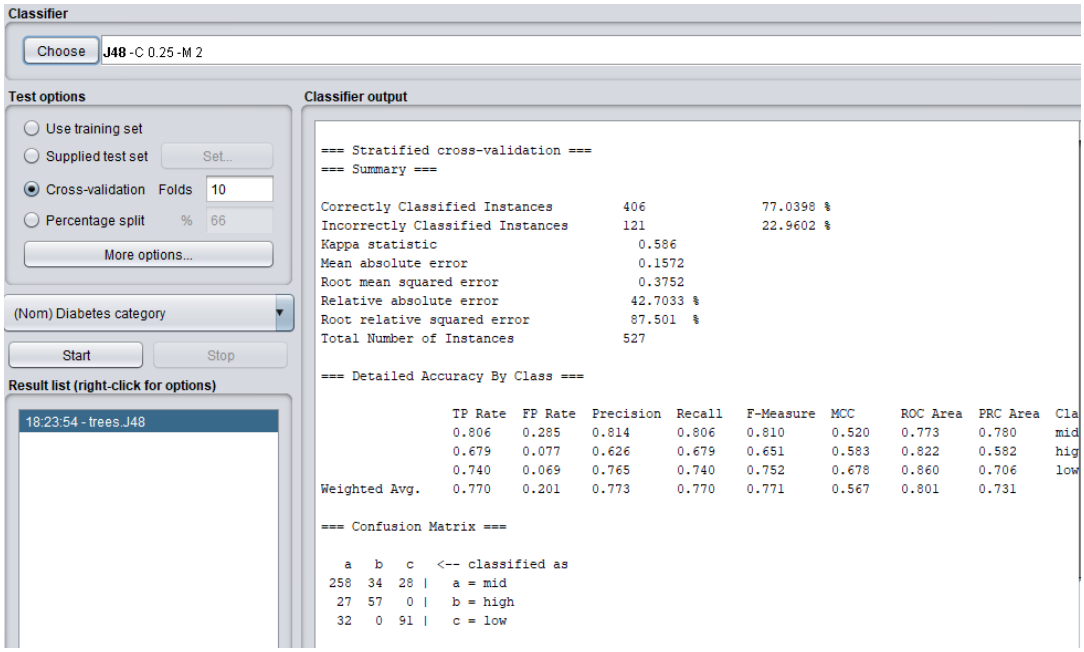
Question 1

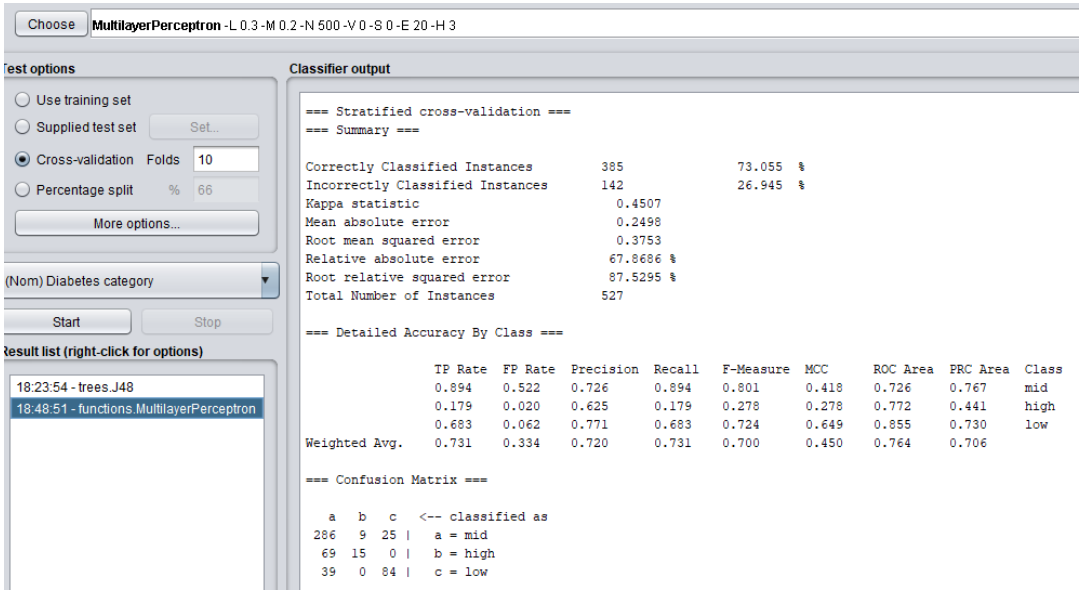
Classification algorithm	weka classifier	Correct classified	Incorrectly classified
Decision Tree	j48	77.03%	22.96%
Neural Network	MLP (hidden layer = 3)	73.05%	26.94%
Neural Network	MLP (hidden layer = 5)	73.62%	26.37%
k-NN	k=3	80.83%	19.16%

The best classification algorithm was **k-Nearest Neighbours** with k= 3 having 80.83% accuracy as highlighted above.

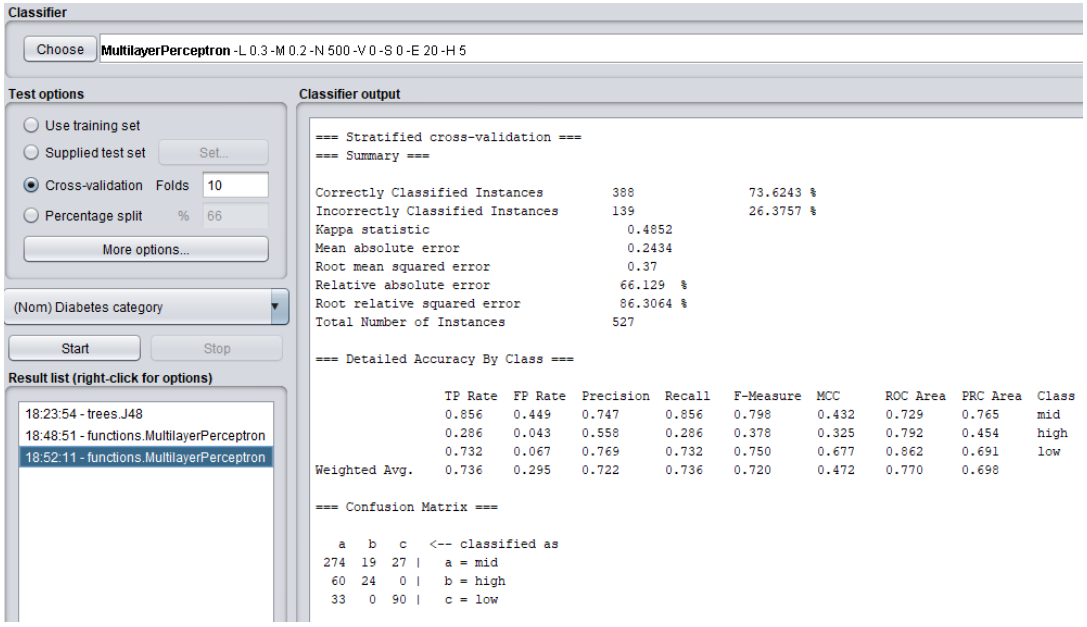
Decision tree



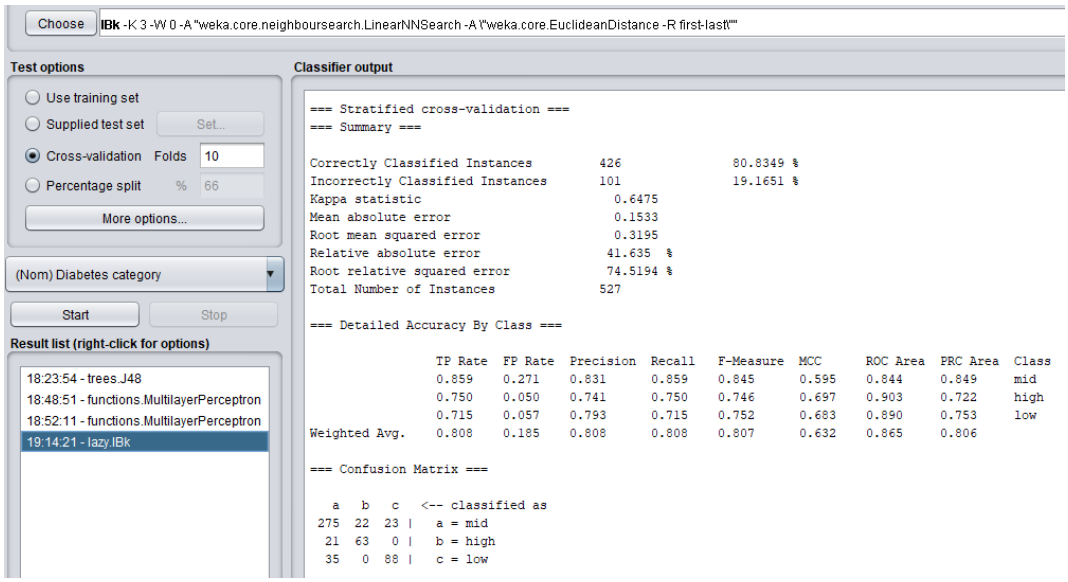
NN with 3 hidden layers



NN with 5 hidden layers



k-NN with k=3



Question 2

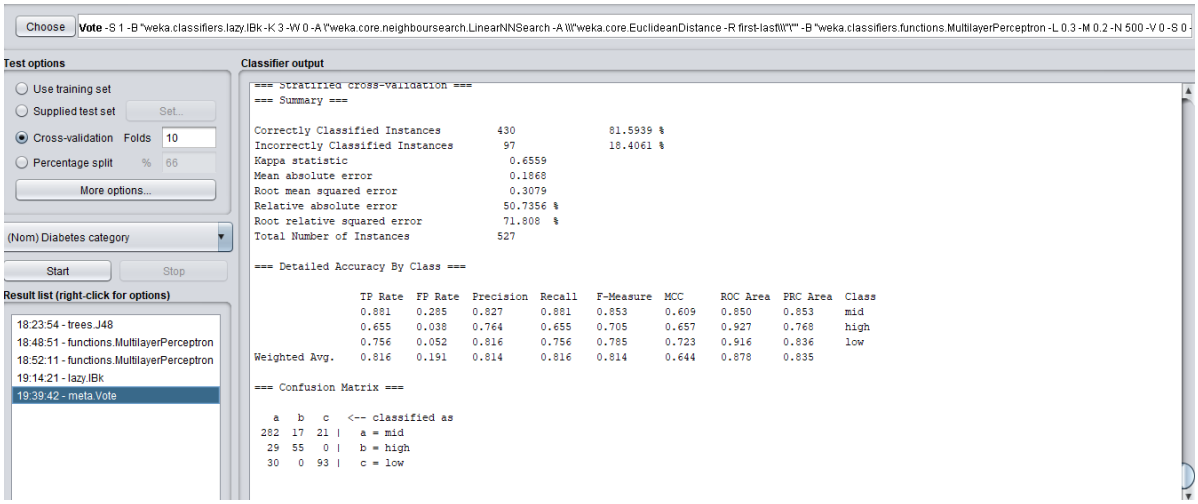
Classification algorithm	Correct classified	Incorrectly classified
Average Probabilities	81.59%	18.40%
Majority Voting	81.02%	18.97%
Minimum Probability	80.26%	19.73%
Maximum Probability	80.64%	19.35%

I used below 3 classifiers for the above 4 different combination rules

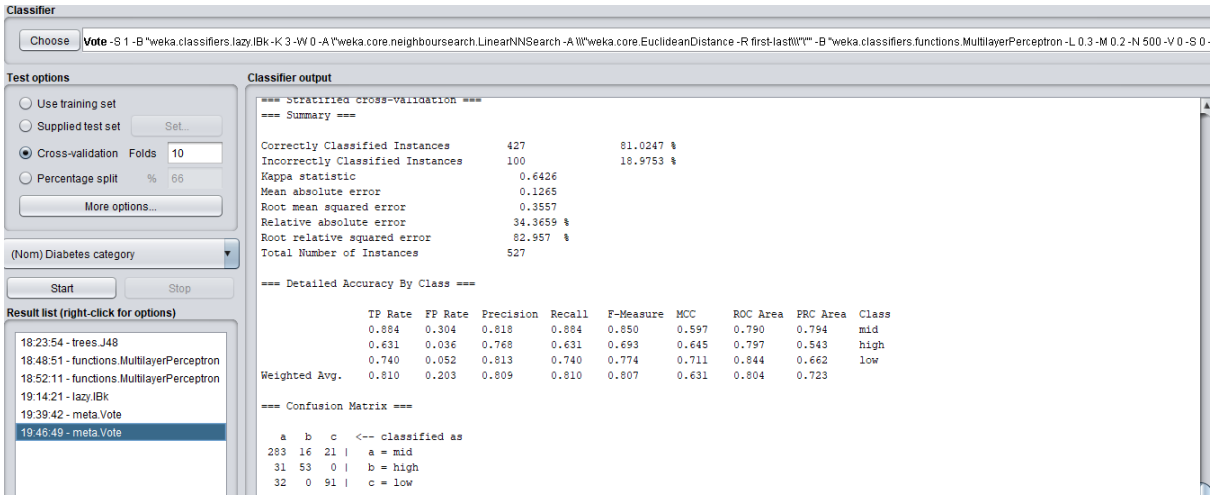
Classification algorithm	weka classifier
Decision Tree	j48
Neural Network	MLP (hidden layer = 3)
k-NN	k=3

The small differences in accuracy when using above 4 combination rules is because
- of higher diversity of the dataset
- less disagreement between the decisions within the voters leading to small differences in accuracies in all 4 combinations
- possible weighting and bias impact on the output of the neuron impacting activation functions
- depending on the problem, complex decisions may require long chains of computational stages based on voting pattern

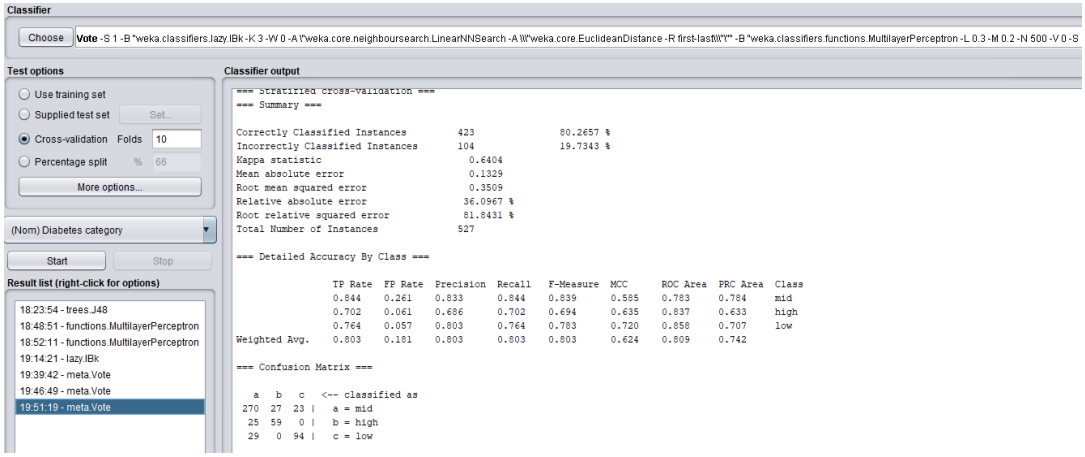
Average Probabilities



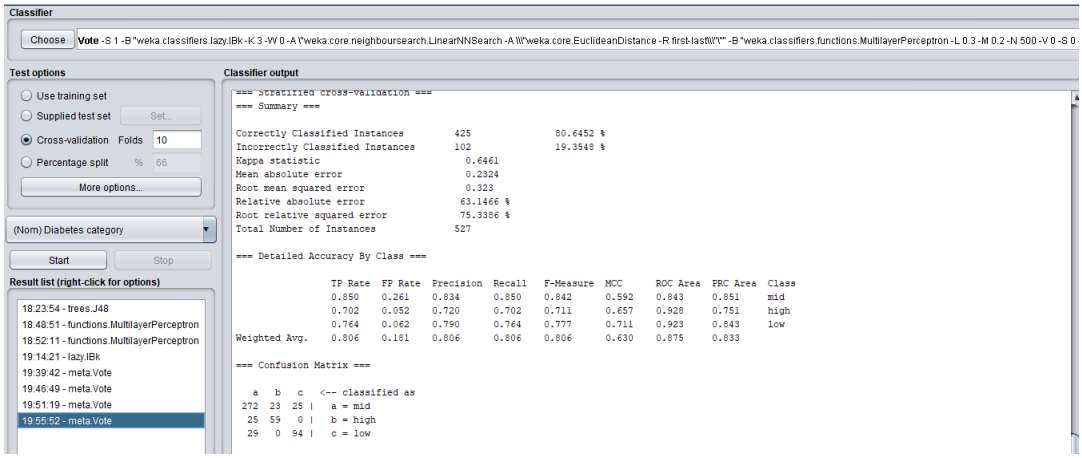
Majority Voting



Minimum Probability

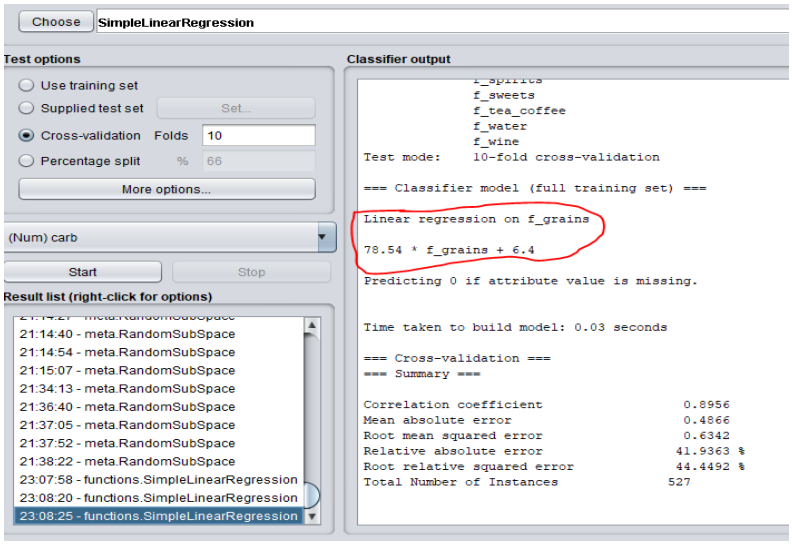


Maximum Probability



Question 3

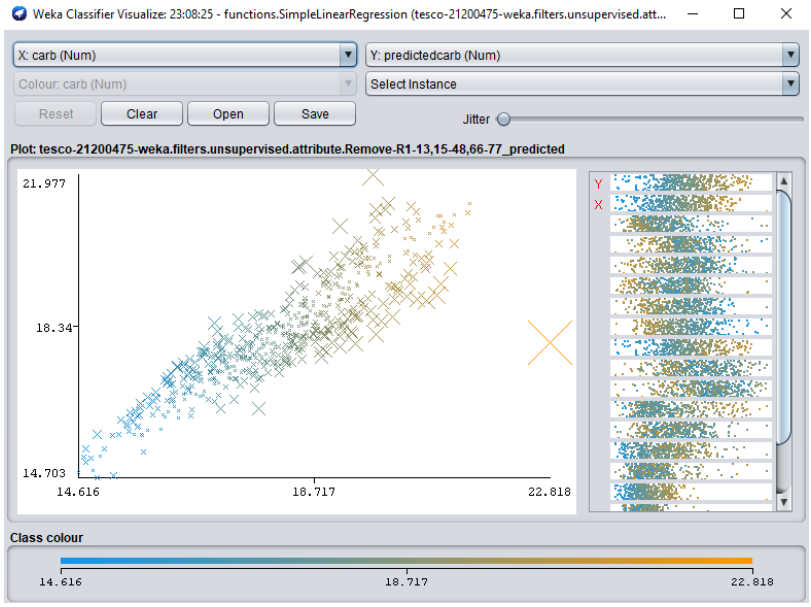
Weka choses feature “f_grains” as it has the highest weight assigned to it to minimize the residual errors.
Equation: predicted_carb = 78.54 * f_grains + 6.4



The model quality is good when we consider the correlation coefficient 89.56% with the predicted values of the feature 'carb'. However, there is 63% root mean squared error which is very high indicating that a combination of 'f_grains' feature alone would not produce the best Linear Regression model.

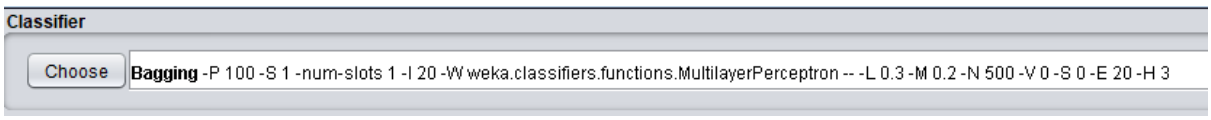
A possible solution to improve the model would be to choose few more features out of 17 attributes and assigning weights for each selected attribute. It would minimize the squared error on training data and improve prediction accuracy.

Analysing predicted vs actual values for the above Linear regression model shows that errors are high.



Question 4

I used the below "Bagging" method for all 3 classifiers (decision tree, Neural network and k-NN)



The below highlighted green cells have best accuracies in each of the iterations.

No of iterations	Accuracy in %ages		
	j48	NN (hidden layer = 3)	k-NN (k=3)
2	76.4706	78.7476	79.1271
4	79.6964	78.9374	80.6452
6	80.6452	79.1271	81.4042
8	79.8861	78.9374	81.5939
10	79.8861	79.1271	81.9734
12	80.2657	79.3169	81.7837
14	81.0247	79.5066	81.4042
16	81.0247	79.6964	81.0247
18	82.1632	79.6964	81.4042
20	81.9734	79.8861	81.5939

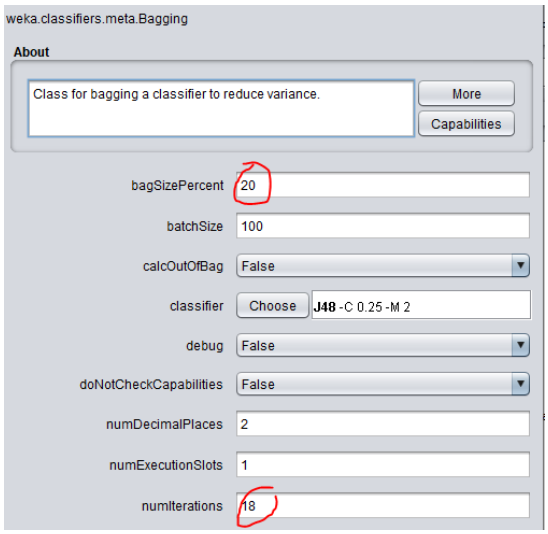
Among all 3 classifiers, bagging method significantly improved the accuracy from 76% to 82% for j48 decision tree.
We can see that level of improvement in accuracy often "level off" after an ensemble has been increased to a certain size. This is visible across all classifiers. For example: j48 decision tree classifier plateau's around interaction size = 14.

For MultiLevelPerceptron with “hidden layer = 3” - As you increase the number of iterations from 2 to 20 in MLP, time taken to build the model in Weka keeps increasing from 30 secs to 5 minutes. Additionally, time taken for validation also keeps increasing. In my case for MLP, accuracy improved marginally from 78 to 79% although number of iterations increased from 2 to 20.

For k-NN (k=3) - accuracy improved from 79 to 81% within first few iterations (2 to 6), after no new diversity is added, so ensemble accuracy will create a plateau.

Consistent improvement was seen in j48 decision tree from 76% to 82%, making it the best classifier with **iteration size = 18** having maximum accuracy 82% among the three classifiers.

I changed the **bagSizePercent** from 20%, 40% upto 100% in upward steps of 20% for iteration size = 18
The accuracy performance increases from 79% to 82% when we increase the **bagSizePercent** from 20% to 100% respectively.



j48 bagSizePercent	accuracy in %
20%	79.5066
40%	80.2657
60%	81.0247
80%	81.0247
100%	82.1632

Question 5

I used the “RandomSubSpace” method for all 3 classifiers (decision tree, Naïve Bayes and k-NN).
Note: Neural network was taking a long time to build model. Hence, I used Naïve Bayes for this question.

The below highlighted green cells have best accuracies in each of the iterations.

No of iterations	Accuracy in %ages		
	j48	Naïve Bayes	k-NN (k=3)
2	76.8501	72.1063	80.8349
4	79.3169	73.055	82.1632
6	81.7837	74.1935	83.112
8	83.112	74.0038	82.9222
10	82.1632	73.055	82.5427
12	83.3017	72.8653	82.9222
14	82.5427	72.6755	82.7324
16	83.112	72.8653	82.3529
18	83.112	72.8653	83.112
20	83.112	73.055	82.7324

Among all 3 classifiers, RandomnSubSpacing method significantly improved the accuracy from 76% to 83% for j48 decision tree

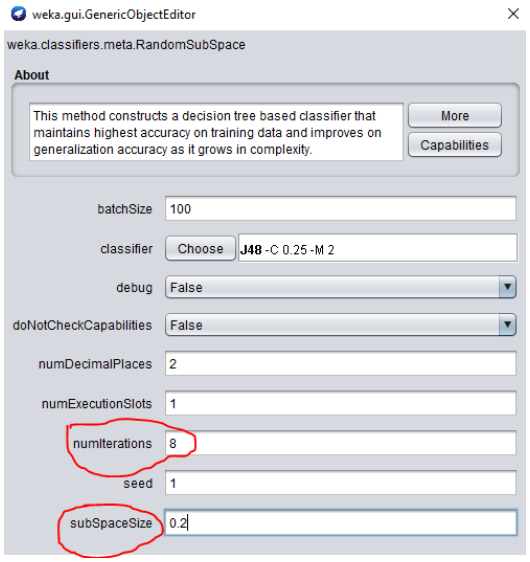
For Naïve Bayes classifier, as you increase the number of iterations from 2 to 8, accuracy improved marginally from 72 to 74%.
After that it was flat and lower sometimes.

For k-NN (k=3), accuracy improved from 80 to 83% within first few iterations (2 to 8), after no new diversity is added, so ensemble accuracy will create a plateau.

We can consider that **iteration size = 8** has maximum accuracy 83% among the 2 of the 3 classifiers.

I changed the **subSpaceSize** from 0.2 to 1.0 in upward steps of 0.2 for iteration size = 8
The accuracy performance decreases from 82% to 65% when we increase the **subSpaceSize** from 0.2 to 1.0 respectively.

j48 bagSizePercent	accuracy in %
0.2	82.5427
0.4	82.7324
0.6	82.5427
0.8	80.8349
1.0	65.4649



Question 6

Set of classifiers is expected to benefit from bagging techniques:
Bagging encourages diversity in the ensemble, works better for “unstable” classifiers - e.g. decision trees, neural networks.

Set of classifiers is expected to benefit from random subspacing techniques:
RandomSubspacing adds more instability into the classifier (diversity) since different features are used when calculating distances.
k-NN would benefit from it

For my dataset, determine the best ensemble strategy for each of these classifiers. Discuss if this is in line with what you expected.
RandomSubspacing method worked well for k-NN classifier since it improved accuracy from 80% to approx. 83%.
Bagging method significantly improved the accuracy from 76% to 82% for j48 decision tree.
Bagging methods worked ordinary for Neural Networks. Accuracy did not improve drastically (78% to 79%) compared to other classifiers.
I used RandomSubSpace for Naïve Bayes and found it helpful to improve the accuracy from 72% to 74%. However, not as much as other classifiers stated above.

The results are in line with my expectations based on the below ensemble theories.
More accurate classifiers contribute more to the ensemble strategies.
Bagging can often reduce variance part of error.
Boosting can often reduce variance and bias, since it focuses on misclassified examples.
Boosting may sometimes increase error, as it is susceptible to noise and may lead to overfitting.