Classification: Diabetes Data Analysis

Ethan Alexander

This report outlines the methodologies and decisions made during the completion of a classification project using diabetes data from the sklearn sample datasets. The analysis involved data exploration, preprocessing, and modeling, with the ultimate goal of identifying the most effective classification approach. All code and visualizations are included in the accompanying Jupyter Notebook.

1. **Data Loading and Exploration**

   To begin, I utilized the provided sample code snippet to load the diabetes data from the sklearn sample datasets. This ensured consistency with the project specifications.

   a. **Correlation Analysis**

      To identify relationships between the features and the target variable, I calculated a correlation matrix and visualized it as a heatmap in the Jupyter Notebook. This step highlighted the features most strongly correlated with the target variable, enabling the selection of the top four most informative features:

      bmi - Body mass index
      s5 - Possibly log of serum triglycerides level (ltg)
      bp - Average blood pressure
      s4 - Total cholesterol / HDL (tch)

   These features were prioritized in the subsequent analysis to ensure a focus on the most relevant predictors.

2. **Data Preprocessing**

   a. **Binarizing the Target Variable**

      After confirming the absence of missing values, I binarized the target vector based on a threshold of 140. Each data point was transformed into a binary value: 0 if the value was greater than or equal to 140 and 1 otherwise. This preprocessing step was necessary to convert the problem into a binary classification task.

b. **Data Splitting**

The dataset was split into training, testing, and validation sets with a 0.6/0.2/0.2 ratio using sklearn's `train_test_split` function. Since the function only supports splitting into two subsets, I applied it twice to achieve the desired three-way split. This approach ensured an appropriate distribution of data across all subsets while maintaining reproducibility.

c. **Transforming Continuous Features to Categorical**

To prepare the data for models that require categorical features, such as Decision Trees and K-Nearest Neighbors (KNN), I created a function to discretize each feature into five bins. This categorical transformation provided an alternative dataset that complemented the continuous version used by other models.

3. Classification Modeling

The classification process utilized four models to predict diabetes outcomes:

1. Gaussian Naive Bayes (GNB)
2. Support Vector Machine (SVM)
3. Decision Tree (DT)
4. K-Nearest Neighbor (KNN)

a. **Model Performance and Comparison**

Each model was trained on the prepared datasets, and performance metrics were calculated, including accuracy, precision, recall, and F1 scores. Confusion matrices were visualized as heatmaps in the accompanying Jupyter Notebook to facilitate model comparison.

b. **Support Vector Machine (SVM)**

For the SVM model, data standardization was performed to prevent bias towards features with wider ranges. Additionally, hyperparameter tuning for the parameters `C`, `gamma`, and `kernel` was conducted to optimize the model's performance. These adjustments contributed to the SVM's superior results compared to the other models.

c. **Decision Tree (DT) and K-Nearest Neighbor (KNN)**

While both the Decision Tree and KNN models offered interpretable results, they were prone to overfitting. The Decision Tree's high variance stems from its ability to create overly complex decision boundaries, fitting both data and noise. Similarly, the KNN model's sensitivity to noise was apparent with small values of k, which caused the model to overfit and misclassify points. Larger values of k mitigated this issue but often resulted in oversimplification, with the majority of data points being categorized into a single class.

d. **Gaussian Naive Bayes (GNB)**

The GNB model, while simple and computationally efficient, exhibited high bias due to its assumption of feature independence and normal distribution. This assumption limited its ability to fully capture the complexity of the dataset.

4. Conclusions

The SVM model emerged as the best classifier for this task, demonstrating superior generalization capabilities. Its ability to handle decision boundaries effectively, along with appropriate hyperparameter tuning, allowed it to outperform the other models. By contrast, the GNB model suffered from high bias, while the Decision Tree and KNN models were more prone to overfitting due to their high variance. Among these, the Decision Tree model was particularly susceptible to overfitting because it tends to create complex decision rules that fit noise in the training data.

Ultimately, the analysis highlights the importance of understanding the trade-offs between bias and variance when selecting and tuning classification models. The performance statistics and visualizations in the Jupyter Notebook provide a comprehensive comparison, supporting the conclusion that the SVM model offers the most robust and reliable predictions for this dataset.