Predictive Modeling for Detection of Type-2 Diabetes: Evaluation of Three Machine Learning Algorithms

CIND 820: Big Data Analytics Project

Initial Result and Codes

Yitayal Mengistu [501211839]

Submitted to:

Dr. Ceni Babaoglu (supervisor)

July 2, 2024

Contents

Table of Figures

Initial Result and Codes

1. Data Preprocessing and Transformation

Visualizing a numeric feature through a histogram or density plot offers valuable insights into its distribution and characteristics, aiding in the determination of whether normalization or transformation is necessary. The relevance and techniques of normalization and transformation are discussed as follows.

1.1. Data Normalization and Scaling

To ensure that the features contributes proportionately or no single feature dominate the learning process of the model due to it scale, transforming the features in a dataset to a common scale without distorting differences in the ranges of values is important. This process prevents features with larger ranges from disproportionately influencing the model. This data preprocessing step is called data normalization(Izenman, 2008)

Data normalization techniques that involve mathematical transformations are applied to numeric features, mainly using mini-max scaling or z-score standardization (standard scaling) (Izenman, 2008). Normalization techniques are not directly applied on categorical data. If necessary, we use label encoding or One-Hot encoding on categorical data before applying normalization. The methods or approaches and implementation procedures for encoding of Categorical data are discussed under the sections “label encoding” and “One-Hot encoding” below.

In this dataset, only bmi (body mass index) is normalized using the standard mini-max scaling technique. The Mini-Max scaling is used to normalize or rescale the range of features to a fixed

rage, typically between 0 and 1 or between -1 and 1 depending on the characteristics of the data and requirements of the machine learning algorithm. With regard to the characteristics of the data, if the data is strictly non-negative, scaling to [0, 1] keeps the values within a non-negative range.

Second Machine learning algorithms that rely on gradient descent optimization, such as logistic regression perform better or converge faster when the input features are normalized to the [0, 1] range. In this particular case, the standard formula for Mini-Maz normalization is(Brownlee, 2020; Giuseppe et al., n.d.).

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Where:

$X\ is\ the\ orginal\ value\ of\ the\ feature$

$X_{min}\ is\ the\ minimum\ value\ of\ the\ feature$

$X_{max}\ is\ the\ maximum\ value\ of\ the\ feature\ and$

$X_{scaled}\ is\ the\ scaled\ value\ of\ X$

On contrary, some Machine learning algorithms that are sensitive to the symmetry of the input data, such as support vector machines (when using kernels like RBF) and principal component analysis perform better when the mean of the data is around zero. Scaling the data to [-1,1] helps center the data around zero. To scale data to [-1, 1], we can modify the standard formula for Mini-Maz normalization shown above (Brownlee, 2020):

$$X_{scaled} = 2 * \frac{X - X_{min}}{X_{max} - X_{min}} - 1$$

1.2. Data Transformation-Log Transformation

The choice of data transformation techniques, such as log transformation, reciprocal transformation or polynomial transformation of numeric variables is an important preprocessing step in many machine learning and statistical modeling tasks for several purposes(Kuhn & Johnson, 2019).

According to (Kuhn & Johnson, 2019; Sharma, 2022;)log transformation may be used to normalize skewed data for models that assume normal distribution and to reduce the variance of data; to handles outliers by compressing the range of data for features with extreme values; to linearize the relationship between features and the target variable; to improve model performance by addressing issues related to skewed data, and non-linearity; and to improve interpretability and get more intuitive insights, especially when dealing with multiplicative processes. For instance, the log-transformed data might help in understanding proportional changes rather than absolute changes.

The general formula for the popular log transformation technique utilized to stabilize the variance, make the data more normally distributed, and reduce the effect of outliers is given as follows:

$\ln(X) = \ln(X + c)$

Where:

$\ln(X) - the\ natural\ log - transformed\ value\ of\ X$

$c -\ is\ a\ constant, often\ set\ to\ 1,\ added\ to\ avoid\ taking\ the$ log of 0 or negative numbers

The following two histograms show the distribution of bmi (body mass index before and after log transformation. After log transformation, the distribution looks closer to normal. The next two box plots also indicate the detection and removal of outliers on the same feature, BMI.

Figure 1- Histogram-BMI before log transformation

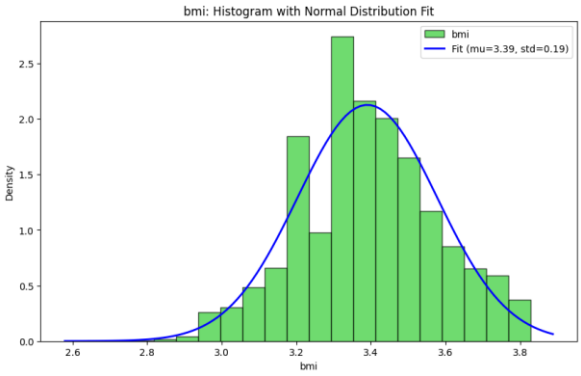


Figure 2-Histogram-BMI after log transformation

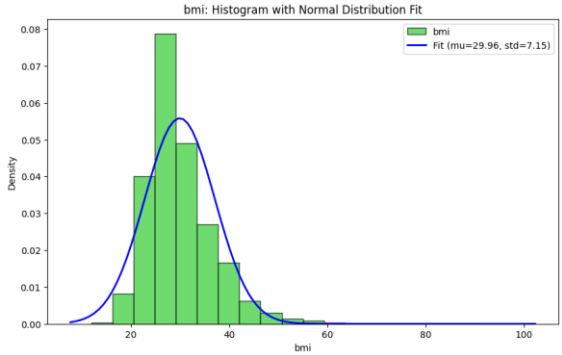


Figure 3- bmi box plot to detect outliers

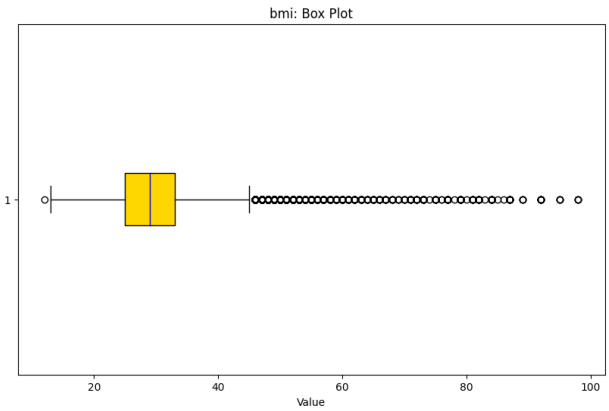
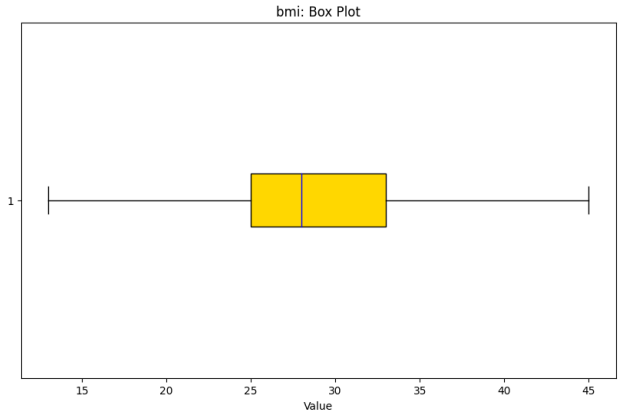


Figure 4- bmi box plot after dropping outliers



1.3. Handling Categorical Data: Label Encoding and One-Hot Encoding

Since Machine Learning Algorithms are based on mathematical operations that require numerical input and often assume that the input feature has an inherent order or meaningful distance between values, they do not handle categorical data directly. Using raw categorical data may lead to misleading interpretations and poor model performance(Roy, 2019) . To efficiently utilize categorical data in machine learning algorithms, it is crucial to transform the data into a suitable numeric format using techniques like label encoding, one-hot encoding, binary encoding or frequency encoding(Bobbitt, 2022; Roy, 2019). These transformations ensure the categorical data is represented in a way that the algorithms can process without introducing misleading relationships.

For this project, however only one-hot encoding scenario is applied to run the ML algorithms turn see the impacts on the performance of the algorithms because label encoding implicitly assumes an ordinal relationship between categories, which may not be true for nominal categories. Another advantage to One-Hot encoding is that it is easier to interpret the encoded data, as each category is represented explicitly (Roy, 2019). Most of the features in this project have binary data

in numerical format and don't require one-hot encoding or normalization. See the comparison on "results and discussion" section.

2. Exploratory Data Analysis (EDA)

To gain comprehensive insights into the dataset, an exploratory data analysis (EDA) has been conducted in this dataset. EDA mainly undertaken to understand the distribution of the dataset, identify outliers, explore relationships between variables, and test hypotheses. Key components included employing descriptive statistics to summarize data, utilizing various visualizations such as histograms, box plots, scatter plots, and heatmaps to visualize data distributions and relationships. Additionally, we assessed relationships between variables using correlation coefficients and scatter plots, complemented by creating summary tables to highlight the dataset's main characteristics. These efforts collectively ensured a thorough understanding of the dataset and informed strategic decisions for the feature selection, analysis and modeling phases. Some selected exploratory data analysis visualizations are found under Appendix-A.

3. Experimental Design

3.1. Feature Selection and Cross Validation on the Training Dataset

More effective and reliable predictive models can be built by selecting the most relevant features from a comprehensive list of potential features in our dataset. This selection process is crucial for enhancing model performance, reducing overfitting, improving interpretability, and increasing efficiency(Kuhn & Johnson, 2019).

In this project, after splitting the dataset into training and testing sets, feature selection was performed on the training set alone to maintain the independence of the testing set and ensure unbiased model evaluation. This precaution avoids overly optimistic performance estimates and ensures the model generalizes well to new, unseen data, reflecting real-world scenarios where the model encounters previously unseen data points.

The feature selection, cross-validation, and performance testing of the three classification models were conducted as follows to ensure robust feature selection and model evaluation, leading to the development of reliable and generalizable predictive models.

1. **Data Splitting**: The dataset was divided into 80:20 training and testing sets using the train-test split method.

   - **Feature Selection:** Feature selection was performed on the training set alone using two techniques: measure of Gini impurity filtering by importance and recursive feature elimination with cross-validation (RFECV).

2. **Cross-Validation:** 5-fold cross-validation was applied to the training dataset. This involved:

   - Splitting the training data into k = 5 subsets (folds).
   - Using k-1 = 4 folds for training and the remaining 1-fold for validation.
   - Repeating this process 5 times, with each fold used once as a validation set.
   - Calculating average performance metrics (such as accuracy, precision, recall) across all 5 folds to obtain a reliable estimate of model performance on unseen data.

3. **Model Evaluation:** After completing cross-validation on the training dataset, the final performance of the three models (Logistic Regression, Decision Tree, and Random Forest)

was evaluated on the independent testing dataset that was originally set aside. This provided an unbiased estimate of how well the models generalize to new, unseen data.

3.2. Univariate Testing for Classification

A chi-square test for categorical features and ANOVA test for numerical features used to identify features that show a strong statistical relationship with the target variable. This step helped quickly filter out obviously irrelevant features based on their individual merit. However, utilizing and comparing additional feature selection techniques was necessary to confirm suitability, especially for decision tree and random forest due to the inherent ability of these two algorithms to handle feature interactions and non-linear relationships without relying heavily on pre-filtering.

3.3. Recursive Feature Elimination with Cross-Validation (RFECV)

The RFECV method, which integrates feature elimination with cross-validation, is used to iteratively remove less important features and evaluates model performance through cross-validation. Due to the inherent capability of Decision Trees and Random Forests to handle feature interactions and non-linear relationships, RFECV was first assumed to help identify the optimal subset of features that collectively contribute most to model accuracy. However, RFECV resulted in very restrictive feature selection for both Decision Trees and Random Forests, only 4 features for Decision Trees and 5 features for Random Forests were selected. For logistic regression, it selected 14 features.

This led us to choose another feature selection alternative, feature Selection using gini importance for Decision Trees and Random Forests and Feature Selection using Model coefficients

and p-values for Logistic Regression. The following figures show feature selection results for logistic regression, decision trees and random forest models using RFECV and feature importance selection criteria.

