Predict Diabetes

CMPU4011 CA1 BUILD A CLASSIFIER

C20305696 LOVELY FERNANDEZ

Introduction

The goal of this project is to develop a machine learning classifier that can predict whether a patient has diabetes based on a set of medical features. This classifier will be trained using a dataset taken from Kaggle that contains information on female patients and their diagnoses for diabetes. It is highly important to be able to predict diabetes based on specific health-relating factors as it can help to diagnose or estimate a likelihood of developing diabetes based on an individual's health record.

Data Exploration and Pre-processing

Descriptive Variables

- Pregnancies: Number of pregnancies the patient has had.
- Glucose: Plasma glucose concentration in the blood.
- BloodPressure: Blood pressure levels.
- SkinThickness: Thickness of the skin
- Insulin: Insulin levels in the blood.
- BMI: Body mass index.
- DiabetesPedigreeFunction: A function which assesses the likelihood of diabetes based on family history.
- Age: Age of the patient.

Features like glucose levels, BMI, and blood pressure are strongly correlated with the onset of diabetes, as they reflect key risk factors for the disease. (Due to the possible dependency in variables, I did not choose Naïve Bayes as a classifier.)

Target Variable

Outcome: A binary variable indicating whether the patient has diabetes (1 = has diabetes, 0 = does not have diabetes).

Data Cleaning and Pre-processing

Identified Missing Values: Glucose, BloodPressure, SkinThickness, Insulin, and BMI (0 values, which were replaced with NaN.)

It was assumed that '0' is a replacement for missing values, as it is biologically improbable to have a value of 0 for features like glucose, blood pressure, skin thickness, and other related features.

I chose to replace the NaN values with the median of each column rather than dropping rows as I wanted to avoid losing a large portion of the dataset - 50% of the rows would have been dropped if I removed all rows with missing values.

Classifier Development and Testing

I chose K-Nearest Neighbours (KNN) and ID3 decision trees as my classifiers to predict diabetes in a patient.

k-Nearest Neighbours (KNN)

KNN is a simple classifier to develop for problems like diabetes prediction where the decision boundary may not be linear. KNN does not make assumptions about the underlying distribution of the data, which is useful in diabetes prediction where the relationships between features (like blood sugar levels, BMI, age, etc…) might be complex and non-linear.

It works by finding the most similar examples to a given instance in the training set, using local patterns to make predictions. This approach is particularly effective for smaller datasets, like medical datasets, where the relationships between features may not be easily captured by more complex models.

Unlike more complex models, KNN doesn't require a separate training phase. This can be a benefit when you want a fast, simple model that can quickly adapt to new data.

ID3 Decision Tree

The ID3 algorithm is used to create decision trees by selecting the best feature to split on, using information gain as the criterion. This classifier was chosen because decision trees provide an easily interpretable model and the ID3 algorithm can capture non-linear relationships between features.

I implemented the ID3 by calculating:

- Entropy to measure uncertainty in the target variable.
- Information Gain to select the best feature for each split.

I also built the ID3 decision tree algorithm from scratch, which involved:

- Calculating *entropy* for the target variable and each feature.
- Using information gain to select the best feature to split the data at each node.
- Recursively building the tree until all instances are classified or no further splitting is possible.

Testing and Evaluation

The dataset was split into training and testing sets (70% for training, 30% for testing). The models were evaluated using the following metrics:

- Accuracy: The percentage of correct predictions on the test set.
- Classification Report: Includes precision, recall, and F1-score

KNN Evaluation

The KNN classifier was manually evaluated by comparing the predicted labels to the actual labels on the test set and I also used the Scikit-learn version to evaluate the model's performance on the test set.

ID3 Decision Tree Evaluation

The ID3 decision tree was evaluated using the accuracy score and classification report. The performance was compared to the Scikit-learn Decision Tree Classifier, which provided an easy-to-interpret decision tree structure and performance metrics.

Results

The custom KNN classifier achieved an accuracy of 67.97% on the test set and the Scikit-learn KNN classifier achieved the same accuracy of 67.97%**%**. This is good as the ideal accuracy is between 70-90% which indicates the model is neither overfitting nor underfitting.

The ID3 decision tree achieved an accuracy of 73.59% using Scikit-learn, while my custom implementation from scratch achieved an accuracy of 85.71% on the test set. This indicates that the custom model is performing significantly better.

Conclusion

Both the KNN and ID3 Decision Tree classifiers performed well in predicting diabetes. The ID3 decision tree, achieved a higher accuracy of 85.71%, demonstrating its ability to capture non-linear relationships effectively and providing a more accurate model for this dataset. However, the KNN classifier was more consistent, achieving an accuracy of 67.97% both in the custom and Scikit-learn implementations.

While KNN might not have outperformed ID3 in terms of accuracy, its consistent performance across different implementations suggests it is a reliable and stable choice. ID3 is preferred when higher accuracy and interpretability are desired, whereas KNN offers a simpler, more consistent approach that may be suitable for fast deployment.