In [

6]:

```
df.columns
```

Out[6]:

```
Index(['ID', 'Age', 'Experience', 'Income', 'ZIP Code', 'Family', 'CCAvg',
       'Education', 'Mortgage', 'Personal Loan', 'Securities Account',
       'CD Account', 'Online', 'CreditCard'],
      dtype='object')
```

In [7]:

```
X = df.drop(['CreditCard'], axis=1)

y = df['CreditCard']
```

In [8]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0
```

In [9]:

```
X_train.shape, X_test.shape
```

Out[9]:

```
((4000, 13), (1000, 13))
```

In [10]:

```
cols = X_train.columns
```

In [11]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

In [12]:

```
X_train = pd.DataFrame(X_train, columns=[cols])
```

In [13]:

```
X_test = pd.DataFrame(X_test, columns=[cols])
```

In [  ]:

14

```
X_train.describe()
```

Out[14]:

|  | ID | Age | Experience | Income | ZIP Code | Famil |
|---|---|---|---|---|---|---|
| **count** | 4.000000e+03 | 4.000000e+03 | 4.000000e+03 | 4.000000e+03 | 4.000000e+03 | 4.000000e+0 |
| **mean** | 1.047148e-16 | 8.820722e-17 | -9.203749e-17 | -4.454770e-17 | 2.580948e-15 | -3.212985e-1 |
| **std** | 1.000125e+00 | 1.000125e+00 | 1.000125e+00 | 1.000125e+00 | 1.000125e+00 | 1.000125e+0 |
| **min** | -1.732418e+00 | -1.941993e+00 | -2.006748e+00 | -1.425882e+00 | -3.807548e+01 | -1.212103e+0 |
| **25%** | -8.646705e-01 | -8.975483e-01 | -8.768221e-01 | -7.560813e-01 | -5.463612e-01 | -1.212103e+0 |
| **50%** | -1.323063e-02 | -2.717733e-02 | -7.648726e-03 | -2.159191e-01 | 1.194385e-01 | -3.433677e-0 |
| **75%** | 8.637108e-01 | 8.431936e-01 | 8.615247e-01 | 5.403081e-01 | 6.648856e-01 | 5.253678e-0 |
| **max** | 1.736489e+00 | 1.887639e+00 | 1.991450e+00 | 3.241119e+00 | 1.592736e+00 | 1.394103e+0 |

In [15]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

svc=SVC()
svc.fit(X_train,y_train)

y_pred=svc.predict(X_test)

print('Model accuracy : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy : 0.7490
```

In [16]:

```
svc=SVC(C=100.0)
svc.fit(X_train,y_train)

y_pred=svc.predict(X_test)

print('C=100.0 Model accuracy with rbf kernel : {0:0.4f}'. format(accuracy_score(y_test, y_
```

```
C=100.0 Model accuracy with rbf kernel : 0.6970
```

In [2 ]:

17

```
linear_svc=SVC(kernel='linear', C=1.0)
linear_svc.fit(X_train,y_train)

y_pred_test=linear_svc.predict(X_test)

print('C=1.0 Model accuracy with linear kernel : {0:0.4f}'. format(accuracy_score(y_test, y
```

```
C=1.0 Model accuracy with linear kernel : 0.7470
```

In [18]:

```
linear_svc=SVC(kernel='linear', C=10.0)
linear_svc.fit(X_train,y_train)

y_pred_test=linear_svc.predict(X_test)

print('C=10.0 Model accuracy with linear kernel : {0:0.4f}'. format(accuracy_score(y_test,
```

```
C=10.0 Model accuracy with linear kernel : 0.7470
```

In [19]:

```
linear_svc=SVC(kernel='linear', C=100.0)
linear_svc.fit(X_train,y_train)

y_pred_test=linear_svc.predict(X_test)

print('C=100.0 Model accuracy with linear kernel : {0:0.4f}'. format(accuracy_score(y_test,
```

```
C=100.0 Model accuracy with linear kernel : 0.7470
```

In [2 ]:

```
    0
```

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_test)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[696   8]
 [245  51]]

True Positives(TP) =  696

True Negatives(TN) =  51

False Positives(FP) =  8

False Negatives(FN) =  245
```

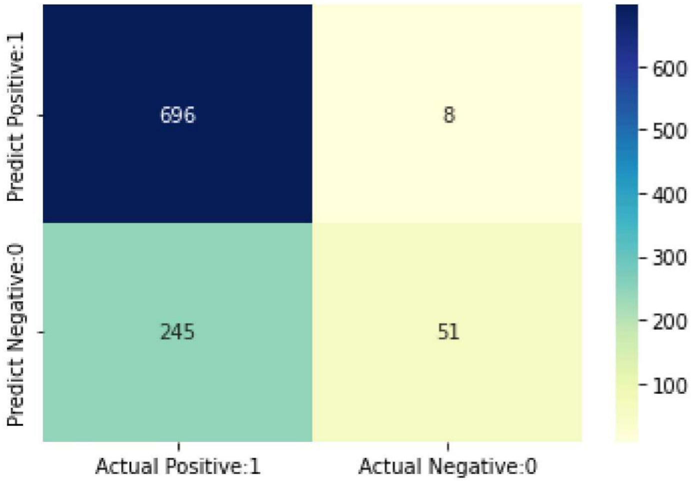
In [21]:

```
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:0'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

Out[21]:

```
<AxesSubplot:>
```



In [2 ]:

2

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.74      0.99      0.85       704
           1       0.86      0.17      0.29       296

    accuracy                           0.75      1000
   macro avg       0.80      0.58      0.57      1000
weighted avg       0.78      0.75      0.68      1000
```

In [2 ]:

In [23]:

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score


kfold=KFold(n_splits=5, shuffle=True, random_state=0)


linear_svc=SVC(kernel='linear')


linear_scores = cross_val_score(linear_svc, X, y, cv=kfold)
```

In [24]:

```
print('Cross-validation with linear kernel:\n\n{}'.format(linear_scores))
```

```
Cross-validation with linear kernel:

[0.704 0.704 0.721 0.686 0.716]
```

In [25]:

```
print('Average cross-validation with linear kernel:{:.4f}'.format(linear_scores.mean()))
```

```
Average cross-validation with linear kernel:0.7062
```

In [26]:

```
rbf_svc=SVC(kernel='rbf')

rbf_scores = cross_val_score(rbf_svc, X, y, cv=kfold)
```

In [27]:

```
print('Cross-validation with rbf kernel:\n\n{}'.format(rbf_scores))
```

```
Cross-validation with rbf kernel:

[0.704 0.703 0.721 0.686 0.716]
```

In [2 ]:

```
    8
```

```
print('Average cross-validation with rbf kernel:{:.4f}'.format(rbf_scores.mean()))
```

```
Average cross-validation with rbf kernel:0.7060
```

In [29]:

```
from sklearn.model_selection import GridSearchCV

from sklearn.svm import SVC

svc=SVC()


parameters = [ {'C':[1, 10, 50], 'kernel':['linear']},
               {'C':[1, 10, 50], 'kernel':['rbf'], 'gamma':[0.1, 0.2, 0.5, 0.9]}
              ]




grid_search = GridSearchCV(estimator = svc,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           verbose=0)


grid_search.fit(X_train, y_train)
```

Out[29]:

```
GridSearchCV(cv=5, estimator=SVC(),
             param_grid=[{'C': [1, 10, 50], 'kernel': ['linear']},
                         {'C': [1, 10, 50], 'gamma': [0.1, 0.2, 0.5, 0.9],
                          'kernel': ['rbf']}],
             scoring='accuracy')
```

In [30]:

```
print('GridSearch CV best: {:.4f}\n\n'.format(grid_search.best_score_))


print('Best results :','\n\n', (grid_search.best_params_))
```

```
GridSearch CV best: 0.7435


Best results :

 {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
```