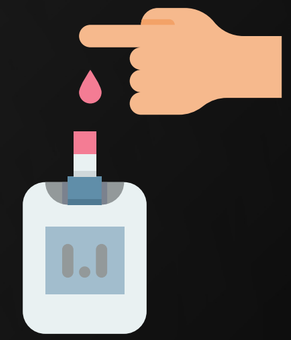
DIABETES PREDICTION ANALYSIS



Diabetes Prediction Dataset

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

Data Analysis

Data analysis is a crucial step in predicting diabetes. By collecting and analyzing various types of data, such as blood sugar levels, body mass index, and family history, we can identify patterns and risk factors associated with diabetes.
This data can then be used to develop predictive models that help healthcare professionals identify patients who are at high risk for developing diabetes. By intervening early, we can prevent or delay the onset of diabetes and improve patient outcomes.

Data Visualization

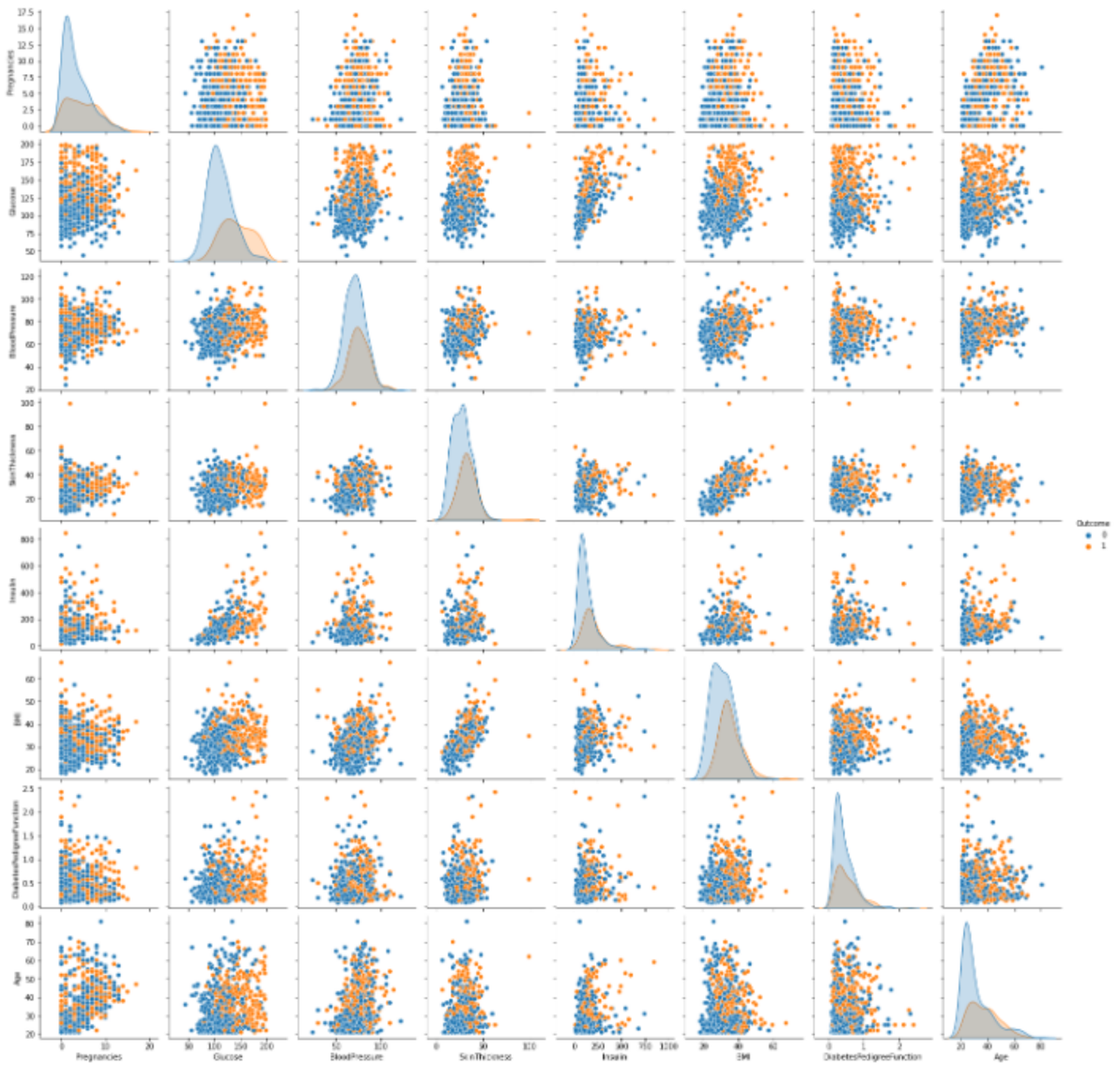
```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	637.000000	637.000000	637.000000	637.000000	637.000000	637.000000	637.000000	637.000000	637.000000
mean	3.780220	118.647152	71.979592	28.596567	128.817910	32.029042	0.425468	32.387755	0.312402
std	3.250842	28.843329	11.289134	8.452620	50.700469	6.408722	0.244526	10.687177	0.463837
min	0.000000	44.000000	38.000000	7.000000	15.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	64.000000	24.000000	110.000000	27.300000	0.239000	24.000000	0.000000
50%	3.000000	114.000000	72.000000	28.596567	128.817910	32.000000	0.361000	29.000000	0.000000
75%	6.000000	136.000000	80.000000	32.000000	128.817910	36.100000	0.583000	40.000000	1.000000
max	13.000000	198.000000	106.000000	60.000000	325.000000	49.700000	1.162000	64.0	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 637 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               637 non-null    int64
 1   Glucose                   637 non-null    float64
 2   BloodPressure             637 non-null    int64
 3   SkinThickness             637 non-null    float64
 4   Insulin                   637 non-null    float64
 5   BMI                       637 non-null    float64
 6   DiabetesPedigreeFunction  637 non-null    float64
 7   Age                       637 non-null    int64
 8   Outcome                   637 non-null    int64
dtypes: float64(5), int64(4)
memory usage: 49.8 KB
```

```
In [130]: sns.pairplot(data = df, hue = 'Outcome')
          plt.show()
```



