Table of Contents

Objective

The objective of this project is to develop a robust model for predicting the likelihood of diabetes in patients based on their medical history and demographic information. Such predictions can be immensely valuable for healthcare professionals in identifying individuals who may be at risk of developing diabetes. Furthermore, pharmaceutical companies are also interested in these predictions as they can aid in customer profiling and developing tailored treatment plans.

Information about the data

Dataset: The dataset used for this project is the Diabetes Prediction Dataset from Kaggle, which comprises a comprehensive collection of medical and demographic data from patients, along with their diabetes status (positive or negative). The dataset encompasses several essential features including age, gender, body mass index (BMI), hypertension, heart disease, smoking history, HbA1c level, and blood glucose level.

Data Preprocessing

Importing the required libraries

In this section, we import the required libraries and modules into the notebook. These libraries provide essential functionalities for data analysis and modeling, such as data manipulation, visualization, and machine learning algorithms. Importing the necessary libraries sets the foundation for the subsequent analysis.

In [63]:
```python
import pandas as pd
import numpy as np
import scipy as sp
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import sklearn.model_selection

from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

import warnings
warnings.filterwarnings("ignore")
```

Data Cleaning

Data cleaning is a critical step in the data preprocessing phase. It involves handling missing values, removing duplicates, and addressing inconsistencies or errors present in the dataset. By ensuring the cleanliness and reliability of the data, we significantly reduce the likelihood of biased or inaccurate analysis.

In [3]:
```python
df = pd.read_csv("diabetes_prediction_dataset.csv")
```

In [4]:
```python
df.head()
```

Out[4]:

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	dia
0	Female	80.0	0	1	never	25.19	6.6	140	
1	Female	54.0	0	0	No Info	27.32	6.6	80	
2	Male	28.0	0	0	never	27.32	5.7	158	
3	Female	36.0	0	0	current	23.45	5.0	155	
4	Male	76.0	1	1	current	20.14	4.8	155	

In [5]:
```python
df.shape
```

Out[5]:
```
(100000, 9)
```

In [7]:
```python
df.dtypes
```

Out[7]:
```
gender                  object
age                    float64
hypertension             int64
heart_disease            int64
smoking_history         object
bmi                    float64
HbA1c_level            float64
blood_glucose_level      int64
diabetes                 int64
dtype: object
```

In [8]:
```
df.isna().sum()
```

Out[8]:
```
gender                 0
age                    0
hypertension           0
heart_disease          0
smoking_history        0
bmi                    0
HbA1c_level            0
blood_glucose_level    0
diabetes               0
dtype: int64
```

In [9]:
```
df['age'] = df['age'].astype(int)
```

In [11]:
```
df['smoking_history'].value_counts()
```

Out[11]:
```
No Info        35816
never          35095
former          9352
current         9286
not current     6447
ever            4004
Name: smoking_history, dtype: int64
```

In [12]:
```
df['gender'].value_counts()
```

Out[12]:
```
Female    58552
Male      41430
Other        18
Name: gender, dtype: int64
```

Upon further examination of the dataset, particularly focusing on the object types, we have identified a significant number of rows (35,816) where the smoking_history predictor is labeled as "No info." While "No info" can be considered as a dimension of information, it is not possible to impute or fill in this missing information when it constitutes more than 30% of the observations.

In [16]:
```
df = df.drop(columns="smoking_history")
```

In [17]:
```
df.describe()
```

Out[17]:

	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	
count	100000.000000	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	100(
mean	41.875660	0.07485	0.039420	27.320767	5.527507	138.058060	
std	22.535417	0.26315	0.194593	6.636783	1.070672	40.708136	
min	0.000000	0.00000	0.000000	10.010000	3.500000	80.000000	
25%	24.000000	0.00000	0.000000	23.630000	4.800000	100.000000	
50%	43.000000	0.00000	0.000000	27.320000	5.800000	140.000000	
75%	60.000000	0.00000	0.000000	29.580000	6.200000	159.000000	
max	80.000000	1.00000	1.000000	95.690000	9.000000	300.000000	

