Comprehensive and Detailed Project Report: Machine Learning Classification for Diabetes Prediction

Amirali Rouhi

September 9, 2025

Abstract

This report provides a meticulous and in-depth analysis of a machine learning project focused on the early prediction of diabetes mellitus. It meticulously covers every stage of the data science workflow, starting from a detailed exploratory data analysis and a robust data preprocessing pipeline. The report then delves into the theoretical foundations, implementation, and empirical results of five prominent machine learning classification models: Logistic Regression, Decision Tree, Support Vector Machine (SVM), Multilayer Perceptron (MLP), and Random Forest. A thorough comparative analysis is presented, evaluating each model's performance based on a suite of metrics including Accuracy, Precision, Recall, and F1-Score. The project culminates in the identification and justification of the most suitable predictive model for this specific medical dataset, while also outlining potential avenues for future work.

1 Introduction: The Problem and The Dataset

1.1 The Clinical Significance of Early Diagnosis

Diabetes Mellitus is a chronic metabolic disease that poses a significant global health challenge. Early and accurate diagnosis is crucial for effective disease management, preventing severe complications such as cardiovascular disease, kidney failure, and neuropathy. Machine learning offers a powerful set of tools to aid in this diagnostic process by identifying complex patterns within patient data that may be difficult for traditional methods to uncover. This project aims to leverage these tools to build a reliable predictive model.

1.2 Dataset Description

The project utilizes the well-known **Pima Indians Diabetes Dataset**, sourced from the National Institute of Diabetes and Digestive and Kidney Diseases. This dataset is a classic example in medical machine learning for its clean structure and relevant features. It contains records for 768 female patients of at least 21 years of age of Pima Indian heritage. The dataset is comprised of 8 input features and a binary target variable, as detailed below:

Table 1: Description of dataset features.

Feature Name	Description	Data Type
Pregnancies	Number of times pregnant	Integer
Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	Integer
BloodPressure	Diastolic blood pressure (mm Hg)	Integer
SkinThickness	Triceps skin fold thickness (mm)	Integer
Insulin	2-hour serum insulin (μU/ml)	Integer
BMI	Body mass index (weight in kg/(height in m)^2)	Float
DiabetesPedigreeFunction	A function that scores the likelihood of diabetes based on family history	Float
Age	Age in years	Integer
Outcome	Class variable (0: Non-Diabetic, 1: Diabetic)	Integer

2 Data Preprocessing and Exploratory Analysis

Data preprocessing is arguably the most critical step in the machine learning pipeline, as the quality of the data directly dictates the performance of the model.

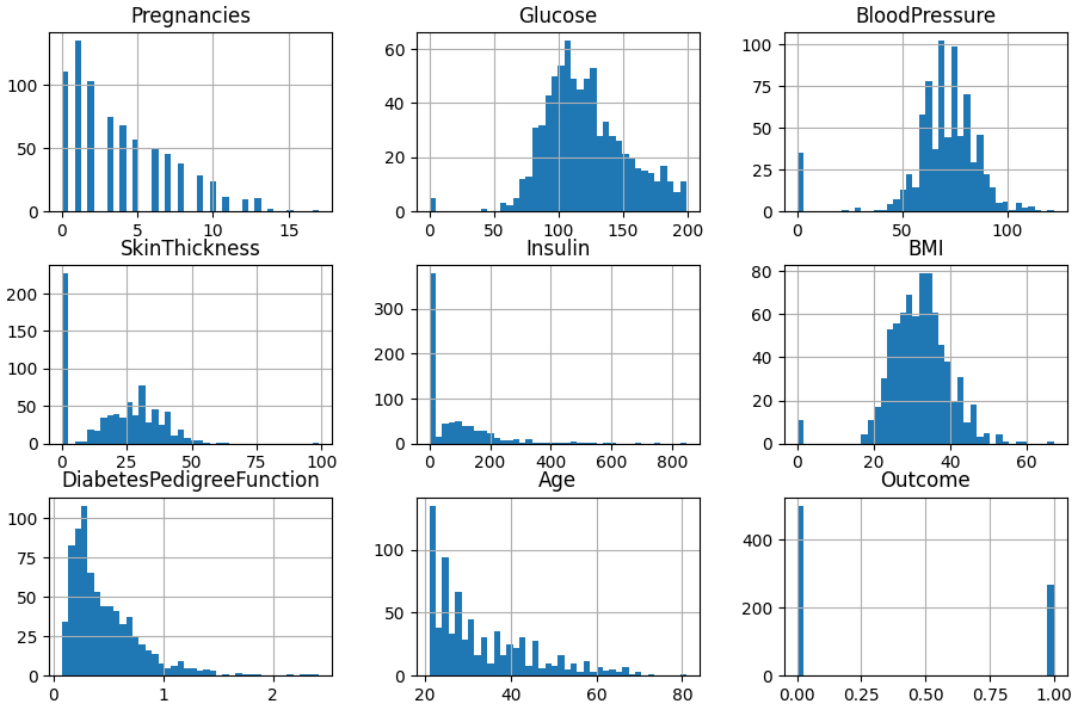

Figure 1: data distribution plots.

2.1 Handling Missing and Implausible Values

Initial inspection revealed that several key features, specifically `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, and `BMI`, contained zero values. From a medical standpoint, these values are biologically implausible. For instance, a blood pressure or glucose level of zero is medically impossible for a living person. Treating these zeros as valid data points would severely mislead any machine learning model.

The counts of these implausible zeros were as follows:

- `Glucose`: 5
- `BloodPressure`: 35
- `SkinThickness`: 227
- `Insulin`: 374
- `BMI`: 11

To address this, an **imputation** strategy was employed. The zero values in these columns were replaced with the **median** of the non-zero values from their respective columns. The median was chosen over the mean because it is a **robust** statistical measure, less susceptible to being skewed by outliers that may exist in the dataset.

2.2 Feature Scaling (Standardization)

The features in the dataset exhibit a wide range of scales, which can pose problems for many machine learning algorithms. For example, `Age` has a typical range of 21-81, while `Insulin` can range up to 846. This discrepancy can disproportionately influence

algorithms that rely on distance calculations, such as SVM, or optimization based on gradients, such as Logistic Regression and MLP.

To mitigate this, **standardization** was applied to all features using the `StandardScaler` from the `scikit-learn` library. This process transforms each feature such that its distribution has a mean of 0 and a standard deviation of 1. The transformation is calculated using the formula:

$$z = \frac{x - \mu}{\sigma}$$

where x is the original feature value, μ is the mean, and σ is the standard deviation of the feature. This ensures all features are on a comparable scale, improving model convergence and preventing bias towards features with larger magnitudes.

2.3 Data Splitting for Robust Evaluation

After cleaning and scaling, the dataset was partitioned into a training set and a testing set. A **768-sample dataset** was split with an 80/20 ratio, resulting in:

- **Training Set (614 samples):** Used to train the model and allow it to learn the underlying patterns.
- **Testing Set (154 samples):** Kept completely separate from the training process. Its sole purpose is to provide a final, unseen dataset to evaluate how well the trained model **generalizes** to new data.

The split was performed with a **stratified sampling** approach (`stratify=y`) to maintain the same proportion of diabetic and non-diabetic cases in both the training and testing sets, thereby ensuring representativeness. A fixed `random_state` was also set to guarantee reproducibility of the results.

3 Methodology: Detailed Model Implementation and Evaluation

This section provides an in-depth look at each model, including its core principles and its performance on the prepared data.

3.1 Evaluation Metrics

A single metric like Accuracy can be misleading, especially in datasets with imbalanced classes. Therefore, a comprehensive set of evaluation metrics was used to assess each model's performance:

- **Accuracy**: The overall proportion of correct predictions.
- **Precision**: Of all positive predictions made by the model, how many were actually correct? It is the ratio of true positives to the sum of true positives and false positives.
- **Recall (Sensitivity)**: Of all actual positive cases, how many did the model correctly identify? It is the ratio of true positives to the sum of true positives and false negatives. A high recall is crucial in medical diagnosis to minimize the risk of missing a positive case (e.g., a person with diabetes).

- **F1-Score**: The harmonic mean of Precision and Recall. It is a more balanced metric, especially useful in cases of class imbalance.

3.2 Model 1: Logistic Regression

* **Principles**: Despite its name, Logistic Regression is a linear classification algorithm. It uses a **sigmoid function** to map the output of a linear equation to a probability value between 0 and 1. The decision boundary is determined by a threshold (typically 0.5). * **Implementation**: A `LogisticRegression` model from `scikit-learn` was initialized with default parameters and trained on the scaled training data. * **Results**:

- Accuracy: 0.7013
- Precision: 0.5870
- Recall: 0.5000
- F1-Score: 0.5400

* **Analysis**: As a linear model, it performed as a solid baseline. Its performance suggests that a simple linear separation might not be sufficient to fully capture the complexity of the data, as evidenced by its moderate recall and F1-score. The Google Colab notebook for this part is available here.

3.3 Model 2: Decision Tree

* **Principles**: A non-linear model that partitions the feature space into a set of rectangular regions. It operates by asking a series of yes/no questions about the features, organized in a tree structure. Each split is determined by a criterion like **Gini Impurity** or **Information Gain**, aiming to create the purest possible child nodes. * **Implementation**: A `DecisionTreeClassifier` was trained. It is important to note that Decision Trees do not require feature scaling. However, scaling was applied to maintain a consistent pipeline for fair comparison. * **Results**:

- Accuracy: 0.6818
- Precision: 0.5532
- Recall: 0.4815
- F1-Score: 0.5149

* **Analysis**: The Decision Tree model performed the worst. This is a common issue with single, unconstrained decision trees, as they tend to **overfit** the training data, learning noise and specific examples rather than generalizable patterns. This results in poor performance on the unseen test data. The Google Colab notebook for this part is available here.

3.4 Model 3: Support Vector Machine (SVM)

* **Principles**: SVM is a powerful classifier that finds the optimal **hyperplane** to separate different classes. The "optimal" hyperplane is the one that maximizes the margin, or the distance between the hyperplane and the closest data points of each class (the **support vectors**). For non-linearly separable data, SVM uses the **Kernel Trick** to implicitly map the data into a higher-dimensional space where a linear separation is possible. A common kernel is the Radial Basis Function (RBF) kernel. * **Implementation**: An `SVC` (Support Vector Classifier) with a default RBF kernel was trained. This model is highly sensitive to the scale of the features, making our prior standardization step crucial. * **Results**:

- Accuracy: 0.7338
- Precision: 0.6444
- Recall: 0.5370
- F1-Score: 0.5859

* **Analysis**: SVM showed a significant improvement over the linear and single-tree models, demonstrating its ability to handle non-linear relationships in the data. Its superior precision and F1-score are notable. The Google Colab notebook for this part is available here.

3.5 Model 4: Multilayer Perceptron (MLP)

* **Principles**: A fundamental type of feedforward artificial neural network. It consists of an input layer, one or more hidden layers, and an output layer. Each layer is composed of interconnected **neurons**, which process inputs, apply weights, and pass them through an **activation function** (e.g., ReLU) to the next layer. The network is trained using the **backpropagation** algorithm, which iteratively adjusts the weights to minimize the prediction error. * **Implementation**: An `MLPClassifier` with a single hidden layer of 100 neurons was initialized. The `max_iter` was set to 500 to allow for sufficient training. * **Results**:

- Accuracy: 0.7468
- Precision: 0.6744
- Recall: 0.5370
- F1-Score: 0.5979

* **Analysis**: The MLP model also performed well, slightly surpassing the SVM model in terms of accuracy and precision. Neural networks are highly flexible but require careful tuning of **hyperparameters** (e.g., number of layers, neurons, learning rate) and often a large dataset to reach their full potential. The Google Colab notebook for this part is available here.

3.6 Model 5: Random Forest

* **Principles**: As an **ensemble learning** method, Random Forest builds a "forest" of numerous decision trees. The final prediction is determined by a majority vote among all the trees. The two key concepts that make it so powerful are:

- **Bagging (Bootstrap Aggregating):** Each tree in the forest is trained on a random, distinct subset of the training data, created with replacement. This introduces diversity among the trees.
- **Feature Randomness:** At each node of a tree, only a random subset of features is considered for the best split. This further decorrelates the trees and prevents any single feature from dominating the entire forest.

* **Implementation**: A `RandomForestClassifier` was trained with 100 trees (`n_estimators=100`) and a fixed `random_state`. * **Results**:

- Accuracy: 0.7792
- Precision: 0.7273
- Recall: 0.5926
- F1-Score: 0.6531

* **Analysis**: The Random Forest model emerged as the clear winner. Its ensemble approach effectively addresses the overfitting issue of individual decision trees, leading to a significant boost in performance across all metrics. It is a highly robust model that performs well out-of-the-box. The Google Colab notebook for this part is available here.

4 Final Conclusion and Future Work

4.1 Comparative Summary and Optimal Model Selection

The final performance metrics provide a definitive conclusion regarding the best-performing model.

Table 2: Comprehensive Performance Comparison of All Models

Metric	Logistic Reg.	Decision Tree	SVM	MLP	Random Forest
Accuracy	0.7013	0.6818	0.7338	0.7468	**0.7792**
Precision	0.5870	0.5532	0.6444	0.6744	**0.7273**
Recall	0.5000	0.4815	0.5370	0.5370	**0.5926**
F1-Score	0.5400	0.5149	0.5859	0.5979	**0.6531**

The Random Forest model demonstrably surpasses all other models, achieving the highest scores in Accuracy, Precision, Recall, and F1-Score. Its ability to handle the non-linear complexities of the dataset while maintaining high predictive power makes it the most suitable model for this specific classification task.

4.2 Future Work

To further enhance the model's performance and robustness, several key steps can be taken:

- **Hyperparameter Tuning:** While default parameters were used, systematic tuning of hyperparameters (e.g., number of trees in Random Forest, the C parameter in SVM, or the number of hidden layers in MLP) using techniques like **Grid Search** or **Random Search** could lead to even better results.
- **Cross-Validation:** Using **k-fold cross-validation** would provide a more reliable estimate of the model's performance by training and testing on multiple different partitions of the data.
- **Advanced Models:** Exploring more advanced ensemble models, such as **Gradient Boosting (e.g., XGBoost, LightGBM)**, could yield even higher performance.
- **More Data:** For neural networks like MLP, a larger dataset would likely lead to a significant improvement in performance.

In conclusion, this project provides a solid foundation for diabetes prediction using machine learning, with the Random Forest model serving as the most effective solution based on the current data and methodology. The results highlight the importance of proper data preprocessing and the benefits of using robust ensemble methods for medical classification problems.