Figure 5- Logistic regression feature selection using feature coefficients

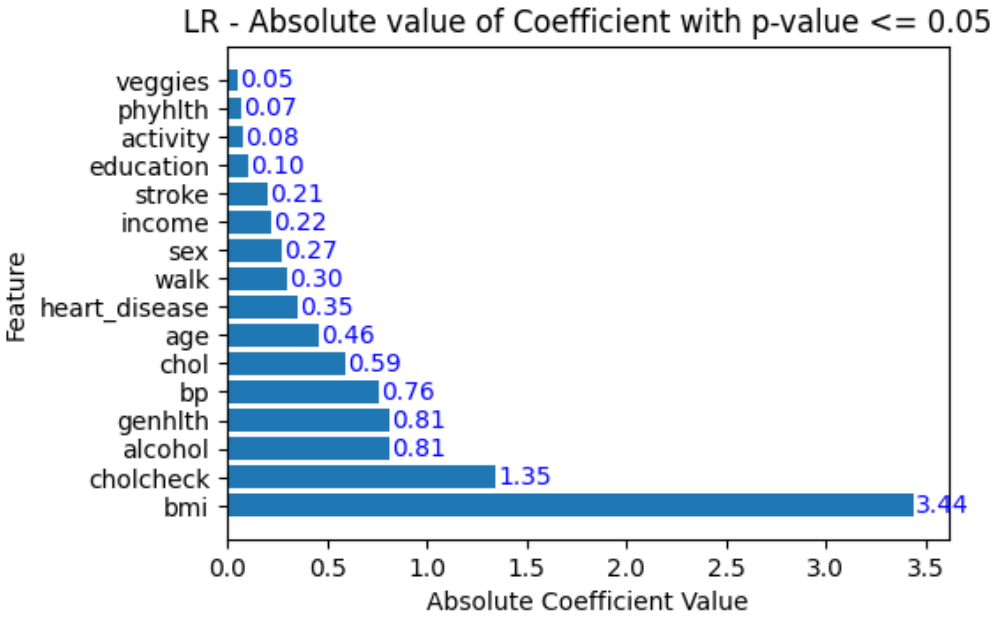


Figure 6- Logistic regression feature selection using RFECV

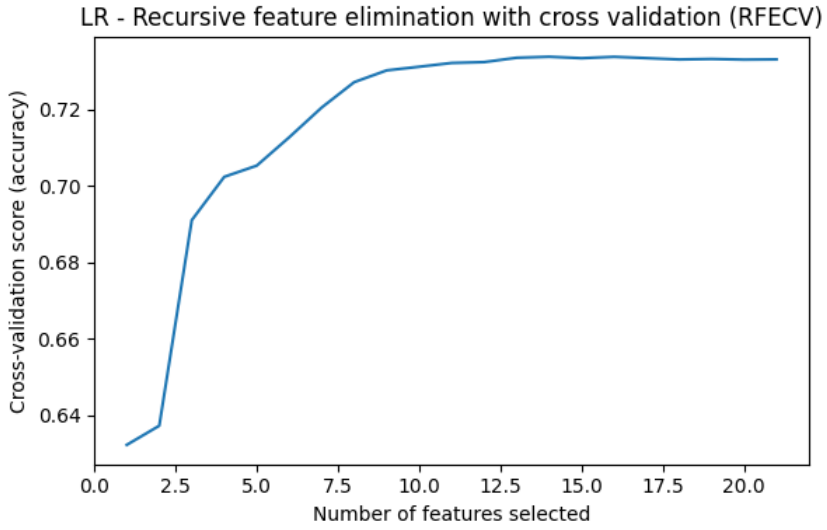


Figure 7-Decision tree feature selection using RFECV

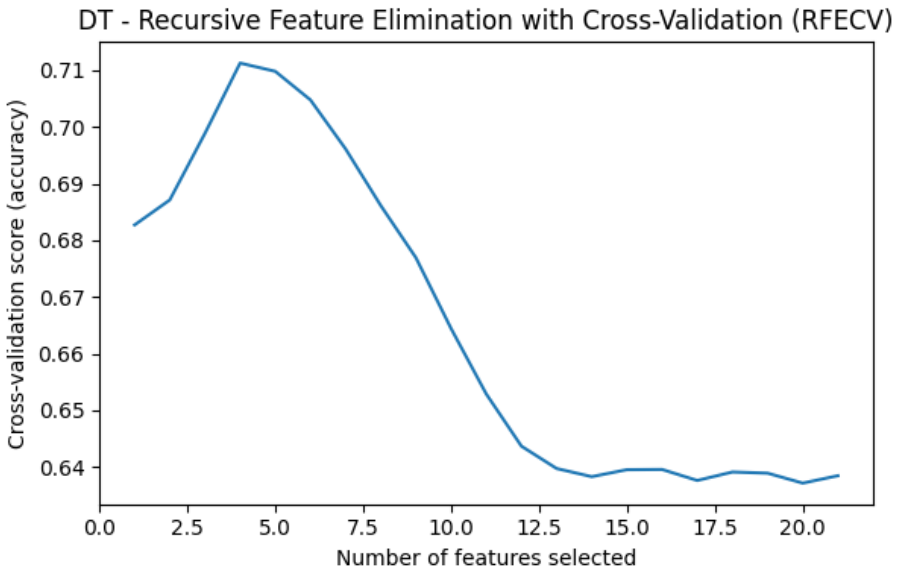


Figure 8- Decision tree feature selection using gini impurity

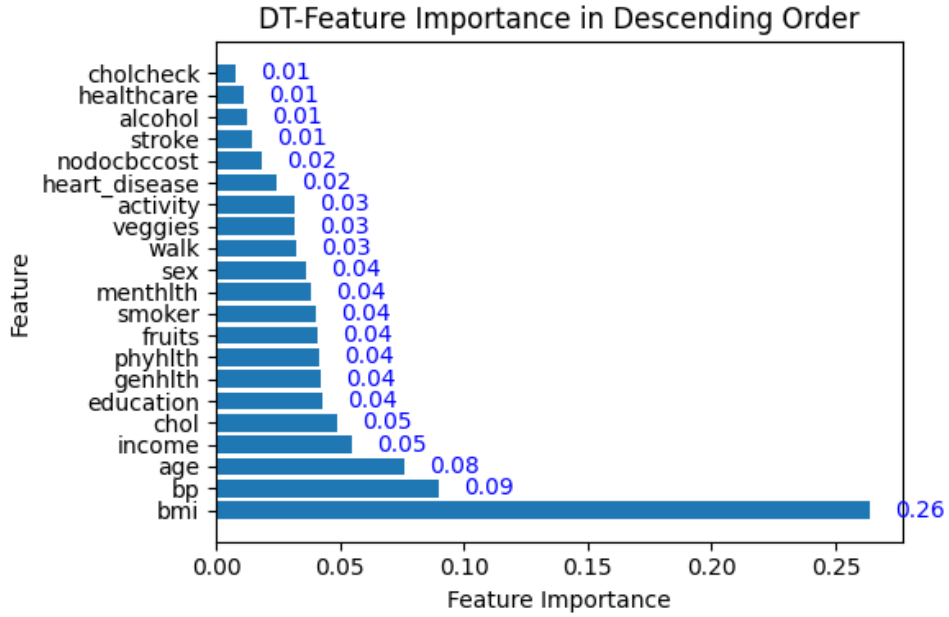


Figure 9- Random Forest feature selection using gini impurity

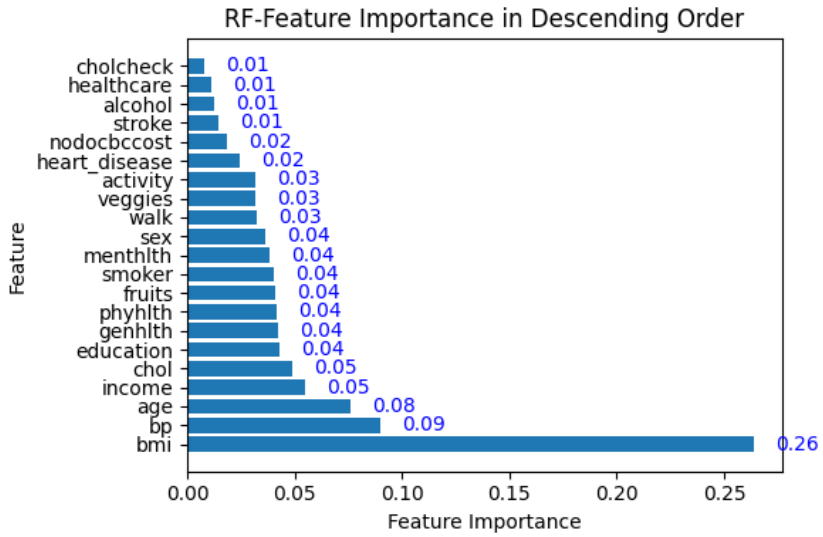
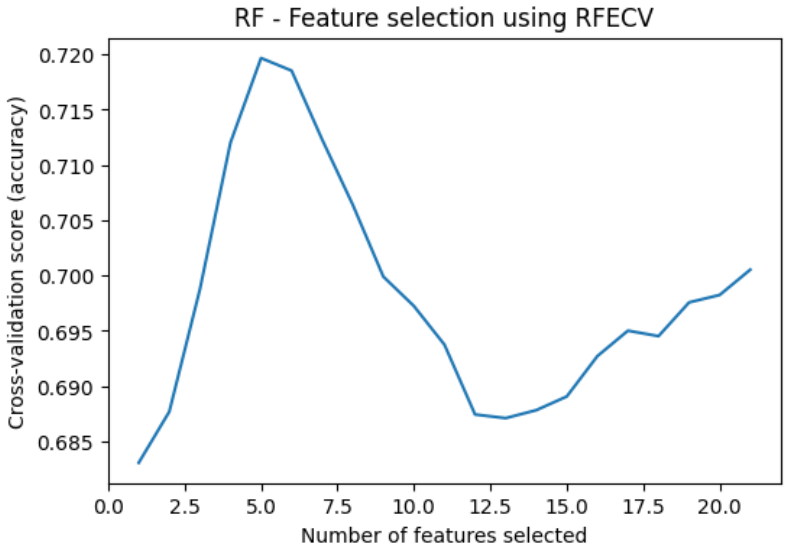


Figure 10- Random Forest feature selection using RFECV



3.4. 5-Fold Cross Validation

Train-Test Split and K-fold Cross-Validation are both techniques used for evaluating machine learning models. Unlike simple train-test splitting, k-fold cross-validation aims to mitigate biases in performance estimation by averaging results across multiple data splits. This method is preferred

for its ability to provide a more accurate gauge of model performance, making it suitable for model evaluation, selection, and assessing generalizability. Additionally, it helps gauge model stability through comparisons of scores across the 5 folds used(Sohil et al., 2022).

In this project, a 5-fold cross-validation technique was utilized on the training dataset to assess the performance of three machine learning models: Logistic Regression, Decision Tree, and Random Forest. First, the dataset is divided in to 5 folds or subsets of about equal size, the cross-validation process is repeated 5 times and, in each iteration, one of the 5 subset is used as the validation set, and the remaining 4(k-1) subsets are used as a training set. This means for each iteration the model is trained on the training set and evaluated on the validation set which results in k=5 performance metrices (accuracy, F1-score) one for each fold. Finally, the k=5 performance metrices are averaged to obtain a single estimation of model's performance for each algorithm. This average performance metrics is often considered a more reliable estimate of the model's performance compared to using a single train test split.