In [18]:
```
df['blood_glucose_level'] = df['blood_glucose_level'].astype(float)
```

In [20]:
```
df.dtypes
```

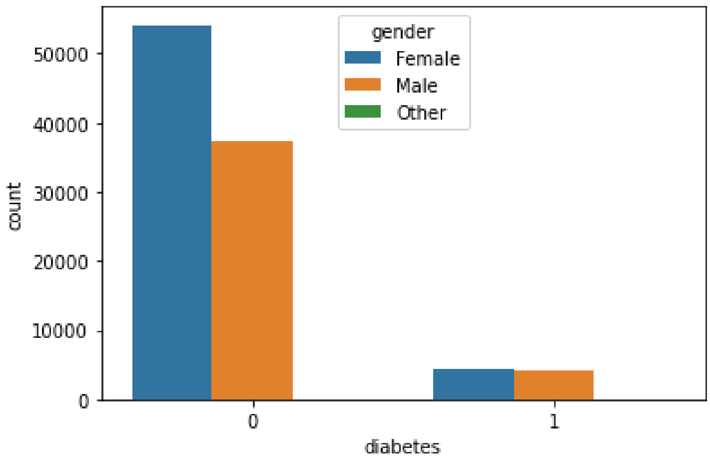
Out[20]:
```
gender                  object
age                      int32
hypertension             int64
heart_disease            int64
bmi                    float64
HbA1c_level            float64
blood_glucose_level    float64
diabetes                 int64
dtype: object
```

The Data Cleaning process for the Diabetes Prediction Dataset has been completed. Below are the final columns and their corresponding data types after the necessary adjustments: Age: This discrete variable has been converted into an integer type to accurately represent the age values. Blood_glucose_level: As a continuous variable, it has been converted into a float type to accurately represent the decimal values associated with blood glucose levels. By appropriately adjusting the data types for these variables, we ensure that they are represented in a format that aligns with their nature and supports further analysis and modeling tasks.

Data Visualisation

In [26]:
```
sns.countplot(x='diabetes',data=df,hue='gender')
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x2dc2eaee520>

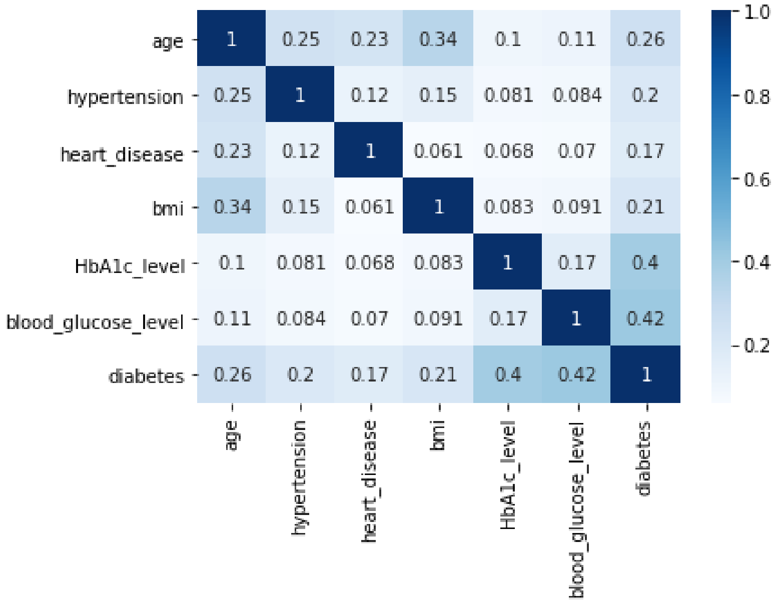


In [21]:
```
df.corr()
```

Out[21]:

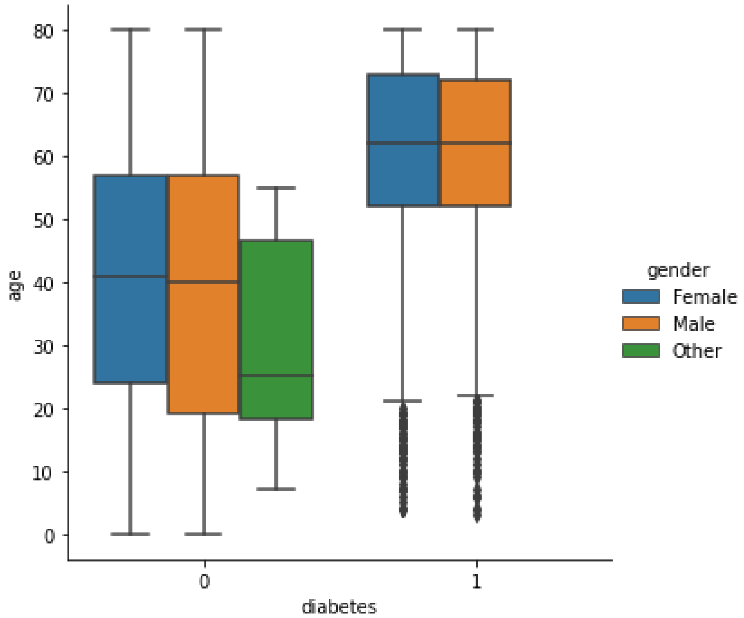
	age	hypertension	heart_disease	bmi	HbA1c_level	blood_glucose_level	dia
age	1.000000	0.251093	0.233254	0.337747	0.101328	0.110631	0.2
hypertension	0.251093	1.000000	0.121262	0.147666	0.080939	0.084429	0.1
heart_disease	0.233254	0.121262	1.000000	0.061198	0.067589	0.070066	0.1
bmi	0.337747	0.147666	0.061198	1.000000	0.082997	0.091261	0.2
HbA1c_level	0.101328	0.080939	0.067589	0.082997	1.000000	0.166733	0.4
blood_glucose_level	0.110631	0.084429	0.070066	0.091261	0.166733	1.000000	0.4
diabetes	0.257933	0.197823	0.171727	0.214357	0.400660	0.419558	1.0

```
In [22]: sns.heatmap(df.corr(), annot = True, cmap = 'Blues')
         plt.show()
```



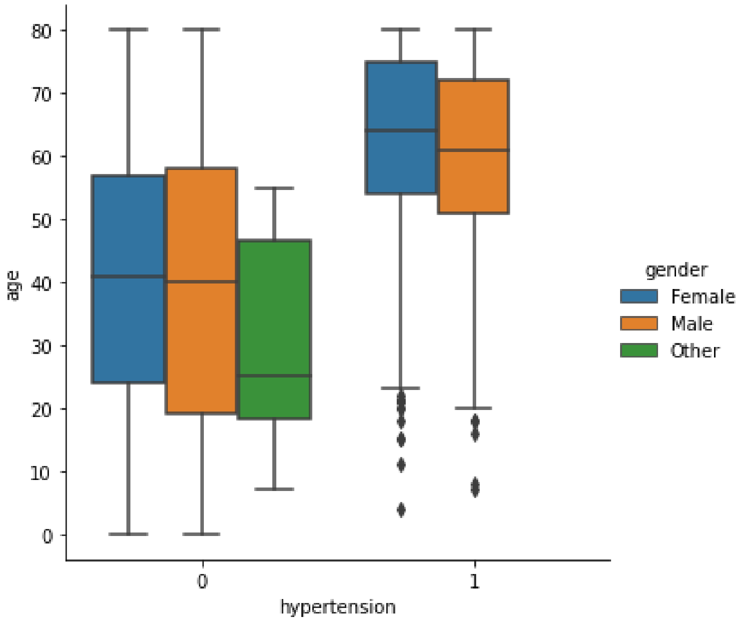
Overall, the heatmap analysis suggests that there are no significant correlations among the predictors in the dataset.

