Machine Learning - Assignment 2

Conor Heffron - 23211267

Q1.1. Evaluate the performance of three basic classifiers on your dataset:

- Decision Tree
- Neural Network
- kNN (k=1)

Carefully consider the evaluation measure(s) that you use for this exercise and justify why you selected the particular evaluation measure(s).

Use the Weka Vote ensemble method (meta -> Vote) to combine the Decision Tree, Neural Network and 1-NN classifiers. Evaluate the performance of the Vote ensemble method with 3 different combination rules (there are 6 possibilities: Average of probabilities, Product of probabilities, Majority voting, Minimum probability, Maximum probability, Median). Provide a justification for the difference in accuracy when using different combination rules.

A.1.1

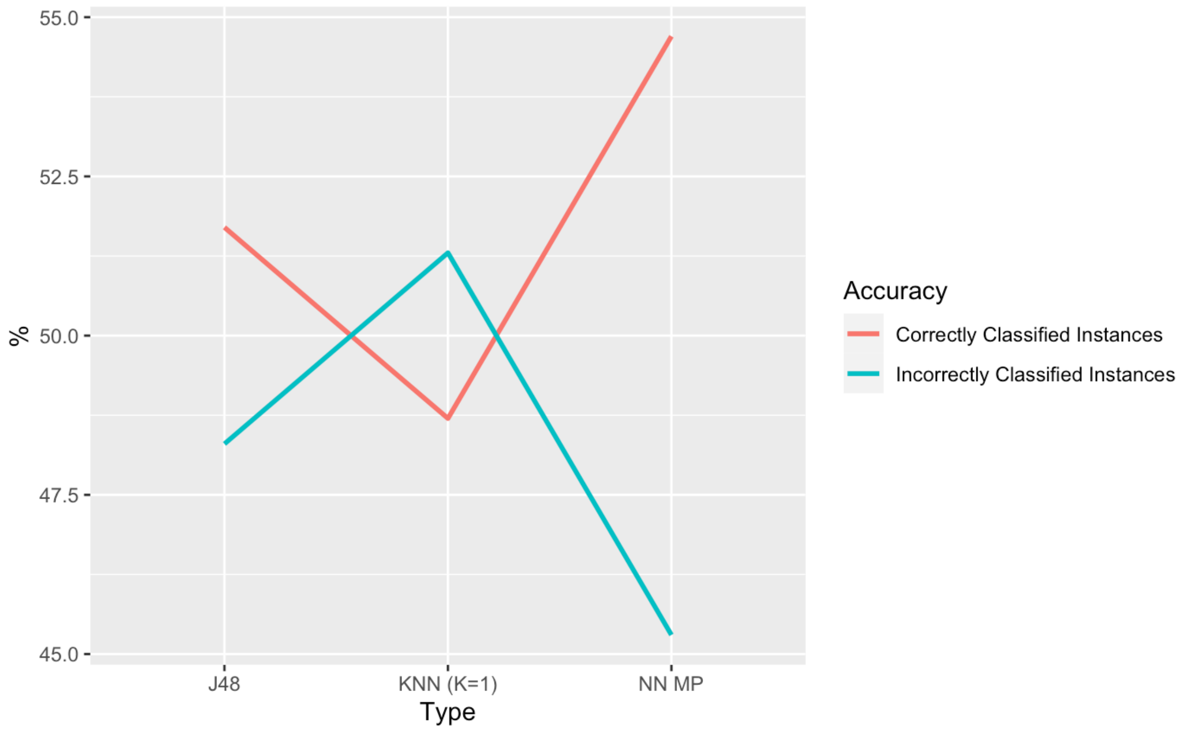
Initially, I ran some python code to normalise the Spotify features and compute z scores to get an idea of top features that may help to discern patterns from the data.

1	z_i_loudness	18263.640
2	z_i_energy	15708.758
3	z_i_danceability	14454.510
4	z_i_mode	12625.000
5	z_i_tempo	12105.081

The top 3 features from the Python script output were ‘loudness’, ‘energy’ and ‘danceability’. I noticed in Weka while using different attribute evaluator functions that ‘acousticness’, ‘speechiness’ and ‘intrumentalness’ features were often ranked higher in this regard while not ranked highly by my script (due to these features having less difference between min and max of column).

