Diabetes Classfier

Author: Predescu Sebastian-Ion
Class: *312CC*
Year: 2025, 1st year, 2nd Semester
Subject: *PCLP3*

---

1 Problem type

This problem is a classification problem: based on some medical information of some patients, we predict whether they have diabetes: *True* for diabetes, *False* otherwise.

2 Dataset Structure

The original database is obtained from Kaggle:

- Total rows: 768
- Train set represents 80% of the initial set of data.
- Test set represents 20% of the initial set of data.

3 CSV Export

Both CSVs are saved with `index=False`.

4 Features

Column	**Type**
Pregnancies	Integer
Glucose	Integer
Blood Pressure	Integer
Skin Thickness	Integer
Insulin	Integer
BMI	Float
Diabetes Pedigree Function	Float
Age	Integer
Outcome (target)	Boolean

5 Exploratory Data Analysis (EDA)

5.1 Missing Values Analysis

Tables 1 and 2 show the count and percentage of missing values for each column in the training and test datasets.

Column	Missing Count	Missing %
Pregnancies	0	0.0
Glucose	0	0.0
BloodPressure	0	0.0
SkinThickness	0	0.0
Insulin	0	0.0
BMI	0	0.0
DiabetesPedigreeFunction	0	0.0
Age	0	0.0
Verdict	0	0.0

Table 1: Missing values in the training set

Column	Missing Count	Missing %
Pregnancies	0	0.0
Glucose	0	0.0
BloodPressure	0	0.0
SkinThickness	0	0.0
Insulin	0	0.0
BMI	0	0.0
DiabetesPedigreeFunction	0	0.0
Age	0	0.0
Verdict	0	0.0

Table 2: Missing values in the test set

Interpretation: There are no missing values in either subset. Therefore, no imputation or deletion strategies are required.

5.2 Descriptive Statistics

Tables 3 show descriptive statistics for the training and test sets.

Column	Mean	Std	Min	25%	50%	75%	Max
Pregnancies	3.74	3.31	0	1	3	6	17
Glucose	120.85	32.03	0	100	117	139	199
BloodPressure	69.42	18.51	0	64	72	80	122
SkinThickness	20.40	15.43	0	0	23	32	63
Insulin	81.44	116.23	0	0	42.5	129.75	846
BMI	31.98	7.74	0	27.1	32	36.38	67.1
DiabetesPedigreeFunction	0.47	0.34	0.08	0.24	0.37	0.61	2.42
Age	32.91	11.50	21	24	29	40	81

Table 3: Descriptive statistics for the training set

Interpretation: The minimum values of 0 for some features like Glucose or BloodPressure may indicate missing or unrealistic measurements and should be handled during preprocessing.

5.3 Variable Distributions for Training Set Analysis

5.3.1 Age

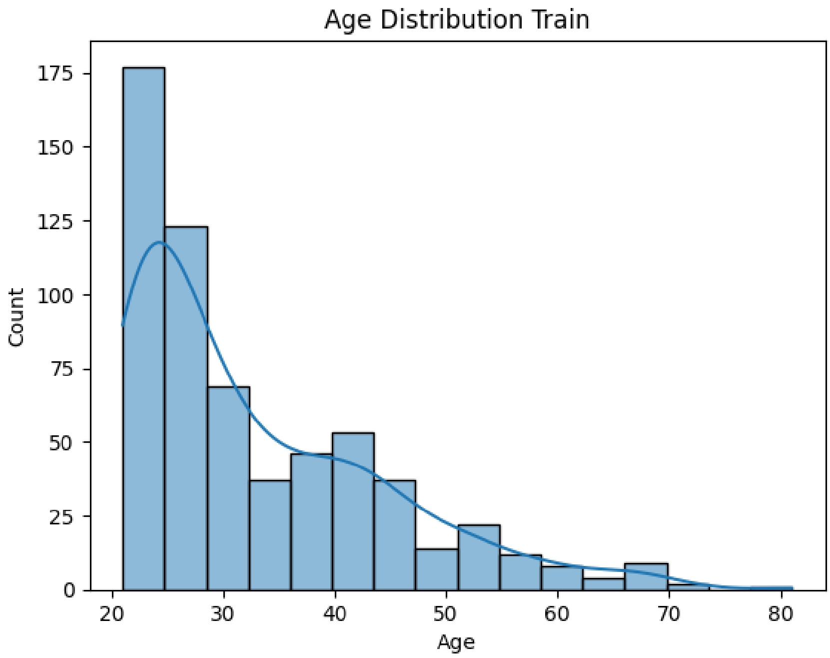

Figure 1: Age Histogram - Training Set

The age distribution is roughly normal, mostly between 20 and 60 years, with some outliers at older ages.

