PREDICTING DIABETES WITH MACHINE LEARNING

TEAM NAME: THE THREE DATAMUSKETEERS

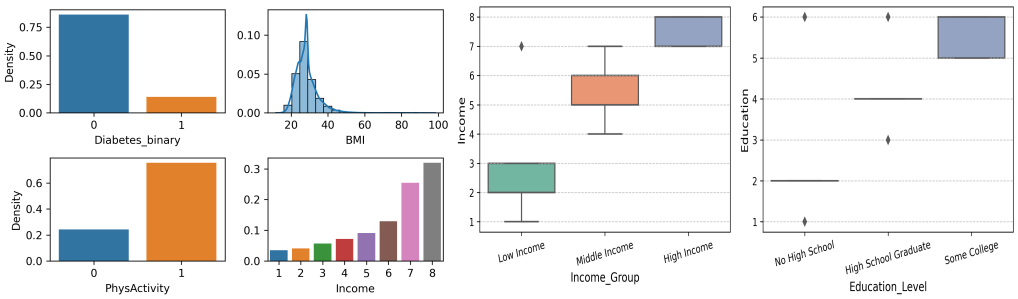
Khaled Rouissi (2103131)
khaled.rouissi@polymtl.ca

Adam Nhaila (2413574)
adam.nhaila@polymtl.ca

Adrien Telitchko (2413607)
adrien.telitchko@polymtl.ca

1 FEATURE DESIGN

Exploratory Data Analysis (EDA) and Data Preprocessing. The competition dataset consists of **202,944 training samples** and **28 features**, including demographics, lab test results, and responses to survey questions. The target variable, ***Diabetes_binary***, is a binary variable that specifies whether a patient has diabetes (Class 1) or not (Class 0). The dataset is divided into two primary subsets: **(1) Mega Training Set:** Comprising 85% of the data. This subset is further partitioned into an 85-15 split to create training and validation sets for model assessment. **(2) Internal Test Set:** Representing 15% of the data, this subset is exclusively reserved for the final evaluation of the selected model. In summary, the final percentages with respect to the original dataset size are: (1) **Training Set:** 72.25%, (2) **Validation Set:** 12.75 %, and (3) **Test Set:** 15 %. These dataset splits ensure a better allocation of data for training, validation, and testing while maintaining the quality and integrity of the Internal Test Set as a reliable proxy for the competition test set. Furthermore, *the full data is shuffled and stratified before each split to avoid biases*. Also, a fixed random state seed of 42 is consistently applied throughout the notebook to ensure reproducibility. We provide a statistical summary of the 28 features alongside the target variable. Particularly, we include the mean, the standard deviation, the minimum, and the maximum values for each variable. This information is provided in Table 4 of the Appendix. When performing data inspection for cleaning, we found that there are **no missing data and no duplicated samples**. Therefore, there is no need to handle missing values and duplicated samples.



(a) Target and Feature Distributions

(b) Outlier Detection in Feature Pairs

Figure 1: **Normalized Feature Distributions and Outlier Detection in the Training Set.** There is a clear imbalance in the target features. Some features are skewed. These observations highlight the need for advanced techniques to address imbalanced data and effectively normalize or transform skewed variables for improved analysis.

We show a snapshot of the analyses in 1a and 1b, respectively, related to target/feature distributions and outlier detection in feature pairs. We keep the full details (distirbution of the 28 features alongside the target variable in Figure 2 and other samples of out-of-distribution in Figure 3 in the Appendix. We observe that the class distribution is **imbalanced** and the positive class (*Diabetes_binary* = 1) constitutes only **13.97% of the Training Set samples**. The BMI shows a skewed distribution with most values concentrated in a specific range and a tail extending toward higher values indicating that individuals with extreme BMI are less common. Similarly, the majority of the population seems to be imbalanced too and physically active (value=1). As for the income, this categorical feature have most values concentrated in higher-income groups. With respect to the outlier analyses, we observe that there are some outliers in low-income group. The education levels show a wide distribution of values, with noticeable outliers. We also investigated the correlations between all features. We provide all of the details the subsection 7.4 in the Appendix and particularly we show the correlation matrix in Figure 4. We observed that the features most highly correlated with the Diabetes_binary target Heart_Disease_Risk include GenHlth, HighBP, DiffWalk, and BMI.

These represent the strongest associations with the target variable. One of the global observations from these analyses is that we have a lot of imbalanced distributions, which may necessitate balancing during training. Also, some variables interactions can help and support advances feature engineering.

Feature Encoding One-hot encoding can be used to encode the four categorical features. However, since it can be observed that all these features have a clear order (e.g., for the *Income_Group* feature, it can be assumed that Low income $<$ Medium income $<$ High income), an **ordinal encoder**, which preserves this natural order, is utilized. The order of the unique values for each categorical feature was manually specified in our code to ensure that the assigned numerical values accurately reflect the intended order. This choice of ordinal encoder helps preserving the number of columns in the transformed training set. In contrast, using one-hot encoding for *BMI_Category* would result in the creation of 4 new sparse features. Hence, a total of 7 additional columns (11 new- 4 existing columns) would be added to the dataset for all four categorical features. The fitted encoding models on the train set are stored in a dictionary for consistent transformation of the validation and test sets.

