Comparing Logistic Regression and Decision Tree for Diabetes Prediction:

An In-depth Evaluation of Performance and Interpretability.

(Summer Research Academy)

By Wayne Cowell, Eduardo Almonte, Md Mustafizur Rahman and Maimouna Diallo .

Faculty Advisor: Dr .Narasim Banavara

Program Director - Graduate Computer Science

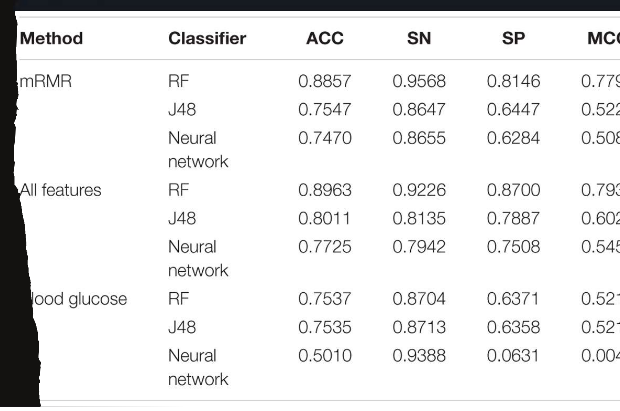
Department of Mathematics and Computer Science, Mercy College

July 14,2023

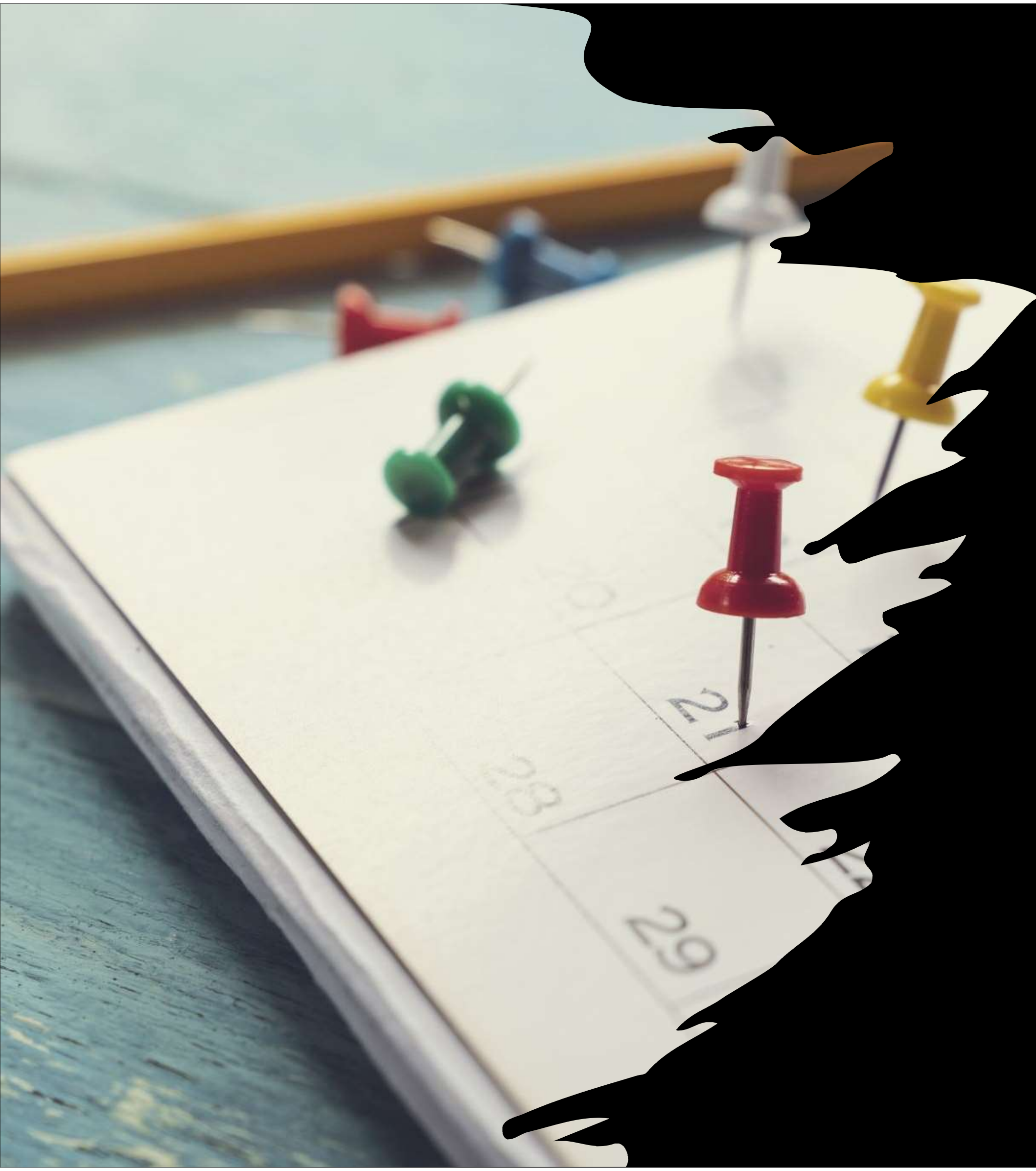
Introduction

- Diabetes is a disorder with a global impact on public health. Detecting early prediction of risk can ease interventions and improve individual quality of life.
- This research aim is to developed a logistic and decision tree model approach for diabetes prediction by analyzing the "Diabetes_Prediction_Dataset.csv"

  (kaggle.com)
- This dataset consist of over 100,000 entries which includes diabetic and non-diabetic cases.



Method	Classifier	ACC	SN	SP	MC
mRMR	RF	0.8857	0.9568	0.8146	0.779
	J48	0.7547	0.8647	0.6447	0.522
	Neural network	0.7470	0.8655	0.6284	0.508
All features	RF	0.8963	0.9226	0.8700	0.793
	J48	0.8011	0.8135	0.7887	0.602
	Neural network	0.7725	0.7942	0.7508	0.545
lood glucose	RF	0.7537	0.8704	0.6371	0.521
	J48	0.7535	0.8713	0.6358	0.521
	Neural network	0.5010	0.9388	0.0631	0.004



Objective

- The primary objective of this diabetes dataset analysis is to predict the outcome of diabetes base on certain predictors such as; smoking, gender and blood glucose within the dataset.
- Can you predict if a person will have diabetes using a Logistic Regression and a Decision Tree model approach?

