```
from sklearn.datasets import load_breast_cancer
import pandas as pd
import numpy as ny
import matplotlib.pyplot  as plt
```

KNN

```
cancer=load_breast_cancer()
```

```
print(cancer.keys())
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

```
X=cancer.data
```

```
cancer.target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 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, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

```
cancer.target_names
```

```
array(['malignant', 'benign'], dtype='<U9')
```

```
cancer.feature_names
```

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

```
X.shape
```

```
(569, 30)
```

```
Y=cancer.target
```

```
Y.shape
```

```
(569,)
```

```
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=4)
```

```
x_train.shape
```

```
(455, 30)
```

```
x_test.shape
```

```
(114, 30)
```

```
y_train.shape
```

```
(455,)
```

```
y_test.shape
```

```
(114,)
```

```
X
```

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])
```

```
type(X)
```

```
numpy.ndarray
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
model=KNeighborsClassifier()
```

```
knnmodel=model.fit(x_train,y_train)
```

```
print(knnmodel)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

```
Yp=knnmodel.predict(x_test)
```

```
Yp
```

```
array([1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 0, 1])
```

```
len(Yp)
```

```
114
```

```
y_test
```

```
array([1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1])
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(455, 30)
(114, 30)
(455,)
(114,)
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
acc1=(accuracy_score(y_test,Yp))
acc1
```

```
0.8771929824561403
```

```
print(confusion_matrix(y_test,Yp))
```

```
[[29  5]
 [ 9 71]]
```

```
print(classification_report(y_test,Yp))
```

```
              precision    recall  f1-score   support

           0       0.76      0.85      0.81        34
           1       0.93      0.89      0.91        80

    accuracy                           0.88       114
   macro avg       0.85      0.87      0.86       114
weighted avg       0.88      0.88      0.88       114
```

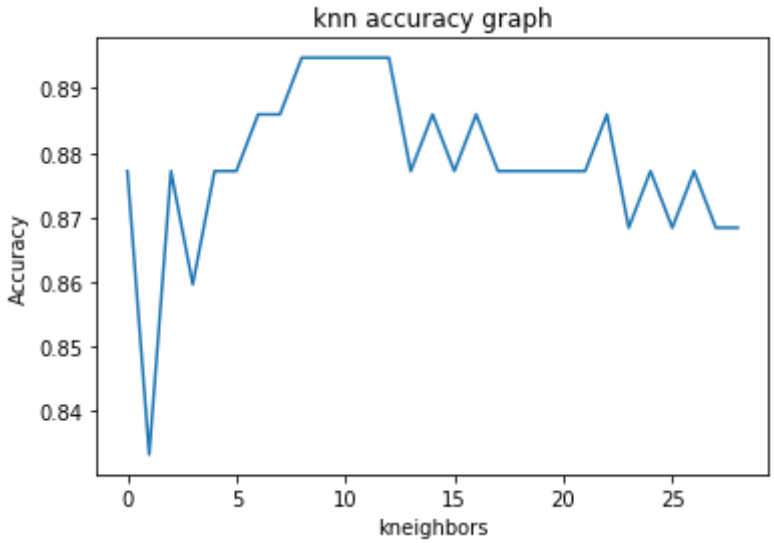
```
ls=[]
```

```
for i in range(1,30):
    model=KNeighborsClassifier(n_neighbors=i)
    knnmodel=model.fit(x_train,y_train)
    Yp=knnmodel.predict(x_test)
    acc=(accuracy_score(y_test,Yp))
```

```
    ls.append(acc)
print(ls)
```

```
[0.8771929824561403, 0.8333333333333334, 0.8771929824561403, 0.8596491228070176, 0.87
```

```
plt.plot(ls)
plt.xlabel("kneighbors")
plt.ylabel("Accuracy")
plt.title("knn accuracy graph")
plt.show()
```



Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
model1=LogisticRegression()
```

```
model1
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
lrmodel=model1.fit(x_train,y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432: FutureWa
  FutureWarning)
```

```
Ypl=lrmodel.predict(x_test)
```

```
Ypl
```

```
array([0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 1])
```

```
y_test
```

```
array([1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1])
```

```
acc2=(accuracy_score(y_test,Ypl))
acc2
```

```
0.9122807017543859
```

```
print(confusion_matrix(y_test,Ypl))
```

```
[[32  2]
 [ 8 72]]
```

```
print(classification_report(y_test,Ypl))
```

```
              precision    recall  f1-score   support

           0       0.80      0.94      0.86        34
           1       0.97      0.90      0.94        80

    accuracy                           0.91       114
   macro avg       0.89      0.92      0.90       114
weighted avg       0.92      0.91      0.91       114
```

```
lrmodel.classes_
```

```
array([0, 1])
```

```
lrmodel.coef_
```

```
array([[ 1.52600655,  0.17074294,  0.0778339 , -0.00609732, -0.09889202,
```

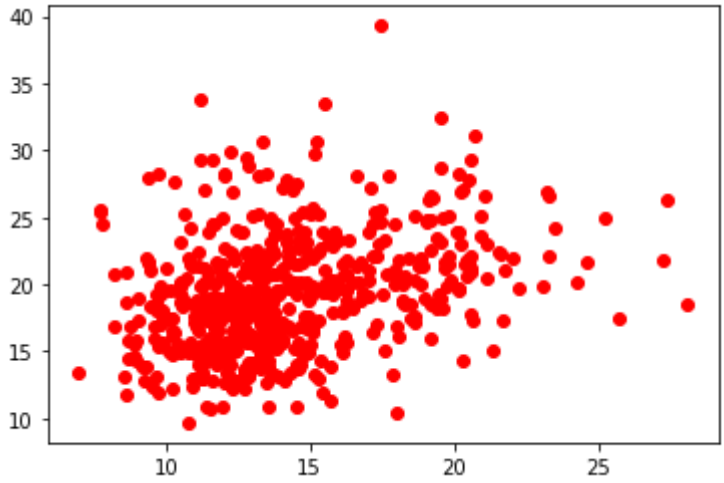
```
lrmodel.intercept_
```

```
array([0.24003215])
```

```
       -0.67690442, -1.10809088, -0.40349885, -0.40321753, -0.07019167]])
```

```
import matplotlib.pyplot as plp

plp.scatter(cancer.data[:,0],cancer.data[:,1],c='r')
```

```
<matplotlib.collections.PathCollection at 0x7f1773b48160>
```



Decision Tree

```
X1=pd.DataFrame(cancer.data,columns=cancer.feature_names)

X1.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

```
Y1=pd.DataFrame(cancer.target,columns=['Cancer Type'])

Y1
```

	Cancer Type
0	0
1	0
2	0
3	0
4	0
...	...
564	0
565	0
566	0
567	0
568	1

569 rows × 1 columns

```
X1.shape
```

```
(569, 30)
```

```
Y1.shape
```

```
(569, 1)
```

```
X1.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	conca
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.08
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.07
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.00
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.02
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.06
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.13
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.42

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier(criterion='entropy', max_depth=2)
dt
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=2,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

```
model2=dt.fit(X1,Y1)
model2
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=2,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

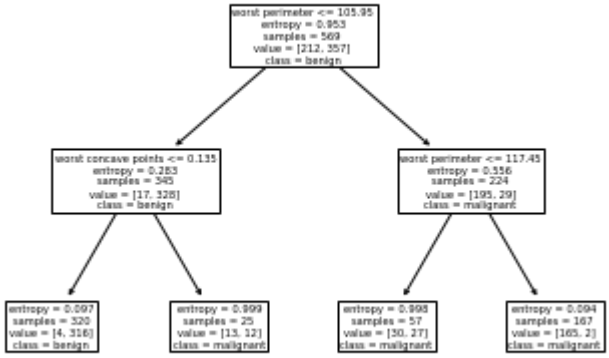
```
Yp2=model2.predict(x_test)
Yp2
```

```
array([0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 0, 1])
```

```
from sklearn import tree
```

```
tree.plot_tree(model2, feature_names=cancer.feature_names, class_names=cancer.target_names
```

```
[Text(167.4, 181.2, 'worst perimeter <= 105.95\nentropy = 0.953\nsamples = 569\nvalue
 Text(83.7, 108.72, 'worst concave points <= 0.135\nentropy = 0.283\nsamples = 345\nv
 Text(41.85, 36.23999999999998, 'entropy = 0.097\nsamples = 320\nvalue = [4, 316]\ncl
 Text(125.55000000000001, 36.23999999999998, 'entropy = 0.999\nsamples = 25\nvalue =
 Text(251.10000000000002, 108.72, 'worst perimeter <= 117.45\nentropy = 0.556\nsample
 Text(209.25, 36.23999999999998, 'entropy = 0.998\nsamples = 57\nvalue = [30, 27]\ncl
 Text(292.95, 36.23999999999998, 'entropy = 0.094\nsamples = 167\nvalue = [165, 2]\nc
```




```
acc3=(accuracy_score(y_test,Yp2))
```

```
acc3
```

```
0.8596491228070176
```

SVM(Support Vector Machine)

```
from sklearn.svm import SVC, LinearSVC
from sklearn import svm
```

```
clf = LinearSVC()
clf.fit(x_train, y_train)
prediction = clf.predict(x_test)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/base.py:929: ConvergenceWarning: L
  "the number of iterations.", ConvergenceWarning)
```

```
acc4=accuracy_score(prediction, y_test)
acc4
```

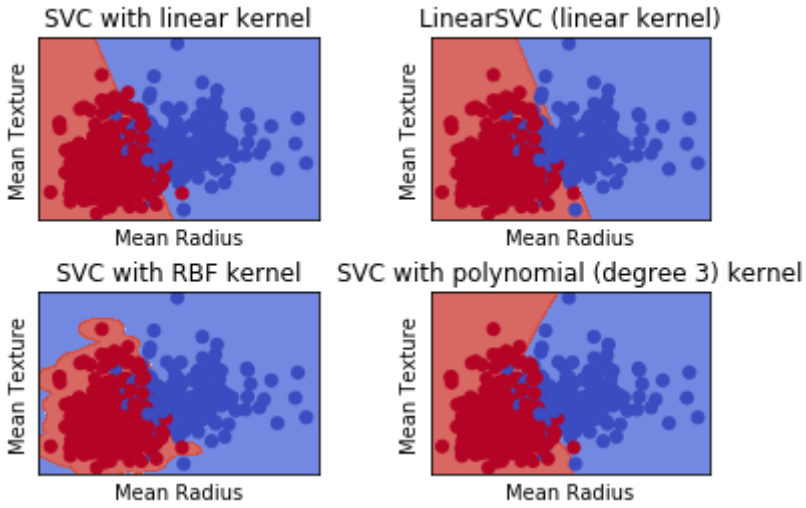
```
0.8947368421052632
```

```
X2 = cancer.data[:, :2]  # we only take the Sepal two features.
Y2 = cancer.target
C = 1.0  # SVM regularization parameter

# SVC with linear kernel
svc = svm.SVC(kernel='linear', C=C).fit(X2, Y2)
# LinearSVC (linear kernel)
lin_svc = svm.LinearSVC(C=C).fit(X2, Y2)
# SVC with RBF kernel
```

```
# SVC with RBF kernel
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X2, Y2)
# SVC with polynomial (degree 3) kernel
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X2, Y2)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/base.py:929: ConvergenceWarning: L
  "the number of iterations.", ConvergenceWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/base.py:193: FutureWarning: The de
  "avoid this warning.", FutureWarning)
```

```
h=.02
# create a mesh to plot in
x_min, x_max = X2[:, 0].min() - 1, X2[:, 0].max() + 1
y_min, y_max = X2[:, 1].min() - 1, X2[:, 1].max() + 1
xx, yy = ny.meshgrid(ny.arange(x_min, x_max, h),
                     ny.arange(y_min, y_max, h))
# title for the plots
titles = ['SVC with linear kernel',
     'LinearSVC (linear kernel)',
      'SVC with RBF kernel',
      'SVC with polynomial (degree 3) kernel']


for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):
   # Plot the decision boundary. For that, we will assign a color to each
   # point in the mesh [x_min, x_max]x[y_min, y_max].
   plt.subplot(2, 2, i + 1)
   plt.subplots_adjust(wspace=0.4, hspace=0.4)

   Z = clf.predict(ny.c_[xx.ravel(), yy.ravel()])

   # Put the result into a color plot
   Z = Z.reshape(xx.shape)
   plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

   # Plot also the training points
   plt.scatter(X2[:, 0], X2[:, 1], c=Y2, cmap=plt.cm.coolwarm)
   plt.xlabel('Mean Radius')
   plt.ylabel('Mean Texture')
   plt.xlim(xx.min(), xx.max())
   plt.ylim(yy.min(), yy.max())
   plt.xticks(())
   plt.yticks(())
   plt.title(titles[i])

plt.show()
```



Accuracy Graph of all Agorithms

```
li1=[acc1,acc2,acc3,acc4]
li2=['KNN','Logistic Regression','Decision Tree','SVM']
x_pos = [i for i, _ in enumerate(li2)]
fig, ax = plp.subplots()
rects1 = ax.bar(x_pos,li1)
plt.xticks(x_pos,li2)
def autolabel(rects):
   for rect in rects1:
      height = rect.get_height()
      ax.text(rect.get_x() + rect.get_width()/2.,1.05*height,
              '%f' % float(height), ha='center', va='bottom')
autolabel(rects1)
plp.show()
```