# Homework 1

Deadline: 20/03/2024

In this homework, you will write a python class called LogisticRegression to implement the logistic regression model. The interface of the class is like the popular machine learning package scikit-learn (scikit-learn https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html). Your task is to complete the four methods *\_\_init\_\_*, *fit*, *predict* and *predict\_prob*.

class LogisticRegression:  
  
 def \_\_init\_\_(self, penalty='l1', \*, tol=1e-4, C=0.1):  
 *"""* ***:param*** *penalty: 'l1' or 'l2' norm for regularization* ***:param*** *tol: tolerance for optimization* ***:param*** *C: See https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression  
 """* # your code here  
  
 def fit(self, X, y):  
 *"""  
 This function is called to train the model given training examples (X and y)* ***:param*** *X: two-dimensional array of size N x d, where N is the number of training examples* ***:param*** *y: labels for the corresponding features in X, it is a one-dimensional array of size N* ***:return****:  
 """* # your code here  
 return self  
  
 def predict(self, X):  
 *"""  
 Given new examples X, this function return prediction* ***:param*** *X: features of new examples, It is a two-dimensional array of size m x d, where m is the number of  
 new examples* ***:return****: one dimensional array of size m, prediction of new examples  
 """* # your code here  
  
 def predict\_prob(self, X):  
 *"""  
 Given new examples X, this function return prediction probability* ***:param*** *X: features of new examples, It is a two-dimensional array of size m x d, where m is the number of  
 new examples* ***:return****: one dimensional array of size m, prediction probability of new examples  
 """* # your code here

Once you have completed the class, you will write a function called *buid\_model* to test your class.

def build\_model(dataset):  
  
 # 1. split the dataset into training, validation and test dataset  
 # your code here  
  
 # 2. use cross-validation techniques to choose model hyper-parameters penalty ('l1' or 'l2') and C  
 # your code here  
 # optimal\_penalty = ...  
 # optimal\_C = ...  
  
 # 3. train model with optimal hyper-parameters  
 model = LogisticRegression(penalty=optimal\_penalty, C=optimal\_C)  
  
 # 4. train model with training + validation datasets  
 # your code here  
  
 # 5. test model performance with test dataset  
 # your code here  
  
 return model

In writing this function, you can use the following steps:

1. Split dataset into training, validation and test data sets
2. Use cross validation techniques to select optimal hyper-parameters *penalty* and *C*
3. Train the model with the optimal hyper-parameters
4. Test model performance with test dataset

At last, you will write a function to load data and call *build\_model*. An example is given below.

def test():  
 data\_file = 'mydatafile'  
 dataset = pd.read\_csv(data\_file)  
 model = build\_model(dataset)

# Dataset

We will use the Iris dataset for testing purpose. The dataset can be downloaded using sklearn API

**from** sklearn **import** datasets

iris **=** datasets**.**load\_iris(as\_frame**=True**)