Implementation:

This implementation was done in python. The size of the datatset is 8291x68 that is there are 8291 examples and 68 features. In this implementation I have added one more column for bias in the dataset because I have initialized by weight vector as all the weights and bias. A new dataset matrix was created of size 8291x69 in which the last column was initially all 1s. The output labels were stored in another array of size 8291x1.

Numpy package was used to initialize arrays and matrices. This made the manipulation of matrices very easy.

For every variant I have included 2 python files. One file finds the best hyperparameters and the other one tests it on development and testing data. They are named as variant_hyper.py for hyperparameters and variant.py for testing of that variant.

There are 4 major functions used in all variants.

1**)newX,X,labels=inputfeat(filename**) : This function takes the file which has the data and extracts out 3 parameters X which are the features for all examples, labels is the output label for all examples and newX is the concatenation of features and labels.

2) **weight,no.of updates=train(epochs,r,w,newX**):This function trains according to each variant and gives out the trained weights and the no. of updates made. The input arguments it takes are no. of epochs, learning rate (r) , intial weight(w), dataset(newX).

For every variant this function changes. For dynamic and margin perceptron the learning rate is getting updated every epoch. Also, data gets shuffled after every epoch

3) predicted_output=predict(w,newX): This function predicts the predicted output for all variants. For the average perceptron it takes in the trained average weights instead of trained weights.

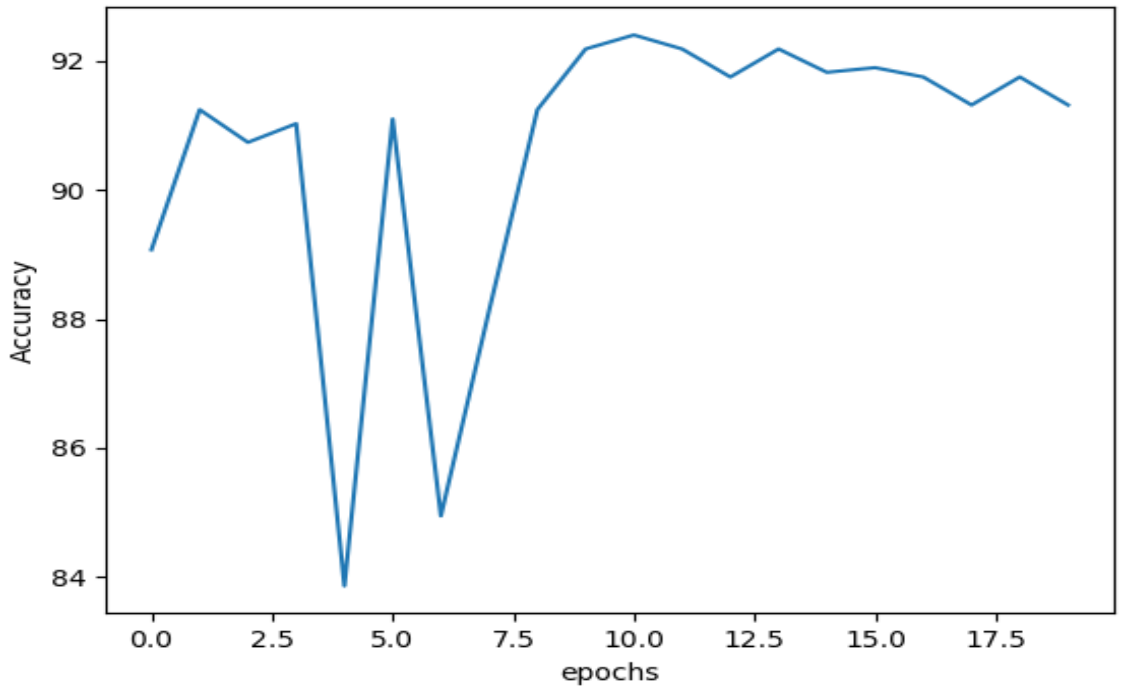
4)**W,acc=traindev(traindev(epochs,r,w,newX,ct,newX2,labels2):** This function trains on 'phishing.train' and tests the output of 'phishing.dev' with its labels for every epoch. This way I can see the development set accuracy for every epoch. With the help of that I can figure out for which epoch I will get max accuracy and I can use that to predict my output for 'phishing.test' for only those many epochs. Basically, it will take the trained weights for those many epochs only and use that for testing on the 'phishing.test'.

In all the cases my cross validation accuracy is very close to my development set accuracy. Also the best parameters from the variant_hyper.py file gives out the best result for my testing data for all my variants.

Best parameters to be used for 10 and 20 epochs are given below and the plot is the learning curve of development set for every epoch for all vaariants.

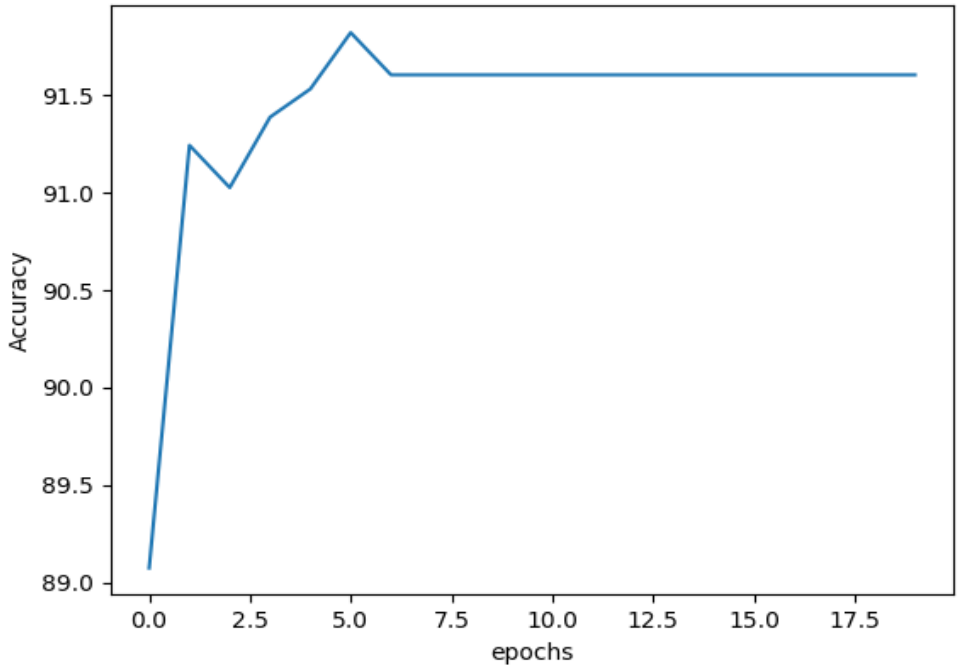
Simple Perceptron:

epochs	Cross validation accuracy	Learning rate	margin	Development accuracy	Test Accuracy	No. of updates
10	90.085	0.01	-	91.678	92.40	7098
20	90.006	0.01	-	87.337	92.619	13988



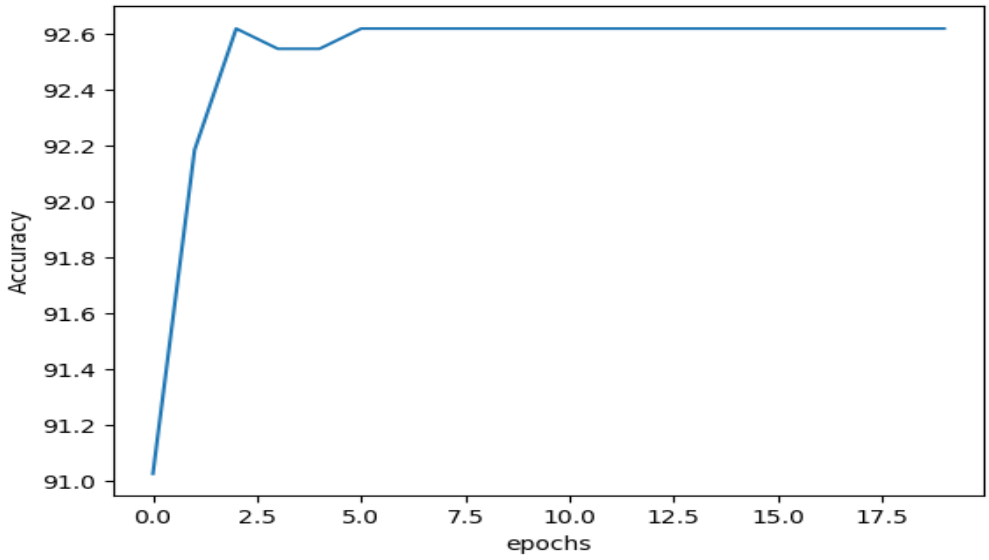
Dynamic Perceptron:

epochs	Cross validation Accuracy	Learning rate	Margin	Development accuracy	Testing accuracy	No. of Updates
10	92.606	0.01	-	91.678	93.053	5732
20	92.509	0.01	-	91.89	92.836	10907



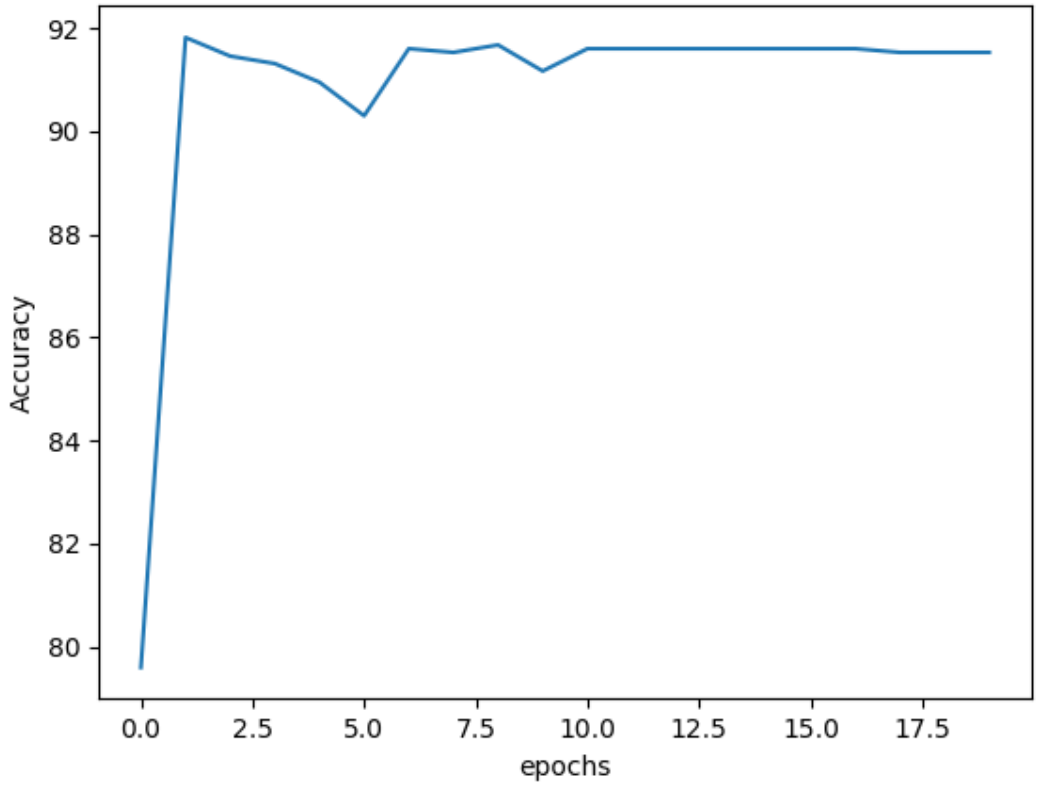
Margin Perceptron:

Epochs	Cross Validation accuracy	Learning rate	margin	Development accuracy	Testing Accuracy	Updates
10	**92.907**	**0.01**	**0.1**	**92.619**	**93.125**	**15046**
20	**93.040**	**0.01**	**0.1**	**92.185**	**93.053**	**28968**



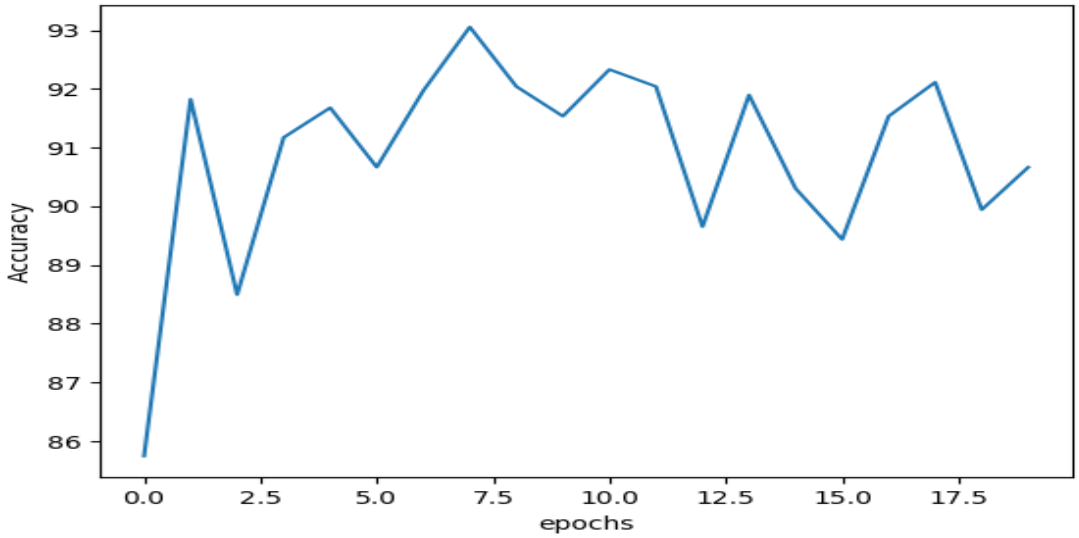
Average Perceptron:

Epochs	Cross Validation accuracy	Learning rate	margin	Development accuracy	Testing Accuracy	Updates
10	90.122	0.1	-	91.389	92.836	5746
20	89.832	0.1	-	91.678	92.764	10927



Aggressive-Margin Perceptron:

Epoch	Cross Validation Accuracy	Learning rate	Margin	Development Accuracy	Testing Accuracy	Updates
10	91.943	0.01	0.01	90.159	93.415	10370
20	91.834	0.01	0.1	82.192	93.125	18323


93
92
91
90
89
88
87
86
Accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epochs