MNIST Digits Classification using SVM & PCA

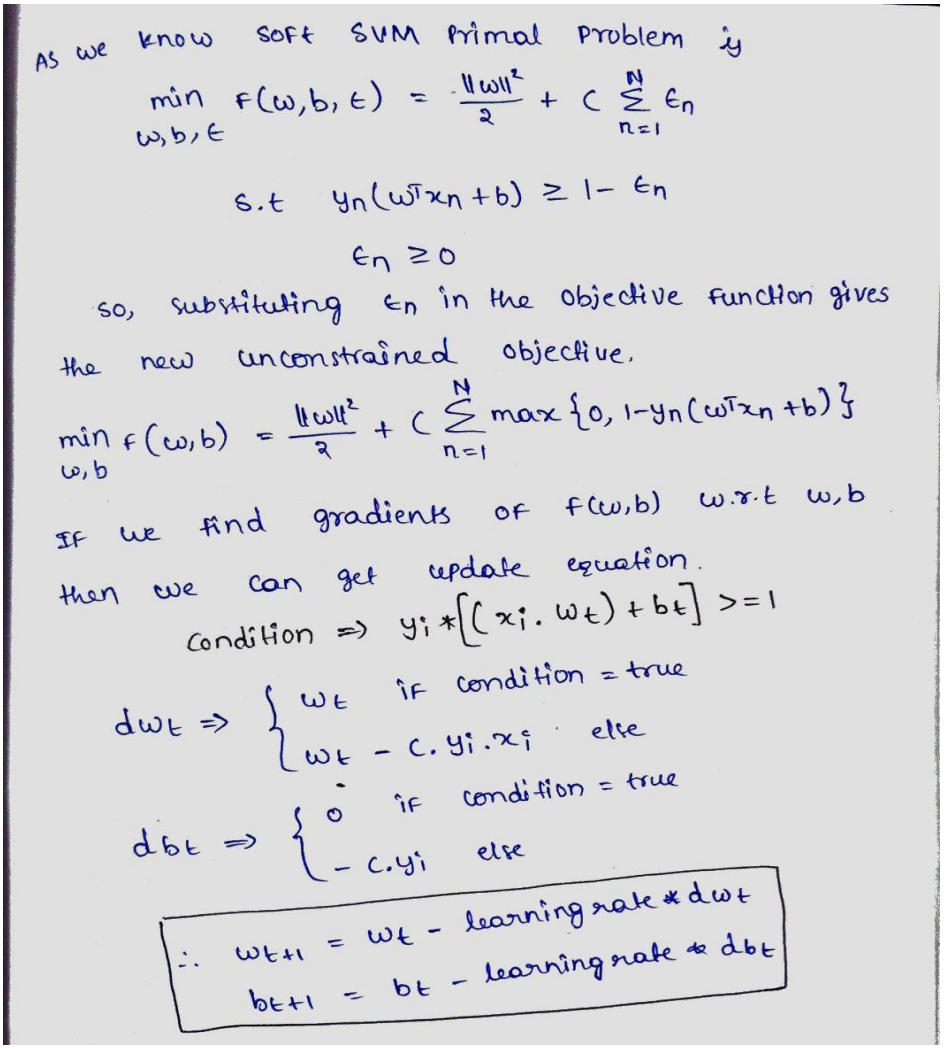
Methodology:

1) Normalisation of Data : Finding min,max value in column and apply this formula to normalise data between -1 and 1.

   Norm(X) = ((X-min)/(max-min))*2 - 1

2) PCA : Finding the eigenvectors of Covariance Matrix of input X . Storing those vectors based on eigenvalues larger to smaller. Pick first K-eigenvectors and transform input X into k dimensions by **dot_product(X,K-comp)**

3) SupportVectorModel :



Using this update equations we will update the weights for **num_iterations** times .

4) MultiClassSVM : Fit SupportVectorModel for all 10 classes pass the data by making y values as 1 for that class and -1 for remaining classes. Find these metric values for each class .
   True Positive(i) = #(y==y_pred==i) .
   False Positive(i) = #(y!=i && y_pred==i)
   False Negative(i) = #(y==i && y_pred!=i)
   Precision(i) = True Positive(i)/(True Positive(i) + False Positive(i))
   Recall(i) = True Positive(i)/(True Positive(i) + False Negative(i))
   F1_Score(i) = 2*Precision(i)*Recall(i)/(Precision(i)+Recall(i))

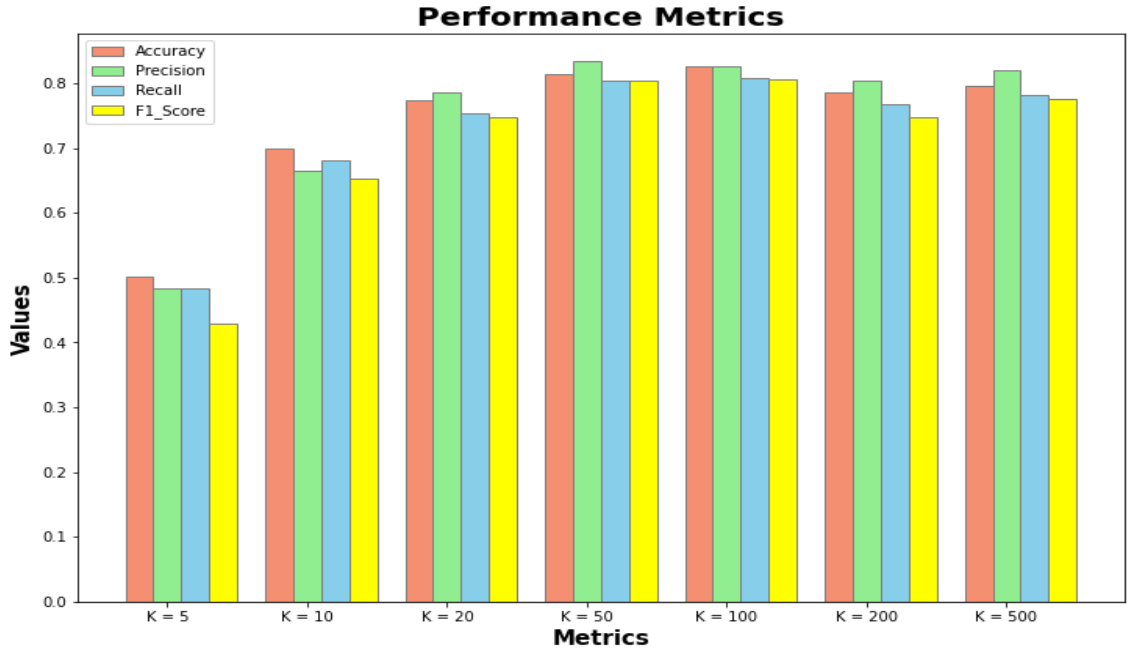
   Precision = Avg(Precision(i) for all i)
   Recall = Avg(Recall(i) for all i)
   F1_Score = Avg(F1_Scorel(i) for all i)

   Plot the results with respect to K value.

Results

```
Learning_rate  : 0.001
num_iters  : 100000
k=5, accuracy=0.5008, precision=0.4827, recall=0.4842, f1_score=0.4281
k=10, accuracy=0.6993, precision=0.6654, recall=0.6803, f1_score=0.6535
k=20, accuracy=0.7744, precision=0.7854, recall=0.7534, f1_score=0.7478
k=50, accuracy=0.8136, precision=0.835, recall=0.8037, f1_score=0.8033
k=100, accuracy=0.826, precision=0.8262, recall=0.8083, f1_score=0.8053
k=200, accuracy=0.785, precision=0.8032, recall=0.7684, f1_score=0.7477
k=500, accuracy=0.7951, precision=0.8193, recall=0.7817, f1_score=0.7759
```

Plot:



Analysis:

In my code hyper_parameter values are

learning_rate = 0.001
num_iters = 100000
C = 1.0

Here accuracy depends on num_iterations upto some number after that it will go to overfit so that it doesn't perform well on new samples .In data set training samples are 60000 so I took num_iterations = 100000 . We are performing stochastic gradient descent on the model so it will update the weights on every feature vector sample.In my results I got high accuracy in 100 components 82.6% . Here I am using a learning rate as 0.001 due to accuracy. I tried with 0.01 but it's highest accuracy among all components is 63.6%.These are my observations in this experiment