Vienna, 29th of April 2019

Vienna University of Technology

Machine Learning 2019 Summer Term

Exercise 1 - Classification

Aleksander Grzymek (01428243)

Carlos A. Vargas Rivera (11823257)

Experiments with the algorithms and parameters:

In this exercise, we decided to use Random Forest, K-Nearest-Neighbours and Naive Bayes algorithms for classifying the datasets. For each algorithm, we have tested different parameters and observed different behaviours for changes in the parameters.

We also tested how the training:test ratio influences the results. We did it by implementing 10-fold cross validation beginning with 95:5 with a 10% step.

While using the Random Forest algorithm we tested the following parameters: the **maximal tree depth**, **the number of trees**, **the minimum samples to split and the minimum samples for a leaf.**

For the Naive Bayes algorithm, we set the **smoothing** parameter **alpha**.

For the K-Nearest-Neighbours we set the number of neighbours, the voting algorithm and the weighting method.

Four Datasets were used for the analysis:

Two datasets were given by the specification of the analysis:

1. Amazon reviews
(https://inclass.kaggle.com/c/184702-tu-ml-ss-19-amazon-commerce-reviews/data)

2. Congressional Voting
(https://inclass.kaggle.com/c/13939/download-all)

Two datasets chosen from open-dataset sources:

3. Cardiotocography Data Set
(https://archive.ics.uci.edu/ml/datasets/cardiotocography)

4. Flags Data Set (http://archive.ics.uci.edu/ml/datasets/Flags)

Characteristics of the datasets:

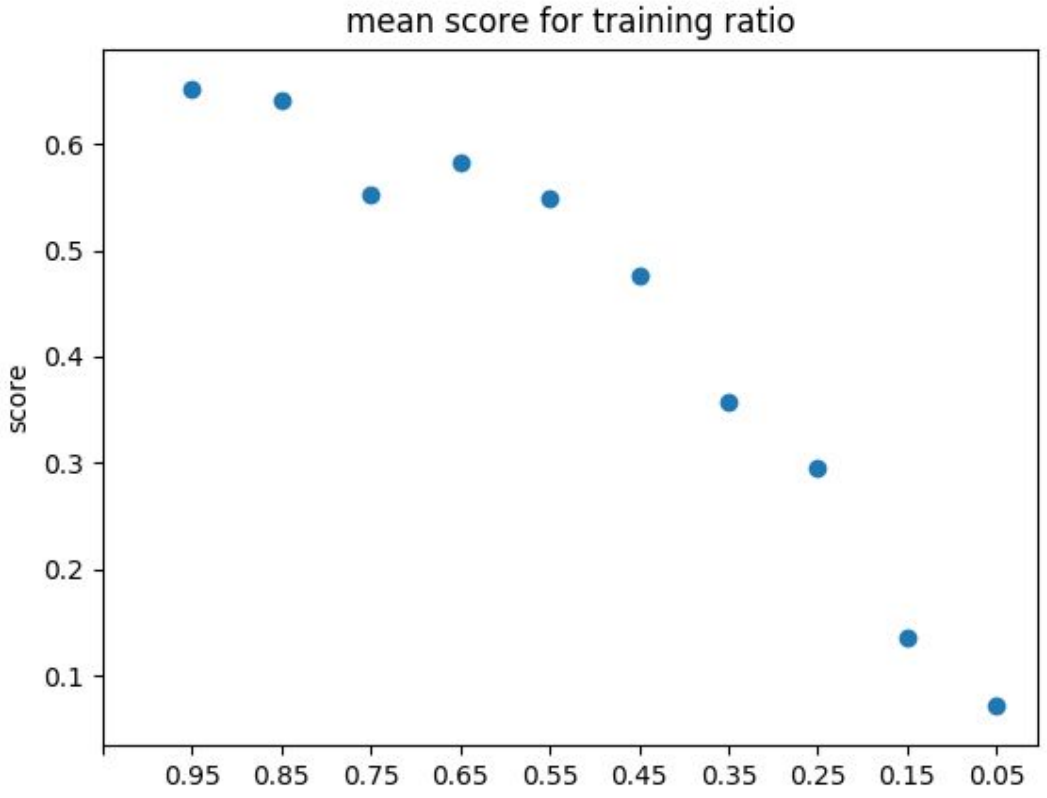
1. Amazon reviews

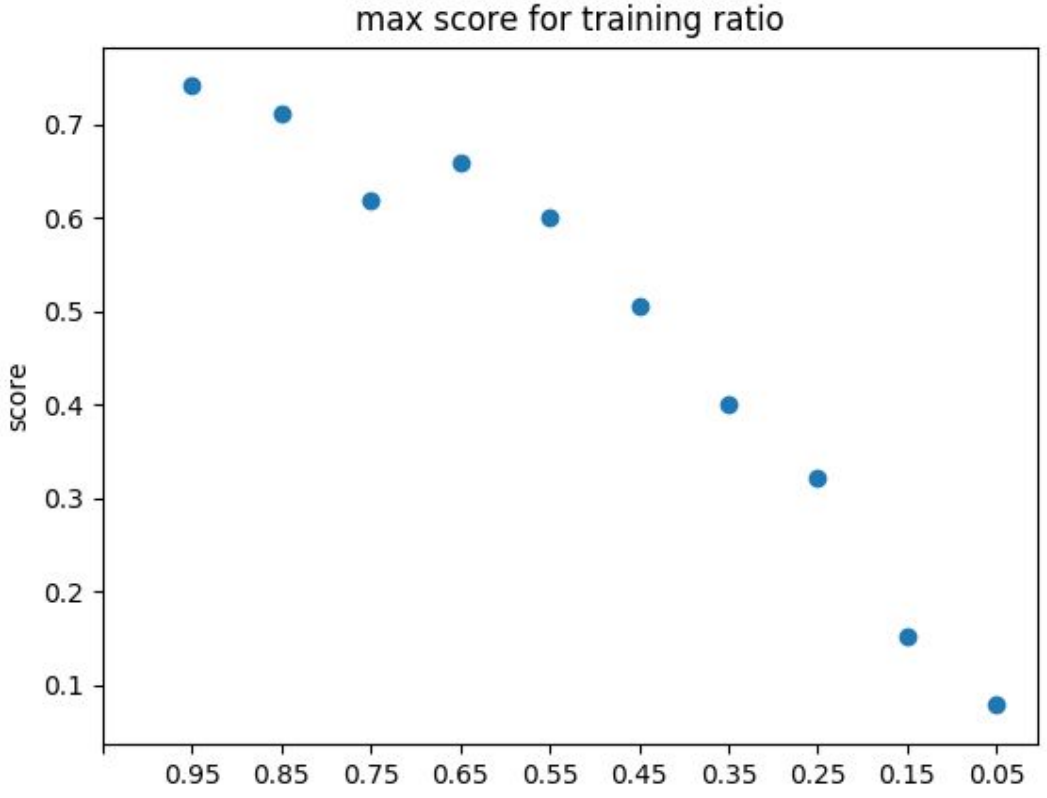
We cannot really state about this dataset as there are only 10002 columns with only 2 described as ID and class. However the computer can still learn from such dataset.

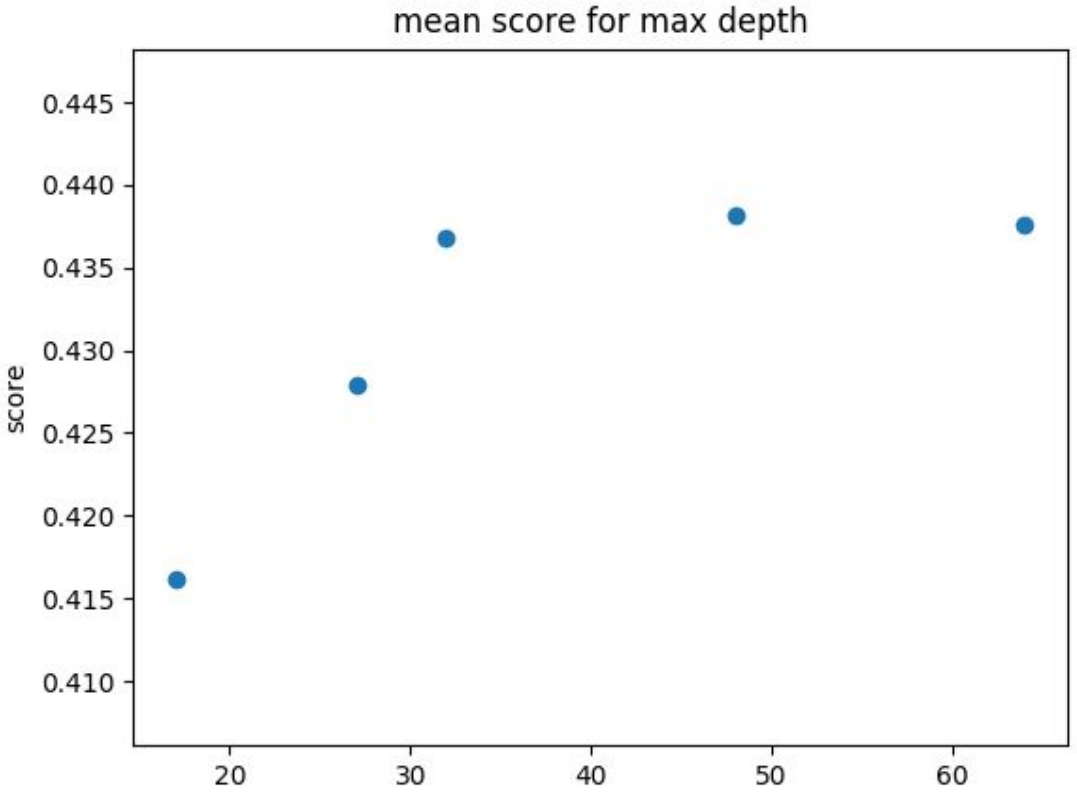
Experiments with the algorithms and parameters.

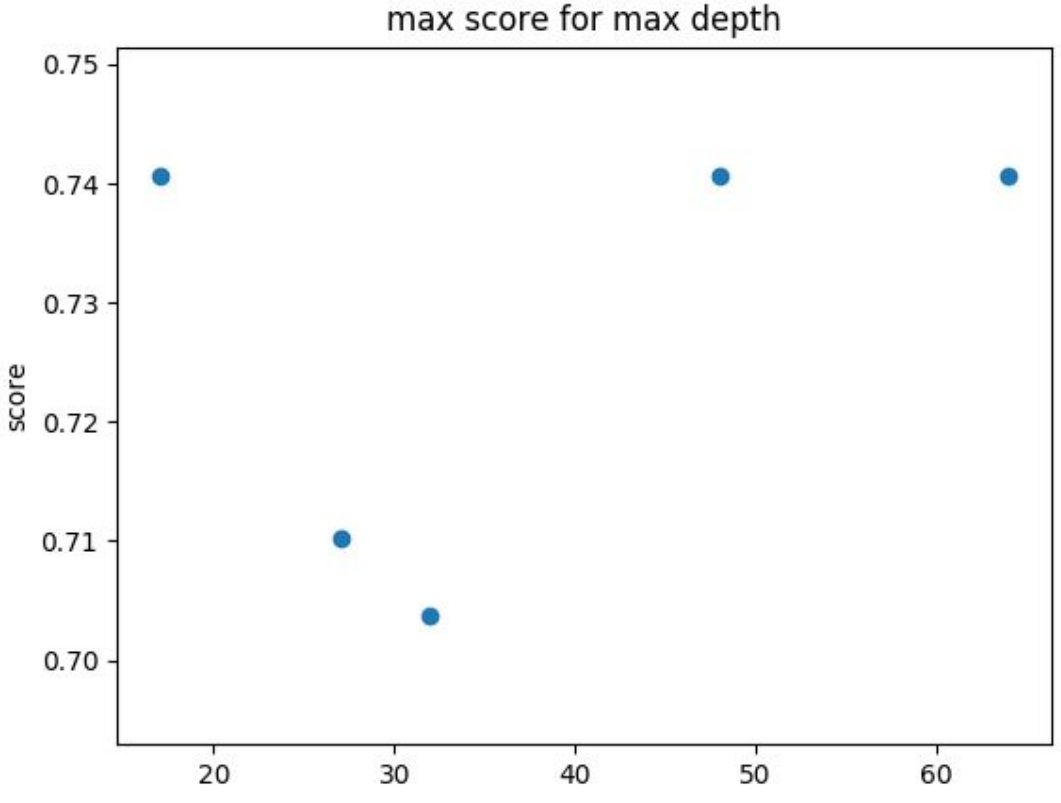
Random Forest:

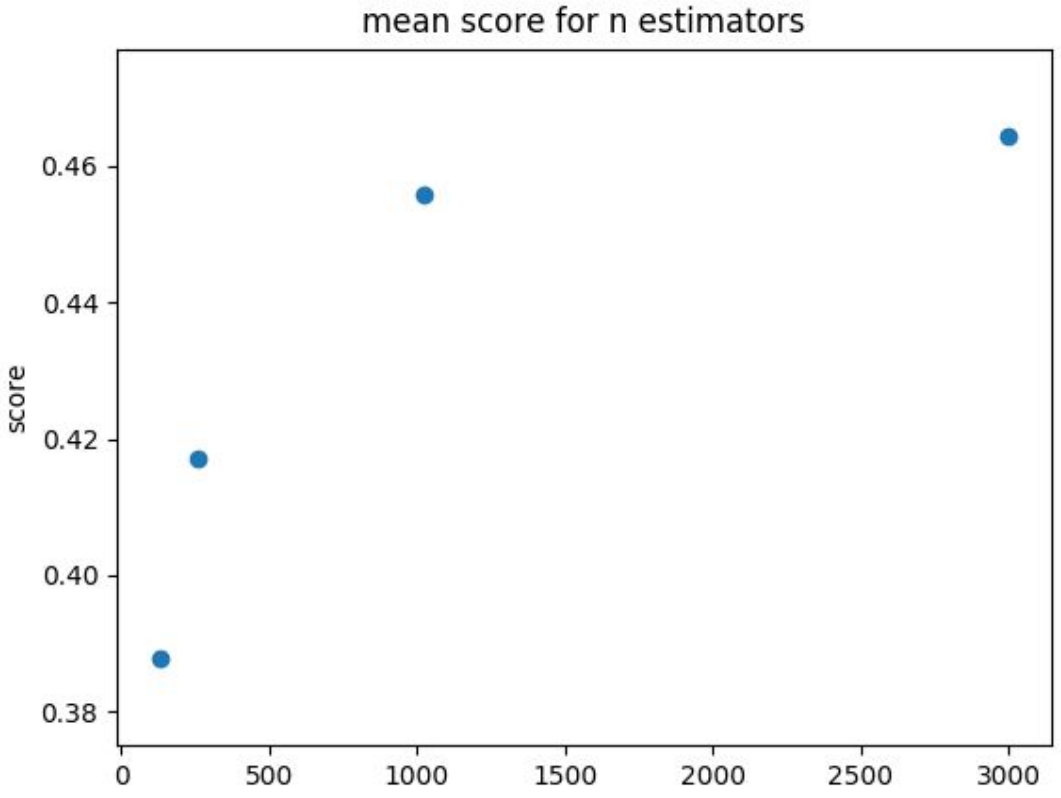
On this dataset we can observe a strong correlation between the training:test ratio and the result. It is almost linear. We found out, the mean score increases with the increasing tree depth, however only until about the depth of 32. We can also see in the figures below, that the mean score is increasing for the increasing number of trees. We tested it up to 20001 trees still getting little improvements.
The behaviour can be observed in the figures below.

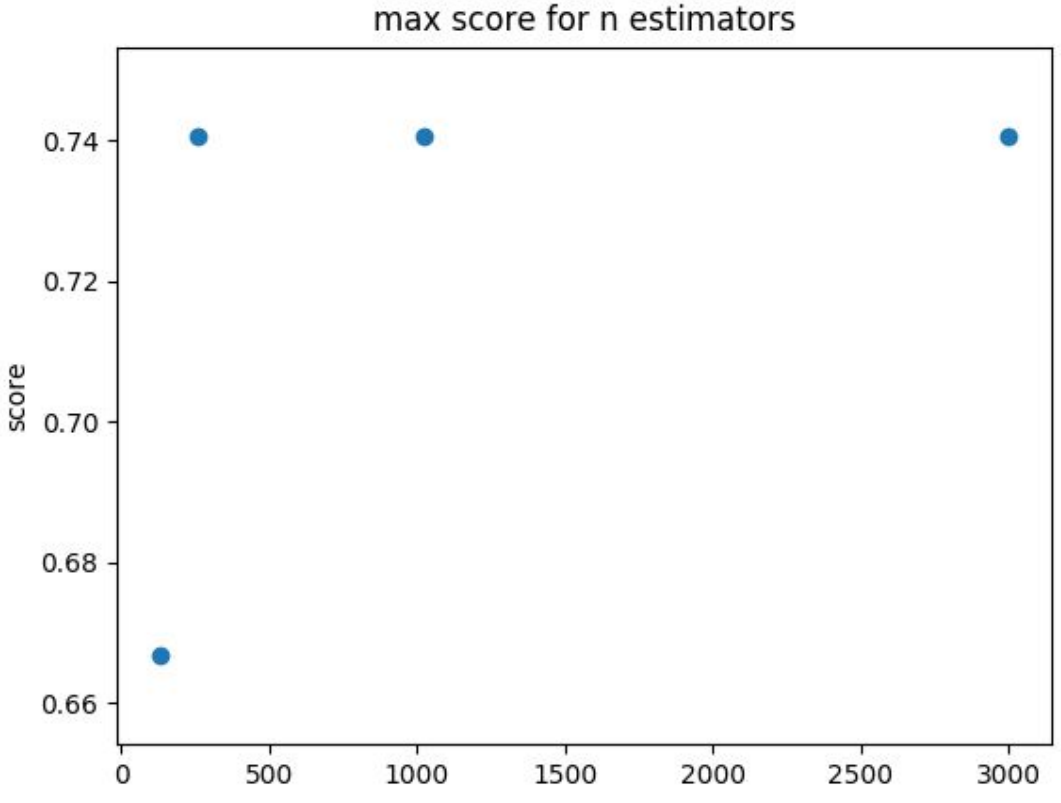

mean score for training ratio
score
0.6
0.5
0.4
0.3
0.2
0.1
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05


max score for training ratio
score
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05


mean score for max depth
score
0.445
0.440
0.435
0.430
0.425
0.420
0.415
0.410
20
30
40
50
60


max score for max depth
score
0.75
0.74
0.73
0.72
0.71
0.70
20
30
40
50
60


mean score for n estimators
score
0.46
0.44
0.42
0.40
0.38
0
500
1000
1500
2000
2500
3000

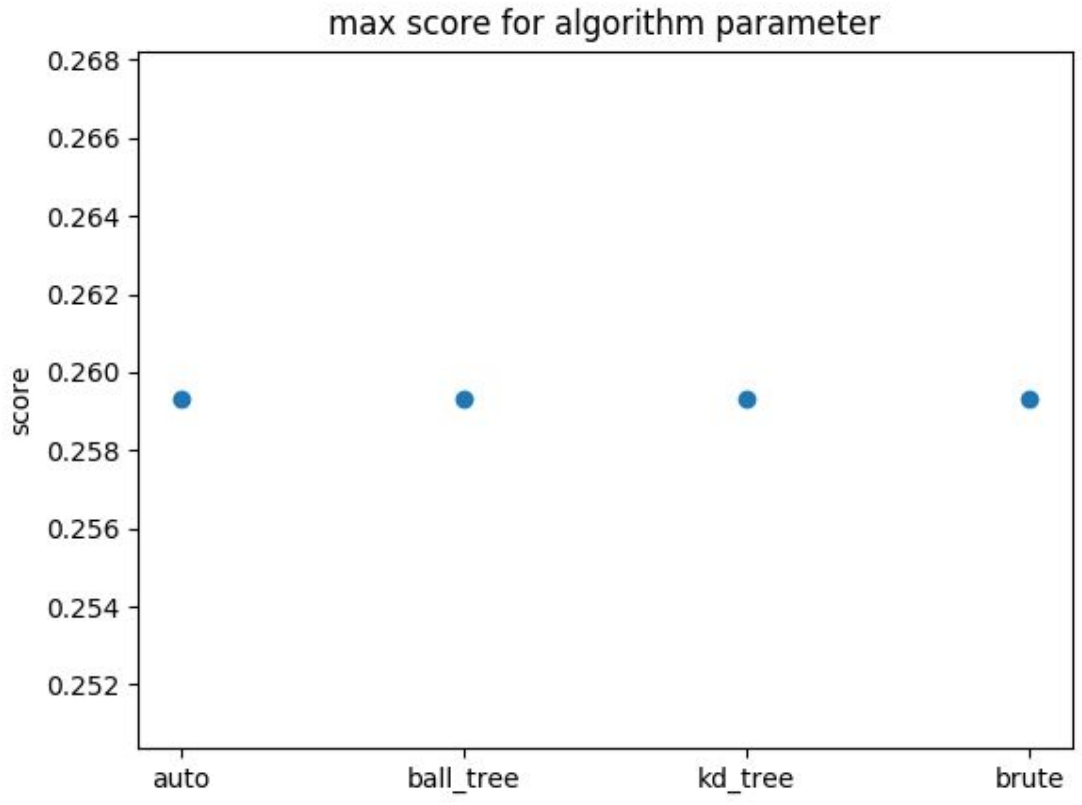

max score for n estimators
score
0.74
0.72
0.70
0.68
0.66
0
500
1000
1500
2000
2500
3000

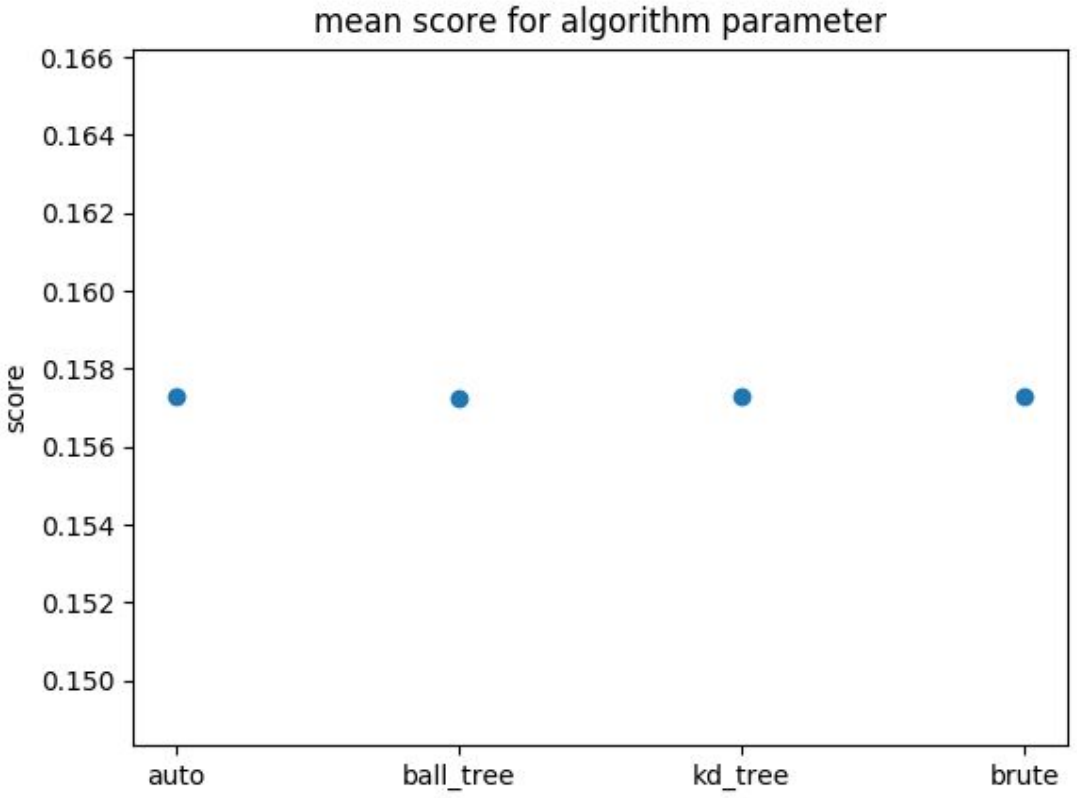
Naive Bayes:

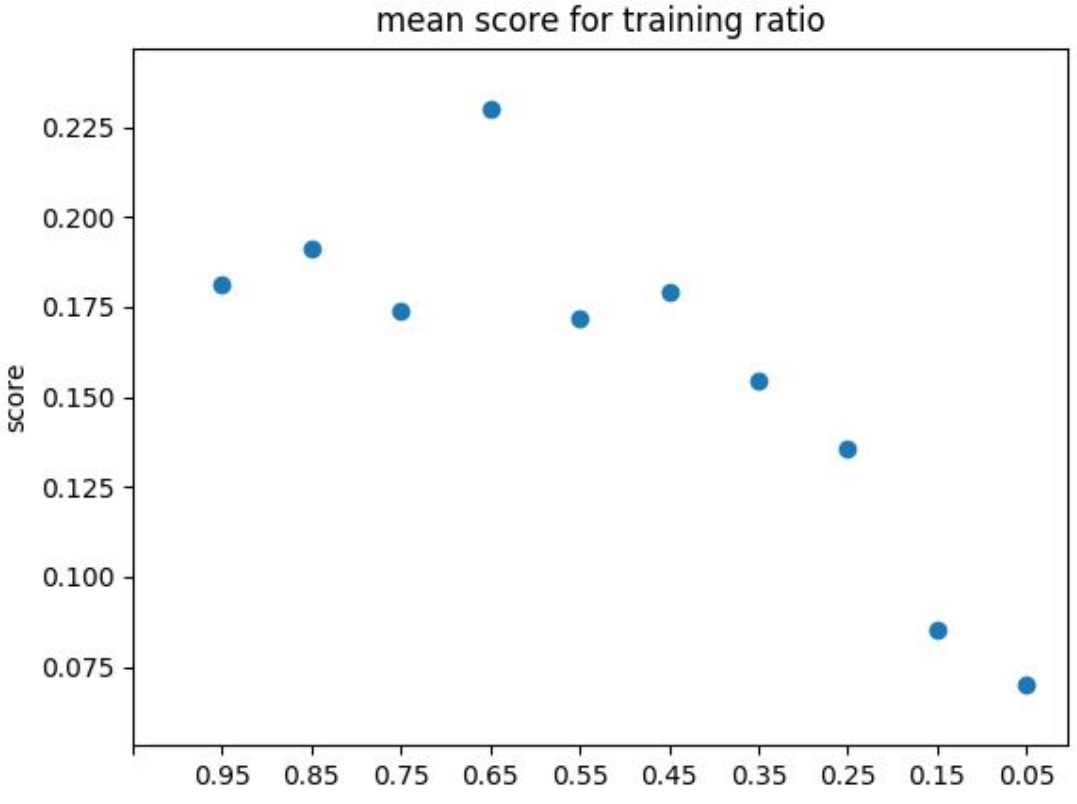
Naive Bayes performed on this dataset almost as well as the random forest. We found out, the optimal alpha (maybe it is only a local optimum) would be about 0.161. We tested the values from 1e-9 to 3.

