1(a) At each iteration t, for each weak learner G_t, we have a specific γ_t, $\epsilon_t = \Pr_{x\sim w_t}[G_t(x) \neq y] = \frac{1}{2} - \gamma_t \leq \frac{1}{2} - \gamma$

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(g(x_i) \neq y_i) \leq \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i G(x_i)) = \prod_{t=1}^{T} Z_t$$

Training error of final classifier

In order to minimize training error, we want to find optimal α_t that minimize its upper bound

$$Z_t(\alpha_t) = \sum_{i=1}^{N} w_t(i)\cdot \exp(-\alpha_t y_i G_t(x_i))$$

$$= e^{-\alpha_t}\sum_{y=G_t(x_i)} w_t(i) + e^{\alpha_t}\sum_{y\neq G_t(x_i)} w_t(i)$$

$$\frac{\partial Z_t(\alpha_t)}{\partial \alpha_t} = e^{-\alpha_t}\cdot(-1)\cdot\sum_{y=G_t(x)} w_t(i) + e^{\alpha_t}\sum_{y\neq G_t(x)} w_t(i) = 0$$

$$\sum_{y\neq G_t(x)} w_t(i) = e^{-2\alpha_t}\sum_{y=G_t(x)} w_t(i)$$

$$\epsilon_t = \frac{\sum_{i=1}^{N} w_t(i)\,\mathbb{1}(y_i \neq G_t(x_i))}{\sum_{i=1}^{N} w_t(i)} = \sum_{y\neq G_t(x)} w_t(i) = 1 - \sum_{y=G_t} w_t(i)$$

→ We get $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$

$$Z_t = e^{-\alpha_t}\sum_{y=G_t(x_i)} w_t(i) + e^{\alpha_t}\sum_{y\neq G_t(x_i)} w_t(i)$$

$$= e^{-\alpha_t}\cdot(1-\epsilon_t) + e^{\alpha_t}\cdot\epsilon_t$$

$$= 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

$\because \epsilon_t = \frac{1}{2} - \gamma_t \leq \frac{1}{2} - \gamma$, where $\gamma_t \geq \gamma$

$$Z_t = \sqrt{1-\gamma_t^2}$$

total training error:

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(g(x_i)\neq y_i) \leq \prod_{t=1}^{T}\sqrt{1-\gamma_t^2} \leq \exp\left(-\frac{1}{2}\sum_t \gamma_t^2\right)$$

$$\exp\left(-\frac{1}{2}\sum_t \gamma_t^2\right) \leq \exp\left(-\frac{1}{2}\sum_t \gamma^2\right) = \exp\left(-\frac{1}{2}T\cdot\gamma^2\right)$$

$$\therefore \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(g(x_i)\neq y_i) \leq \exp\left(-\frac{\gamma^2}{2}T\right)$$

1.(b) Yes. $0<\gamma\leq 1$, $0<e^{-\frac{\gamma^2}{2}}<1$ since $\left(e^{-\frac{\gamma^2}{2}}\right)^T$ serve as an upper bound of total training loss, as T increase i.e., when adding more weak classifiers, total training loss converges to zero.

2.(a) I will use $f(\alpha\cdot a_1+(1-\alpha)a_2)\leq \alpha f(a_1)+(1-\alpha)\cdot f(a_2)$ as definition of convexity to justify. Since convexity is preserved under summation of nonnegative terms, we only need show convexity inside $\sum_{i=1}^{n}$.
We can try counterexample in terms of $\alpha_1=0.5$, $a_1=0$, $a_2=2$, $y=0$

left hand side $=\left(0-\frac{1}{1+\exp(-1)}\right)^2\approx 0.53$

right hand side $=0.5\left(0-\frac{1}{1+\exp(0)}\right)^2+0.5\left(0-\frac{1}{1+\exp(2)}\right)^2\approx 0.51$

$0.53>0.51$ which violates inequality $\Rightarrow$ not convex

2.(b) Similarly, we will ~~try~~ to find a contereexample to check the convexity inside $\sum_{i=1}^{n}$, e.g. $\alpha=0.5$, $a_1=-1$, $a_2=1$, $y=1$

left hand side $=(1-\max(0,0.5\times -1+0.5\times 1))=1$
right hand side $=0.5\times(1-\max(0,1))^2+0.5\times(1-\max(0,-1))^2=0.5$
$1>0.5$, which violates inequality $\Rightarrow$ not convex

