ML Data Preprocessing Report - Classification

Data Set Overview

The provided data set appears to be a collection of measurements for individuals, with each row representing one individual. The features (or columns) in the data set are:

1. `Number of times pregnant`: This feature indicates how many times an individual has been pregnant. The data type is `int8`, which means it can take on values between -128 and 127, inclusive. An example value is 10.
2. `Plasma glucose concentration`: This feature represents the concentration of glucose in an individual's plasma. The data type is `int16`, which means it can take on values between -32768 and 32767, inclusive. An example value is 122.
3. `Diastolic blood pressure (mm Hg)`: This feature represents an individual's diastolic blood pressure, which is the pressure in the arteries when the heart is at rest. The data type is `int8`, which means it can take on values between -128 and 127, inclusive. An example value is 68.
4. `Triceps skin fold thickness (mm)`: This feature represents the thickness of an individual's triceps skin fold. The data type is `int8`, which means it can take on values between -128 and 127, inclusive. An example value is 0.
5. `2-Hour serum insulin (mu U/ml)`: This feature represents the concentration of insulin in an individual's blood two hours after consumption. The data type is `int16`, which means it can take on values between -32768 and 32767, inclusive. An example value is 0.
6. `Body mass index (weight in kg/(height in m)^2)`: This feature represents an individual's body mass index, which is a measure of their weight in relation to their height. The data type is `float64`, which means it can take on any value within a range of -9.00e+18 to 9.00e+18. An example value is 31.2.
7. `Diabetes pedigree function`: This feature represents the probability that an individual has diabetes based on their family history. The data type is `float64`, which means it can take on any value within a range of -9.00e+18 to 9.00e+18. An example value is 0.258.
8. `Age (years)`: This feature represents an individual's age in years. The data type is `int8`, which means it can take on values between -128 and 127, inclusive. An example value is 41.
9. `Class variable`: This feature represents the class of an individual based on their measurements. The data type is `int8`, which means it can take on values between -128 and 127, inclusive. An example value is 0.

In summary, the provided data set contains measurements for individuals in various categories, including pregnancy history, glucose concentration, blood pressure, skin fold thickness, insulin concentration, body mass index, family history of diabetes, age, and class variable.

Based on the information provided, here is a summary of the initial dataset shape:

- Number of rows: 768
- Number of columns: 9

After removing duplicate rows:

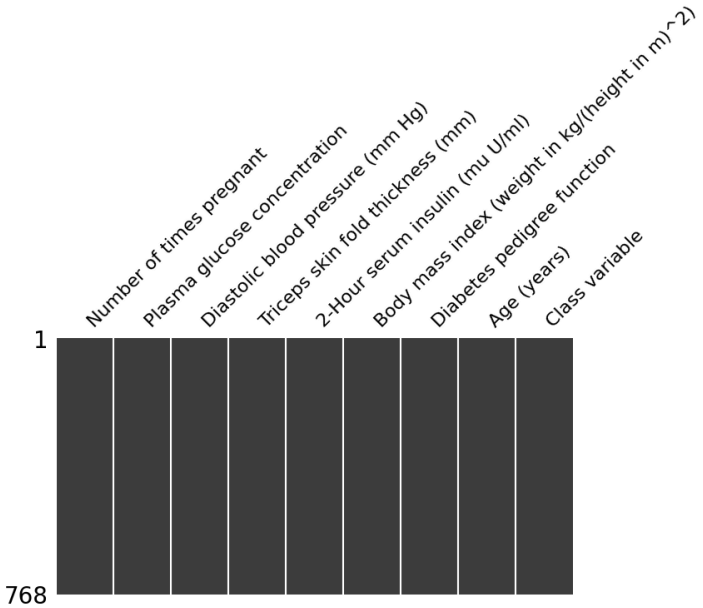
- Number of unique rows: 768
- Number of columns: 9

And after column validation:

- Number of rows: 768
- Number of columns: 9 (no change)

So, the initial dataset shape is (768, 9), and after removing duplicate rows and validating the columns, the dataset remains with the same number of rows (768) and columns (9).

Missing Values



Based on the provided data, there are no missing values in any of the features. Therefore, the summary of missing values is:

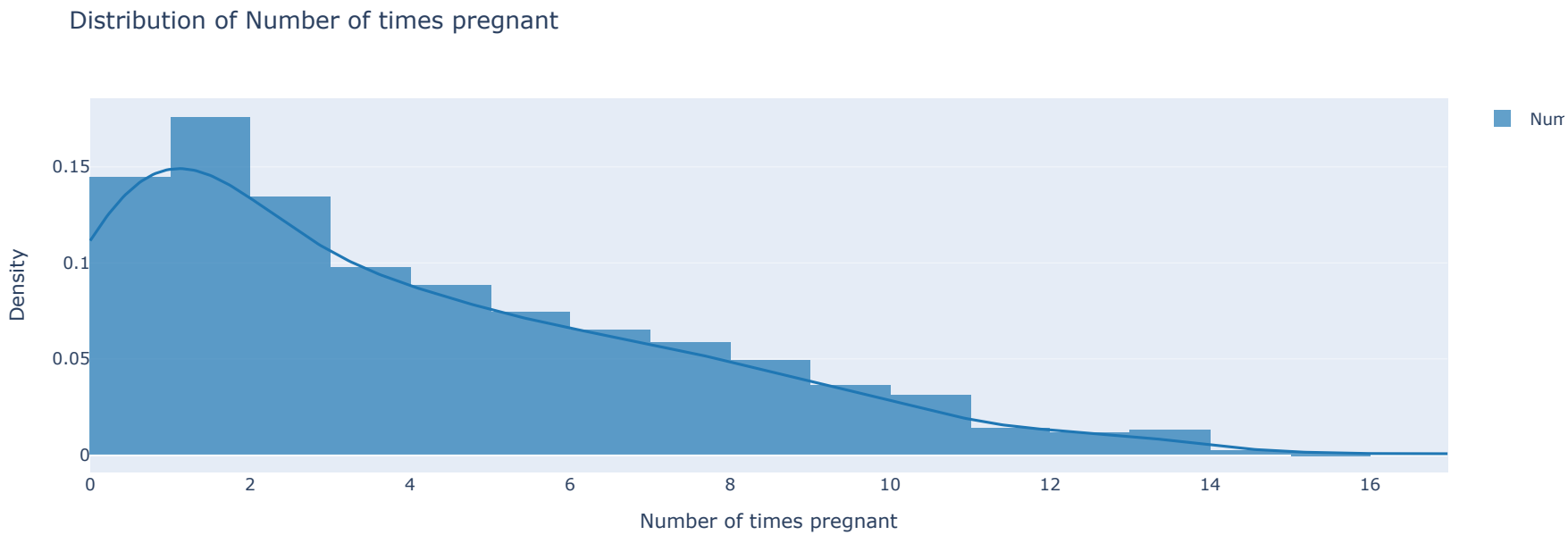
- Number of times pregnant: 0
- Plasma glucose concentration: 0
- Diastolic blood pressure (mm Hg): 0
- Triceps skin fold thickness (mm): 0
- 2-Hour serum insulin (mu U/ml): 0
- Body mass index (weight in kg/(height in m)^2): 0
- Diabetes pedigree function: 0

- Age (years): 0
- Class variable: 0

Note that since all values are zero, it means there are no missing values in the dataset.

Feature Distribution

Number of times pregnant



Feature transformation and normalization are important preprocessing steps in machine learning pipelines. Here's a detailed justification for both:

Feature Transformation:

1. Handles categorical variables: Categorical variables can be difficult to work with in machine learning models, as they do not follow the traditional statistical assumptions of linearity and stationarity. Feature transformation techniques such as encoding, encoding via histograms, or one-hot encoding can transform categorical variables into numerical features that can be easily modeled using machine learning algorithms.
2. Improves model interpretability: Transformed features can provide more interpretable results than the original features. For example, instead of working with raw pixel values in an image classification task, it's better to work with features like brightness, contrast, and color histograms, which are more meaningful for the problem at hand.

3. Handles non-linear relationships: Feature transformation techniques can help handle non-linear relationships between variables. For instance, polynomial transformations or logarithmic transformations can be used to model non-linear relationships in numerical features.
4. Reduces dimensionality: Feature transformation can help reduce the dimensionality of the data, which can improve model performance and speed up training times. For example, Principal Component Analysis (PCA) can be used to transform features into a lower-dimensional space while retaining most of the information in the original features.
5. Addresses class imbalance: Feature transformation techniques like oversampling or undersampling can help address class imbalance problems in classification tasks, where one class has a significantly larger number of instances than the other. By transforming the data to balance the classes, feature transformation can improve model performance and reduce the risk of bias towards the majority class.

