In [147]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn import preprocessing
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, roc_auc_score
from imblearn.over_sampling import SMOTE
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

Data PreprPreprocessing

In [48]:

```
disease = pd.read_csv("framingham.csv")
```

In [49]:

```
disease
```

Out[49]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp
0	1	39	4.0	0	0.0	0.0	0	0
1	0	46	2.0	0	0.0	0.0	0	0
2	1	48	1.0	1	20.0	0.0	0	0
3	0	61	3.0	1	30.0	0.0	0	1
4	0	46	3.0	1	23.0	0.0	0	0
...	...	...	...	...	...	...	...	..
4235	0	48	2.0	1	20.0	NaN	0	0
4236	0	44	1.0	1	15.0	0.0	0	0
4237	0	52	2.0	0	0.0	0.0	0	0
4238	1	40	3.0	0	0.0	0.0	0	1
4239	0	39	3.0	1	30.0	0.0	0	0

4240 rows × 16 columns

Renaming the male column to Sex_male to know that 1 is male and 0 is female

In [50]:

```
disease.rename(columns={'male' : 'Sex_male'}, inplace = True)
```

In [51]:
```
disease
```

Out[51]:

	Sex_male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevale
0	1	39	4.0	0	0.0	0.0	0	
1	0	46	2.0	0	0.0	0.0	0	
2	1	48	1.0	1	20.0	0.0	0	
3	0	61	3.0	1	30.0	0.0	0	
4	0	46	3.0	1	23.0	0.0	0	
...	...	...	...	...	...	...	...	
4235	0	48	2.0	1	20.0	NaN	0	
4236	0	44	1.0	1	15.0	0.0	0	
4237	0	52	2.0	0	0.0	0.0	0	
4238	1	40	3.0	0	0.0	0.0	0	
4239	0	39	3.0	1	30.0	0.0	0	

4240 rows × 16 columns

Handling the missing values

In [54]:
```
disease.isnull().sum()
```

Out[54]:
```
Sex_male             0
age                  0
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
dtype: int64
```

In [55]:
```
disease['cigsPerDay'].skew()
```

Out[55]:
```
1.2470523561848126
```

In [56]:
```
#As the skew of cigsPerDay is greater than 0 so we fill the null value of t
disease['cigsPerDay'] = disease['cigsPerDay'].fillna(disease['cigsPerDay'].
```

In [57]:
```
disease['totChol'].skew()
```

Out[57]:
```
0.8718805634765354
```

In [58]:
```
#As the skew is 0.87 which is greater than 0 so we fill the null value of t
disease['totChol']=disease['totChol'].fillna(disease['totChol'].mean())
```

In [59]:
```
#As the BPMeds is in binary form so we use mode to fill the null values
disease['BPMeds']=disease['BPMeds'].fillna(disease['BPMeds'].mode()[0])
```

In [60]:
```
disease['BMI'].skew()
```

Out[60]:
```
0.9821832986950597
```

In [61]:
```
#As the skew is 0.98 which is greater than 0 so we fill the null value of B
disease['BMI']=disease['BMI'].fillna(disease['BMI'].median())
```

In [62]:
```
disease['glucose'].skew()
```

Out[62]:
```
6.2149483495346765
```

In [63]:
```
#As the skew is 6.214 which is highly greater than 0 so we fill the null va
disease['glucose']=disease['glucose'].fillna(disease['glucose'].median())
```

