Cairo University

Faculty of Engineering
Cairo University

MACHINE LEARNING PROJECT

Eng: Mohamed Shawky

Name	Sec	BN	ID
Ahmed Samy Ibrahim Mohamed	1	8	9210073
Kareem Samy Fawzy El Sayed	2	2	9210838
Nancy Ayman Mohamed Salahuddin	2	27	9213410
Yara Hisham Shaaban Mahrous	2	31	9211350

Table of Contents

Problem definition:

The problem is a binary classification task aimed at predicting whether a patient is at risk of developing diabetes based on various health indicators. Given medical and demographic data such as glucose levels, blood pressure, BMI, age, and insulin levels, the goal is to develop a predictive model that can accurately identify individuals who are likely to develop diabetes. Early detection can enable timely intervention and better disease management.

Motivation:

The motivation for addressing this problem lies in the growing global burden of diabetes, a chronic disease that can lead to severe complications such as heart disease, kidney failure, and vision loss if left unmanaged. By leveraging machine learning algorithms to predict diabetes risk early, healthcare providers can implement preventive measures, recommend lifestyle changes, and personalize treatment plans. This proactive approach can improve patient outcomes, reduce healthcare costs, and enhance the quality of life for individuals at risk. Additionally, such predictive models can support public health initiatives by identifying high-risk populations and optimizing resource allocation for diabetes prevention programs.

Evaluation Metrics:

1. Accuracy
2. Macro recall

The contribution of each team member

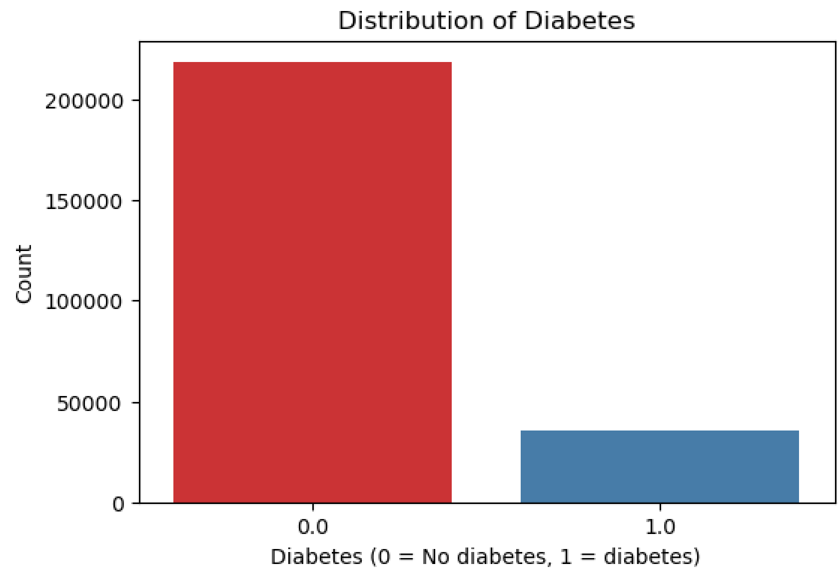
Name	Conttibution
Ahmed Samy	Logistic regression- Reandom forest- ZeroR
Kareem Samy	Logistic regression- Reandom forest-ZeroR
Nancy Ayman	Visualization- preceptron-SVM
Yara HIsham	Visualization- preceptron-SVM

Exploratory Data Analysis (EDA):

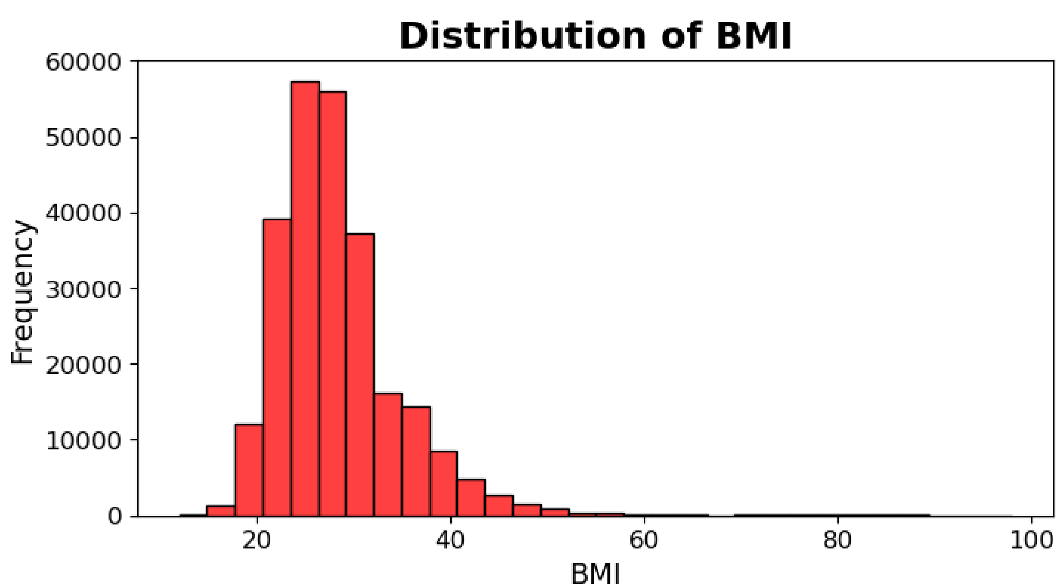
The first step was understanding the dataset, the features corelations and get insights for the output of the classification modes.

A- Data set distribution & outliers:

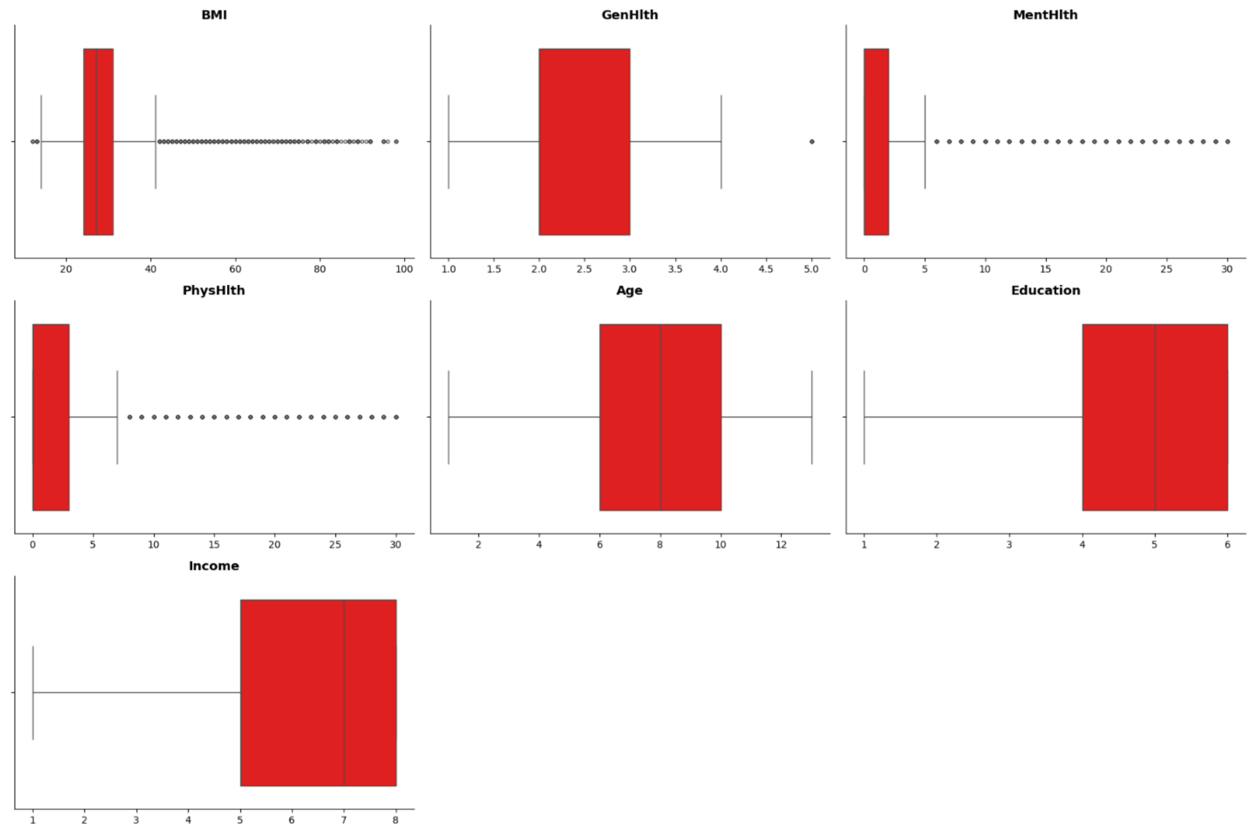
1- The data set distribution over the the prediction target (diabetes) is bias toward healthy people(not diabetic) which is realistic as the majority of the humans don't suffer for diabetes. However that is considered as a challenge for out project and biased data leads to baise model trainging and miss classification especially for the diabetes class.



2- BMI (Body Mass Index) is a key predictor of diabetes risk across different age groups. Tracking BMI trends over time can help identify the age range (e.g., 20-40 years) where individuals are most likely to be diagnosed with diabetes. This insight enables targeted awareness campaigns, encouraging people in this high-risk age group to undergo regular health screenings for early detection and prevention.



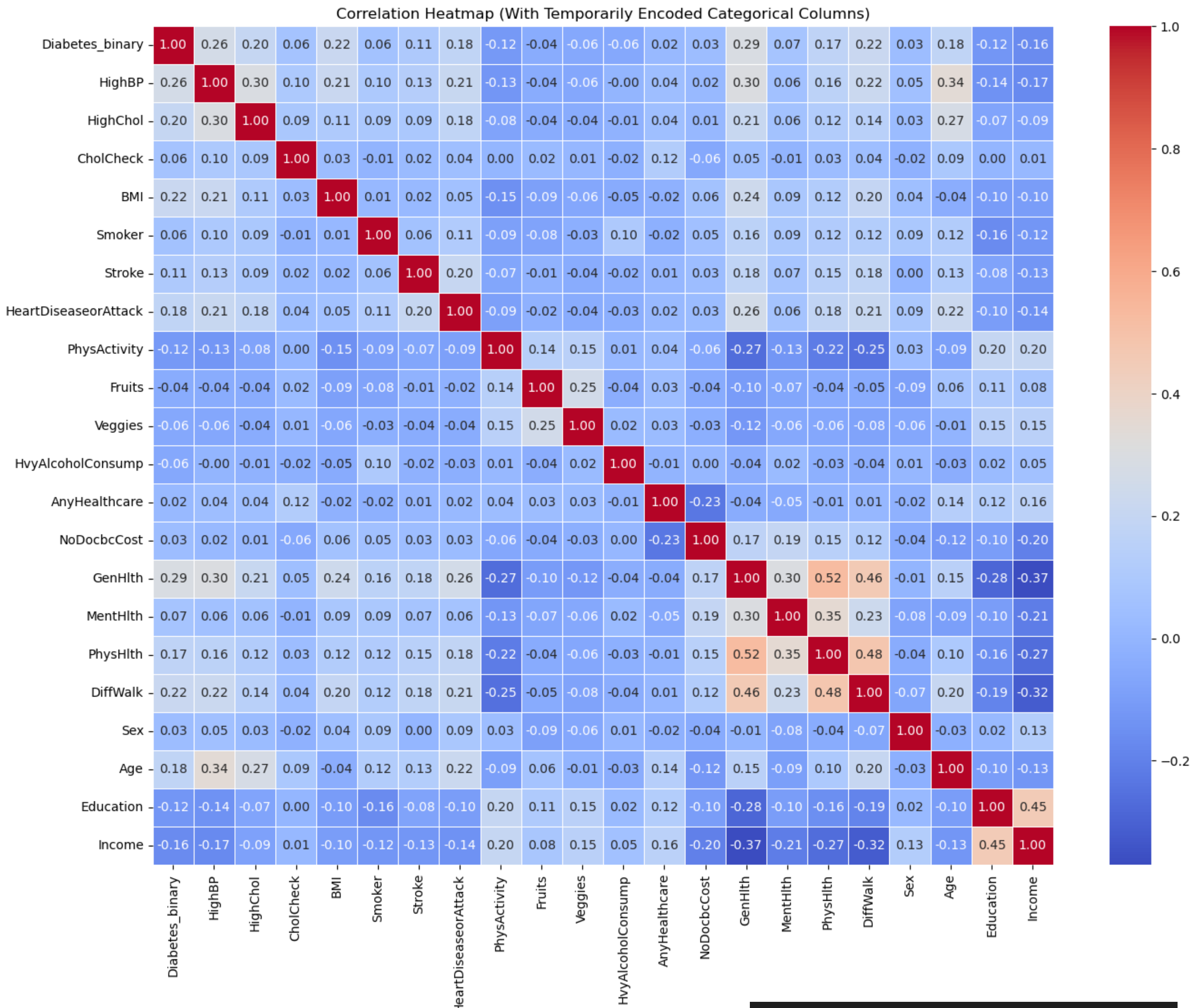
3- Features outiers analysis:



As shown there are 4 features that have outliers that needs to be elimiated. But after revising the dataset description, physHlth and MentHlth are both indicators for range (0-30) so despite being rare values they are crutical in the classification as they define a group of peaple with their characteristic.

```
Feature 'BMI' has 9847 outliers.
Feature 'GenHlth' has 12081 outliers.
Feature 'MentHlth' has 36208 outliers.
Feature 'PhysHlth' has 40949 outliers.
Feature 'Age' has 0 outliers.
Feature 'Education' has 0 outliers.
Feature 'Income' has 0 outliers.
```

B- Features overall correcation:

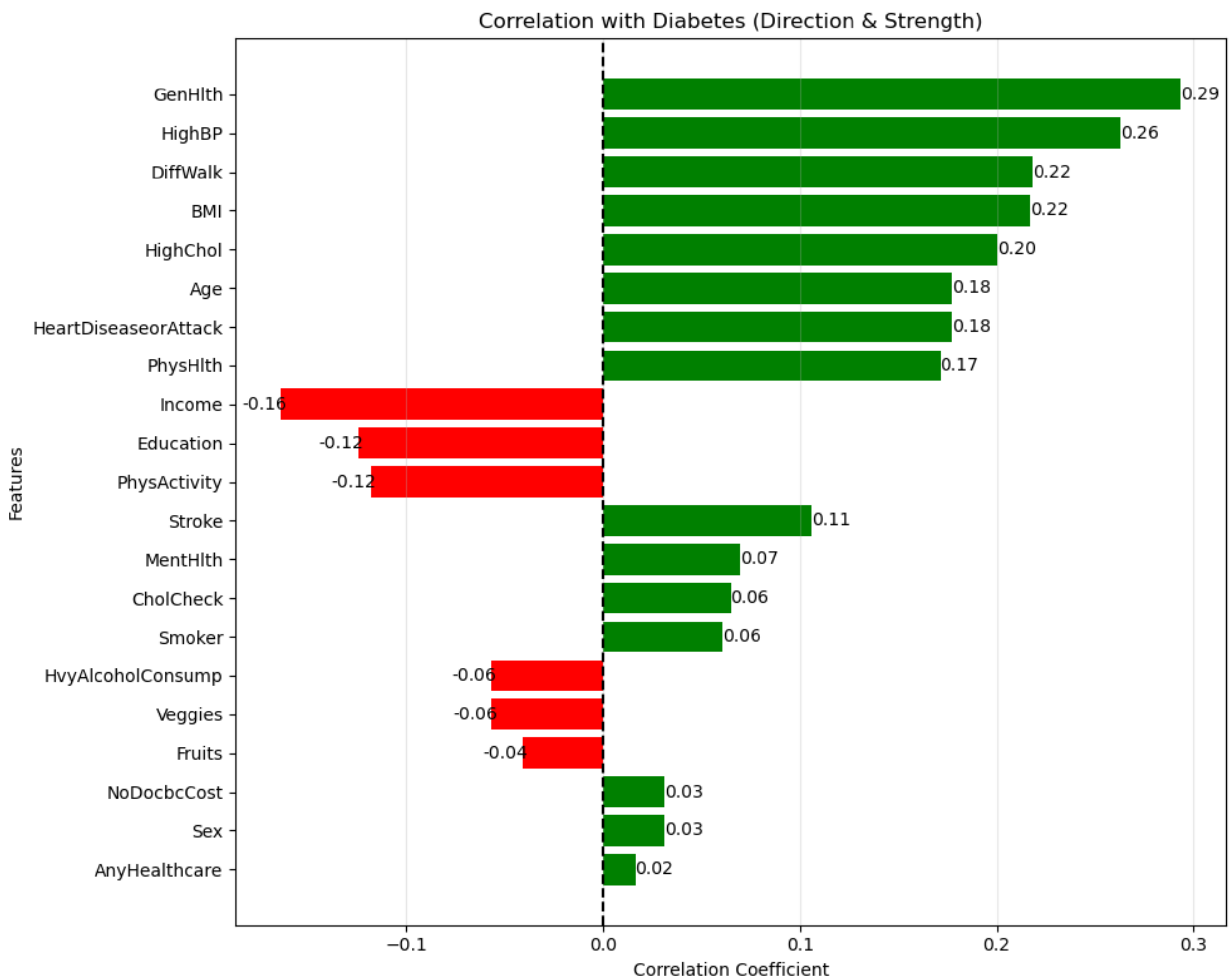


The analysis shows strong links between general health, physical health, and mobility issues, with income and age also playing key roles. These correlations highlight important diabetes risk factors while revealing some redundant features that may need combining or removing.

```
Top Correlated Column Pairs:
      Feature 1 Feature 2  Correlation
324     GenHlth  PhysHlth     0.524364
369    PhysHlth  DiffWalk     0.478417
388    DiffWalk   GenHlth     0.456920
461   Education    Income     0.449106
476      Income   GenHlth    -0.370014
346    MentHlth  PhysHlth     0.353619
419         Age    HighBP     0.344452
479      Income  DiffWalk    -0.320124
323     GenHlth  MentHlth     0.301674
309     GenHlth    HighBP     0.300530
```

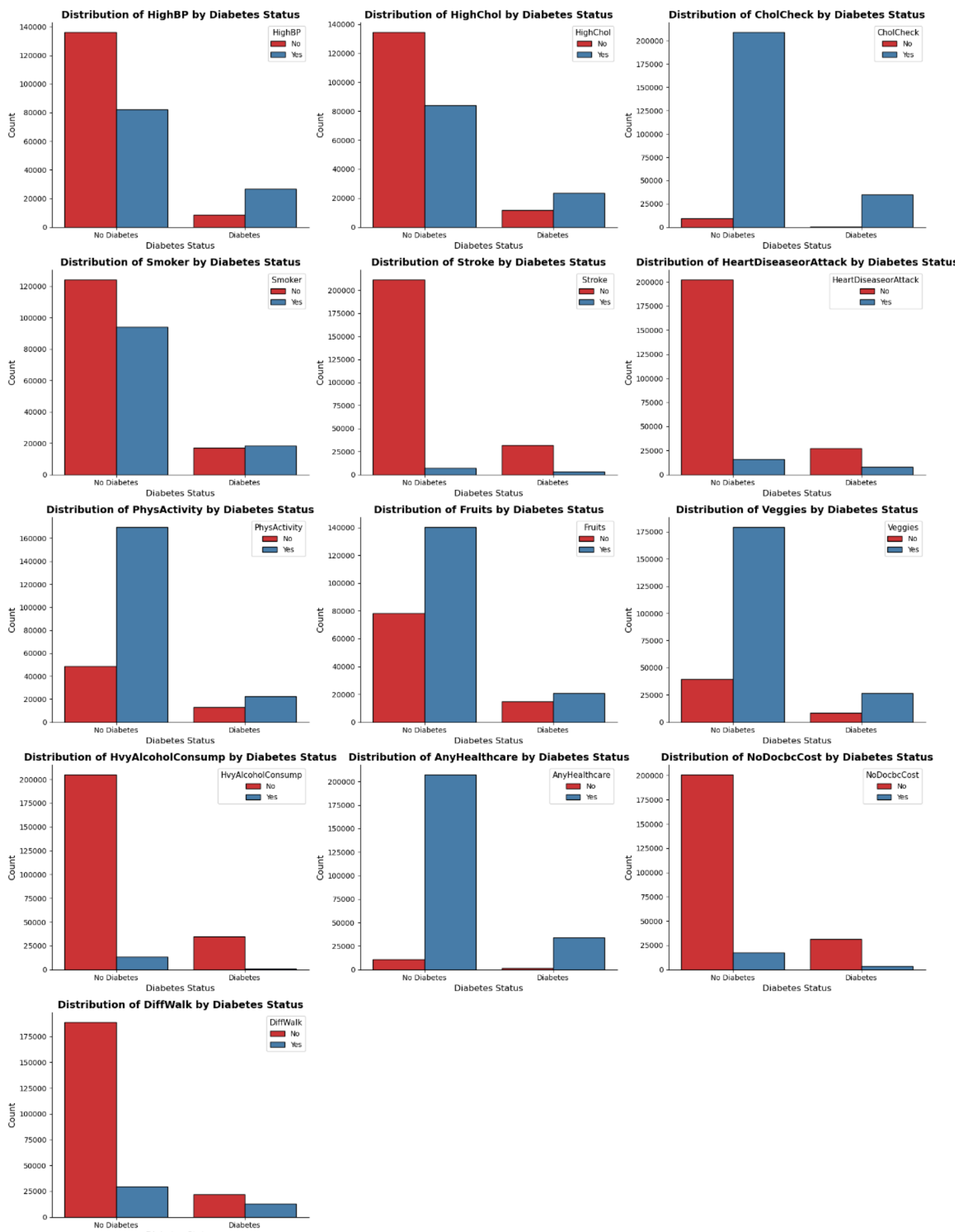
C- Diabetes and features correlation:

This graph for all features correlation for diabetes:

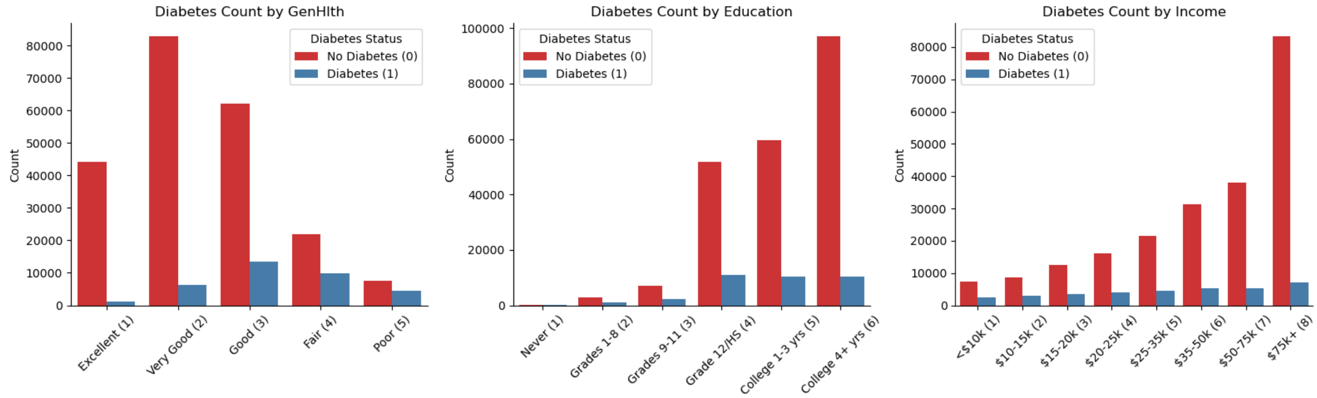


D-diabetes per each feature:

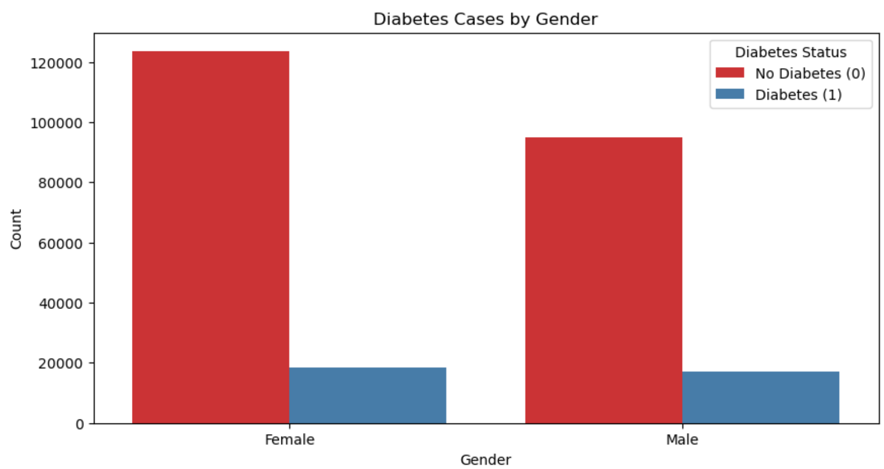
1- The graphs below show how specific habits and medical conditions correlate with diabetes in our dataset. Each visualization compares one of these binary features against diabetes status: **Medical Conditions**, **Health Habits**, **Access Barriers**, and **Mobility Issues**



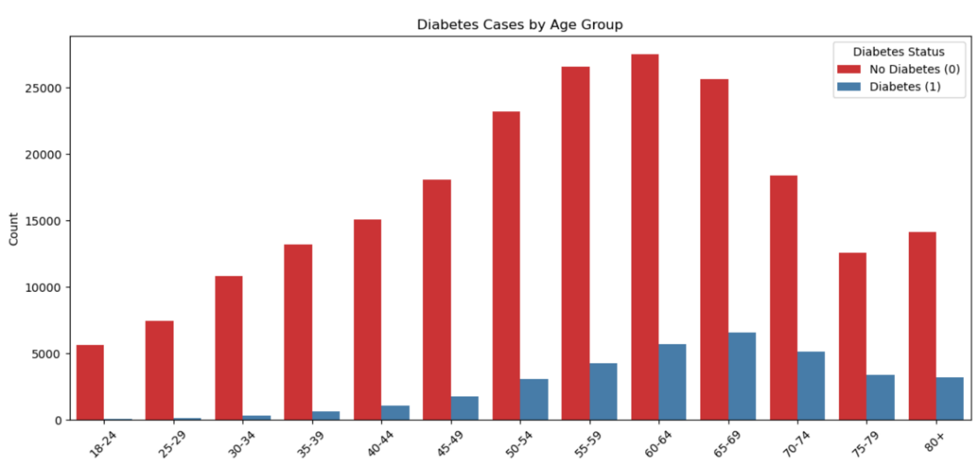
2- diabetes distribuition grouping by general health, education or income.



3- Does diabetes can be classified be gender? Such question could be very helpful for classification problem.



4- The graph show that the range(20-40) most people are classified as diabetic which align with BMI distribution over ages (more evidence for there strong correlation)



Models Analysis

All models are validated using cross validation of 5 folds

The balance range between the classes range between 1:6 as: $\frac{diabetic}{\text{not diabetic}} = \frac{1}{6}$

Outliers removal:

```python
def drop_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df_cleaned = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
    return df_cleaned

diabetes_cleaned = drop_outliers_iqr(diabetes, 'BMI')
diabetes_cleaned = drop_outliers_iqr(diabetes_cleaned, 'GenHlth')
```

Biased and erroric dataset problem:

Since some of the data was omited and other manipulated by the datset owner, the over all accuracy of all classes wasn't the best angle for addressing out problem. Essentially as this is a medical problem so we need to no ensure FN classification for diabetic class(if some one should be diagnosed as diabetic the model can't recklessly classify the patient as heathy but vice versa is alowed with some percent)

New goal: get good balance between diabetes class macro recall and over all accuracy.

Standard scaling of feature:

Since the data scale was different for all features, most models needs approximate equal ranges so standard scaling was essential.

There was 3 scaler that was tested against the dataset for analysis and picking the best fit for the data:

- Standardscaler (applied during the models training)
- MinMaxscaler
- RobustScaler

1- Baseline(ZeroR)

Before working on developing machine learning models, we used some dummy models as a baseline model to compare the performance of more complex models. By comparing the performance of a complex model to that of a simple model, we can determine if the complex model is actually providing useful predictions or if it is overfitting the data. Dummy models also help identify if the problem has any inherent bias or if the dataset is imbalanced. Overall, starting with a dummy model is a good way to get a baseline understanding of the data and the problem before moving on to more complex models. We have 2 strategies:

Actual classes distribution:

```
Actual class distribution:
Diabetes_binary
0.0    218334
1.0     35346
```

Most frequent:

Description: Always predicts the majority class

```
Classification Report:
              precision    recall  f1-score   support

 No Diabetes      0.862     1.000     0.926     65605
    Diabetes      0.000     0.000     0.000     10499

    accuracy                          0.862     76104
   macro avg      0.431     0.500     0.463     76104
weighted avg      0.743     0.862     0.798     76104
```

Uniform:

Description: Random predictions with equal probability

```
Classification Report:
              precision    recall  f1-score   support
...
   macro avg      0.501     0.502     0.425     76104
weighted avg      0.763     0.500     0.575     76104
```

2- Logistic regression

Model overview

Logistic regression is a powerful tool for predicting categorical outcomes. It is used in a wide variety of fields, including marketing, medicine, and finance. For example, logistic regression can be used to predict the likelihood that a customer will buy a product, the likelihood that a transaction to be fraud or not or the likelihood that a company will go bankrupt.

Advantages:

- ❖ It is a simple method to predict categorical outcomes.
- ❖ It can be used to predict the probability of an outcome for any given combination of predictor values.
- ❖ It is relatively easy to interpret the results of a logistic regression model.

Disadvantages:

- ❖ It can be sensitive to outliers in the data.
- ❖ It can be difficult to interpret the results of a logistic regression model when there are multiple independent variables.
- ❖ It can be computationally expensive to fit a logistic regression model with a large number of independent variables.

Overall, logistic regression is a powerful tool for predicting categorical outcomes. It is relatively easy to use and interpret, and it can be used in a wide variety of fields.

Model applying:

I. Using SMOTE for bias compensation:

SMOTE is used rebalaning the classes, and with the help of grid search to find the best parameters for the classification.