I completed my initial analysis in R. I have taken these features into account while using Euclidean distance and Hierarchical clustering to segment data from each genre into 5 clusters or groups to help identify a pattern like outliers for Latin music samples or a lower/smaller cluster graph in Rock music for ‘speechiness’ (more instrumental) whereas as Rap has a higher rate of ‘speechiness’, EDM has higher rates of ‘energy’ but less ‘acousticness’, higher rates of danceability in Pop and EDM etc can be inferred from the plots and data alike. I found this very interesting as I am a Spotify user with an eclectic taste in music.

Then, I ran the 3 classifiers individually without combination rule(s) via Voting and no bagging.



Enabling weighting for KNN made no difference, see Weka console output in zip attached.

When we look at TP and FP rates by genre - it appears Pop and Latin were the most difficult genres to classify whereas Rock, Rap and EDM were the most straight forward.

Neural network - Multilayer Perceptron (NN MP) was particularly strong at correctly classifying Rock, Rap and EDM samples and so was the best overall.

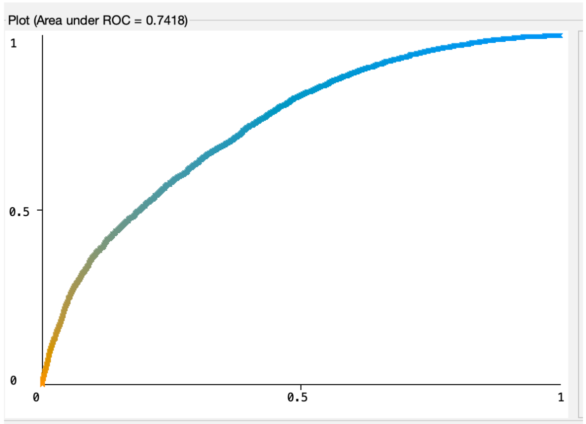
KNN was best at classifying Latin music samples (weighting made no difference), followed closely by decision tree (J48). However KNN was a bad measure overall for this data, J48 was better than KNN but not as good as NN MP.

MP NN was just about the best for Pop music samples but the difference between the 3 classifiers correctly classifying an observation was negligible for Pop samples. The ROC Area plot for Latin is clearly not as good a measure as say for example the equivalent for Rock samples. The ideal ROC curve has an AUC equal to 1. So suffice to say Rock has a better measure when 0.906 (Rock) > 0.742 (Latin).

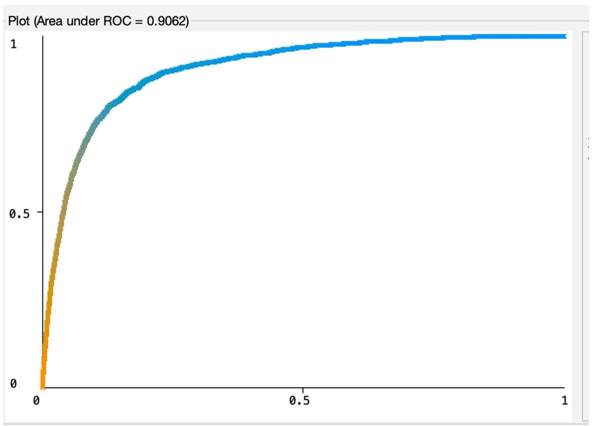
TP Rate	FPRate	Precision	Recall	F-Measure	MCC	ROCArea	PRCArea	Class
0.623	0.107	0.628	0.623	0.625	0.518	0.846	0.681	edm
0.386	0.117	0.435	0.386	0.409	0.282	0.742	0.404	latin
0.362	0.153	0.372	0.362	0.367	0.211	0.696	0.330	pop

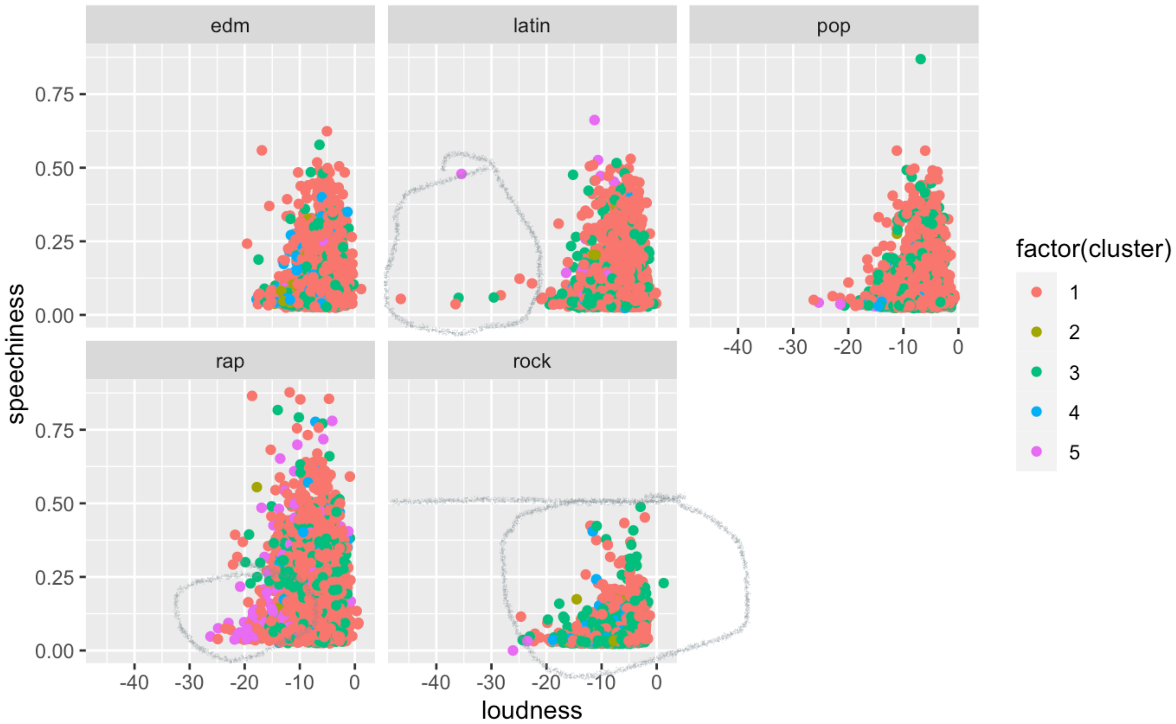
TP Rate	FPRate	Precision	Recall	F-Measure	MCC	ROCArea	PRCArea	Class
0.632	0.096	0.631	0.632	0.632	0.536	0.866	0.652	rap
0.729	0.093	0.631	0.729	0.677	0.602	0.906	0.682	rock
0.547	0.113	0.541	0.547	0.543	0.431	0.811	0.552	WeightedAvg

