An Application of Decision Tree Predictor

1. Introduction

1.1 What is Decision Tree

Employing the tree data structure, decision tree is a hierarchical structure and a supervised learning method (Suthaharan, 2016, pp. 237-269), where the branch notes show the path of a decision point to another, and leaf notes suggests the final decisions (Song and Lu, 2015). Decision tree method is often used to identify features and patterns for making predictions and classifications when it comes to large data (Myles, 2004). Fundamentally, it uses set theory to split data into different sets or subsets for purposes such as training and testing. Random Forest, as an ensemble learning method, and a combination classifier which combines different decision trees together (Efron, Tibshirani, 1994) to realize more precise outcomes. With the advancement of machine learning algorithms, it is common to apply such methods to fields like medical health, stock predictions, transportation, marketing strategies, etc., and almost all of these areas are all closely connected with our everyday life. It is encouraging for us to explore what we are concerned about in our lives by further working on these approaches.

1.2 The Question of Predicting Diabetes

As one of the most significant health problems in the world, around 1.5 million deaths are caused by diabetes every year, according to the World Health Organization. As a group where all members are interested in healthcare problems, when we were approaching the health topic, it was unsurprising to discover that there are at least two family members or known relatives of our group members suffering from diabetes. As a chronic disease, it is not just an increase in glucose. We have seen patients with diabetes not only have to pay attention to everything or every environment and regulate dietary for every single day, and get regular treatment, but also struggles to prevent from or fight with its complications such as heart and kidney diseases, visual impairment and neurological damage, which was often described by them as “walking with the death”.

However, it seems so common but most of them overlook their body indicators. Statistics show that every one in two adults suggests this disease without knowing their conditions (International Diabetes Federation, 2023), so the prediction on diabetes to improve early diabetes diagnosis and personalized treatment has come to our attention. We intend to combine what we learned about sets, trees, with probability science to design machine learning models with Random Forest classifier and Naïve Bayes classifier to predict whether

a person has diabetes. We want to study different kinds of models and compare these two to choose the more accurate one. Therefore, in this report, we will explain how we explain our methods and the data analysis and present the code and models we built with the final conclusion.

1.3 Project Progress

Having decided on our topic of predicting whether a patient has diabetes or not, we started searching a large portion of data so that we are able to train our model for an accurate result. We collected a dataset regarding Diabetes published by the National Institute of Diabetes and Digestive and Kidney Diseases. This data contains 8 health-related attributes of pregnancies, glucose levels, blood pressure, skin thickness, insulin levels, age, and BMI.

With the application of sets, trees and probability, we will focus on the using Random Forest classifier and Naïve Bayes classifier for our decision tree predictor to approach this dataset. After cleaning the data, we split our data into training and test sets and trained the models and compared their accuracy. For all these processes, we used the software Python and applied its library scikit-learn for the implementation.

2. Analysis

2.1 Random Forest classifier

Decision trees are a common method in machine learning. They remain unaffected by feature scaling and various transformations, and they are robust to irrelevant features, generating interpretable models (Ho, 1998). However, they often lack accuracy. In particular, deep trees can overfit, capturing highly irregular patterns and exhibiting low bias and high variance on the training set. Random Forest is a method that averages multiple deep decision trees to reduce variance. In this approach, decision trees are trained on different subsets of a dataset. This comes at a slight increase in bias and some loss of interpretability but often significantly improves overall performance in the final model.

The training algorithm for Random Forest applies the general algorithm of bagging to tree learning. Given a training set X = x1, ..., xn and target Y = y1, ..., yn, the bagging method iteratively samples with replacement B times from the training set and then trains tree models on these samples:
For $b = 1, ..., B$:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
2. Train a classification or regression tree f_b on X_b, Y_b. (Ho, 2002)

After training, predictions for an unknown sample x can be obtained by averaging predictions from all individual regression trees:

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b\,(x')$$

This bagging method reduces variance without increasing bias, leading to improved performance. Even if predictions from individual trees are sensitive to noise in the training set, this issue does not arise when using multiple trees as long as these trees are uncorrelated.

Furthermore, the standard deviation of predictions from all individual regression trees on sample x' can serve as an estimate of prediction uncertainty:

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}\left(f_b(x') - \hat{f}\right)^2}{B-1}}$$

The number of samples or trees, denoted by B, is a hyperparameter. Typically, several hundred to several thousand trees are used, depending on the size and nature of the training set.

In this project, we applied the random forest classifier from library Sklearn and achieved high accuracy.

