Report on mRMR

By: Xiao PAN, Fubang ZHAO and Xiangnan YUE
02/04/2018

Part 1. Introduction to the task

Our task is to select the Max-Dependency features using the mRMR method described in the paper. In this part, we will give an introduction on the two main tasks of our implementation: the calculation of mutual information and the implementation of the first-order incremental selection which is described in the paper.

1.1 The calculation of Mutual Information

The most important step in the feature selection algorithm is to calculate the mutual information between target and one feature and between two features. Mutual information is a perfect statistic for measuring the degree of relatedness between two random variables. The standard formula of mutual information is given by

$$I(x;y) = \iint p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\,dxdy.$$

However, in the practical case, since we only have finite instance of the random variables, we can't have the exact distribution of the two random variables. Therefore, in order to extract the mutual information from the finite data set, we proposed some method for computing an approximated mutual information.

We consider both discrete(categorical) and continuous data.

1. Mutual information between discrete and continuous data can be calculated using Nearest Neighbor method. [1]
2. Mutual information between two continuous data can be calculated using the formula $I(X,Y) = H(X) + H(Y) - H(X,Y)$
3. Mutual information between two discrete data can be calculated by replacing the theoretical probability by the frequency counted on the data set.

Illustration of the k Nearest Neighbor Method:
To compute $I(X,Y)$, where X is a categorical variable and y is a continuous variable.

Notations:

- N denotes the total number of samples
- X_k denotes the k-th category of the variable X.
- $N_{k(i)}$ denotes the total number of samples the category of point i.
- m_i denotes the number of nearest neighbor of point i in the continuous variable y.
- d_i denotes the distance from point i to the k-th neighbor in the category of point i.

Then the mutual information

$$I_i = \psi(N) - \psi(N_{k(i)}) + \psi(k) - \psi(m_i),$$

And we take average over all data points

$$I(X, Y) = \frac{1}{N}\sum_i I_i = \psi(N) - \frac{1}{N}\sum_i \psi(N_{k(i)}) + \psi(k) - \frac{1}{N}\sum_i \psi(m_i)$$

In our implementation, the variable k is some fixed small integer of user's choice. Too small k will lead to sampling error; too high k will lead to high coarse-graining error.

1.2 The implementation of First-Order Incremental Selection Based on Mutual Information
In the paper the authors have proved the equivalence between Max-Dependency and mRMR in the first-order incremental searching case. In the implementation we also use the first-order incremental search. In the following we give the algorithm that we use in the coding.

Notations:

- X : the feature matrix (n samples, p features), X_i : the i-th feature (column) of X
- y : the target
- S : the selected set of features
- F : the set of features which are not selected yet

Algorithm:

```
Input: k_min, k_max, nb_features

        # k varies in the range from k_min to k_max.
        # nb_features is the number of features to select from all the features.

Initialization:
XYMI: a vector of shape $p$
        # used to store the mutual information between feature and target
XXMI: a matrix of shape $p \times p$
        # used to store the mutual information between two features
S: empty list
        # used to store the indexes of the selected features
F: list of all indexes features
        # used to store the indexes of the non-selected features

Step 1:
For every feature $X_i$ ,compute the mutual information $I(X_i, Y)$ using Nearest Neighbor
Method, store the value in XYMI [i].
$S_1 = argmax\ XYMI$
$S.\ add(S_1)$
$F.\ remove(S_1)$
```

```
Step 2:
Iterate until size of S reach nb) features:
    For the last selected feature $S_{last}$ , compute the mutual information $I(X_{S_{last}}, X_j)$ for
    $j \in F$ , store the value in XXMI [$S_{last}, j$]

    Compute the average of each column of XXMI, we get the index of the column
    with the maximum mean.
    $S_{new} = argmax\ XXMI[:, k]$
    $S.\ add(S_{new})$
    $F.\ remove(S_{new})$

Output: $S$
```

Part 2 Description of the Data Set

We will test our algorithm on several data sets from different sources: (i) UCI Arrhythmia; (ii) UCI Automobile; (iii)

2.1 Data Sets -- an overview

(1) Randomly Generated data
Firstly, we applied our algorithm and the selectKBest method of sklearn to the simulation data generated randomly by using the library of make_classification of sklearn. This is a good way to test the algorithm firstly before applying to the real data.

