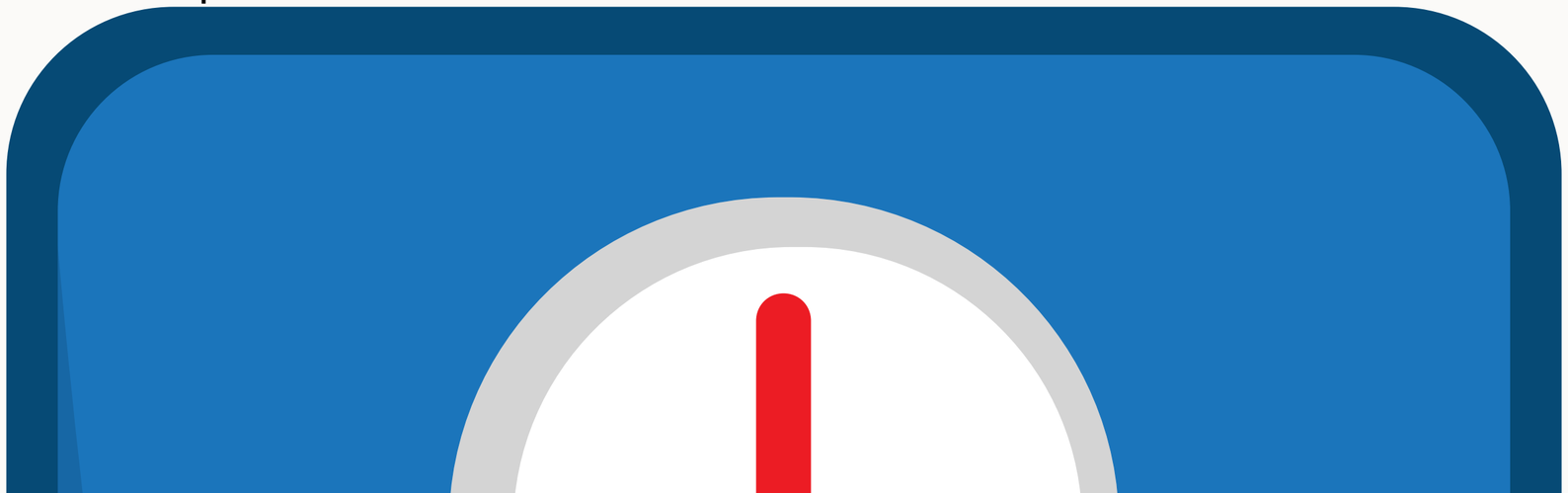
# DIABETES PREDICTIVE MODEL

*FERNANDO BUENO EIMBCKE*
*EDUARDO REA WILLIAMS*
*YAEL ANTONIO CALZADA MARTIN*
*FABIAN MORENO GARZA*

# CONTENT

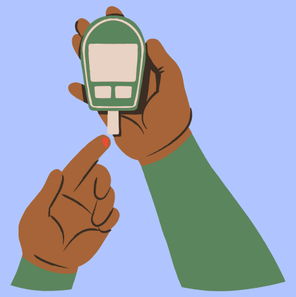
**DIABETES AFFECTS MILLIONS WORLDWIDE. OUR MODEL AIMS TO TACKLE THIS CHALLENGE BY ANALYZING VARIOUS PATIENT CHARACTERISTICS AND RISK FACTORS TO PREDICT DIABETES ONSET. THIS APPROACH IS NOT JUST ABOUT PREDICTION; IT'S A STEP TOWARDS GREATER AWARENESS AND PROACTIVE HEALTH MANAGEMENT IN THE FIGHT AGAINST DIABETES.**



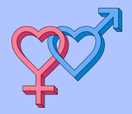


# MODEL VARIABLES

01 HBA1C LEVELS



02 GENDER



03 SMOKING HISTORY

04 HEART DISEASE

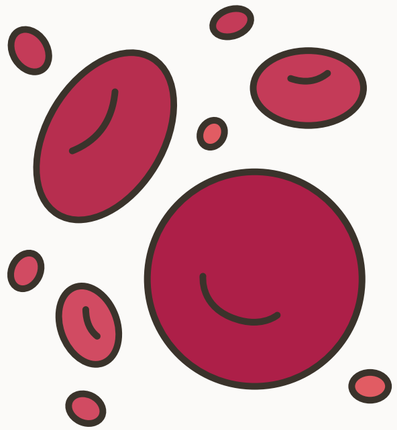
05 BMI

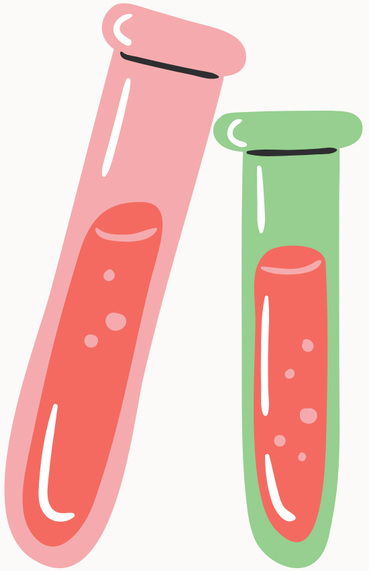

06 BLOOD GLUCOSE


07 DIABETES

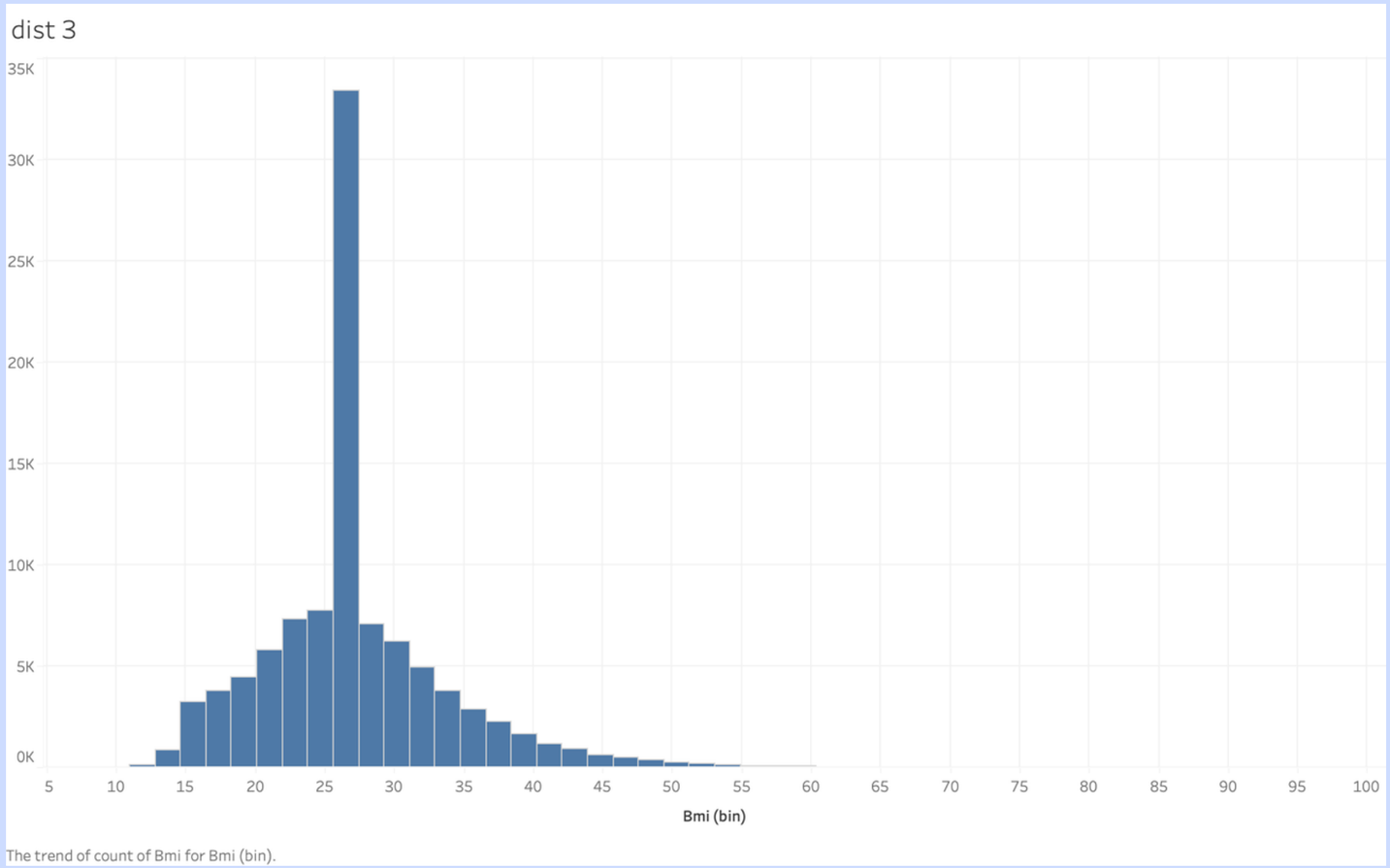
# HBA1C

HbA1c level, also known as hemoglobin A1c or glycated hemoglobin, is a measure of the average blood sugar (glucose) levels over the past two to three months. It is used primarily to monitor the long-term blood sugar control in people with diabetes. This test helps in understanding how well diabetes is being managed and can guide treatment adjustments if needed.

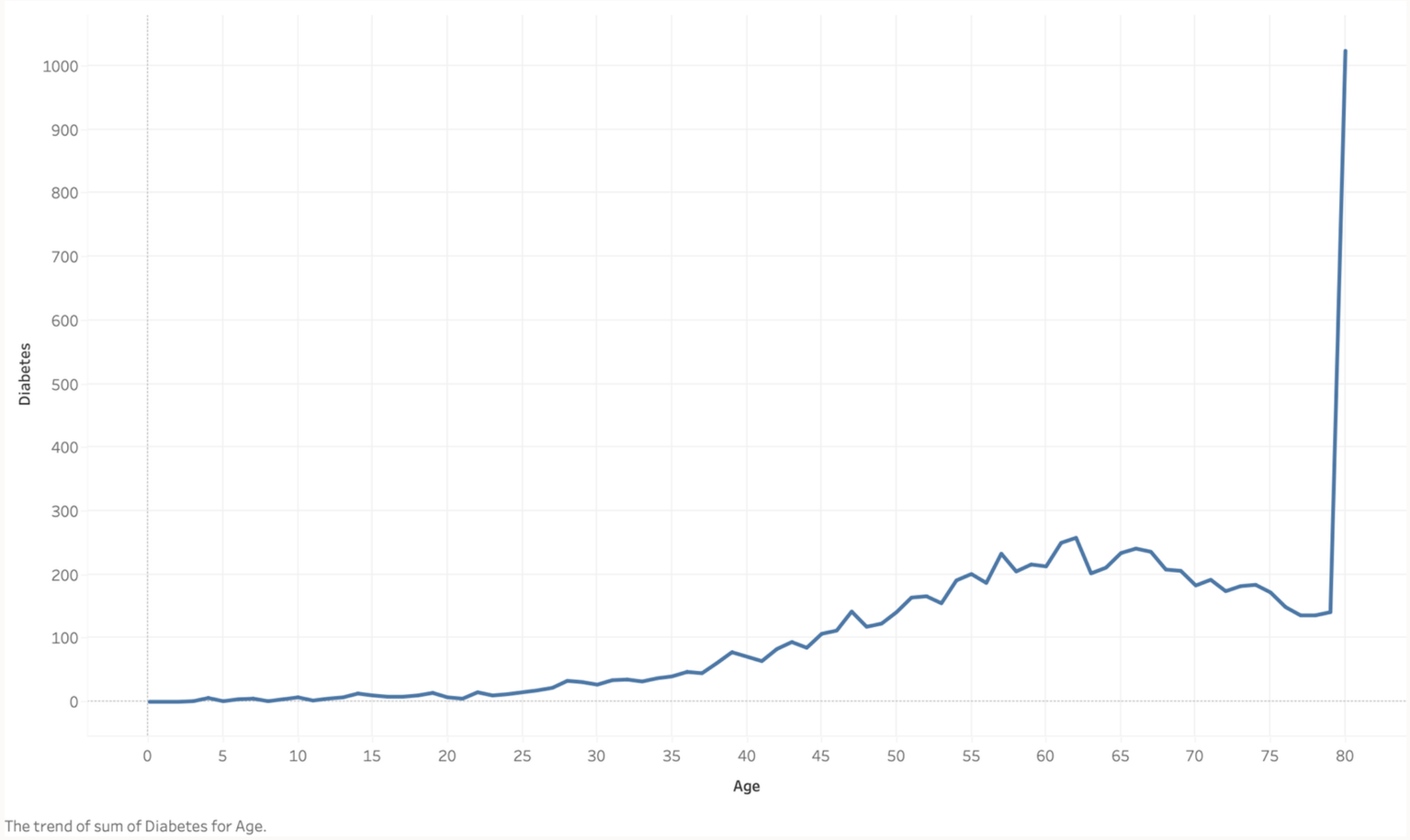




# BMI

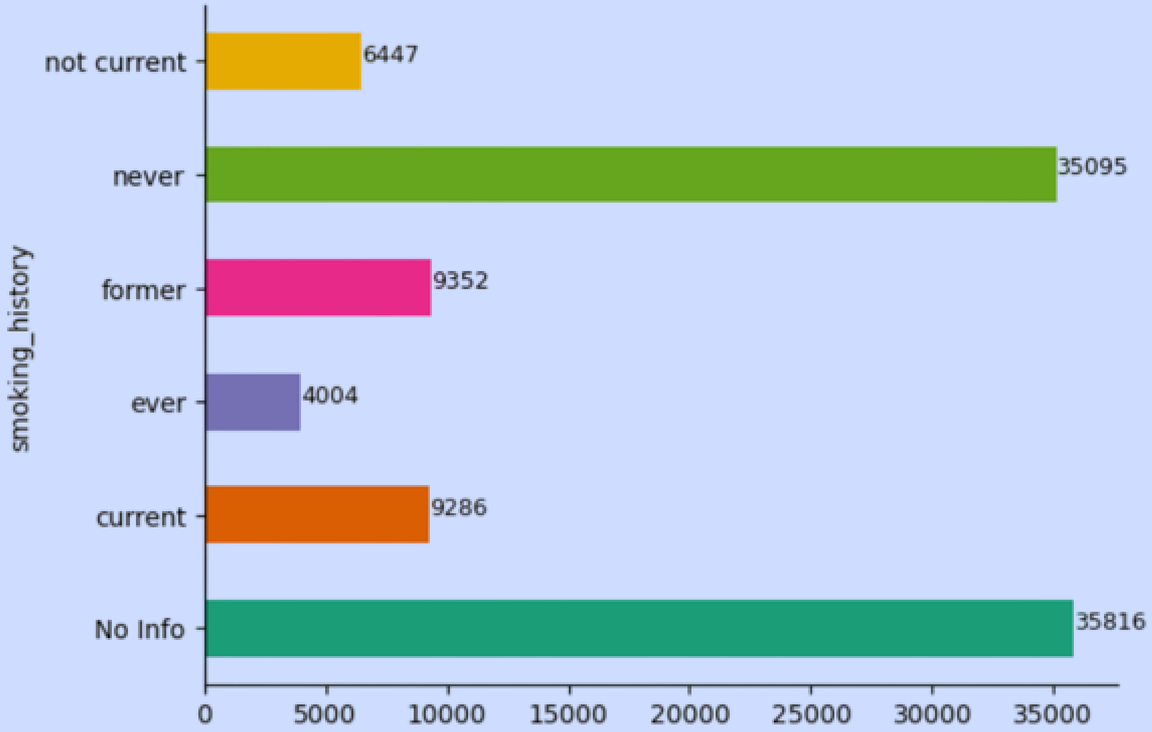

dist 3
35K
30K
25K
20K
15K
10K
5K
0K
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
Bmi (bin)
The trend of count of Bmi for Bmi (bin).

# DIABETES VS AGE

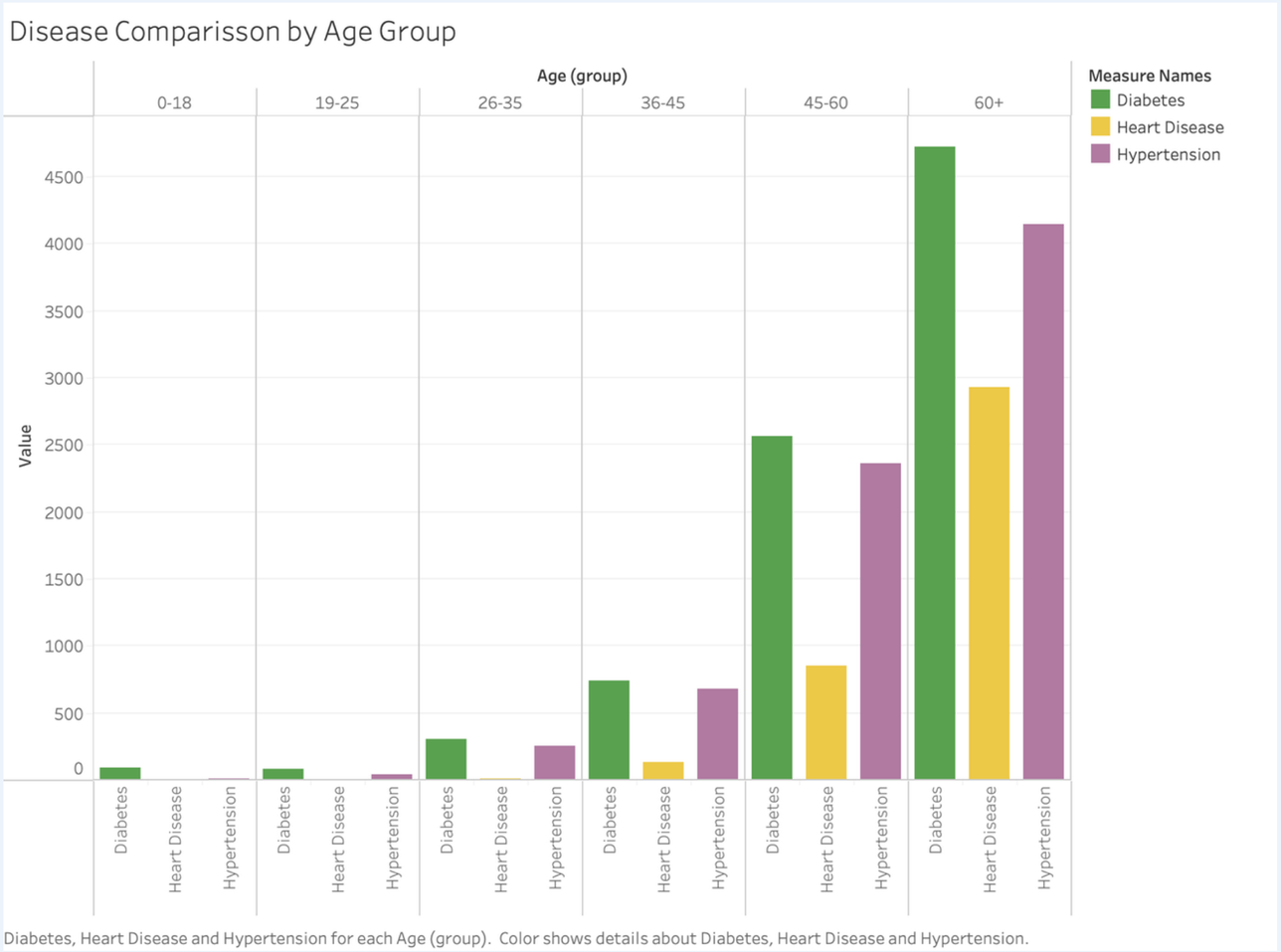



The trend of sum of Diabetes for Age.