2.2 Naïve Bayes classifier

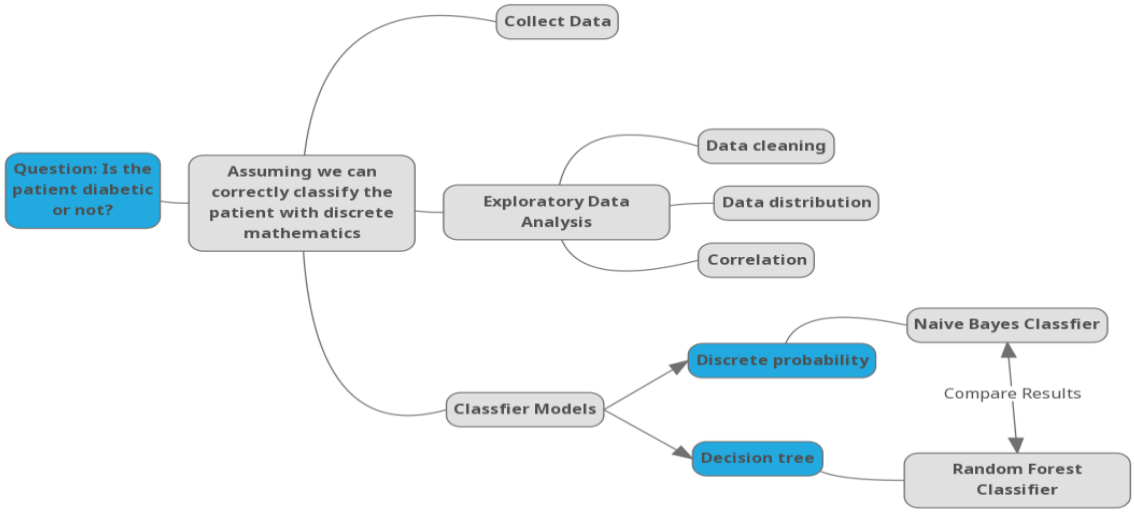
The Naïve Bayes Model (NBM) is based on Bayes' Rule with the hypothesis of variables are independent. For a given training dataset, firstly we train the combined probability distribution of input and output and build up the model. Secondly, based on this model, for a given input x, we use NBM to find out the output y with the highest probability of post-test.

$$P(x|y\ ) = P(x_1|y)xP(x_2|y)x \cdots P(x_n|y)$$

In this project, we also used the Bernoulli Naïve Bayes Classifier, a variant of the Naïve Bayes classifier, is particularly to uncover the Boolean feature. It provided valuable insights into its performance for predicting diabetes based on the dataset's attributes. With an achieved accuracy of 0.65, the Bernoulli Naïve Bayes model demonstrated moderate success in classifying individuals as diabetic or non-diabetic. However, compared to the Random Forest Classifier, its performance was notably lower. The model's lower accuracy suggests potential limitations in capturing the complex relationships among the health-related features present in the dataset. While the Naïve Bayes algorithm assumes independence between features, the nature of health attributes might not entirely adhere to this assumption, affecting the model's predictive power. Furthermore, the removal of outliers, especially in crucial variables such as BMI, glucose, and blood pressure, might have influenced the model's performance negatively. Despite its limitations, the analysis using the Bernoulli Naïve Bayes Classifier provides valuable insights into the challenges and nuances of applying probabilistic models in

healthcare-related predictive analytics, highlighting areas for further exploration and model refinement.

2.3 Strategy



2.4 Exploratory data analysis

EDA (Exploratory data analysis) acts as a foundational step in the machine learning pipeline, laying the groundwork for creating models by enhancing understanding, identifying patterns, and guiding data preprocessing and modeling decisions. In this project we used Jupyter Notebook to prompt codes for EDA.

The libraries we used in EDA:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Firstly, we finished some basic data exploration about the dataset, which includes showing the head of the dataset, finding out the shape and size of the data and renaming the output column.

```
df = pd.read_csv('Healthcare-Diabetes.csv')
df.head()
```

	Id	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1	6	148	72	35	0	33.6	0.627	50	1
1	2	1	85	66	29	0	26.6	0.351	31	0
2	3	8	183	64	0	0	23.3	0.672	32	1
3	4	1	89	66	23	94	28.1	0.167	21	0
4	5	0	137	40	35	168	43.1	2.288	33	1

```
print('Shape :',df.shape)
print('Size :',df.size)
```

```
Shape : (2768, 10)
Size : 27680
```

```
df.rename(columns = {'Outcome' : 'Diabetic'}, inplace = True)
df.head(5)
```

	Id	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Diabetic
0	1	6	148	72	35	0	33.6	0.627	50	1
1	2	1	85	66	29	0	26.6	0.351	31	0
2	3	8	183	64	0	0	23.3	0.672	32	1
3	4	1	89	66	23	94	28.1	0.167	21	0
4	5	0	137	40	35	168	43.1	2.288	33	1

2.4.1 Data Cleaning

Data cleaning is a fundamental aspect of Exploratory Data Analysis (EDA) that involves identifying and rectifying errors, inconsistencies, and outliers in the dataset. It's essential for preparing data to be used effectively in machine learning models. We have implemented several key steps involved in data cleaning during EDA:

