CS7641-Project

Introduction

Diabetes is a serious chronic disease in which individuals lose the ability to effectively regulate their blood glucose levels. While there is no cure for diabetes, making lifestyle modifications and receiving early treatment can reduce the harms of this disease [1]. Hence, models for diabetes risk assessment can be an important tool for addressing this disease. We aim to uncover the factors that strongly predict diabetes. We use the BRFSS 2015 dataset available for free on Kaggle [3]. Each sample has 21 features. Some of the features are binary while others are continuous. The labels are either no diabetes, prediabetes, or diabetes. Prior work for classification tasks includes the use of SVMs, Random Forests, XGBoost, and Logistic Regression [2]. A detailed summary of various methods and their performance is also provided in [5].

Link to dataset: https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

Problem Definition

The aim is to identify the health indicators from the features in our dataset that can most reliably predict diabetes in patients. Additionally, although standard classification techniques simply find correlations among data, we aim to also identify causal relationships [4]. We can identify features that are most informative to potential diabetes by answering this question. The key motivation of our project is that this can lead to earlier diagnosis and better health outcomes for these patients.

Overview

Data Pre-Processing

Luckily, our dataset was clean and hence did not require any interpolation techniques. However, we conducted an experiment to try various feature recovery techniques by randomly deleting some data, which is further described in the data pre-processing section below. Also, we analyzed the dataset by computing the correlation matrix of diabetes with individual features in the dataset to understand their contribution. We also leveraged sophisticated feature selection algorithms such as Principal Components Analysis and Lasso Selection to

understand the dataset and determine if we can select fewer features that can still reliable predict diabetes.

Unsupervised Learning

We used kNN data cleaning. For the features with missing data, we used kNN to determine the nearest neighbors based on the other features and then computed the average of the missing feature for the nearest 7 neighbors and used that value to replace the missing feature value. We also used an autoencoder and PCA for feature selection.

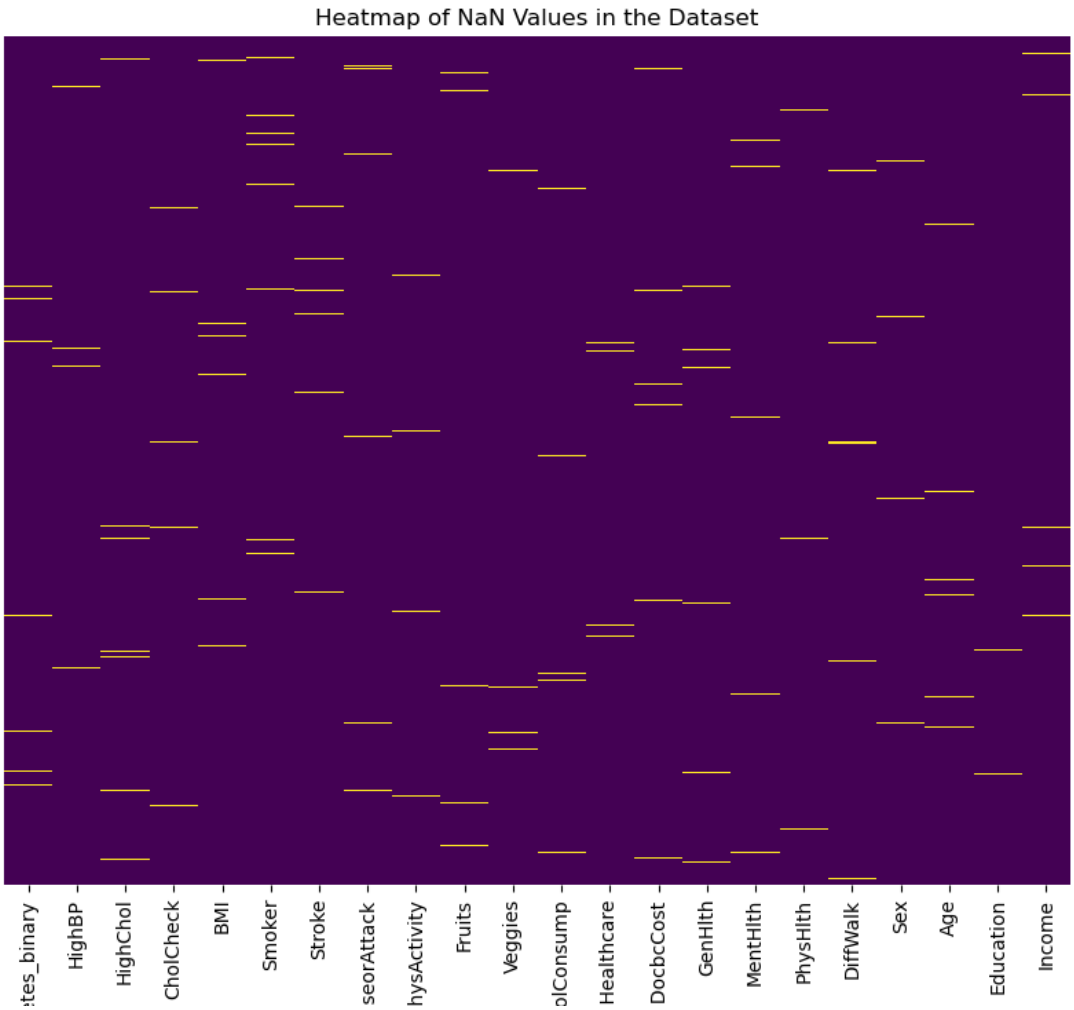
Supervised Learning

We explored whether a given feature has the potential to predict diabetes, which is a classification problem. First, we used linear models including logistic regression and support vector machines on the selected features. Then, we used non-linear classifiers such as neural networks and kernel SVM to determine whether non-linearity can further boost performance. Additionally, we explored the possibility of using a large language model (LLM). The motivation was to check if textual annotation of features (such as alcohol consumption, mental health, etc.) can guide the model better, assuming it was seen by the model during the extensive pre-training stage. Finally, we also carried out a causal analysis to find causal relationships between the features and diabetes.

Data Pre-processing

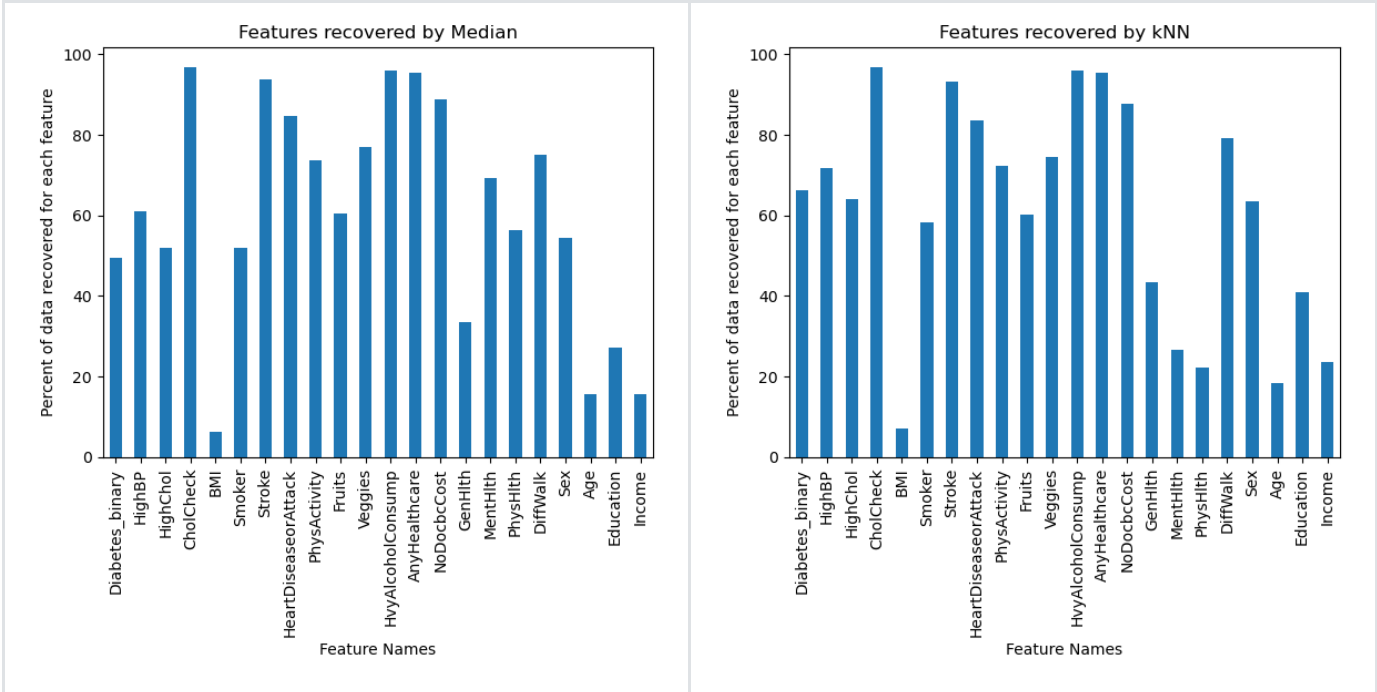
We started with a dataset that was already clean and did not have any missing features. We chose the dataset where the output labels (0 and 1) were equally balanced, with a total of N = 70,692 data points.

