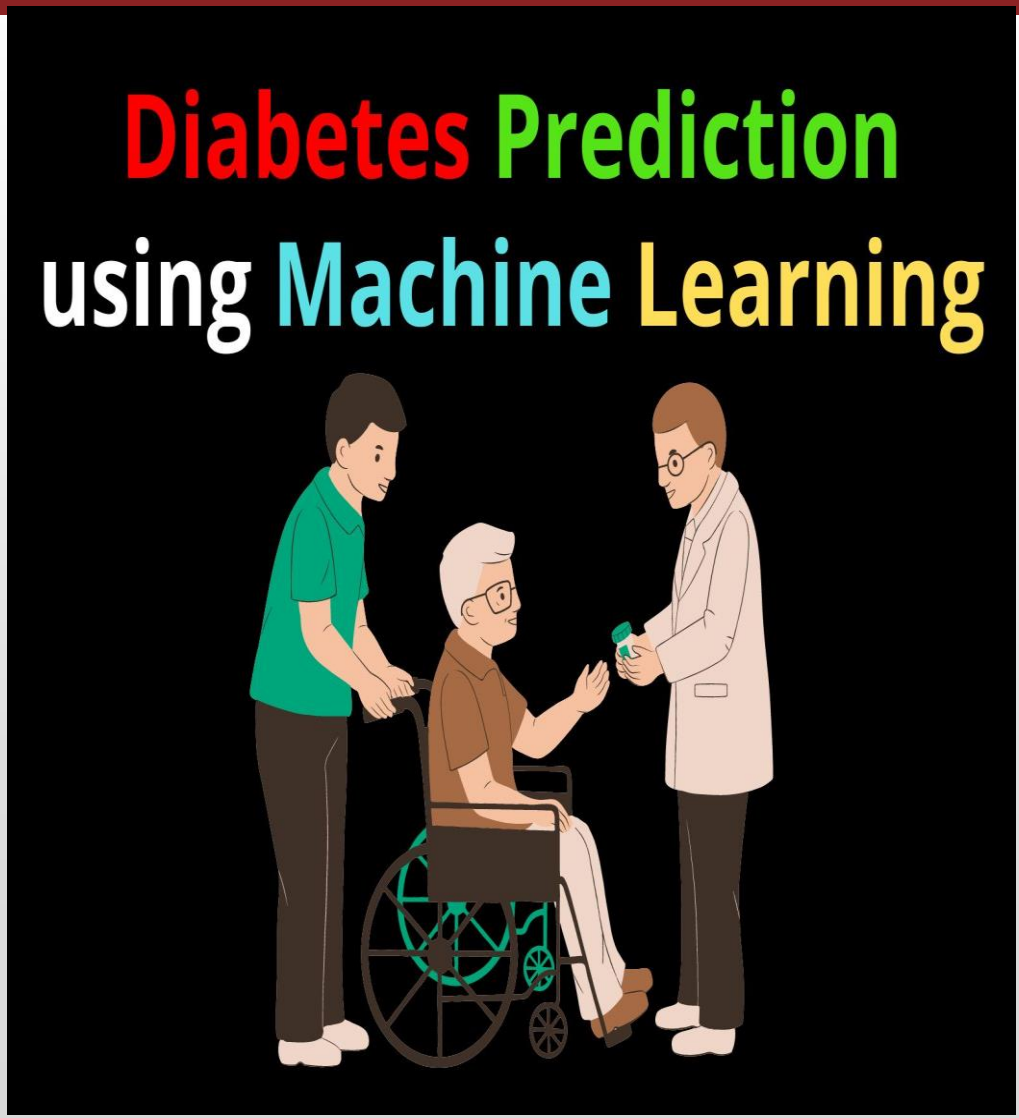
Diabetes Prediction

Introduction

- Diabetes is a global health concern, affecting millions of people across all age groups. It is a chronic condition where the body is unable to regulate blood sugar levels effectively, leading to serious health complications such as heart disease, kidney failure, blindness, and even premature death.
- Timely and accurate detection of diabetes is crucial for managing the disease and preventing its severe consequences.



Objective:

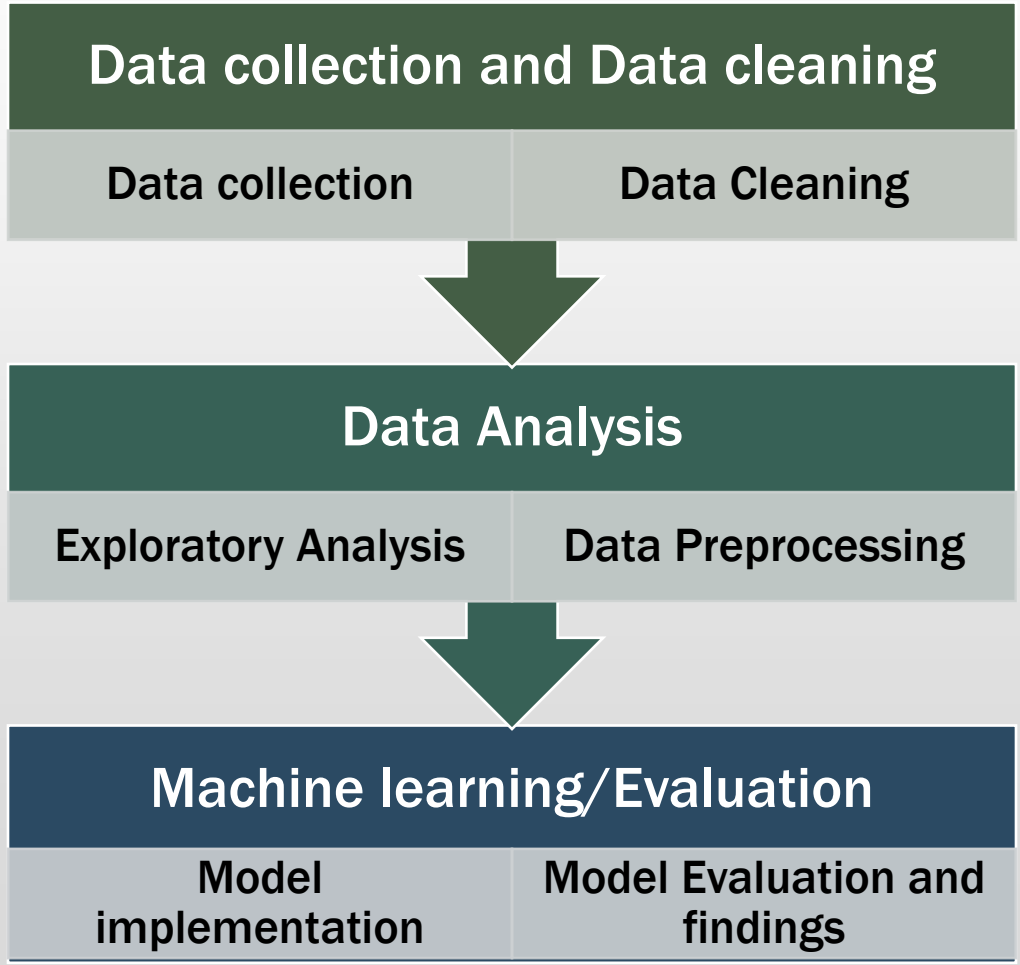
- To develop a machine learning model that can predict the likelihood of diabetes based on demographic and health-related variables such as:
  - Age
  - Blood glucose levels
  - Hypertension status
  - Smoking history
  - BMI and other relevant health metrics

Importance:

- Early detection of diabetes allows healthcare providers to intervene at a stage when the disease is still manageable.
- Machine learning models can enhance traditional diagnostic tools by identifying hidden patterns and risk factors from large datasets.
- Improved accuracy in prediction models can lead to better resource allocation in healthcare, such as targeted screenings for high-risk populations.
- By predicting diabetes in its early stages, patients can adopt lifestyle changes or treatments that significantly reduce the risk of complications.

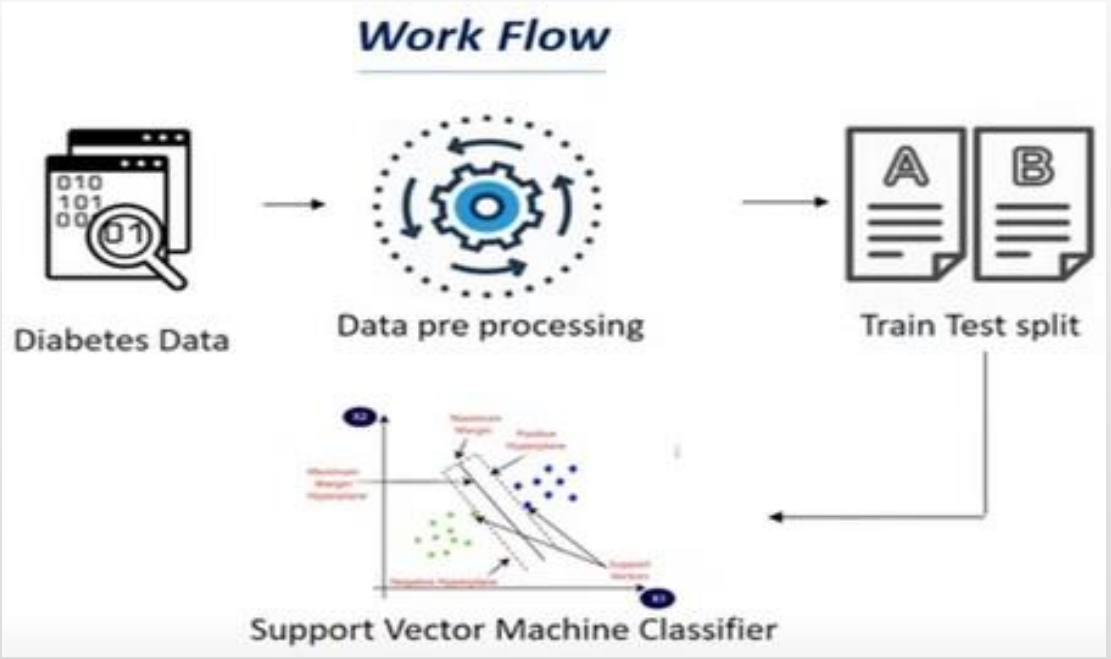
Methodology
Data Collection and Preprocessing:

- The dataset used, diabetes.csv, includes important health metrics like age, blood glucose levels, hypertension status, and smoking history.
- Handling missing values(this data have no missing value using the .isna() function
- Used a label encoder in encoding categorical variables(smoking history), and categorizing age groups (Young Adults, Adults, Old Aged).
- Bar plots and count plots were used to analyze the distribution of variables such as gender, smoking history, and age.

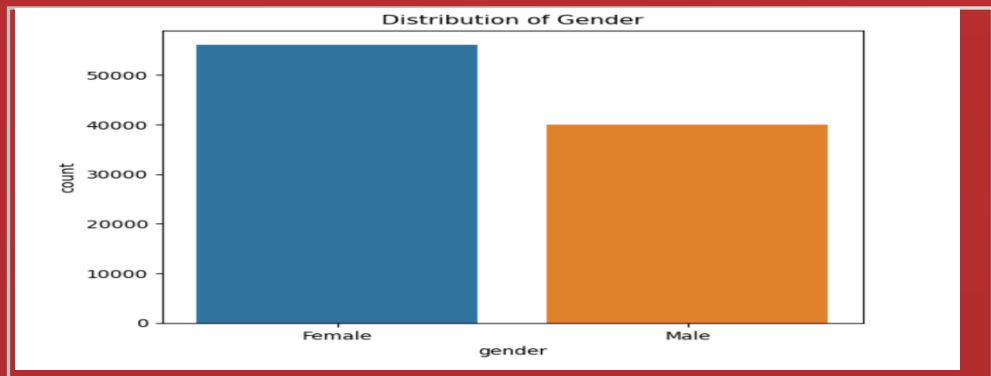


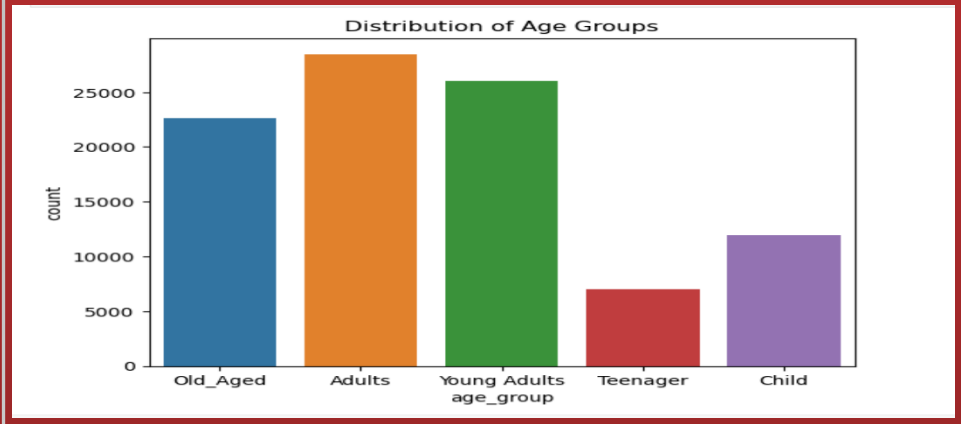
Tools and Algorithms

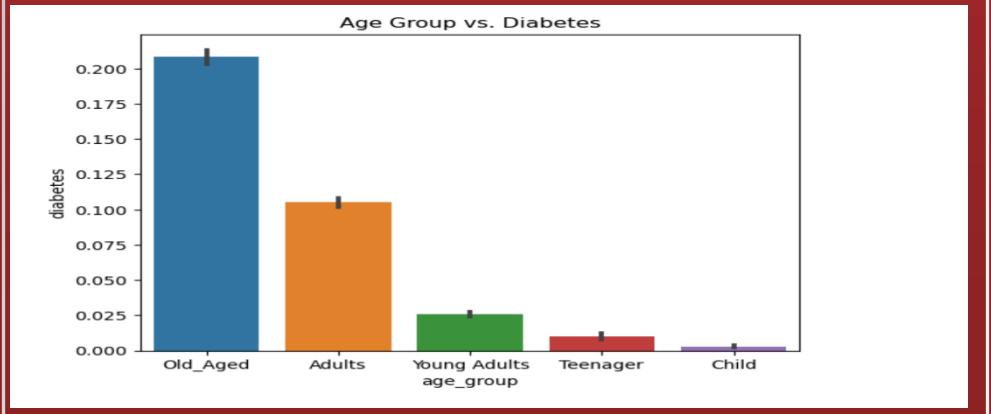
- Python libraries was used for data manipulation and modelling:
  - Pandas, NumPy (Data Manipulation)
  - Matplotlib, Seaborn (Data Visualization)
  - Scikit-learn (Modelling)
- **5 different algorithms was tested for this work:**
  - Linear Regression
  - Ridge Regression
  - Lasso Regression
  - Gradient Boosting
  - Random Forest Regressor
- **Model Training:**
  - Data was splited into 80% training and 20% test
  - Performance metrics used was Mean Squared Error (MSE), R-squared, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)



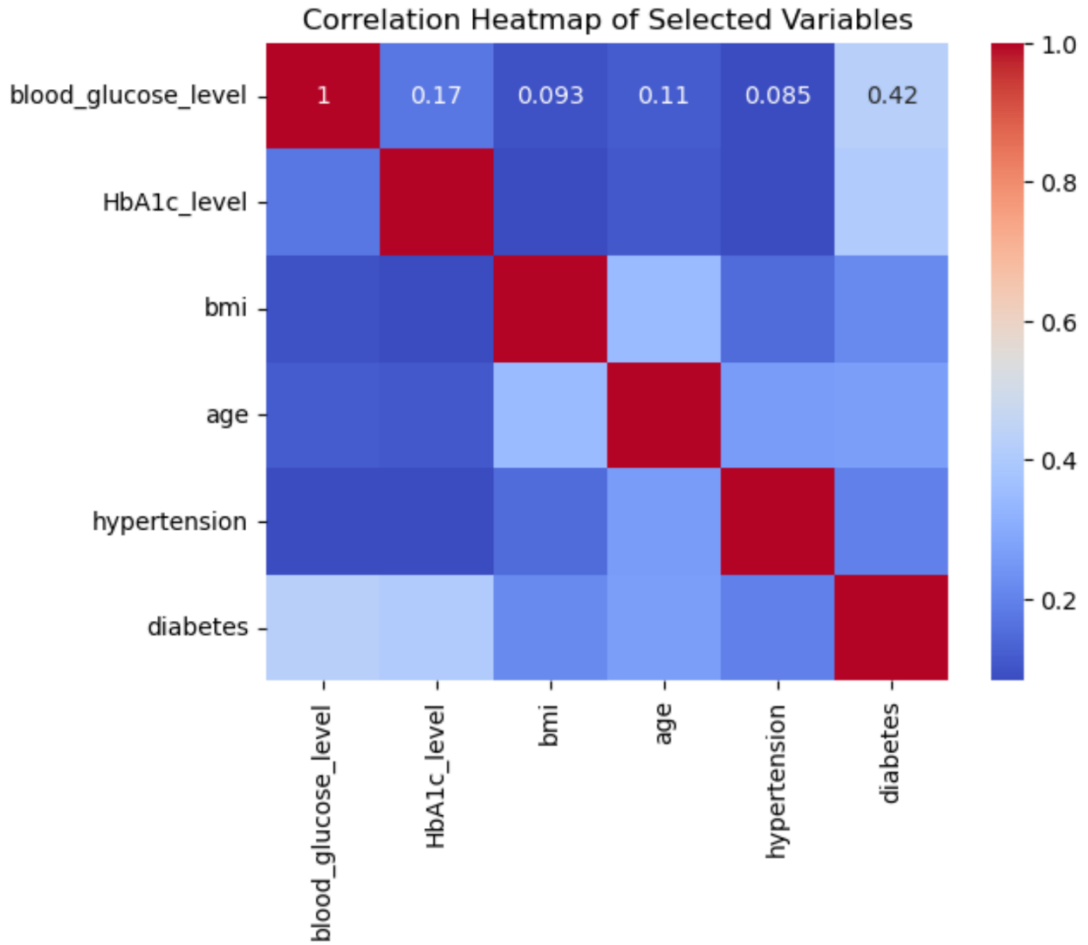
- The dataset showed that there are more **Adults** in the datasets in terms of population size, followed by **Young Adults** and **Old Aged** groups.
- **Females** are more represented in the dataset compared to males.
- Relationship between **age** and **diabetes**, with older individuals having a higher likelihood of developing diabetes.







• Key Findings from EDA



Correlation Analysis:

- Blood glucose level and diabetes have a moderate positive correlation (0.42), suggesting it is a key predictor for diabetes.
- Age and hypertension showed a very weak correlation, indicating little relationship.

RESULTS SECTION

Linear Regression Model

- MSE, MAE, and RMSE: These error values are relatively small, indicating that the model's predictions are generally close to the actual values.
- R^2 Score: The moderate R^2 of 0.3467 suggests that the model doesn’t explain much of the variability in the target variable, meaning it could be underfitting.

```
Mean Squared Error (MSE): 0.05268877957834909
R-squared (R2 Score): 0.3466683842462852
Mean Absolute Error (MAE): 0.1579083727735746
Root Mean Squared Error (RMSE): 0.22954036590183674
```

Ridge Regression Model

```
# Training the model and assessing feature importance
# Initialize and train a Ridge Regression model
ridge_model = Ridge(alpha=1.0)  # You can tune the alpha value
ridge_model.fit(X_train, y_train)

# Make predictions
y_pred_ridge = ridge_model.predict(X_test)

# Evaluate the model
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
r2_ridge = r2_score(y_test, y_pred_ridge)

print(f"Ridge - MSE: {mse_ridge}, R²: {r2_ridge}")
```

```
Ridge - MSE: 0.052688764589597384, R²: 0.3466685701041746
```

- The MSE for both Ridge and Linear Regression is nearly identical. In fact, the difference is so small that it suggests Ridge has not significantly changed the overall error.
- R^2 scores are also extremely close between the two models, This shows that Ridge has not significantly improved the proportion of variance explained by the model.

Lasso Regression Model

```
# Training the model and assessing feature importance
# Initialize and train a Lasso Regression model
lasso_model = Lasso(alpha=0.1)  # You can tune the alpha value
lasso_model.fit(X_train, y_train)

# Make predictions
y_pred_lasso = lasso_model.predict(X_test)

# Evaluate the model
mse_lasso = mean_squared_error(y_test, y_pred_lasso)
r2_lasso = r2_score(y_test, y_pred_lasso)

print(f"Lasso - MSE: {mse_lasso}, R²: {r2_lasso}")
```

```
Lasso - MSE: 0.061551101574536644, R²: 0.23677714752691836
```

- Lasso's MSE (0.0615) is worse than both Ridge and Linear Regression, indicating that Lasso has a higher prediction error.
- Lasso's R^2 (0.2367) is lower than Ridge (0.3467) and Linear Regression (0.3467), meaning that Lasso explains a lower proportion of the variance in the target variable.
- This indicates that Lasso was less effective in capturing the underlying patterns in the data compared to Ridge and Linear Regression.

Gradient Boosting Model

```
# Training the model and assessing feature importance
# Now, train your model without the warning
gb_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.5, random_state=42)
gb_model.fit(X_train, y_train)

# Make predictions
y_pred_gb = gb_model.predict(X_test)

# Evaluate the model
mse_gb = mean_squared_error(y_test, y_pred_gb)
mae_gb = mean_absolute_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)
```

```
# Print evaluation metrics
print(f"Gradient Boosting - MSE: {mse_gb}")
print(f"Gradient Boosting - MAE: {mae_gb}")
print(f"Gradient Boosting - R²: {r2_gb}")
```

```
Gradient Boosting - MSE: 0.02511773913414889
Gradient Boosting - MAE: 0.05711154011618834
Gradient Boosting - R²: 0.6885444448719571
```

- MSE of 0.0251 suggests that the model has relatively small prediction errors.
- MAE of 0.0571 also indicates that your model's predictions are fairly close to the actual values.
- R^2 of 0.6885 means that approximately 68.85% of the variance in the target variable is explained by the model.

Random Forest Regressor

```
# Training the model and assessing feature importance
# Initialize the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)  # You can tune n_estimators and other hyperparameters

# Train the model
rf_model.fit(X_train, y_train)  # No need to scale features for Random Forest
```

```
RandomForestRegressor(random_state=42)
```