Dataset

- Dataset obtained from Kaggle
- Valuable insights into factors influencing diabetes occurrence
- Enables pattern exploration and future prediction
- Medical and demographic data from patients
- Includes age, gender, BMI, hypertension, heart disease, smoking history, HbA1c levels, and blood glucose levels

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

100000 rows × 9 columns

Check for Null Values

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   gender               100000 non-null  object 
 1   age                  100000 non-null  float64
 2   hypertension         100000 non-null  int64  
 3   heart_disease        100000 non-null  int64  
 4   smoking_history      100000 non-null  object 
 5   bmi                  100000 non-null  float64
 6   HbA1c_level          100000 non-null  float64
 7   blood_glucose_level  100000 non-null  int64  
 8   diabetes             100000 non-null  int64  
dtypes: float64(3), int64(4), object(2)
```

Data Wrangling

Clean and structured data allows for accurate and effective analysis

Tools

Data Preprocessing:

- Consistency achieved through various techniques
- Utilized Google Collab, a cloud-based Jupyter notebook

Techniques employed:

- Python libraries: Pandas, NumPy, and Scikit-learn
- Robust functionality for handling missing values, normalizing features, and addressing data inconsistencies
- Data visualization tools: Matplotlib and Seaborn

Machine Learning:

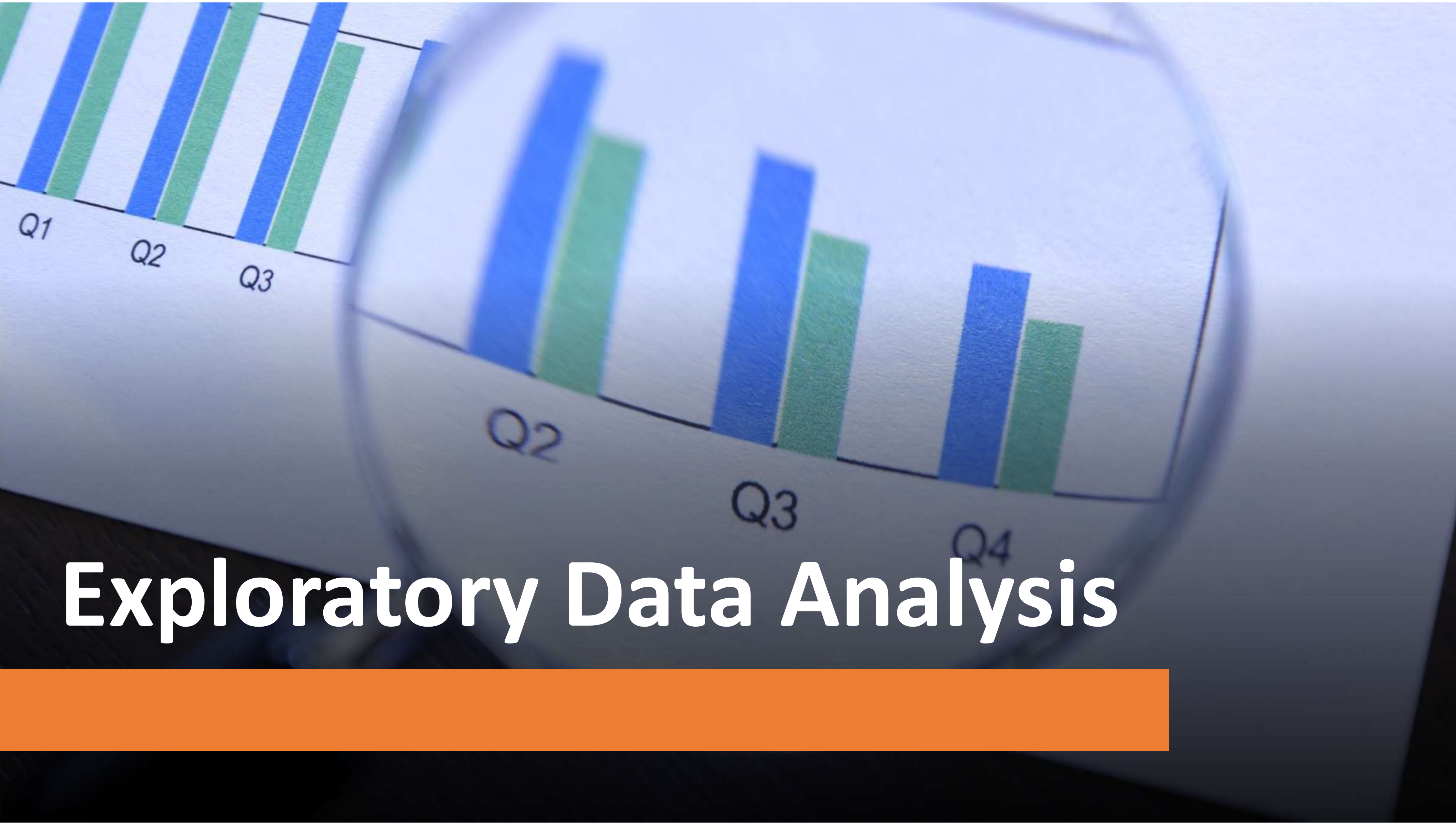