However, even though we started with a clean dataset, we were interested in exploring a few data-cleaning algorithms. We conducted an experiment where we simulated data that was missing data for certain features and we tried to recover those features using unsupervised learning to see how much of the missing information could be regained. We started by randomly deleting 15% of the data for each feature and replacing these data points with NaN values. The resulting dataset with missing elements is represented by the heatmap below.



After deleting some data randomly, we tried to use the mean, mode, and median of each feature to fill in the NaN values and compared the new dataset with recovered features to the original dataset. We also used kNN on this same dataset to see if it would give a higher accuracy in feature recovery than the simple arithmetic measures. In this experiment, a total of 10,560 data points were deleted and replaced with NaN among all the features. We found that for our data, using the median of each feature to fill the NaN values resulted in the highest accuracy with 60% of the deleted elements recovered to their original values. kNN performed the next best with 58% of the deleted elements that were recovered. For our dataset, the mean and mode of each feature performed very poorly in recovering the original data before deletion.

Taking a closer look at the individual features that were recovered, we found that using the median or kNN seemed to perform equally well for most of the features, but the median worked better for a few features. The breakdown for these two feature recovery methods can be seen in the graphs below. We believe that the median worked better for some of these features because the original data for these features was very skewed towards one value. Hence, using that value to replace all the missing data results in a higher recovery accuracy than using kNN. Another important note is that BMI was the one feature that had a very low recovery accuracy for all the tested methods. This could indicate that the information in the other features does not predict BMI well.



(a) Using Median for Feature Recovery

(b) Using kNN for Feature Recovery

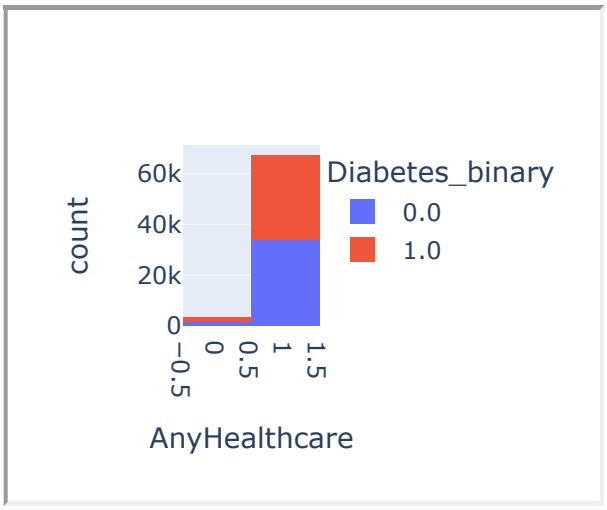
Lasso Selection

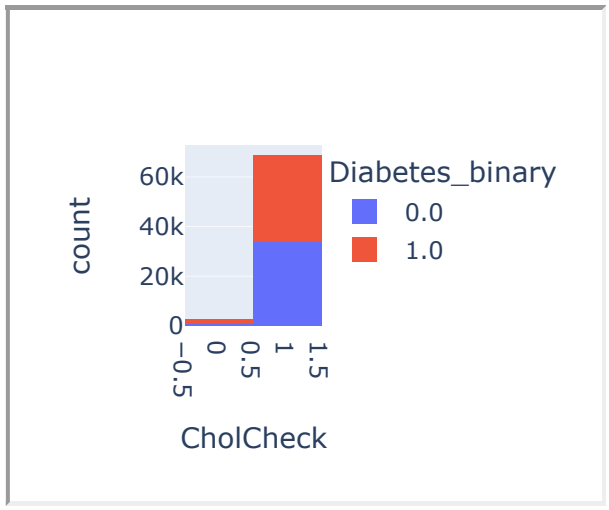
We conducted feature selection using lasso selection, where we utilized a linear support vector machine (SVC) which used L1 norm as its penalty. Then, we used sklearn's SelectFromModel class to select the most informative features. After experimenting with several regularization parameters, we settled on using C = 0.0003. From this selection, we got a total of 9 features which are: ['HighBP', 'HighChol', 'HeartDiseaseorAttack', 'PhysActivity', 'Veggies', 'GenHlth', 'DiffWalk', 'Education', 'Income']

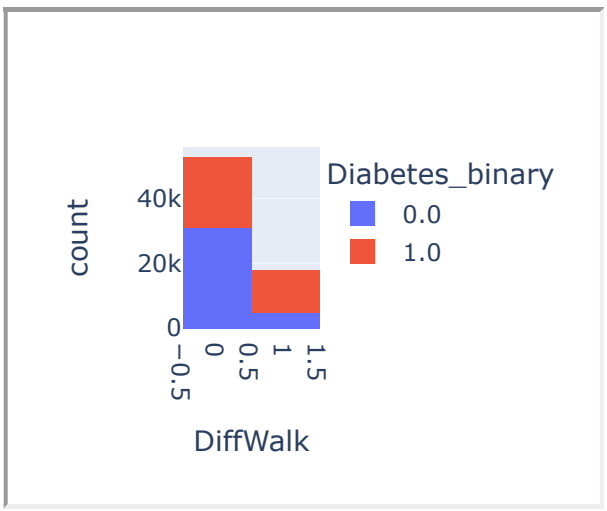
Visualization

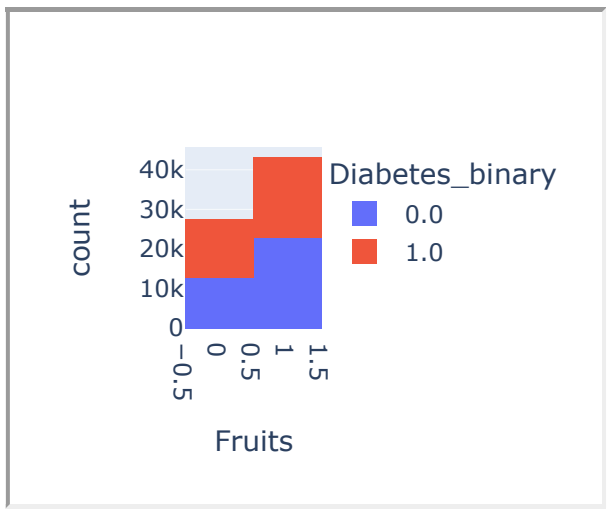
To understand the correlations between each feature with diabetes better, we plotted the following graphs.

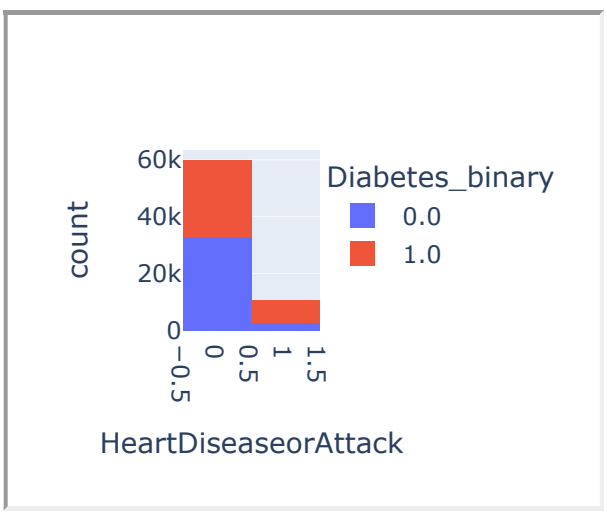
For each binary feature, we used Plotly to give interactive histograms on counts of people having that feature or symptom and whether they have diabetes.