3 (i) $g_m(x_i)=\left[\frac{\partial L(y_i,F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

$L(Y,F)=\sum_{i=1}^{n}-y_i\cdot F(x_i)+\log(1+\exp(y_i F(x_i)))$

$g_m(x_i)=-y_i F_{m-1}(x_i)+\frac{\exp(y_i\cdot F_{m-1}(x_i))\cdot y_i}{1+\exp(y_i F_{m-1}(x_i))}$

In order to obtain $h(x,a_m)$, we need to find a_m, s.t.

$a_m=\arg\min_{a,\beta}\sum_{i=1}^{N}(-g_m(x_i)-\beta h(x_i,a))^2$ $\Rightarrow$ optimization problem

$=\arg\min_{a,\beta}\sum_{i=1}^{N}\left(y_i F_{m-1}(x_i)-\frac{y_i\cdot\exp(y_i F_{m-1}(x_i))}{1+\exp(y_i\cdot F_{m-1}(x_i))}-\beta\cdot h(x_i;a)\right)^2$

3(ii) In order to obtain step size ρ_m, we need to solve

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, \bar{F}_{m-1}(x_i) + \rho_m h(x_i; a_m))$$

$$\sum_{i=1}^{N} L(y_i, \bar{F}_{m-1}(x_i) + \rho_m h(x_i; a_m))$$

$$= \sum_{i=1}^{N} y_i (\bar{F}_{m-1}(x_i) + \rho h(x_i; a_m)) + \log(1 + \exp(y_i \bar{F}_{m-1}(x_i) + y_i \rho h(x_i, a_m)))$$

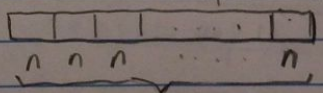
$$\frac{\partial L}{\partial \rho} = \sum_{i=1}^{N} y_i h(x_i; a_m) + \frac{\exp(y_i \bar{F}_{m-1}(x_i)) \times \exp(y_i \rho h(x_i; a_m)) \times y_i h(x_i; a_m)}{1 + \exp(y_i \bar{F}_{m-1}(x_i)) \times \exp(y_i \rho h(x_i; a_m))}$$

$$= 0$$

We can see both numerator and denominator have ρ and take a sum. It is a nonlinear function with respect to ρ, so we can NOT find a closed form solution

EC1(a)
```
def fun(X, K):
    Z = zeros((n-2, m-2))
    for i = 0, 1, ... n-3
        for j = 0, 1, ... m-3
            Z[i, j] = sum_{ñ=0}^{2} sum_{m̃=0}^{2} X[i+ñ, j+m̃] × K[ñ, m̃]
    return Z
```

(b) Since we count index from 0, for row index i of A, $i = 0, 1, \cdots (n-2) \times (m-2)$



m blocks, each block is of size n

Kernel size is 3×3.

First $\lfloor \frac{i}{n-2} \rfloor$ blocks are zeros

Then we can see a repeating pattern which repeats 9 = 1, 2, 3 blocks

1) $i \% (n-2)$ are zeros
2) then are $K(:, 9)^T$
3) followed by $n - 3 - i \% (n-2)$

The rest $(m-3) - \lfloor \frac{i}{n-2} \rfloor$ blocks are zeros

4.
(a)Adaboost:
Algorithm:

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

During we generate weak learners, we use maximum weighted information gain as criteria. In concrete,

- The entropy of a r.v. X with distribution $(p(x_1), \ldots, p(x_n))$

$$H(X) = \sum_{i=1}^{n} -p(x_i) \log_2 p(x_i)$$

- The conditional entropy

$$H(X|Y) = \sum_{j=1}^{m} p(y_j) H(X|y_j)$$

- Information Gain

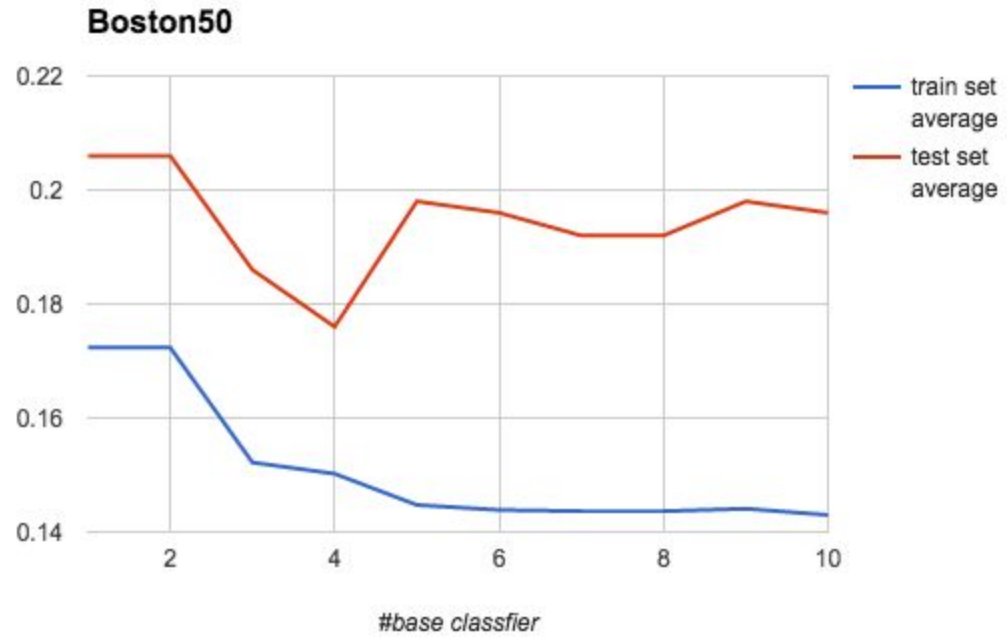
$$IG(X|Y) = H(X) - H(X|Y)$$

However, when we calculate p(x_i) or p(y_j), we do not use count(x_i)/sum_over_i(count(x_i)) as p(x_i). Instead, we use p(x_i)=weight(x_i)/sum_over_i(weight_i), p(j_i)=weight(x_j)/sum_over_j(weight_j)

For each feature(other than 4th feature), we first generate split candidates by enumerating feature in percentiles. We compare the information gain respect to different split candidates(including 4th feature) to obtain the best split for each feature and then use information gain respect to best splits of different features to get best feature and split. We can use them to binary split dataset.
After each split, we obtain two sub-datesets, representing left and right nodes, which, along with their weights, can be feed into information gain generator to obtain next node(best feature and split).
When we go to leaf, we use alpha_t and h_t to obtain weighted sum and compare with 0 in order to get predicted class. Then compare the class we predict with the original target information, count the number of error then add all leaves up and divided by N = final_error_rate.
For boston50, B=[1,2,3,4,5,6,7,8,9,10]

boston50	b=1	2	3	4	5	6	7	8	9	10
folder=0	0.17324561	0.17324561	0.14692982	0.14692982	0.12719298	0.125	0.125	0.125	0.125	0.11403509
	0.28	0.28	0.14	0.14	0.2	0.2	0.2	0.2	0.2	0.18
1	0.13157895	0.13157895	0.14912281	0.13157895	0.13157895	0.13157895	0.13157895	0.13157895	0.13157895	0.13157895
	0.18	0.18	0.22	0.14	0.18	0.16	0.16	0.16	0.16	0.16
2	0.19298246	0.19298246	0.14692982	0.14692982	0.14692982	0.14692982	0.14692982	0.14692982	0.14692982	0.14692982
	0.14	0.14	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
3	0.15789474	0.15789474	0.15789474	0.15789474	0.15789474	0.15789474	0.15789474	0.15789474	0.15789474	0.15789474
	0.26	0.26	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16
4	0.15350877	0.15350877	0.13815789	0.13815789	0.13815789	0.13815789	0.13815789	0.13815789	0.13815789	0.13815789
	0.4	0.4	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34
5	0.16447368	0.16447368	0.16885965	0.16885965	0.16885965	0.16885965	0.16885965	0.16885965	0.16885965	0.16885965
	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
6	0.1622807	0.1622807	0.1622807	0.13596491	0.14254386	0.13596491	0.13377193	0.13377193	0.13815789	0.13815789
	0.28	0.28	0.28	0.26	0.38	0.38	0.34	0.34	0.4	0.4
7	0.20175439	0.20175439	0.14254386	0.14254386	0.14254386	0.14254386	0.14254386	0.14254386	0.14254386	0.14254386

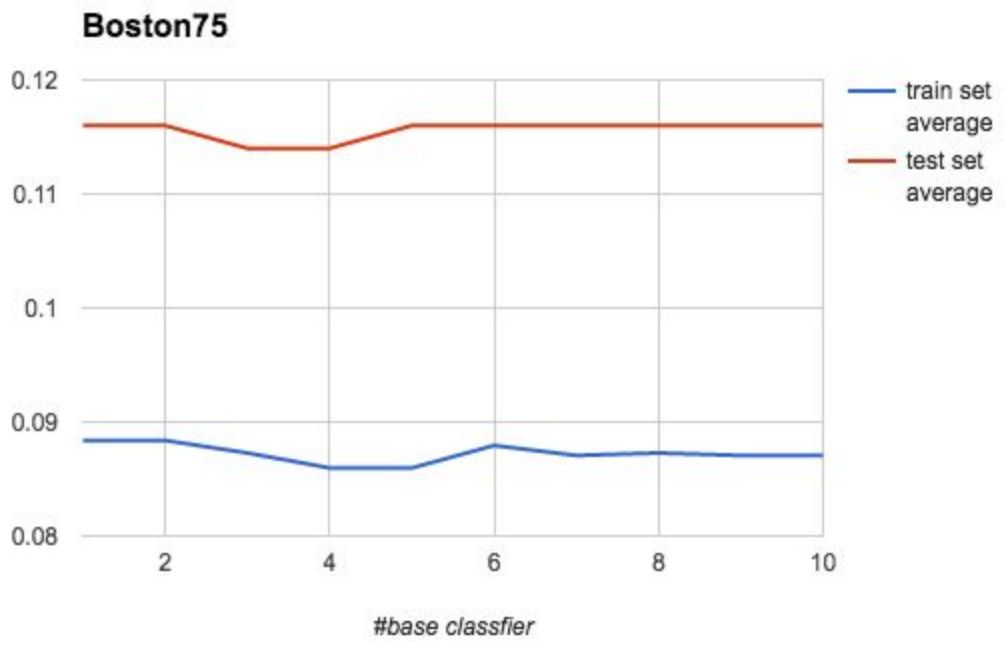
	0.12	0.12	0.24	0.24	0.24	0.24	0.24	0.24	0.24	0.24
8	0.20394737	0.20394737	0.15570175	0.17982456	0.13815789	0.13815789	0.13815789	0.13815789	0.13815789	0.13815789
	0.08	0.08	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
9	0.18201754	0.18201754	0.15350877	0.15350877	0.15350877	0.15350877	0.15350877	0.15350877	0.15350877	0.15350877
	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22	0.22
train set average	0.17236842	0.17236842	0.15219298	0.1502193	0.14473684	0.14385965	0.14364035	0.14364035	0.14407895	0.14298246
test set average	0.206	0.206	0.186	0.176	0.198	0.196	0.192	0.192	0.198	0.196
train set std	0.02184633	0.02184633	0.00883747	0.01438871	0.01189074	0.01249038	0.01264535	0.01264535	0.0123685	0.01434016
test set std:	0.09551963	0.09551963	0.08581375	0.08380931	0.09897474	0.09951884	0.0926067	0.0926067	0.10332473	0.10346014



For boston75

boston75	b=1	2	3	4	5	6	7	8	9	10
folder=0	0.08333333	0.08333333	0.08333333	0.08333333	0.09868421	0.09868421	0.09868421	0.09868421	0.09868421	0.09868421
	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
1	0.09210526	0.09210526	0.09210526	0.09210526	0.09210526	0.09210526	0.09210526	0.09210526	0.09210526	0.09210526
	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08	0.08
2	0.09649123	0.09649123	0.09649123	0.09649123	0.09649123	0.09649123	0.09649123	0.09649123	0.09649123	0.09649123
	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
3	0.08991228	0.08991228	0.07894737	0.07894737	0.07894737	0.07894737	0.07894737	0.07894737	0.07894737	0.07894737
	0.2	0.2	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
4	0.0877193	0.0877193	0.0877193	0.0877193	0.0877193	0.0877193	0.0877193	0.0877193	0.0877193	0.0877193
	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
5	0.09210526	0.09210526	0.09210526	0.09210526	0.09210526	0.09649123	0.09210526	0.09210526	0.09210526	0.09210526
	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
6	0.07894737	0.07894737	0.07894737	0.07894737	0.06359649	0.0745614	0.0745614	0.0745614	0.0745614	0.0745614
	0.18	0.18	0.18	0.18	0.2	0.2	0.2	0.2	0.2	0.2
7	0.08114035	0.08114035	0.08114035	0.08114035	0.08114035	0.08114035	0.08114035	0.08114035	0.08114035	0.08114035
	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16	0.16
8	0.09210526	0.09210526	0.09210526	0.08552632	0.08552632	0.0877193	0.0877193	0.0877193	0.0877193	0.0877193
	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
9	0.08991228	0.08991228	0.08991228	0.08333333	0.08333333	0.08552632	0.08114035	0.08333333	0.08114035	0.08114035
	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
train set average	0.08837719	0.08837719	0.0872807	0.08596491	0.08596491	0.0879386	0.0870614	0.0872807	0.0870614	0.0870614

test set average	0.116	0.116	0.114	0.114	0.116	0.116	0.116	0.116	0.116	0.116
train set std	0.00528595	0.00528595	0.00594941	0.00570175	0.00964912	0.00768796	0.00753633	0.00739136	0.00753633	0.00753633
test set std:	0.05782733	0.05782733	0.05517246	0.05517246	0.05782733	0.05782733	0.05782733	0.05782733	0.05782733	0.05782733



Discussion:
Test set average error is higher than train set, which is normal, since test set is unseen dataset. When we increase the number of base classifier, the average error rate goes down at first. Because more base classifiers can compensate existing base classifiers' shortcoming and . However, because each weak learner is restricted as two layer tree, we may can only five two layer trees that can help compensating the shortcoming. when we increase the number of base classifiers, the last base classifier has virtually zero weight and make little contribution or even noise direction. So the right part of curve remains the same or goes up.
(b)
Random Forest:
Algorithm:

- Create k bootstrap samples $S^1, \ldots, S^k$
- Learn an un-pruned decision tree on each sample
- Learning: At each internal node
  - Randomly select $m < d$ features
  - Determine the best split using only these features
- Prediction: Use output from all trees in the forest
  - Classification: Majority vote

If we have dataset X, bootstrap samples are samples of size N drawing from X with replacement.

- The entropy of a r.v. X with distribution $(p(x_1), \ldots, p(x_n))$

$$H(X) = \sum_{i=1}^{n} -p(x_i) \log_2 p(x_i)$$

- The conditional entropy

$$H(X|Y) = \sum_{j=1}^{m} p(y_j) H(X|y_j)$$

- Information Gain

$$IG(X|Y) = H(X) - H(X|Y)$$

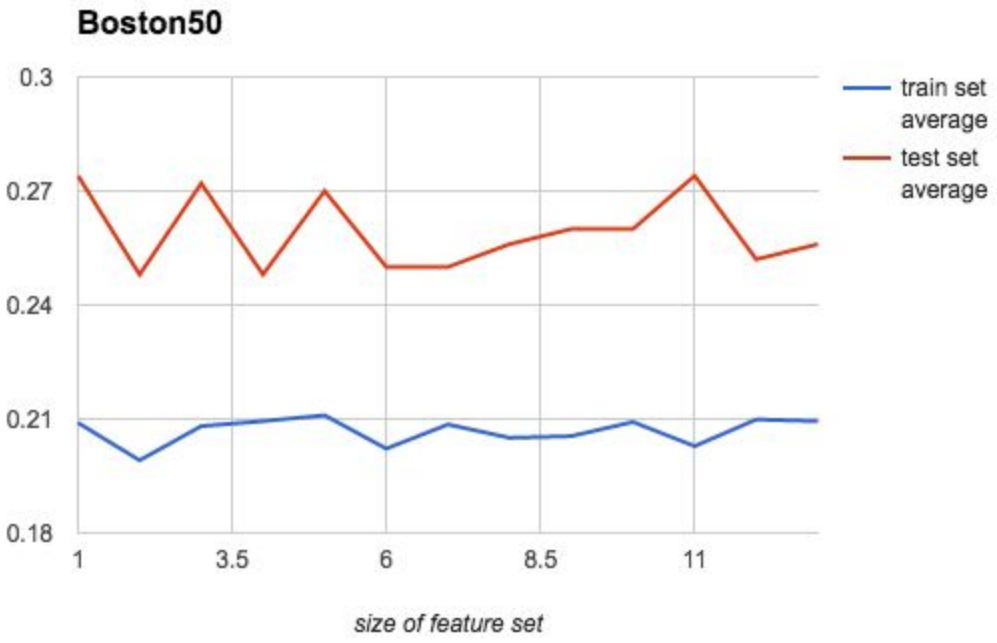
When we calculate p(x_i) or p(y_j), we use count(x_i)/sum_over_i(count(x_i)) as p(x_i) and p(y_j)=count(x_j)/sum_over_j(count(y_j)).

For each bootstrap sample, we generate a bunch of trees.
For each tree, we generated nodes layer by layer. For each node, we enumerate features into multiple split and calculate information gain using foregoing formula. And compare information gain of best split of different features
After we have forest, we can classify it by equal weighted vote of different trees.
For boston50,

boston50	m=1	2	3	4	5	6	7	8	9	10	11	12	13
folder=0	0.21491228	0.19078947	0.20833333	0.20394737	0.20175439	0.18859649	0.21491228	0.2127193	0.19517544	0.19517544	0.1995614	0.21052632	0.21710526

	0.56	0.46	0.54	0.48	0.38	0.26	0.38	0.38	0.38	0.36	0.48	0.46	0.4
1	0.203 94737	0.192 98246	0.186 40351	0.210 52632	0.208 33333	0.184 21053	0.208 33333	0.203 94737	0.184 21053	0.206 14035	0.190 78947	0.212 7193	0.199 5614
	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26	0.26
2	0.217 10526	0.219 29825	0.217 10526	0.212 7193	0.228 07018	0.219 29825	0.217 10526	0.210 52632	0.217 10526	0.223 68421	0.219 29825	0.214 91228	0.223 68421
	0.26	0.12	0.2	0.14	0.36	0.3	0.16	0.2	0.36	0.36	0.24	0.22	0.32
3	0.201 75439	0.184 21053	0.214 91228	0.212 7193	0.212 7193	0.199 5614	0.221 49123	0.206 14035	0.203 94737	0.221 49123	0.199 5614	0.214 91228	0.208 33333
	0.28	0.32	0.26	0.28	0.28	0.3	0.28	0.3	0.28	0.24	0.28	0.3	0.3
4	0.208 33333	0.214 91228	0.201 75439	0.219 29825	0.228 07018	0.228 07018	0.208 33333	0.208 33333	0.206 14035	0.214 91228	0.212 7193	0.217 10526	0.225 87719
	0.16	0.12	0.32	0.12	0.26	0.16	0.24	0.26	0.16	0.18	0.26	0.16	0.16
5	0.223 68421	0.195 17544	0.212 7193	0.214 91228	0.225 87719	0.217 10526	0.217 10526	0.225 87719	0.230 26316	0.208 33333	0.214 91228	0.208 33333	0.208 33333
	0.18	0.12	0.16	0.16	0.1	0.14	0.12	0.1	0.16	0.16	0.16	0.14	0.16
6	0.192 98246	0.188 59649	0.192 98246	0.186 40351	0.192 98246	0.173 24561	0.188 59649	0.190 78947	0.186 40351	0.190 78947	0.188 59649	0.195 17544	0.199 5614
	0.38	0.44	0.4	0.38	0.42	0.44	0.44	0.44	0.36	0.42	0.42	0.4	0.36
7	0.206 14035	0.192 98246	0.201 75439	0.199 5614	0.179 82456	0.199 5614	0.179 82456	0.186 40351	0.199 5614	0.195 17544	0.201 75439	0.210 52632	0.192 98246
	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34	0.34
8	0.210 52632	0.225 87719	0.230 26316	0.230 26316	0.230 26316	0.212 7193	0.217 10526	0.210 52632	0.221 49123	0.234 64912	0.214 91228	0.214 91228	0.208 33333
	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
9	0.210 52632	0.186 40351	0.214 91228	0.203 94737	0.201 75439	0.199 5614	0.212 7193	0.195 17544	0.210 52632	0.201 75439	0.186 40351	0.199 5614	0.210 52632
	0.28	0.26	0.2	0.28	0.26	0.26	0.24	0.24	0.26	0.24	0.26	0.18	0.22
train set average	0.208 99123	0.199 12281	0.208 11404	0.209 42982	0.210 96491	0.202 19298	0.208 55263	0.205 04386	0.205 48246	0.209 21053	0.202 85088	0.209 86842	0.209 42982
test set average	0.274	0.248	0.272	0.248	0.27	0.25	0.25	0.256	0.26	0.26	0.274	0.252	0.256
train set std	0.008 14952	0.014 2392	0.012 10916	0.011 27841	0.016 28723	0.016 31673	0.012 91628	0.011 0196	0.014 11197	0.013 56107	0.011 40562	0.006 79825	0.010 01355

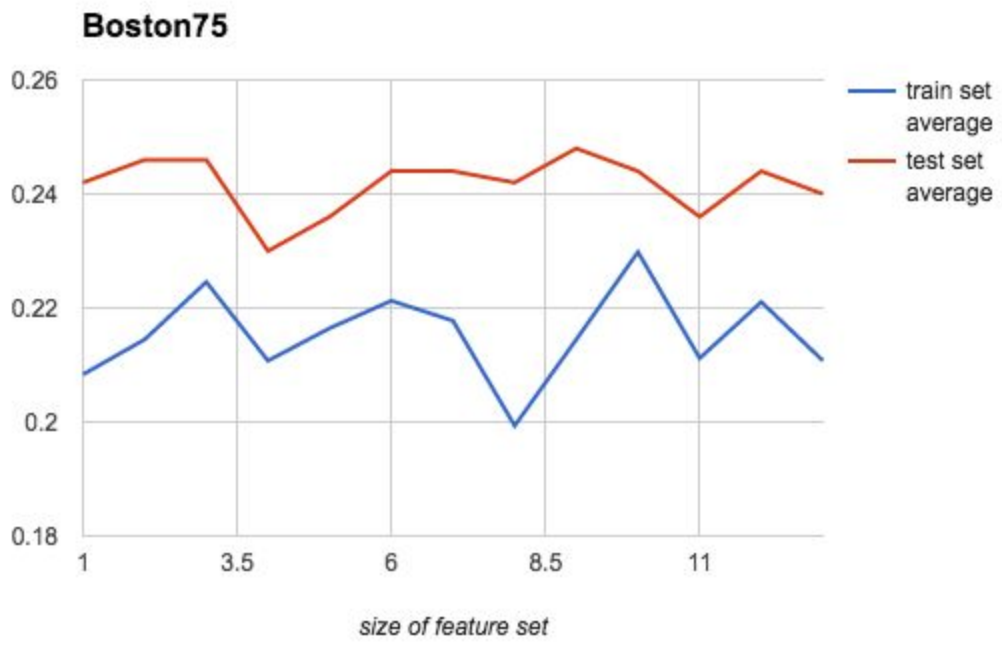
test set std:	0.132 07574	0.137 17143	0.131 20976	0.126 87001	0.113 9298	0.106 67708	0.114 62984	0.115 16944	0.104 69002	0.108 44353	0.117 32008	0.122 37647	0.105 37552



For boston75,

boston75	m=1	2	3	4	5	6	7	8	9	10	11	12	13
folder=0	0.245 61404	0.175 4386	0.214 91228	0.258 77193	0.214 91228	0.258 77193	0.258 77193	0.212 7193	0.212 7193	0.254 38596	0.243 42105	0.263 15789	0.171 05263
	0.18	0.14	0.14	0.18	0.16	0.18	0.18	0.16	0.14	0.18	0.18	0.18	0.16
1	0.153 50877	0.175 4386	0.252 19298	0.142 54386	0.168 85965	0.164 47368	0.146 92982	0.135 96491	0.186 40351	0.201 75439	0.177 63158	0.197 36842	0.192 98246
	0.18	0.22	0.26	0.16	0.18	0.18	0.2	0.18	0.22	0.2	0.22	0.2	0.2
2	0.247 80702	0.254 38596	0.245 61404	0.236 84211	0.243 42105	0.247 80702	0.217 10526	0.228 07018	0.276 31579	0.241 22807	0.274 12281	0.252 19298	0.219 29825
	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
3	0.223 68421	0.223 68421	0.223 68421	0.223 68421	0.221 49123	0.223 68421	0.223 68421	0.214 91228	0.221 49123	0.221 49123	0.219 29825	0.223 68421	0.223 68421
	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0.6
4	0.225 87719	0.179 82456	0.192 98246	0.175 4386	0.195 17544	0.127 19298	0.155 70175	0.190 78947	0.219 29825	0.188 59649	0.127 19298	0.192 98246	0.153 50877

	0.44	0.38	0.38	0.38	0.42	0.38	0.38	0.42	0.42	0.38	0.38	0.42	0.38
5	0.203 94737	0.217 10526	0.203 94737	0.203 94737	0.210 52632	0.217 10526	0.212 7193	0.179 82456	0.217 10526	0.214 91228	0.210 52632	0.217 10526	0.186 40351
	0.62	0.64	0.6	0.58	0.62	0.64	0.64	0.62	0.64	0.64	0.64	0.64	0.62
6	0.206 14035	0.245 61404	0.258 77193	0.219 29825	0.195 17544	0.219 29825	0.201 75439	0.214 91228	0.223 68421	0.184 21053	0.153 50877	0.199 5614	0.179 82456
	0.14	0.2	0.2	0.14	0.12	0.18	0.18	0.18	0.2	0.16	0.08	0.14	0.18
7	0.217 10526	0.245 61404	0.263 15789	0.203 94737	0.208 33333	0.254 38596	0.221 49123	0.166 66667	0.160 08772	0.265 35088	0.214 91228	0.239 03509	0.258 77193
	0.14	0.16	0.16	0.14	0.14	0.16	0.14	0.14	0.14	0.16	0.14	0.14	0.14
8	0.155 70175	0.236 84211	0.179 82456	0.241 22807	0.232 45614	0.260 96491	0.274 12281	0.258 77193	0.228 07018	0.245 61404	0.228 07018	0.212 7193	0.252 19298
	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
9	0.203 94737	0.190 78947	0.210 52632	0.201 75439	0.274 12281	0.239 03509	0.265 35088	0.190 78947	0.199 5614	0.280 70175	0.263 15789	0.212 7193	0.269 73684
	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
train set average	0.208 33333	0.214 47368	0.224 5614	0.210 74561	0.216 44737	0.221 27193	0.217 76316	0.199 34211	0.214 47368	0.229 82456	0.211 18421	0.221 05263	0.210 74561
test set average	0.242	0.246	0.246	0.23	0.236	0.244	0.244	0.242	0.248	0.244	0.236	0.244	0.24
train set std	0.030 63903	0.029 88899	0.027 53394	0.031 98223	0.027 58367	0.041 47521	0.040 4015	0.032 60767	0.028 53912	0.031 13417	0.044 2423	0.022 44569	0.038 02723
test set std:	0.214 74636	0.211 4805	0.204 1666	0.203 42075	0.214 62525	0.211 24393	0.211 62231	0.212 12261	0.215 25798	0.211 4332	0.216 48095	0.216 11108	0.208 61448



Discussion:
Test set has a higher average error rate then train set which is normal, since test set are unseen date set.
When the size of feature set is very small, there are very limited choice we can make to decide best feature and split, then a related high error rate. When the feature set is very larger, the variance of trees can be appropriately reduced, since trees generated are more correlated. Sometimes, we see error rate fluctuates a bit, because all trees make equal weighted votes and including more feature candidates have to balance variance between trees and performance of each tree, which is hard to capture.