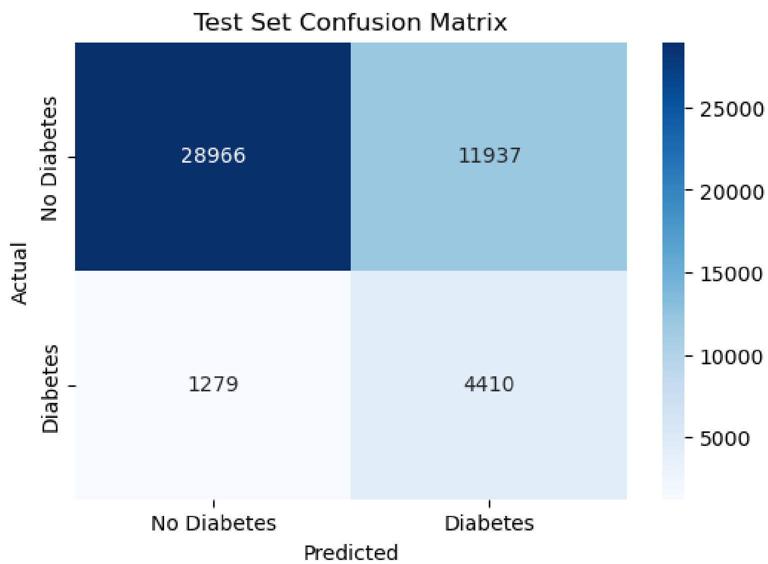
```
param_grid = [
    {
        'penalty': ['l1', 'l2'],
        'C': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0],
        'solver': ['liblinear', 'saga']
    },
    {
        'penalty': ['l2', 'l1'],
        'C': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0],
        'class_weight': ['balanced', {0:1, 1:6}],
        'solver': ['liblinear', 'saga']
    }
]
```

The grid seach for paramters result

Best Parameters: {'C': 0.0001, 'class_weight': {0: 1, 1: 6}, 'penalty': 'l1', 'solver': 'saga'}

These parameters result in test accuracy of *Test Accuracy: 36.25%* this contradicts the goal of balancing between the recall and overall accuracy. Hence the parameters were modified to:

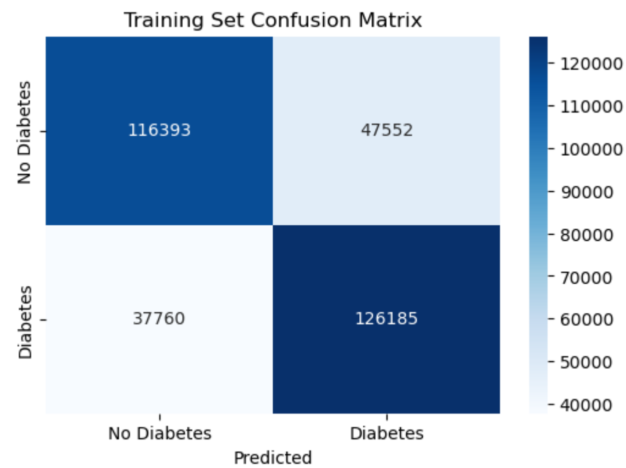
C=0.0001, penalty='l1', solver='saga' " the result of mode training:




```
Training Accuracy: 73.98%
Test Accuracy: 71.63%
              precision    recall  f1-score   support

 No Diabetes       0.96      0.71      0.81     40903
    Diabetes       0.27      0.78      0.40      5689

    accuracy                           0.72     46592
   macro avg       0.61      0.74      0.61     46592
weighted avg       0.87      0.72      0.76     46592
```



II. Training without SMOTE:

Another way of balancing the classes is to use the balance attribute of the logisitic regression.

The best parameters form the grid search are:

Best Parameters: {'C': 0.0001, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}

As its test accuray was *62.16%*

So the best paramenters achieve the goal.

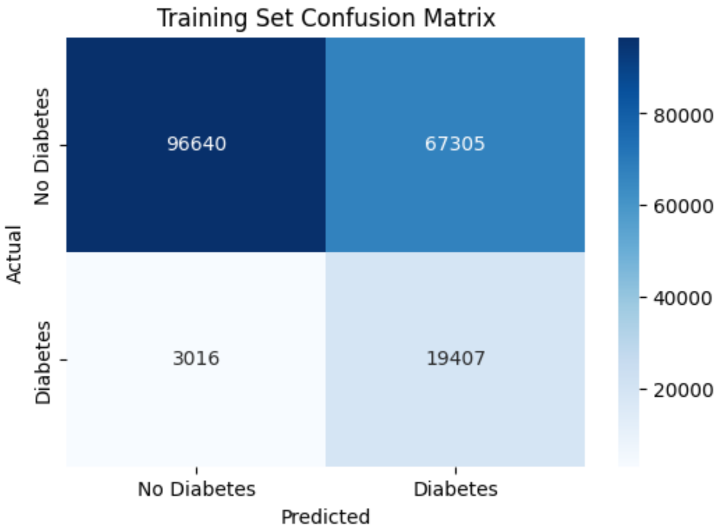
```
param_grid = [
    {
        'penalty': ['l1', 'l2'],
        'C': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0],
        'solver': ['liblinear', 'saga']
    },
    {
        'penalty': ['l2', 'l1'],
        'C': [0.0001, 0.001, 0.01, 0.1, 1.0, 10.0],
        'class_weight': ['balanced', {0:1, 1:6}],
        'solver': ['liblinear', 'saga']
    }
]
```

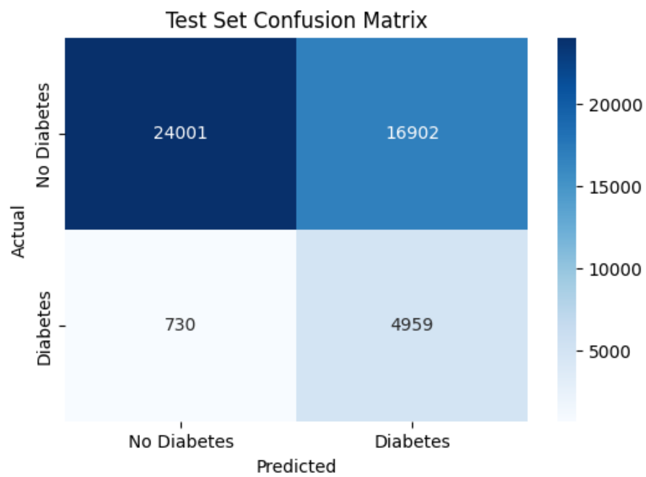
```
Training Accuracy: 62.27%
Test Accuracy: 62.16%

Classification Report (Test Set):
              precision    recall  f1-score   support

 No Diabetes       0.97      0.59      0.73     40903
    Diabetes       0.23      0.87      0.36      5689

    accuracy                           0.62     46592
   macro avg       0.60      0.73      0.55     46592
weighted avg       0.88      0.62      0.69     46592
```





comparing SMOTE:

removing SMOTE does help in increasing recall but at the price of decreasing the accuracy by almost same factor.

III. Binary classification threshord manipulation:

Decreasing the threshold magnifies the recall as prefered but diminishes the accuracy So its not our best option.

```
Training Accuracy with Threshold 0.4: 45.02%
Test Accuracy with Threshold 0.4: 44.87%

Classification Report (Test Set) with Threshold = 0.4
              precision    recall  f1-score   support

 No Diabetes       0.99      0.38      0.55     40903
    Diabetes       0.18      0.96      0.30      5689

    accuracy                           0.45     46592
   macro avg       0.58      0.67      0.42     46592
weighted avg       0.89      0.45      0.52     46592
```

IV. Train only using top 10 correlated features:

The result is almost the same as using the full dataset features

Conlusion: decreasing the dimensionality is a good choise.

```
Training Accuracy: 62.27%
Test Accuracy: 62.16%
              precision    recall  f1-score   support

 No Diabetes       0.97      0.59      0.73     40903
    Diabetes       0.23      0.87      0.36      5689

    accuracy                           0.62     46592
   macro avg       0.60      0.73      0.55     46592
weighted avg       0.88      0.62      0.69     46592
```

3- Preceptron Model:

Model overview

The perceptron is a fundamental binary linear classifier and the building block of neural networks. Introduced by Frank Rosenblatt in 1957, it serves as the simplest form of an artificial neuron, making it a key concept in machine learning and deep learning.