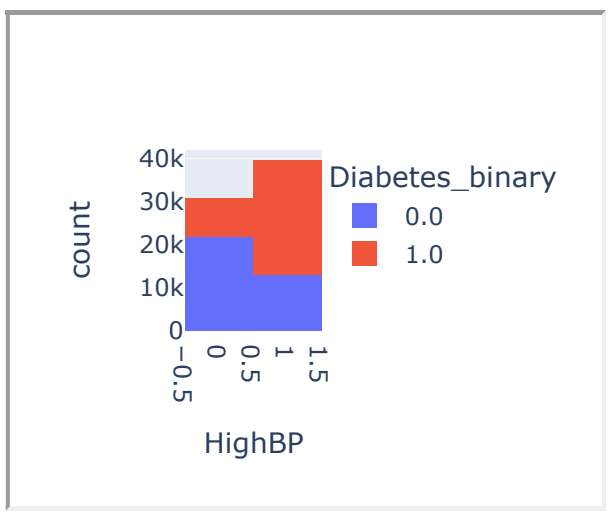

count
60k
40k
20k
0
−0.5
0
0.5
1
1.5
AnyHealthcare
Diabetes_binary
0.0
1.0

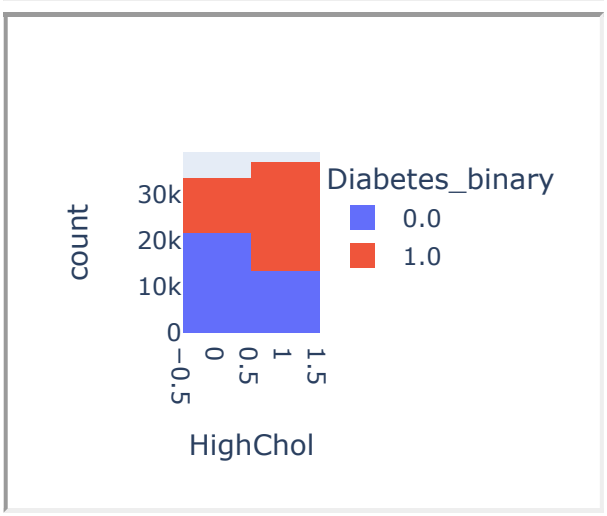

count
60k
40k
20k
0
−0.5
0
0.5
1
1.5
CholCheck
Diabetes_binary
0.0
1.0

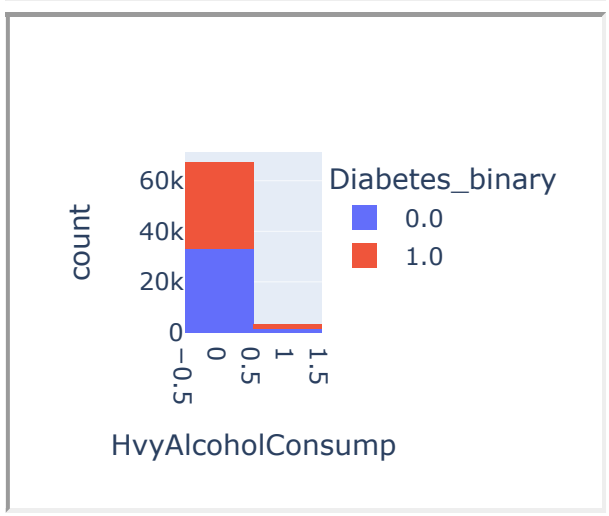

count
40k
20k
0
−0.5
0
0.5
1
1.5
DiffWalk
Diabetes_binary
0.0
1.0

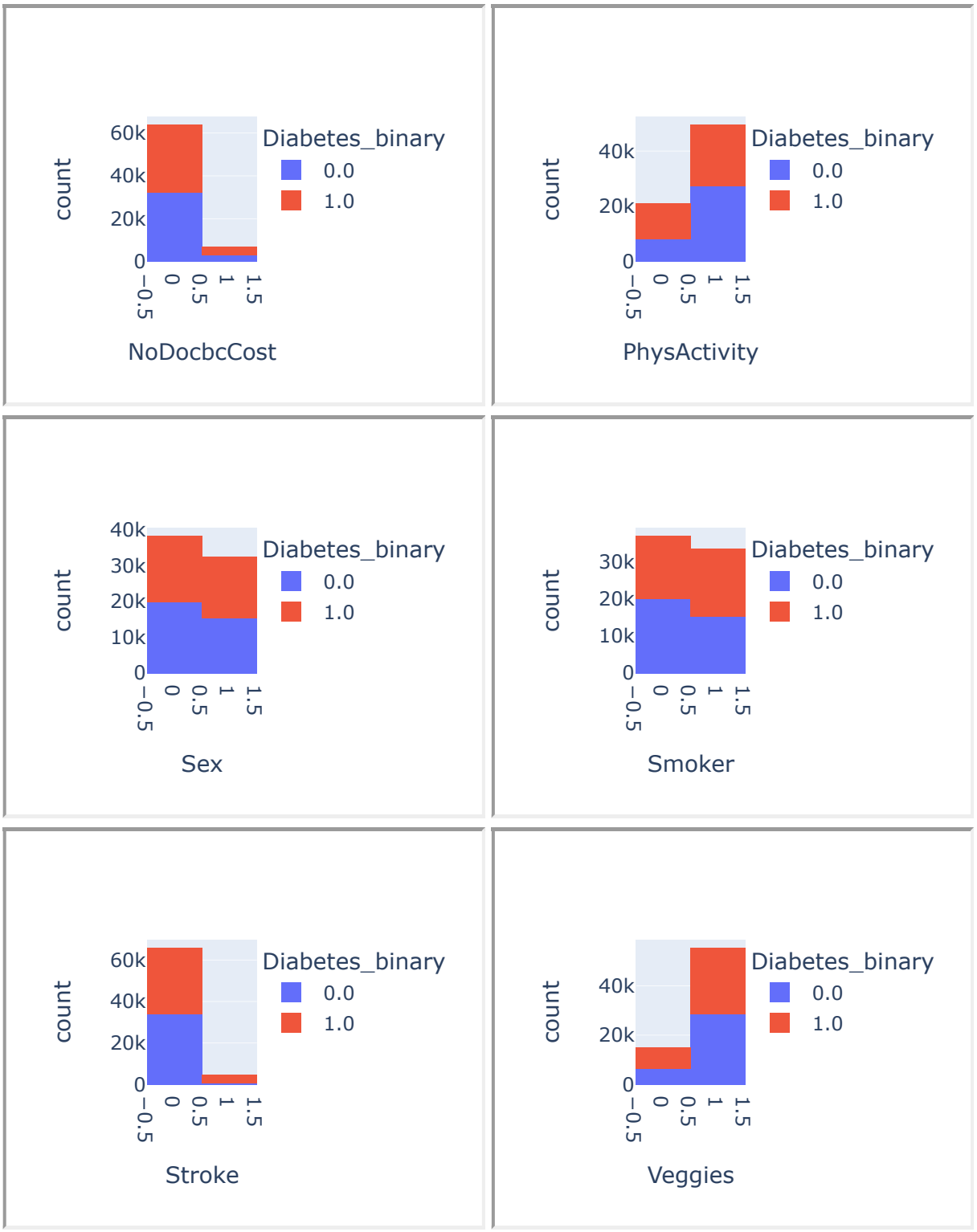

count
40k
30k
20k
10k
0
−0.5
0
0.5
1
1.5
Fruits
Diabetes_binary
0.0
1.0


count
60k
40k
20k
0
−0.5
0
0.5
1
1.5
HeartDiseaseorAttack
Diabetes_binary
0.0
1.0

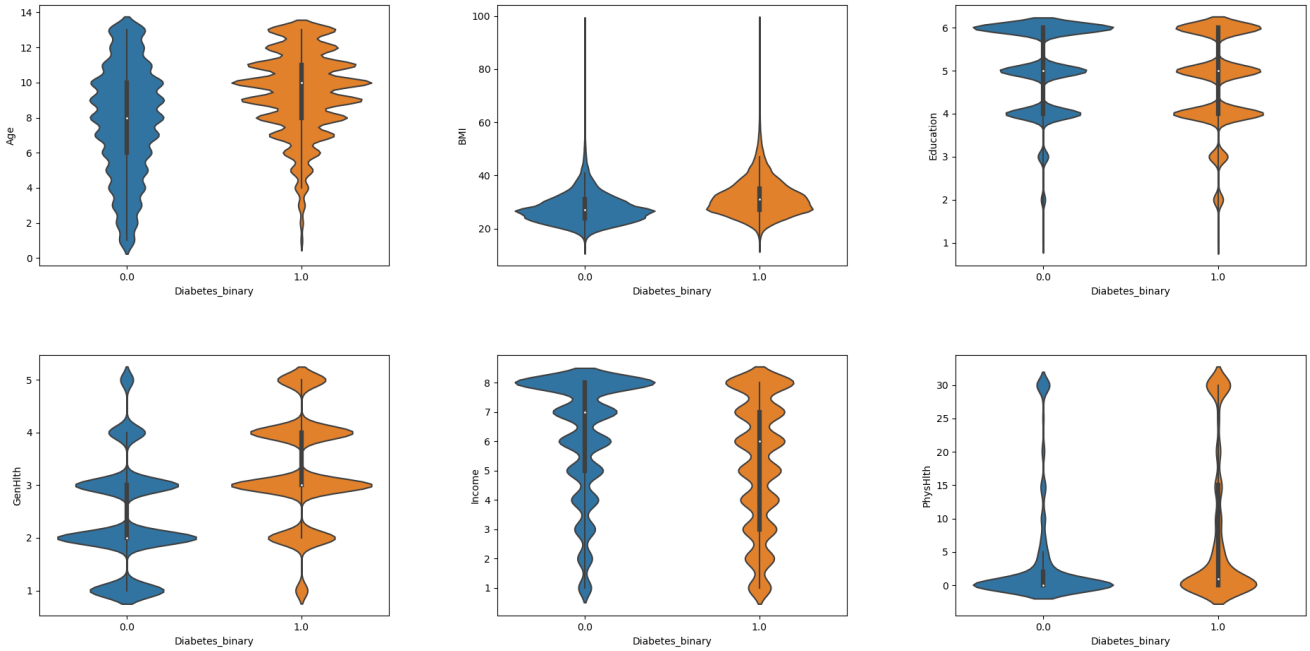

count
40k
30k
20k
10k
0
−0.5
0
0.5
1
1.5
HighBP
Diabetes_binary
0.0
1.0


count
30k
20k
10k
0
−0.5
0
0.5
1
1.5
HighChol
Diabetes_binary
0.0
1.0

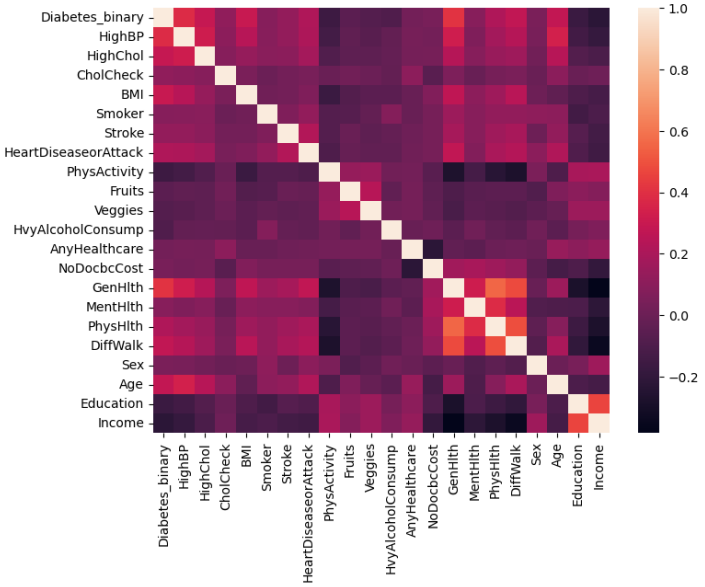

count
60k
40k
20k
0
−0.5
0
0.5
1
1.5
HvyAlcoholConsump
Diabetes_binary
0.0
1.0



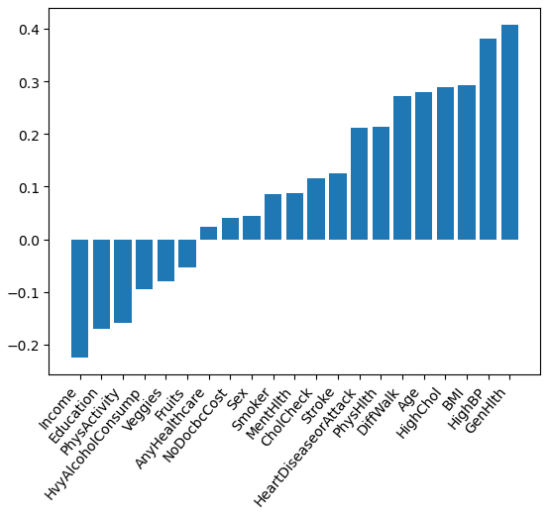
For each non-binary feature, we used Seaborn to plot a violin plot to understand the distribution of that feature value for people with and without diabetes.



To understand the strengths of the correlation between each feature and whether people have diabetes, we used Seaborn to plot a heatmap to visualize this through colors. We also plotted this barplot using Matplotlib to rank the correlation coefficient between a feature and diabetes to help us understand the correlation strengths better.



(a) Heatmap



(b) Correlation of individual feature with diabetes

Observations from the heatmap:

- We observe a few notable off-diagonal elements: Physical Health (PhysHlth) and General Health (GenHlth) are highly correlated, as one would expect
- Education and Income are also positively correlated which is aligned with our expectations.

Observations from the correlation:

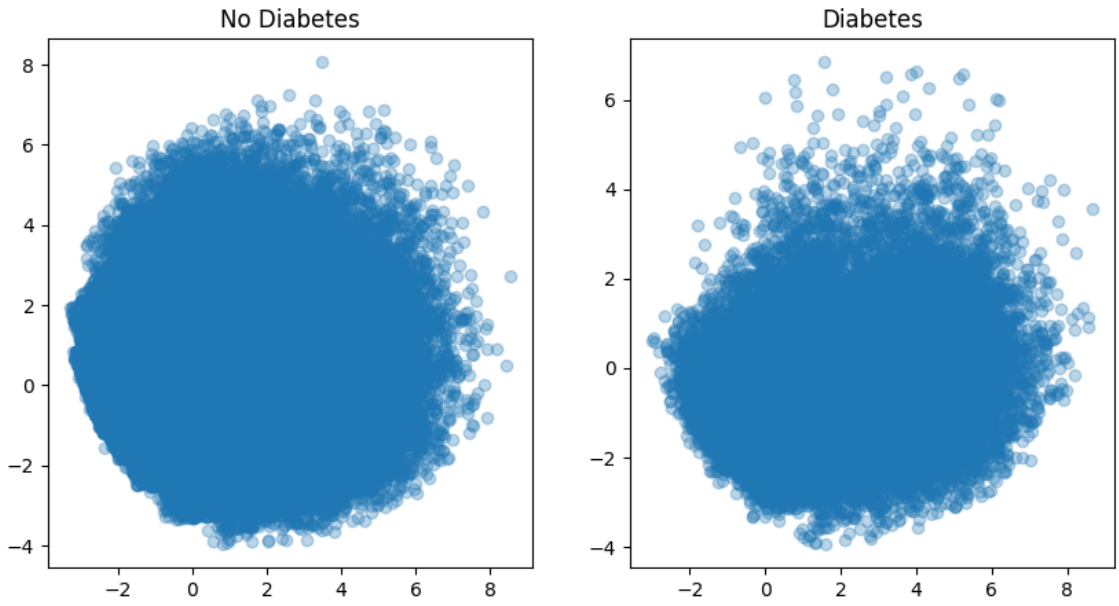
- Most results are aligned with expectations e.g. BMI and Age are positively correlated with diabetes.
- *DiffWalk*, which stands for difficulty in walking, is also positively correlated. However, intuitively speaking, it should not have a causal effect. Rather, there must be some common variable capturing 'Poor Health' which leads to both: difficulty in walking and diabetes. This feature served as the inspiration for the Causal Analysis discussed in the latter sections.