Feature Scaling and Normalization

After encoding categorical features, the Train Set was scaled and normalized to ensure consistent feature ranges, aiding model convergence and reducing bias toward features with larger values.
(1) No Scaling Needed: 17 features, including binary variables and the encoded *Age_Group* (value 0), required no scaling: *HighBP, HighChol, CholCheck, Smoker, Stroke, HeartDiseaseorAttack, PhysActivity, Fruits, Veggies, HvyAlcoholConsump, AnyHealthcare, NoDocbcCost, DiffWalk, Sex, Healthy_Diet, Mental_Health_Risk, and Age_Group*.
(2) Scaling Applied: The remaining 11 features (*BMI, GenHlth, MentHlth, PhysHlth, Age, Education, Income, BMI_Category, Heart_Disease_Risk, Education_Level, Income_Group*) were scaled using MinMaxScaler for simplicity, although various scalers (e.g., StandardScaler, RobustScaler, QuantileTransformer) were tested to address skewed distributions and enhance model performance. Results comparing scaler impacts are detailed in the results section.

Feature Engineering Newly engineered features could help the models learn more complex patterns in the training set. It is important to note that the provided dataset already includes some engineered features, and we have been able to understand the equations used to create them. Some of these features are as follows:
Healthy_Diet = *Fruits x Veggies*
Heart_Disease_Risk = *HighBP + HighChol + Smoker + Stroke + HeartDiseaseorAttack*
BMI_Category: The four-class classification of the Body Mass Index.
To assess the impact of feature engineering, we added interaction terms and squared features using Python's polynomial function with degree two, increasing the scaled training set to 434 features. However, this did not improve the F1 score. Consequently, we limited new features to interactions among *BMI*, *Age*, *GenHlth*, and *Heart_Disease_Risk*, adding six new features.

Feature Selection To evaluate feature importance, we used Recursive Feature Elimination (RFE), classifier-based methods (Decision Tree, Random Forest, Logistic Regression), and correlation matrix analysis. No features showed extremely high correlation (r$>$0.9) as shown in Figure 4 and feature importance varied by classifier as seen in Figure 5 . Given the small feature set, we dropped only the *Age_group* feature to preserve valuable information.

Data Balancing and Augmentation Strategies Considering that the training set is imbalanced, with the positive class representing only 13.97% of the samples, one of the major challenges in this competition has been balancing the dataset to improve model performance, particularly with respect to the F1 score. To address this, multiple data balancing techniques and augmentation strategies were tested. Below is a description of some of these strategies.

Oversampling Techniques:
(a) *Random Oversampling*: consists of randomly selecting samples from the minority class until matching the number of samples in the majority class.
(b) *SMOTE (Synthetic Minority Over-sampling Technique)*: consists of creating new synthetic samples for the minority class by interpolating between the existing samples and their nearest neighbors.
(c) *Borderline-SMOTE*: is similar to SMOTE but the new generated samples will be sampled from neighbors near to the decision boundary, to better handle difficult-to-classify samples.
(d) *ADASYN (Adaptive Synthetic Sampling)*: consists of adaptively generate new synthetic samples

for the minority class focusing on the difficult-to-classify samples.

Undersampling Techniques:
(a) *Random Undersampling*: consists of randomly removing samples from the majority class until matching the number of samples in the minority class.
(b) *RepeatedEditedNearestNeighbours (RENN)*: consists of removing noisy samples from the majority class, which are misclassified by their nearest neighbors.

Training Dataset Extension: In addition to creating new synthetic data, we considered that collecting new real samples, particularly from the positive class, could be beneficial for model training. To achieve this, we searched for public diabetes datasets that closely align with the features of our training set. Multiple scientific articles Xie et al. (2019); Petersen & Association (2008) and datasets, such as the Pima Indians Diabetes Database Smith & Jones (1988), the Diabetes 130-US Hospitals for Years 1999-2008 Clore & Strack (2014), and several Behavioral Risk Factor Surveillance System (BRFSS) Annual Data Centers for Disease Control and Prevention (2024) from the Centers for Disease Control and Prevention, were analyzed. After reviewing these datasets, we determined that the BRFSS datasets were the near to our training features and could be merged to introduce new samples. It is worth noting that we used and modified a publicly available Kaggle notebook Teboul (2019) designed for cleaning BRFSS yearly data. We cleaned and merged the 2013 and 2015 datasets, then imported and combined them with the training set. All of the above techniques were tested, and the comparison results are presented in the Results section.

2 ALGORITHMS

We explored a diverse set of learning algorithms to address the diabetes prediction problem. Below is a selection of the algorithms we have tried.

2.1 INDIVIDUAL ALGORITHMS

- **Logistic Regression:** is a binary classifier that uses a linear decision boundary to classify data. The model computes a linear combination of input features, and then applies the sigmoid function to perform binary classification and scale the output to the [0,1] interval. This allows interpreting the output as a probability.
- **K-Nearest Neighbors (KNN):** is a model-free algorithm. To classify a new data point, it identifies its k closest points in the training set using a chosen distance metric. It then assigns the new data point to the most common class among these neighbors.
- **Decision Tree:** operates as a tree-like model. The process starts with the entire dataset at the root. At each node, it selects the best feature and threshold to split the data, minimizing impurity. This repeats for subsets, forming branches. To classify, the model traverses the tree until it reaches a leaf node, which provides the predicted class.
- **Multi-Layer Perceptron:** is an neural network algorithm consisting of an input layer, one or more hidden layers and an output layer. Each layer is fully connected,and the model is trained using backpropagation to adjust weights.

2.2 ENSEMBLE ALGORITHMS

Ensemble algorithms combine multiple machine learning models to mitigate their individual weaknesses, such as overfitting, thereby enhancing performance on unseen datasets:

- **Bagging Algorithms:** Bagging consists of training multiple models on different subsets of the training data and then aggregating their predictions. These subsets are created by bootstrapping, which involves random sampling with replacement.
  - **Random Forest:** consists of combining multiple decisions trees. Each tree is trained on a bootstrap sample of the training dataset. Each tree independently gives a prediction and the predictions are aggregated by majority voting.
  - **Balanced Random Forest:** is an extension of Random Forest that addresses class imbalance issues. Instead of drawing samples proportionally to the original class distribution, it performs under-sampling of the majority class or oversampling of the minority class to create balanced training subsets for each tree.

  - **Extremely Randomized Trees (Extra Trees):** further reduces variance and prevents overfitting by introducing additional randomness into the tree-building process. Instead of selecting the best possible split from a random subset of features, Extra Trees randomly select both the feature and the split threshold for each node.
- **Boosting Algorithms:** Boosting builds models sequentially. The first model predicts on the entire dataset, and misclassified points get increased weights. Subsequent models focus on these harder cases. After training all models, their predictions are combined.
  - **AdaBoost:** was the first successful boosting technique. It is designed for binary classification and relies solely on decision trees.
  - **XGBoost:** enhances boosting by supporting parallel processing and adding regularization. It also includes a tree learning algorithm that handles missing values internally.
  - **LightGBM:** grows trees leaf-wise. It splits the leaf with the highest potential to reduce loss, resulting in deeper trees that capture more complex patterns.
  - **CatBoost:** It handles categorical features directly. It uses ordered target encoding instead of preprocessing methods like one-hot encoding. It also uses symmetric trees.
- **Voting:** is an ensemble technique that aggregates the predictions of multiple independent models to give a prediction. It can be performed using hard, soft or weighted soft voting.
- **Stacking:** is an ensemble technique that uses meta-learner algorithm, usually logistic regression, to combine the predictions from the base learners.

3 METHODOLOGY

Our methodology encompassed several key decisions to ensure finding the most robust model that can maximize the F1-score.

- **Training/Validation/Test Splits:** As explained in Section 1, we split the training set into an 85/15% Mega Training/Test split, followed by an 85/15% Training/Validation split on the Mega Training set. The data was shuffled and stratified to ensure robust testing. This division ensures that we have enough testing milestones to validate the performance of the model. Especially, validation results closely matched Kaggle submissions, providing insight into model performance without requiring multiple submissions.
- **Data Encoding, Normalization, and Class Balancing:** Encoding, normalization, and balancing using various methods were performed to minimize bias towards the majority class or higher values.
- **Validation:** We used the Validation Set to evaluate the models performance and performed k-fold cross-validation when applicable to assess model stability and reduce overfitting.
- **Regularization Strategy:**The best and most suitable regularization coefficient and type were determined through multiple rounds of Optuna optimization (a Bayesian optimization framework) for each algorithm. This process helped apply the necessary regularization to prevent overfitting without excessively penalizing the model.
- **Optimization Tricks:** Various types of scalers and sampling techniques were tested and compared for each model. Additionally, boosting with early stopping was implemented when applicable. Ensemble techniques and optimizing the decision boundary of the predicted probabilities helped to improve the F1-score.
- **Hyperparameter Settings:** A comprehensive list of key hyperparameters, such as learning rate, number of trees, and depth for ensemble models, was optimized using a hyperparameter search and Optuna optimization to identify the optimal values.

4 RESULTS

Our models were evaluated mainly based on F1-score, with accuracy, precision, and recall also considered to understand the strengths and weaknesses of each algorithm. Table 1 summarizes the impact of dataset balancing techniques and augmentation strategies. Multiple classification models were trained using their default parameters on eight different training sets generated through the resampling techniques described in Section 1. The results in Table 1 clearly show that the original

Table 1: **Impact of Various Sampling Techniques on Validation F1 Score Across Different Classification Models.** RENN and oversampling with additional data emerged as the most two promising techniques, achieving the highest validation F1 scores across the classifiers.

Model	Original	Over	Under	SMOTE	Borderline	ADASYN	Add_data	RENN
Logistic Regression	23.48	43.56	43.82	43.82	43.95	43.00	43.50	**45.48**
Decision Tree	29.75	28.44	33.98	30.68	31.22	30.36	**49.41**	39.23
Random Forest	24.80	34.43	42.52	37.72	37.59	36.81	**73.90**	45.20
Balanced RF	43.17	34.83	42.59	37.58	37.33	37.17	**73.67**	41.62
Extra Trees	26.90	23.52	41.63	38.59	37.94	37.14	**83.81**	43.56
LGBM	24.29	44.08	43.61	36.30	36.23	34.39	43.86	**45.82**
XGBoost	25.57	43.91	42.68	34.47	35.27	34.17	44.37	**45.43**
CatBoost	25.81	44.14	43.62	33.77	33.41	32.84	44.87	**45.59**
AdaBoost	25.10	43.88	44.02	44.31	44.46	43.50	43.50	**45.70**
KNN	26.4	36.34	39.15	36.96	38.28	36.63	**60.75**	40.84
MLP	27.59	42.20	43.73	41.93	42.33	40.76	**44.17**	43.94

imbalanced training dataset significantly hindered the performance of all tested classifiers. Balancing the distribution between the positive and negative classes substantially improved model performance. Among the evaluated techniques, RENN and oversampling with additional data emerged as the most two promising techniques, achieving the highest validation F1 scores across the classifiers. Overall, tree-based algorithms performed well with the resampled datasets, while boost-based algorithms exhibited slightly better performance with the RENN sampling technique. Notably, the Balanced Random Forest algorithm outperformed all others when using the original imbalanced training set, likely due to its default mechanism of correcting class weights. Additionally, the performance of the K-Nearest Neighbors (KNN) algorithm was highly dependent on the training dataset. Its F1 score ranged from 26.4% on the imbalanced dataset to 40.8% and 60.7% when using RENN and additional data sampling techniques, respectively.

Table 2: **Impact of Various Scaling Techniques on Validation F1 Score Across Different Classification Models.** StandardScaler provided the best performance for the majority of classifiers.

Model	Standard Scaler	MinMax Scaler	Robust Scaler	MaxAbs Scaler	Quantile Transformer	Power Transformer
Logistic Regression	43.73	**43.79**	43.72	43.79	43.30	43.65
Decision Tree	**34.62**	33.98	34.54	33.72	34.41	33.89
Random Forest (RF)	**42.74**	42.48	42.53	42.61	42.64	42.69
Balanced RF	**42.74**	42.69	42.44	42.62	42.67	42.68
Extra Trees	41.55	41.52	41.83	41.18	41.65	**41.85**
KNN	39.37	39.15	**39.94**	39.24	38.58	38.86
LGBM	43.63	43.61	43.51	43.54	**43.78**	43.70
XGBoost	42.71	42.68	**42.88**	42.78	42.61	42.64
CatBoost	43.94	43.61	**43.97**	43.84	43.67	43.67
AdaBoost	43.51	**44.02**	43.58	43.74	43.64	43.62
Multi-layer Perceptron	42.32	43.19	42.33	**43.23**	41.73	41.51

Table 2 summarizes the impact of different scaling techniques on the F1 score of various models. All models were tested using the same initial training set, with the undersampling technique and their default parameters. It was observed that the StandardScaler provided the best performance for the majority of classifiers. Specific scaling and normalization techniques can be used to slightly boost the performance of the models.

Table 3 summarizes the final results of 17 algorithms tested on the Test Set. It presents nine algorithms with their best hyperparameters, seven majority voting ensembles, and one stacking ensemble. The table shows that Extra Trees outperformed all the algorithms with an F1 score of 83.9%, followed by a soft weighted ensemble of Random Forest and Extra Trees with an F1 score of 81.6%. XGBoost had the lowest F1 score, followed by AdaBoost and Logistic Regression. It is worth mentioning that the results on our internal test set were almost identical to those on the validation set, with a difference of less than 1%. Moreover, the results closely matched the competition scores. In fact, the final submission using the Extra Trees model achieved a public set score of 84% and a private set score of 83%, compared to our internal test score of 83.9%.

Table 3: **Test Metrics for Different Models.** Extra Trees outperformed overall followed by a soft weighted ensemble of Random Forest and Extra Trees.

	Test Accuracy (%)	Test F1 Score (%)
Logistic Regression	78.48	45.80
Decision Tree	87.04	59.93
Random Forest	93.39	72.49
Extra Trees	95.96	**83.91**
XGBoost	80.27	45.55
LGBM	79.32	46.19
AdaBoost	78.24	45.71
CatBoost	78.49	45.90
MLP	78.61	45.82
Ens.(Hard, LGBM, Decision Tree, and Logistic Regression)	74.48	46.02
Ens.(Soft, LGBM, Decision Tree, and Logistic Regression)	84.51	56.99
Ens.(Soft weighted, LGBM, Decision Tree, and Logistic Regression)	87.11	60.13
Ens.(Soft, Extra Trees, Decision Tree, Logistic Regression, MLP, and LGBM)	88.06	64.69
Ens.(Hard, Extra Trees, Decision Tree, Logistic Regression, MLP, and LGBM)	74.96	46.94
Ens.(Soft weighted, Extra Trees, Decision Tree, Logistic Regression, MLP, and LGBM)	93.03	75.09
Ens.(soft weighted, Extra Trees and Random Forest)	95.46	81.57
Ens. Stacking w/ Logistic Regression on Random forest and Decision Tree	94.28	74.75

5 DISCUSSION

During this competition, more than ten classification models were tested and tuned to perform a classification task of diabetes prediction. The dataset imbalance was one of the major causes that significantly hindered the performance of the models and made it more bias toward the majority negative class. Several sampling and data augmentation techniques were tested and they improved the performance of all the models. In fact, balancing the data and either undersampling by removing the noisy sample using RENN technique or adding new real positive samples boosted the performance of the models by at least 50%.