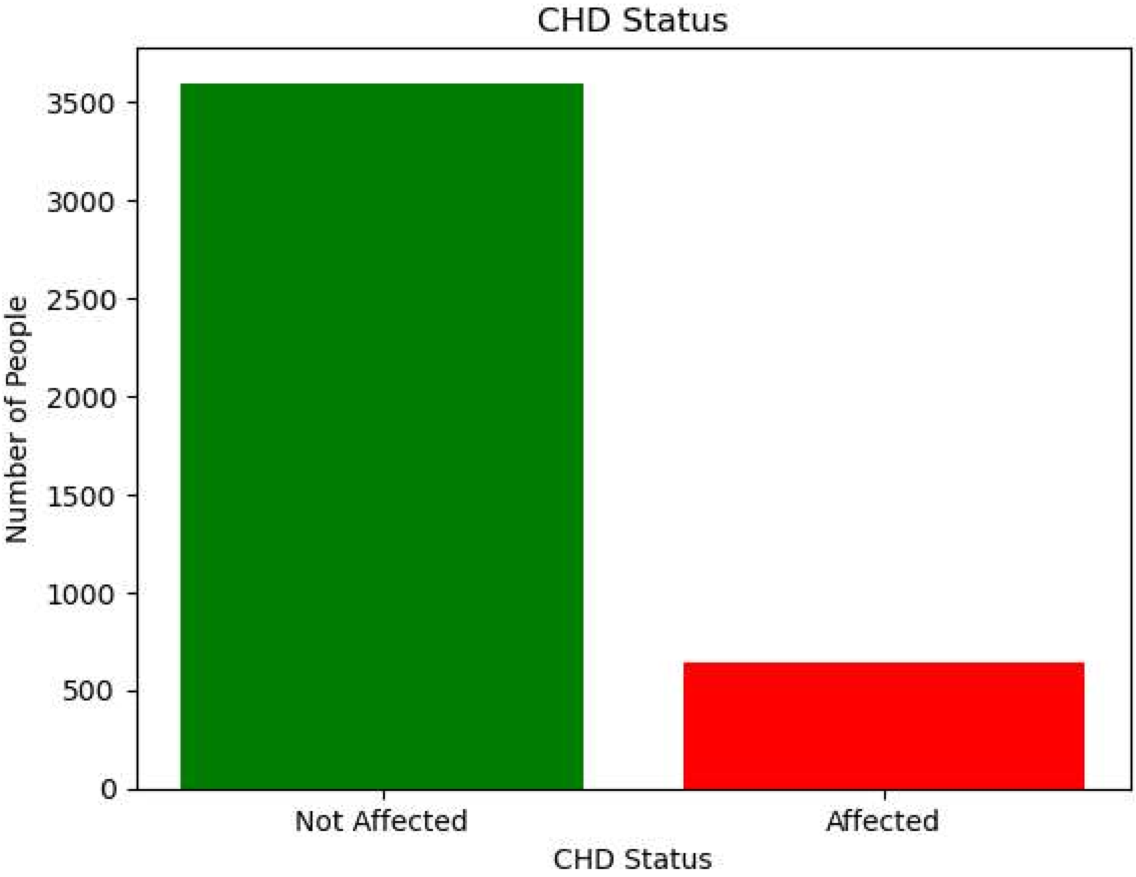
In [64]:
```
disease.isnull().sum()
```

Out[64]:
```
Sex_male           0
age                0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
heartRate          1
glucose            0
TenYearCHD         0
dtype: int64
```

Making a Bar Plot to Count people affected by CHD

In [65]:

```
counts = disease["TenYearCHD"].value_counts()

plt.bar(["Not Affected", "Affected"], counts, color=["green", "red"])
plt.title("CHD Status")
plt.ylabel("Number of People")
plt.xlabel("CHD Status")
plt.show()
```



Spliting the data

In [66]:

```
X = np.asarray(disease[['age', 'Sex_male', 'cigsPerDay',
                        'totChol', 'sysBP', 'glucose']])
y = np.asarray(disease['TenYearCHD'])

# normalization of the dataset
X = preprocessing.StandardScaler().fit_transform(X)

# Train test split
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.3, random

print(f"Train set: {X_train.shape} {y_train.shape}")
print(f"Test set: {X_test.shape} {y_test.shape}")
```

```
Train set: (2968, 6) (2968,)
Test set: (1272, 6) (1272,)
```

In [67]:

```
# As you can see from the barplot there is a bias in data so we use smote t
# Handle class imbalance with SMOTE (Oversampling the minority class)
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

print(f"Train set: {X_resampled.shape} {y_resampled.shape}")
print(f"Test set: {X_test.shape} {y_test.shape}")
```

```
Train set: (5024, 6) (5024,)
Test set: (1272, 6) (1272,)
```

Fitting Logistic Regresssion Model for Heart Disease Prediction

In [111]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_resampled,y_resampled)
y_pred = model.predict(X_test)
```

Checking the Accuracy of the model

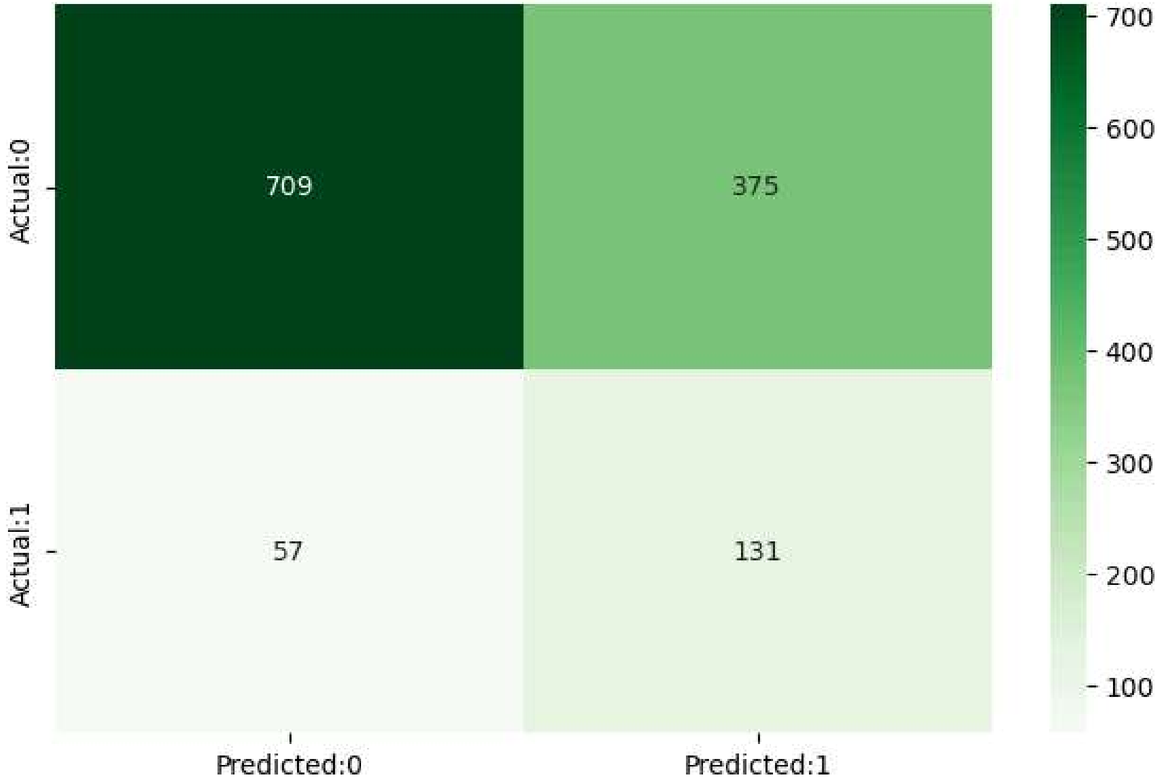
In [114]:

```
score_lr=round(accuracy_score(y_test,y_pred)*100,2)
roc_lr= round(roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])*100,
print("Accuracy of the model is =",score_lr)
print("ROC-AUC Score:",roc_lr)
```

```
Accuracy of the model is = 66.04
ROC-AUC Score: 72.07
```

In [115]:

```
# Confusion matrix
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                           columns = ['Predicted:0', 'Predicted:1'],
                           index =['Actual:0', 'Actual:1'])

plt.figure(figsize = (8, 5))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")

plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```



```
The details for confusion matrix is =
              precision    recall  f1-score   support

           0       0.93      0.65      0.77      1084
           1       0.26      0.70      0.38       188

    accuracy                           0.66      1272
   macro avg       0.59      0.68      0.57      1272
weighted avg       0.83      0.66      0.71      1272
```

Fitting Random Forest Clasifier for Heart Disease Prediction

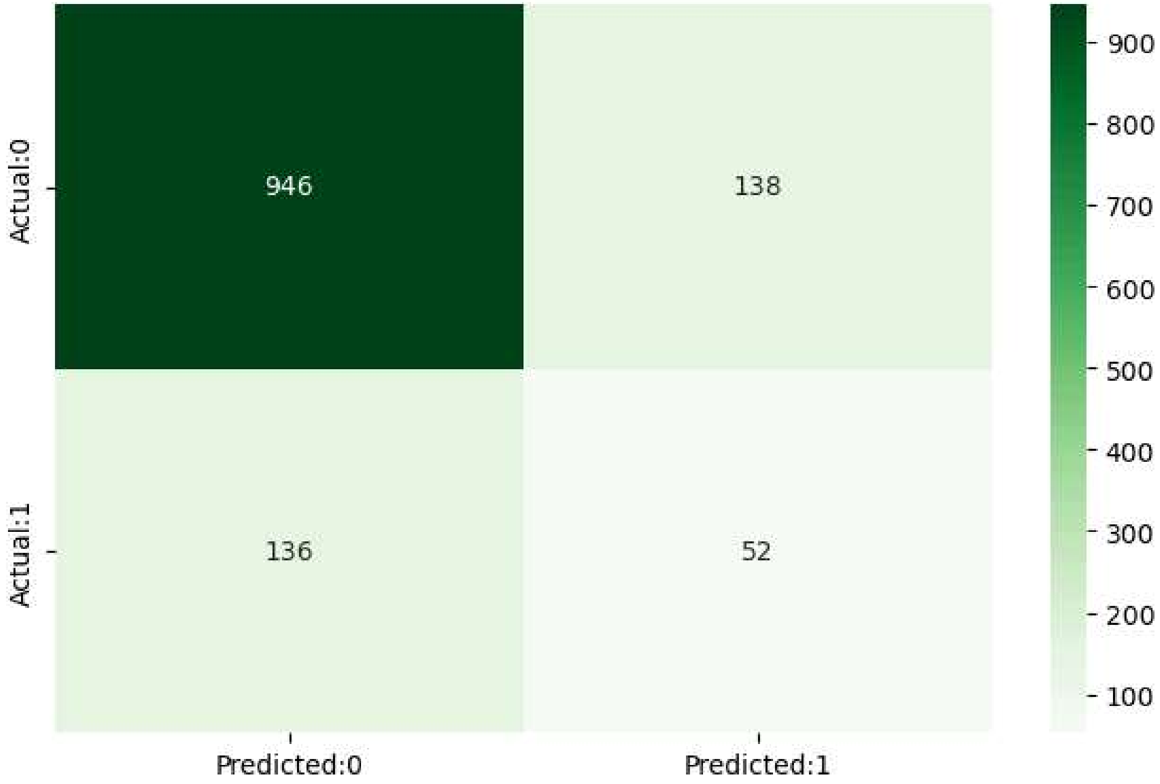
In [116]:

```
# Train a model
model_RFC = RandomForestClassifier(class_weight='balanced', random_state=42
model_RFC.fit(X_resampled, y_resampled)

# Evaluate
y_pred = model_RFC.predict(X_test)

print(classification_report(y_test, y_pred))

score_rf = round(accuracy_score(y_test,y_pred)*100,2)
roc_rf = round( roc_auc_score(y_test, model_RFC.predict_proba(X_test)[:, 1]

print("Accuracy of the model: ",score_rf)
print("ROC-AUC Score:",roc_rf)
```

```
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      1084
           1       0.27      0.28      0.28       188

    accuracy                           0.78      1272
   macro avg       0.57      0.57      0.57      1272
weighted avg       0.79      0.78      0.79      1272

Accuracy of the model:  78.46
ROC-AUC Score: 68.62
```

In [117]:

```python
# Confusion matrix
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                           columns = ['Predicted:0', 'Predicted:1'],
                           index =['Actual:0', 'Actual:1'])

plt.figure(figsize = (8, 5))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")

plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```



```
The details for confusion matrix is =
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      1084
           1       0.27      0.28      0.28       188

    accuracy                           0.78      1272
   macro avg       0.57      0.57      0.57      1272
weighted avg       0.79      0.78      0.79      1272
```

Fitting SVC while using gaussian rbf kernel for Heart Disease Prediction

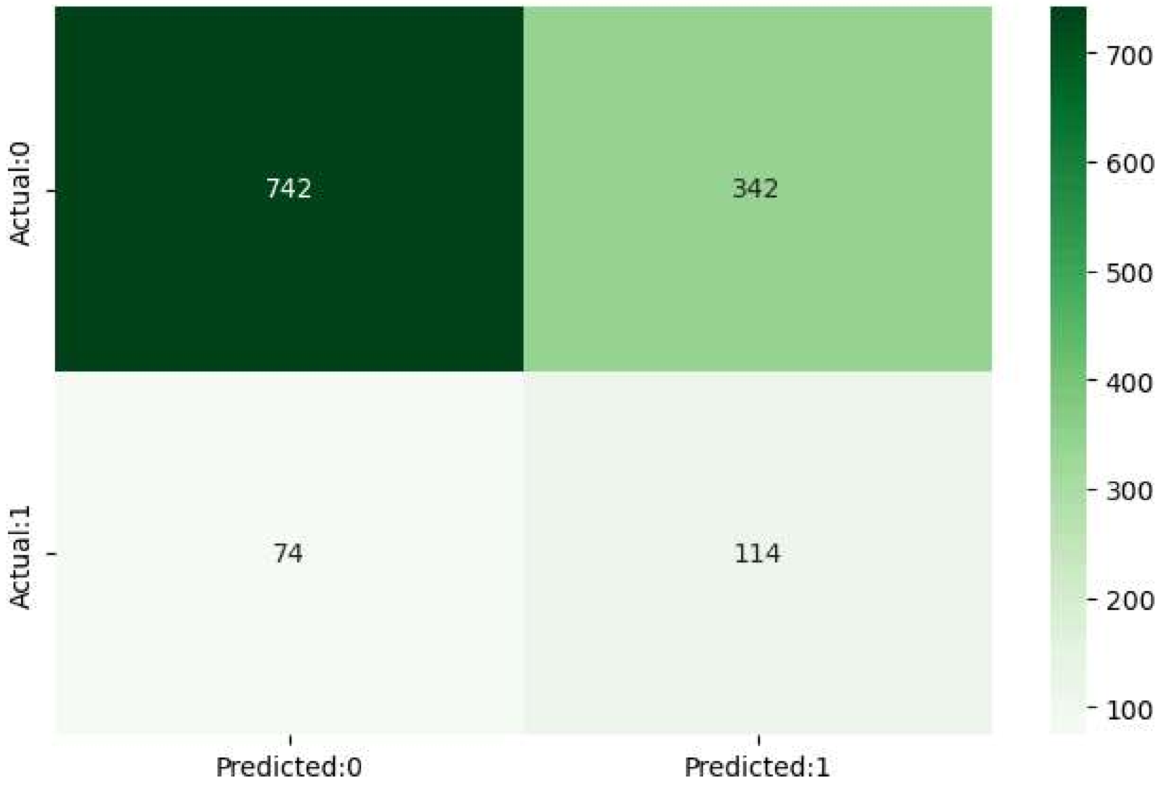
In [121]:

```
# Train the SVC model with class weights
svc = SVC(kernel='rbf', class_weight='balanced', probability=True, random_s
svc.fit(X_resampled, y_resampled)

# Make predictions and evaluate the model
y_pred = svc.predict(X_test)

# Evaluate performance
roc_svc_rbf = round(roc_auc_score(y_test, svc.predict_proba(X_test)[:, 1])*
score_svc_rbf = round(accuracy_score(y_test,y_pred)*100,2)


print(classification_report(y_test, y_pred))
print("Acurracy of the model is :",score_svc_rbf)
print("ROC-AUC Score:",roc_svc_rbf)
```

```
              precision    recall  f1-score   support

           0       0.91      0.68      0.78      1084
           1       0.25      0.61      0.35       188

    accuracy                           0.67      1272
   macro avg       0.58      0.65      0.57      1272
weighted avg       0.81      0.67      0.72      1272

Acurracy of the model is : 67.3
ROC-AUC Score: 69.62
```

In [122]:

```
# Confusion matrix
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                           columns = ['Predicted:0', 'Predicted:1'],
                           index =['Actual:0', 'Actual:1'])

plt.figure(figsize = (8, 5))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")

plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```



```
The details for confusion matrix is =
              precision    recall  f1-score   support

           0       0.91      0.68      0.78      1084
           1       0.25      0.61      0.35       188

    accuracy                           0.67      1272
   macro avg       0.58      0.65      0.57      1272
weighted avg       0.81      0.67      0.72      1272
```

Fitting SVC while using linear kernel for Heart Disease Prediction

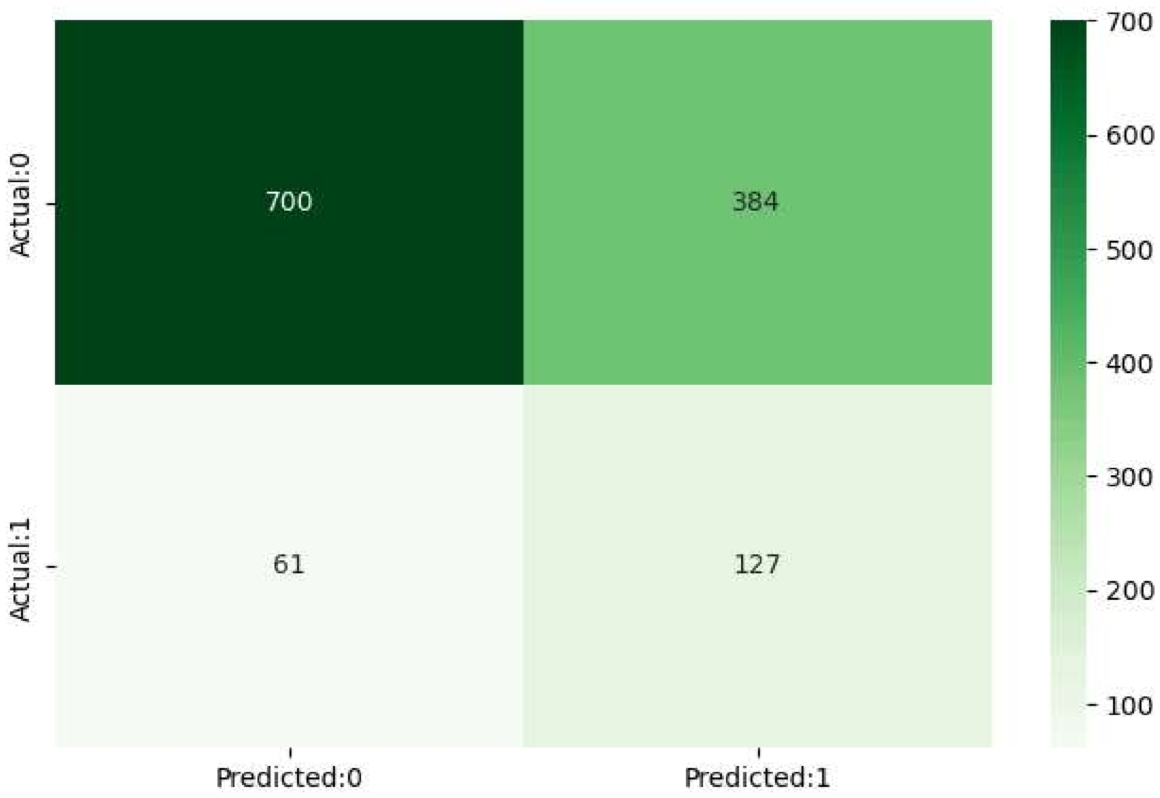
In [123]:

```
# Train the model
linear_svc_model=SVC(kernel="linear", class_weight="balanced", probability=
linear_svc_model.fit(X_resampled,y_resampled)

# Make predictions and evaluate the model
y_pred = linear_svc_model.predict(X_test)

# Evaluate the model
score_linearsvc = round(accuracy_score(y_test,y_pred)*100,2)
roc_linearsvc = round(roc_auc_score(y_test, linear_svc_model.predict_proba(
print(classification_report(y_test, y_pred))
print("Acurracy of the model is :",score_linearsvc)
print("ROC-AUC Score:", roc_linearsvc)
```

```
              precision    recall  f1-score   support

           0       0.92      0.65      0.76      1084
           1       0.25      0.68      0.36       188

    accuracy                           0.65      1272
   macro avg       0.58      0.66      0.56      1272
weighted avg       0.82      0.65      0.70      1272

Acurracy of the model is : 65.02
ROC-AUC Score: 71.87
```

In [124]:

```
# Confusion matrix
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                           columns = ['Predicted:0', 'Predicted:1'],
                           index =['Actual:0', 'Actual:1'])

plt.figure(figsize = (8, 5))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")

plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```



```
The details for confusion matrix is =
              precision    recall  f1-score   support

           0       0.92      0.65      0.76      1084
           1       0.25      0.68      0.36       188

    accuracy                           0.65      1272
   macro avg       0.58      0.66      0.56      1272
weighted avg       0.82      0.65      0.70      1272
```

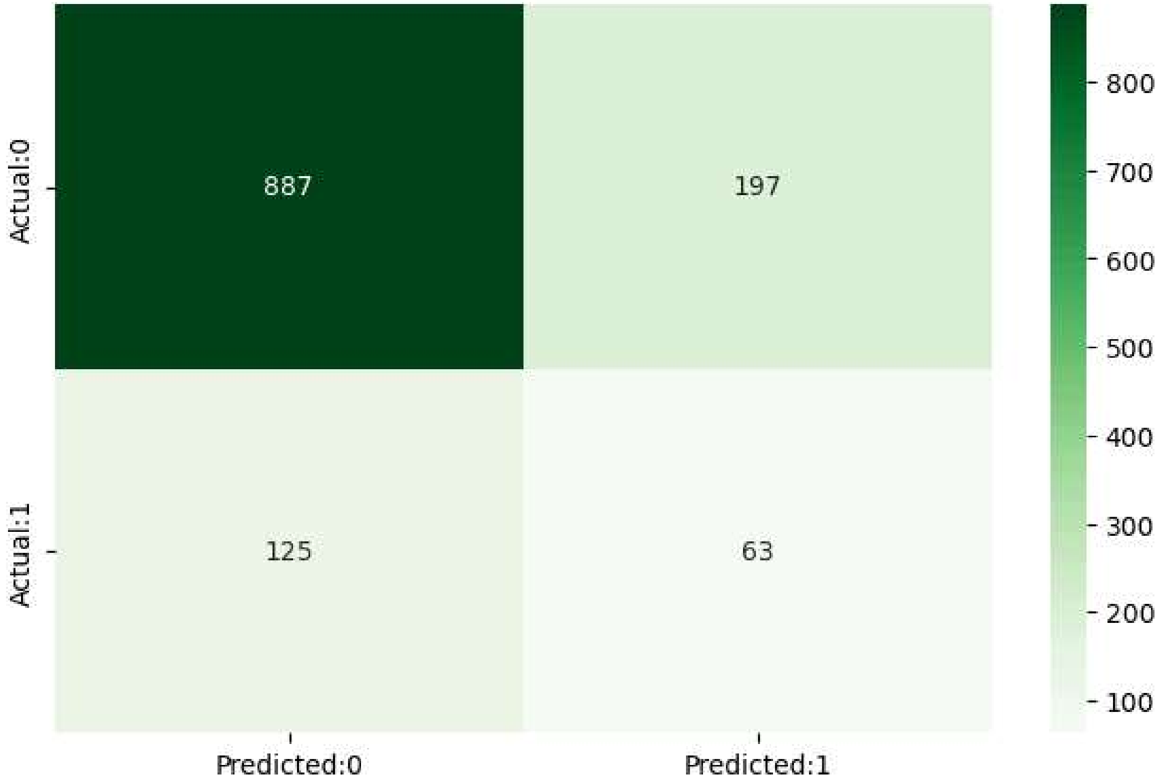
In [131]:

```
# Train the model
poly_svc_model=SVC(kernel="poly", degree=4, coef0=1, class_weight="balanced
poly_svc_model.fit(X_resampled,y_resampled)

# Make predictions and evaluate the model
y_train_predict =poly_svc_model.predict(X_resampled)
y_pred = poly_svc_model.predict(X_test)

# Evaluate the model
score_polysvc = round(accuracy_score(y_test,y_pred)*100,2)
roc_polysvc = round(roc_auc_score(y_test,poly_svc_model.predict_proba(X_tes

print(classification_report(y_test, y_pred))
print("Acurracy of the model is :",score_polysvc)
print("ROC-AUC Score:", roc_polysvc)
print("Acurracy of the model on training is :",accuracy_score(y_resampled,y
```

```
              precision    recall  f1-score   support

           0       0.90      0.69      0.78      1084
           1       0.23      0.54      0.32       188

    accuracy                           0.67      1272
   macro avg       0.56      0.61      0.55      1272
weighted avg       0.80      0.67      0.71      1272

Acurracy of the model is : 66.9
ROC-AUC Score: 65.18
Acurracy of the model on training is : 0.7231289808917197
```

In [109]:

```
# Confusion matrix
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                           columns = ['Predicted:0', 'Predicted:1'],
                           index =['Actual:0', 'Actual:1'])

plt.figure(figsize = (8, 5))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")

plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```



```
The details for confusion matrix is =
              precision    recall  f1-score   support

           0       0.88      0.82      0.85      1084
           1       0.24      0.34      0.28       188

    accuracy                           0.75      1272
   macro avg       0.56      0.58      0.56      1272
weighted avg       0.78      0.75      0.76      1272
```

Final Output

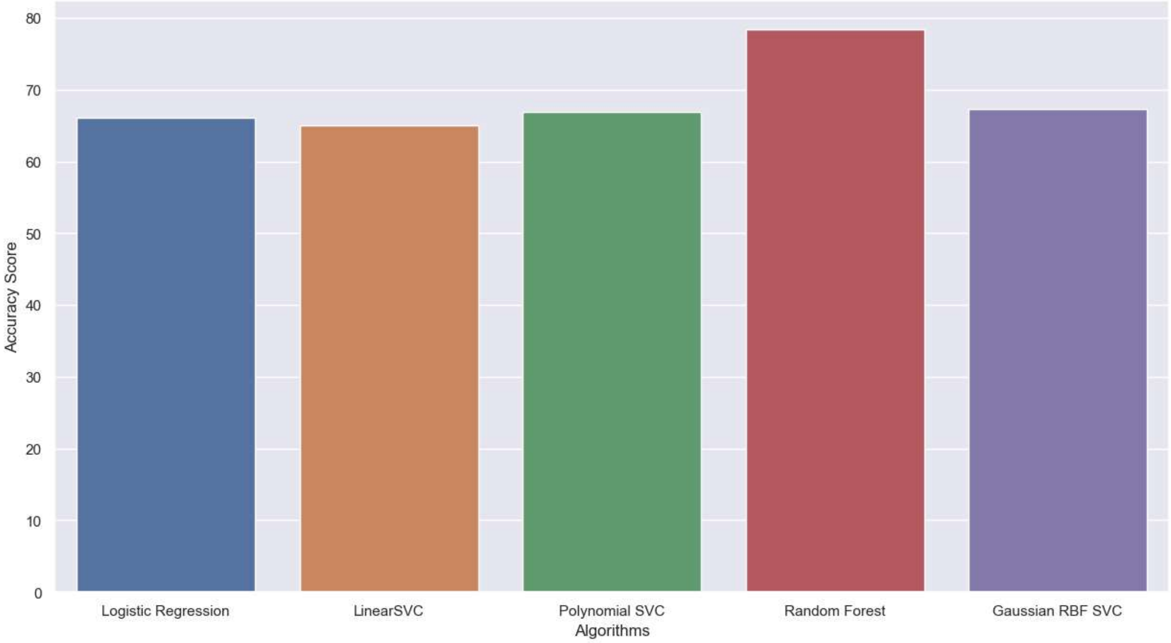
In [142]:

```
scores= [score_lr,score_linearsvc,score_polysvc,score_rf,score_svc_rbf]
rocs = [roc_lr,roc_linearsvc,roc_polysvc,roc_rf,roc_svc_rbf]
algorithms = ["Logistic Regression","LinearSVC","Polynomial SVC","Random Fo

for i in range(len(algorithms)):
    print(f"The Accuracy and ROC score achieved using {algorithms[i]} is: {
```

```
The Accuracy and ROC score achieved using Logistic Regression is: 66.04%
and 72.07%
The Accuracy and ROC score achieved using LinearSVC is: 65.02% and 71.87%
The Accuracy and ROC score achieved using Polynomial SVC is: 66.9% and 6
5.18%
The Accuracy and ROC score achieved using Random Forest is: 78.46% and 6
8.62%
The Accuracy and ROC score achieved using Gaussian RBF SVC is: 67.3% and
69.62%
```

In [145]:

```
sns.set(rc={'figure.figsize': (15, 8)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy Score")

# Explicitly specify x and y
sns.barplot(x=algorithms, y=scores)
plt.show()
```



In [ ]: