Project 1: Supervised Learning - Classification

Alexander Svarfdal Gudmundsson Jan Babin

October 3, 2024

Contents

1 Introduction

1.1 Background

Diabetes is a chronic condition characterised by elevated blood sugar levels. While genetic factors can play a role, a significant amount of diabetes Type 2 cases is associated with lifestyle choices. According to the World Health Organization [**WHO2016**], 422 million people worldwide are living with diabetes, with numbers continuing to rise. The societal impact of diabetes is significant: a diminished quality of life and a higher risk of serious health complications which in turn may lead to increased healthcare costs. Therefore, there is medical merit in early predictions of diabetes to refine treatment and management strategies.

1.2 Objective

The goal of this project is to build a machine learning model that can identify patterns in lifestyle and demographic data associated with diabetes and thereby predict whether a given person has diabetes / is very likely do develop it. While the model is purely intended for academic purpose, the project allows us to explore how machine learning can be used to analyse health-related data and provide insights into potential risk factors.

1.3 Dataset

The dataset at hand, "Diabetes, Hypertension and Stroke Prediction," is from Kaggle and contains health-related data in CSV format intended for predicting diabetes, hypertension and stroke. For this project, we are solely focussing on diabetes. The dataset consists of 70,692 observations and 18 features, some of which are outlined as follows (for a full overview, see Table 8 in the Appendix):

- **Age**: Coded in 13 age groups (see Figure 1).
- **Sex**: Binary variable representing male (0.0) and female (1.0).
- **BMI**: Body Mass Index, a continuous variable.
- **Lifestyle indicators**: Such as smoking status (whether the individual has smoked at least 100 cigarettes in their lifetime), physical activity in the past 30 days (excluding job-related activity), daily fruit and vegetable consumption, and heavy alcohol consumption (based on defined weekly limits for men and women).
- **General health and mental/physical health**: Self-reported general health on a scale from 1 (excellent) to 5 (poor), along with the number of days with poor mental and physical health over the past 30 days.
- **Pre-existing conditions**: Information on high cholesterol, coronary heart disease or myocardial infarction, and difficulty walking.

The target variable is *Diabetes*, which is represented as a binary category (0.0 for no diabetes, 1.0 for diabetes). No missing or null values were identified in the dataset, ensuring that data cleaning was minimal and straightforward. Furthermore, the dataset is perfectly balanced, in that, 50% of observations correspond to diabetes, the other 50% do not.

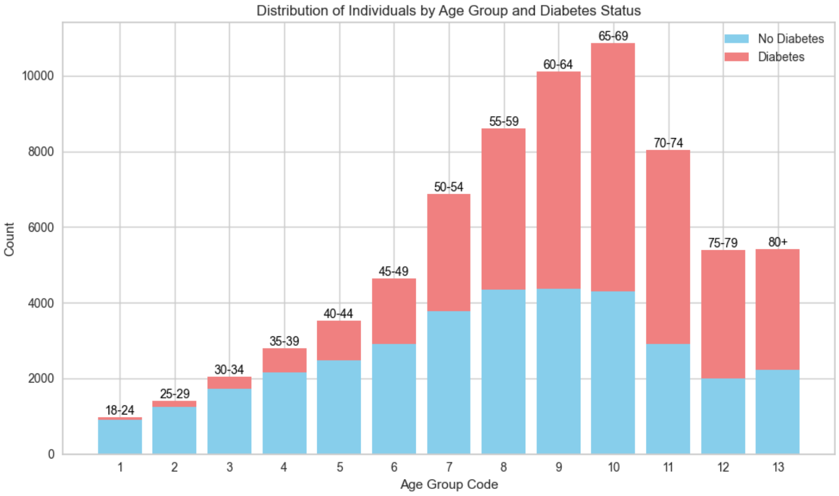

Figure 1: Distribution of Age Groups in the Dataset: Counts of individuals across thirteen defined age groups, ranging from 18 to 24 years to 80 years and older, according to the AGEG5YR scheme.

2 Process

The steps involved in the analysis follow a standard supervised machine learning pipeline, from data preparation to model selection and evaluation.

2.1 Data Loading and Exploration

The dataset was loaded into a pandas DataFrame. Initial exploration was conducted to understand the structure of the data:

- Data Integrity: We confirmed that the dataset contained no missing or null values.
- Feature Exploration: The features were analysed for their unique values and data types, revealing that many variables were binary, and the rest were either categorical or continuous. Continuous variables included features like *BMI*, *Age*, *MentHlth*, and *PhysHlth*, while categorical variables included *GenHlth* (self-reported health scale from 1 to 5) and binary indicators for lifestyle factors like smoking, physical activity, and alcohol consumption.
- Cyclical values: None of your features were cyclical (e.g. month of the year where December and January seem to be far apart numerically, but actually are not).

2.2 Preprocessing

2.2.1 One-Hot Encoding

Since the data set's features were numeric, we tried one-hot encoding features that seemed categorical into binary columns to prevent the model from interpreting these as continuous data, where actually the distance between values might not bear any meaning.

2.2.2 Scaling

Certain features with a wide range of values were standardised using sklearn's `StandardScaler`. These included: *BMI*, *MentHlth*, *PhysHlth*, and *Age*: Standardisation ensures that the models (e.g., logistic regression, support vector machines) are not biased towards features with larger numeric ranges. This is important for algorithms that rely on the magnitude of features during decision-making.

2.2.3 Train-Test Split

The dataset was split into a training set (80%) and a test set (20%) using `train_test_split` from sklearn. We decided on this split for our relatively large data set in order to retrieve both substantial training and test set sizes. This reduces overfitting (model has enough training to learn nuances without memorising specific instances) as well as yielding a robust set of unseen data for testing and is a common industry practice. Stratification was applied to ensure that the class distribution of the target variable (*Diabetes*) remained balanced across the training and test sets.

2.3 Feature Correlation Analysis

To explore potential multicollinearity among features, a correlation matrix was generated and visualised using Seaborn's `heatmap` function. The goal was to identify highly correlated features that could negatively impact model performance by redundancy. We utilised pandas' inbuilt correlation function that by default calculates the Pearson correlation coefficient, which is a measure of linear correlation.

2.4 Model Selection and Evaluation

2.4.1 PyCaret Experiment Setup

For model selection, we utilised PyCaret's `ClassificationExperiment` to quickly get a comparison between multiple classification models. The `ClassificationExperiment` tries different models to rank the models based on accuracy, using default hyperparameters based on common practices. We then selected the 5 best performing models to fine-tune further via hyperparameter optimisation.
PyCaret's built-in random grid-search was used to tune the hyperparameters of the selected models, optimising for accuracy. The random grid-search works by randomly choosing a parameter value from a given (or a default) range. It then tries out different combinations of these random hyperparameters and chooses the best one based on accuracy.
Cross-validation was automatically performed by `ClassificationExperiment`. We employed 10-fold cross-validation, where the dataset is split into 10 equal parts (folds). A model is trained on 9 of these and tested on the remaining one. This process is repeated 10 times, each time using a different fold for testing. This method ensures that every observation is used for both training and testing, which allows for a robust evaluation of the model's performance while also improving generalisability / reduce risk of overfitting. The final performance metrics are averaged in an attempt to provide a reliable estimate of model accuracy on unseen data.

2.4.2 Feature Selection

Some rudimentary feature selection was employed, using PyCaret's `ClassificationExperiment`'s inbuilt feature selection parameter. This reduces the set of features, the model is trained on, to only those deemed important based on statistical tests and feature importance techniques, which may help improve model performance.

2.4.3 Model Stacking and Ensembling

In an effort to improve prediction accuracy, two ensemble learning methods were applied:

- A hard voting ensemble (that is, majority vote) was created that involved the 5 top-performing models chosen previously, using PyCaret's `(blend_models)`. The resulting blended model runs its base models on an input and will output whatever was predicted most often among these.
- Furthermore, a combined model using stacking was also built. Stacking works by training a model to aggregate the predictions of all of the predictors in a more sophisticated way than blending.

3 Results

For the results section, we primarily focussed on evaluating different models' performances using accuracy, while also incorporating F1 score, recall, and precision. Accuracy was chosen as our main metric because it is a straightforward measure that comprehensively includes all possible outcomes of the confusion matrix. According to an article by Google [**Google**], accuracy serves as a robust quality estimate for balanced data sets.

3.1 Preprocessing

One-hot encoding features, that were already binary but numerical did not make any difference in the model's performance. This goes to show that a model can be trained on binary features equivalently well, regardless of whether they are represented by float or boolean values. However, one-hot encoding the 5-category-feature *GenHlth* resulted in slightly higher accuracy values for all models evaluated by `ClassificationExperiment`. Therefore, we proceeded with this encoding.
Scaling continuous attributes actually resulted in some of the best models performing worse then without. We also learnt that `ClassificationExperiment` takes care of preprocessing steps (involving scaling) as needed for different models. Hence, we chose to omit manual scaling and let PyCaret take care of it.

3.2 Feature Correlation Analysis

The features *GenHlth_5.0* and *PhysHlth* exhibited strong linear correlation ($r = 0.52 \geq 0.5$), albeit on the lower end. Omitting *GenHlth_5.0*, however, decreased accuracy across our top 5 models, implying the degree of correlation was not high enough to outweigh having more features / data to train on.

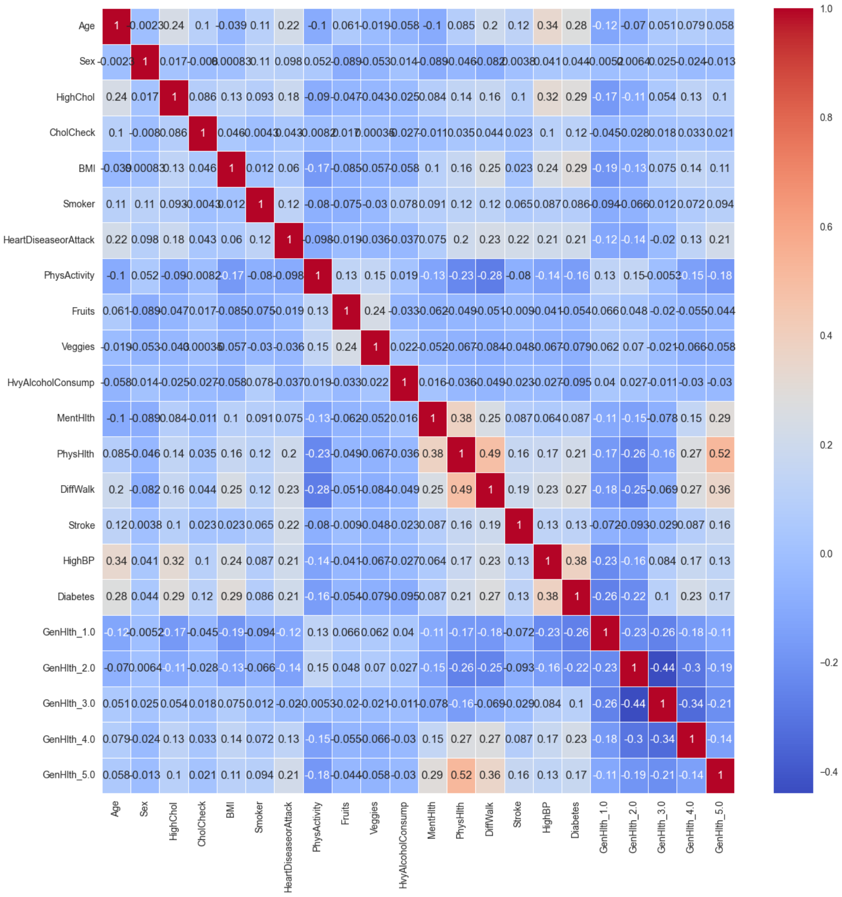

Figure 2: Correlation matrix showing relationships between features in the dataset (Pearson correlation). Higher values indicate stronger correlations.

3.3 ClassificationExperiment

With this setup, PyCaret's `ClassificationExperiment` yielded the following results:

Model	Classifier	Accuracy	Recall	Precision	F1
`gbc`	Gradient Boosting Classifier	0.7508	0.7897	0.7329	0.7601
`lightgbm`	Light Gradient Boosting Machine	0.7495	0.7926	0.7298	0.7598
`ada`	Ada Boost Classifier	0.7482	0.7731	0.7366	0.7543
`lr`	Logistic Regression	0.7470	0.7743	0.7343	0.7537
`lda`	Linear Discriminant Analysis	0.7467	0.7798	0.7314	0.7548
`ridge`	Ridge Classifier	0.7465	0.7797	0.7312	0.7546
`svm`	SVM - Linear Kernel	0.7436	0.7991	0.7206	0.7567
`rf`	Random Forest Classifier	0.7268	0.7631	0.7117	0.7364
`nb`	Naive Bayes	0.7246	0.7251	0.7244	0.7247
`et`	Extra Trees Classifier	0.7111	0.7336	0.7022	0.7175

Table 1: Performance of Top 10 Models (sorted by accuracy, top scores highlighted)

Our rudimentary feature selection analysis produced the following results:

Model	Classifier	Accuracy	Recall	Precision	F1
`gbc`	Gradient Boosting Classifier	0.6986	0.7403	0.6834	0.7106
`ada`	Ada Boost Classifier	0.6974	0.7058	0.6943	0.6998
`lightgbm`	Light Gradient Boosting Machine	0.6962	0.7248	0.6857	0.7046
`svm`	SVM - Linear Kernel	0.6913	0.7110	0.6858	0.6964
`lr`	Logistic Regression	0.6902	0.6982	0.6872	0.6926
`ridge`	Ridge Classifier	0.6896	0.7000	0.6858	0.6927
`lda`	Linear Discriminant Analysis	0.6894	0.6992	0.6859	0.6924
`rf`	Random Forest Classifier	0.6676	0.6736	0.6657	0.6696
`et`	Extra Trees Classifier	0.6614	0.6431	0.6677	0.6551
`qda`	Quadratic Discriminant Analysis	0.6588	0.5318	0.7129	0.6091

Table 2: Performance of Top 10 Models after feature selection (sorted by accuracy, top scores highlighted)

As can be seen, feature selection affected the best performing models negatively, we therefore proceeded keeping all features in the data. After optimising the hyperparameters, the resulting scores of the five best models were as shown (note that most models did not benefit from hyperparameter tuning):

Model	Classifier	Accuracy	Recall	Precision	F1
`gbc`	Gradient Boosting Classifier	0.7508	0.7897	0.7329	0.7601
`lightgbm`	Light Gradient Boosting Machine	0.7507	0.7915	0.7319	0.7604
`ada`	Ada Boost Classifier	0.7482	0.7731	0.7366	0.7543
`lr`	Logistic Regression	0.7471	0.7741	0.7345	0.7537
`lda`	Linear Discriminant Analysis	0.7467	0.7798	0.7314	0.7548

Table 3: Best Results of Top 5 Models (sorted by accuracy, top scores highlighted)

3.4 Model Stacking and Ensembling

We ensembled the 5 best tuned models using blending:

Model	Accuracy	Recall	Precision	F1
`Blended Model`	0.7490	0.7779	0.7354	0.7560

Table 4: Blending Results (Mean Performance Metrics)

The scores suggest that this ensemble would perform worse than a Gradient Boosting model alone. Our stacking ensemble yielded the following results:

Model	Accuracy	Recall	Precision	F1
`Stacking Ensemble`	0.7507	0.7800	0.7369	0.7578

Table 5: Stacking Ensemble Results (Mean Performance Metrics)

Hence, a Gradient Boosting model would outperform a stacking ensemble, too. Since our 5 best models included models that were based on a similar method (e.g., Light Gradient Boosting and Gradient Boosting), we then tried to ensemble hand-picked models in an effort to diversify the set of base models, that is, avoid near-perfect correlation between models. However, the best combination we found still involved Gradient Boosting Classifier, Light Gradient Boosting Classifier and Ada Boost Classifier. Blending performed better than stacking, the results are shown below:

Model	Accuracy	Recall	Precision	F1
`Blending Ensemble`	0.7522	0.7867	0.7360	0.7604

Table 6: Blending Ensemble Results with selected base models (gbc, ada, lightgbm) (Mean Performance Metrics)

This was the best model we could find for our data.

3.5 Testing the best model

When we were done training the best model on the training dataset, the model was evaluated on the test data. Table 7 shows the accuracy results for both the training and test sets.

Dataset	Accuracy
Training Data	0.7534
Test Data	0.7492

Table 7: Accuracy of the ensemble model on training and test data.

The results show that the blended ensamble performs well on the training and test data. Since there is little to no difference between the training and the test accuracy it means that the model is able to generalise very well on unseen data, that also indicates that our model is not overfitting. Overfitting means that the model would be learning the training data too closely, but thats not the case for our model. The final test accuracy of 74.92% is the best that we could do for this project.

4 Discussion and Future Work

While wrapping up this project up we checked other solutions on Kaggle using the same dataset like this one. We found that our accuracy score, other key metrics and the model as a whole is not that good in comparison to other models on the web. Other solutions implemented feature selection based on correlation of the features to the target class diabetes. We tested that out but found out that it made the top models from `ClassificationExperiment` perform worse. Furthermore, some of our models did not benefit from the hyperparameter tuning and the original models performed better. There are two reasons why that might be:

- The default values for the hyperparameters were already so good that the tuning did not manage to find any better ones.
- Usage of random grid search for the hyperparameter tuning, its possible that a better search method like grid search would have been better, we tried to use grid search for the tuning but found out soon that we did not have the computational time or power to move forward with that.

4.1 Future Work

The current model performs decently, there are some ways we could further improve it:

- **Feature Selection:** We could have tried out more extravagant feature selection, using feature imporatnce and only take the most important ones into the model training.
- **Hyperparameter Tuning:** Instead of using random grid search we could have used a better search method like grid search to get the better hyperparameters.
- **Interpretability:** Because the best model is a ensamble it might be hard to interperate it. It might be better to choose non ensambled model from the top tuned models for easier interpretability.

In conclusion, while the current model works decently, further improvements can be made by refining features and further tune some hyperparameters.

Appendix

Feature	**Description**
Age	Coded in 13 age groups (e.g., 1: 18-24, 2: 25-29, etc.)
Sex	Sex of the individual (0: Male, 1: Female)
HighChol	High cholesterol (0: No, 1: Yes)
CholCheck	Checked cholesterol in the last 5 years (0: No, 1: Yes)
BMI	Body Mass Index (continuous variable)
Smoker	Smoked at least 100 cigarettes in their lifetime (0: No, 1: Yes)
HeartDiseaseorAttack	History of coronary heart disease or myocardial infarction (0: No, 1: Yes)
PhysActivity	Engaged in physical activity in the past 30 days, excluding work (0: No, 1: Yes)
Fruits	Consumes fruit 1 or more times per day (0: No, 1: Yes)
Veggies	Consumes vegetables 1 or more times per day (0: No, 1: Yes)
HvyAlcoholConsump	Heavy alcohol consumption (men: 14+ drinks/week, women: 7+ drinks/week) (0: No, 1: Yes)
GenHlth	Self-reported general health (1: Excellent, 2: Very good, 3: Good, 4: Fair, 5: Poor)
MentHlth	Days of poor mental health in the past 30 days (0 to 30)
PhysHlth	Days of poor physical health in the past 30 days (0 to 30)
DiffWalk	Difficulty walking or climbing stairs (0: No, 1: Yes)
Stroke	History of stroke (0: No, 1: Yes)
HighBP	High blood pressure (0: No, 1: Yes)
Diabetes	Presence of diabetes (0: No, 1: Yes)

Table 8: Full list of dataset features used in the analysis.