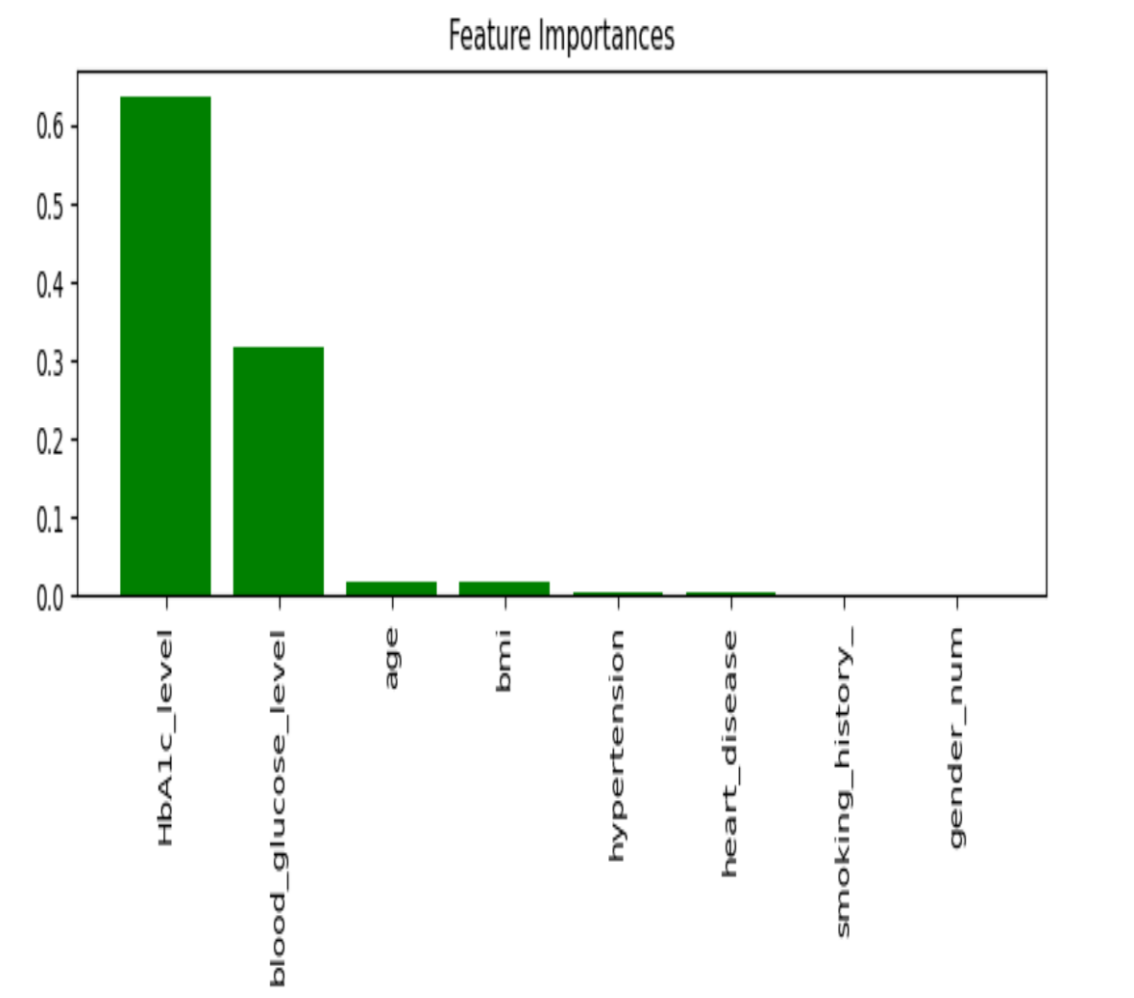
```
# Training the model and assessing feature importance
# Make predictions on the test set
y_pred_rf = rf_model.predict(X_test)

# Evaluate the Random Forest model
mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
rmse_rf = np.sqrt(mse_rf)
```

```
print(f"Random Forest - MSE: {mse_rf}")
print(f"Random Forest - MAE: {mae_rf}")
print(f"Random Forest - R²: {r2_rf}")
print(f"Random Forest - RMSE: {rmse_rf}")
```

```
Random Forest - MSE: 0.028125371194898167
Random Forest - MAE: 0.0527750386381537
Random Forest - R²: 0.6512503353942448
Random Forest - RMSE: 0.16770620499819966
```

- (MAE) - 0.0528: This is lower than the Gradient Boosting MAE (0.0571), suggesting that the Random Forest model has slightly smaller average absolute errors in its predictions.
- R^2 - 0.6513: This is lower than the Gradient Boosting R^2 (0.6885), meaning the Gradient Boosting model explains a bit more of the variance in the target variable compared to the Random Forest model.
- Random Forest model has a slightly higher MAE, it has a lower MSE and RMSE compared to the Gradient Boosting model. The Gradient Boosting model has a better R^2

Conclusion



- The study showed that Random Forest and Gradient Boosting were the most effective models for predicting diabetes, with R^2 scores of 0.65 and 0.69.
- These models effectively identified key health factors, like HbA1c levels and blood glucose levels.
- On the other hand, Lasso Regression didn't perform as well.

Recommendations:

- Use Random Forest or Gradient Boosting for health-related predictions due to their strong performance.
- Focus on feature selection to simplify models and improve interpretability.
- Expand data collection to include additional health indicators for better predictions.