Unsupervised Models

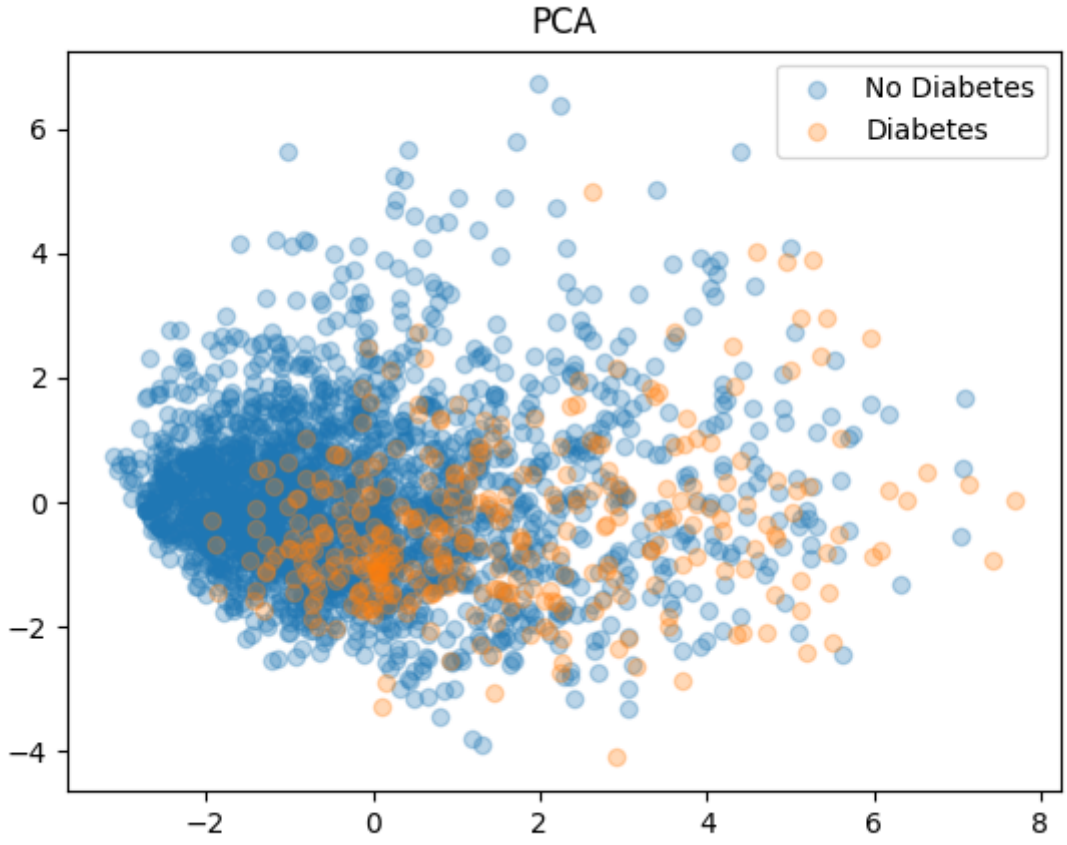
Principal Component Analysis

We first used Principal Component Analysis (PCA) on the entire dataset. We reduced the dimension of the data to 2 main components and visualized the result to determine whether we can easily classify the people with or without diabetes.

The following graph shows the result of the entire dataset:



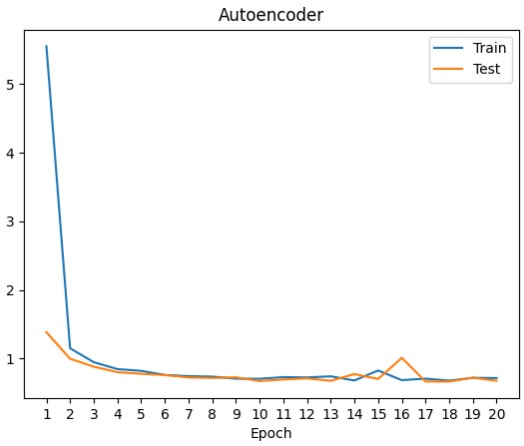
From the graph above, we noticed the dataset size is too large to have an obvious visualization, then we uniformly randomly sampled 0.01 times from the original dataset. The reason we use a uniformly random sample is uniform sample can theoretically keep the distribution of the original dataset. The following graph shows the result with a 0.01 portion of the data:



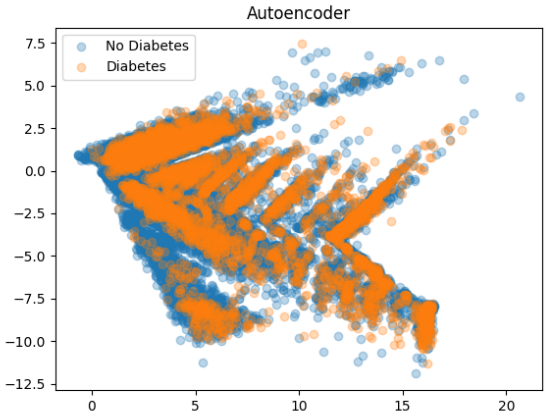
From the above two visualizations, we noticed that two features are not sufficient enough for classification. However, we also noticed that the value of the y-axis for people with diabetes is lower than for people without diabetes. Therefore, we believe there is potential to classify diabetes, and we further run an auto-encoder and visualize the data distribution after dimension reduction.

Autoencoder

The following graph indicates the changes in loss function (Mean Squared Error) during the training process. The Y-axis is the MSE value and the x-axis is the epoch. During the training process, we set the learning rate as 0.001, and the regularizer as 0.00001, with Adam Optimizer. We can observe both the training and testing loss decrease during the training time. Moreover, we illustrate the data distribution as follows:



(a) Autoencoder Loss



(b) 2-D Latent Distribution

The distribution shape can be observed differently as PCA, but there is not a clear boundary between the people with diabetes and those without diabetes. We then did more analysis based on the result.

Supervised Models

In the supervised part of our project, we took a well-rounded approach to machine learning by using different models to handle various aspects of our analysis. We employed 2 different neural networks, which have a detailed structure and training process, and also used traditional machine-supervised learning algorithms such as SVM, Logistic Regression, and Random Forest Classifier. The dataset is **randomly shuffled** and split into train:val:test in the ratio 0.7:0.15:0.15 for all the models. Additionally, we also trained each of these models on the following subset of features which were found to be useful in the Feature Selection stage: ['HighBP', 'HighChol', 'HeartDiseaseorAttack', 'PhysActivity', 'Veggies', 'GenHlth', 'DiffWalk', 'Education', 'Income'].

We take a relatively different approach to the three traditional machine learning techniques that we used. In terms of data preprocessing, Among the 21 features, the following 7 features are continuous and are hence z-score normalized: ['BMI', 'GenHlth', 'MentHlth', 'PhysHlth', 'Age', 'Education', 'Income']. Since each traditional method has several hyperparameters, we studied the performance of the models across 3 dimensions:

- *Stability:* each configuration was run on 5 fixed random seeds. We track both the mean and standard deviation of the results. A lower standard deviation is desired
- *Training dataset size:* We hypothesize that the size of the dataset is more than sufficient for our task. To test this, we randomly sample a small subset of the training data for training and test our model on the original validation and test subsets (both 15% of the original complete dataset). The fraction of training data is varied in the range [0.0001, 0.001, 0.01, 0.1, 0.5, 1]. Note that the validation and test dataset are fixed i.e. 15% and 15% of the original dataset. We only change the amount of training data. For example, a ratio of

0.001 means 0.001*70% = 0.7% of the total data is used for training while still using 15% for validation and testing.

- *Model-specific Hyperparameters:*
  - Logistic Regression: Since our features are normalized and number of samples » number of features, we explore the possibility of feature selection using L1. The L1 regularization penalty is varied in the same range as above: [0.01, 1, 10, 100]
  - SVM: Margin is the most important hyperparameter which controls the amount of L2 regularization. We vary it in the range [0.01, 1, 10, 100]. Also, the popular RBF kernel is used. (There is no option of L1 regularization for SVMs)
  - Random Forest: We vary the number of estimators (trees in the RF) in the range [10, 20, 50, 100]

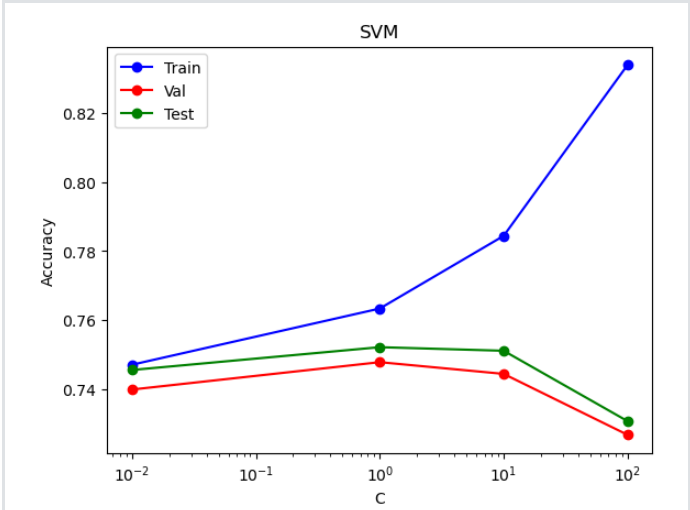
Each of these traditional machine learning models has two sets of experiments:

1. We report results on varying the hyperparameters for each of the three models. The entire training dataset is used in this case. Note that for SVM and Logistic Regression, the value of hyperparameter C denotes inverse regularization i.e. smaller value of C indicates higher regularization (consistent with the skLearn specification).
2. Motivated by the fact that 49,000 training examples, each with 21 features, might be more than required, we vary the size of just the training dataset and report results. Note that we only report the test accuracy (on the entire test set) to avoid cluttered plots.

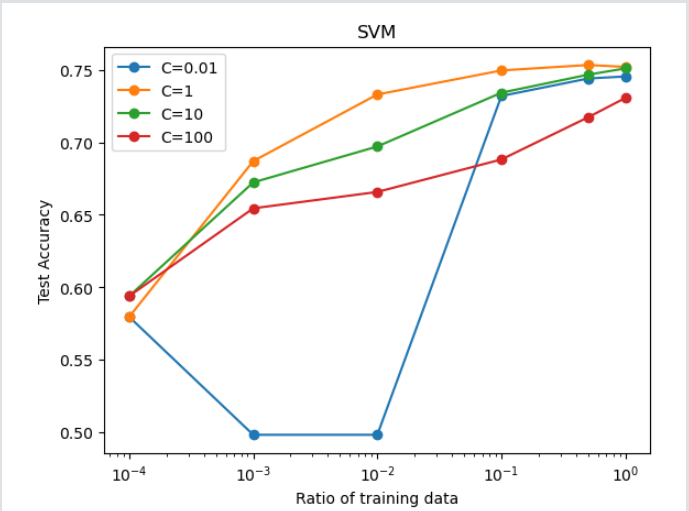
Results for each of the models on this set of experiments are shown below. A detailed comparison can be seen in the Results section.

SVM

In (a) we see that there is a sweet spot (at C = 1) for which validation accuracy peaks. On the other hand, training accuracy goes up with increasing C (decreasing regularization). Similarly, The performance of SVM increases and then saturates as we increase the amount of training data (from (b)). The plot for C=0.01 is unpredictable since it imposes heavy regularization, making the model unstable for smaller training subsets.



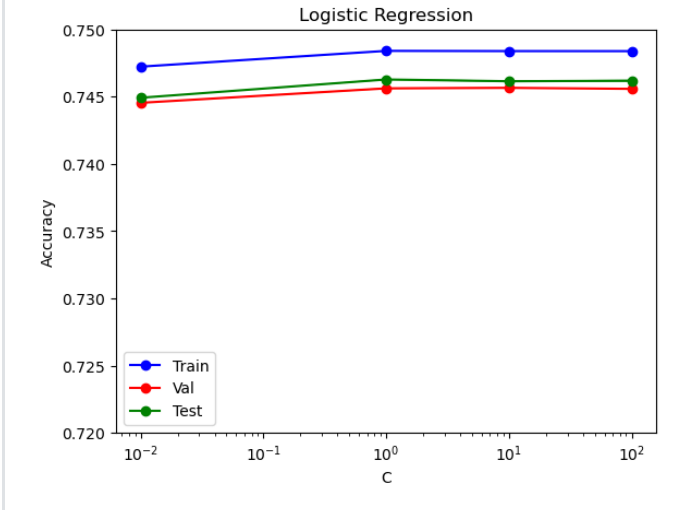
(a) Variation in Accuracy with C



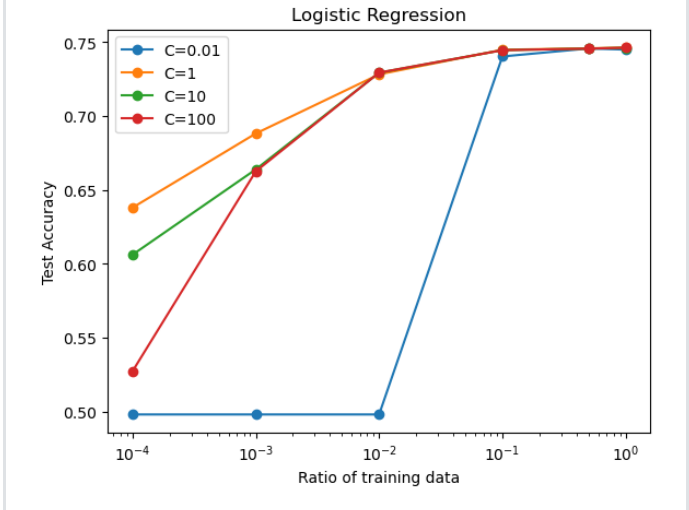
(b) Variation in Accuracy with Training Data

Logistic Regression

The performance of logistic regression is not very sensitive to the regularization. This is good news since we don't need to worry much about tuning. Compared to SVM, the performance of LR saturates very early, at around 10% of the training data. This indicates that LR needs less data than SVM. One potential explanation is that SVM (with RBF kernel) is a non-linear method, hence requiring more data.



(a) Variation in Accuracy with C

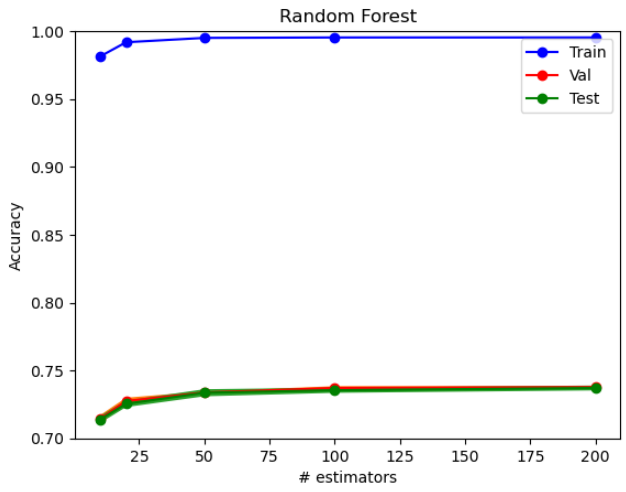


(b) Variation in Accuracy with Training Data

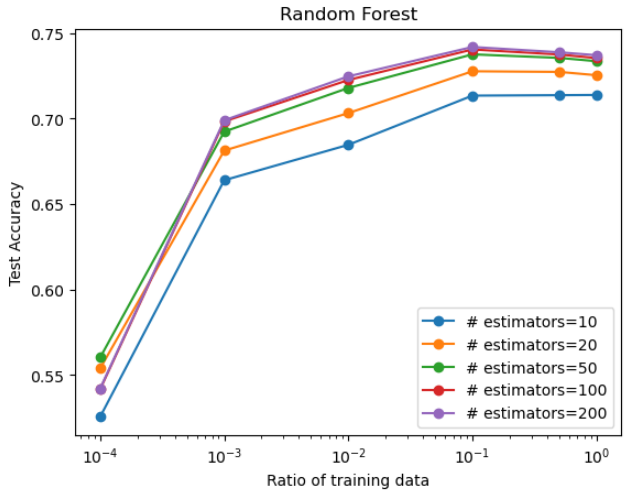
Random Forest

We see in the figure below, RF heavily overfits since training accuracy is much higher than the test. Also, the performance saturates as the number of estimators increases. Unlike regularization in SVM, adding more estimators does not hurt performance because bagging reduces variance. RF performance peaks at a ratio of around 10% but slightly goes down

thereafter (at 50% and 100% of the training data). This indicates that with more trees, the model overfits the training data as we increase it.



(a) Variation in Accuracy with C



(b) Variation in Accuracy with Training Data

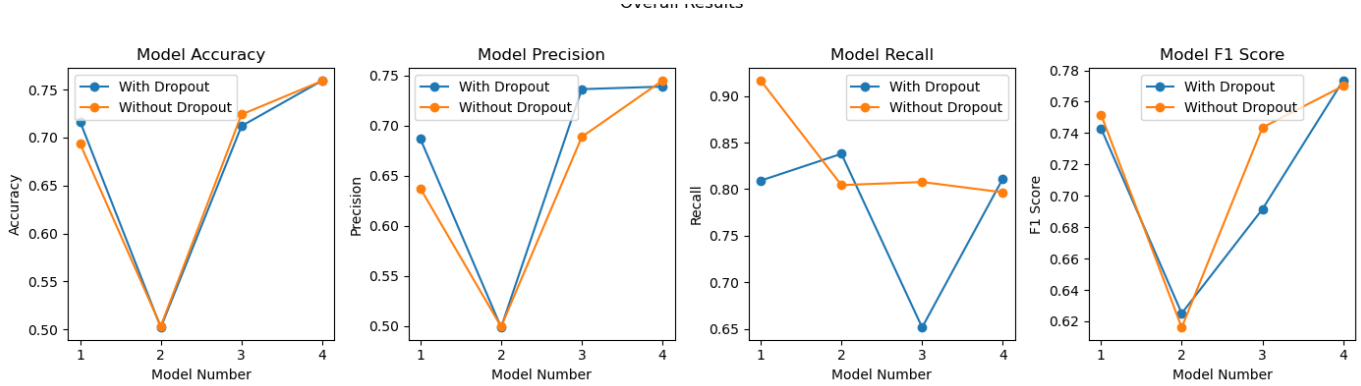
Neural Networks

We used two different Neural Networks to tackle our problem. Both models consisted of four layers each, featuring a ReLU activation function and concluding with a sigmoid activation. One of the models included additional dropout layers after each layer to prevent overfitting. Both models employed Adam optimization with a decreasing learning rate during iterations. To prevent overfitting, we implemented early stopping based on the validation loss, selecting the model with the least validation error for evaluation. We utilized a batch size of 16 and allowed each model to run for 30 epochs unless halted by early stopping.

To enhance our results, we deployed these models in four different methods:

1. **All Training Data:**
    - Both models were trained on the entire dataset and evaluated on the testing data, resulting in an overall accuracy of approximately 70%.
2. **Selected Features Only:**
    - The models were then trained exclusively on the selected features. However, due to the limited data, the overall accuracy experienced a significant drop.
3. **Weighted Selected Features:**
    - We used all the training data but assigned a weightage of 3 to the selected features. This step significantly improved accuracy, precision, and recall since these features played a crucial role in our predictions.
4. **Normalized Training Data with Weighted Features:**
    - In the final method, we normalized all the training data and assigned a three times weightage to the selected features. This approach was built on the positive trend observed with increasing weightage and yielded the best overall accuracy.

The results of each of the models can be seen below. A detailed comparison can be seen in the results section.



LLM Exploration

We explored the possibility of using large language models to predict whether people would have diabetes based on input features.

Given the CSV file, we use the first column “Diabetes_binary” as the ground truth label for our model, with values of 0 and 1. For the remaining columns, we join the features by having the feature column name and value and join them with a space.

We used the “bert-base-uncased” model for training. The training arguments we used are num_train_epochs=1, per_device_train_batch_size=32, warmup_steps=100, weight_decay=0.01. An example of input for the model is **highbp : 1.0 highchol : 0.0 cholcheck : 1.0 bmi : 26.0 smoker : 0.0 stroke : 0.0 heartdiseaseorattack : 0.0 physactivity : 1.0 fruits : 0.0 veggies : 1.0 hvyalcoholconsump : 0.0 anyhealthcare : 1.0 nodocbccost : 0.0 genhlth : 3.0 menthlth : 5.0 physhlth : 30.0 diffwalk : 0.0**. One result output is below

```
Case 5:

Input: "highbp : 1.0 highchol : 0.0 cholcheck : 1.0 bmi : 47.0 smoker : 0.0 stroke

True Label: Diabetes

Predicted Label: Diabetes
```

We used per_device_eval_batch_size=64 and evaluation_strategy=”epoch” to evaluate our training model. See the results and conclusions section for detailed evaluation numbers. Our exploration shows the potential of using LLM for classification. Future research could perform more fine-tuning and enhance model performances on such tasks.

Results and Conclusions

Given below is a summary of the best-performing models. Following the standard methodology for tuning hyperparameters, we report the test accuracy, precision, recall, and F1 scores of the model which gives the best validation performance.

Model	Accuracy	Precision	Recall	F1 score
SVM	0.752	0.726	0.810	0.766
LR	0.746	0.737	0.767	0.752
RF	0.737	0.721	0.777	0.748
NN-4 (With Droupout)	0.759	0.741	0.818	0.771
LLM	0.737	0.712	0.783	0.748

As we can see from the table above, the order of performance is NN > SVM > LR > RF > LLM. All methods, at their best performance, give similar accuracy of around 73-76%. The difference is minimal. Among these models, LR and RF saturate very quickly, at around 10% of the training dataset (or 4900 samples). It is interesting to note that the recall for all these models is slightly higher than precision. However, the difference is not significant (~4%).

The best test performance of the four models on the chosen subset of features (reported in the Data Pre-Processing section)

Model	Test Accuracy (subset)	Test Accuracy (all)
SVM	0.731	0.752
LR	0.727	0.746
RF	0.709	0.737
NN-2 (Without Droupout)	0.727	0.733
LLM	0.725	0.737

There is a noticeable drop in performance on keeping only the top 9 features discovered in the feature selection stage.

Causal analysis

All machine learning methods rely on correlations to predict the desired target variable. However, it has been known (for centuries) that *correlation does not imply causation*.

A famous example of this phenomenon is the strong correlation between ice cream sales and drownings. If we plot the two, we see a strong positive correlation between the two. However, there is no causal relation between the two. How can we explain this? One can easily predict

that there is a common variable that influences both: temperature. As the temperature increases, more people visit the beach. With more people, there are more drownings and higher ice cream sales! Such a variable (temperature) is called a **confounder**.

Relevance to our problem

In the case of diabetes prediction, we care not only about correlation but also causation since we want to suggest remedies for preventing diabetes. If a feature X is highly correlated (either positively or negatively) with diabetes, we want to establish if it's a causal relationship or simply a spurious correlation. If we can somehow establish a causal relationship i.e. X leads to diabetes, we can devise policies that target the feature X (e.g. obesity).

The analysis was motivated by the presence of some features such as 'Difficulty in walking' (*DiffWalk* in the dataset) which, by intuition, should not directly lead to diabetes. There must be some confounding variable(s) which lead to both: Difficulty in Walking and Diabetes. A person could be finding it difficult to walk due to several other reasons such as a fracture and hence the causal relationship between the two should be very weak. In the feature pre-processing section, we see that there is a significant (negative) correlation between Diabetes and *DiffWalk*. Does this mean that we should prevent walking injuries to prevent diabetes? Intuitively, the answer is no. Can we somehow show it rigorously using the data?

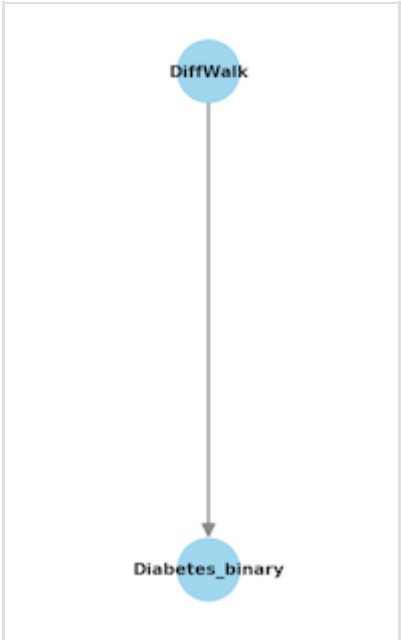
Procedure

Having motivated the problem, we now turn our attention to Causal Inference. It supplies us with tools to answer such questions. For brevity purposes, we only mention the procedure through which one can identify causal relationships. For a detailed explanation of each step, please refer to the documentation of the dowhy library, which we use for analysis.

