

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

# Load your dataset
# Replace 'your_dataset.csv' with the actual file path or URL of your dataset
df = pd.read_csv('diabetes.csv').head()

# Pie Chart
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
df['BMI'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90)
plt.title('Pie Chart')

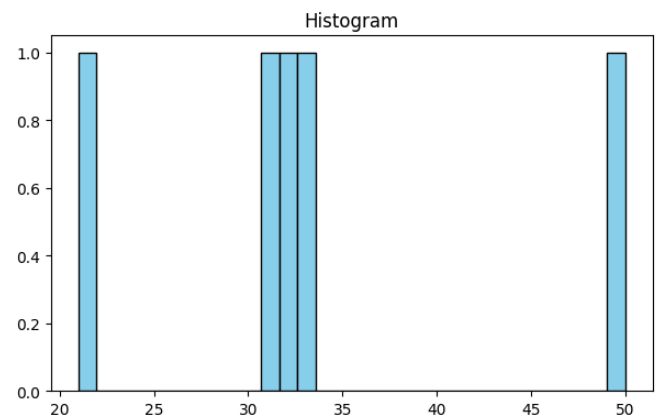
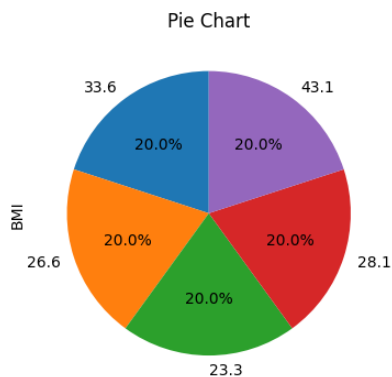
# Histogram
plt.subplot(2, 2, 2)
plt.hist(df['Age'], bins=30, color='skyblue', edgecolor='black')
plt.title('Histogram')

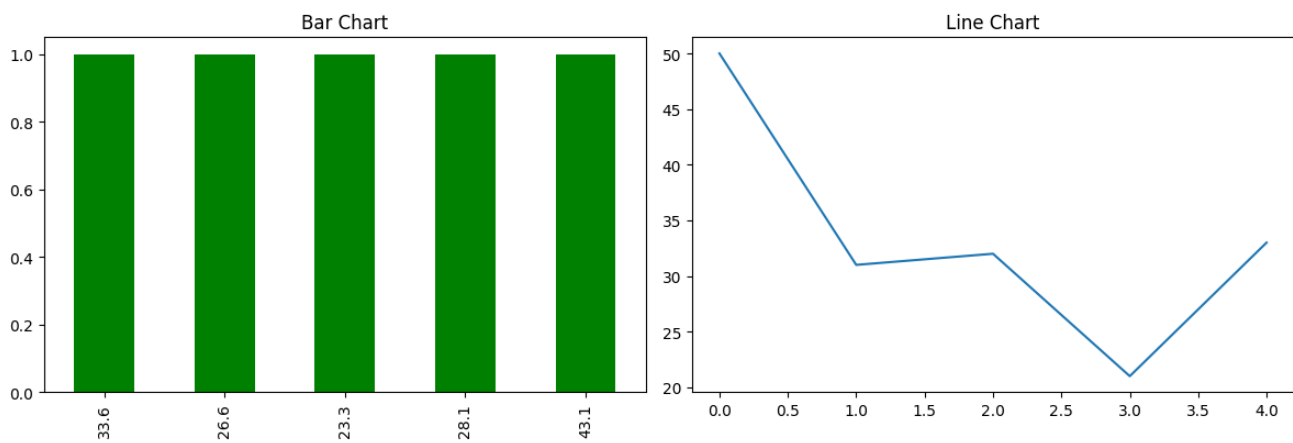
# Bar Chart
plt.subplot(2, 2, 3)
df['BMI'].value_counts().plot(kind='bar', color='green')
plt.title('Bar Chart')

