DIABETESE PREDICTION

Predict whether a given personnel is diabetic or not based on available features.

Name: Misbah Ghafoor, Reg # 2024-MS-DS-118

Contents

1. Introduction

Problem Statement

Diabetes is a chronic health condition that affects how the body processes blood sugar (glucose). The problem is to analyze set of features to predict whether a patient is likely to develop diabetes. By identifying patterns and correlations in the data, we aim to create a model that can predict the risk of diabetes, enabling early diagnosis and more effective management of the disease.

Success Criteria

The success of this project will be measured using the following criteria:

- Data cleaning and preparation
- Identification of key features influencing diabetes risk.
- Train a supervise machine learning model to classify given personnel either diabetic or not based on given set of new features.

Data Science Team:

Name	Misbah Ghafoor
Registration#	2024-MS-DS-118

2. Data Collection

The data for this project is sourced from Kaggle, a renowned platform for open datasets. The dataset contains information related to patient demographics, health metrics, and diabetes-related factors. This data will serve as the foundation for building models to predict the likelihood of diabetes and identify key risk factors.

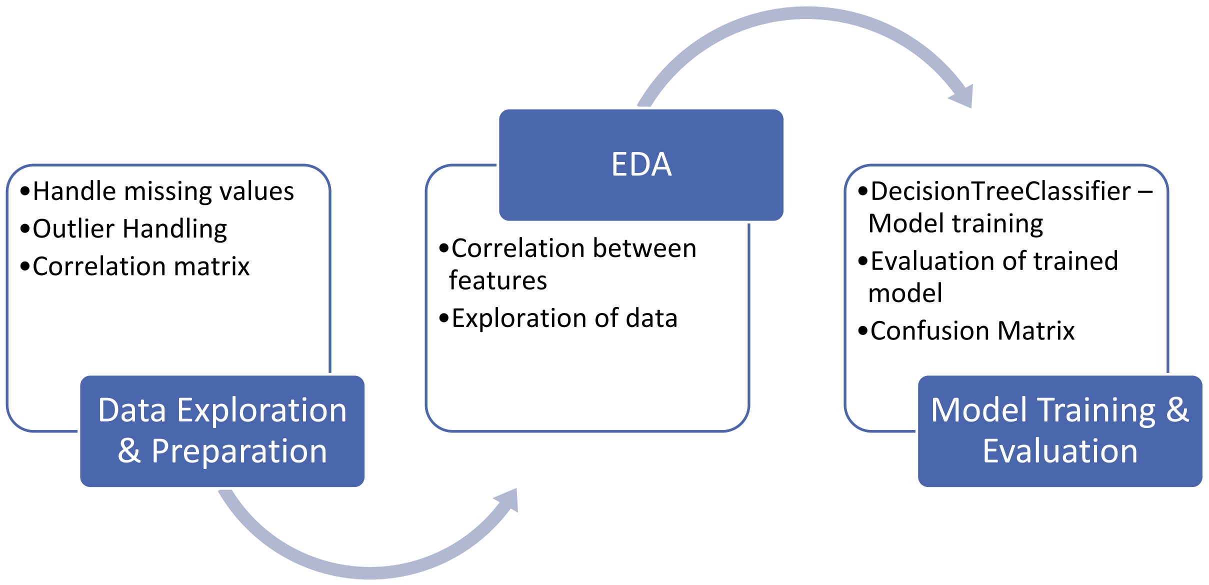
Data Source: [https://www.kaggle.com/datasets/mathchi/diabetes-data-set]

Key Components of the Dataset:

Sr #	Feature Title	Feature Description
1	Pregnancies	Number of times pregnant
2	Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3	Blood Pressure	Diastolic blood pressure (mm Hg)
4	Insulin	2-Hour serum insulin (mu U/ml)
5	BMI	Body mass index (weight in kg/(height in m)^2)
6	Diabetes Pedigree Function	Diabetes pedigree function
7	Age	Age (years)
8	Skin Thickness	Triceps skin fold thickness (mm)
9	Outcome	Class variable (0 or 1)

3. Project Methodology

The project is divided into three distinct parts according to the project requirements. The pipeline for the project is illustrated in the diagrams below.


•Handle missing values
•Outlier Handling
•Correlation matrix
Data Exploration & Preparation
EDA
•Correlation between features
•Exploration of data
•DecisionTreeClassifier – Model training
•Evaluation of trained model
•Confusion Matrix
Model Training & Evaluation

4. Data Exploration and Preparation

Data Cleaning: Handle missing values, duplicates, and outliers.

```
df.info()
```

```
<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
```

Handling Missing Values

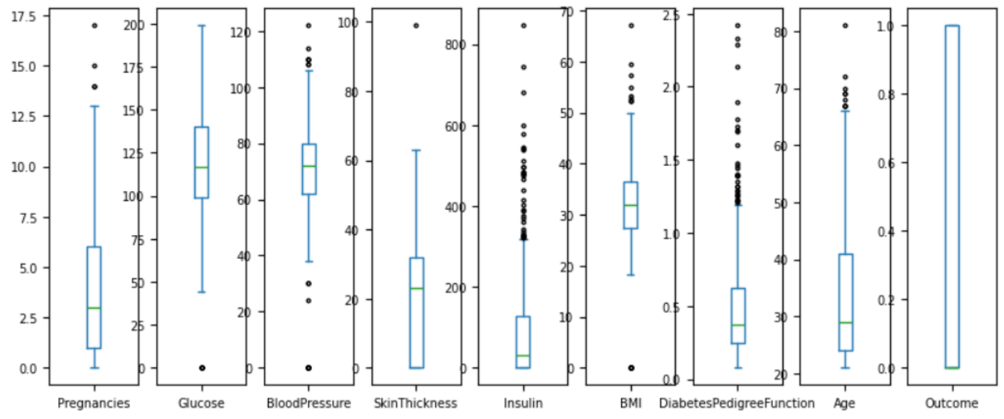
```
df.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

There are no missing values in the data.

Outliers Handling

```
df.plot(kind='box', subplots=True, sharex=False , sharey=False, figsize=(9, 4))
plt.show()
```



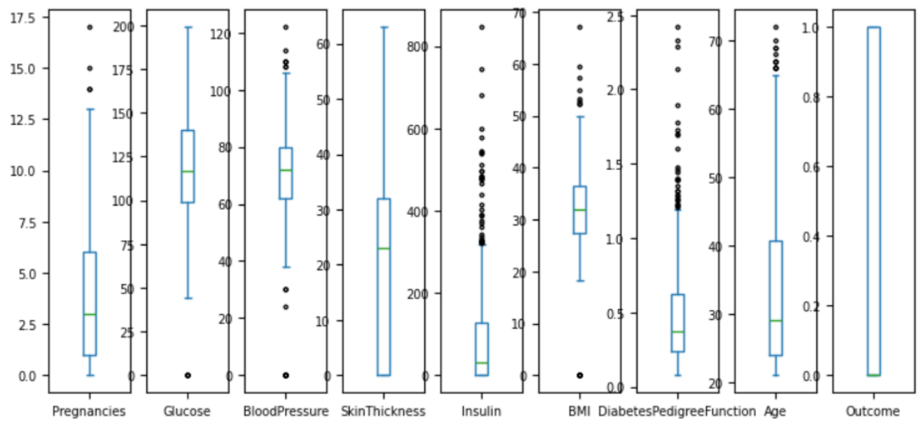
I will remove outliers from the **'Age'** and **'SkinThickness'** columns due to extreme values that could distort the analysis. For the other columns outliers were not removed as they represent valid medical data, and removing them could eliminate important information for predicting diabetes risk.