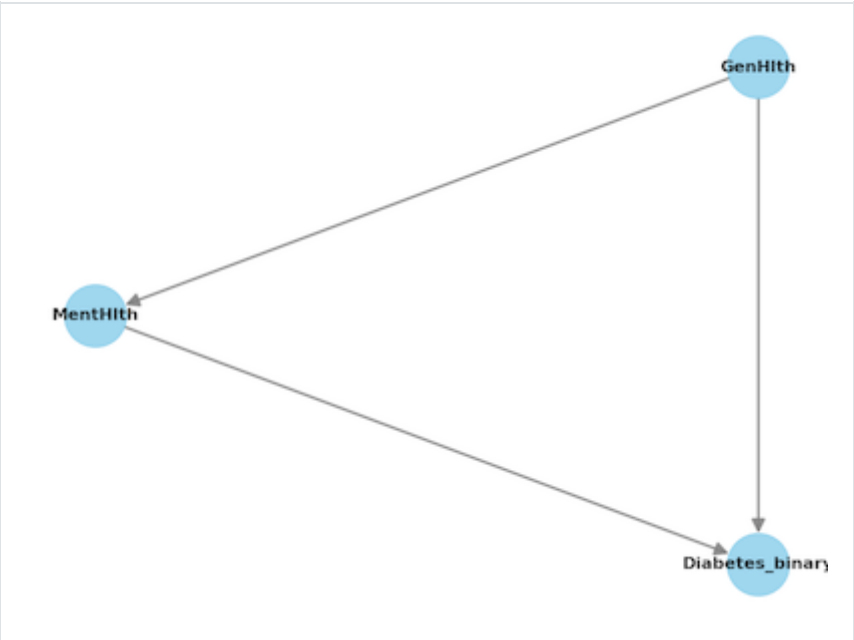
1. Design a directed acyclic graph (DAG). Each node is a feature and the arrows indicate causal relationships. The graph encodes our intuition of potential causal effects which cannot be determined from data alone.
2. Intervene on a particular feature by explicitly setting its value. This is how we get rid of spurious correlations since a confounder's effect is nullified when we intervene on the value of the feature.
3. Compute the average causal effect across a dataset of setting the value of that variable i.e. $ `\sum_w E[Y = 1 | X = 1, W = w] P(W = w) - \sum_w E[Y = 1 | X = 0, W = w] P(W = w)` $ Here, Y is the outcome (diabetes), X is the feature we intervene on and W is the set of confounding features. Such an analysis is called the backdoor analysis since the confounder W serves as a backdoor influence. We are computing the effect of changing X on the outcome Y.

Example 1

Let Y = Diabetes, X = *DifficultWalk*, and W = *GeneralHealth*. We hypothesize that X and Y could be highly correlated but it is due to a common confounding variable W called *GeneralHealth* which influences both.



(a) DAG for simple case



(b) DAG with confounder W

On training a simple linear estimation model for the below DAG, we get a causal estimate of 0.31.

In other words, $ `E[Y = 1 | X = 1] - E[Y = 1 | X = 0] = b = 0.31` $. Standard machine learning models give us such correlation-based estimates. We have not incorporated any causality-based assumptions in this setup as of yet (figure (a) above).

By conditioning on the value of *GeneralHealth*, the coefficient drops to 0.11 (DAG (b) in figure above). This sharp drop suggests that GeneralHealth influences both: diabetes and difficulty in walking. On knowing its value, the direct causal effect of *DiffWalk* drops as expected!

We carry the same analysis for different confounding variables W and report the change in the causal estimate for *DiffWalk* below:

Confounder (W)	Causal estimate (c)	Change from baseline (c - b)
General Health	0.11	**-0.2**
BMI	0.24	-0.07
Age	0.26	-0.05
Sex	0.32	+0.01

As we can observe, General Health as a confounder leads to the highest decrease in the causal estimate of Diabetes from DiffWalk. This is in line with our expectations. On the other hand, *Sex* has a negligible change in the causal estimate. This is expected because our causal

assumption that *Sex* determines *DiffWalk* is not justified. Other factors such as BMI and Age also decrease the causal estimate by a moderate amount.

Note that 0.11 is still a non-zero value, which indicates that GeneralHealth is not the perfect confounding variable.

Example 2

We follow the same analysis as before. This time, we consider *HeartDisease* as the treatment variable i.e. Y = Diabetes, X = *HeartDisease*, and W (confounder) is varied. The motivation is similar: intuitively, heart disease should not lead to diabetes. However, one can imagine some confounding variable that captures the overall health of the individual. Then, it is not surprising to guess that poor health leads to both Heart Disease and Diabetes!

The baseline causal estimate i.e. $ `E[Y = 1 | X = 1] - E[Y = 1 | X = 0] = b = 0.29` $.

We carry the same analysis for different confounding variables W and report the change in the causal estimate for *HeartDisease* below:

Confounder (W)	Causal estimate (c)	Change from baseline (c - b)
General Health	0.15	**-0.14**
BMI	0.27	-0.02
Age	0.22	-0.07
Sex	0.29	0.00

Again, General Health leads to the lowest decrease in the causal estimate. BMI, Age, and Sex lead to negligible changes, thereby indicating they are not true confounders.

Conclusion

Through causal analysis, we established that features such as *DifficultWalking* and *HeartDisease* may show significant statistical correlation with Diabetes but they are not true causal factors.

Next Steps

In this project, we focused on the prediction of diabetes from 21 interpretable features. We first visualized the dataset using techniques such as histograms. On applying standard supervised learning techniques such as SVM, RF, LR, and Neural networks, we attained an accuracy of 76% on diabetes prediction. On analyzing the performance of each method as a function of the dataset size, we discovered that just 10% of the dataset is enough to capture the

distribution. We then tried out an LLM for the same task and surprisingly discovered that it performs almost as well as standard supervised learning methods. Finally, we applied causal inference techniques to identify whether certain features cause diabetes or are merely correlated with diabetes due to confounding factors.

Several potential research directions could be pursued:

- Combine multiple datasets with potentially different features and use unsupervised learning to combine them.
- Ensemble various techniques (LLM, RF, SVM, NN, etc) and use voting to combine the best of all models.
- Experiment with different DAGs capturing various causal relationships.

Acknowledgements

We would like to thank Prof. Roozbahani for the lectures and the TAs for their help.

References

[1] Diabetes Health Indicators Code, 11 2021.Abhari, Shahabeddin, et al. “Artificial intelligence applications in type 2 diabetes mellitus care: focus on machine learning methods”. Healthcare Informatics Research, vol. 25, no. 4, 2019, p. 248.

[2] Diabetes Health Indicators Code, 11 2021.

[3] Diabetes Health Indicators Dataset, 11 2021.

[4] Pedro Sanchez, Jeremy P Voisey, Tian Xia, Hannah I Watson, Alison Q O’Neil, and Sotirios ATsaftaris. Causal machine learning for healthcare and precision medicine. Royal Society Open Science, 9(8):220638, 2022.

[5] Zahid Ullah, Farrukh Saleem, Mona Jamjoom, Bahjat Fakieh, Faris Kateb, Abdullah Marish Ali, Babar Shah, et al. Detecting high-risk factors and early diagnosis of diabetes using machine learning methods. Computational Intelligence and Neuroscience, 2022, 2022.

Completed Timeline

We use the GanttChart to track the completed timeline.

Link: https://docs.google.com/spreadsheets/d/19UiVypS2rqBWYJJoYNOczcpD6UlQnUlm/edit?usp=drive_link&ouid=107541174236383849913&rtpof=true&sd=true

Contribution

All members contributed to the midterm report, final report, and the video.

Amitej: Found dataset. Wrote problem definition section. Data Cleaning. Feature Selection. Unsupervised Learning.

Mithilesh: Found dataset. Wrote Background section. Data Cleaning. Supervised Learning Models (SVM, DT, RF). Causal Analysis.

Sheraz: Found dataset. Recorded presentation videos. Data Cleaning. Neural Networks, Unsupervised Learning.

Sixuan: Found dataset. Wrote Methods section. Data Cleaning. Feature Selection. PCA. Autoencoder.

Yiyang: Found dataset. Wrote Potential results and discussion section. Data Cleaning. Feature Visualization. LLM classification exploration.