(1) Handling missing values and duplicate values:
Identifying and dealing with missing data is crucial. Techniques like imputation (filling missing values based on other data) or deletion (removing rows or columns with missing values) are employed based on the nature and extent of missingness.
The code below calculates the total number of nulls in each column and checks for duplicates.

```
df.isna().sum()
```

```
Id                        0
Pregnancies               0
Glucose                   0
BloodPressure             0
SkinThickness             0
Insulin                   0
BMI                       0
DiabetesPedigreeFunction  0
Age                       0
Diabetic                  0
dtype: int64
```

```
df.duplicated().sum()
```

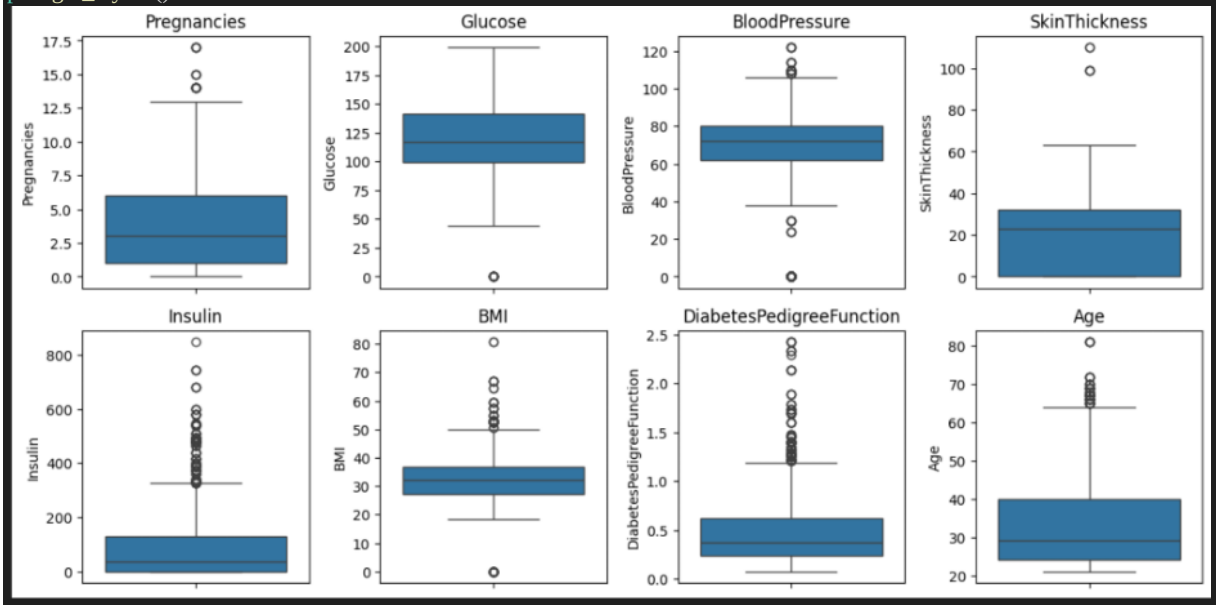
```
0
```

The results show that there are no missing or duplicate values in the dataset.

(2) Dealing with outliers:
Outliers can significantly impact model performance. EDA involves detecting outliers through statistical methods or visualization techniques and deciding whether to remove, transform, or handle them separately based on domain knowledge.
In our project, we used boxplot in library seaborn and matplotlib to visualize the outliers.

```
fig, axes = plt.subplots(2,4, figsize = (12,6))
```

```
sns.boxplot(df['Pregnancies'], ax = axes[0,0])
axes[0,0].set_title('Pregnancies')
sns.boxplot(df['Glucose'], ax = axes[0,1])
axes[0,1].set_title('Glucose')
sns.boxplot(df['BloodPressure'], ax = axes[0,2])
axes[0,2].set_title('BloodPressure')
sns.boxplot(df['SkinThickness'], ax = axes[0,3])
axes[0,3].set_title('SkinThickness')
sns.boxplot(df['Insulin'], ax = axes[1,0])
axes[1,0].set_title('Insulin')
sns.boxplot(df['BMI'], ax = axes[1,1])
axes[1,1].set_title('BMI')
sns.boxplot(df['DiabetesPedigreeFunction'], ax = axes[1,2])
axes[1,2].set_title('DiabetesPedigreeFunction')
sns.boxplot(df['Age'], ax = axes[1,3])
axes[1,3].set_title('Age')
plt.tight_layout()
```



The results show that all the variables contain outliers. In the cases of BMI, Glucose, and BloodPressure, outliers are observed with values equal to 0, which is implausible. Therefore, due to its unusual behavior we will remove these observations from the dataset. The code below is an example of variable Glucose.

```
# the variable glucose doesn't contain any outlier above the upper quartile.
# Therefore, we will only remove the outlier present below the lower quartile.
q1 = df['Glucose'].quantile(.25)
q3 = df['Glucose'].quantile(0.75)
iqr = q3-q1
lower_quartile = q1 - (iqr * 1.5)
lower_array = np.where(df['Glucose']<=lower_quartile)[0]
df.drop(index=lower_array).reset_index(inplace = True)
df.shape
```

```
(2768, 10)
```

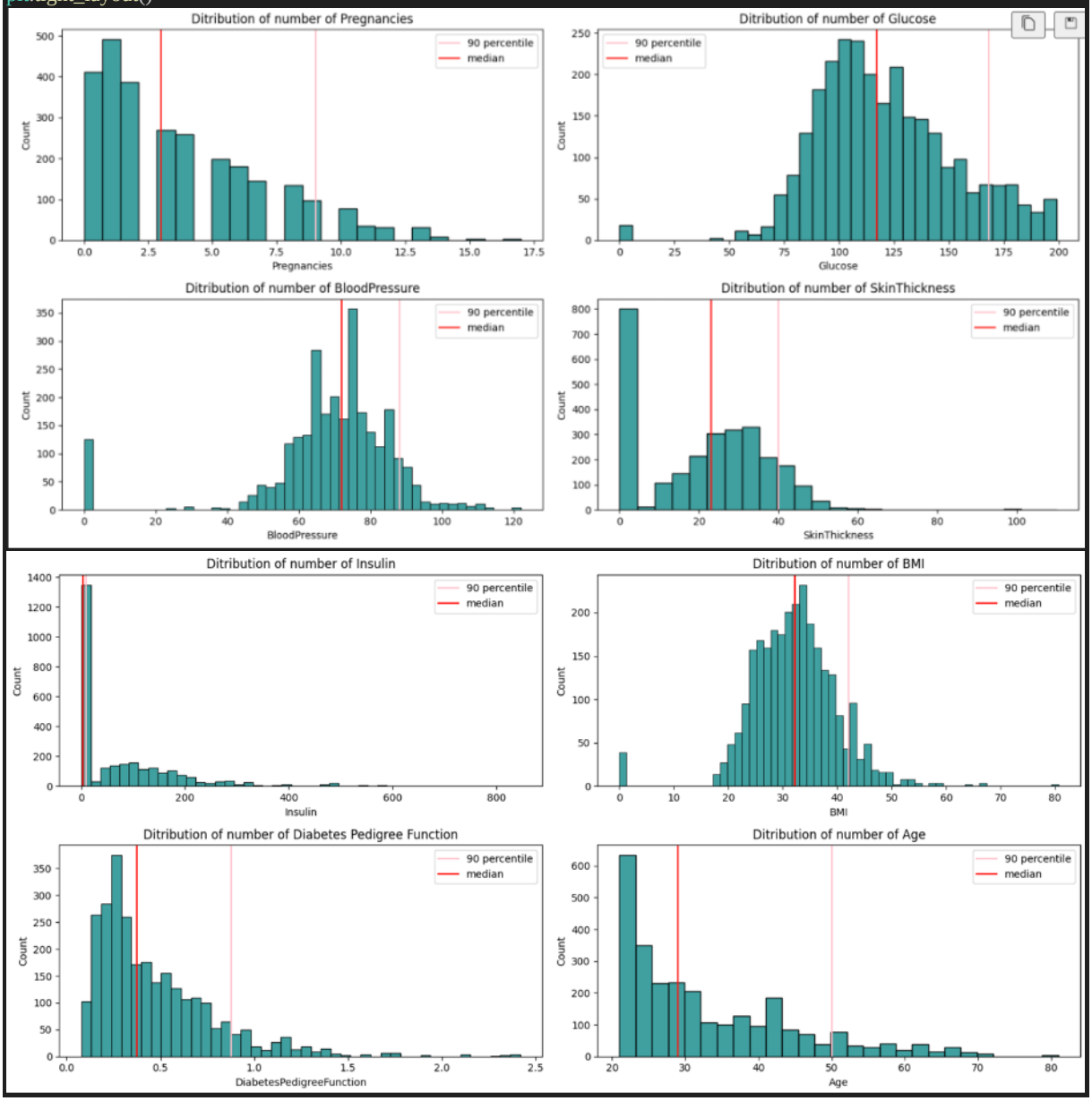
2.4.2 Distribution analysis

Understanding data distributions through EDA is crucial for several reasons:
Model Selection: Different machine learning models assume different distributions. Knowing the data distribution helps in selecting models that align well with the data.

