Title: Analyzing Symptoms for Early Diabetes Detection

Authors: Angela Cheng, Crystal Diaz, Janet Pham, Laurand Osmeni

Introduction

- Why was the project undertaken?

  > Our team undertook this project to address the need for data science and visualization techniques in improving diabetes detection and prediction by leveraging data science and visualization techniques. According to American Diabetes Association approximately 37.3 million Americans affected by diabetes, which accounts for 11.3% of the population, and around 1.9 million individuals living with type 1 diabetes, including about 244,000 children and adolescents, the significance of effective disease management cannot be overstated. By providing valuable insights through analytical methods and visualizations, our goal was to empower healthcare professionals to improve outcomes for individuals affected by diabetes.

- What was the research question, the tested hypothesis or the purpose of the research?

  > How can data science and visualization techniques support the early detection and prediction of diabetes?

  > From the dataset and accompanying plots, potential hypotheses arise regarding factors that may contribute to an increased likelihood of diabetes. These hypotheses involve the influence of age, certain symptoms as polyuria and polydipsia, obesity, gender, and specific symptoms like visual blurring and delayed healing on the prevalence of diabetes.

Selection of Data

- What is the source of the dataset? Characteristics of data?

  > The dataset was obtained from the reputable UCI website, a reliable source for machine learning datasets. It serves the purpose of predicting the risk of early stage diabetes. The dataset contains 520 instances of newly diagnosed diabetic individuals, with 17 attributes describing diabetes related signs and symptoms. It is specifically designed for early stage

diabetes risk prediction.

Dataset
https://raw.githubusercontent.com/Jamham1020/CST383_Project/main/diabetes_data_upload.csv

Source: Early stage diabetes risk prediction dataset.. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5VG8H.

- Any munging or feature engineering?

  Yes, to prepare the dataset for analysis, we performed data cleaning and feature engineering. This involved handling missing values, removing duplicates, and fixing inconsistencies. We also created new features and transformed existing ones to better capture relevant information. These steps improved the quality and usefulness of the data for our analysis.

Methods.

- What materials/APIs/tools were used or who was included in answering the research question?

  Methods and materials we used for data analysis libraries were Pandas and NumPy, along with machine learning classification decission tree, for building and training predictive models. Visualization tools like Matplotlib and Seaborn were used to present the findings and visualize the data.

  Collaborative platforms such as Slack, Colab, GitHub and Jupyter Notebook were employed to facilitate communication among team members and share code. Everyone on the team participated in answering the research question, providing input, suggestions, and making improvements to the project.

  Also, we searched and looked at the valuable insights from healthcare professionals through the clinician professional website, https://diabetes.org. Their expertise helped us gain domain specific knowledge on aspects related to diabetes detection and management.

  Source: Statistics About Diabetes | ADA. (n.d.). https://diabetes.org/about-us/statistics/about-diabetes

Results

- What answer was found to the research question; what did the study find? Was the tested hypothesis true? Any visualizations?

  > Our study found that data science and visualization techniques effectively support the early detection and prediction of diabetes. Key factors such as age, gender, and symptoms like polyuria and polydipsia were identified as significant predictors of diabetes. Visualizations including bar plots, violin plots, scatter plots, and confusion matrices provided valuable insights into the distribution of diabetes cases and the performance of predictive models.

  > Yes, the tested hypothesis were true. The hypothesis that data science and visualization techniques can support the early detection and prediction of diabetes was supported by the study's findings.

  > The study utilized various visualizations, such as bar plots, violin plots, scatter plots, and confusion matrices, to enhance understanding of patterns, correlations, and predictive models in diabetes detection and prediction.

Discussion

- What might the answer imply and why does it matter? How does it fit in with what other researchers have found? What are the perspectives for future research? Survey about the tools investigated for this assignment.

  > The implications of the findings suggest that age, gender, and specific symptoms such as polyuria and polydipsia play important roles in predicting the likelihood of developing diabetes. Understanding these factors can help healthcare professionals identify individuals at higher risk and implement preventive measures or early interventions.

  > The research on diabetes risk factors, age is a well established factor, with older individuals being more susceptible to developing diabetes. Gender differences in diabetes prevalence have also been documented, with some studies indicating a higher risk for males. Also, symptoms like polyuria and polydipsia are recognized as common early signs of diabetes.

The future research perspectives could focus on exploring the interplay between different risk factors and their impact on diabetes prediction. Investigating additional variables, lifestyle factors, and genetic markers, could enhance the accuracy of predictive models.

The tools investigated in this assignment, they encompassed various analytical techniques commonly used in data exploration and machine learning. These tools included data visualization techniques of histograms, box plots, scatter plots, graphviz etc.. correlation analysis, classification algorithms Decision Tree, and evaluation metrics of accuracy, confusion matrix, ROC curve.

Summary

- Most important findings.

  The study found that age, gender, and specific symptoms such as frequent urination and increased thirst are important factors in the early detection and prediction of diabetes. Visualizations, including bar plots, violin plots, and scatter plots, were used to illustrate the distribution of diabetes cases and correlations between variables. These findings highlight the value of data science and visualization techniques in helping the understanding and management of diabetes.

Presentation video with a demo addressing the rubric

https://www.youtube.com/watch?v=Y5HTCSYkSCs

Project on GitHub

https://github.com/Jamham1020/CST383_Project

References/Bibliography

Statistics About Diabetes | ADA. (n.d.). https://diabetes.org/about-us/statistics/about-diabetes

Early stage diabetes risk prediction dataset.. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5VG8H.

```python
# -*- coding: utf-8 -*-
"""
X23 Group 1 - Final Project CST383
"""
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import graphviz
from matplotlib import rcParams
from PIL import Image
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_sc
from sklearn.model_selection import cross_val_score, GridSearchCV, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.preprocessing import LabelEncoder



df = pd.read_csv("https://raw.githubusercontent.com/Jamham1020/CST383_Project/main/
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Age                520 non-null    int64
 1   Gender             520 non-null    object
 2   Polyuria           520 non-null    object
 3   Polydipsia         520 non-null    object
 4   sudden weight loss 520 non-null    object
 5   weakness           520 non-null    object
 6   Polyphagia         520 non-null    object
 7   Genital thrush     520 non-null    object
 8   visual blurring    520 non-null    object
 9   Itching            520 non-null    object
 10  Irritability       520 non-null    object
 11  delayed healing    520 non-null    object
 12  partial paresis    520 non-null    object
 13  muscle stiffness   520 non-null    object
 14  Alopecia           520 non-null    object
 15  Obesity            520 non-null    object
 16  class              520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

```python
# Number of columns and rows of instances and attributes.
df.shape
```

```
(520, 17)
```

```python
# Check for missing values
df.isna().sum()
```

```
Age                   0
Gender                0
Polyuria              0
Polydipsia            0
sudden weight loss    0
weakness              0
Polyphagia            0
Genital thrush        0
visual blurring       0
Itching               0
Irritability          0
delayed healing       0
partial paresis       0
muscle stiffness      0
Alopecia              0
Obesity               0
class                 0
dtype: int64
```

```python
# Encode the categorical variables
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df['Polyuria'] = label_encoder.fit_transform(df['Polyuria'])
df['Polydipsia'] = label_encoder.fit_transform(df['Polydipsia'])
df['sudden weight loss'] = label_encoder.fit_transform(df['sudden weight loss'])
df['weakness'] = label_encoder.fit_transform(df['weakness'])
df['Polyphagia'] = label_encoder.fit_transform(df['Polyphagia'])
df['Genital thrush'] = label_encoder.fit_transform(df['Genital thrush'])
df['visual blurring'] = label_encoder.fit_transform(df['visual blurring'])
df['Itching'] = label_encoder.fit_transform(df['Itching'])
df['Irritability'] = label_encoder.fit_transform(df['Irritability'])
df['delayed healing'] = label_encoder.fit_transform(df['delayed healing'])
df['partial paresis'] = label_encoder.fit_transform(df['partial paresis'])
df['muscle stiffness'] = label_encoder.fit_transform(df['muscle stiffness'])
df['Alopecia'] = label_encoder.fit_transform(df['Alopecia'])
df['Obesity'] = label_encoder.fit_transform(df['Obesity'])
df['class'] = label_encoder.fit_transform(df['class'])
```

```
# Display the first few rows of the df.
df.head()
```

|	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	v blu
0	40	1	0	1	0	1	0	0	
1	58	1	0	0	0	1	0	0	
2	41	1	1	0	0	1	1	0	
3	45	1	0	0	1	1	1	1	
4	60	1	1	1	1	1	1	0	

