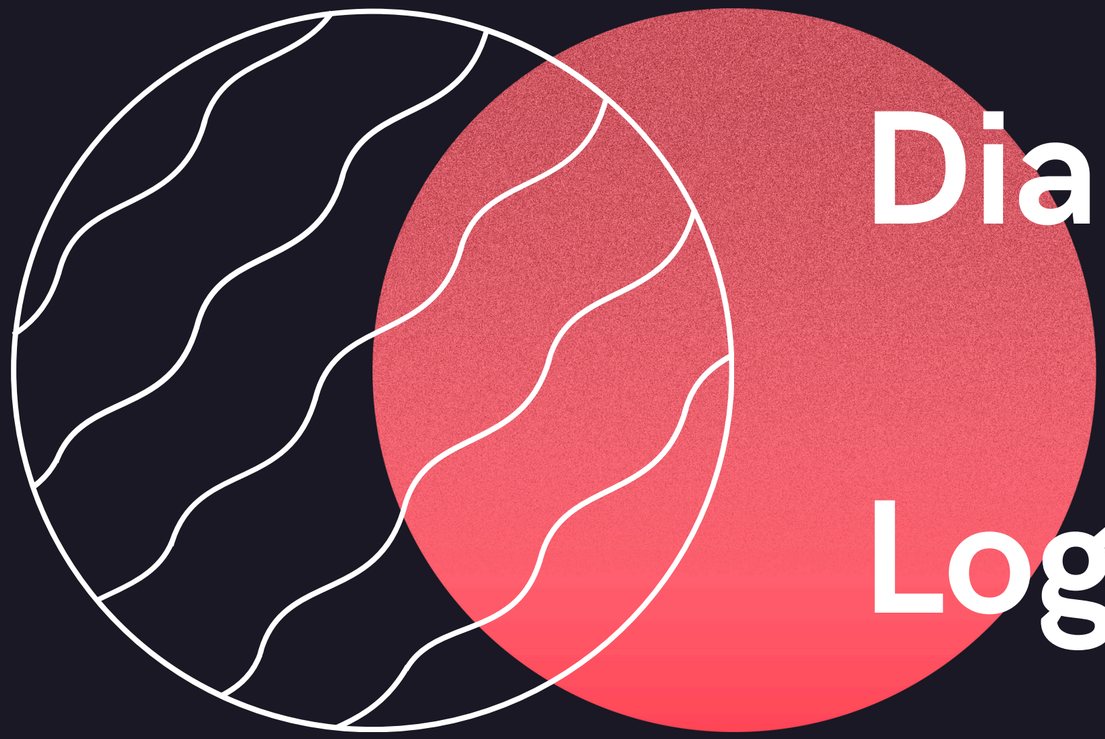


Diabetes Prediction - Logistic Regression

Goal

Our goal is to use machine learning to develop an effective model for diabetes prediction

Agenda:

- Exploratory Data Analysis
- Data Preparation
- Model Training
- Evaluate Model
- Model Deployment
- Conclusions and Limitations