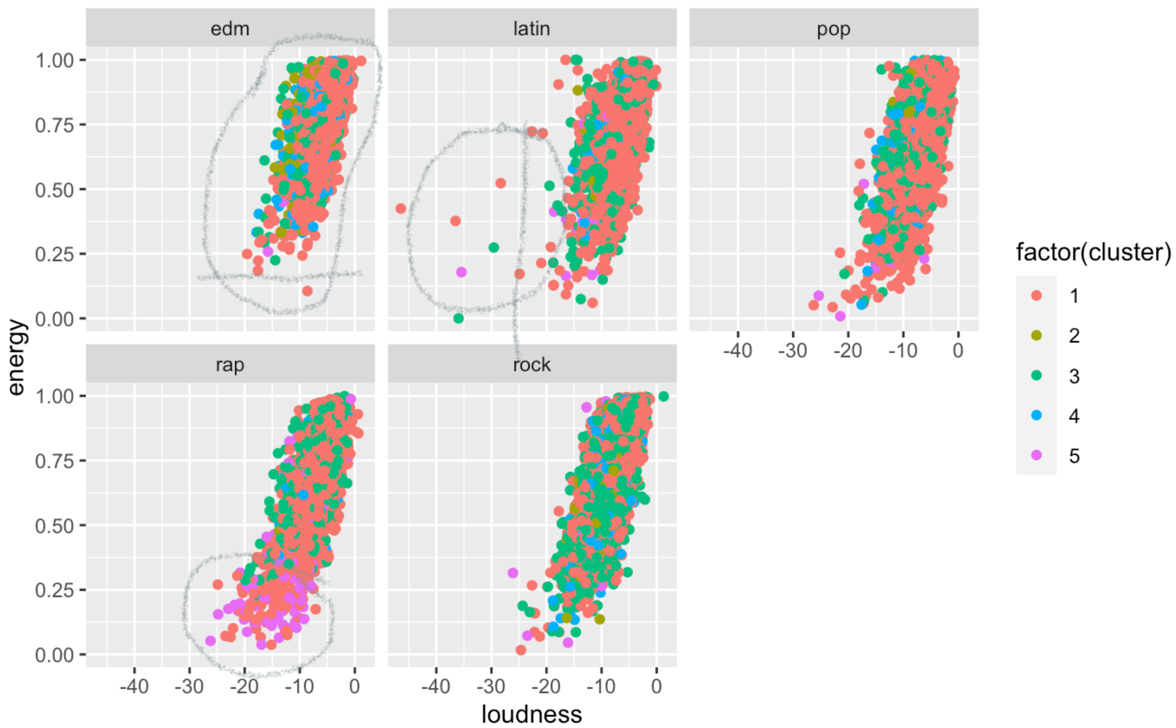
Latin ROC Plot / Threshold Curve

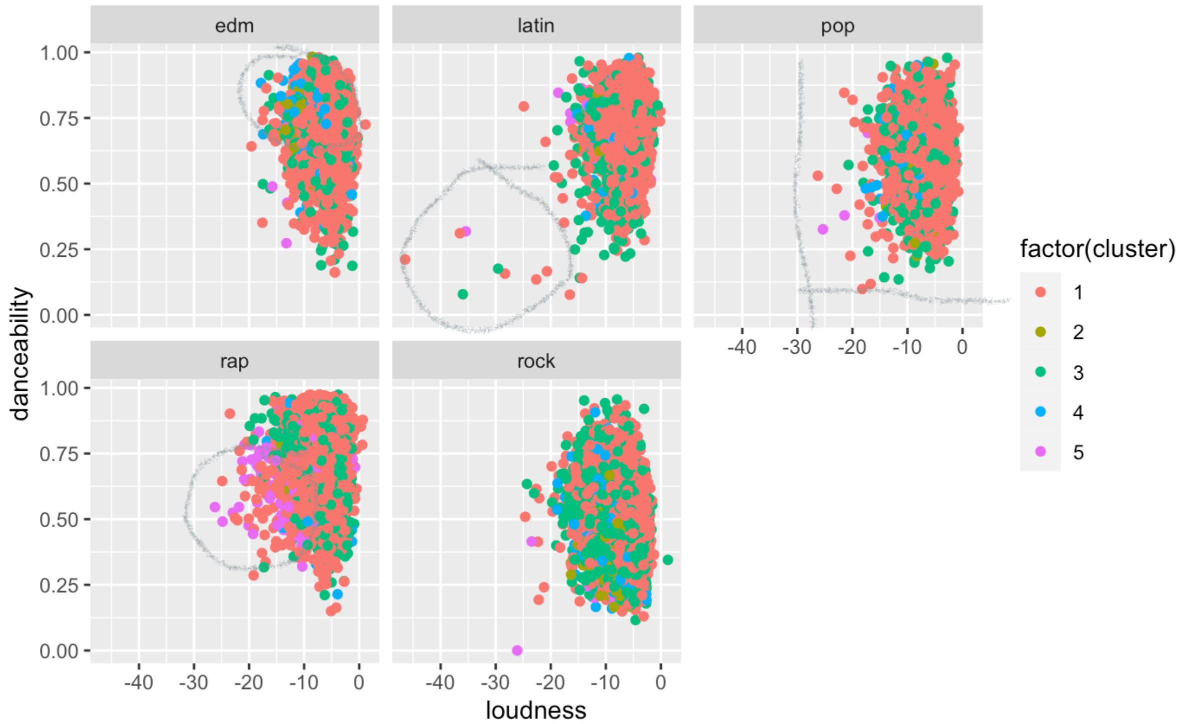


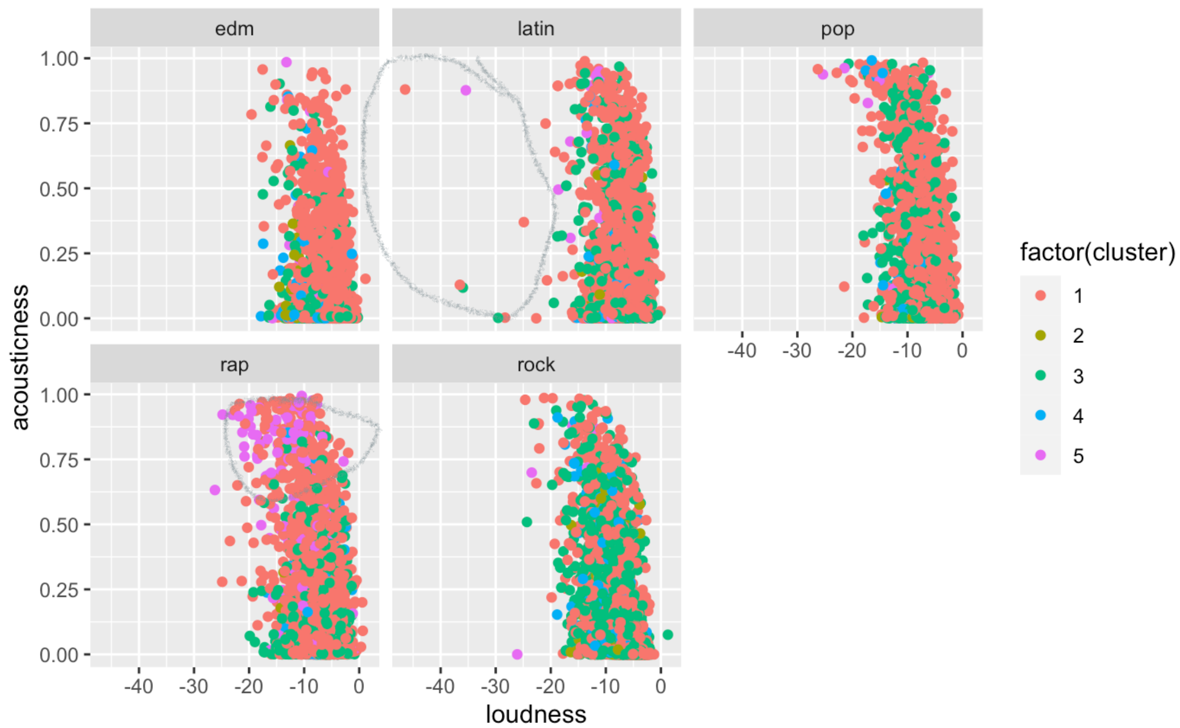
Rock ROC Plot / Threshold Curve




edm
latin
pop
rap
rock
speechiness
loudness
0.00
0.25
0.50
0.75
-40
-30
-20
-10
0
factor(cluster)
1
2
3
4
5


edm
latin
pop
rap
rock
energy
loudness
0.00
0.25
0.50
0.75
1.00
-40
-30
-20
-10
0
factor(cluster)
1