- Accurate prediction of diabetes occurrence
- Scikit-learn

Machine Learning Models:

- Logistic regression Model
- Decision Tree Model

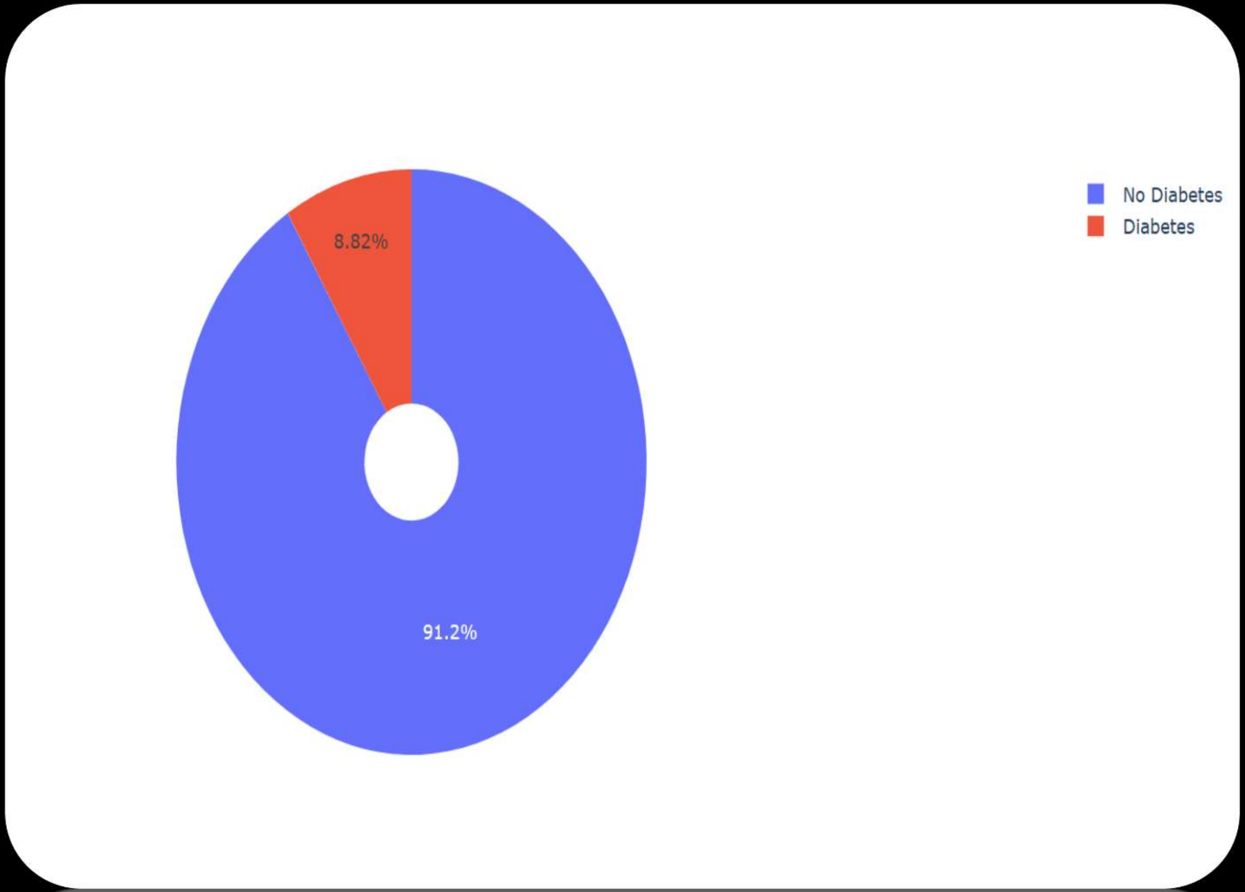
	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	1	80.0	0	1	0	25.19	6.6	140	0
1	1	54.0	0	0	0	27.32	6.6	80	0
2	0	28.0	0	0	0	27.32	5.7	158	0
3	1	36.0	0	0	2	23.45	5.0	155	0
4	0	76.0	1	1	2	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99994	1	36.0	0	0	0	24.60	4.8	145	0
99996	1	2.0	0	0	0	17.37	6.5	100	0
99997	0	66.0	0	0	1	27.83	5.7	155	0
99998	1	24.0	0	0	0	35.42	4.0	100	0
99999	1	57.0	0	0	2	22.43	6.6	90	0

96128 rows × 9 columns

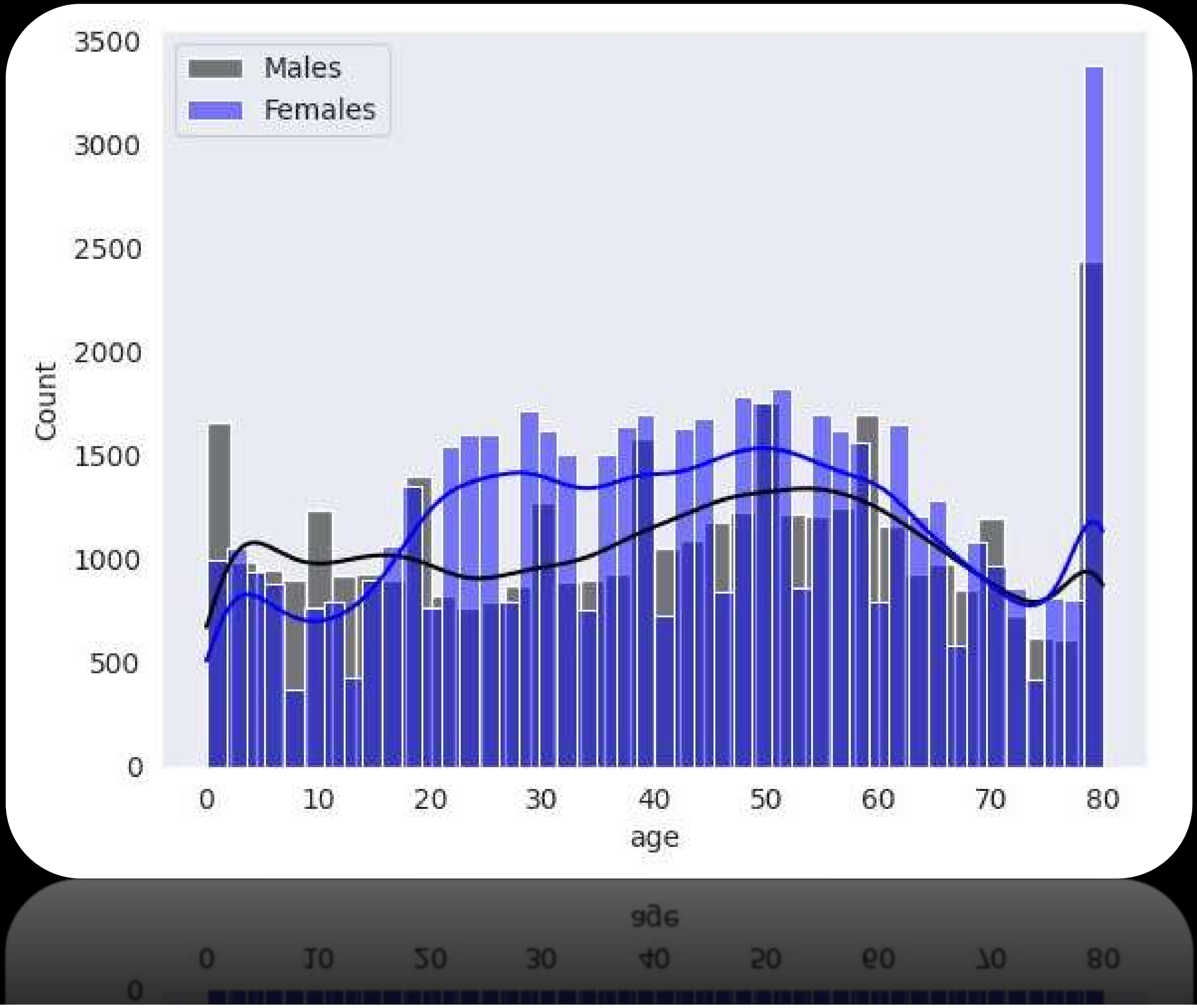


Exploratory Data Analysis

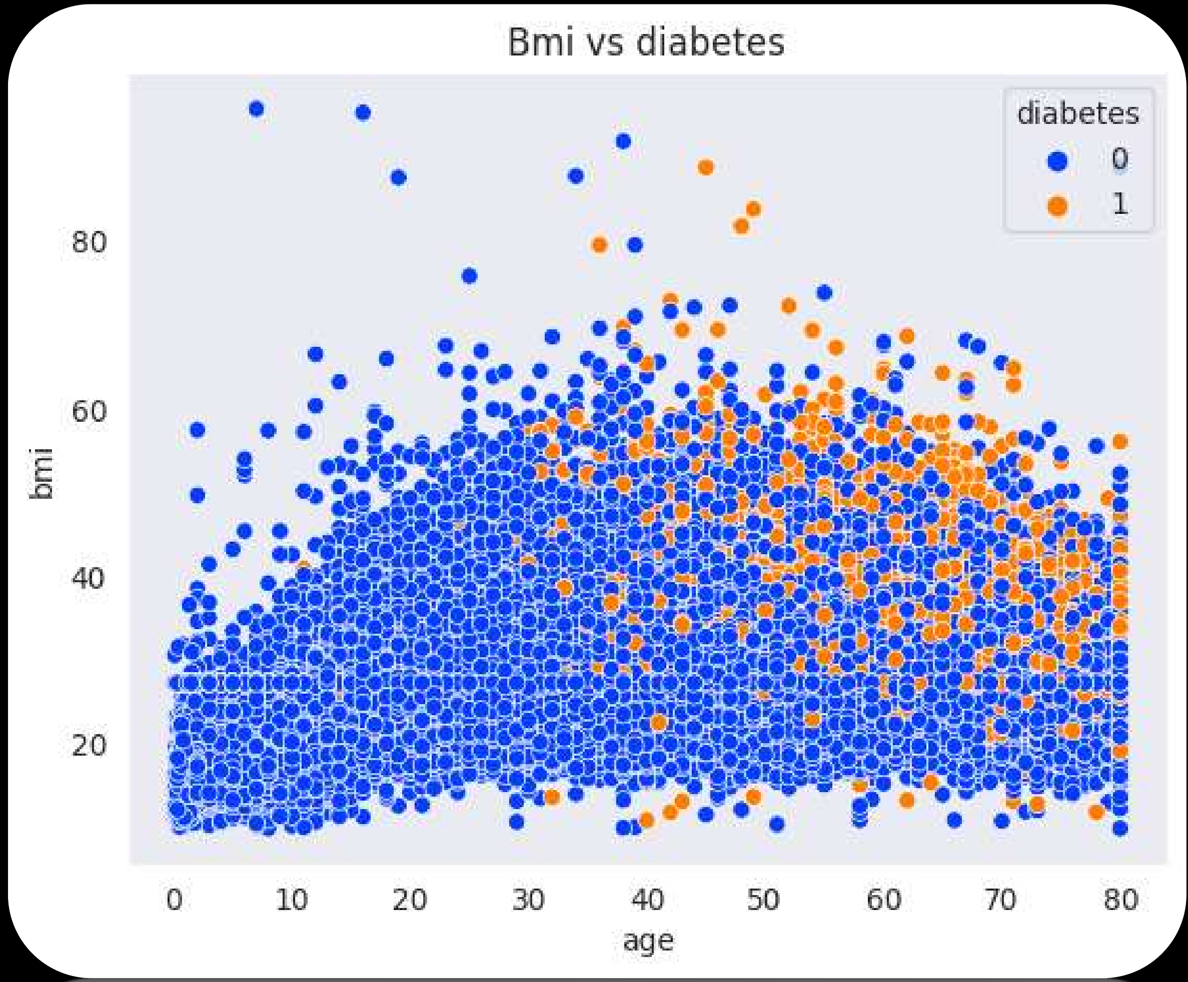
Percentage of Diabetic and Non-diabetic