Feature Engineering: Understanding distributions aids in creating new features or transforming existing ones to better suit model assumptions and improve performance.
Data Preprocessing: Handling skewed data, outliers, or non-normal distributions might require specific preprocessing techniques for optimal model training.
In our project, we used histplot in library seaborn and matplotlib to visualize distribution analysis.

```
fig, axes = plt.subplots(4,2, figsize = (15,15))
sns.histplot(df['Pregnancies'], ax = axes[0,0], color = 'teal')
axes[0,0].set_title('Ditribution of number of Pregnancies')
axes[0,0].axvline(df['Pregnancies'].quantile(.90), color = 'pink', label = '90 percentile')
axes[0,0].axvline(df['Pregnancies'].quantile(.50), color = 'red', label = 'median')
axes[0,0].legend()

sns.histplot(df['Glucose'], ax = axes[0,1],color = 'teal')
axes[0,1].set_title('Ditribution of number of Glucose')
axes[0,1].axvline(df['Glucose'].quantile(.90), color = 'pink', label = '90 percentile')
axes[0,1].axvline(df['Glucose'].quantile(.50), color = 'red', label = 'median')
axes[0,1].legend()

sns.histplot(df['BloodPressure'], ax = axes[1,0], color = 'teal')
axes[1,0].set_title('Ditribution of number of BloodPressure')
axes[1,0].axvline(df['BloodPressure'].quantile(.90), color = 'pink', label = '90 percentile')
axes[1,0].axvline(df['BloodPressure'].quantile(.50), color = 'red', label = 'median')
axes[1,0].legend()

sns.histplot(df['SkinThickness'], ax = axes[1,1], color = 'teal')
axes[1,1].set_title('Ditribution of number of SkinThickness')
axes[1,1].axvline(df['SkinThickness'].quantile(.90), color = 'pink', label = '90 percentile')
axes[1,1].axvline(df['SkinThickness'].quantile(.50), color = 'red', label = 'median')
axes[1,1].legend()

sns.histplot(df['Insulin'], ax = axes[2,0], color = 'teal')
axes[2,0].set_title('Ditribution of number of Insulin')
axes[2,0].axvline(df['Pregnancies'].quantile(.90), color = 'pink', label = '90 percentile')
axes[2,0].axvline(df['Pregnancies'].quantile(.50), color = 'red', label = 'median')
axes[2,0].legend()

sns.histplot(df['BMI'], ax = axes[2,1], color = 'teal')
axes[2,1].set_title('Ditribution of number of BMI')
axes[2,1].axvline(df['BMI'].quantile(.90), color = 'pink', label = '90 percentile')
axes[2,1].axvline(df['BMI'].quantile(.50), color = 'red', label = 'median')
axes[2,1].legend()

sns.histplot(df['DiabetesPedigreeFunction'], ax = axes[3,0], color = 'teal')
axes[3,0].set_title('Ditribution of number of Diabetes Pedigree Function')
axes[3,0].axvline(df['DiabetesPedigreeFunction'].quantile(.90), color = 'pink', label = '90 percentile')
axes[3,0].axvline(df['DiabetesPedigreeFunction'].quantile(.50), color = 'red', label = 'median')
axes[3,0].legend()
```

```
sns.histplot(df['Age'], ax = axes[3,1], color = 'teal')
axes[3,1].set_title('Ditribution of number of Age')
axes[3,1].axvline(df['Age'].quantile(.90), color = 'pink', label = '90 percentile')
axes[3,1].axvline(df['Age'].quantile(.50), color = 'red', label = 'median')
axes[3,1].legend()

plt.tight_layout()
```



Based on the histograms provided in the two images, we can observe the following distribution patterns for each variable:

(1) Number of Pregnancies:

- The majority of counts are for fewer pregnancies, with a peak at 0-2.5 range, indicating most subjects have had few or no pregnancies.
- There is a right-skewed distribution, with a tail extending towards higher numbers of pregnancies, but these cases are less frequent.

- The median number of pregnancies is below the 90th percentile, showing that over half the subjects have a number of pregnancies less than the median value.

(2) Glucose:

- The distribution appears to be roughly normal (bell-shaped) with a slight right skew.
- The peak count is around the 100 glucose level.
- The median glucose level is left of the 90th percentile, suggesting that over half of the individuals have glucose levels below this median.

(3) Blood Pressure:

- The distribution has a normal appearance with a peak at around 60-80 range.
- The tail is less pronounced compared to the number of pregnancies, indicating fewer outliers in blood pressure measurements.
- The median blood pressure is less than the 90th percentile.

(4) Skin Thickness:

- This variable shows a right-skewed distribution with most counts concentrated at the lower end.
- There are fewer individuals with higher skin thickness measurements.
- The median is well below the 90th percentile.

(5) Insulin:

- The distribution is highly right-skewed and clusterd
- A large number of counts are at the lower end, near 0, suggesting many individuals have low insulin measurements.
- The median is very close to the lower end, and the 90th percentile is significantly higher, indicating a small proportion of individuals with very high insulin levels.

(6) BMI (Body Mass Index):

- The distribution is somewhat normal but with a right skew.
- The majority of the BMI values are concentrated around the 20-40 range.
- The median BMI is below the 90th percentile.

(7) Diabetes Pedigree Function:

- The distribution is right-skewed.
- Most individuals have a diabetes pedigree function value at the lower end, indicating a lower genetic predisposition to diabetes.
- The median is significantly lower than the 90th percentile.