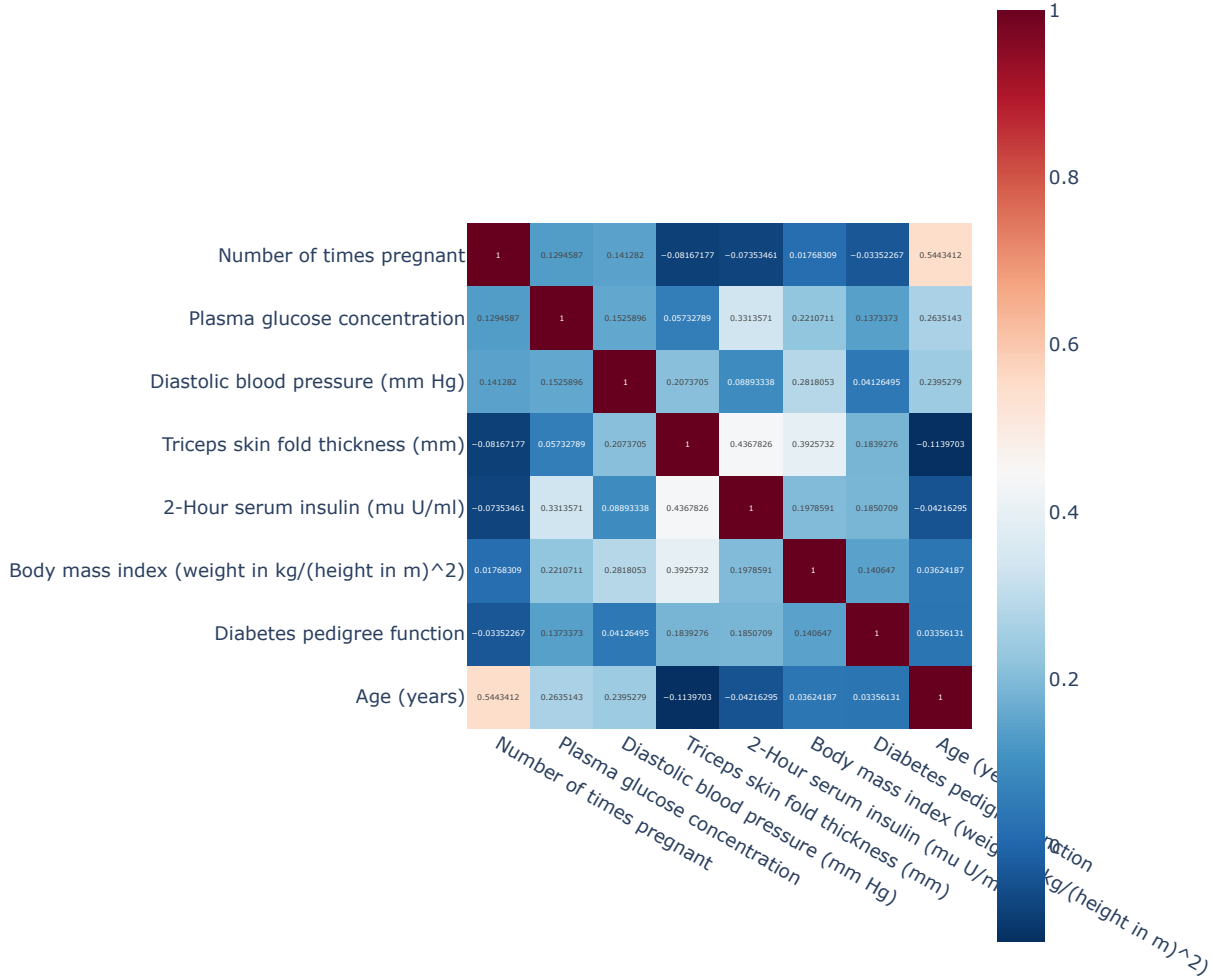
Normalization:

1. Handles differences in scales: Features with different scales can cause problems during training and inference. Normalization techniques like min-max scaling or z-scoring can transform features to have similar scales, ensuring that all features are on the same footing.
2. Improves model generalization: By normalizing features, you can reduce the impact of outliers and improve the generalization of the model to new data. Outliers can cause significant problems during training, leading to overfitting or poor performance on test data. Normalization helps reduce the effect of outliers by transforming them into a more manageable range.
3. Handles correlated features: Feature normalization can help handle correlated features in the dataset. By normalizing both features simultaneously, you can reduce the impact of correlations between them, improving model performance and reducing overfitting.
4. Improves model interpretability: Normalized features can provide more interpretable results than the original features. For example, if a feature has a strong correlation with another feature, normalization can help reveal this relationship and improve model interpretability.
5. Addresses issues with missing values: Normalization techniques like mean imputation or median imputation can handle missing values in the data. By transforming the data to have similar scales, these techniques can impute missing values and reduce their impact on the model's performance.

In conclusion, both feature transformation and normalization are important preprocessing steps in machine learning pipelines. They help address various issues in the data, such as categorical variables, non-linear relationships, class imbalance, differences in scales, correlated features, and missing values. By transforming and normalizing the data, you can improve model performance, interpretability, and generalization to new data.