PairPlot

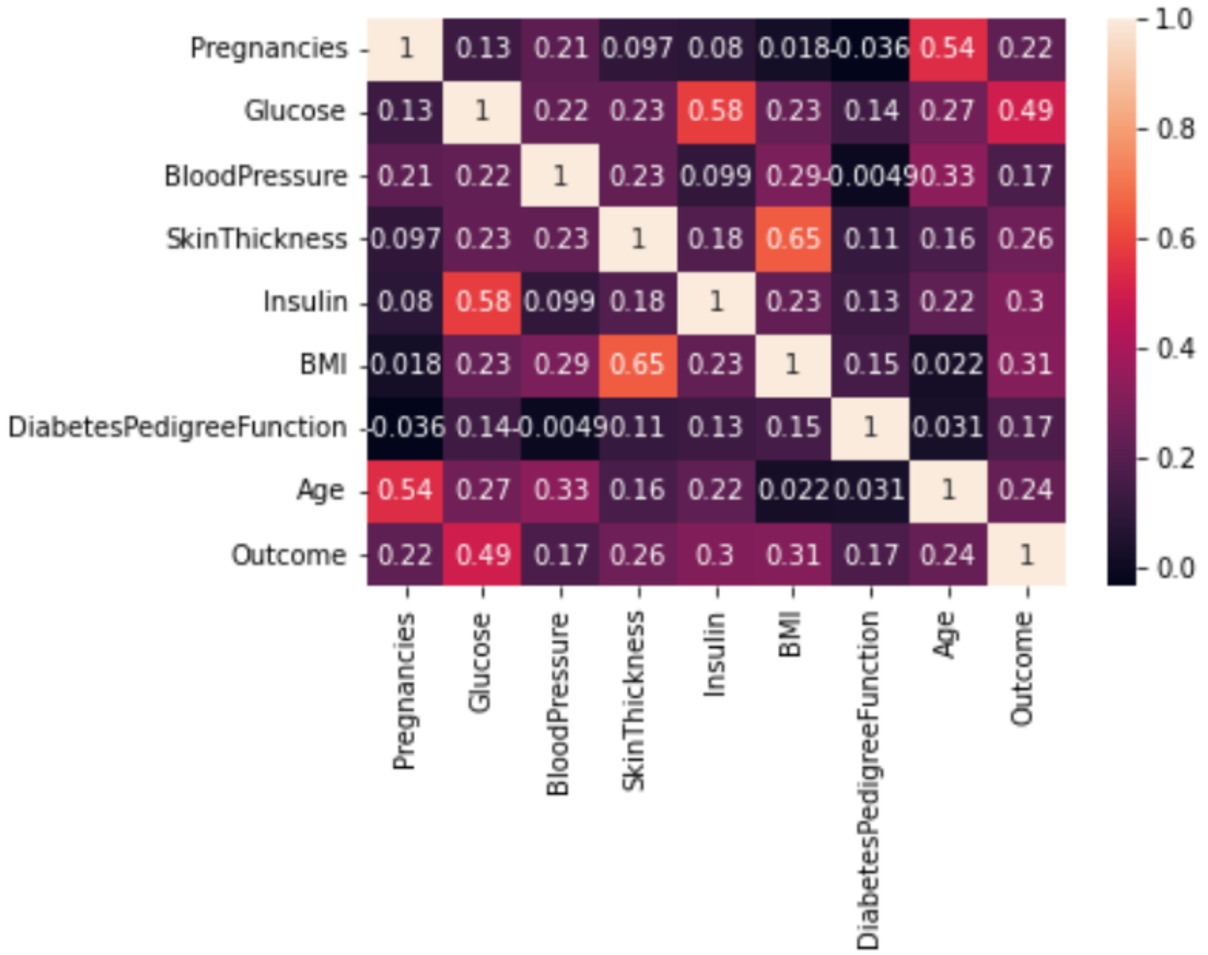
- Skinthickness linearly increases with BMI
- Blood Pressure increases as Glucose increases
- The More the Skinthickness more is the Insulin levels.

HeatMap

- A heat map is a two-dimensional representation of data in which various values are represented by colors. A simple heat map provides an immediate visual summary of information across two axes, allowing users to quickly grasp the most important or relevant data points. More elaborate heat maps allow the viewer to understand complex data sets.

```
sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```



Modeling

Cleaning The Data

- Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled.

- Finding the null values in the Dataset and removing them or replacing them with the mean values

- Finding the outliers and removing them from the dataset

Cleaning The Data

```
In [403]: 1 df.isnull().sum()

Out[403]: Pregnancies                 0
          Glucose                     0
          BloodPressure               0
          SkinThickness               0
          Insulin                     0
          BMI                         0
          DiabetesPedigreeFunction    0
          Age                         0
          Outcome                     0
          dtype: int64
```

Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
148	72	35	0	33.6	0.627	
85	66	29	0	26.6	0.351	
183	64	0	0	23.3	0.672	
89	66	23	94	28.1	0.167	
137	40	35	168	43.1	2.288	
...	...	...	...	...	...	
101	76	48	180	32.9	0.171	
122	70	27	0	36.8	0.340	
121	72	23	112	26.2	0.245	
126	60	0	0	30.1	0.349	
93	70	31	0	30.4	0.315	

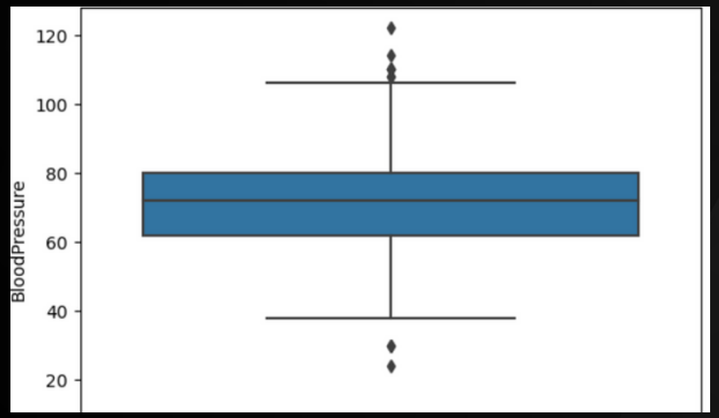
```
df[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = df[["Glucose", "BloodPressure", "SkinThickness",
                                                                          "Insulin", "BMI"]].replace(0, np.NaN)
```

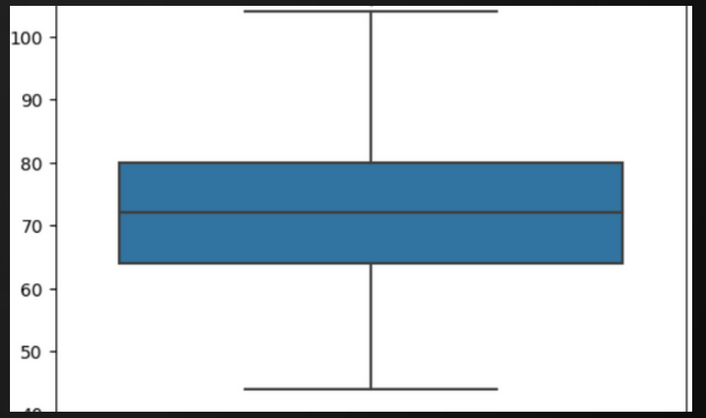
Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
148.0	72	35.0	NaN	33.6	0.627	50
85.0	66	29.0	NaN	26.6	0.351	31
183.0	64	NaN	NaN	23.3	0.672	32
89.0	66	23.0	94.0	28.1	0.167	21
116.0	74	NaN	NaN	25.6	0.201	30
...	...	...	...	...	...	...
101.0	76	48.0	180.0	32.9	0.171	63
122.0	70	27.0	NaN	36.8	0.340	27
121.0	72	23.0	112.0	26.2	0.245	30
126.0	60	NaN	NaN	30.1	0.349	47
93.0	70	31.0	NaN	30.4	0.315	23

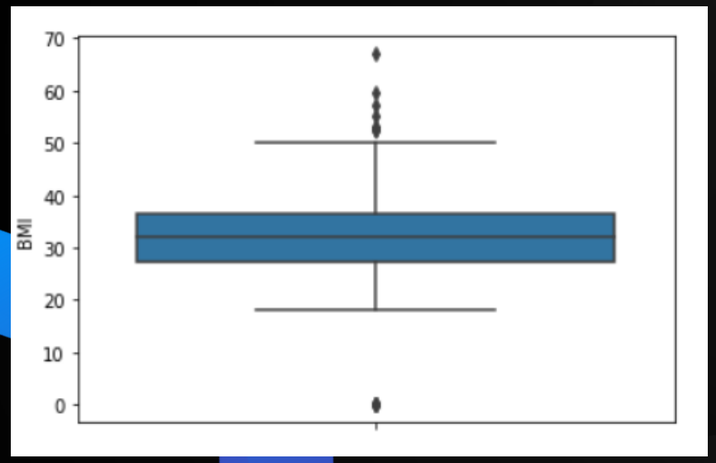
```
: 1 df.isnull().sum()

: Pregnancies                 0
  Glucose                     5
  BloodPressure               0
  SkinThickness             171
  Insulin                   302
  BMI                         0
  DiabetesPedigreeFunction    0
  Age                         0
  Outcome                     0
  dtype: int64
```

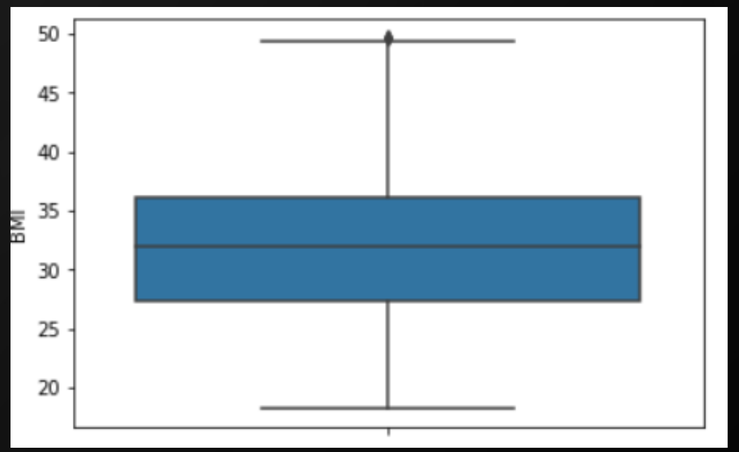
```
df["Glucose"].fillna(df["Glucose"].mean(), inplace = True)
df["BloodPressure"].fillna(df["BloodPressure"].mean(), inplace = True)
df["SkinThickness"].fillna(df["SkinThickness"].mean(), inplace = True)
df["Insulin"].fillna(df["Insulin"].mean(), inplace = True)
df["BMI"].fillna(df["BMI"].mean(), inplace = True)
df
```

Removing Outliers





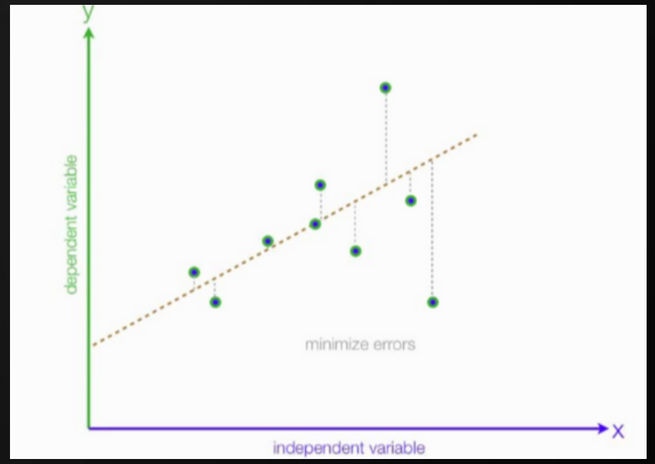




Regression :

Linear Regression

Linear regression is a statistical method that can be used to predict the likelihood of developing diabetes based on certain factors. It works by finding a line of best fit that shows the relationship between two variables, such as age and blood sugar levels. This line can then be used to predict future values based on the known data.



Performing Regresssion

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test= train_test_split(x,y,test_size=.20,random_state=87)
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
```

```
x_train=sc.fit_transform(x_train)
```

```
x_test=sc.transform(x_test)
```

```
from sklearn.linear_model import LinearRegression
```

```
reg=LinearRegression()
```

```
r2
```

```
0.16183028033577462
```

```
adj_r2
```

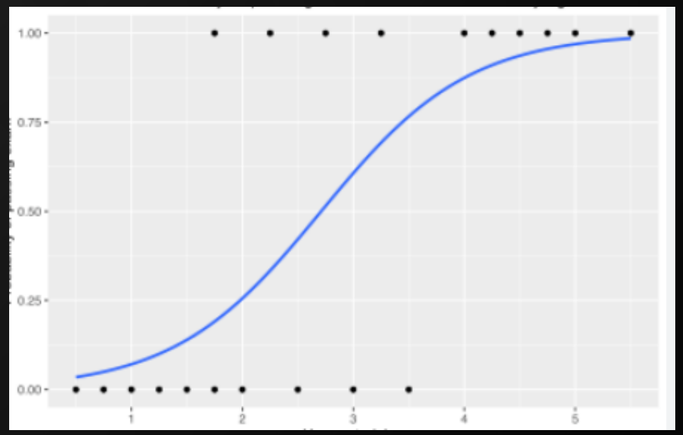
```
0.15181231555891128
```

Performing linear regression involves fitting a linear model to a set of data points to understand and predict the relationship between a dependent variable (often called the target or response variable) and one or more independent variables (often called features or predictors). The goal is to find the best-fitting line that minimizes the sum of squared differences between the predicted values and the actual data points.

Classification :

Logistic Regression

Logistic regression is a statistical method used to analyze data and predict the likelihood of an event occurring. In diabetes prediction, logistic regression can be used to determine the probability of a person developing diabetes based on their age, weight, blood pressure, and other factors.



Performing Classification

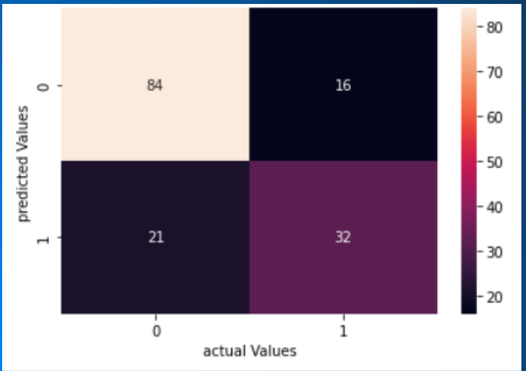
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train,y_test= train_test_split(x,y,test_size=.20,random_state=87)
```

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

```
lr.fit(x_train,y_train)
```

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.77	0.84	0.80	86
1	0.60	0.50	0.55	42
accuracy			0.73	128
macro avg	0.69	0.67	0.67	128
weighted avg	0.72	0.73	0.72	128



Performing logistic regression involves building a statistical model to predict a binary outcome (i.e., a yes/no or 0/1 outcome) based on one or more independent variables. Logistic regression is commonly used for classification tasks where the dependent variable is categorical

```
accuracy_2=(tn+tp)/(tn+tp+fn+fp)
accuracy_2
```

```
0.7581699346405228
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,y_pred)
```

```
0.7218867924528302
```

Decision Tree

Decision Tree

Decision trees are a popular machine learning method used in diabetes prediction. They work by breaking down a dataset into smaller and more manageable subsets, creating a tree-like structure of decisions that ultimately lead to a prediction. For example, a decision tree for diabetes prediction might start with the question of whether the patient is overweight or not. Depending on the answer, the tree would then branch out to further questions until a prediction is made.

One benefit of decision trees is that they are easy to interpret and visualize. This makes them useful for identifying which factors are most important in predicting diabetes. However, decision trees can also be prone to overfitting, where the model fits too closely to the training data and performs poorly on new data. It's important to carefully tune the parameters of the decision tree algorithm to avoid this issue

Decision Tree

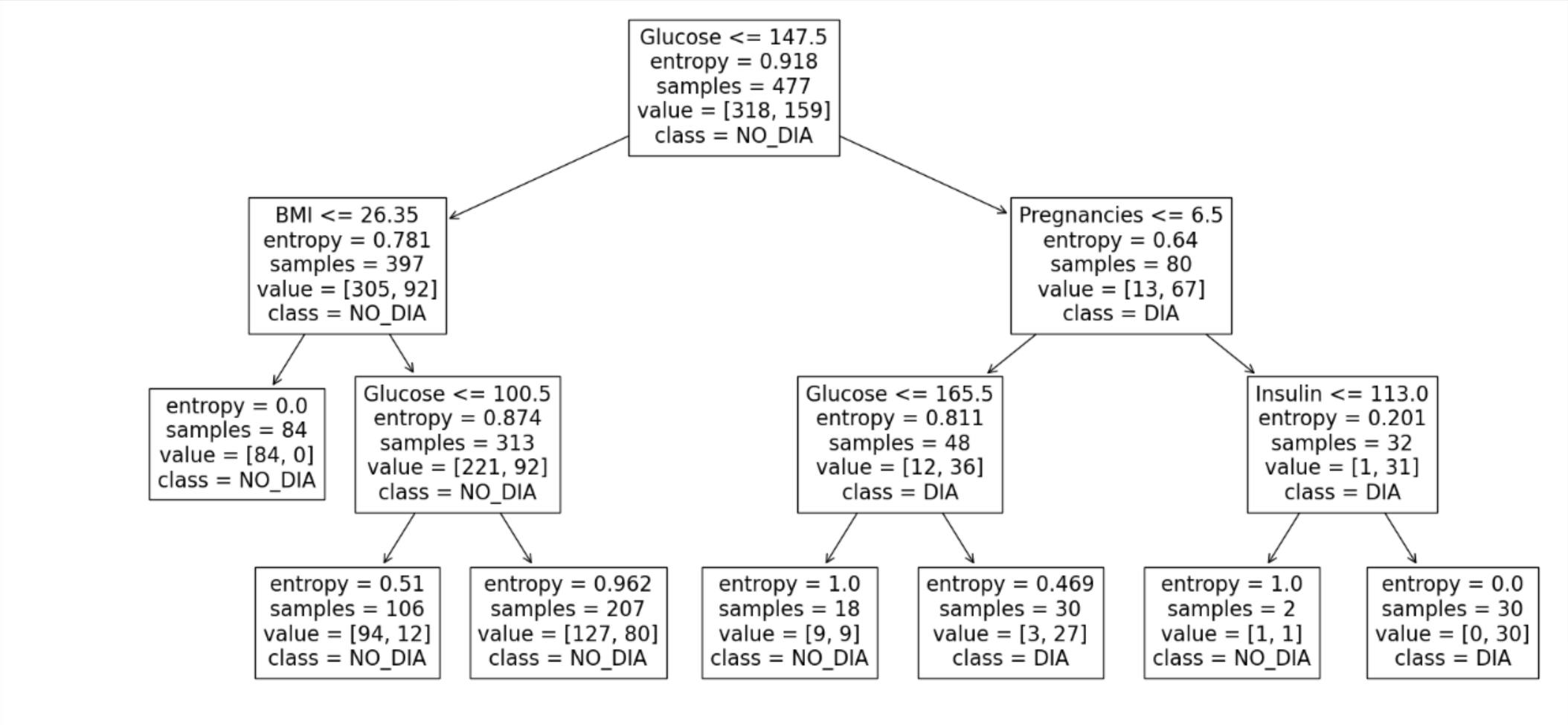
A Decision Tree is a supervised Machine learning algorithm.
It is used in both classification and regression algorithms.
The decision tree is like a tree with nodes

We can use the following code to create our decision tree

```
In [316]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train,y_test= train_test_split(x,y,test_size=.25,random_state=67)

In [317]:
from sklearn.tree import DecisionTreeClassifier

In [318]:
dtc=DecisionTreeClassifier(criterion='entropy',max_depth=3)

In [319]:
dtc.fit(x_train,y_train)

Out[319]:
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

```
In [335]:
import matplotlib.pyplot as plt

plt.figure(figsize=(20,10))
from sklearn import tree
tree.plot_tree(dtc)
plt.show()
```

Decision Tree



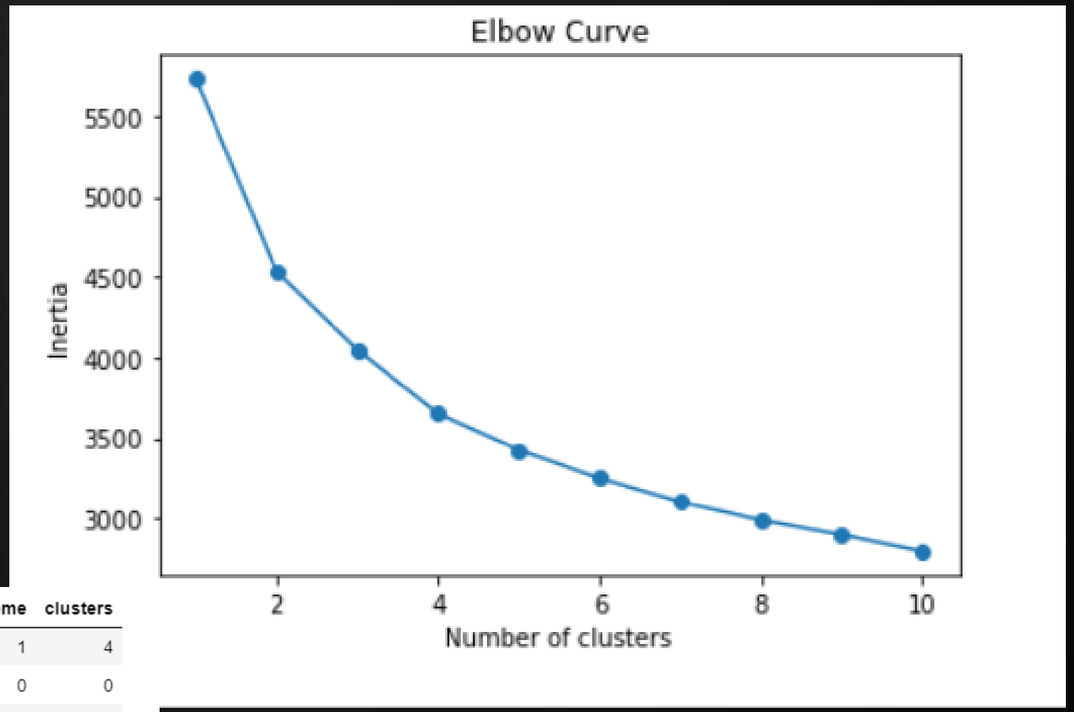
Clustering

Clustering

Clustering is a technique used to group similar data points together. In diabetes prediction, clustering can be used to identify groups of patients with similar risk factors. By analyzing these groups, doctors can make more accurate predictions about which patients are at high risk for having diabetes.
One example of how clustering has been used in diabetes prediction is through the analysis of electronic health records. By clustering patients based on their medical history and other risk factors, researchers were able to identify several distinct subgroups of patients with different levels of diabetes risk. This information can be used to develop personalized prevention and treatment plans for each patient.

Clustering

The elbow method is a graphical method for finding the optimal K value in a k-means clustering algorithmThe elbow graph shows the within-cluster-sum-of-square (WCSS) values on the y-axis corresponding to the different values of K (on the x-axis). The optimal K value is the point at which the graph forms an elbow.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	clusters
0	6	148.0	72	35.000000	128.81791	33.6	0.627	50	1	4
1	1	85.0	66	29.000000	128.81791	26.6	0.351	31	0	0
2	8	183.0	64	28.596567	128.81791	23.3	0.672	32	1	4
3	1	89.0	66	23.000000	94.00000	28.1	0.167	21	0	2
5	5	116.0	74	28.596567	128.81791	25.6	0.201	30	0	0
6	3	78.0	50	32.000000	88.00000	31.0	0.248	26	1	2
10	4	110.0	92	28.596567	128.81791	37.6	0.191	30	0	0

Our team

KRISHNA BHADAURIA

JAHNVI CHAURASIA

KASHISH sAHU

SHIKHA SINGH

THANK YOU

For watching this presentation :)