```
In [23]: sns.catplot(data = df, x = 'diabetes', y = 'age', kind = 'box', hue = 'gender')
         plt.show()
```



The Box Plot for Diabetes vs. Age reveals that individuals with diabetes tend to have a higher median age compared to those without diabetes. Moreover, the plot indicates a general trend of increasing age among individuals with diabetes, with a few outliers where diabetes is observed in individuals under the age of 20. This finding highlights the association between higher age and the likelihood of developing diabetes. It suggests that age plays a significant role in the prevalence of diabetes, with older individuals being more susceptible to the condition. The presence of outliers below the age of 20 may indicate cases of early-onset diabetes or other factors contributing to diabetes at a younger age. By considering the distribution of ages and the presence of outliers, we can better understand the relationship between age and diabetes. This insight can inform healthcare professionals and researchers in identifying age-related risk factors and designing targeted interventions to address diabetes in different age groups.

In [25]:
```
sns.catplot(data = df, x = 'hypertension', y = 'age', kind = 'box', hue = 'gender')
plt.show()
```



The Box Plot for Hypertension vs. Age, categorized by gender, shows a similarity to the previously observed Diabetes vs. Age plot. This similarity was expected, as there is a natural correlation between age and the likelihood of developing both diabetes and hypertension. Age can be considered a confounding factor in this context. The resemblance in the boxplots reinforces the notion that as age increases, the likelihood of developing hypertension also tends to increase. This aligns with the general understanding that age is a significant risk factor for hypertension.

Preparaing data for Modelling

In [27]:
```
#Defining dependent and independent variables.
x= df.drop('diabetes',axis=1)
y= df['diabetes']
```

In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state
```

In [45]:
```
ohe_gender=OneHotEncoder(sparse=False)
```

In [48]:
```
X_train_gender=ohe_gender.fit_transform(X_train[['gender']])
X_test_gender=ohe_gender.transform(X_test[['gender']])
```

In [52]:
```
X_train_rem=X_train.drop(columns=['gender'])
X_test_rem=X_test.drop(columns=['gender'])
```

In [54]:
```
X_train_transformed=np.concatenate((X_train_rem,X_train_gender),axis=1)
X_test_transformed=np.concatenate((X_test_rem,X_test_gender),axis=1)
```

In [50]:
```
X_train_gender
```

Out[50]:
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       ...,
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
```

In [55]:
```
X_train_transformed.shape
```

Out[55]:
```
(70000, 9)
```

In [56]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(70000, 7)
(70000,)
(30000, 7)
(30000,)
```

Now that we have prepared our training and testing datasets, we can proceed with evaluating the performance. During this evaluation, we will assess various metrics such as accuracy, precision, recall, F1 score, and training time for each classifier..

By systematically evaluating the performance metrics of each classifier, we can gain insights into their strengths and weaknesses.

Evaluation of model

In [32]:
```
#Created a dictionary to store the results.
Results = {}
```

Decision Tree Algorithm

A non-parametric classifier that models the decision rules as a tree. It's a powerful method that works well for both classification and regression problems. It's also easy to interpret the decision rules and the importance of the features.

In [67]:
```
dtc = DecisionTreeClassifier()
dtc.fit(X_train_transformed,y_train)

y_pred = dtc.predict(X_test_transformed)

Results['Decision Trees'] = [accuracy_score(y_test, y_pred),
                             precision_score(y_test, y_pred, average='weighted'),
                             recall_score(y_test, y_pred, average='weighted'),
                             f1_score(y_test, y_pred, average='weighted')]
```

In [66]:
```
# Creating a DataFrame from the Results
df_Results = pd.DataFrame.from_dict(Results, orient='index', columns=['Accuracy', 'Preci

df_Results
```

Out[66]:

	Accuracy	Precision	Recall	F1-Score
Decision Trees	0.953733	0.953884	0.953733	0.953808

Conclusion

The Result Dataframe provides precision, recall, and F1 values for each tested model. The scores indicate the accuracy achieved on the test dataset.

The F1 score serves as a combined measure of precision and recall. A higher F1 score implies better performance in terms of both metrics, making it an optimal criterion for model evaluation.

The Decision tree model demonstrates the accuracy of 0.953733 and the F1 score of 0.953808

In [ ]:
```

```