Diabetes Risk Prediction using Machine Learning

Name - Uzair Sunkad
USN- 1RVU22CSE182
Date - 11-11-2024
Institution-RV University

Abstract:

This project, Diabetes Risk Prediction using Machine Learning, aims to develop a predictive model to identify individuals at risk of diabetes. Given the rising prevalence of diabetes worldwide and its association with severe health complications, early identification of at-risk individuals is crucial for timely intervention and management. Using a dataset with key health indicators—such as glucose levels, blood pressure, BMI, and age—this study explores patterns and builds machine learning models to predict diabetes risk.

The analysis includes comprehensive data preprocessing, exploratory data analysis (EDA), and model development with evaluation metrics to gauge performance. Results demonstrate the model's potential to assist in proactive health management by accurately distinguishing between diabetic and non-diabetic cases. This project emphasizes the impact of machine learning in healthcare, providing actionable insights for diabetes prediction and contributing to early intervention efforts.

Table of Contents:

Introduction

Background

Diabetes is a significant and growing health issue globally, with an increasing number of individuals diagnosed each year. This chronic condition can lead to severe complications, including heart disease, nerve damage, and kidney failure, making early detection essential. Although diabetes cannot be cured, it can be effectively managed through timely intervention and proper treatment. Understanding the factors that contribute to diabetes risk can aid in proactive management and reduce long-term health impacts.

As someone who seen his family member personally experience gestational diabetes, I am motivated to use this project to better understand diabetes, particularly its impact on women. The goal is to utilize machine learning models to identify individuals at risk, enabling early interventions and ultimately improving the management of diabetes.

Objective

The primary objective of this project is to use machine learning techniques to predict the likelihood of an individual developing diabetes. The objective of this project is to build a machine learning model that can predict whether an individual is at risk of diabetes based on the provided features. The model aims to achieve high accuracy in distinguishing between diabetic and non-diabetic individuals, with the goal of providing insights into early diabetes detection.

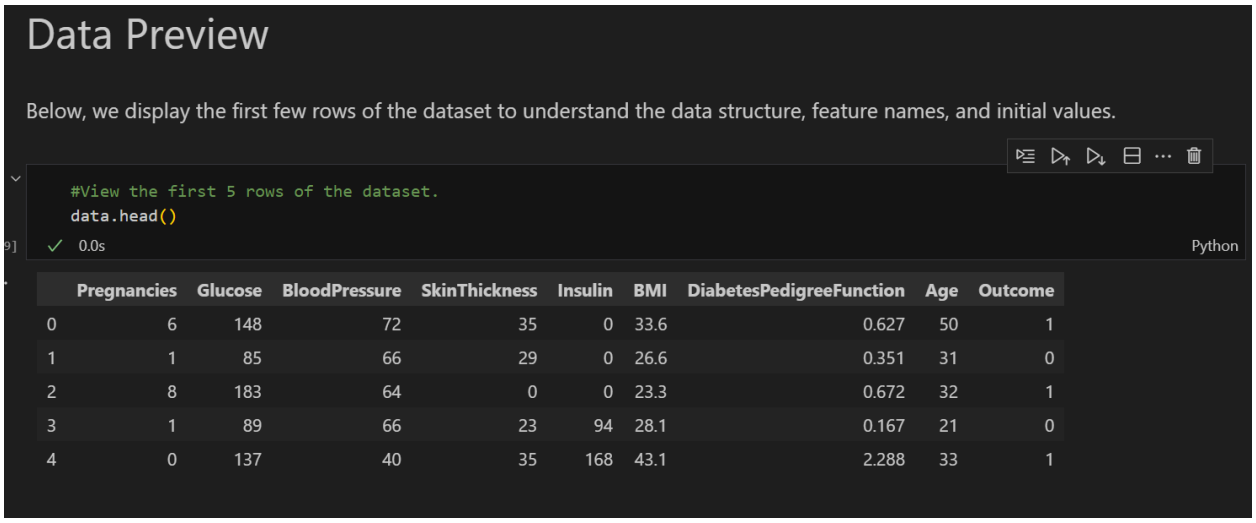
Data Description

Data Source

The dataset for this project is sourced from Kaggle, specifically the Diabetes Dataset. This dataset includes various medical and demographic features commonly associated with diabetes risk. Data set link : https://www.kaggle.com/datasets/saurabh00007/diabetescsv

Dataset Overview

- **Number of Rows**: 768
- **Number of Columns**: 9
- **Features**: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age
- **Variable Descriptions**
- **Pregnancies**: Number of times the individual has been pregnant.
- **Glucose**: Plasma glucose concentration during a 2-hour oral glucose tolerance test.
- **BloodPressure**: Diastolic blood pressure in mm Hg.
- **SkinThickness**: Triceps skinfold thickness in mm.
- **Insulin**: 2-hour serum insulin concentration (mu U/ml).
- **BMI**: Body mass index, calculated as weight in kg/(height in m)^2.
- **DiabetesPedigreeFunction**: A score indicating the likelihood of diabetes based on family history.
- **Age**: Age of the individual in years.
- **Outcome**: Class variable (0 = not diabetic, 1 = diabetic), used as the target for prediction.



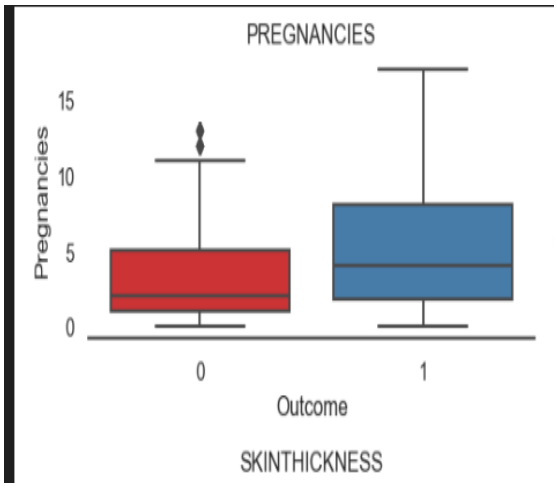
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Data Quality

Upon initial inspection, the dataset may contain some issues that could affect analysis:

- **Missing Values**: Certain fields, such as BMI and Insulin, may have missing or null values.
- **Outliers**: Some features could contain outliers, potentially skewing analysis.
- **Inconsistencies**: The range of values for certain variables may suggest data entry errors, requiring further validation and cleaning.

SkinThickness	Insulin
35	0
29	0
0	0
23	94
35	168



Data Preprocessing

Data Cleaning

To prepare the data for analysis, several data cleaning steps were performed:

- **Handling Missing Values**: Missing values in key features, such as BMI and Insulin, were addressed by imputing median values.
- **Removing Duplicates**: The dataset was checked for any duplicate records to ensure the integrity of the data, and duplicates were removed if found.
- **Outlier Detection and Treatment**: Outliers in features such as Glucose and BloodPressure were identified using statistical techniques (e.g., IQR method) and addressed as appropriate to maintain data quality.

```
# Check for any zero values in the specified columns after replacement
columns_to_check = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
for column in columns_to_check:
    zero_count = (data[column] == 0).sum()
    print(f"Zero values in {column}: {zero_count}")
```

✓ 0.0s

```
Zero values in Glucose: 0
Zero values in BloodPressure: 0
Zero values in SkinThickness: 0
Zero values in Insulin: 0
Zero values in BMI: 0
```

Feature Engineering

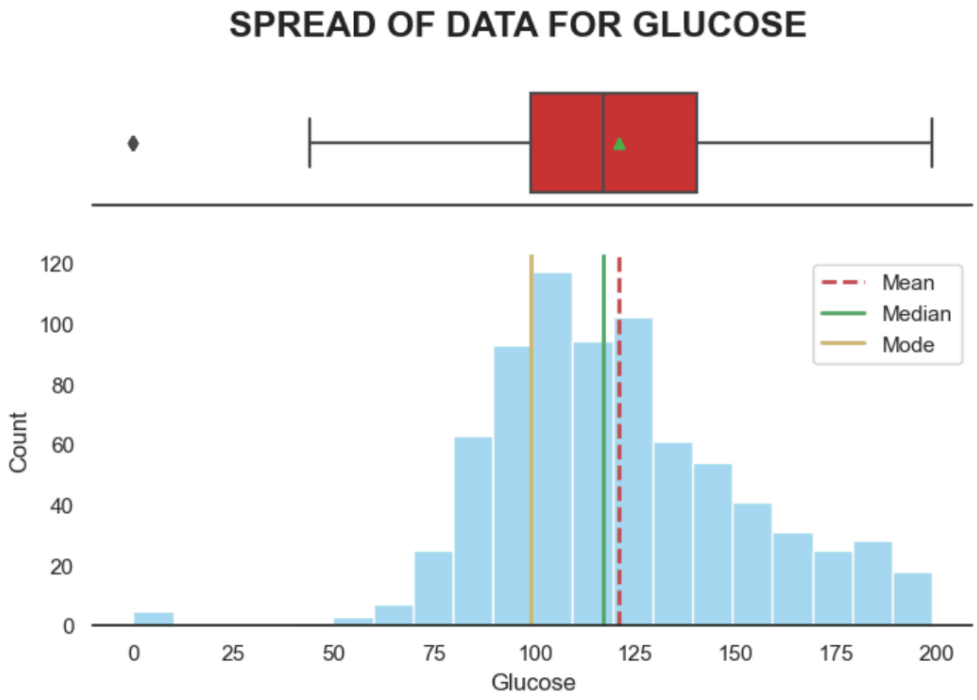
New features were created to enhance model performance and capture potentially meaningful patterns in the data:

- **BMI Category**: BMI was categorized into groups (e.g., underweight, normal, overweight, obese) to provide additional context for the model.
- **Age Grouping**: Age was binned into ranges to capture age-related risk patterns.

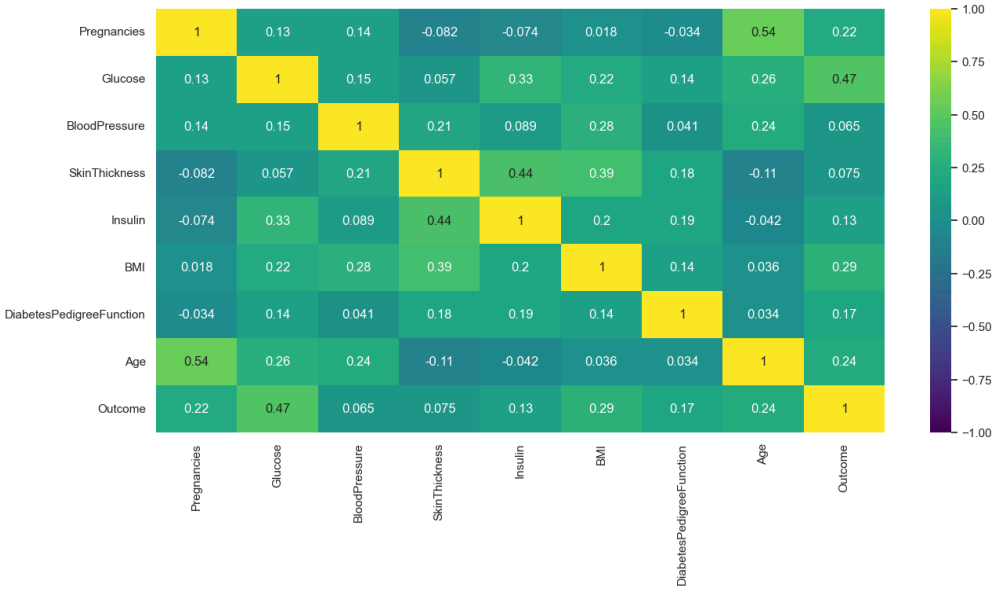
Feature Selection

To improve model performance and reduce dimensionality, a feature selection process was applied:

- **Correlation Analysis**: A correlation heatmap was used to identify redundant features. Features with high correlations were evaluated, and one of each pair was considered for removal to prevent multicollinearity.
- **Univariate Analysis**: Statistical tests were conducted to confirm the significance of each feature in predicting diabetes risk.



- **Bivariate Analysis:** Using this, we examined the relationships between the variables in the dataset. A correlation matrix is plotted using a heatmap to visualize how different features are correlated with each other. The correlation coefficients range from -1 to 1, where(1) A correlation of 1 indicates a perfect positive correlation (2) A correlation of -1 indicates a perfect negative correlation (3) A correlation of 0 indicates no linear relationship.



Data Transformation

To ensure that features were suitable for machine learning models, several transformations were applied:

- **Scaling**: Numeric features like Glucose, BloodPressure, and BMI were scaled using standardization or normalization to bring them to a common scale.

Exploratory Data Analysis (EDA)

1. **Overview of Insights**
   - The exploratory analysis aimed to identify key patterns and relationships within the dataset that might inform the modeling process.
2. **Visualizations and Findings**
   - Histograms were used to observe the distributions of features like `Age`, `BMI`, `BloodPressure`, and others.
   - A heatmap was generated to examine correlations among features, revealing that certain variables, such as `Glucose`, had stronger correlations with the target variable `Outcome`.
3. **Relationships and Patterns**
   - The analysis highlighted a positive relationship between `Glucose` and `Outcome`, indicating that higher glucose levels are more likely associated with positive diabetes diagnoses.
   - Some features displayed skewness, which may influence the model's performance and will be considered during model selection.
4. **Summary of EDA**
   - Key features like `Glucose`, `BMI`, and `Age` appear to play a significant role in predicting diabetes outcomes. These insights will guide the choice of model and the evaluation process.

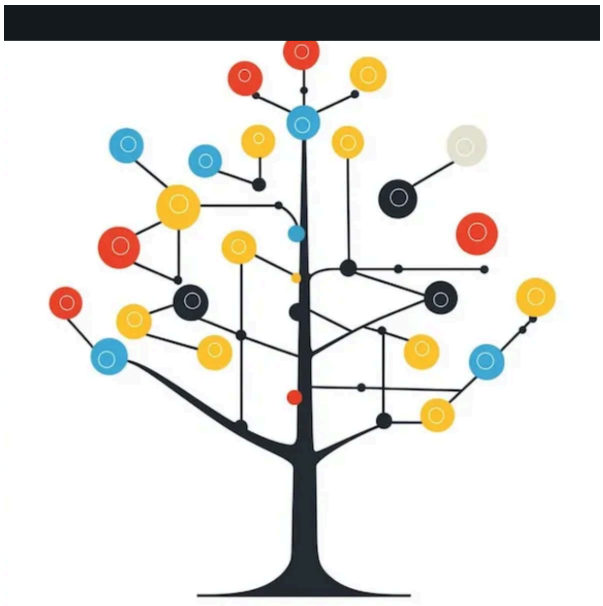
Modeling

Model Selection

The Decision Tree Classifier was chosen for this classification problem because it is a non-linear model capable of handling both numerical and categorical features effectively. Decision Trees are interpretable, easy to visualize, and often serve as a good baseline model for classification tasks. In this specific case, we are dealing with a healthcare dataset where the goal is to predict whether a patient is at risk for diabetes. A Decision Tree was selected because it provides clear

decision boundaries and allows for easy interpretation of how different features impact the decision-making process.

Additionally, Decision Trees are effective when the data has complex relationships or interactions between features. While simpler models like linear regression may struggle with non-linear data patterns, Decision Trees can handle such relationships well. However, one of the key assumptions is that there might be some level of overfitting if the tree grows too deep, which is something we observe initially in our model evaluation.



Training and Validation

To train and evaluate the model, the dataset was split into two primary subsets: training and test The train-test split allows for an unbiased evaluation by training the model on one portion of the data and testing it on a separate, unseen portion. The typical practice is to reserve around 70-80% of the data for training and the remaining 20-30% for testing.

In this case, the model was trained on the scaled training dataset (`X_train_scaled_df`), ensuring that the features were standardized to prevent certain features from dominating due to their scale. The split ensures that the model is evaluated on data it has not seen during training, providing a realistic measure of its performance.

```
# Splitting data into training and test set:
#The Stratify arguments maintains the original distribution of classes in the target variable while splitting the data in
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,stratify=y)
print(X_train.shape, X_test.shape)
```

✓ 0.0s Python

```
537, 8) (231, 8)
```

Hyperparameter Tuning:

To address the issue of overfitting, hyperparameter tuning was conducted using a grid search approach. The two key hyperparameters adjusted were:

(a) max_depth=5: This hyperparameter limits the depth of the decision tree. By restricting the depth, the model is less likely to create overly complex decision boundaries that are tailored to noise in the training data, thus reducing overfitting.
(b) min_samples_leaf=10: This hyperparameter ensures that each leaf node of the tree contains at least 10 samples. By enforcing a minimum number of samples per leaf, the tree avoids splitting the data into very small and unrepresentative groups, which helps with generalization.

The tuning process helps optimize the model's ability to balance fitting the training data and generalizing to new, unseen data, as seen by the improvement in the test performance after tuning.

```
# Fitting the Decision Tree model with max_depth=5 and min_samples_leaf=10
d_tree = DecisionTreeClassifier(random_state=1, max_depth=5, min_samples_leaf=10)
d_tree.fit(X_train_scaled_df, y_train)
```

Evaluation Metrics

The performance of the Decision Tree Classifier was evaluated using several key metrics:

Accuracy: This is the proportion of correct predictions out of all predictions made. While accuracy is a useful metric, it can be misleading if the data is imbalanced (e.g., if one class significantly outweighs the other). In this case, since the dataset likely contains imbalanced classes, accuracy alone might not provide a complete picture of the model's performance.

Precision: Precision is the ratio of correctly predicted positive instances to the total predicted positives. It is crucial in scenarios where false positives are costly or undesirable. For this problem, a higher precision means fewer patients are incorrectly predicted as being at risk for diabetes when they are not.

Recall: Recall is the ratio of correctly predicted positive instances to the total actual positives. This metric is particularly important in medical applications where missing a positive case (i.e., failing to identify a patient at risk of diabetes) can have serious consequences. The model should identify as many true positives as possible.

F1 Score: The F1 Score is the harmonic mean of precision and recall, providing a single measure of the model's performance that balances the trade-off between the two metrics. It's especially valuable when you want to optimize both recall and precision simultaneously.

Confusion Matrix: A confusion matrix provides a summary of the model's predictions by showing the number of true positives, false positives, true negatives, and false negatives, helping assess the model's classification performance.

AUC-ROC Curve: The AUC-ROC curve illustrates the model's ability to distinguish between classes by plotting the true positive rate against the false positive rate at different thresholds, with the area under the curve (AUC) representing the model's overall classification performance.

By using these metrics, we ensure that the model's performance is evaluated from multiple angles, addressing both the risk of overfitting and the critical importance of accurate predictions for patients at risk of diabetes.

```
Training Accuracy: 0.8342644320297952
Test Accuracy: 0.7272727272727273
Training Precision: 0.8397979871138976
Test Precision: 0.7315010570824525
Training Recall: 0.8342644320297952
Test Recall: 0.7272727272727273
Training F1 Score: 0.8359587392115196
Test F1 Score: 0.729044222802469
```

Results

Performance Summary:

After applying hyperparameter tuning to the Decision Tree model (with `max_depth=5` and `min_samples_leaf=10`), the model's performance significantly improved, addressing the initial overfitting issue. Below is a summary of the key metrics:These results demonstrate a balanced performance between training and test sets, indicating that the model is generalizing better and reducing overfitting compared to the initial iteration.

Metric	Training	Test
Accuracy	83.43%	72.73%
Precision	0.840	0.732
Recall	0.834	0.727
F1 Score	0.836	0.729

Comparison of Models:

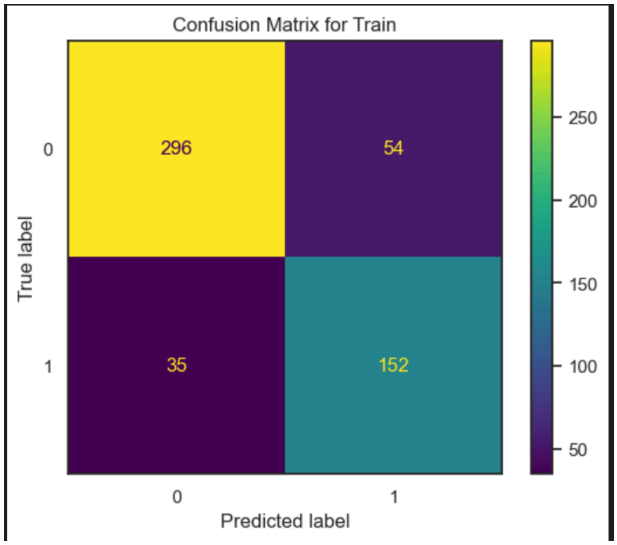
Since only the Decision Tree model was used in this analysis, a comparison with other models isn't available at this time. However, future analyses could involve comparing the Decision Tree's performance with other models such as Random Forest, Gradient Boosting, or Logistic Regression to assess which provides the best trade-off between accuracy, precision, recall, and F1 score.

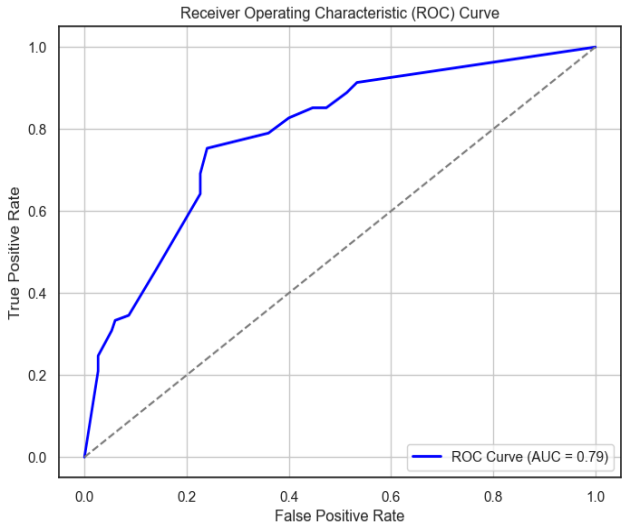
Interpretation:

The Decision Tree model provides useful insights into the factors contributing to diabetes risk, with **Glucose**, **Age**, and **BMI** being the most significant predictors. However, while the model performs well in classifying patients at risk, it struggles with certain instances where **Age** or **BMI** plays a more prominent role than **Glucose**. The model's performance in terms of **recall** suggests that it correctly identifies a large portion of high-risk patients, but there is room for improvement, particularly in its ability to recognize cases where other factors (such as age or BMI) are key indicators.

Visualizations:

1. **Confusion Matrix**: The confusion matrix can be visualized to display the number of true positives, false positives, true negatives, and false negatives for both the training and test datasets. This will provide insight into where the model is making errors, particularly in terms of misclassifying high-risk patients.
2. **ROC Curve**: The ROC curve, along with the AUC score, can help evaluate the model's ability to discriminate between at-risk and not-at-risk patients across all classification thresholds.
3. **Feature Importance Plot**: A plot of feature importance can be created to highlight which features, such as **Glucose**, **Age**, and **BMI**, are driving the model's predictions. This will confirm that glucose is indeed the most influential feature in determining diabetes risk.





Discussion

Key Findings:

The analysis highlighted that Glucose, Age, and BMI are the most significant factors in identifying individuals at risk of diabetes, with Glucose having the highest impact. These findings suggest that early monitoring of glucose levels, especially among pregnant women, could help prevent future health complications. The Decision Tree model successfully categorized women into Lower Risk and Higher Risk profiles based on these features, although it struggled when Age or BMI played a more critical role than glucose levels.

Challenges:

During the analysis, one of the main challenges was model overfitting, where the Decision Tree performed exceptionally well on the training data but poorly on the test set. This was addressed through hyperparameter tuning by adjusting parameters like max_depth and min_samples_leaf, which helped mitigate overfitting and improved the model's generalization. Another challenge was ensuring the model recognized the importance of features like Age and BMI equally with Glucose for more accurate predictions.

Limitations:

- **Misclassification**: The model sometimes failed to correctly identify high-risk individuals where Age or BMI were more influential than Glucose levels. This limitation could be overcome with better feature engineering and more advanced models.
- **Feature Imbalance**: Not all features may have been adequately preprocessed, or additional features could be incorporated to improve the model's predictive power.
- **Model Limitations**: While the Decision Tree provided useful insights, it could benefit from incorporating more complex models like Random Forest or Gradient Boosting to improve accuracy and recall.

Insights for Business or Scientific Impact:

For healthcare providers or businesses in the health sector, these findings can help in targeting at-risk populations, particularly among middle-aged and overweight women. Early identification of high-risk patients could lead to timely interventions, reducing the long-term impact of diabetes and associated healthcare costs. Additionally, this model could be applied in public health initiatives to develop screening programs based on Glucose, Age, and BMI thresholds.

In a research context, these results contribute to a deeper understanding of the key factors influencing diabetes risk, which could lead to more refined models for predicting disease progression and improving preventive measures. Further analysis and more sophisticated models could enhance the accuracy of these predictions and inform health policy decisions.

Conclusion and Future Work

Summary of Outcomes:

This analysis successfully identified key factors—Glucose, Age, and BMI—that contribute to predicting diabetes risk, with Glucose being the most influential feature. The Decision Tree model, after hyperparameter tuning, improved its generalization ability, reducing overfitting and achieving more balanced performance between the training and test sets. However, the model faced challenges in cases where Age or BMI played a more significant role than Glucose.

Overall, the project met its objectives of building a model that can classify individuals based on their risk of diabetes and offer valuable insights for healthcare professionals.

Suggestions for Future Improvements:

- **Explore More Advanced Models**: The Decision Tree model could be further enhanced by exploring alternative algorithms such as Random Forest, Gradient Boosting Machines, or Ensemble Methods, which are known for better handling overfitting and improving performance.
- **Feature Engineering**: Additional preprocessing or the creation of new features could be explored to better capture the relationships between variables, particularly to address misclassifications where Age or BMI is more significant than Glucose.
- **Expanding the Dataset**: Incorporating more diverse data sources or gathering additional data could lead to better generalization and more accurate predictions across different populations.
- **Fine-tune Hyperparameters Further**: Conducting a more exhaustive search for optimal hyperparameters through methods like Grid Search or Random Search could improve model performance, especially in terms of recall.

Implications:

The outcomes of this analysis have practical implications for healthcare providers, particularly in developing targeted interventions for individuals at risk of diabetes based on easily measurable factors like Glucose, Age, and BMI. By identifying high-risk patients early, healthcare systems can provide more effective prevention strategies. Future research could further refine these models and extend them to broader populations, potentially improving public health strategies and reducing the burden of diabetes-related complications. Additionally, continued work in feature engineering and exploring more sophisticated algorithms could lead to more accurate and robust predictive models.

References

1. **Pima Indians Diabetes Database** – The dataset used for this analysis, available from the UCI Machine Learning Repository.
   URL: https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes
2. **Scikit-learn Documentation** – Official documentation for the machine learning library used in model development and evaluation. URL: https://scikit-learn.org/stable/
3. **James, G., Witten, D., Hastie, T., & Tibshirani, R.** (2013). *An Introduction to Statistical Learning: with Applications in R.* Springer.

4. **Breiman, L.** (1986). *Classification and Regression Trees.* Wadsworth & Brooks/Cole.
5. **Chollet, F.** (2017). *Deep Learning with Python.* Manning Publications.
6. **Dataset Link-https: //www.kaggle.com/datasets/saurabh00007/diabetescsv**

Appendices

1. Data Preprocessing:

```
print ("#"*40,"\n","Features : \n\n", data.columns.tolist()) #get name of columns/features
print ("#"*40,"\nMissing values :\n\n", data.isnull().sum().sort_values(ascending=False))
print ("#"*40,"\nUnique values :  \n\n", data.nunique())  #  count of unique values
```

✓ 0.0s

```
import matplotlib.pyplot as plt
import seaborn as sns

def dist_box(data):
    # Function plots a combined graph for univariate analysis of a continuous variable
    # to check spread, central tendency, dispersion, and outliers
    Name = data.name.upper()
    fig, (ax_box, ax_dis) = plt.subplots(nrows=2, sharex=True, gridspec_kw={"height_ratios": (.25, .75)}, figsize=(8, 5))
    mean = data.mean()
    median = data.median()
    mode = data.mode().tolist()[0]

    sns.set_theme(style="white")
    sns.set_palette(sns.color_palette("Set1", 8))
    fig.suptitle("SPREAD OF DATA FOR " + Name, fontsize=18, fontweight='bold')

    sns.boxplot(x=data, showmeans=True, orient='h', ax=ax_box)
    ax_box.set(xlabel='')

    # Improve the visualization by setting the background to white
    sns.despine(top=True, right=True, left=True)  # Remove side line from graph

    # Use histplot instead of distplot (since distplot is deprecated)
    sns.histplot(data, kde=False, ax=ax_dis, color="skyblue", bins=20)

    ax_dis.axvline(mean, color='r', linestyle='--', linewidth=2)
    ax_dis.axvline(median, color='g', linestyle='-', linewidth=2)
    ax_dis.axvline(mode, color='y', linestyle='-', linewidth=2)

    plt.legend({'Mean': mean, 'Median': median, 'Mode': mode})
    plt.show()  # Explicitly call show to render the plot

# Assuming `data` is your DataFrame
# Select all numerical columns (excluding 'Outcome' as it's the target variable)
numerical_cols = data.select_dtypes(include='number').columns.to_list()
numerical_cols.remove('Outcome')  # Exclude the target variable

# Apply the dist_box function for each numerical column
for col in numerical_cols:
    dist_box(data[col])  # Plot for each numerical column
```

(2) Model Training and Hyperparameter Tuning:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Fitting the Decision Tree model
d_tree = DecisionTreeClassifier(random_state=1)
d_tree.fit(X_train_scaled_df, y_train)

# Function to calculate metrics and plot confusion matrix
def get_metrics_score(model, X_train, X_test, y_train, y_test):
    # Making predictions
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)

    # Calculate accuracy, precision, recall, and F1 score for train and test data
    accuracy_train = accuracy_score(y_train, y_pred_train)
    accuracy_test = accuracy_score(y_test, y_pred_test)

    precision_train = precision_score(y_train, y_pred_train, average='weighted')
    precision_test = precision_score(y_test, y_pred_test, average='weighted')

    recall_train = recall_score(y_train, y_pred_train, average='weighted')
    recall_test = recall_score(y_test, y_pred_test, average='weighted')

    f1_train = f1_score(y_train, y_pred_train, average='weighted')
    f1_test = f1_score(y_test, y_pred_test, average='weighted')

    # Print and return the metrics
    print(f"Training Accuracy: {accuracy_train}")
    print(f"Test Accuracy: {accuracy_test}")
    print(f"Training Precision: {precision_train}")
    print(f"Test Precision: {precision_test}")
    print(f"Training Recall: {recall_train}")
    print(f"Test Recall: {recall_test}")
    print(f"Training F1 Score: {f1_train}")
    print(f"Test F1 Score: {f1_test}")
```

(3) Model Evaluation

```
    # Confusion matrix plotting
    make_confusion_matrix(y_train, y_pred_train, "Confusion Matrix for Train")
    make_confusion_matrix(y_test, y_pred_test, "Confusion Matrix for Test")

    # Return metrics in a dictionary for further use
    return {
        'accuracy_train': accuracy_train,
        'accuracy_test': accuracy_test,
        'precision_train': precision_train,
        'precision_test': precision_test,
        'recall_train': recall_train,
        'recall_test': recall_test,
        'f1_train': f1_train,
        'f1_test': f1_test
    }

# Function to plot the confusion matrix
def make_confusion_matrix(y_true, y_pred, title):
    ConfusionMatrixDisplay.from_predictions(y_true, y_pred)
    plt.title(title)
    plt.grid(False)  # Turn off the grid
    plt.show()

# Calculating different metrics and plotting confusion matrices
score_list_dt = get_metrics_score(d_tree, X_train_scaled_df, X_test_scaled_df, y_train, y_test)
```

(4) Extended Visualizations

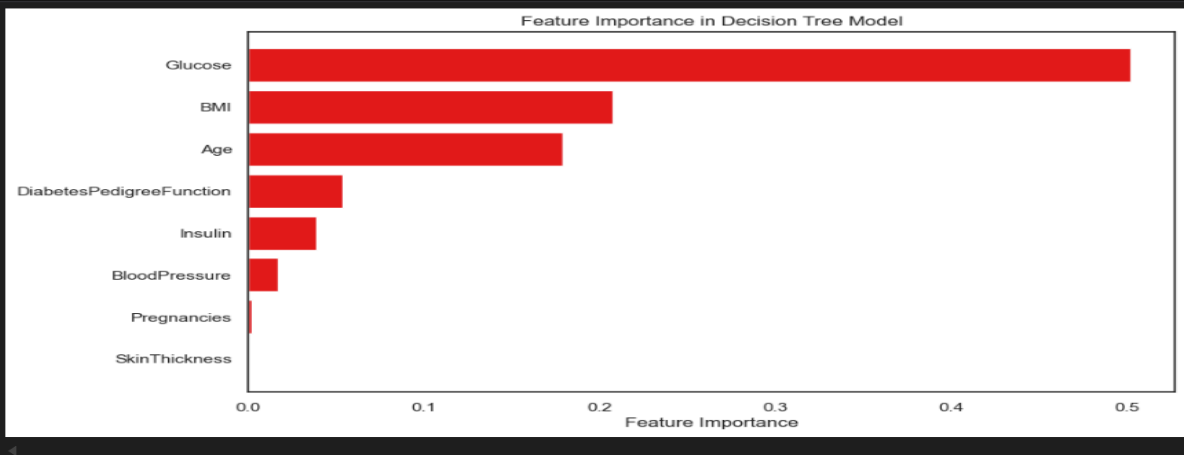
```
import matplotlib.pyplot as plt
import numpy as np

# Plot feature importance from the fitted Decision Tree model
feature_importance = d_tree.feature_importances_  # Use 'd_tree' instead of 'best_model'
features = X_train_scaled_df.columns  # Ensure 'X_train_scaled_df' has columns attribute

# Sort the feature importance
indices = np.argsort(feature_importance)

plt.figure(figsize=(10, 6))
plt.barh(features[indices], feature_importance[indices], align='center')
plt.xlabel('Feature Importance')
plt.title('Feature Importance in Decision Tree Model')
plt.show()
```

0.2s




```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Function to plot the ROC Curve
def plot_roc_curve(model, X_test, y_test):
    # Get predicted probabilities
    y_prob = model.predict_proba(X_test)[:, 1]  # Probabilities for the positive class (1)

    # Compute ROC curve and AUC
    fpr, tpr, thresholds = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)

    # Plot ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC Curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')  # Diagonal line
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.grid(True)
    plt.show()

# Plot the ROC curve after model training
plot_roc_curve(d_tree, X_test_scaled_df, y_test)
```

0.3s