Lab Sessions Day 3

Exercise 1

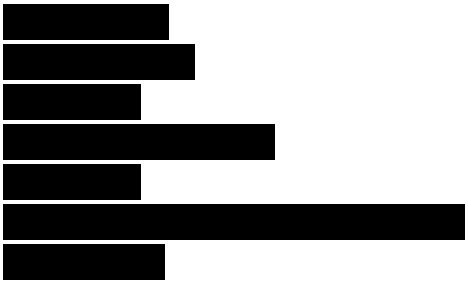
Linear regression

```
#Make two vector X and y

X=np.array([1,2,4,3,5])
y=np.array([1,3,3,2,5])

#With simple linear regression we want to model our data as follows:
#y = B0 + B1 * x


#We can start off by estimating the value for B1 as:
#B1 = sum((Xi-mean(X)) * (yi-mean(y))) / sum((Xi- mean(X))^2)
```

```
#We can calculate B0 using B1 and some statistics from our dataset, as follows:
#B0 = mean(y) – B1 * mean(X)
```

```
#Making Predictions (y_hat is a predicted y)
y_hat=B0+B1*X
```

#Evaluation
RMSE = sqrt(sum((y_hat_i – yi)^2)/n)



Exercise 2

Logistic regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

cancer=load_breast_cancer()
X_train,X_test,y_train,y_test=train_test_split(cancer.data,cancer.target,stratify=cancer.target,ra
ndom_state=42)

######default C=1#####
lgr=LogisticRegression().fit(X_train,y_train)
print("training set score: %f" % lgr.score(X_train, y_train))
print('\n'"test set score: %f" % lgr.score(X_test, y_test))

######increase C to 100#####
lgr100=LogisticRegression(C=100).fit(X_train,y_train)
print('\n'"training set score of lgr100: %f" % lgr100.score(X_train, y_train))
print('\n'"test set score of lgr100: %f" % lgr100.score(X_test, y_test))

Change C value and compare the performance metric
######decrease C to 0.01#####
lgr001=LogisticRegression(C=0.01).fit(X_train,y_train)
print('\n'"training set score of lgr001: %f" % lgr001.score(X_train, y_train))
print('\n'"test set score of lgr001: %f" % lgr001.score(X_test, y_test))

import matplotlib.pyplot as plt
```

```
plt.plot(lgr.coef_.T,'o',label='C=1')
plt.plot(lgr100.coef_.T,'+',label='C=100')
plt.plot(lgr001.coef_.T,'-',label='C=0.01')
plt.xticks(range(cancer.data.shape[1]),cancer.feature_names,rotation=90)
plt.ylim(-5,5)
plt.legend()
plt.show()

###If we desire a more interpretable model, using L1 regularization might help
###As LogisticRegression applies an L2 regularization by default, the result
###looks similar to Ridge in Figure ridge_coefficients. Stronger regularization
###pushes coefficients more and more towards zero, though coefficients never
###become exactly zero.

import numpy as np
import math
n=np.arange(-2,3)
print(n)
r=pow(float(10),n)
print(r)
for C in r:
    lr_l1=LogisticRegression(C=C,penalty="l1").fit(X_train,y_train)
    print('\n'"Training Accuracy of L1 LogRess with C=%f:%f"%(C,lr_l1.score(X_train,y_train)))
    print('\n'"Test Accuracy of L1 LogRegss with C=%f: %f"%(C,lr_l1.score(X_test,y_test)))
    plt.plot(lr_l1.coef_.T,'o',label="C=%f"%C)
plt.xticks(range(cancer.data.shape[1]),cancer.feature_names,rotation=90)
plt.ylim(-5,5)
plt.legend(loc='best')
plt.show()
```