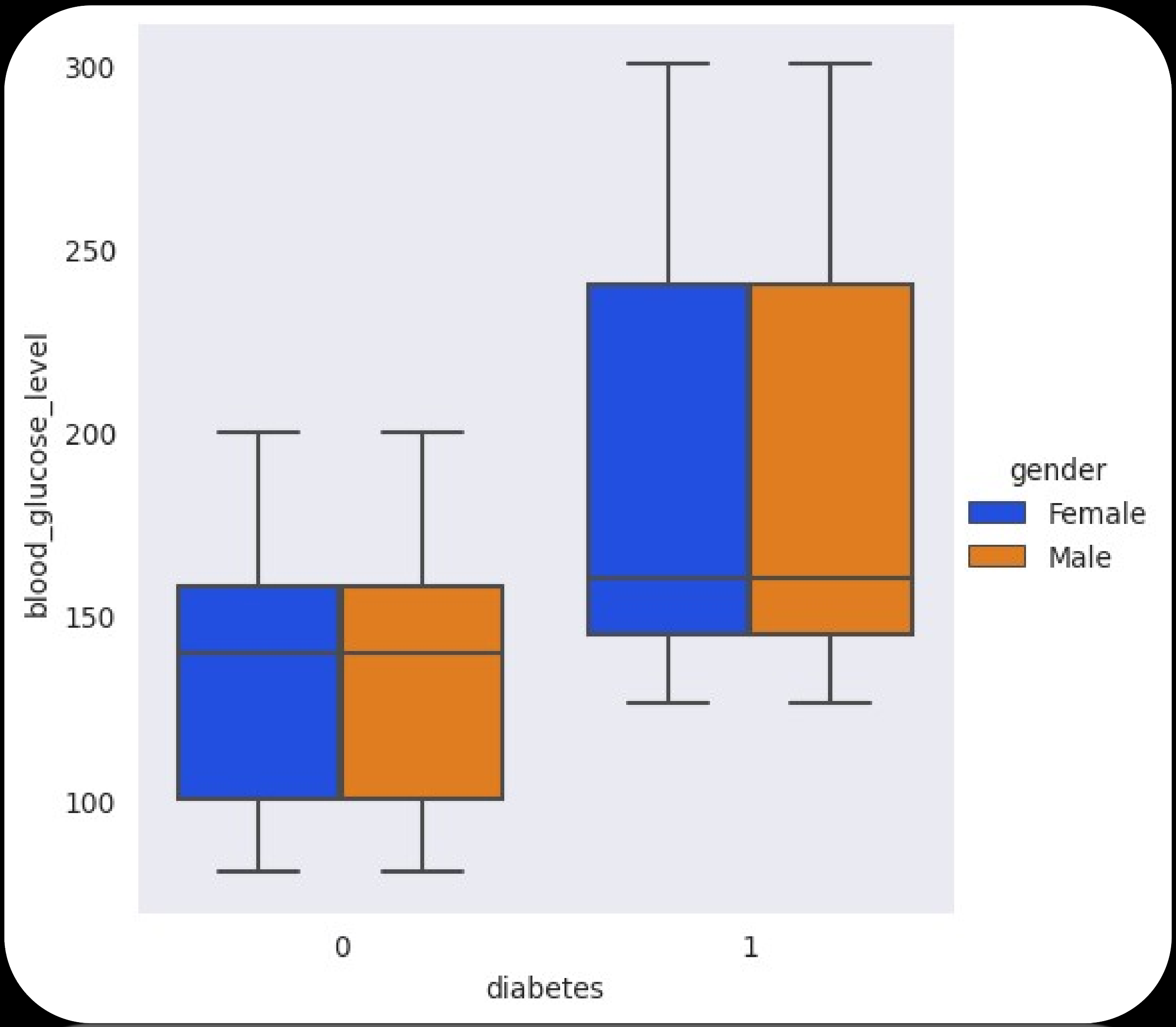
Diabetes by age.



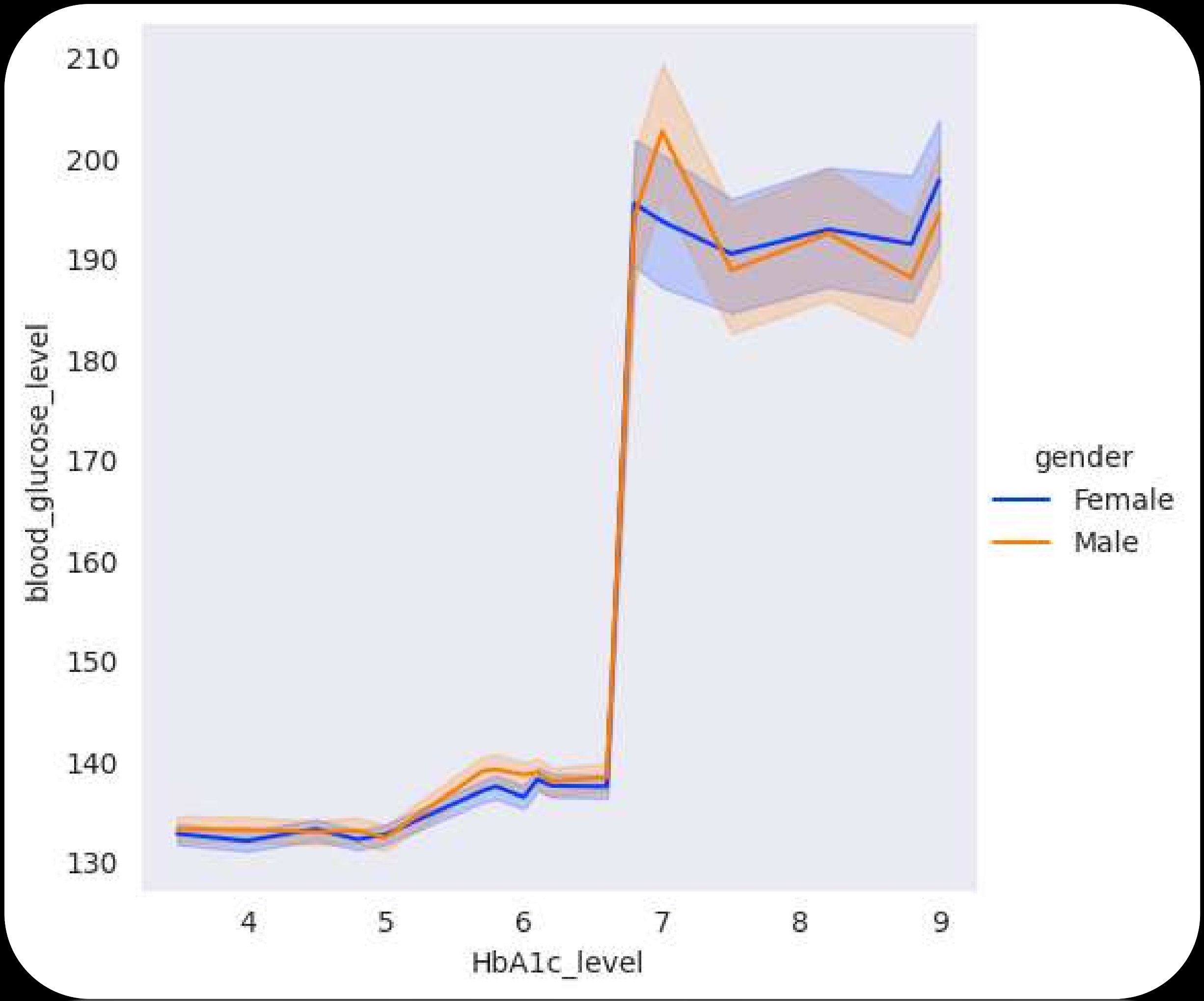
BMI vs Diabetes



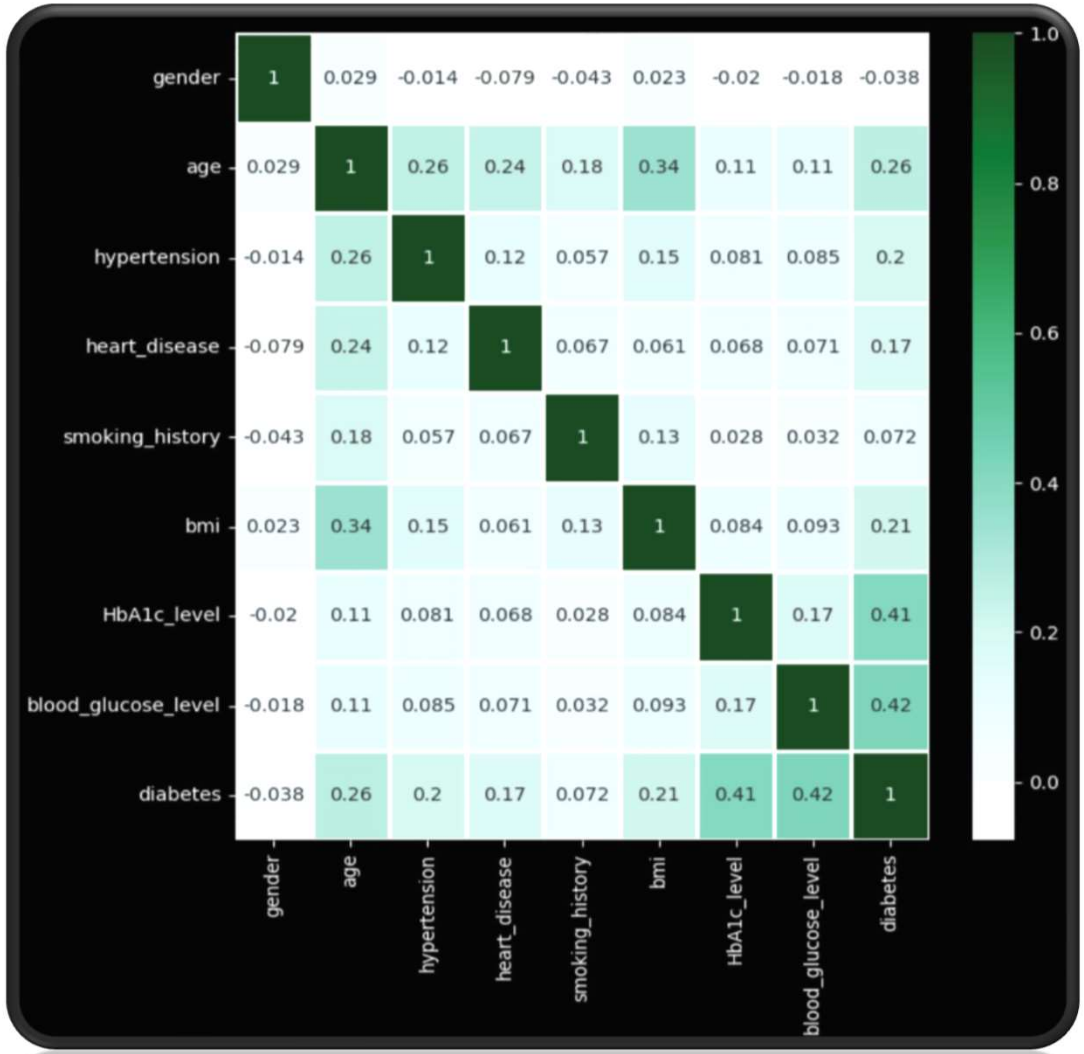
Diabetes regarding Blood Glucose Level

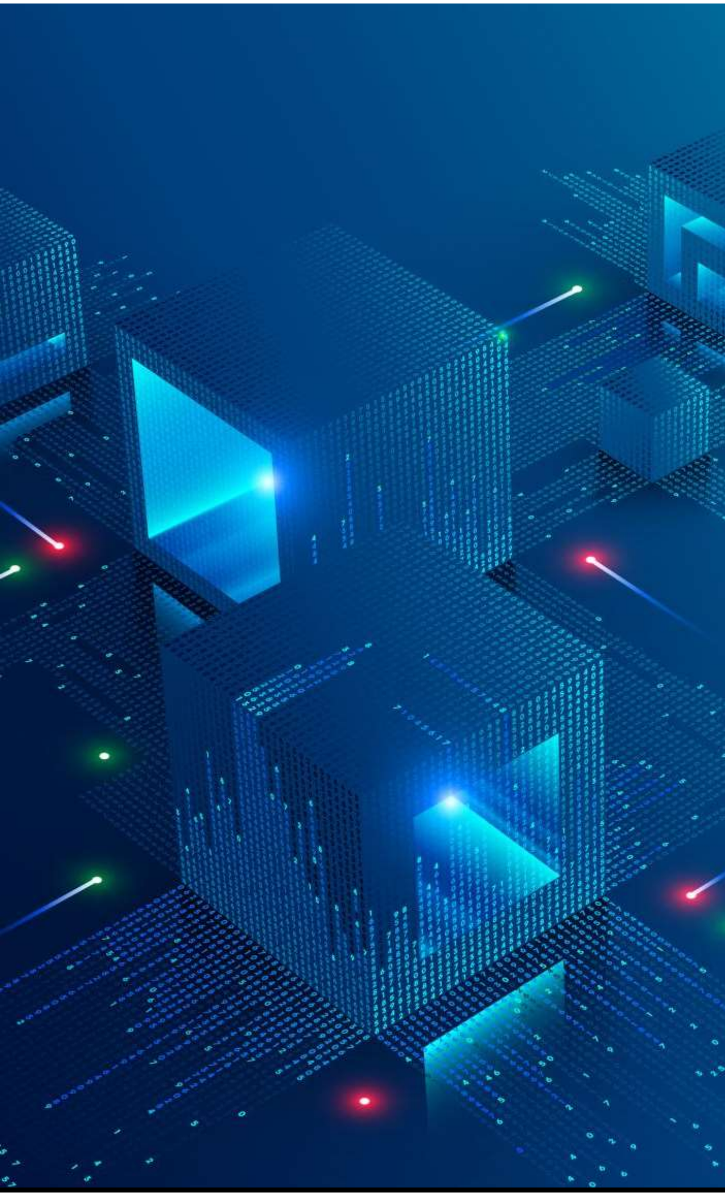


HbA1c Level and Blood Glucose Level



Relationship between the variables





Machine Learning

- Field of study focused on algorithms and statistical models
- Enables computers to learn from data and make predictions or decisions
- Extracts patterns, relationships, and insights from datasets

Machine Learning Models:

- Logistic Regression
- Decision Tree

Logistic Regression

A predictive algorithm using independent variables to predict the dependent variable of categorical type.

Splitting the dataset

