REPORT

PRE PROCESSING THE TRAINING DATA

First of all I observed that there were around 3800 features which had all the entries as 0.
So I filtered out those columns from the training data, and only around 231 features were left finally. This helped in increasing the speed of the classifiers which I used throughout the assignment. Moreover I save this filtered data in a file and loaded directly from it each time. This saved a lot of time.

SVM

Training SVM, was first taking lot of time, then I changed the cache parameter in the SVM model function to 8 or 10 GB, then it took about an hour for the linear kernel and more for the gaussian one.
Due to such a huge time consumption. I only fitted the model once, predicted on test data and made the submission on Kaggle. Accuracies I got were
Gaussian Kernel : 70.676 %
Linear Kernel : 89.380 %

Gaussian NB :

The normal Gaussian NB model gave the 5 fold CV scores as :
[0.70005783 0.69361265 0.6985606 0.70903483 0.72805552]
Multinomial NB required non negative values of X
Binomial NB works for boolean X values

DECISION TREES

Columns :
min_samples_split
min_samples_leaf
Mean 5 fold CV score

2	1	0.901890
2	2	0.892444
2	3	0.893755
3	1	0.900438
3	2	0.892855
3	3	0.893948

Random Forests will be surely better than Decision Trees, so quickly proceeding to them.

RANDOM FOREST

Columns :
n_estimators
max_features
bootstrap
Mean 5 fold CV score

7	None	True	0.92773
7	None	False	0.90452
7	auto	True	0.91444
7	auto	False	0.92340
7	log2	True	0.90493
7	log2	False	0.91331
10	None	True	0.93218
10	None	False	0.90518
10	auto	True	0.92409
10	auto	False	0.93130
10	log2	True	0.91672
10	log2	False	0.92094
13	None	True	0.93317
13	None	False	0.90483
13	auto	True	0.92440
13	auto	False	0.93066
13	log2	True	0.91570
13	log2	False	0.92441

Tried one with n_estimators = 100
[0.93947182 0.943452 0.94184552 0.93869683 0.93946793]
NO normalisation
NO PCA

As I saw Random Forest was working the best, started trying for different params :

n_estimator = 500 feature = 'auto' boot = None	: 0.93241
n_estimators = 700 feature = 'auto' boot = True	: 0.92650
PCA(n_components=500) n_estimator = 300 feature = 'auto' boot = True	: 0.92911
n_estimator = 50 feature = 'auto' boot = True min_split = 2 criterion="entropy"	: 0.92316
PCA components = 100, rest same as above	: 0.92064
PCA(n_components=400, whiten=True) n_estimators = 300	: 0.87416
pca : 220 n_estimator = 700 feature = 'auto' boot = None	: 0.93303
No normalisation, PCA = 150, n_estimator = 500 feature = 'sqrt' boot = None	: 0.93620
PCA components = 100, rest same as above	: 0.93665
No PCA n_estimators = 500	: 0.94691
nest = 100 oob_score=True	: 0.94135
max_feature = 'log2' from 'auto'	: 0.93774
Removed the zero columns max_feature = 0.60 nest = 100	: 0.94631
n_estimator = 100 feature = 0.70 boot = False Rest same as previous one	: 0.94654

Concluded PCA was not helping, so just used the filtered out data (removed the zero columns).

Now tried another Ensembling Method : **Extra Trees Classifier :** This one unlike the Random Forests chooses the decision boundary randomly, whereas RF select the best split at each point.

The basic implementation of ExtraTreesClassifier	: 0.94825
Varying min_split with nest = 100, here = 2	: 0.94607
min_split = 3	: 0.94649
min_split = 4	: 0.94562
This time used the filtered data with min_split = 2	: 0.94966
with random state = 0	: 0.94664
Increased n_estimators to 500	: 0.95025
Increased n_estimators to 700	: 0.95025
max_features = 0.60	: 0.95137
max_features = 0.50	: 0.95167
max_features = 0.40	: 0.95160
max_features = 0.55	: 0.95224
max_features = 0.55, n_estimators = 500	: 0.95253
max_features = 0.70, n_estimators = 500	: 0.95279

(in CV 0.70 race the best mean score, also limiting max_depth in CV for small n_estimators gave higher score for value ~ 450, but didn't gave any improvement on submitting for test data)

I also read online that we can increase accuracy by using just using multiple submissions files and submitting combination of them as a voting classifier would do on multiple models. This gave a pretty good accuracy but not better than my best one.

Finally used the **Voting Classifier** : It takes multiple models as inputs and averages out their drawbacks due to presence of other ones, and gives better accuracy then all the inputs 0.95363

So tried multiple combinations of Extra Tree Classifiers with varied parameters (the best ones) and I got my best accuracy with n_estimators = 100 for each one of them, used 4 of them :

```
clf4 = ExtraTreesClassifier(n_estimators=nest, max_features=0.68)
clf5 = ExtraTreesClassifier(n_estimators=nest, max_features=0.70, max_depth = 450)
clf6 = ExtraTreesClassifier(n_estimators=nest, max_features=0.72, bootstrap = False)
clf7 = ExtraTreesClassifier(n_estimators=nest, max_features=0.72, max_depth = 450, bootstrap = False)

eclf = VotingClassifier(estimators=[('1', clf4), ('2', clf5), ('3', clf6), ('4', clf7)], voting='hard')
```

Finally stood 14th in the Leaderboard.