RAW DATA

```
# Display the first few rows
df.head()
```

✓ 0.0s

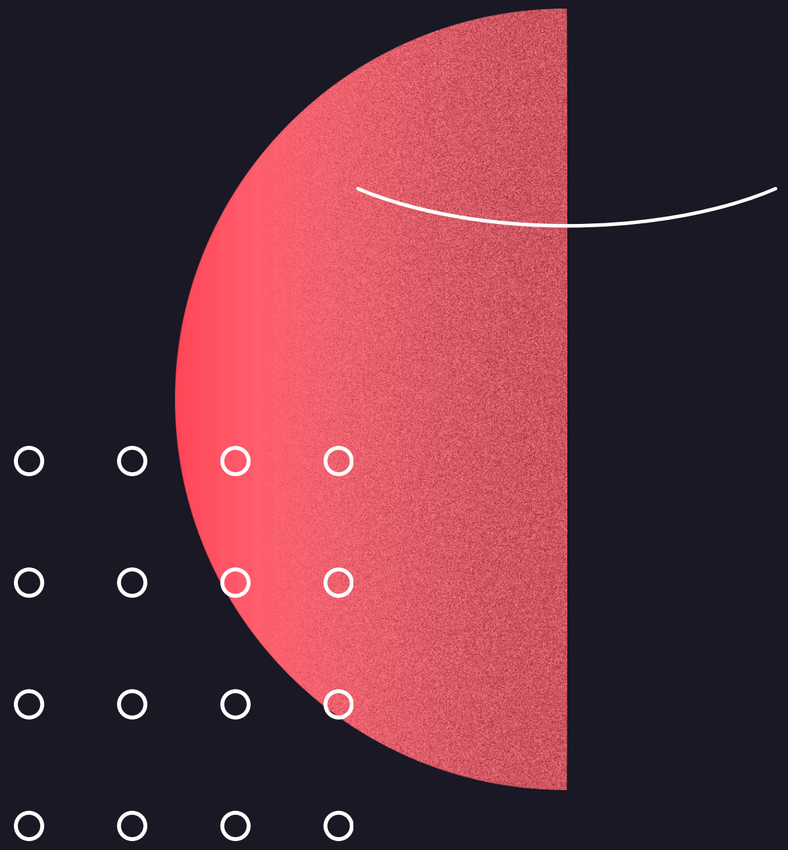
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Shape: (768, 9)

Outcome:
1: Diabetic
0: Non-Diabetic

Context: Dataset taken from women in India

Exploratory Data Analysis



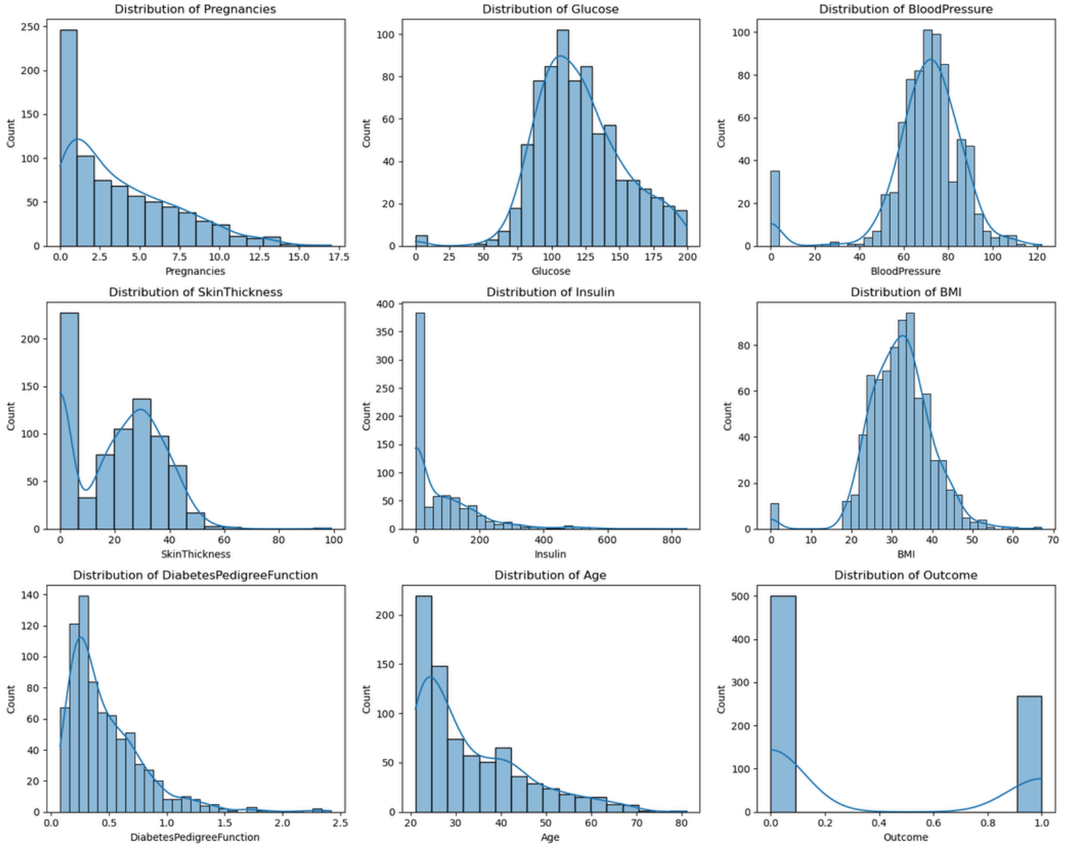
DISTRIBUTION OF FEATURES

- **Pregnancies, SkinThickness, Insulin, Age and DiabetesPedigreeFunction:**

These features are positively skewed, with most values concentrated at lower ranges and tails extending to the right.

- **Glucose, BloodPressure, and BMI:**

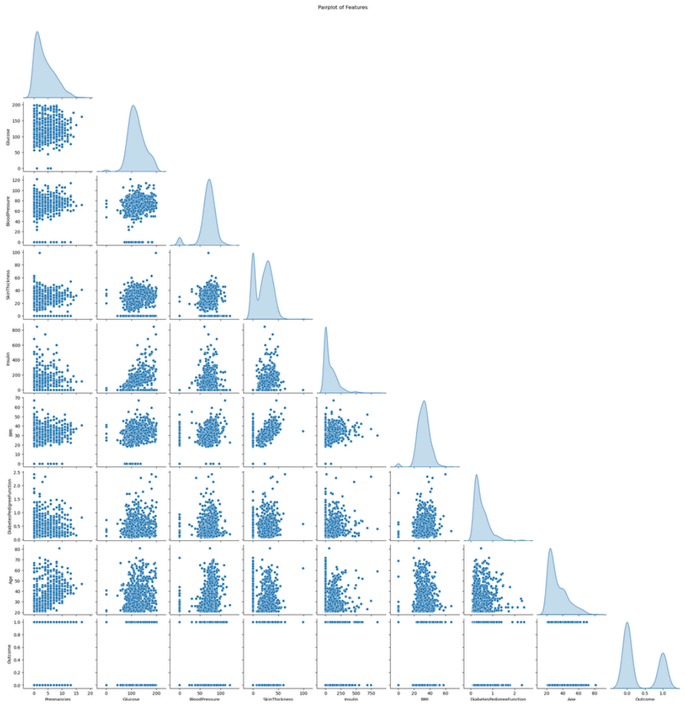
These features exhibit approximately normal distributions, though with some noticeable outliers at extreme values.