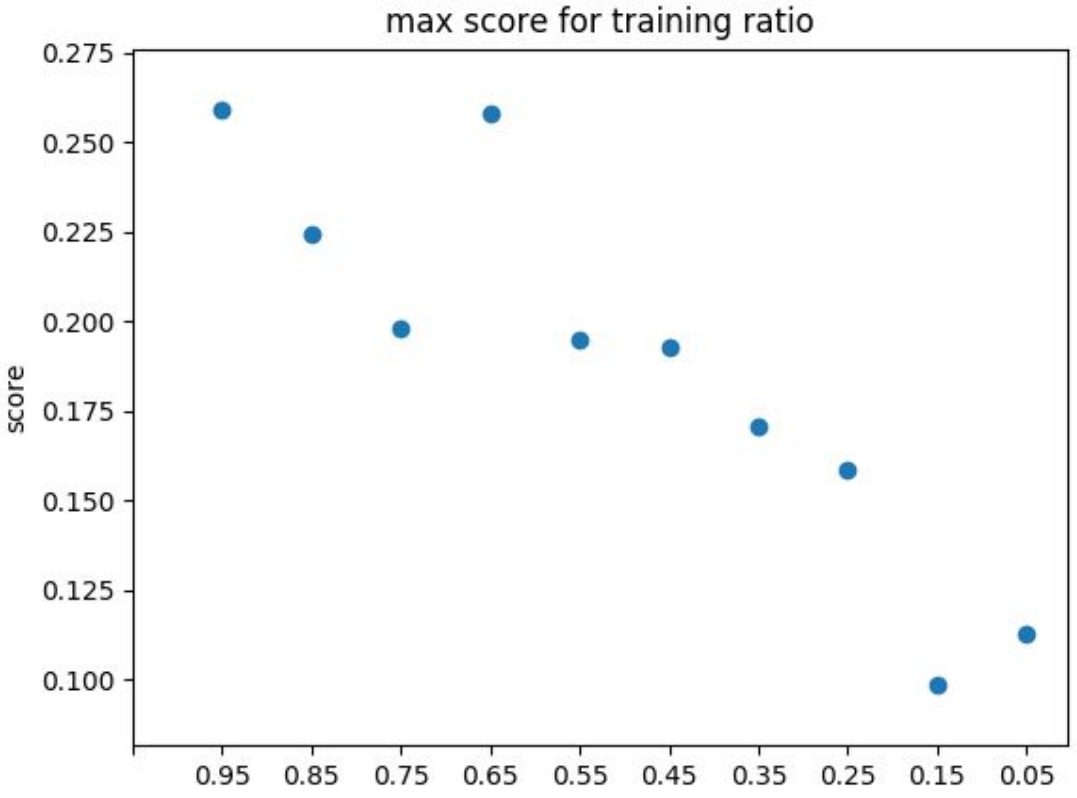
K-Nearest-Neighbours

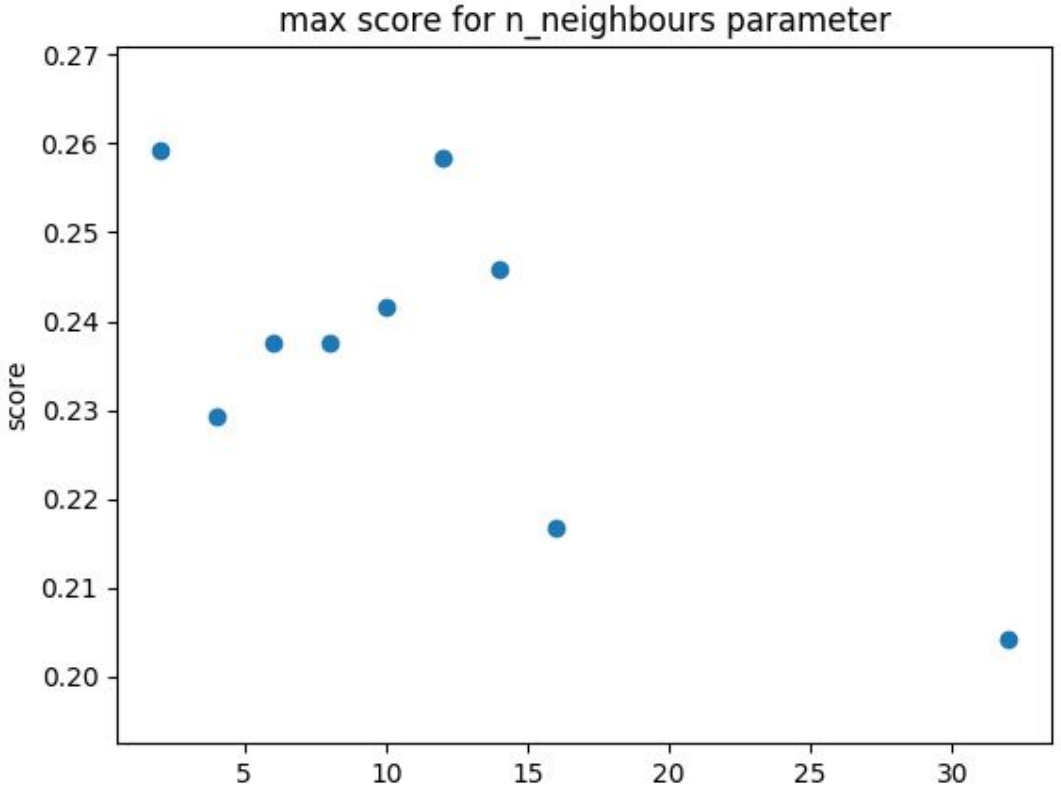
K-Nearest Neighbours got a very low score in this dataset and we dropped any further reasearch on it.

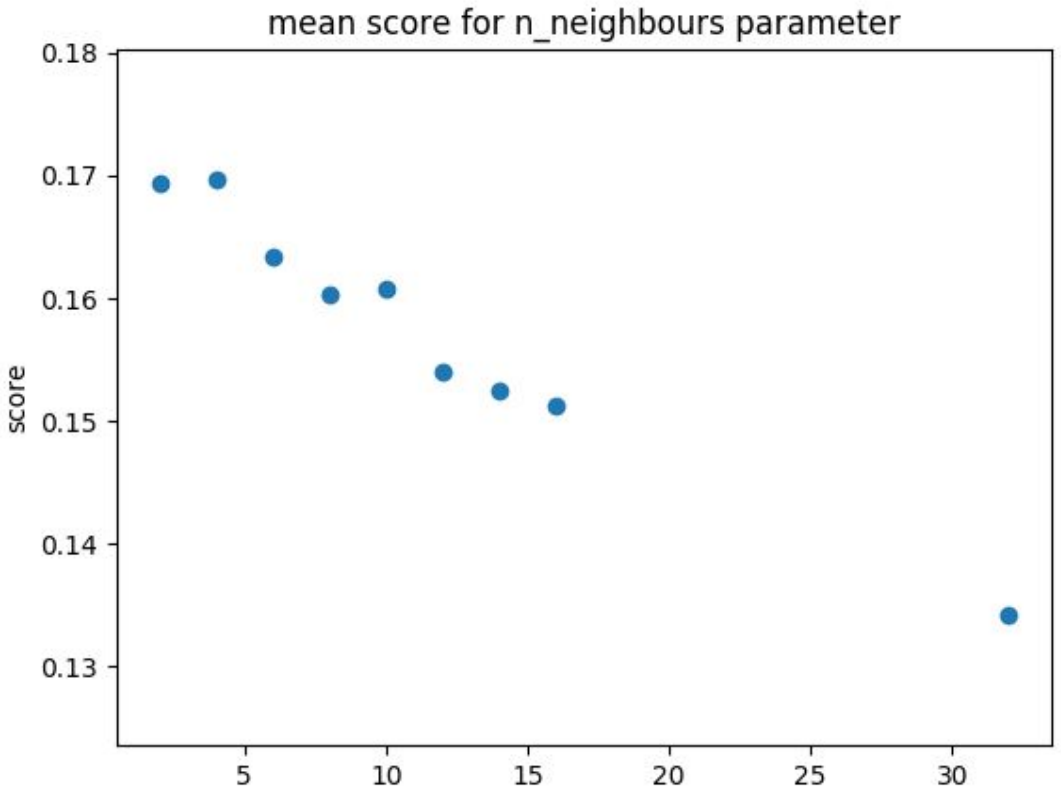


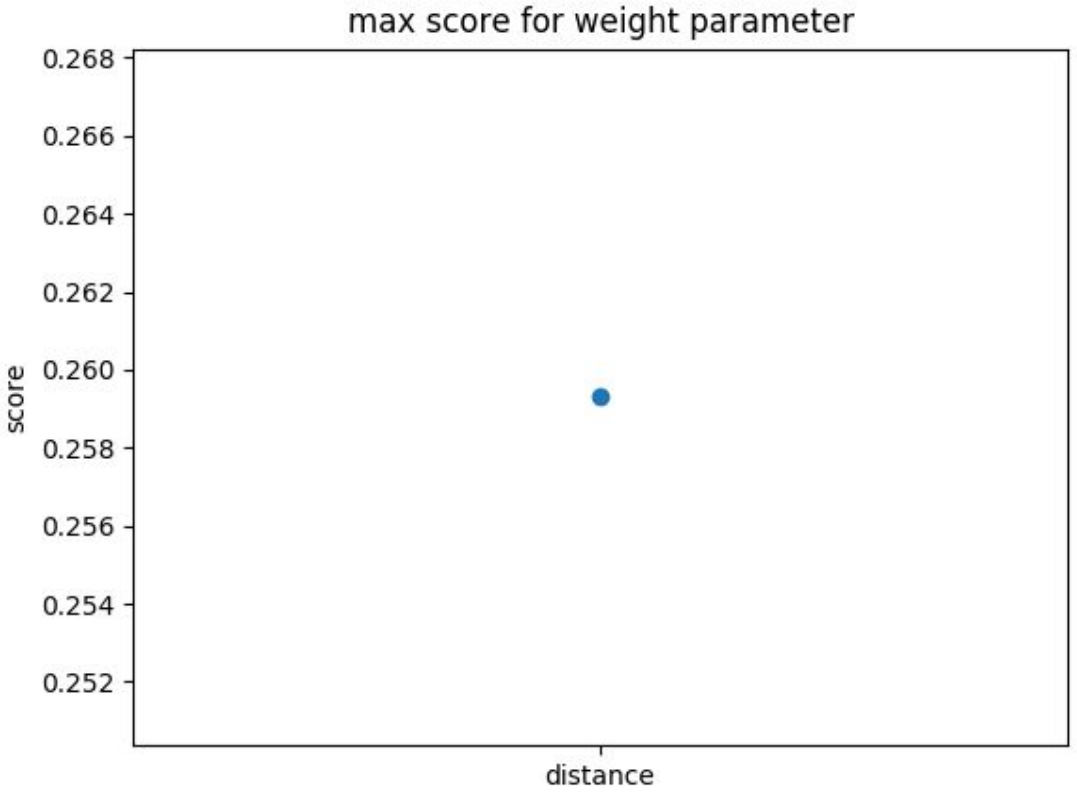

mean score for algorithm parameter
score
0.166
0.164
0.162
0.160
0.158
0.156
0.154
0.152
0.150
auto
ball_tree
kd_tree
brute

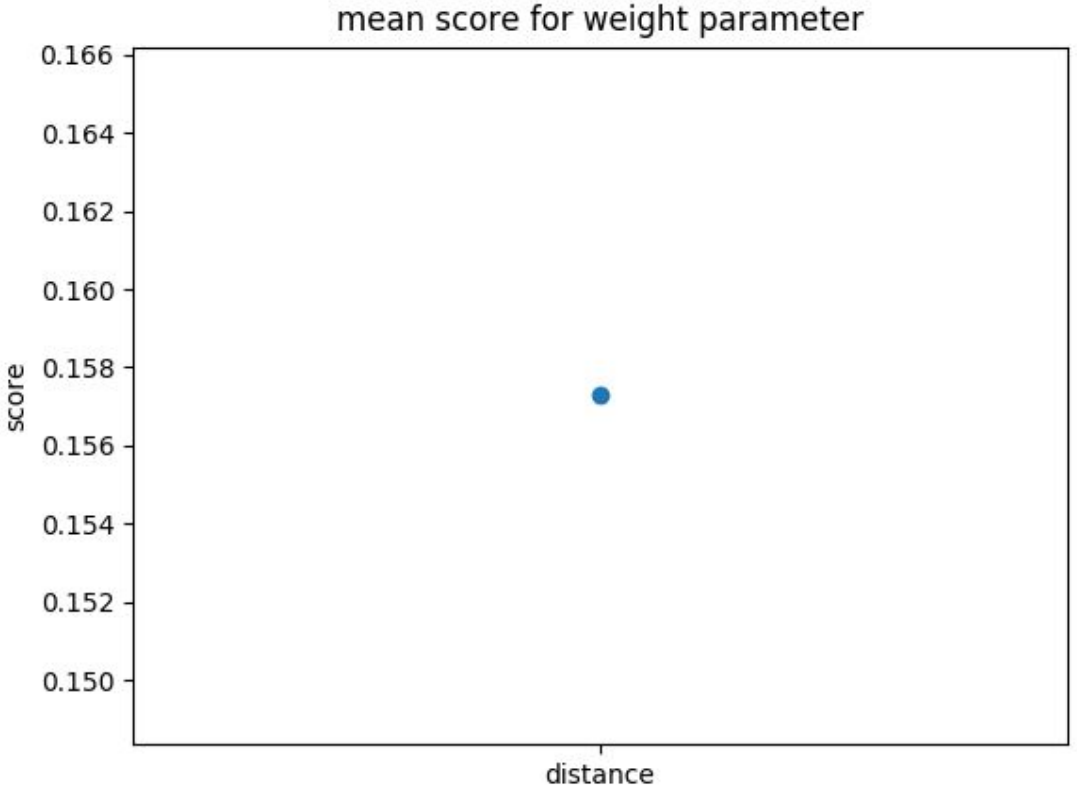

mean score for training ratio
score
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05


max score for training ratio
score
0.275
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05


max score for n_neighbours parameter
score
0.27
0.26
0.25
0.24
0.23
0.22
0.21
0.20
5
10
15
20
25
30


mean score for n_neighbours parameter
score
0.18
0.17
0.16
0.15
0.14
0.13
5
10
15
20
25
30


max score for weight parameter
0.268
0.266
0.264
0.262
0.260
0.258
0.256
0.254
0.252
score
distance


mean score for weight parameter
0.166
0.164
0.162
0.160
0.158
0.156
0.154
0.152
0.150
score
distance

Further research:
We also tested many other classifiers from scikit learn on this dataset, however as they did not performed well, we enclose them in the compressed file, but do not describe here.

2. Congressional Voting

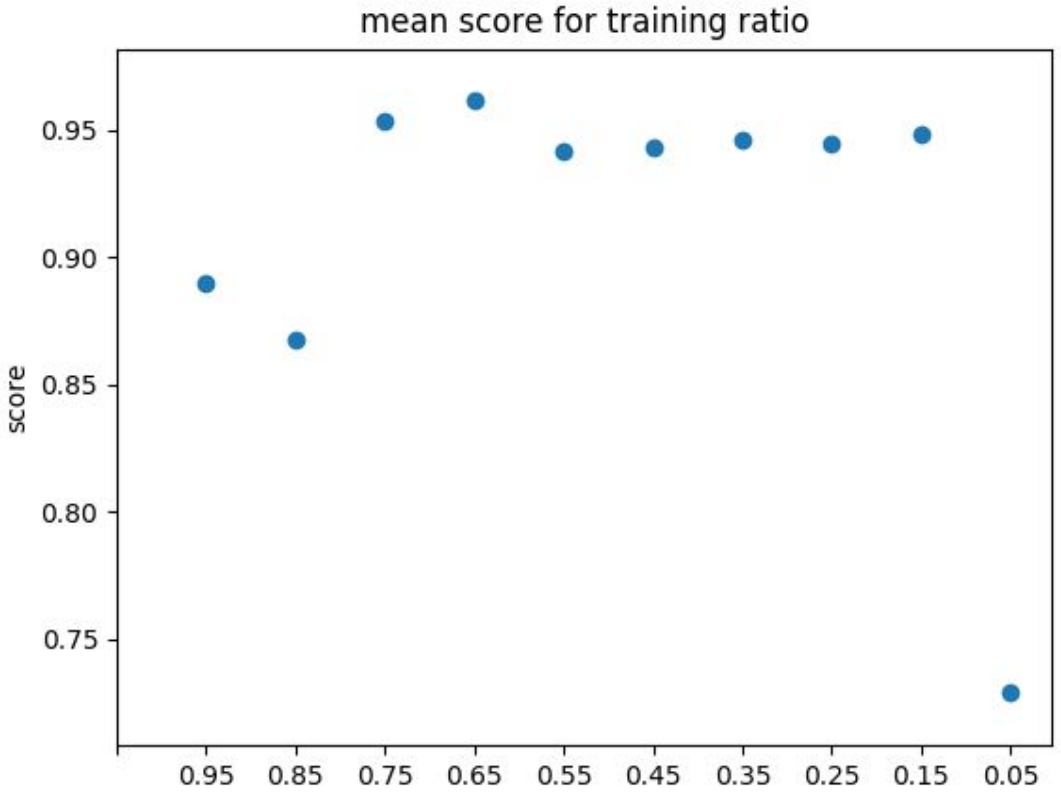
This dataset has 18 columns representing the voting of some congress representants, the first column is the ID of the representant, the second column is the class to predict (republican or democrat), the other 16 colums are having values of “y” and “n” which represent yes or not votes of each representant regarding the following topics (handicapped-infants, water-project-cost-sharing, adoption-of-the-budget-resolution, physician-fee-freeze, el-salvador-aid, religious-groups-in-schools, anti-satellite-test-ban, aid-to-nicaraguan-contras, mx-missile, immigration, synfuels-crporation-cutback, education-spending, superfund-right-to-sue, crime, duty-free-exports and export-administration-act-south-africa). There is a total of 211 missing values “unknown” in all the columns across the representants, there are also some representants with no missing values.

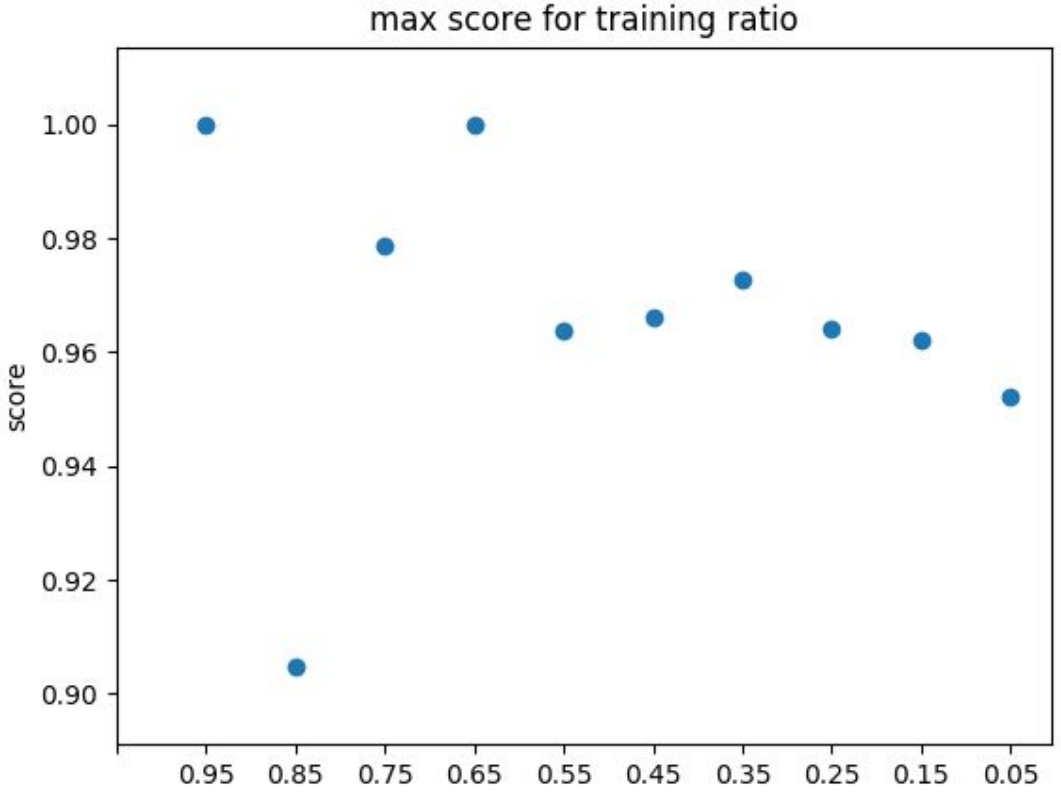
The label to predict is located in the second column “class” with values democrat or republican.

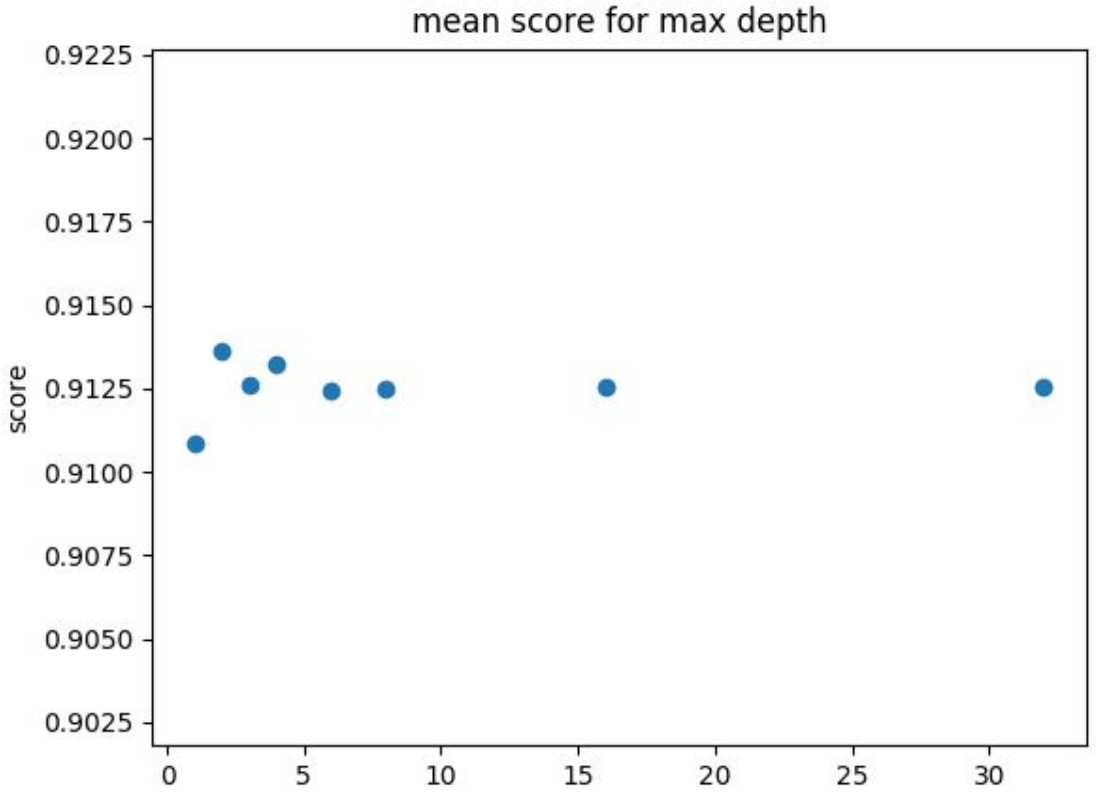
There were executed data transformation task replacing the values “y” and “n” with 1 and 0 respectively.

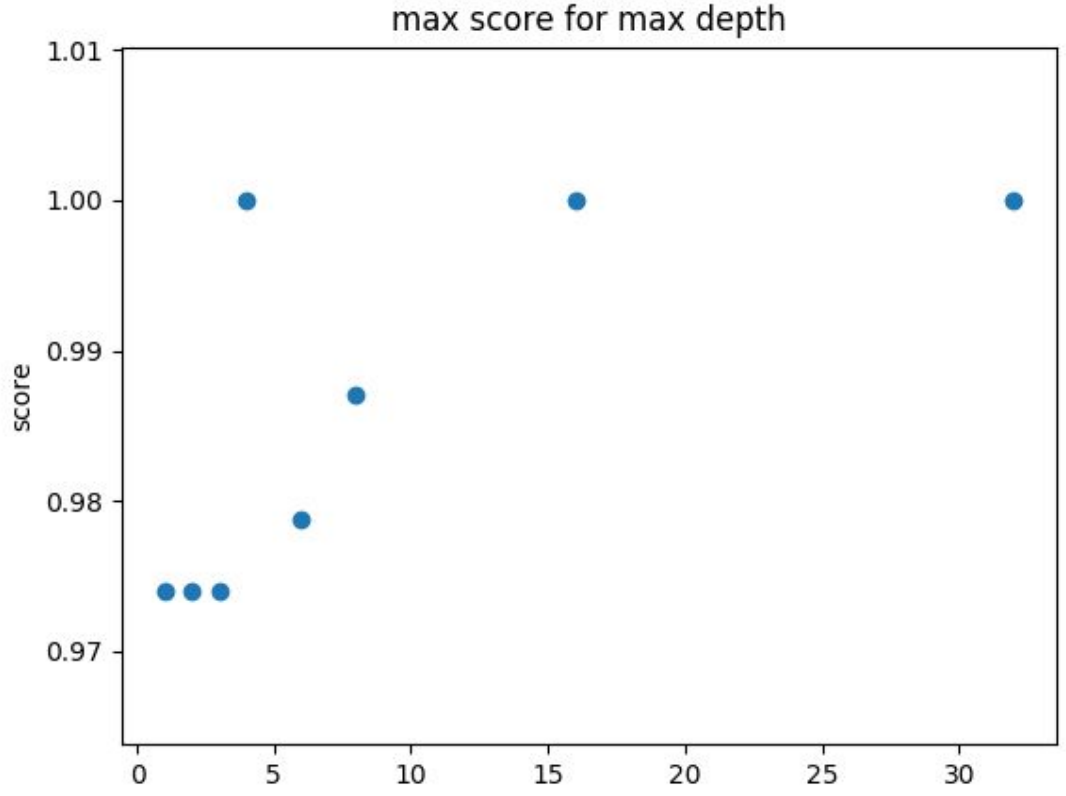
The missing values are handled using the mean of every column.

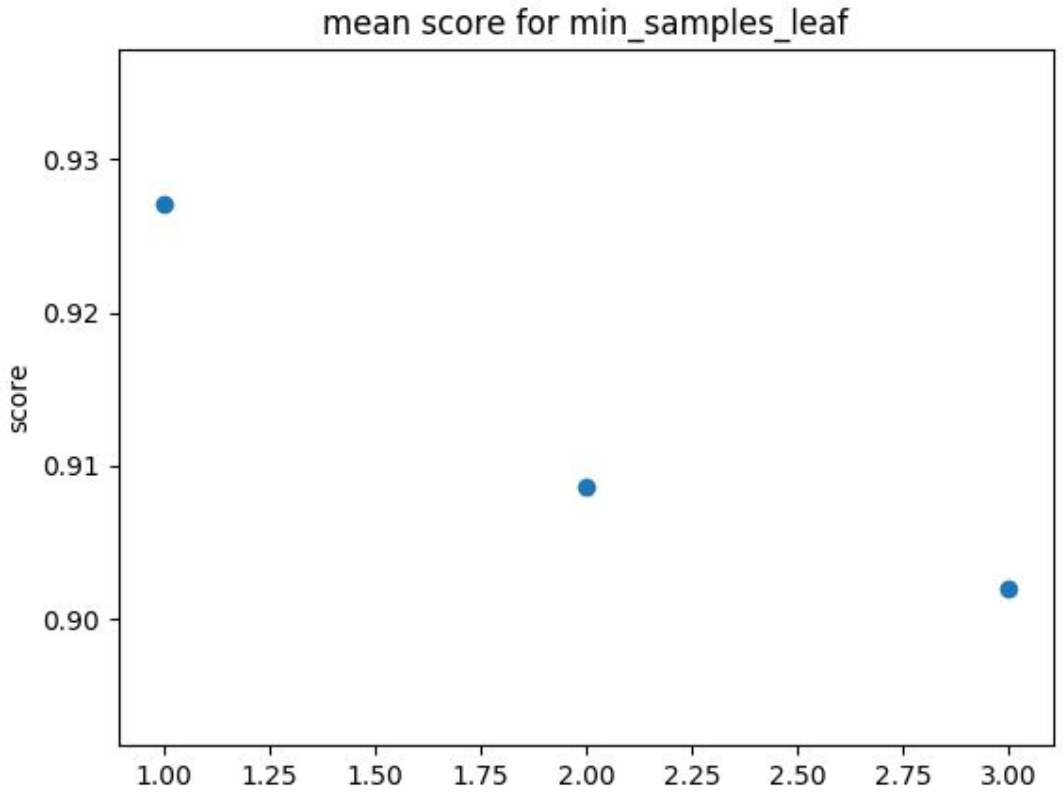
Experiments with the algorithms and parameters:
Random Forest:
This algorithm performed well for every not very low training:test ratio. The maximal depth of the tree has no significant influence on the results. The minimum number of samples for leaf and for branch, other than the default, only makes the score worse. For this dataset rather lower number of estimators performed better. The optimum was somewhere between 17 and 200. The computation time strongly depends on the number of estimators, and is almost independend from the the maximal tree depth with a slightly tendency to drop with higher depth. The following figures depict the results.

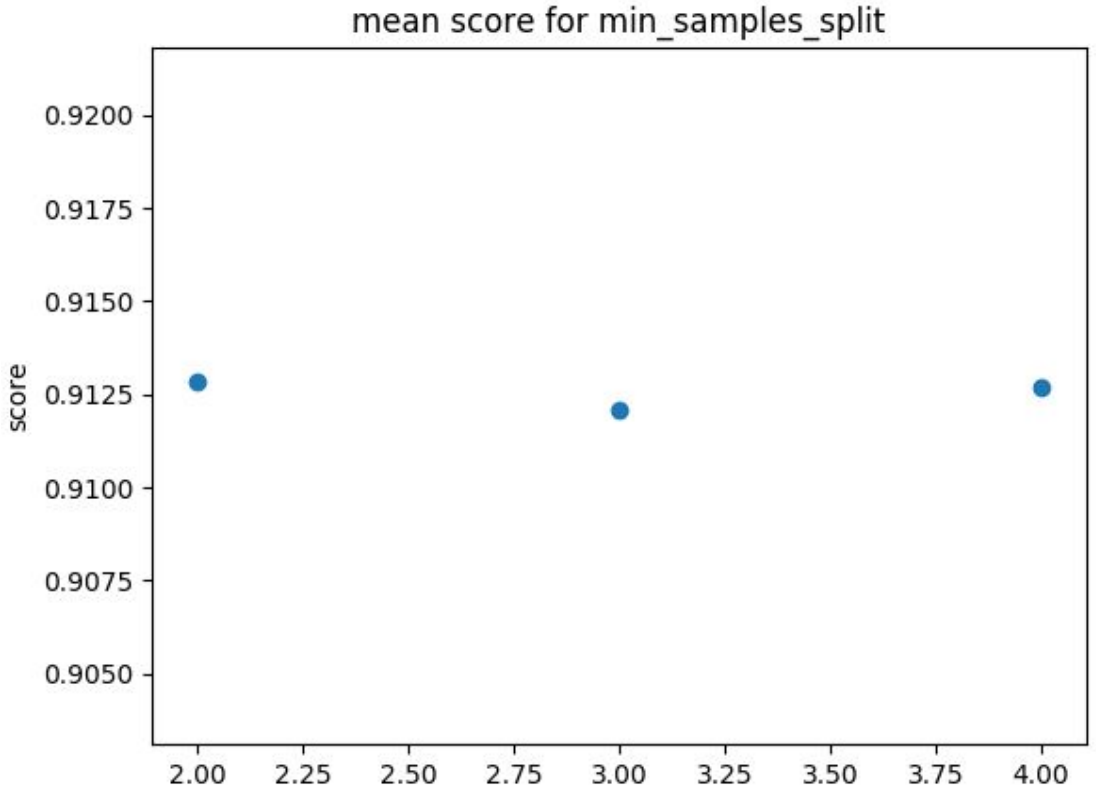

mean score for training ratio
score
0.95
0.90
0.85
0.80
0.75
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05

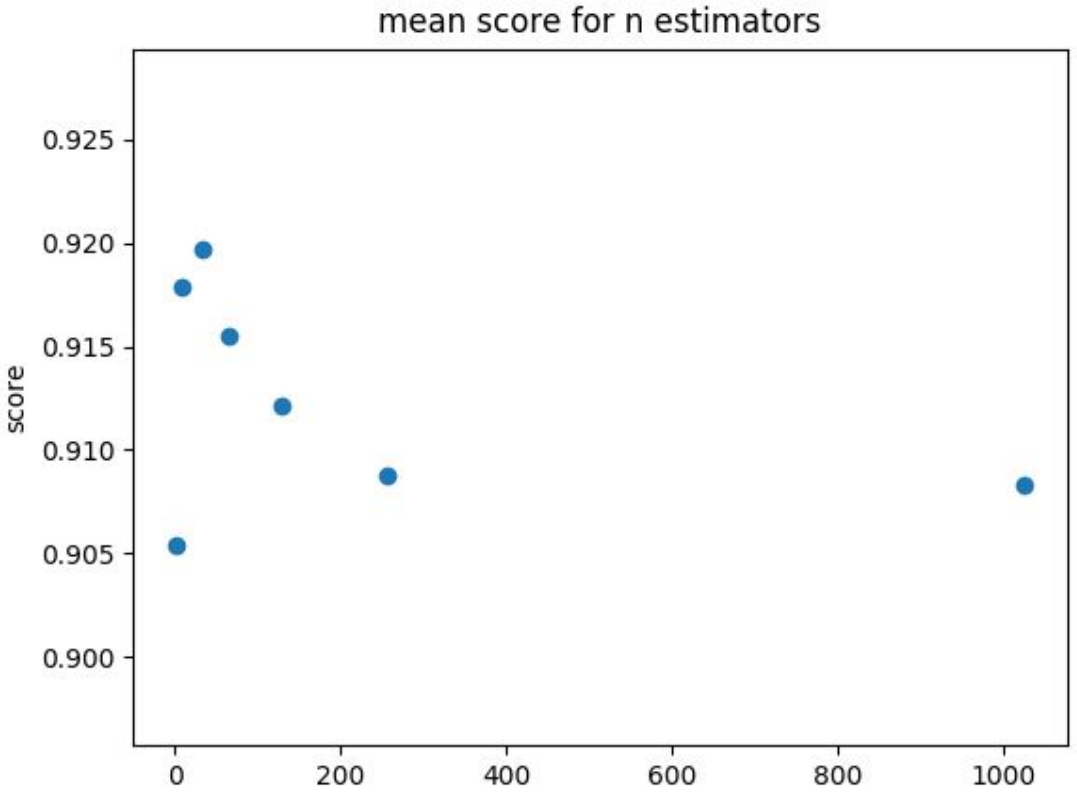

max score for training ratio
score
1.00
0.98
0.96
0.94
0.92
0.90
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05

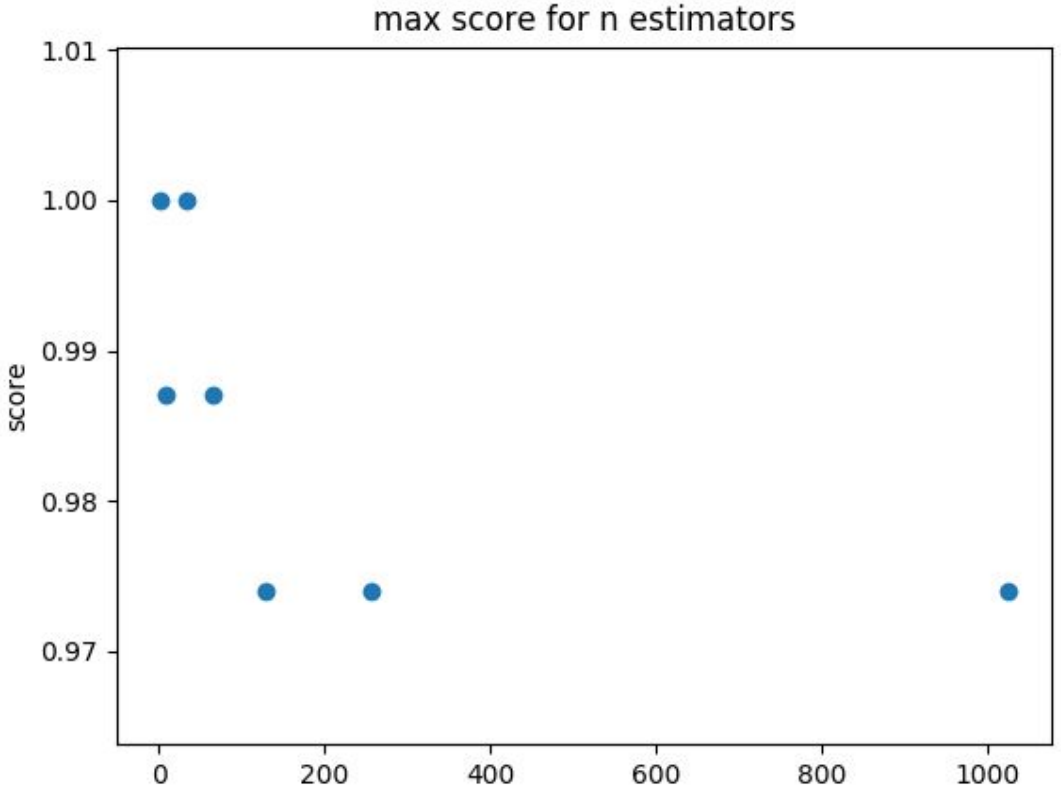

mean score for max depth
score
0.9225
0.9200
0.9175
0.9150
0.9125
0.9100
0.9075
0.9050
0.9025
0
5
10
15
20
25
30

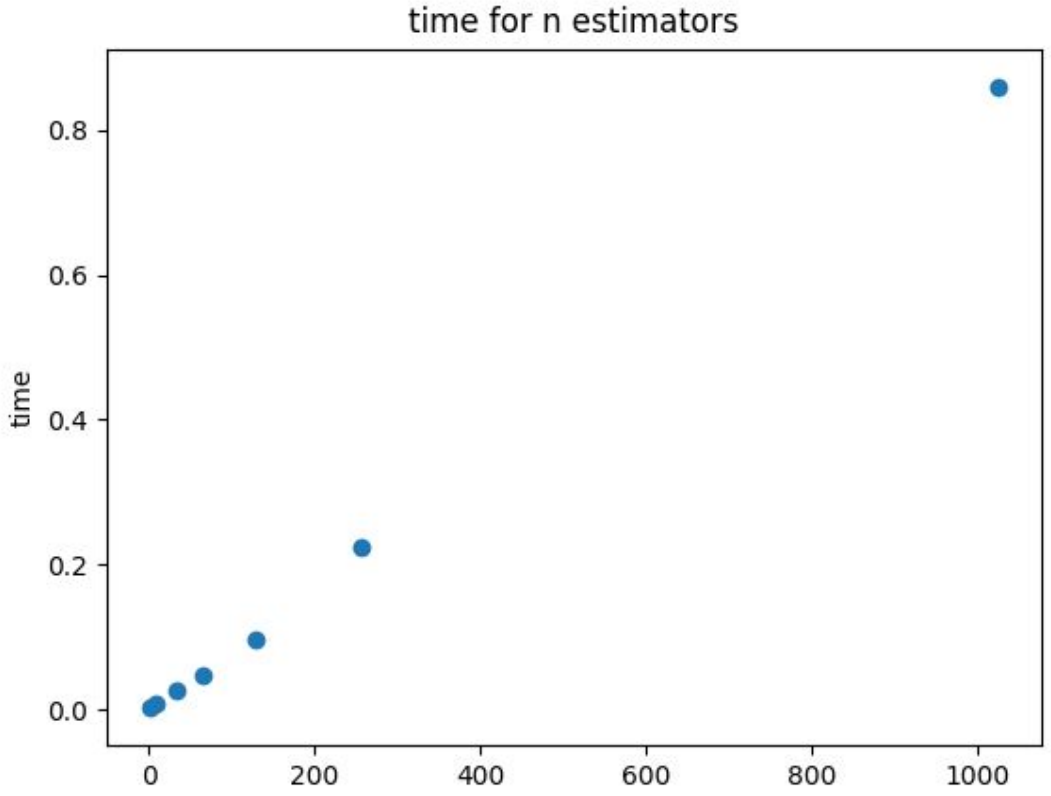

max score for max depth
score
1.01
1.00
0.99
0.98
0.97
0
5
10
15
20
25
30

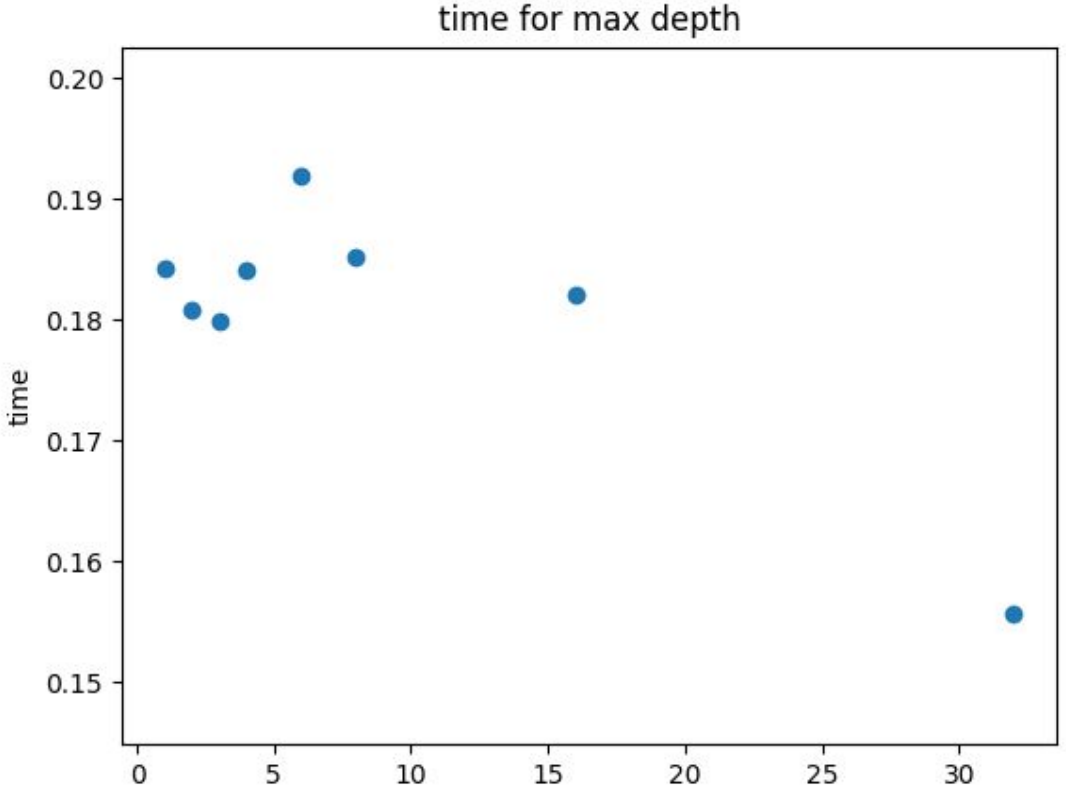

mean score for min_samples_leaf
score
0.93
0.92
0.91
0.90
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00


mean score for min_samples_split
score
0.9200
0.9175
0.9150
0.9125
0.9100
0.9075
0.9050
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00

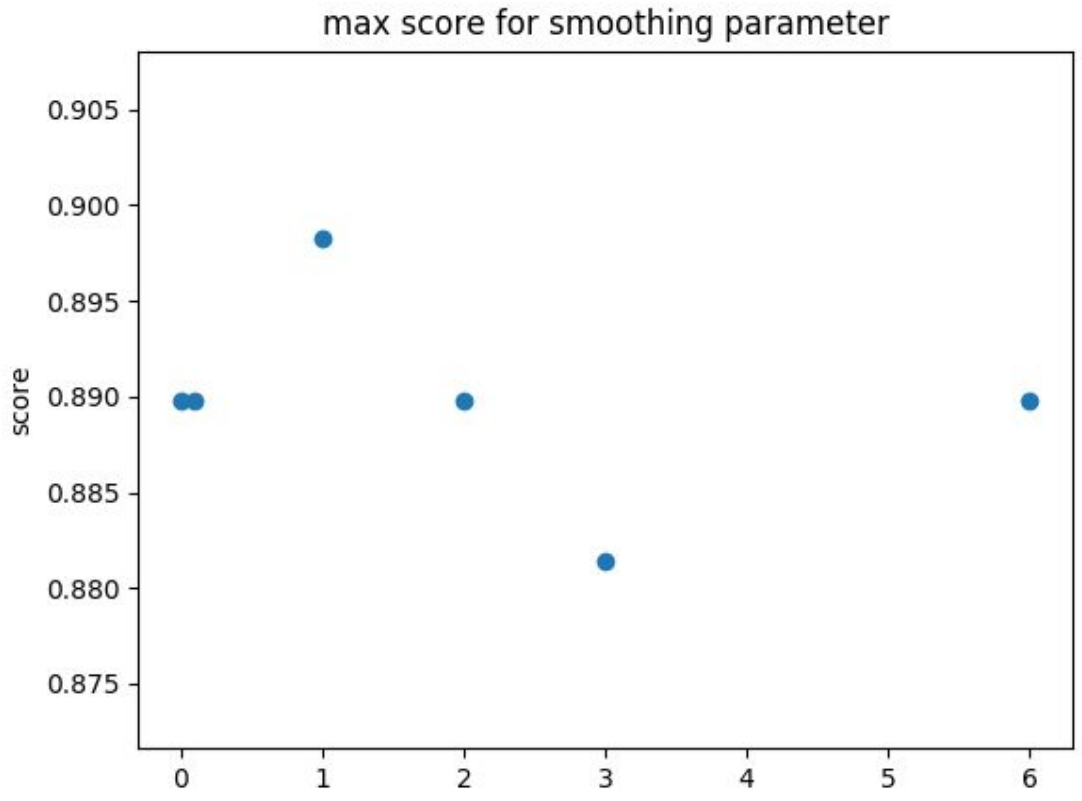

mean score for n estimators
score
0.925
0.920
0.915
0.910
0.905
0.900
0
200
400
600
800
1000

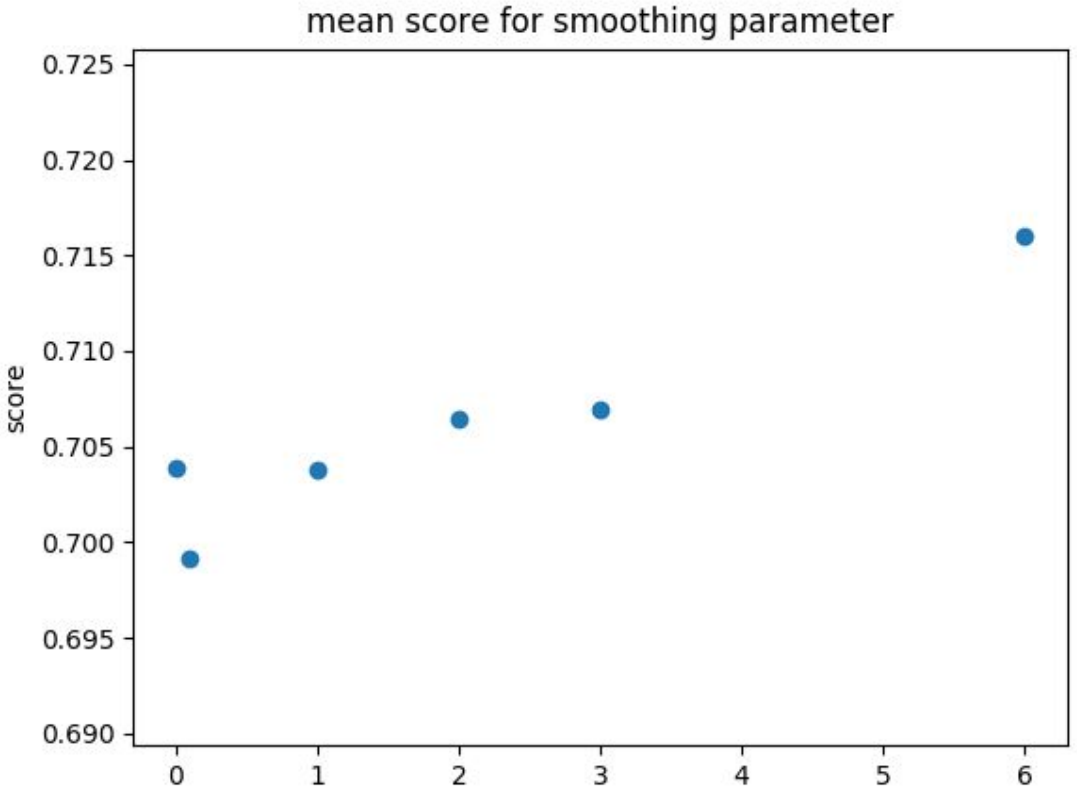

max score for n estimators
score
1.01
1.00
0.99
0.98
0.97
0
200
400
600
800
1000

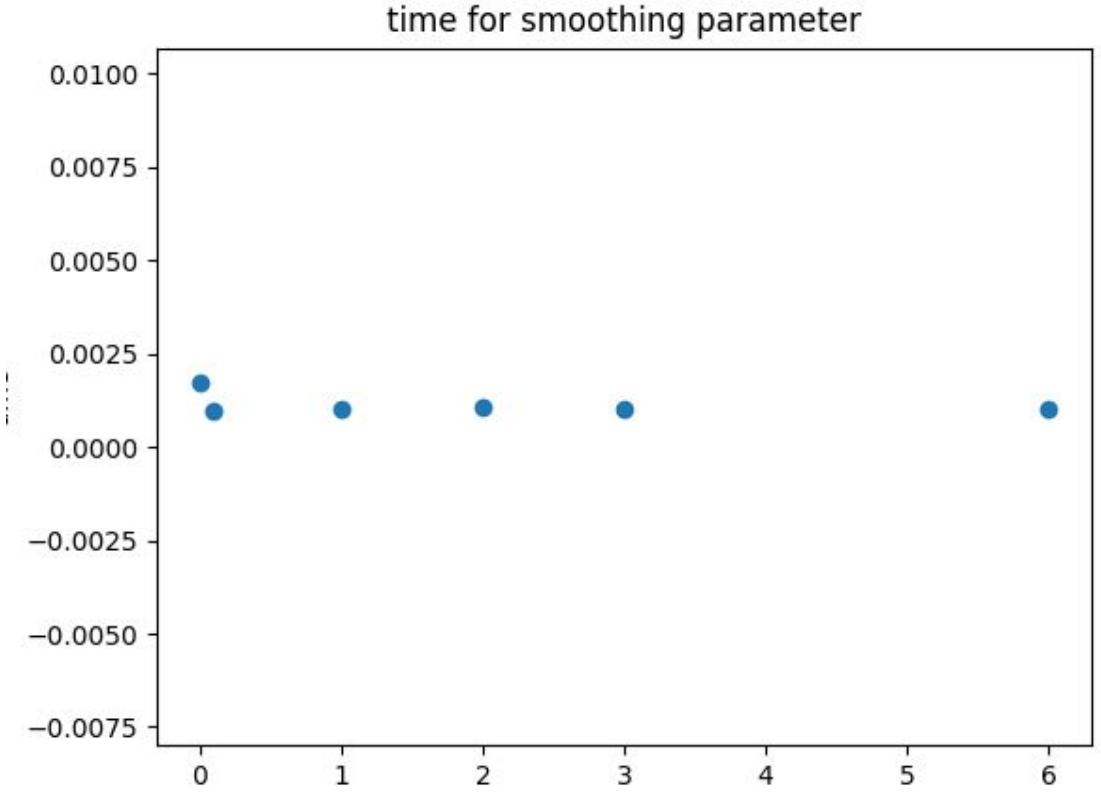

time for n estimators
time
0.0
0.2
0.4
0.6
0.8
0
200
400
600
800
1000

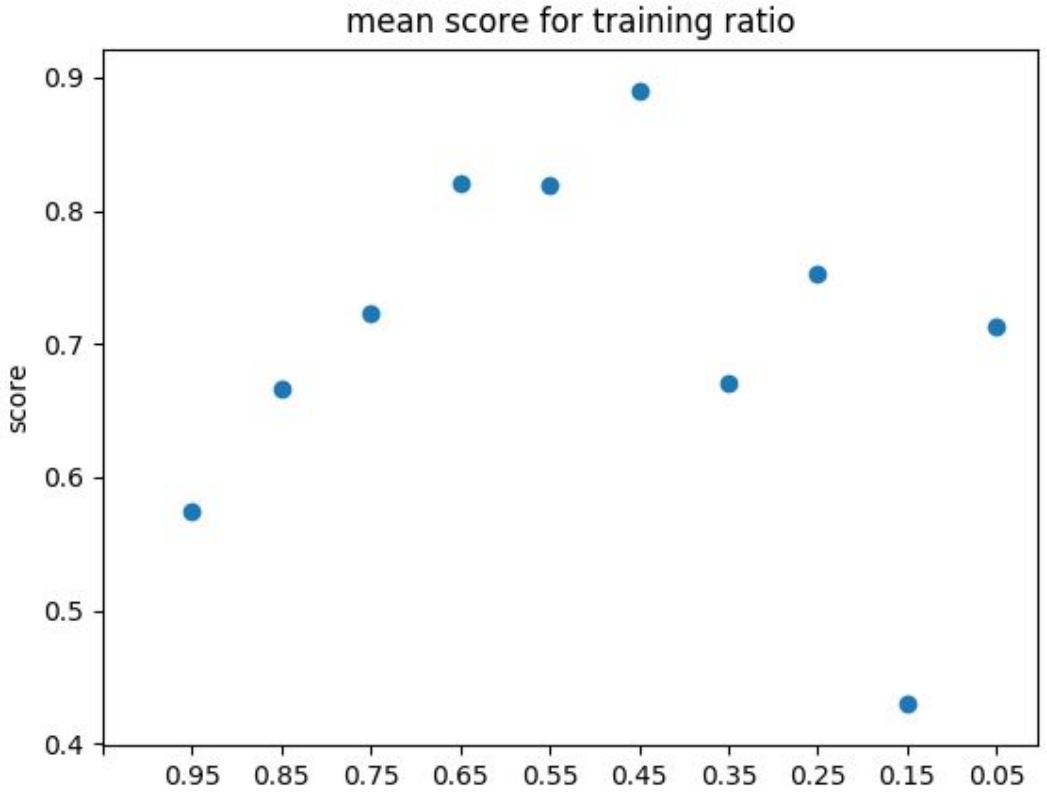

time for max depth
time
0.15
0.16
0.17
0.18
0.19
0.20
0
5
10
15
20
25
30

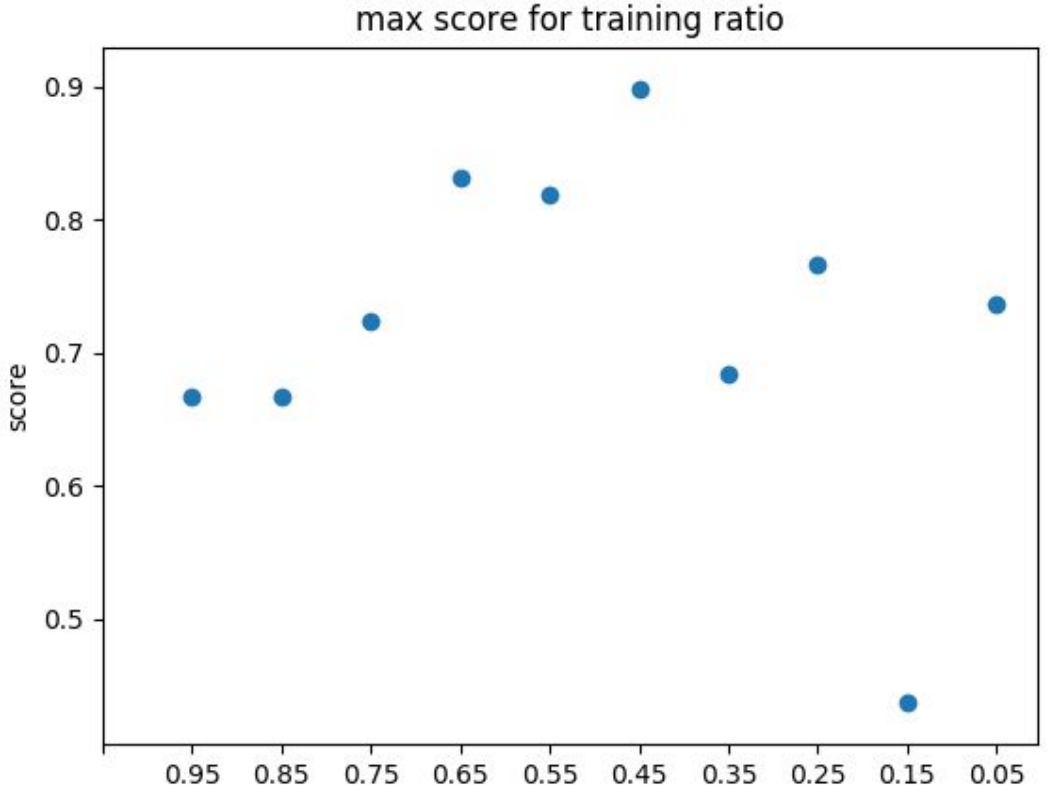
Naive Bayes:
For this dataset the default smoothing paramter had the best max value, however the mean seems to increase for increasing smoothing parameter. There are no significant differences in the computation time. The Naive Bayes performs best for the middle range of the training:test ratio. The described behaviour can be observed in the following figures.




mean score for smoothing parameter
score
0.725
0.720
0.715
0.710
0.705
0.700
0.695
0.690
0
1
2
3
4
5
6


time for smoothing parameter
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
−0.0075
0
1
2
3
4
5
6


mean score for training ratio
score
0.9
0.8
0.7
0.6
0.5
0.4
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05


max score for training ratio
score
0.9
0.8
0.7
0.6
0.5
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05

K Nearest Neighbours

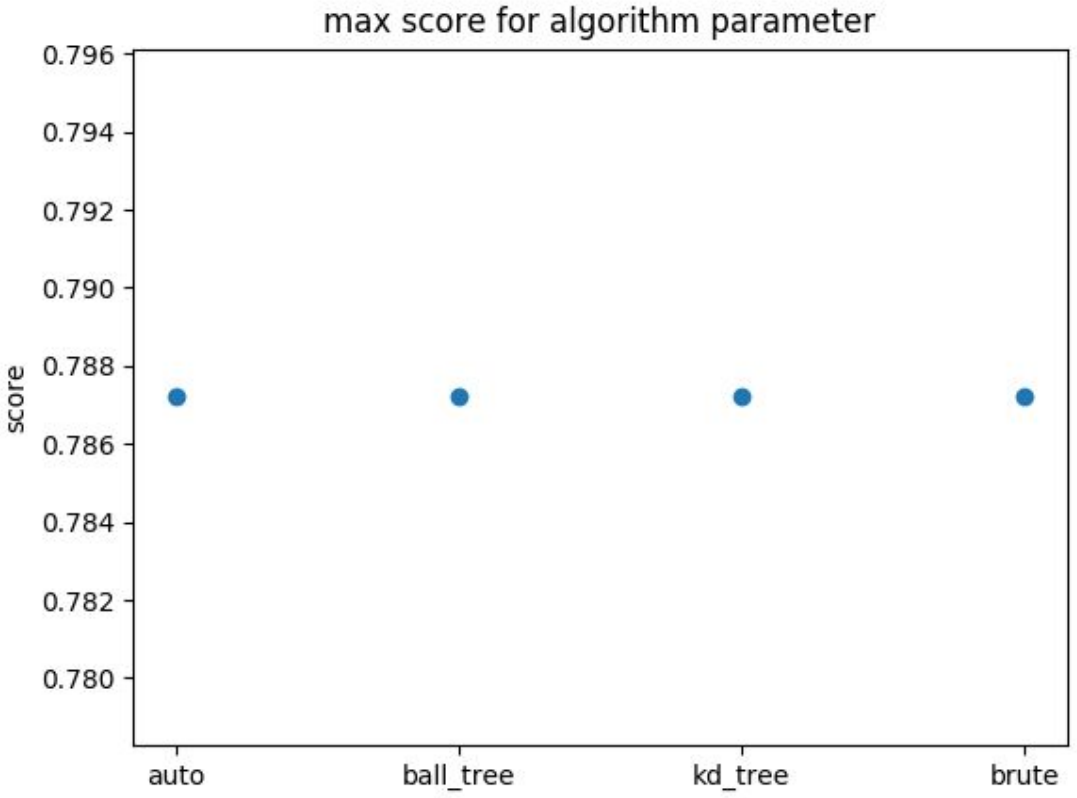
With this algorithm, there are no changes in the max and mean score of the voting algorithm parameter (autom ball_tree, kd_tree, brute).

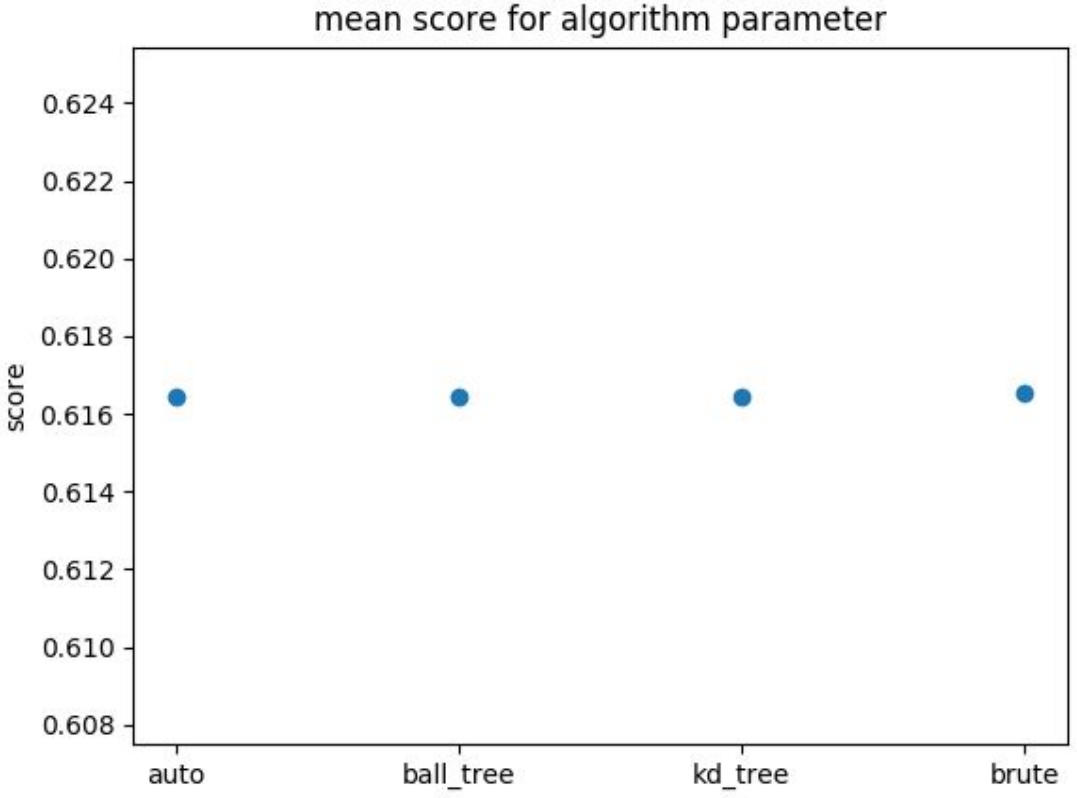
Regarding the training ratio, the behave of the scores where sparse. Nevertheless, the highest score reached was with the percent of 25% samples for training.

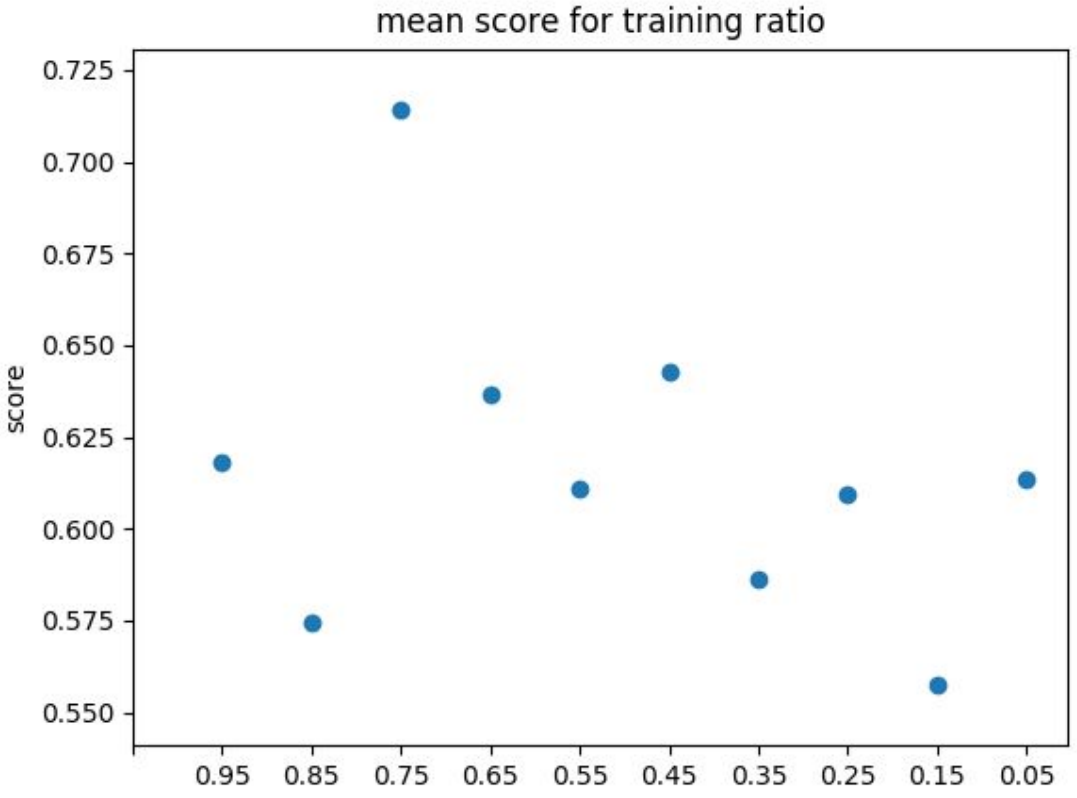
In contrast to the number of neighbours, the mean and max score changed differently. The max score was reached using 8 neighbours and taking into account the mean, the best score was reached using 10 neighbours.

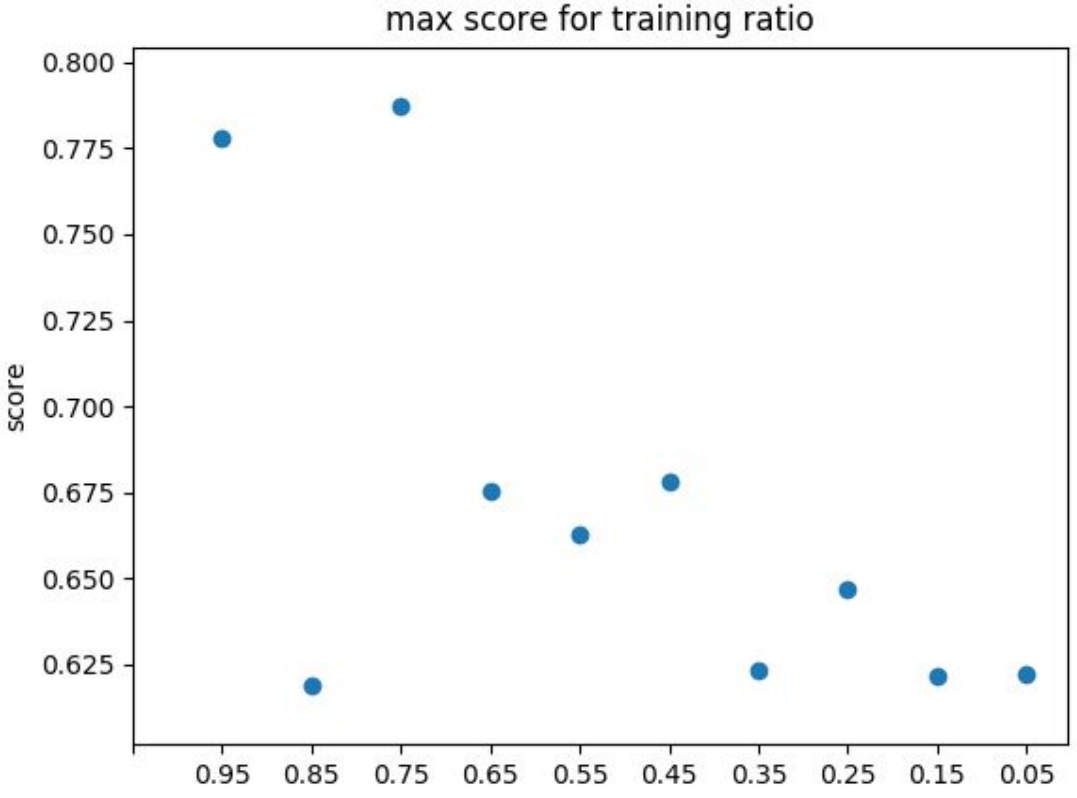
Taking into account the distance of the nodes of the neighbours generated also a better score in the max score parameter but not in the mean parameter, nevertheless, this is the parameter where the score was optimized.

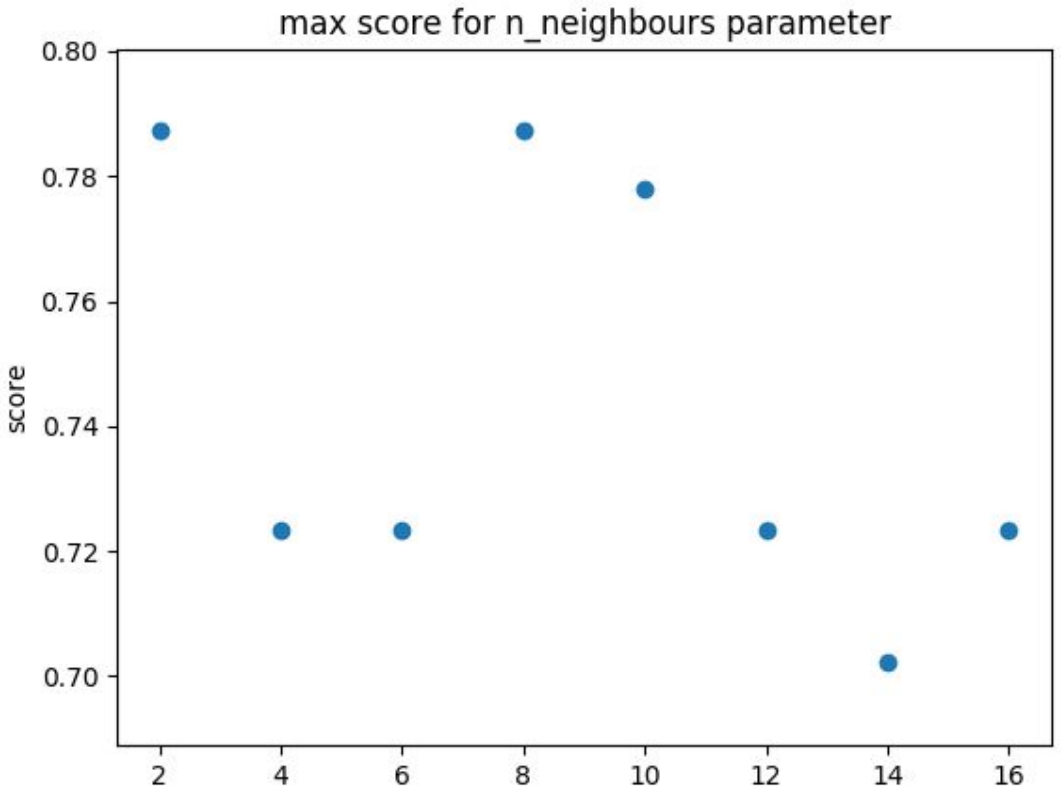
The number of neighbours did not affect the execution time.

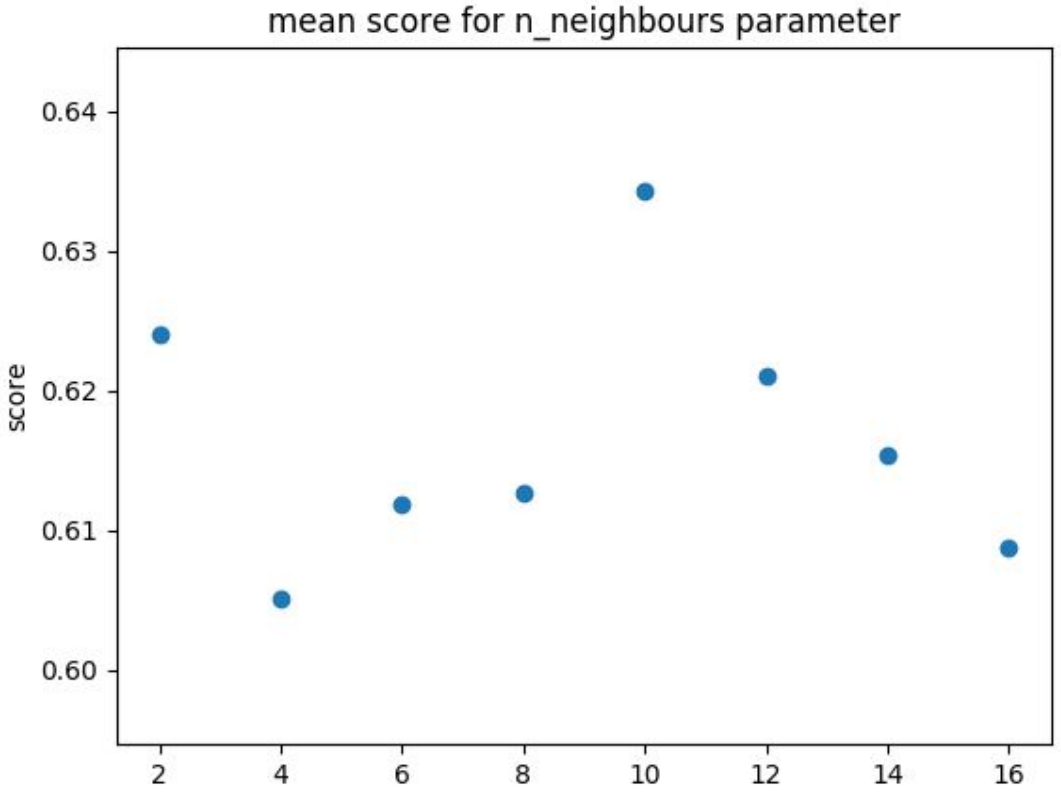

max score for algorithm parameter
score
0.796
0.794
0.792
0.790
0.788
0.786
0.784
0.782
0.780
auto
ball_tree
kd_tree
brute

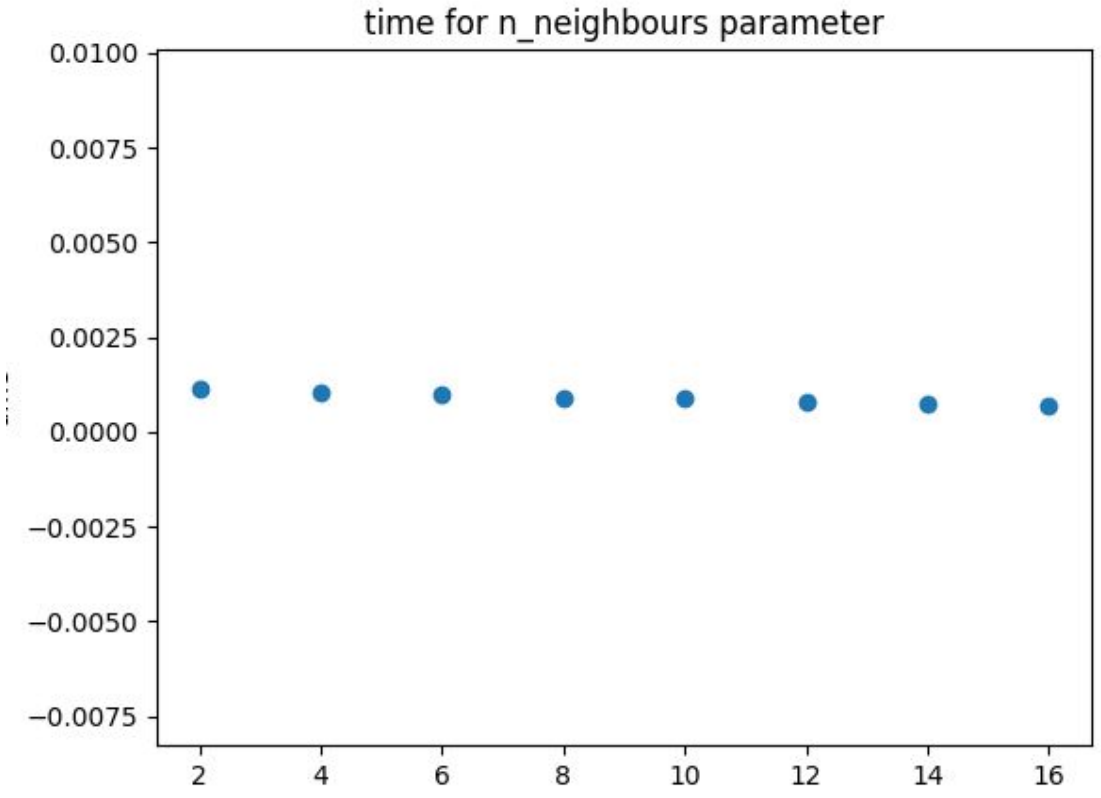

mean score for algorithm parameter
score
0.624
0.622
0.620
0.618
0.616
0.614
0.612
0.610
0.608
auto
ball_tree
kd_tree
brute

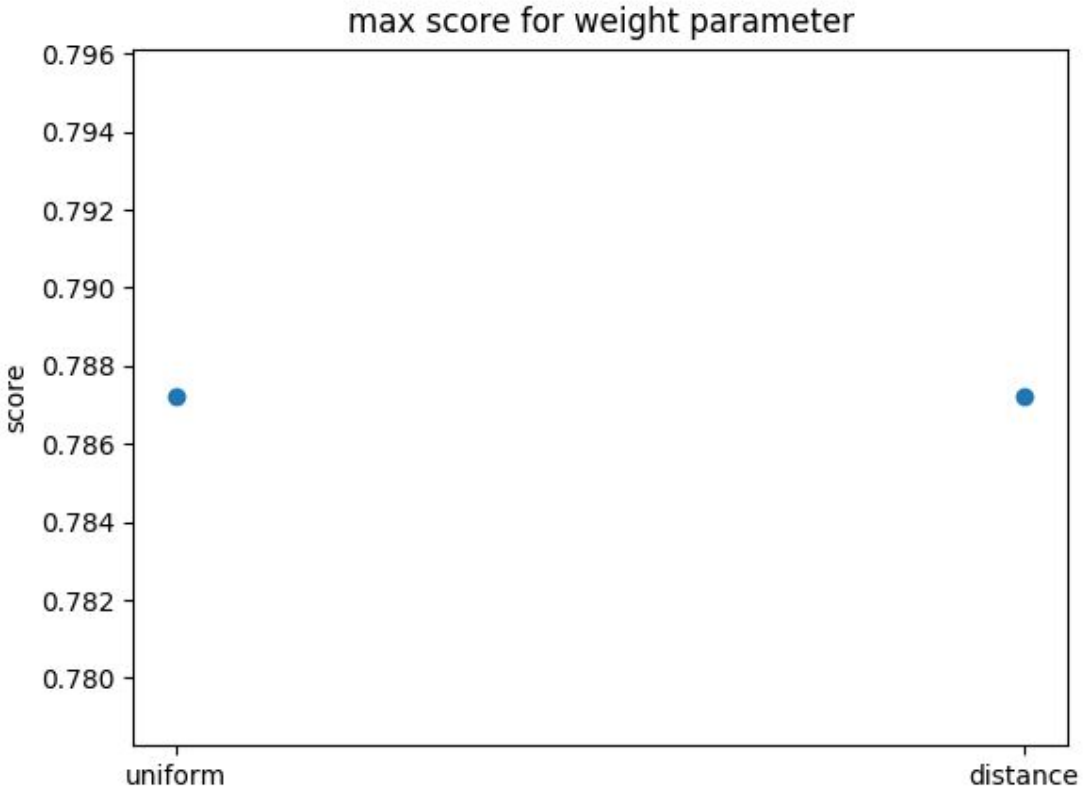

mean score for training ratio
score
0.725
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05

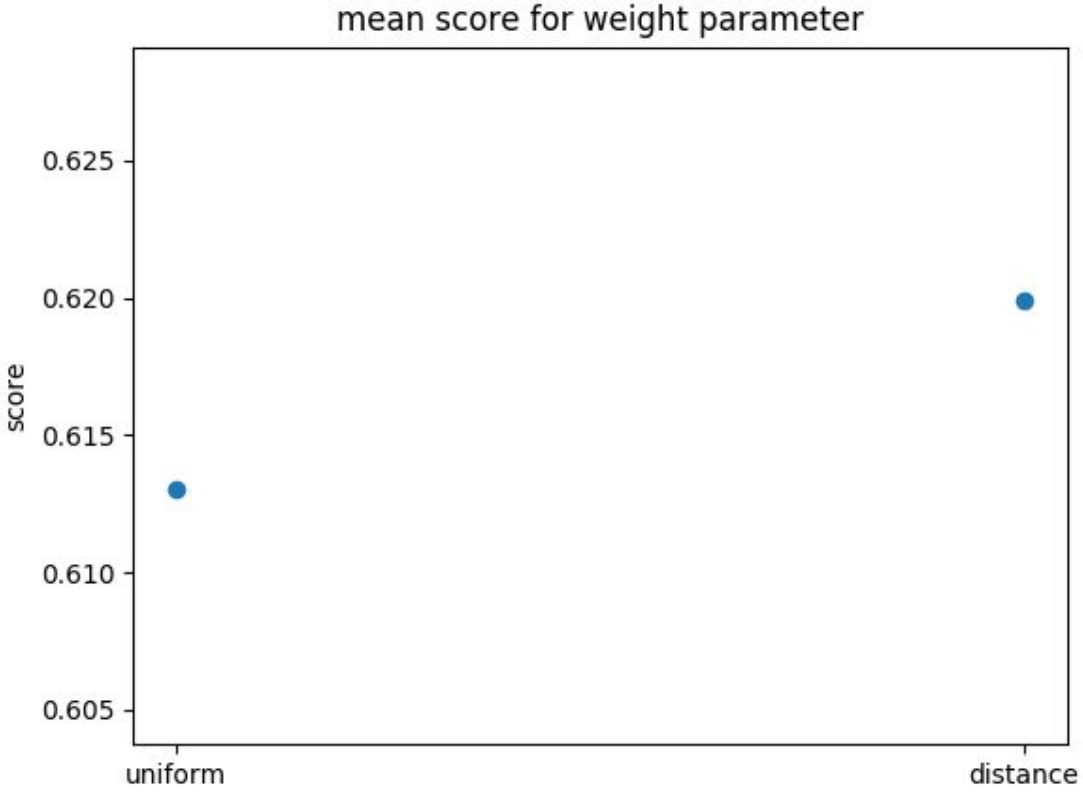

max score for training ratio
score
0.800
0.775
0.750
0.725
0.700
0.675
0.650
0.625
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05


max score for n_neighbours parameter
score
0.80
0.78
0.76
0.74
0.72
0.70
2
4
6
8
10
12
14
16


mean score for n_neighbours parameter
score
0.64
0.63
0.62
0.61
0.60
2
4
6
8
10
12
14
16


time for n_neighbours parameter
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
−0.0075
2
4
6
8
10
12
14
16


max score for weight parameter
0.796
0.794
0.792
0.790
0.788
0.786
0.784
0.782
0.780
score
uniform
distance


mean score for weight parameter
0.625
0.620
0.615
0.610
0.605
score
uniform
distance

3. Cardiotocography Data Set

This dataset consists of measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms classified by expert obstetricians.
2126 fetal cardiotocograms (CTGs) were processed and the respective diagnostic features measured. The features are classified in 10 classes of heart behaviour by 3 expert obstetricians and a consensus classification label assigned to each of them. The dataset classifies the data with two different kind of labels: One with respect to a morphologic pattern (A, B, C, SH, AD, DE, LD, FS, SUSP) and the other label represents the fetal state (NSP, where Normal=1, Suspect=2 and Pathologic=3). Therefore, the dataset can be used either for 10-class or 3-class analysis.

Quantitative analysis of the characteristics:

A. 2126 samples with 21 attributes and no missing values.
B. Types of attributes:
   There are 21 features attributes with numeric values and the other 10 classes attributes are nominal-valued.
   C. Characteristics:

The feature variables are represented by the following:
21 numeric attributes. The numbers in the square bracket represent the minimum and the maximum value of the corresponding attribute.

• LB: FHR baseline (beats per minute) [106, 160]
• AC: n of accelerations per second [0, 26]
• FM: n of fetal movements per second [0, 564]
• UC: n of uterine contractions per second [0, 23]
• DL: n of light decelerations per second [0, 16]
• DS: n of severe decelerations per second [0, 1]
• DP: n of prolonged decelerations per second [0, 4]
• ASTV: percentage of time with abnormal short term variability [12, 87]
• MSTV: mean value of short term variability [0.2, 7,0]
• ALTV: percentage of time with abnormal long term variability [0, 91]
• MLTV: mean value of long term variability [0.0, 50.7]
• Width: width of FHR histogram [3, 180]
• Min: minimum of FHR histogram [50, 159]
• Max: Maximum of FHR histogram [122, 238]

• Nmax: n of histogram peaks [0, 18]
• Nzeros: n of histogram zeros [0, 10]
• Mode: histogram mode [60, 187]
• Mean: histogram mean [73, 182]
• Median: histogram median [77, 186]
• Variance: histogram variance [0, 269]
• Tendency: histogram tendency [-1, 1]

The 10-class nominal labels represent the following morphologic patterns:

• 1. A: calm sleep
• 2. B: REM sleep
• 3. C: calm vigilance
• 4. D: active vigilance
• 5. SH: shift pattern (A or Susp with shifts)
• 6. AD: accelerative/decelerative pattern (stress situation)
• 7. DE: decelerative pattern (vagal stimulation)
• 8. LD: largely decelerative pattern
• 9. FS: flat-sinusoidal pattern (pathological state)
• 10. SUSP: suspect pattern
• CLASS: class code for the classes above (from 1 to 10 respectively). This is the label used for the analysis.

The NSP label that represents the fetal state is not used for the exercise, although it could be a great opportunity to extend the analysis.

D. Further Analysis

We generated box plots to analyze the behaviour of the feature variables. Those graphics show boxes that represent the amount of data in between the first quartile to the third quartile, the horizontal line goes through the box at the median value, the whiskers go from each quartile to the minimum or maximum and the dots represent the outliers that have values outside 1.5 times the interquartile range above the upper quartile and below the lower quartile (see Appendices).

We noticed that 11 of 21 features have the min value of zero, we tried to find out how much percentage of zero value these features have. Some of them have a high number of zero value that led us to the question of how both classifiers we have chosen would deal with this situation.

Moreover, we pointed outliers in all the feature variables for all the classes, it is necessary to analyse where the data comes from and what are we classifying, taking

in consideration that it belongs to human beings heart's behaviours and the classifications are identifying morphologic patterns so we hypothesised that those outliers will contribute to predict the classes in this dataset. We found previous analysis with the same dataset[1] that considered the process to delete the outliers from the original dataset and measure the f1-score, it scored lower values for the prediction model without the outliers than with them. We will continue in the next exercise with the analysis using the original dataset.

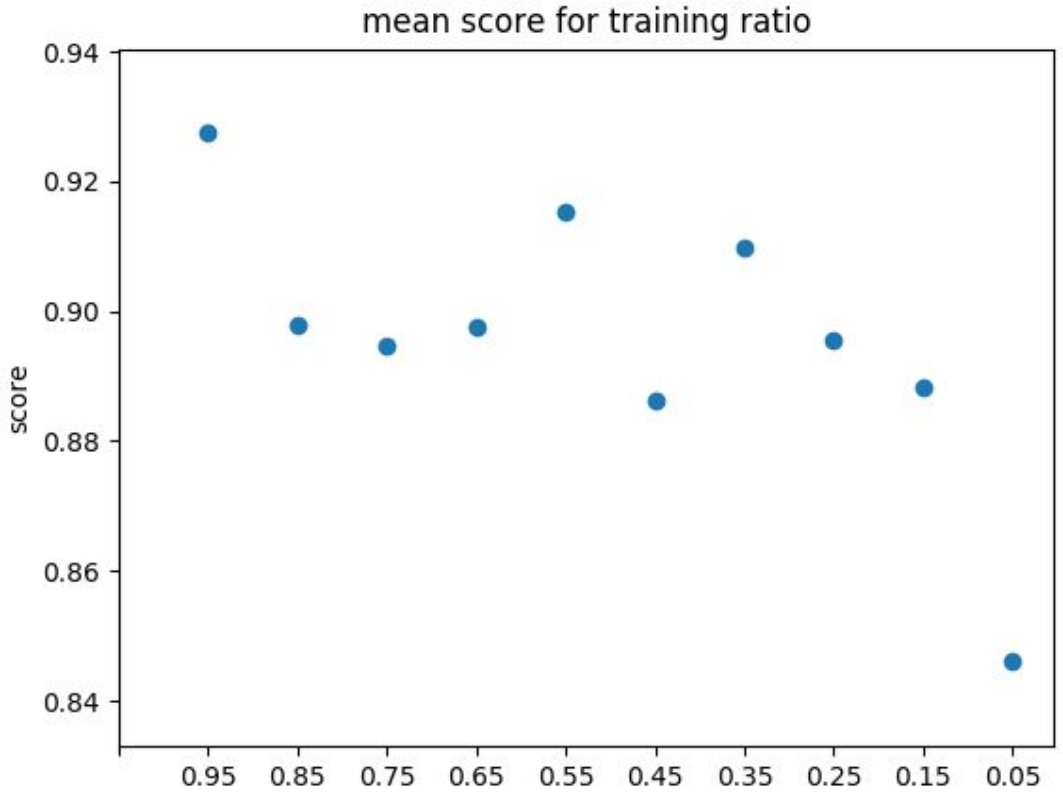
Experiments with the algorithms:

Scaling:

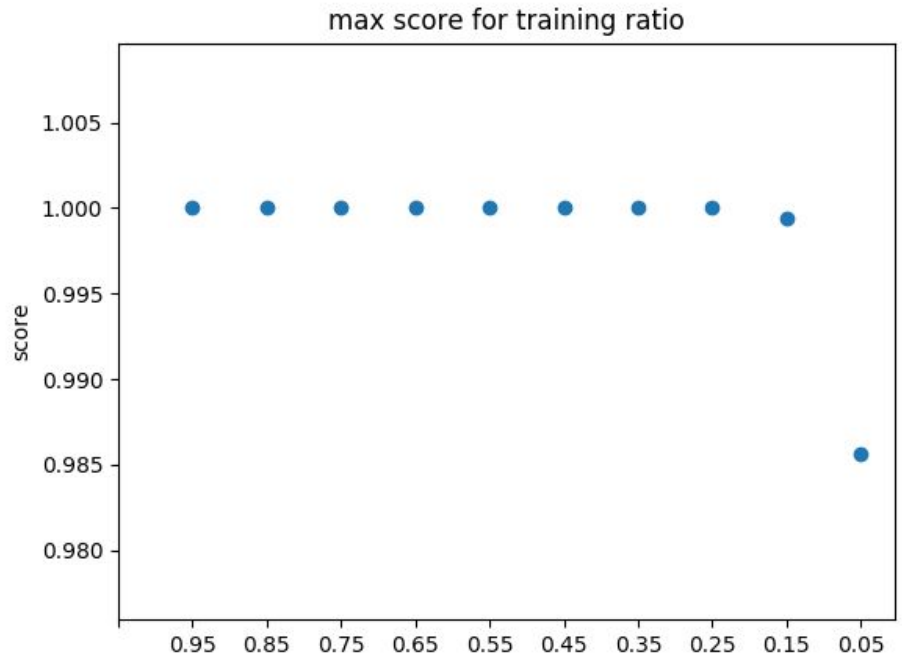
We used the MinMaxScaler as the sklearn Naive Bayes does not accept negative values

Random Forest

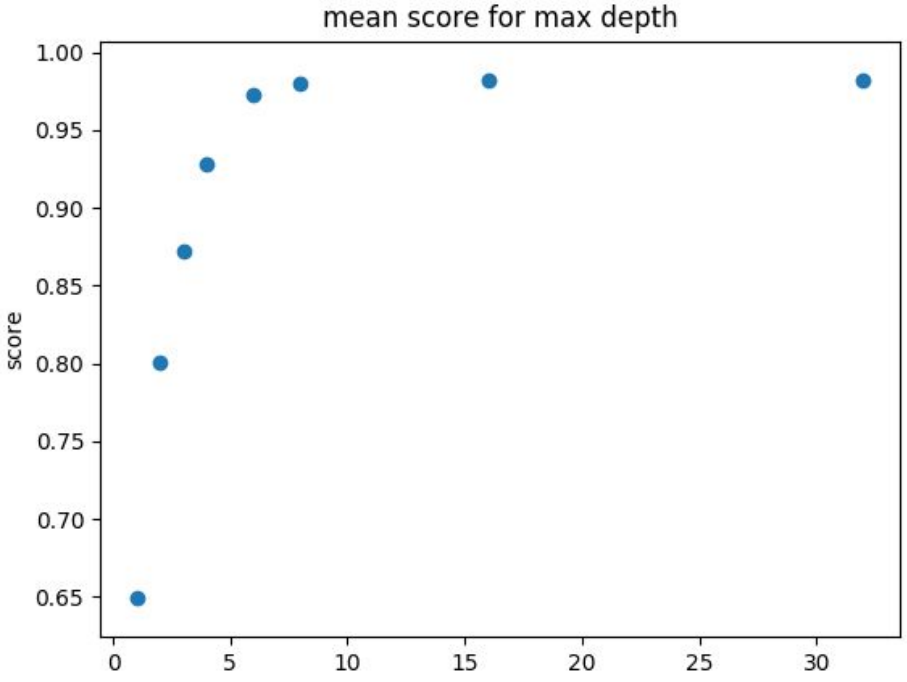
The following figures present the results of our experiments with the parameters. The figures present the mean value for a parameter and the maximal score achieved for a parameter. In the first figure we can see a tendency that for higher training to test ratio, the score is higher.

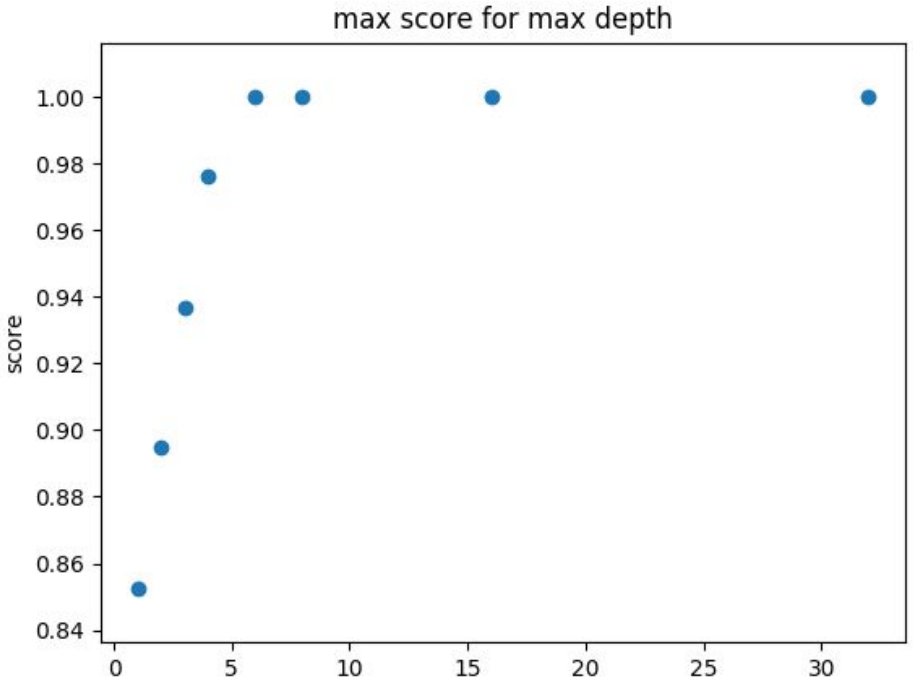


However the maximal score is really dropping only for very low ratios.

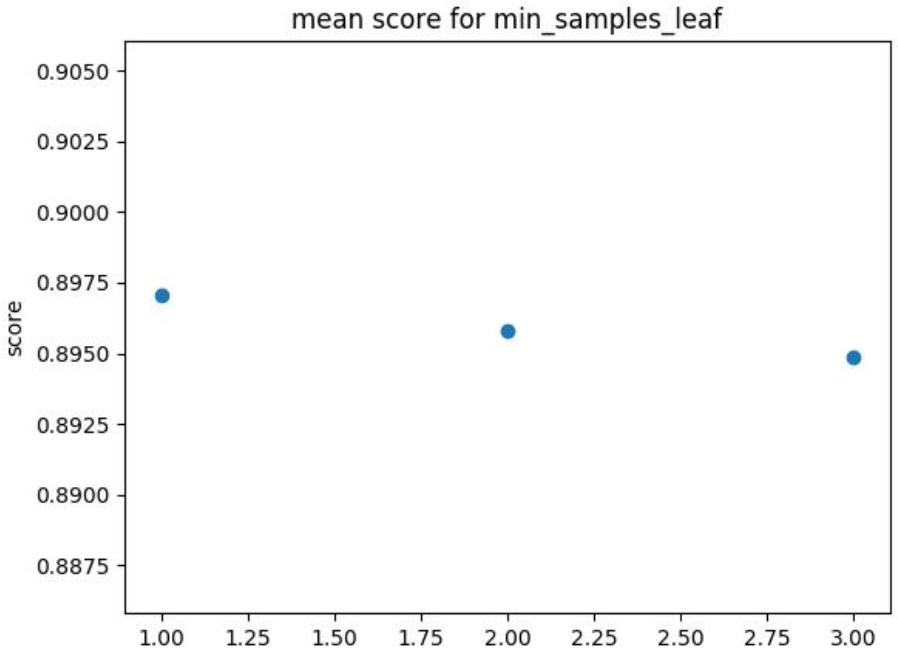


The following two figures show a great importance of the tree depth on this dataset. The deeper the tree the better the score.



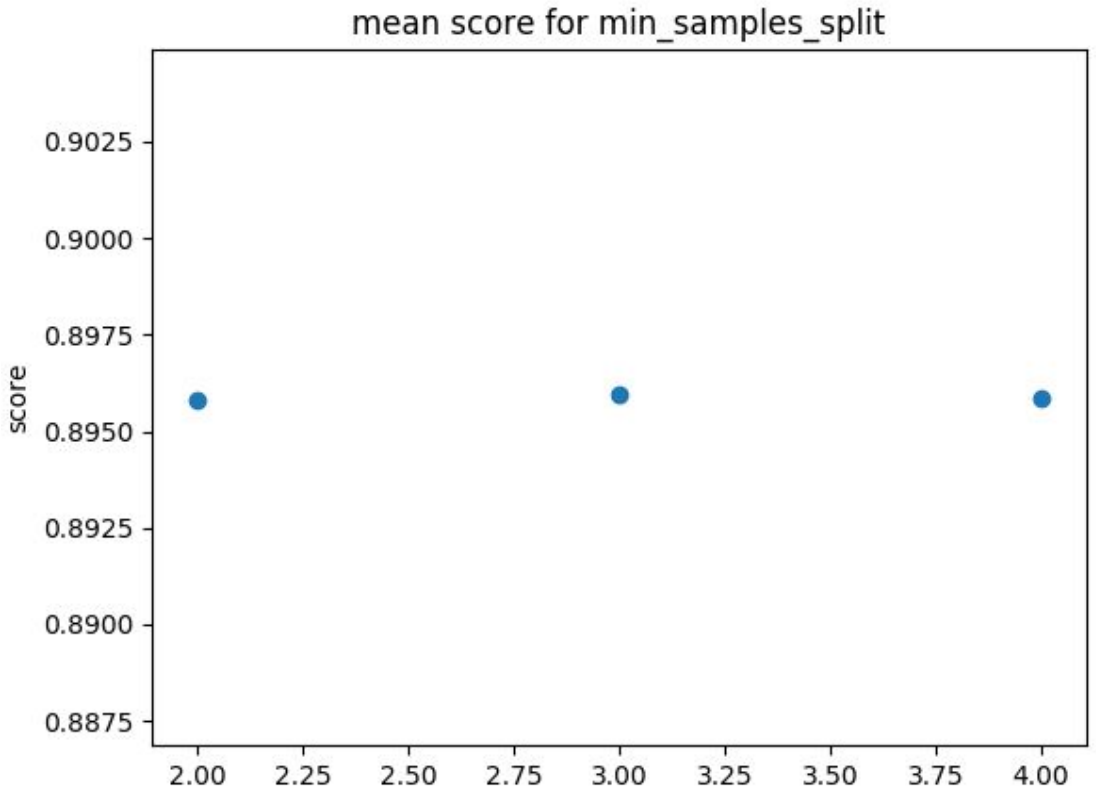


The higher minimal number of samples in a leaf generally reduces the learning



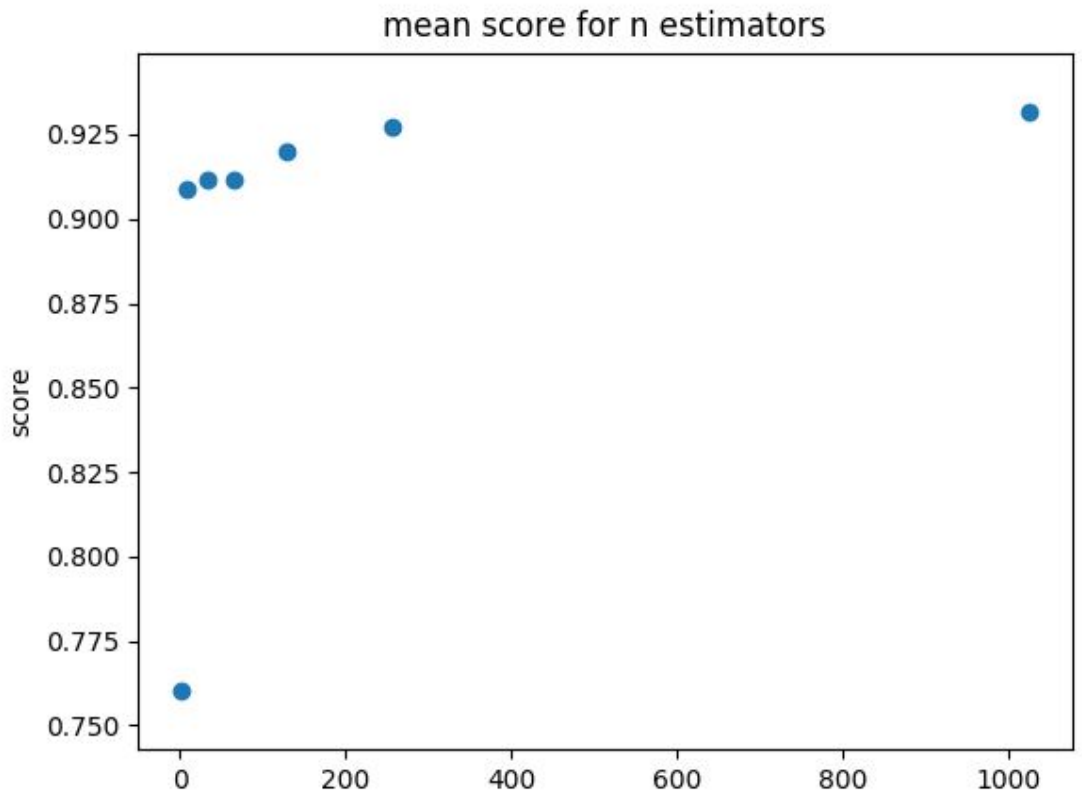
score.

The minimal number of samples for a split does not significantly influence the results.

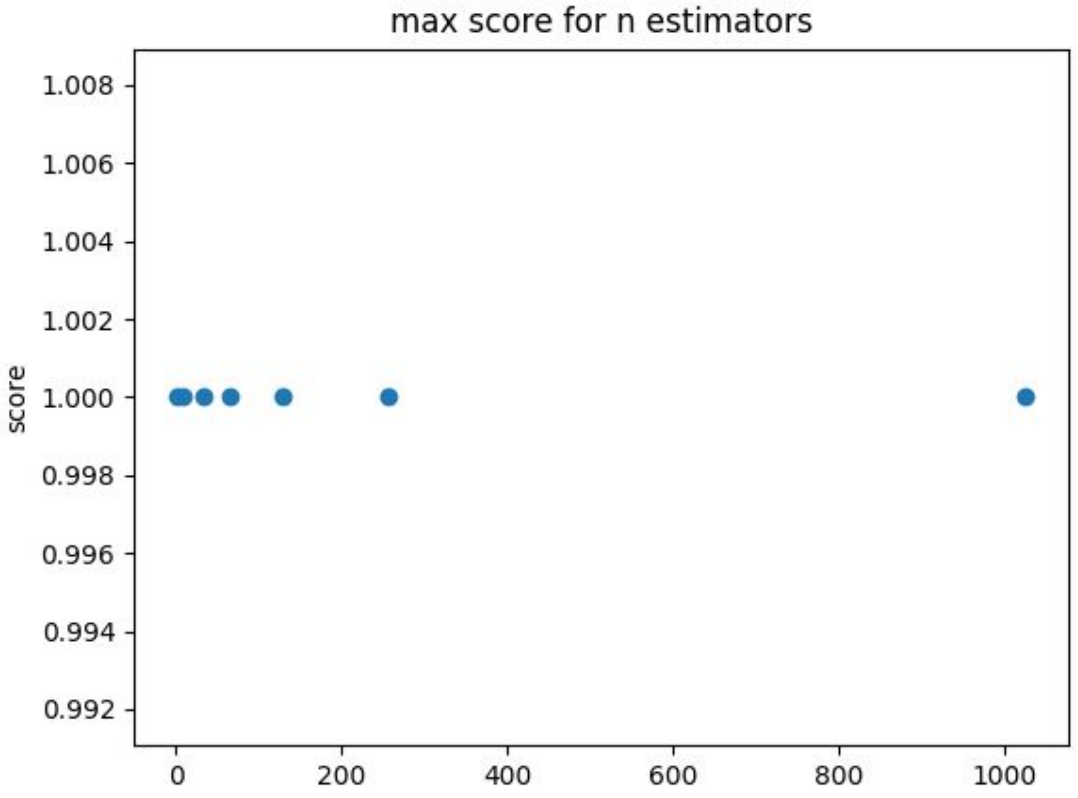


In the following figure, we can see there is a slight increment of the mean score compared with the number of trees in a forest. Having 1 tree represent a minimum of 0.76 score, then after an increment of 10 trees we can see scores above to 0.91 until aproximatelly 0.93 having 300 number of trees. Then the tendency established in

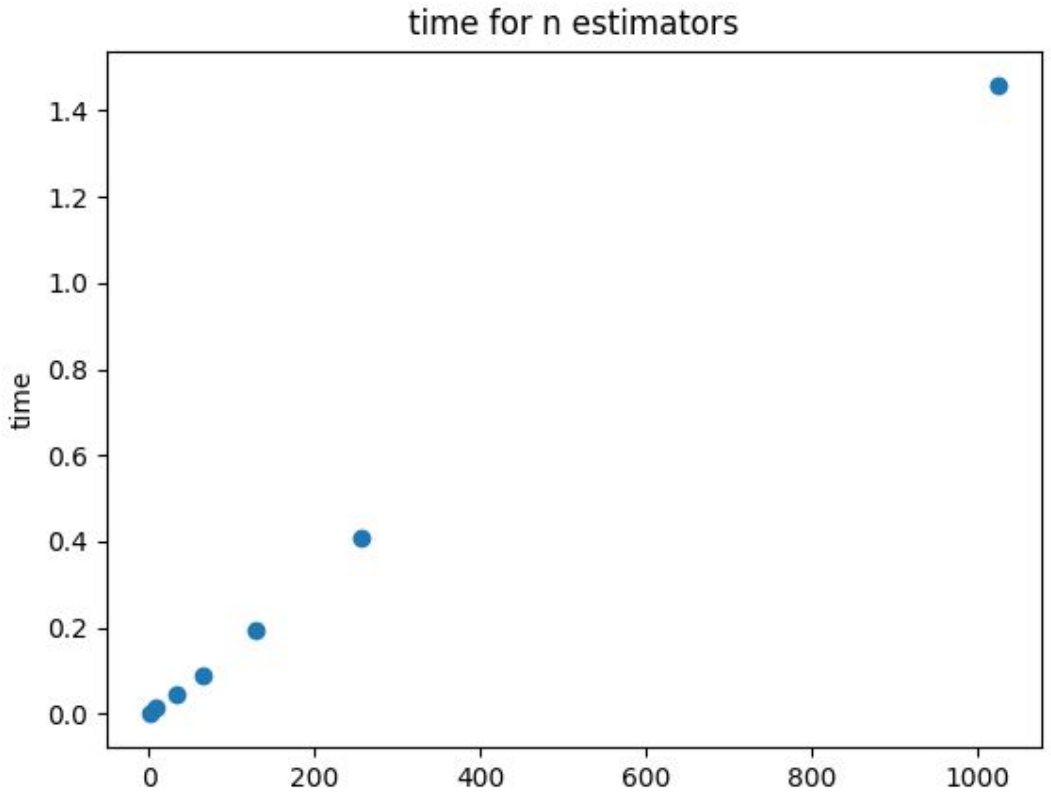
0.935 not having a signicativelly increment even testing with more than 1000 trees.



In the following figure, we can identify that as much as the number of trees in the forest increases, there is not a notable change in the max possible score of the algorithm.

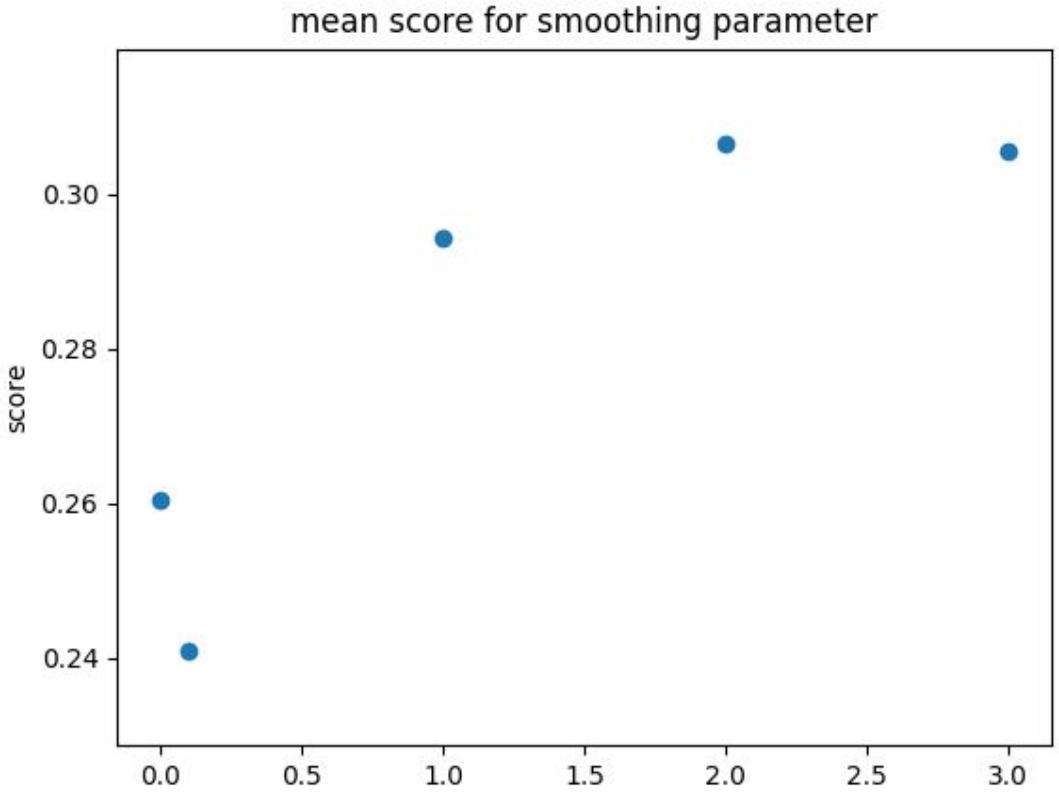


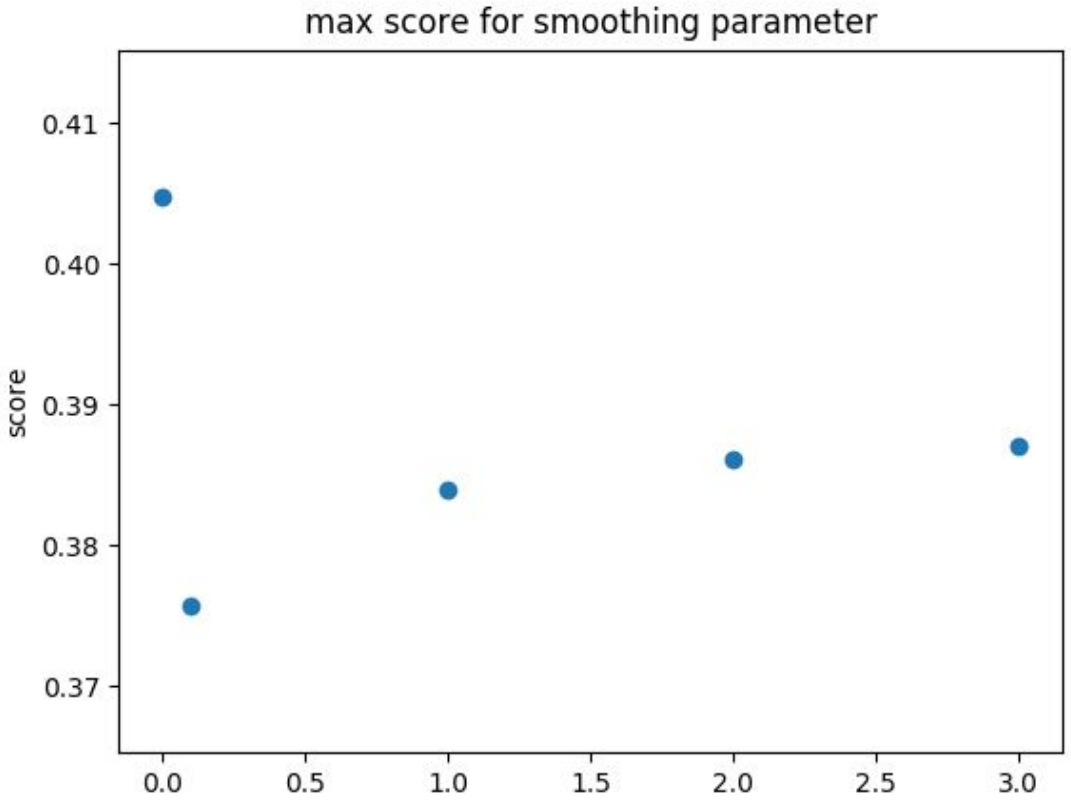
In the following figure, we can see the direct relationship in between the time needed to process the dataset compare with the increment the number of trees in each forest, as much the number of trees increase, the time for processing also does.



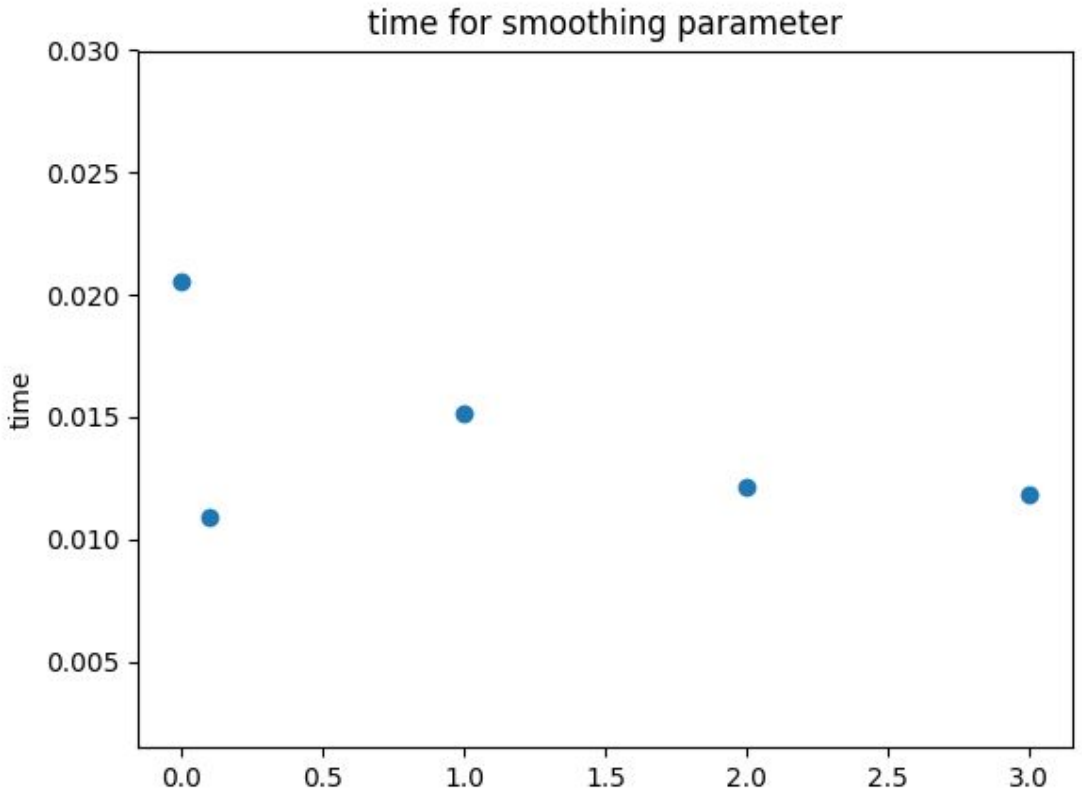
Naive Bayes

For the Naive Bayes algorithm we have tested different alpha values. We can see an improvement for a value near 0, but also some little improvements for increasing values. The standard parameter alpha = 1 got the lowest max score in the test. The mean score is in favour of higher values (following two graphics).

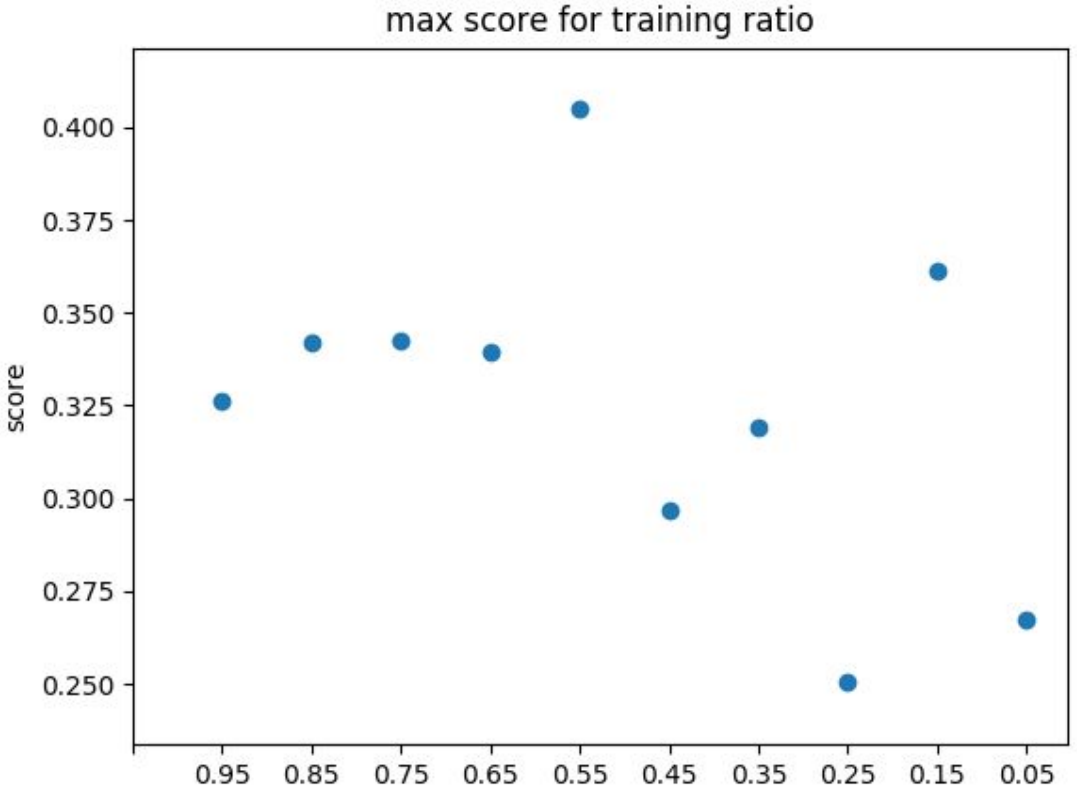

mean score for smoothing parameter
score
0.30
0.28
0.26
0.24
0.0
0.5
1.0
1.5
2.0
2.5
3.0



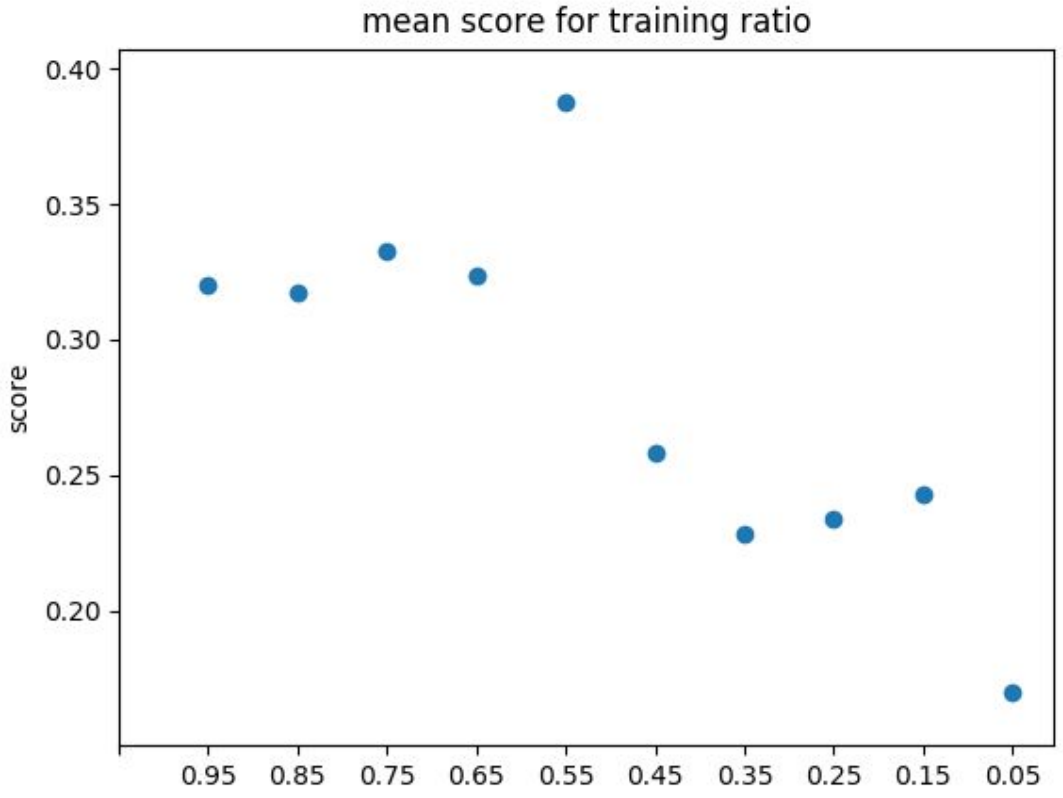
The following figure shows the time necessary for computation. As we run the test only once, there might be some factors influencing the execution time and we could same they are quite the same for all parameters.



Regarding the max score estimator compare with the percent of samples for training and testing we can observe the behaviour is sparse. The best score was reached having 55% of data for training and 45% for testing.

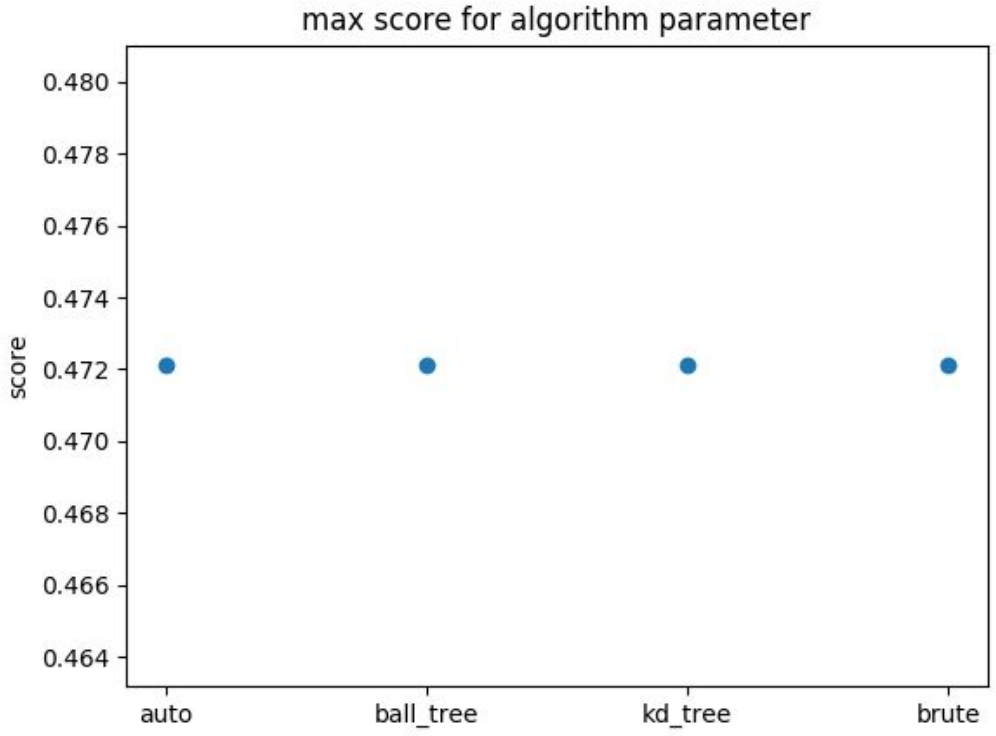


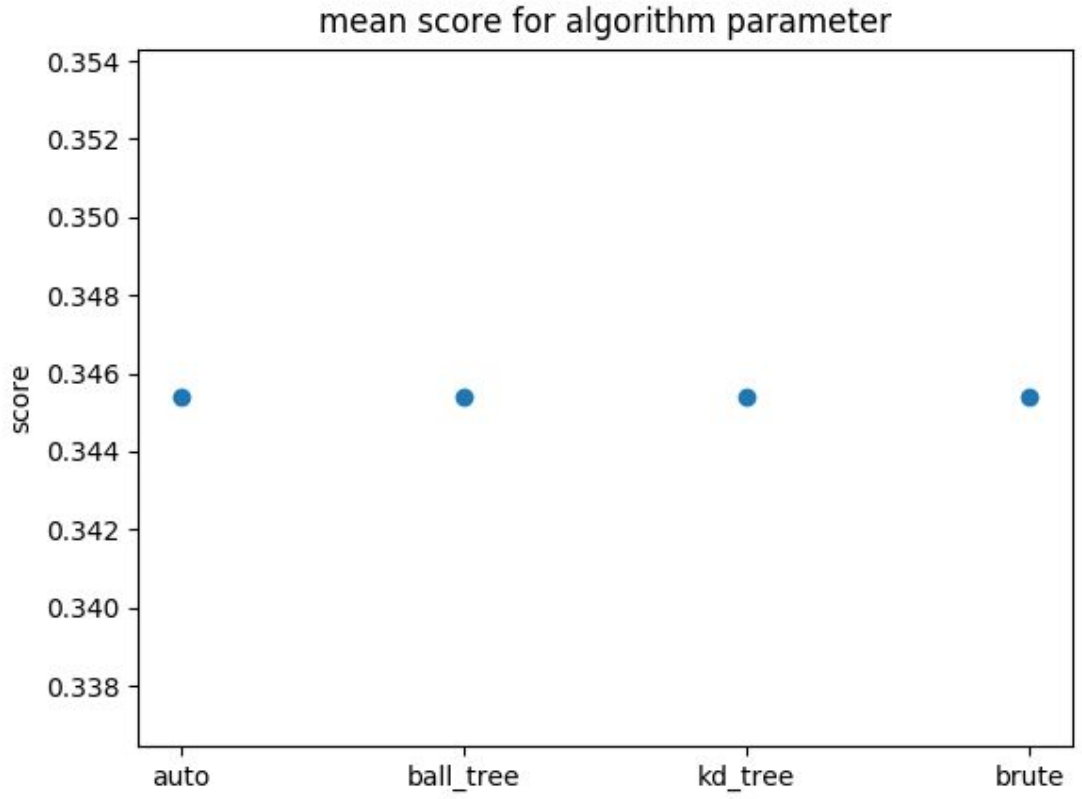
In the following figure we can see there is no an increment of the mean score regarding the percent of sample's split after having 55% of the samples for training and the rest for testing, with this value the highiest score was close to 0.4. Having few samples for taining (below 50%) induces a weak model with scores in between 0.3 and 0.35. In the other hand, having more than 55% of samples for training induces overfitting in the model and generate also lower scores in the mean (below 0.25 until almost 0.05).



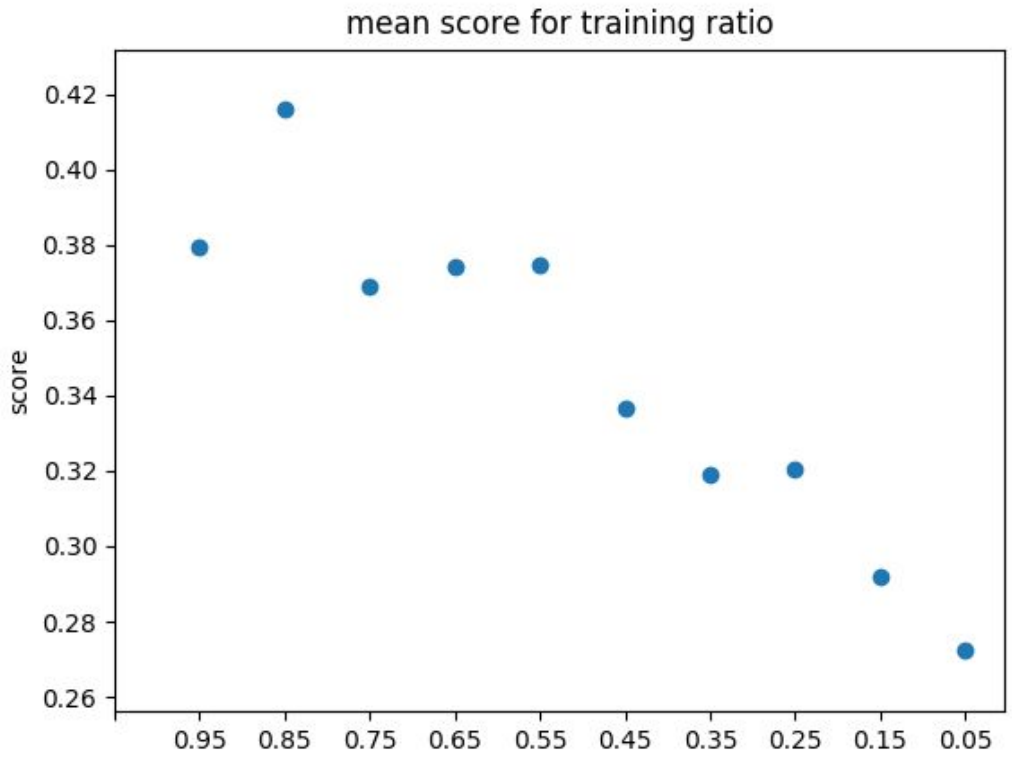
K-Nearest Neighbours

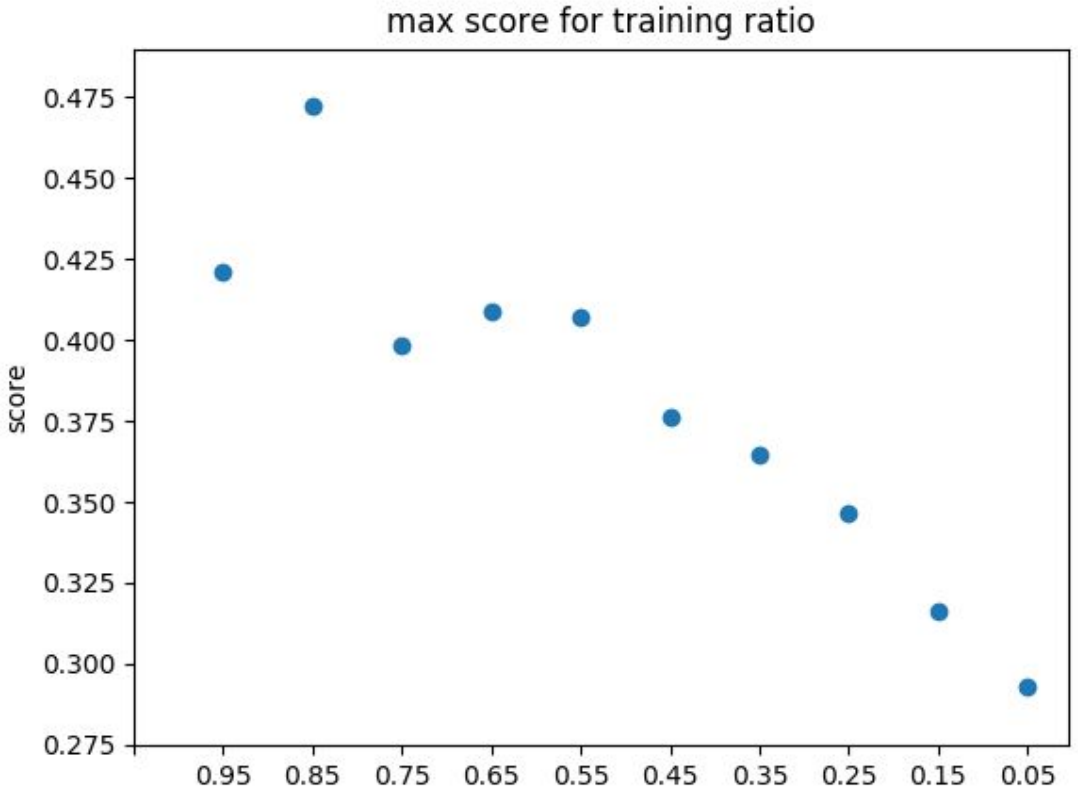
As we can see in the following two figures, the voting algorithm chosen does not really affect the results.


max score for algorithm parameter
score
0.480
0.478
0.476
0.474
0.472
0.470
0.468
0.466
0.464
auto
ball_tree
kd_tree
brute

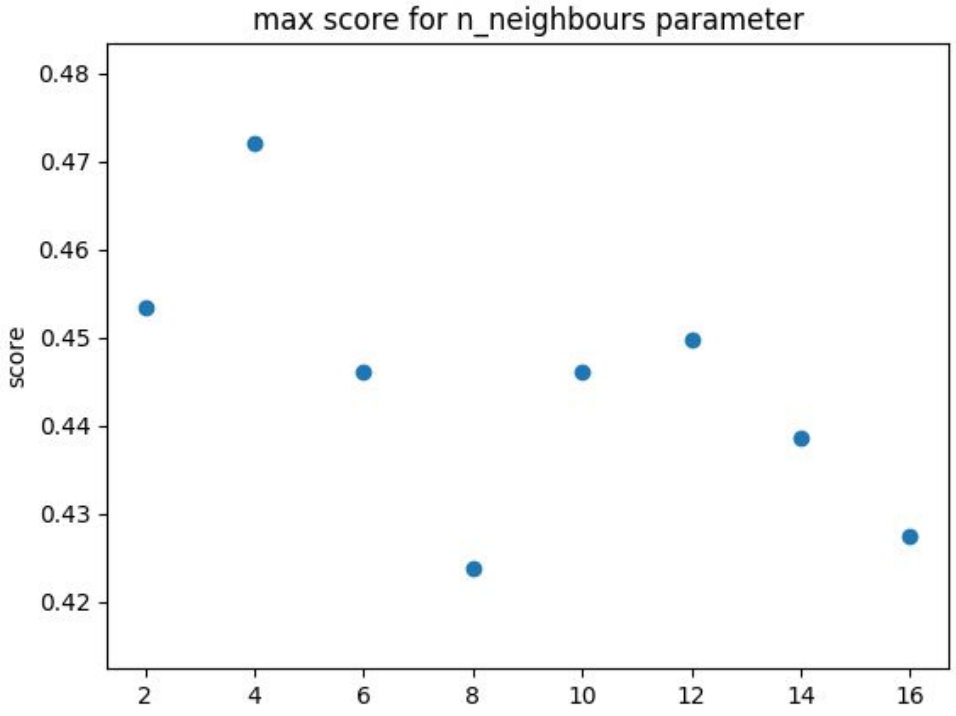

mean score for algorithm parameter
score
0.354
0.352
0.350
0.348
0.346
0.344
0.342
0.340
0.338
auto
ball_tree
kd_tree
brute

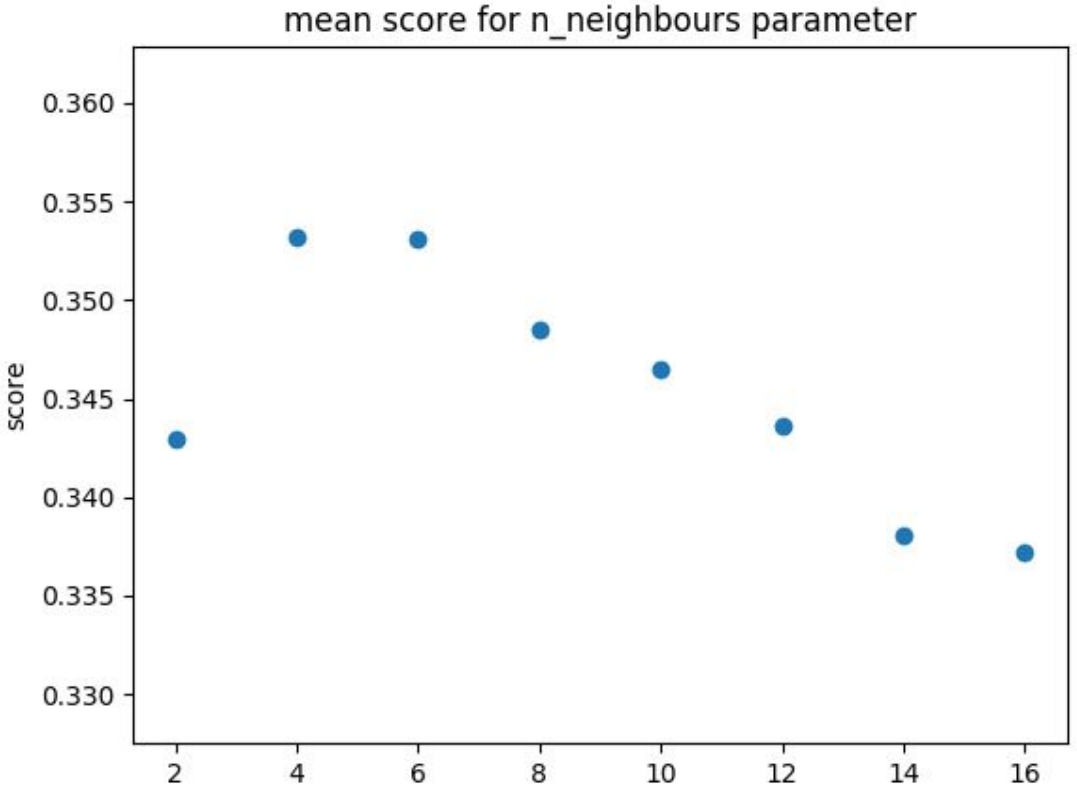
With dropping training:test ratio the efficiency of this the k nearest neighbours on this dataset is also dropping, almost linearly with some outliers.



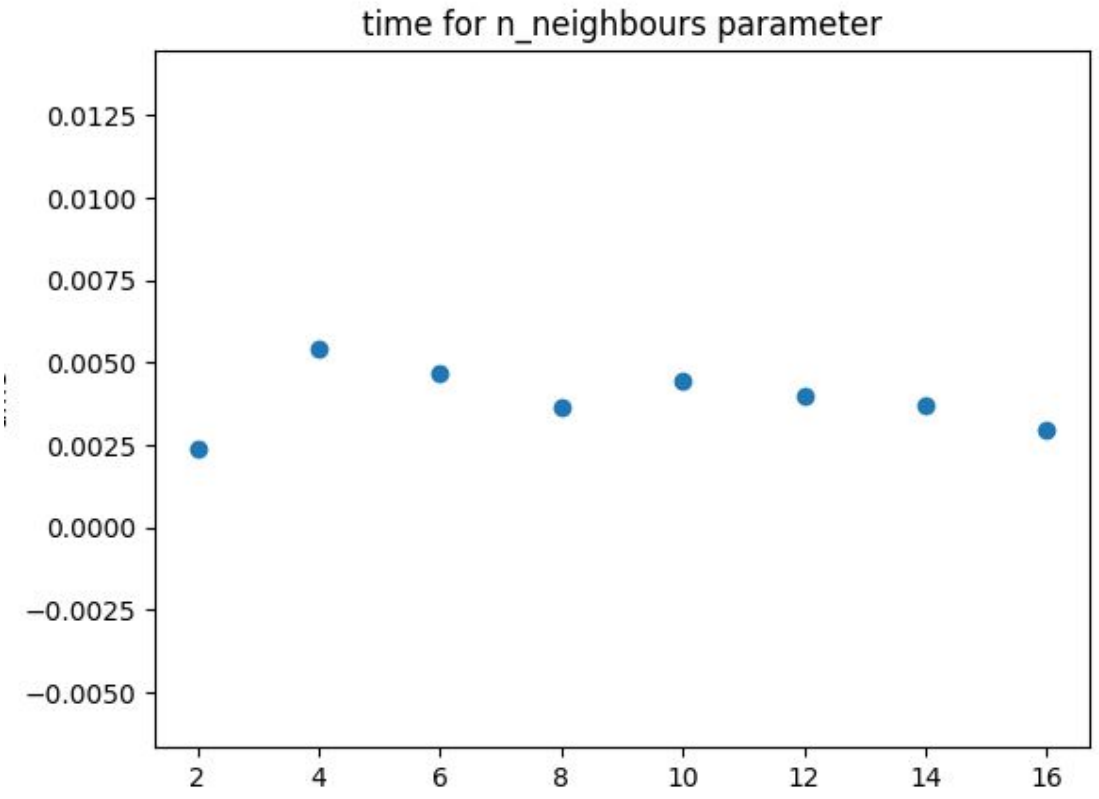


The following two figures and especially the second one describe the influence of the number of neighbours used for classification. As we can see for this dataset it is optimal to use 4 estimators.

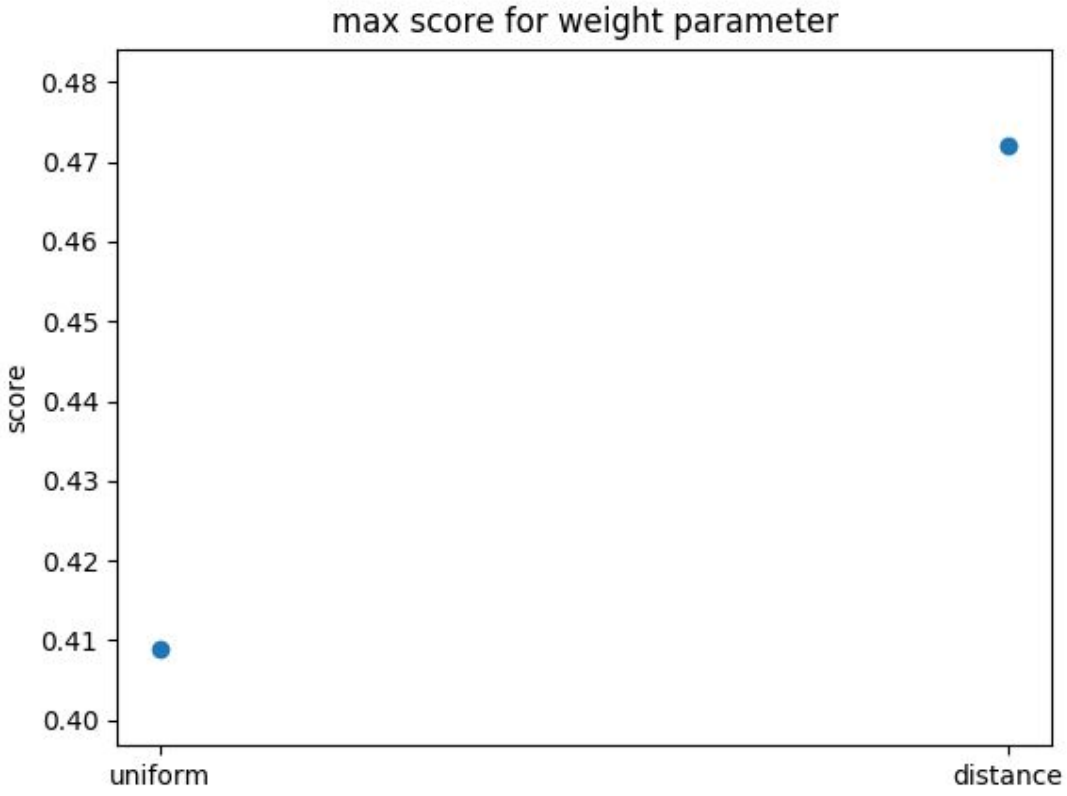

max score for n_neighbours parameter
score
0.48
0.47
0.46
0.45
0.44
0.43
0.42
2
4
6
8
10
12
14
16

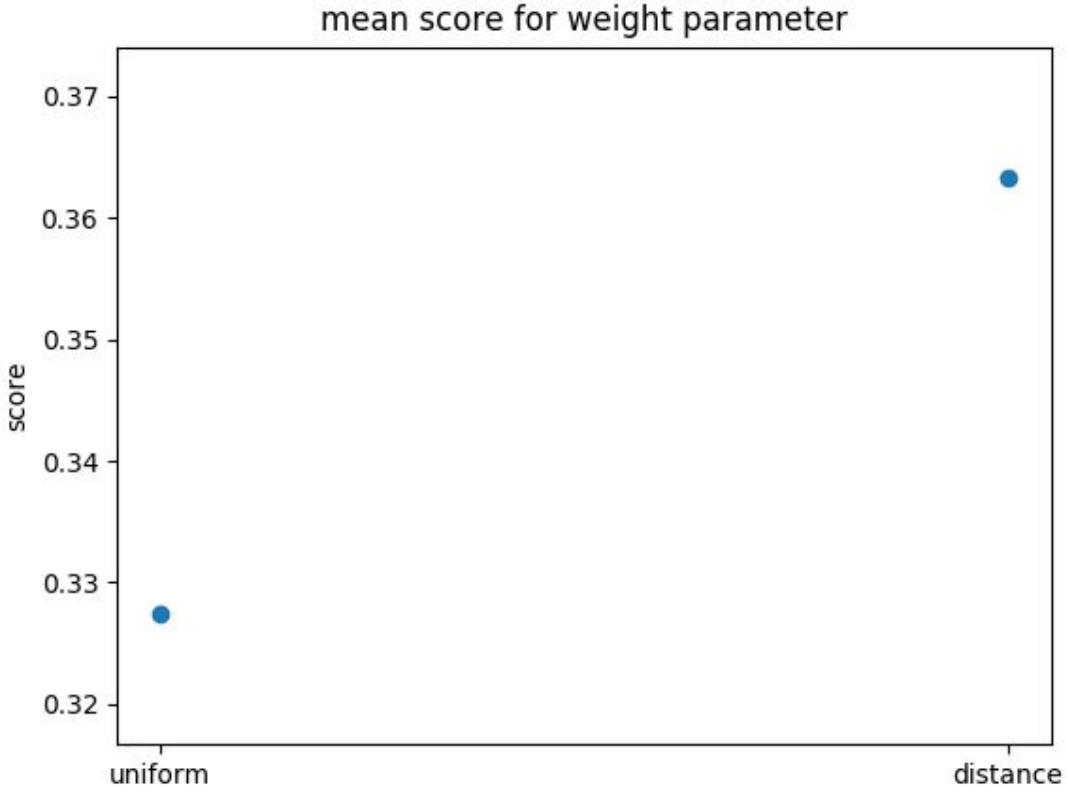

mean score for n_neighbours parameter
score
0.360
0.355
0.350
0.345
0.340
0.335
0.330
2
4
6
8
10
12
14
16

The computation time is higher for a higher score achieved by the estimators which is surprising, because we would assume more estimators to take more time.



Using the weight parameter set to distance gave better results than uniform distribution of the weights.


max score for weight parameter
0.48
0.47
0.46
0.45
0.44
0.43
0.42
0.41
0.40
score
uniform
distance


mean score for weight parameter
0.37
0.36
0.35
0.34
0.33
0.32
score
uniform
distance

4. Flags Data Set (http://archive.ics.uci.edu/ml/datasets/Flags)

This data file contains details of various nations and their flags. In this data file the fields are separated by spaces (not commas). With this data you can try things like predicting the religion of a country from its size and the colours in its
Flag.

A. 194 samples with 30 attributes and no missing values.
B. There are 10 attributes with numeric values and the others
are nominal-valued.
C. Distribution/histograms of values in the attributes
D. Characteristics
The feature variables are represented by the following 30 attributes.
• 1. name Name of the country concerned
• 2. **landmass** 1=N.America, 2=S.America, 3=Europe, 4=Africa, 4=Asia, 6=Oceania - [nominal] One hot encoding was used in pre-processing.
• 3. zone Geographic quadrant, based on Greenwich and the Equator 1=NE, 2=SE, 3=SW, 4=NW - [nominal] One hot encoding was used in pre-processing.
• 4. area in thousands of square km [0 - 22402]
• 5. population in round millions [0 - 1008]
• 6. language 1=English, 2=Spanish, 3=French, 4=German, 5=Slavic, 6=Other, Indo-European, 7=Chinese, 8=Arabic, 9=Japanese/Turkish/Finnish/Magyar, 10=Others - [nominal] One hot encoding was used in pre-processing.

• 7. religion 0=Catholic, 1=Other Christian, 2=Muslim, 3=Buddhist, 4=Hindu, 5=Ethnic, 6=Marxist, 7=Others - [nominal] One hot encoding was used in pre-processing.
• 8. bars Number of vertical bars in the flag [0 - 5]
• 9. stripes Number of horizontal stripes in the flag [0 - 14]
• 10. colours Number of different colours in the flag [1 - 8]
• 11. red 0 if red absent, 1 if red present in the flag [0 - 1]
• 12. green same for green [0 - 1]
• 13. blue same for blue [0 - 1]
• 14. gold same for gold (also yellow) [0 - 1]
• 15. white [0 - 1]
• 16. black [0 - 1]
• 17. orange and brown [0 - 1]
• 18. mainhue predominant colour in the flag (tie-breaks decided by taking the topmost hue, if that fails then the most central hue, and if that fails the leftmost hue) - [nominal] One hot encoding was used in pre-processing.

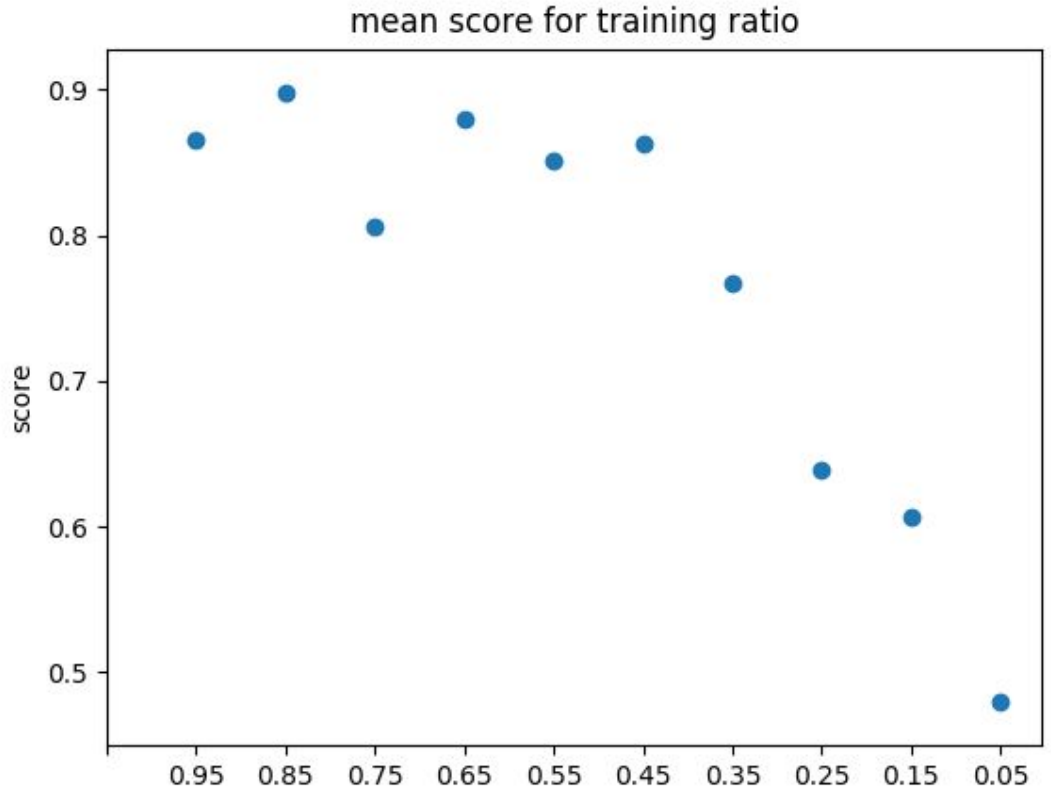
• 19. circles Number of circles in the flag - another column was added to identify if the attribute is present or not in the flag (In the preprocessing we skipped the quantity of the number of instances of the attribute in the flag) [0 - 1]
• 20. crosses Number of (upright) crosses - another column was added to identify if the attribute is present or not in the flag (In the preprocessing we skipped the quantity of the number of instances of the attribute in the flag) [0 - 1]
• 21. saltires Number of diagonal crosses - another column was added to identify if the attribute is present or not in the flag (In the preprocessing we skipped the quantity of the number of instances of the attribute in the flag) [0 - 1]
• 22. quarters Number of quartered sections - another column was added to identify if the attribute is present or not in the flag (In the preprocessing we skipped the quantity of the number of instances of the attribute in the flag) [0 - 1]
• 23. sunstars Number of sun or star symbols - another column was added to identify if the attribute is present or not in the flag (In the preprocessing we skipped the quantity of the number of instances of the attribute in the flag) [0 - 1]
• 25. triangle 1 if any triangles present, 0 otherwise - [0 - 1]
otherwise 0
• 27. animate 1 if an animate image (e.g., an eagle, a tree, a human hand) present, 0 otherwise - [0 - 1]
• 28. text 1 if any letters or writing on the flag (e.g., a motto or slogan) - [0 - 1]
• 29. topleft colour in the top-left corner (- [nominal] One hot encoding was used in pre-processing.
• 30. botright Colour in the bottom-left corner - [nominal] One hot encoding was used in pre-processing.

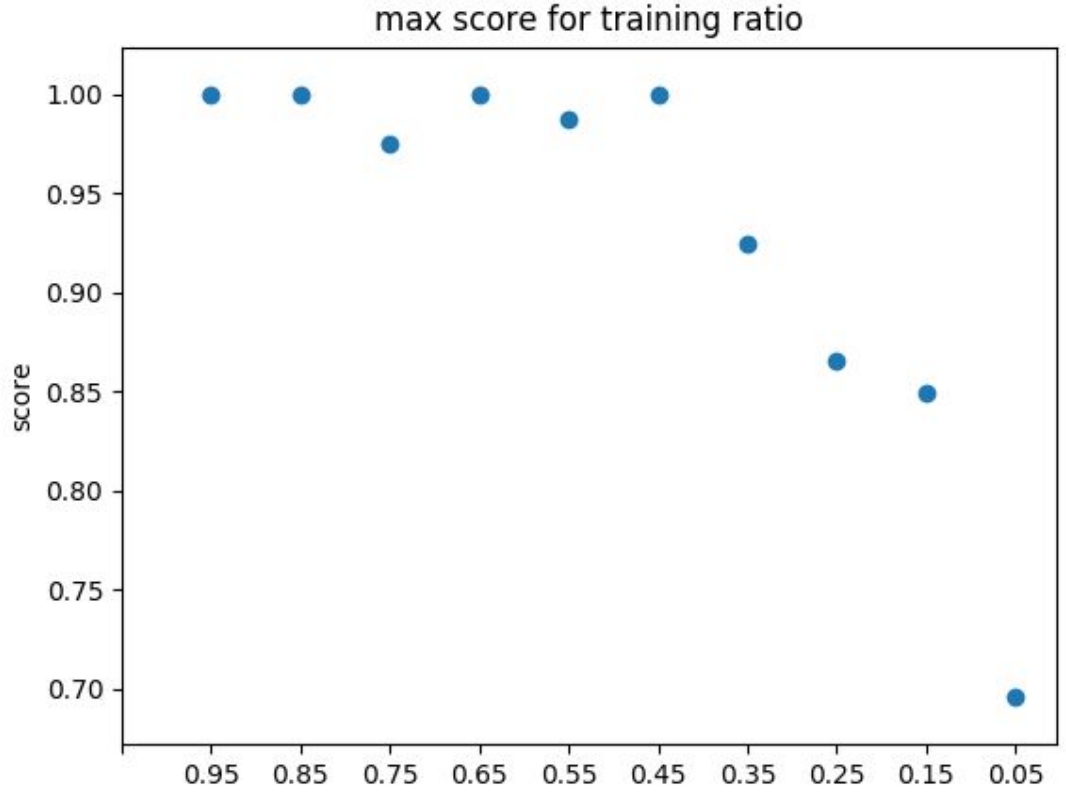
E. Further Analysis

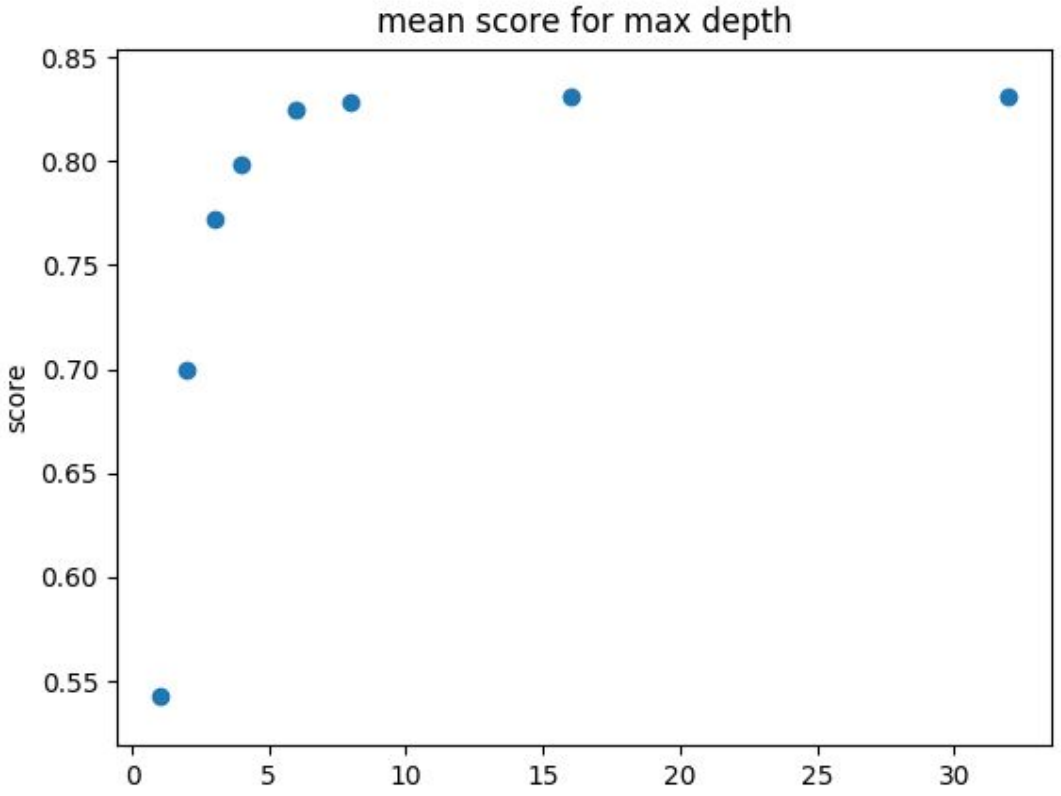
We generated box plots to analyze the behaviour of the feature variables. Those graphics show boxes that represent the amount of data in between the first quartile to the third quartile, the horizontal line goes through the box at the median value, the whiskers go from each quartile to the minimum or maximum and the dots represent the outliers that have values outside 1.5 times the interquartile range above the upper quartile and below the lower quartile (see Appendices).

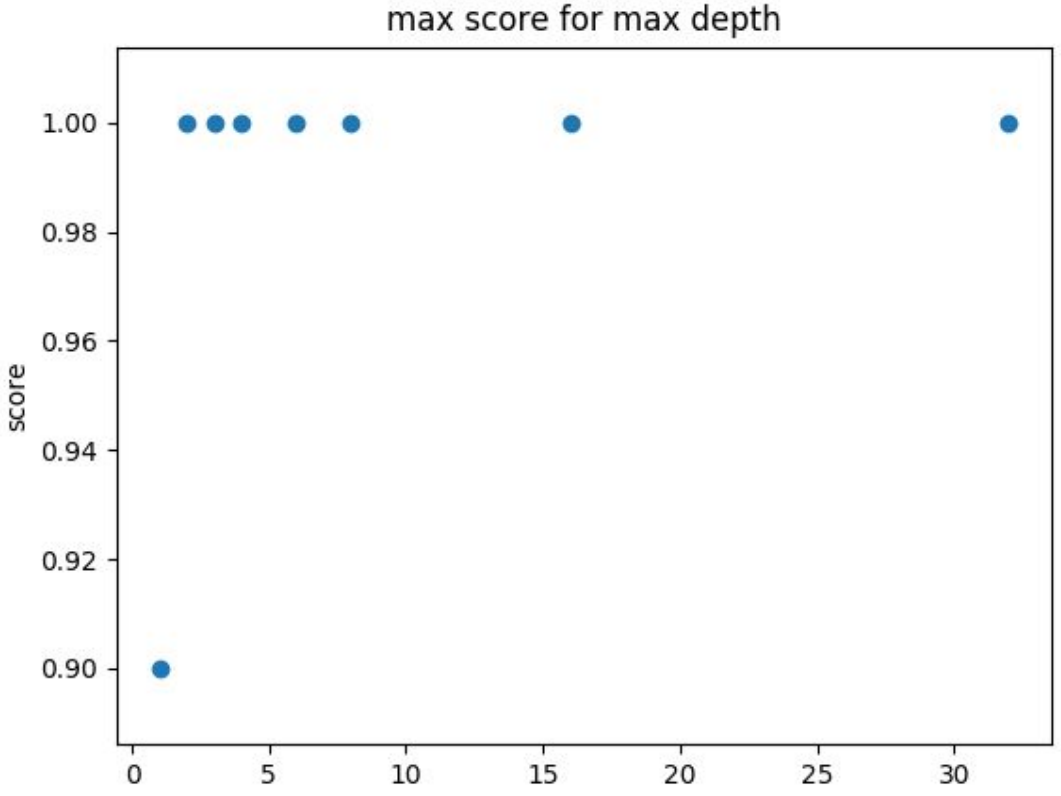
The classifier used was the feature **landmass** to predict the continent the country has a place: Using One hot Encoding it is set six new columns to represent the continents: N.America, S.America, Europe, Africa, Asia and Oceania.

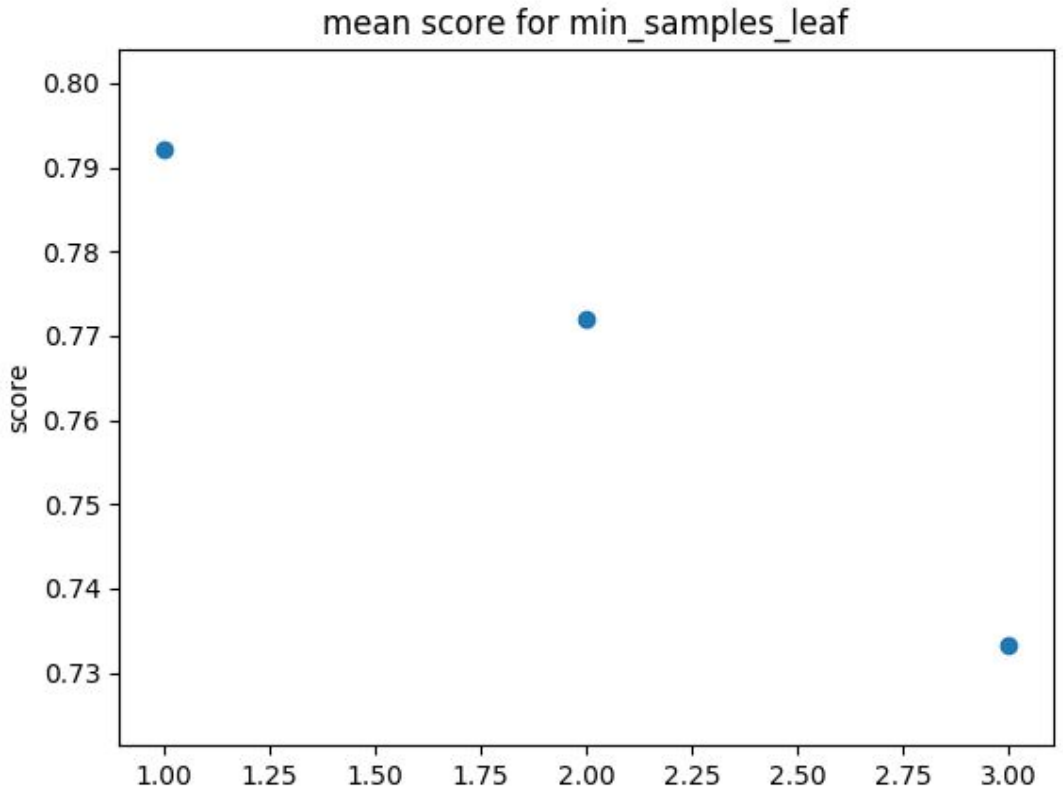
Experiments with the algorithms and parameters:
Random Forest
The random forest algorithms behave for this dataset similarly to the cardiography dataset. See figures below.

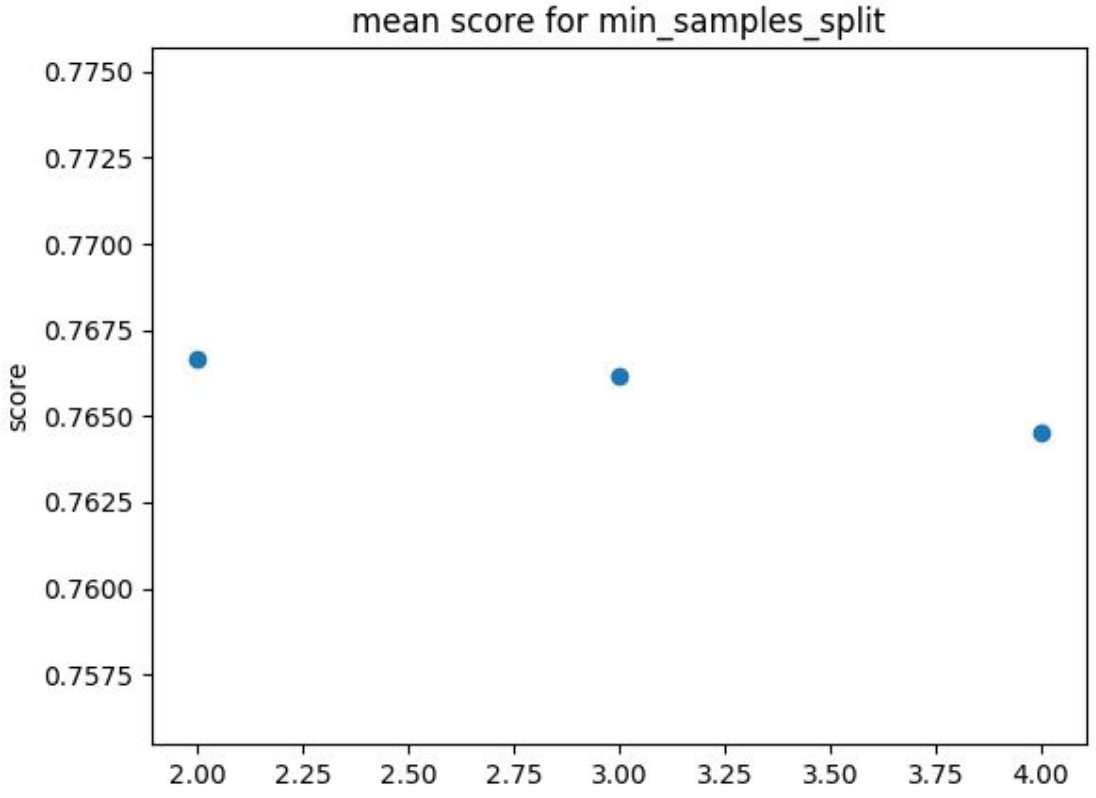

mean score for training ratio
score
0.9
0.8
0.7
0.6
0.5
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05

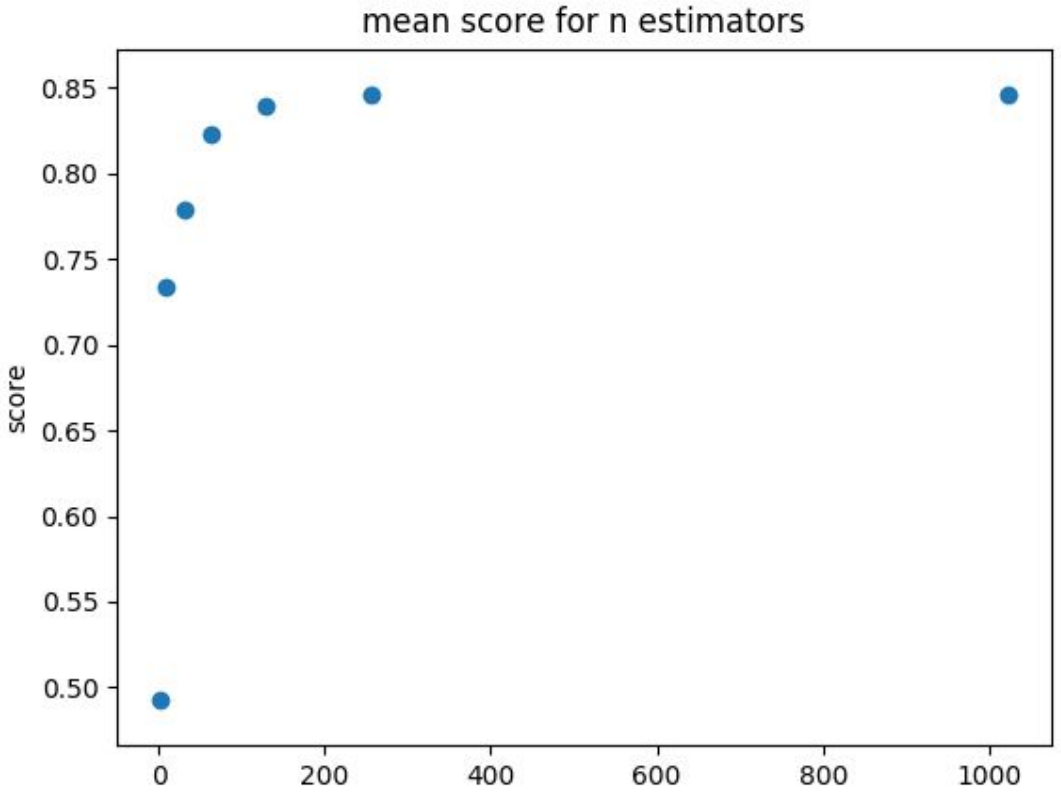

max score for training ratio
score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.95
0.85
0.75
0.65
0.55
0.45
0.35
0.25
0.15
0.05

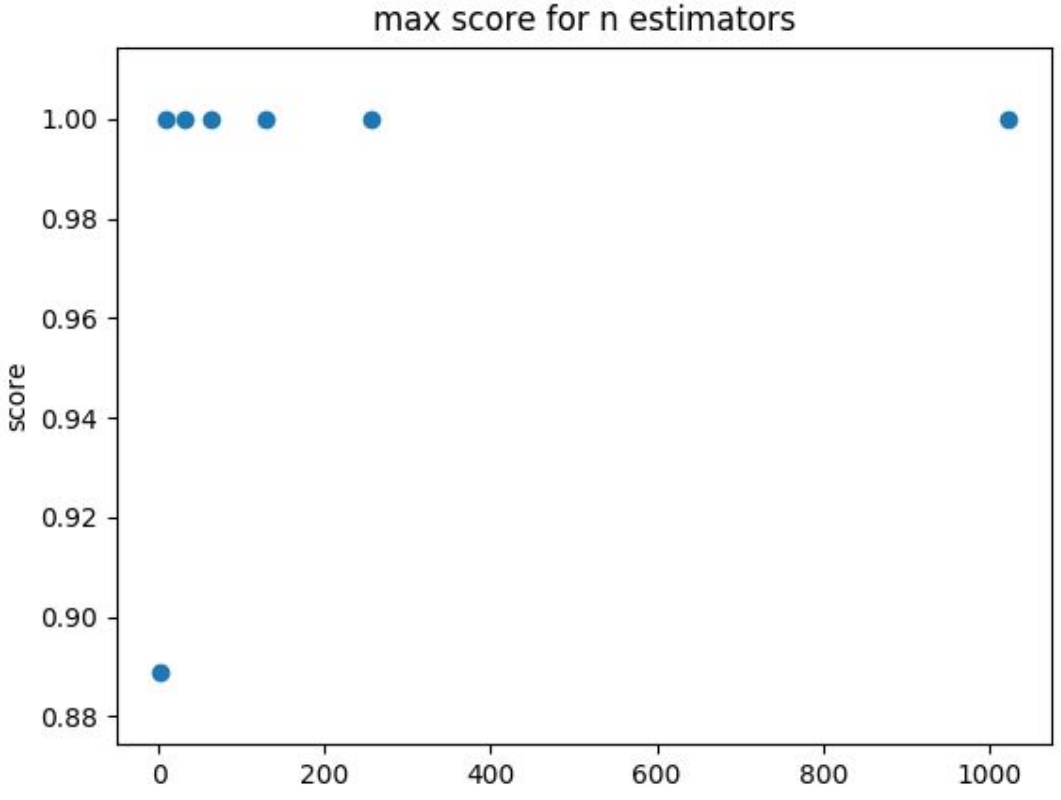

mean score for max depth
score
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0
5
10
15
20
25
30


max score for max depth
score
1.00
0.98
0.96
0.94
0.92
0.90
0
5
10
15
20
25
30


mean score for min_samples_leaf
score
0.80
0.79
0.78
0.77
0.76
0.75
0.74
0.73
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

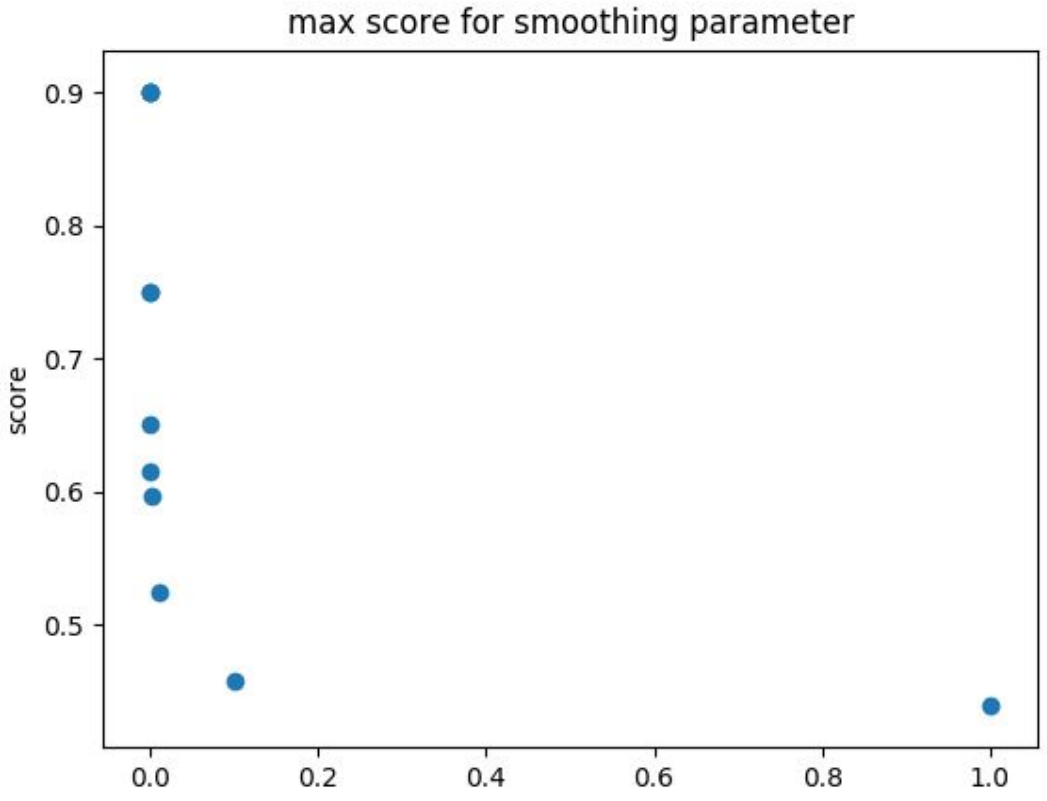

mean score for min_samples_split
score
0.7750
0.7725
0.7700
0.7675
0.7650
0.7625
0.7600
0.7575
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00

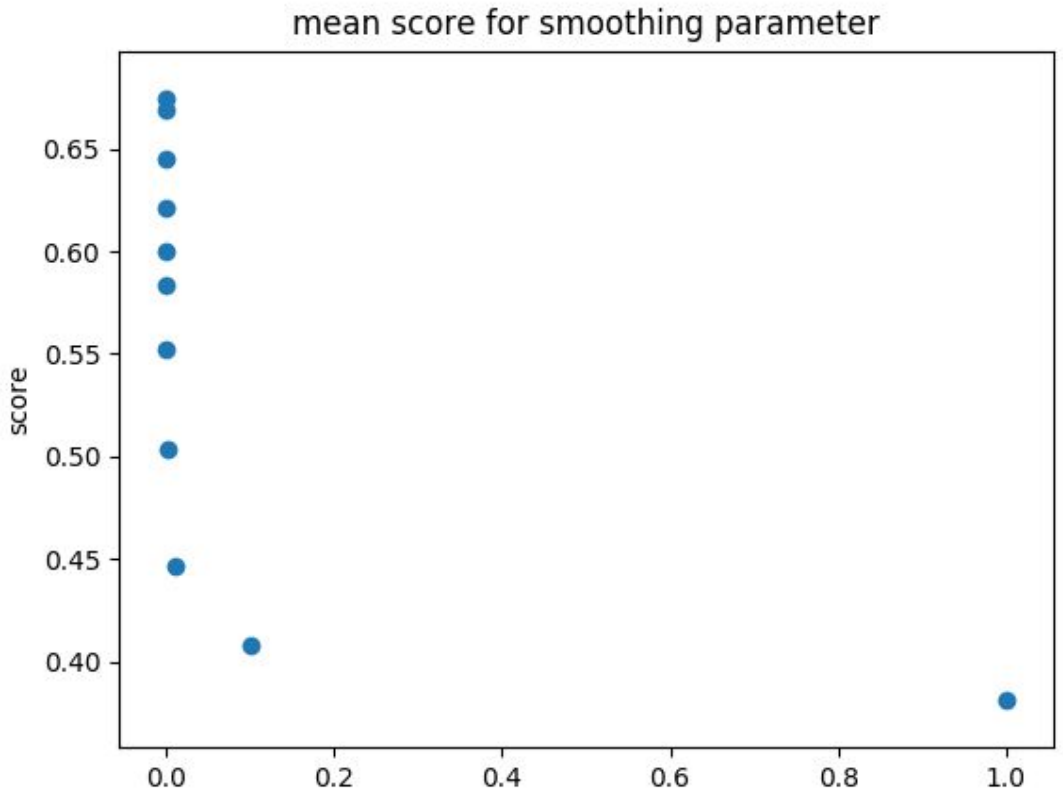

mean score for n estimators
score
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
200
400
600
800
1000


max score for n estimators
score
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0
200
400
600
800
1000

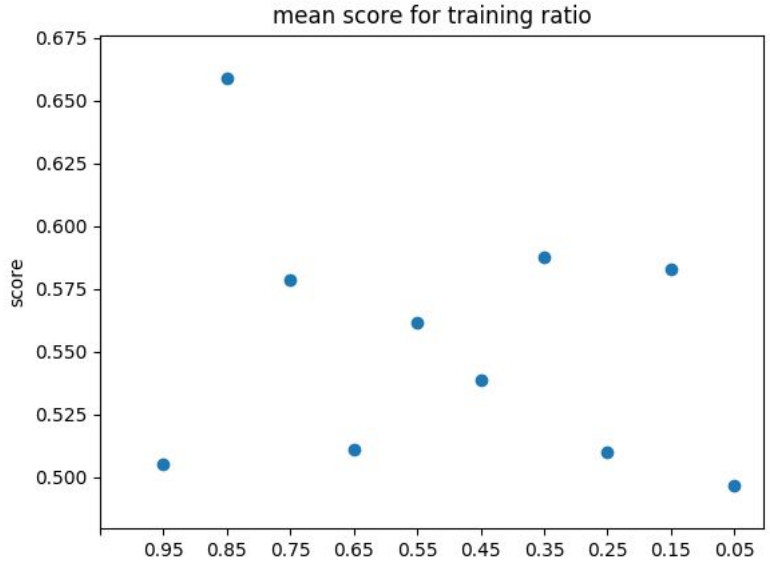
Naive Bayes

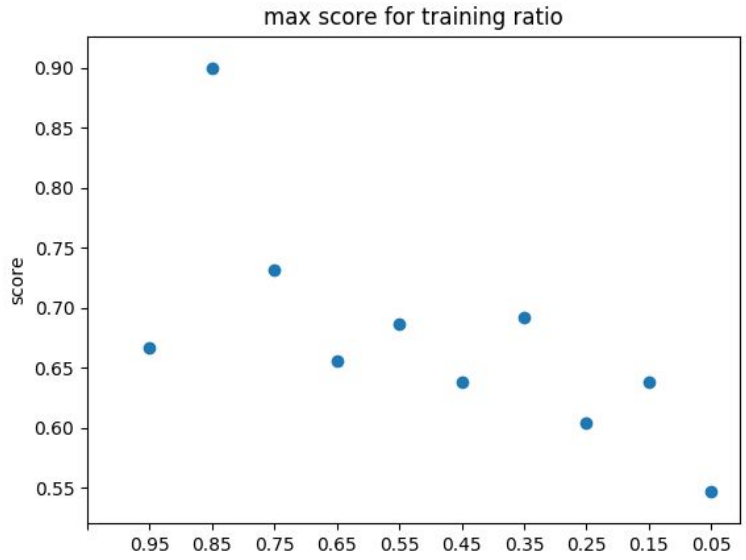
For this dataset the alpha parameter was significant. The closer to zero, the better was the score.





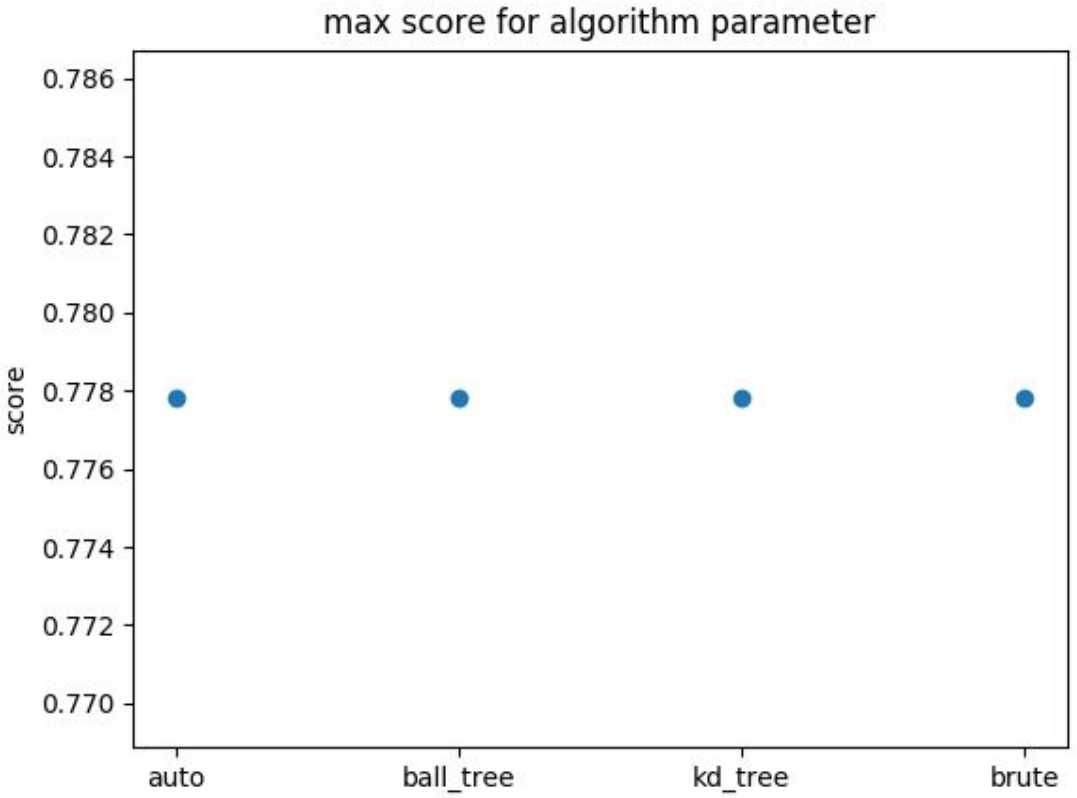
The best score (an outlier) was achieved for 0.85 training:test ratio. The other ratios achieve quite similar scores.

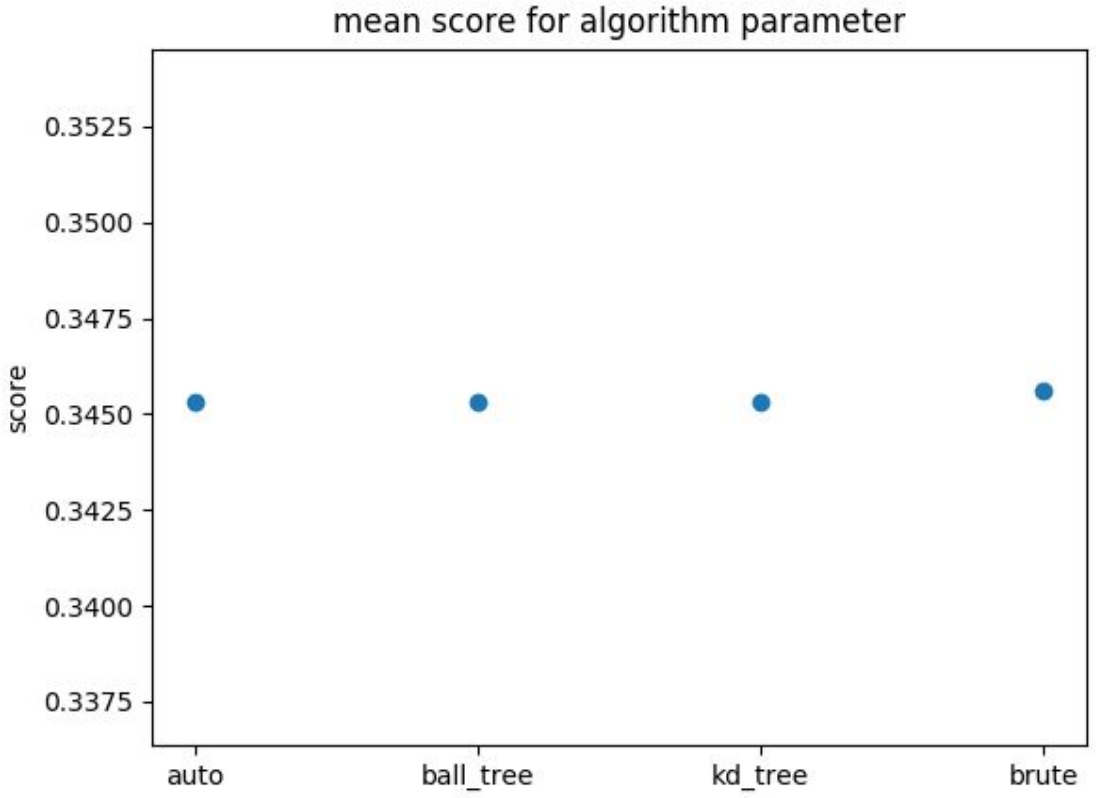




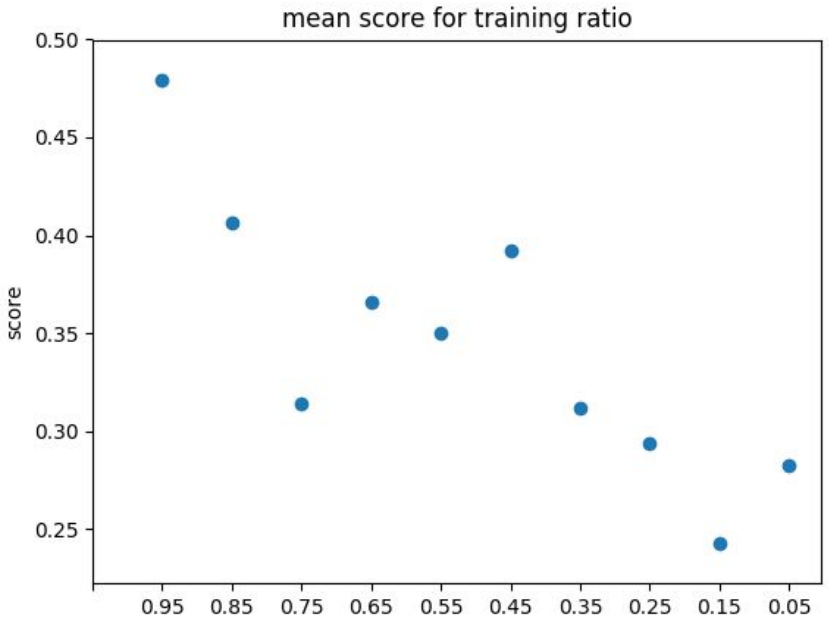
K-Nearest-Neighbours

Here again the algorithm makes no real difference to the score.

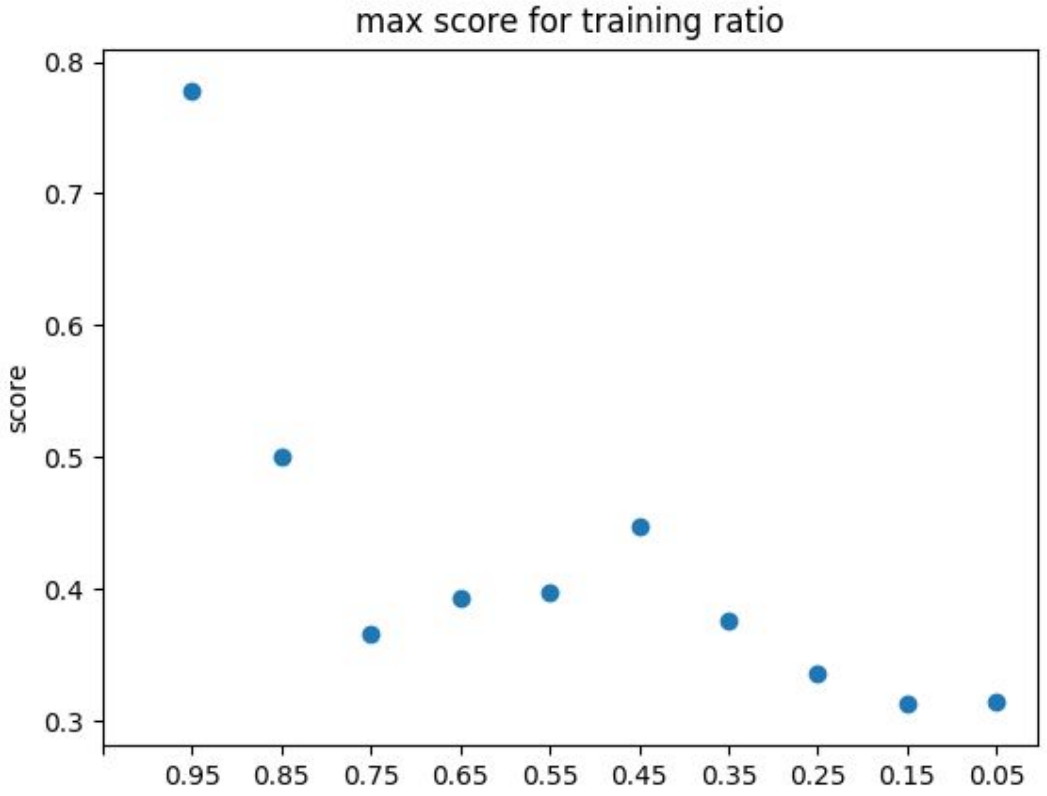




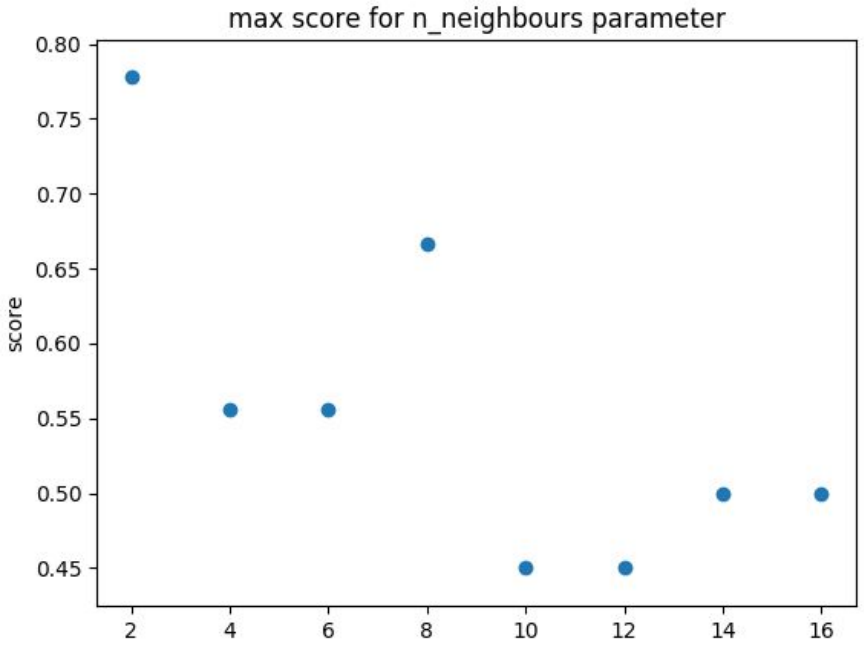
However the training ratio is important for this algorithm and we can see almost

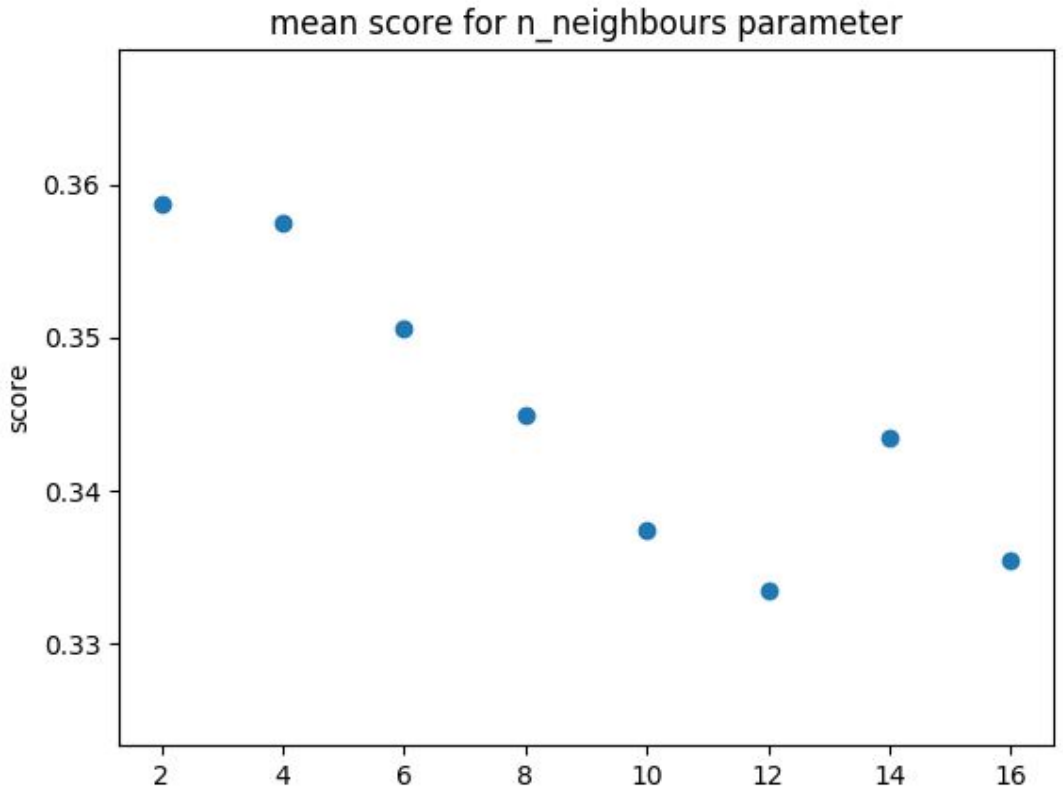


linear correlation.

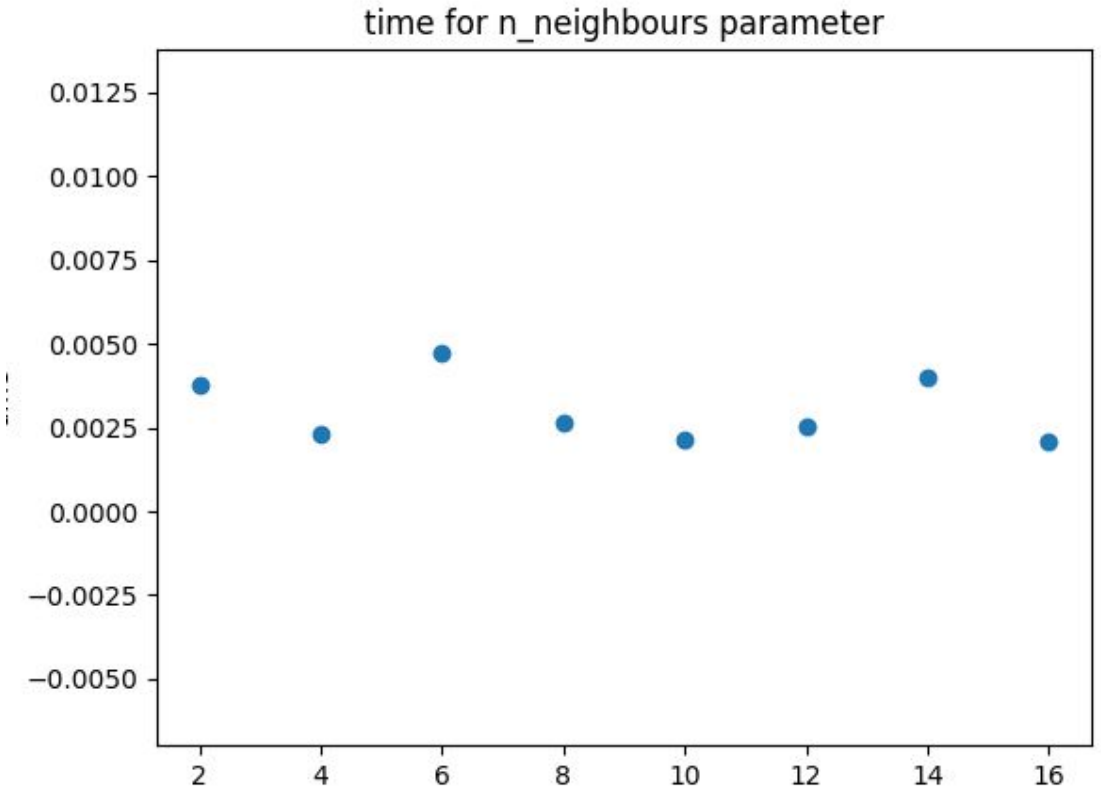


The best was the run with two estimators.

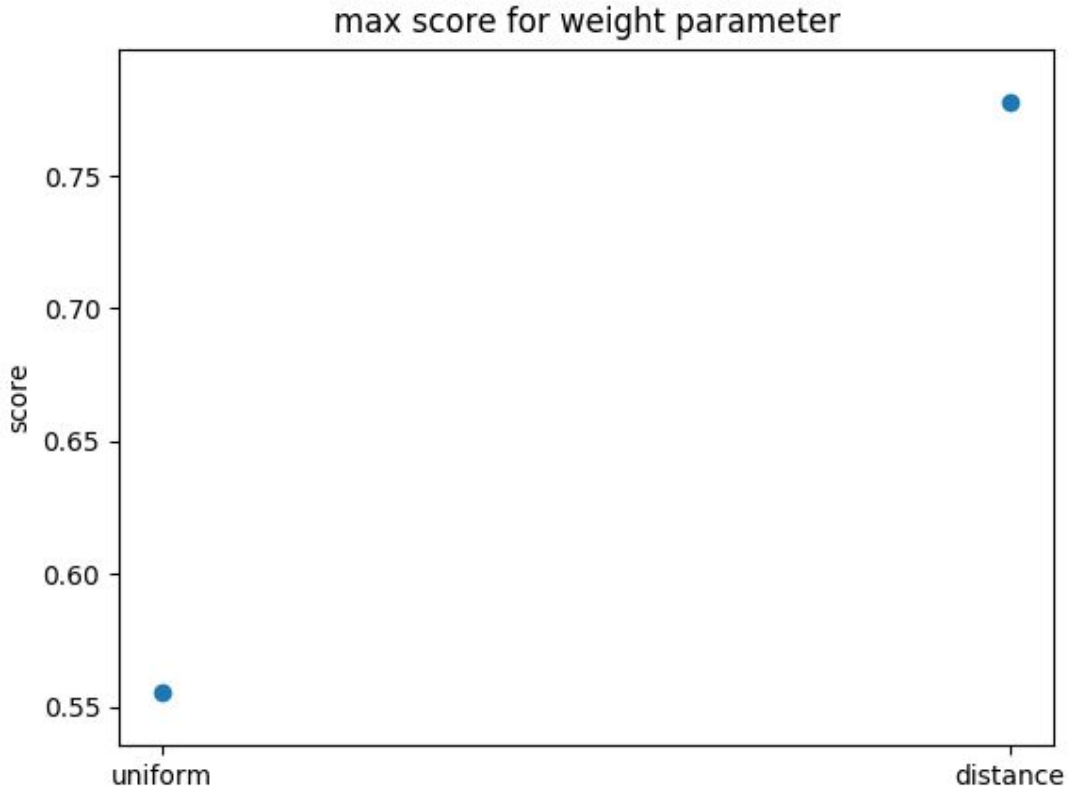


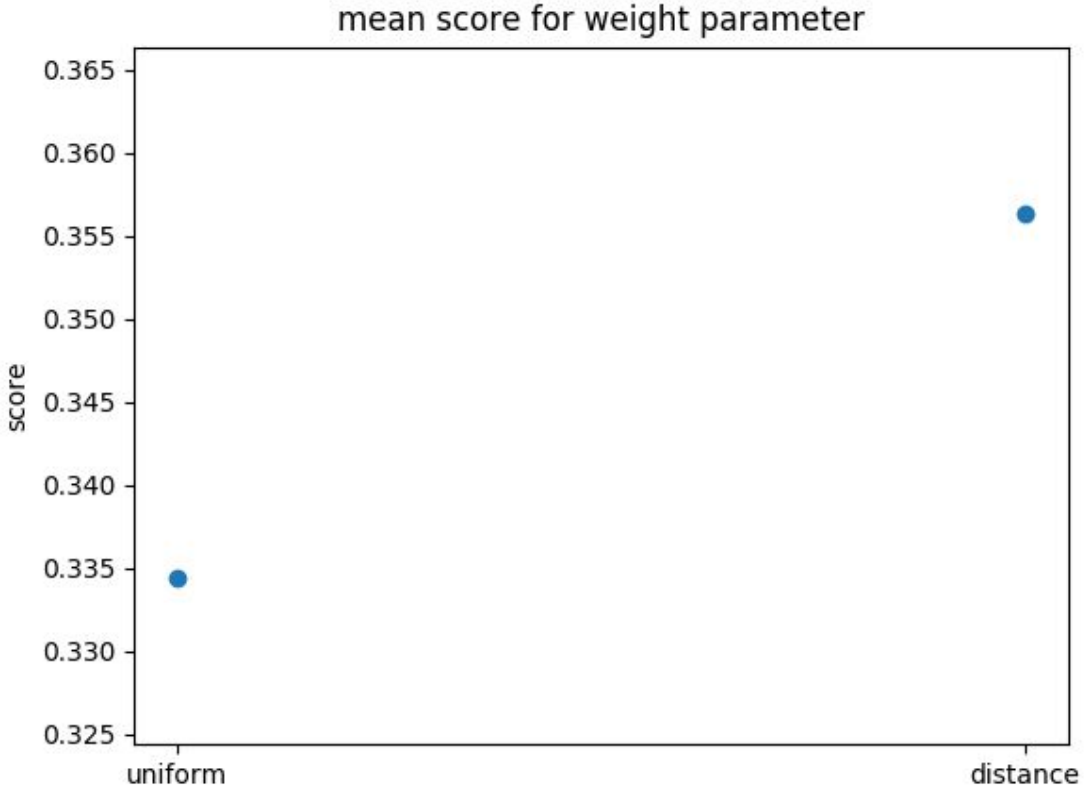


The time necessary for computation is again the same for all of the parameters.



The distance weighting function again improves the score.


max score for weight parameter
score
0.75
0.70
0.65
0.60
0.55
uniform
distance


mean score for weight parameter
score
0.365
0.360
0.355
0.350
0.345
0.340
0.335
0.330
0.325
uniform
distance

Experiment with the datasets and classifiers, by evaluating their performance.

Chose a number of performance measures. Argue why you chose them, what they measure, and whether they are sufficient.

Experiment with different parameter settings

And report on it - report not only one (best/random) result from a classifier on a specific dataset, but several results!

Compare results among classifiers and datasets.

Aggregated comparison, e.g. pick best settings for each combination

Significance testing against at least one baseline

• Evaluate the effect of pre-processing (mostly scaling)

Compare results w/o pre-processing vs. applied pre-processing methods (be careful about built-in pre-processing in some implementation!)

• Record (approximate) runtimes of the classifiers

• Summarise your results - tables or figures