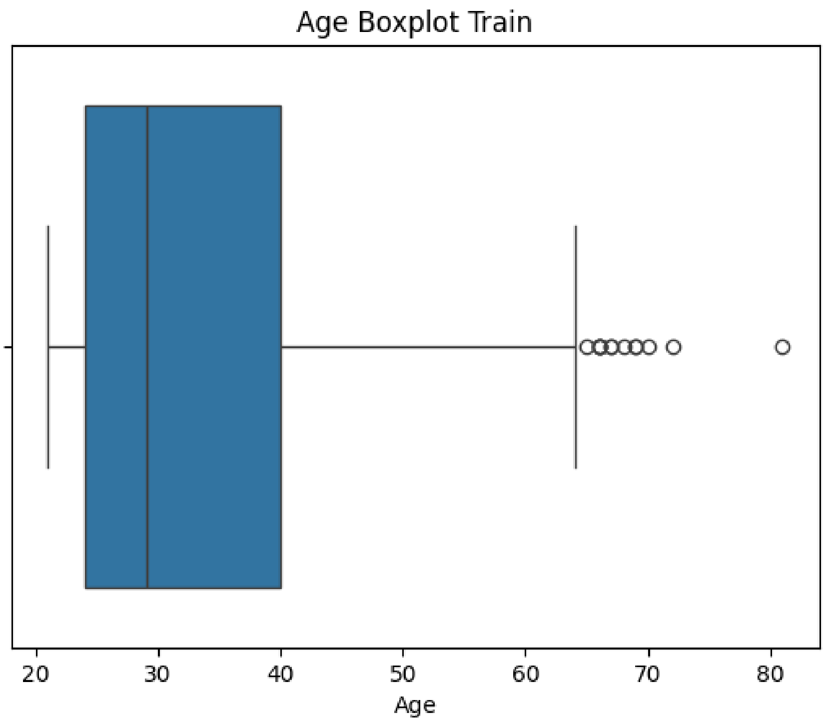

Figure 2: Age Boxplot - Training Set

The boxplot confirms a few outliers beyond 70 years old, but most data falls within expected ranges.

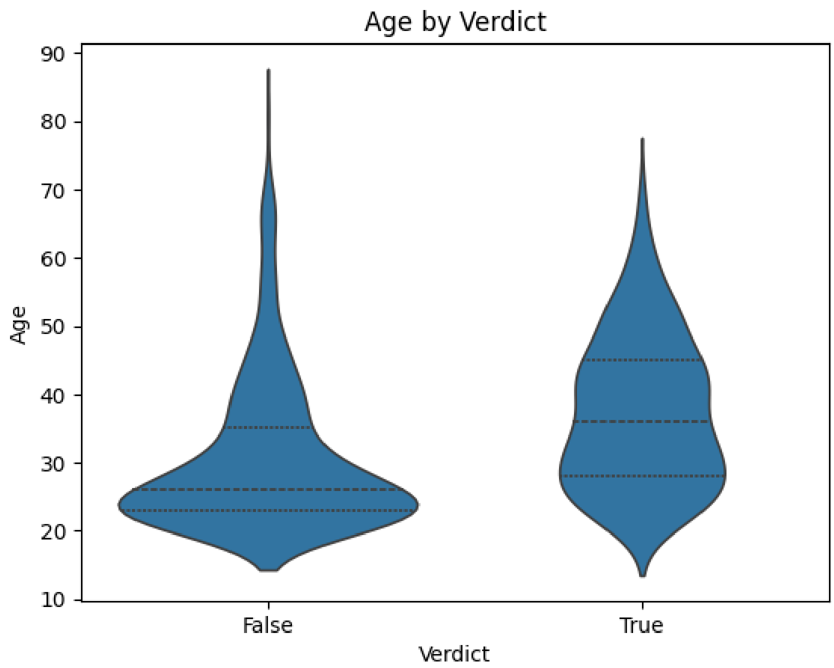

Figure 3: Age Distribution by Verdict - Training Set

Older patients tend to have a higher likelihood of diabetes, as shown by higher density for diabetic patients above age 40.