3.5. Optimization of Key Model Parameters

Parameter tuning and optimization techniques are critical steps in improving the performance of classification algorithms. Key parameters can be adjusted and alternative optimization techniques can be used to fine tune the performance of the models based on the available dataset and problem under consideration. To find the optimal combination of parameters for the three-classifier considered in this project, GridSearch with cross validation technique was used.

Given the large working dataset in this project, predicting outcomes from a large, complex dataset with numerous features and data points significantly impacts training time and convergence. Accordingly, top 3 parameters with different ranges of values for each classification algorithms are selected for comparative analysis of model performance. The optimized values are given in table 1 below.

Optimization of Key Parameters - Logistic Regression

The top 3 parameters selected for fine tuning the Logistic regression model performance is regularization strength, regularization penalty (penalty) and solver. To determine the optimal values for regularization strength, penalty, and solver for the Logistic Regression model, a technique called Grid Search with Cross-Validation (GridSearchCV) was used (Géron, 2019). This method exhaustively searches through a specified parameter grid to find the best combination of parameter values based on cross-validation performance. Using GridSearchCV, the performance of the Logistic Regression model with different combinations of Regularization strength (C), penalty, and solver values can be systematically evaluated to determine the optimal accuracy of the model (Géron, 2019). The typical ranges of three most important parameters in logistic regression areas follows:

Regularization strength prevent overfitting by adding a penalty for larger coefficients in the model. Regularization Strength (C in Logistic Regression) controls the inverse of regularization strength. A smaller C value indicates stronger regularization. The most commonly used values are: [0.01, 0.1, 1, 10, 100].

Regularization penalty specifies the type of regularization to be applied. Common options are l1 (Lasso) for promoting sparsity, l2 (Ridge) for promoting smaller coefficients, and 'elasticnet' which is a combination of both. The most commonly used options are: [l1, l2, elasticnet, none].

The solver parameter specifies the optimization algorithm used to fit the model. The choice of solver can significantly impact the convergence speed, accuracy, and overall performance of the model, particularly for large datasets or datasets with specific characteristics(Géron, 2019). Not all solvers support all penalties and solvers in this context refers to algorithm to use in the optimization problem. Common solvers include: liblinear, lbfgs, newton-cg, sag, saga(Géron, 2019).

Optimization of Key Parameters - Decision Tree

To fine-tune the performance of a decision tree model for classification purpose, the three most important parameters considered in this analysis include: maximum depth which controls the maximum depth of the tree to prevent overfitting; minimum samples split to specify the minimum number of samples required to split an internal node, controlling tree size and preventing overfitting and minimum samples leaf to set the minimum number of samples that must be present in a leaf node, avoiding leaves based on very few samples and improving robustness. These parameters are crucial for managing the complexity of the decision tree and achieving a good balance between bias and variance, leading to better performance on both training and test datasets(Géron, 2019).

To find the optimal combination of three most relevant hyperparameters for a Decision Tree classifier, GridSearchCV was used in a similar manner to the Logistic Regression. For a Decision Tree, the typical ranges for these hyperparameters used in python code are given as follows:

- Values for max_depth: [None, 10, 20, 30, 40, 50]

- Values for min_samples_split: [2, 5, 10, 20]
- Values for min_samples_leaf: [1, 2, 5, 10]

Optimization of Key Parameters - Random Forests

Similarly, to fine-tune the performance of random forests model for classification purpose, three most important parameters considered in this analysis include the number of estimators, maximum features, and maximum depth. The number of estimators parameter is used to specify the number of trees in the forest. In relation to this, more trees generally lead to better performance but at a higher computational cost. The max features parameter is used to control the number of features considered for each split. Adjusting this parameter helps reduce overfitting and introduces randomness, improving model generalization. The max depth parameter is used to limit the maximum depth of each tree, preventing overfitting and controlling the complexity of the model(Raschka & Mirjalili, 2019) . These parameters are crucial for managing the complexity and diversity of the trees in the Random Forest, which in turn helps in achieving better performance on both training and test datasets. Below are some key parameters for the three models along with their default values and optimized values using GridSearchCV method.

Table 1: Optimization of models' Parameter using GridSearch with cross validation

Algorithms	Parameters To be optimized	Default values	Optimized values (GridSearchCV)
Logistic Regression	Regularization strength	1	0.1
	Penalty	12	L1
	Solver	lbfgs	saga
Decision Tree	Max depth	none	10
	Min samples split	2	20
	Min samples leaf	1	1

Random Forest	Number of estimators	100 trees	300 trees
	Max features	auto	sqrt
	Max depth	none	10

To optimize the values of these parameters of the Random Forest, GridSearchCV is used in a similar manner as the LR and DT models above. For Random Forests, the typical ranges for these hyperparameters used in python code are as follows(Raschka & Mirjalili, 2019).

- n_estimators: [50, 100, 200, 300]
- max_features: [sqrt, log2, None]
- max_depth: [None, 10, 20, 30, 40, 50]

4. Modelling Classification Algorithms: Logical and Theoretical Farmwork

4.1. Logistic Regression for Classification

logistic regression is a robust algorithm for binary classification due to its simplicity, interpretability, and efficacy in handling linearly separable datasets(Fullerton, 2009). By optimizing coefficients and employing regularization, logistic regression provides reliable predictions while offering insights into underlying data relationships(Fullerton, 2009).

logistic regression estimates the probability of an instance belonging to a specific class (Y=1) using the sigmoid function, transforming a linear combination of inputs to a value between 0 and 1. The algorithm classifies an instance as Y=1 if the predicted probability p(x) is 0.5 or higher; otherwise, it predicts Y=0. The decision boundary is determined where the linear combination of input features equals zero, separating predictions of Y=1 and Y=0(Train, 2001).

Logistic regression computes a weighted sum of input features, transformed by the logistic function to generate a probability score between 0 and 1, representing the likelihood of an instance belonging to the positive class. During training, coefficients are optimized to maximize the likelihood of observed data, typically using techniques like gradient descent to iteratively adjust coefficients and minimize the logistic loss function. For predictions, new instances are evaluated based on probability scores derived from learned coefficients, with a threshold (often 0.5) determining classification into the positive or negative class(Ilyas et al., 2020).

4.2. Decision Tree for Classification

A decision tree is a versatile machine learning algorithm used for both classification and regression tasks. In classification, it segments the data into subsets based on feature values, forming a tree-like structure of decisions. Decision trees operate by recursively splitting the data based on feature values to create homogeneous subsets. The aim is to develop a model that predicts the target variable by learning simple decision rules derived from the data features(Fullerton, 2009; Gong, 2022).

The process of building a decision tree involves several key steps. The algorithm begins with the entire dataset as the root node. At each node, the impurity measure (such as Gini impurity or entropy) is calculated for each feature. The feature that provides the highest information gain or lowest impurity after the split is selected. The data is then divided into subsets based on this feature. This process is repeated recursively for each subset, creating new decision nodes and further splits until a stopping criterion is met. These criteria can include reaching a maximum tree depth, having a minimum number of samples per leaf, or achieving complete purity in the subsets (all instances in a node belong to the same class)(Aghaei et al., 2021; Hohman et al., 2020).

During prediction, a new instance is classified by passing it down the tree starting from the root node. At each decision node, the algorithm follows the branch that corresponds to the feature value of the instance. This continues until a leaf node is reached, and the class label of that leaf node is assigned to the instance(Gong, 2022).

The accuracy and efficiency of a decision tree are heavily influenced by the choice of impurity measures and the method of splitting(Aghaei et al., 2021). Gini impurity measures the likelihood of an incorrect classification of a randomly chosen element, while entropy measures the level of disorder or impurity. Information gain, which guides feature selection at each split, is calculated by comparing the entropy or Gini impurity of the dataset before and after the split (Aghaei et al., 2021; Breiman, 2001a).