Moreover, finding the most suitable hyperparameters for each model helped optimize the F1 score. The initial division of the training set into training, validation, and test sets enabled us to predict the models' performance without the need for multiple Kaggle submissions. Furthermore, optimizing and slightly adjusting the decision boundary between the positive and negative classes in most models resulted in an additional 1-2% performance improvement. It was observed that the Multilayer Perceptron's performance was relatively good, but it was unable to surpass the performance of tree-based and boosting algorithms. It should also be noted that we tested a tabular transformer and dimensionality reduction algorithms, but their performances did not surpass those of the methods presented. The tabular transformer was not included, as it was stated that pre-trained models could not be used in this competition.

In conclusion, it is worth to mention that the limitations of computational resources and the unsuitability of certain algorithms for GPU usage constrained our ability to fully explore and enhance specific models and ideas. Additionally, given the significant performance gap between our model and those of other teams, there is a possibility that the additional sampling and data we used for certain algorithms are some sort of data leakage of the competition test dataset. In fact, data leakage and model performance issues tie into the broader debate on intelligence, especially with the rise of GPT models. While these models are highly useful, many responses merely reproduce internet information, and despite strong benchmark performances, they struggle with simple human tasks like the ARC challenge.

6 STATEMENT OF CONTRIBUTIONS

Khaled Rouissi: Led many parts of the project, developed pre-processing pipelines, conducted hyperparameter tuning, contributed to model evaluation metrics, and wrote the majority of the report.

Adam Nhaila: Managed algorithm selection and implementation, conducted hyperparameter tuning, and coordinated the ensemble modeling strategy, and authored part of the report.

Adrien Telitchko: Handled data visualization, performed a comparative analysis of model performances, conducted hyperparameter tuning, and authored part of the report.

We hereby state that all the work presented in this report is that of the authors.

REFERENCES

Centers for Disease Control and Prevention. Behavioral risk factor surveillance system (brfss) annual data, 2024. URL https://www.cdc.gov/brfss/annual_data/annual_data.htm. Accessed: 2024-12-03.

Cios Krzysztof DeShazo Jon Clore, John and Beata Strack. Diabetes 130-US Hospitals for Years 1999-2008. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5230J.

Matt Petersen and American Diabetes Association. Economic costs of diabetes in the u.s. in 2007. *Diabetes Care*, 31(3):596–615, 2008. doi: 10.2337/dc08-9017. URL https://doi.org/10.2337/dc08-9017.

John Smith and Michael Jones. Pima indians diabetes database, 1988. Available at https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database.

Alex Teboul. Diabetes health indicators dataset notebook, 2019. URL https://www.kaggle.com/code/alexteboul/diabetes-health-indicators-dataset-notebook. Accessed: 2024-12-03.

Zidian Xie, Olga Nikolayeva, Jiebo Luo, and Dongmei Li. Building risk prediction models for type 2 diabetes using machine learning techniques. *Preventing Chronic Disease*, 16:190109, 2019. doi: 10.5888/pcd16.190109. URL http://dx.doi.org/10.5888/pcd16.190109.

7 APPENDIX

7.1 REMARKS

Please note that we have used ChatGPT to edit the grammar in this report and to edit the grammar of the descriptive text (i.e. docstring) within the functions of our code.

7.2 NORMALIZED DISTRIBUTIONS

Figure 2 shows that the Train Set features are a mix of binary, categorical, and continuous variables. Notably, the *Age_Group* feature has only one unique value and should be removed, as it will not add any useful information. Furthermore, 16 features, such as *HighBP, Smoker* and *Sex* are binary with two unique values. Seven features such as *Income, BMI_Category* and *Education* have between 3 and 8 unique values, while 4 features (BMI, MentHlth, PhysHlth, and Age) are continuous variables. Here are some more details with respect to the data analysis:

- HighBP, HighChol, CholCheck features show relatively balanced distributions, except for CholCheck, where almost all individuals fall into one category.
- BMI: The distribution is right-skewed, with most individuals having a BMI in the lower range but some outliers extending to higher values.
- Smoker, Stroke: Smoking prevalence is roughly balanced, while strokes are much less common in the population.
- HeartDiseaseorAttack: The majority of individuals do not report this condition.
- PhysActivity, Fruits, Veggies: High levels of physical activity and frequent fruit and vegetable consumption are observed, though there's some variation.
- HvyAlcoholConsum, AnyHealthcare: Heavy alcohol consumption and healthcare access have clear imbalances, with fewer individuals in the "1" category.
- NoDocbcCost: Very few individuals report avoiding doctor visits due to cost.
- GenHlth: General health is distributed across categories, with higher levels being more common.
- MentHlth, PhysHlth: These continuous variables show skewed distributions with the majority reporting fewer issues.
- DiffWalk: A minority reports difficulty walking.

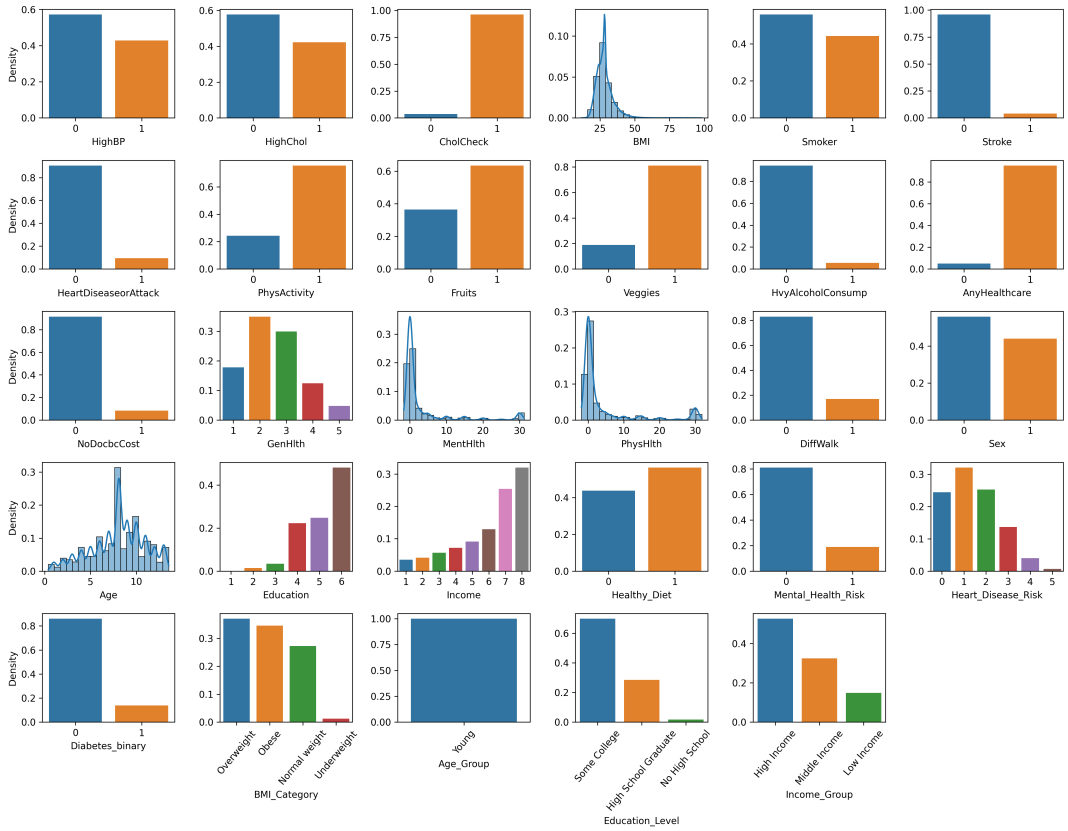


Figure 2: Normalized Distributions of the Target Variable and All Features in the Training Set

- Age: The distribution shows a concentration around specific age ranges, with a gradual tapering.
- Education: More individuals fall into higher educational levels.
- Income: Most income categories are higher, consistent with a well-off population distribution.
- Diabetes_binary: Diabetes is imbalanced, with fewer individuals in the "1" category.
- BMI_Category: Overweight and obese categories dominate, with fewer individuals in the underweight or normal categories.
- Age_Group, Income_Group: Younger and middle-income individuals dominate these categories.

7.3 Outlier Detection

Figure 3 provides an analysis of outlier detection across different features. We observe a significant number of outliers in the BMI feature and the Obese category. Similarly, a notable presence of outliers is also evident in the Education and Education_level features.

7.4 Correlation Analysis

Figure 4 displays the correlations between all features. We observed that the features most highly correlated with the Diabetes_binary target Heart_Disease_Risk include GenHlth, HighBP, DiffWalk, and BMI. These represent the strongest associations with the target variable. One of the global observations from these analyses is that we have a lot of imbalanced distributions, which may necessitate balancing during training. Also, some variables interactions can help and support advances feature engineering. Additionally, we observe that there are some features that are highly correlated

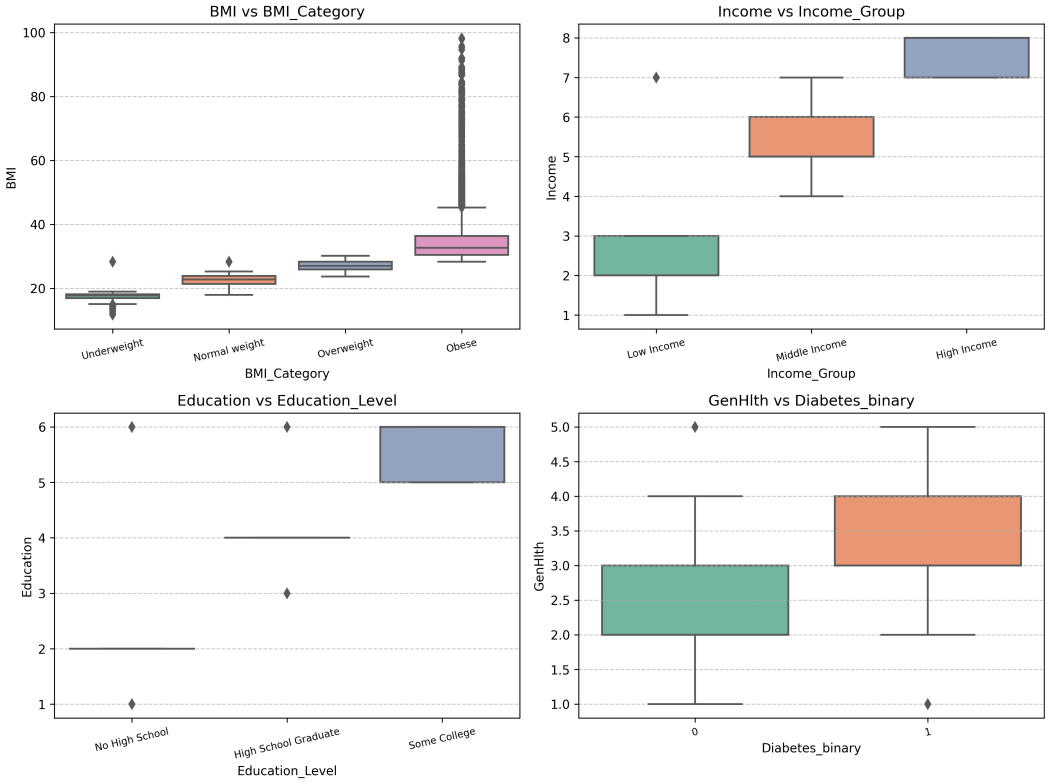


Figure 3: Outlier Detection on Feature Pairs.

with each other, such as veggies and fruits with respect to Healthy_Diet. These correlations can help us in determining new engineered features to boost models' performance.

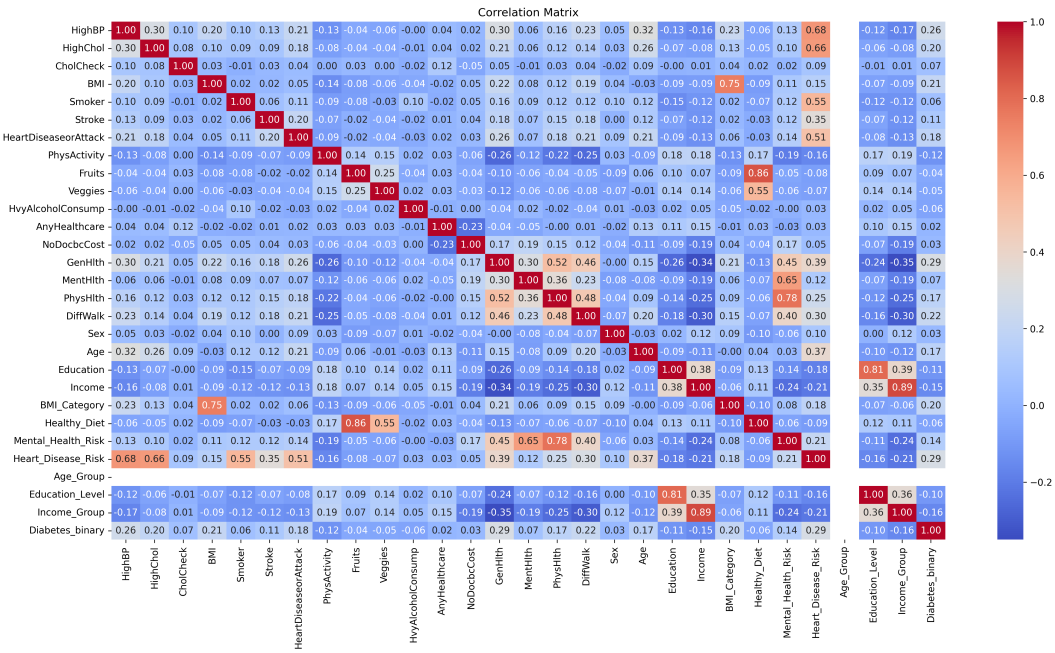


Figure 4: Correlation Matrix of the Target Variable and All Features in the Training Set

7.5 FEATURE IMPORTANCE

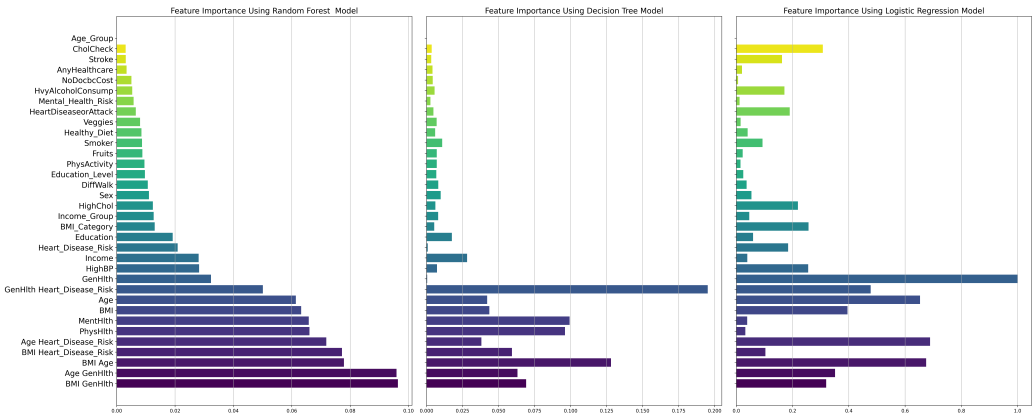


Figure 5: Feature Importance Using Different Classifiers

Knowing which features are most impactful can make complex models more interpretable, can simplify the training of the models, and can reduce overfitting. Figure 5 shows that the three models Random Forest, Decision Trees, and Logistic Regression models have very differnt feature importance mapping.

