Importing required library

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, precision_s
```

Loading dataset

In [3]:
```
data = pd.read_csv("diabetes.csv")
```

In [8]:
```
data.head(1)
```

Out[8]:

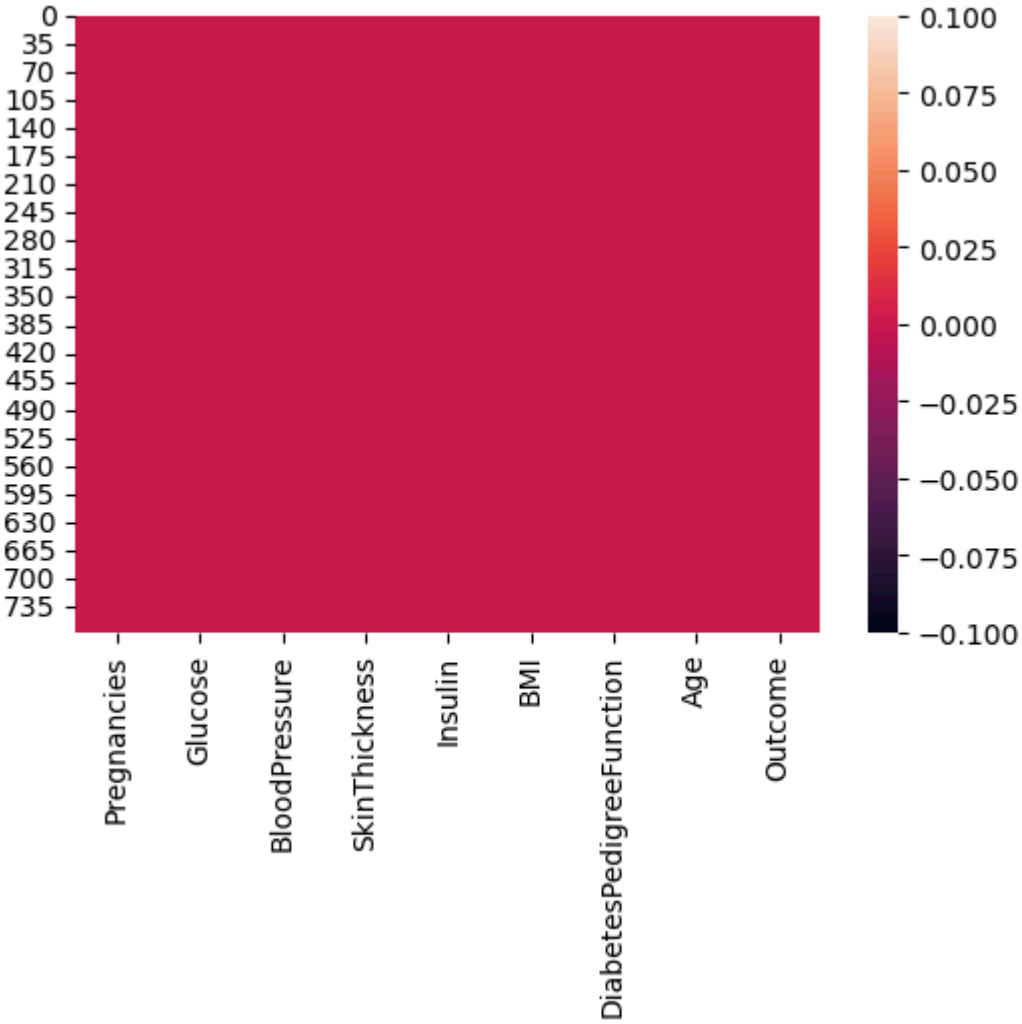
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPed
0	6	148	72	35	0	33.6	

In [9]:
```
data.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
```

In [15]:
```
plt.figure(figsize=(6,4))
sns.heatmap(data.isnull()) #if there is blank then there is null values
# plt.xticks(rotation = 45)
```

Out[15]:
```
<Axes: >
```



In [16]:
```
data.describe()
```

Out[16]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.0
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.9
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.8
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.3
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.0
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.6
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.1

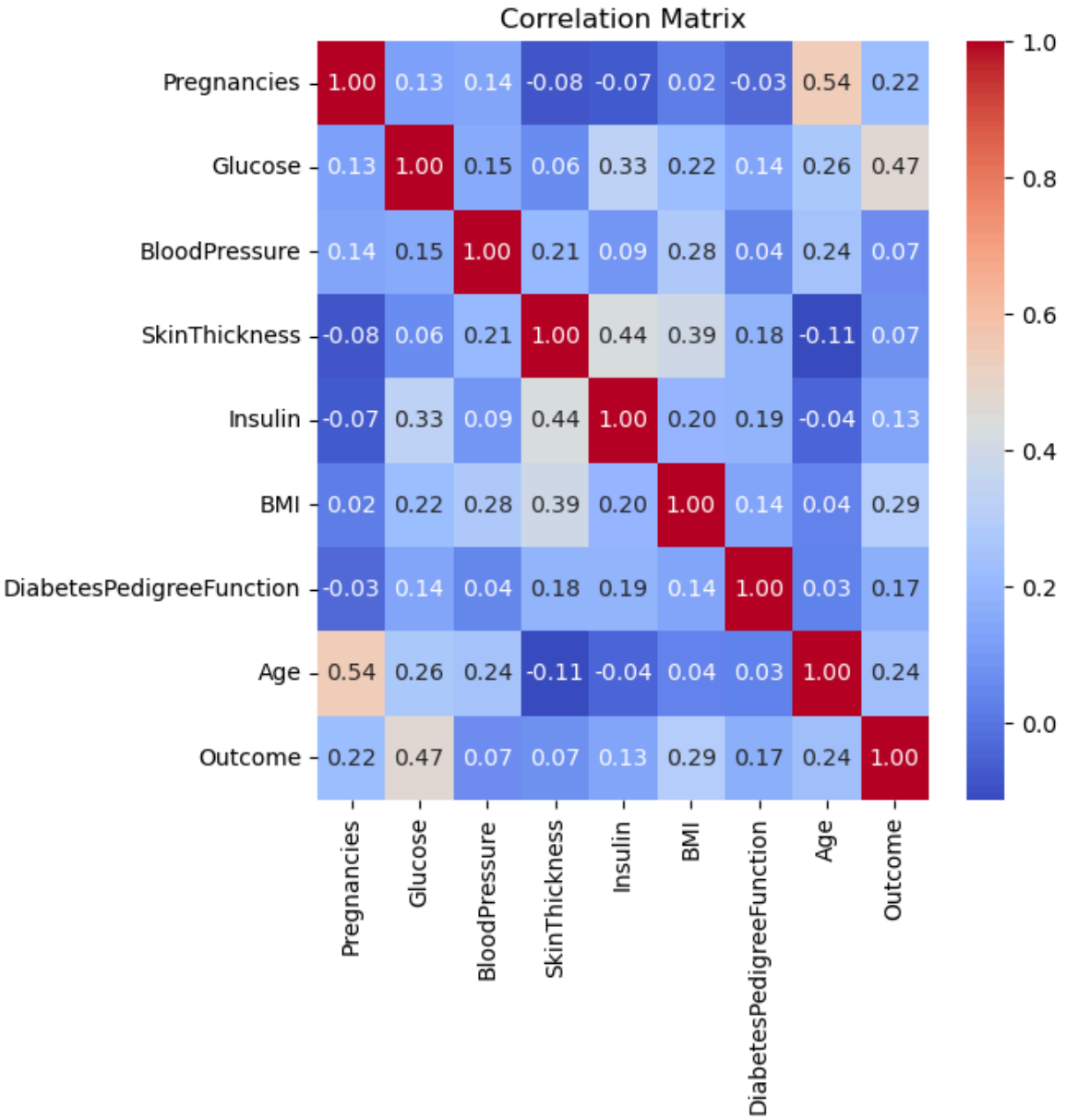
Co-relation Martrix

In [17]:
```
correlation = data.corr()
```


In [18]:
```
print(correlation)
```

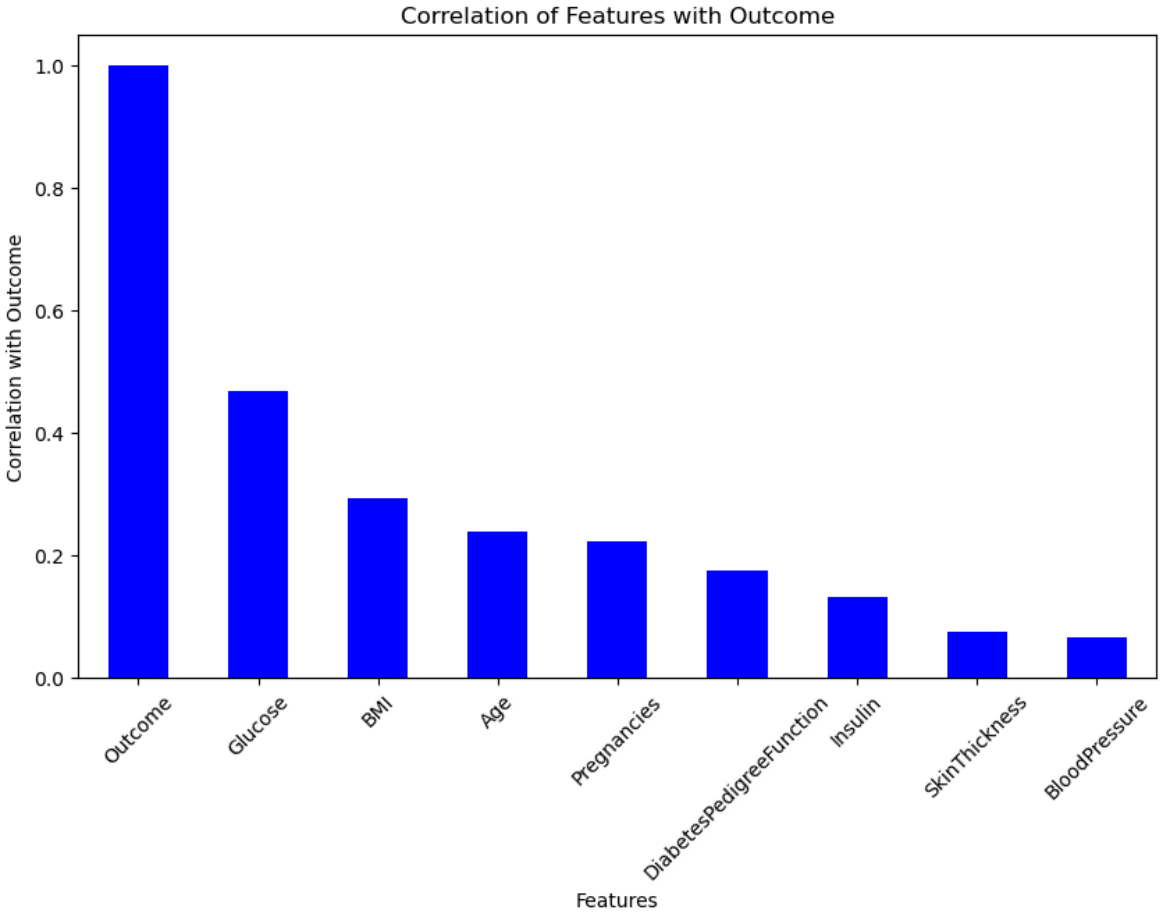
```
                          Pregnancies   Glucose  BloodPressure  SkinThickn
ess  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081
672
Glucose                      0.129459  1.000000       0.152590       0.057
328
BloodPressure                0.141282  0.152590       1.000000       0.207
371
SkinThickness               -0.081672  0.057328       0.207371       1.000
000
Insulin                     -0.073535  0.331357       0.088933       0.436
783
BMI                          0.017683  0.221071       0.281805       0.392
573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183
928
Age                          0.544341  0.263514       0.239528      -0.113
970
Outcome                      0.221898  0.466581       0.065068       0.074
752