To avoid overfitting, which can result from an overly complex tree, pruning techniques are employed. Pre-pruning, or early stopping, halts the tree's growth once it reaches a predefined level of complexity. Post-pruning involves removing branches from a fully grown tree to reduce complexity and enhance generalization to unseen data. These techniques ensure the model remains robust and performs well on new, unseen data (Aghaei et al., 2021; Bierman, 2001a).

4.3. Random Forests for Classification

A random forest is a powerful ensemble machine learning algorithm used for both classification and regression tasks. In classification, it constructs a multitude of decision trees during training and outputs the class that is the mode of the classes of the individual trees. Random forests operate by creating multiple decision trees from various subsets of the data, thereby reducing the risk of overfitting and improving the model’s accuracy (Breiman, 2001b).

The process of building a random forest involves several key steps. First, the algorithm begins by generating numerous bootstrap samples from the original dataset. Each bootstrap sample is used to build an individual decision tree. During the construction of each tree, a random subset of features is selected at each split, ensuring that each tree is different from the others. This randomness helps in making the model robust and less prone to overfitting. The trees are grown to their maximum depth without pruning.(Breiman, 2001b; Gong, 2022)

During prediction, a new instance is classified by passing it through all the trees in the forest. Each tree provides a class prediction, and the final output is determined by the majority vote among the trees. This ensemble approach ensures that the model benefits from the wisdom of the crowd, leading to more accurate and stable predictions(Breiman, 2001b).

The accuracy and robustness of a random forest are influenced by several factors, such as the number of trees in the forest and the number of features considered for splitting at each node. Increasing the number of trees generally improves the model's performance but also increases computational complexity. Random forests typically use the Gini impurity or entropy measures to evaluate splits, similar to individual decision trees(Breiman, 2001b; Gong, 2022).

To further enhance the model's performance, hyperparameter tuning can be employed. Key parameters to tune include the number of trees (n_estimators), the maximum depth of each tree (max_depth), the minimum number of samples required to split an internal node (min_samples_split), and the minimum number of samples required to be at a leaf node (min_samples_leaf). Proper tuning of these parameters helps in balancing the model's bias and variance, leading to optimal performance on new, unseen data(Breiman, 2001b; Gong, 2022).

Random forests are widely appreciated for their ability to handle large datasets with higher dimensionality and for their robustness against overfitting. They form the backbone of many advanced machine learning applications and serve as a benchmark for comparing the performance of other algorithms(Cichosz et al., 2016; Rani, 2020).

4.4. Consequences of Diabetes Prediction Errors: Type-I and Type-II Errors

In medical testing, understanding and managing type I and type II errors is crucial, particularly for conditions like diabetes. A type I error, also known as a false positive, occurs when a model or a test incorrectly predicts the presence of a condition that is not actually present. In the context of diabetes, this means diagnosing a patient with diabetes when they do not have it. This can lead to unnecessary stress, anxiety, and potentially harmful treatments. On the other hand, a Type II error, or false negative, happens when a model or test fails to predict the presence of a condition that is actually present. For diabetes, this means failing to diagnose a patient who actually has the disease. This can result in a lack of necessary treatment, leading to serious health complications(Fawcett, 2006; Provost et al., 2001).

The relative importance of Type I and Type II errors varies by condition. Type II errors are generally more critical for conditions where early diagnosis and treatment are crucial for preventing severe outcomes or further transmission, such as cancer, diabetes, and HIV. Missing these conditions can lead to significant health deterioration or public health risks. On the other hand, Type I errors are more critical in scenarios where diagnostic tests can cause significant anxiety, invasive follow-up procedures, and unnecessary treatments. Examples include certain cancer screenings and prenatal testing. While Type I errors can cause harm, they often do not carry the same immediate health risks as Type II errors(Fawcett, 2006; McHugh, 2009; Provost et al., 2001).

In this evaluation of model predictions, minimizing type-II error (false negatives) is prioritized, where early diagnosis and treatment are crucial to avoid serious health complications from diabetes. However, minimizing false positives is also important to avoid unnecessary stress, anxiety, and unnecessary treatments.

In the above context, recall is the most relevant metric to directly address type II errors by showing how well the model identify all positive instances. Recall measures the proportion of actual positives that are correctly identified by the model. While accuracy measures the overall correctness of the model by considering both true positives and true negatives, Precision and Specificity are the most relevant metrics for evaluating Type I errors. Precision focuses on the correctness of positive predictions and indicates how many of the positive predictions are actually correct and Specificity focuses on the correctness of negative predictions and indicates how well the model avoids false positives. The definition and description of different evaluation metrices rea given in the following section. These metrices provide a comprehensive understanding of a model's performance.

4.5. Evaluation Metrices

Below are the most commonly used classification model performance evaluation matrices(Powers, 2007).

a) **Accuracy**: The proportion of correctly classified instances out of the total instances. It is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

Where:

$TP = True\ Positive,$

$TN = True\ Negative,$

$FN = False\ Negative\ and$

$FP = False\ Positive$

a) **Precision:** The proportion of true positive predictions (correctly predicted positives) out of all positive predictions. It is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

b) **Recall (Sensitivity):** The proportion of true positive predictions out of all actual positive instances. It is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

c) **F1 Score:** The harmonic means of precision and recall, providing a single metric that balances both measures. It is calculated as:

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

d) **Specificity:** The proportion of true negative predictions (correctly predicted negatives) out of all actual negative instances. It is calculated as:

$$Recall = \frac{TN}{TN + FP}$$

e) **Receiver Operating Characteristic (ROC Curve):** A graphical representation of the trade-off between true positive rate (Sensitivity) and false positive rate (1 - Specificity) across various threshold values. AUC (Area Under the Curve) is often used to summarize the ROC curve.

f) **Confusion Matrix:** A table that summarizes the performance of a classification model, showing the counts of true positives, true negatives, false positives, and false negatives.

g) **Precision-Recall Curve:** A curve that shows the trade-off between precision and recall across various threshold values.

5. Results and Discussion

In this project, we aimed to model the risk of type 2 diabetes using three machine learning algorithms: Logistic Regression (LR), Decision Tree (DT), and Random Forest (RF). Each model was evaluated under three scenarios: the base model, feature selection with 5-fold cross-validation, and hyperparameter optimization. The performance of these model scenarios was evaluated using four key metrics: accuracy, precision, recall, and F1 score. The comparative analysis of these models under different scenarios provides insights into their predictive capabilities and the impact of feature selection and hyperparameter tuning on model performance. Table 2 shows the results of Evaluation Metrics for Three classification Algorithms in Three Different Scenarios are shown on table -2.

5.1. Comparative Analysis of Model Performance: LR, DT and RF

5.1.1. Logistic Regression Performance

The logistic regression (LR) model consistently demonstrated robust performance across all scenarios. The base model exhibited strong predictive capability, achieving an accuracy of 74.04 percent and an F1 score of 74.77. When feature selection was applied, the model maintained its performance, indicating a better balance in predicting positive cases. Hyperparameter optimization further didn't refined the model, further achieving only marginal increases in recall and F1 score, thus enhancing its overall predictive performance. This consistency across scenarios highlights the robustness of the logistic regression model in predicting type 2 diabetes risk.

5.1.2. Decision Tree Performance

The decision tree (DT) model, in contrast, showed varied performance across different scenarios. The base model performed poorly with an accuracy of 0.6368 and an F1 score of 62.44 percent, indicating limitations in its default configuration. Feature selection did not significantly improve the model's performance, with accuracy remaining around 64.12 percent. However, hyperparameter optimization led to substantial improvements, boosting accuracy to 72.55 percent and F1 score to 73.53 percent. These results suggest that while a decision tree model can be effective, it requires careful tuning of hyperparameters to achieve optimal performance.

5.1.3. Random Forest Performance

The random forest (RF) model displayed strong performance initially, with the base model achieving an accuracy of 73.85 percent and an F1 score of 75.04 percent. Interestingly, feature selection slightly decreased the model's performance, suggesting that the default feature set was

already well-suited for the task. However, hyperparameter optimization enhanced the model, achieving an accuracy of 73.66 percent and an F1 score of 74.99 percent.