7.6 MORE RESULTS

Table 4: Summary statistics for the Training dataset

Feature	Count	Mean	Std	Min	Max
Unnamed: 0	146,626	126,710.23	73,261.55	0.00	253,678.00
HighBP	146,626	0.43	0.49	0.00	1.00
HighChol	146,626	0.42	0.49	0.00	1.00
CholCheck	146,626	0.96	0.19	0.00	1.00
BMI	146,626	28.36	6.27	11.67	98.31
Smoker	146,626	0.44	0.50	0.00	1.00
Stroke	146,626	0.04	0.20	0.00	1.00
HeartDiseaseorAttack	146,626	0.09	0.29	0.00	1.00
PhysActivity	146,626	0.76	0.43	0.00	1.00
Fruits	146,626	0.63	0.48	0.00	1.00
Veggies	146,626	0.81	0.39	0.00	1.00
HvyAlcoholConsump	146,626	0.06	0.23	0.00	1.00
AnyHealthcare	146,626	0.95	0.22	0.00	1.00
NoDocbcCost	146,626	0.08	0.28	0.00	1.00
GenHlth	146,626	2.51	1.07	1.00	5.00
MentHlth	146,626	3.19	7.43	-1.68	31.25
PhysHlth	146,626	4.25	8.74	-1.87	31.86
DiffWalk	146,626	0.17	0.37	0.00	1.00
Sex	146,626	0.44	0.50	0.00	1.00
Age	146,626	8.04	2.90	0.40	13.54
Education	146,626	5.15	0.98	1.00	6.00
Income	146,626	6.15	1.99	1.00	8.00
Healthy_Diet	146,626	0.56	0.50	0.00	1.00
Mental_Health_Risk	146,626	0.19	0.39	0.00	1.00
Heart_Disease_Risk	146,626	1.43	1.15	0.00	5.00

7.7 BEST HYPERPARAMETERS FOUND FOR EACH MODEL

Logistic Regression

- C: 15.0684
- $penalty$: l2
- max_iter: 973
- tol: 0.0128
- $class\_weight$: None
- $solver$: newton-cg

Decision Tree

- $criterion$: log_loss
- $splitter$: random
- max_depth: 48
- $min\_samples\_split$: 8
- $min\_samples\_leaf$: 2
- $class\_weight$: None

Random Forest

- $n\_estimators$: 100
- $criterion$: gini
- max_depth: None
- $min\_samples\_split$: 2
- $bootstrap$: True
- $class\_weight$: None
- n_jobs: -1

XGBoost

- $n\_estimators$: 435
- max_depth: 16
- $learning\_rate$: 0.0075
- $subsample$: 0.8918
- $colsample\_bytree$: 0.8281
- $gamma$: 0.2647
- min_child_weight: 9
- $scale\_pos\_weight$: 0.6499
- $booster$: dart

CatBoost

- $iterations$: 639
- $depth$: 6
- $learning\_rate$: 0.0145
- $l2\_leaf\_reg$: 0.1113
- $border\_count$: 224
- $bagging\_temperature$: 0.1705
- $grow\_policy$: Lossguide
- $random\_strength$: 8.1798
- $max\_ctr\_complexity$: 5
- $min\_data\_in\_leaf$: 2

- $subsample$: 0.8421

Extra Trees

- $n\_estimators$: 100
- $criterion$: gini
- max_depth: None
- $min\_samples\_split$: 2
- $bootstrap$: False
- $class\_weight$: None
- n_jobs: -1

AdaBoost

- $n\_estimators$: 416
- $learning\_rate$: 1.3173
- $algorithm$: SAMME.R

LightGBM

- $n\_estimators$: 392
- max_depth: 4
- $learning\_rate$: 0.1906
- num_leaves: 48
- $feature\_fraction$: 0.7282
- $bagging\_fraction$: 0.8210
- $bagging\_freq$: 9
- $min\_child\_samples$: 35
- $lambda\_1$: 0.0018
- $lambda\_2$: 0.0005
- $scale\_pos\_weight$: 0.8566
- $boosting\_type$: dart

Multi-layer Perceptron (MLP)

- $hidden\_layer\_sizes$: (20,)
- $activation$: logistic
- $solver$: adam
- $alpha$: 0.0006
- $learning\_rate$: constant
- $learning\_rate\_init$: 0.0003
- $batch\_size$: 64
- max_iter: 973
- $early\_stopping$: True
- $warm\_start$: False