5.3.2 Glucose

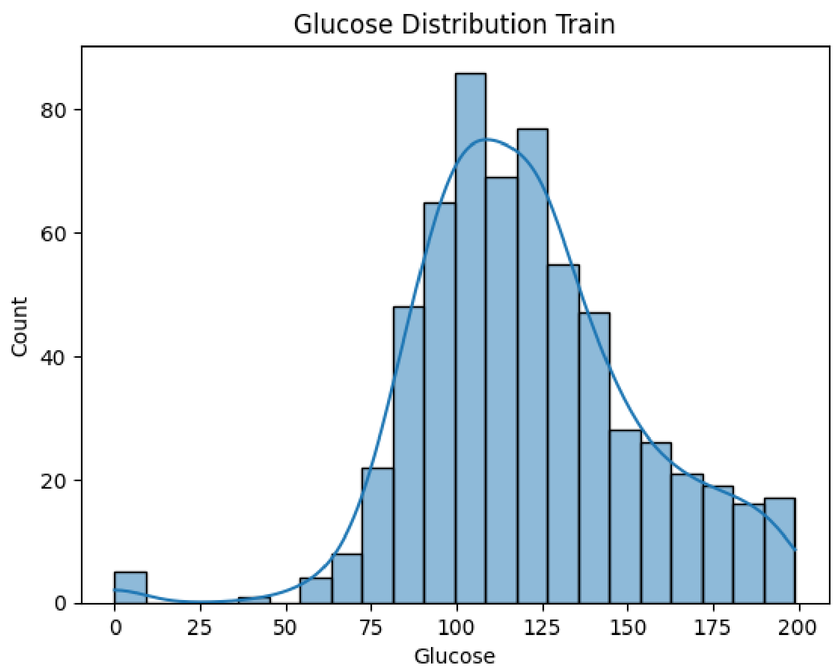

Figure 4: Glucose Histogram - Training Set

Glucose is right-skewed, with a long tail of high values corresponding to diabetic cases.

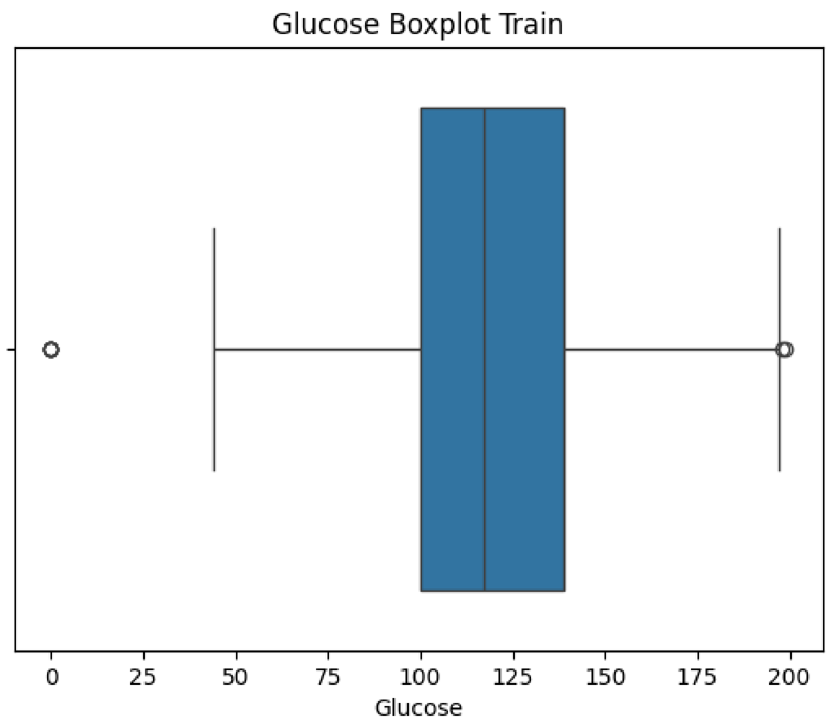

Figure 5: Glucose Boxplot - Training Set

There are significant high-value outliers which may need treatment.

Figure 6: Glucose Distribution by Verdict - Training Set

Diabetic patients generally have elevated glucose values, supporting its strong predictive role.