5.2. Impact of Feature Selection and Cross-Validation on Model Performance

Feature selection had a varied impact on the models. For logistic regression, it resulted in marginal improvements, indicating that the initial feature set was already informative. For the decision tree and random forest models, feature selection had minimal impact, with some performance metrics even declining. This suggests that these models may inherently manage feature relevance effectively. Cross-validation, on the other hand, proved essential in providing a more robust evaluation of model performance, preventing overfitting and ensuring generalizability.

5.3. Impact of Hyperparameter Optimization on Model Performance

Hyperparameter optimization played a critical role in enhancing model performance, particularly for the decision tree and random forest models. Logistic regression showed consistent performance across all scenarios, indicating its inherent robustness and lesser dependency on hyperparameter tuning. For the decision tree and random forest models, hyperparameter optimization led to significant performance improvements, underscoring the importance of tuning to achieve optimal model configurations.

Overall, logistic regression and random forest outperformed the decision tree model in all scenarios. The random forest model, particularly with hyperparameter optimization, demonstrated the best balance between precision and recall, making it the most reliable model for predicting type 2 diabetes risk. While the decision tree model's performance improved significantly with

hyperparameter tuning, it remained less reliable compared to logistic regression and random forest. These findings highlight the importance of hyperparameter optimization and cross-validation in enhancing model performance and ensuring reliable predictions.

Table 2:valuation metrics for three classification algorithms with three Scenarios

Model	Model Scenarios	Data	Accuracy	precision	recall	F1 - score	TP	TN	FP	FN
LR	Base Model	Test	0.7304	0.7284	0.7680	0.7477	5145	4759	1918	1554
	Feature Selection, CV	Training	0.7341	0.7201	0.7643	0.7416	20412	18864	7932	6292
		Test	0.7396	0.728	0.7663	0.7467	5134	4759	1918	1565
	Parameter optimization	Training	0.7342	0.7202	0.7646	0.7417	20418	18864	7932	6286
		Test	0.7396	0.7279	0.7665	0.7467	5135	4758	1919	1564
DT	Base Model	Test	0.6368	0.6477	0.6227	0.6244	4038	4481	2196	2661
	Feature selection, CV	Training	0.6380	0.6477	0.6028	0.6244	16098	18040	8756	10606
		Test	0.6412	0.6521	0.61	0.6293	4074	4504	2173	2625
	Parameter optimization	Training	0.7431	0.7245	0.78	0.7526	20910	18846	7950	5794
		Test	0.7255	0.7100	0.7613	0.7353	5100	4605	2072	1599
RF	Base Model	Test	0.7385	0.7186	0.7853	0.7504	5261	4617	2060	1438
	Feature selection, CV	Training	0.6908	0.6809	0.7162	0.6981	19126	17835	8961	7578
		Test	0.6957	0.6873	0.7202	0.7034	4825	4482	2195	1874
	Parameter optimization	Training	0.7316	0.7093	0.7833	0.7445	20919	18225	8571	5785
		Test	0.7436	0.715	0.7884	0.7499	8282	4572	2105	1417

In conclusion, the comparative analysis underscores the significance of model selection, feature engineering, and hyperparameter tuning in predictive modeling. While logistic regression and random forest models showed robust performance, the decision tree model required extensive

tuning to achieve comparable results. For predicting type 2 diabetes risk, the random forest with hyperparameter optimization is recommended for its superior balance of precision and recall, providing the most reliable predictions among the evaluated models.

5.4. Most Predictive Features for Diabetes Risk prediction

The results of feature selection using feature importance indicate that most features in this dataset contribute to model accuracy, although the magnitude of their contribution varies significantly. For example, as shown in the figures below, bmi (body mass index) alone contributes about 26 percent to the overall performance for decision tree and random forest models.

Table 3: DT and RF - Features importances based on Gini impurity

Feature	Importance score	percent	Cumulative Contribution (%)
bmi	0.2636	26.36%	26.36%
bp	0.0902	9.02%	35.38%
age	0.0759	7.59%	42.98%
income	0.0546	5.46%	48.43%
chol	0.0489	4.89%	53.32%
education	0.0427	4.27%	57.59%
genhlth	0.0424	4.24%	61.83%
phyhlth	0.0414	4.14%	65.97%
fruits	0.0413	4.13%	70.10%
smoker	0.0402	4.02%	74.12%
menthlth	0.0380	3.80%	77.92%
sex	0.0362	3.62%	81.54%
walk	0.0321	3.21%	84.75%
veggies	0.0317	3.17%	87.92%
activity	0.0316	3.16%	91.08%
heart_disease	0.0241	2.41%	93.49%
nodocbccost	0.0188	1.88%	95.37%

Additionally, the top five features: bmi, bp, age, income, and chol(cholesterol), collectively account for approximately 53 percent of the model's performance. Feature selection based on Gini impurity for decision tree and random forest models indicated that the top three predictors of diabetes risk are BMI, BP, and age, which together contribute about 43 percent of the prediction accuracy.

For logistic regression, important features were selected based on the absolute values of the coefficients and their significance level, with p-values less than or equal to 0.05. The coefficients for bmi and cholcheck were the most significant, with absolute values greater than 1. Specifically, bmi has a coefficient of 3.44, and cholcheck has a coefficient of 1.35. The top five relevant features for logistic regression include bmi, cholcheck, alcohol consumption, general health (genhlth), and high blood pressure(bp).

6. Conclusion and Policy Implications

The comparative analysis of the logistic regression (LR), decision tree (DT), and random forest (RF) models provide valuable insights into their performance in predicting the risk of type 2 diabetes. The key findings and policy implications are the following:

The random forest model, particularly with hyperparameter optimization, demonstrated the best balance between precision and recall, making it the most reliable model for predicting type 2 diabetes risk. Although logistic regression showed robust performance, the random forest's superior

predictive capabilities make it the preferred choice. The decision tree model, while improved with hyperparameter tuning, remained less reliable compared to logistic regression and random forest.

Feature selection had a varied impact on the models, with logistic regression showing marginal improvements and decision tree and random forest models showing minimal impact in performance. Cross-validation proved essential in providing a more robust evaluation of model performance, preventing overfitting and ensuring generalizability. Implementing rigorous cross-validation procedures should be a standard practice in health predictive modeling to ensure that models generalize well to new data, thereby increasing their reliability and effectiveness in real-world applications.

Hyperparameter optimization played a critical role in enhancing model performance, particularly for the decision tree and random forest models. Logistic regression showed consistent performance across all scenarios, indicating its inherent robustness and lesser dependency on hyperparameter tuning. Allocating resources towards the optimization of machine learning models, including hyperparameter tuning, can significantly improve the accuracy and reliability of predictive analytics in healthcare. This is especially important for complex models like random forests and decision trees.

The analysis identified bmi (body mass index), bp(blood pressure), age, and cholesterol levels as the most predictive features across all models. For logistic regression, additional important features included alcohol consumption and general health status. The identification of key predictors such as bmi, bp, age, and cholesterol levels underscores the importance of targeted public health interventions. Policies aimed at reducing obesity and managing blood pressure through lifestyle changes, dietary regulations, and routine health check-ups could significantly mitigate the risk of type 2 diabetes.

Public health campaigns focusing on regular cholesterol checks and promoting healthy lifestyle choices, such as reduced alcohol consumption and improved general health awareness, can enhance the effectiveness of diabetes prevention strategies. These targeted interventions can lead to more efficient allocation of resources and improved health outcomes in the population.

The comparative analysis underscores the significance of model selection, feature engineering, and hyperparameter tuning in predictive modeling for health outcomes. The random forest model, with its superior balance of precision and recall, is recommended for predicting type 2 diabetes risk. Policymakers and healthcare providers should focus on implementing targeted interventions based on the most significant predictors identified, to effectively mitigate the risk of type 2 diabetes and improve public health outcomes.

