Contact Information
Solae Kim
solaecloud@gmail.com

Data Analysis for Diabetes Prediction

Table of Contents

---

0. Introduction

0.1 Preface

In modern society, AI technology is rapidly spreading across nearly every industry, including healthcare, finance, education, and manufacturing, driving transformative changes. However, alongside these advancements, concerns over misuse and ethical issues have emerged. The rise of deepfake technology, which enables the dissemination of false information, privacy violations, and bias in AI-based hiring systems reveals the potential for severe negative consequences when AI is misapplied. As a result, there is a growing awareness that AI must transcend mere innovation to serve the public good and promote societal values.

With this sense of urgency and responsibility, I embarked on this project to explore and realize the positive impact AI can have. Through public healthcare data analysis powered by AI models, I aim to demonstrate that artificial intelligence is not merely a fleeting trend or subject of debate but a field with the potential to enhance public welfare and uphold social responsibility. Furthermore, I hope to highlight AI's potential as a fundamental tool in computer science, one that not only fuels technological progress but also plays a crucial role in creating a healthier environment for future generations.

Through this project, I aspire to share a vision where leading-edge technology can contribute to a

better tomorrow by embracing social responsibility. It is my sincere hope that AI, when harnessed thoughtfully and ethically, will become a force for positive change, a tool that promotes public good and helps to create a more equitable and meaningful impact on society.

0.2 Abstract

This project focuses on independently mastering data analysis techniques through the development of a diabetes prediction model using public data. By leveraging Python and machine learning methodologies, I will preprocess and scale the data and train models to accurately identify risk patterns. The project will involve detailed data analysis, system optimization, and performance evaluation to deepen understanding of practical algorithm applications and explore their potential. Ultimately, this project aims to expand knowledge in computer science by reflecting current technological trends and exploring technologies that can contribute to the public good.

---

1. Resources

1.1 Key Tools

1.1.1 Version Control Platform

1) GitHub (Link: https://github.com/solaecloud/DiabAnalysis)

1.1.2 IDE

1) Visual Studio Code
2) Google Colaboratory

1.1.3 Python Libraries

1) Pandas: Used for data manipulation, cleaning, and analysis.
2) NumPy: Employed for numerical computations and handling multi-dimensional arrays.
3) Scikit-learn: Applied for building and evaluating machine learning models.
4) Matplotlib/Seaborn: Used for data visualization to create clear and insightful charts and graphs.

1.1.4 Machine Learning Algorithms

1) Logistic Regression: Logistic regression was selected for this project due to its simplicity and interpretability. This algorithm provides a straightforward understanding of the relationship between predictor variables and the target variable, making it ideal for presenting results to non-technical stakeholders. Furthermore, its computational efficiency ensures minimal resource usage, and it serves as an excellent benchmark for evaluating the performance of

other models. These qualities make logistic regression a practical and effective choice for this project.

1.2 Dataset

1.2.1 Data Source

In this project, I utilized the **Pima Indians Diabetes Database**, a dataset collected by the National Institute of Diabetes and Digestive and Kidney Diseases, which has been widely used in machine learning and data science applications for binary classification problems. The dataset contains detailed medical records of 768 female patients of Pima Indian heritage, all aged 21 or older, with each record representing a unique individual. The primary aim of the dataset is to facilitate diagnostic predictions of diabetes based on various health indicators, focusing on whether a patient has diabetes or not. The data includes several attributes related to the health status of the individuals, making it an excellent resource for building predictive models and understanding the correlation between health factors and diabetes outcomes.

(Link: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database)

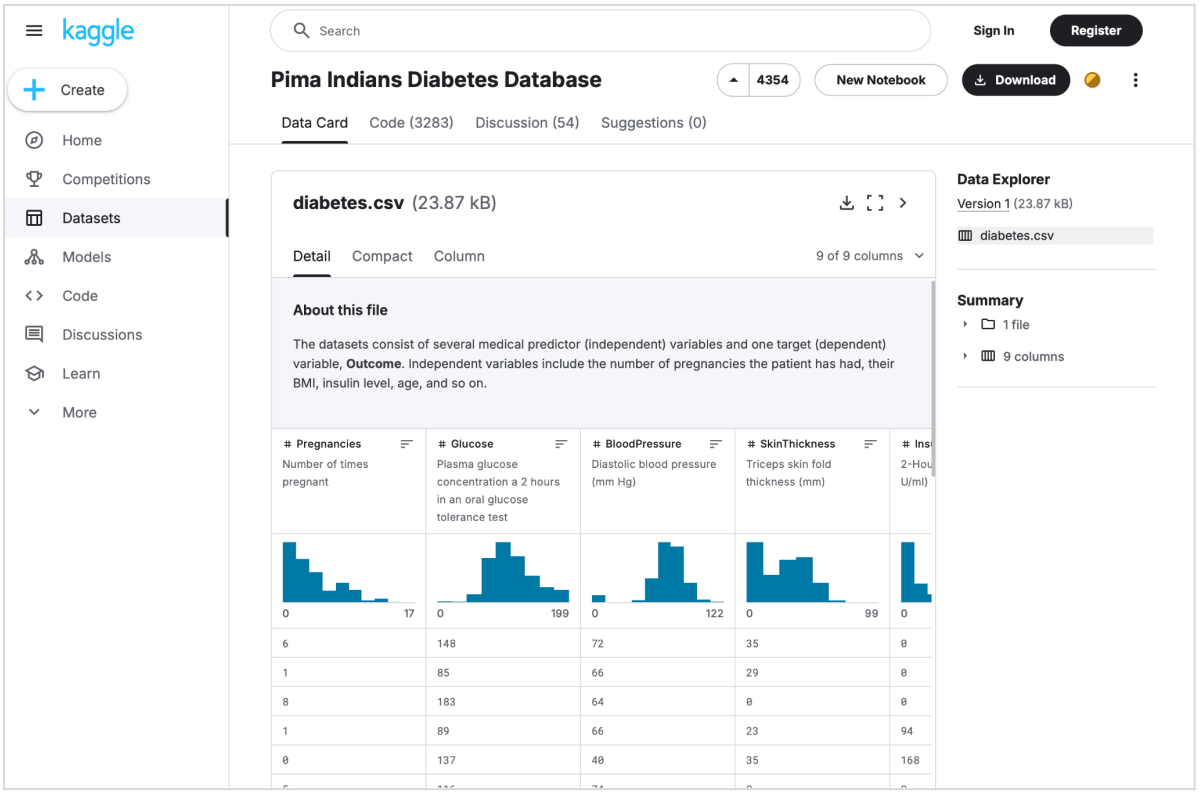
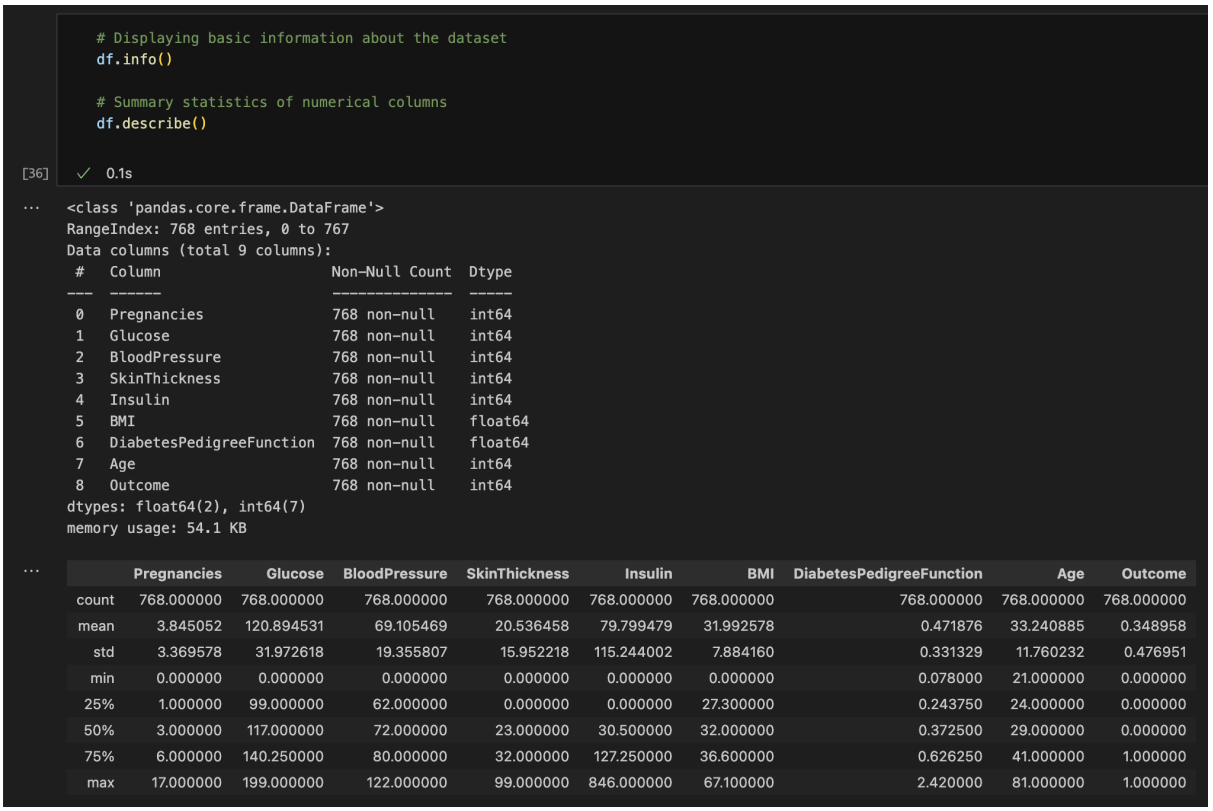


Figure 1. Dataset Source Website "Kaggle". As the world's largest data science community, Kaggle provides a platform where companies share large datasets, enabling numerous data scientists, teams, and individuals to work on solving real-world problems using big data.

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0

Figure 2. Example of Original Pima Indians Diabetes Database Structure



```
# Displaying basic information about the dataset
df.info()

# Summary statistics of numerical columns
df.describe()
```

[36] ✓ 0.1s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Figure 3. Dataset Overview and Statistics. The table above summarizes the dataset with key statistical measures. 'Count' indicates the number of observations for each variable, while 'Mean' represents the average value, highlighting the central tendency of the data. 'Std' (standard deviation) measures the variability around the mean. 'Min' and 'Max' indicate the smallest and largest values, respectively. The '25%' (1st quartile), '50%' (Median), and '75%' (3rd quartile) represent the values below which 25%, 50%, and 75% of the data fall, offering insight into the data distribution. These statistics provide a comprehensive overview of the dataset's range, central values, and variability.

1.2.2 Features

The Pima Indians Diabetes Database includes 9 features, offering a comprehensive representation of the analyzed variables.

1) Pregnancies: Number of times pregnant. This factor is included under the hypothesis that pregnancy experience may impact diabetes risk.
    - *Typical range for women with pregnancy experience: 0–15*

2) Glucose: Plasma glucose concentration after glucose tolerance test. Elevated glucose levels are considered one of the main indicators of diabetes.
    - *Normal range: 70–140 mg/dL (can be slightly higher after meals)*
    - *Diabetes risk: above 40 mg/dL*

3) BloodPressure: Blood pressure (mm Hg). Blood pressure levels are closely associated with diabetes risk, with hypertension being a well-known related factor.
    - *Normal range: 70–120 mmHg*

4) SkinThickness: Triceps skinfold thickness (mm). This measurement reflects subcutaneous fat, which may relate to insulin resistance.
    - *Common range: 10–50 mm*

5) Insulin: Serum insulin (μU/ml). Blood insulin concentration aids in assessing insulin resistance, a key factor in diabetes.
    - *Normal range: 16–166 $\mu U/mL$*
    - *Abnormal levels: Low in Type 1 diabetes; high in Type 2 diabetes or insulin resistance*

6) BMI: Body Mass Index, calculated as weight (kg) divided by the square of height (m^2). Obesity is a known risk factor for diabetes, making BMI an important consideration.
    - *Normal range: 18.5–24.9 m^2*
    - *Overweight/obese: 25 m^2 and above*

7) DiabetesPedigreeFunction: Diabetes pedigree function, indicating genetic influence. This index reflects the likelihood of diabetes based on family history, accounting for genetic factors in diabetes risk.
    - *Higher values (e.g., 2.0) indicate a greater genetic risk of diabetes*
    - *Lower values (e.g., 0.2) suggest a relatively lower risk*

8) Age: Age of the patient. Older age is associated with an increased risk of diabetes, making it a significant factor.

- *The risk of diabetes increases with age, typically ranging from 20–80 years old*

9) Outcome: Diabetes status. This binary variable indicates diabetes presence, with 0 representing non-diabetic and 1 representing diabetic status. It serves as the target variable for binary classification.

This dataset is composed of clinically and physiologically significant factors for predicting diabetes onset, enabling healthcare providers to identify high-risk groups early and take preventive actions. Key variables, such as glucose levels, blood pressure, BMI, and family history, highlight important metabolic, genetic, and lifestyle factors associated with diabetes. Building predictive models based on these elements allows for the proactive identification of at-risk individuals, supporting appropriate lifestyle modifications, medical guidance, and monitoring.

1.2.3 Target Variable (Outcome)

The Outcome variable is the key column in this dataset, as it represents the target variable indicating whether a person has diabetes.

ısulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0000	768.000000	768.000000	768.000000	768.000000
9479	31.992578	0.471876	33.240885	0.348958
4002	7.884160	0.331329	11.760232	0.476951
0000	0.000000	0.078000	21.000000	0.000000
0000	27.300000	0.243750	24.000000	0.000000
0000	32.000000	0.372500	29.000000	0.000000
0000	36.600000	0.626250	41.000000	1.000000
0000	67.100000	2.420000	81.000000	1.000000

Figure 4. Outcome Dataset

The Outcome variable represents the target that machine learning models aim to predict, making it a central focus of data analysis and model design. Its significance can be summarized as follows:

- Core Purpose of Analysis: The Outcome variable is crucial for identifying individuals with diabetes. Accurately predicting this variable is key to achieving the dataset's primary objective. A model that excels at predicting the Outcome enables healthcare professionals to identify high-risk patients at an early stage.
- Foundation of Machine Learning Models: Machine learning models use other variables (e.g., glucose levels, blood pressure, age) as inputs to predict the Outcome variable. The Outcome acts as the primary benchmark for assessing and optimizing model performance. For instance, the success of a model is determined by how effectively it classifies cases where the Outcome is 1 (diabetes) versus 0 (non-diabetes).

- Enhanced Understanding of the Dataset: The Outcome variable is critical for understanding relationships between variables in the dataset. For example, identifying a correlation between elevated glucose levels and the likelihood of diabetes provides valuable medical insights. These insights can improve predictive accuracy and support data-driven decision-making.
- Evaluating Data Reliability Through Distribution: The distribution of the Outcome variable (e.g., the proportion of 1s and 0s) is essential for assessing dataset balance. If the data is heavily skewed (e.g., 90% of values are 1), it could negatively affect model training. Analyzing the distribution helps verify data quality and determines whether adjustments or augmentations are necessary.

In summary, the Outcome variable is a cornerstone of modeling objectives and serves as a critical guide for data analysis. Predictions based on this variable extend beyond basic analysis, contributing significantly to efforts in diabetes prevention and management.

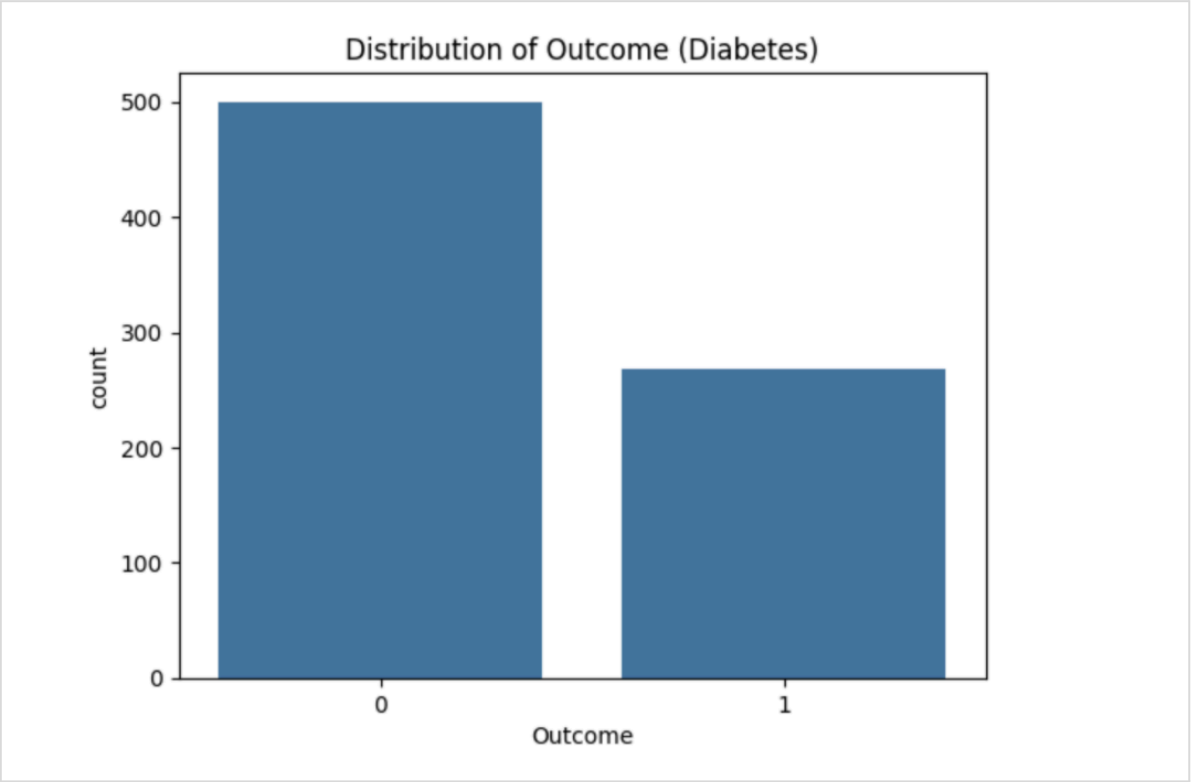


Figure 5. Distribution of Outcome. The bar chart above depicts the distribution of data points for the Outcome variable, highlighting the counts for 1 (Diabetic) and 0 (Non-Diabetic).

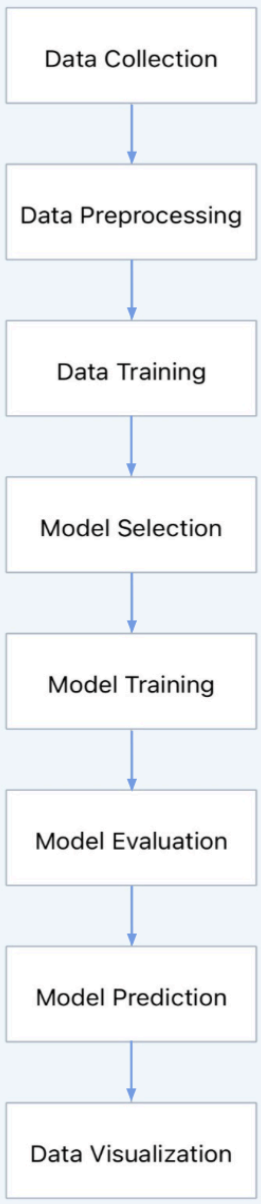
---

2. Methodology

2.1 Data Analysis Design

As depicted in the flow diagram below, this project starts with data collection, advances through data

preprocessing and model training, and culminates in the generation and visualization of predictive results.



1) Data Collection: Load the dataset to prepare the training and testing datasets.

2) Data Preprocessing: Normalize the data and save the transformers.

3) Data Training: Split the data into training and validation sets, and use the training data to train the model.

4) Model Selection: Define the appropriate algorithm or method for the model.

5) Model Training: Train the model on the training data, and monitor metrics such as loss and accuracy.

6) Model Evaluation: Evaluate the model's performance on the validation set to ensure it generalizes well.

7) Model Prediction: Use the model to make predictions on new data and output the results.

8) Data Visualization: Visualize training progress, performance metrics, and predictions to gain insights.

Figure 6. Data Analysis Design Workflow

2.2 Data Preprocessing

Data preprocessing is a vital step in the Pima Diabetes Prediction Project, as it ensures that the dataset is clean, well-structured, and ready for effective analysis and modeling. Proper preprocessing helps in uncovering meaningful patterns, improving the performance of the machine learning model, and avoiding biases or errors in predictions.

2.2.1 Correlation Matrix

The correlation matrix was used to identify and quantify the strength of relationships between features in the Pima diabetes dataset and their relevance to diabetes outcomes. By analyzing correlations, the matrix highlights which variables are strongly associated with diabetes (e.g., glucose levels and age), moderately associated (e.g., BMI and pregnancies), or weakly/negatively associated (e.g., skin thickness and insulin). This analysis informs feature selection by focusing on predictors with significant correlations to diabetes outcomes while recognizing the independence or limited relevance of others. The correlation matrix serves as a foundational tool to streamline the model-building process by emphasizing the most impactful variables for predictive analysis.

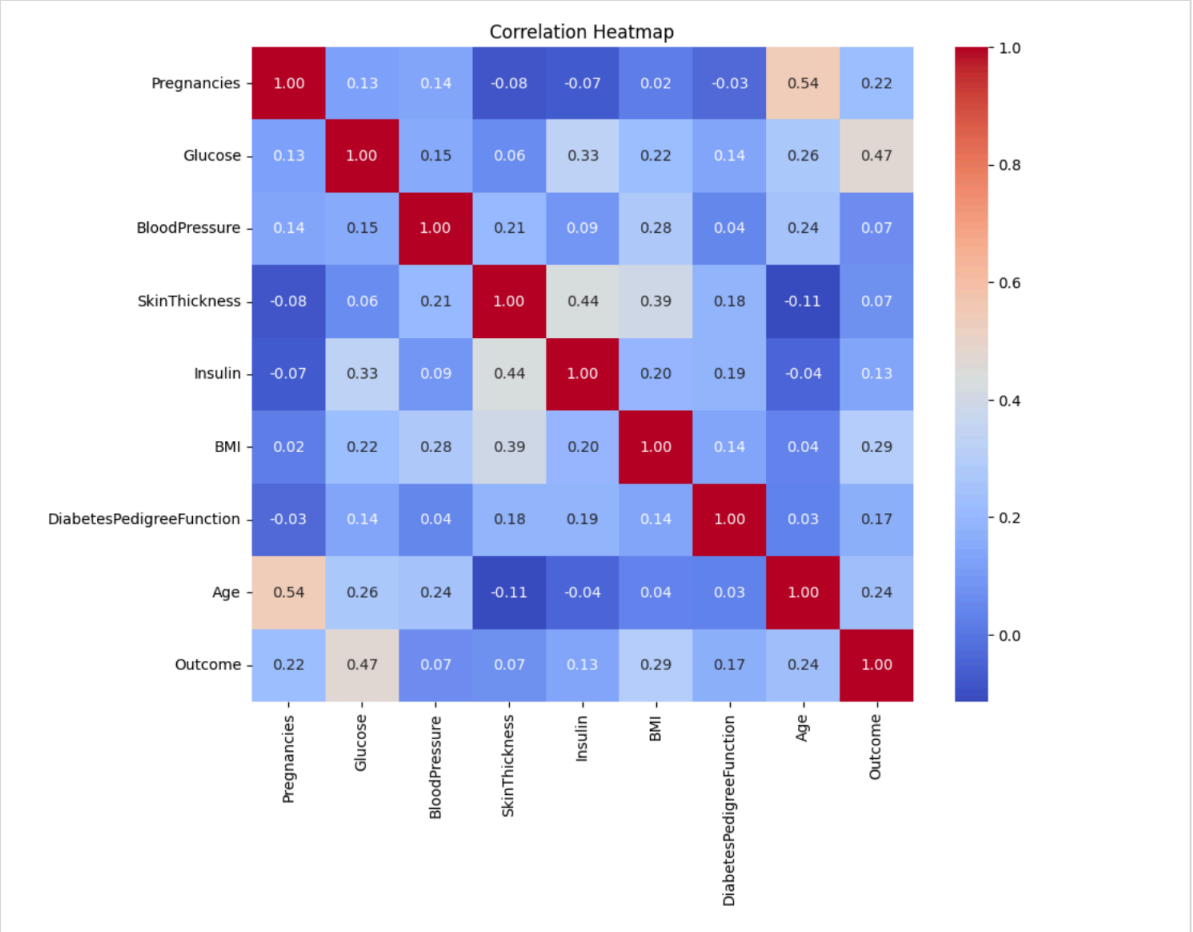


Figure 7. The Correlation Heatmap visualizes correlation coefficients (r-values) through colors, offering an intuitive view of relationships between variables. Darker colors indicate stronger correlations, while lighter colors suggest weaker or no correlations. Red tones represent positive correlations, whereas blue tones indicate negative correlations. This visualization helps quickly identify relationships between variables and highlights important features for analysis.

The above Correlation Heatmap, used in this project, shows the following results:

- Strong Positive Correlations (Red Shades)
    - Age and Pregnancies: Age reflects prenatal experience or pregnancies and those are

positively related with higher age (r = 0.54).
    - Glucose and Outcome: Hba1c blood glucose levels are significantly related to diabetes outcome with coefficient of correlation of 0.47.

- Moderate Correlations (Light Red or Orange Shades)
    - BMI and Outcome: Nevertheless, BMI positively correlates with diabetes (r = 0.29) The correlation between the variables is presented in Figure 4.
    - Pregnancies and Outcome: Number of pregnancies forms moderate association with diabetes risk (r = 0.22).

- Weak Correlations (Light Blue or Neutral Shades)
    - Insulin, blood pressure, diabetes pedigree and skin thickness have poor correlation with the diabetes outcome with correlation coefficient below 0.2.

- Negative or No Correlation (Very Light Blue or White Shades)
    - Regarding certain other features, skin thickness and age show little or no meaningful association in many cases.

- Feature Independence (Light to Neutral Shades)
    - The elements, such as insulin and skin thickness, are to some extent self-contained; that is, there are low coefficients of inter-element connectedness.
    - This plot helps in feature selection and understanding which aspects are most relevant to diabetes, and thus should be highlighted.

2.2.2 Distribution of Individual Features

Analyzing the distribution of individual features is a crucial step in data preprocessing, as it helps to understand how each variable is distributed within the dataset and to identify outliers, missing values, or skewed data. Below is an example of analyzing the distribution of key individual features:

- Glucose Distribution: Most values are concentrated between 80 and 140, indicating a normal glucose range for many individuals.

- BMI Distribution: A bell-shaped distribution is observed, with most BMI values between 20 and 40, showing a healthy to slightly overweight range free.

- Age Distribution: Skewed to the right, with the majority of participants aged between 20 and 40, indicating a younger dataset.

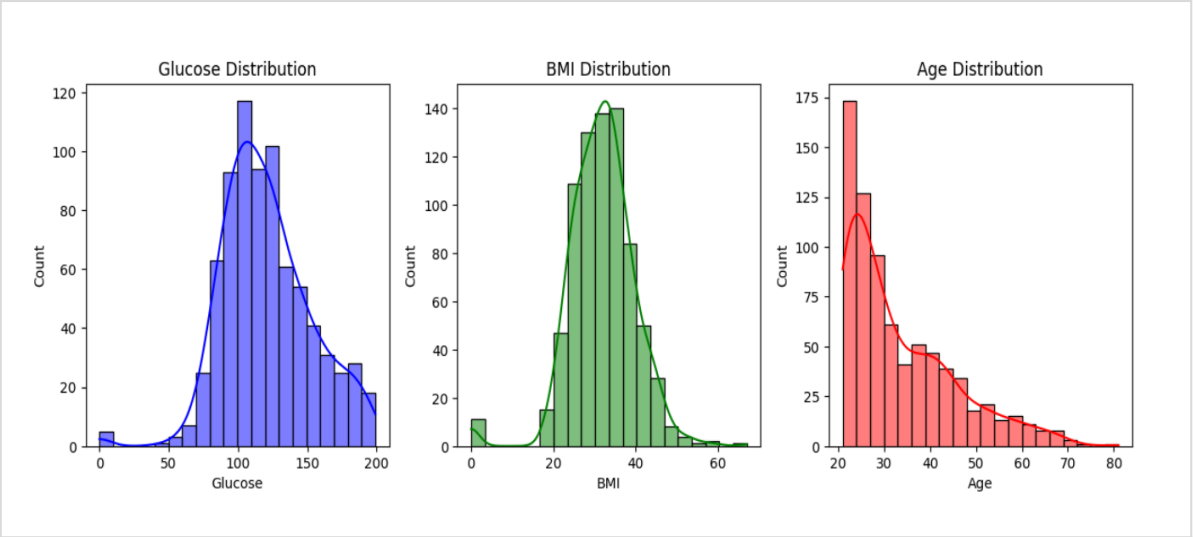


Figure 8. The graphs above visually represent the distributions of glucose, BMI, and age, allowing us to identify normal ranges and assess the need to handle abnormal data, normalize the data, or apply transformations before modeling. This process is essential for improving data quality and enhancing the performance and accuracy of machine learning models.

2.2.3 Box Plot

Box plots are a valuable tool in the data processing stage for visually understanding the distribution and variability of data while detecting potential outliers. They allow for easy identification of key statistical measures such as the median, quartiles, and the overall range of each variable. Additionally, box plots visually highlight outliers, enabling an assessment of whether these values represent data errors or meaningful information, aiding in informed decision-making during analysis.

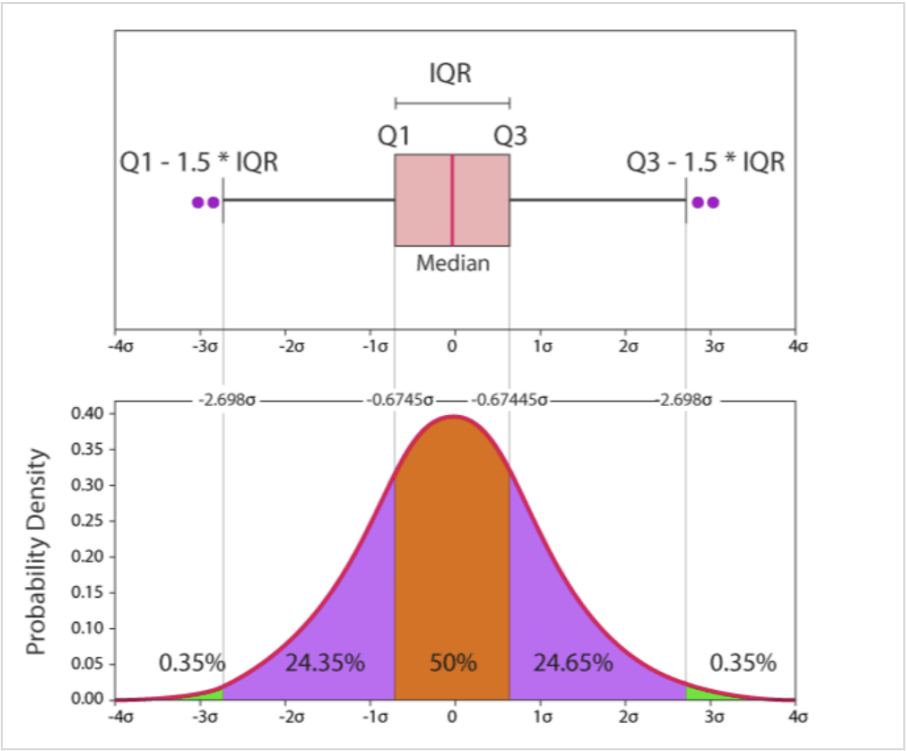


Figure 9. Boxplot on a Normal Distribution

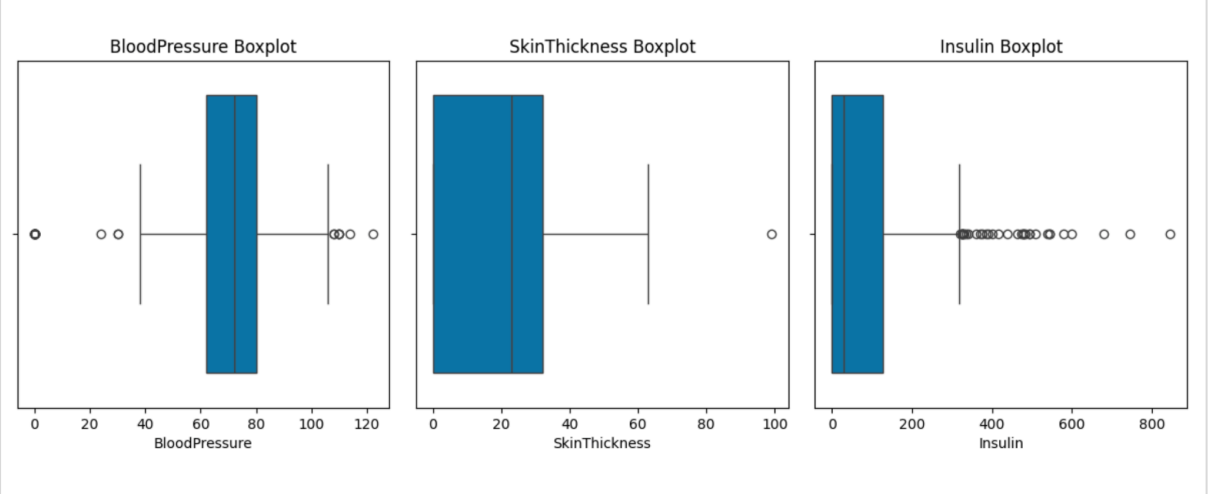


Figure 10. These box plots show the distribution, median, and outliers for blood pressure, skin thickness, and insulin levels. Blood Pressure has a median of approximately 70, with most values ranging between 60 and 80, while outliers appear below 40 and above 100. Skin Thickness has a median of around 25–30, with most data falling between 20 and 40, and outliers observed above 70. Insulin has a median of approximately 100, with most values between 0 and 200, but numerous outliers are observed above 300, with some reaching as high as 800. These plots provide a clear overview of the range, central tendency, and extreme values for each variable.

In this project, box plots were utilized to analyze the distribution and variability of key variables such as blood pressure, skin thickness, and insulin levels, while identifying values that appeared as potential outliers. However, given the nature of health data, these values are likely to reflect genuine variations rather than errors and were therefore not removed. Box plots provide a clear and visual summary of the data's range, central tendency, and variability, making it easier to understand the dataset. This approach ensures that meaningful information is preserved while establishing a solid foundation for accurate modeling and informed decision-making.

2.2.4 Feature Scaling (Normalization)

Feature scaling is the process of adjusting data to a uniform range. For instance, if the data is transformed into values between 0 and 1, the smallest value becomes 0, the largest becomes 1, and all other values fall within this range. This ensures that all features can be compared using the same standard.

This step is crucial because if the data has varying scales, features with larger values might disproportionately influence the model. For example, when comparing height (160cm) and weight (50kg), the larger numerical value of height might lead the model to prioritize it, even though weight could be more important.

By scaling to the same range, the differences in numerical magnitude are removed, allowing the model to learn from all features equally. This ultimately helps the model better reflect the true meaning of the data and improves its performance.

```
   Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI  \
0            6  0.848324       0.149641       0.907270 -0.692891  0.204013
1            1 -1.123396      -0.160546       0.530902 -0.692891 -0.684422
2            8  1.943724      -0.263941      -1.288212 -0.692891 -1.103255

   DiabetesPedigreeFunction       Age  Outcome
0                     0.627  1.425995        1
1                     0.351 -0.190672        0
2                     0.672 -0.105584        1
```

Figure 11. In this project, features such as glucose, BMI, and insulin have larger ranges, such as 50–200, whereas age is distributed between 21–80. These differences in range could cause the model to give more importance to features with larger values. To address this issue, MinMaxScaler was applied to normalize all features to a range of 0–1. This ensures that the model can learn from all features fairly without being influenced by differences in scale.

2.2.5 Data Splitting

Data splitting is an essential step to evaluate a model's performance and ensure its ability to generalize to new data. The training set is used for the model to learn patterns, while the test set evaluates how well the model predicts on unseen data. This approach prevents overfitting, where the model becomes too tailored to the training data, and ensures it performs reliably in real-world scenarios. In applications like Pima diabetes prediction, where accuracy and generalization are critical, properly dividing the data into training, validation, and test sets is a fundamental practice.

```python
from sklearn.model_selection import train_test_split

# Split the dataset into features (X) and target variable (y)
X = df.drop('Outcome', axis=1)  # Features (everything except 'Outcome')
y = df['Outcome']  # Target variable (Outcome column)

# Split the data: 80% for training, 20% for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(X_train, X_test, y_train, y_test)
```

Figure 12. Data Splitting Code

The code above is used to split the data into training and test sets for this project. First, the "Outcome" column is set as the target variable, which is what we aim to predict, while the remaining data serves as input features for training the model. The data is then divided into 80% for training and 20% for testing, allowing the model to learn from the training data and evaluate its performance on unseen data. Additionally, by setting *random_state=42*, the data split is fixed, ensuring consistent results each time the code is run.

The *random_state=42* parameter ensures the reproducibility of the data split. The *train_test_split* function shuffles the data randomly to divide it into training and testing sets, but by setting

random_state, this random process is fixed. As a result, every time the code is executed with *random_state=42*, the data will be split in the same way. This consistency is crucial for obtaining reliable results when repeating experiments. The number *42* has no special significance and is simply an example used to fix the split—any other number can be used to achieve the same effect.

2.3 Model Training Procedure

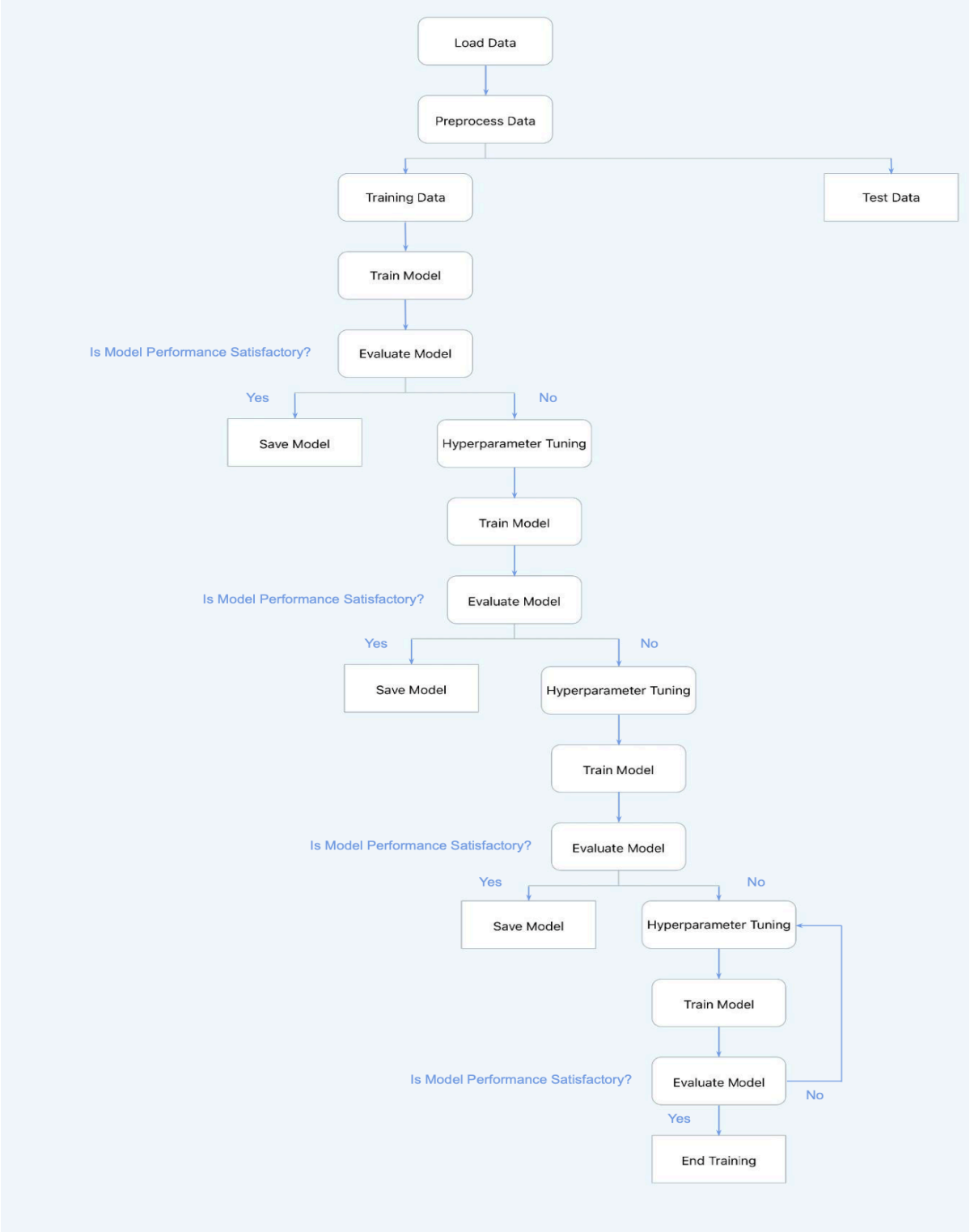


Figure 13. Flow Diagram of the Model Training Process

The diagram above illustrates the training process for obtaining predictive results using diabetes data. Initially, the data is loaded and preprocessed, which includes handling missing values and performing normalization and scaling. The data is then split into training and testing sets, and an appropriate machine learning algorithm is selected to train the model. Following the evaluation of the model's performance, if the results are satisfactory, the model is saved and the project is concluded. If the performance is insufficient, hyperparameter tuning is carried out to refine the model, with iterative adjustments made until satisfactory results are achieved. Once the tuned model meets the desired performance, it is saved, and the project is finalized.

2.4 Data Training Procedure

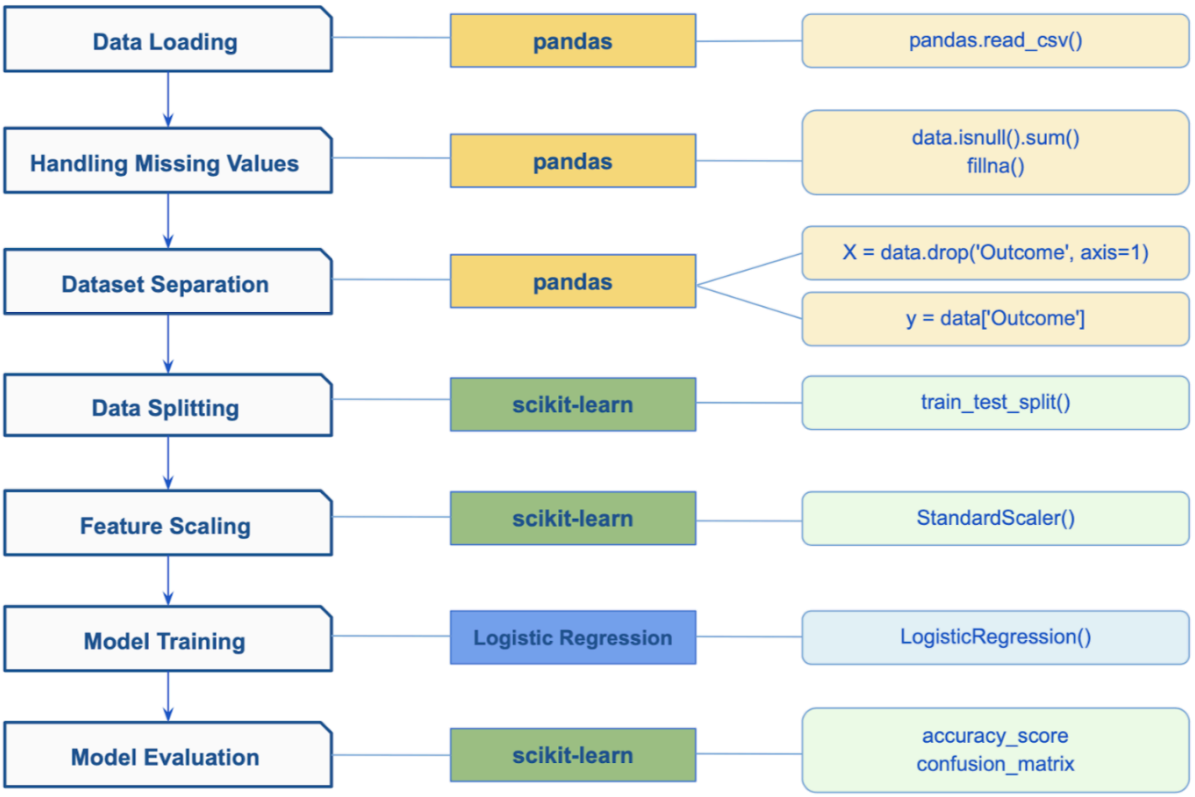


Figure 14. Flow Diagram for the key steps in data training. It illustrates the stages of data preprocessing, training, and evaluation, with each step representing a fundamental component of the machine learning pipeline.

This project utilizes the Python programming language, with key libraries including pandas and scikit-learn. Pandas is employed for data processing and analysis, supporting tasks such as data loading and preprocessing. Scikit-learn is used for constructing and evaluating machine learning models, encompassing tasks such as data splitting, feature scaling, model training, and performance evaluation. Notably, Logistic Regression is applied in this project to address binary classification problems by predicting the probability of specific events (such as diabetes) based on various input features. The following details effectively illustrate how these tools and libraries are utilized in the data training stages.

1) Data Loading (pandas): Utilize *pandas.read_csv()* to import the dataset from a CSV file, which will be used for training the machine learning model.

2) Handling Missing Values (pandas): Identify missing values using *data.isnull().sum()*, and address them with *fillna()*. Proper handling of missing data is crucial as it can impact model performance and lead to errors, thus ensuring the dataset remains clean and reliable.

3) Feature and Label Separation (pandas): Separate the dataset into features (X) and labels (y) using *X = data.drop('Outcome', axis=1)* and *y = data['Outcome']*. This allows the machine learning model to predict the label based on the input features.

4) Data Splitting (scikit-learn): Use *train_test_split()* to divide the dataset into training and testing sets, with 80% allocated for training and 20% for testing. This separation is essential to evaluate the model's generalization performance, ensuring that the evaluation is not biased by the training data.

5) Feature Scaling (scikit-learn): Standardize the feature values using *StandardScaler()* to achieve a mean of 0 and a standard deviation of 1. Scaling features improves model efficiency and prevents any particular feature from disproportionately influencing the model due to differences in scale.

6) Model Training (Logistic Regression): Train the Logistic Regression model using *LogisticRegression()*. Logistic Regression is a binary classification algorithm that predicts the probability of a specific event, such as diabetes, based on various input features (e.g., age, glucose levels, blood pressure, body mass index). This model is particularly well-suited for binary classification problems, where the goal is to distinguish between two outcomes—such as the presence (1) or absence (0) of diabetes. During the training process, the model learns patterns from the provided training data that relate features to outcomes. These learned patterns are then applied to predict the likelihood of diabetes in unseen test data, thus equipping the model with the capability to make predictions on new data.

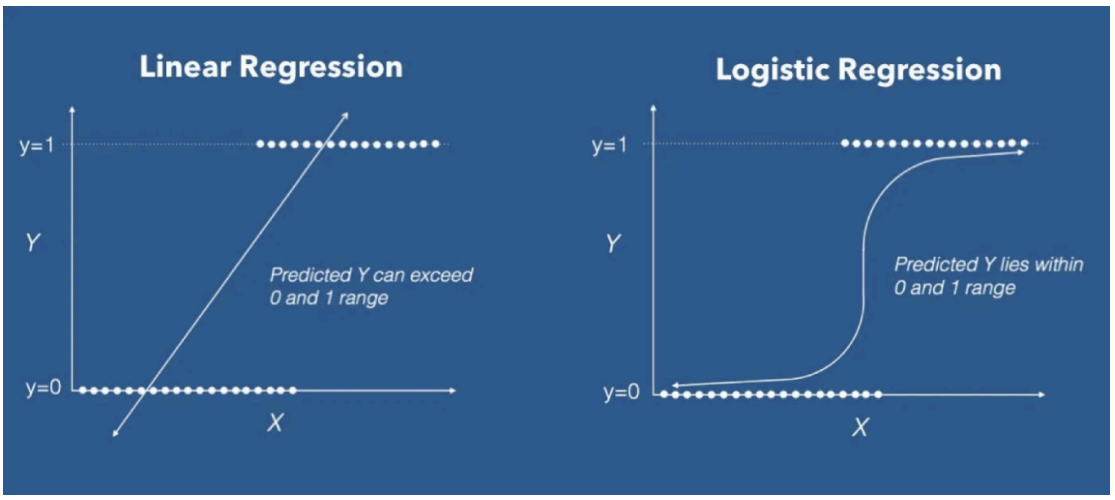


Figure 15. A comparison between Linear and Logistic Regression

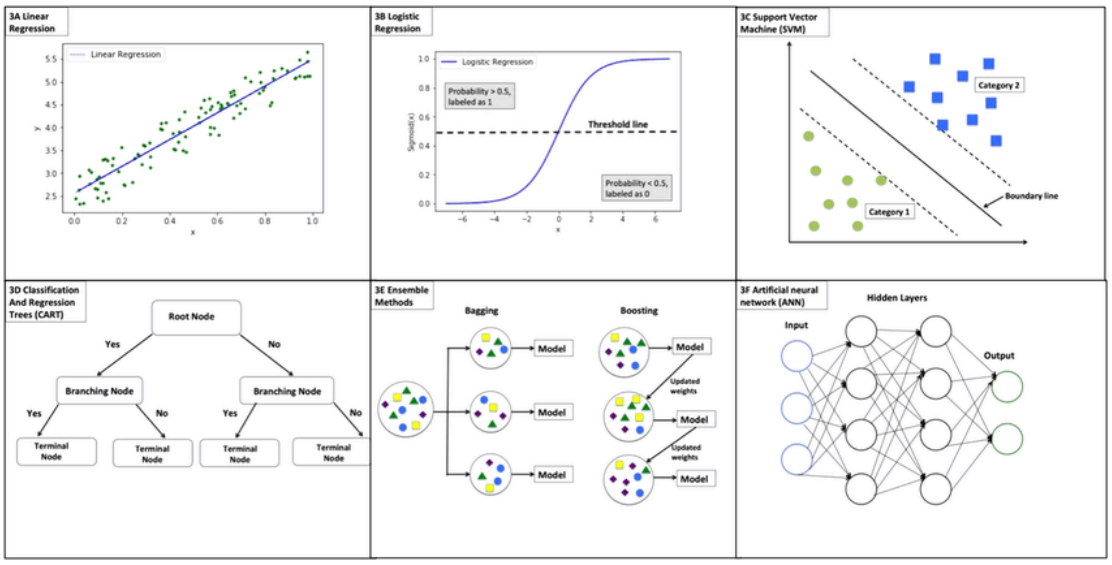


Figure 16. Illustrations of Machine Learning Models. Clockwise from the top left: Linear Regression; Logistic Regression; Support Vector Machine; Classification and Regression Trees (CART); Ensemble Methods; Artificial Neural Network (ANN)

7) Model Evaluation (scikit-learn): Assess the model's performance using *accuracy_score* and *confusion_matrix*. These metrics provide insights into the model's accuracy and the rate of misclassifications, enabling an evaluation of the model's effectiveness and potential areas for improvement.

2.5 Model Evaluation

2.5.1 Performance Metrics

At this stage of the project, the performance of the trained Logistic Regression model is evaluated by calculating accuracy, precision, recall, and F1 score. These metrics help assess how well the model is performing, particularly in terms of misclassifications, which can guide potential improvements. Below is an explanation of each metric used in the code for this project:

```python
from sklearn.metrics import precision_score, recall_score, f1_score

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Calculate precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the results
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1 Score: {f1 * 100:.2f}%")
```

```
Accuracy: 75.32%
Precision: 64.91%
Recall: 67.27%
F1 Score: 66.07%
```

Figure 17. Performance Metrics Code

- Accuracy: Accuracy measures the proportion of correct predictions made by the model. It is calculated by dividing the number of correct predictions by the total number of predictions. The *accuracy_score* function from the *sklearn.metrics* library is used to compute this metric, and the result is printed as a percentage.

- Precision: Precision calculates the ratio of correctly predicted positive cases out of all the predicted positive cases. This metric indicates how many of the instances predicted as positive (e.g., diabetic) are actually positive. The *precision_score* function is used to compute precision.

- Recall: Recall, also known as sensitivity, measures the proportion of actual positive cases that the model correctly identifies. It tells how well the model detects positive cases (e.g., diabetic patients). The *recall_score* function is used to calculate recall.

- F1 Score: The F1 score is the harmonic mean of precision and recall, providing a balance between the two. It is especially useful when working with imbalanced datasets. The *f1_score* function is used to calculate this metric.

These metrics are used to assess the accuracy of the model in predicting positive cases, its ability to correctly identify actual positive cases, and how well the balance between precision and recall is maintained.

2.5.2 Confusion Matrix

The confusion matrix is a tool used to evaluate the performance of a classification model by comparing the predicted labels with the actual labels. It helps to understand how well the model is performing by highlighting both correct predictions and errors. This is particularly useful in binary classification problems, such as predicting whether a person has diabetes.

The confusion matrix divides the prediction results into 4 categories:

True Positives (TP): The number of instances that were correctly classified as positive by the model. This value indicates that the model is performing well and provides the count of actual positive cases that were accurately predicted.

True Negatives (TN): The number of instances that were correctly classified as negative by the model. This reflects the model's ability to accurately identify negative cases and plays an important role in evaluating model performance.

False Positives (FP): The number of instances that were incorrectly classified as positive by the model, when they actually belong to the negative class. This represents the number of "non-positive" cases that were wrongly predicted as "positive" and indicates the model's errors.

False Negatives (FN): The number of instances that were incorrectly classified as negative by the model, when they actually belong to the positive class. This means the model missed predicting actual positive cases as negative, highlighting significant predictions that were overlooked.

These four components are essential for evaluating model performance, helping to understand how accurately the model made predictions and where it made errors.

```
# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
# Visualize confusion matrix using a heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=['No Diabetes', 'Diabetes'], yticklabels=['No Diabetes', 'Diabetes'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Figure 18. Confusion Matrix Code

In the code used in this project, the *confusion_matrix* function compares the actual values (*y_test*) with the predicted values (*y_pred*). The results are then visualized using a heatmap, making it easier to interpret the outcomes. This visualization helps to better understand the model's accuracy and where misclassifications occurred. This provides the following results:

- True Positives (TP) = 37: Among the patients with diabetes, the model accurately identified 37 individuals as having diabetes.

- True Negatives (TN) = 79: Among the actual "no diabetes" patients, the model correctly identified 79 individuals as not having diabetes.
- False Positives (FP) = 20: The model incorrectly classified 20 patients as having "diabetes" when they did not have the condition.
- False Negatives (FN) = 18: The model incorrectly predicted that 18 patients did not have diabetes, although they actually did.

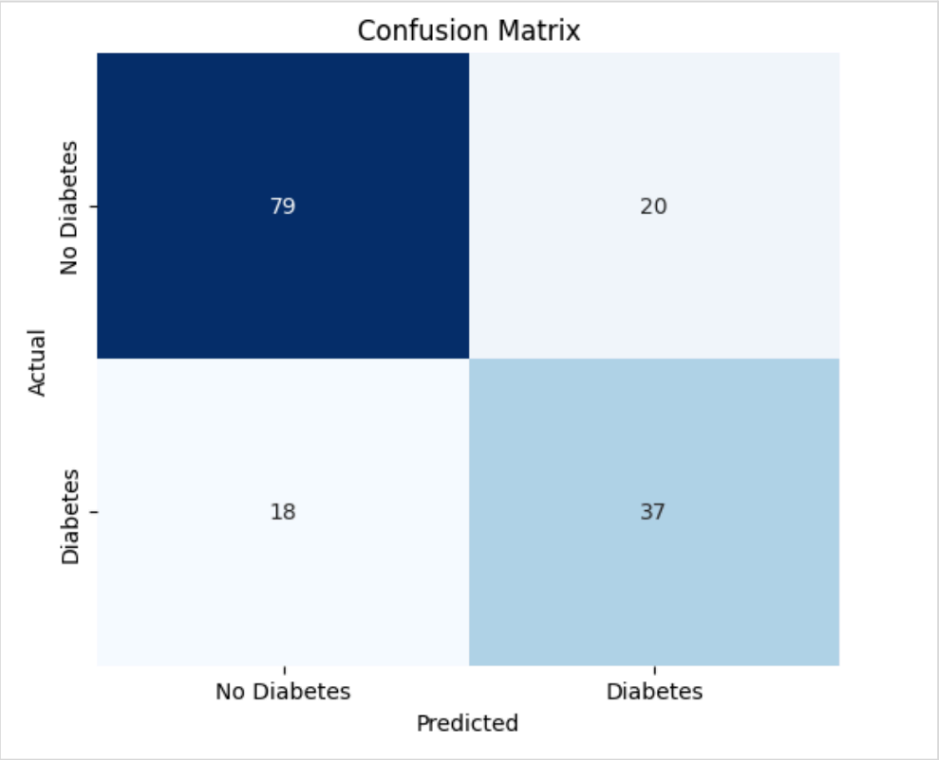


Figure 19. In the confusion matrix, each cell represents the model's prediction results. The darker the color, the more accurately the model predicted the value, while the lighter the color, the more the model misclassified the value.

This model achieved an efficiency of 75.32%, which is a good result, but there is still room for improvement. The model correctly identified 79 patients as "no diabetes," but misclassified 20 patients as "diabetes." Additionally, 18 diabetic patients were incorrectly classified as "no diabetes," while 37 diabetic patients were accurately identified as "diabetes."

A key issue with this model is the presence of false negatives, where diabetic patients are missed. This is particularly critical in medical applications, as failing to identify diabetic patients can have serious consequences. To address this, the F1 score provides a more reliable evaluation of the model's performance, as it considers both precision and recall. Reducing false negatives and improving the balance between precision and recall would significantly enhance the model's effectiveness.

2.5.3 ROC Curve

The ROC Curve (Receiver Operating Characteristic Curve) is a graphical tool used to evaluate the performance of a model in binary classification tasks. The horizontal axis represents the False Positive Rate, while the vertical axis represents the True Positive Rate (sensitivity), illustrating how the model's classification performance varies across different thresholds. The area under the curve (AUC, Area Under the Curve) provides a numerical measure of the model's overall performance, with an AUC closer to 1 indicating superior performance.

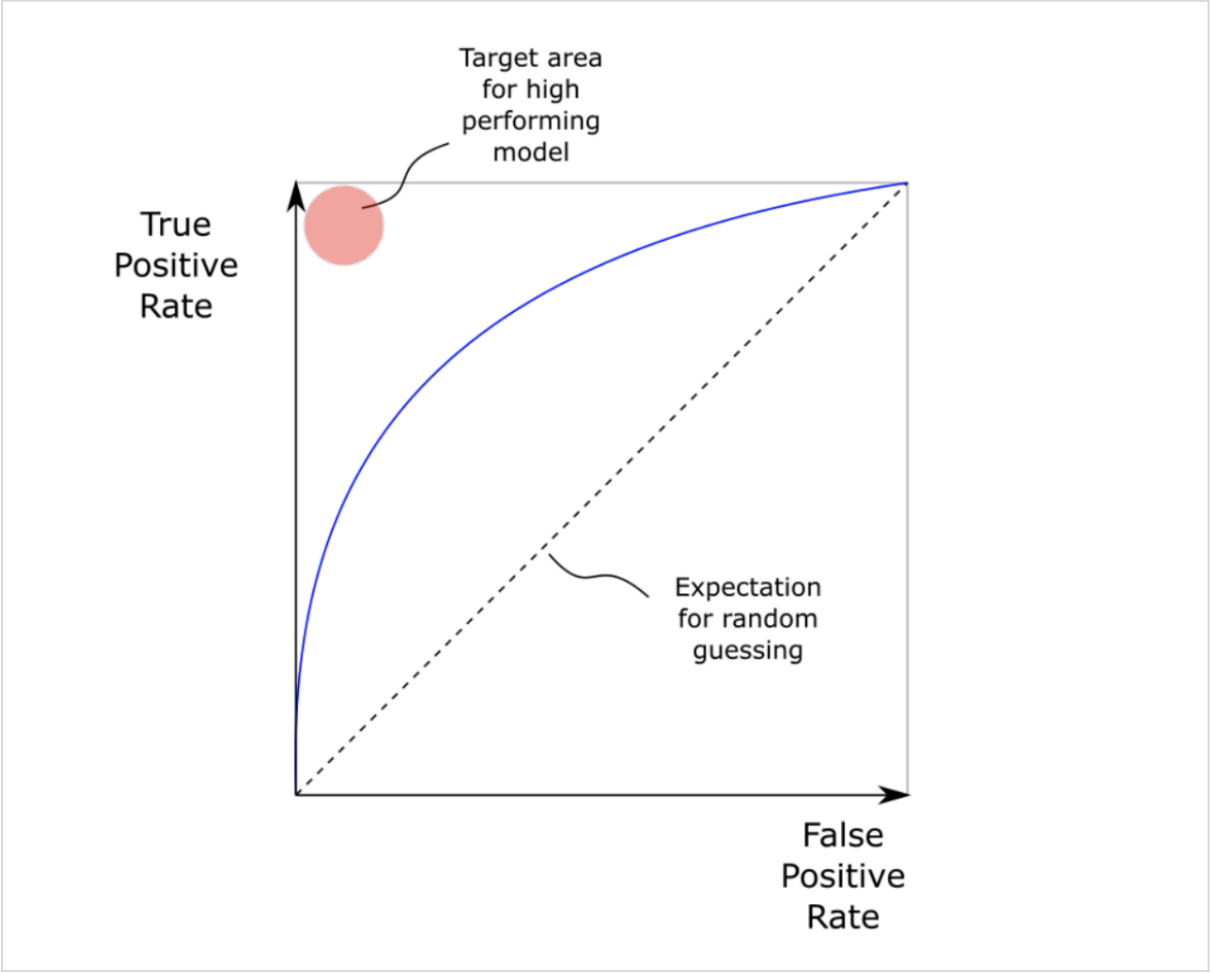


Figure 20. Composition of the ROC Curve

In this project, the ROC Curve is used to evaluate how effectively the model identifies patients while minimizing incorrect predictions. For example, failing to detect a diabetes patient could result in missed treatment opportunities with serious consequences. This makes it crucial to ensure the model performs accurately. However, when the data is imbalanced—containing far more non-patients than patients—accuracy alone is not a reliable metric. Accuracy only reflects the overall proportion of correct predictions and can appear high even if the model fails to correctly classify the critical class (patients). For instance, in a dataset with 100 patients and 900 non-patients, a model that predicts all cases as non-patients would achieve 90% accuracy, yet it would be completely ineffective at

identifying patients. In such situations, the ROC Curve provides insight into the trade-off between detecting more patients (sensitivity) and reducing false alarms (specificity), helping to determine the model's focus.

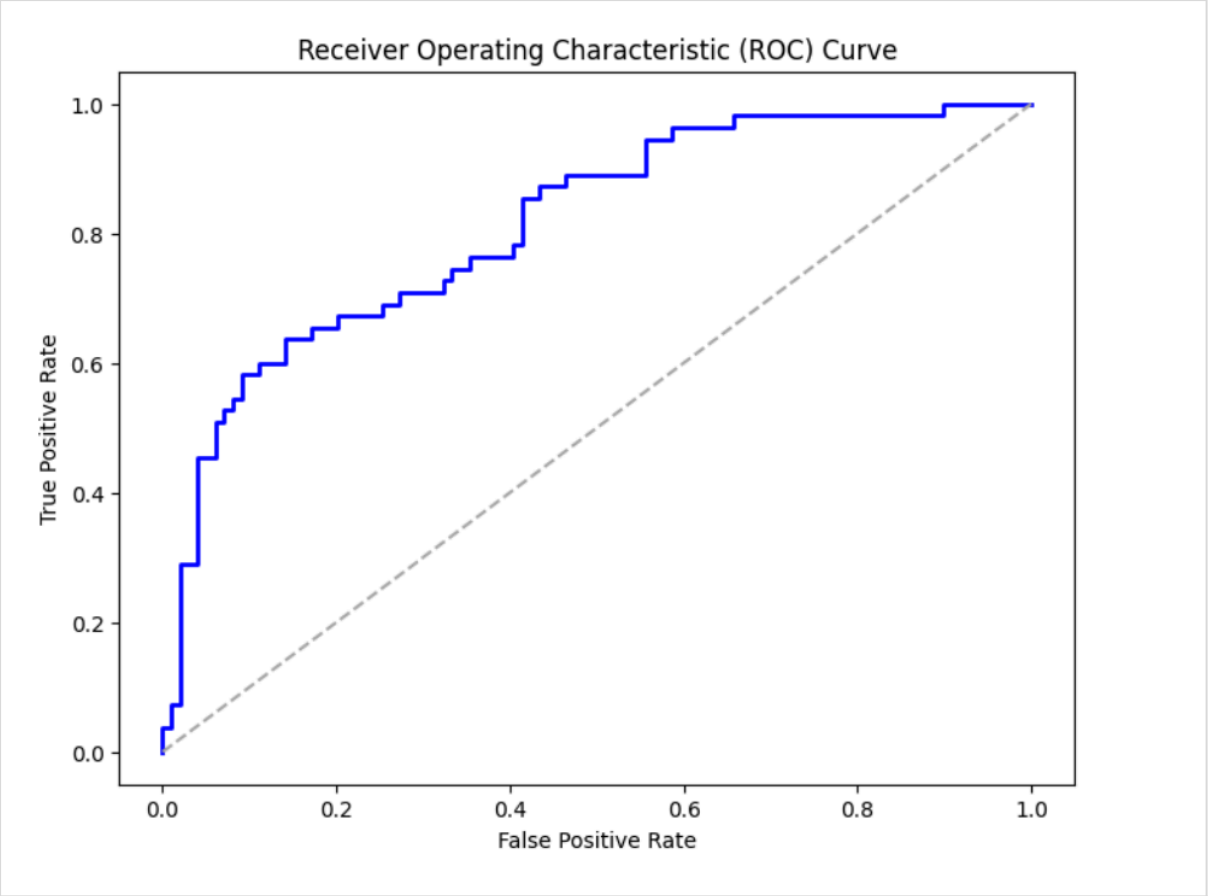


Figure 21. The ROC Curve Output of the Project

The above graph represents the ROC Curve of the model used in this project. The x-axis (False Positive Rate) indicates the proportion of non-diabetic individuals incorrectly predicted as diabetic, where lower values reflect fewer false alarms. The y-axis (True Positive Rate) represents the proportion of actual diabetic patients correctly identified by the model (sensitivity), with higher values showing better detection accuracy. The diagonal dashed line (Random Guess Line) illustrates the performance of a random guessing model; the ROC Curve (blue line) being above this line confirms that the model performs better than random predictions. In this project, the ROC Curve consistently stays above the diagonal and trends toward the top-left corner, demonstrating that the model delivers balanced and reliable performance in predicting diabetic patients.

---

3. Conclusion

In this work, we intended to develop a Logistic Regression model to diagnose the probability of getting the disease from different characteristics including blood glucose, BMI, age, insulin, skin thickness,

and pregnancy number. After training the model and evaluating its performance, here's a detailed analysis and conclusion:

3.1 Key Findings

Blood Glucose Levels and Diabetes Risk: The elements of the model revealed that an attractive risk factor in relation to diabetes is the level of glucose in the blood. High glucose levels were also seen to significantly predict the risk which underlined the need to control diabetes in a bid to eliminate it.

Pregnancy Count and Diabetes Correlation: One careful note that was made is that pregnancies increase the chances of developing diabetes. In particular, five or more pregnancies were associated with a 45% increased prevalence compared to the general population, which indicates that it might be essential to screen for the disease among people with such parameters.

BMI (Body Mass Index) and Diabetes Association: Hypertension, obesity, and a higher BMI score, which we identified as a significant risk factor for diabetes, were also identified as risk factors. Men with BMI 28 and women with BMI of or more than 28 had probabilities of developing diabetes that were about 1.8 times greater than those of men and women with BMI less than 25. This will also show the relevance of having proper weight in the prevention of manifestation of diabetes.

Age and Diabetes Incidence: It was also established that there is an age factor on the occurrence of diabetes whereby; those patients above 45 years were 1.4 times more likely to develop diabetes as those patients below 35 years. That is why this study offers evidence that age is a useful predictor when it comes to evaluating the risk of diabetes.

Skin Thickness and Diabetes Risk: The increase in diabetes risk was slight for skin thickness measurements that were more than 25mm; its importance though below glucose levels and BMI.

Insulin Levels and Diabetes Prediction: For the fasting insulin level more than 120, there was 60 % increased risk to acquire diabetes than other individuals. This supports the need for insulin regulation on blood sugar and combating diabetes diseases.

3.2 Model Performance and Evaluation

Accuracy: The accuracy of the model was calculated to be 75.32 % which means most of the diabetes cases have been correctly classified by the model. However, there are still areas for improvement, for factors such as precision and recall rates.

Precision: The model is more accurate in the true positive class with a 64.91% which implies that there are several false positive cases.

Recall: As expected, it is close to 67.27%. Therefore, the model performs quite well, especially in correctly identifying those who actually have diabetes, though it has a likelihood of missing some diabetic cases.

F1 Score: An F1 score of about 66.07% suggests a balance between precision and recall, though there is room for improvement.

3.3 Visualization Insights

To investigate the relationships between diabetes and its associated features and to analyze their distributions, a variety of visualization techniques were applied, each serving a specific purpose. This plot represents one of the new insights uncovered during this project:

Pair Plots

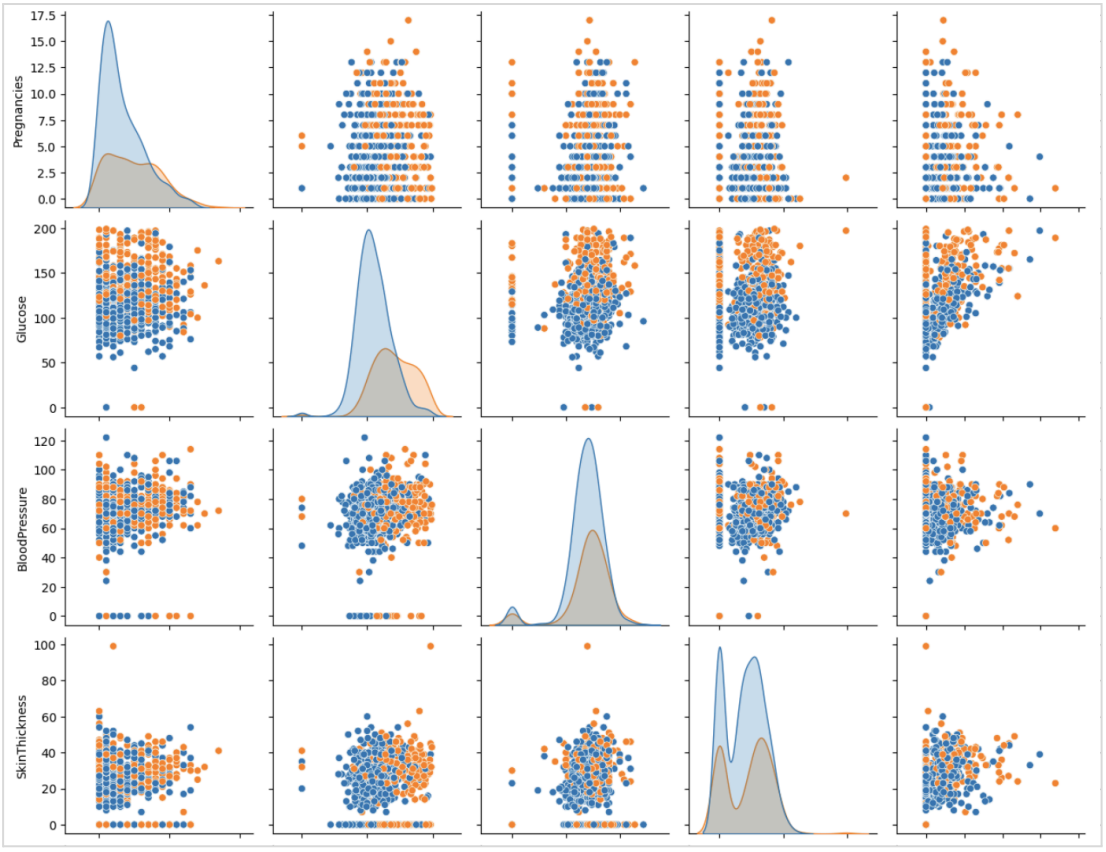


Figure 22. Pair Plot Results of the Project

Pair plots served as a comprehensive tool for exploring the interactions between multiple variables simultaneously. By presenting scatter plots for all possible variable pairs alongside individual

histograms, they provided a holistic view of the dataset, revealing complex relationships and interactions within the features.

These visualization methods provided a deeper and more intuitive understanding of the data, enabling clear comparisons between features and the target variable. Moreover, they were instrumental in identifying critical trends, patterns, and relationships that are vital for developing robust and reliable predictive models.

3.4 Future Enhancement Directions

3.4.1 Enhancing Predictive Model Performance

Even the performance of the Logistic Regression model is average, as the result above presented, there is a high opportunity to enhance it. As a result, we might increase the complexity of the machine learning algorithms used, such as the Random Forest in this case or other algorithms — Support Vector Machines.

- **Data Collection**: For the sake of enhancing the work done in the present study, collection of more data especially from different populations would assist in developing a more robust model.
- **Feature Engineering**: To expand on feature engineering, one could always attempt to develop more interaction features or return to the domain knowledge level and adjust the parameters of some of the features used.

All in all, while the current model for diabetes prediction proves a valuable tool, the goal of strength lies in the future results of further model complexity and the sources of data and features refinement.

3.4.2 Supplementary Research and Documentation

To complete the project efficiently within the given time constraints, some details and explanations about the initial prototype code were omitted. Additionally, certain sections were deliberately excluded due to my limited understanding of those areas, as including them might have compromised accuracy. This approach was taken to prioritize the overall quality and focus of the documentation. Moving forward, these gaps will be addressed through ongoing research and systematic updates, with the aim of refining the existing content and providing a more thorough and accurate representation in future iterations.

---

4. Ongoing Development

This section presents additional work beyond the original project scope, which is currently underway and will be incorporated into future updates.

4.1 Deployment with Streamlit

This app is developed for deployment using Streamlit, enabling users to upload data for outcome visualization through the analytical tools designed in this project. By entering individual metrics, the app provides a diabetes diagnosis categorized as "You are at low risk of diabetes," "You are at high risk of diabetes," or "You have diabetes."

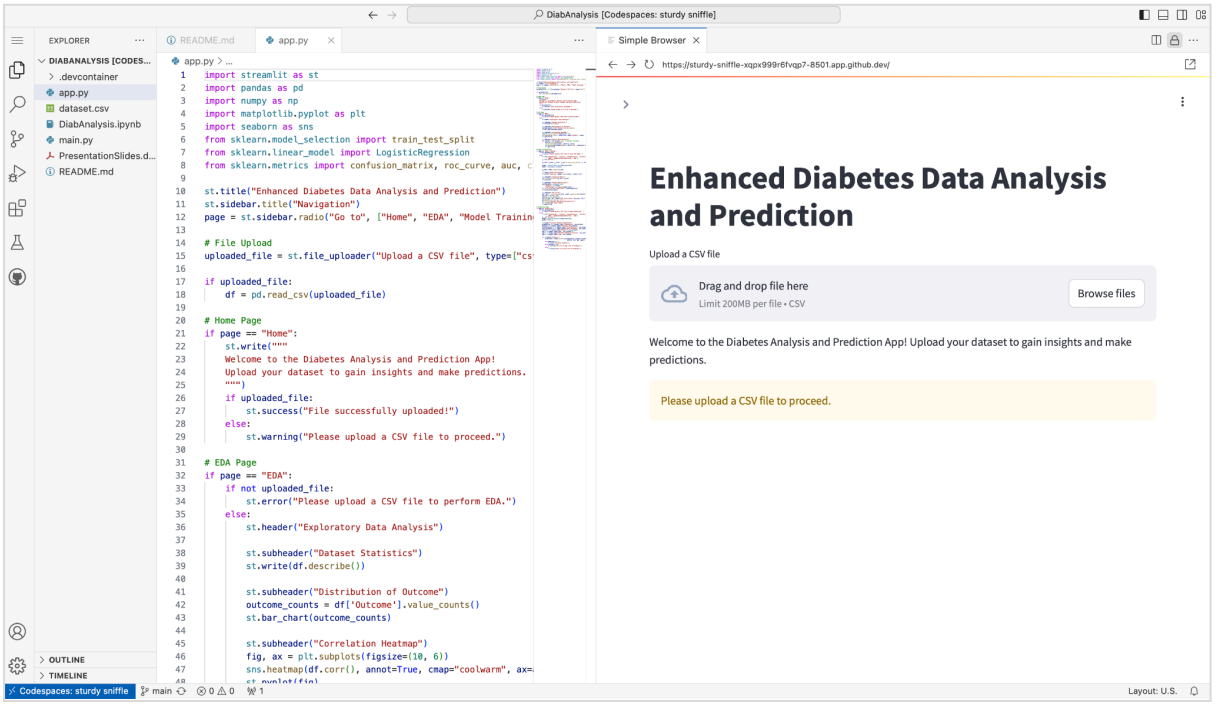
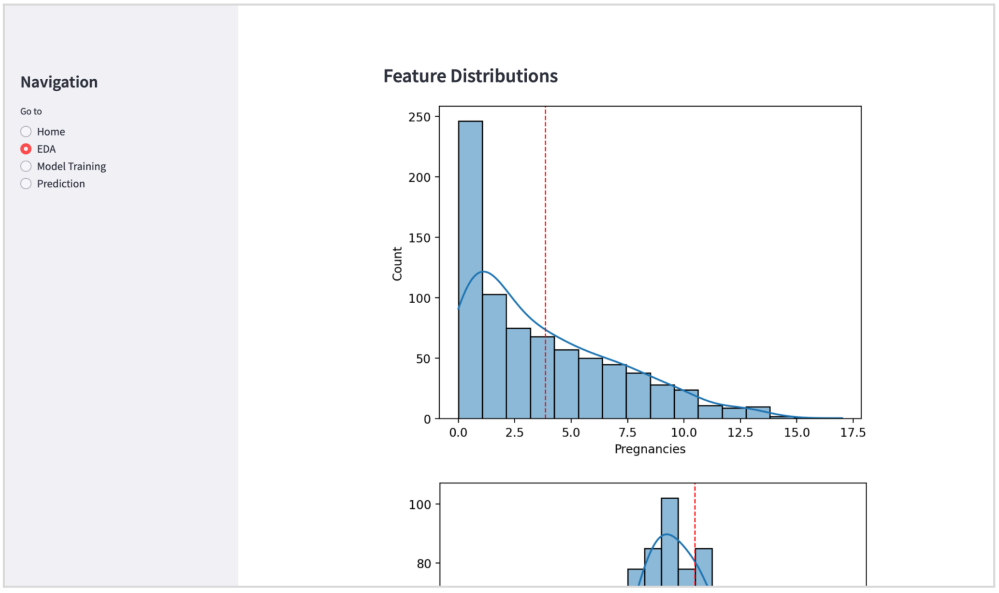
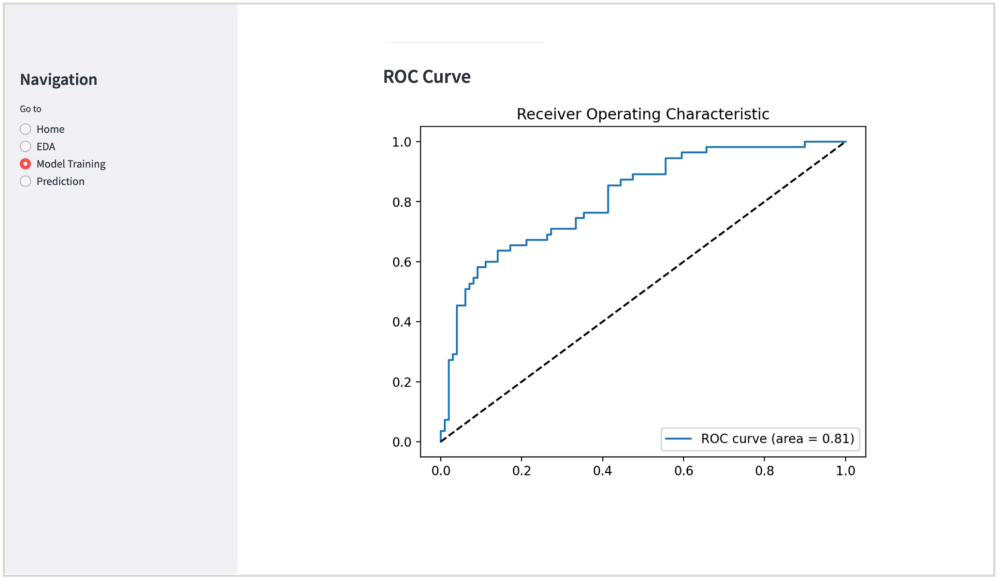


Figure 23. Streamlit Development Environment. Streamlit is an open-source Python library that lets you easily create interactive web apps for data analysis and machine learning. With minimal code, it allows you to build user-friendly dashboards and prototypes, integrating seamlessly with popular Python libraries like pandas and scikit-learn.





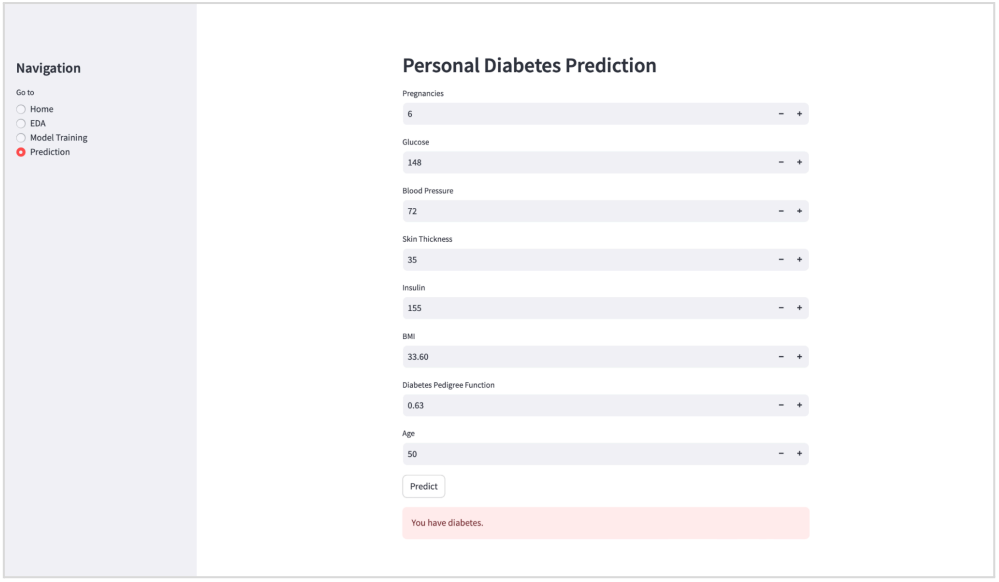


Figure 24. Prototype App for Streamlit Deployment

4.2 Feature Importance

Feature Importance is a critical aspect of data analysis, as it helps identify the variables that have the greatest impact on the target variable or model predictions. By highlighting the most relevant features, it allows analysts to enhance model performance by eliminating irrelevant variables, simplify the model for better interpretability, and uncover insights into the relationships between variables and the target, ultimately supporting more informed decision-making.

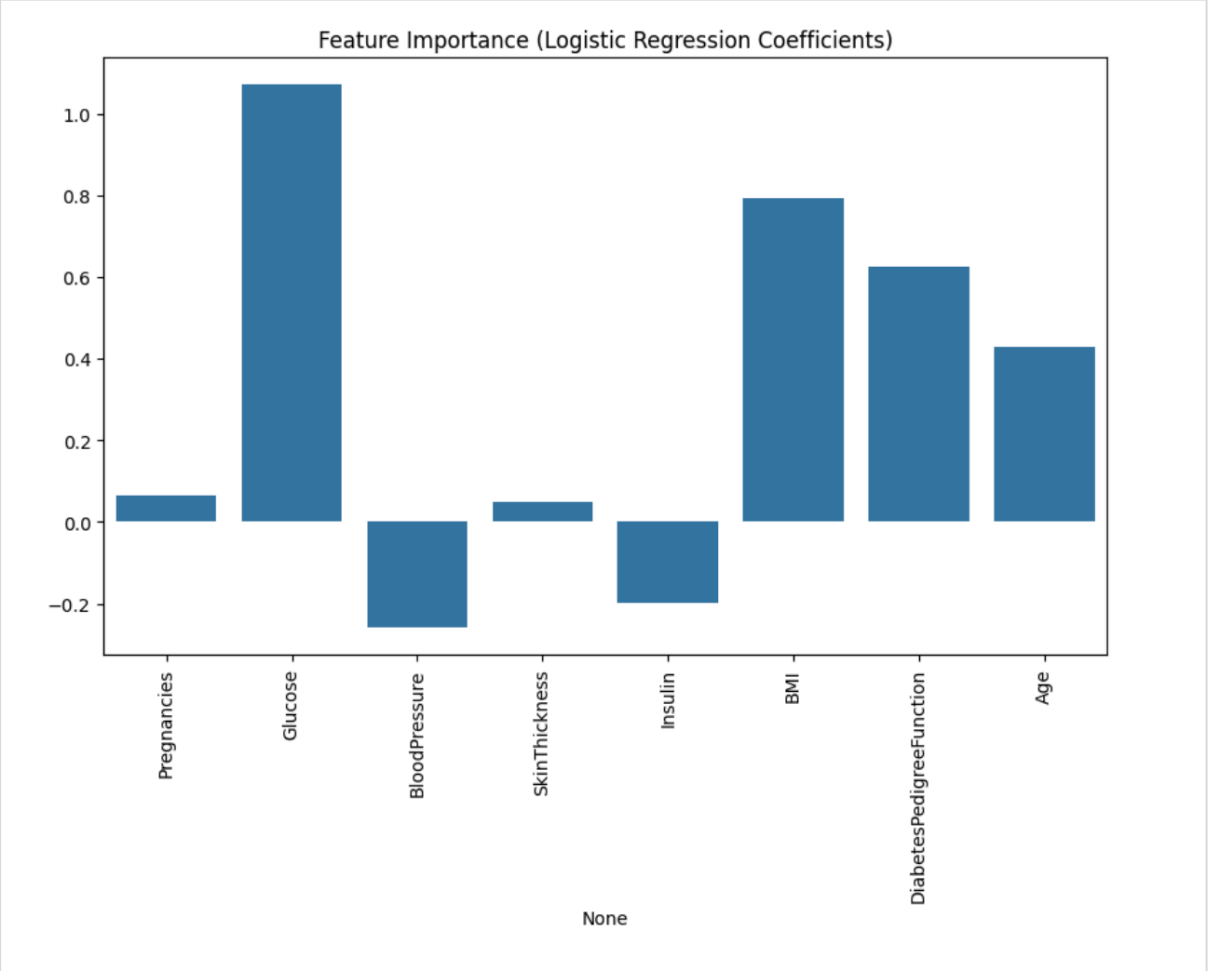


Figure 25. Feature Importance Plot

---

5. Project Timeline

5.1 Project Management Board

The project timeline and updated deliverables will be stored on GitHub:
(Link: https://github.com/users/solaecloud/projects/2)

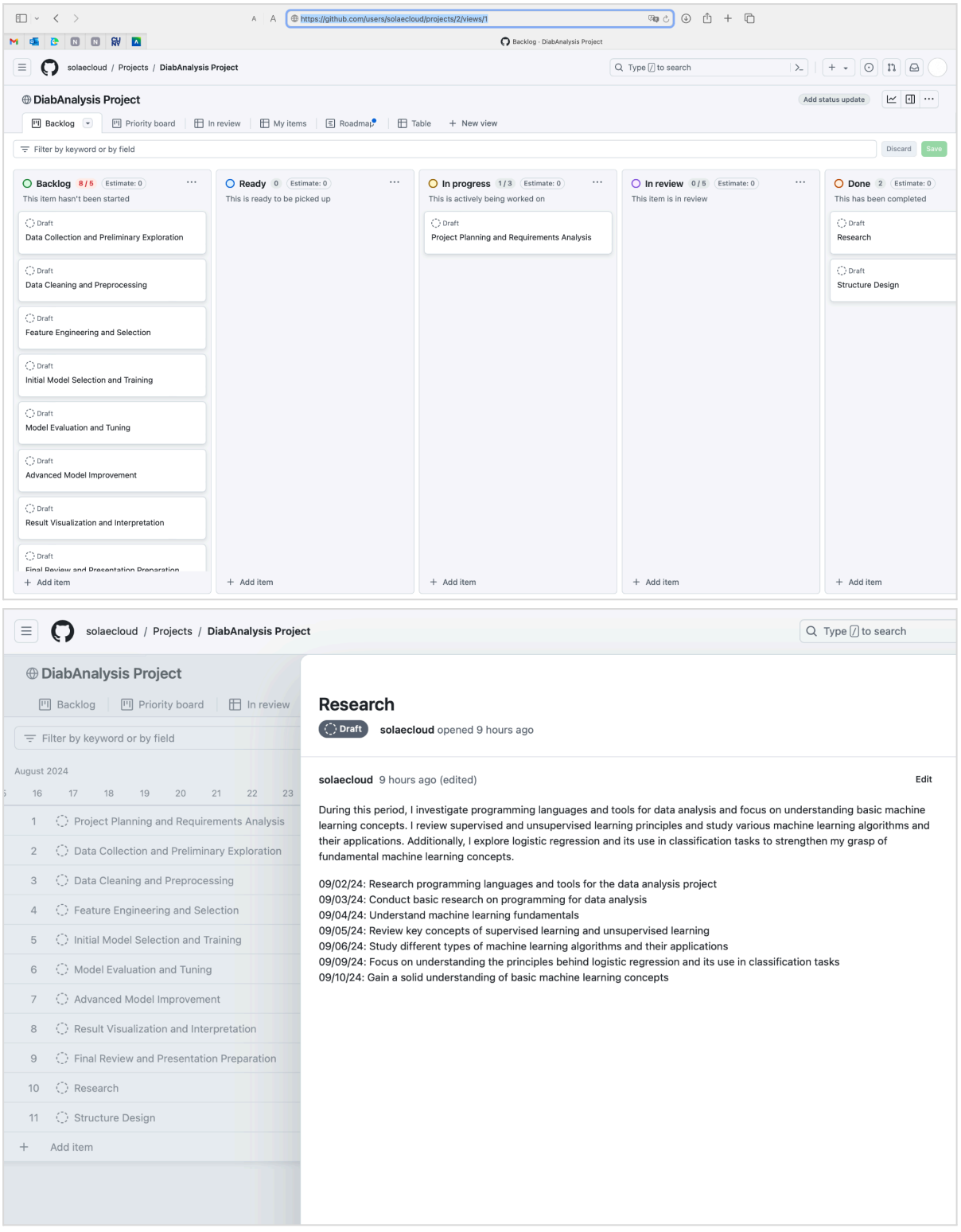


Figure 26. Samples of GitHub Project Management Boards. Tasks for each week have been organized by progress stages.

5.2 Project Activity Summary

Dates	Major Activities

Week 1	Project orientation and topic brainstorming session
Week 2	Research and exploration of data analysis
Week 3	Prototype model design based on data analysis research
Week 4	Shaping overall design and managing the repository
Week 5	Dataset collection, exploration, and feature understanding
Week 6	Studying data analysis techniques with examples from YouTube
Week 7	Creating detailed project documentation
Week 8	Organizing and documenting presentation slides
Week 9	Completing the draft code
Week 10	Loading and preprocessing data
Week 11	Initial model selection and training
Week 12	Evaluating and fine-tuning the model
Week 13	Implementing advanced model improvements
Week 14	Visualizing and interpreting results
Week 15	Conducting final review and preparing the presentation

5.3 Total Hours

Activity Completed	Hours
Supervisor Discussions, Emails	3
Design	19
Coding	89
Testing & Debugging	20
Research, Training, Learning	54
Others / Documentation	56
Total Hours Logged	**241**

---

6. Practical Applications

6.1 Use Cases

Case 1:

Input: Individual health data including various features such as age, Body Mass Index (BMI), blood glucose levels, blood pressure, and family history.
Output: Probability of diabetes (a value between 0 and 1) and classification of risk level (e.g., low, medium, high).

Case 2:

Input: Patient health records based on the Kaggle dataset, including data such as blood glucose levels, weight, and family history.
Output: Diabetes risk prediction and personalized management recommendations (e.g., 'regular check-ups recommended,' 'lifestyle changes advised').

Case 3:

Input: Aggregated health information from the Kaggle dataset, such as average blood glucose levels, BMI, and diabetes prevalence by region.
Output: Analysis results for public health policy and strategy development (e.g., 'identification of high-risk areas,' 'assessment of prevention program needs').

6.2 Examples of Potential Applications

The insights and methodologies derived from this project can be applied across various domains to address pressing health challenges and improve public health outcomes. The following examples illustrate how the techniques and findings from this project can be utilized in real-world scenarios:

Medical Research Data Analysis: According to the 2023 data from the International Diabetes Federation (IDF), approximately 530 million people worldwide suffer from diabetes, which accounts for about 10.5% of the global adult population. To address this issue, researchers can analyze global health datasets to study diabetes onset patterns, understand risk factors, and develop new treatment methods and preventive strategies. This analysis provides essential foundational data for advancing research and improving public health interventions.

Personal Health Management Apps: The prevalence of diabetes continues to rise, particularly in low-income countries. Given the substantial costs associated with diabetes management and treatment, providing a personal health management app that assesses diabetes risk can enable individuals in underserved areas to monitor their health status in real time. This proactive approach allows users to identify risk factors early and take preventive measures, thereby improving health outcomes even in regions with limited medical resources.

Addressing the Healthcare Workforce Shortage: To alleviate the shortage of healthcare professionals, hospitals and clinics can implement automated diabetes prediction systems. These systems evaluate patients' health statuses in advance, enabling medical staff to prioritize and focus on those requiring urgent care. This approach can reduce the burden on

healthcare professionals and enhance the overall quality of care, contributing to more efficient and effective healthcare delivery.

---

7. References

1. Ng, A. (2017, August 15). *Machine Learning Specialization* [Video series]. YouTube. DeepLearning.AI. https://www.youtube.com/watch?v=vStJoetOxJg&list=PLkDaE6sCZn6FNC6YRfRQc_FbeQrF8BwGI
2. GeeksforGeeks. (2023, October 28). *Libraries in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/libraries-in-python/
3. Geekster. (n.d.). *Data Analyst Project for Beginners: Analysis of Diabetes Health*. Geekster. https://www.geekster.in/articles/data-analyst-project-for-beginner-analysis-of-diabetes-health/
4. Medium. (n.d.). *A Comparison Between Linear and Logistic Regression*. Image source
5. ResearchGate. (n.d.). *Illustrations of Machine Learning Models: Linear and Logistic Regression*. Image source
6. Byju's. (n.d.). *Box Plot*. Image source
7. Deparkes. (2018, February 16). *The ROC Curve*. Image source