# Line Chart
plt.subplot(2, 2, 4)
plt.plot(df['Age'], label='Line Chart')
plt.title('Line Chart')

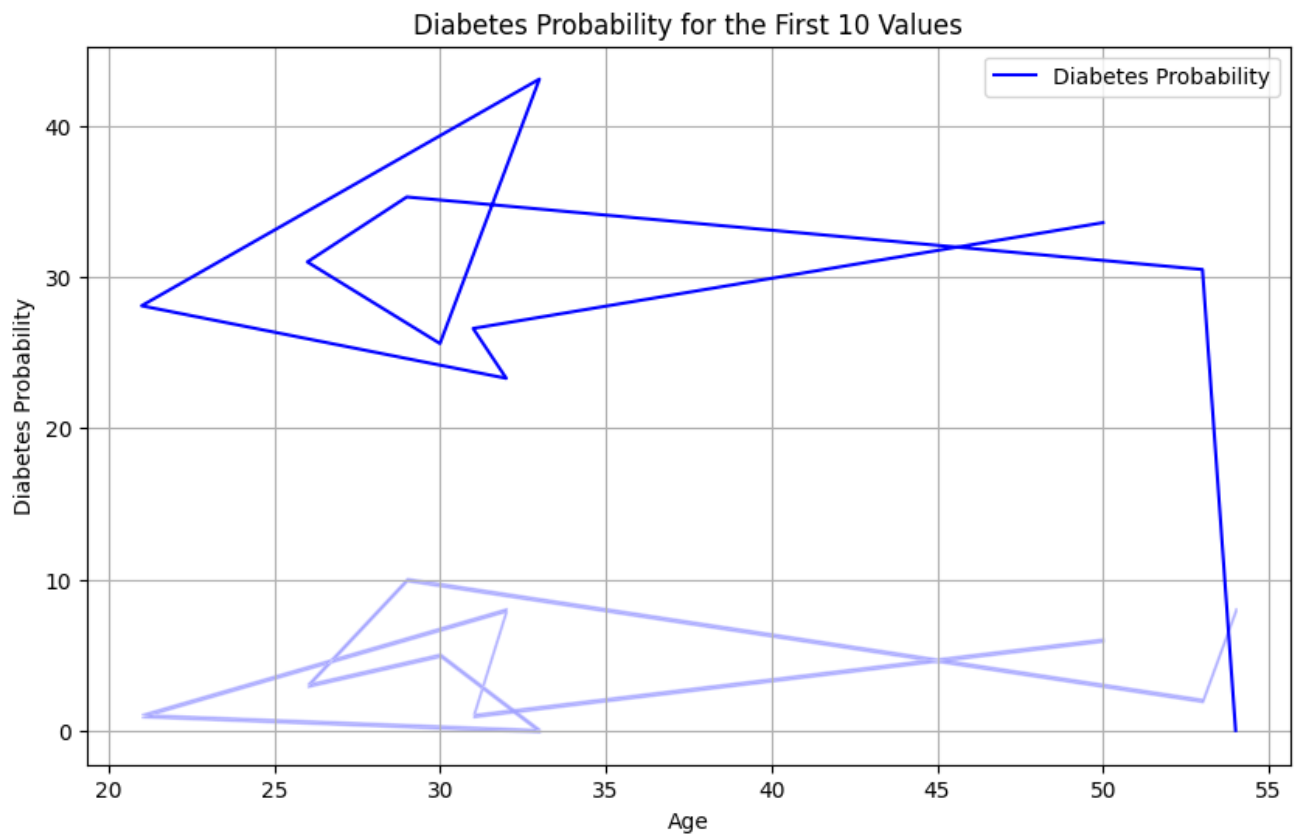
# Adjust layout to prevent overlapping
plt.tight_layout()

# Show the plots
plt.show()
```





```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('diabetes.csv')
# Take only the first 10 values
subset_df = df.head(10)
# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(subset_df['Age'], subset_df['BMI'], label='Diabetes Probability', color='blue')
plt.fill_between(subset_df['Age'], subset_df['Pregnancies'] - 0.1, subset_df['Pregnancies'])
# Adding labels and title
plt.xlabel('Age')
plt.ylabel('Diabetes Probability')
plt.title('Diabetes Probability for the First 10 Values')
# Display the plot
plt.grid(True)
plt.legend()
plt.show()
```



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report
data = pd.read_csv('diabetes.csv')
print("The shape of dataset:")
print(data.shape)
print("Displaying first 5 rows of dataset:")
print(data.head())
print("Describing dataset:")
print(data.describe())
correlation_matrix = data.corr().round(2)
print("The heatmap for dataset:")
sns.heatmap(data=correlation_matrix, annot=True)
plt.show()
```

The shape of dataset:

(768, 9)

Displaying first 5 rows of dataset:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

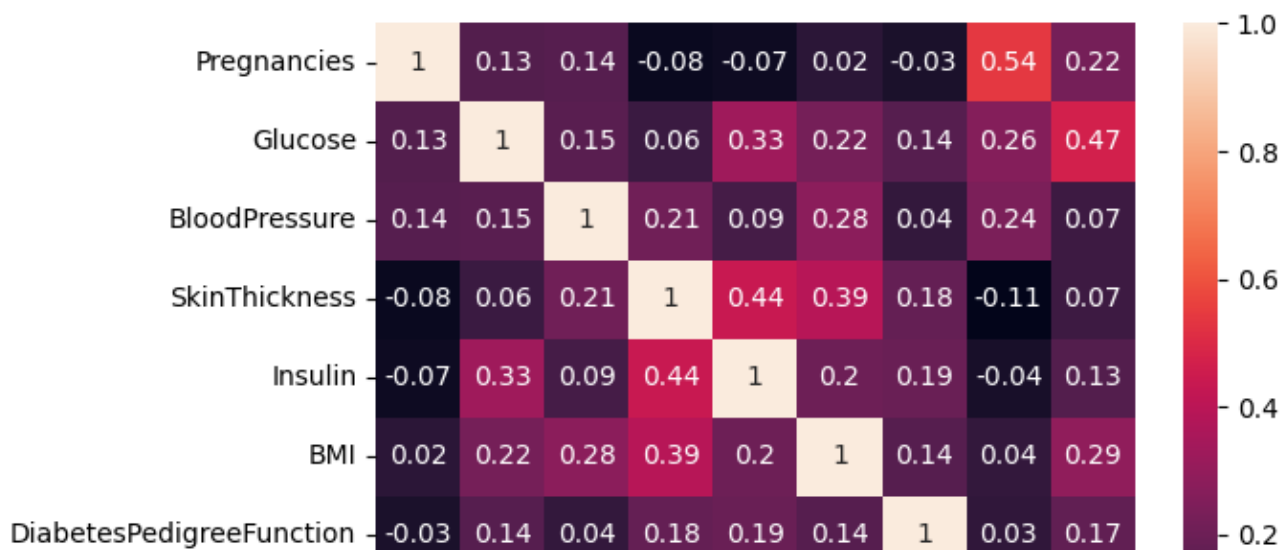
	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
1	0.351	31	0
2	0.672	32	1
3	0.167	21	0
4	2.288	33	1

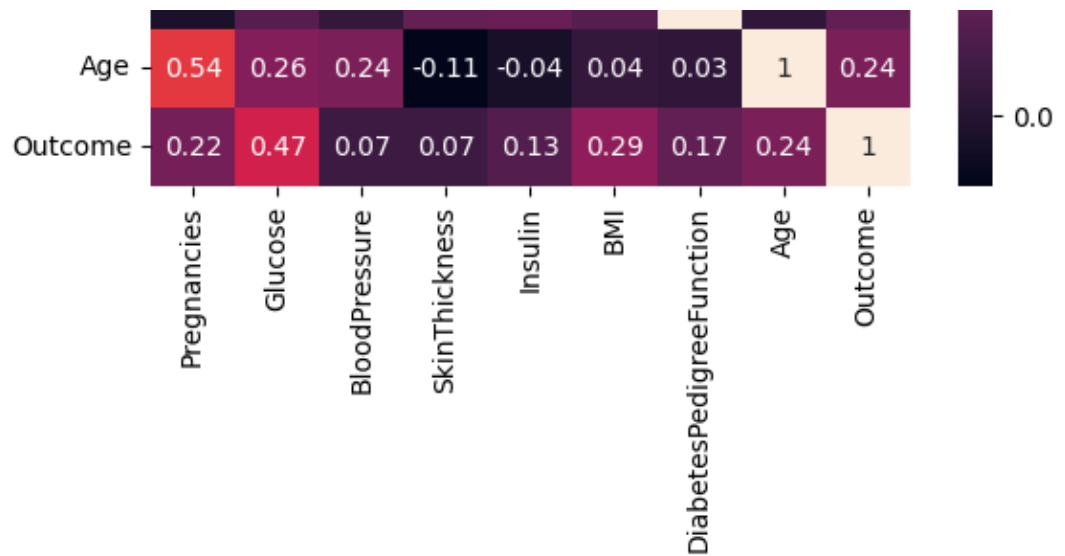
Describing dataset:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	\
count	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	
std	3.369578	31.972618	19.355807	15.952218	115.244002	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	31.992578	0.471876	33.240885	0.348958
std	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.078000	21.000000	0.000000
25%	27.300000	0.243750	24.000000	0.000000
50%	32.000000	0.372500	29.000000	0.000000
75%	36.600000	0.626250	41.000000	1.000000
max	67.100000	2.420000	81.000000	1.000000

The heatmap for dataset:





```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Read the dataset
data = pd.read_csv('diabetes.csv')

# Drop rows with missing values
data = data.dropna()

# Separate features and target variable
X = data.drop('Age', axis=1)
y = data['Age']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize and train the RandomForestRegressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
predictions = model.predict(X_test)

# Evaluate the model
# ... mean_squared_error(X_test, predictions) ...
```

```
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
```

```
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared Score: {r2:.2f}")
```

```
Mean Squared Error: 107.15
R-squared Score: 0.33
```

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

```
# Load diabetes dataset from CSV file
data = pd.read_csv('diabetes.csv')
```

```
# Define age categories
age_bins = [40, 50, 60, 70, 80]
age_labels = ['40-50', '50-60', '60-70', '70-80']
```

```
# Categorize patients into age groups
data['AgeCategory'] = pd.cut(data['Age'], bins=age_bins, labels=age_labels, right=False)
```

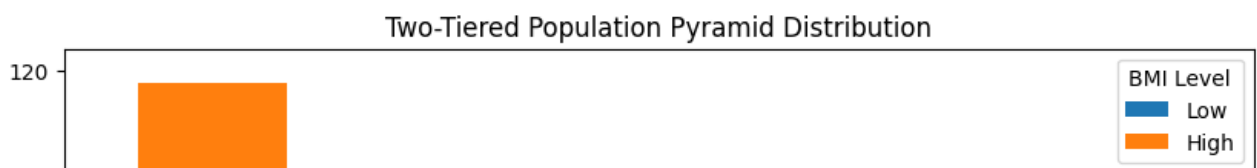
```
# Define BMI threshold
BMI_threshold = 0.06
```

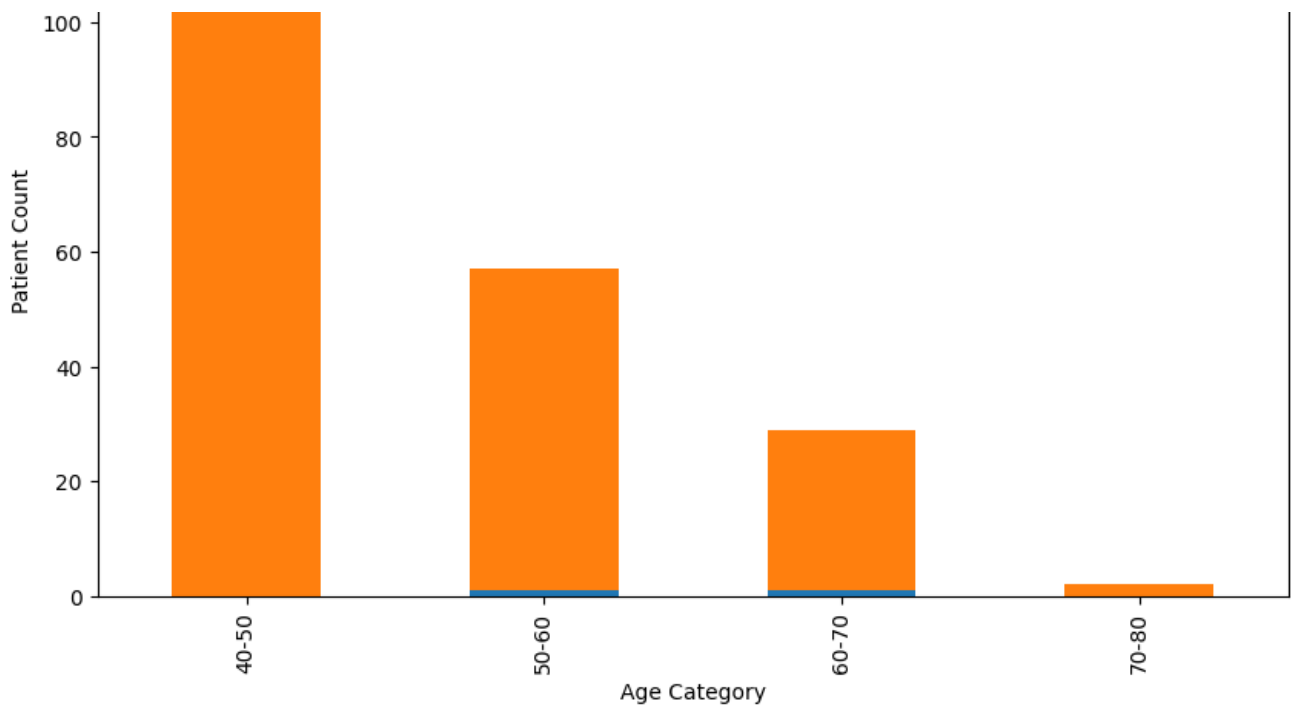
```
# Categorize patients into low/high BMI levels
data['BMILevel'] = pd.cut(data['BMI'], bins=[-float('inf'), BMI_threshold, float('inf')],
```

```
# Count patients for each age category and BMI level
age_BMI_counts = data.groupby(['AgeCategory', 'BMILevel']).size().unstack(fill_value=0)
```

```
# Count patients with complications for each age category, BMI level, and complication ty
complication_counts = data[data['BloodPressure'] != 'No Complication'].groupby(['AgeCateg
```

```
# Plot stacked bar-population pyramid graph
fig, ax = plt.subplots(figsize=(10, 6))
age_BMI_counts.plot(kind='bar', stacked=True, ax=ax)
ax.set_title('Two-Tiered Population Pyramid Distribution')
ax.set_xlabel('Age Category')
ax.set_ylabel('Patient Count')
plt.legend(title='BMI Level', loc='upper right')
plt.show()
```





```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the diabetes dataset from a CSV file
diabetes_df = pd.read_csv('/content/diabetes.csv')

# Take the first 100 rows of the dataset
diabetes_df_first_100 = diabetes_df.head(100)

# Drop rows with missing values
diabetes_df_cleaned = diabetes_df_first_100.dropna()

# Alternatively, you can fill missing values with the mean
# diabetes_df_cleaned = diabetes_df_first_100.fillna(diabetes_df_first_100.mean())

# Split the cleaned dataset into features (X) and target variable (y)
X = diabetes_df_cleaned.drop('BMI', axis=1)
y = diabetes_df_cleaned['BMI']
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a linear regression model
model = LinearRegression()

# Train the model on the training set
model.fit(X_train, y_train)

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

# Evaluate the model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Plot the predictions against the actual values with different markers and colors
plt.scatter(y_test, y_pred, marker='o', color='blue', label='Actual vs. Predicted')
plt.scatter(y_test, y_test, marker='x', color='red', label='Actual vs. Actual')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Linear Regression on First 100 Rows of Diabetes Dataset')
plt.legend()
plt.show()
```

Mean Squared Error: 47.48391412498061

R-squared: 0.07359486724544784