5.4 Variable Distributions for Test Set Analysis

5.4.1 Age

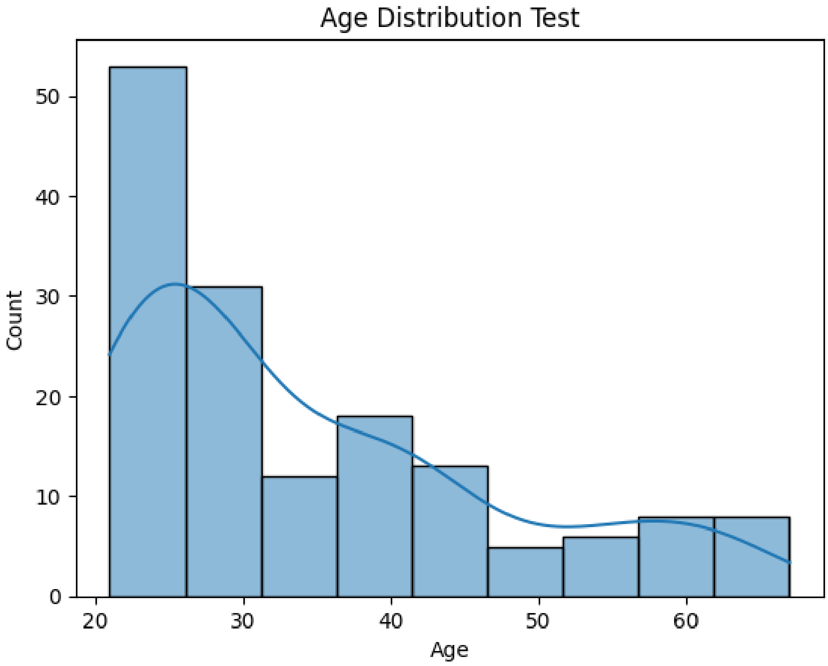

Figure 7: Age Histogram - Test Set

Test set age distribution matches training set, indicating consistent sampling.

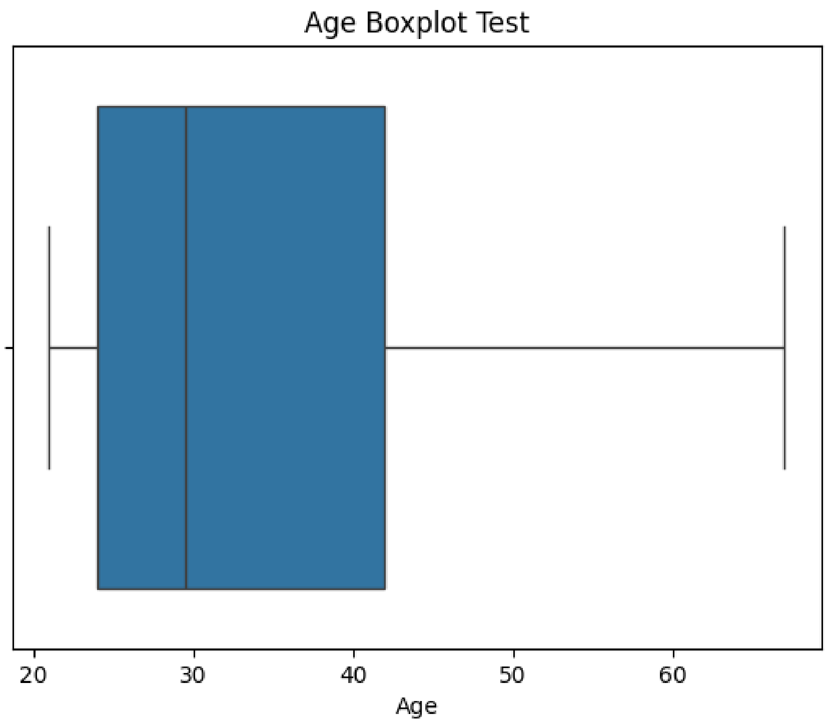

Figure 8: Age Boxplot - Test Set

Similar outliers at older ages are present in test data.

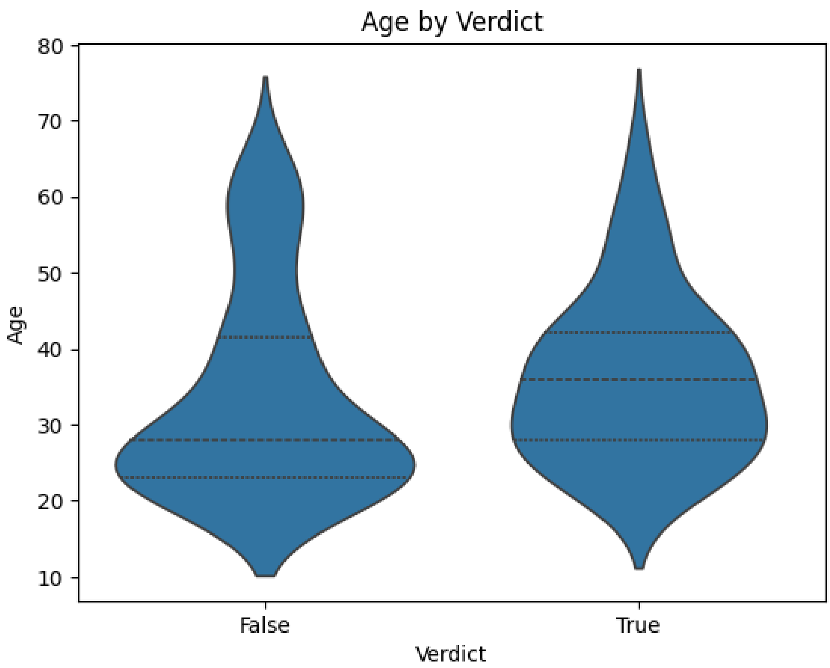

Figure 9: Age Distribution by Verdict - Test Set

Again, diabetic patients tend to be older.

5.4.2 Glucose

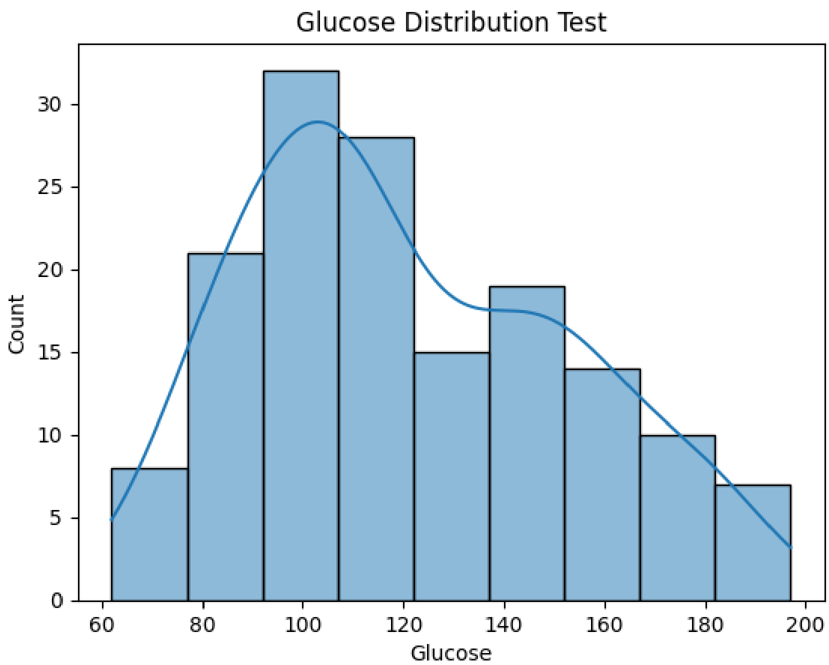

Figure 10: Glucose Histogram - Test Set

Distribution closely resembles training set.

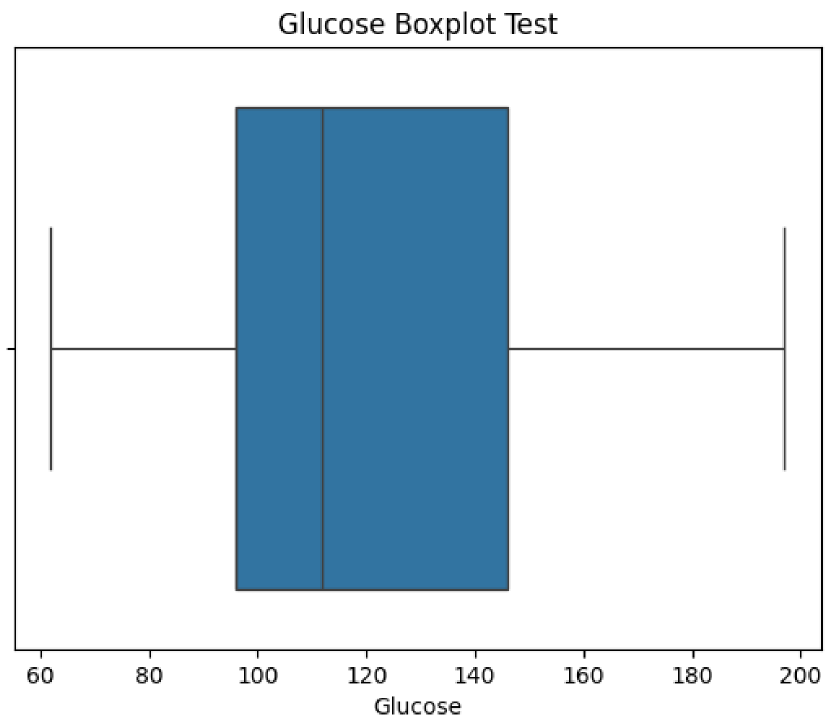

Figure 11: Glucose Boxplot - Test Set

Outliers are present similarly in test set.

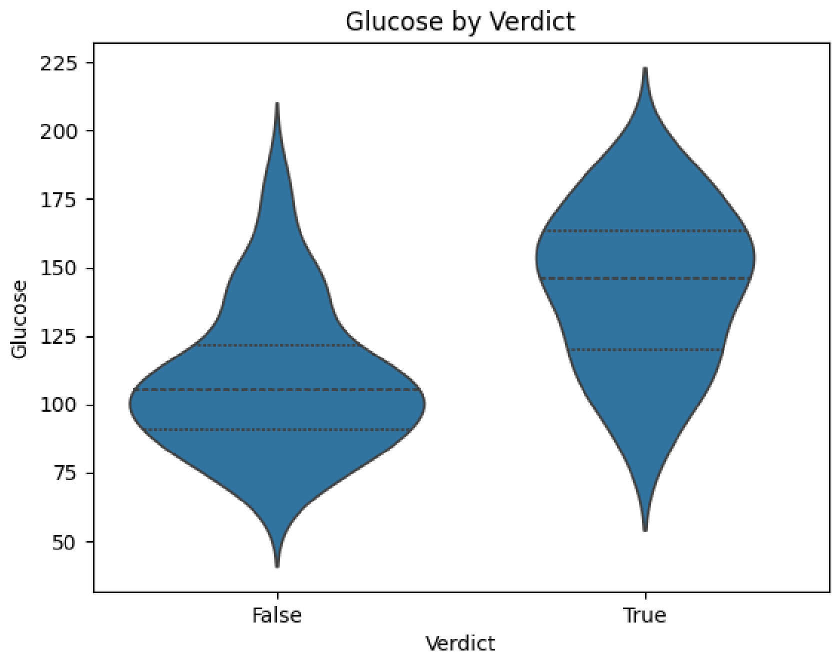

Figure 12: Glucose Distribution by Verdict - Test Set

Diabetic patients have generally higher glucose levels.

5.4.3 Categorical Variables

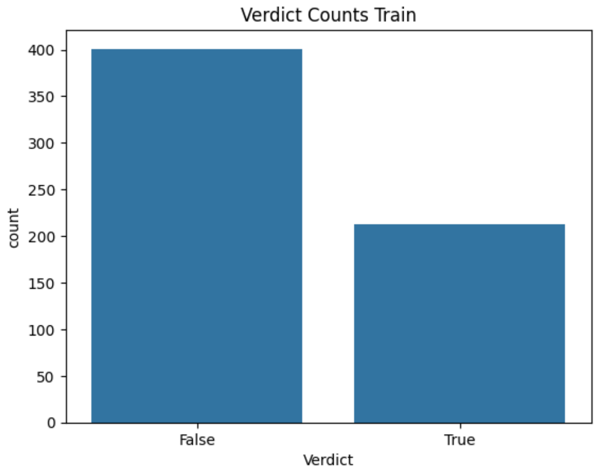

Figure 13: Distribution of the Target Variable Verdict

The dataset is imbalanced with many more non-diabetic (False) than diabetic (True) cases. Class imbalance techniques like oversampling, undersampling, or class weighting should be considered during model training to avoid bias.

5.5 Outlier Detection

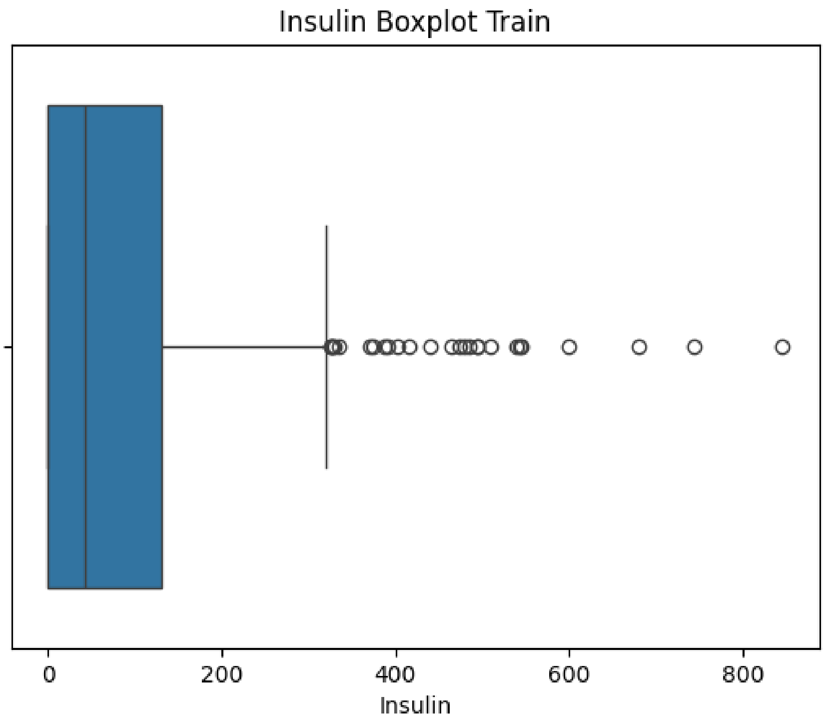

Figure 14: Boxplot of Insulin - Training Set

There are many high-value outliers in Insulin. These extreme values may be measurement errors or rare cases. Imputing zero insulin values and applying robust scaling or outlier trimming could improve model stability.

5.6 Correlation Analysis

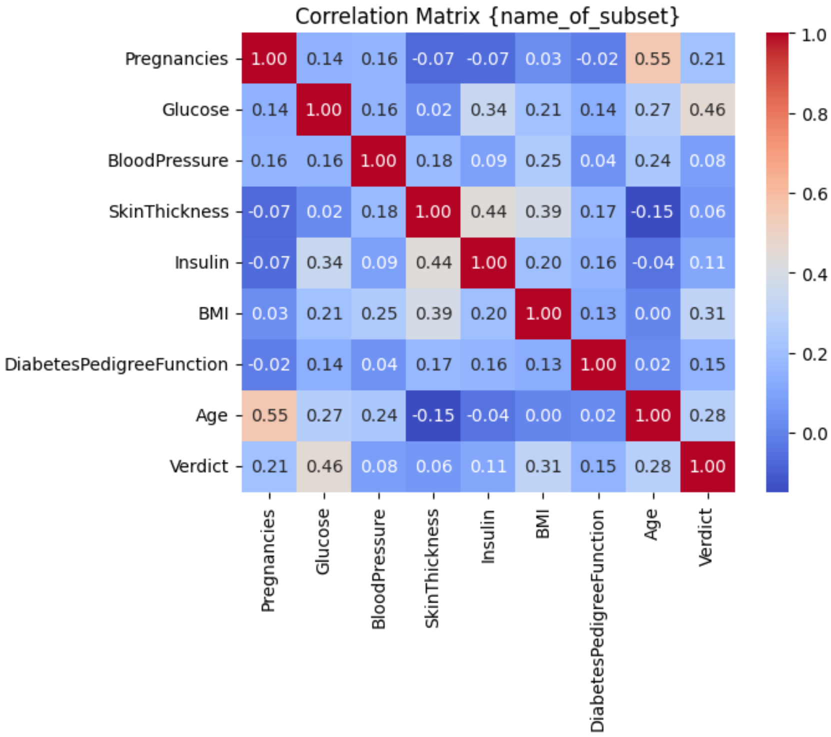

Figure 15: Correlation Matrix of Numerical Variables (Training Set)

Strong positive correlation between Glucose and Verdict confirms glucose is a critical diabetes predictor. Moderate correlations for BMI and Age suggest they also provide useful signals. Features with very low correlation could be candidates for removal or transformation.

5.7 Relationship to Target Variable

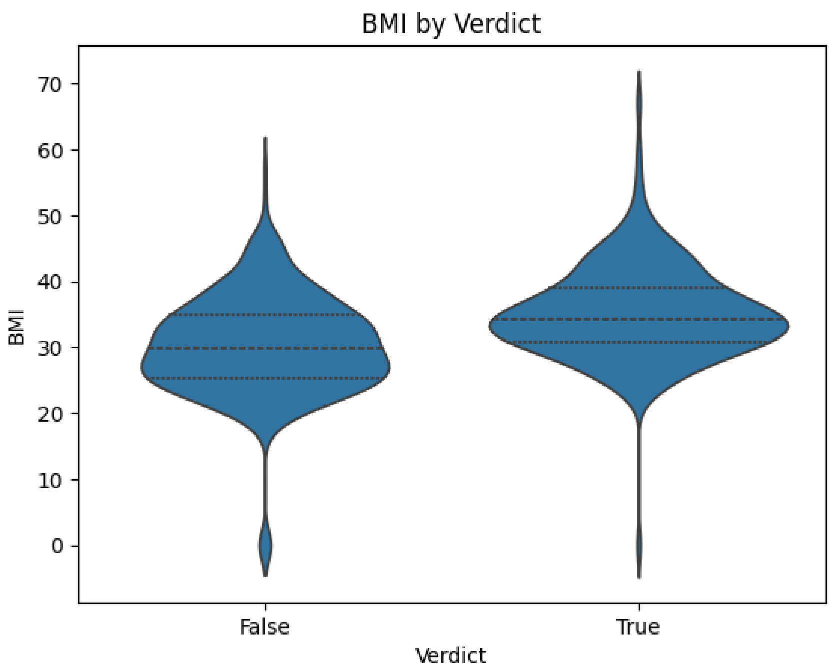

Figure 16: BMI Distribution by Verdict

Diabetic patients generally have higher BMI values, although some overlap exists. This indicates BMI is a helpful predictor but not definitive on its own, suggesting models should consider multiple features jointly.

5.8 Comments and Recommendations

- Zero values in Glucose, BloodPressure, and Insulin likely represent missing data; median imputation or domain-specific strategies should be used.
- Outliers in Insulin and BMI should be addressed using IQR-based filtering or robust scaling.
- The imbalanced target variable requires special techniques like SMOTE, class weighting, or balanced sampling.
- Features with strong correlations to the target should be prioritized, and interactions or transformations explored for improved predictive power.
- Numerical features should be standardized/scaled to improve model convergence and performance.

6 Model Evaluation

To evaluate the performance of our classifiers, we implemented a complete evaluation pipeline using the `scikit-learn` library. This pipeline includes model training, prediction, metric computation, and confusion matrix visualization.

6.1 Models Used

We selected two classification models:

- **Logistic Regression** – A linear model suitable for binary classification tasks.
- **Random Forest Classifier** – An ensemble method that builds multiple decision trees and aggregates their predictions.

Both models were trained using:

```
logistic = LogisticRegression().fit(X_train, Y_train)
forest = RandomForestClassifier(random_state=42).fit(X_train, Y_train)
```

6.2 Prediction and Metric Evaluation

After training, predictions were made on the test set:

```
y_pred_logic = logistic.predict(X_test)
y_pred_forest = forest.predict(X_test)
```

To evaluate performance, the following classification metrics were computed:

- **Accuracy** – Overall percentage of correct predictions.
- **Precision** – How many of the predicted positives were actual positives.
- **Recall** – How many of the actual positives were identified.
- **F1-score** – Harmonic mean of precision and recall.

The metrics were formatted in a custom text report using the following function:

```
def make_report(model_name, true_Y, predicted_Y):
    accuracy = accuracy_score(true_Y, predicted_Y)
    precision = precision_score(true_Y, predicted_Y)
    recall = recall_score(true_Y, predicted_Y)
    f1 = f1_score(true_Y, predicted_Y)
    report = (
        f"{model_name}:\n"
        f"Accuracy: {accuracy:.2f}\n"
```

```
        f"Precision:{precision:.2f}\n"
        f"Recall: {recall:.2f}\n"
        f"F1-score: {f1:.2f}\n"
    )
    return report
```

6.3 Confusion Matrix Visualization

To better understand the prediction performance, confusion matrices were plotted for both classifiers:

- **True Positives (TP)**: Correctly predicted diabetic patients.
- **True Negatives (TN)**: Correctly predicted non-diabetic patients.
- **False Positives (FP)**: Non-diabetic predicted as diabetic.
- **False Negatives (FN)**: Diabetic predicted as non-diabetic.

The matrices were plotted using the following function:

```
def build_cmat(name, true_Y, predicted_Y):
    cm = confusion_matrix(true_Y, predicted_Y)
    figure, ax = plt.subplots()
    disp = ConfusionMatrixDisplay(cm, display_labels=[False, True])
    disp.plot(ax=ax, cmap=plt.cm.Blues)
    ax.set_title(name)
    plt.close(figure)
    return figure
```

6.4 Summary

The entire evaluation is wrapped in a single function `evaluate()` which:

- Trains both models,
- Makes predictions on the test set,
- Generates a performance report,
- Returns two confusion matrix figures.

This approach ensures reproducible and interpretable comparison between classifiers.

7 Code Repository

The complete project is on GitHub right here.
Follow the instructions in the repository's `README.md` to clone and run the code.