ass4ml

[25]:
```
# Step 1: Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
↪recall_score
```

[2]:
```
# Read the dataset
df = pd.read_csv('diabetes.csv')
```

[3]:
```
df.head()
```

[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	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

[4]:
```
df.tail()
```

[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
     Pedigree  Age  Outcome
763     0.171   63        0
764     0.340   27        0
765     0.245   30        0
766     0.349   47        1
767     0.315   23        0
```

[5]:
```
df.isnull()
```

[5]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  \
0          False    False          False          False    False  False
1          False    False          False          False    False  False
2          False    False          False          False    False  False
3          False    False          False          False    False  False
4          False    False          False          False    False  False
..           …      …              …              …        …      …
763        False    False          False          False    False  False
764        False    False          False          False    False  False
765        False    False          False          False    False  False
766        False    False          False          False    False  False
767        False    False          False          False    False  False

     Pedigree    Age  Outcome
0       False  False    False
1       False  False    False
2       False  False    False
3       False  False    False
4       False  False    False
..        …    …       …
763     False  False    False
764     False  False    False
765     False  False    False
766     False  False    False
767     False  False    False

[768 rows x 9 columns]
```

[7]:
```
df.isnull().sum()
```

[7]:
```
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
Pedigree         0
```

```
Age              0
Outcome          0
dtype: int64
```

[8]:
```
df.notnull()
```

[8]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0           True     True           True           True     True  True
1           True     True           True           True     True  True
2           True     True           True           True     True  True
3           True     True           True           True     True  True
4           True     True           True           True     True  True
..           …       …              …              …        …    …
763         True     True           True           True     True  True
764         True     True           True           True     True  True
765         True     True           True           True     True  True
766         True     True           True           True     True  True
767         True     True           True           True     True  True

     Pedigree   Age  Outcome
0        True  True     True
1        True  True     True
2        True  True     True
3        True  True     True
4        True  True     True
..        …    …        …
763      True  True     True
764      True  True     True
765      True  True     True
766      True  True     True
767      True  True     True

[768 rows x 9 columns]
```

[9]:
```
df.notnull().sum()
```

[9]:
```
Pregnancies      768
Glucose          768
BloodPressure    768
SkinThickness    768
Insulin          768
BMI              768
Pedigree         768
Age              768
Outcome          768
dtype: int64
```

[11]:
```
df.shape
```

[11]:
```
(768, 9)
```

[12]:
```
df.describe()
```

[12]:

	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	Pedigree	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

[13]:
```
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   Pedigree       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
```

[14]:
```
# Assuming the dataset is clean, we separate features and labels
X = df.drop('Outcome', axis=1)  # Features
```

```
y = df['Outcome']                    # Labels
```

[15]:
```
# Step 5: Split the data 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)
```

[16]:
```
# Step 6: Train the KNN model
knn = KNeighborsClassifier(n_neighbors=5)  # You can adjust the number of␣
 ↪neighbors
knn.fit(X_train, y_train)
```

[16]:
```
KNeighborsClassifier()
```

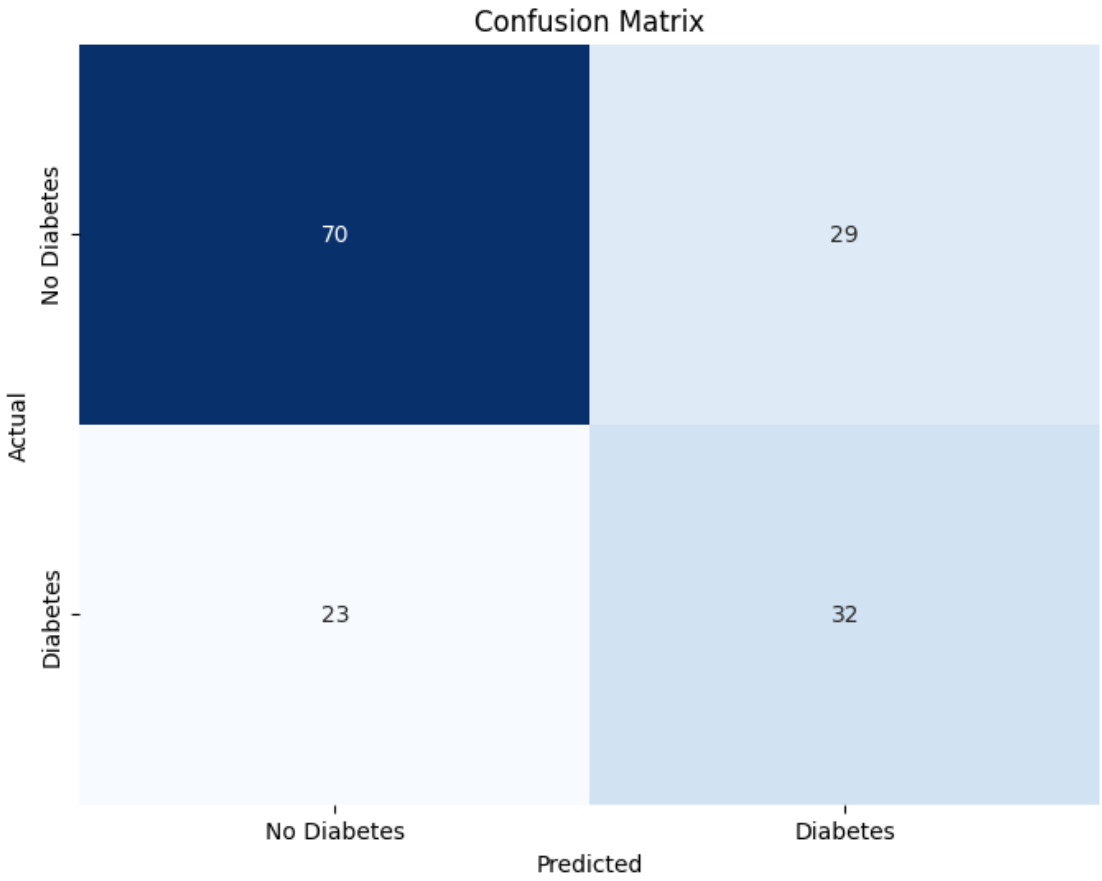
[17]:
```
# Step 7: Make predictions
y_pred = knn.predict(X_test)
```

[18]:
```
# Step 8: Evaluate the model
conf_matrix = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
error_rate = 1 - accuracy
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

[21]:
```
# Step 9: Print results
print(f"Accuracy: {accuracy:.2f}")
print(f"Error Rate: {error_rate:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

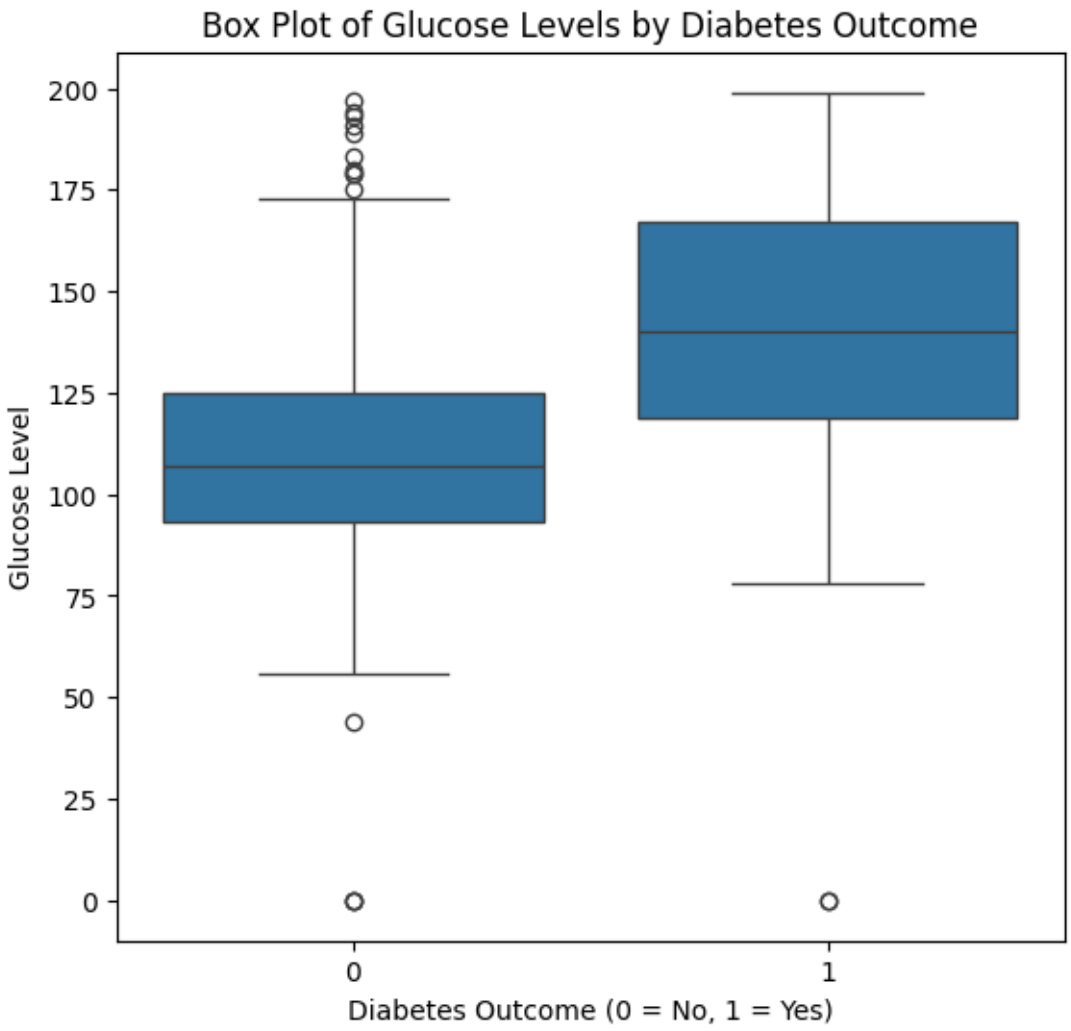
```
Accuracy: 0.66
Error Rate: 0.34
Precision: 0.52
Recall: 0.58
```

[26]:
```
# Step 10: Visualize the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['No Diabetes', 'Diabetes'],
            yticklabels=['No Diabetes', 'Diabetes'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

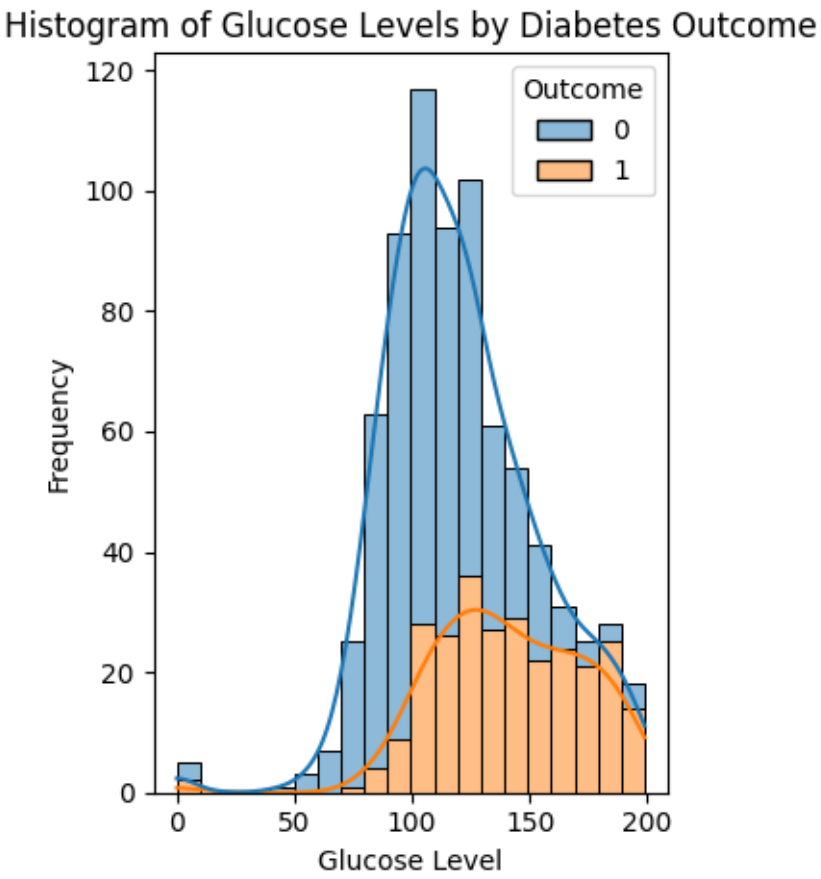


[27]:
```
# Step 4: Visualize the distribution of "Glucose" levels
plt.figure(figsize=(14, 6))

# Box plot
plt.subplot(1, 2, 1)
sns.boxplot(x='Outcome', y='Glucose', data=df)
plt.title('Box Plot of Glucose Levels by Diabetes Outcome')
plt.xlabel('Diabetes Outcome (0 = No, 1 = Yes)')
plt.ylabel('Glucose Level')
```

[27]:
```
Text(0, 0.5, 'Glucose Level')
```



[28]:
```
# Histogram
plt.subplot(1, 2, 2)
sns.histplot(df, x='Glucose', hue='Outcome', multiple='stack', bins=20,␣
 ↪kde=True)
plt.title('Histogram of Glucose Levels by Diabetes Outcome')
plt.xlabel('Glucose Level')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```


Histogram of Glucose Levels by Diabetes Outcome
Outcome
0
1
Frequency
Glucose Level
0
20
40
60
80
100
120
0
50
100
150
200