Advantages:

- Efficient for linearly separable problems with low computational cost.
- Forms the basis for more complex models (MLPs, deep learning).
- Can update weights incrementally with new data (useful for streaming data).

Disadvantages:

- Fails on non-linearly separable data (e.g., XOR problem).
- Unlike logistic regression, it doesn't provide class probabilities.
- Requires standardized/normalized data for stable training.
- If data isn't perfectly separable, the algorithm may not converge.

Model applying:

I. Using SMOTE for bias compensation:

SMOTE is used rebalaning the classes, and with the help of grid search to find the best parameters for the classification.

```
param_grid = [
    {
        'penalty': ['l1', 'l2', 'elasticnet'],
        'alpha': [0.0001, 0.001, 0.01, 0.1],
        'eta0': [0.001, 0.01, 0.1],
        'class_weight': ['balanced', None]
    }
]
```

The grid seach for paramters result

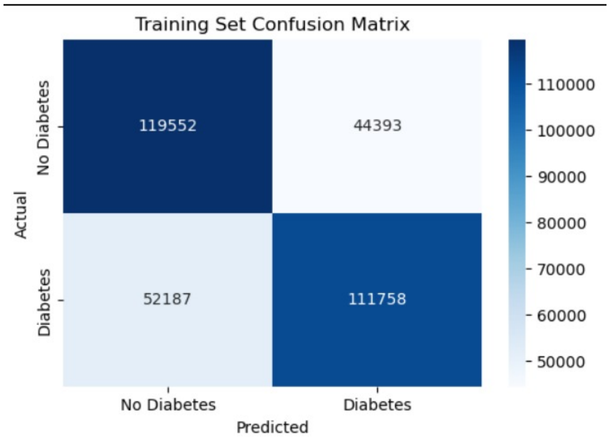
Best Parameters: {'alpha': 0.0001, 'class_weight': 'balanced', 'eta0': 0.1, 'penalty': 'l1'}

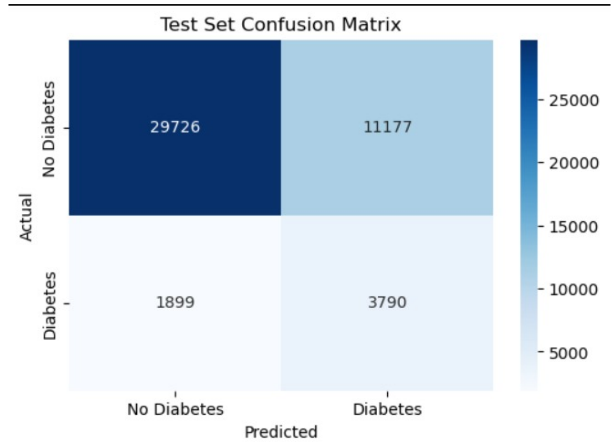
These parameters result in test accuracy of *71.94%*

the result of mode training:

```
Training Accuracy: 70.54%
Test Accuracy: 71.94%
```

	precision	recall	f1-score	support
No Diabetes	0.94	0.73	0.82	40903
Diabetes	0.25	0.67	0.37	5689
accuracy			0.72	46592
macro avg	0.60	0.70	0.59	46592
weighted avg	0.86	0.72	0.76	46592





II. Training without SMOTE:

Another way of balancing the classes is to use the balance attribute of the logisitic regression direclty.

```
param_grid = [
    {
        'penalty': ['l1', 'l2', 'elasticnet'],
        'alpha': [0.0001, 0.001, 0.01, 0.1],
        'eta0': [0.001, 0.01, 0.1],
        'class_weight': ['balanced', None]
    }
]
```

The best parameters form the grid search are:

Best Parameters: {'alpha': 0.01, 'class_weight': 'balanced', 'eta0': 0.1, 'penalty': 'l1'}

As its test accuray was *66.16% to increase the accuracy alittle bit the parameters are:*

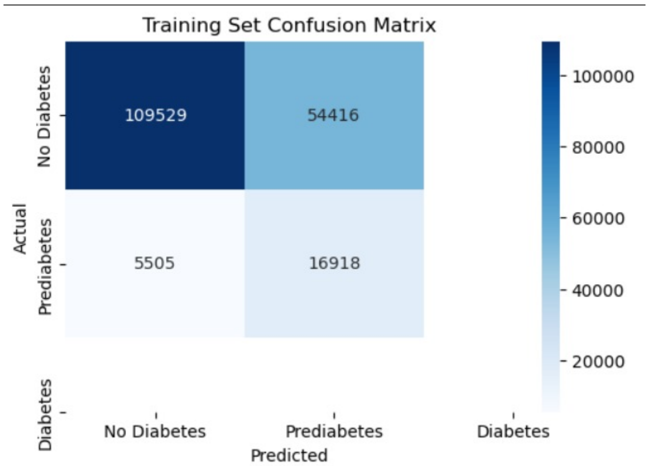
random_state=42, alpha=0.001, penalty='l1', eta0=0.1, class_weight='balanced'

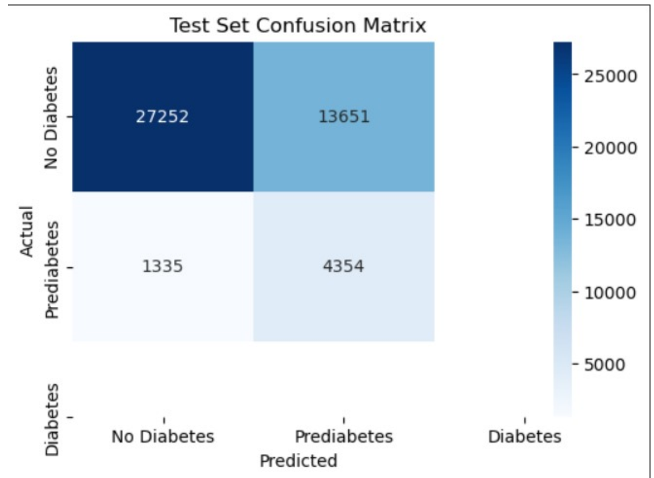
```
Training Accuracy: 67.85%
Test Accuracy: 67.84%

Classification Report (Test Set):
              precision    recall  f1-score   support

 No Diabetes       0.95      0.67      0.78     40903
    Diabetes       0.24      0.77      0.37      5689

    accuracy                           0.68     46592
   macro avg       0.60      0.72      0.58     46592
weighted avg       0.87      0.68      0.73     46592
```





comparing SMOTE:

removing SMOTE decreased the accuracy but increased the recall.

III. Binary classification threshord manipulation:

Decreasing the threshold magnifies the recall as prefered but diminishes the accuracy so its not our best option.

```
Training Accuracy with Threshold 0.1: 70.39%
Test Accuracy with Threshold 0.1: 70.58%

Classification Report (Test Set) with Threshold = 0.1
              precision    recall  f1-score   support

 No Diabetes       0.95      0.70      0.81     40903
    Diabetes       0.25      0.73      0.38      5689

    accuracy                           0.71     46592
   macro avg       0.60      0.72      0.59     46592
weighted avg       0.86      0.71      0.75     46592
```

IV. Train only using top 10 correlated features:

The result is almost the same as using the full dataset features

Conlusion: decreasing the dimensionality is a good choise.

```
Training Accuracy: 70.38%
Test Accuracy: 70.33%
              precision    recall  f1-score   support

 No Diabetes       0.95      0.69      0.80     40903
    Diabetes       0.26      0.76      0.39      5689

    accuracy                           0.70     46592
   macro avg       0.61      0.73      0.60     46592
weighted avg       0.87      0.70      0.75     46592
```

4- Random forest

Model overview

Random Forest is a popular machine learning algorithm that falls under the category of ensemble learning methods. It is a type of decision tree algorithm that generates multiple decision trees and combines their predictions to produce the final output.