```
# Display descriptive statistics of the df
df.describe()
```

|	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polypha
count	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000	520.000
mean	48.028846	0.630769	0.496154	0.448077	0.417308	0.586538	0.45!
std	12.151466	0.483061	0.500467	0.497776	0.493589	0.492928	0.498
min	16.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	39.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	47.500000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000
75%	57.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
max	90.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000

```
# Display a histogram of the age column in the df.
sns.histplot(df.Age)
plt.show()
```
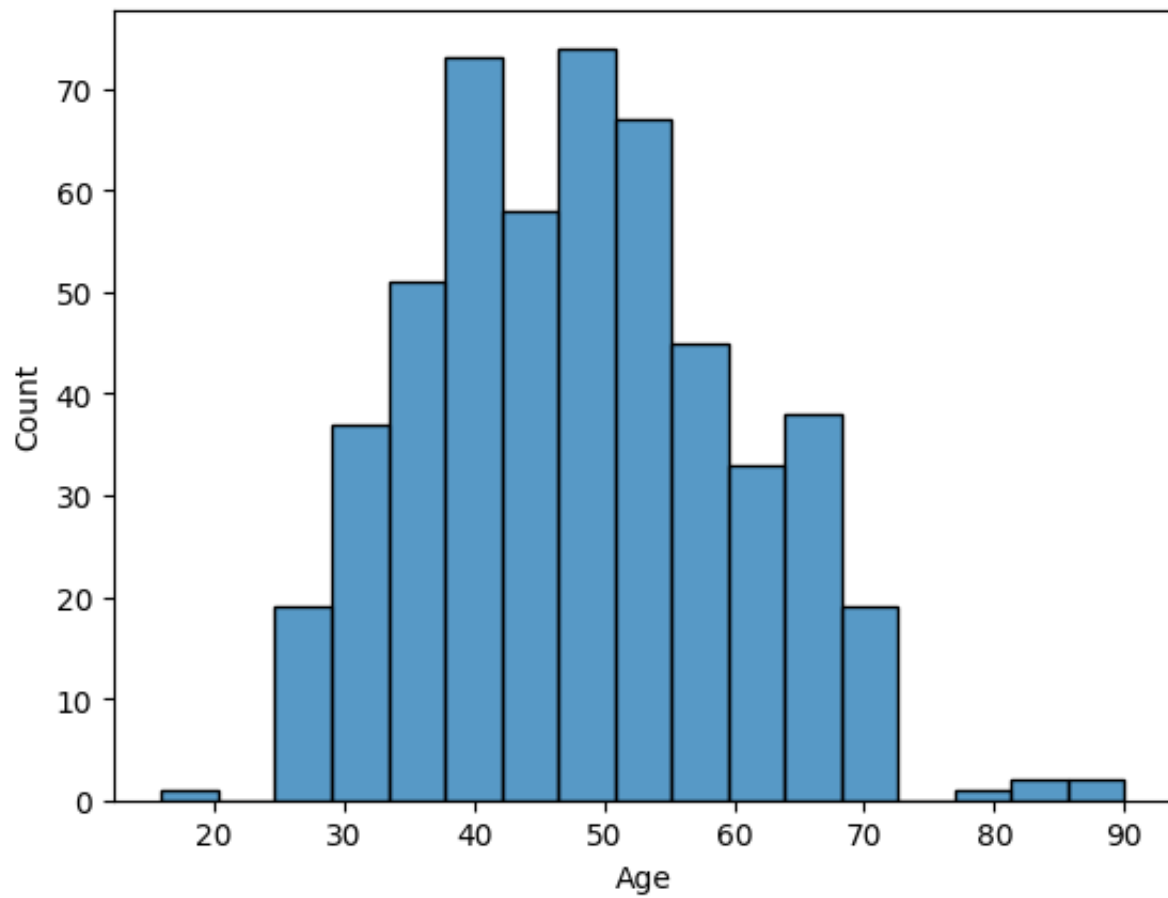
```
# Plot the grouped bar plot, group the data by Gender to calculate the counts.
gender_counts = df.groupby(['Gender', 'class']).size().unstack()
plt.figure(figsize=(8, 6))
gender_counts.plot(kind='bar', width=0.4)
plt.xlabel("Gender")
plt.ylabel("Count")
plt.title("Distribution of Positive and Negative Cases by Gender")
plt.legend(title="Class", labels=["Negative", "Positive"])
plt.xticks([0, 1], ["Male", "Female"], rotation=0)
plt.show()
```

<Figure size 800x600 with 0 Axes>

Distribution of Positive and Negative Cases by Gender
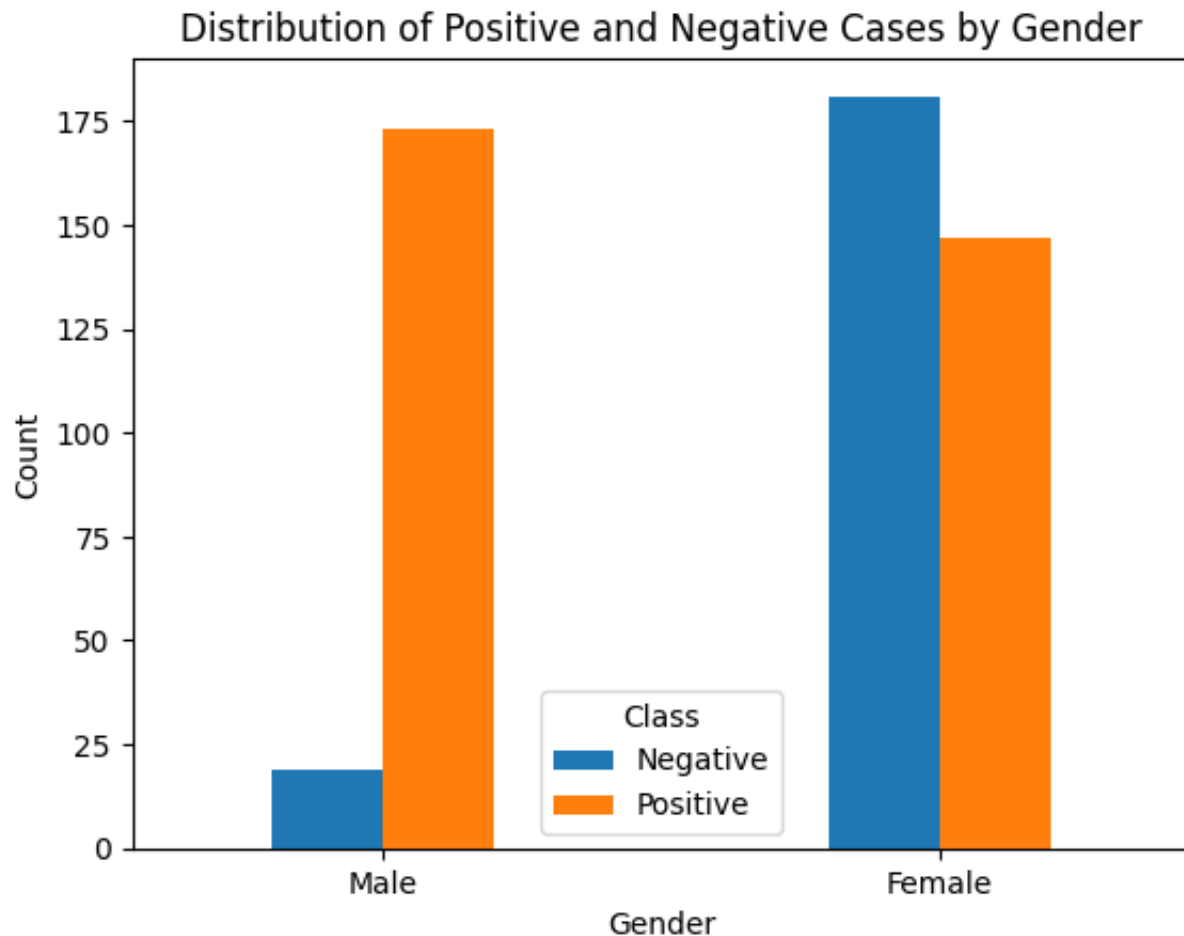
```python
# Define diabetics by age and gender.
gender = {0: "Male", 1: "Female"}
age = df.groupby(pd.cut(df['Age'], bins=[0, 30, 60, 100])).mean()['class']
gender_diabetic = df.groupby('Gender').mean()['class']
color_age = ['skyblue', 'orange', 'green']
color_gender = ['skyblue', 'orange']
fig, axes = plt.subplots(1, 2, figsize=(10, 4))
axes[0].bar(age.index.astype(str), age.values, color=color_age)
axes[0].set_xlabel("Age Group")
axes[0].set_ylabel("Proportion of Diabetics")
axes[0].set_title("Diabetics within Age")
gender_labels = [gender[gender_value] for gender_value in gender_diabetic.index]
axes[1].bar(gender_labels, gender_diabetic.values, color=color_gender)
axes[1].set_xlabel("Gender")
axes[1].set_ylabel("Proportion of Diabetics")
axes[1].set_title("Diabetics within Gender")
plt.tight_layout()
plt.show()
```
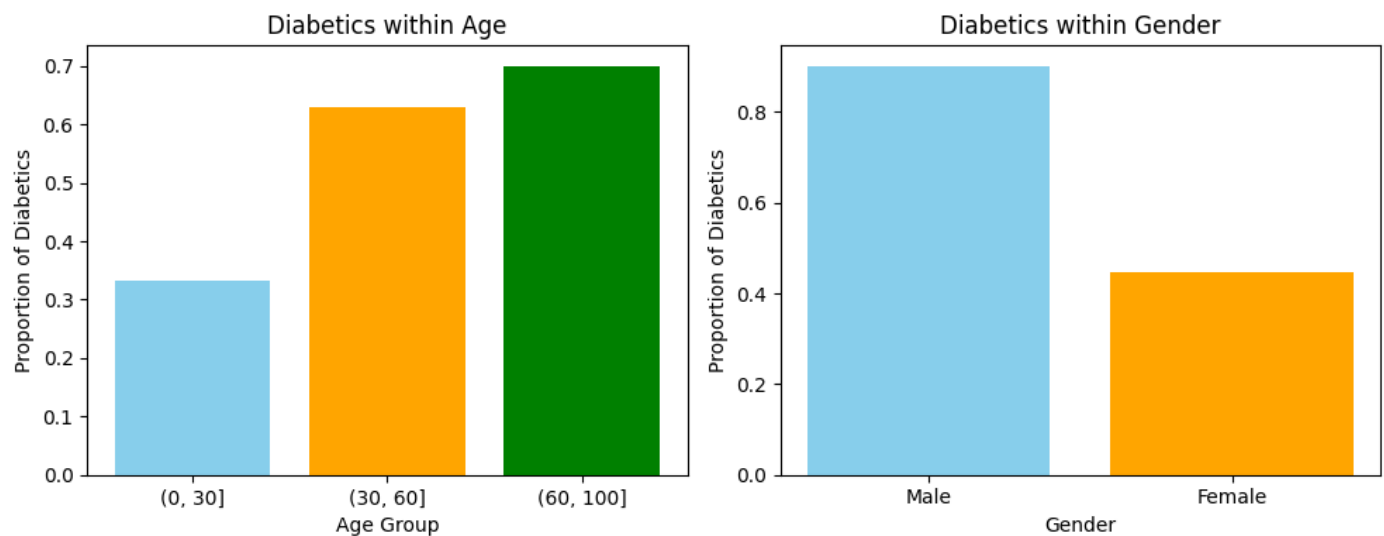
```
# Plot violin plot of polyuria by gender.
plt.figure(figsize=(8, 6))
sns.violinplot(x='Polyuria', y='Age', hue='Gender', data=df)
plt.xlabel("Polyuria")
plt.ylabel("Age")
plt.title("Violin Plot of Polyuria by Gender")
plt.legend(title="Gender", loc="upper right")
plt.show()
```
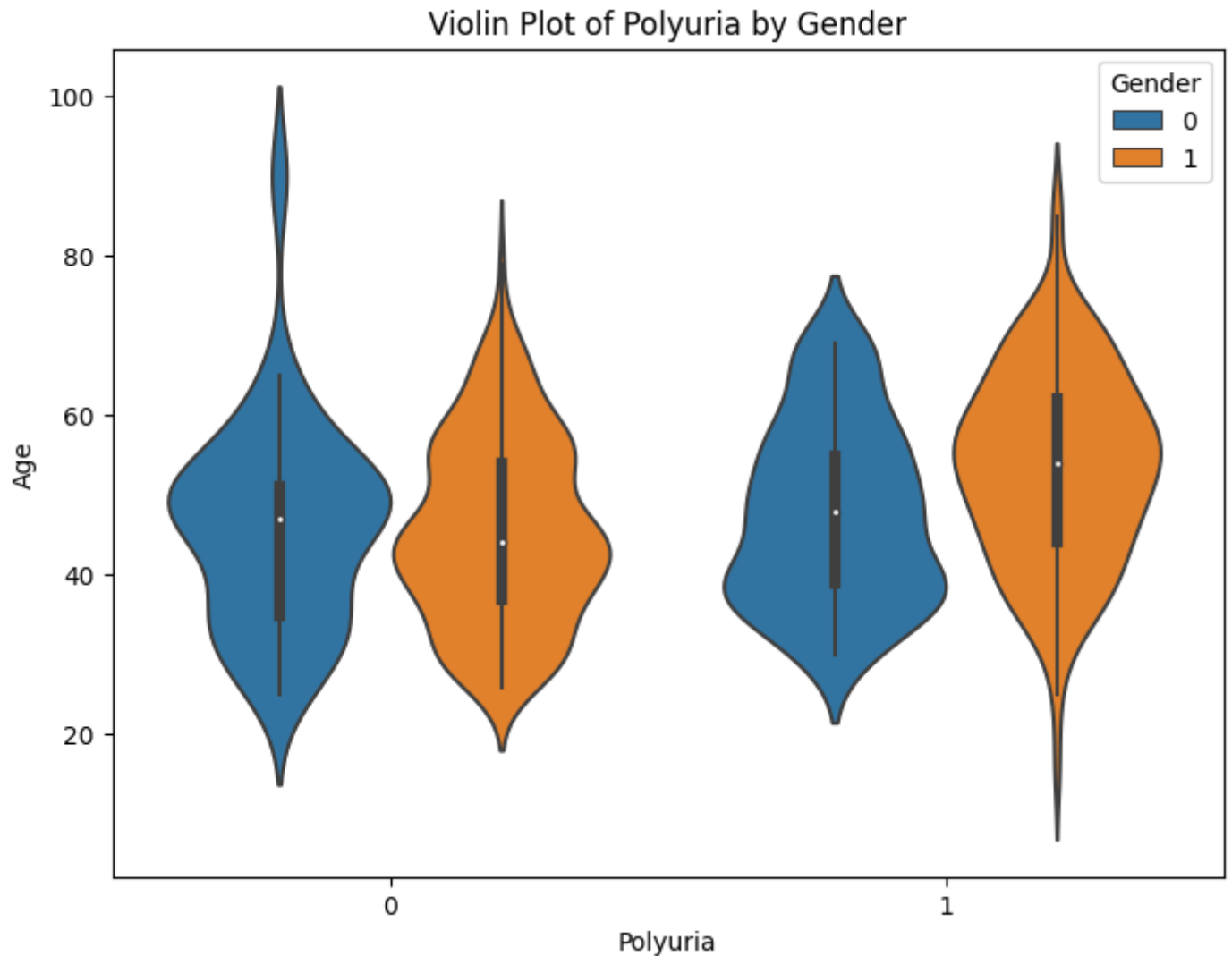


```
# Analyze the correlation between variables
corr = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr, linewidths=0.25, cmap='YlGnBu', annot=False)
plt.title("Correlation Heatmap")
plt.show()
```
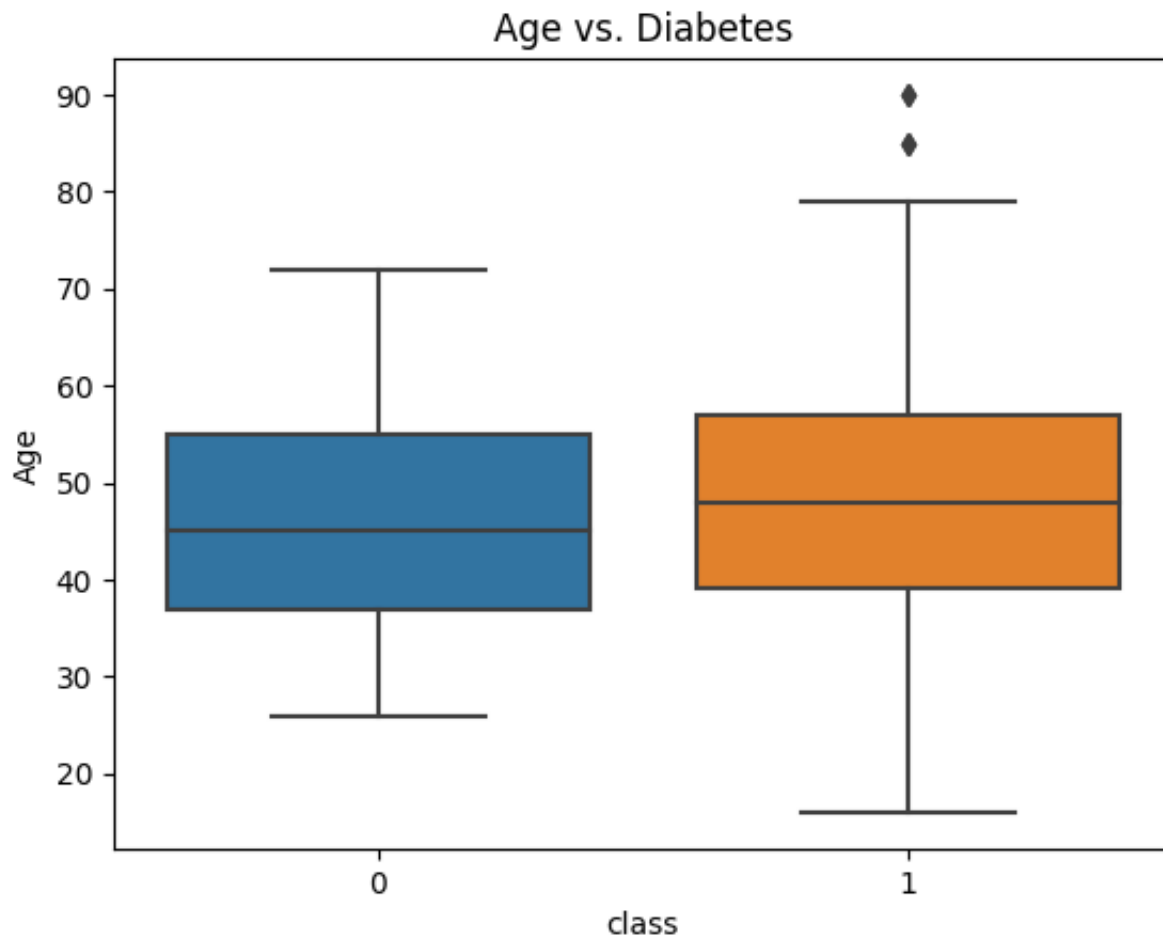
Correlation Heatmap
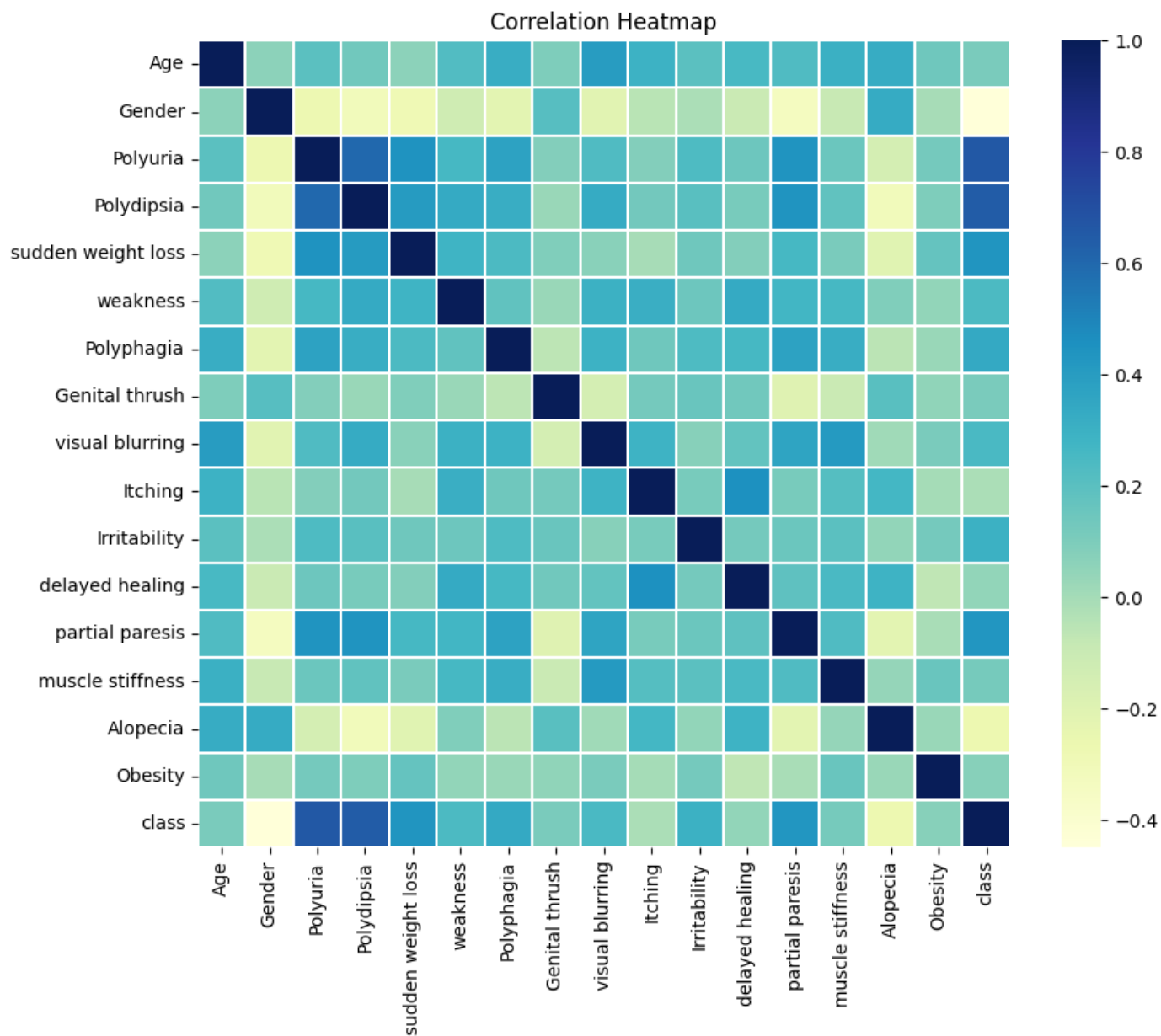
```
# Check the distribution of the target variable
sns.countplot(x = 'class', data = df)
plt.title('Distribution of Diabetes Cases')
plt.show()
```


Distribution of Diabetes Cases
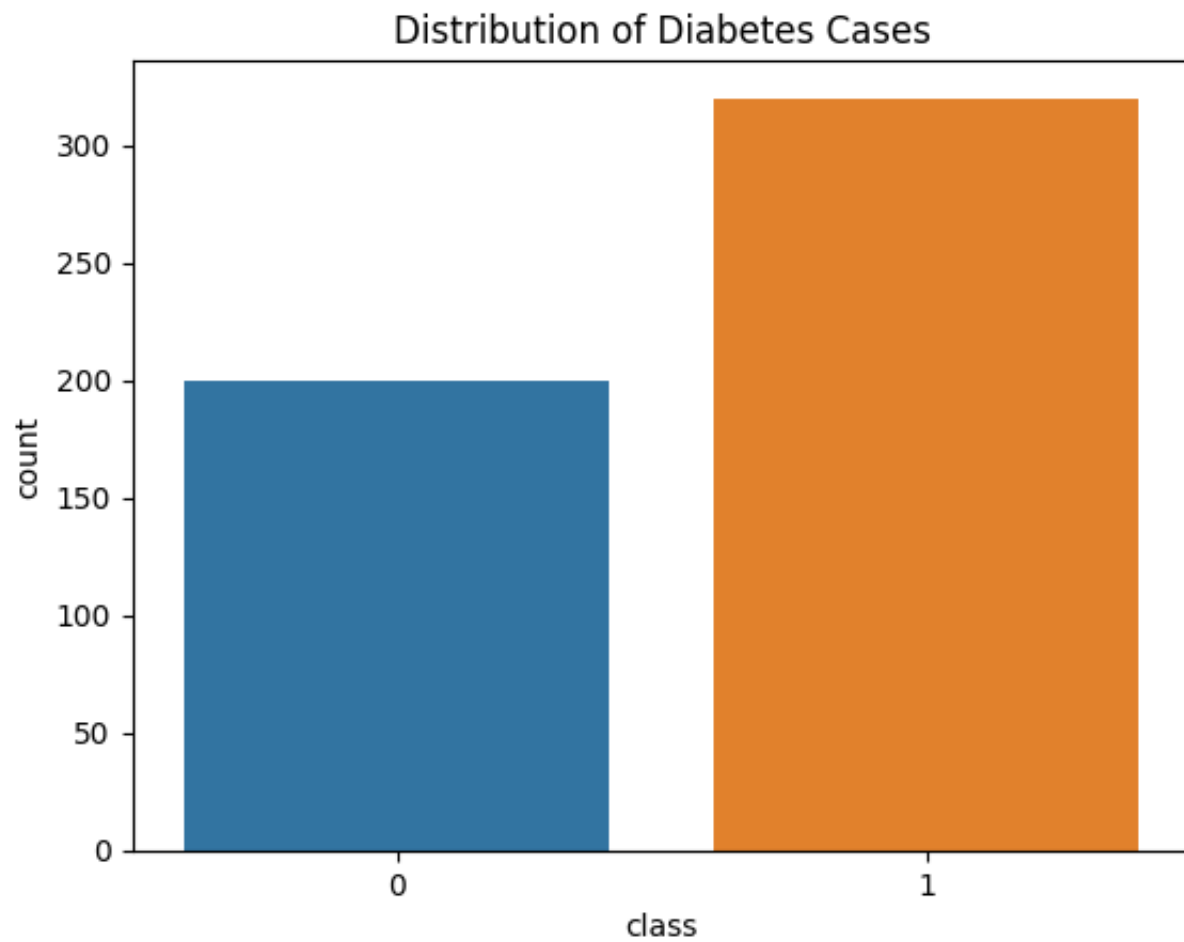
```
# Boxplot to visualize age vs diabetics.
sns.boxplot(x = 'class', y = 'Age', data = df)
plt.title('Age vs. Diabetes')
plt.show()
```



```
# split the dataset into training and testing sets
X = df.drop('class', axis = 1)
y = df['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# train a Random Forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# make predictions on the test set
y_pred = clf.predict(X_test)
```

```python
# Calculate accuracy and display confusion matrix
accuracy = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
print("Accuracy: ", accuracy)
print("Confusion Matrix: ")
print(confusion_mat)
categories = ['True Negative', 'False Positive', 'False Negative', 'True Positive']
labels = ['TN', 'FP', 'FN', 'TP']
colors = ['lightblue', 'lightgreen', 'salmon', 'gold']
plt.figure(figsize=(6, 4))
plt.bar(categories, confusion_mat.ravel(), color=colors)
plt.xlabel('Categories')
plt.ylabel('Count')
plt.title('Confusion Matrix')
for i in range(len(categories)):
    plt.text(i, confusion_mat.ravel()[i], labels[i], ha='center', va='bottom')
plt.show()
```

```
Accuracy:  0.9903846153846154
Confusion Matrix:
[[33  0]
 [ 1 70]]
```
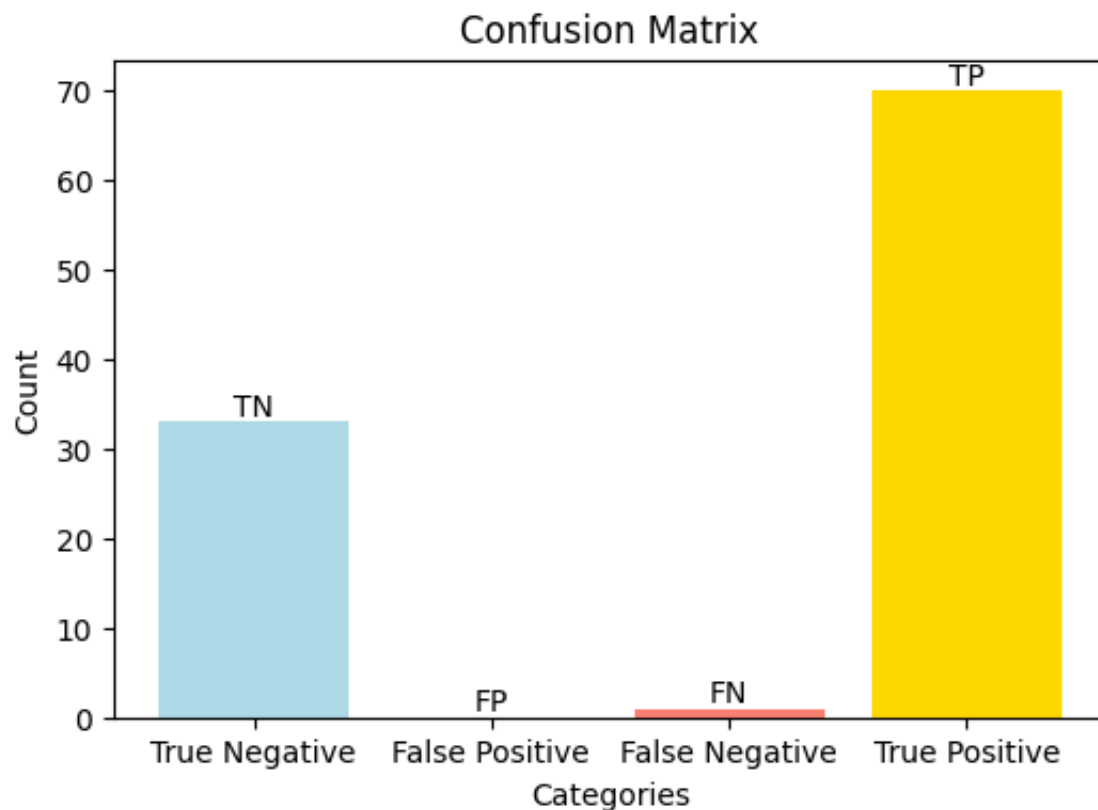


```python
# The pair plot using age, gender, polyuria, and polydipsia to visualize insights i
vars = ['Age', 'Gender', 'Polyuria', 'Polydipsia']
df_var = df[vars]
```

```python
g = sns.pairplot(df_var, hue='Polyuria')
g.axes[0, 0].set_ylabel('Age')
g.axes[1, 0].set_xlabel('Gender')
g.axes[1, 0].set_ylabel('Age')
g.axes[1, 1].set_xlabel('Polyuria')
g.fig.suptitle('Pair Plot of Age, Gender, Polyuria, and Polydipsia', y=1.02)
plt.show()
```

Pair Plot of Age, Gender, Polyuria, and Polydipsia
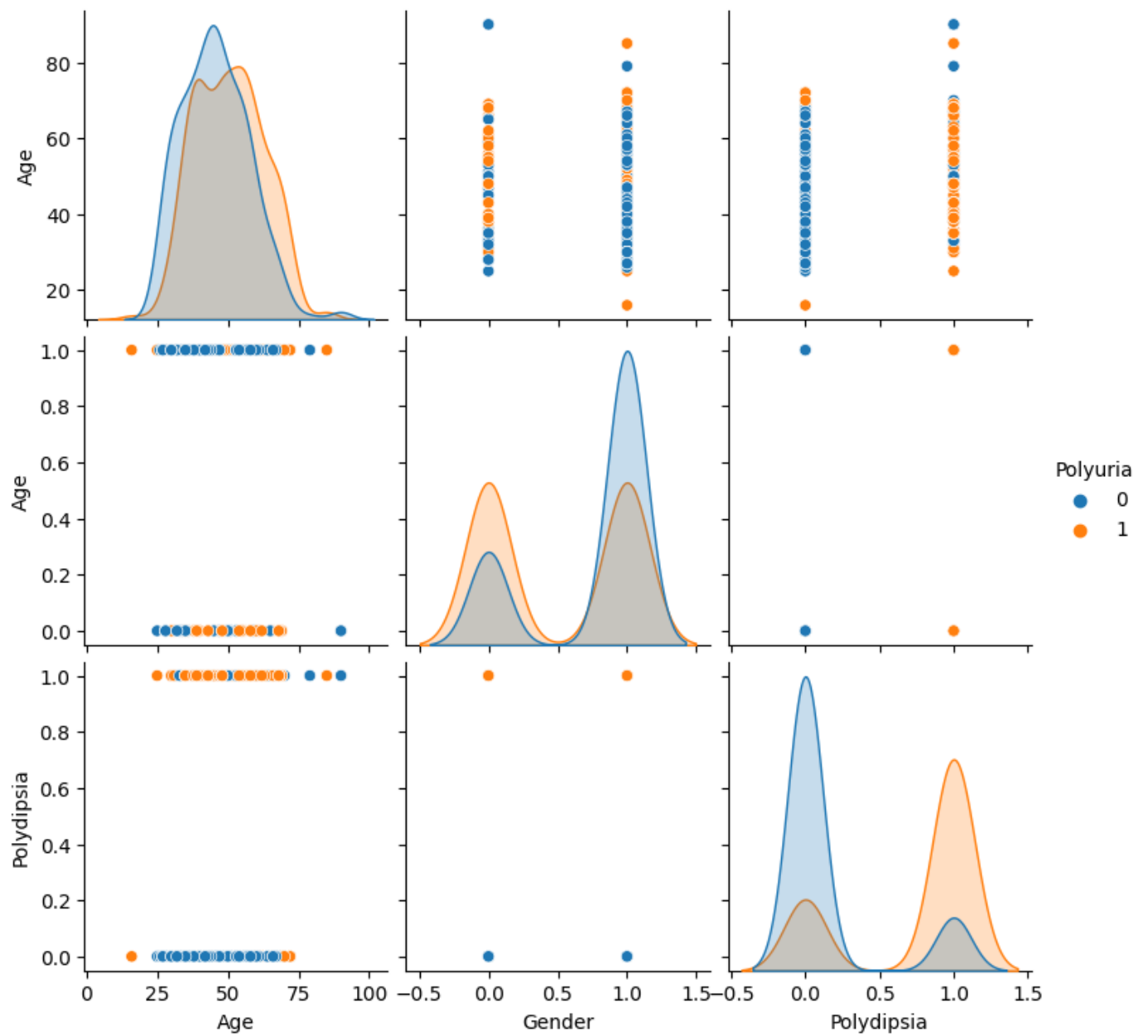
```python
# calculate the probabilities for positive class predictions
y_pred_prob = clf.predict_proba(X_test)[:, 1]

# calculate the ROC AUC (Receiver Operating Area Under Curve) score
# ROC AUC score is a measure of how well the model distinguishes between positive a
# Higher score indicates a better predictive ability.
# AOC of 0.5 suggests no discrimination (ability to diagnose patients with and with
# AOC of 0.7 - 0.8 is considered acceptable, 0.8 - 0.9 is excellent, and more than
auc_score = roc_auc_score(y_test, y_pred_prob)
print("ROC AUC Score: ", auc_score)
```
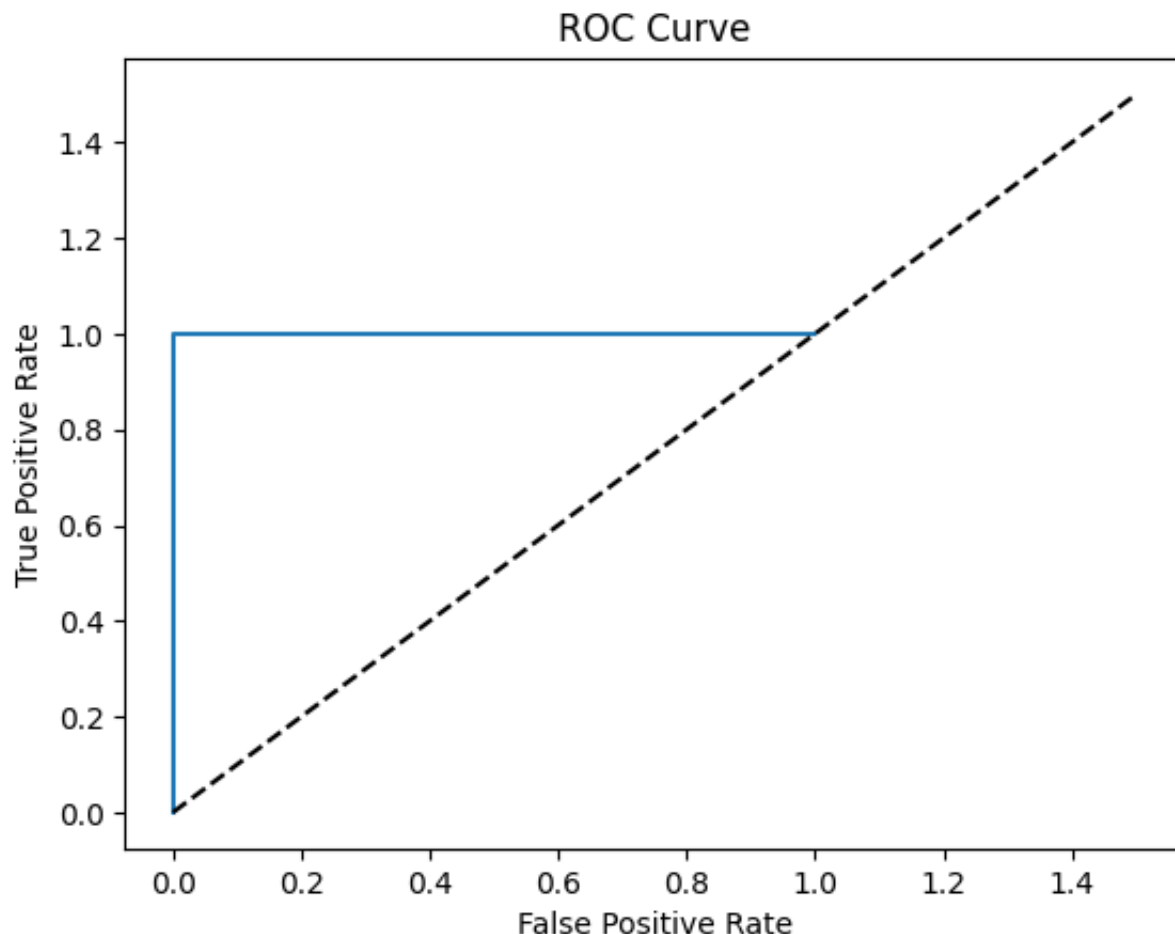
    ROC AUC Score: 1.0

```python
# creat ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot(fpr, tpr)
plt.plot([0, 1.5], [0, 1.5], 'k--') #random guessing curve
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```

```
# Swarm plot of Age vs Polyuria.
plt.figure(figsize=(8, 6))
sns.swarmplot(x='Polyuria', y='Age', hue='class', data=df, palette='Set1', dodge=Tr
plt.legend(title="Prediction", loc="upper right")
plt.title("Polyuria vs. Age")
plt.xlabel("Polyuria")
plt.ylabel("Age")
plt.show()
```
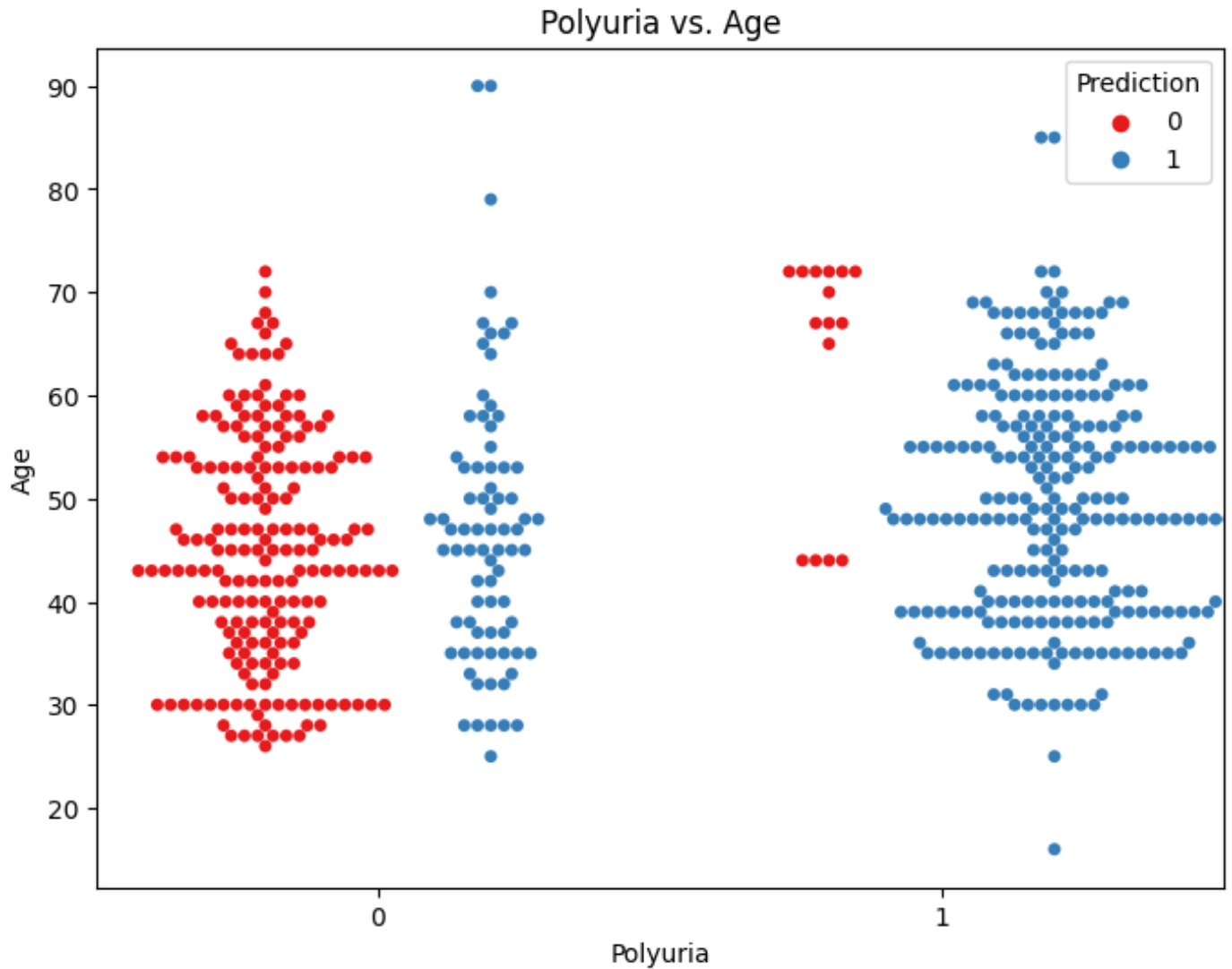
```
# Strip plot to visualize between age and polydipsia.
plt.figure(figsize=(8, 6))
sns.stripplot(x='Polydipsia', y='Age', data=df, hue='class', palette='Set1', jitter
plt.title("Age vs. Polydipsia")
plt.xlabel("Polydipsia")
plt.ylabel("Age")
plt.legend(title="Prediction", loc="upper right")
plt.show()
```
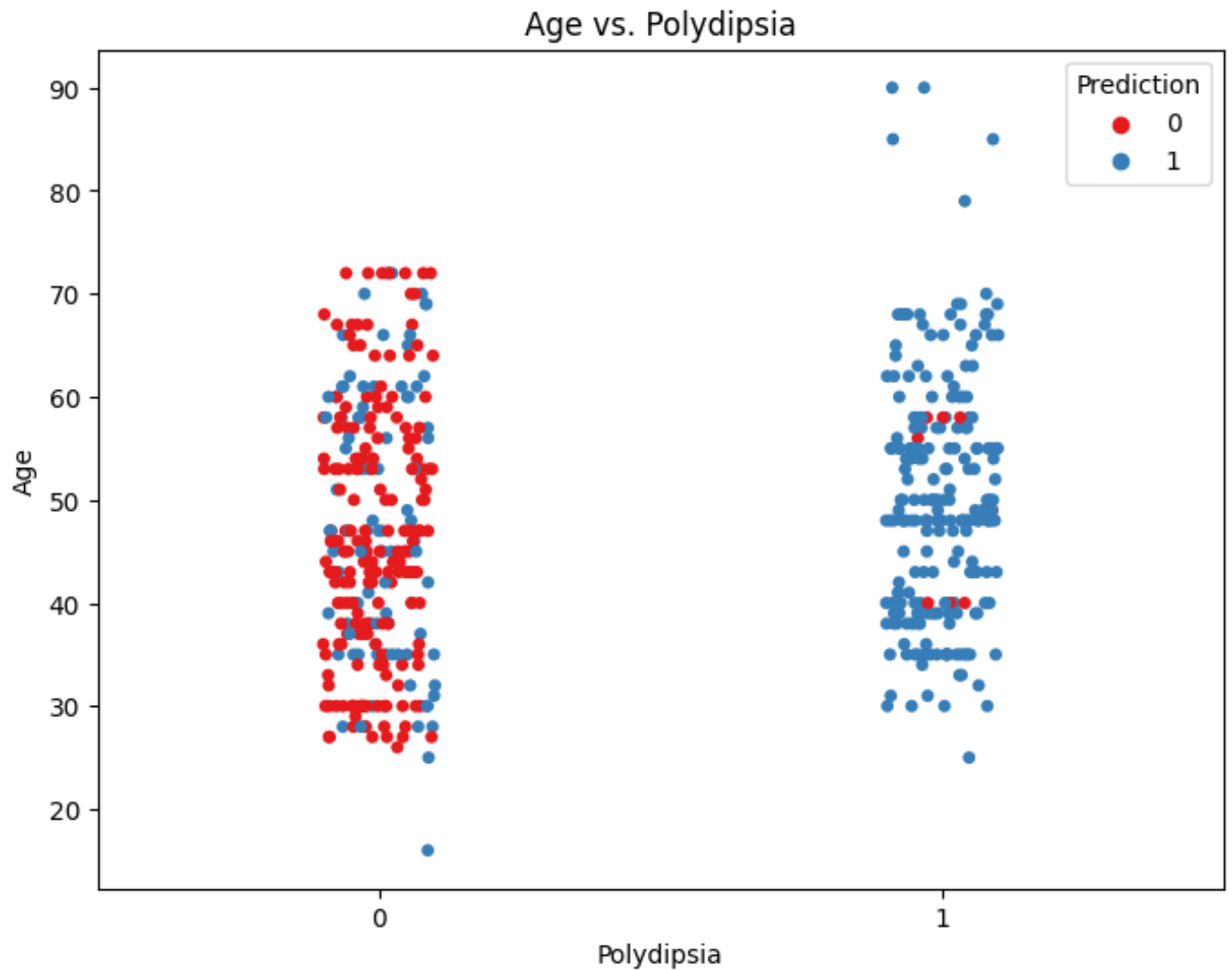
```python
# Visualize heatmap of the confusion matrix with annotations to indicate the preser
plt.figure(figsize=(6, 4))
sns.heatmap(confusion_mat, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.xticks(np.arange(2) + 0.5, ['Negative', 'Positive'])
plt.yticks(np.arange(2) + 0.5, ['Negative', 'Positive'])
plt.show()
```
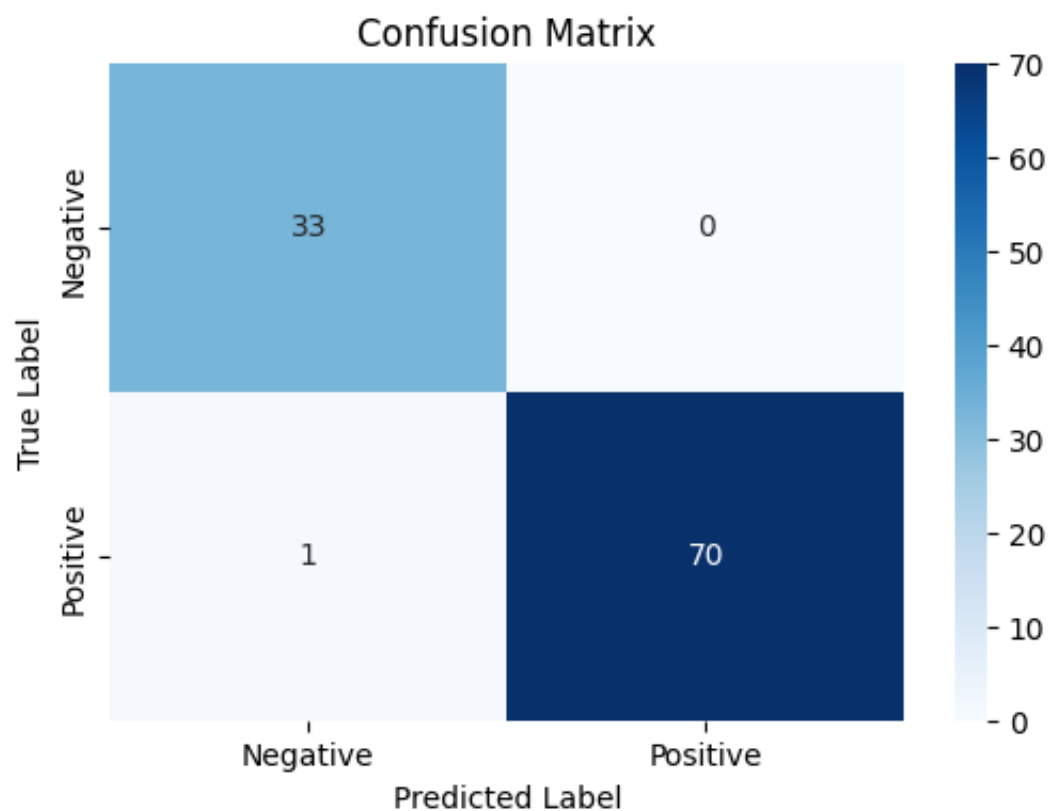
```
# Calculate the percentage of positive and negative cases
positive_count = df['class'].value_counts()[1]
negative_count = df['class'].value_counts()[0]
total_count = len(df)
positive_percentage = (positive_count / total_count) * 100
negative_percentage = (negative_count / total_count) * 100
plt.figure(figsize=(4, 4))
labels = ['Positive', 'Negative']
sizes = [positive_percentage, negative_percentage]
colors = ['orange', 'green']
explode = (0.05, 0)
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', st
plt.title("Distribution of Diabetes Cases")
plt.axis('equal')
plt.show()
```


Distribution of Diabetes Cases
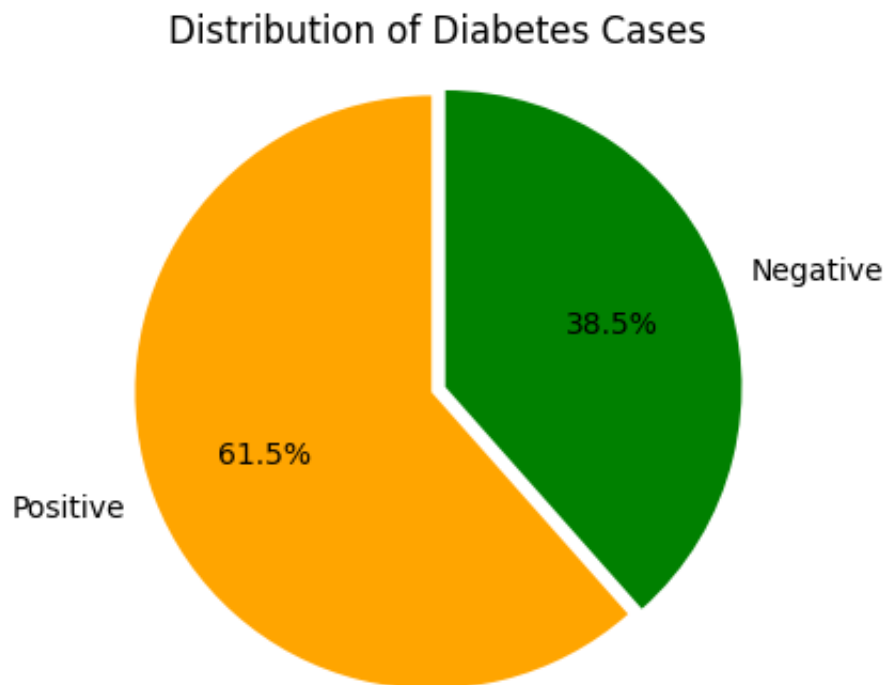
```
# Create separate sets of data for training the model and evaluating its performanc
diabetics = ['Age', 'Gender', 'Polyuria', 'Polydipsia', 'sudden weight loss', 'weak
train_size = 0.75
random_state = 38
X_train, X_test, y_train, y_test = train_test_split(df[diabetics], df['class'], tra
X_train = X_train.values
X_test = X_test.values
y_train = y_train.values
y_test = y_test.values
```

```python
# Decision Tree Classification
classification = DecisionTreeClassifier(max_depth=4, min_samples_split=5, random_st
classification.fit(X_train, y_train)
neg_pos = ['Negative', 'Positive']
data = export_graphviz(classification, precision=2,
                       feature_names=diabetics,
                       proportion=True,
                       class_names=neg_pos,
                       filled=True, rounded=True,
                       special_characters=True)
```
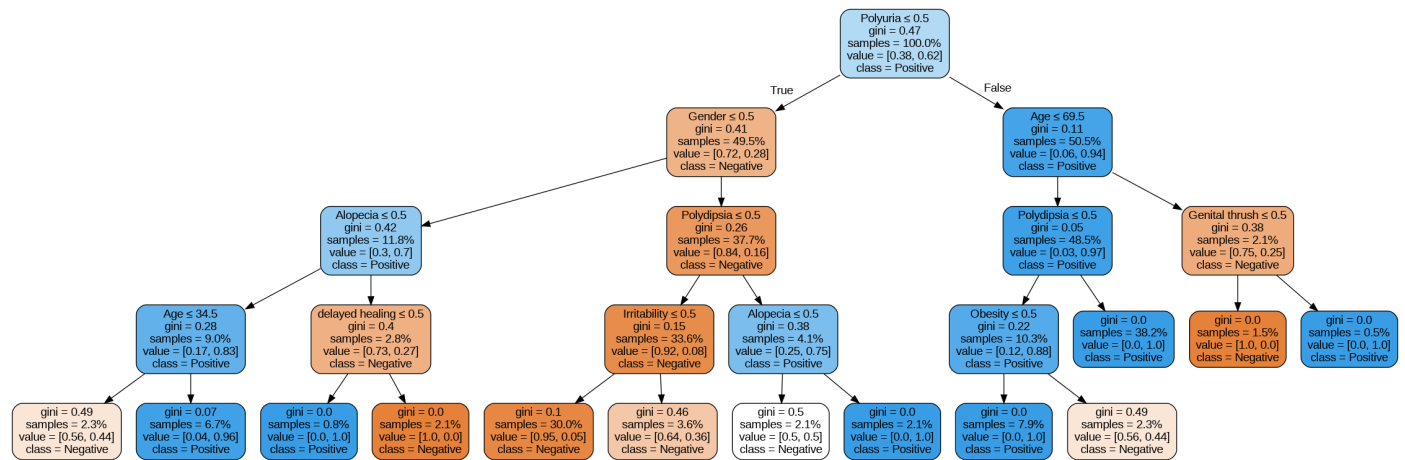
```python
# Graphviz to visualize and render the decision tree graph.
graph = graphviz.Source(data)
graph.format = 'png'
graph.filename = 'graph'
graph.render(view=False, format='png', cleanup=True)
Image.open('graph.png').show()
```

Polyuria ≤ 0.5
gini = 0.47
samples = 100.0%
value = [0.38, 0.62]
class = Positive

True

False

Gender ≤ 0.5
gini = 0.41
samples = 49.5%
value = [0.72, 0.28]
class = Negative

Age ≤ 69.5
gini = 0.11
samples = 50.5%
value = [0.06, 0.94]
class = Positive

Alopecia ≤ 0.5
gini = 0.42
samples = 11.8%
value = [0.3, 0.7]
class = Positive

Polydipsia ≤ 0.5
gini = 0.26
samples = 37.7%
value = [0.84, 0.16]
class = Negative

Polydipsia ≤ 0.5
gini = 0.05
samples = 48.5%
value = [0.03, 0.97]
class = Positive

Genital thrush ≤ 0.5
gini = 0.38
samples = 2.1%
value = [0.75, 0.25]
class = Negative

Age ≤ 34.5
gini = 0.28
samples = 9.0%
value = [0.17, 0.83]
class = Positive

delayed healing ≤ 0.5
gini = 0.4
samples = 2.8%
value = [0.73, 0.27]
class = Negative

Irritability ≤ 0.5
gini = 0.15
samples = 33.6%
value = [0.92, 0.08]
class = Negative

Alopecia ≤ 0.5
gini = 0.38
samples = 4.1%
value = [0.25, 0.75]
class = Positive

Obesity ≤ 0.5
gini = 0.22
samples = 10.3%
value = [0.12, 0.88]
class = Positive

gini = 0.0
samples = 38.2%
value = [0.0, 1.0]
class = Positive

gini = 0.0
samples = 1.5%
value = [1.0, 0.0]
class = Negative

gini = 0.0
samples = 0.5%
value = [0.0, 1.0]
class = Positive

gini = 0.49
samples = 2.3%
value = [0.56, 0.44]
class = Negative

gini = 0.07
samples = 6.7%
value = [0.04, 0.96]
class = Positive

gini = 0.0
samples = 0.8%
value = [0.0, 1.0]
class = Positive

gini = 0.0
samples = 2.1%
value = [1.0, 0.0]
class = Negative

gini = 0.1
samples = 30.0%
value = [0.95, 0.05]
class = Negative

gini = 0.46
samples = 3.6%
value = [0.64, 0.36]
class = Negative

gini = 0.5
samples = 2.1%
value = [0.5, 0.5]
class = Negative

gini = 0.0
samples = 2.1%
value = [0.0, 1.0]
class = Positive

gini = 0.0
samples = 7.9%
value = [0.0, 1.0]
class = Positive

gini = 0.49
samples = 2.3%
value = [0.56, 0.44]
class = Negative

✓ 0s   completed at 5:37 AM   ● ✕