2
3
4
5





We can see from the above diagrams a depiction of 5 clusters per playlist genre. Latin appears to have the most outliers that are somewhat distant from the main cluster of points.

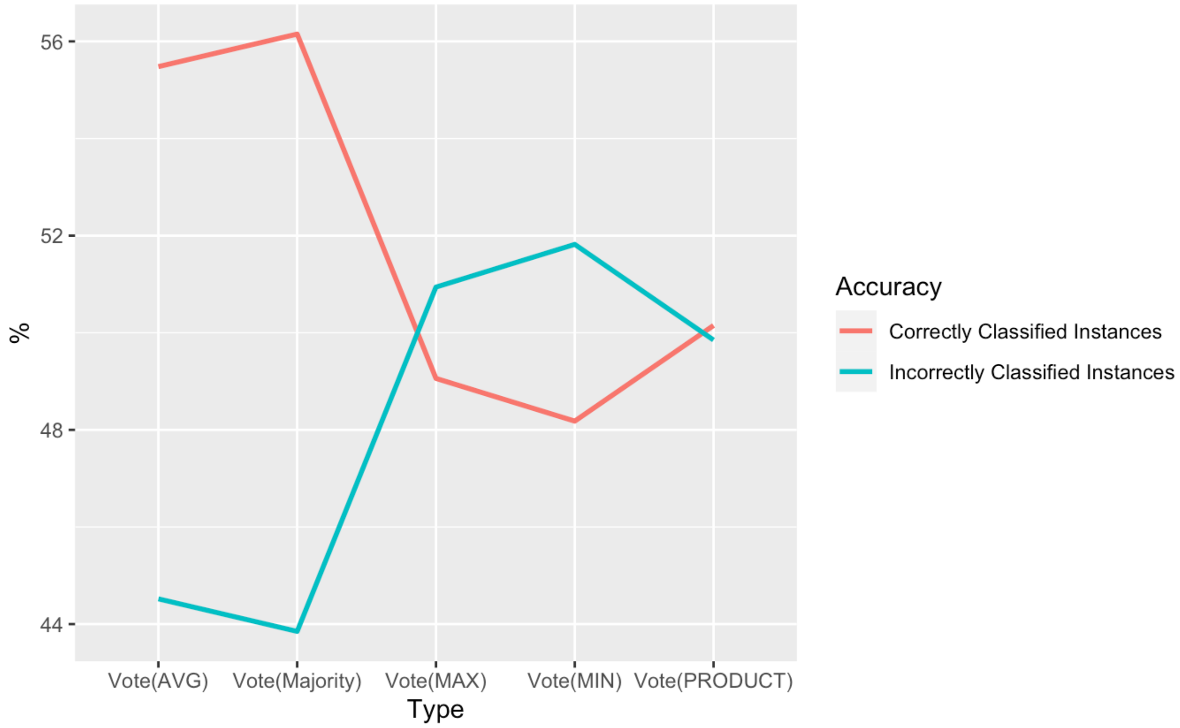
I used the Weka vote ensemble to combine the decision tree (J48), NN MP and KNN(lbk where k=1) classifiers. I would like to compare the results of 3 combination rules:

1. Majority Voting

2. Average of Probabilities

3. Product of Probabilities

In addition, I ran for min/max of probabilities combination rules but the results were inaccurate – there was more samples classified incorrectly than correctly for both of these as shown in the plot below.



Majority voting was the most accurate followed closed by the **average of probabilities**. The product of probabilities was quite a bit less accurate in relative terms - 56% (Majority Vote):55% (Average of Probs.):50% (Product of Probs.) correctly classified. Product of probabilities only just had a positive classification rate of 50%:49%.

	Accuracy	n	%	Type
1	Correctly Classified Instances	12102	55.48	avg
2	Incorrectly Classified Instances	9710	44.52	avg
3	Correctly Classified Instances	12248	56.15	majority vote
4	Incorrectly Classified Instances	9564	43.85	majority vote
5	Correctly Classified Instances	10700	49.06	max
6	Incorrectly Classified Instances	11112	50.94	max
7	Correctly Classified Instances	10509	48.18	min
8	Incorrectly Classified Instances	11303	51.82	min
9	Correctly Classified Instances	10938	50.15	product
10	Incorrectly Classified Instances	10874	49.85	product

We can now see that there is a benefit to use a combination of rules to classify this data. The subtleties of classifying Pop and Latin music in particular was troublesome for each classifier used. However overall and for Rap, Rock and EDM – Multilayer Perceptron appears to be the best option.

I think majority voting performs best here because this gives each classifier an equal weighting to vote on what is the correct genre per record. The downside is that it took a long time to run all of this through Weka once NN MP was included, the runtime increased further with an increasing bag size for Weka runs for the next question.

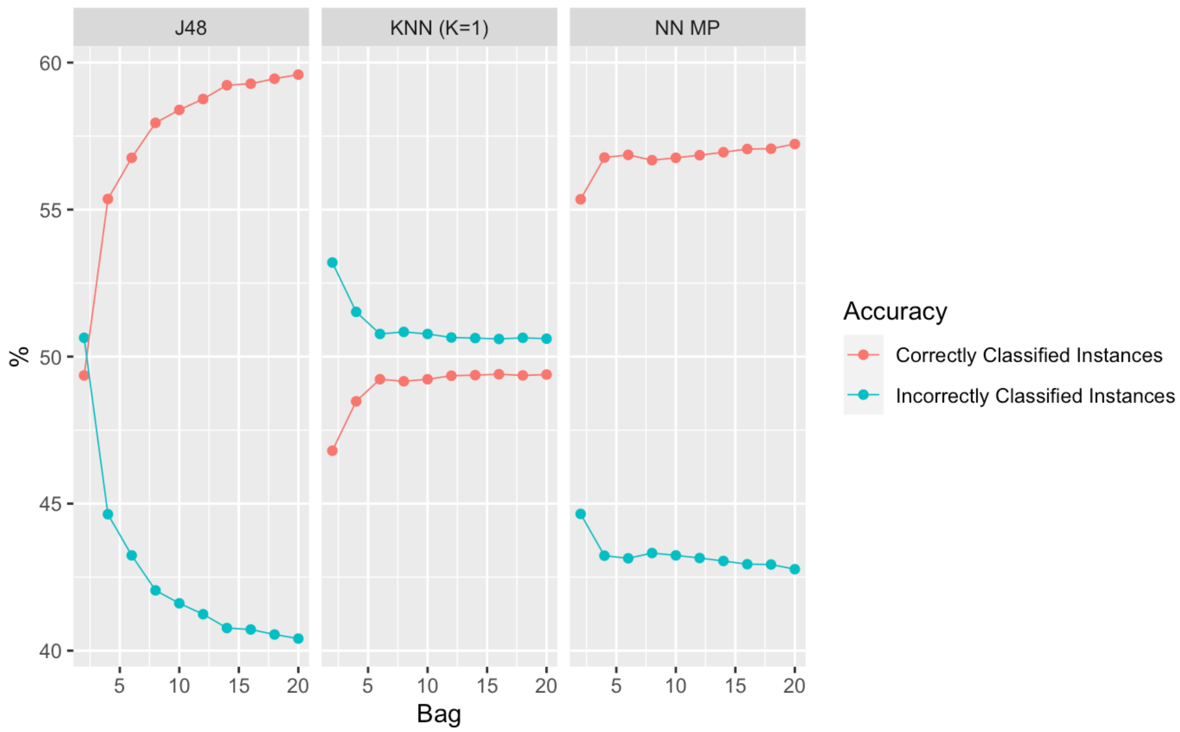
1.2. Return to the full data set and apply ensembles with bagging using the three classifiers from Task (a). Investigate the performance of these classifiers as the ensemble size increases (e.g., in steps of 2 from 2 to 20 members). Using the best performing ensemble size, investigate how changing the number of instances in the bootstrap samples affects classification performance (i.e. the "bag size").

A.1.2

I applied ensembles with bagging using the three classifiers from the previous task. All three classifiers had a significant improvement at the start which became more linear and predictable when increasing the bag size increments of 2.

The biggest improvement was for J48 tree (59% correct:40% incorrect classification of samples). The NN MP also showed an improvement in accuracy. The KNN classifier where K=1 showed small improvement but still had more samples classified incorrectly than correctly when bag size was set to 20. Please see plot below to help give a better visualisation of this information.

See relevant attachments for further information on workings and Weka console output.
Note: This took quite some time to run with increasing bag size (number of iterations 2-> 20) with 10-fold cross-validation. Due to this I have rushed some workings and descriptions.



1.3. Apply ensembles with random sub spacing using the three classifiers from Task (a). Investigate the performance of these classifiers as the ensemble size increases (e.g., in steps of 2 from 2 to 20 members). Using the best performing ensemble size, investigate how changing the number of features used when applying random sub spacing affects classification performance (i.e. the “subspace size”).

A.1.3

Encourages diversity in the ensemble, works well for *KNN and* J48 decision trees (Random Forest).

The performance improved greatly with the correctly classified samples rate jumping to 58.5% with sub space size of 16.

Console output:

```
=== Run information ===

Scheme:       weka.classifiers.meta.Vote -S 1 -B "weka.classifiers.trees.J48 -C
0.25 -M 2" -B "weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 16
-W weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0" -B
"weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E
20 -H a" -B "weka.classifiers.lazy.IBk -K 1 -W 0 -A
\"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R
first-last\\\"\"" -batch-size 2 -R MAJ
Relation:     spotify-23211267
Instances:    21812
Attributes:   13
              danceability
              energy
              key
              loudness
              mode
              speechiness
              acousticness
              instrumentalness
              liveness
              valence
              tempo
              duration_ms
              playlist_genre
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Vote combines the probability distributions of these base learners:
        weka.classifiers.trees.J48 -C 0.25 -M 2
        weka.classifiers.meta.RandomSubSpace -P 0.5 -S 1 -num-slots 1 -I 16 -W
weka.classifiers.trees.REPTree -- -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0
        weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -
S 0 -E 20 -H a
```

```
        weka.classifiers.lazy.IBk -K 1 -W 0 -A
"weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R
first-last\""
using the 'Majority Voting' combination rule.
.
.
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       12761               58.5045 %
Incorrectly Classified Instances      9051               41.4955 %
Kappa statistic                          0.4806
Mean absolute error                      0.166
Root mean squared error                  0.4074
Relative absolute error                 51.9432 %
Root relative squared error            101.9247 %
Total Number of Instances            21812

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area
PRC Area  Class
                 0.668    0.108    0.640      0.668    0.654      0.552    0.780
0.502     edm
                 0.439    0.105    0.494      0.439    0.465      0.350    0.667
0.323     latin
                 0.362    0.133    0.406      0.362    0.383      0.239    0.614
0.275     pop
                 0.685    0.090    0.665      0.685    0.674      0.588    0.797
0.520     rap
                 0.770    0.083    0.670      0.770    0.717      0.652    0.844
0.557     rock
Weighted Avg.    0.585    0.104    0.576      0.585    0.579      0.476    0.740
0.436

=== Confusion Matrix ===

    a    b    c    d    e   <-- classified as
 3260  379  721  329  190 |    a = edm
  473 1809  799  732  311 |    b = latin
  844  728 1583  410  812 |    c = pop
  318  574  356 3088  175 |    d = rap
  198  174  441   87 3021 |    e = rock
```

1.4. Based on the lectures, which set of classifiers is expected to benefit from bagging techniques more and which set of classifiers is expected to benefit from random sub spacing techniques more? For your dataset, determine the best ensemble strategy for each of these classifiers. Discuss if this is in line with what you expected.

A.1.4

Bagging can often reduce the variance part of error and we see that as we increment bag size by 2. I think bagging is suited to more complex runs such as: Deep Decision Trees (J48) and NN MP with added complexity. It could be suitable for KNN when k is very low.

I do not think it is suitable for small decision trees or simple linear models.

Overall, I think it is best suited to NN MP as it can learn and model non-linear and complex relationships, work well when training data is noisy and fast performance once a network is trained is possible.

However, for NN MP, many training examples is probably needed. The training time can be very long especially if added as a combination rule for voting ensemble. It is also quite complicated and difficult to understand the internal workings unlike some other classifiers.

1.5. Some of the features may be correlated with others or have dependencies on other features. Build a linear regression model to predict the energy feature from this set of features: tempo, loudness, liveness. Compute the regression model using Linear Regression and SGD. Show the regression model and comment on the quality of the model and any differences you observe between SGD and Linear Regression.

A.1.5

0.0339 (loudness) + 0.0778 (liveness) + 0.0001 (tempo). Tempo and loudness are significant variables for predicting energy values while liveness may not be a great source of info.

- Liveness > 0.05 so normally would accept null hypothesis or insignificant relationship to energy.
- Loudness and Tempo < 0.05 which is significant.
- Tempo in particular seems to have a very significant relationship as it is much closer to *zero.*

```
=== Run information ===

Scheme:       weka.classifiers.functions.LinearRegression -S 1 -R 1.0E-8 -num-
decimal-places 4
Relation:     spotify-23211267
Instances:    21812
Attributes:   13
              danceability
              energy
              key
              loudness
              mode
              speechiness
              acousticness
              instrumentalness
```

```
              liveness
              valence
              tempo
              duration_ms
              playlist_genre
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===


Linear Regression Model

energy =

     -0.1548 * danceability +
      0.0004 * key +
      0.0339 * loudness +
     -0.0028 * mode +
      0.0585 * speechiness +
     -0.2387 * acousticness +
      0.0904 * instrumentalness +
      0.0778 * liveness +
      0.1345 * valence +
      0.0001 * tempo +
      0      * duration_ms +
      0.016  * playlist_genre=pop,latin,rock,edm +
      0.0128 * playlist_genre=latin,rock,edm +
      0.0298 * playlist_genre=rock,edm +
      0.0057 * playlist_genre=edm +
      0.9101

Time taken to build model: 0.05 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.796
Mean absolute error                      0.0822
Root mean squared error                  0.105
Relative absolute error                 58.826  %
Root relative squared error             60.5253 %
Total Number of Instances            21812
```

Overall the correlation coefficient is > 0.7 so this represents a strong relationship between variables (ranges from -1 to 1 or weak to strong correlation).

The smaller the p-value is, the more significant the factor is. P-value = 0.05 is a reasonable threshold.

The P values for tempo, loudness and liveness is < 2.2e-16, this means a significant relationship with the response variable (energy) in the model as P value is much less than 0.05.

```
Call:
lm(formula = energy ~ tempo + loudness + liveness, data = spotify_23211267)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.53589 -0.08146  0.00745  0.08633  1.25599

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 8.834e-01  4.736e-03  186.54   <2e-16 ***
tempo       5.312e-04  3.237e-05   16.41   <2e-16 ***
loudness    3.872e-02  2.877e-04  134.57   <2e-16 ***
liveness    1.189e-01  5.382e-03   22.09   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1253 on 21808 degrees of freedom
Multiple R-squared:  0.4783,      Adjusted R-squared:  0.4782
F-statistic:  6665 on 3 and 21808 DF,  p-value: < 2.2e-16

                   2.5 %       97.5 %
(Intercept) 0.8740980328 0.8926619375
tempo       0.0004677832 0.0005946626
loudness    0.0381533842 0.0392812685
liveness    0.1083588727 0.1294589853


0.1740097
```