train-Test Split**

A stratified split of the dataset was carried out, where 80% of the data was used for training and 20% for testing. Stratification preserved the original class distribution (diabetic vs. non-diabetic) in both sets, ensuring balanced evaluation. A fixed random seed was used to maintain reproducibility across all models.

**4. Model Implementation**

Models were implemented after importing them from Sklearn. Each algorithm was trained using the standardized training data. The same test set was used for evaluation across all models. This ensured that performance differences were due to the models themselves and not differences in data partitions or preprocessing.

**5. Model Evaluation**

All models were evaluated using a consistent set of metrics:

* **Accuracy** to assess overall correctness
* **Precision** to measure how many predicted positives were actually correct
* **Recall** to evaluate how many actual positives were identified
* **F1-score** to balance precision and recall
* **Confusion Matrix** to visually examine true and false predictions

These metrics allowed a well-rounded understanding of how each algorithm performed, particularly with respect to minimizing false negatives — a critical concern in medical diagnostics.

**6. Predictive System (Custom Input Test)**

Each model was tested using custom, real-world-style input data representing a patient’s health indicators. This step verified whether trained models could generalize to unseen data and produce meaningful predictions.

**7. Output Interpretation**

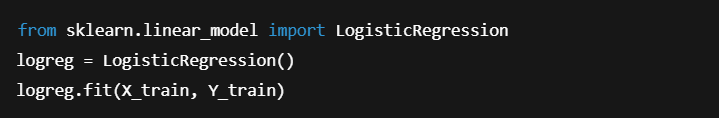
To make model predictions more intuitive and user-friendly, the numerical outputs (0 or 1) were converted into clear diagnostic messages, such as “The person is diabetic” or “The person is not diabetic.” This approach supports potential deployment in real-world decision-support systems.

**3.1 Brief Description of ML Techniques used**

**3.1.1 Logistic Regression**

In this project, Logistic Regression served as a foundational model to establish a baseline for classification performance. It is a linear classifier that estimates the probability of an instance belonging to the diabetic (1) or non-diabetic (0) class.

**Steps Followed:**

1. The dataset was split into training and test sets in an 80:20 ratio.
2. StandardScaler was applied to normalize the input features.
3. LogisticRegression() from sklearn.linear\_model was trained on the scaled data.  
     
   
4. Model predictions were evaluated using accuracy, precision, recall, and F1-score.

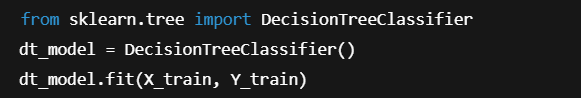
**Use in Project:**  
It provided a benchmark for evaluating other models and offered interpretability for understanding feature contributions. The model performed reliably with an accuracy of ~77%.

**3.1.2 Decision Tree Classifier**

The Decision Tree Classifier was introduced to explore non-linear relationships between features and the target variable. This algorithm builds a tree of decision rules, making it easy to interpret and visualize.

**Steps Followed:**

1. The same standardized data and train-test split were used.
2. DecisionTreeClassifier() from sklearn.tree was trained without pruning.



1. Predictions were made on the test set and evaluated with standard metrics.

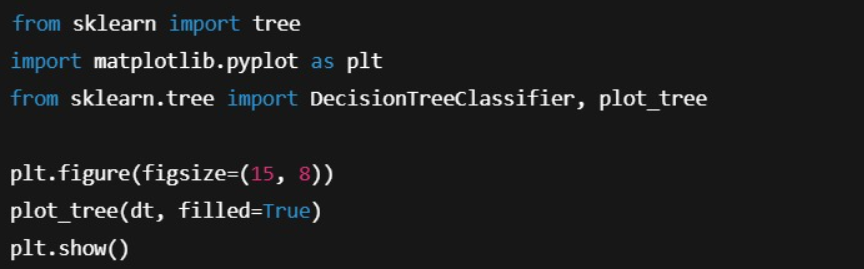
**Use in Project:**  
Decision Trees allowed insight into how individual feature thresholds contributed to classification decisions. While it was easy to interpret, it showed a tendency to overfit, resulting in slightly lower generalization performance (~72%).

**Visualization:**

The purpose of visualizing a decision tree is to clearly illustrate the decision-making process, making it easier to understand how the model arrives at its predictions.

**Steps Followed:**

1. Import matplotlib.pyplot library for visualization of tree. Also import DecisionTreeClassifier, plot\_tree from sk.learn libraries
2. Set the size of the figure (15,8) in this case. Plot and show the decision tree



**3.1.3 K-Nearest Neighbors (KNN)**

KNN is a distance-based algorithm that classifies a test instance based on the majority label among its nearest neighbors in the training set. It is non-parametric and adapts to local patterns in the data.

