```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, recall_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/diabetes2.csv')
df.sample(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
631	0	102	78	40	90	34.5	0.238	24	0
411	1	112	72	30	176	34.4	0.528	25	0
640	0	102	86	17	105	29.3	0.695	27	0
16	0	118	84	47	230	45.8	0.551	31	1
638	7	97	76	32	91	40.9	0.871	32	1

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

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

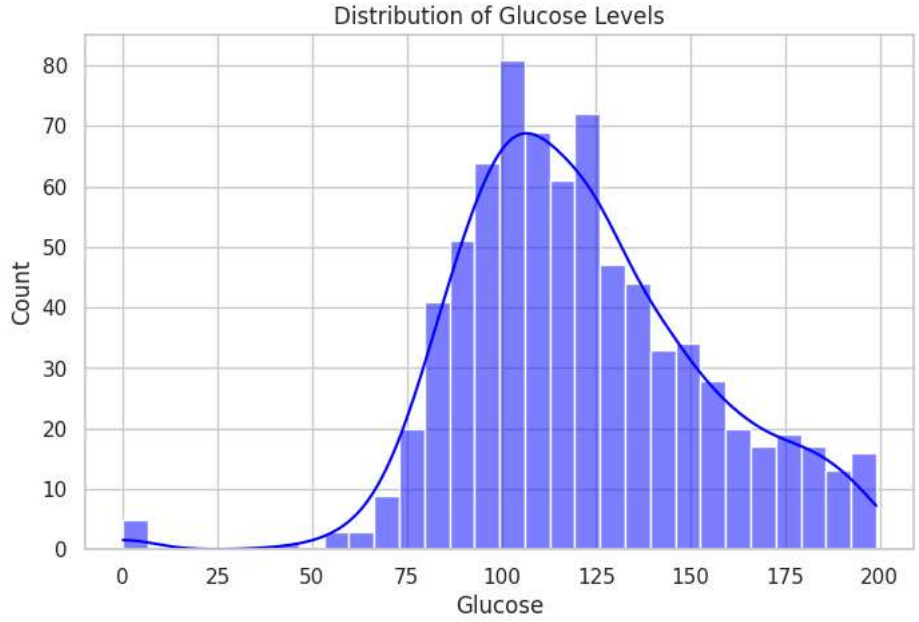
	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

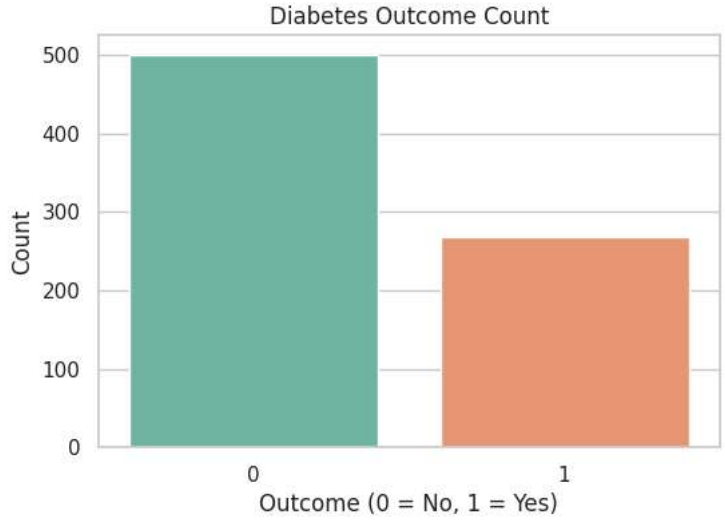
```
df.duplicated().sum()
```

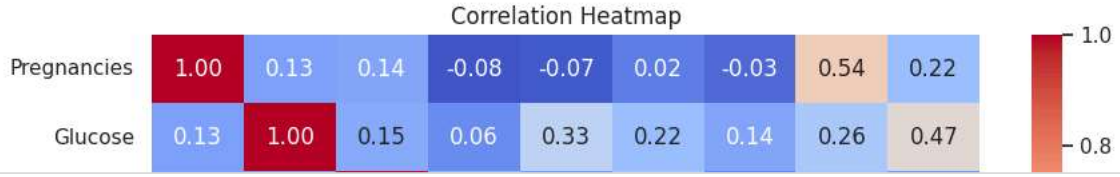
```
np.int64(0)
```

```
import seaborn as sns

sns.set(style="whitegrid")

# Distribution of Glucose levels
plt.figure(figsize=(8,5))
sns.histplot(df['Glucose'], kde=True, bins=30, color="blue")
plt.title("Distribution of Glucose Levels")
plt.xlabel("Glucose")
plt.ylabel("Count")
plt.show()

# Outcome count (0 = Non-diabetic, 1 = Diabetic)
plt.figure(figsize=(6,4))
sns.countplot(x='Outcome', data=df, palette='Set2')
plt.title("Diabetes Outcome Count")
plt.xlabel("Outcome (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()

# Correlation heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()

# Age distribution by outcome
plt.figure(figsize=(8,5))
sns.boxplot(x="Outcome", y="Age", data=df, palette="Set3")
plt.title("Age Distribution by Diabetes Outcome")
plt.show()
```




```
/tmp/ipython-input-1448572228.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='Outcome', data=df, palette='Set2')
```





```
X_train, X_test, y_train, y_test = train_test_split(df.drop(['Outcome'], axis=1),
                                                    df['Outcome'],
                                                    test_size=0.2,
                                                    random_state=2)
```

```
std = StandardScaler()
```

```
X_train = std.fit_transform(X_train)
X_test = std.transform(X_test)
```

```
model = LogisticRegression(max_iter=1000, class_weight='balanced')
```

```
model.fit(X_train, y_train)
```

LogisticRegression

LogisticRegression(class_weight='balanced', max_iter=1000)

```
y_pred = model.predict(X_test)
```

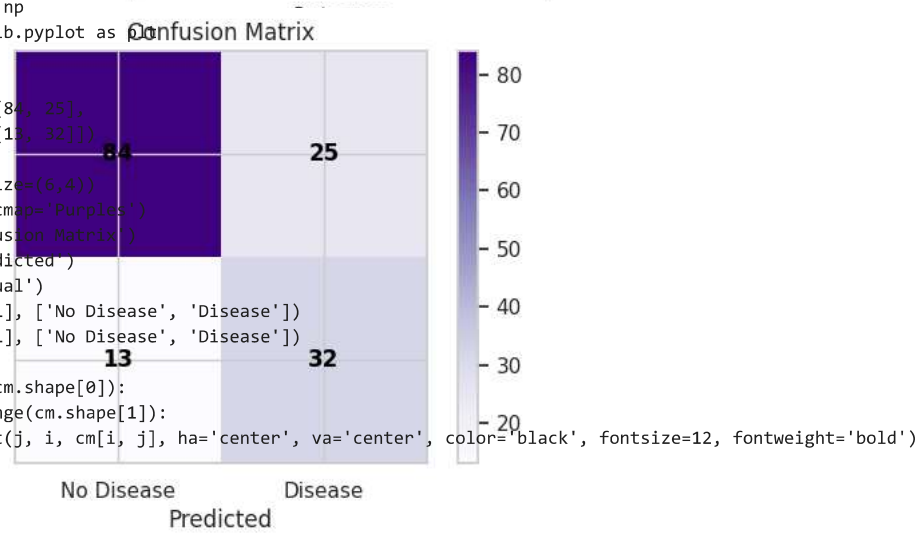
```
print('Accuracy: ', accuracy_score(y_test, y_pred))
print('Recall: ', recall_score(y_test, y_pred))
print('f1 score: ', f1_score(y_test, y_pred))
print('Confusion Matrix: ', confusion_matrix(y_test, y_pred))
```

```
Accuracy:  0.7532467532467533
Recall:  0.7333333333333333
f1 score:  0.6346153846153846
Confusion Matrix:  [[89 26]
 [12 33]]
```

```
sns.boxplot(x="Outcome", y="Age", data=df, palette="Set3")
```

```
/tmp/ipython-input-1448573228.py:29: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

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


cm = np.array([[84, 25],
               [13, 32]])
plt.figure(figsize=(6,4))
plt.imshow(cm, cmap='Purples')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks([0, 1], ['No Disease', 'Disease'])
plt.yticks([0, 1], ['No Disease', 'Disease'])
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        plt.text(j, i, cm[i, j], ha='center', va='center', color='black', fontsize=12, fontweight='bold')

plt.colorbar()
plt.show()
```




```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.76      0.81       109
           1       0.56      0.73      0.63        45

    accuracy                           0.75       154
   macro avg       0.72      0.75      0.72       154
weighted avg       0.78      0.75      0.76       154
```

Start coding or generate with AI.