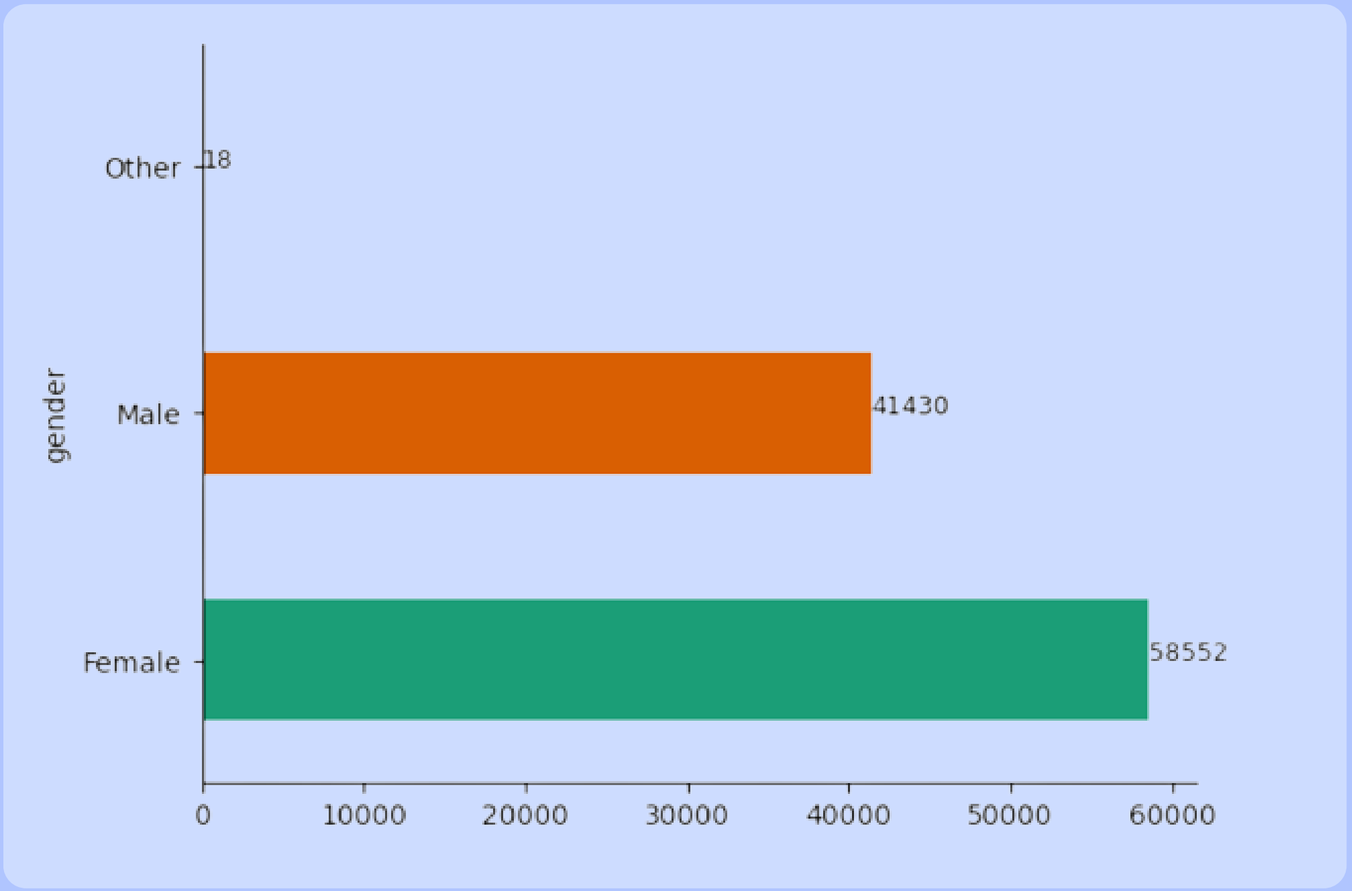
# Smoking history

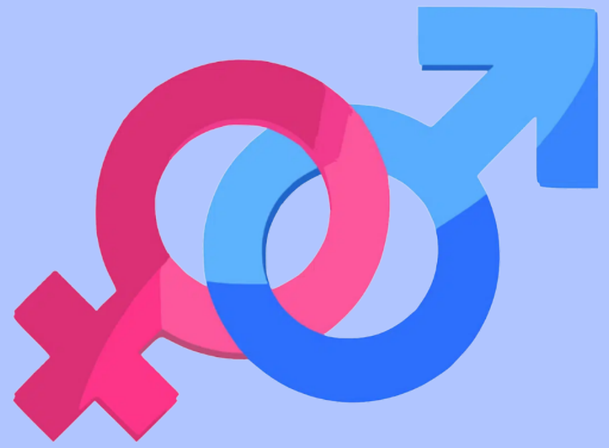



# HEART DISEASE

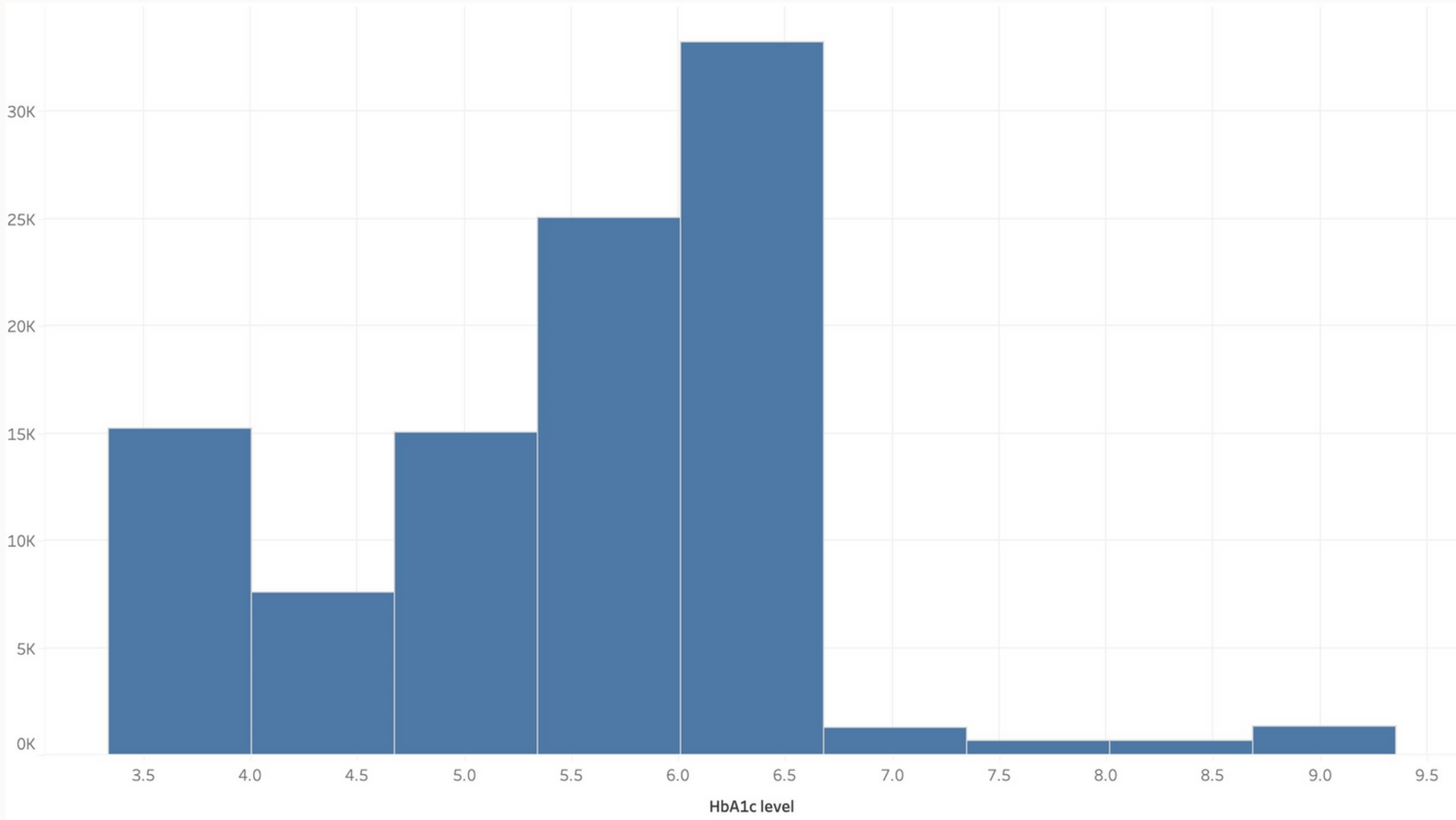



# Gender


gender
Other
Male
Female
18
41430
