2 - 线性回归

一元线性回归

```
xm = np.mean(x)
ym = np.mean(y)
syy = np.mean((y-ym)**2)
syx = np.mean((y-ym)*(x-xm))
sxx = np.mean((x-xm)**2)
beta1 = syx/sxx
beta0 = ym - beta1*xm
print("xbar={0:7.2f}, ybar={1:7.2f}".format(xm,ym))
print("sxx={0:7.2f},  syy={1:7.2f}".format(np.sqrt(sxx),np.sqrt(syy)))
print("beta0={0:7.2f}, beta1={1:7.2f}".format(beta0,beta1))
yhat=beta0+beta1*x
RSS = np.mean((y-yhat)**2)
print("RSS = {0:7.2f}".format(RSS))
```

多元线性回归

用sklearn包：

```
regr = linear_model.LinearRegression()
regr.fit(X_tr,y_tr)
regr.coef_
regr.intercept_
y_tr_pred = regr.predict(X_tr)
RSS = np.mean((y_tr_pred-y_tr)**2)/(np.std(y_tr)**2)
print("Normalized RSS={0:f}".format(RSS))
plt.scatter(y_test,y_test_pred)
plt.plot([0,350],[0,350],'r’)
```

coefficient of determination, R_k^2,

```
 xm = np.mean(X[:,k])
    sxy = np.mean((X[:,k]-xm)*(y-ym))
    sxx = np.mean((X[:,k]-xm)**2)
    beta1[k] = sxy/sxx
    beta0[k] = ym - beta1[k]*xm
    Rsq[k] = (sxy)**2/sxx/syy
```

用np包

```
ones = np.ones((ns_train,1))
A = np.hstack((ones,X_tr))
A.shape
out = np.linalg.lstsq(A,y_tr)
beta = out[0]
```