```
# Calculate Q1 and Q3
Q1 = df['SkinThickness'].quantile(0.25)
Q3 = df['SkinThickness'].quantile(0.75)
IQR = Q3 - Q1

# Detect outliers
upper_limit= Q3+IQR*2
lower_limit= Q1-IQR*2

# Display outliers for all columns
df=df.loc[(df['SkinThickness']<= upper_limit) & (df['SkinThickness']>= lower_limit)]
```

```
Q1=df['Age'].quantile(0.25)
Q3=df['Age'].quantile(0.75)
IQR = Q3 - Q1
upper_limit=Q3+IQR*2
lower_limit=Q1-IQR*2
df= df.loc[(df['Age']<=upper_limit) & (df['Age']>=lower_limit)]
```

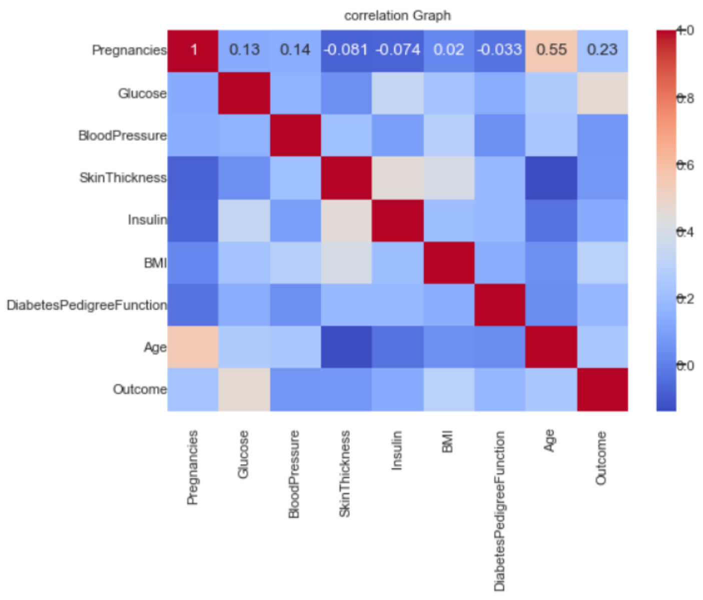
Box plot after outlier removal:

```
df.plot(kind='box', subplots=True, sharex=False , sharey=False, figsize=(9, 4))
plt.show()
```



Correlation between Features

```
plt.figure(figsize=(6,4))
sns.heatmap(df.corr(),annot=True, cmap="coolwarm")
plt.title ("correlation Graph")
plt.show()
```

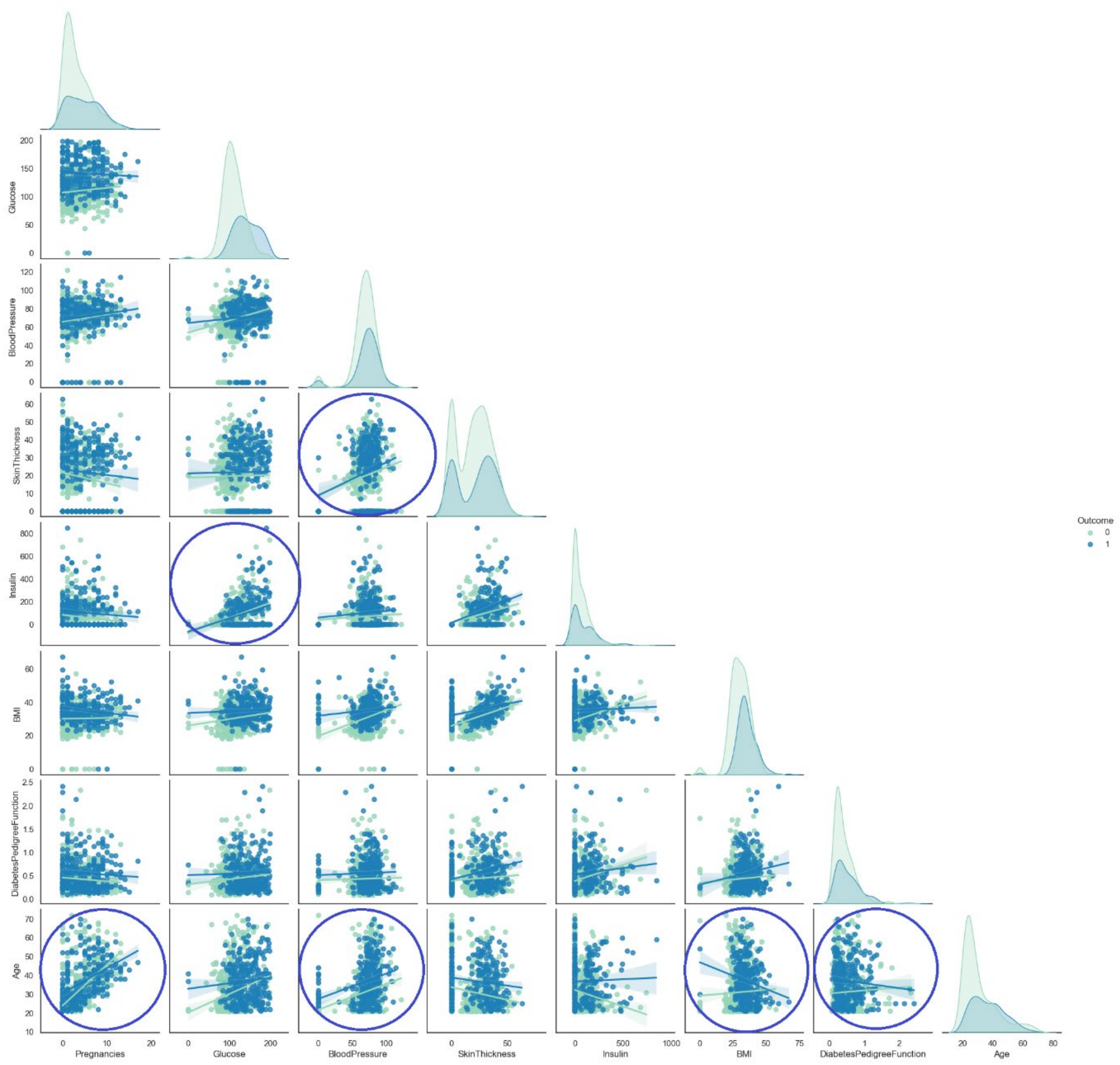


There is no strong correlation between the features themselves.

5. EDA

```
sns.set(font_scale=2)
plt.figure(figsize=(10, 8))
sns.set_style("white")
sns.set_palette("bright")
sns.pairplot(df, kind = 'reg', corner = True, palette ='YlGnBu', hue='Outcome' )
plt.show()
```

```
<Figure size 1000x800 with 0 Axes>
```



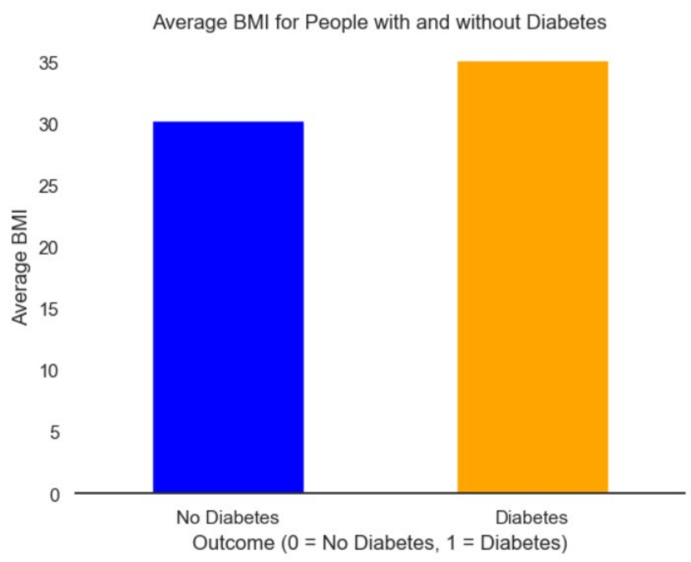
From the chart it is clear that patients who are identified as diabetic have strong correlation between age and number of pregnancies, strong negative correlation between age and BMI, positive correlation between age and pressure and strong positive correlation between Glucose and Insulin level.

Average BMI for people with and without diabetes

```
# Group by Outcome and calculate mean BMI
bmi_avg = df.groupby('Outcome')['BMI'].mean()
print(bmi_avg)
```

```
Outcome
0    30.313026
1    35.144195
Name: BMI, dtype: float64
```

```
# Plotting the average BMI for each outcome group
bmi_avg.plot(kind='bar', color=['blue', 'orange'])
plt.title('Average BMI for People with and without Diabetes')
plt.xlabel('Outcome (0 = No Diabetes, 1 = Diabetes)')
plt.ylabel('Average BMI')
plt.xticks([0, 1], ['No Diabetes', 'Diabetes'], rotation=0)
plt.show()
```



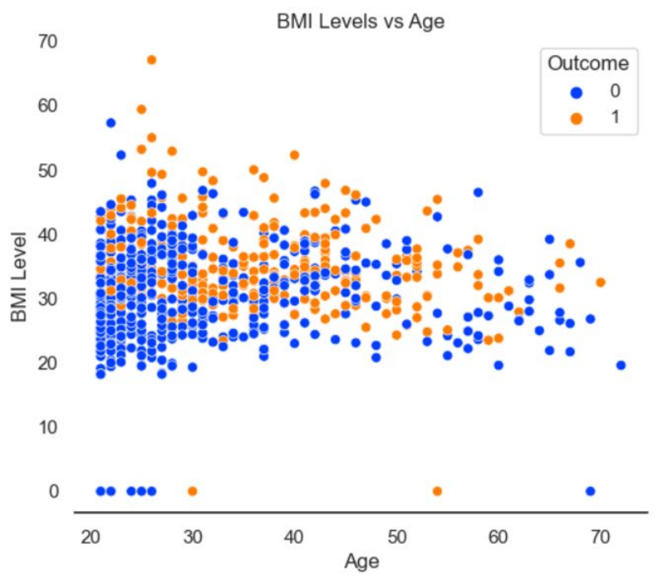
Age group composition

```
bins = [0, 20, 30, 40, 50, 60, 70, 80, 100]
labels = ['0-20', '21-30', '31-40', '41-50', '51-60', '61-70', '71-80', '81+']
df['Age_Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

# Filter data for high BMI (BMI > 30)
high_bmi_age_group = df[df['BMI'] > 30].groupby('Age_Group').size()

# Show the count of people with high BMI in each age group
print(high_bmi_age_group)
```

```
Age_Group
0-20       0
21-30    223
31-40    103
41-50     92
51-60     33
61-70     12
71-80      1
81+        0
```

Age vs. BMI Scatter plot

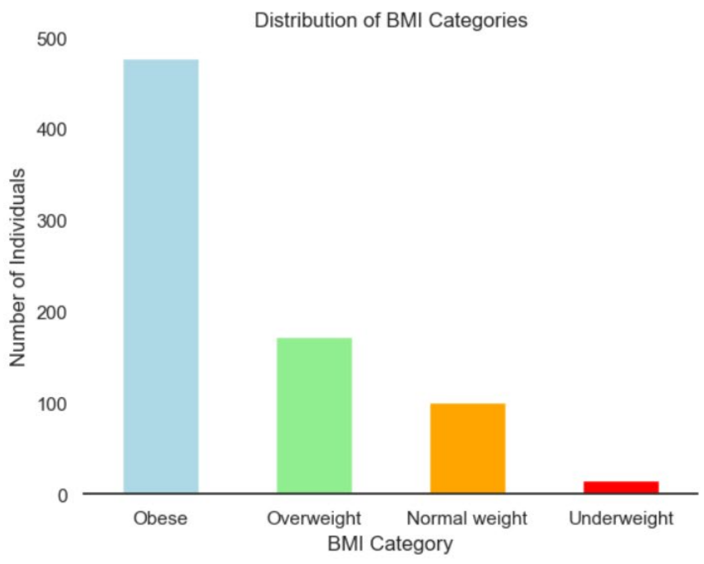
```
# Scatter plot for BMI level (y-axis) and Age (x-axis)
plt.figure(figsize=(6, 5))
sns.scatterplot(x='Age', y='BMI', data=df, color='blue', hue='Outcome')
plt.title('BMI Levels vs Age')
plt.xlabel('Age')
plt.ylabel('BMI Level')
sns.despine(left=True)
plt.show()
```



From the chart it is clear that BMI values are higher for the age group from 20 to 30, but for all the other records, BMI is mixed between 20 and 40.

Weight Category vs. BMI

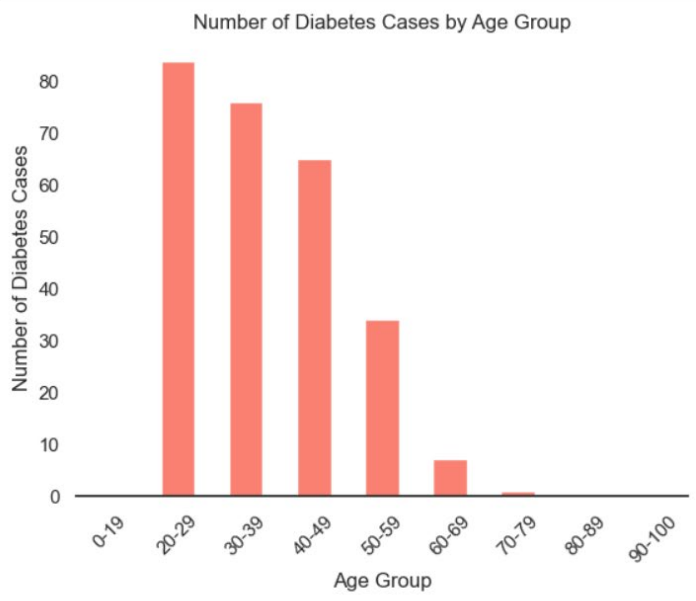
```python
import matplotlib.pyplot as plt

# Categorizing BMI
def categorize_bmi(bmi):
    if bmi < 18.5:
        return 'Underweight'
    elif 18.5 <= bmi < 24.9:
        return 'Normal weight'
    elif 25 <= bmi < 29.9:
        return 'Overweight'
    else:
        return 'Obese'

# Apply categorization
df['BMI_Category'] = df['BMI'].apply(categorize_bmi)

# Plotting the distribution of BMI categories
bmi_category_counts = df['BMI_Category'].value_counts()
bmi_category_counts.plot(kind='bar', color=['lightblue', 'lightgreen', 'orange', 'red'])
plt.title('Distribution of BMI Categories')
plt.xlabel('BMI Category')
plt.ylabel('Number of Individuals')
plt.xticks(rotation=0)
plt.show()
```



by comparing the average BMI for both groups, we can see that being overweight or obese is linked to a higher chance of having diabetes.

Number of diabetes cases by age Group

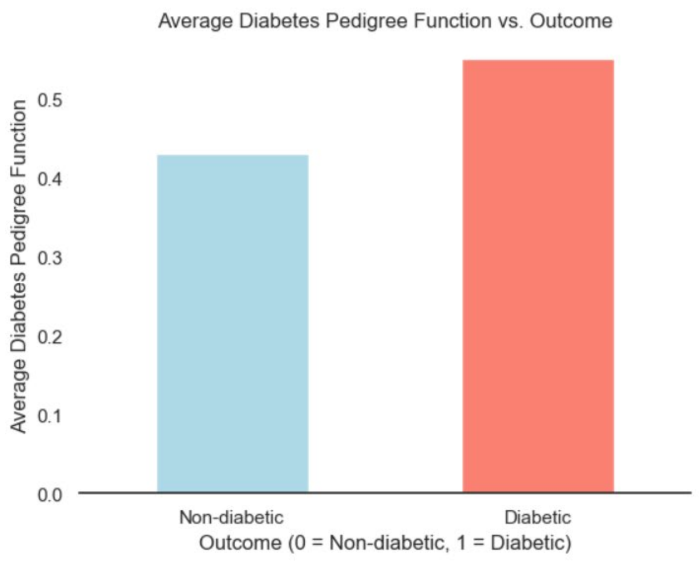
```
import pandas as pd
import matplotlib.pyplot as plt

# Create age groups (e.g., 20-30, 30-40, etc.)
bins = [0, 20, 30, 40, 50, 60, 70, 80, 90, 100]
labels = ['0-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89', '90-100']

# Add a new column 'Age_Group'
df['Age_Group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

# Filter the data to get the count of diabetes cases (Outcome=1) for each age group
age_group_diabetes = df[df['Outcome'] == 1].groupby('Age_Group').size()

# Plotting the number of diabetes cases per age group
age_group_diabetes.plot(kind='bar', color='salmon')
plt.title('Number of Diabetes Cases by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Number of Diabetes Cases')
plt.xticks(rotation=45)
plt.show()
```



The age group 20 to 29 is dominating in terms of number of diabetic outcomes, closely followed by age group from 30 to 39.

Average Diabetes Pedigree Function

```
# Group by Outcome and calculate average Diabetes Pedigree Function
pedigree_vs_outcome = df.groupby('Outcome')['DiabetesPedigreeFunction'].mean()

# Plotting the results
pedigree_vs_outcome.plot(kind='bar', color=['lightblue', 'salmon'])
plt.title('Average Diabetes Pedigree Function vs. Outcome')
plt.xlabel('Outcome (0 = Non-diabetic, 1 = Diabetic)')
plt.ylabel('Average Diabetes Pedigree Function')
plt.xticks([0, 1], ['Non-diabetic', 'Diabetic'], rotation=0)
plt.show()
```



6. Model Training & Evaluation

Imports, Train Test Split, Model Training

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import ConfusionMatrixDisplay, classification_report
```

```
# Prepare the data (features and target)
X = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
        'BMI', 'DiabetesPedigreeFunction', 'Age']]  # Features
y = df['Outcome']  # Target

# Split data into training and testing sets (70% training, 30% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize the Decision Tree classifier
model = DecisionTreeClassifier(random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)
```

Model Evaluation

```
# Evaluate the model
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.75      0.71      0.73       147
           1       0.53      0.59      0.56        83

    accuracy                           0.67       230
   macro avg       0.64      0.65      0.64       230
weighted avg       0.67      0.67      0.67       230
```

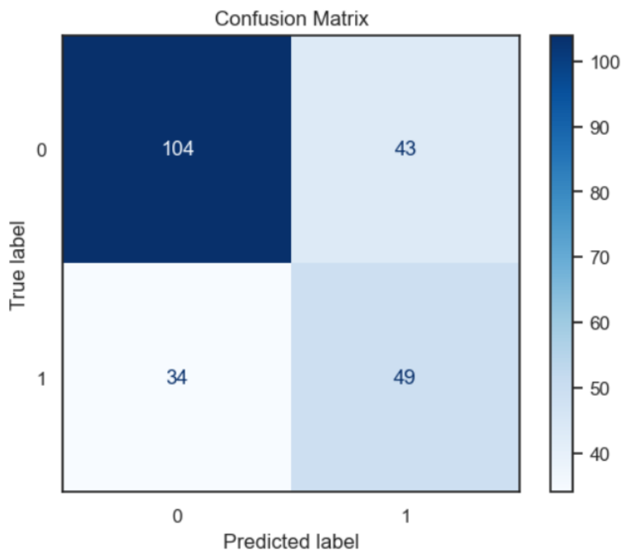
Feature Importance

```
# Get feature importances
importances = model.feature_importances_

# Create a DataFrame with feature names and their importance values
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': importances
})

# Sort features by importance
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Display feature importances
print(feature_importance_df)
```

```
                    Feature  Importance
1                   Glucose    0.419540
6  DiabetesPedigreeFunction    0.173145
5                       BMI    0.123563
0               Pregnancies    0.087519
7                       Age    0.081132
2             BloodPressure    0.062585
3             SkinThickness    0.045517
4                   Insulin    0.006998
```

Confusion Matrix

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, cmap='Blues')
plt.title('Confusion Matrix')
plt.show()
```



Conclusion & Future Suggestions

As part of this project, I have successfully cleaned the data and prepared it for evaluation and model training purposes. Afterward, I have also did exploratory data analysis and found useful insights from the data. I have also trained a machine learning model and evaluated its performance, including the generation of confusion matrix.

There are several aspects that can be further evaluated, such as:

1. Outlier handling for other features
2. Training different machine learning models
3. Feature engineering for development of robust machine learning model.