58552
0
10000
20000
30000
40000
50000
60000

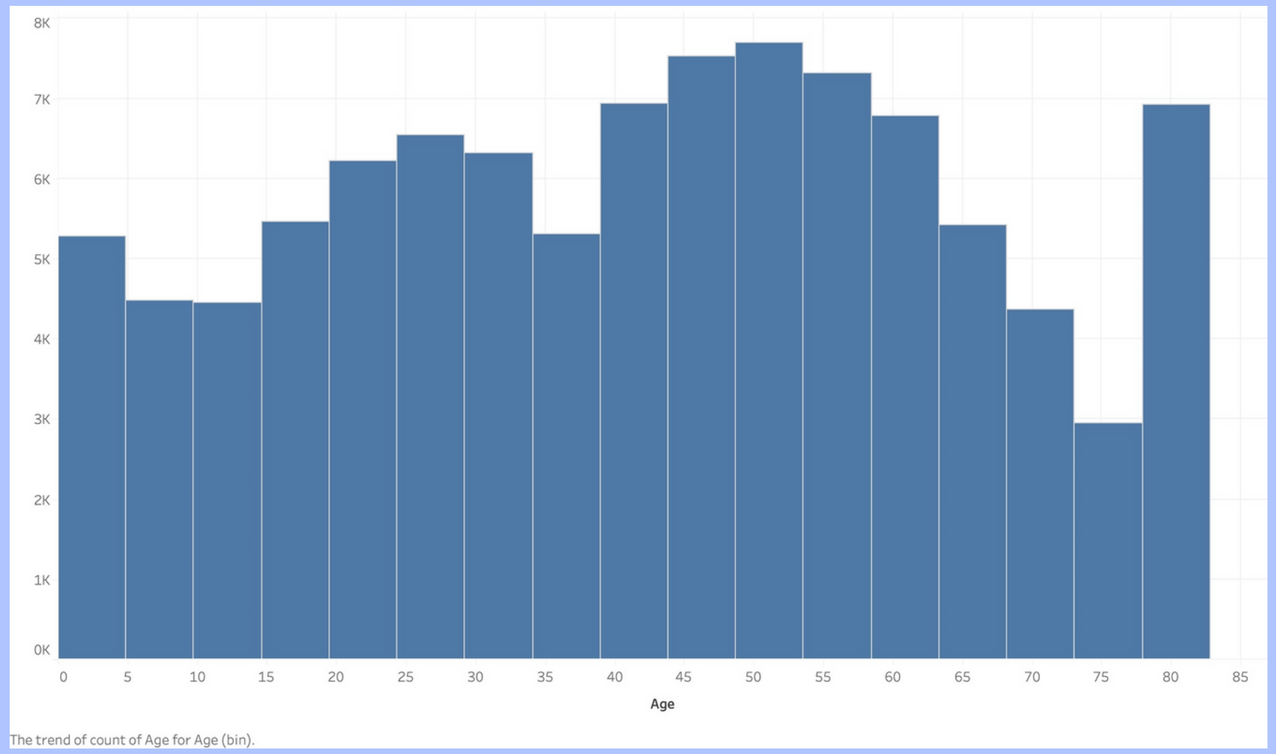


# HBA1C DISTRIBUTION



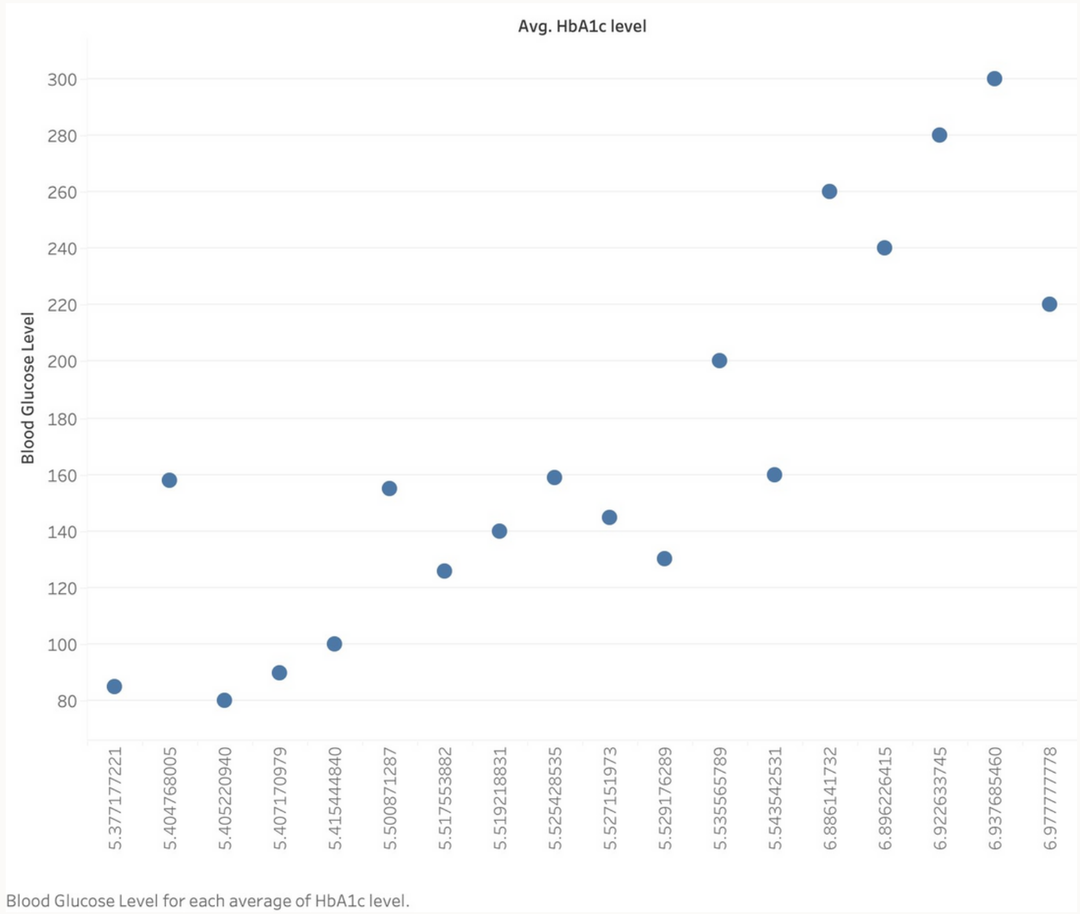

The trend of count of HbA1c level for HbA1c level (bin).

# Age



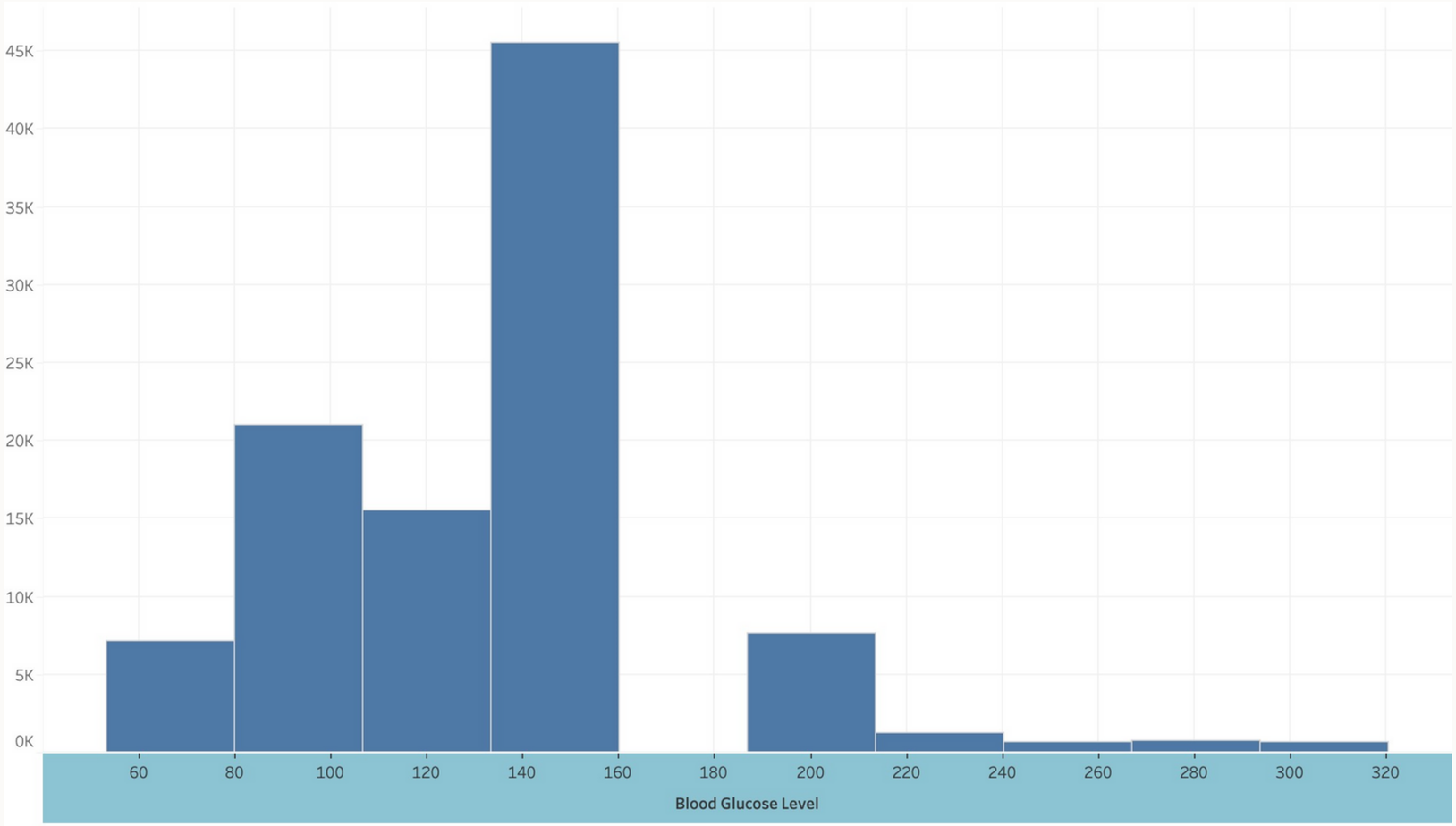

The trend of count of Age for Age (bin).

# HBA1C Levels vs. Blood Glucose Levels



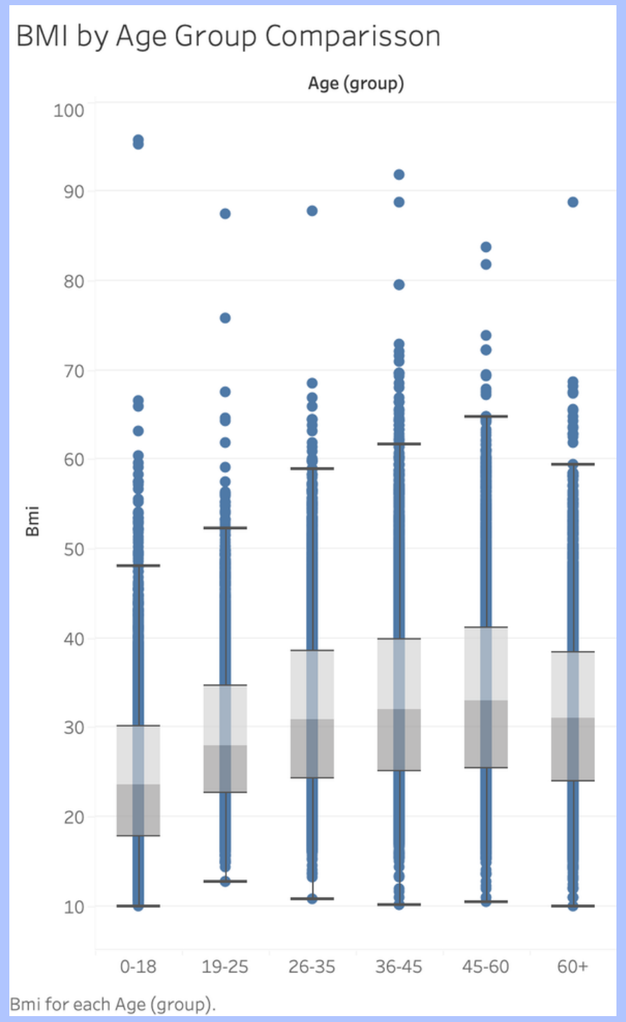

Blood Glucose Level for each average of HbA1c level.

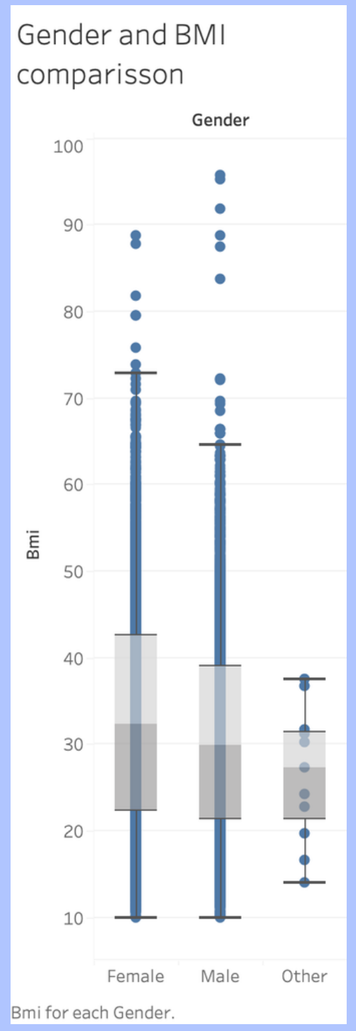
# BLOOD GLUCOSE

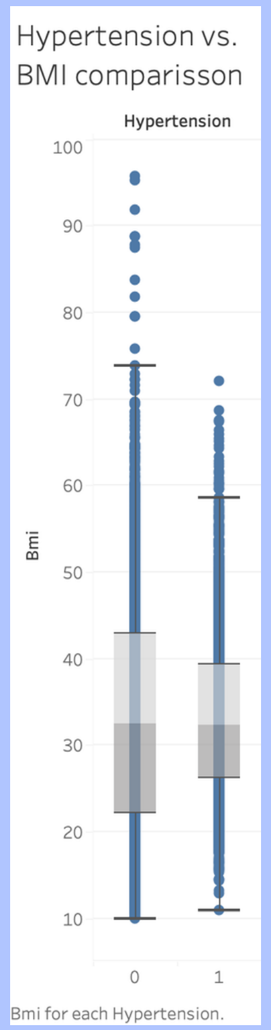



The trend of count of Blood Glucose Level for Blood Glucose Level (bin).

# Variables comparison

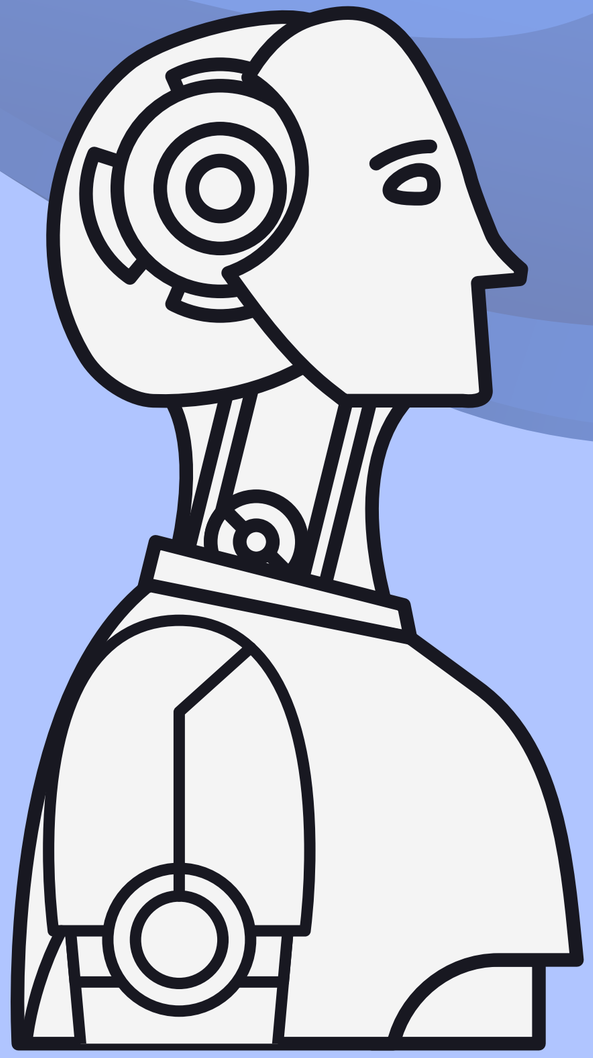



# PROCEDURE

- Load the CSV file.
- Convert it into a PySpark SQL table.
- Perform data processing and cleaning.
- Transfer the data to Pandas.
- Apply One-Hot Encoding to the categorical variables.
- Normalize the numerical variables.
- Split the data into features (X) and target (y).
- Divide the data into training and test sets.
- Train the model and evaluate its performance.
- Test the model with different data making predictions.



# MODELS EVALUIATION

## 10 Epochs

```
[72] # Evaluate the model on the test set
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f"Accuracy: {accuracy*100}%")
     print(f"Loss: {loss*100}%")

     955/955 [==============================] - 2s 2ms/step - loss: 0.1726 - accuracy: 0.9133
     Accuracy: 91.3306713104248%
     Loss: 17.26486384868622%
```

## 70 Epochs

```
[37] # Evaluate the model on the test set
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f"Accuracy: {accuracy*100}%")
     print(f"Loss: {loss*100}%")

     955/955 [==============================] - 3s 3ms/step - loss: 0.1706 - accuracy: 0.9154
     Accuracy: 91.54012203216553%
     Loss: 17.055977880954742%
```

## 32 Epochs

```
[54] # Evaluate the model on the test set
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f"Accuracy: {accuracy*100}%")
     print(f"Loss: {loss*100}%")

     955/955 [==============================] - 2s 2ms/step - loss: 0.1709 - accuracy: 0.9159
     Accuracy: 91.58921241760254%
     Loss: 17.085206508636475%
```

## 100 Epochs

```
[20] # Evaluate the model on the test set
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f"Accuracy: {accuracy*100}%")
     print(f"Loss: {loss*100}%")

     955/955 [==============================] - 2s 2ms/step - loss: 0.1720 - accuracy: 0.9140
     Accuracy: 91.39612317085266%
     Loss: 17.197634279727936%
```

## 50 Epochs

```
[24] # Evaluate the model on the test set
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f"Accuracy: {accuracy*100}%")
     print(f"Loss: {loss*100}%")

     955/955 [==============================] - 2s 2ms/step - loss: 0.1715 - accuracy: 0.9135
     Accuracy: 91.35030508041382%
     Loss: 17.153489589691162%
```

# MODEL PACIENT 1

## 10 Epochs

```
[76] print(f"Predicted probability of having diabetes: {prediction[0][0] * 100}%")

     Predicted probability of having diabetes: 77.41220593452454%

[77] # Interpretar la predicción basada en el umbral del 65%
     predicted_diabetes = (prediction[0][0] > 0.65).astype(int)
     # Print a personalized message based on the prediction
     if predicted_diabetes == 1:
         print("Diabetes Prediction: Probably yes")
     else:
         print("Diabetes Prediction: Probably no")

     Diabetes Prediction: Probably yes
```

```
# Make an example prediction
new_patient = pd.DataFrame({
    'age': [80],
    'hypertension': [0],
    'heart_disease': [1],
    'bmi': [25.19],
    'HbA1c_level': [6.6],
    'blood_glucose_level': [140],
    'gender_Female': [1],
    'gender_Male': [0],
    'gender_Other': [0],
    'smoking_history_No Info': [0],
    'smoking_history_current': [0],
    'smoking_history_ever': [0],
    'smoking_history_former': [0],
    'smoking_history_never': [1],
    'smoking_history_not current': [0]
})
```

## 70 Epochs

```
print(f"Predicted probability of having diabetes: {prediction[0][0] * 100}%")

Predicted probability of having diabetes: 77.88918614387512%

# Interpretar la predicción basada en el umbral del 65%
predicted_diabetes = (prediction[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")

Diabetes Prediction: Probably yes
```

## 32 Epochs

```
print(f"Predicted probability of having diabetes: {prediction[0][0] * 100}%")

Predicted probability of having diabetes: 84.32970643043518%

# Interpretar la predicción basada en el umbral del 65%
predicted_diabetes = (prediction[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")

Diabetes Prediction: Probably yes
```

## 50 Epochs

```
[28] print(f"Predicted probability of having diabetes: {prediction[0][0] * 100}%")

     Predicted probability of having diabetes: 63.7865424156189%

[29] # Interpretar la predicción basada en el umbral del 65%
     predicted_diabetes = (prediction[0][0] > 0.65).astype(int)
     # Print a personalized message based on the prediction
     if predicted_diabetes == 1:
         print("Diabetes Prediction: Probably yes")
     else:
         print("Diabetes Prediction: Probably no")

     Diabetes Prediction: Probably no
```

## 100 Epochs

```
print(f"Predicted probability of having diabetes: {prediction[0][0] * 100}%")

Predicted probability of having diabetes: 76.34917497634888%

# Interpretar la predicción basada en el umbral del 65%
predicted_diabetes = (prediction[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")

Diabetes Prediction: Probably yes
```

# MODEL PACIENT 2

## 10 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 3.273367471176569e-10%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

```
# Make an example prediction
new_patient2 = pd.DataFrame({
    'age': [20],
    'hypertension': [0],
    'heart_disease': [0],
    'bmi': [23.86],
    'HbA1c_level': [5.7],
    'blood_glucose_level': [85],
    'gender_Female': [1],
    'gender_Male': [0],
    'gender_Other': [0],
    'smoking_history_No Info': [0],
    'smoking_history_current': [0],
    'smoking_history_ever': [0],
    'smoking_history_former': [0],
    'smoking_history_never': [1],
    'smoking_history_not current': [0]
})
```

## 70 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 1.621143150276616e-09%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

## 32 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 1.1252598852706797e-08%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

## 50 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 6.519260500418738e-18%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

## 100 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 3.738927223384536e-08%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

# MODEL PACIENT 3

## 10 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 9.18702632188797%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

```
# Make an example prediction
new_patient2 = pd.DataFrame({
    'age': [44],
    'hypertension': [0],
    'heart_disease': [0],
    'bmi': [27.32],
    'HbA1c_level': [6.5],
    'blood_glucose_level': [200],
    'gender_Female': [1],
    'gender_Male': [0],
    'gender_Other': [0],
    'smoking_history_No Info': [0],
    'smoking_history_current': [0],
    'smoking_history_ever': [0],
    'smoking_history_former': [0],
    'smoking_history_never': [1],
    'smoking_history_not current': [0]
})
```

## 70 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 9.684423357248306%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

## 32 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 11.815376579761505%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

## 50 Epochs

```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 15.794315934181213%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

## 100 Epochs

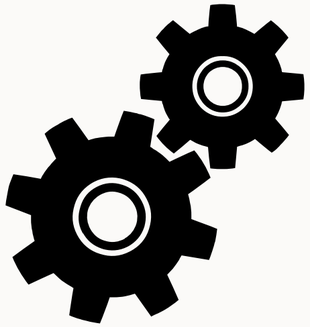
```
print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")
```

```
Predicted probability of having diabetes: 7.330609858036041%
```

```
# Interpret prediction based on 65% threshold
predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
# Print a personalized message based on the prediction
if predicted_diabetes2 == 1:
    print("Diabetes Prediction: Probably yes")
else:
    print("Diabetes Prediction: Probably no")
```

```
Diabetes Prediction: Probably no
```

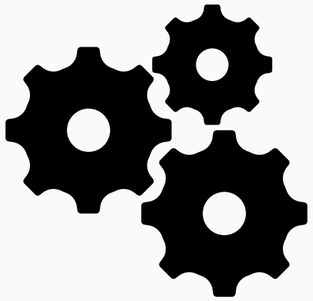
# MODEL SMALL DATA



## 10 Epochs with Small data

```
[20] # Evaluate the model on the test set
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f"Accuracy: {accuracy*100}%")
     print(f"Loss: {loss*100}%")

     110/110 [==============================] - 0s 2ms/step - loss: 0.2295 - accuracy: 0.8911
     Accuracy: 89.10806179046631%
     Loss: 22.94875532388687%
```



## 10 Epochs with Big data

```
[72] # Evaluate the model on the test set
     loss, accuracy = model.evaluate(X_test, y_test)
     print(f"Accuracy: {accuracy*100}%")
     print(f"Loss: {loss*100}%")

     955/955 [==============================] - 2s 2ms/step - loss: 0.1726 - accuracy: 0.9133
     Accuracy: 91.3306713104248%
     Loss: 17.26486384868622%
```

# MODEL SMALL DATA

## 10 Epochs

```
# Make an example prediction
new_patient = pd.DataFrame({
    'age': [80],
    'hypertension': [0],
    'heart_disease': [1],
    'bmi': [25.19],
    'HbA1c_level': [6.6],
    'blood_glucose_level': [140],
    'gender_Female': [1],
    'gender_Male': [0],
    'gender_Other': [0],
    'smoking_history_No Info': [0],
    'smoking_history_current': [0],
    'smoking_history_ever': [0],
    'smoking_history_former': [0],
    'smoking_history_never': [1],
    'smoking_history_not current': [0]
})
```

```
[74] # Normalize new patient data
     # We use the same scaler that I used for the training data
     new_patient[['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']] = scaler.transform(new_patient[['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']])
```

```
[75] # Make the prediction with the model
     prediction = model.predict(new_patient)

     1/1 [==============================] - 0s 33ms/step
```

```
[76] print(f"Predicted probability of having diabetes: {prediction[0][0] * 100}%")

     Predicted probability of having diabetes: 77.41220593452454%
```

```
[77] # Interpretar la predicción basada en el umbral del 65%
     predicted_diabetes = (prediction[0][0] > 0.65).astype(int)
     # Print a personalized message based on the prediction
     if predicted_diabetes == 1:
         print("Diabetes Prediction: Probably yes")
     else:
         print("Diabetes Prediction: Probably no")

     Diabetes Prediction: Probably yes
```

# MODEL SMALL DATA - PACIENT 2

## 10 Epochs

```
# Make an example prediction
new_patient2 = pd.DataFrame({
    'age': [20],
    'hypertension': [0],
    'heart_disease': [0],
    'bmi': [23.86],
    'HbA1c_level': [5.7],
    'blood_glucose_level': [85],
    'gender_Female': [1],
    'gender_Male': [0],
    'gender_Other': [0],
    'smoking_history_No Info': [0],
    'smoking_history_current': [0],
    'smoking_history_ever': [0],
    'smoking_history_former': [0],
    'smoking_history_never': [1],
    'smoking_history_not current': [0]
})
```

```
[79] # Normalize new patient data
     # We use the same scaler that I used for the training data
     new_patient2[['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']] = scaler.transform(new_patient2[['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']])
```

```
[80] # Make the prediction with the model
     prediction2 = model.predict(new_patient2)

     1/1 [==============================] - 0s 19ms/step
```

```
[81] print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")

     Predicted probability of having diabetes: 3.273367471176569e-10%
```

```
[82] # Interpret prediction based on 65% threshold
     predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
     # Print a personalized message based on the prediction
     if predicted_diabetes2 == 1:
         print("Diabetes Prediction: Probably yes")
     else:
         print("Diabetes Prediction: Probably no")

     Diabetes Prediction: Probably no
```

# MODEL SMALL DATA - PACIENT 3

## 10 Epochs

```
[83] # Make an example prediction
     new_patient2 = pd.DataFrame({
         'age': [44],
         'hypertension': [0],
         'heart_disease': [0],
         'bmi': [27.32],
         'HbA1c_level': [6.5],
         'blood_glucose_level': [200],
         'gender_Female': [1],
         'gender_Male': [0],
         'gender_Other': [0],
         'smoking_history_No Info': [0],
         'smoking_history_current': [0],
         'smoking_history_ever': [0],
         'smoking_history_former': [0],
         'smoking_history_never': [1],
         'smoking_history_not current': [0]
     })
```

```
[84] # Normalize new patient data
     # We use the same scaler that I used for the training data
     new_patient2[['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']] = scaler.transform(new_patient2[['age', 'bmi', 'HbA1c_level', 'blood_glucose_level']])
```

```
[85] # Make the prediction with the model
     prediction2 = model.predict(new_patient2)

     1/1 [==============================] - 0s 24ms/step
```

```
[86] print(f"Predicted probability of having diabetes: {prediction2[0][0] * 100}%")

     Predicted probability of having diabetes: 9.18702632188797%
```

```
     # Interpret prediction based on 65% threshold
     predicted_diabetes2 = (prediction2[0][0] > 0.65).astype(int)
     # Print a personalized message based on the prediction
     if predicted_diabetes2 == 1:
         print("Diabetes Prediction: Probably yes")
     else:
         print("Diabetes Prediction: Probably no")

     Diabetes Prediction: Probably no
```

# CONCLUSION

**The model using balanced data without repetition achieved greater accuracy, successfully predicting 4 out of 6 tests. In contrast, the model with balanced data, which involved data elimination and no duplication, achieved a success rate of 3 out of 6. The optimal number of epochs for training the model was determined to be 10 epochs and 50 epochs. Also, more information such as other factors like genetics, accurate smoking exposure history, diet, etc., is needed for the model to be even more precise and to avoid making assumptions.**