In [149]:

```
#IMPORTING PYTHON LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
```

In [150]:

```
#TO CHANGE OR TO CHOOSE THE DIRECTORY/PATH OF THE DATASET.
import os
#os.chdir()
data=pd.read_csv("diabetes.csv")
data
```

Out [150]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [151]:

```
#TO COPY FILE FROM ONE VBARIABLE TO ANOTHER VARIABLE USE COPY().
d1=data.copy()
d1.head()
```

Out [151]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [152]:

```
#TO KNOW ABOUT THE INFORMATION OF THE DATASET
d1.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 [153]:

```
#TO KNOW ABOUT THE DESCRIPTION OF THE DATASET
d1.describe()
```

Out [153]:

	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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

In [154]:

```
#TO CHECK WHETHER NULL VALUES ARE PRESENT IN THE DATASET OR NOT
d1.isna().sum()
```

Out [154]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [155]:

```
#TO KNOW ABOUT THE TOTAL COUNT OF ELEMENTS PRESENT IN THE COLUMNS.
d1.count()
```

Out [155]:

```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

In [156]:

```
#TO PRINT SPECIFIED NUMBER OF ELEMENTS OF THE DATASET FROM 1ST TO LAST.
#by default the count will be 5
d1.head()
```

Out [156]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [157]:

```
#TO PRINT SPECIFIED NUMBER OF ELEMENTS OF THE DATASET FROM LAST TO 1ST.
#by default the count will be 5
d1.tail()
```

Out [157]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

AS THERE IS NO NULL VALUES IN THE DATASET, THERE WILL BE NO NEED OF PREPROCESSING.

AS ALSO WE KNOW TO PREPROCESS THE DATA WE CAN USE TO DO,

1.ANALYZE THE DATASET AND CHECK FOR NULL VALUES.

2.IF NULL VALUE PRESENTS THAN:

```
    1.REPLACE NULL VALUES WITH 0.
    2.REPLACE NULL VALUES WITH PADDING[METHOD='PAD'] OR BACKWARD FILLING [METHOD='BFILL']
    3.ELSE WE CAN REPLACE NULL VALUES WITH MEAN/MODE/MEDIAN.
```

In [158]:

```
#TO KNOW THE SIZE OF THE DATASET
d1.shape
```

Out [158]:

```
(768, 9)
```

In [159]:

```
#TO KNOW THE SIZE OF THE DATASET
d1.size
```

Out [159]:
```
6912
```

In [160]:
```
d1.columns
```

Out [160]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

CONSIDERING VARIABLES

In [161]:
```
col=d1[['Outcome']]
col
x=d1.drop(col,axis=1)
x.head()
```

Out [161]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [162]:
```
y=d1['Outcome']
y.head()
```

Out [162]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

In [163]:
```
x.shape
```

Out [163]:
```
(768, 8)
```

In [164]:
```
y.shape
```

Out [164]:
```
(768,)
```

MODEL BUILDING

In [165]:
```
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.3,random_state=1)
```

In [166]:
```
X_train.shape
```

Out [166]:
```
(537, 8)
```

In [167]:
```
X_test.shape
```

Out [167]:
```
(231, 8)
```

In [168]:
```
Y_train.shape
```

Out [168]:
```
(537,)
```

In [169]:
```
Y_test.shape
```

Out [169]:
```
(231,)
```

KNN(K-NEAREST NEIGHBOR) CLASSIFIER

In [170]:
```
knn=KNeighborsClassifier()
model=knn.fit(X_train,Y_train)
Y_pred1=model.predict(X_test)
Y_pred1
```

Out [170]:
```
array([1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

In [171]:

```
#Evaluation_Metrics
value_KNN=metrics.classification_report(Y_test,Y_pred1,digits=4)
print(value_KNN)
```

```
              precision    recall  f1-score   support

           0     0.7853    0.8767    0.8285       146
           1     0.7353    0.5882    0.6536        85

    accuracy                         0.7706       231
   macro avg     0.7603    0.7325    0.7410       231
weighted avg     0.7669    0.7706    0.7641       231
```

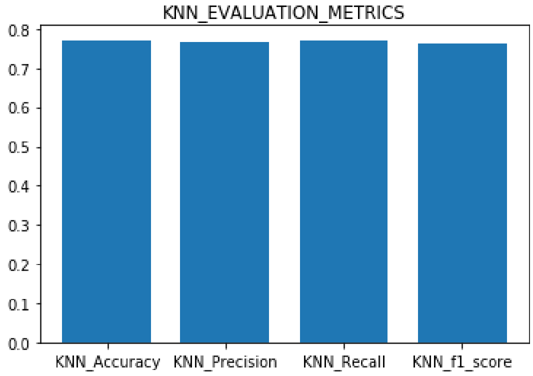
In [172]:

```
#CONSIDER WEIGHTED METRICS
KNN_Accuracy=0.7706
KNN_Precision=0.7669
KNN_Recall=0.7706
KNN_f1_score=0.7641
```

accuracy_KNN=(metrics.accuracy_score(Y_test,Y_pred1)) print("accuracy_KNN :",accuracy_KNN) precision_KNN=(metrics.precision_score(Y_test,Y_pred1)) print("precision_score :",precision_KNN) recall_KNN=(metrics.recall_score(Y_test,Y_pred1)) print("recall_score :",recall_KNN) f1_score_KNN=(metrics.f1_score(Y_test,Y_pred1)) print("f1_score :",f1_score_KNN)

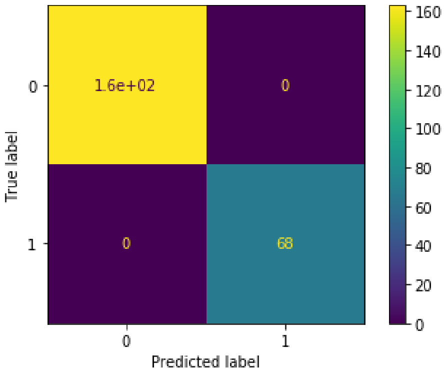
In [173]:

```
#PLOTTING
import matplotlib.pyplot as plt
x=['KNN_Accuracy','KNN_Precision','KNN_Recall','KNN_f1_score']
y=[KNN_Accuracy,KNN_Precision,KNN_Recall,KNN_f1_score]
plt.title("KNN_EVALUATION_METRICS")
plt.bar(x,y,width=0.75)
plt.show()
```



In [174]:

```
#CONFUSION_MATRIX
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(knn,X_test,Y_pred1)
plt.show()
```



DECISION TREE CLASSIFIER

In [191]:

```
DT=DecisionTreeClassifier()
model=DT.fit(X_train,Y_train)
Y_pred2=model.predict(X_test)
Y_pred2
```

Out [191]:

```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [192]:

```
#Evaluation_Metrics
value_DT=metrics.classification_report(Y_test,Y_pred2,digits=4)
print(value_DT)
```

```
              precision    recall  f1-score   support

           0     0.7500    0.8014    0.7748       146
           1     0.6133    0.5412    0.5750        85

    accuracy                         0.7056       231
   macro avg     0.6817    0.6713    0.6749       231
weighted avg     0.6997    0.7056    0.7013       231
```

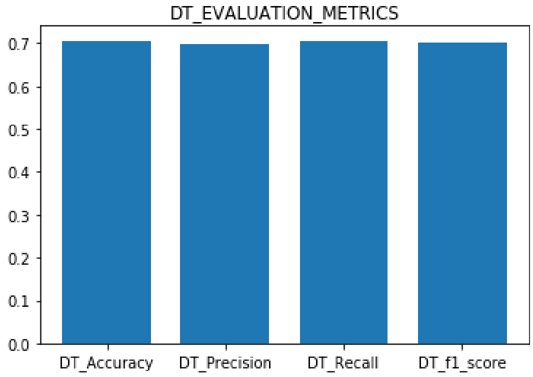
In [193]:

```
#CONSIDER WEIGHTED Avg METRICS
DT_Accuracy=0.7056
DT_Precision=0.6997
DT_Recall=0.7056
DT_f1_score=0.7013
```

accuracy_DT=(metrics.accuracy_score(Y_test,Y_pred2)) print("accuracy_DT :",accuracy_DT) precision_DT=(metrics.precision_score(Y_test,Y_pred2)) print("precision_score :",precision_DT) recall_DT=(metrics.recall_score(Y_test,Y_pred2)) print("recall_score :",recall_DT) f1_score_DT=(metrics.f1_score(Y_test,Y_pred2)) print("f1_score :",f1_score_DT)

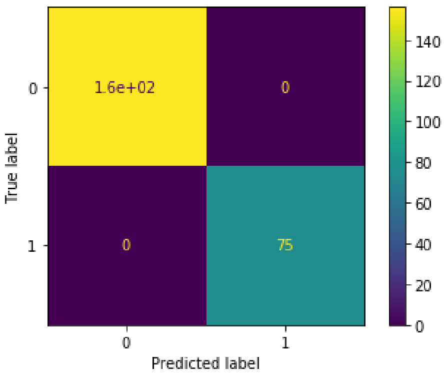
In [194]:

```
#PLOTTING
import matplotlib.pyplot as plt
x=['DT_Accuracy','DT_Precision','DT_Recall','DT_f1_score']
y=[DT_Accuracy,DT_Precision,DT_Recall,DT_f1_score]
plt.title("DT_EVALUATION_METRICS")
plt.bar(x,y,width=0.75)
plt.show()
```



In [195]:

```
#CONFUSION_MATRIX
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(DT,X_test,Y_pred2)
plt.show()
```



SVM(Support Vector Machine)

In [198]:

```
from sklearn.svm import SVC
svm=SVC(kernel='linear')
```

```
model=svm.fit(X_train,Y_train)
Y_pred3=model.predict(X_test)
Y_pred3
```

```
Out [198]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
                  1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
                  0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
                  0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
                  0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
                  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
                  0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                  1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
                  1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                  0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                  0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

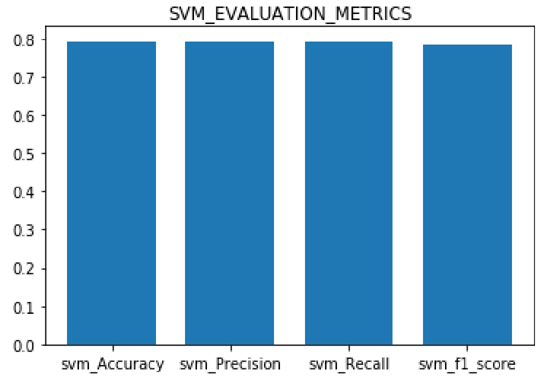
```
In [199]:
#Evaluation_Metrics
value_svm=metrics.classification_report(Y_test,Y_pred3,digits=4)
print(value_svm)
```

```
              precision    recall  f1-score   support

           0     0.7917    0.9110    0.8471       146
           1     0.7937    0.5882    0.6757        85

    accuracy                         0.7922       231
   macro avg     0.7927    0.7496    0.7614       231
weighted avg     0.7924    0.7922    0.7840       231
```

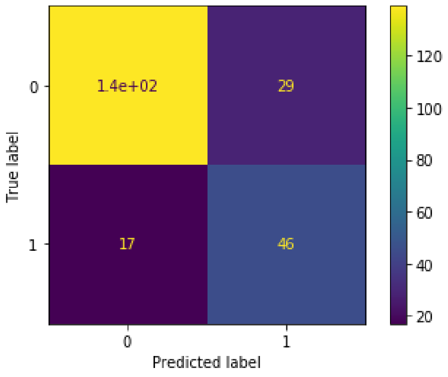
```
In [200]:
#CONSIDER WEIGHTED Avg METRICS
svm_Accuracy=0.7922
svm_Precision=0.7924
svm_Recall=0.7922
svm_f1_score=0.7840
```

accuracy_svm=(metrics.accuracy_score(Y_test,Y_pred3)) print("accuracy_svm :",accuracy_svm) precision_svm=(metrics.precision_score(Y_test,Y_pred3)) print("precision_score :",precision_svm) recall_svm=(metrics.recall_score(Y_test,Y_pred3)) print("recall_score :",recall_svm) f1_score_svm=(metrics.f1_score(Y_test,Y_pred3)) print("f1_score :",f1_score_svm)

```
In [201]:
#PLOTTING
import matplotlib.pyplot as plt
x=['svm_Accuracy','svm_Precision','svm_Recall','svm_f1_score']
y=[svm_Accuracy,svm_Precision,svm_Recall,svm_f1_score]
plt.title("SVM_EVALUATION_METRICS")
plt.bar(x,y,width=0.75)
plt.show()
```



```
In [296]:
#CONFUSION_MATRIX
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(DT,X_test,Y_pred3)
plt.show()
```



RANDOM FOREST CLASSIFIER

In [239]:

```
#RANDOM_FOREST
from sklearn.preprocessing import StandardScaler #for preprocessing
std=StandardScaler()
x_train=std.fit_transform(X_train)
x_test=std.fit_transform(X_test)
```

In [286]:

```
#fitting
from sklearn.ensemble import RandomForestClassifier
#n_estimators - the required no.of trees in the random forest ...bydefault n=10
classifier1=RandomForestClassifier(n_estimators=10,criterion="entropy")
classifier1.fit(x_train,Y_train)
```

Out [286]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [287]:

```
#predicting
Y_pred4=classifier1.predict(x_test)
Y_pred4
```

Out [287]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [288]:

```
#Evaluation_Metrics
value_RF=metrics.classification_report(Y_test,Y_pred4,digits=4)
print(value_RF)
```

```
              precision    recall  f1-score   support

           0     0.7836    0.9178    0.8454       146
           1     0.8000    0.5647    0.6621        85

    accuracy                         0.7879       231
   macro avg     0.7918    0.7413    0.7537       231
weighted avg     0.7897    0.7879    0.7780       231
```

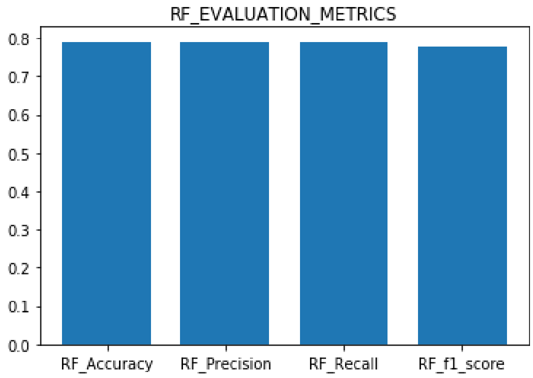
In [289]:

```
#CONSIDER WEIGHTED Avg METRICS
RF_Accuracy=0.7879
RF_Precision=0.7897
RF_Recall=0.7879
RF_f1_score=0.7780
```

accuracy_RF=(metrics.accuracy_score(Y_test,Y_pred4)) print("accuracy_RF :",accuracy_RF) precision_RF=(metrics.precision_score(Y_test,Y_pred4)) print("precision_score :",precision_RF) recall_RF=(metrics.recall_score(Y_test,Y_pred4)) print("recall_score :",recall_RF) f1_score_RF=(metrics.f1_score(Y_test,Y_pred4)) print("f1_score :",f1_score_RF)

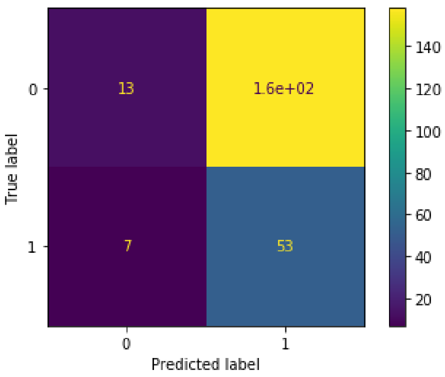
In [294]:

```
#PLOTTING
import matplotlib.pyplot as plt
x=['RF_Accuracy','RF_Precision','RF_Recall','RF_f1_score']
y=[RF_Accuracy,RF_Precision,RF_Recall,RF_f1_score]
plt.title("RF_EVALUATION_METRICS")
plt.bar(x,y,width=0.75)
plt.show()
```



In [298]:

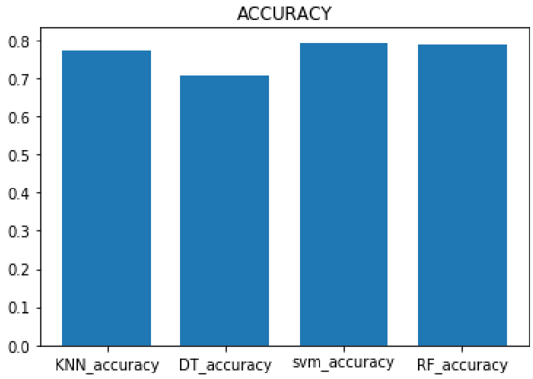
```
#CONFUSION_MATRIX
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier1,X_test,Y_pred4)
plt.show()
```



PLOTTINGS (comparision)

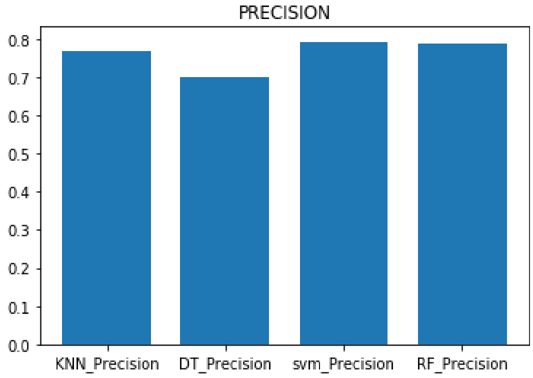
In [301]:

```
#PLOTTING
import matplotlib.pyplot as plt
x=['KNN_accuracy','DT_accuracy','svm_accuracy','RF_accuracy']
y=[KNN_Accuracy,DT_Accuracy,svm_Accuracy,RF_Accuracy]
plt.title("ACCURACY")
plt.bar(x,y,width=0.75)
plt.show()
```



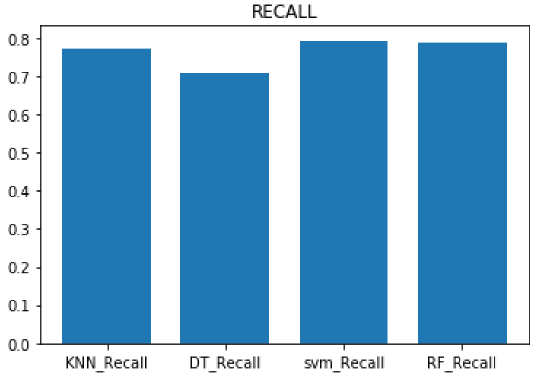
In [302]:

```
#PLOTTING
import matplotlib.pyplot as plt
x=['KNN_Precision','DT_Precision','svm_Precision','RF_Precision']
y=[KNN_Precision,DT_Precision,svm_Precision,RF_Precision]
plt.title("PRECISION")
plt.bar(x,y,width=0.75)
plt.show()
```



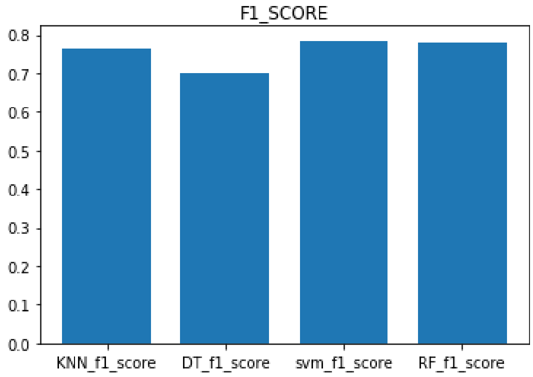
In [303]:

```
#PLOTTING
import matplotlib.pyplot as plt
x=['KNN_Recall','DT_Recall','svm_Recall','RF_Recall']
y=[KNN_Recall,DT_Recall,svm_Recall,RF_Recall]
plt.title("RECALL")
plt.bar(x,y,width=0.75)
plt.show()
```



In [305]:

```
#PLOTTING
import matplotlib.pyplot as plt
x=['KNN_f1_score','DT_f1_score','svm_f1_score','RF_f1_score']
y=[KNN_f1_score,DT_f1_score,svm_f1_score,RF_f1_score]
plt.title("F1_SCORE")
plt.bar(x,y,width=0.75)
plt.show()
```



In [ ]: