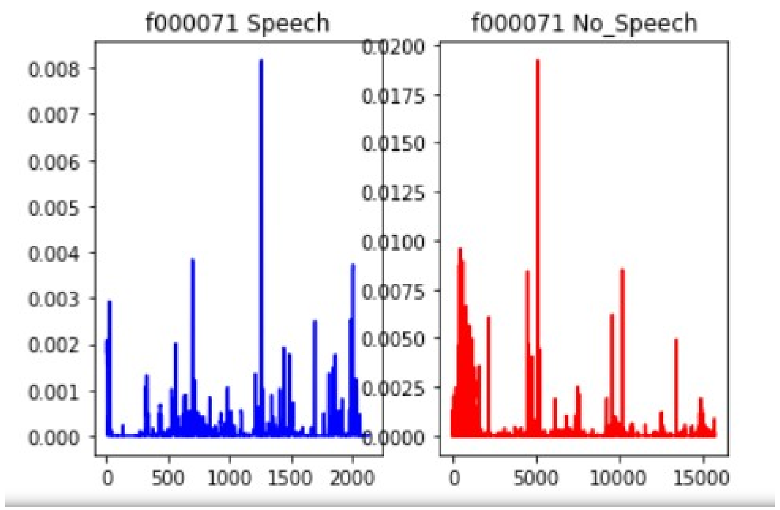
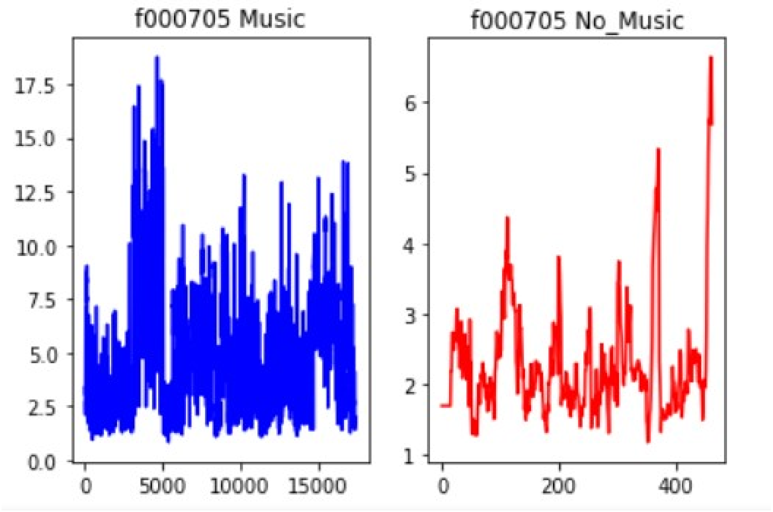
Music & Speech Detection in Radio Broadcasts

Table of Content

*Things to Note: Due to weak computer, not all files provided was used, a portion was randomly chosen and used to train classifier

- Diagrams created based on data
- Discussion on Features, Outliers and Distributions of Result
- Most accurate algorithm
- Observation on overfitting behavior
- Variation in Classification accuracy based on relevant parameters

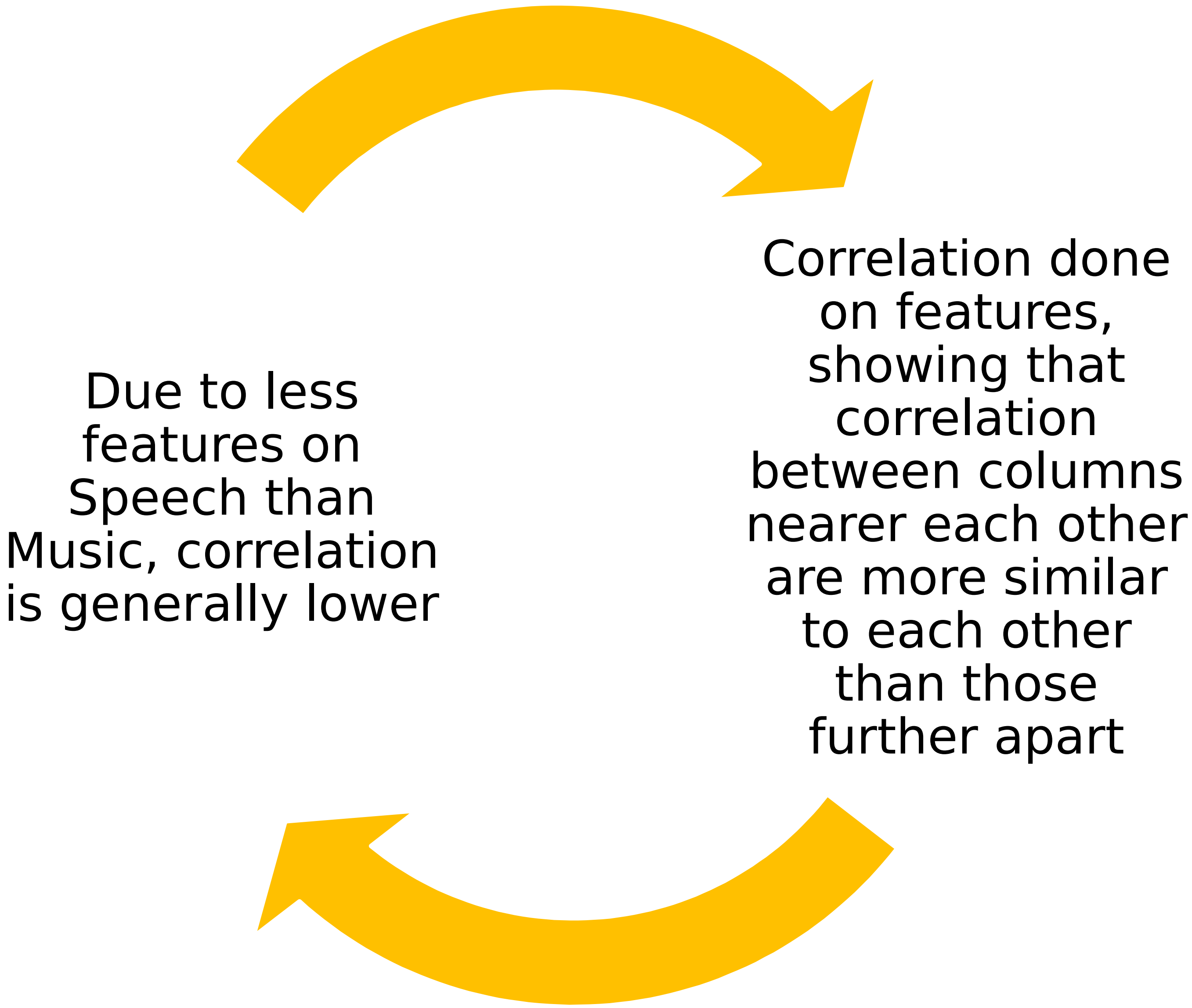




Informative & Uninformative Features

- When there is music, there is generally a higher frequency range than when it has no music.
  - F000657 No_music, frequency limited to between -0.10 and 0.20
- Number of rows detected classifying as speech or no music is generally shorter than “music” or “no speech”.
- Opposites groups (music, no speech) vs (no music, speech)

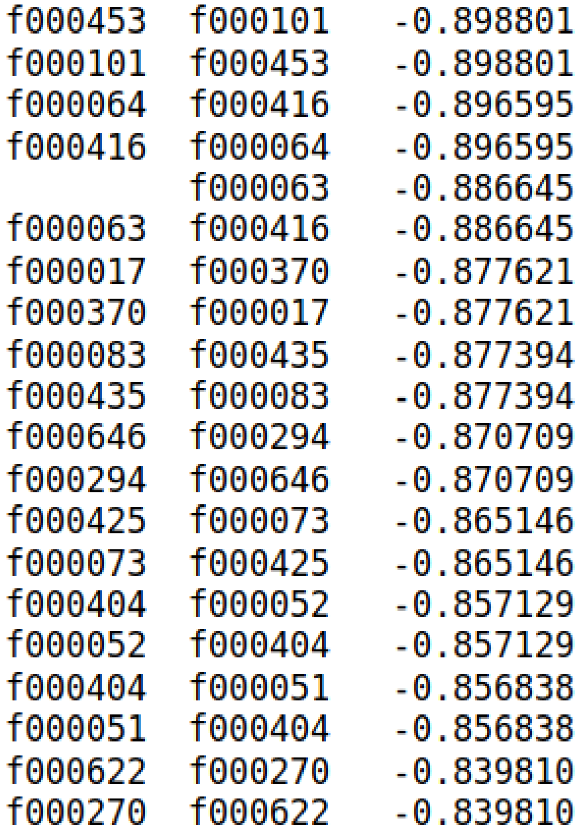
A brief look on correlation



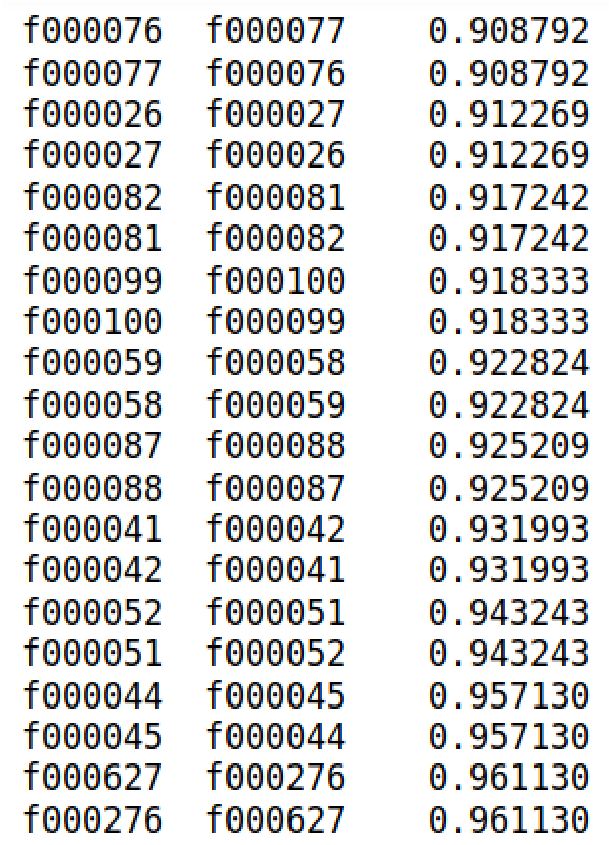
Due to less features on Speech than Music, correlation is generally lower

Correlation done on features, showing that correlation between columns nearer each other are more similar to each other than those further apart

Pearson Correlation Music

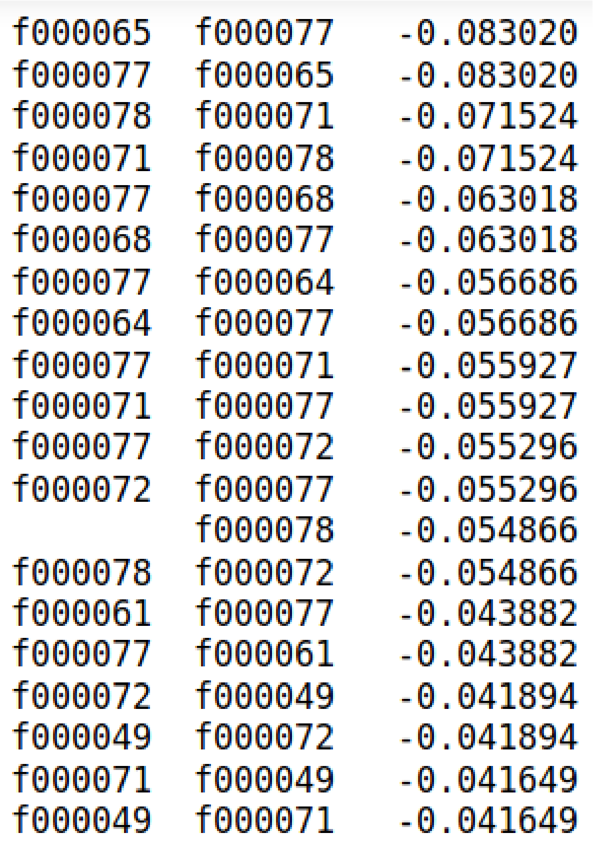


```
f000453  f000101   -0.898801
f000101  f000453   -0.898801
f000064  f000416   -0.896595
f000416  f000064   -0.896595
         f000063   -0.886645
f000063  f000416   -0.886645
f000017  f000370   -0.877621
f000370  f000017   -0.877621
f000083  f000435   -0.877394
f000435  f000083   -0.877394
f000646  f000294   -0.870709
f000294  f000646   -0.870709
f000425  f000073   -0.865146
f000073  f000425   -0.865146
f000404  f000052   -0.857129
f000052  f000404   -0.857129
f000404  f000051   -0.856838
f000051  f000404   -0.856838
f000622  f000270   -0.839810
f000270  f000622   -0.839810
```

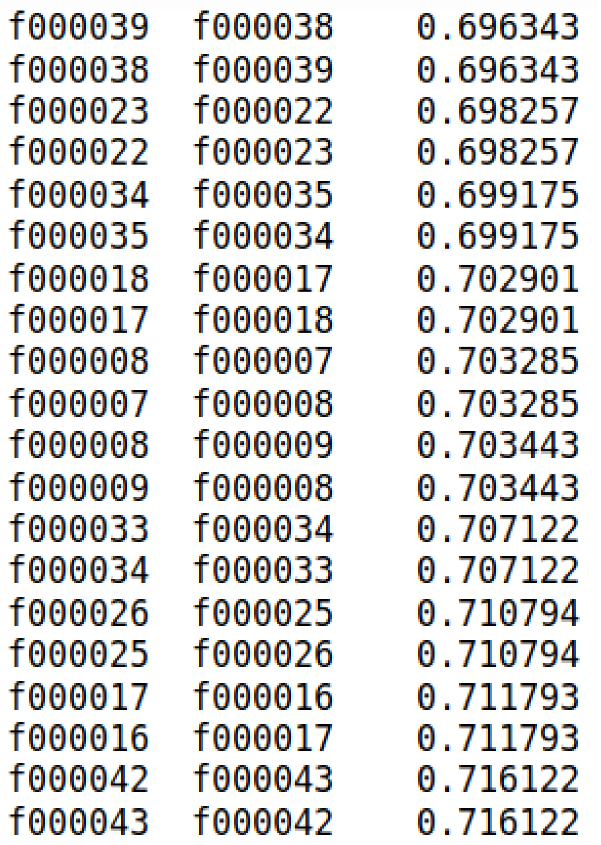


```
f000076  f000077    0.908792
f000077  f000076    0.908792
f000026  f000027    0.912269
f000027  f000026    0.912269
f000082  f000081    0.917242
f000081  f000082    0.917242
f000099  f000100    0.918333
f000100  f000099    0.918333
f000059  f000058    0.922824
f000058  f000059    0.922824
f000087  f000088    0.925209
f000088  f000087    0.925209
f000041  f000042    0.931993
f000042  f000041    0.931993
f000052  f000051    0.943243
f000051  f000052    0.943243
f000044  f000045    0.957130
f000045  f000044    0.957130
f000627  f000276    0.961130
f000276  f000627    0.961130
```

Pearson Correlation Speech



```
f000065  f000077   -0.083020
f000077  f000065   -0.083020
f000078  f000071   -0.071524
f000071  f000078   -0.071524
f000077  f000068   -0.063018
f000068  f000077   -0.063018
f000077  f000064   -0.056686
f000064  f000077   -0.056686
f000077  f000071   -0.055927
f000071  f000077   -0.055927
f000077  f000072   -0.055296
f000072  f000077   -0.055296
         f000078   -0.054866
f000078  f000072   -0.054866
f000061  f000077   -0.043882
f000077  f000061   -0.043882
f000072  f000049   -0.041894
f000049  f000072   -0.041894
f000071  f000049   -0.041649
f000049  f000071   -0.041649
```

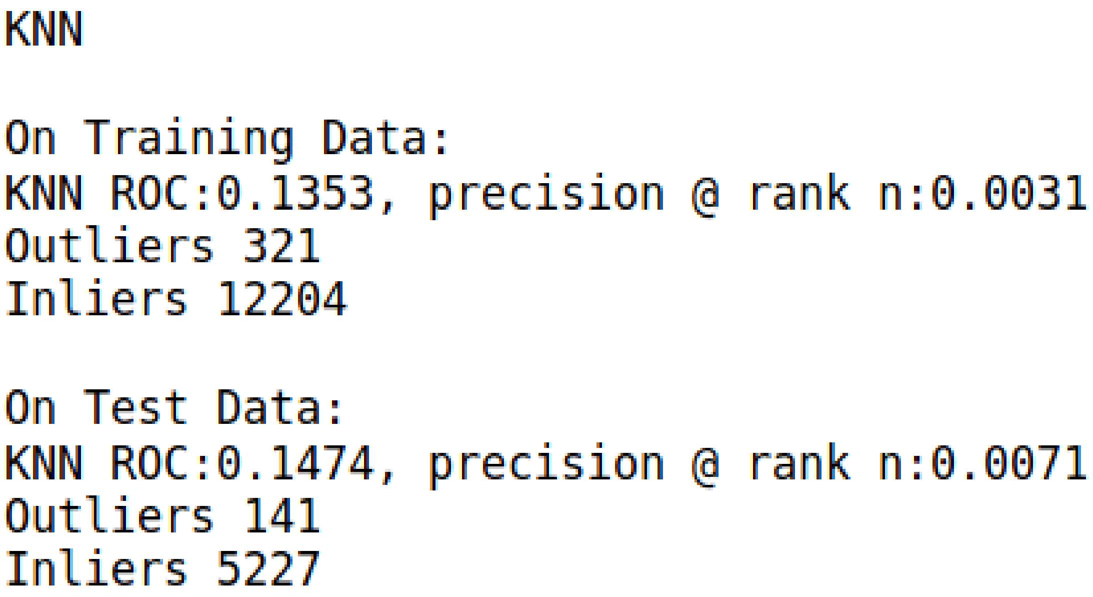


```
f000039  f000038    0.696343
f000038  f000039    0.696343
f000023  f000022    0.698257
f000022  f000023    0.698257
f000034  f000035    0.699175
f000035  f000034    0.699175
f000018  f000017    0.702901
f000017  f000018    0.702901
f000008  f000007    0.703285
f000007  f000008    0.703285
f000008  f000009    0.703443
f000009  f000008    0.703443
f000033  f000034    0.707122
f000034  f000033    0.707122
f000026  f000025    0.710794
f000025  f000026    0.710794
f000017  f000016    0.711793
f000016  f000017    0.711793
f000042  f000043    0.716122
f000043  f000042    0.716122
```

Outliers found in Data

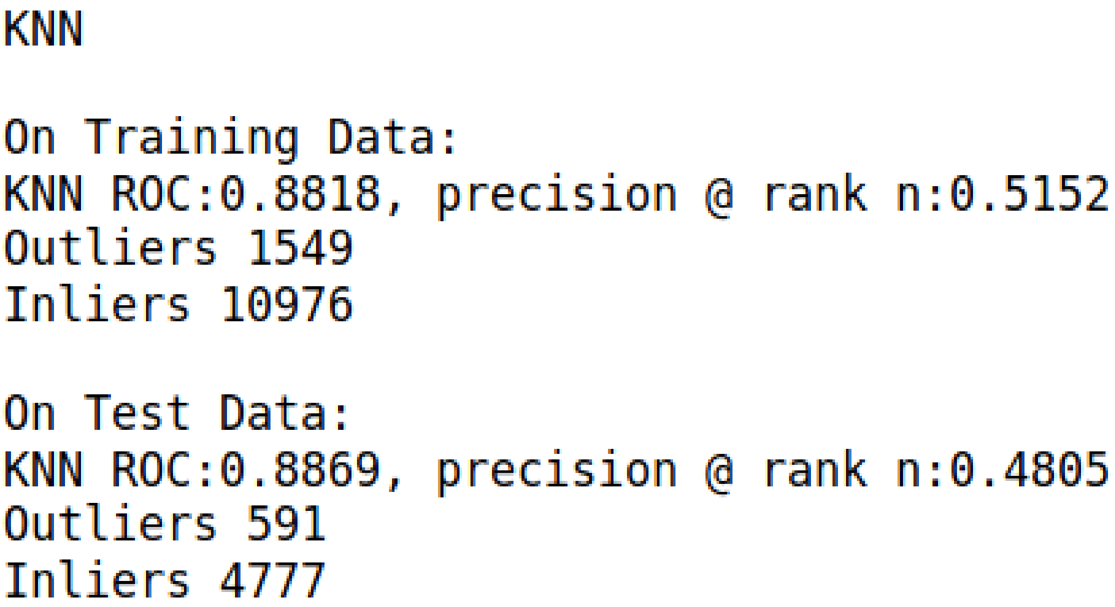
Outliers found using statistical method, with ROC as line of best fit separating

Music



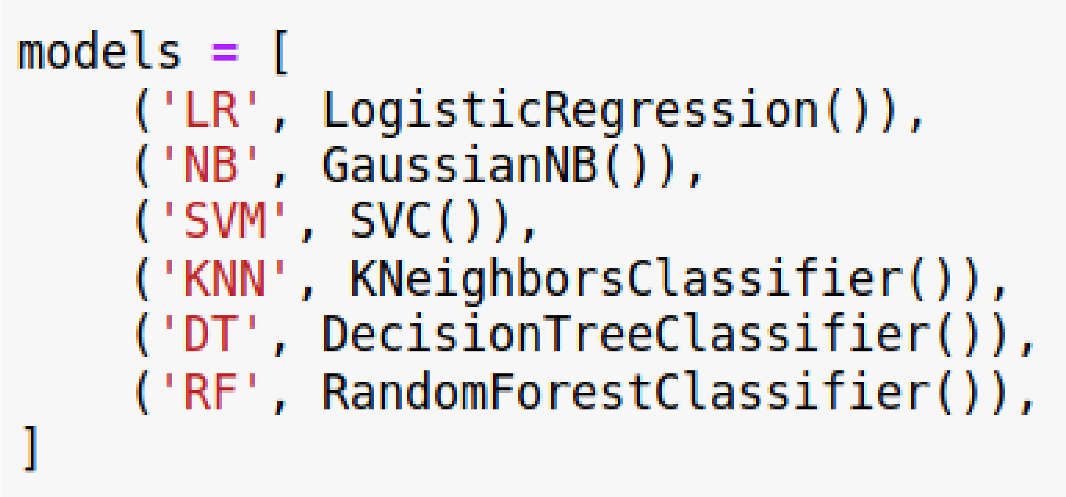
```
KNN

On Training Data:
KNN ROC:0.1353, precision @ rank n:0.0031
Outliers 321
Inliers 12204

On Test Data:
KNN ROC:0.1474, precision @ rank n:0.0071
Outliers 141
Inliers 5227
```

Speech



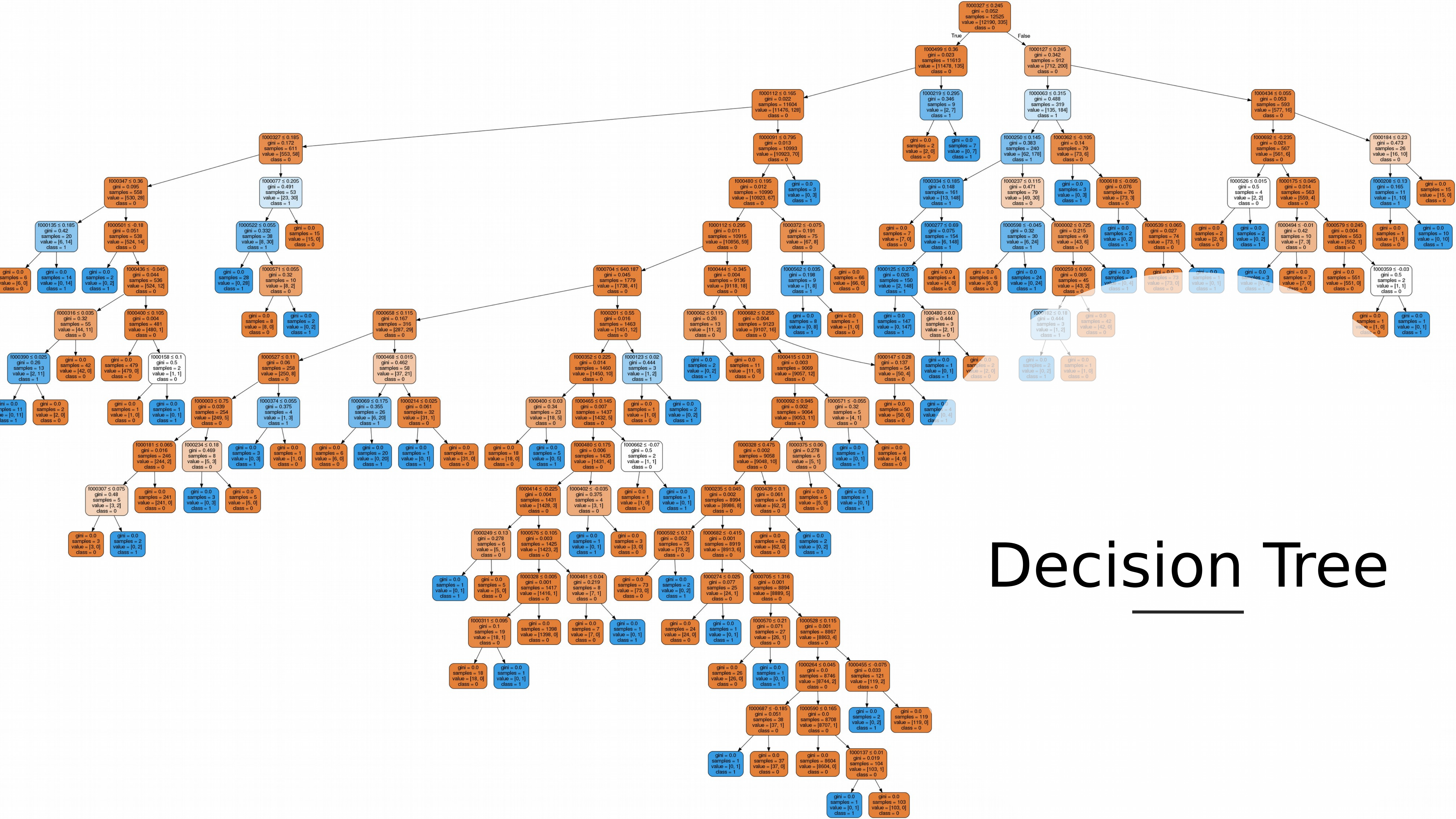
```
KNN

On Training Data:
KNN ROC:0.8818, precision @ rank n:0.5152
Outliers 1549
Inliers 10976

On Test Data:
KNN ROC:0.8869, precision @ rank n:0.4805
Outliers 591
Inliers 4777
```

Most accurate algorithm

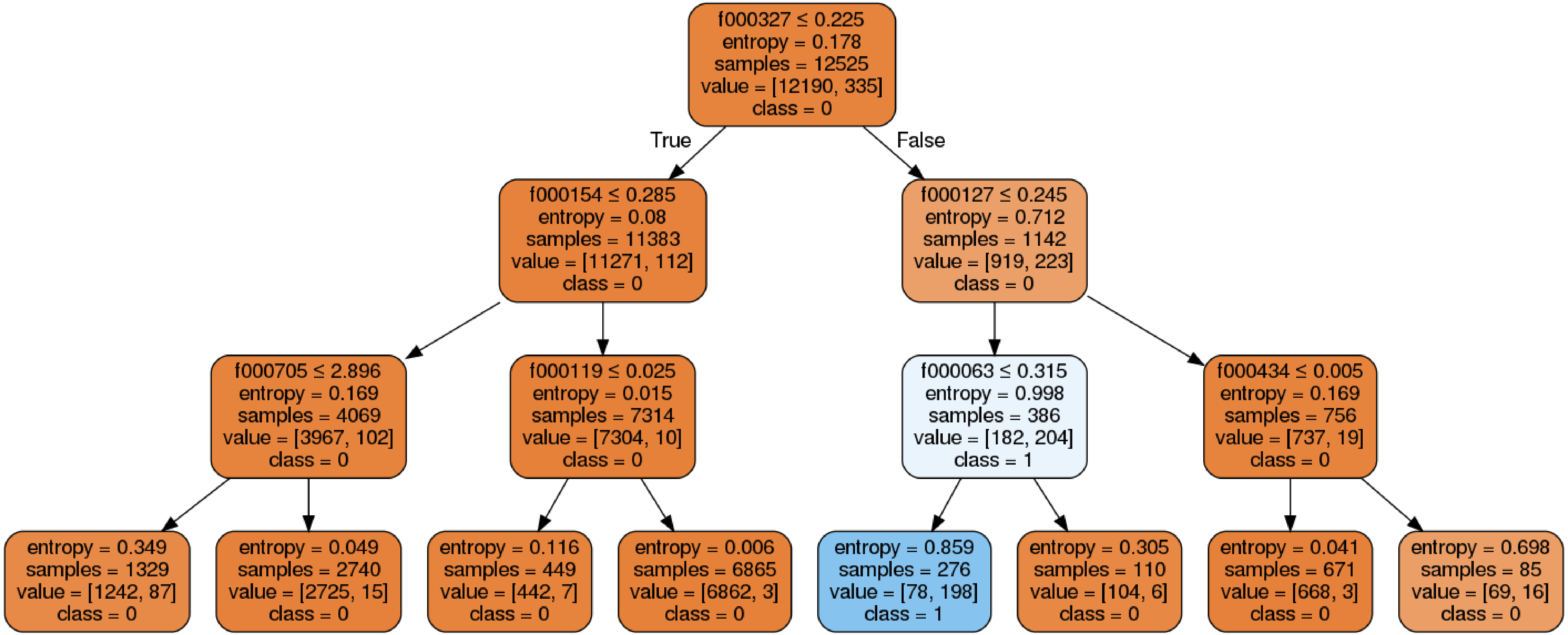


```
models = [
    ('LR', LogisticRegression()),
    ('NB', GaussianNB()),
    ('SVM', SVC()),
    ('KNN', KNeighborsClassifier()),
    ('DT', DecisionTreeClassifier()),
    ('RF', RandomForestClassifier()),
]
```

- Depending on either music or speech, but generally, logistic regression or random forest
- This is further validated with testing using feature selection, where logistic regression or random forest remains most accurate



Decision Tree



f000327 ≤ 0.225
entropy = 0.178
samples = 12525
value = [12190, 335]
class = 0

True

False

f000154 ≤ 0.285
entropy = 0.08
samples = 11383
value = [11271, 112]
class = 0

f000127 ≤ 0.245
entropy = 0.712
samples = 1142
value = [919, 223]
class = 0

f000705 ≤ 2.896
entropy = 0.169
samples = 4069
value = [3967, 102]
class = 0

f000119 ≤ 0.025
entropy = 0.015
samples = 7314
value = [7304, 10]
class = 0

f000063 ≤ 0.315
entropy = 0.998
samples = 386
value = [182, 204]
class = 1

f000434 ≤ 0.005
entropy = 0.169
samples = 756
value = [737, 19]
class = 0

entropy = 0.349
samples = 1329
value = [1242, 87]
class = 0

entropy = 0.049
samples = 2740
value = [2725, 15]
class = 0

entropy = 0.116
samples = 449
value = [442, 7]
class = 0

entropy = 0.006
samples = 6865
value = [6862, 3]
class = 0

entropy = 0.859
samples = 276
value = [78, 198]
class = 1

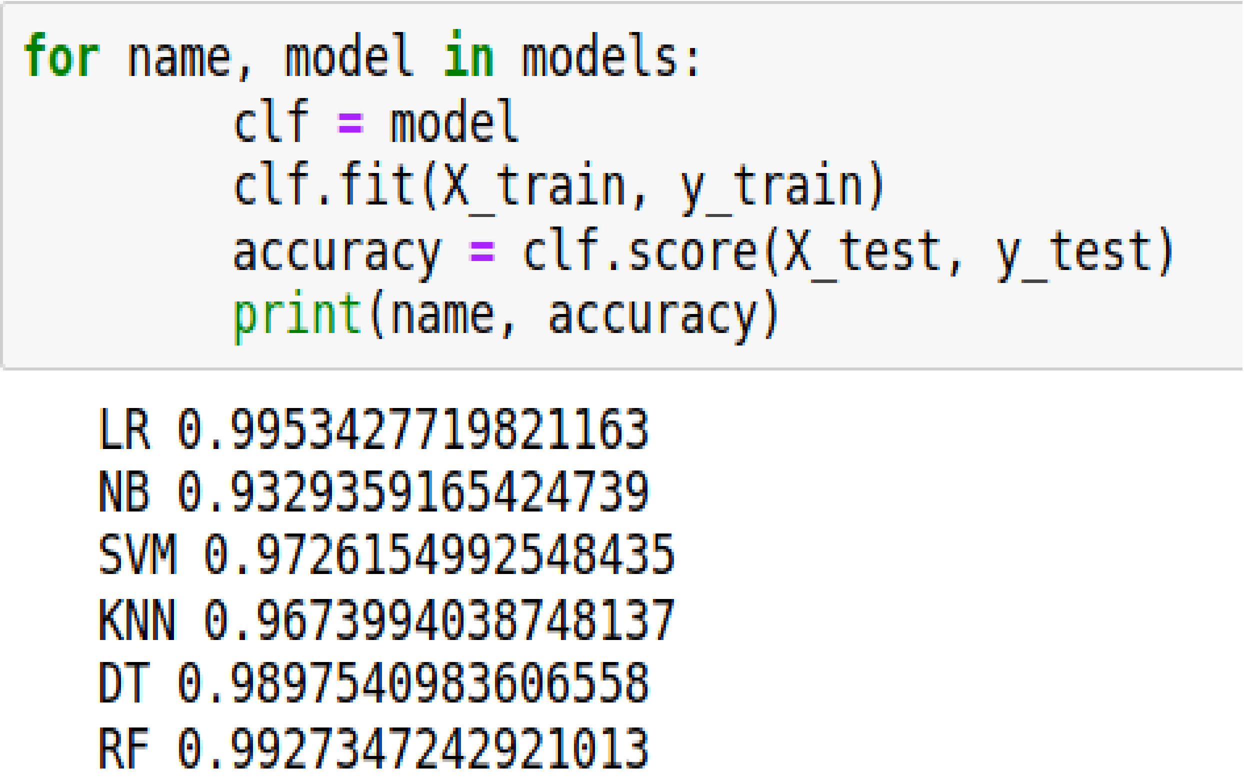
entropy = 0.305
samples = 110
value = [104, 6]
class = 0

entropy = 0.041
samples = 671
value = [668, 3]
class = 0

entropy = 0.698
samples = 85
value = [69, 16]
class = 0

Simple fitting and scores

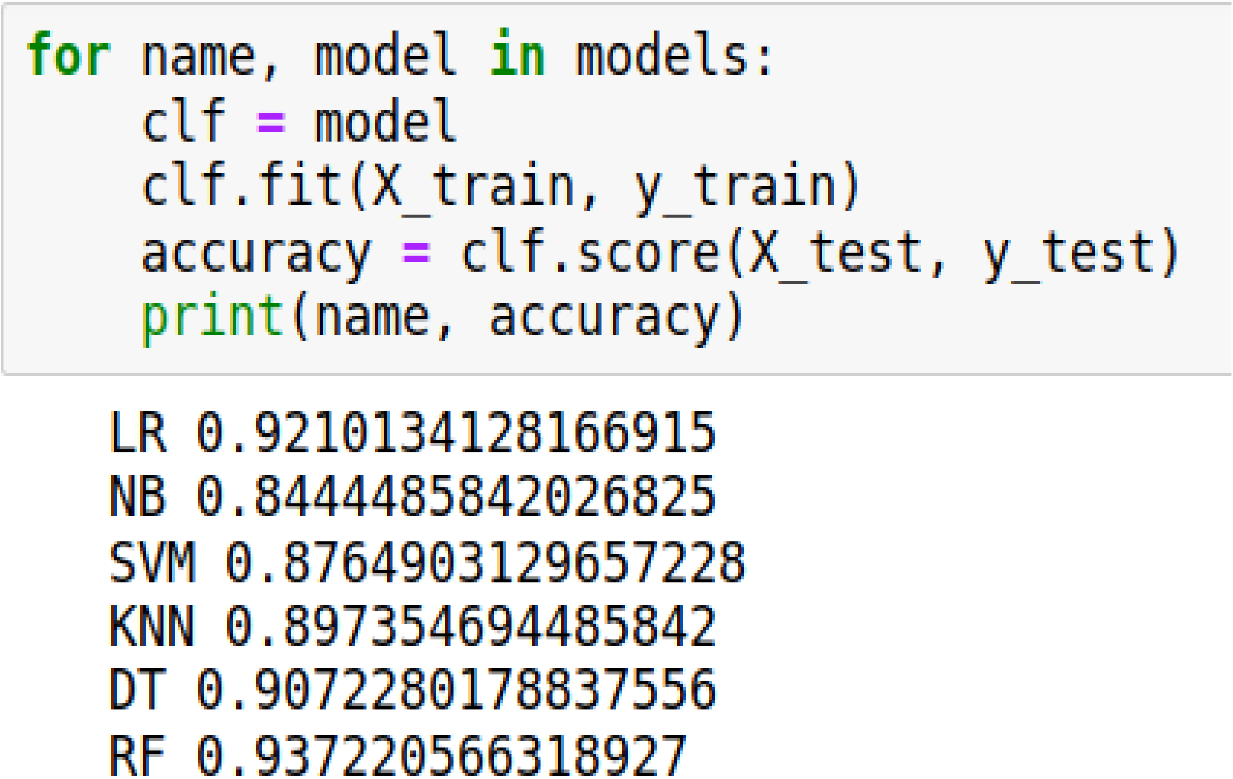
Music



```
for name, model in models:
        clf = model
        clf.fit(X_train, y_train)
        accuracy = clf.score(X_test, y_test)
        print(name, accuracy)
```

```
LR 0.9953427719821163
NB 0.9329359165424739
SVM 0.9726154992548435
KNN 0.9673994038748137
DT 0.9897540983606558
RF 0.9927347242921013
```

Speech

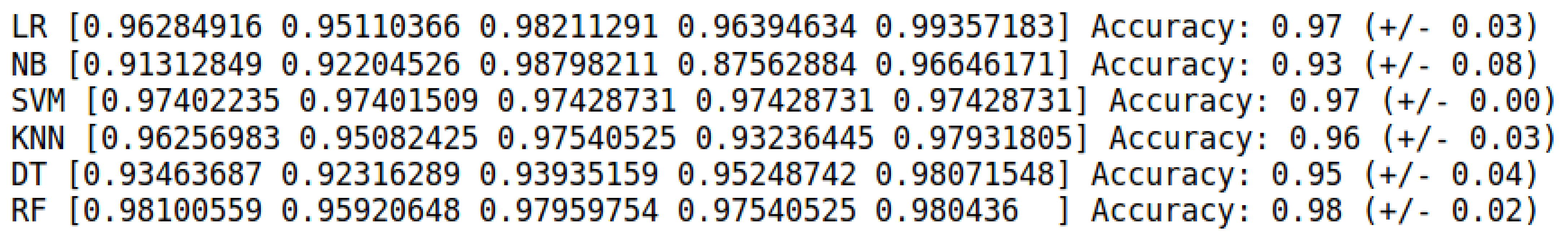


```
for name, model in models:
    clf = model
    clf.fit(X_train, y_train)
    accuracy = clf.score(X_test, y_test)
    print(name, accuracy)
```

```
LR 0.9210134128166915
NB 0.844485842026825
SVM 0.8764903129657228
KNN 0.89735469448584 2
DT 0.9072280178837556
RF 0.937220566318927
```

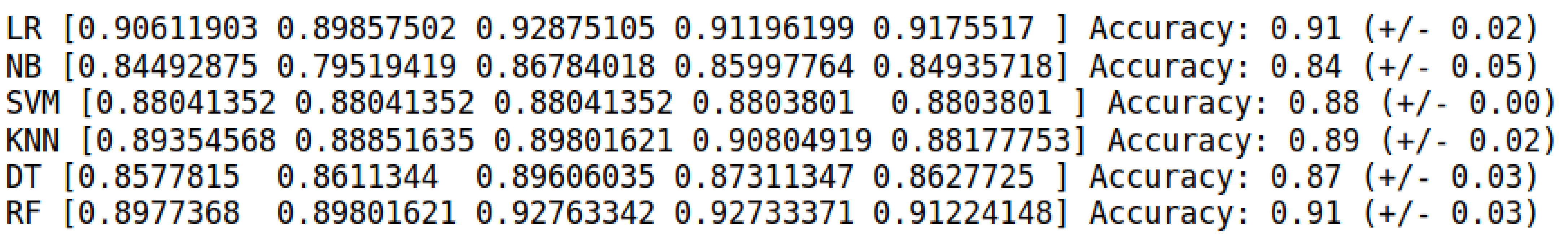
Cross validation and Average of accuracy

Music

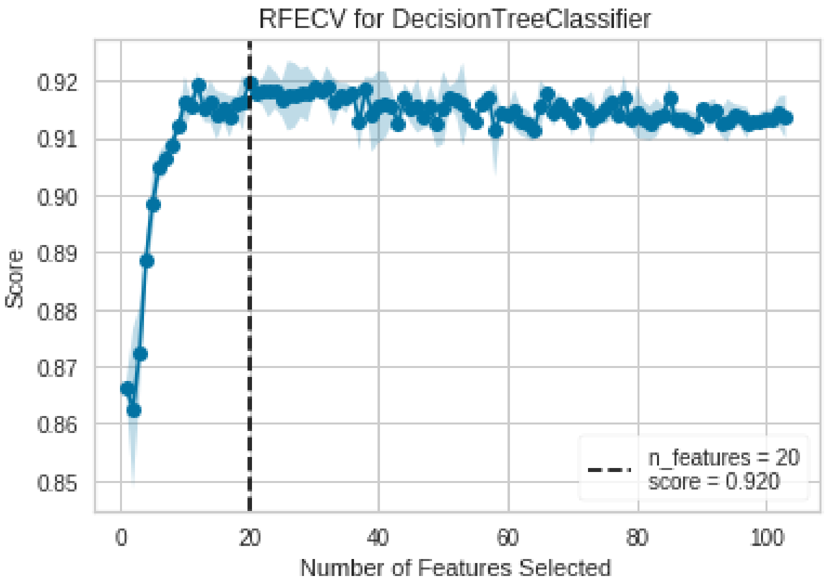


```
LR [0.96284916 0.95110366 0.98211291 0.96394634 0.99357183] Accuracy: 0.97 (+/- 0.03)
NB [0.91312849 0.92204526 0.98798211 0.87562884 0.96646171] Accuracy: 0.93 (+/- 0.08)
SVM [0.97402235 0.97401509 0.97428731 0.97428731 0.97428731] Accuracy: 0.97 (+/- 0.00)
KNN [0.96256983 0.95082425 0.97540525 0.93236445 0.97931805] Accuracy: 0.96 (+/- 0.03)
DT [0.93463687 0.92316289 0.93935159 0.95248742 0.98071548] Accuracy: 0.95 (+/- 0.04)
RF [0.98100559 0.95920648 0.97959754 0.97540525 0.980436  ] Accuracy: 0.98 (+/- 0.02)
```

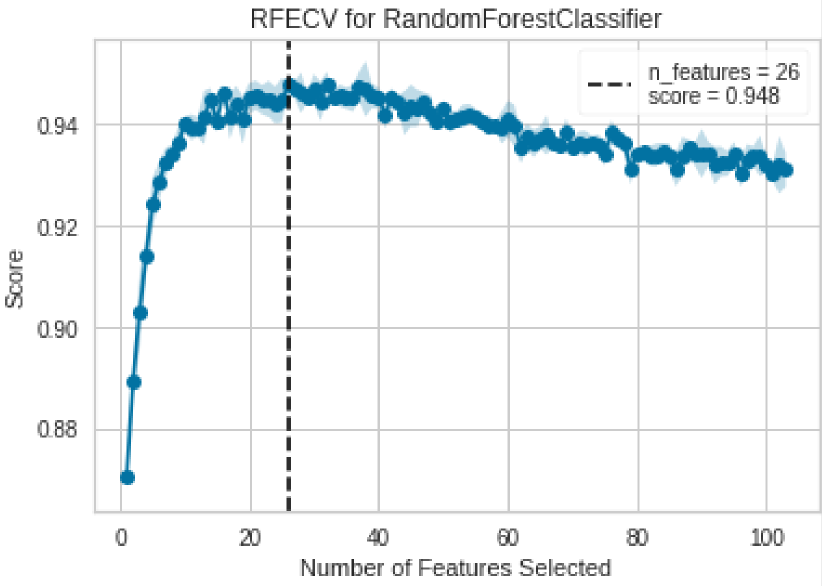
Speech



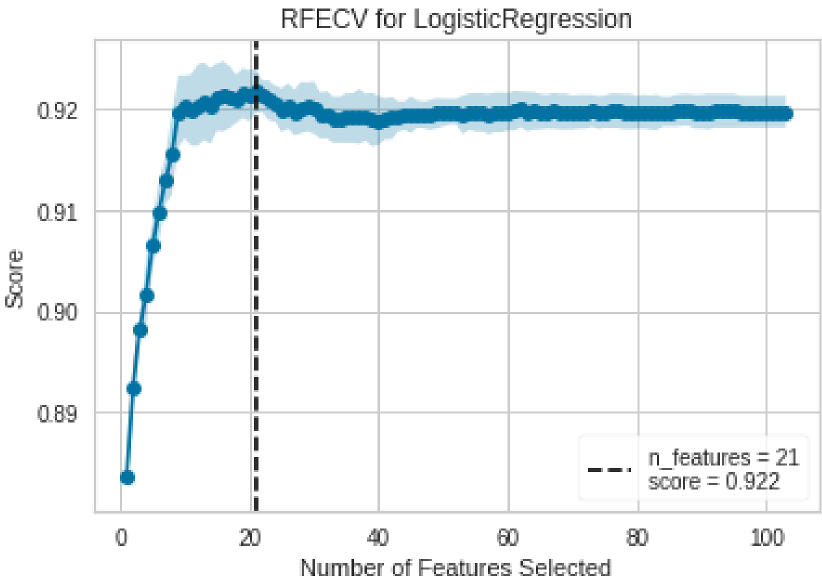
```
LR [0.90611903 0.89857502 0.92875105 0.91196199 0.9175517 ] Accuracy: 0.91 (+/- 0.02)
NB [0.84492875 0.79519419 0.86784018 0.85997764 0.84935718] Accuracy: 0.84 (+/- 0.05)
SVM [0.88041352 0.88041352 0.88041352 0.8803801  0.8803801 ] Accuracy: 0.88 (+/- 0.00)
KNN [0.89354568 0.88851635 0.89801621 0.90804919 0.88177753] Accuracy: 0.89 (+/- 0.02)
DT [0.8577815  0.8611344  0.89606035 0.87311347 0.8627725 ] Accuracy: 0.87 (+/- 0.03)
RF [0.8977368  0.89801621 0.92763342 0.92733371 0.91224148] Accuracy: 0.91 (+/- 0.03)
```



RFECV for DecisionTreeClassifier



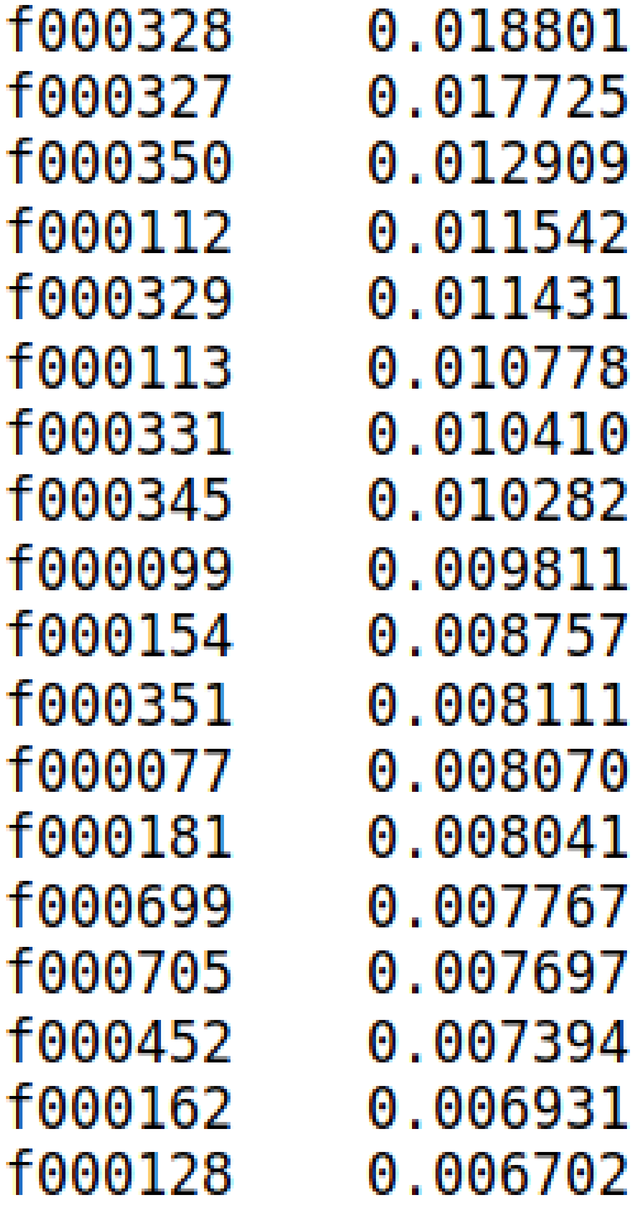
RFECV for RandomForestClassifier



RFECV for LogisticRegression

Recursive Feature Elimination Cross Validation – Speech

Feature importance Random forest - music

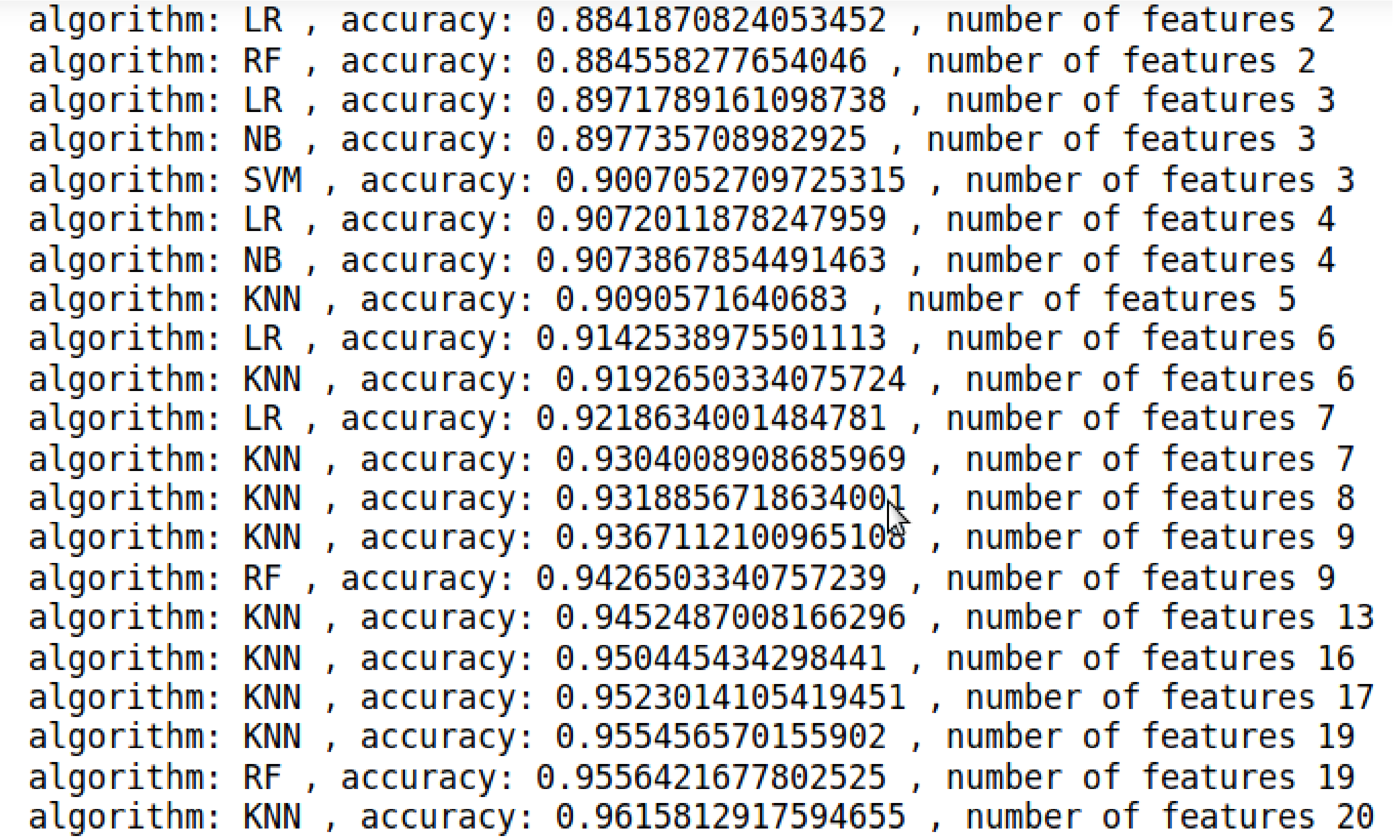


```
f000328    0.018801
f000327    0.017725
f000350    0.012909
f000112    0.011542
f000329    0.011431
f000113    0.010778
f000331    0.010410
f000345    0.010282
f000099    0.009811
f000154    0.008757
f000351    0.008111
f000077    0.008070
f000181    0.008041
f000699    0.007767
f000705    0.007697
f000452    0.007394
f000162    0.006931
f000128    0.006702
```

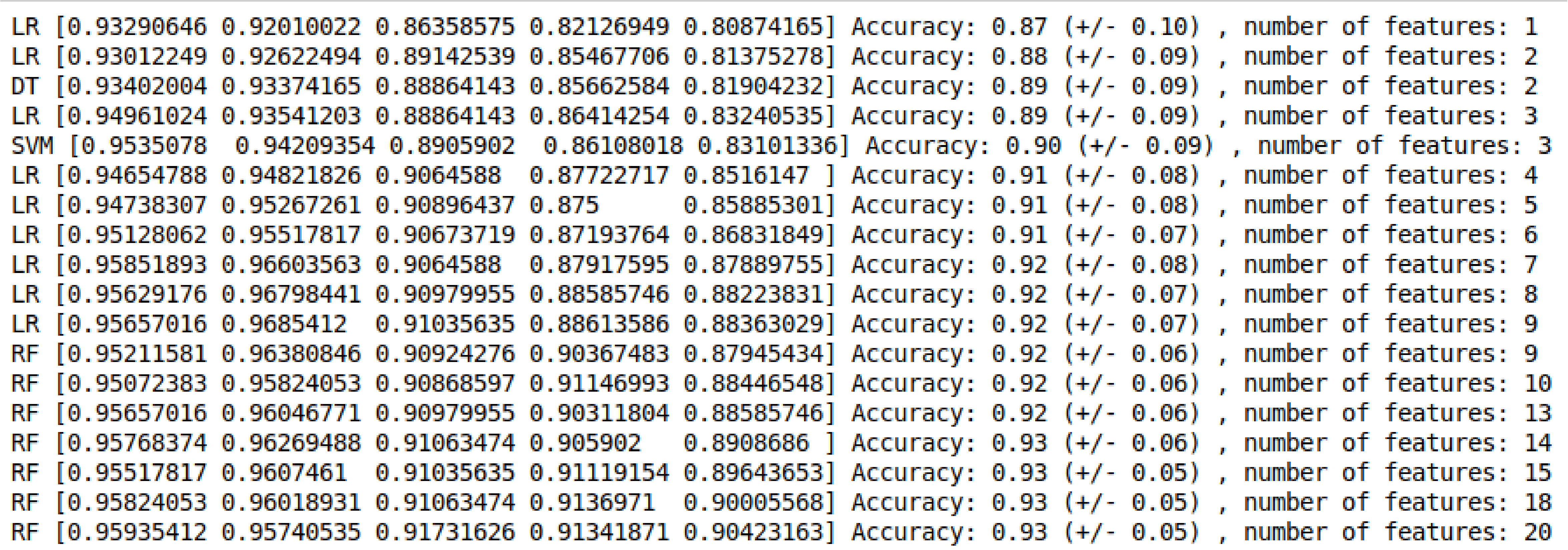
Overfitting KNeighbors - speech

Simple fitting and prediction

When trying to best feature, initial results seem to imply that K-neighbors was most accurate algorithm, however, further observations using cross validating reveals it was overfitted, lowering it’s score.



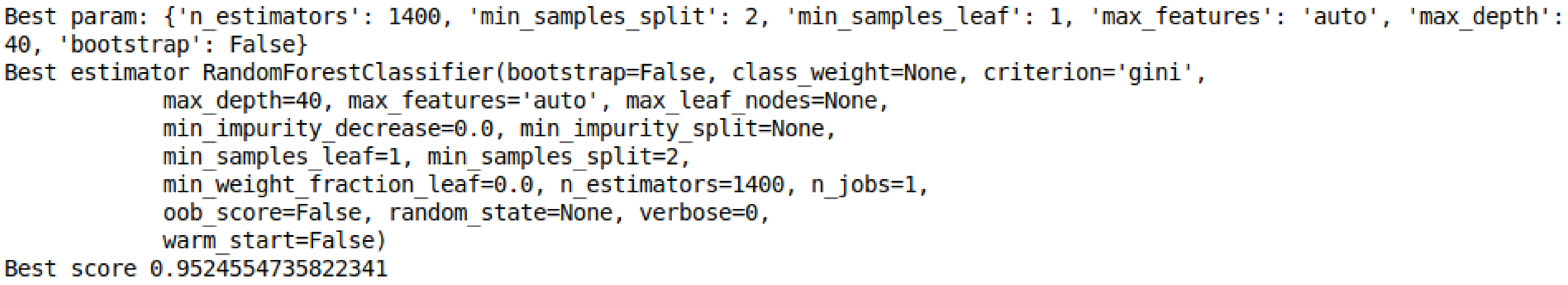
```
algorithm: LR , accuracy: 0.8841870824053452 , number of features 2
algorithm: RF , accuracy: 0.884558277654046 , number of features 2
algorithm: LR , accuracy: 0.8971789161098738 , number of features 3
algorithm: NB , accuracy: 0.897735708982925 , number of features 3
algorithm: SVM , accuracy: 0.9007052709725315 , number of features 3
algorithm: LR , accuracy: 0.9072011878247959 , number of features 4
algorithm: NB , accuracy: 0.9073867854491463 , number of features 4
algorithm: KNN , accuracy: 0.9090571640683 , number of features 5
algorithm: LR , accuracy: 0.9142538975501113 , number of features 6
algorithm: KNN , accuracy: 0.9192650334075724 , number of features 6
algorithm: LR , accuracy: 0.9218634001484781 , number of features 7
algorithm: KNN , accuracy: 0.9304008908685969 , number of features 7
algorithm: KNN , accuracy: 0.9318856718634001 , number of features 8
algorithm: KNN , accuracy: 0.9367112100965108 , number of features 9
algorithm: RF , accuracy: 0.9426503340757239 , number of features 9
algorithm: KNN , accuracy: 0.9452487008166296 , number of features 13
algorithm: KNN , accuracy: 0.950445434298441 , number of features 16
algorithm: KNN , accuracy: 0.9523014105419451 , number of features 17
algorithm: KNN , accuracy: 0.955456570155902 , number of features 19
algorithm: RF , accuracy: 0.9556421677802525 , number of features 19
algorithm: KNN , accuracy: 0.9615812917594655 , number of features 20
```



```
LR [0.93290646 0.92010022 0.86358575 0.82126949 0.80874165] Accuracy: 0.87 (+/- 0.10) , number of features: 1
LR [0.93012249 0.92622494 0.89142539 0.85467706 0.81375278] Accuracy: 0.88 (+/- 0.09) , number of features: 2
DT [0.93402004 0.93374165 0.88864143 0.85662584 0.81904232] Accuracy: 0.89 (+/- 0.09) , number of features: 2
LR [0.94961024 0.93541203 0.88864143 0.86414254 0.83240535] Accuracy: 0.89 (+/- 0.09) , number of features: 3
SVM [0.9535078  0.94209354 0.8905902  0.86108018 0.83101336] Accuracy: 0.90 (+/- 0.09) , number of features: 3
LR [0.94654788 0.94821826 0.9064588  0.87722717 0.8516147 ] Accuracy: 0.91 (+/- 0.08) , number of features: 4
LR [0.94738307 0.95267261 0.90896437 0.875      0.85885301] Accuracy: 0.91 (+/- 0.08) , number of features: 5
LR [0.95128062 0.95517817 0.90673719 0.87193764 0.86831849] Accuracy: 0.91 (+/- 0.07) , number of features: 6
LR [0.95851893 0.96603563 0.9064588  0.87917595 0.87889755] Accuracy: 0.92 (+/- 0.08) , number of features: 7
LR [0.95629176 0.96798441 0.90979955 0.88585746 0.88223831] Accuracy: 0.92 (+/- 0.07) , number of features: 8
LR [0.95657016 0.9685412  0.91035635 0.88613586 0.88363029] Accuracy: 0.92 (+/- 0.07) , number of features: 9
RF [0.95211581 0.96380846 0.90924276 0.90367483 0.87945434] Accuracy: 0.92 (+/- 0.06) , number of features: 9
RF [0.95072383 0.95824053 0.90868597 0.91146993 0.88446548] Accuracy: 0.92 (+/- 0.06) , number of features: 10
RF [0.95657016 0.96046771 0.90979955 0.90311804 0.88585746] Accuracy: 0.92 (+/- 0.06) , number of features: 13
RF [0.95768374 0.96269488 0.91063474 0.905902   0.8908686 ] Accuracy: 0.93 (+/- 0.06) , number of features: 14
RF [0.95517817 0.9607461  0.91035635 0.91119154 0.89643653] Accuracy: 0.93 (+/- 0.05) , number of features: 15
RF [0.95824053 0.96018931 0.91063474 0.9136971  0.90005568] Accuracy: 0.93 (+/- 0.05) , number of features: 18
RF [0.95935412 0.95740535 0.91731626 0.91341871 0.90423163] Accuracy: 0.93 (+/- 0.05) , number of features: 20
```

With a Cross validation K Neighbors does not have the best accuracy

Random Forest:



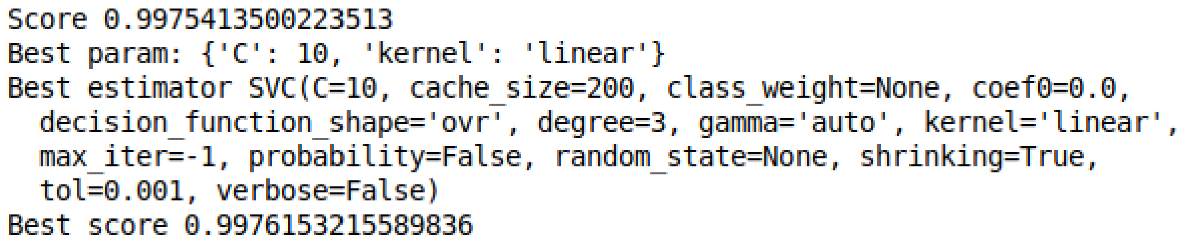
```
Best param: {'n_estimators': 1400, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth':
40, 'bootstrap': False}
Best estimator RandomForestClassifier(bootstrap=False, class_weight=None, criterion='gini',
            max_depth=40, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=1400, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
Best score 0.9524554735822341
```

Variations in classification accuracy based on parameters

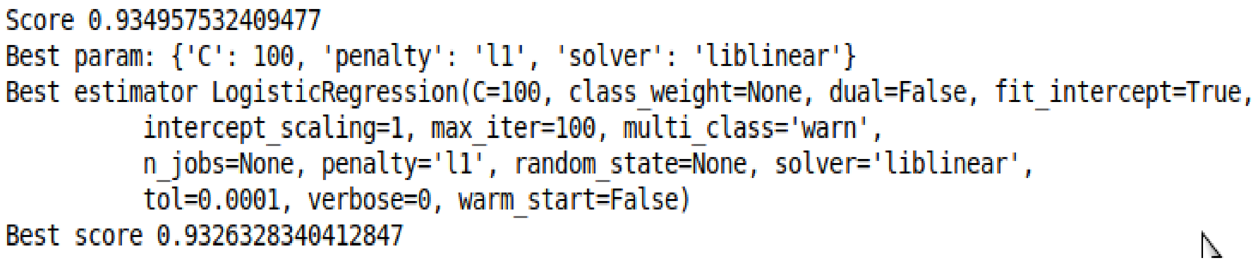
- Through hyperparameter tuning, accuracy of all algorithms are improved.
  - Some notable improvements include Random forest, which saw marked increase of up to 5%

Hyperparameter Tuning - Speech

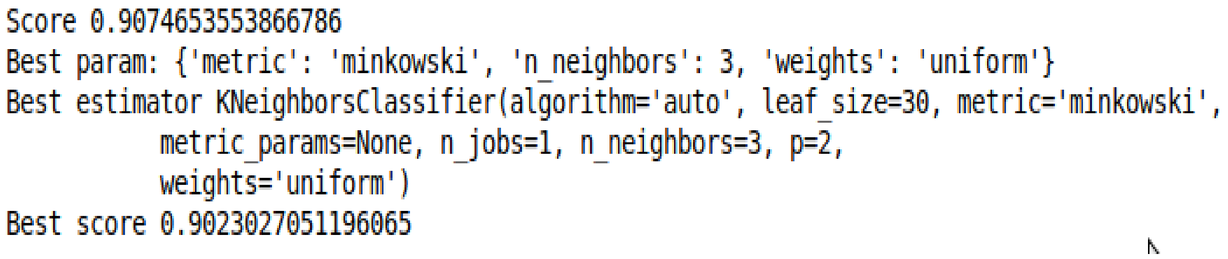
SVC, K-Neighbors, Logistic Regression, Decision Tree



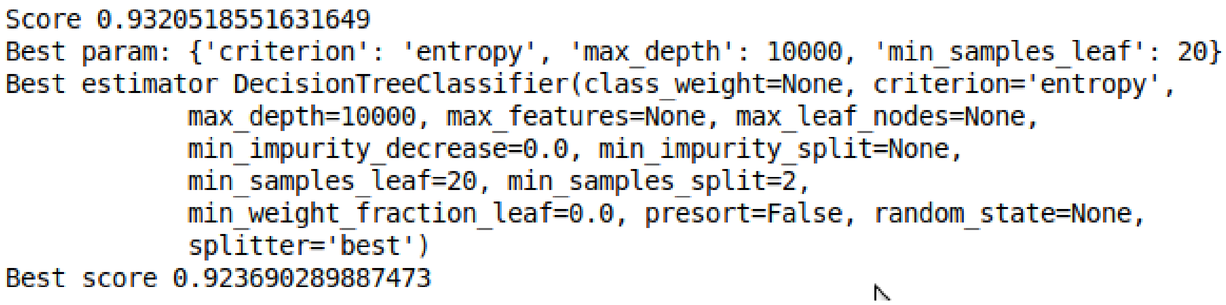
```
Score 0.99754135002235l3
Best param: {'C': 10, 'kernel': 'linear'}
Best estimator SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Best score 0.9976153215589836
```



```
Score 0.934957532409477
Best param: {'C': 100, 'penalty': 'l1', 'solver': 'liblinear'}
Best estimator LogisticRegression(C=100, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=None, solver='liblinear',
          tol=0.0001, verbose=0, warm_start=False)
Best score 0.9326328340412847
```



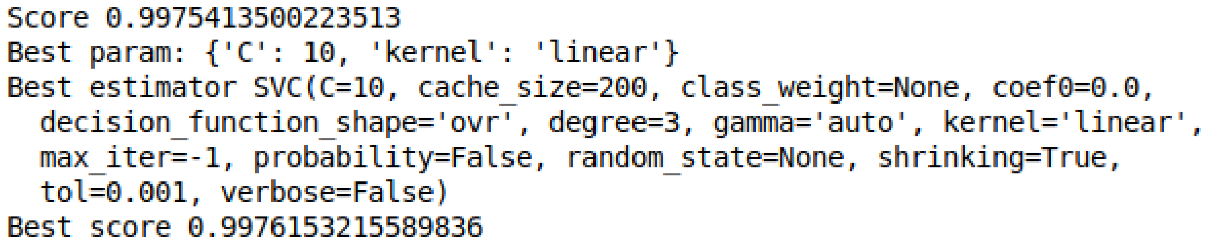
```
Score 0.9074653553866786
Best param: {'metric': 'minkowski', 'n_neighbors': 3, 'weights': 'uniform'}
Best estimator KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')
Best score 0.9023027051196065
```



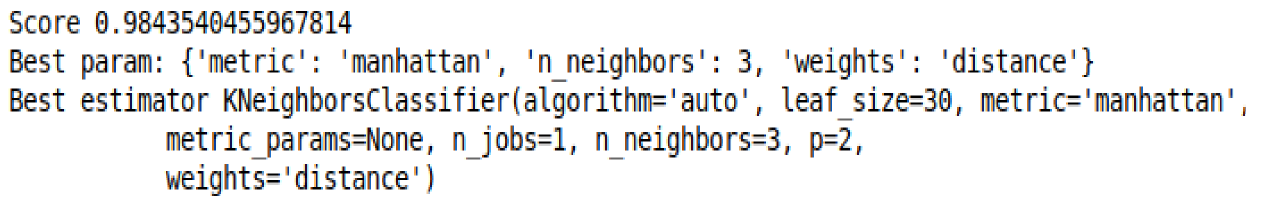
```
Score 0.9320518551631649
Best param: {'criterion': 'entropy', 'max_depth': 10000, 'min_samples_leaf': 20}
Best estimator DecisionTreeClassifier(class_weight=None, criterion='entropy',
            max_depth=10000, max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=20, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
Best score 0.923690289887473
```

Hyperparameter Tuning - Music

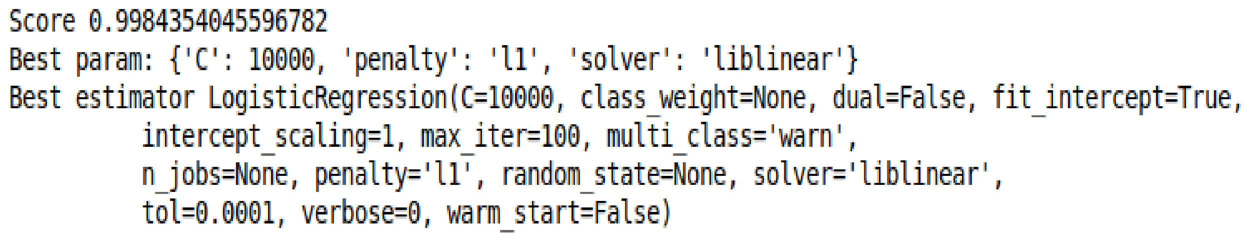
SVC, K-Neighbors, Logistic Regression, Decision Tree, Random Forest



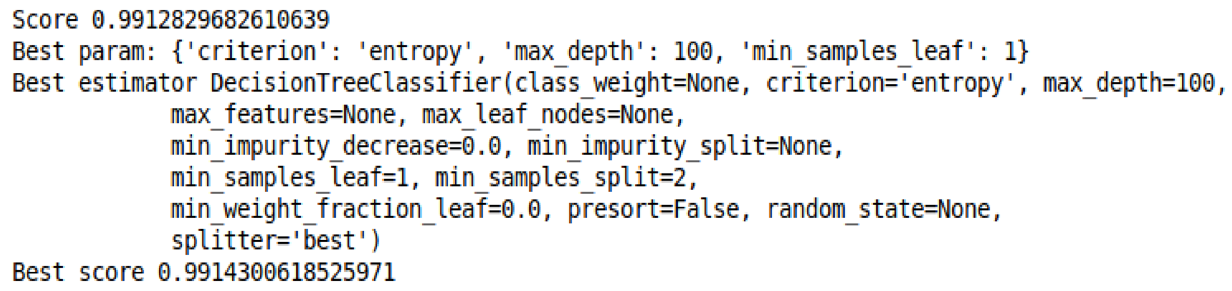
```
Score 0.9975413500223513
Best param: {'C': 10, 'kernel': 'linear'}
Best estimator SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Best score 0.9976153215589836
```



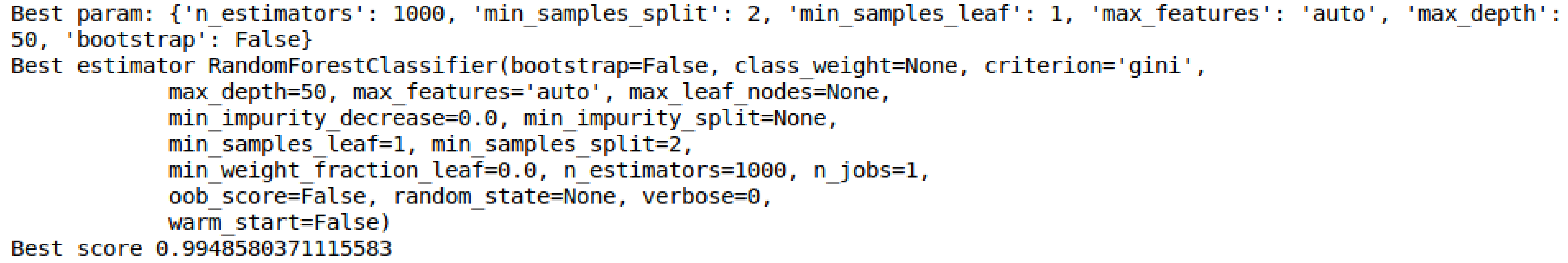
```
Score 0.9843540455967814
Best param: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Best estimator KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='manhattan',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='distance')
```



```
Score 0.9984354045596782
Best param: {'C': 10000, 'penalty': 'l1', 'solver': 'liblinear'}
Best estimator LogisticRegression(C=10000, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=None, solver='liblinear',
          tol=0.0001, verbose=0, warm_start=False)
```



```
Score 0.9912829682610639
Best param: {'criterion': 'entropy', 'max_depth': 100, 'min_samples_leaf': 1}
Best estimator DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=100,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
Best score 0.9914300618525971
```

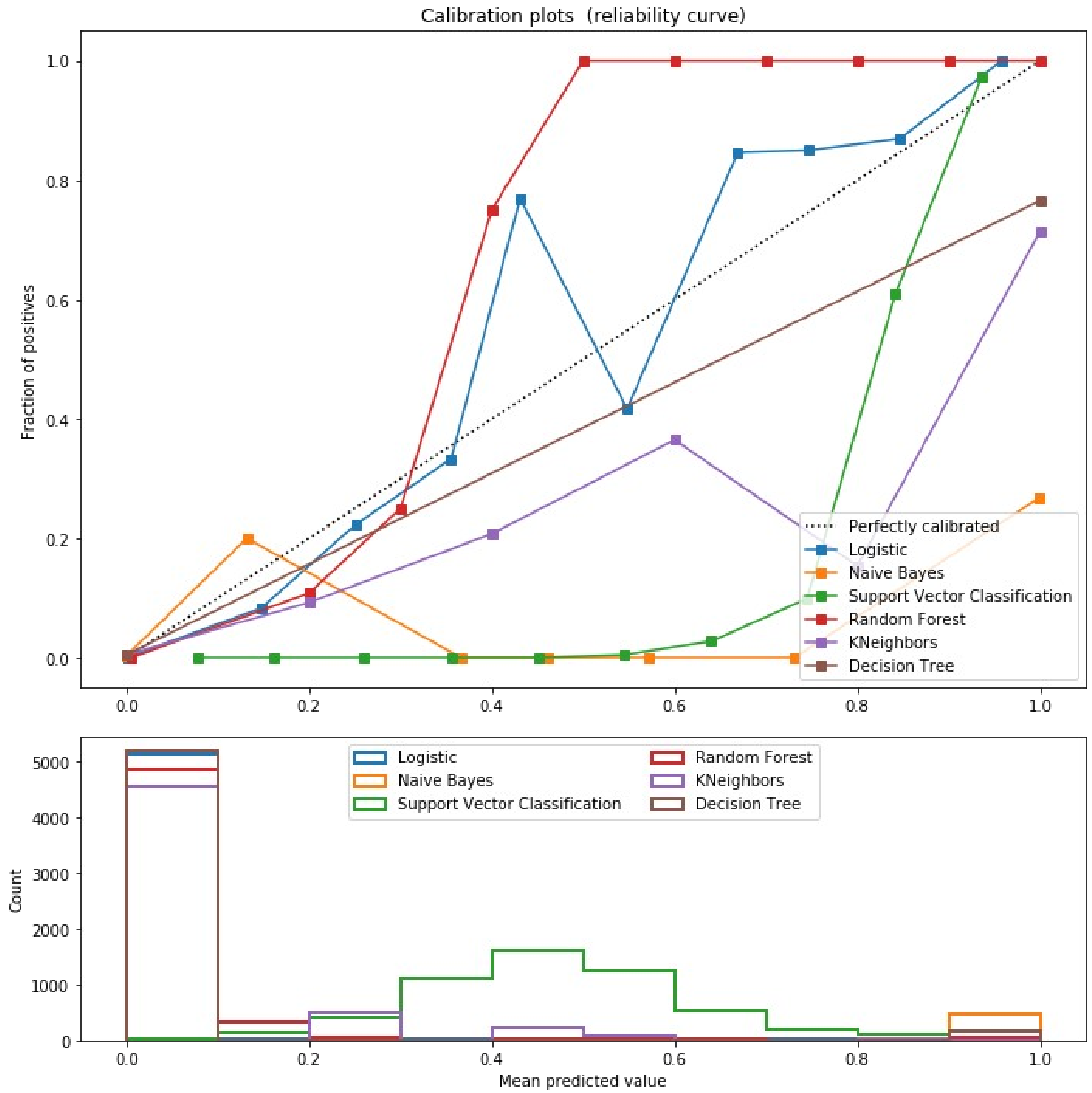


```
Best param: {'n_estimators': 1000, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth':
50, 'bootstrap': False}
Best estimator RandomForestClassifier(bootstrap=False, class_weight=None, criterion='gini',
            max_depth=50, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=1000, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
Best score 0.9948580371115583
```

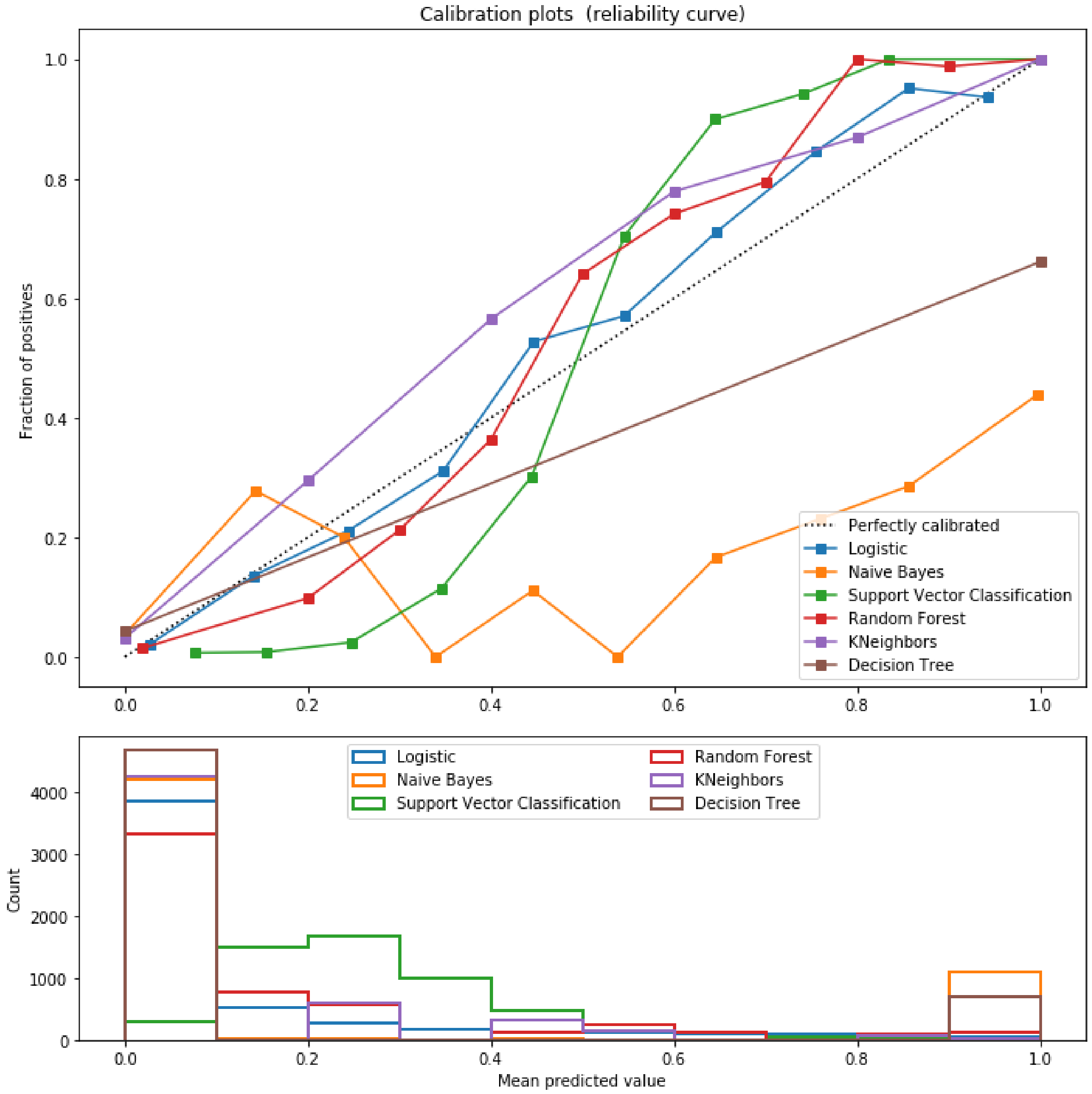
Calibration of Classifiers

- A side exercise done for the practical was to compare the calibrations of the classifiers.
- As can be observed, logistic regression is the closest to the perfect calibration line. Might be due to logistic regression directly optimizing log-loss, whilst other classifiers gave more biased probabilities

Music



Speech Calibration Plots


Calibration plots (reliability curve)
Fraction of positives
Perfectly calibrated
Logistic
Naive Bayes
Support Vector Classification
Random Forest
KNeighbors
Decision Tree
Count
Mean predicted value