```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial import Voronoi, voronoi_plot_2d
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.svm import LinearSVC
import math

a = np.asarray([[1,2],[1,4],[2,3],[2,7],[3,3],[3,5],[3,6],[4,2],[4,4],[4,12],[4,15],[5,3],[5,11],[5,13],
[6,12],[6,13],[6,14],[7,14],[8,12],[9,7],[10,6],[10,8],[10,9],[10,10],[11,8],[11,9],[11,11],[12,10]])

ay = ([1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0])


ay2 = np.asarray([1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

c = np.asarray([[1,2],[1,4],[1,11],[2,3],[2,7],[3,3],[3,5],[4,15],[4,12],[5,11],[5,13],[6,12],[6,13],
[6,14],[7,9],[7,14],[8,2],[8,12],[9,7],[10,6],[10,8],[10,9],[10,10],[11,3],[11,8],[11,9],[11,11],[12,10],
[13,3],[13,5],[14,4],[14,13],[14,14],[15,13]])

#######naive bayes

model = GaussianNB()
model.fit(a, ay)
predicted= model.predict([[1,2],[14,6],[7,8]])
print predicted
plt.scatter(a[:, 0], a[:, 1], c=ay, s=100)
plt.show()

#######k-nearest neighbor

vor = Voronoi(a)
voronoi_plot_2d(vor)
plt.show()

vor = Voronoi(c)
voronoi_plot_2d(vor)
plt.show()

######logistic regression

def sigmoid(x):
    a = []
    for item in x:
        a.append(1/(1+math.exp(-item)))
    return a

Xaxis = np.arange(-10., 10., 0.2)
#Xaxis is an one-dimensional array with 100 elements
#begin from -10.0 to 10.0, with interval of 0.2 for each two elements
sig = sigmoid(Xaxis)

plt.plot(Xaxis, sig, color="blue")
plt.show()

t = a[:,0].reshape(28,1)
ty= ay2.reshape(28,1)
X_test = np.linspace(0, 13, 200)
# instantiate a logistic regression model, and fit with X and y
# C = Inverse of regularization strength; must be a positive float. Like in support vector machines,
smaller values specify stronger regularization.
lr = LogisticRegression(C=1e5)
print t.shape, ty.shape
lr = lr.fit(t, ty)

########Linear Regression
```

```
ols = LinearRegression()
ols.fit(t, ty)
loss1 = X_test * ols.coef_ + ols.intercept_
print loss1, X_test
plt.plot(X_test, loss1[0], linewidth=1)
plt.axhline(.5, color='.5')

# check the accuracy on the training set
print lr.score(t, ty)

loss = sigmoid([i * lr.coef_ + lr.intercept_ for i in X_test])
print lr.coef_, lr.intercept_
plt.scatter(t, ty, c=ty, s=100)
plt.plot(X_test, loss, color="blue")
plt.show()

########SVM

# instantiate a Linear Support Vector Classification, and fit with X and y
lsvc = LinearSVC()
lsvc = lsvc.fit(a, ay)

# check the accuracy on the training set
print lsvc.score(a, ay)

xx, yy = np.meshgrid(np.arange(min(a[:,0])-1, max(a[:,0])+1, .02), np.arange(min(a[:,1])-1, max(a[:,1])
+1, .02))

Z = lsvc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.5)
plt.scatter(a[:, 0], a[:, 1], c=ay, cmap=plt.cm.coolwarm, s=100)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())
plt.show()
```