Problem 3

I used the subset.py file found at https://github.com/cjlin1/libsvm/blob/master/tools/subset.py

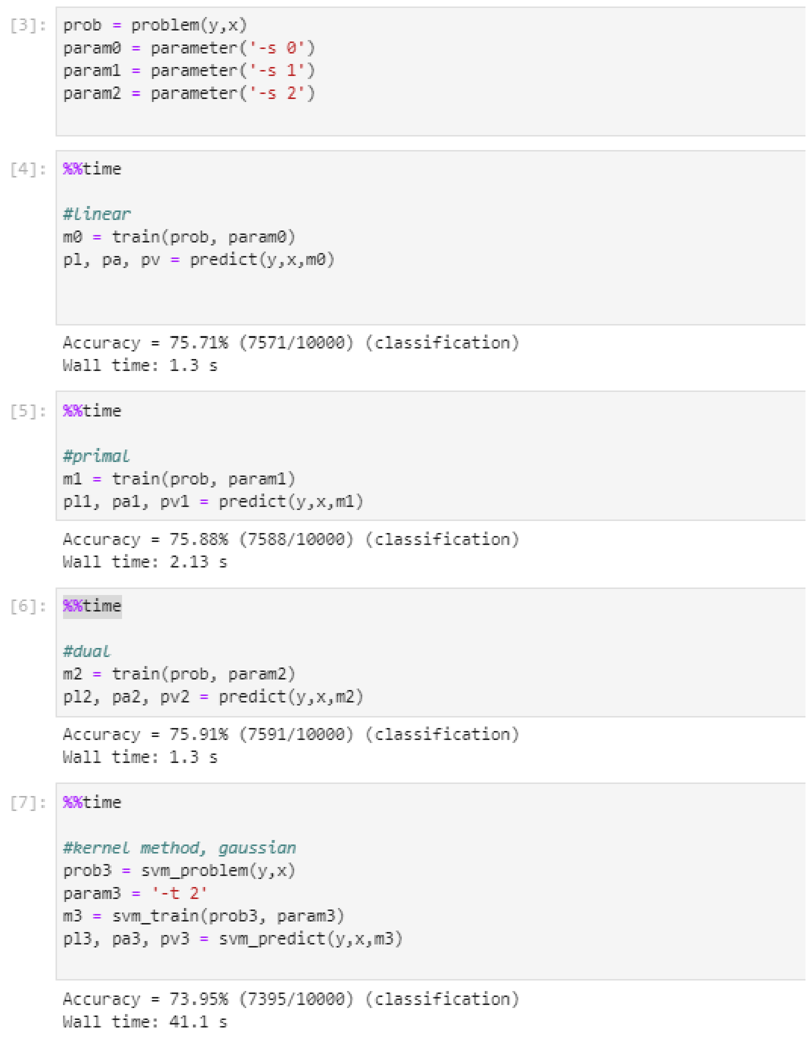
to randomly sampled 10000 data and saved to new file new.libsvm.binary

the script used at terminal:

$ python subset.py -1 covtype.libsvm.binary 10000 new.libsvm.binary

The new file and subset.py is submitted along with the code.

Below is the output of linear, primal, dual, kernel svm respectively.



```
[3]: prob = problem(y,x)
     param0 = parameter('-s 0')
     param1 = parameter('-s 1')
     param2 = parameter('-s 2')
```

```
[4]: %%time

     #linear
     m0 = train(prob, param0)
     pl, pa, pv = predict(y,x,m0)

     Accuracy = 75.71% (7571/10000) (classification)
     Wall time: 1.3 s
```

```
[5]: %%time

     #primal
     m1 = train(prob, param1)
     pl1, pa1, pv1 = predict(y,x,m1)

     Accuracy = 75.88% (7588/10000) (classification)
     Wall time: 2.13 s
```

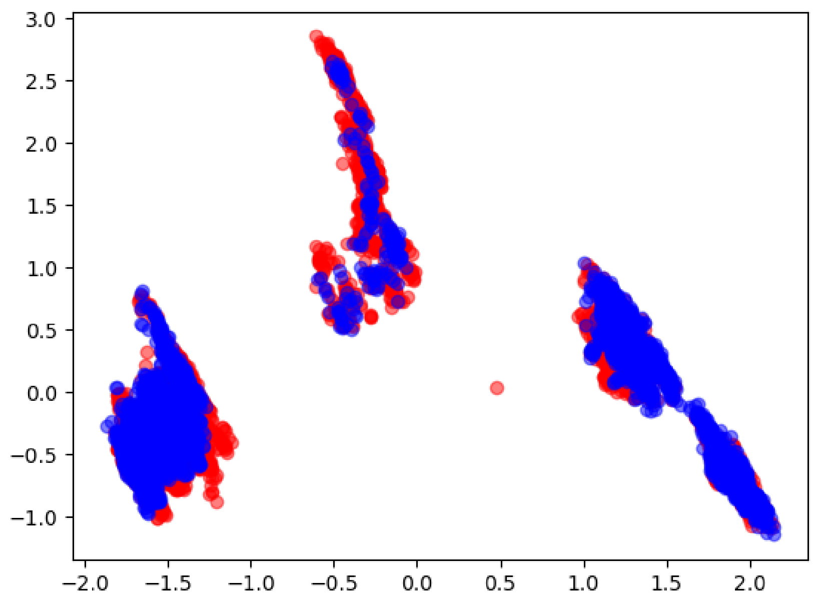
```
[6]: %%time

     #dual
     m2 = train(prob, param2)
     pl2, pa2, pv2 = predict(y,x,m2)

     Accuracy = 75.91% (7591/10000) (classification)
     Wall time: 1.3 s
```

```
[7]: %%time

     #kernel method, gaussian
     prob3 = svm_problem(y,x)
     param3 = '-t 2'
     m3 = svm_train(prob3, param3)
     pl3, pa3, pv3 = svm_predict(y,x,m3)

     Accuracy = 73.95% (7395/10000) (classification)
     Wall time: 41.1 s
```

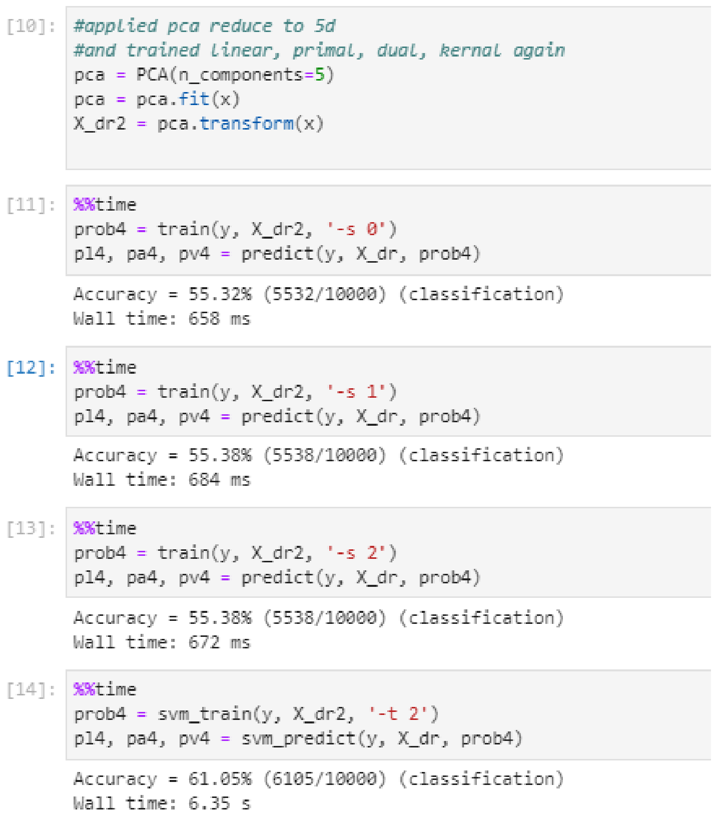
As we can see, linear, primal, and dual roughly produced the same accuracy, which is expected in svm. However, the primal took longer than the other two.

The kernel method produced a slightly lower accuracy. The training time is quadratic for kernel whereas the others are linear time. The kernel method is not preferred here.

After applying PCA to 2D, here's a visualization of data



Also performed dimension reduction to 5D using PCA



```
[10]: #applied pca reduce to 5d
      #and trained linear, primal, dual, kernal again
      pca = PCA(n_components=5)
      pca = pca.fit(x)
      X_dr2 = pca.transform(x)
```

```
[11]: %%time
      prob4 = train(y, X_dr2, '-s 0')
      pl4, pa4, pv4 = predict(y, X_dr, prob4)

      Accuracy = 55.32% (5532/10000) (classification)
      Wall time: 658 ms
```

```
[12]: %%time
      prob4 = train(y, X_dr2, '-s 1')
      pl4, pa4, pv4 = predict(y, X_dr, prob4)

      Accuracy = 55.38% (5538/10000) (classification)
      Wall time: 684 ms
```

```
[13]: %%time
      prob4 = train(y, X_dr2, '-s 2')
      pl4, pa4, pv4 = predict(y, X_dr, prob4)

      Accuracy = 55.38% (5538/10000) (classification)
      Wall time: 672 ms
```

```
[14]: %%time
      prob4 = svm_train(y, X_dr2, '-t 2')
      pl4, pa4, pv4 = svm_predict(y, X_dr, prob4)

      Accuracy = 61.05% (6105/10000) (classification)
      Wall time: 6.35 s
```

The accuracy and the running time both decreased compared to before.