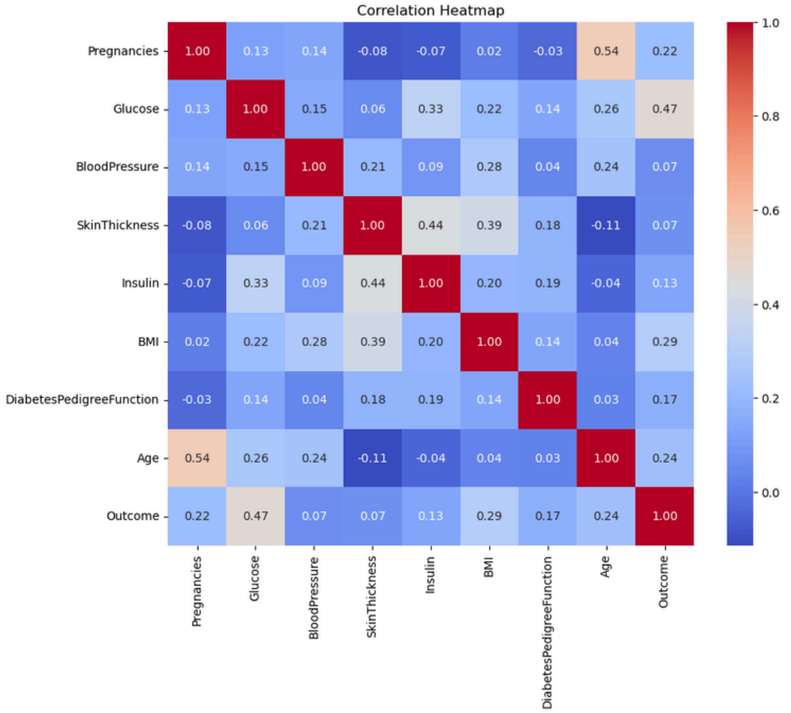


CORRELATION BETWEEN FEATURES

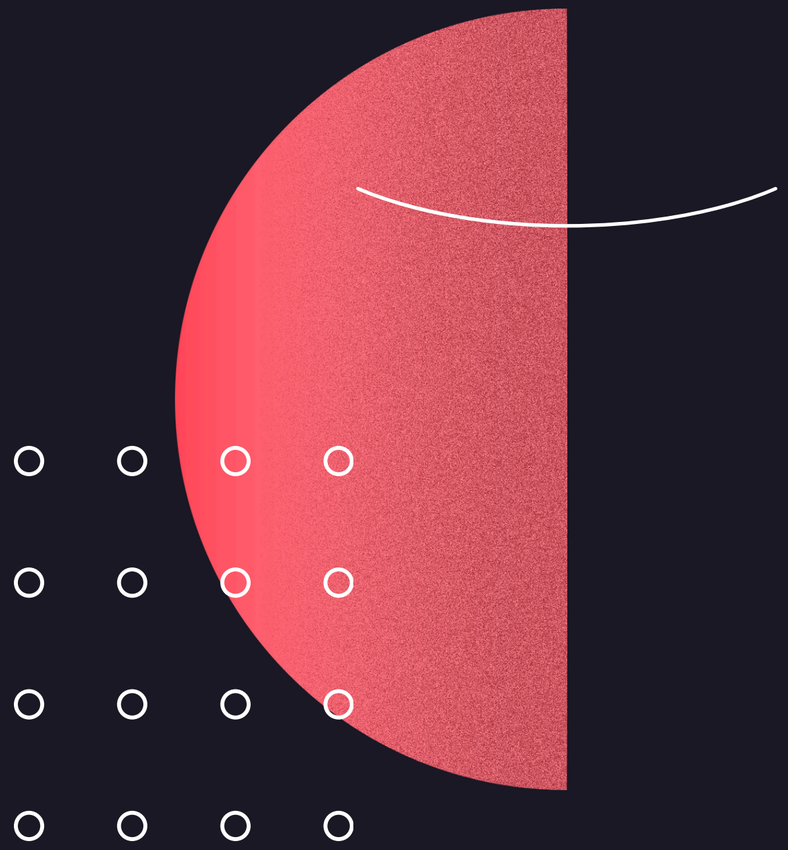
- Glucose, BMI, Age, and Pregnancies show the strongest relationships with the target variable (Outcome), indicating these features might be the most predictive for diabetes classification.

- Moderate correlations (e.g., SkinThickness and BMI, Insulin and SkinThickness) suggest some redundancy, which are addressed during feature selection.





Data Preparation



OUTLIERS

```python
# Replace zero values with NaN for relevant columns
zero_columns = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
df[zero_columns] = df[zero_columns].replace(0, np.nan)

for column in zero_columns:
    df[column].fillna(df[column].median(), inplace = True)
```

✓ 0.0s

- Replace zero values with NaN for relevant columns: Glucose, BloodPressure, SkinThickness, Insulin, and BMI.
- Fill missing values (NaN) with the median of the respective column to handle outliers and ensure robust statistics.

FEATURE SCALING

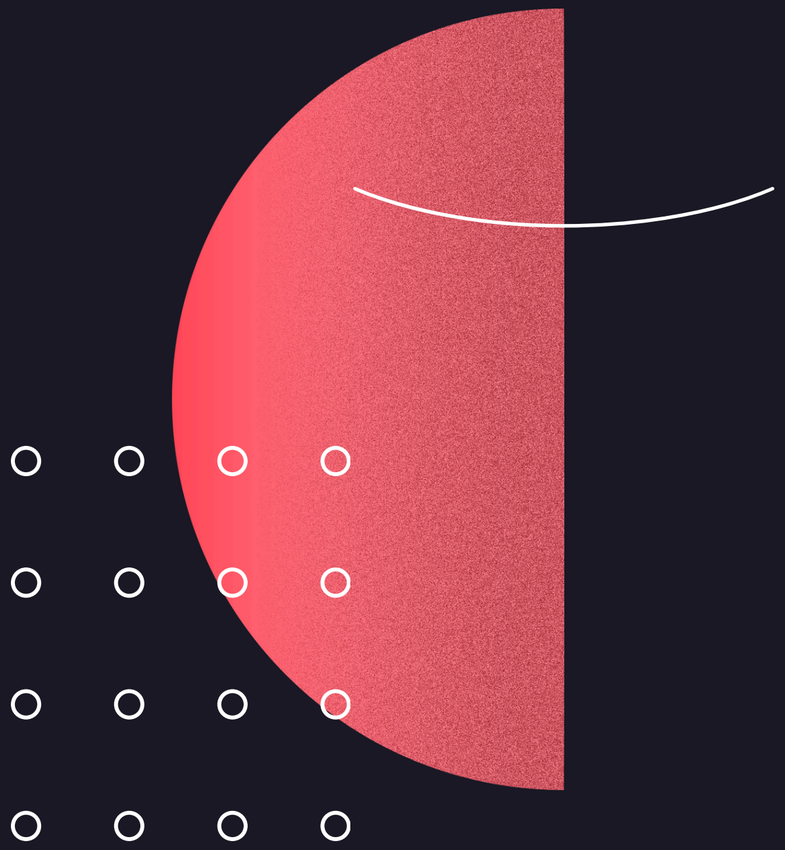
$$Z = \frac{x - \mu}{\sigma}$$

```python
# Standardization (Z-score normalization because we are predicting a boolean outcome with a logistic regression)
scaler = StandardScaler()
df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']] = scaler.fit_transform(
    df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]
)
```

✓ 0.0s

- Use standardization (Z-score normalization) to scale numerical features for better performance in models like Logistic Regression.
- Scaled features ensure equal importance of features regardless of their original scales.

Model Training



Defining Features and Targets

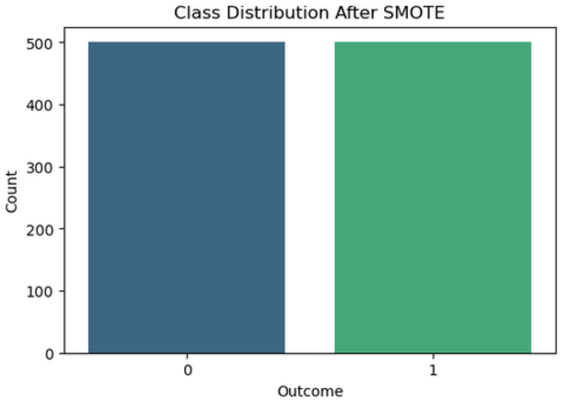
We define the Outcome Column as the target (y)
The rest of the columns are the features (X)

Handling Class Imbalance (SMOTE)

Since the dataset is strongly biased (0: 500, 1: 268),
we use SMOTE to rebalance the dataset

```
Before SMOTE:
Outcome
0    500
1    268
Name: count, dtype: int64

After SMOTE:
Outcome
1    500
0    500
Name: count, dtype: int64
```



Regularization

We use Elastic Net for Regularization

```
Fitting 10 folds for each of 36 candidates, totalling 360 fits
Best ElasticNet Parameters: {'alpha': 0.1, 'l1_ratio': 0.1}
Best ElasticNet Score: -0.1780
```

Feature Selection

We use the Elastic Net ratios to find the best features

```
Selected Features:
['Pregnancies', 'Glucose', 'SkinThickness', 'BMI', 'DiabetesPedigreeFunction', 'Age']
```

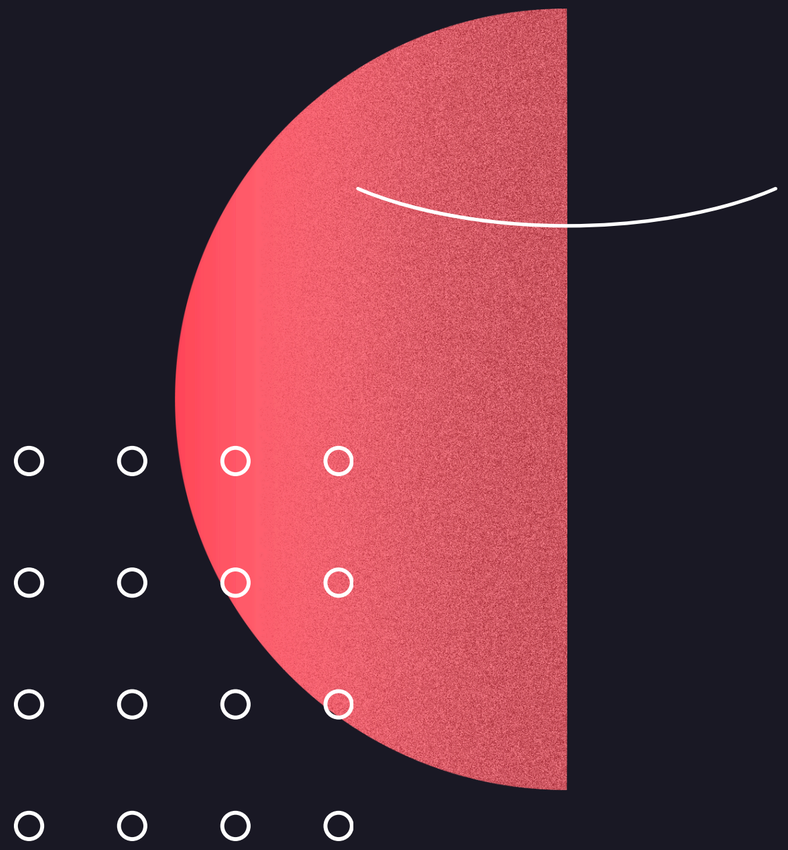
Hyperparameter Tuning

We performed hyperparameter tuning using GridSearchCV with cross-validation, systematically testing combinations of parameters like:

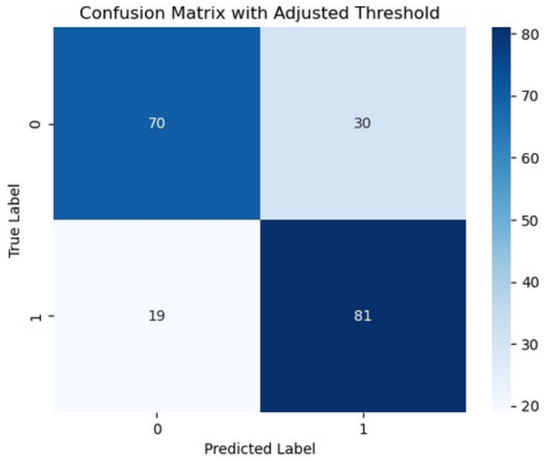
- Regularization strength (C)
- Penalty types (l1, l2, elasticnet)
- Solvers (liblinear, saga)

optimizing the model based on the ROC-AUC score to improve classification performance.

Evaluate Model

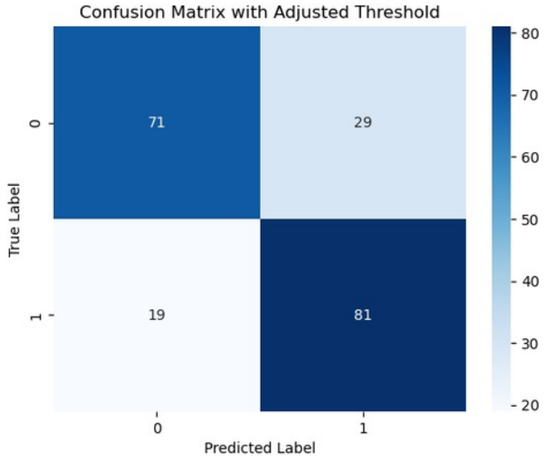


MODEL 1: SELECTED FEATURES



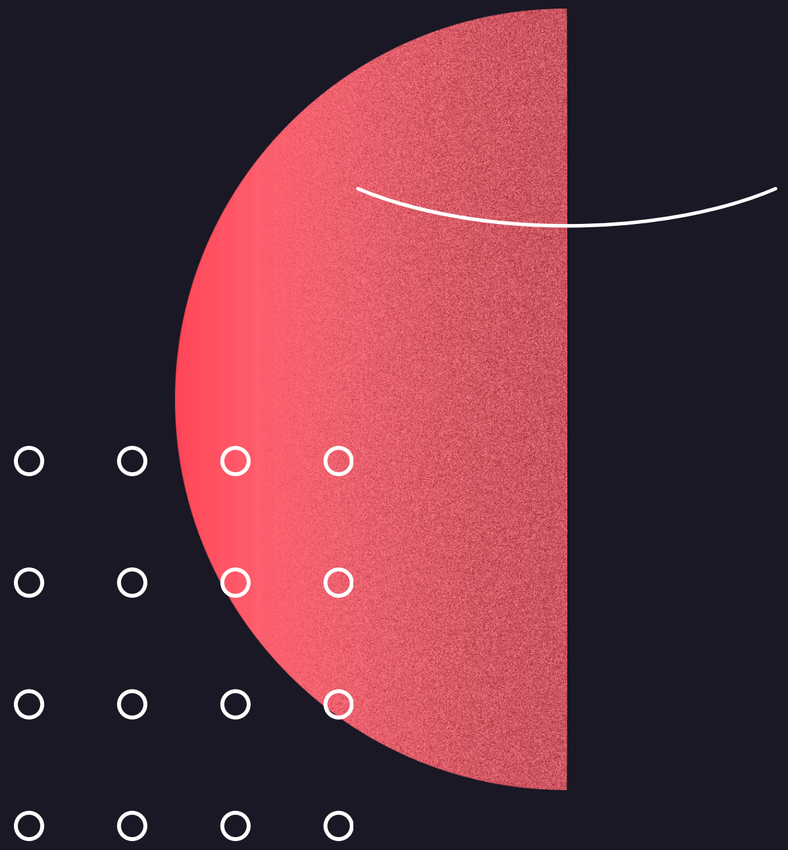
VS.

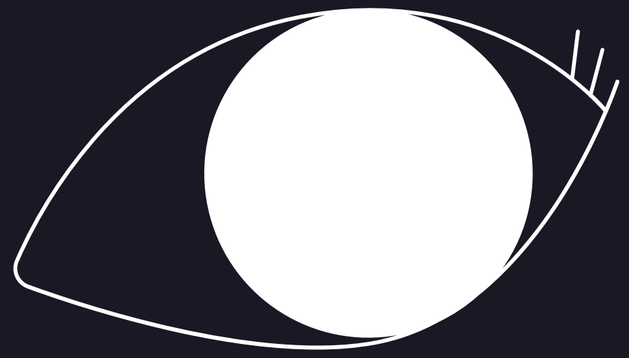
MODEL 2: ALL FEATURES



Metric	Matrix 1: Selected Features	Matrix 2: All Features	Conclusions
True Negatives (TN)	70 (Correct Non-Diabetic Predictions)	71 (Correct Non-Diabetic Predictions)	All Features (fewer false positives)
False Positives (FP)	30 (Non-Diabetic predicted as Diabetic)	29 (Non-Diabetic predicted as Diabetic)	All Features (better precision)
True Positives (TP)	81 (Correct Diabetic Predictions)	81 (Correct Diabetic Predictions)	Same
False Negatives (FN)	19 (Diabetic predicted as Non-Diabetic)	19 (Diabetic predicted as Non-Diabetic)	Same
Precision	0.730	0.736	All Features (slightly higher)
Recall	0.81	0.81	Same
Accuracy	0.755	0.76	All Features (slightly better)
Model Complexity	Lower (Fewer features, simpler model)	Higher (All features, more complexity)	Selected Features (simpler, faster)
Risk of Overfitting	Lower (less chance of overfitting)	Higher (more features may cause overfitting)	Selected Features (more robust)

Model Deployment

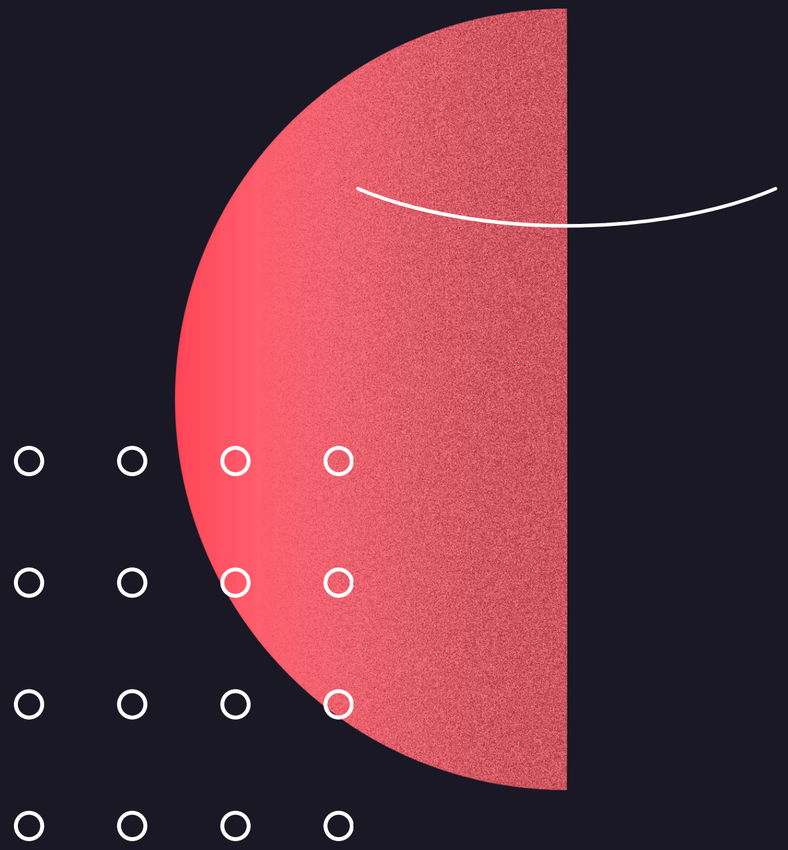




HERE ARE SOME PREDICTIONS

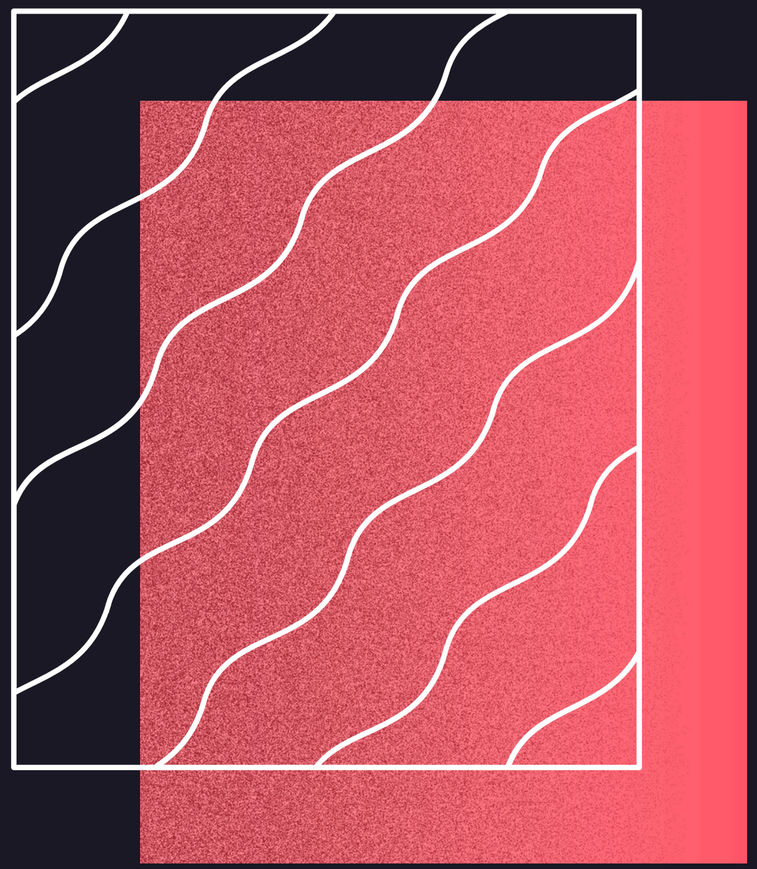
Our model

Conclusions and Limitations



CONCLUSION

- Glucose and BMI are critical predictors of diabetes, as shown in feature importance.
- The adjusted logistic regression model effectively predicts diabetes with a well-balanced precision-recall tradeoff.
- The selected features model is a better choice for real-world deployment due to its comparable or better metrics and simplicity.



LIMITATIONS

- Class Imbalance:
  - Slight imbalance in the dataset could affect predictions for minority class (diabetic).
- Dataset Size:
  - Limited dataset size may affect generalizability to larger populations.
- Threshold Adjustment:
  - Tuning threshold improves recall but may increase false positives, which could be a concern in medical contexts.