(8) Age:

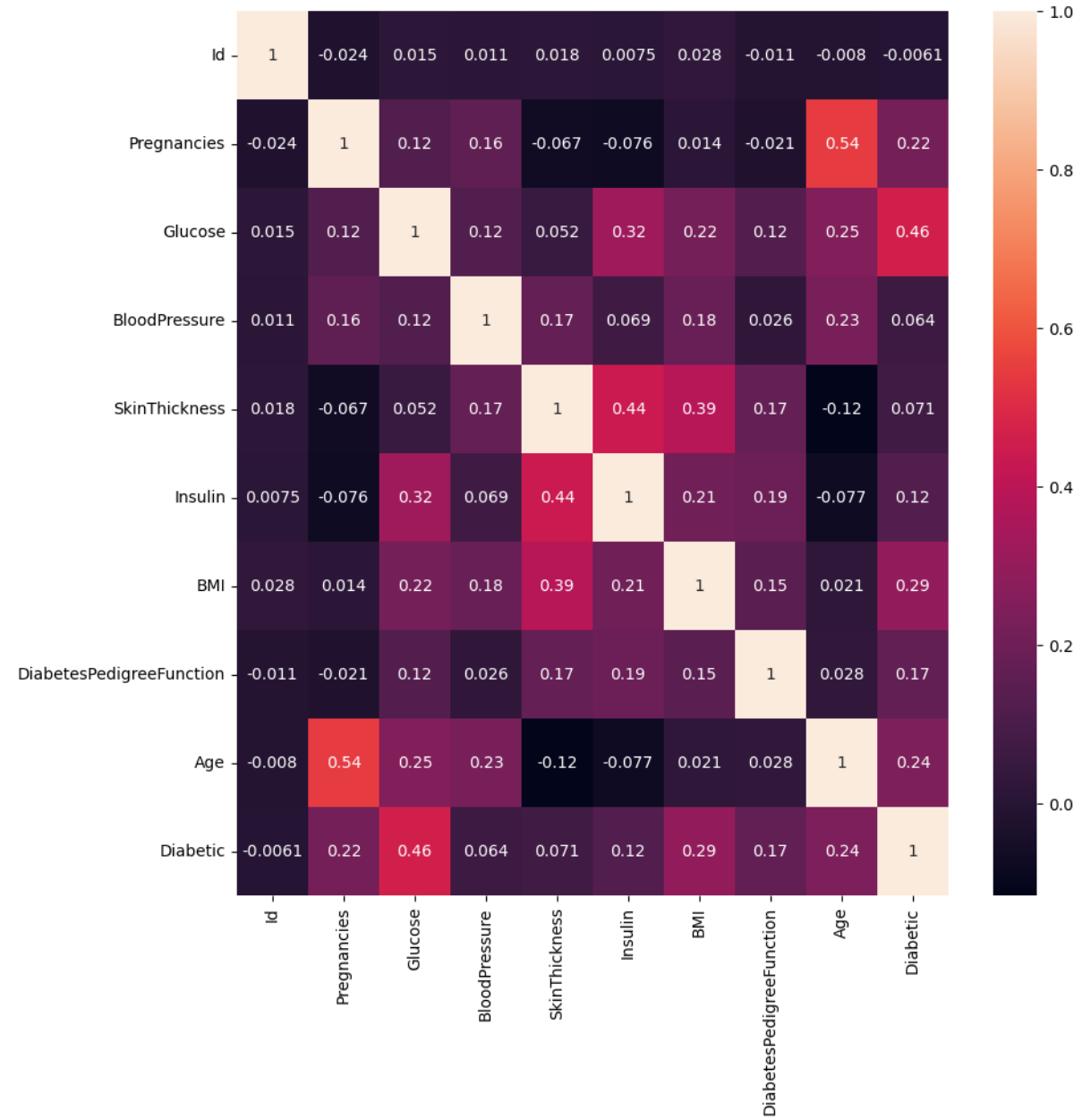
- The age distribution is right-skewed, with a large concentration of younger individuals.
- Few individuals are in the older age brackets.
- The median age is less than the 90th percentile, indicating that the majority are younger than the median age.

In general, the variables show a common pattern of being right-skewed, with medians that are typically lower than the 90th percentiles. This suggests that for most of these variables, a smaller number of individuals have high values.

2.4.3 Correlation analysis

A heatmap correlation is a visual representation of the correlation matrix using colors to show the strength and direction of relationships between variables in a dataset.

```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(), annot = True)
```



The variables Glucose (0.46), BMI (0.28), Age (0.23), and Pregnancies (0.22) exhibit a more pronounced correlation with diabetes compared to other variables, suggesting their potential as predictor variables. Moreover, DiabetesPedigreeFunction (0.16) and Insulin (0.12) also display some level of correlation.

However, SkinThickness and BloodPressure don't show significant correlation with the predictor variable. Although they might have interactive potential, confirming their relationship requires further investigation. Hence, at this stage, we opt not to include them in our model.

2.5 Model results

The libraries and functions we used in machine learning:

```
from sklearn.model_selection import train_test_split, PredefinedSplit, GridSearchCV
from sklearn.naïve_bayes import BernoulliNB
from sklearn.metrics import recall_score, precision_score, accuracy_score, f1_score, confusion_matrix,
ConfusionMatrixDisplay
from sklearn.ensemble import RandomForestClassifier
```

Feature engineering, remove unnecessary columns based on the correlation heatmap:

```
df = df.drop(['Id','BloodPressure', 'SkinThickness','Insulin', 'DiabetesPedigreeFunction'], axis = 1)
```

The variables Glucose (0.46) and BMI (0.28) are normally distributed and highly correlated with diabetes, making them strong potential predictor variables. Age (0.23) and Pregnancies (0.22) also show a notable correlation.

The Naïve Bayes Model:

```
# defining predictor variable
y = df['Diabetic']
# defining target variable
x = df.copy()
x = x.drop(columns = [ 'Diabetic' ])
# split the dataset
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, stratify = y, random_state = 0)
# Naïve Bayes model
nb = BernoulliNB(binarize=0.0)
model = nb.fit(x_train, y_train)
y_pred = model.predict(x_test)
# evaluating the model
print(f"Accuracy: {round(accuracy_score(y_test, y_pred),3)}")
print(f"Precision: {round(precision_score(y_test, y_pred),3)}")
print(f"recall: {round(recall_score(y_test, y_pred),3)}")
print(f"F1 Score: {round(f1_score(y_test, y_pred),3)}")
```

```
Accuracy: 0.652
Precision: 0.333
recall: 0.01
F1 Score: 0.02
```

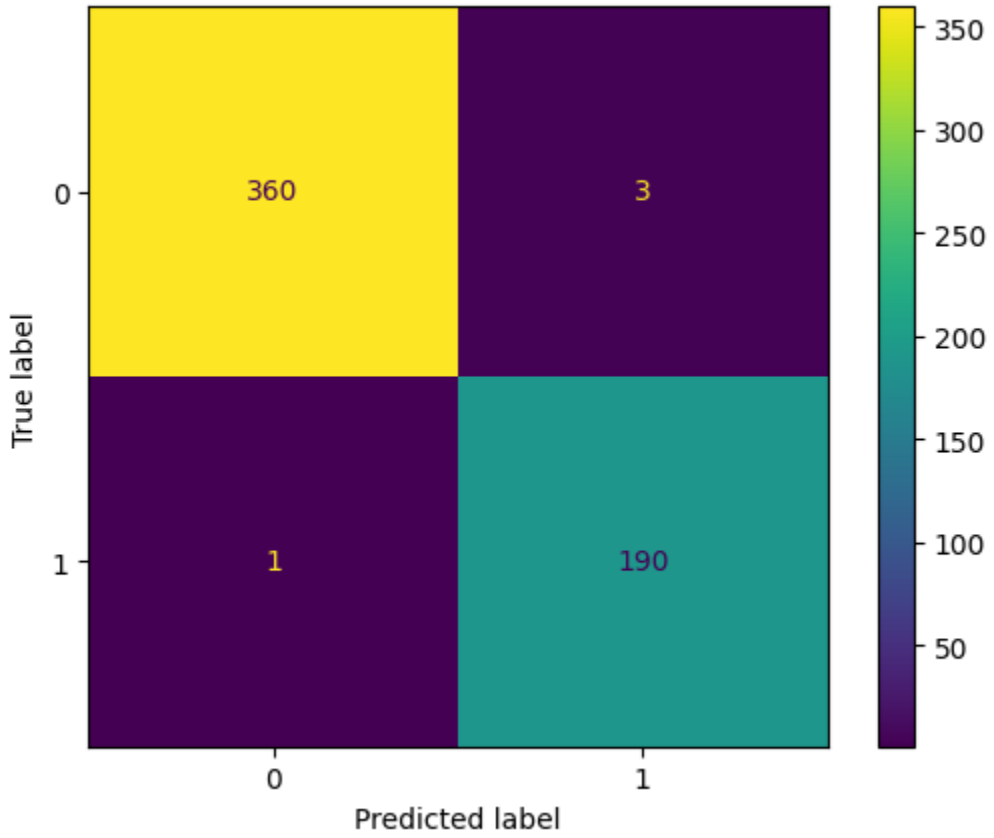
These metrics provide an assessment of the model's performance:

Accuracy: 0.652
The model correctly predicted 65.2% of the samples. Not a satisfying result.
Precision: 0.333
Only 33.3% of the samples predicted as positive were actually true positives. This suggests a high rate of false positives.
Recall: 0.01
The model identified only 1% of the actual positive samples.
F1 Score: 0.02
With a low F1 score, the model exhibits poor performance when considering both precision and recall.
The Naïve Bayes model's performance seems unsatisfactory. Its results imply numerous misclassifications.