References

Aghaei, S., Gómez, A., & Vayanos, P. (2021). *Strong Optimal Classification Trees*. http://arxiv.org/abs/2103.15965

BOBBITT, Z. (2022, August 8). *Label Encoding vs. One Hot Encoding: What's the Difference?* Statology.

Breiman, L. (2001a). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324/METRICS

Breiman, L. (2001b). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brownlee, J. (2020, August 28). *How to Use StandardScaler and MinMaxScaler Transforms in Python*. Https://Machinelearningmastery.Com/Standardscaler-and-Minmaxscaler-Transforms-in-Python/.

Cichosz, S. L., Johansen, M. D., & Hejlesen, O. (2016). Toward Big Data Analytics: Review of Predictive Models in Management of Diabetes and Its Complications. In *Journal of Diabetes Science and Technology* (Vol. 10, Issue 1, pp. 27–34). SAGE Publications Inc. https://doi.org/10.1177/1932296815611680

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/J.PATREC.2005.10.010

Fullerton, A. S. (2009). A Conceptual Framework for Ordered Logistic Regression Models. *Http://Dx.Doi.Org/10.1177/0049124109346162*, *38*(2), 306–347. https://doi.org/10.1177/0049124109346162

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. Concepts, Tools, and Techniques to Build Intelligent Systems* (N. Tache, Ed.; 2nd ed.). O'Reilly Media, Inc.

Giuseppe, C., Ayyadevara, V. K., & Perrier, A. (n.d.). *Hands-On Machine Learning on Google Cloud Platform. https://www.oreilly.com/library/view/hands-on-machine-learning/9781788393485/fd5b8a44-e9d3-4c19-bebb-c2fa5a5ebfee.xhtml*. O'reilly.

Gong, D. (2022). Top 6 Machine Learning Algorithms for Classification. *Towards Data Science*.

Hohman, F., Wongsuphasawat, K., Kery, M. B., & Patel, K. (2020). Understanding and Visualizing Data Iteration in Machine Learning. *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–13. https://doi.org/10.1145/3313831.3376177

Ilyas, A., Zampetakis, M., & Daskalakis, C. (2020). *A Theoretical and Practical Framework for Regression and Classification from Truncated Samples*.

Izenman, A. J. (2008). *Modern Multivariate Statistical Techniques*. https://doi.org/10.1007/978-0-387-78189-1

Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection*. Chapman and Hall/CRC. https://doi.org/10.1201/9781315108230

McHugh, M. (2009). The odds ratio: calculation, usage, and interpretation. *Biochemia Medica*, 120–126. https://doi.org/10.11613/BM.2009.011

Min–max normalization - Hands-On Machine Learning on Google Cloud Platform [Book]. (n.d.). Retrieved July 3, 2024, from https://www.oreilly.com/library/view/hands-on-machine-learning/9781788393485/fd5b8a44-e9d3-4c19-bebb-c2fa5a5ebfee.xhtml

(PDF) Normalization and Standardization: Methods to preprocess data to have consistent scales and distributions. (n.d.). Retrieved July 3, 2024, from https://www.researchgate.net/publication/377123133_Normalization_and_Standardization_Methods_to_preprocess_data_to_have_consistent_scales_and_distributions

Powers, D. M. W. (2007). *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation* .

Provost, F., Org, P., & Fawcett, T. (2001). *Robust Classification for Imprecise Environments*. *42*, 203–231.

Rani, K. J. (2020). Diabetes Prediction Using Machine Learning. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 294–305. https://doi.org/10.32628/CSEIT206463

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning Machine Learning and Deep Learning with Python, Scikit-learn, and TensorFlow 2*. Packt Publishing.

Roy, B. (2019). All about Categorical Variable Encoding. *Medium*.

Sharma, R. (2022, November 8). *What's Wrong with Your Statistical Model? Skewed Data.* Https://Builtin.Com/Data-Science/Skewed-Data.

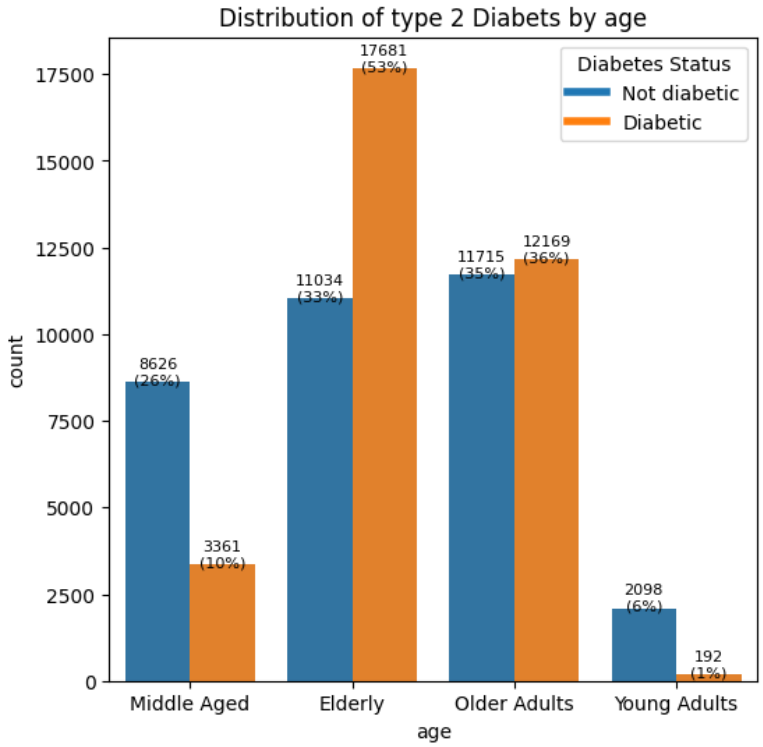
Sohil, F., Sohali, M. U., & Shabbir, J. (2022). An introduction to statistical learning with applications in R. *Statistical Theory and Related Fields*, *6*(1), 87–87. https://doi.org/10.1080/24754269.2021.1980261

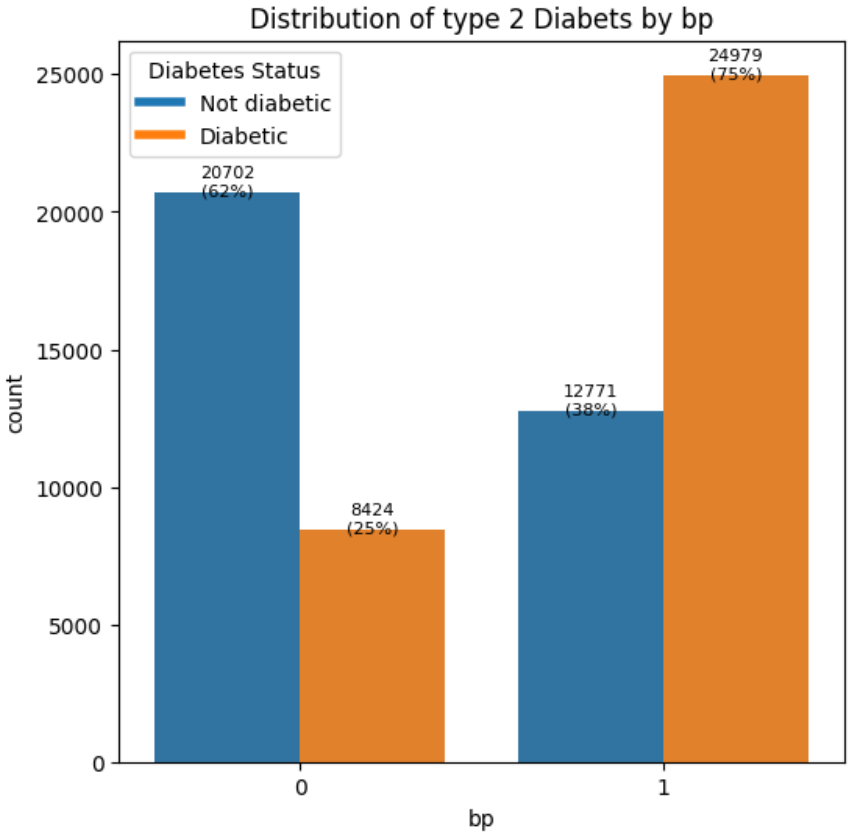
Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, *45*(4), 427–437. https://doi.org/10.1016/J.IPM.2009.03.002