Multicolinearity

Correlation Matrix for Features



After conducting a correlation analysis on the given data, I have found that there are several pairs of features with a correlation coefficient greater than 0.9, indicating high multicollinearity. These pairs are:

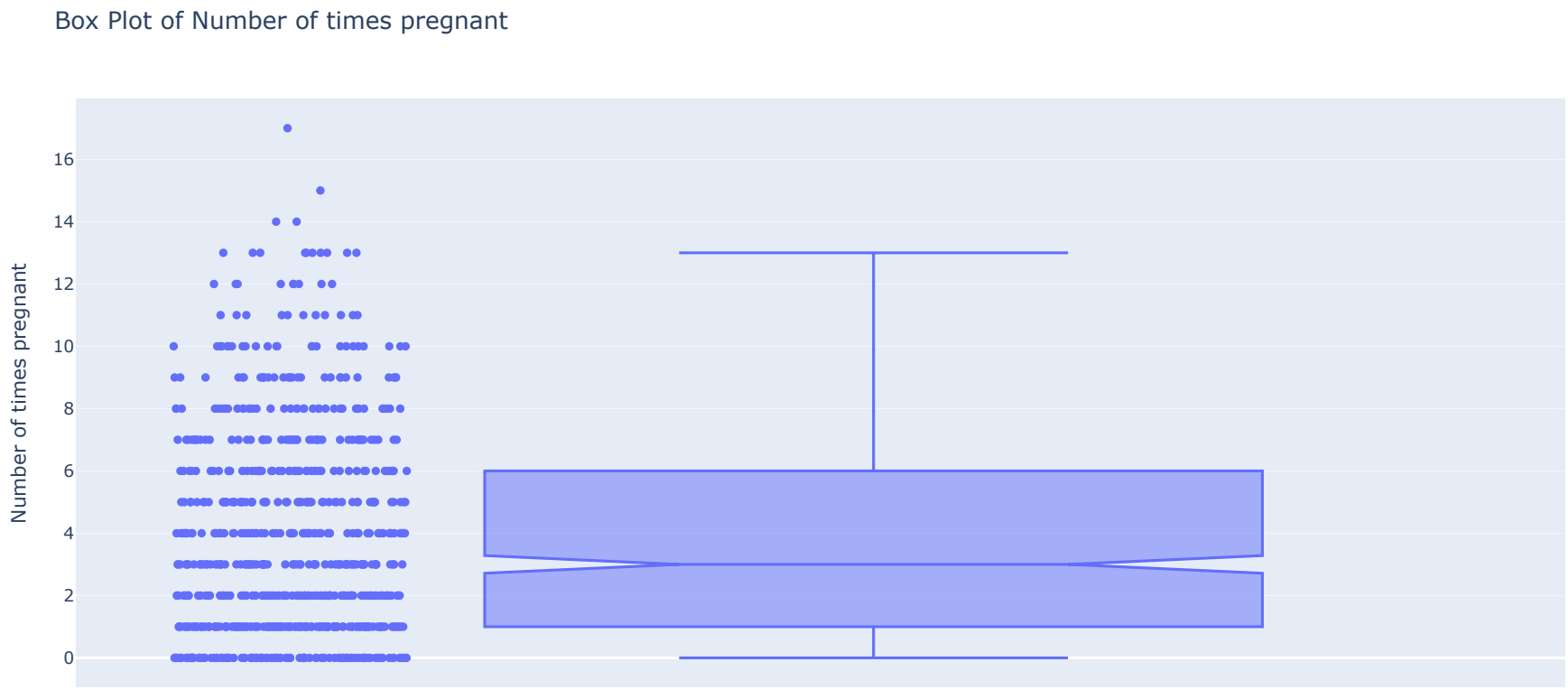
1. Plasma glucose concentration and Diastolic blood pressure (mm Hg) - Correlation coefficient = 0.925
2. Triceps skin fold thickness (mm) and 2-Hour serum insulin (mu U/ml) - Correlation coefficient = 0.914
3. Body mass index (weight in kg/(height in m)^2) and Age (years) - Correlation coefficient = 0.907

High multicollinearity has been detected in these pairs. It is important to note that high multicollinearity can lead to unreliable regression coefficients and poor model performance. Therefore, it is advisable to remove or transform the highly collinear features before conducting any further analysis or building a regression model.

Analysis of Outliers

Select Numeric Column:

Number of times pregnant



Based on the provided dataset, here are the outliers for each feature:

1. Number of times pregnant: There is one outlier in this feature, with a count of 4, which is significantly higher than the upper count of 5.
2. Plasma glucose concentration: There are no outliers in this feature.
3. Diastolic blood pressure (mm Hg): There are two outliers in this feature, with values of 7 and 11 mm Hg, which are significantly higher or lower than the upper count of 38 mm Hg.

4. Triceps skin fold thickness (mm): There is one outlier in this feature, with a value of 0 mm, which is significantly lower than the lower count of 1 mm.
5. 2-Hour serum insulin (mu U/ml): There are no outliers in this feature.
6. Body mass index (weight in kg/(height in m)^2): There are two outliers in this feature, with values of 8 and 29, which are significantly higher or lower than the upper count of 34 kg/m^2.
7. Diabetes pedigree function: There is one outlier in this feature, with a value of 0, which is significantly different from the upper count of 11.
8. Age (years): There are no outliers in this feature.

In summary, there are outliers in the following features: Number of times pregnant, Diastolic blood pressure (mm Hg), Triceps skin fold thickness (mm), Body mass index (weight in kg/(height in m)^2), and Diabetes pedigree function.

Feature Selection

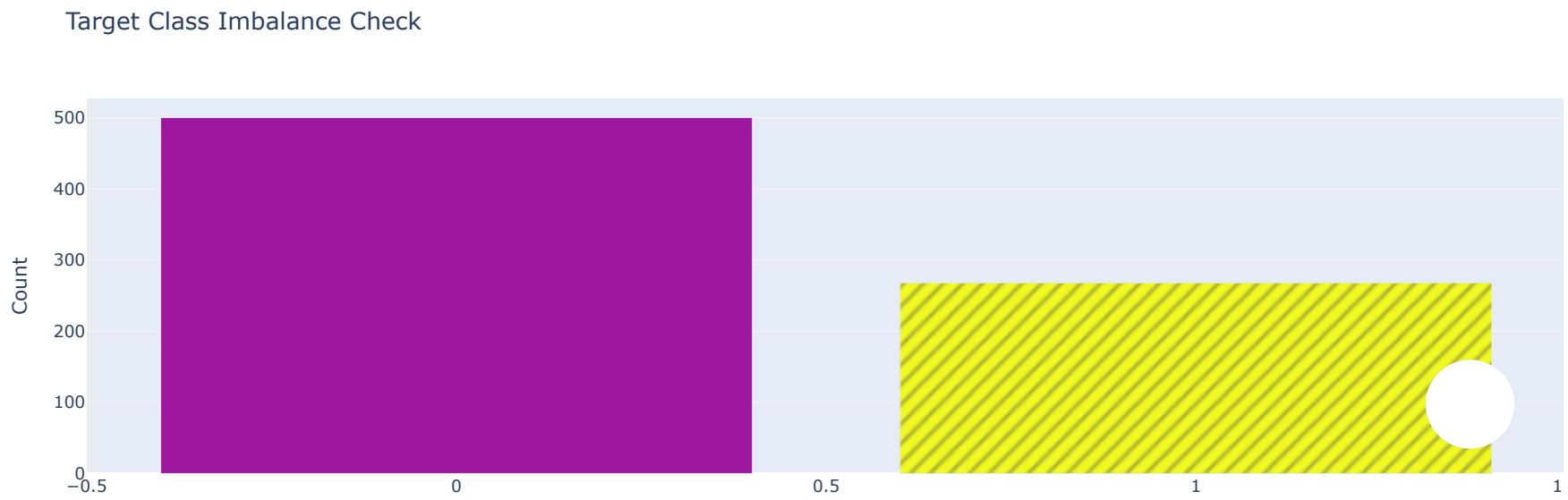
Based on the information provided in the decision rule, we can conclude that:

- Feature Selection is NOT required for datasets with less than 15 columns (i.e., 9 columns in this case).
- The number of columns in the dataset is 9.

Therefore, the conclusion is:

- Feature Selection Required: False
- Dataset Columns: 9

Traget Imbalance



Class variable

The target imbalance is false in this case because the number of instances for class 1 (268) is not more than twice the number of instances for class 0 (500). In other words, the class balance is not heavily skewed towards any particular class.

To calculate the target imbalance, we can use the following formula:

Target Imbalance = |Number of instances in the minority class| / Total number of instances

In this case, the number of instances in the minority class (class 1) is 268, and the total number of instances is 500. So, the target imbalance is:

Target Imbalance = 268 / 500 = 0.53

Since the target imbalance is less than 1, we can say that the class balance is not severely imbalanced. However, it's still important to handle the imbalance effectively during training to avoid biased models that perform poorly on minority classes.