Advantages:

- Random Forest has a high accuracy rate due to the combination of multiple decision trees.
- It is robust to outliers and noise in the dataset.
- Random Forest provides a measure of feature importance, which can be useful for feature selection and interpretation.
- It is able to handle large datasets and can be parallelized for faster processing.
- The combination of multiple decision trees reduces the risk of overfitting and increases generalization.

Disadvantages:

- Random Forest models are often difficult to interpret due to their complexity and the number of decision trees.
- The training and prediction process of Random Forest can be computationally expensive, especially for large datasets.
- The memory usage of Random Forest can be high due to the number of decision trees.
- Random Forest can be biased towards the majority class in imbalanced datasets, leading to lower accuracy for the minority class.

Model applying:

I. Using SMOTE for bias compensation:

SMOTE is used rebalaning the classes, and with the help of grid search to find the best parameters for the classification. The grid seach for paramters result:

Best Parameters: {'class_weight': 'balanced', 'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 200}

Test Accuracy: 82.06%

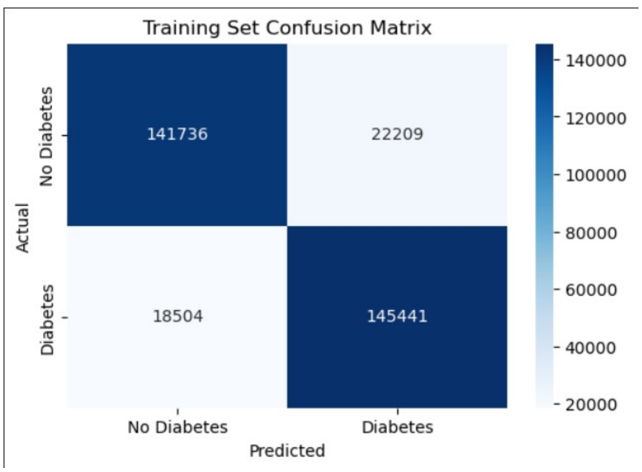
```
rf_param_grid = [
    {
        'n_estimators': [100, 200],
        'max_depth': [10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [2, 4],
        'class_weight': [None, 'balanced'],
    },
    {
        'n_estimators': [100, 200],
        'max_features': ['sqrt', 'log2'],
        'bootstrap': [True, False],
        'class_weight': [None, 'balanced'],
    }
]
```

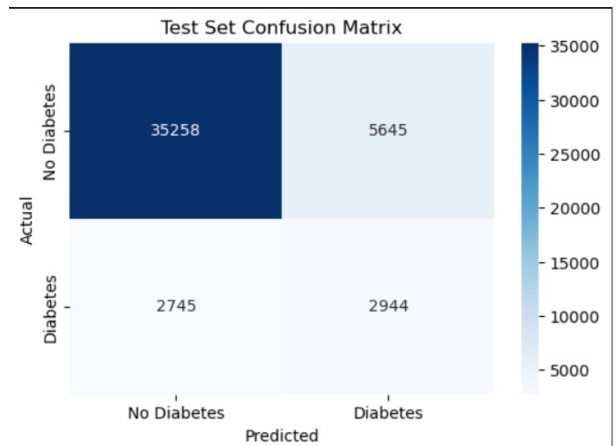
the result of mode training:

```
Training Accuracy: 87.58%
Test Accuracy: 81.99%
              precision    recall  f1-score   support

 No Diabetes       0.93      0.86      0.89     40903
    Diabetes       0.34      0.52      0.41      5689

    accuracy                           0.82     46592
   macro avg       0.64      0.69      0.65     46592
weighted avg       0.86      0.82      0.83     46592
```





II. Training without SMOTE:

Another way of balancing the classes is to use the balance attribute of the sklearn random forest direclty. The grid parametes are the same as SMOTE grid params

The best parameters form the grid search are:

Best Parameters: {class_weight='balanced', criterion='entropy', max_depth=10, min_samples_leaf=4, min_samples_split=10,n_estimators=100 }

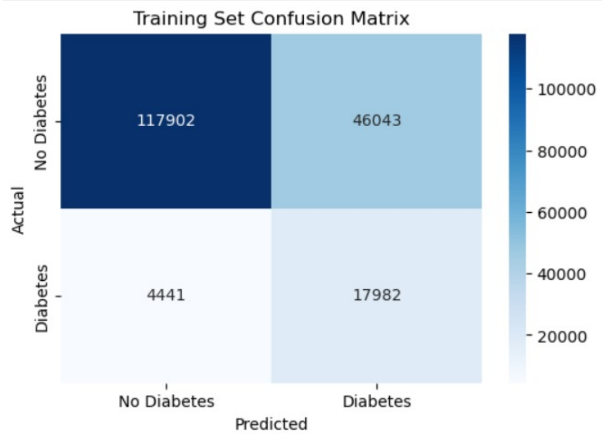
As its test accuray was *72.36% (meets the goal)*

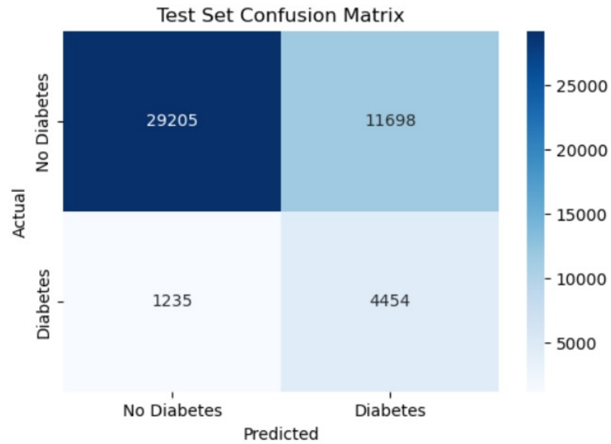
```
Training Accuracy: 72.98%
Test Accuracy: 72.36%

Classification Report (Test Set):
              precision    recall  f1-score   support

 No Diabetes       0.96      0.72      0.82     40903
    Diabetes       0.28      0.78      0.41      5689

    accuracy                           0.72     46592
   macro avg       0.62      0.75      0.61     46592
weighted avg       0.88      0.72      0.77     46592
```





III. Binary classification threshord manipulation:

Decreasing the threshold magnifies the recall as prefered but diminishes the accuracy so its not our best option.

```
Training Accuracy with Threshold 0.4: 63.83%
Test Accuracy with Threshold 0.4: 63.35%

Classification Report (Test Set) with Threshold = 0.4
              precision    recall  f1-score   support

 No Diabetes       0.97      0.60      0.74     40903
    Diabetes       0.23      0.88      0.37      5689

    accuracy                           0.63     46592
   macro avg       0.60      0.74      0.56     46592
weighted avg       0.88      0.63      0.70     46592
```

IV. Train only using top 10 correlated features:

The result is almost the same as using the full dataset features

Conlusion: decreasing the dimensionality is a good choise.

```
Training Accuracy: 72.80%
Test Accuracy: 72.16%
              precision    recall  f1-score   support

 No Diabetes       0.96      0.71      0.82     40903
    Diabetes       0.27      0.78      0.41      5689

    accuracy                           0.72     46592
   macro avg       0.62      0.75      0.61     46592
weighted avg       0.88      0.72      0.77     46592
```

5- SVM:

Model overview

Support Vector Machines (SVMs) are a class of supervised learning algorithms used for classification and regression analysis. SVMs work by finding the hyperplane that best separates the data into different classes. The optimal decision boundary is the hyperplane that maximizes the margin between the two classes. In the case where the data is not linearly separable, SVMs use a kernel trick to map the data into a higher-dimensional space where the data can be linearly separated.