(2) UCI Arrhythmia
https://archive.ics.uci.edu/ml/datasets/arrhythmia
The object of the task is to distinguish between the presence and absence of cardiac arrhythmia and to classify them in one of 16 groups. The difficulties lay on the fact that some groups have very little samples and some has even no sample. The algorithm has to deal with the extreme and unbalanced case. The label y is categorical and it consists of our experiment for discrete y and continuous x.

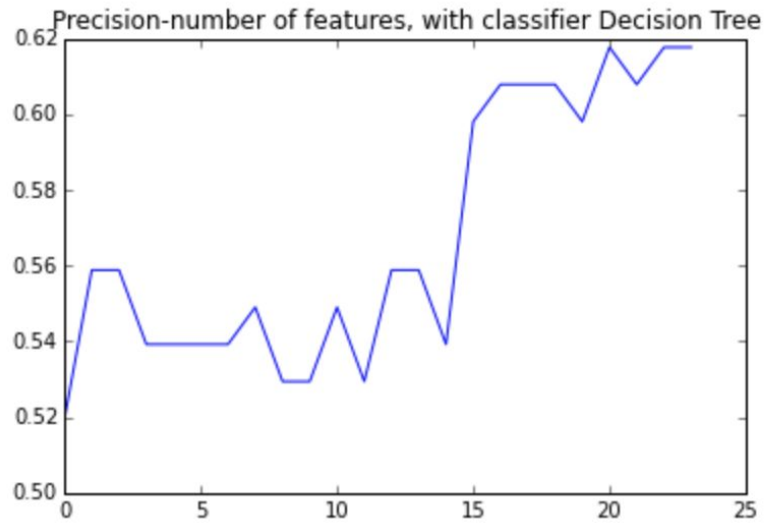
(3) Tianchi Diabetes / Blood Sugar Prediction
With 42 features, the object is to predict the patient's blood sugar percentage. We list this data source as a comparison between Lasso and our feature selector with linear-regression.

2.2 Model Selection Pipeline

We first gave an example for the data set UCI Arrhythmia, then we experimented on the Tianchi Diabetes / Blood Sugar Prediction.