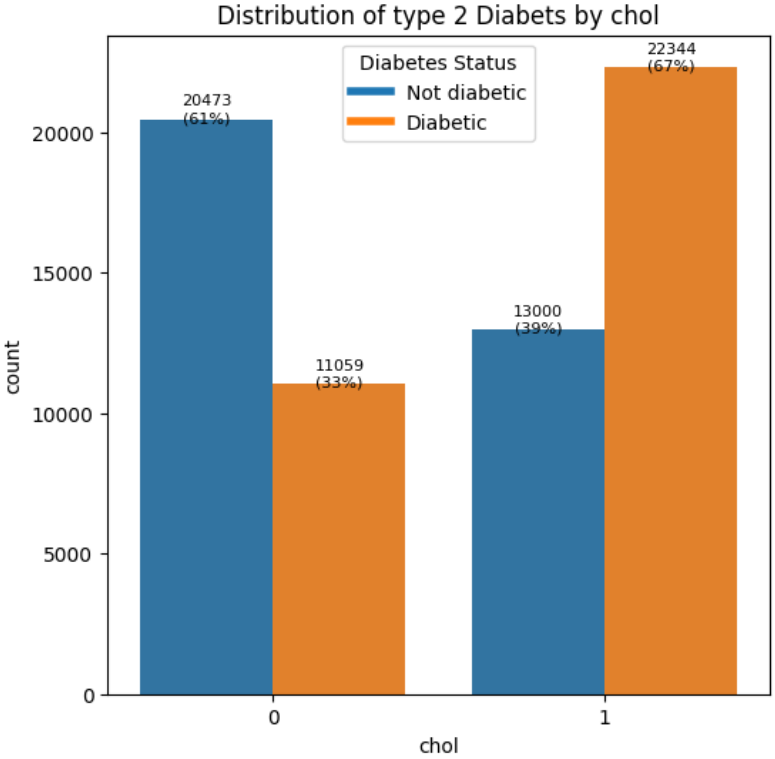
Train, K. E. (2001). *Discrete Choice Methods with Simulation*. Cambridge University Press. https://doi.org/10.1017/CBO9780511805271

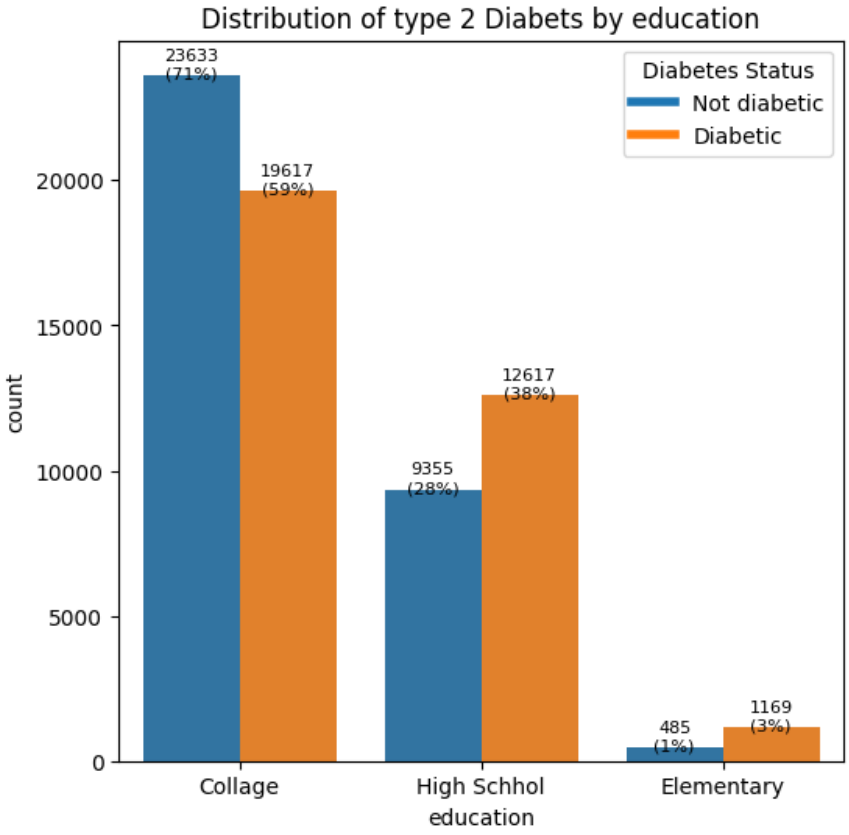
Appendix:

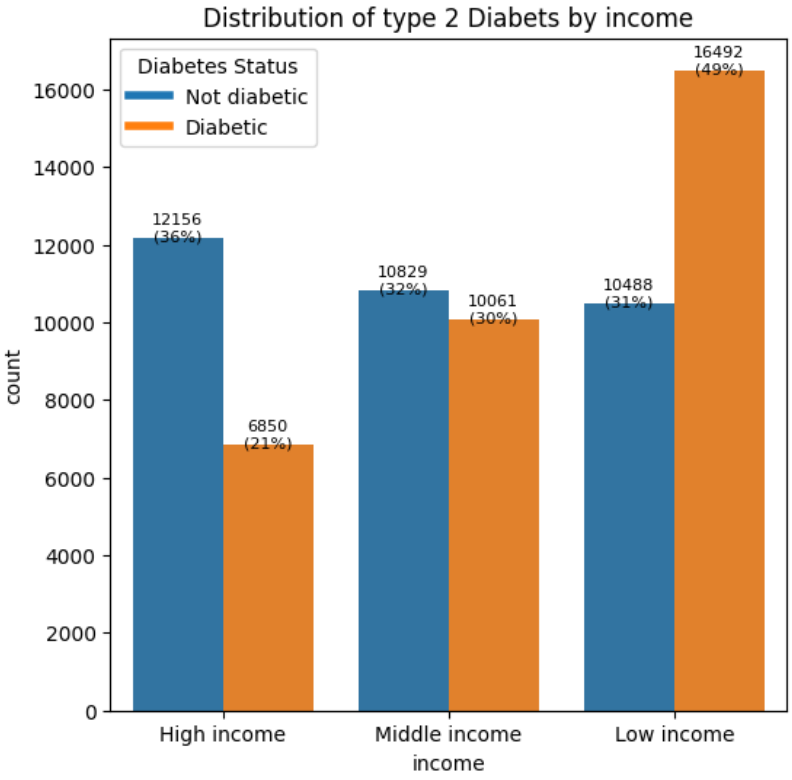
Appendix 1-Distribution of diabetes by some selected features: age, high blood pressure, cholesterol, education, and income level




Distribution of type 2 Diabets by bp
Diabetes Status
Not diabetic
Diabetic
20702
(62%)
8424
(25%)
12771
(38%)
24979
(75%)
count
bp
0
1
0
5000
10000
15000
20000
25000


Distribution of type 2 Diabets by chol
Diabetes Status
Not diabetic
Diabetic
20473
(61%)
11059
(33%)
13000
(39%)
22344
(67%)
count
chol
0
1
0
5000
10000
15000
20000


Distribution of type 2 Diabets by education
Diabetes Status
Not diabetic
Diabetic
23633
(71%)
19617
(59%)
12617
(38%)
9355
(28%)
485
(1%)
1169
(3%)
count
0
5000
10000
15000
20000
Collage
High Schhol
Elementary
education


Distribution of type 2 Diabets by income
Diabetes Status
Not diabetic
Diabetic
12156
(36%)
6850
(21%)
10829
(32%)
10061
(30%)
10488
(31%)
16492
(49%)
count
0
2000
4000
6000
8000
10000
12000
14000
16000
High income
Middle income
Low income
income