D68 Final Report
Dog vs Cat classifier

Huan Ma
1000955120

Definition

Task

Build a dog vs cat classifier. A dog vs cat classifier takes input as an image of a dog or cat, and the output will be a statement about whether the input is a picture of a cat or a picture of a dog. This is a supervised learning task, more specifically, a binary classification task, because every picture has a label either dog or cat. We choose 1 to represent dog and -1 to represent cat.

Data

Data is available at https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data. Only the train.zip will be used as our data since the data size of the training data folder is already large enough, i.e. we separate the data in the training data folder as training data, validation data and test data. Training data and validation data are used to train classifiers, test data are used to choose the best performance classifier from 3 algorithms.
The training data folder contains 25,000 images of dogs and cats. The number of images of dogs and cats are equally 12500. Each image in this folder has the label as part of the filename.

Metrics

$$Misclassification\ rate = \frac{1}{n}\Sigma\, I(y_i \neq y_i')$$

where

- n is the number of images in the data subset
- y_i' is the predicted label of the image, 1 for dog, -1 for cat
- y_i is the true label of the image
- $I(y_i \neq y_i')$ is a function that if $y_i \neq y_i'$ is true, then the function equal to 1, equal to 0 otherwise

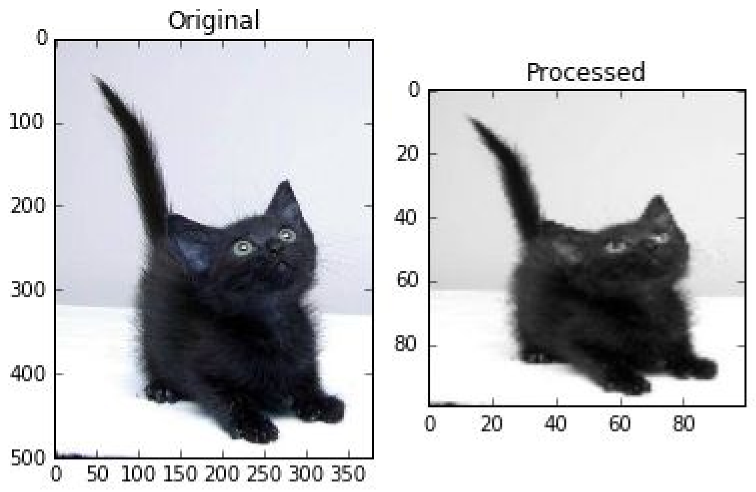
A smaller misclassification rate is better in general.

Methodology

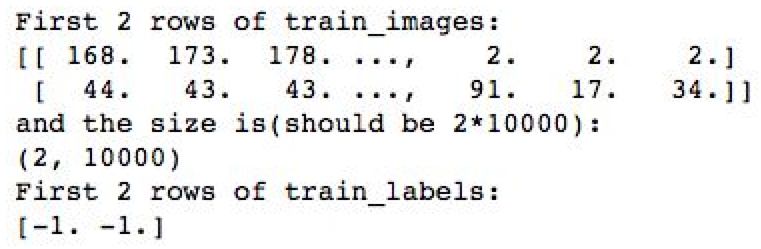
1. Image Preprocessing

i) Normalize and resize the images to a standard format where each image is grayscaled and has size 100*100.

Here is an example of what the images look like now:



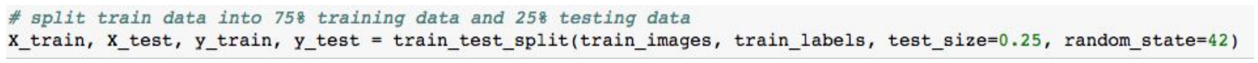
The first 2 rows of the processed data is represented:



```
First 2 rows of train_images:
[[ 168.  173.  178. ...,    2.    2.    2.]
 [  44.   43.   43. ...,   91.   17.   34.]]
and the size is(should be 2*10000):
(2, 10000)
First 2 rows of train_labels:
[-1. -1.]
```

We can notice that the first 2 rows are both cat images because the label is -1.

ii) Split data into training data and testing data by 3:1 randomly, leave testing data for comparing the three classifiers trained by different algorithms later, i.e 75% training data, 25% testing data



```
# split train data into 75% training data and 25% testing data
X_train, X_test, y_train, y_test = train_test_split(train_images, train_labels, test_size=0.25, random_state=42)
```

Size of training data: 18750
Size of cat training data: 9391
Size of dog training data: 9359

Thus, we can confirm that the split is balanced on cats and dogs.

iii) Split the training data into training data and validation data by 4:1 randomly for training by each algorithm.

```
# split training data into 80% training data and 20% validation data
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
```

2. Set up baseline by

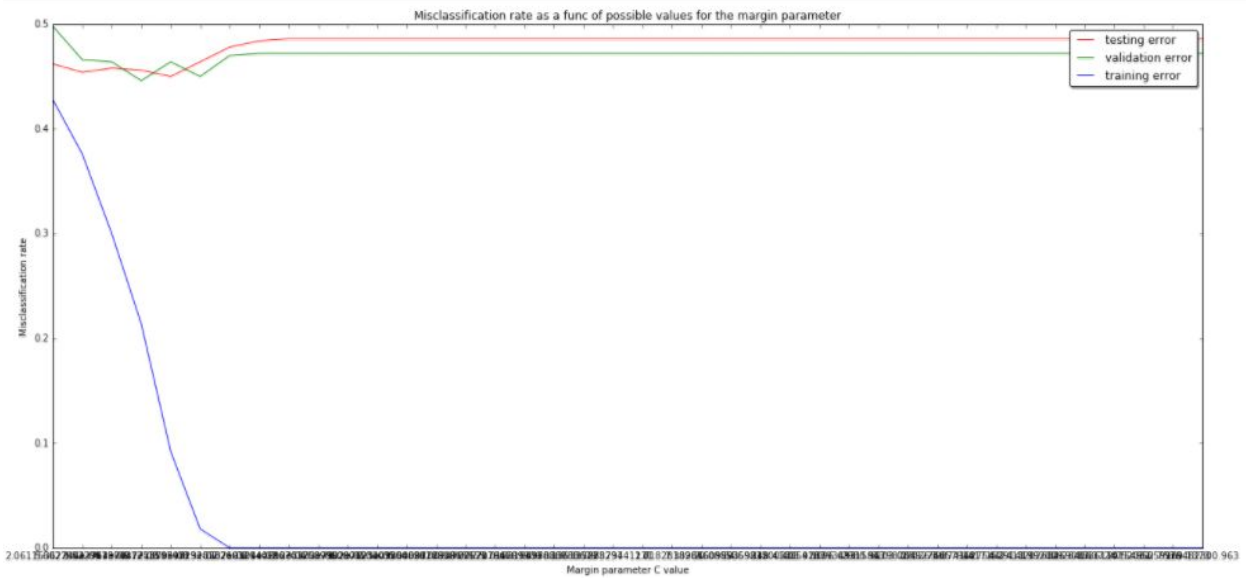
i) learning classifiers by linear kernel SVM, Random Forest and AdaBoost

ii) using k fold Cross Validation and Grid Search to find the optimal parameters for each algorithm

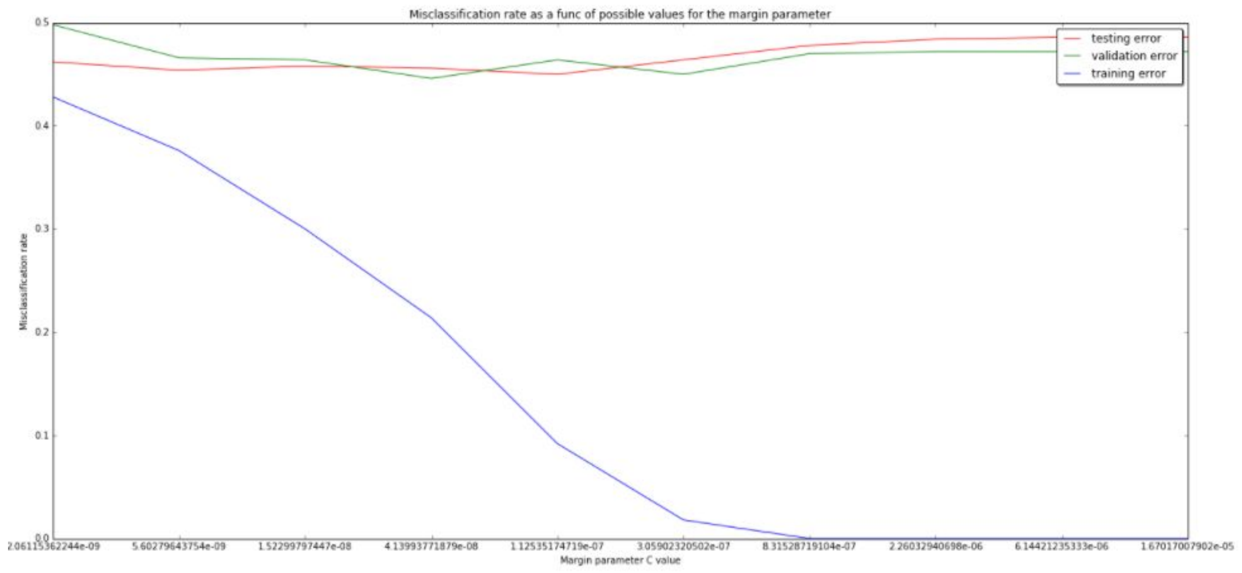
a) Linear SVM

Possible value for margin parameter c: exp(-20), exp(-19), …, exp(19), exp(20)

```
Best value for C:
4.13993771879e-08
which is:
exp(-17.0)
Misclassification rate on validation data for linear kernel SVM:
0.446
Misclassification rate on testing data for linear kernel SVM:
0.456
```