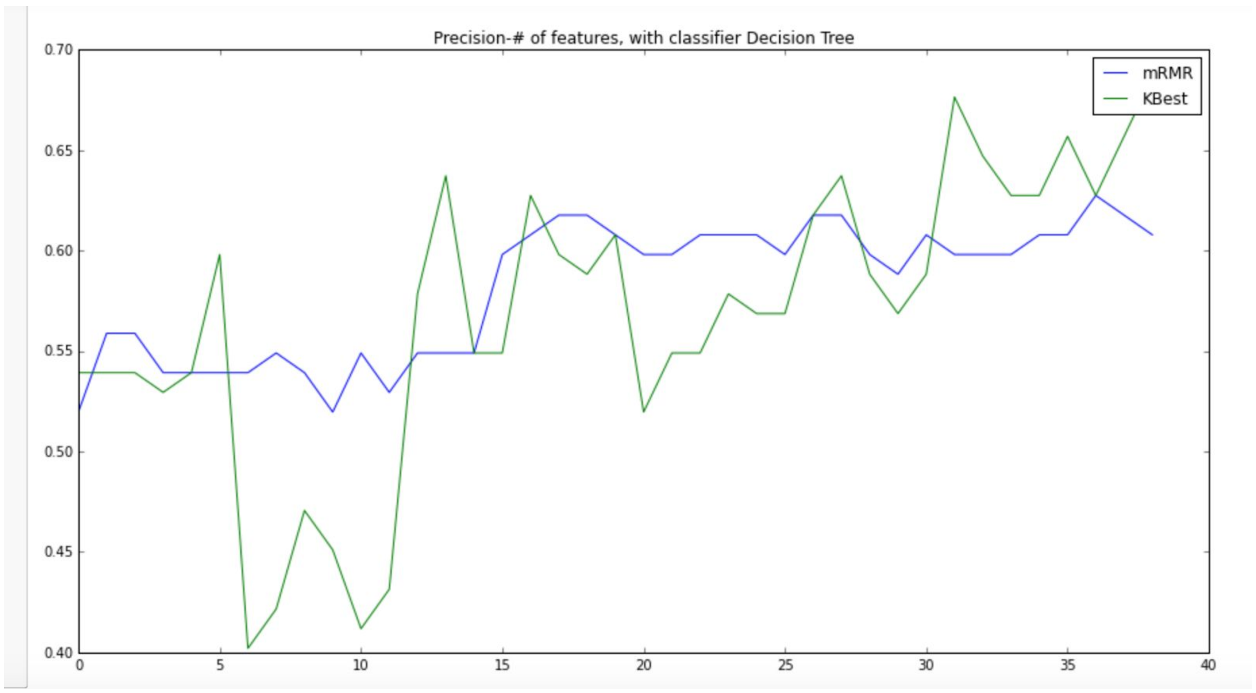
2.2.1 The k-precision graph
We first run our mRMR model for the selection of features given a fixed number of k. The selected features are then tested with a chosen classifier and a graph was plotted based on the (k, precision) results.

With all the features (279 in total), the decision tree will only get a precision of 0.578, with our self-implemented feature selection algorithm mRMR we got the following plot:



Graph 1. Precision - # of features selected, with DT Classifier.

However, we may also want to compare with the sklearn.feature_selection.**SelectKBest** to see the difference. SelectKBest requires first entering k value, where we used a range from 1 to 40 (note that we only have about 400s samples, which makes larger size of features meaningless). The speed is much faster than our selector, yet with Decision Tree Classifier, it showed that our selector is competitive with respect to the precision, and perhaps more stable.



Graph 2. Precision - # of features selected, mRMR (blue) v.s. KBest (green), tested under Decision Tree Classifier.

2.2.2 Compact Feature Subsets and Backward Selection
In order to select the best candidate feature set (k is unknown), we need to calculate the cross-validation error to find a relatively stable range of small error. We used the backward selection to select the best k.

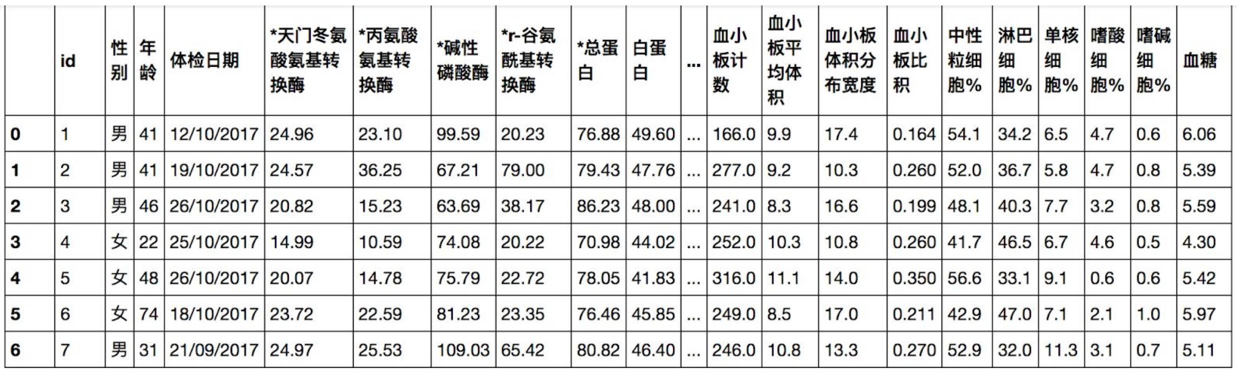
We saw from the graph above that Ω (the range of k, where the error change is consistently small) is between [17 , 39] . We choose k = 37 as the optimal size of our candidate feature set. Remember the backward selection tries to exclude one redundant feature from the candidate feature set at a time. Each feature will be removed as an experiment to find the largest error reduction, and the backward algorithm stops if no further improvement can be achieved.

We executed the backward on our example for the wrapper with Decision Tree Classifier and for the parameter “random_state = None”, the backward algorithm reduced 3-4 features on average. Codes can be found in “example.ipynb”.

2.2.3 Experiment on Blood Sugar Prediction
We will go through the basics for doing data analysis, and then create a framework for general data analysis tasks. To be more specific (i) data storage, feature extraction and clean;
(ii) statistical analysis; (iii) training, estimation and prediction; (iv) automatique model selection et cross validation; Codes can be found in “example.ipynb” (note that when selecting more

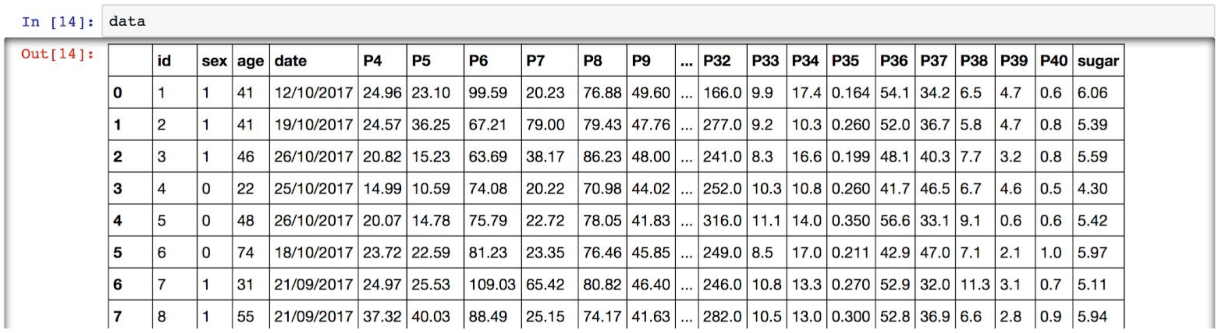
features, the mutual informations calculated in our KNN way for the diabetes data risks being negative, and we only got it work for the simplest case when n_feature <= 5).

We concentrated on the last step (iv) and gave some brief introduction for the first three. As the data source includes Chinese character, we used “encoding='gbk'” when importing the data.



	id	性别	年龄	体检日期	*天门冬氨酸氨基转换酶	*丙氨酸氨基转换酶	*碱性磷酸酶	*r-谷氨酰基转换酶	*总蛋白	白蛋白	...	血小板计数	血小板平均体积	血小板体积分布宽度	血小板比积	中性粒细胞%	淋巴细胞%	单核细胞%	嗜酸细胞%	嗜碱细胞%	血糖
0	1	男	41	12/10/2017	24.96	23.10	99.59	20.23	76.88	49.60	...	166.0	9.9	17.4	0.164	54.1	34.2	6.5	4.7	0.6	6.06
1	2	男	41	19/10/2017	24.57	36.25	67.21	79.00	79.43	47.76	...	277.0	9.2	10.3	0.260	52.0	36.7	5.8	4.7	0.8	5.39
2	3	男	46	26/10/2017	20.82	15.23	63.69	38.17	86.23	48.00	...	241.0	8.3	16.6	0.199	48.1	40.3	7.7	3.2	0.8	5.59
3	4	女	22	25/10/2017	14.99	10.59	74.08	20.22	70.98	44.02	...	252.0	10.3	10.8	0.260	41.7	46.5	6.7	4.6	0.5	4.30
4	5	女	48	26/10/2017	20.07	14.78	75.79	22.72	78.05	41.83	...	316.0	11.1	14.0	0.350	56.6	33.1	9.1	0.6	0.6	5.42
5	6	女	74	18/10/2017	23.72	22.59	81.23	23.35	76.46	45.85	...	249.0	8.5	17.0	0.211	42.9	47.0	7.1	2.1	1.0	5.97
6	7	男	31	21/09/2017	24.97	25.53	109.03	65.42	80.82	46.40	...	246.0	10.8	13.3	0.270	52.9	32.0	11.3	3.1	0.7	5.11

An overview of the diabetes data

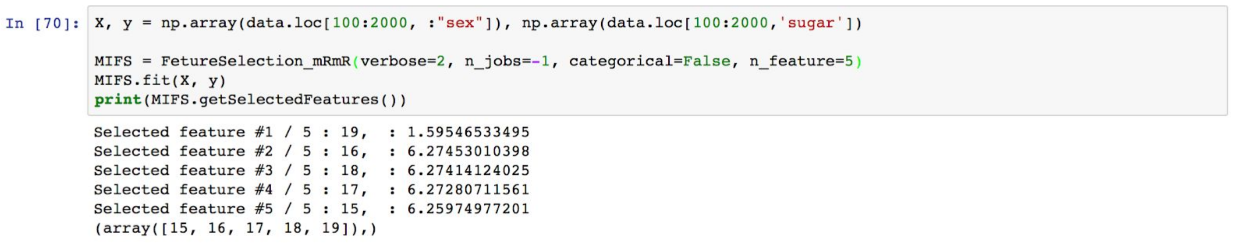


```
In [14]: data
```

Out[14]:

	id	sex	age	date	P4	P5	P6	P7	P8	P9	...	P32	P33	P34	P35	P36	P37	P38	P39	P40	sugar
0	1	1	41	12/10/2017	24.96	23.10	99.59	20.23	76.88	49.60	...	166.0	9.9	17.4	0.164	54.1	34.2	6.5	4.7	0.6	6.06
1	2	1	41	19/10/2017	24.57	36.25	67.21	79.00	79.43	47.76	...	277.0	9.2	10.3	0.260	52.0	36.7	5.8	4.7	0.8	5.39
2	3	1	46	26/10/2017	20.82	15.23	63.69	38.17	86.23	48.00	...	241.0	8.3	16.6	0.199	48.1	40.3	7.7	3.2	0.8	5.59
3	4	0	22	25/10/2017	14.99	10.59	74.08	20.22	70.98	44.02	...	252.0	10.3	10.8	0.260	41.7	46.5	6.7	4.6	0.5	4.30
4	5	0	48	26/10/2017	20.07	14.78	75.79	22.72	78.05	41.83	...	316.0	11.1	14.0	0.350	56.6	33.1	9.1	0.6	0.6	5.42
5	6	0	74	18/10/2017	23.72	22.59	81.23	23.35	76.46	45.85	...	249.0	8.5	17.0	0.211	42.9	47.0	7.1	2.1	1.0	5.97
6	7	1	31	21/09/2017	24.97	25.53	109.03	65.42	80.82	46.40	...	246.0	10.8	13.3	0.270	52.9	32.0	11.3	3.1	0.7	5.11
7	8	1	55	21/09/2017	37.32	40.03	88.49	25.15	74.17	41.63	...	282.0	10.5	13.0	0.300	52.8	36.9	6.6	2.8	0.9	5.94

After data clean, 41 features in total, last column is the blood sugar percentage (y)

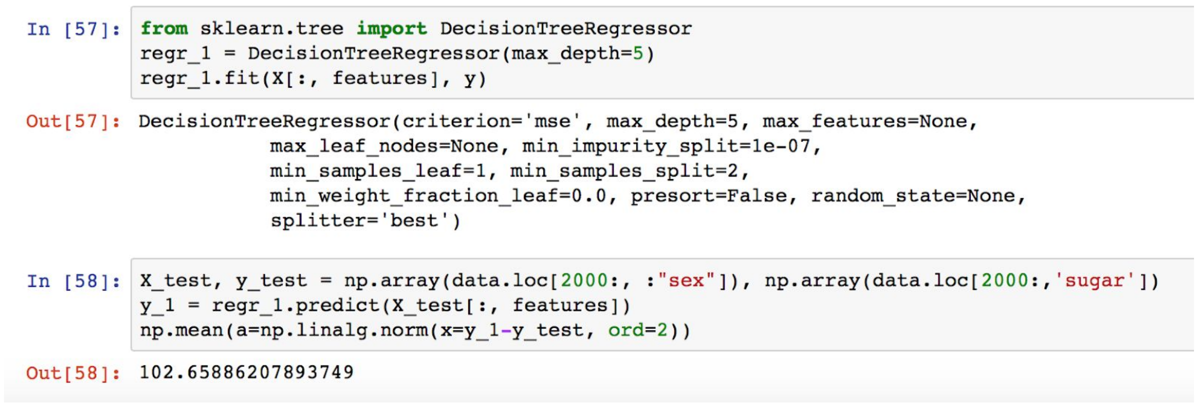


```
In [70]: X, y = np.array(data.loc[100:2000, :"sex"]), np.array(data.loc[100:2000,'sugar'])

         MIFS = FetureSelection_mRmR(verbose=2, n_jobs=-1, categorical=False, n_feature=5)
         MIFS.fit(X, y)
         print(MIFS.getSelectedFeatures())

         Selected feature #1 / 5 : 19,   : 1.59546533495
         Selected feature #2 / 5 : 16,   : 6.27453010398
         Selected feature #3 / 5 : 18,   : 6.27414124025
         Selected feature #4 / 5 : 17,   : 6.27280711561
         Selected feature #5 / 5 : 15,   : 6.25974977201
         (array([15, 16, 17, 18, 19]),)
```

Select most relevant features



```
In [57]: from sklearn.tree import DecisionTreeRegressor
         regr_1 = DecisionTreeRegressor(max_depth=5)
         regr_1.fit(X[:, features], y)

Out[57]: DecisionTreeRegressor(criterion='mse', max_depth=5, max_features=None,
                    max_leaf_nodes=None, min_impurity_split=1e-07,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                    splitter='best')

In [58]: X_test, y_test = np.array(data.loc[2000:, :"sex"]), np.array(data.loc[2000:,'sugar'])
         y_1 = regr_1.predict(X_test[:, features])
         np.mean(a=np.linalg.norm(x=y_1-y_test, ord=2))

Out[58]: 102.65886207893749
```

Test for error

Part 3 Short Description of the Library Code and Difficulties Encountered

3.1 Short Description of the Library Code

We referred to two open source codes in sklearn:

1. sklearn.feature_selection.mutual_info_classif [1] (For classification tasks)
2. sklearn.feature_selection.mutual_info_regression [2] (For regression tasks)

[1]http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html
[2]http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html

3.2 Difficulties Encountered and Solution

a). The complexity of computation of distribution of joint probability.

Because of the complexity of computation of distribution of joint probability of two features, we made use of the k nearest neighbour to replace the density estimation method in the paper.

b). The realization of parallel computation

In the algorithm of mRmR, it contains a lot of computation of arrays and matrix, so, to accelerate the speed of computing, we parallelize the calculation using the module ‘joblib’ of sklearn to make sure that all cpus could be utilized at the same time.

c). The problem of dividing by zero

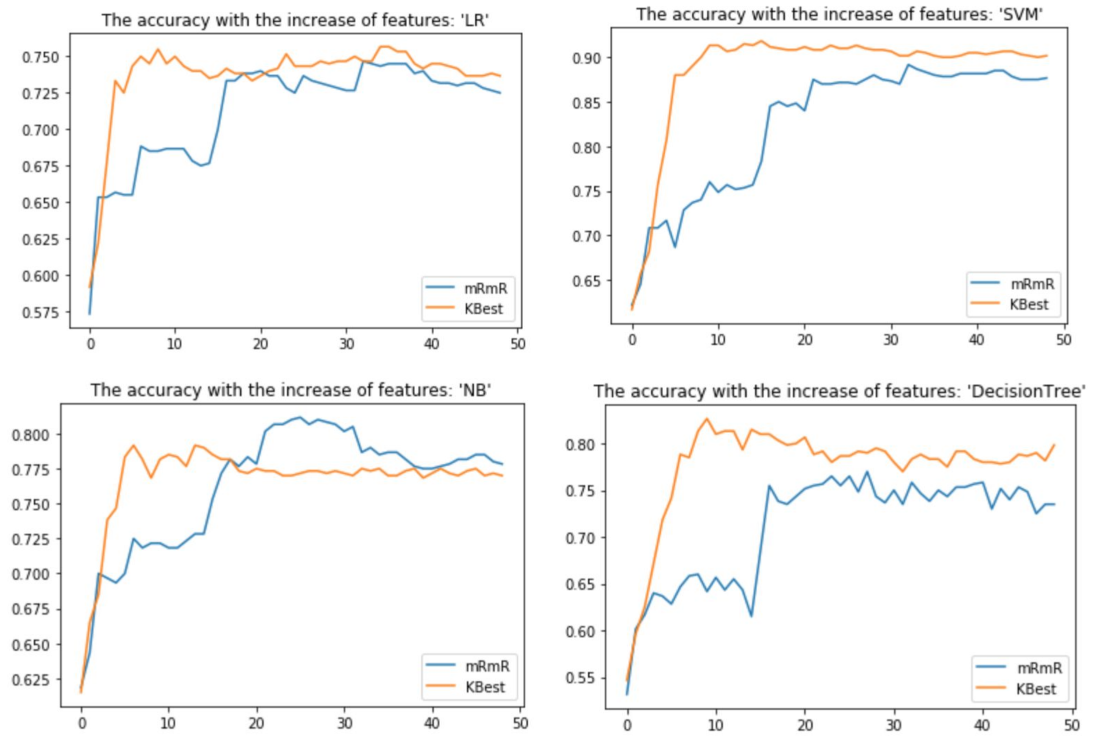
Actually, in daily life, the data sometimes is small, when apply the normal entropy function shown in Geometric k-nearest neighbor estimation of entropy and mutual information

Warren M. Lord, Jie Sun, and Erik M. Bollt, it encounters always the problem of division by zero(which is not a problem for the simulated data case). To solve this problem, we choose to replace it with another function which could be found in the code. These two functions shows the similar performance when it comes the issue of accuracy.

Part 4 Analysis of the Results

In this part, we would like to give the analysis of the results of our experiments. We compared the precision accuracy with the selected features with the one of SelectKBest method.

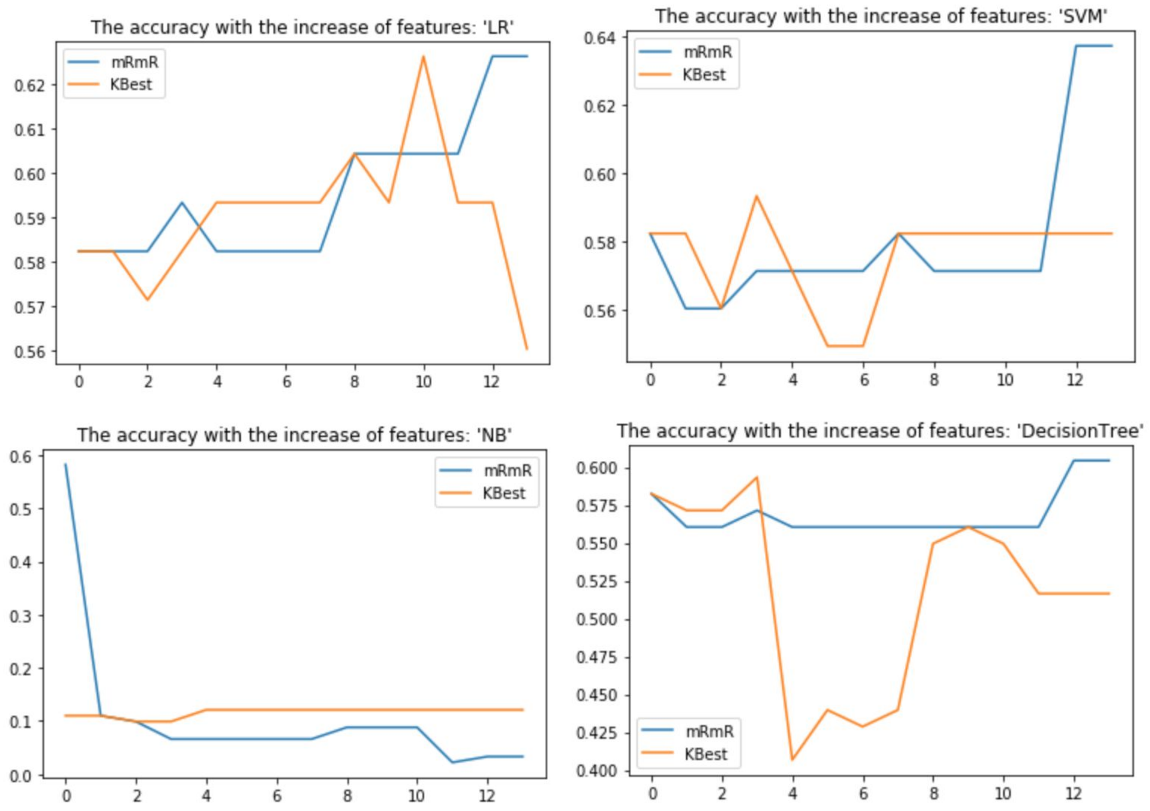
4.1 The result of generated data.



Graph 3. Precision - # of features selected, mRMR (blue) v.s. KBest (green), tested under Logistic Regression, SVM, Naive Bayes and Decision Tree separately(Simulation data).

From the plots, we can see that the performances of our mRmR algorithm and the SelectKBest algorithm could obtain the similar effect when we set the number of features equals to 50. For the case of Naive Bayes, the performance of mRmR even performs better.

4.2 The result of real data



Graph 4. Precision - # of features selected, mRMR (blue) v.s. KBest (green), tested under Logistic Regression, SVM, Naive Bayes and Decision Tree separately(Real data).

The results of real data also shows the similar performance with SelectKBest method. The performance for the real-world data seems to be not that ideal, the reasons for it may be the limit of the scale of the samples and features. However, according to the performance of simulated data, we believe it could also perform well in the case we have enough scale of data. We would try more real-world data in the further work.

Part 5 Conclusion

In this section we make a conclusion on the method that we utilized for the implementation and on the library that we referred.

5.1 A critical view of the method

(1) Computation time
With the algorithm and the parallel method we have applied, the feature selection is quite fast, certainly much more fast than the Max Dependency method.

However, compared to the SelectKBest method, our method seems to be a little bit slower even if we have already applied the parallel computation.

(2) Limitations
The main limitations lie in the limitations of mutual information estimation based on kNN method:

For kNN, the testing time is proportional to the size of the training set. So The larger the training set, the longer it takes to classify a test document. kNN is inefficient for very large training sets.

(3) Good points

1. We have added the a ‘parallel’ choice to speed up the computation.
2. We make use of KNN to simulate the density distribution.

5.2 A critical view of the library used in the task.

SelectKBest: In the case of classification problem, this library works very well. However, it could not be applied to the regression problem, so we have not shown the comparation between the performances of SelectKBest and mRmR in the case of regression problem.