```
[119]  1 #Split data into a training and testing set
       2 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

[118]  1 x_train.shape

     (67289, 8)

[120]  1 x_test.shape

     (28839, 8)

[121]  1 y_train.shape

     (67289,)

[122]  1 y_test.shape
```

- Train the logistic regression model using the training set.
- Evaluate the model's performance by making predictions on the testing set.
- Compare the predicted values against the testing values.
- Assess how well the model predicted the testing data

[41]
```
#Logistic Regression model
model = LogisticRegression()
```

[42]
```
#fit the model
model.fit(x_train, y_train)
```

```
#run the model to do predictions on the testing data set
y_pred = model.predict(x_test)
```

[123]
```
y_pred
```

```
array([0, 1, 0, ..., 0, 0, 0])
```

[124]
```
y_test
```

```
44521    0
35753    1
44898    0
74695    0
22543    0
        ..
33329    0
86186    0
80232    0
75150    0
23031    0
Name: diabetes, Length: 28839, dtype: int64
```

```
confusion_matrix(y_test, y_pred)
```

```
array([[26125,   236],
       [  929,  1549]])
```

Use performance metrics to measure the model's effectiveness

```
[296]  1 print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.96	0.99	0.98	26271
1	0.87	0.60	0.71	2568
accuracy			0.96	28839
macro avg	0.92	0.80	0.84	28839
weighted avg	0.95	0.96	0.95	28839

Decision Tree

The tree is constructed by recursively partitioning the data into smaller subsets, using the features that best separate the target variable's values.

Splitting the dataset

[205]
```
#DecisionTree
#Split the data into training and testing datasets

xTrain, xTest, yTrain, yTest = train_test_split(x,y,test_size=0.3)
```

```
#shapes
xTrain.shape
```

```
(67289, 8)
```

[207]
```
yTrain.shape
```

```
(67289,)
```

[208]
```
xTest.shape
```

```
(28839, 8)
```

[209]
```
yTest.shape
```

```
(28839,)
```

- Build a random decision tree using a maximum depth of 3

- Fit Training data to the model

- Compare the predicted values against the testing values

- Assess how well the model predicted the testing data

```
[242]
#Build a random descision tree. Using a maximum depth of 3
decTree = DecisionTreeClassifier(max_depth=3)
```

```
#fit the data to the model
decTree.fit(xTrain, yTrain)
```

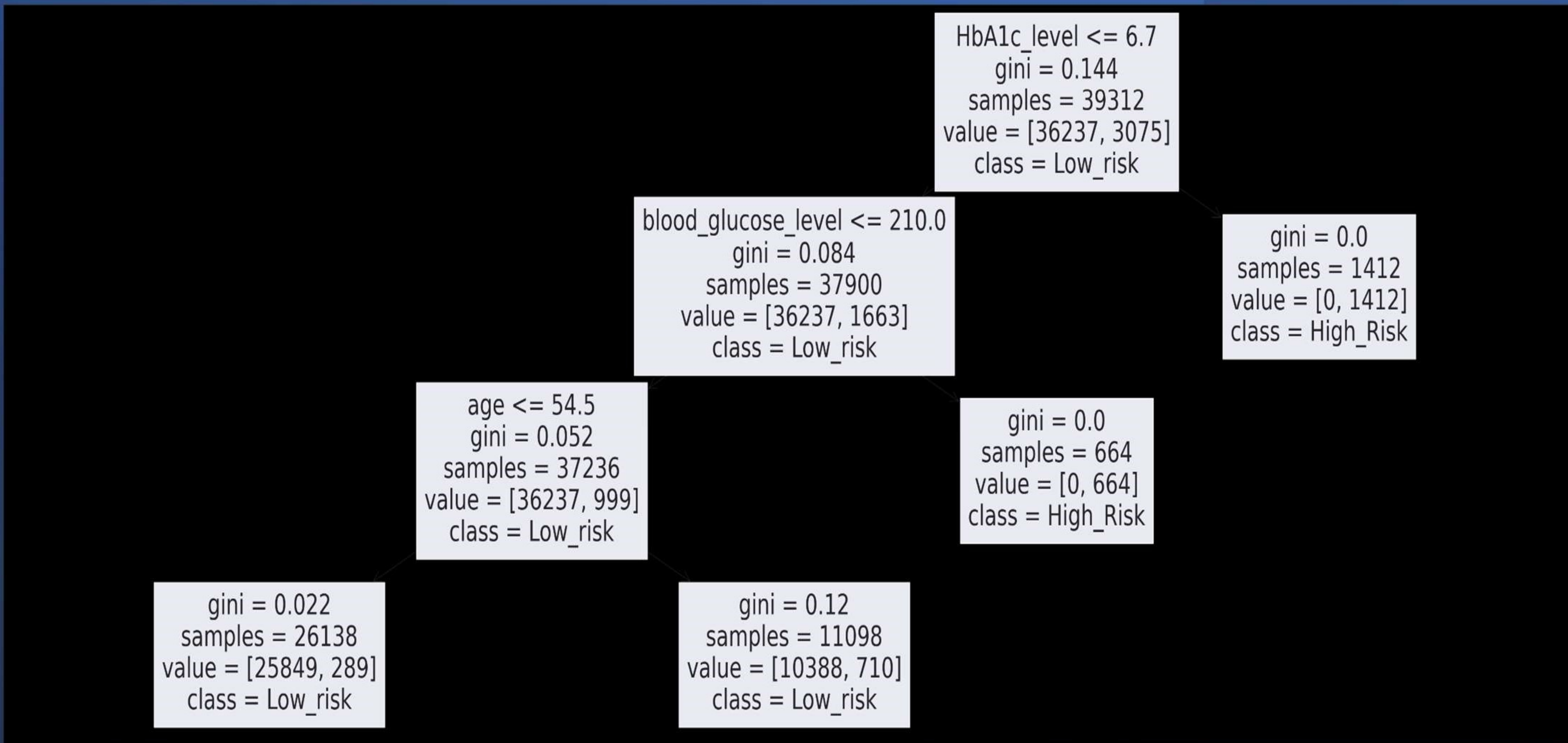
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3)
```

```
[212]
#Do predictions with decision tree using the testing data
yTestPred = decTree.predict(xTest)
```

```
[214]
#Evaluate the model
print('Confusion Matrix for Testing Data')
print(confusion_matrix(yTest, yTestPred))
```

```
Confusion Matrix for Testing Data
[[26288     0]
 [  844  1707]]
```

Decision Tree Graph



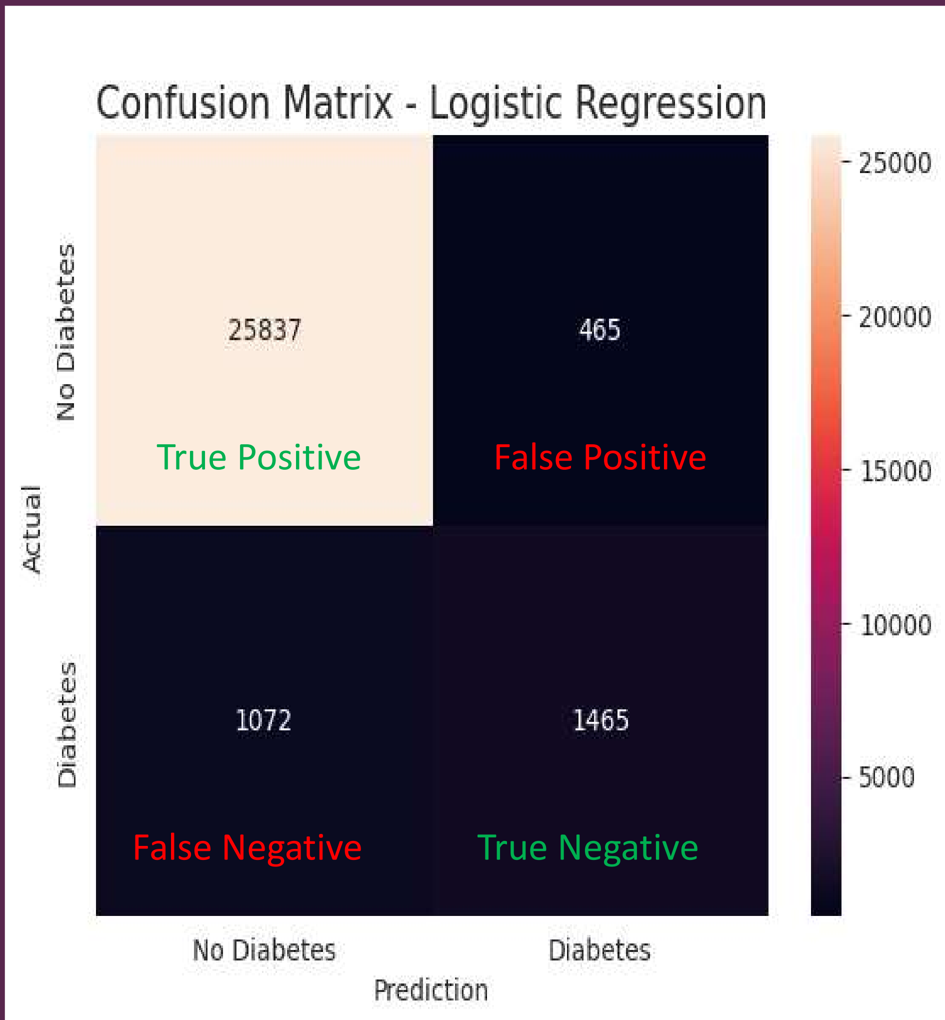
Use performance metrics to measure the model's effectiveness

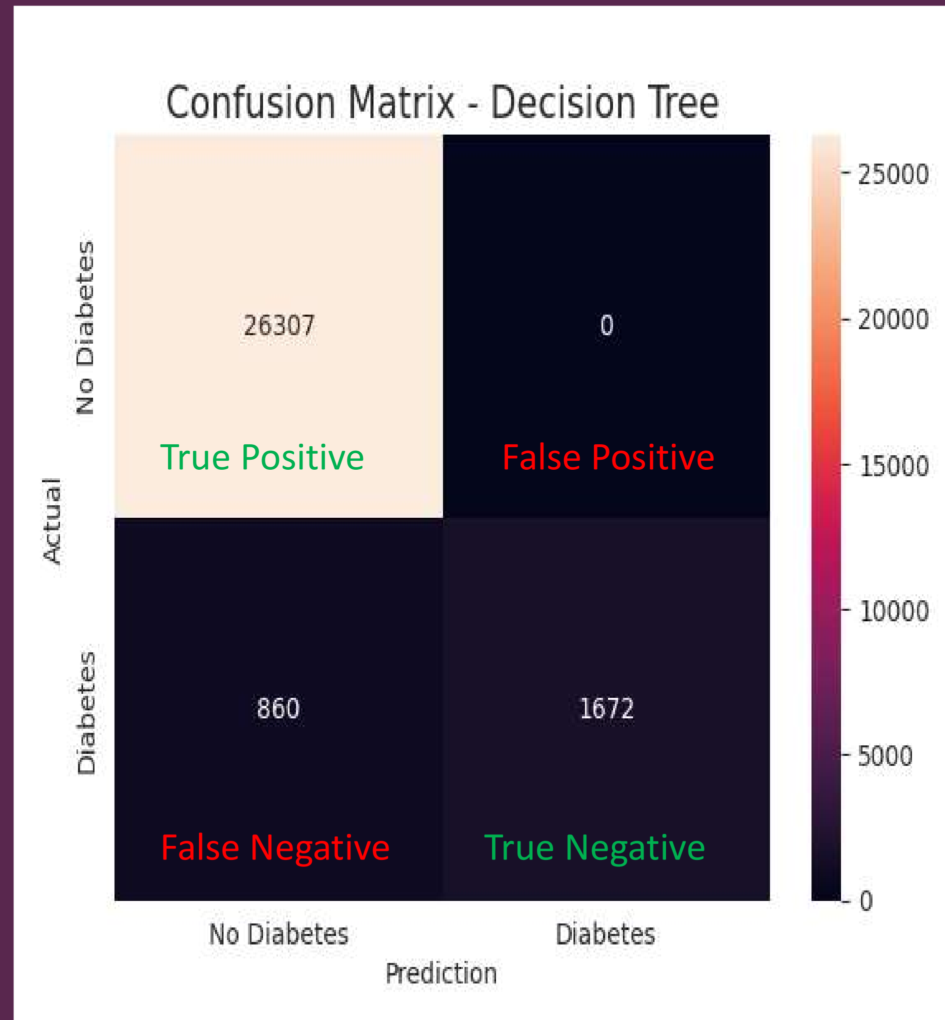
```
#classifation report for testing data
print(classification_report(yTest, yTestPred))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     26345
           1       1.00      0.66      0.79      2494

    accuracy                           0.97     28839
   macro avg       0.98      0.83      0.89     28839
weighted avg       0.97      0.97      0.97     28839
```


Confusion Matrix - Logistic Regression
Actual
No Diabetes
Diabetes
25837
True Positive
465
False Positive
1072
False Negative
1465
True Negative
No Diabetes
Diabetes
Prediction
25000
20000
15000
10000
5000


Confusion Matrix - Decision Tree
Actual
No Diabetes
Diabetes
26307
True Positive
0
False Positive
860
False Negative
1672
True Negative
No Diabetes
Diabetes
Prediction
25000
20000
15000
10000
5000
0

LOGISTIC REGRESSION VS DECISION TREE

	LOGISTIC REGRESSION	DECISION TREE
ACCURACY	95%	97%
PRECISION (0)	96%	97%
PRECISION (1)	84%	100%
F1 SCORE (0)	98%	98%
F1 SCORE (1)	71%	79%

RESULTS

- The decision tree had an overall better performance than the logistic regression
- Our findings showed that the independent variables, HbA1c and blood glucose level had the highest correlation to our dependent variable, diabetes.
- Even after dividing our dataset by gender, this relation remains true. But men had slightly higher HbA1c and blood glucose levels than women.

Limitations

- Missing data and outlines
- Complex interaction

Future Work

CLINICAL APPLICATION

ADVANCE MACHINE LEARNING MODEL

Thank You

References Kaggle.com(2023)

1. Harrell Jr., F. E. (2015). Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis (2nd Edition). Springer.
2. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd Edition). Springer.