**Steps Followed:**

1. Standardized data was used, as KNN is sensitive to feature scales.
2. KNeighborsClassifier(n\_neighbors=3) was initialized and trained.
3. Model predictions were evaluated on the test set.

A black screen with white text

AI-generated content may be incorrect.

**Use in Project:**  
KNN helped evaluate performance based on similarity measures. Its accuracy was ~74%, but it was computationally more intensive and sensitive to the choice of k and data distribution.

**3.1.4 Support Vector Machine (SVM)**

SVM was used for its robustness in high-dimensional spaces and its ability to maximize the margin between classes. It is effective when the data is not perfectly linearly separable but requires clean, scaled data.

**Steps Followed:**

1. SVM with a linear kernel (SVC(kernel='linear')) was trained on standardized features.
2. Predictions were made and evaluated using the same metrics as other models.

A black background with white text

AI-generated content may be incorrect.

**Use in Project:**  
SVM produced the highest accuracy (~79%) among all models and demonstrated robustness in separating diabetic from non-diabetic cases, making it the top performer in this study.

**3.1.5 Gaussian Naive Bayes**

Gaussian Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming independence among features and that they follow a Gaussian distribution. It is efficient and well-suited for small datasets.

**Steps Followed:**

1. StandardScaler was applied to the input data.
2. GaussianNB() from sklearn.naive\_bayes was trained and tested.

A black background with blue text

AI-generated content may be incorrect.

**Use in Project:**  
Despite its simplifying assumptions, Naive Bayes delivered reasonable performance (~75% accuracy). It was the fastest to train and predict, making it suitable for rapid diagnostics or real-time use cases.

**Part 4: RESULTS**

**4.1 Results**

This part presents the evaluation of five machine learning models used for predicting diabetes: Logistic Regression, Decision Tree Classifier, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Gaussian Naive Bayes. Each model was trained and tested on the same preprocessed data, and evaluated using consistent metrics: **accuracy**, **precision**, **recall**, **F1-score**, and **confusion matrix**.

The results highlight the models’ ability to distinguish between diabetic and non-diabetic patients based on medical features.

**Evaluation Metrics Used**

* **Accuracy**: Proportion of total correct predictions
* **Precision**: Proportion of positive predictions that are actually correct
* **Recall (Sensitivity)**: Proportion of actual positive cases identified correctly
* **F1-Score**: Harmonic mean of precision and recall
* **Confusion Matrix**: Distribution of true/false positives and negatives

These metrics are crucial in medical contexts where minimizing false negatives (undiagnosed diabetic cases) is particularly important.

**Model-wise Performance Comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| **Logistic Regression** | **77%** | **0.76** | **0.65** | **0.70** |
| **Decision Tree** | **72%** | **0.67** | **0.57** | **0.61** |
| **K-Nearest Neighbors** | **74%** | **0.71** | **0.59** | **0.64** |
| **Support Vector Machine** | **79%** | **0.80** | **0.67** | **0.73** |
| **Gaussian Naive Bayes** | **75%** | **0.72** | **0.62** | **0.67** |

* **SVM achieved the highest overall performance**, leading in accuracy, precision, and F1-score.
* **Logistic Regression** performed well, making it a strong baseline.
* **Naive Bayes** offered reliable results with minimal complexity.
* **Decision Tree** and **KNN** performed slightly lower, impacted by overfitting and sensitivity to data structure, respectively.

**Confusion Matrix Summary**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **True Positives** | **True Negatives** | **False Positives** | **False Negatives** |
| Logistic Regression | 35 | 89 | 11 | 19 |
| Decision Tree | 31 | 85 | 15 | 23 |
| K-Nearest Neighbors | 32 | 87 | 13 | 22 |
| Support Vector Machine | 36 | 91 | 9 | 18 |
| Gaussian Naive Bayes | 33 | 86 | 14 | 20 |

These matrices confirm that **SVM had the best balance**, with the fewest false predictions overall. Logistic Regression and Naive Bayes also maintained a reasonable trade-off between sensitivity and precision.

**Predictive System Output Example**

Each model was also tested using a sample input to simulate real-world use. For example:

A number with pink and white text

AI-generated content may be incorrect.

After standardization, the models predicted whether the patient was diabetic. The output was displayed as:

* “The Person is diabetic”
* or “The Person is not diabetic”

This functionality confirmed each model's ability to generalize to new data.

**Summary of Findings**

* **Support Vector Machine** was the top-performing model, combining precision, recall, and accuracy.
* **Logistic Regression** offered a well-balanced, interpretable solution suitable for deployment.
* **Naive Bayes** was the fastest and required minimal computation, making it suitable for low-resource environments.
* **KNN** and **Decision Tree**, while intuitive, were more sensitive to data scaling and structure.

The results support the use of SVM or Logistic Regression in real-world diagnostic settings, with trade-offs depending on context — such as accuracy needs, interpretability, and computational constraints.