The Random Forest Model:

```
x_tr, x_val, y_tr, y_val = train_test_split(x_train, y_train, stratify = y_train, random_state = 0, test_size = 0.20)

# defining the hyperparamters in Randomforest Model
cv_params = {
            'max_depth' : [8,10,14,20],
            'n_estimators' : [20, 40, 60, 80, 100],
            'min_samples_leaf' : [0.25, 0.5, 1],
            'min_samples_split' : [0.001, 0.01, 0.05],
            'max_features' : ['sqrt'],
            'max_samples' : [.5, .9]
}

# use GridsearchCV to find optimal parameters
split_index = [0 if x in x_val.index else -1 for x in x_train.index]
custom_split = PredefinedSplit(split_index)
clf = RandomForestClassifier(random_state = 0)
rf = GridSearchCV(clf, cv_params, cv = custom_split, refit = 'f1', n_jobs = -1, verbose = 1)
rf.fit(x_train, y_train)
rf.best_params_

# using optimal paramters
rf_op = RandomForestClassifier(n_estimators = 80, max_depth = 20, max_features = 'sqrt', max_samples = 0.9,
min_samples_leaf = 1,
                        min_samples_split = 0.001, random_state = 0)
rf_op.fit(x_train, y_train)
y_pred = rf_op.predict(x_test)
print(f"Accuracy: {round(accuracy_score(y_test, y_pred),3)}")
print(f"Precision: {round(precision_score(y_test, y_pred),3)}")
print(f"recall: {round(recall_score(y_test, y_pred),3)}")
print(f"F1 Score: {round(f1_score(y_test, y_pred),3)}")
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = rf_op.classes_)
disp.plot()
```

```
Accuracy: 0.993
Precision: 0.984
recall: 0.995
F1 Score: 0.99
```



Compared with the Naïve Bayes model, the Random Forest model's performance seems satisfactory.

3. Conclusion

3.1 Key Insights Derived from Analysis

From the comprehensive analysis and results obtained:

- The dataset, comprising 2768 patient records with 8 health-related numerical attributes, facilitated the development of predictive models for diabetes.
- All 8 health features were utilized in training the models, and after outlier removal and feature selection, predictors significantly correlated with the outcome were retained.
- Models Developed: Two classifiers were implemented:
    1. **Bernoulli Native Bayes Classifier**
    2. **Random Forest Classifier**
- Model Performances:
    1. **Bernoulli Native Bayes Classifier:** Achieved an accuracy of 0.65.

2. **Random Forest Classifier:** Demonstrated significantly higher accuracy, reaching 0.984 on the test set.

- The comparison between the two classifiers revealed a substantial performance gap, emphasizing the Random Forest Classifier's superiority in accurately predicting diabetes based on the dataset's features.

3.2 Weaknesses and Model Limitations

Despite the promising results, our models possess certain limitations:

- **Outlier Handling Impact:** The removal of outliers in BMI, glucose, and blood pressure might have influenced the models negatively. While necessary for data integrity, outlier removal could have led to the loss of crucial information, potentially affecting the models' robustness.
- **Model Limitations:** Both models have their limitations, as evident from the lower accuracy of the Bernoulli Native Bayes Classifier. The Bernoulli Native Bayes Classifier exhibited comparatively lower accuracy, indicating potential challenges in capturing the underlying data distributions accurately.
- **Native Bayes Classifier**: The Naïve Bayes Classifier assumes independence between features, which might not entirely hold true in health-related attributes. This assumption limitation can affect the model's ability to capture intricate relationships among variables, potentially impacting predictive accuracy.
- **Random Forest Classifier**: While the Random Forest Classifier achieved high accuracy, there might be a risk of overfitting due to its complexity, especially when working with a limited dataset. Employing techniques like cross-validation or regularization might mitigate this risk.

Reference

Efron, B., & Tibshirani, R. J. (1994). An introduction to the bootstrap. CRC press, 436.

Ho, T. K. (1998). The random subspace method for constructing decision forests. IEEE transactions on pattern analysis and machine intelligence, 20(8), 832-844.

Ho, T. K. (2002). A data complexity analysis of comparative advantages of decision forest constructors. Pattern Analysis & Applications, 5, 102-112.

Atlas, D. (2021). International diabetes federation. IDF Diabetes Atlas, 10th edn. Brussels, Belgium: International Diabetes Federation. [Accessed on December 7, 2023], from https://www.diabetesatlas.org.

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., & Brown, S. D. (2004). An introduction to decision tree modeling. Journal of Chemometrics, 18(3), 275-285. https://doi.org/10.1002/cem.873

Song, Y. Y., & Ying, L. U. (2015). Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry, 27(2), 130.

Suthaharan, S. (2016). Decision Tree Learning. In: Machine Learning Models and Algorithms for Big Data Classification. (Vol 36. Integrated Series in Information Systems). Springer, Boston, MA. https://doi.org/10.1007/978-1-4899-7641-3_10

World Health Organization. (n.d.). Diabetes. [Accessed on December 7, 2023], from https://www.who.int/health-topics/diabetes#tab=tab_1

Dataset Source

https://www.kaggle.com/datasets/mathchi/diabetes-data-set?select=diabetes.csv