Advantages:

- SVMs can perform well in high-dimensional spaces, making them useful for solving complex problems with many features.
- SVMs are less prone to overfitting than other classification algorithms because they try to maximize the margin between classes, which helps prevent the model from being too closely fit to the training data.
- SVMs can be used for both linear and nonlinear classification and regression tasks, and different kernel functions can be used to handle different types of data.
- SVMs are computationally efficient and can work well with small and large datasets.

Disadvantages:

- SVM performance depends heavily on the choice of kernel, which can be challenging to choose correctly.
- SVMs are sensitive to the scale of the input features, so data preprocessing is required to standardize the features.
- SVMs can be computationally intensive, particularly for large datasets and complex kernel functions.
- SVMs are designed for binary classification problems, which means they need to be modified for multi-class classification tasks.

Model applying:

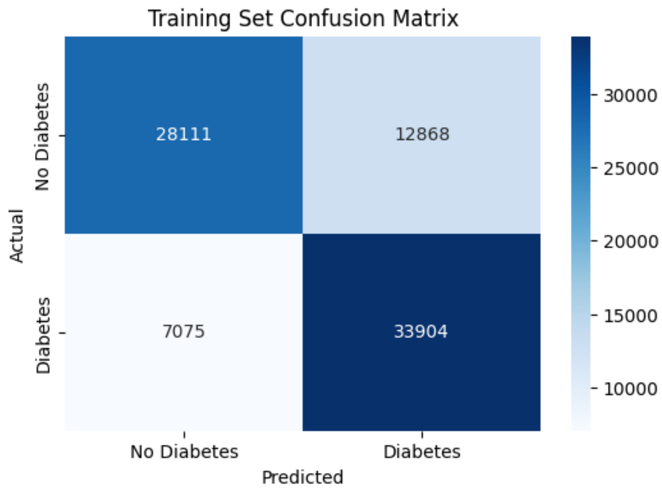
Given that SVM models are computationally intensive to train, using one device to find good parameters is inefficient. So the parameters search was distributed over multiple devices. The best reached paramters were selected as the best parameters for the data.

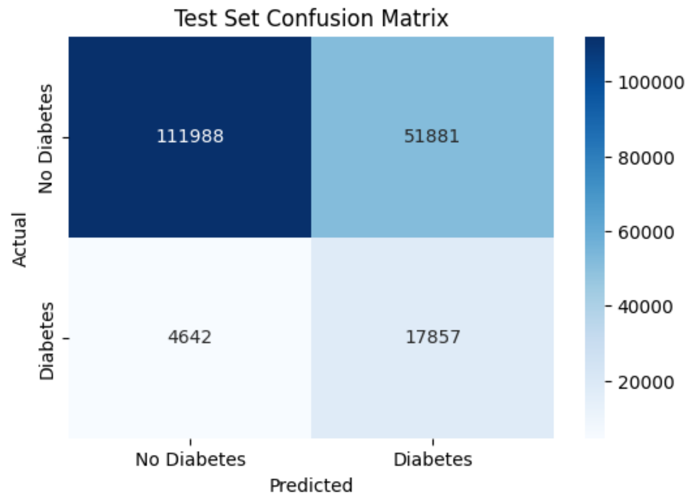
I. Using SMOTE for bias compensation:

```
Training Accuracy: 75.67%
Test Accuracy: 69.67%
              precision    recall  f1-score   support

 No Diabetes       0.96      0.68      0.80    163869
    Diabetes       0.26      0.79      0.39     22499

    accuracy                           0.70    186368
   macro avg       0.61      0.74      0.59    186368
weighted avg       0.88      0.70      0.75    186368
```





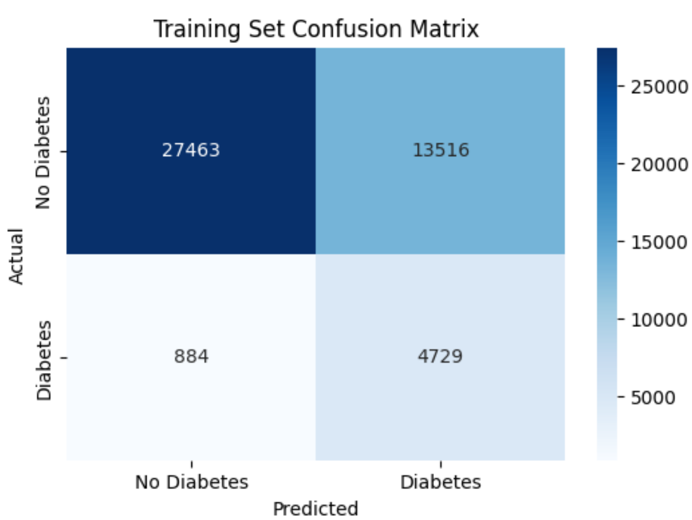
II. Training without SMOTE:

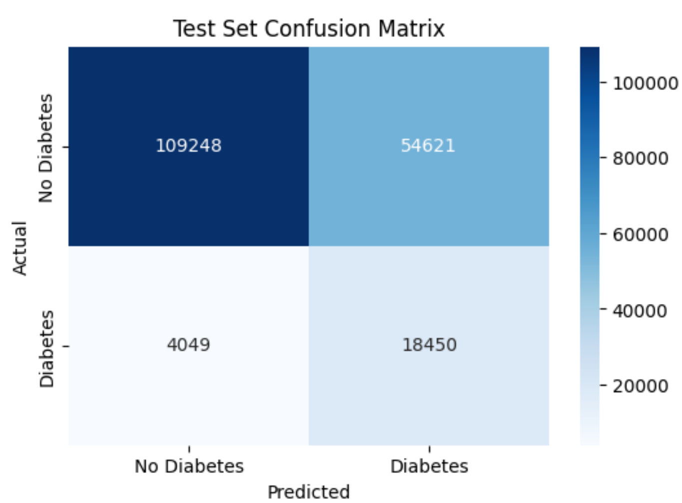
```
Training Accuracy: 69.09%
Test Accuracy: 68.52%

Classification Report (Test Set):
              precision    recall  f1-score   support

 No Diabetes       0.96      0.67      0.79    163869
    Diabetes       0.25      0.82      0.39     22499

    accuracy                           0.69    186368
   macro avg       0.61      0.74      0.59    186368
weighted avg       0.88      0.69      0.74    186368
```





III. Binary classification threshord manipulation:

```
Training Accuracy with Threshold 0.4: 87.71%
Test Accuracy with Threshold 0.4: 87.65%

Classification Report (Test Set) with Threshold = 0.4
              precision    recall  f1-score   support

 No Diabetes       0.89      0.99      0.93    163869
    Diabetes       0.43      0.07      0.12     22499

    accuracy                           0.88    186368
   macro avg       0.66      0.53      0.53    186368
weighted avg       0.83      0.88      0.84    186368
```

IV. Train only using top 10 correlated features:

```
Training Accuracy: 69.11%
Test Accuracy: 68.67%
              precision    recall  f1-score   support

 No Diabetes       0.96      0.67      0.79    163869
    Diabetes       0.25      0.80      0.38     22499

    accuracy                           0.69    186368
   macro avg       0.61      0.74      0.59    186368
weighted avg       0.88      0.69      0.74    186368
```