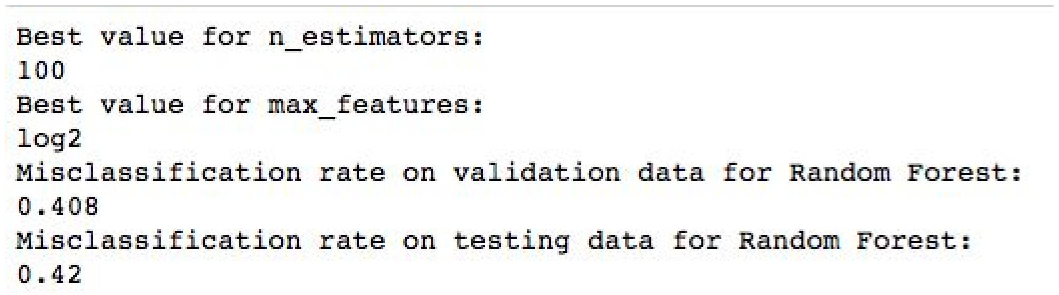
Zoom in the graph to have a better understanding of this plot:

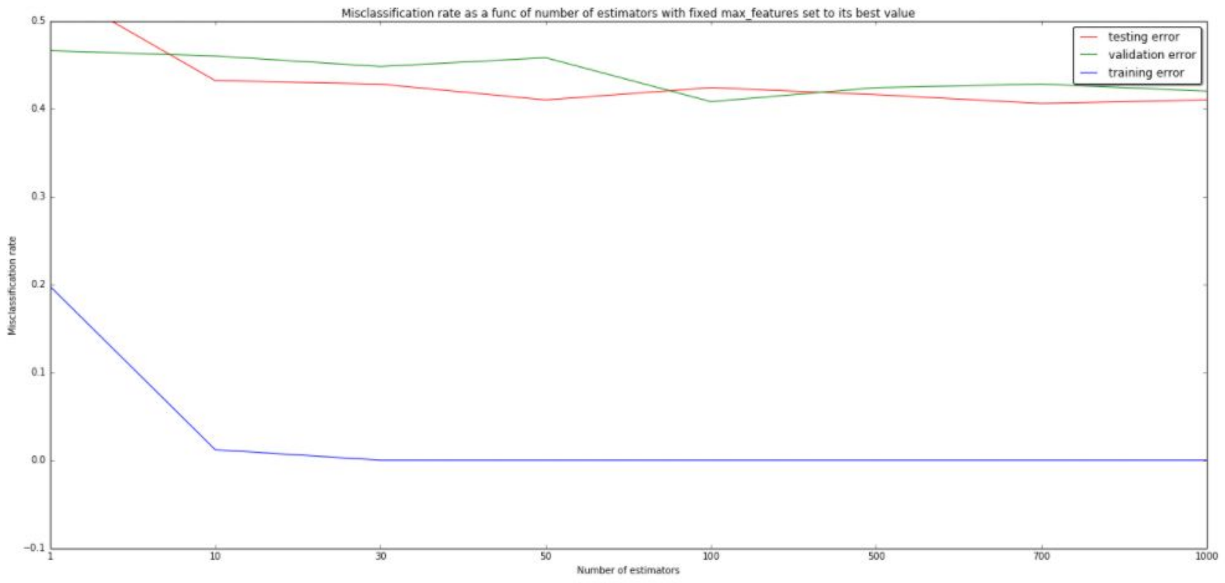


We can confirm the optimal value for the margin parameter is 4.13993771879e-08 which is exp(-17.0) from the plot above as well, at this point, validation error is minimized.

b) Random Forest

Possible values for n_estimators are: [1, 10, 30, 50, 100, 500, 700, 1000], for max_features are: ['auto', 'sqrt', 'log2']


```
Best value for n_estimators:
100
Best value for max_features:
log2
Misclassification rate on validation data for Random Forest:
0.408
Misclassification rate on testing data for Random Forest:
0.42
```



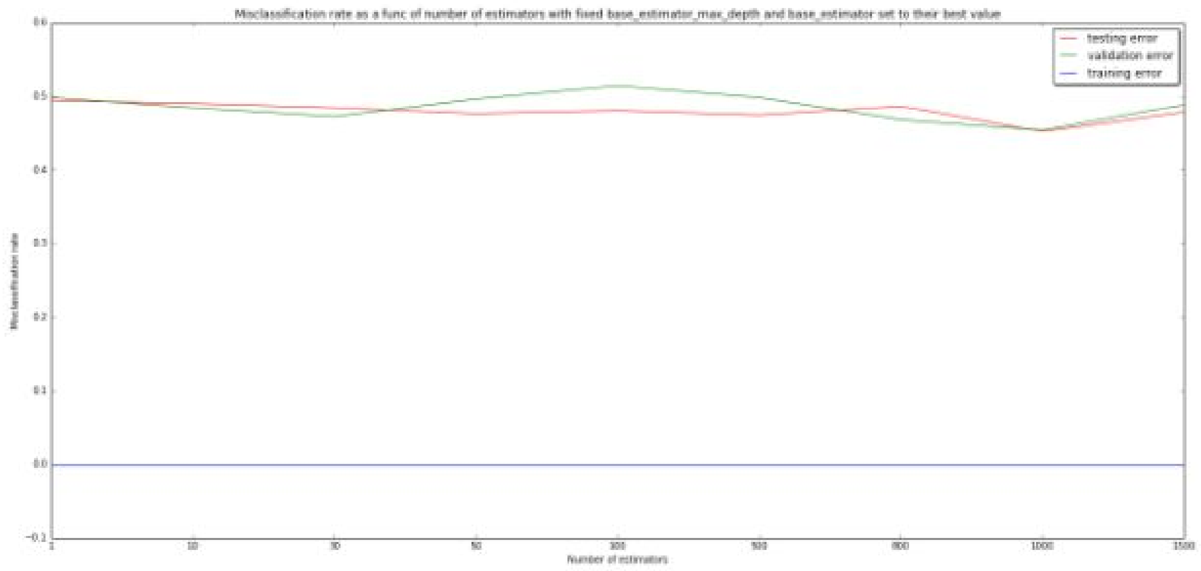
Again, we can notice that when n_estimator = 100, the minimum validation error is achieved.

c) AdaBoost

Possible values for several parameters are as follows:

```
tuned_parameters= {'base_estimator__max_depth':[1,50],
            'base_estimator':[DecisionTreeClassifier(max_features=2),
                              DecisionTreeClassifier(max_features=10)],
                     'n_estimators':[1, 10, 30, 50, 100, 500, 800, 1000, 1500]
```

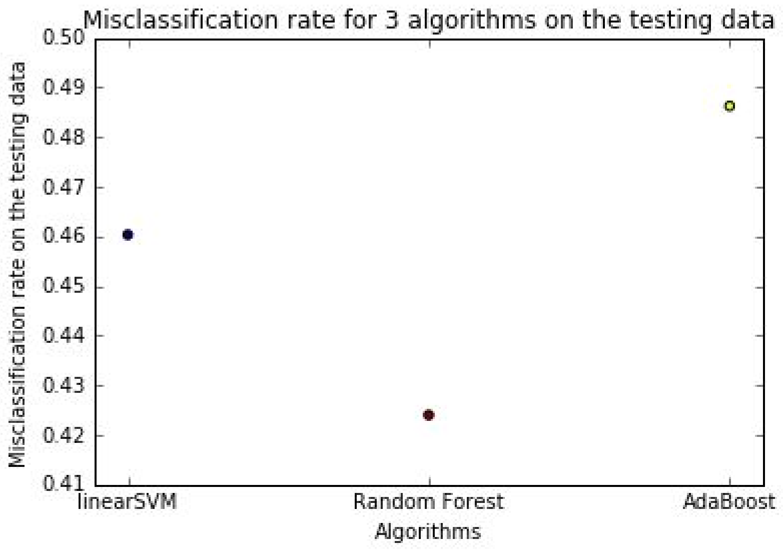
```
Best value for base_estimator__max_depth:
50
Best value for n_estimators:
1000

Best value for base_estimator:
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=Non
e,
            max_features=2, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
Misclassification rate on validation data for AdaBoost:
0.424
Misclassification rate on testing data for AdaBoost:
0.474
```



From the plot above, at the point where n_estimators = 1000, we have the minimum validation error with fixed other parameters at their optimal values.

iii) Compare the performance of the 3 classifiers on the leave-out testing data and choose the classifier which has the lowest misclassification rate.

```
Misclassification rate for linear SVM on the testing data:
0.46032
Misclassification rate for Random Forest on the testing data:
0.424
Misclassification rate for AdaBoost on the testing data:
0.48624
```



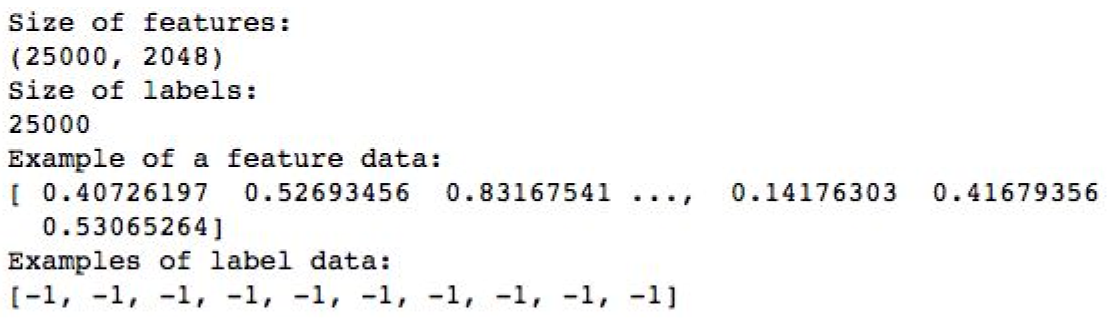
iv) Conclusion

We choose Random Forest classifier with n_estimators = 100, max_features= 'log2' as the baseline. It has the lowest misclassification rate at 0.424 on the leave-out testing data as well as the lowest validation misclassification rate among linear SVM, Random Forest and AdaBoost classifiers.

3. **Learning**

i) Do feature extraction by loading pre-trained neural network Inception-v3 by tensorflow on the data to figure out the weights on the second-to-last layer.

Name extracted features data as features, name label data as labels. Here is some description of the extracted data(notice that the length of features is 2048 now instead of 100*100):



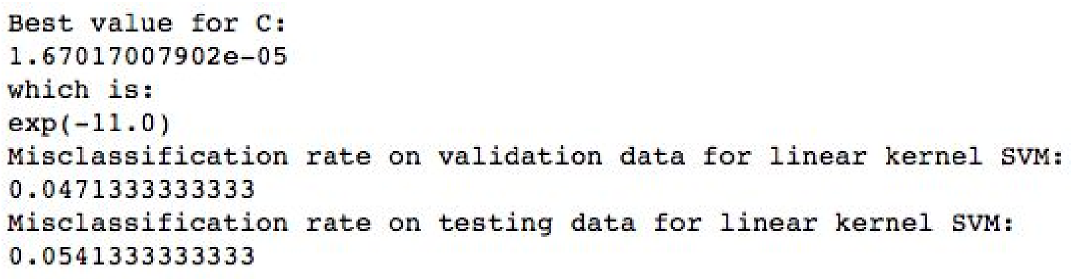
```
Size of features:
(25000, 2048)
Size of labels:
25000
Example of a feature data:
[ 0.40726197  0.52693456  0.83167541 ...,  0.14176303  0.41679356
  0.53065264]
Examples of label data:
[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
```

ii) Learn classifiers by linear kernel SVM, Random Forest and AdaBoost

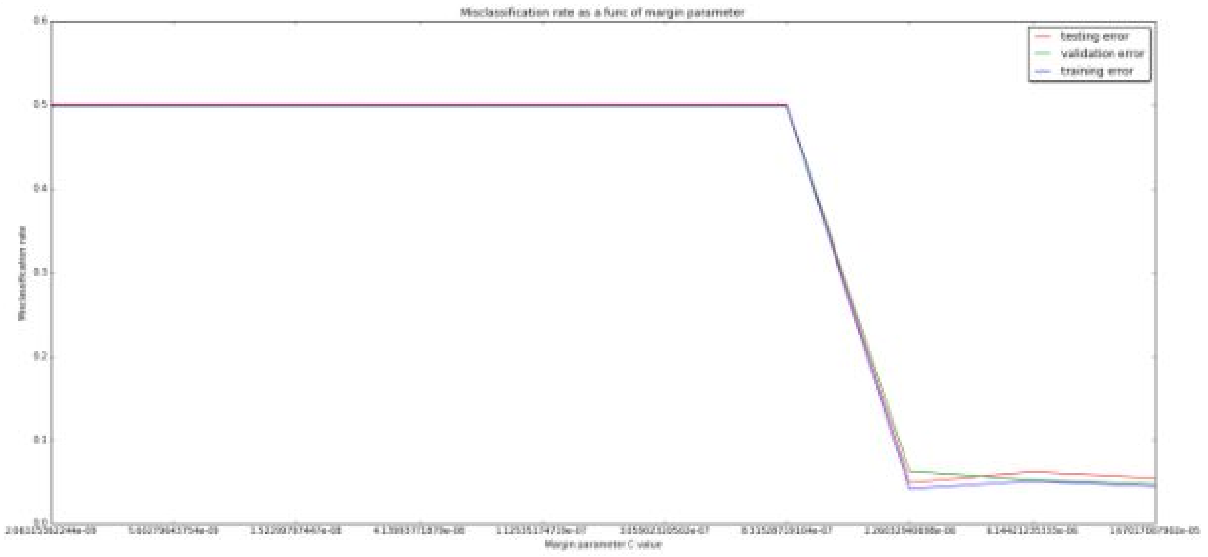
iii) Use k fold Cross Validation and Grid Search to find the optimal parameters for the algorithms

a) linear SVM

Possible value for margin parameter c: exp(-20), exp(-19), …, exp(-11)



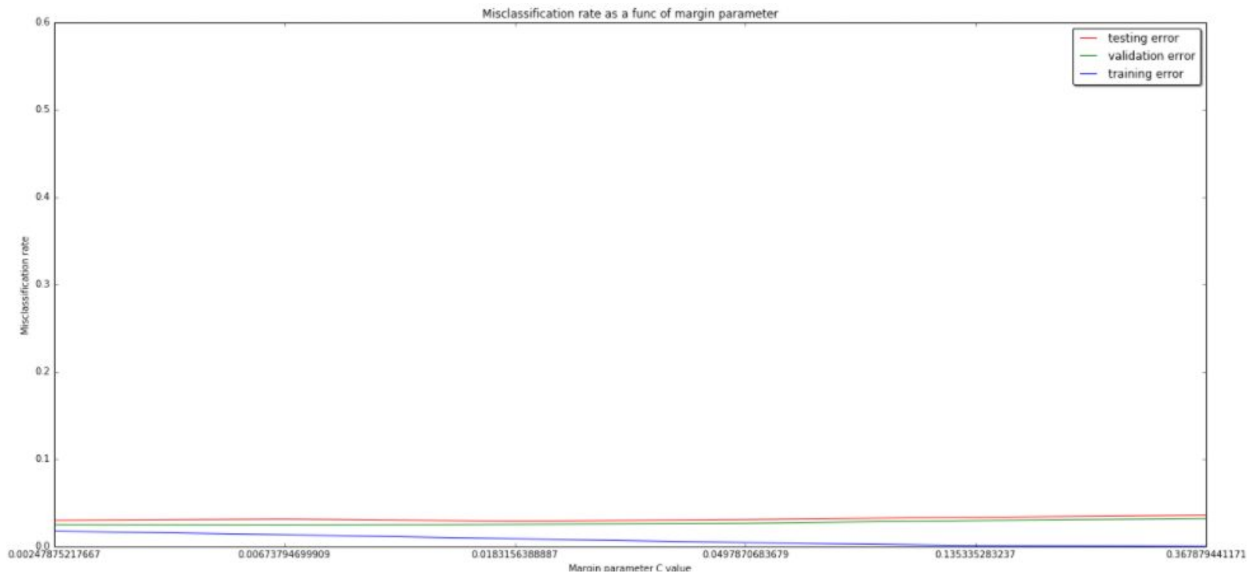
```
Best value for C:
1.67017007902e-05
which is:
exp(-11.0)
Misclassification rate on validation data for linear kernel SVM:
0.0471333333333
Misclassification rate on testing data for linear kernel SVM:
0.0541333333333
```



From the plot, we can notice that the best value chosen by grid search and cross validation is exp(-11) which is the last possible value in the possible value sets for margin parameter. We need to change the possible value sets for margin parameter to check if better value for margin parameter exists.

Choose possible margin parameter values to be exp(-11), exp(-10),....,exp(-1)

```
Best value for C:
0.00673794699909
which is:
exp(-5.0)
Misclassification rate on validation data for linear kernel SVM:
0.0242
Misclassification rate on testing data for linear kernel SVM:
0.0312
```



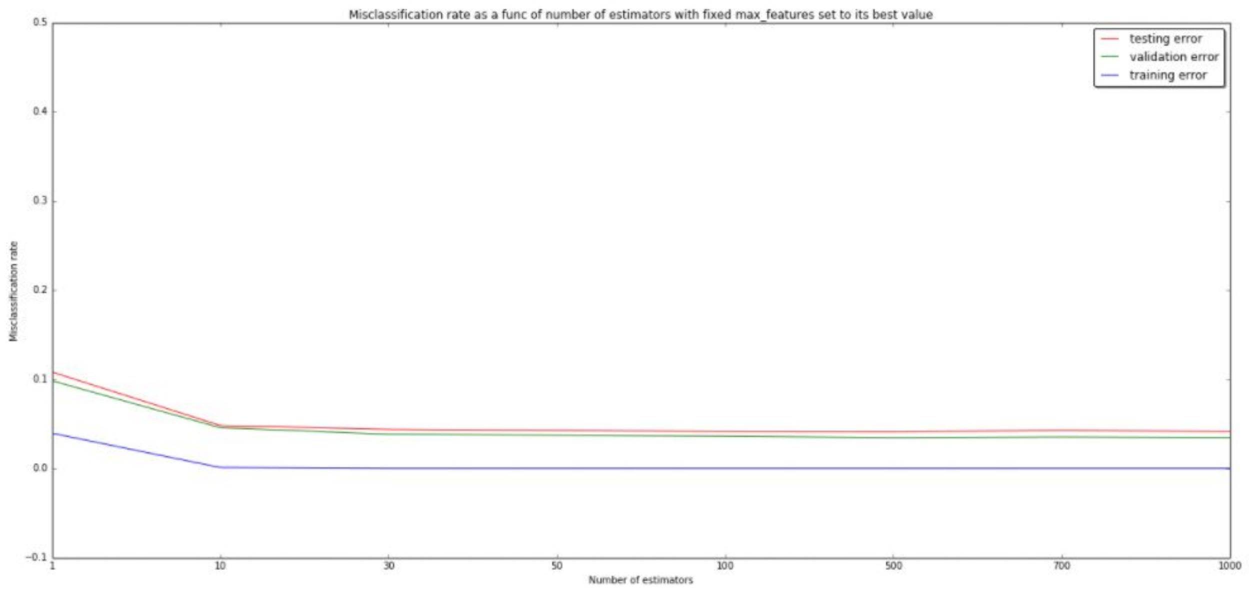
The lines in the plot are flat but we can still observe that the minimum validation error occurs at 0.00673794699909 which is exp(-5).

b) Random Forest

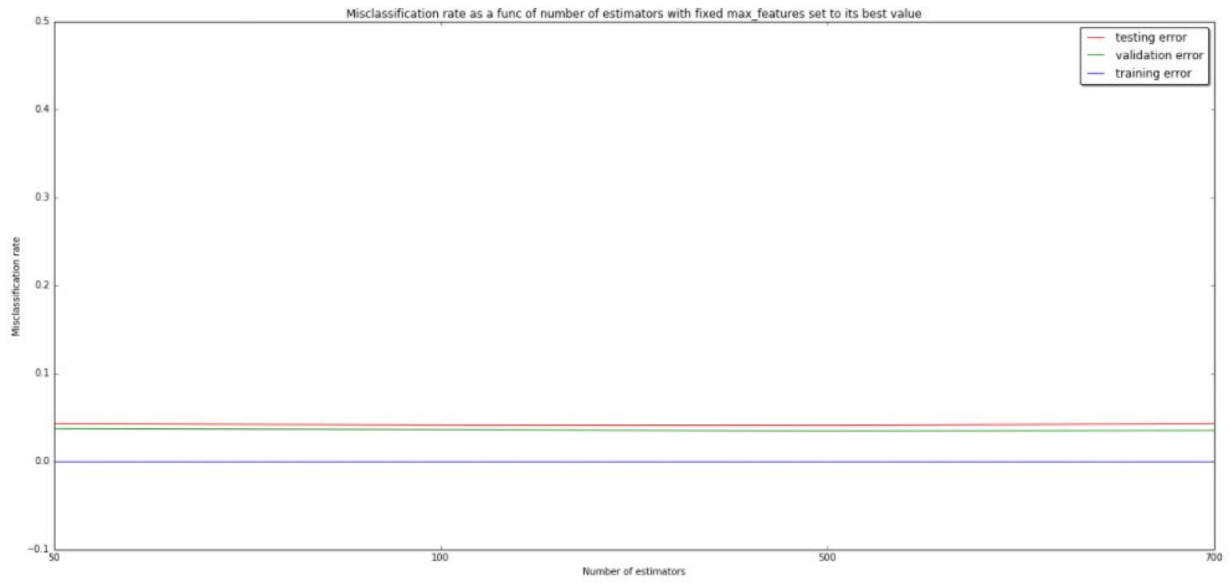
Possible values are:

```
tuned_parameters={'n_estimators': [1, 10, 30, 50, 100, 500, 700, 1000],
    'max_features': ['auto', 'sqrt', 'log2']}
```

```
Best value for n_estimators:
500
Best value for max_features:
sqrt
Misclassification rate on validation data for Random Forest:
0.0347333333333
Misclassification rate on testing data for Random Forest:
0.0410666666667
```



Zoom in this plot to check if our chosen optimal values by grid search and cross validation are proper or not:



The three lines are still quite flat, the reason may be that the difference between each error on different number of estimators are quite small. This assumption is proved by displaying the values of the validation error directly.

```
[1, 10, 30, 50, 100, 500, 700, 1000]
[ 0.0982      0.0456      0.0382      0.03713333  0.036       0.03446667
  0.0352      0.0346    ]
When n_estimators = 500, the validation error is minimized which is 0.03446667
```

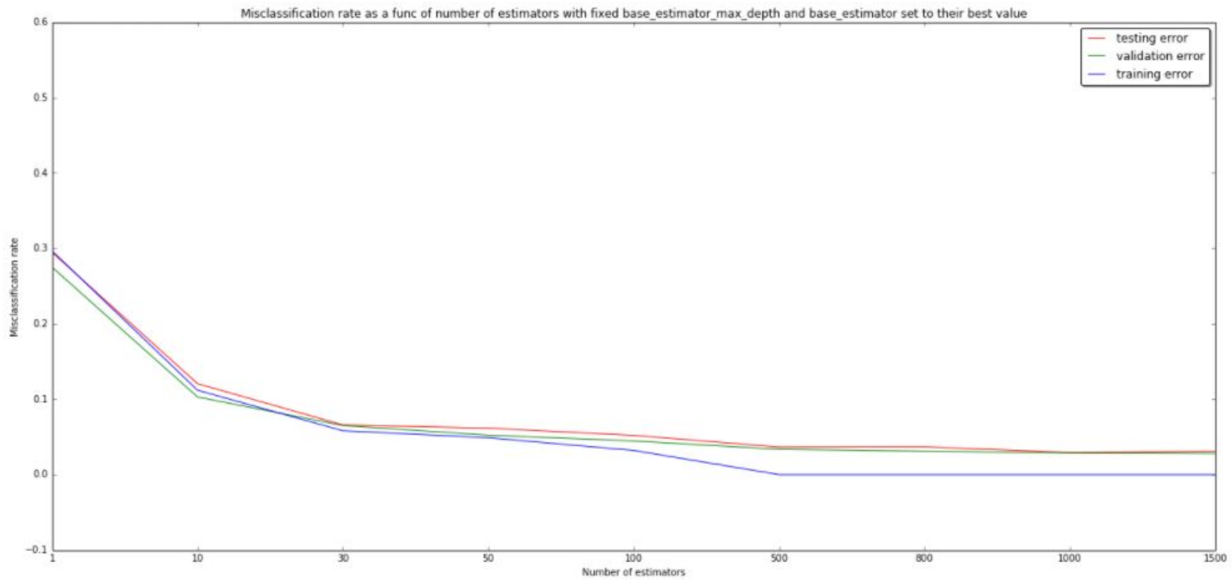
Thus, we can conclude that the chosen value by grid search and cross validation is indeed optimal.

c) AdaBoost

Possible values are:

```
tuned_parameters= {'base_estimator__max_depth':[1],
            'base_estimator':[DecisionTreeClassifier(max_features=2),
                              DecisionTreeClassifier(max_features=10)],
                   'n_estimators':[1, 10, 30, 50, 100, 500, 800, 1000, 1500]}
```

```
Best value for base_estimator__max_depth:
1
Best value for n_estimators:
1500

Best value for base_estimator:
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=10, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
Misclassification rate on validation data for AdaBoost:
0.114733333333
Misclassification rate on testing data for AdaBoost:
0.122933333333
```

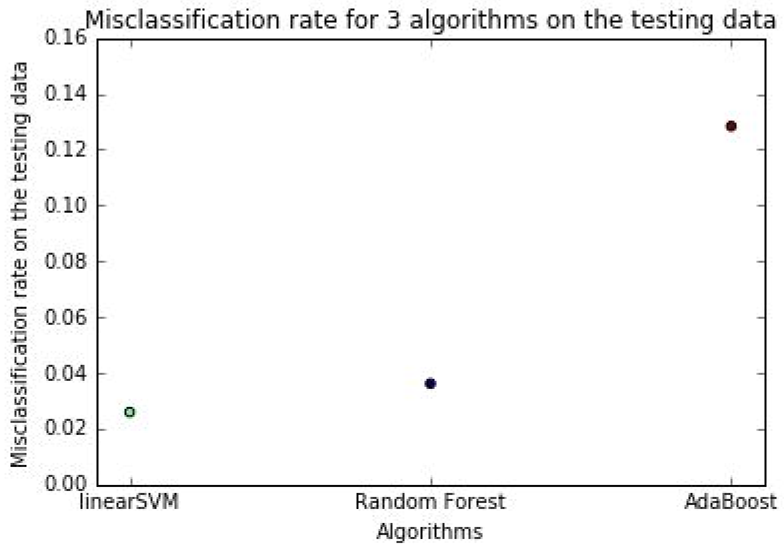


We can observe that the validation error rate is minimized at n_estimators = 1500.

4. Conclusion

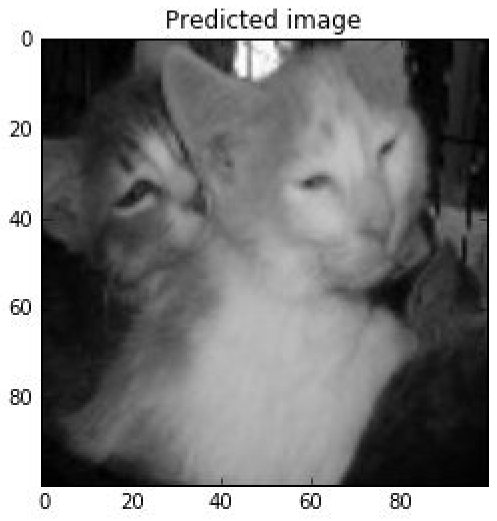
Compare the test error rates on leave-out testing data of the three optimal classifiers by linear SVM, Random Forest and AdaBoost.

```
Misclassification rate for linear SVM on the testing data:
0.02592
Misclassification rate for Random Forest on the testing data:
0.03616
Misclassification rate for AdaBoost on the testing data:
0.12832
```

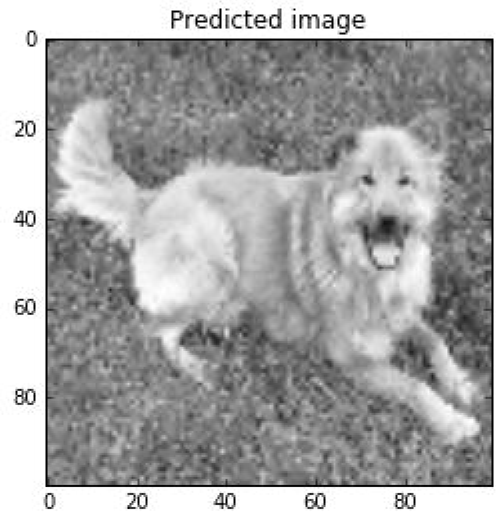


Clearly, linear SVM has a better performance. It has the lowest misclassification rate on the leave-out testing data at 0.02592 among linear SVM, Random Forest and AdaBoost classifiers and 0.02592 is much lower than the misclassification rate of the baseline which is 0.424. We choose the linear SVM classifier to be our optimal classifier where c=exp(-5).

Here are some examples using our classifier to determine if the image is a dog image or a cat image.



```
Processing /Users/lordlavon/Desktop/input/test/16.jpg...
'This is more like a cat image.'
```



```
Processing /Users/lordlavon/Desktop/input/test/4.jpg...
'This is more like a dog image.'
```

Code

Look at ‘D68FinalProject.ipynb’