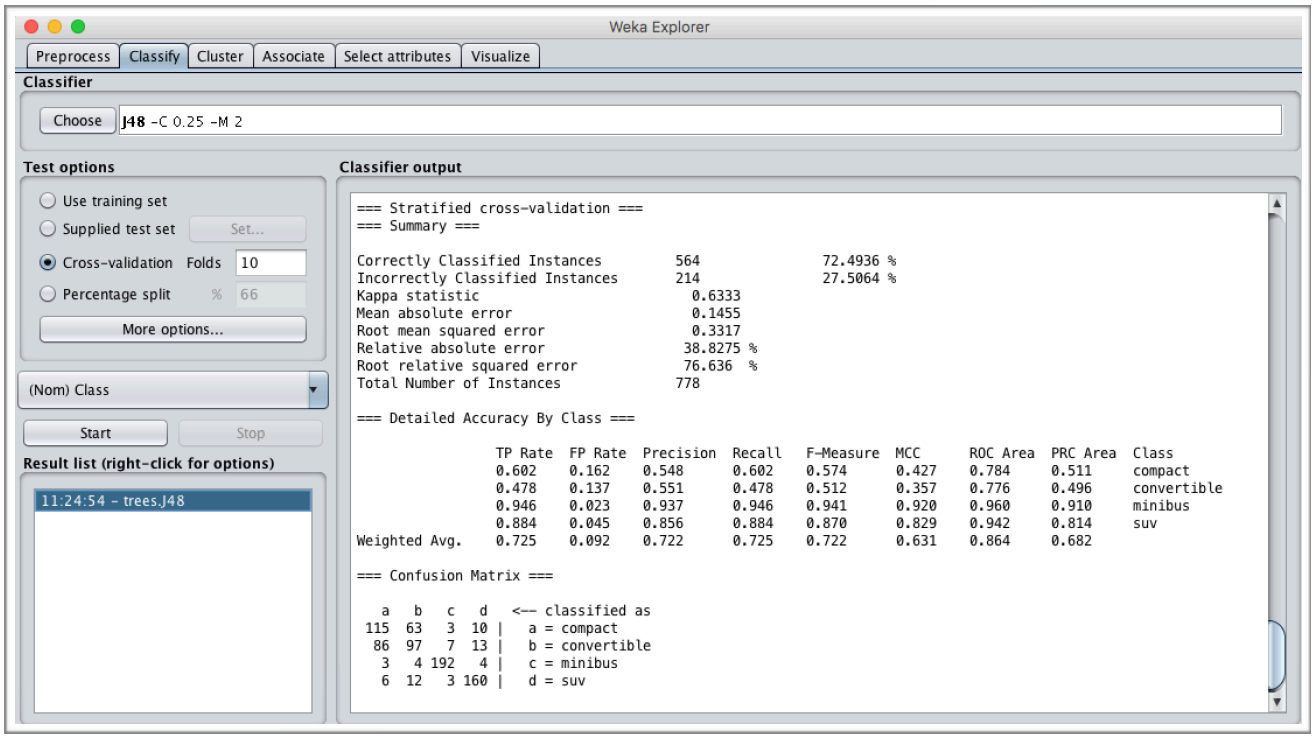
Name: Gurpreet Singh

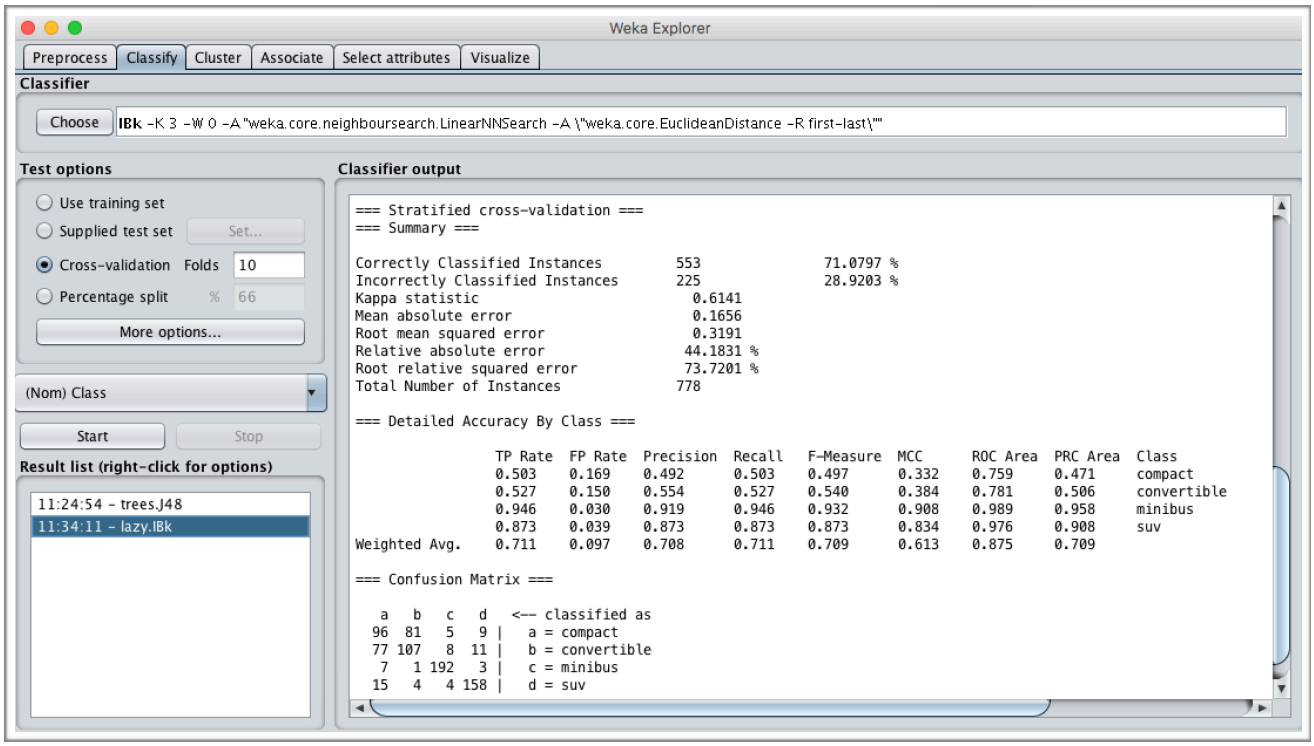
Data Set used:
http://claritytrec.ucd.ie/~alawlor/comp47460/datasets/ml/vehicles/17202278.arff

Performance of Decision Trees (J48) and 3-NN classifiers on the data.

On evaluation with the Decision Trees(J48), I received the correctly classified instances as 72.49%, with cross-validation as 10.



On evaluation with the K-NN (IBK) with 3 nearest neighbours, I received the correctly classified instances as 71.07%, with cross-validation as 10.



Name: Gurpreet Singh

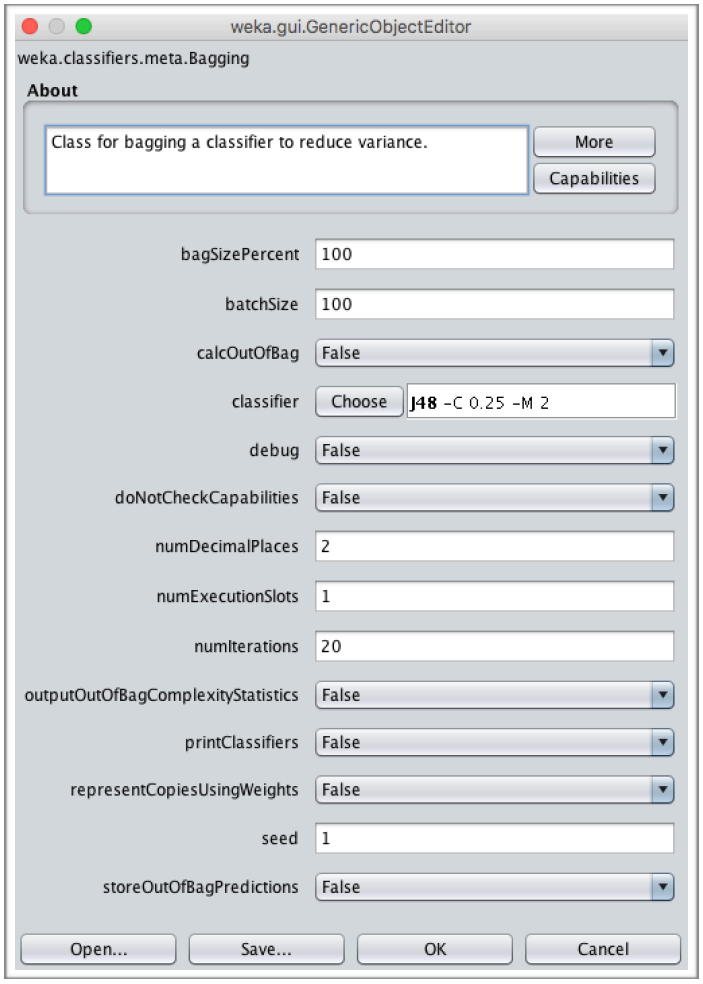
Applying ensembles with bagging using Decision Trees (J48) and 3-NN classifiers. Investigating the performance of both classifiers as the ensemble size increases, in steps of 20 from 20 to 100 members.

Bagging with:

- **J48**
- **3-NN**

Steps:

1. Using Weka, click on the Classify tab.
2. Click Choose, select method classifiers->meta->Bagging.
3. Click Bagging in the box to the right. The configuration interface of the method appears.
4. Click Choose, select <J48/IBK(3-NN)>. (one at a time)
5. Set the “numIterations” to <20,40,60,80,100 (one at a time)> Click OK button.
6. Click Start button to build the ensemble.

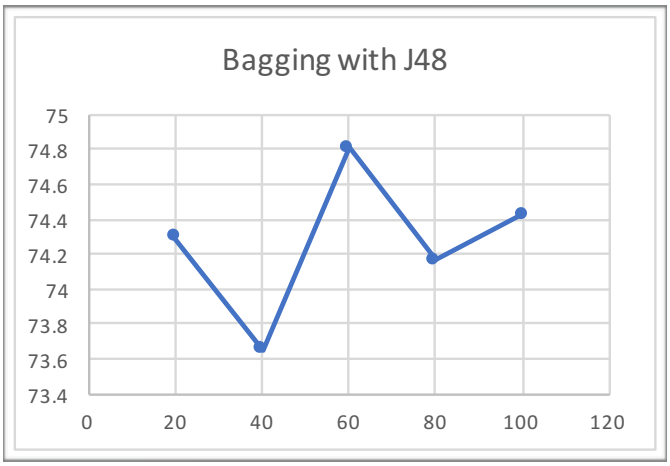


Bagging J48:

Bagging with J48

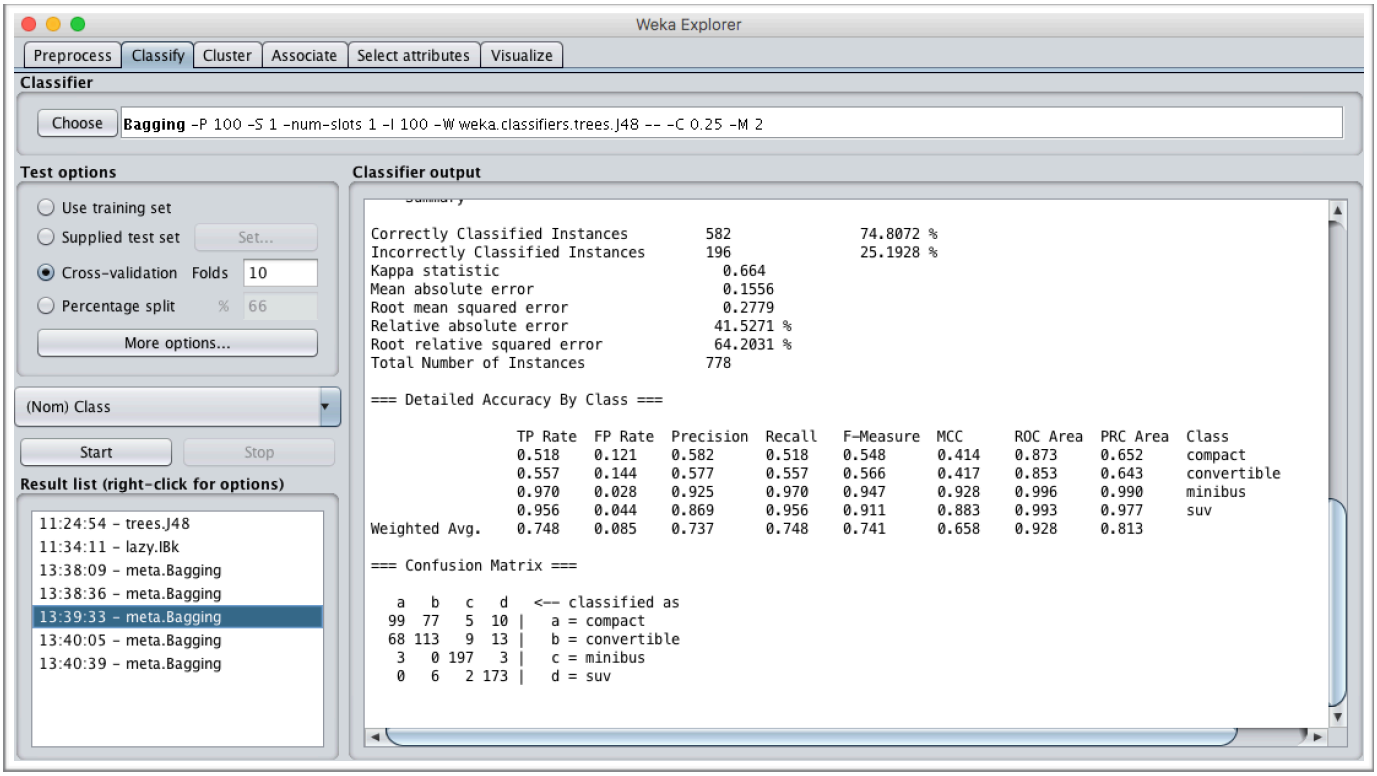
Ensemble Size	% Correct	% Incorrect
20	74.2931	25.7069
40	73.6504	26.3496
60	**74.8072**	**25.1928**
80	74.1645	25.8355
100	74.4216	25.5784

Visualisation of the performance



The best performance I received with the ensemble size of 60, with cross-validation as 10.

Name: Gurpreet Singh

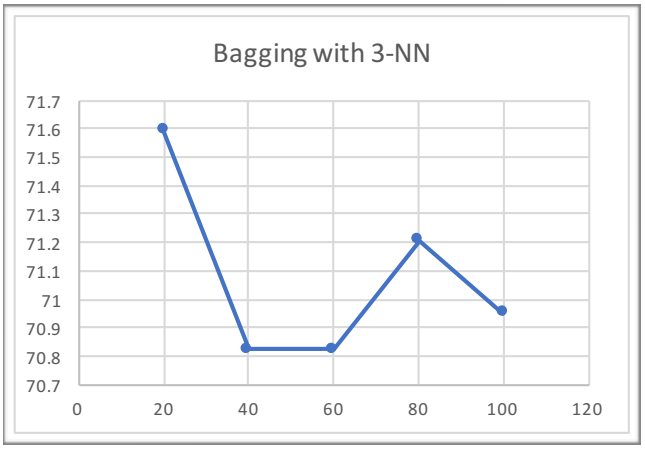


Bagging 3-NN:

Bagging with 3-NN

Ensemble Size	% Correct	% Incorrect
20	**71.5938**	**28.4062**
40	70.8226	29.1774
60	70.8226	29.1774
80	71.2082	28.7918
100	70.9512	29.0488

Visualisation of the performance



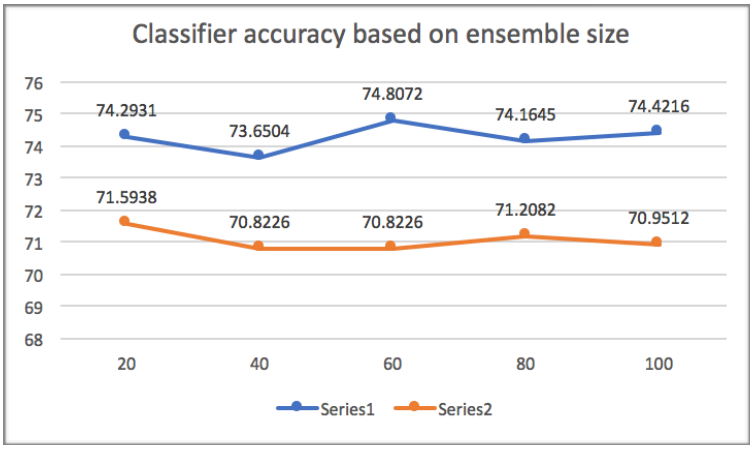
- *For both of the classifiers **the best <u>ensemble size</u> are, J48 with '60' and for 3-NN with '20'**.*
- *Therefore calculated the accuracy on **bag-size** from <u>10 to 100, with the step size of 10.</u>*
- *Right table shows the classifier's <u>accuracy on **varying** the bag size and **fixed** ensemble size as 60 for J48 and 20 for 3-NN</u>.*

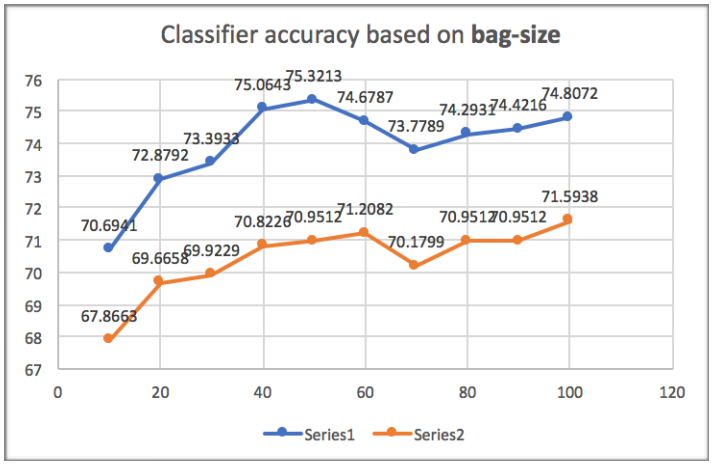
<u>Classifier (Ensemble Size)</u>

Bag Size	J48 (60)	3-NN (20)
10	70.6941	67.8663
20	72.8792	69.6658
30	73.3933	69.9229
40	75.0643	70.8226
50	**75.3213**	70.9512
60	74.6787	71.2082
70	73.7789	70.1799
80	74.2931	70.9512
90	74.4216	70.9512
100	74.8072	**71.5938**

Name: Gurpreet Singh

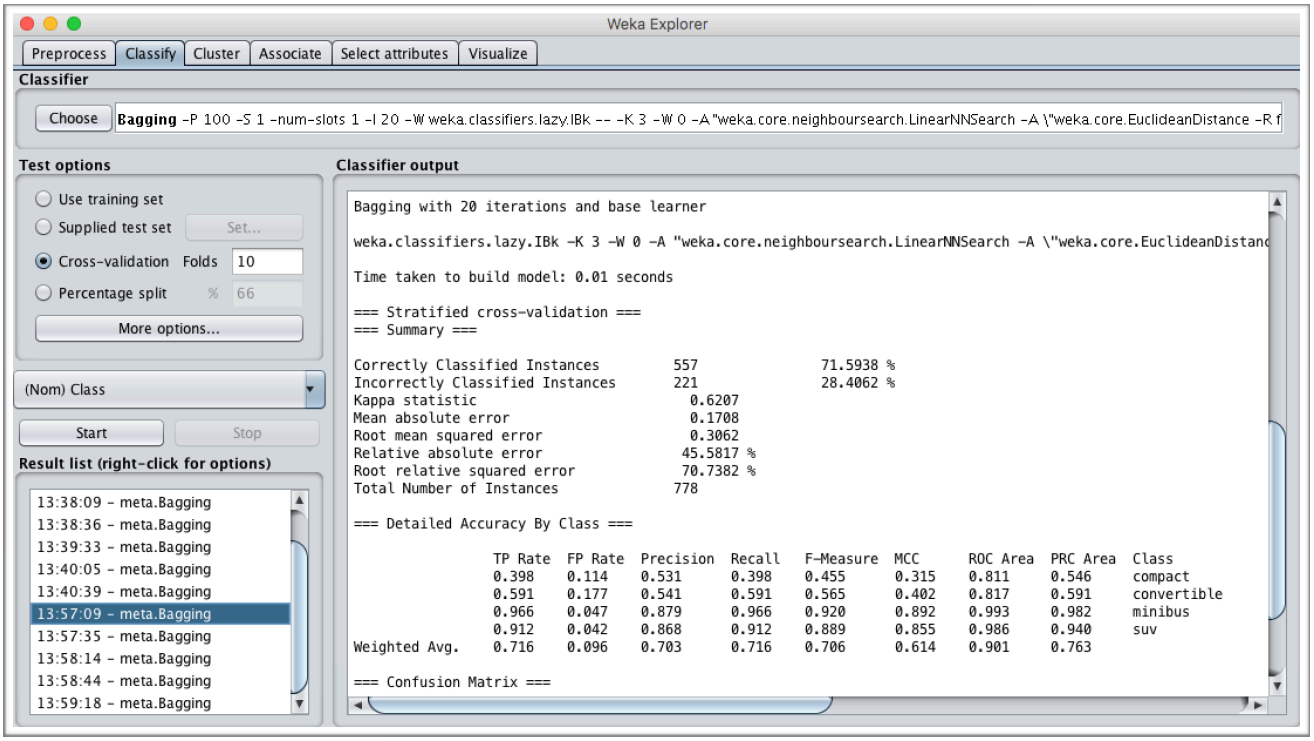
Below is the plot showing the performances on ensemble size and bag-size for best ensemble size.





Series 1: J48 & Series 2: 3-NN

The best performance I received with the ensemble size of 20, with cross-validation as 10.

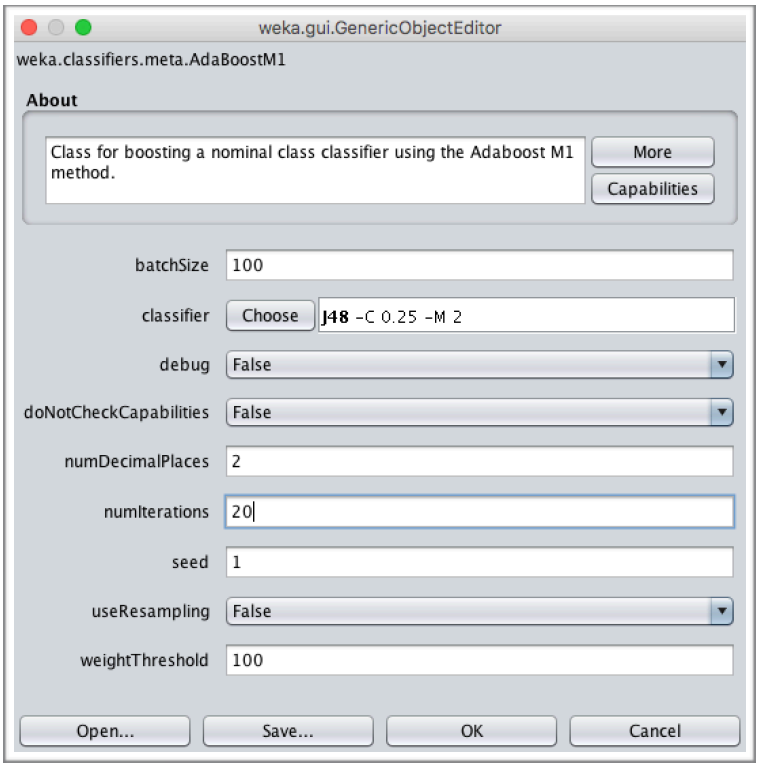


Name: Gurpreet Singh

Applying ensembles with boosting using both classifiers and Investigating the performance of both classifiers as the ensemble size increases, in steps of 20 from 20 to 100 members.

Steps:

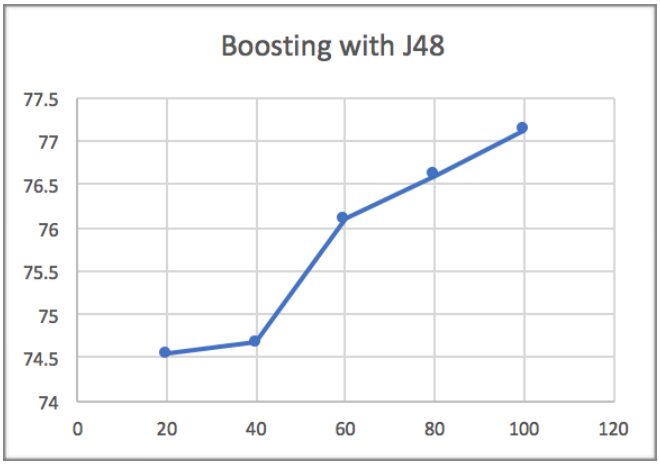
1. Using Weka, click on the Classify tab.
2. Click Choose, select method classifiers->meta->AdaBoostM1.
3. Click AdaBoostM1 in the box to the right. The configuration interface of the method appears.
4. Click Choose, select <J48/IBK(3-NN)>. (one at a time)
5. Set the numIterations to <20,40,60,80,100 (one at a time)>.
6. Click OK button.
7. Click Start button to build the ensemble.



Boosting with J48:

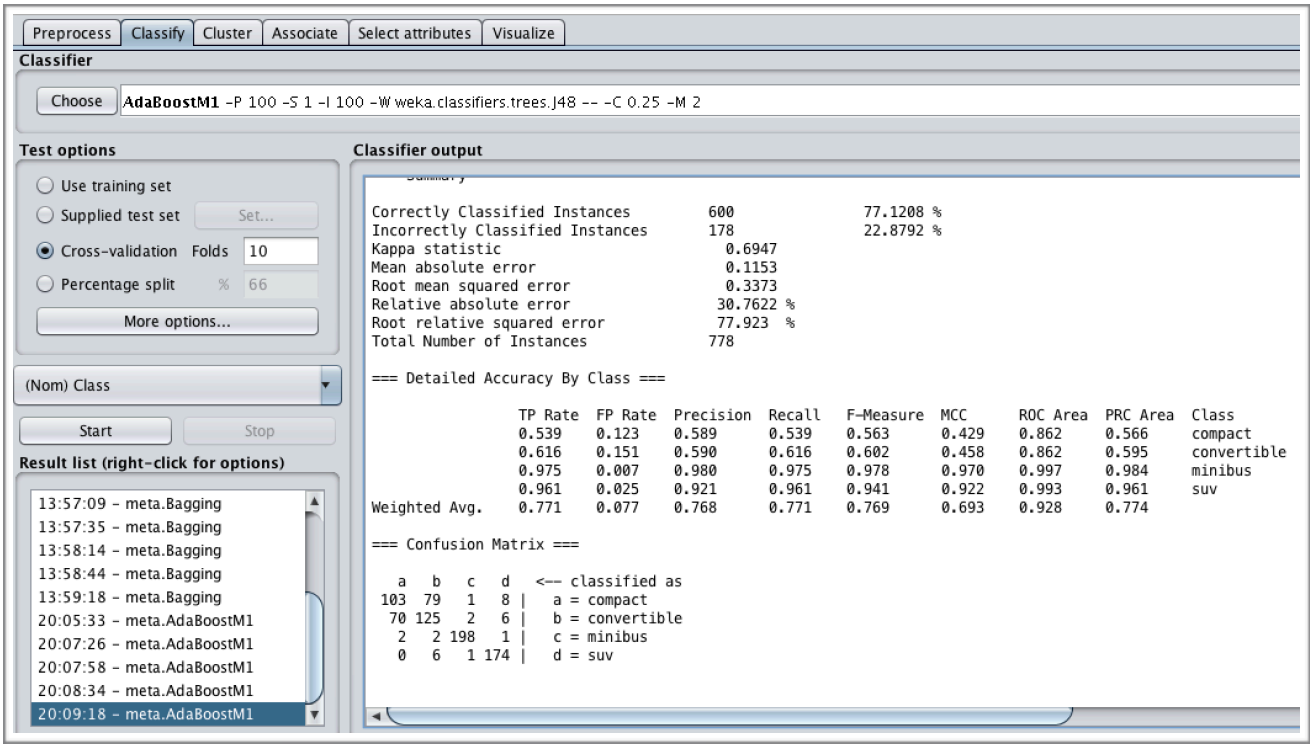
Ensemble Size	% Correct	% Incorrect
20	74.5501	25.4499
40	74.6787	25.3213
60	76.0925	23.9075
80	76.6067	23.3933
100	**77.1208**	**22.8792**

Visualisation of the performance



Name: Gurpreet Singh

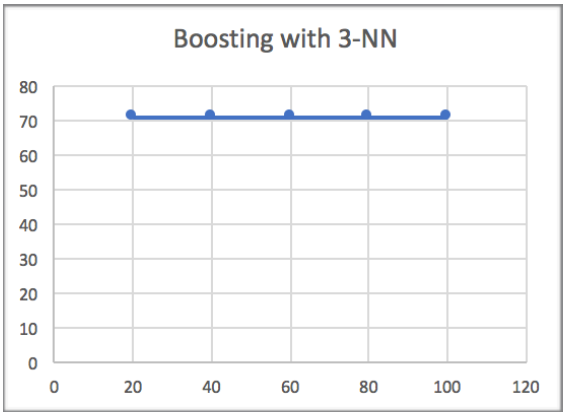
The best performance I received with the ensemble size of 100, with cross-validation as 10.



Boosting with 3-NN:

Ensemble Size	% Correct	% Incorrect
20	70.6941	29.3059
40	70.6941	29.3059
60	70.6941	29.3059
80	70.6941	29.3059
100	70.6941	29.3059

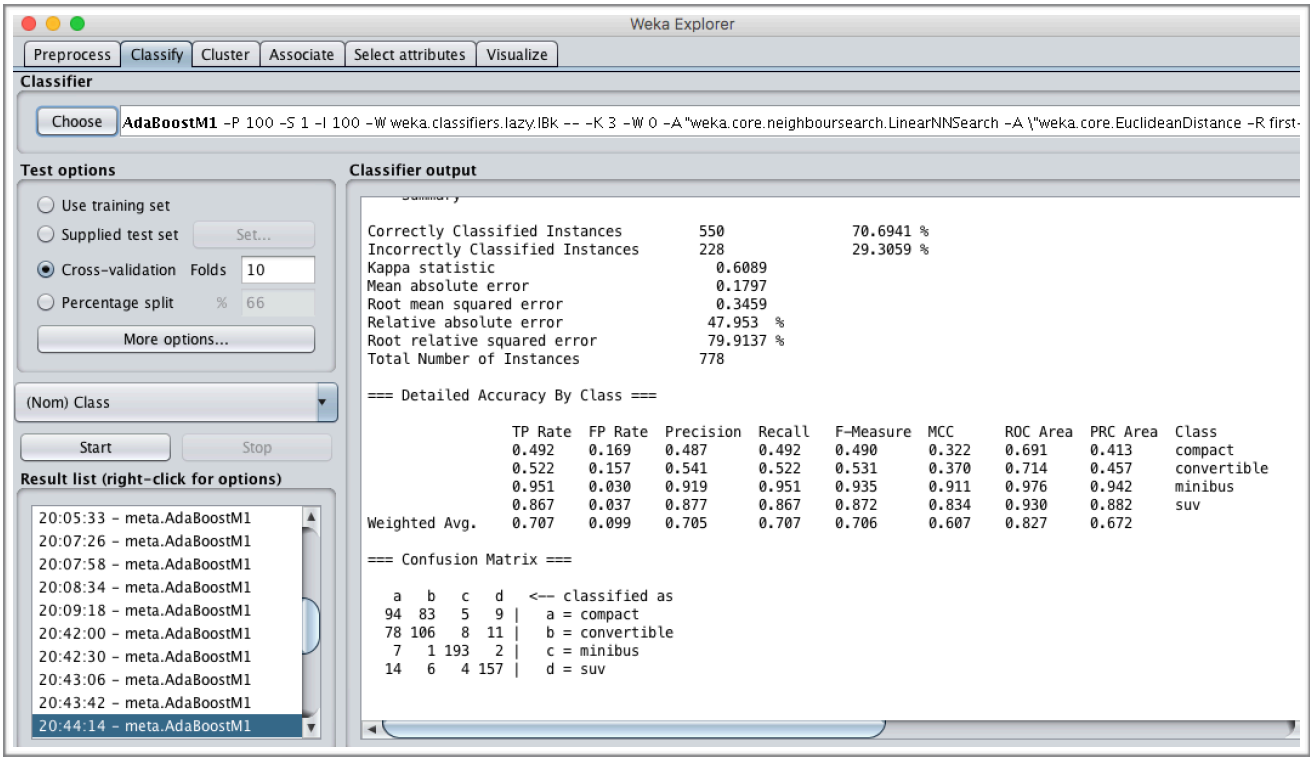
Visualisation of the performance



*Here for every 'numiterations' I received the **same** correctly and incorrectly classified instances, there by for this on researching I got to know that the direct application of boosting to k-NN (in this case 3-NN) classifiers is not available, when using re-weighting, or resampling. k-NN is in-stable with respect to inputs.[12] There-fore the performance didn't increased / decreased.*

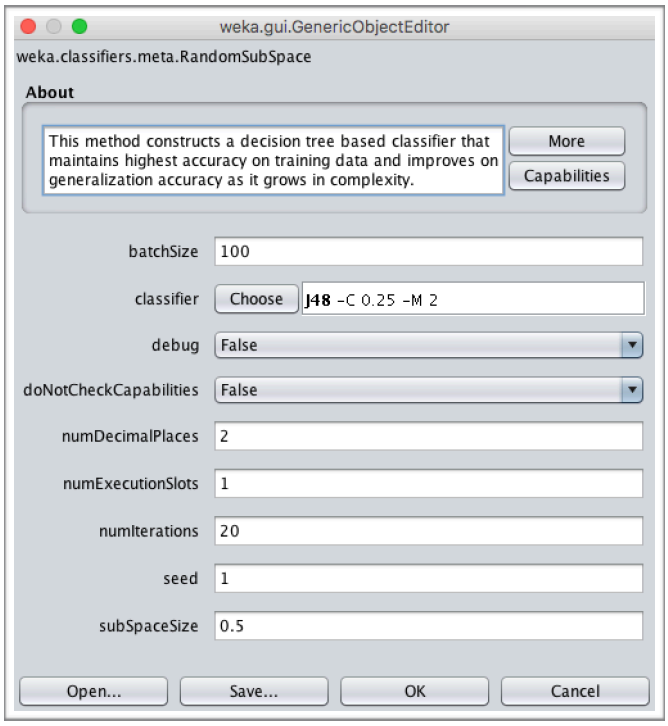
Name: Gurpreet Singh

The performance for 20/40/60/80/100 as *all were same, with cross-validation as 10*:



Applying ensembles with random sub-spacing using both classifiers, and Investigating the performance of both classifiers as the ensemble size increases, in steps of 20 from 20 to 100 members.
Using the best performing ensemble size, investigating, how changing the number of features used when applying random sub-spacing affects classification performance (i.e. the "subspace size").

1. Using Weka, click on the Classify tab.
2. Click Choose, select method classifiers->meta->RandomSubSpace.
3. Click RandomSubSpace in the box to the right. The configuration interface of the method appears.
4. Click Choose, select <J48/IBK(3-NN)> (one at a time).
5. Click J48/IBK(3-NN). In the configuration window, if KNN then set to 3. Close window.
6. Set the numIterations to <20,40,60,80,100 (one at a time)>.
7. Click OK button.

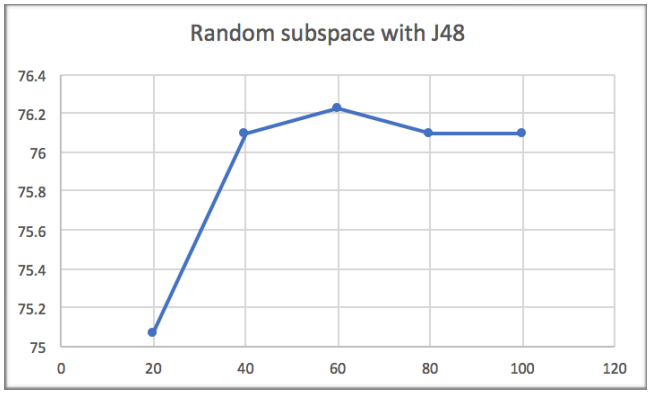


Visualisation of the performance

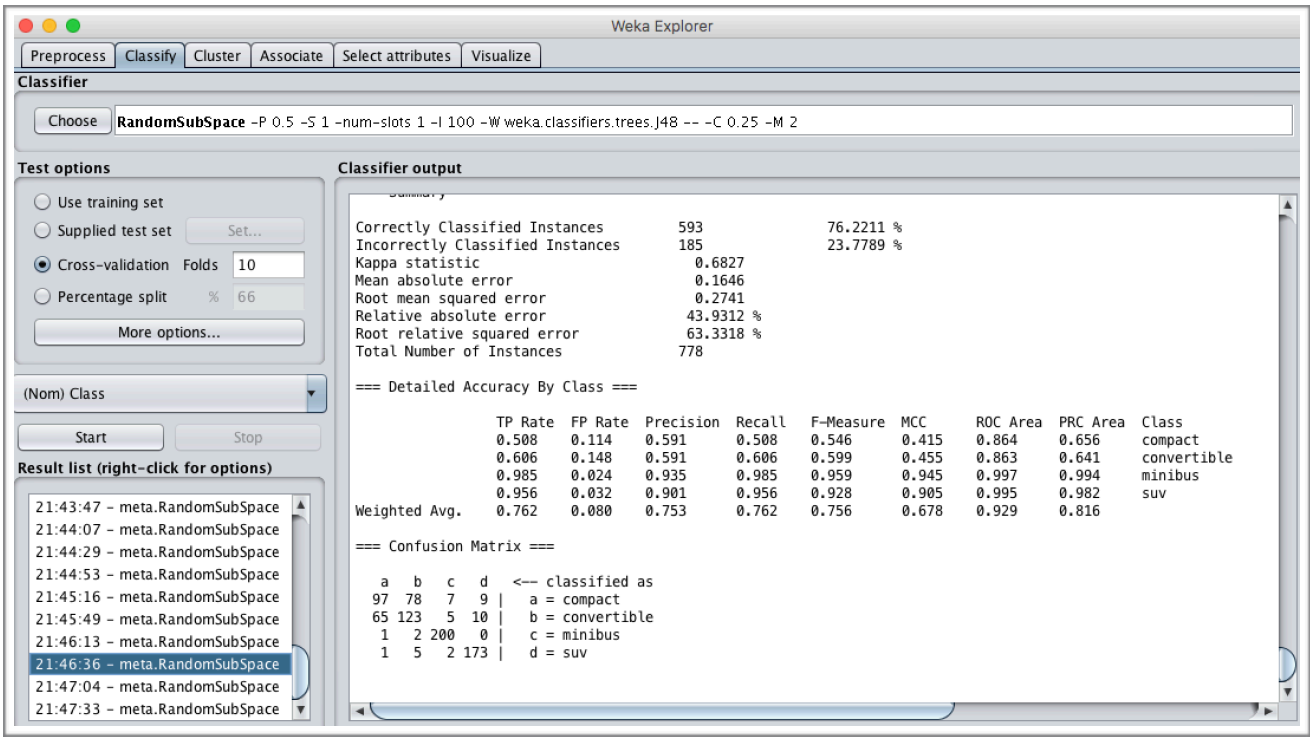
Name: Gurpreet Singh

Random subspace with J48:

Ensemble Size	% Correct	% Incorrect
20	75.0643	24.9357
40	76.0925	23.9075
60	**76.2211**	**23.7789**
80	76.0925	23.9075
100	76.0925	23.9075



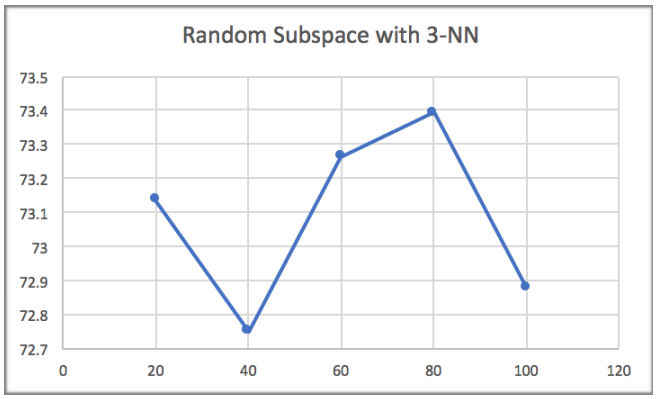
Best accuracy with numIterations as 60, with cross-validation as 10:



Random subspace with 3-NN:

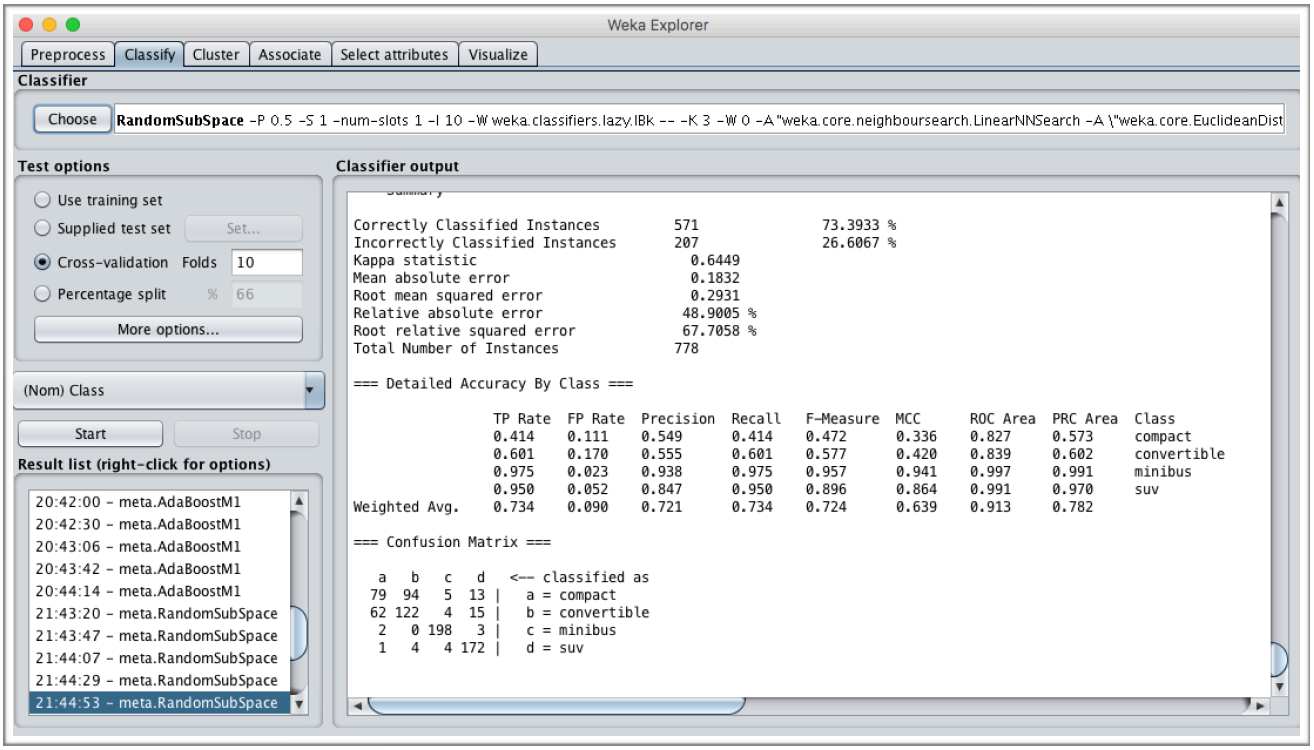
Visualisation of the performance

Ensemble Size	% Correct	% Incorrect
20	73.1362	26.8638
40	72.7506	27.2494
60	73.2648	26.7352
80	**73.3933**	**26.6067**
100	72.8792	27.1208



Name: Gurpreet Singh

Best accuracy with numIterations as 80, with cross-validation as 10:



Best Performance:

We can clearly see from the above set that "**Random Subspace with J48 Decision Tree** with **Ensemble Size** as **60**" performed best, with correct percentage as 76.2211, with cross-validation as 10.

In the tables shown below it can be seen *different subspace sizes* (0.25,0.5,0.75,0.95) to see how the value of RandomSubSpace *affect the accuracy of the models,* with the **J48 Decision Tree and 3-NN** with *Ensemble Size 20/40/60/80/100, previously*.

Random Subspace with J48 Decision Tree:

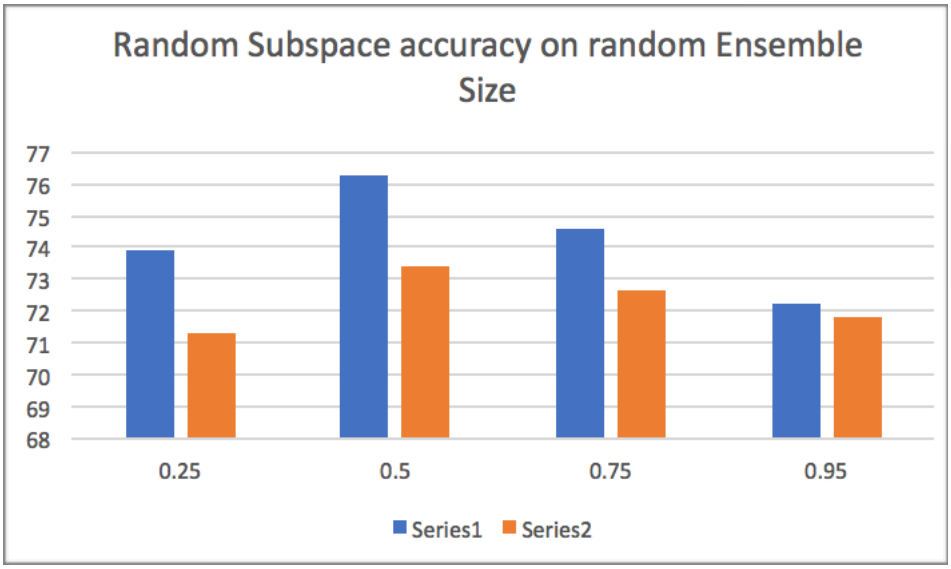
Subspace Ensemble size	0.25	0.5	0.75	0.95
20	**75.1928**	75.0643	74.8072	72.108
40	72.7506	76.0925	**74.9357**	**72.2365**
60	73.9075	**76.2211**	74.5501	**72.2365**
80	73.9075	76.0925	73.9075	71.9794
100	74.2931	76.0925	73.7789	72.108

Name: Gurpreet Singh

Random Subspace with 3-NN:

Subspace Ensemble size	0.25	0.5	0.75	0.95
20	71.0797	73.1362	71.8509	72.108
40	70.5656	72.7506	72.2365	**72.365**
60	71.0797	73.2648	**73.0077**	71.9794
80	**71.3368**	**73.3933**	72.6221	71.8509
100	71.2082	72.8792	72.6221	72.2365

Here from the above table we can see clearly that the *best performance of J48* was with the ensemble size 60, where as on the other hand the *best accuracy with 3-NN* has with ensemble size of 80. Below is the visualisation of both of the best accuracies based on the variable



Series 1: J48 & Series 2: 3-NN

Conclusion:

- In both the above cases it can be seen that a subspace value of 0.5 performs better than 0.25.
- This is because working with 25% of the features leaves too much of important information in order to produce an accurate model.
- Hence, it can be seen that even tough with the variable ensemble size, the *subspace=0.5* has always performed better for the J48 algorithm for my dataset.
- It can be further seen that the J48 has performed better than 3-NN in almost all the ensemble size leaving 0.95.

Summary:

Name: Gurpreet Singh

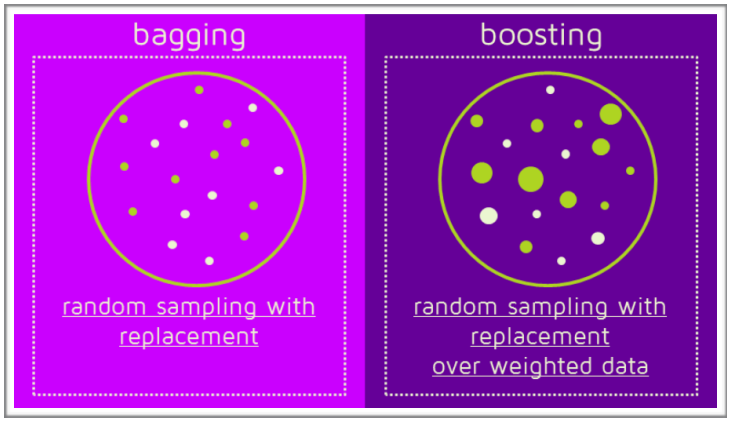
In the below table I have summarised all the above table, and we can see the best accuracy of both the J48 and 3-NN's with Bagging, Boosting and Random Subspace on its respective ensemble size (and subspace for Random subspace).

Method's Accuracy	J48	3-NN
Bagging	74.8072	71.5938
Ensemble Size	60	20
Boosting	**77.1208**	70.6941
Ensemble Size	**100**	20/40/60/80/100
Random SubSpace	76.2211	73.3933
Ensemble Size & Subspace	60 0.5	80 0.5

These differences are because:

Bagging:
Bagging is an ensemble generation method that uses variations of samples used to train base classifiers. For each classifier to be generated, Bagging selects (with replacement (No weights)) N samples from the training set with size N and train a base classifier. This is repeated until the desired size of the ensemble is reached.
Bagging should be used with unstable classifiers, that is, classifiers that are sensitive to variations in the training set such as Decision Trees.
That is why J48 is having (74.8072%) better score than 3-NN with (71.5938%).

Boosting:
Boosting generates an ensemble by adding classifiers that correctly classify "difficult samples". For each iteration, *boosting weights the samples*, so that, *misclassified samples* by the ensemble can have a *higher weight.* Hence, in next run those have *higher probability of being selected for training the new classifier.*
Boosting approach is very sensitive to noise and is only effective using weak classifiers.
Hence we can see that with J48 this outperformed than the remaining with the accuracy of 77.1208%.



Name: Gurpreet Singh

Random Subspace:
Random Subspace is an approach that uses *variations in the features* instead of *variations in the* samples.

That is why we can see that the performance of random subspace is better than bagging in both the cases J48 and 3-NN, whereas for *3-NN random subspace gave the best accuracy.*

To sum up the best performance was with J48 with Boosting and having the ensemble size as 100.

Name: Gurpreet Singh

References:

[1]: http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/
[2]: http://www.edupristine.com/blog/curse-dimensionality
[3]: Lecture Notes on Feature selection.
[4]: http://nikhilbuduma.com/2015/03/10/the-curse-of-dimensionality/
[5]: https://stats.stackexchange.com/questions/120080/whatre-the-differences-between-pca-and-autoencoder
[6]: diagnosing-bias-vs-variance by Andrew Ng
[7]: Bias-Variance Tradeoff and Ensemble Methods by Tom Dietterich & Rich Maclin
[8]: http://www-scf.usc.edu/~csci567/17-18-bias-variance.pdf
[9]: https://stats.stackexchange.com/questions/22329/how-does-centering-the-data-get-rid-of-the-intercept-in-regression-and-pca
[10]: http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/
[11]: http://www.juanshishido.com/logisticcoefficients.html
[12]: Boosting k-nearest neighbor classifier by means of input space projection by Nicolás & Domingo.
[13]:Bagging, Boosting and the Random Subspace Method for Linear Classifiers by Marina Skurichina and Robert P. W. Duin.
[14]: https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/13