                            Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies               -0.073535  0.017683                 -0.033523
Glucose                    0.331357  0.221071                  0.137337
BloodPressure              0.088933  0.281805                  0.041265
SkinThickness              0.436783  0.392573                  0.183928
Insulin                    1.000000  0.197859                  0.185071
BMI                        0.197859  1.000000                  0.140647
DiabetesPedigreeFunction   0.185071  0.140647                  1.000000
Age                       -0.042163  0.036242                  0.033561
Outcome                    0.130548  0.292695                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.263514  0.466581
BloodPressure             0.239528  0.065068
SkinThickness            -0.113970  0.074752
Insulin                  -0.042163  0.130548
BMI                       0.036242  0.292695
DiabetesPedigreeFunction  0.033561  0.173844
Age                       1.000000  0.238356
Outcome                   0.238356  1.000000
```

In [19]:
```
plt.figure(figsize=(6, 6))
sns.heatmap(correlation, annot=True, fmt=".2f", cmap="coolwarm")    #dark
plt.title("Correlation Matrix")
plt.show()
```



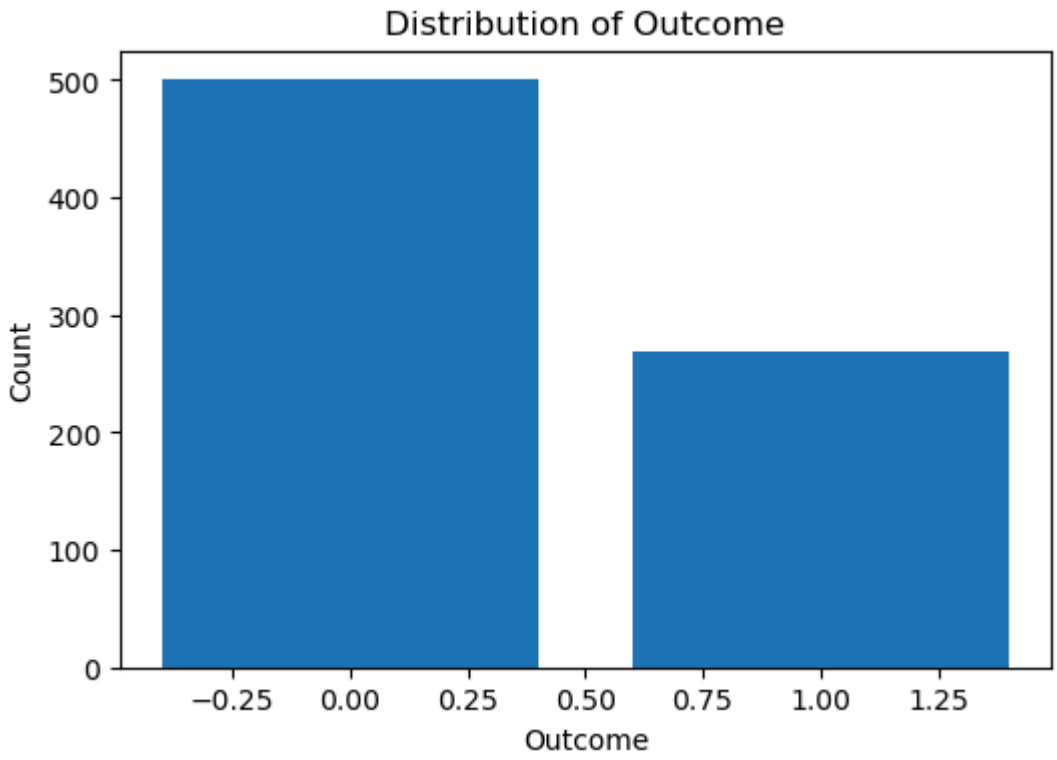
In [20]:
```
correlation_with_outcome = correlation['Outcome'].abs().sort_values(ascen

plt.figure(figsize=(10, 6))
correlation_with_outcome.plot(kind='bar', color='blue')
plt.xlabel('Features')
plt.ylabel('Correlation with Outcome')
plt.title('Correlation of Features with Outcome')
plt.xticks(rotation=45)
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [ ]:

In [21]:
```
outcome_counts = data['Outcome'].value_counts()
plt.figure(figsize=(6, 4))
plt.bar(outcome_counts.index, outcome_counts.values,)
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.title('Distribution of Outcome')
plt.show()
```



Train test split

In [22]:
```
x = data.drop('Outcome', axis = 1)
y = data['Outcome']

x_train,x_test, y_train, y_test = train_test_split(x, y , test_size = 0.2
```

In [23]:
```
x_train.head(2)
```

Out[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesP
602	1	124	74	36	0	27.8	
429	1	95	82	25	180	35.0	

Training The model

1. Logistic Regression

In [24]:
```
model = LogisticRegression(max_iter = 1000)
model.fit(x_train, y_train)
```

Out[24]:
```
LogisticRegression(max_iter=1000)
```

Making Predictions

In [25]:
```
predictions = model.predict(x_test)
predictions
```

Out[25]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0])
```

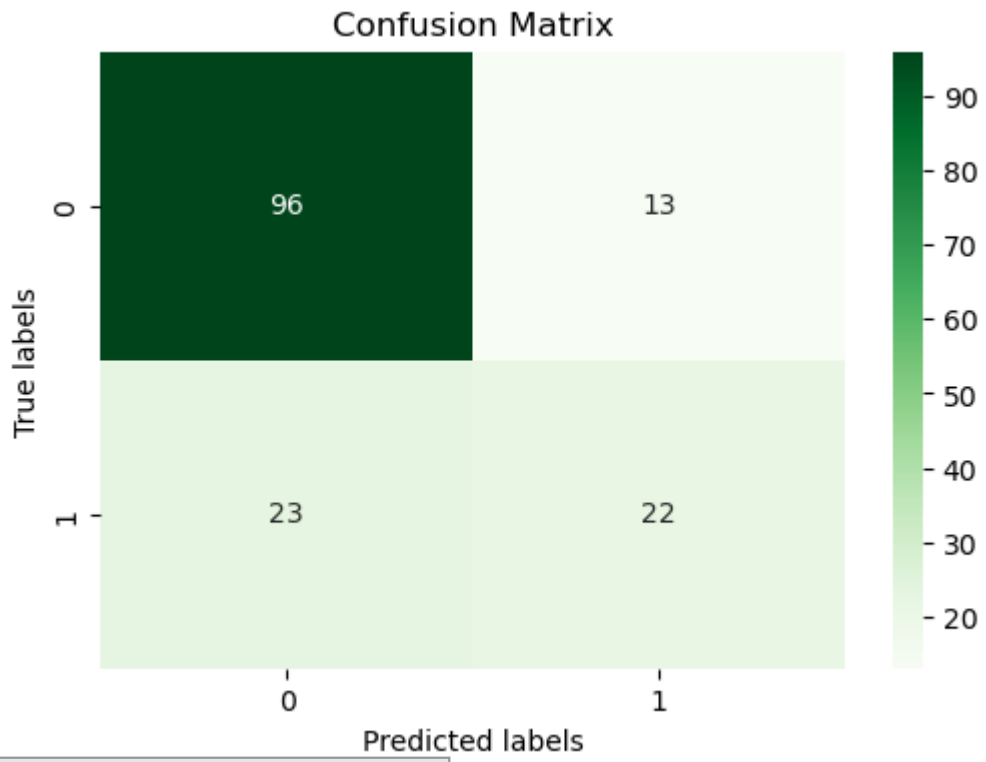
Evalution

In [26]:
```
accuracy = accuracy_score(predictions, y_test)
```

In [27]:
```
accuracy
```

Out[27]:
```
0.7662337662337663
```

In [28]:
```
conf_matrix = confusion_matrix(y_test, predictions)

plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Greens")
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```



In [29]:
```
lr_precision = precision_score(y_test, predictions)
lr_precision
```

Out[29]:
```
0.6285714285714286
```

In [30]:
```
f1 = f1_score(y_test, predictions)
f1
```

Out[30]:
```
0.5499999999999999
```

2. Support Vector Machine

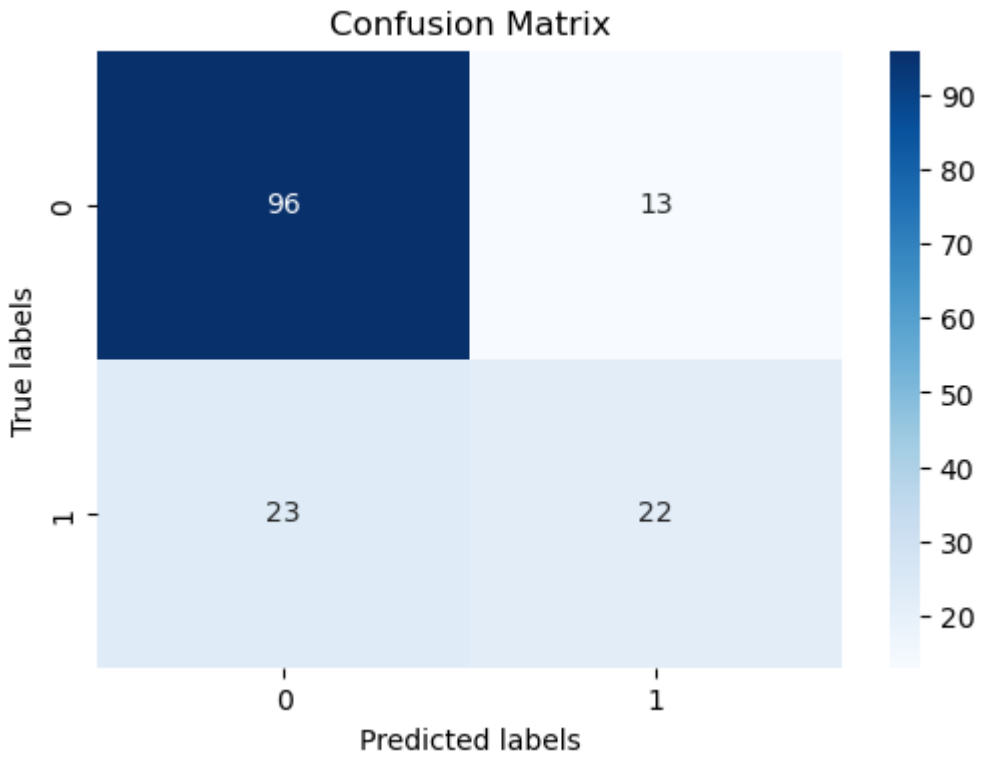
In [31]:
```
svm_model = SVC(kernel='linear')
svm_model.fit(x_train, y_train)
svm_y_pred = svm_model.predict(x_test)
```

Evaluation

In [32]:
```
svm_accuracy = accuracy_score(y_test, svm_y_pred)
print("SVM Accuracy:", svm_accuracy)
```

```
SVM Accuracy: 0.7532467532467533
```

In [33]:
```
conf_matrix_svm = confusion_matrix(y_test, svm_y_pred)
# print("Confusion Matrix for SVM:")
# print(conf_matrix_svm)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted labels")
plt.ylabel("True labels")
plt.title("Confusion Matrix")
plt.show()
```



In [34]:
```
svm_precision = precision_score(y_test, svm_y_pred)
svm_precision
```

Out[34]: 0.6

In [35]:
```
f1 = f1_score(y_test, svm_y_pred)
f1
```

Out[35]: 0.5249999999999999

In [ ]:

In [36]:

In [ ]:

In [ ]: