In [47]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv(r'Documents\Machine-Learning-with-Python-master\diabetes.csv')
df.head()
```

Out[47]:

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

In [2]:

```python
print("shape of data set is {}".format(df.shape))
```

```
shape of data set is (768, 9)
```

In [3]:

```python
df.groupby('Outcome').size()
```

Out[3]:

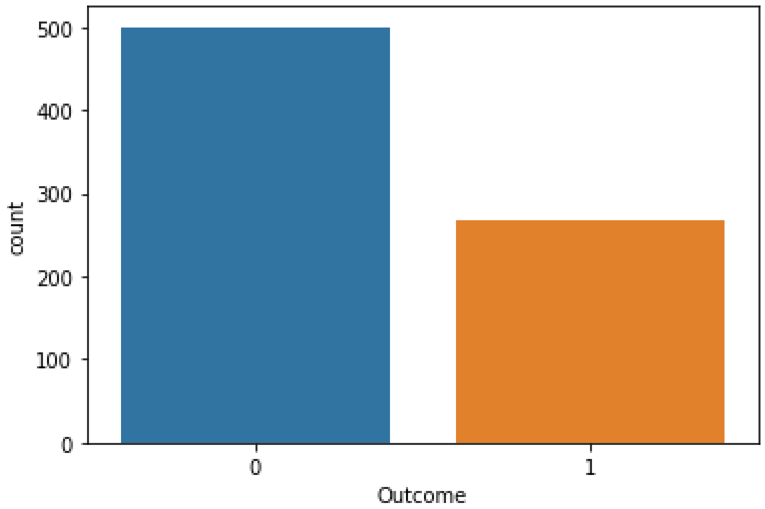
```
Outcome
0    500
1    268
dtype: int64
```

In [26]:

```python
import seaborn as sns
sns.countplot(df['Outcome'],label='count')
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f665eaf748>
```



In [5]:

```
df.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesP
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [21]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
BloodPressure                  768 non-null int64
SkinThickness                  768 non-null int64
Insulin                        768 non-null int64
BMI                            768 non-null float64
DiabetesPedigreeFunction       768 non-null float64
Age                            768 non-null int64
Outcome                        768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [82]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(df.loc[:,df.columns!='O
utcome'],df['Outcome'],stratify=df['Outcome'],random_state=66)
x_train.head()
```

Out[82]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
612	7	168	88	42	321	38.2	0.787
557	8	110	76	0	0	27.8	0.237
26	7	147	76	0	0	39.4	0.257
70	2	100	66	20	90	32.9	0.867
73	4	129	86	20	270	35.1	0.231

In [8]:

```
x_test.head()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
506	0	180	90	26	90	36.5	0.314
709	2	93	64	32	160	38.0	0.674
257	2	114	68	22	0	28.7	0.092
518	13	76	60	0	0	32.8	0.180
432	1	80	74	11	60	30.0	0.527

In [9]:

```
y_train.head()
```

Out[9]:

```
612    1
557    0
26     1
70     1
73     0
```

```
Name: Outcome, dtype: int64
```

In [10]:
```python
y_test.head()
```

Out[10]:
```
506    1
709    1
257    0
518    0
432    0
Name: Outcome, dtype: int64
```

In [83]:
```python
# K-nearest_neighbors classifer
list_of_training_accuracy = []
list_of_testing_accuracy = []
from sklearn.neighbors import KNeighborsClassifier
training_accuracy = []
testing_accuracy = []
neighbors = list(range(1,10))
for no_of_neighbors in neighbors:
    kn = KNeighborsClassifier(n_neighbors=no_of_neighbors).fit(x_train,y
_train)
    y_pre = kn.predict(x_test)
    print("predicted values of k = {} is ".format(no_of_neighbors))
    print(y_pre)
```

```
predicted values of k = 1 is
[1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0
0
 0 0 1 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0
0
 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 1 1 1 0 0 1
1
 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 1 0 0
0
 1 1 1 0 1 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 1 1 0 1 1 0 0 1 0 0 1 1 0 0 1 1
0
 0 0 1 0 0 0 0]
predicted values of k = 2 is
[1 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 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 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
1
 0 1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0
0
```

```
 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1
0
 0 0 1 0 0 0 0]
predicted values of k = 3 is
[1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 0 0 1 1 0 0 0 1
0
 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
0
 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0
1
 0 1 0 0 1 0 1 0 1 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 0 0
0
 0 1 1 0 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1
0
 1 0 1 0 1 0 0]
predicted values of k = 4 is
[1 0 0 0 0 1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
0
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0
0
 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 0
0
 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1
0
 1 0 1 0 1 0 0]
predicted values of k = 5 is
[1 0 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0
1
 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0
0
 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0
1
 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 0 0
0
 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1
0
 1 0 1 0 1 1 0]
predicted values of k = 6 is
[0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0
 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
0
 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0
1
 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 1 0 0
 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 1 0 0
0
 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1
0
 1 0 1 0 1 1 0]
predicted values of k = 7 is
[1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0
 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0
0
 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0 0 1 1 1 0 0 0 1 1 0 0
1
 1 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 0 0
1

 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1
1
 1 0 1 1 1 1 0]
predicted values of k = 8 is
[1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0
 0 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
0
 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0
0
 1 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0
0
 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1
0
 0 0 1 0 1 0 0]
predicted values of k = 9 is
[1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0
 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
0
 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0
1
 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0
1
 1 1 0 0 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 1
0
 0 0 1 0 1 0 0]
```

In [118]:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
for no_of_neighbors in neighbors:
    kn = KNeighborsClassifier(n_neighbors=no_of_neighbors).fit(x_train,y
_train)
    training_accuracy.append(kn.score(x_train,y_train))
    testing_accuracy.append(kn.score(x_test,y_test))
print("list of training accuracy of diffrent k values models")
print(training_accuracy)
print('\n')
print('list of testing accuracy of diffrent k values models')
print(testing_accuracy)
```

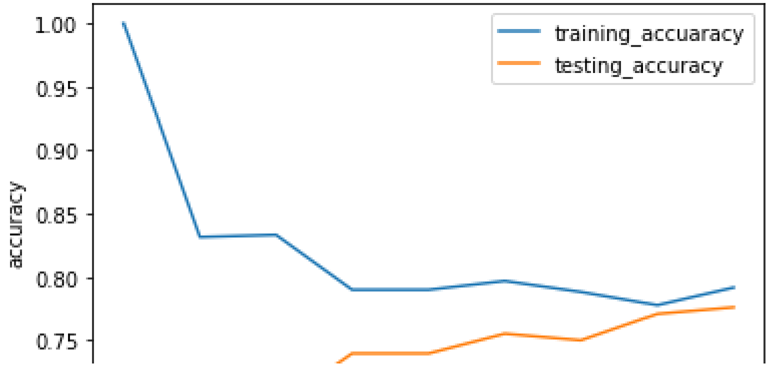
```
list of training accuracy of diffrent k values models
[1.0, 0.8315972222222222, 0.8333333333333334, 0.7899305555555556, 0.7899
305555555556, 0.796875, 0.7881944444444444, 0.7777777777777778, 0.791666
6666666666, 1.0, 0.8315972222222222, 0.8333333333333334, 0.7899305555555

556, 0.7899305555555556, 0.796875, 0.7881944444444444, 0.777777777777777
8, 0.7916666666666666, 1.0, 0.8315972222222222, 0.8333333333333334, 0.78
99305555555556, 0.7899305555555556, 0.796875, 0.7881944444444444, 0.7777
777777777778, 0.7916666666666666]


list of testing accuracy of diffrent k values models
[0.6875, 0.7239583333333334, 0.6979166666666666, 0.7395833333333334, 0.7
395833333333334, 0.7552083333333334, 0.75, 0.7708333333333334, 0.7760416
666666666, 0.6875, 0.7239583333333334, 0.6979166666666666, 0.73958333333
33334, 0.7395833333333334, 0.7552083333333334, 0.75, 0.7708333333333334,
0.7760416666666666, 0.6875, 0.7239583333333334, 0.6979166666666666, 0.73
95833333333334, 0.7395833333333334, 0.7552083333333334, 0.75, 0.77083333
33333334, 0.7760416666666666]
```

In [85]:

```
plt.plot(neighbors,training_accuracy,label="training_accuaracy")
plt.plot(neighbors,testing_accuracy,label="testing_accuracy")
plt.ylabel('accuracy')
plt.xlabel('neighbors')
plt.legend()
plt.savefig('knn_accuracy_comapare_model')
```



In [122]:

```
# from the above figure we can say that we have to neighbor some where a
round 9
kn = KNeighborsClassifier(n_neighbors = 9).fit(x_train,y_train)
list_of_training_accuracy.append(kn.score(x_train,y_train))
list_of_testing_accuracy.append(kn.score(x_test,y_test))
print("Accuracy of the K-Neares_neighbors_classifer on training set is "
,kn.score(x_train,y_train))
print("Accuracy of the K-Neares_neighbors_classifer on testing set is ",
kn.score(x_test,y_test))
print()
print("Confusion Matrix ")
print(confusion_matrix(y_test,kn.predict(x_test)))
print()
print("Report")
print(classification_report(y_test,kn.predict(x_test)))
```

```
Accuracy of the K-Neares_neighbors_classifer on training set is  0.79166
66666666666
Accuracy of the K-Neares_neighbors_classifer on testing set is  0.776041
6666666666

Confusion Matrix
[[105  20]
 [ 23  44]]

Report
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       125
           1       0.69      0.66      0.67        67

    accuracy                           0.78       192
   macro avg       0.75      0.75      0.75       192
weighted avg       0.77      0.78      0.77       192
```

In [124]:

```
# logistic Regression
from sklearn.linear_model import LogisticRegression
lgre = LogisticRegression().fit(x_train,y_train)
ypre = lgre.predict(x_test)
```

```
print("predicted values",ypre)
print("test set values",list(y_test))
list_of_training_accuracy.append(lgre.score(x_train,y_train))
list_of_testing_accuracy.append(lgre.score(x_test,y_test))
print()
print("Accuracy of Logistic_Regression on training set is ",lgre.score(x
_train,y_train))
print("Accuracy of Logistic_Regression on testing set is ",lgre.score(x_
test,y_test))
print()
print("Confusion Matrix ")
print(confusion_matrix(y_test,lgre.predict(x_test)))
print()
print("Report")
print(classification_report(y_test,lgre.predict(x_test)))
```

```
predicted values [1 0 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
 1 1 0 1 0 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0

 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
1
 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0
1
 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1
1
 0 0 0 0 1 0 0]
test set values [1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1,
1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
0, 1, 0, 1, 1, 0]

Accuracy of Logistic_Regression on training set is  0.78125
Accuracy of Logistic_Regression on testing set is  0.7708333333333334

Confusion Matrix
[[110  15]
 [ 29  38]]

Report
              precision    recall  f1-score   support
              precision    recall  f1-score   support

           0       0.79      0.88      0.83       125
           1       0.72      0.57      0.63        67

    accuracy                           0.77       192
   macro avg       0.75      0.72      0.73       192
weighted avg       0.77      0.77      0.76       192
```

```
C:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432: Futu
reWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

In [126]:

```
#decision trees

from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth = 3).fit(x_train,y_train)
ypre=dtree.predict(x_test)
print("Predicted values of Outcome using Decison Trees classfier")
print(ypre)
print()
print("actual values of Outcome of the order set")
print(list(y_test))
list_of_training_accuracy.append(dtree.score(x_train,y_train))
list_of_testing_accuracy.append(dtree.score(x_test,y_test))
print()
print("Accuracy of Decision Trees on training set is ",dtree.score(x_tra
in,y_train))
print("Accuracy of Decision Trees on testing set is ",dtree.score(x_test
,y_test))
print()
print("Confusion Matrix ")
print(confusion_matrix(y_test,kn.predict(x_test)))
print()
print("Report")
print(classification_report(y_test,kn.predict(x_test)))
```

```
Predicted values of Outcome using Decison Trees classfier
[1 0 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1
1
 1 0 0 1 0 0 0 1 0 0 1 0 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0
0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 1 0 0 1 0 0
1
 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0
1
 0 0 0 0 1 1 0 1 0 1 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 1 1
0
 0 0 0 0 1 0 0]

actual values of Outcome of the order set
[1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1,
0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0]

Accuracy of Decision Trees on training set is  0.7725694444444444
Accuracy of Decision Trees on testing set is  0.7395833333333334

Confusion Matrix
[[105  20]
 [ 23  44]]

Report
              precision    recall  f1-score   support


           0       0.82      0.84      0.83       125
           1       0.69      0.66      0.67        67

    accuracy                           0.78       192
   macro avg       0.75      0.75      0.75       192
weighted avg       0.77      0.78      0.77       192
```

In [127]:

```
#support vector Machine
from sklearn.svm import SVC
svc = SVC()
svc.fit(x_train, y_train)
print("Accuracy on training set: {:.2f}".format(svc.score(x_train, y_tra
in)))
print("Accuracy on test set: {:.2f}".format(svc.score(x_test, y_test)))
print()
print("Confusion Matrix ")
print(confusion_matrix(y_test,kn.predict(x_test)))
print()
print("Report")
print(classification_report(y_test,kn.predict(x_test)))
```

```
Accuracy on training set: 1.00
Accuracy on test set: 0.65

Confusion Matrix
[[105  20]
 [ 23  44]]

Report
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       125
           1       0.69      0.66      0.67        67

    accuracy                           0.78       192
   macro avg       0.75      0.75      0.75       192
weighted avg       0.77      0.78      0.77       192
```

```
C:\anaconda\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: Th
e default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'a
uto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [128]:

```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.fit_transform(x_test)
svc = SVC()
svc.fit(x_train_scaled, y_train)
print("Accuracy on training set: {:.2f}".format(svc.score(x_train_scaled
, y_train)))
print("Accuracy on test set: {:.2f}".format(svc.score(x_test_scaled, y_t
est)))
print()
print("Confusion Matrix ")
print(confusion_matrix(y_test,kn.predict(x_test)))
print()
print("Report")
print(classification_report(y_test,kn.predict(x_test)))
```

```
Accuracy on training set: 0.77
Accuracy on test set: 0.77

Confusion Matrix
[[105  20]
```

```
[[105  20]
 [ 23  44]]

Report
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       125
           1       0.69      0.66      0.67        67

    accuracy                           0.78       192
   macro avg       0.75      0.75      0.75       192
weighted avg       0.77      0.78      0.77       192
```

```
C:\anaconda\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: Th
e default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'a

uto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [129]:

```
svc = SVC(C=1000)
svc.fit(x_train_scaled, y_train)
list_of_training_accuracy.append(svc.score(x_train_scaled,y_train))
list_of_testing_accuracy.append(svc.score(x_test_scaled,y_test))
print("Accuracy on training set: {:.3f}".format(
    svc.score(x_train_scaled, y_train)))
print("Accuracy on test set: {:.3f}".format(svc.score(x_test_scaled, y_t
est)))
print()
print("Confusion Matrix ")
print(confusion_matrix(y_test,kn.predict(x_test)))
print()
print("Report")
print(classification_report(y_test,kn.predict(x_test)))
```

```
Accuracy on training set: 0.790
Accuracy on test set: 0.797

Confusion Matrix
[[105  20]
 [ 23  44]]

Report
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       125
           1       0.69      0.66      0.67        67
```

```
    accuracy                           0.78       192
   macro avg       0.75      0.75      0.75       192
weighted avg       0.77      0.78      0.77       192
```

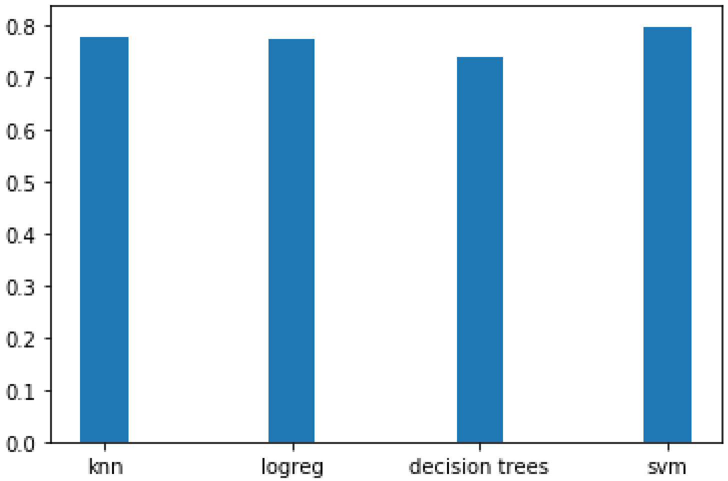
```
C:\anaconda\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: Th
e default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'a
uto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [109]:

```
list_of_models=['knn','logreg','decision trees','svm']
plt.bar(list_of_models,list_of_testing_accuracy,width = 0.25)
```

Out[109]:

```
<BarContainer object of 4 artists>
```



from the above bar graph we can observe Support Vector Machine has got the highest accuracy

In [ ]: