```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, precision_score, recall_score,
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt
```

```
# Load dataset
data = pd.read_csv("/content/diabetes.csv")

# Display first 5 rows
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes
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	

Next steps: Generate code with data New interactive sheet

```
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
```

```
X = data.drop("Outcome", axis=1)   # Features
y = data["Outcome"]                # Target (0 = No Diabetes, 1 = Diabetes)
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
C_values = [0.1, 1, 10]
results = {}

for C in C_values:
    svm = SVC(C=C, kernel='rbf')
    svm.fit(X_train_scaled, y_train)

    y_pred = svm.predict(X_test_scaled)

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    results[C] = {
        "model": svm,
        "accuracy": acc,
        "precision": prec,
        "recall": rec,
        "f1_score": f1
    }
```

```
for C, metrics in results.items():
    print(f"\nSVM Results for C = {C}")
    print(f"Accuracy : {metrics['accuracy']:.4f}")
    print(f"Precision: {metrics['precision']:.4f}")
    print(f"Recall   : {metrics['recall']:.4f}")
    print(f"F1-Score : {metrics['f1_score']:.4f}")
```

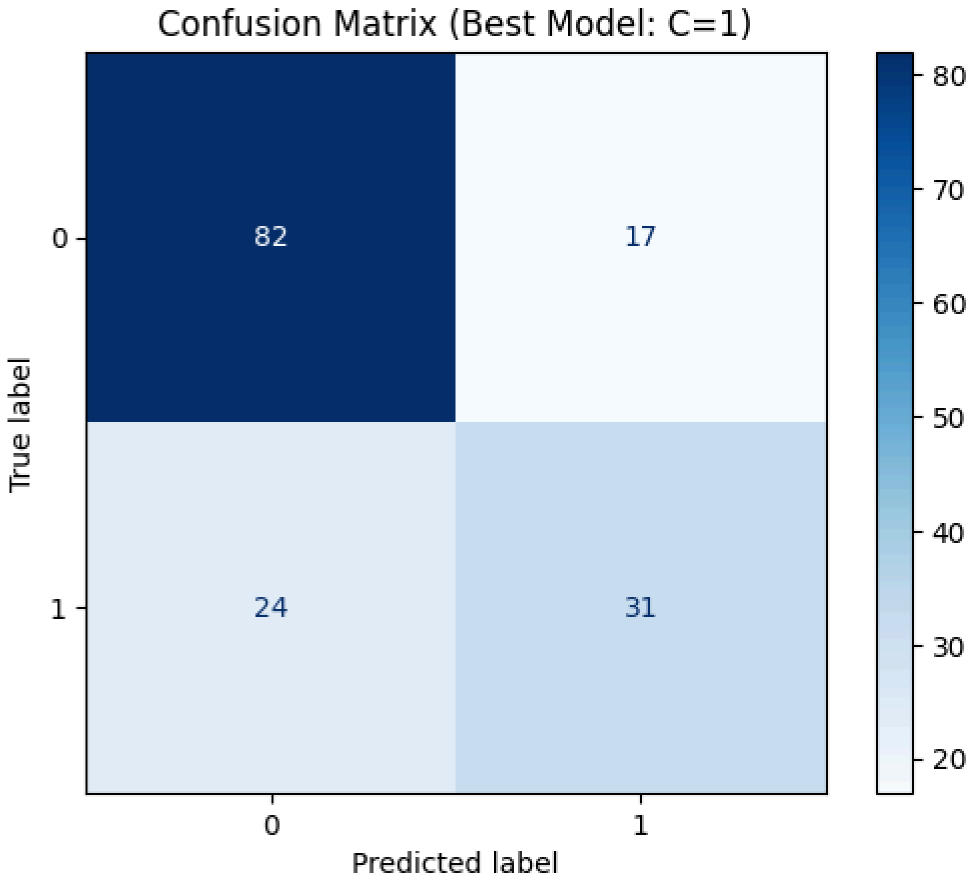
```

SVM Results for C = 0.1
Accuracy : 0.7468
Precision: 0.7857
Recall   : 0.4000
F1-Score : 0.5301

SVM Results for C = 1
Accuracy : 0.7338
Precision: 0.6458
Recall   : 0.5636
```

```
F1-Score : 0.6019

SVM Results for C = 10
Accuracy : 0.7143
Precision: 0.6078
Recall   : 0.5636
F1-Score : 0.5849
```

```
best_C = max(results, key=lambda x: results[x]['f1_score'])
best_model = results[best_C]['model']

print("Best C value:", best_C)
```

```
Best C value: 1
```

```
y_best_pred = best_model.predict(X_test_scaled)

cm = confusion_matrix(y_test, y_best_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap="Blues")
plt.title(f"Confusion Matrix (Best Model: C={best_C})")
plt.show()
```



```
Start coding or generate with AI.
```