



Dimensionality Reduction Methods Comparison

Dissertation or thesis submitted for Master's in Data Analytics for
Business Intelligence

Department of Mathematics, University of Leicester

September 2022

Submitted by: Mohit Baliyan (mb856)

Student ID: 209002494

Academic Supervisor: Dr. Evgeny Mirkes

Program Director: Dr. Andrew Morozov

1. Abstract

Dimensionality Reduction is one of the major sub problems from Data Mining's 6 historical problems. Principal Component Analysis is one of the mathematical tools which is used widely to project high dimensional data into lower dimensional data. There is no validated choice of selecting the number of principle components to represent the original data. There are three main heuristics to select a number of informative (useful, meaningful, etc.) components namely Kaiser, Broken Stick and Conditional Number rule. In this study, we used 23 classification databases and systematically tested all three heuristics including original dimension using three classifiers namely Fisher Discriminant, Logistic Regression and K Nearest Neighbor. In the investigation, we demonstrated that there is no one universal answer to which intrinsic dimension is better. However, in our study, we found that in average dimensionality reduction on base of conditional number is better.

2. Acknowledgements

My first and big appreciation goes to my academic supervisor, Dr Evgeny Mirkes, for his marvelous supervision, guidance and encouragement. Sincere gratitude is extended to his generous participation in guiding, constructive feedback, kind support, and advice during my masters. Thank you very much Dr Evgeny.

Also, I greatly appreciate my program director, Dr Andrew Morozov for his great feedback, excellent encouragement and guidance. Thanks a lot you both.

Many thanks to all of the members of staff in University of Leicester for their kind support during my masters study. Also, I extend my thanks to all my colleagues at University of Leicester for their continuous encouragement and support, as well as to all of my friends and colleagues from Leicester and outside for their time, advice and moral support.

Last, but not least, my warm and heartfelt thanks go to my family for their tremendous support and hope they had given to me. Without that hope, this thesis would not have been possible. Thank you all for the strength you gave me. I love you all!

3. Table of Contents

S.No.	Chapters	Page Number
1.	Abstract	2
2.	Acknowledgements	3
3.	Table of Contents	4
4.	List of Figures	5
6.	List of Tables	6
7.	Introduction (Problem Formulation)	7
8.	Background Literature	8
9.	Methods	9
10.	Results and Comparisons	19
11.	Conclusion	52
12.	References	53
13.	Appendix	54

4. List of Figures

S.No.	Name of Figures	Page Number
1.	PCA Example	10
2.	Scree Plot	12
3.	Broken Stick Rule	13
4.	Choice of K: Climate Database	27
5.	Choice of K: 2 Databases	28
6.	Choice of K: 3 Databases	29
7.	Threshold for Logistic: Climate Database	32
8.	Threshold for Logistic: 2 Databases	33
9.	Threshold for Logistic: 3 Databases	34
11.	Threshold for Logistic: Folds in Climate	35
12.	Threshold for Fisher: Climate Database	36
13.	Threshold for Fisher: 2 Databases	37
14.	Threshold for Fisher: 3 Databases	38
15.	Threshold for Fisher: Folds in Climate	39

5. List of Tables

S.No	Name of Tables	Page Number
1.	Dimensions Retained	25
2.	Selected Best K	29
3.	Performance Measure: Fisher	46
4.	Fisher: Average Based Ranking	47
5.	Fisher: p Based Ranking	48
6.	Performance Measure: Logistic	49
7.	Logistic: Average Based Ranking	50
8.	Logistic: p Based Ranking	51
9.	Performance Measure: KNN	52
10.	KNN: Average Based Ranking	53
11.	KNN: p Based Ranking	54

6. Introduction (Problem Formulation)

The objective of dimensionality reduction is to reduce multicollinearity, saving computational time and space. Apart from this, most of the models don't perform well on high dimensional data like KNN. Principal Components Analysis is used to reduce the high dimensional data with an objective to remove noise and retain the principal components with high variance (or say retain those which contains more information). But there are many heuristics approaches to retain principal components. But we don't have any mathematical proof or demonstration which can guarantee that this heuristic approach will retain highly relevant principal components which lead to achieving best performance during modelling.

For instance, we have Database A having dimensions of (1000, 55) and we need it to reduce into lower dimensions by projecting it into lower dimensions subspace and we use Principal components Analysis to reduce this higher dimensional Database A. So, what PCA does, it projects all the features into other axes with an objective to reduce the noise and increase the variance across all axes. So, now to reduce the original data into lower dimensional, we need to retain the principal components, but we don't know which heuristic approach will lead us to getting reduced Database which could possibly have the intrinsic dimension.

We currently have 3 main heuristics approaches to retain the principal components which are given below:

1. Kaiser Rule
2. Broken Stick Rule
3. Conditional Number

So, the objective of this study is to investigate these heuristics approaches on 23 Database to know if there is any universal answer which could lead us to intrinsically represent the original data.

7. Background Literature

According to (James et al. 2013), there is not any single universal adapted answer to how many principal components a someone retains. So that, they are successfully able to find the intrinsic dimension of the data. The author stated that it should be dataset dependent or what is the domain of the dataset it derived from.

However, geo.fu.berlin (2018) suggested this 3-step simple approach to retain the highly relevant principal components which are given below.

1. Visualize the Scree Plot and retain PC components based on that plot.
2. Based on Variance Explained methods
3. Kaiser Rule.

Kaiser rule state that all PC with eigenvalue greater than mean eigenvalue are informative. The Broken Stick rule Jackson et al. (1993), is another approach to retain the highly relevant principal component. It's basically, based on a stick of unit length and split into X number of units by randomly to model the X variances and then select the breaking points from this uniform distribution.

In general, the Broken stick rule states that all eigenvalues which are greater than uniformly estimated length of fragment of randomly broken stick. The conditional number rule states that

all principal components with eigenvalues which are greater than maximal eigenvalue divided by 10 are statistically significant.

Joaquín A. et al. (2013) stated one of the analyses they did on variable selection. They first split the variables into disjoint sets and ensure to keep one variable from each disjoint group. They used 2 Branch and Bounds and found that both work effectively when analyzing the variable selection problem.

8. Methods

In this section, we will explain all the methods in brief which were used in this study.

Dimensionality Reduction and PCA

PCA or Principal components analysis is a machine learning technique which is used to reduce the dimensionalities or variables in data sets to a minimum number without losing the useful information from it and still get the valuable results. This technique is used in confusion datasets having hundreds of variables and dimensions and helps to get hidden dimensions of data as well. For example, a weather forecast company needs to predict the weather in parts of the UK and needs to consider whatever variables are associated with it. A weather forecast algorithm predicts weather considering various variables, inputs and confusion datasets which needs to be considered for prediction, but it is not easy to identify if all variables need to be used or not, in which case PCA can be used to reduce the dimensions of data and given desired results. For this problem there can be several data variables in data sets, for example: intensity of winds, pressure, humidity, past weather data and many more like this. Unfortunately, we cannot figure

out what is happening because the data appears clouded, unclear and even redundant. A very complex system needs to be built if we must predict weather efficiently but it's a challenge to use various algorithms and time and space complexity will be more inefficient. In this case the data set should be cleaned in a form where all data variables will give a standardized value and where we need to select the components of algorithm and can see a clear picture with less dimensions of data.

PCA can be of different types with respect to prediction or how much information needs to be gained. It can be 2-Dimension or 3-Dimensions which gives two or three components on different axis and actual plot and graphical representation of data variables can be seen.

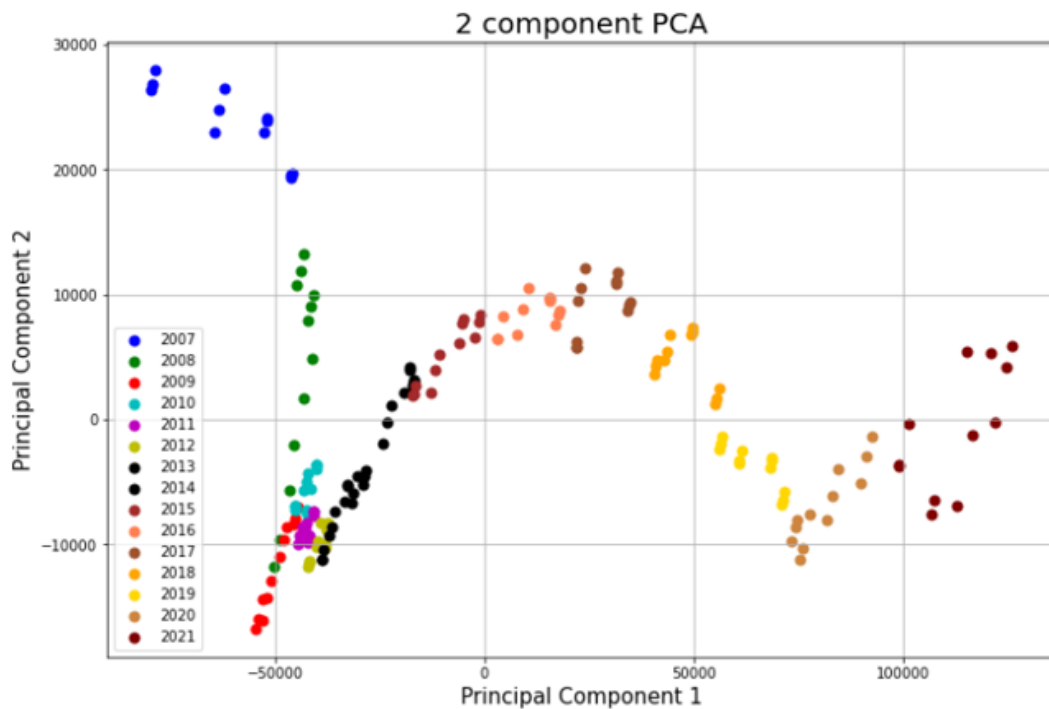


Figure1. PCA Example

Above is a common example of change in different datasets throughout the year and reducing 15 features of data into 2. Basically, what this analysis does is shrink the dimensionality and reduce the features to 2 and 3 components having the same meaning. There must be one variable which we can plot. PCA has played a huge role in this research project where hundreds of dimensions had to be dealt but data variables were not correlated to each other, and it was hard to show some useful information gained. And this algorithm clearly helped in this thesis.

Dimensionality reductions have been done using the algorithm below. The process of how this algorithm has been used in this research project is:

- Dataset d +1 dimensions was considered to make it useful with dimension d
- Computation of mean value for every data variable
- Computation of correlation coefficient matrix
- Computation of eigen vectors and eigen values
- Associating each component value (2 components in our case) with eigen vectors and generating matrix to plot the graph with 2 components

Mathematically, it can be shown as:

$$cov_{x,y} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{N - 1}$$

$cov_{x,y}$ covariance of features x and y

x_i specific training example from feature x

y_i specific training example from feature y

\bar{x} mean over all examples of feature x

\bar{y} mean over all examples of feature y

N total amount of examples

Kaiser Rule

The Kaiser Criterion is an algorithm which also works with dimensions of the data and eigen vectors and values of correlation matrix. This method is commonly used in exploratory data analysis. The main idea of this algorithm or technique is to work on eigen values and take them under 1.0. This algorithm should be used when we know there can be high correlation in data

variables which are getting used. The important part with this technique is summarizing the data which is nothing but using eigen values to summarize by using given data set. An eigen value is mainly a value to know how good the model and components are. An eigenvalue of 1.0 means that the factor contains the same amount of information as a single variable. The use of Kaiser Rule can also be used to understand how many important components of data variables can be used for PCA. Suppose a data set has 100 dimensions and 3 dimensions giving the highest values of eigen vector, which means the components are giving useful information then PCA or other algorithms can be implemented using those 3 features of data. A basic plot from kaiser rule is given below, showing the eigen values of the dataset components:

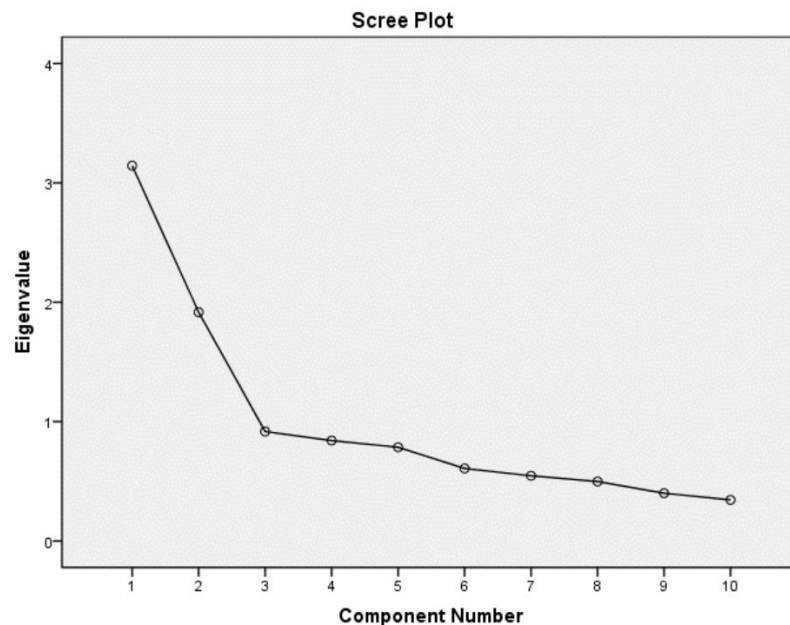


Figure2. Scree Plot

Broken Stick Rule

The biggest challenge when dealing with a large data set remains the dimensionality reduction and what components need to be chosen and what can be avoided. It all depends on eigen values calculation. There are several methods in statistics to identify these components depending on the nature of the problem. It is very useful to understand which technique gives best results,

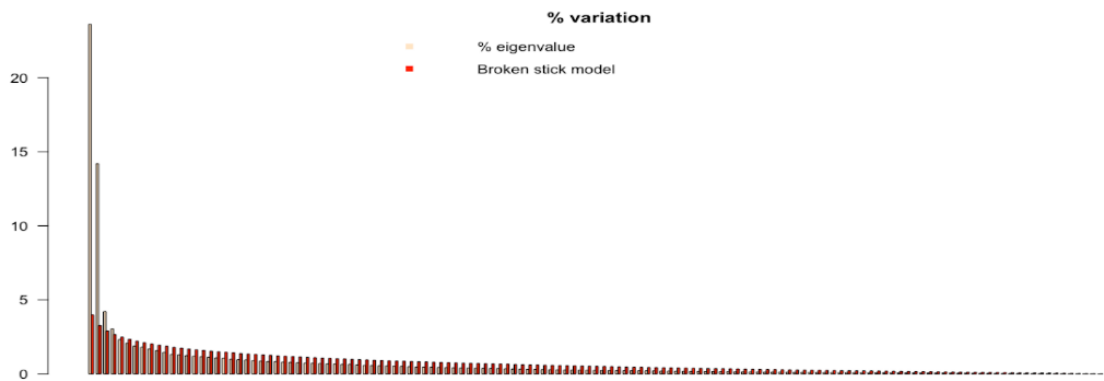
hence in this thesis several methods like above have been used and differentiated to figure out what values of components to use. Another technique is the broken stick rule, it gives the components of data set which gives more variances in p parts. If we divide quantity into random parts, the proportion of largest part can be given as $(1/p) \sum (1/i)$ where the summation is over the values $i=k..p$. For example, if $p=7$ then,

$$E1 = (1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7) / 7 = 0.37,$$

$$E2 = (1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7) / 7 = 0.228,$$

$$E3 = (1/3 + 1/4 + 1/5 + 1/6 + 1/7) / 7 = 0.156, \text{ and so on}$$

The broken stick rule giving variances in parts of data sets values is shown below:



Broken stick rule is considered as one of the best rules to understand principal components of data sets.

Figure3. Broken Stick Rule

Conditional Number Rule

The condition number is an application of the derivative and is formally defined as the value of the asymptotic worst-case relative change in output for a relative change in input. Condition

number rule is also known as decision rule and works on criteria of conditions and outcomes of condition. After every step it calculates if-then condition. The whole idea is to check if the data set variable giving the desired knowledge gain or not. It calculates eigen values for every variable and checks if the eigen value can be considered or not. If not, then it will run for each data variable until it's 0. This rule has few disadvantages as it only works if variables are not correlated to each other and does not work well with complex data sets and problems like this thesis. A condition number for a matrix and computational task measures how sensitive the answer is to perturbations in the input data and to roundoff errors made during the solution process.

Classifiers

Fisher

The popular dimensionality reduction technique which is PCA works on giving components with variances and removes noise and bad data and does not include label information for components. The challenge is to decide what the label is for projections of components and Fisher's algorithm is used in this case which works with maximizing the given formula:

$$J(w) = w^T S_B w / w^T S_W w$$

where S_B is the "between classes scatter matrix" and S_W is the "within classes scatter matrix". Note that due to the fact that scatter matrices are proportional to the covariance matrices we could have defined J using covariance matrices – the proportionality constant would have no effect on the solution. The definitions of the scatter matrices are:

$$S_B = \sum_c N_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T$$

Where,

$$\mu_c = \frac{1}{N_c} \sum_{i \in c} x_i$$

$$\bar{x} = \frac{1}{N} \sum_i x_i = \frac{1}{N} \sum_c N_c \mu_c$$

and N_c is the number of cases in class c .

It follows analysis of variance which is also ANOVA and regression analysis which considers one dependent variable as linear combination for others.

KNN

K nearest neighbor is a classifier who works on competitive or lazy learning and did its training during prediction only. I basically saved all the training examples and label the test sample based on the majority of the samples's label which are closer to it. For instance, we have 15 samples in our database, and we split into train and test of 10 and 5 samples. So, let's say we must classify the 11th sample, it will compute the nearest K samples in the 10 training samples, samples in majority belong to same class in these K samples will be our classification for this 11th sample. There are various approaches to select the number of K in KNN, elbow method of calculating performance on various value of K to choose the optimal K. Method to find the closest samples would be Euclidean distance.

Logistic Regression

The Logistic Regression Classifier we used is inspired from linear regression. Hypothetical model used in linear regression is limited to predict between 0 and 1 by applying the sigmoid function on it. So that it can be $(1 / (1 + e^{-z}))$, where $z = XW + b$. Cost function in optimizing the value of W to reduce the cost function or loss function would be cross entropy because our logistic model will predict discrete values despite continuous values.

Leave one out cross validation (LOOCV)

Leave one out cross validation is a special case of K fold cross validation where $K = n$, n denoted number of samples. In leave one out cross validation, each sample will be used as a validation set individually and act as a hold-out case for the validation set. For example, if we have 10 samples or $n = 10$ then in first iteration 1st sample will be validation set, then in second iteration only 2nd sample will be validation set and so on. The leave one cross validation's test error gives the actual estimate of the prediction error. Leave one out cross validation (or sometime called LOOCV) suffers from high variance, this happened because of the slight change in training set throughout the n cross-validation steps. As only one example changes in every fold, so there is not a practical change in training at each fold. Leave one out cross validation is also computationally expensive, because it will take equal to n times of model training on $n-1$ samples at each fold. Berrar *et. al*(2018).

Stratified Cross Validation

Stratified Cross Validation technique is one of the cross-validation techniques which is used to evaluate a machine learning model performance. It gives the actual performance of the model, and it may give worse results if a model suffers from overfitting. Stratified Cross Validation technique is inspired from stratified sampling. Stratified sampling refers to distributing the data into samples by maintaining the ratio of category of the samples. For instance, let's say we have 100 samples of people, in which we have 40 people who suffered from cancer while the remaining 60 are fine. So, the ratio of patients with normal people is $2/3$. If we divided this sample into 2 sub samples or say folds using stratified sampling, then we will have to maintain ratio of $2/3$ of patients to normal people in both the samples. So, 40% of 50 is 20, then 20 in first sub sample should be people suffering from cancer, while 60% of 50 is 30, then 30 in first sub sample should be normal people. KSV Muralidhar (2014).

Then the 1st sample should contain 20 patients and 30 normal, same for 2nd sample would be that it should contain 20 patients and 30 normal. So, stratified 10th fold cross validation will divide the datasets into 10 folds where each fold should contain samples from all categories in same proportion across all folds.

Balanced Accuracy Score

We used balanced accuracy metrics to evaluate the performance of our classifiers. It is defined as an average of sensitivity and specificity.

$$\text{Balanced Accuracy} = (\text{Sensitivity} + \text{Specificity}) / 2$$

Sensitivity is also called recall or true positive rate as it measures the rate of true positive prediction by the classifier.

$$\text{Sensitivity} = \text{TP} / (\text{TP} + \text{FN})$$

While Specificity is also called selectivity or true negative rate as it measures the rate true negative prediction by the classifier.

$$\text{Specificity} = \text{TN} / (\text{TN} + \text{FP})$$

Where, TP = True Positive,

FN = False Negative

TN = True Negative

FP = False Positive

Testing and Ranking Protocol

We used 2 ranking protocols to rank the performance of a classifier in 4 scenarios on a database's original dimensions, kaiser, broken stick and conditional number. First is average ranking and second one is T-test based significance testing ranking.

Average Ranking

In this approach, we will rank the performance of classifier on all the heuristics approach using average method. For instance, on a Breast Cancer database, Fisher classifier achieved a balanced accuracy of 0.9587 on original database, 0.9507 on reduced database using Kaiser rule, 0.9293 on reduced database using Broken Stick Rule, 0.9501 on reduced database using Conditional Number. So, using average ranking we will rank original 1, kaiser rule 2, broken stick rule 3 and conditional number rule 4. For another example, on a Banknote dataset, Fisher classifier achieved a balanced accuracy of 0.9877 on original database, 0.78 on kaiser led reduction, 0.78 Broken stick rule led reduction and 0.91 conditional number rule led reduction.

As we can see balanced accuracy achieved on kaiser and broken stick rule led reduction was same and on rank 3. We can assign a rank of average of both i.e.. $(3 + 4) / 2$ equivalent to 3.5 to both.

T-test based significance testing

In t-test paired test, it is used to find the significantly different between two samples assuming from normal distribution. The objective is to check if there is any significant difference between the two samples.

Basic Assumptions or needs to conduct this test.

1. Both the samples should be continuous in nature.
2. Both samples should follow the normal distribution.
3. Samples should be large and randomly selected from the population.

$$t = \frac{\sum d}{\sqrt{\frac{n(\sum d^2) - (\sum d)^2}{n-1}}}$$

Where, where **t** = t-statistic, **d** is the difference between two samples and **n** is the samples size.

9. Results and Comparisons

Heuristics Approach to calculate number of principal components

First, we calculated the number of principle components retained by all the three heuristic approaches of selection namely Kaiser, Broken Stick and Conditional Number. To produce these results, we simply first implemented Principal Components Analysis algorithm separately which is presented below. This algorithm has two methods, one for calculating eigen values and eigen vectors and the other method. This first method will take X matrix as input and will give the eigen values and eigen vectors as output. The second method of this algorithm will take X matrix and number of principal components we want to retain as an input and will give the transformed X matrix after projection as an output.

Algorithm: PCA

First Method

Step1: def Eigen(X):

Step2: X_stand = X - mean(X) # centralize the data

Step3: X_cov = Corr(X) # calculate correlation matrix

Step4: e, v = eig(X_cov) # find the eigenvalues and eigenvectors

Step5: idx = sort(e) # sort eigen vector according to eigen values in descending order

Step6: e = e[idx] # eigen values

Step7: v = v[:, idx] # eigen vectors

Step8: return e, v

Second Method

Projection of X

```
Step1: def transformation (x, no_of_components):  
Step2:     e, v = eig(x)  
Step3:     p = v [: : no_of_components]  
Step4:     mean = mean (x, 0)  
Step5:     x_stand = x - mean  
Step6:     x_transform = dot_product(x_stand, p)  
Step7:     return x_transform
```

Then we developed an algorithm called Selection Methods which is presented below. This algorithm has 3 methods namely Kaiser, Broken Stick Rule and Conditional Number. All these methods will take X matrix as an input, apart from this, these methods used above PCA algorithm to calculate the eigen values and eigen vectors for that X matrix. These methods will give a number of principal components to retain according to that rule as an output. Additionally, these methods would give 1 as an output if any heuristic approach calculated to not retain any principal components. In that case, these methods will give 1 as output. For example, for Database Climate, according to Broken Stick rule 0 number of principal components could be retained but method Broke Stick will give 1 as an output.

Algorithm: Selection Methods

```
# Function return number of components
```

```
def conditional_number():  
    pca = PCA()  
    e, v = pca.eig_vector(x)  
    e_max = e[0] # selection  
    condition = e_max / 10  
    no_of_components = count_nonzero(e > condition)  
    if no_of_components == 0:  
        return 1  
    else:  
        return no_of_components
```

```
# Function return number of components
```

```
def kaiser_rule():  
    pca = PCA()  
    e, v = pca.eig_vector(x)  
    # selection  
    return np.argmax(e < 1)
```

```
# function return number of components
```

```
def broken_stick():  
  
    pca = PCA()  
    e, v = pca.eig_vector(x)  
  
    # calculate the proportional variance  
    prop_var = e / sum(e)
```

```
# calculate the expected length of the k-th the longest segment
```

```
p = np.size(e)
```

```
g = np.zeros(p)
```

```
k = 0
```

```
while k < p:
```

```
    i = k
```

```
    while i < p:
```

```
        g[k] = g[k] + (1 / (i + 1))
```

```
        i = i + 1
```

```
    k = k + 1
```

```
g = g / p
```

```
'''
```

In the Broken-Stick model, the individual percentages of variance of the components are compared with the

values expected from the “broken stick” distribution. The two distributions are compared element-by-element,

and first value $d + 1$ where the expected values larger than the observed value determines the dimension.

```
'''
```

```
no_of_components = argmax(prop_var < g)
```

```
if no_of_components == 0:
```

```
    return 1
```

```
else:
```

```
    return no_of_components
```

Now, for each database we calculate components to retain using Kaiser rule, Broken Stick Rule and Conditional Number. We automated this process for all databases and produced results

presented in the table below. The algorithm we produced to automate this process is given below.

Algorithm: Dimension Estimation

```
def main():  
    # create list of databases  
    files = os.listdir('./Databases/')  
  
    # initialize lists to store each database name, no of attributes, cases and respective dimensions  
    databases = []  
    original_attributes = []  
    cases = []  
    kaiser_comps = []  
    broken_comps = []  
    conditional_comps = []  
  
    for file in files:  
        # load one database at a time  
        data = loadmat('./Databases/' + file)  
        x = data['X']  
  
        # standardise data  
        x = stats.zscore(x)  
  
        # apply selection methods for calculating respective dimensions  
        methods = SelectionMethods(x)  
  
        # for kaiser rule  
        kaiser_comps.append(methods.kaiser_rule())
```



```

# for broken stick
broken_comps.append(methods.broken_stick())

# for conditional number
conditional_comps.append(methods.conditional_number())

# original dimensions
(m, n) = x.shape
databases.append(file)
cases.append(m)
original_attributes.append(n)

# create dictionary, then convert into pandas dataframe to further save as a csv file
dictionary = {'Databases': databases, 'Cases': cases, 'Attributes': original_attributes, 'PCA-K':
kaiser_comps,
              'PCA-BS': broken_comps, 'PCA-CN': conditional_comps}

df = pd.DataFrame(dictionary)
df.to_csv('dimensions.csv')

```

Databases	Cases	Attributes	PCA-K	PCA-BS	PCA-CN
Banknote	1372	4	2	2	3
Blood	748	4	2	2	3
Breast Cancer	569	30	6	3	5
Climate	540	18	10	1	18
Cryotherapy	90	6	3	1	6
Diabetic	1151	19	5	3	8
Egg Eye State	14980	14	4	4	5
HTRU2	17898	8	2	2	4
Immunotherapy	90	7	3	1	7

LfW_faces	13233	128	51	55	57
Liver	579	10	4	1	7
Maledon	2600	500	224	1	362
Minboone	130064	50	4	1	1
Musk	476	166	23	9	7
Musk2	6598	166	25	13	6
Plrx	182	10	4	1	6
Qsar	1055	41	11	6	15
Skin	245057	3	1	1	2
Sonar	208	60	13	6	11
Spect	267	22	7	3	12
Spectf	267	44	10	3	6
Telescope	19020	10	3	1	6
Vertebral	310	6	2	1	5

Table1. Dimension Estimation

As we can see in the results, we calculated the number of principle components to retain using these three heuristics approach namely Kaiser, Broken Stick and Conditional Number rule. In the table above, cases represented the number of samples in that database, attributes represented the number of features which means original dimensions for that database is (Cases, Attributes). PCA-K represents the number of principal components retained by Kaiser rule, while PCA-BS represents the number of principal components retained by Broken Stick and PCA-CN represents the number of principal components retained by Conditional number approach to regain principal components.

- One of the insights we took from the above table is that Conditional Number approach retained more principal number of components than Kaiser rule and Broken Stick the most number of times.

K selection in KNN

For selecting the best K for each database, we used leave one out cross validation technique. Below is presented the algorithm we used to select K for that database.

Algorithm: Best K Selection

```

Step1:  def best_k(X, y):
Step2:      x = zscore(x)          # standardize x
Step3:      best_n = 0
Step4:      best_score = 0
Step5:      for i in range(1, 30, 2):      # iterate like 1, 3, 5 ...
Step6:          cv = LeaveOneOut()      # initialize leave one out cross validation object
Step7:          y_true, y_predict = list(), list() # enumerate splits
Step8:          for train_ix, test_ix in cv.split(x): # split data
Step9:              x_train, x_test = x[train_ix, :], x[test_ix, :]
Step10:             y_train, y_test = y[train_ix], y[test_ix]
Step11:             model = KNN(k=i)
Step12:             fit(x_train, y_train)      # fit the model
Step13:             y_hat = predict(x_test) # prediction using trained model
Step14:             y_true.append(y_test[0])    # store
Step15:             y_predict.append(y_hat[0])  # store
Step16:             score = balanced_accuracy_score(y_true, y_predict) # calculate score
Step17:             if score > best_score:      # assigned best score achieved
Step18:                 best_score = score
Step19:                 best_n = i
Step20:     return best_n

```

Where `balanced_accuracy_score` will calculate balanced accuracy after taking two vectors and `zscore` to standardize the `x`, it will take vector as an input. In producing this algorithm, we used `zscore`(SciPy) and `leave-one-out` cross validation function from (sklearn), while used our own K nearest neighbor classifier.

So, the above presented algorithm will be used later in this study to select the best K for classifier K nearest neighbor classifier for any database. This algorithm will take X and y as an input to calculate the best K for that database.

For instance, selecting the best K of KNN classifier for database Climate, we choose a range from 1 to 30 and then calculate the balanced accuracy at each odd number between 1 to 30 i.e., 1, 3, 5, 7,, 29. So, that K at which we achieved highest balanced accuracy will be our choice of K for that Database in KNN.

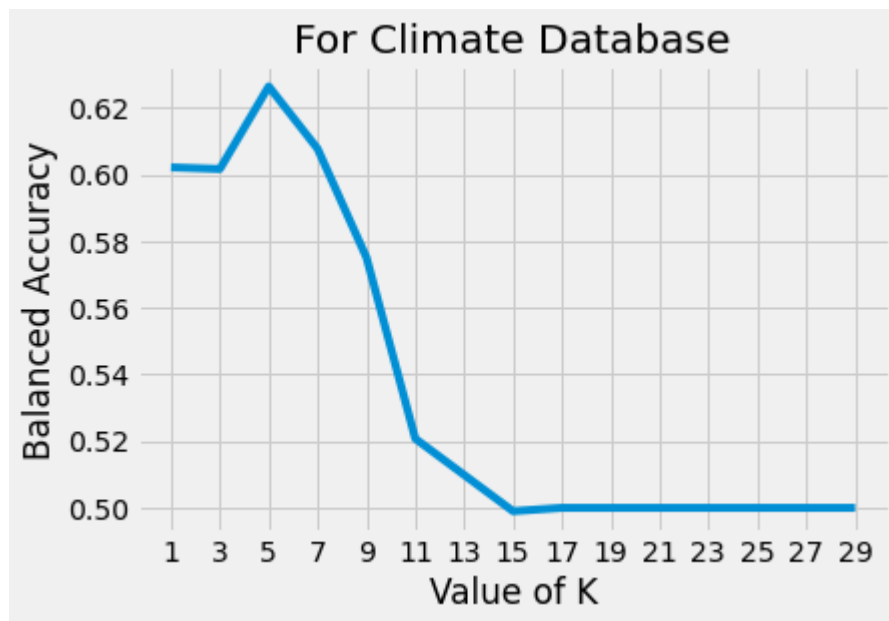


Figure4. Choice of K: Climate Database

- At K = 5, our classifier KNN achieved the highest balanced accuracy, so K = 5 will be our optimal choice when using KNN on Climate database in this study.

Why did we choose a different K in KNN classifier for each database?

The reason for selecting different K for each database is because classifier KNN's performance varies based on the selection of K. On one database, classifier KNN achieved highest balanced accuracy on one value, while on the other database, it achieved highest balanced accuracy on other value. For instance, Let's say we take two databases namely Climate and Blood and applied leave one out cross validation technique with choice of selection of K from 1 to 30 (odd values only).

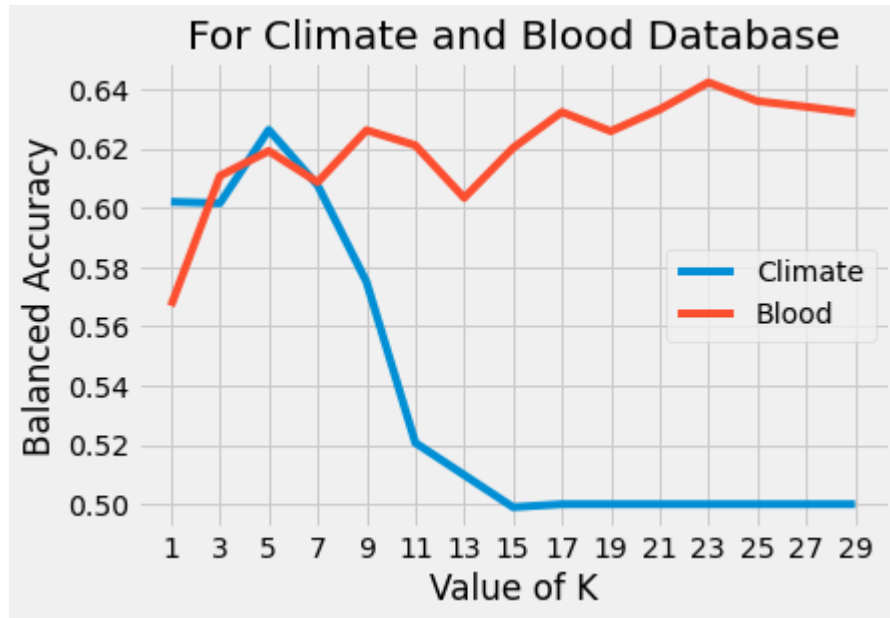


Figure5. Choice of K: 2 Databases

- As we can conclude from the above plot, classifier on Climate database achieved the highest balanced accuracy at $K = 5$, while classifier on Blood database achieved the highest balanced accuracy at $K = 23$
- So, the optimal choice of choosing K for Climate database will be 5, while for Blood Database it will be 23.

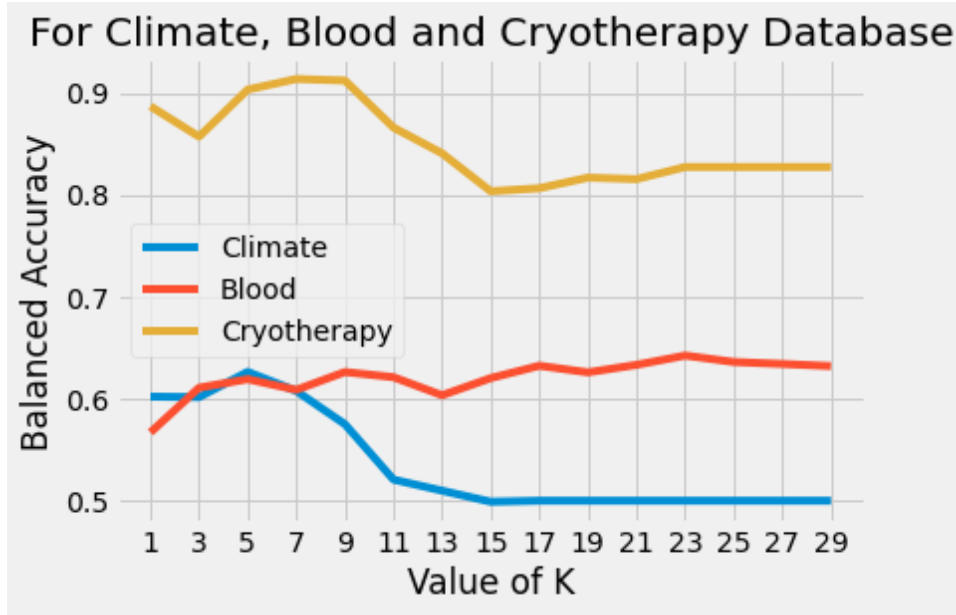


Figure6. Choice of K: 3 Databases

- We did a similar demonstration on 3 databases. From the above plot, the optimal choice of choosing K for Climate database is 5, for Blood Database it is 23 and for Cryotherapy it is 7.

These demonstrated that classifiers of different databases may need different values of K to achieve the highest balanced accuracy. Similarly, we calculated the best K or optimal choice of K for every database we used in this study using the algorithm we presented above name **Best K selection** which is presented in the table below.

Databases	Best K
Banknote	17
Blood	23
Breast Cancer	5
Climate	5
Cryotherapy	7

Diabetic	29
Immunotherapy	5
Liver	1
Maledon	17
Musk	5
Plrx	1
Qsar	7
Sonar	1
Spect	25
Spectf	11
Vertebral	1

Table. Selected Best K

Why do we choose the same K throughout each fold in one database?

The optimal choice of K for each database would be different, but we will use the same optimal choice throughout each fold during performance evaluation for that database. For example, we calculated optimal choice of $K = 5$ for classifier on climate database, as we will use stratified cross validation, so during each fold for climate database we will use $K = 5$ only. Because we used leave one out cross validation technique to select optimal K for each database which guarantee us that this will be our optimal choice throughout each fold as well.

So, from the demonstration and above study, we can conclude that, in this study we will use value of K according to the table we presented above. So, when we use K nearest neighbor classifier later in this study, to evaluate the performance on database, we will take that value of K, which presented in that table.

Threshold Selection for Logistic Regression

In selecting the decision boundary during prediction for logistic regression classifier, we used threshold selection technique to achieve the lowest error rate. After predicting the y i.e., y_{predict} using logistic model, we defined set of unique values with all possible points from y_{predict} . After defining the threshold array, we added borders to both sides. Then we checked all distinct values in threshold array one by one and calculated the error at each threshold. This

gave us the possible threshold where we achieved the least error. We called this threshold the best threshold. Presented below is the algorithm we produced for selecting the best threshold.

Threshold Selection (Dr. Evgeny Mirkes)

Algorithm:

Step1: threshold_selection(y, z)

Step2: threshold = distinct(z)

Step3: threshold = (threshold[1:] + threshold[: -1]) / 2

Step4: best_error = n, best_threshold = 0

Step5: for x in threshold:

Step6: error = sum(z != y)

Step7: if error < best_error :

Step8: best_error = error

Step9: best_threshold = x

Step10: return best_threshold

Here, z predicted continuous value by logistic regression classifier, y actual values, sum (z!=y) will give a number of incorrect classified examples by logistic regression classifier. While distinct function will take vector as input and will provide a vector with distinct values. In producing this we used sklearn implementation of Logistic regression(sklearn) for training and used our predict function for prediction.

For instance, we took a climate database and illustrated how we achieved different balanced accuracy at different thresholds or decision boundaries.

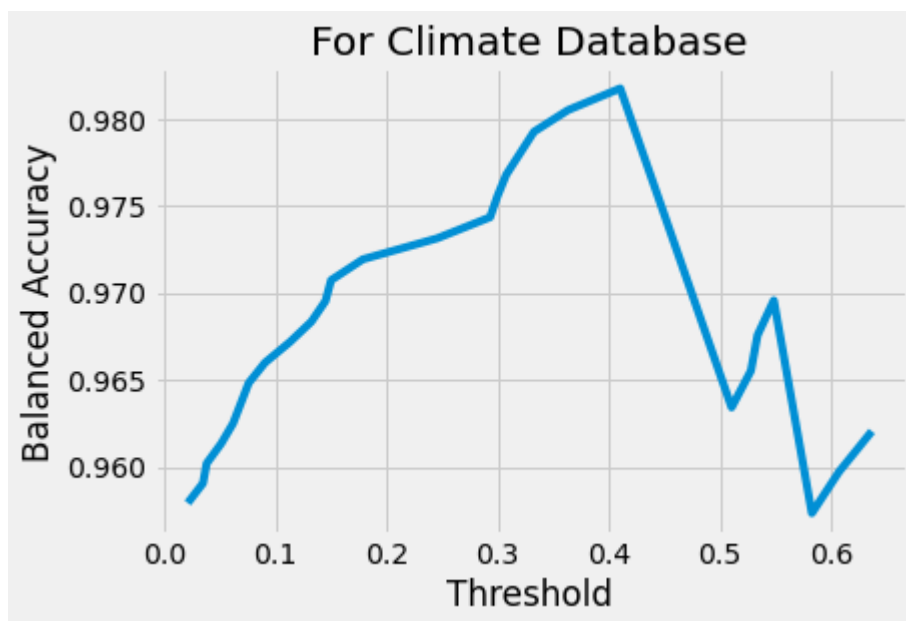


Figure7. Threshold for Logistic: Climate Database

- As we can see from the above plot, at threshold around 0.4, our logistic regression classifier achieved the highest balanced accuracy which means different threshold will lead to different performance.

Why different thresholds for different databases?

In this demonstration, we choose database namely Climate and Breast Cancer and calculated balanced accuracy at different thresholds.

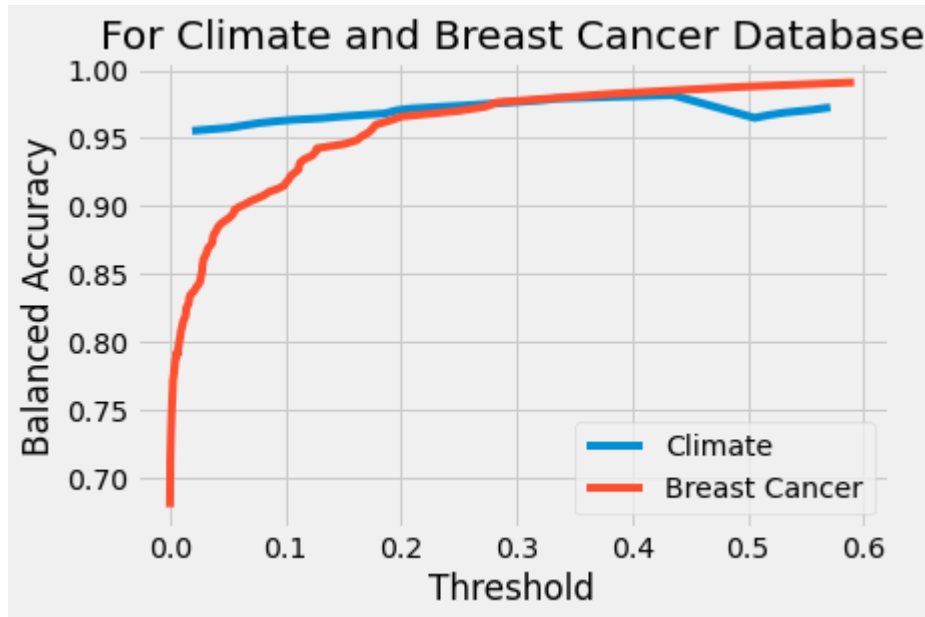


Figure8. Threshold for Logistic: 2 Databases

- From the above plots, we can conclude that optimal threshold will be different for different databases as for each database classifier achieved highest balanced accuracy at different thresholds.

Similarly, we took three databases namely Climate, Breast Cancer and Blood database and calculated balanced accuracy for all three databases at different thresholds.

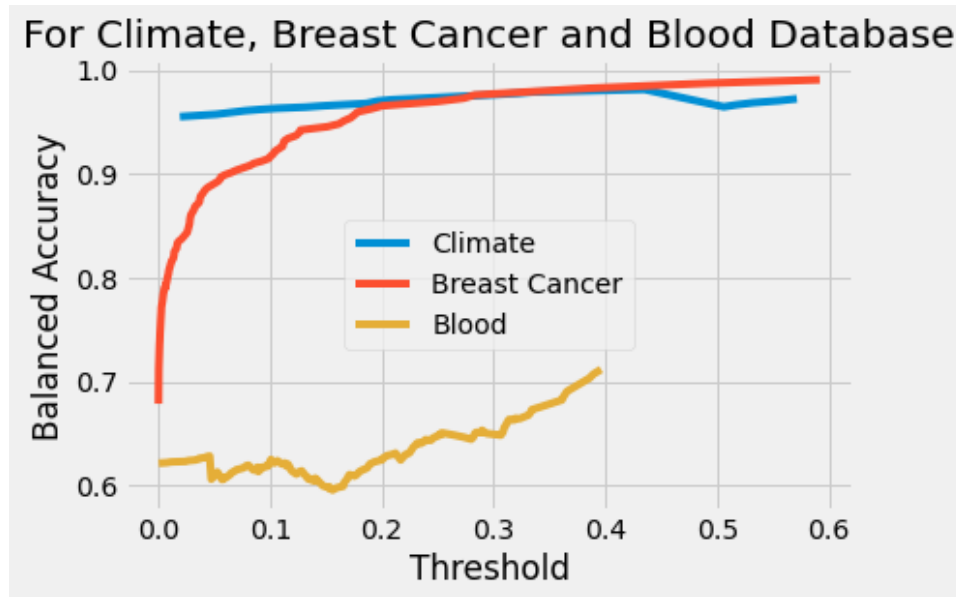


Figure9. Threshold for Logistic: 3 Databases

- As from the above plots, for all the three databases, they achieved their best results at different thresholds which clearly guarantee that threshold selection will be different and database dependent.
- Apart from these, we can't use the same threshold at each fold during performance evaluation using stratified cross validation for that database.

Why will we calculate different thresholds at each fold for a single database?

To demonstrate that threshold should be different at each fold and should be calculated at each fold separately, we took a database namely Climate and will show in next section that our classifier logistic regression will achieve highest balanced accuracy at different thresholds, or our best threshold will be different at each fold for getting highest balanced accuracy for logistic regression classifier.

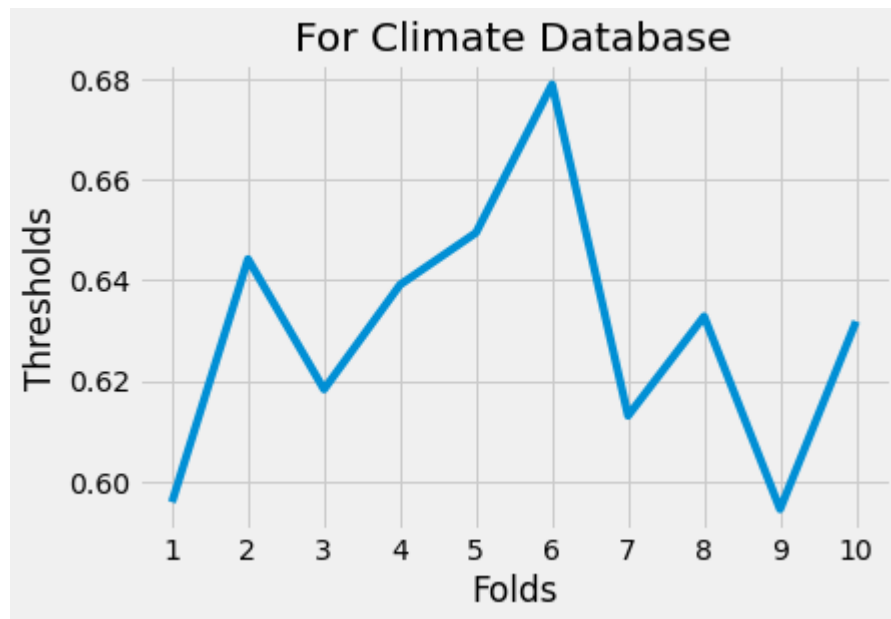


Figure10. Threshold for Logistic: Folds in Climate Database

- As we can see from the above plot, at each fold, the highest balanced accuracy for that fold is achieved at different thresholds. From the above plots, we also can conclude that the best threshold regions for these 10 folds for climate Database is between 0.55 to 0.68

From all the demonstrations and discussions, we can conclude that threshold selection will be database dependent then fold dependent. So, when we logistic regression classifier we will calculate threshold selection at each fold before prediction irrespective of database or folds.

Threshold Selection for Fisher Discriminant Classifier

Similarly, we used the same algorithm for threshold selection in Fisher Discriminant Classifier as we used in Logistic Regression, because performance of Fisher Classifier also depends on the selection of threshold. For instance, we took a database Climate and will calculate balanced accuracy across the possible thresholds produced by our threshold selection algorithm, then we will see at which threshold we will get the best-balanced accuracy.

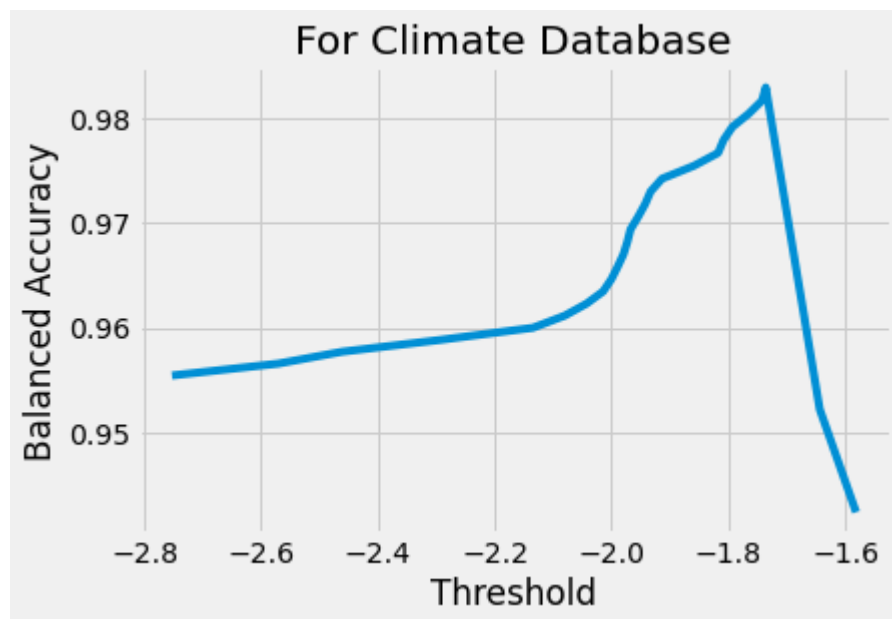


Figure12. Threshold for Fisher: Climate Database

- As we can see from the above plots, our classifier Fisher achieved different balanced accuracy at different thresholds selection, so we will use that threshold during prediction which will give us balanced accuracy. The highest balanced accuracy of more than 98% was achieved at around -1.7 while if we move towards 0, the Fisher Discriminant classifier experienced a huge decline in balanced accuracy.

Why different thresholds for different databases?

Similarly, as we demonstrated in Logistic Regression, we chose the same database namely Climate and Breast Cancer and calculated balanced accuracy at different thresholds using Fisher classifier.

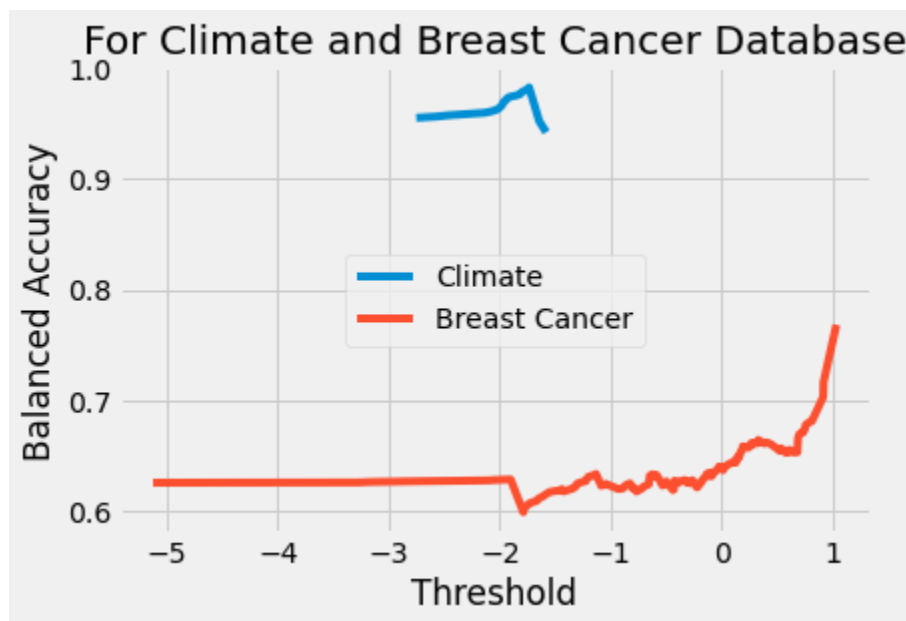


Figure13. Threshold for Fisher: 2 Databases

- From the above plot, we can conclude that there was a huge difference between optimal threshold for both datasets, as we can see optimal threshold for breast cancer dataset is 1 where Fisher Discriminant classifier achieved a highest balanced accuracy of nearly 80%, while for Climate Dataset optimal threshold was nearly -2 where classifier was able to achieve highest balanced accuracy nearly 99%. It showed clearly that optimal threshold would be different for any database.

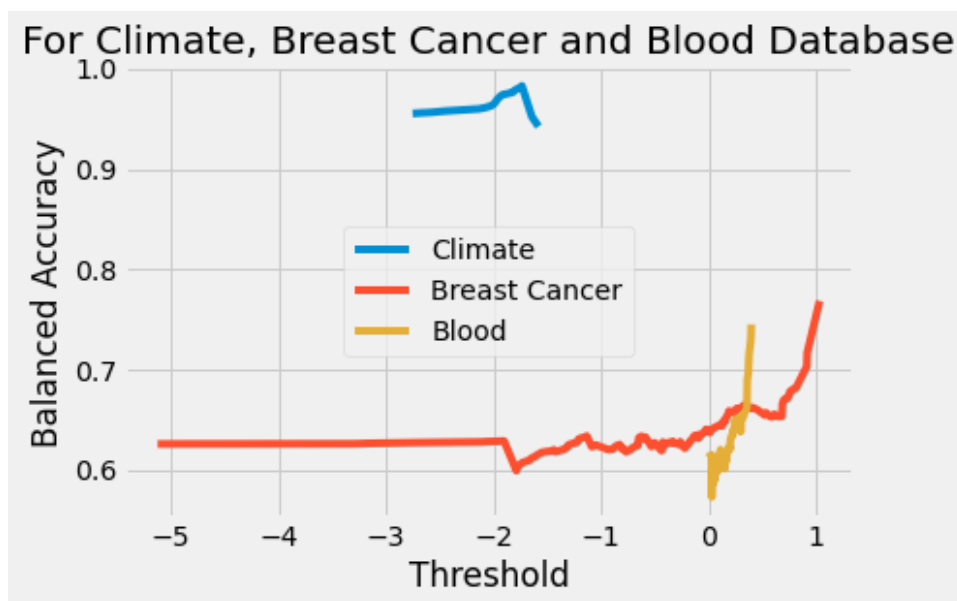


Figure14. Threshold for Fisher: 3 Databases

- In this plot, we added another database namely Blood database for calculating optimal threshold for database separately to achieved highest balanced accuracy. In this plot we can see for blood database, optimal threshold was around 0.5 to achieved highest balanced accuracy which was far away from the first two database.

So, the above demonstration proved that the optimal threshold for achieving highest balanced accuracy would be different on different databases for the Fisher Discriminant Classifier. So, we need to calculate optimal threshold for each database separately during classification for performance measurement.

Why will we calculate different thresholds at each fold for a single database?

To demonstrate that threshold should be different at each fold and should be calculated at each fold separately, we took a database namely Climate and will show in next section that our classifier logistic regression will achieve highest balanced accuracy at different thresholds, or our best threshold will be different at each fold for getting highest balanced accuracy for logistic regression classifier.

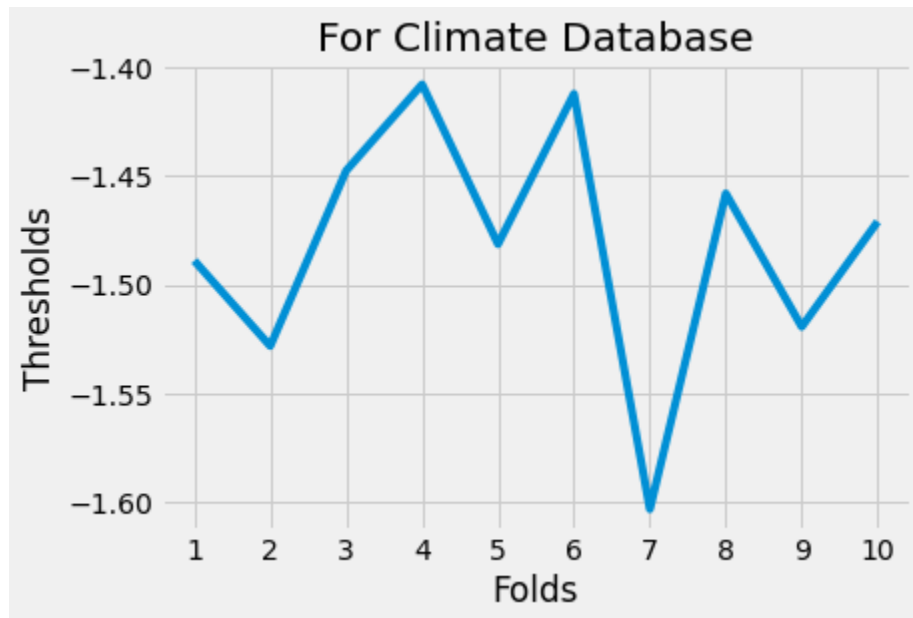


Figure12. Threshold for Fisher: Folds in Climate Database

- The above graph showed that at first fold optimal threshold was -1.48 , while at second fold it was a slight increase to getting highest balanced accuracy. For the third fold it was -1.44 , fourth fold needed the largest threshold to achieve highest balanced accuracy, while for seventh fold, the optimal threshold was the smallest in all the folds, it was -1.60 to achieve the highest balanced accuracy at this fold.

As we can see from the above plot on classification by Fisher Discriminant Classifier on Climate Database, at each fold we need to calculate optimal or best threshold to achieve the highest balanced accuracy. From that demonstration we can clearly conclude that there is a need to calculate optimal threshold at each fold separately for same database as well for better classification as we did for logistic regression classifier. As in logistic regression, similarly for Fisher Discriminant Classifier we can say that optimal threshold will be database dependent then fold dependent as well. So, it will be good practice to calculate the best threshold for each sample before prediction for Logistic Regression Classifier as well as for Fisher Discriminant Classifier.

Why did we use Stratified Cross Validation technique and Balanced Accuracy during performance evaluation?

Stratified Cross Validation is used to maintain the ratios of every classes samples in all folds equally to remove the threat of biases while balanced accuracy gives the quality of our model in predicting positive and negative classes as well.

Create Folds

First, before evaluating the performance of each classifier on each database, we will create the folds for each database separately and save them internally using their indices. Saving the indices for folds locally ensured us that for any database we used same fold with same samples for our performance by any classifier which all the three classifiers will be evaluated on same folds with same samples. So, it will avoid any kind of random splitting during performance for any classifier. For instance, a dataset Climate will first be splitting into 10 folds using Stratified Cross Validation(sklearn), during splitting we saved the indices of samples belonging to each fold and saved them different files and gave naming basis on the folds. Presented below

Algorithm to create folds and save them locally

```
# create folds and write their indices to HDD

def save_folds(x, y, database):

    # create dataframe df and later join vectors in x and vector y

    df = pd.DataFrame(x)

    df['y'] = y

    # create directory for saving folds
```

```

os.mkdir('Folds-Databases/' + database)

# initialize Stratified K fold cross validation
skf = StratifiedKFold(n_splits=10, shuffle=False)

# looping over folds
fold_no = 1
for train_index, test_index in skf.split(df, y):
    # slicing and save indices to HDD
    np.savetxt('Folds-Databases/' + database + '/train_fold_' + str(fold_no) + '.txt', train_index)
    np.savetxt('Folds-Databases/' + database + '/test_fold_' + str(fold_no) + '.txt', test_index)
    fold_no = fold_no + 1

```

Performance Evaluation

Finally, we will calculate the balanced accuracy for all three heuristics approach using all the three classifiers one by one and save them using 10 times 10 folds cross validation to showed that

results we got are not random. We also saved balanced accuracy at every iterations to save 100 size sample for all methods to perform p testing further.

We automated this task and produced an algorithm, one such algorithm

Algorithm: Performance Evaluation

```
def balanced_accuracy(file, dimensions, k):  
    # load database  
    data = loadmat('Databases/' + file)  
    x = data['X']  
    y = data['Y']  
  
    # standardise  
    x = zscore(x)  
  
    # transformation  
    pca = PCA()  
    dim = dimensions.loc[dimensions['Databases'] == file]  
    x_kaiser = pca.transformation(x, dim["PCA-K"].values[0])  
    x_bs = pca.transformation(x, dim["PCA-BS"].values[0])  
    x_cn = pca.transformation(x, dim["PCA-CN"].values[0])  
  
    # n repetitions on stratified K fold cross validation  
    n = 10  
  
    total_balanced_accuracy = 0  
    total_balanced_accuracy_k = 0  
    total_balanced_accuracy_bs = 0  
    total_balanced_accuracy_cn = 0
```

```

b_full = []
b_kaiser = []
b_bs = []
b_cn = []

counter = 0

for i in range(0, n):

    balance_accuracy = 0
    balanced_accuracy_k = 0
    balanced_accuracy_bs = 0
    balanced_accuracy_cn = 0

    # stratified 10-fold cross validation
    for j in range(1, 11):
        # selecting indices of training set
        train_set = pd.read_csv('Folds-Databases/' + file + '/train_fold_' + str(j) + '.txt',
                                header=None)
        train_index = train_set.to_numpy()
        mt, nf = train_index.shape
        train_index = np.reshape(train_index, mt)
        train = list(train_index)
        train = [int(item) for item in train]

        # selecting indices of test set for jth fold
        test_set = pd.read_csv('Folds-Databases/' + file + '/test_fold_' + str(j) + '.txt',
                                header=None)
        test_index = test_set.to_numpy()

```

```

mv, nf = test_index.shape
test_index = np.reshape(test_index, mv)
test = list(test_index)
test = [int(item) for item in test]

# getting x, y for current folds from indices above
x_train = np.take(x, train, axis=0)
x_test = np.take(x, test, axis=0)

x_train_k = np.take(x_kaiser, train, axis=0)
x_test_k = np.take(x_kaiser, test, axis=0)

x_train_bs = np.take(x_bs, train, axis=0)
x_test_bs = np.take(x_bs, test, axis=0)

x_train_cn = np.take(x_cn, train, axis=0)
x_test_cn = np.take(x_cn, test, axis=0)

y_train = np.take(y, train, axis=0)
y_test = np.take(y, test, axis=0)

# without dimensionality reduction
model1 = KNN(k=k)
model1.fit(x_train, y_train)
# predict on test set
y_test_predict = model1.predict(x_test)
acc1 = balanced_accuracy_score(y_test, y_test_predict)
balance_accuracy = balance_accuracy + acc1

```

```

b_full.append(acc1)

# for kaiser
model2 = KNN(k=k)
model2.fit(x_train_k, y_train)
# predict on test set
y_test_predict_k = model2.predict(x_test_k)
acc2 = balanced_accuracy_score(y_test, y_test_predict_k)
balanced_accuracy_k = balanced_accuracy_k + acc2
b_kaiser.append(acc2)

# for BS
model3 = KNN(k=k)
model3.fit(x_train_bs, y_train)
y_test_predict_bs = model3.predict(x_test_bs)
acc3 = balanced_accuracy_score(y_test, y_test_predict_bs)
balanced_accuracy_bs = balanced_accuracy_bs + acc3
b_bs.append(acc3)

# for CN
model4 = KNN(k=k)
model4.fit(x_train_cn, y_train)
y_test_predict_cn = model4.predict(x_test_cn)
acc4 = balanced_accuracy_score(y_test, y_test_predict_cn)
balanced_accuracy_cn = balanced_accuracy_cn + acc4
b_cn.append(acc4)

total_balanced_accuracy = total_balanced_accuracy + balanced_accuracy / 10

```

```

total_balanced_accuracy_k = total_balanced_accuracy_k + balanced_accuracy_k / 10
total_balanced_accuracy_bs = total_balanced_accuracy_bs + balanced_accuracy_bs / 10
total_balanced_accuracy_cn = total_balanced_accuracy_cn + balanced_accuracy_cn / 10

```

```

counter = counter + 1

```

```

print(counter)

```

```

return [total_balanced_accuracy / 10,
        total_balanced_accuracy_k / 10,
        total_balanced_accuracy_bs / 10,
        total_balanced_accuracy_cn / 10, b_full, b_kaiser, b_bs, b_cn]

```

The following are the results we produced using the above processes:

Fisher

Databases	BA	BA-Kaiser	BA-Brok	BA-Cond
Banknote	0.9877	0.7848	0.7848	0.9137
Blood	0.5675	0.6017	0.6017	0.5675
Breast Cancer	0.9587	0.9507	0.9293	0.9501
Climate	0.8025	0.6354	0.499	0.8025
Cryotherapy	0.9025	0.785	0.805	0.9025
Diabetic	0.7241	0.5866	0.5888	0.6121
Immunotherapy	0.5393	0.5964	0.55	0.5393
Liver	0.5525	0.5166	0.4975	0.5456
Maledon	0.5558	0.5738	0.6135	0.57
Musk	0.7185	0.7218	0.6838	0.6874
Plrx	0.5046	0.4923	0.51	0.4869
Qsar	0.8401	0.7764	0.8022	0.7833
Sonar	0.6484	0.6539	0.6819	0.7007
Spect	0.6863	0.6838	0.6722	0.6684
Spectf	0.5914	0.6257	0.5424	0.6145
Vertebral	0.7926	0.6405	0.6481	0.7926

Table4. Performance Measure: Fisher

Ranking based on average

Databases	Full Dimension	Kaiser	Broken	Condition
Banknote	1	3.5	3.5	2
Blood	3.5	1.5	1.5	3.5
Breast Cancer	1	2	4	3
Climate	1.5	3	4	1.5
Cryotherapy	1.5	4	3	1.5
Diabetic	1	4	3	2
Immunotherapy	3.5	1	2	3.5
Liver	1	3	4	2
Maledon	4	2	1	3
Musk	2	1	4	3
Plrx	2	3	1	4
Qsar	1	4	2	3
Sonar	4	3	2	1
Spect	1	2	3	4
Spectf	3	1	4	2
Vertebral	1.5	4	3	1.5

Table4. Fisher: Average Based Ranking

Ranking based on p value

Databases	Full Dimension	Kaiser	Broke	Condition
-----------	----------------	--------	-------	-----------

Banknote	1	-1	-1	0
Blood	-1	1	1	-1
Breast Cancer	1	0	-1	0
Climate	1	0	-1	1
Cryotherapy	1	-1	0	1
Diabetic	1	-1	-1	0
Immunotherapy	-1	1	-1	-1
Liver	1	0	-1	1
Maledon	-1	0	1	0
Musk	1	1	-1	0
Plrx	1	-1	1	-1
Qsar	1	-1	0	-1
Sonar	-1	0	0	1
Spect	0	0	0	0
Spectf	0	1	-1	1
Vertebral	1	-1	-1	1

Table5. Fisher: p Based Ranking

Logistics

Databases	BA	BA-Kaiser	BA-Brok	BA-Cond
Banknote	0.9884	0.7725	0.7725	0.9175
Blood	0.5642	0.6036	0.6036	0.5642
Breast Cancer	0.9744	0.9684	0.944	0.9728
Climate	0.8334	0.6355	0.499	0.8334
Cryotherapy	0.8875	0.81	0.805	0.8875
Diabetic	0.7274	0.6024	0.5949	0.6308
Immunotherapy	0.6536	0.5393	0.55	0.6536
Liver	0.5611	0.5183	0.4975	0.5705
Maledon	0.5538	0.5742	0.6135	0.5681
Musk	0.7656	0.7361	0.6659	0.6743
Plrx	0.5162	0.4923	0.51	0.4931
Qsar	0.8438	0.8185	0.7945	0.8129
Sonar	0.6676	0.6842	0.6908	0.7117
Spect	0.6862	0.6669	0.6996	0.6858
Spectf	0.6801	0.6157	0.5553	0.6666
Vertebral	0.7755	0.6431	0.6481	0.7755

Table6. Performance Measure: Logistic

Ranking based on average

Databases	Full Dimension	Kaiser	Broken	Condition
Banknote	1	3.5	3.5	2
Blood	3.5	1.5	1.5	3.5
Breast Cancer	1	3	4	2
Climate	1.5	3	4	1.5
Cryotherapy	1.5	3	4	1.5
Diabetic	1	3	4	2
Immunotherapy	1.5	4	3	1.5
Liver	2	3	4	1
Maledon	4	2	1	3
Musk	1	2	4	3
Plrx	1	4	2	3
Qsar	1	2	4	3
Sonar	4	3	2	1
Spect	2	4	1	3
Spectf	1	3	4	2
Vertebral	1.5	4	3	1.5

Table7. Logistic: Average Based Ranking

Ranking based on p value

Databases	Full Dimension	Kaiser	Broken	Condition
Banknote	1	-1	-1	0
Blood	-1	1	1	-1
Breast Cancer	1	0	-1	1
Climate	1	0	-1	1
Cryotherapy	1	-1	-1	1
Diabetic	1	-1	-1	0
Immunotherapy	1	-1	-1	1
Liver	1	0	-1	1
Maledon	-1	0	1	0
Musk	1	0	-1	-1
Plrx	1	-1	1	-1
Qsar	1	0	-1	0
Sonar	0	0	0	0
Spect	0	0	0	0
Spectf	1	0	-1	1
Vertebral	1	-1	1	1

Table8. Logistic: p Based Ranking

KNN

Databases	Best K	BA	BA-Kaiser	BA-Broken	BA-Cond
Banknote	17	0.9967	0.859	0.859	0.9618
Blood	23	0.5953	0.5689	0.5689	0.5953
Breast Cancer	5	0.9589	0.9584	0.9293	0.9466
Climate	5	0.637	0.517	0.499	0.637
Cryotherapy	7	0.925	0.8075	0.8075	0.925
Diabetic	29	0.6564	0.6212	0.6215	0.6405
Immunotherapy	5	0.6607	0.5402	0.4804	0.6607
Liver	1	0.5842	0.5806	0.5948	0.5842
Maledon	17	0.5696	0.5896	0.5985	0.5746
Musk	5	0.821	0.8177	0.7636	0.7216
Plrx	1	0.5174	0.4545	0.5272	0.5274
Qsar	7	0.8312	0.8376	0.828	0.8376
Sonar	1	0.6989	0.6809	0.6852	0.7235
Spect	25	0.7586	0.7479	0.6972	0.7677
Spectf	11	0.5885	0.612	0.5709	0.6525
Vertebral	1	0.7995	0.5948	0.6379	0.7995

Table9. Performance Measure: KNN

Ranking based on average

Databases	Full Dimension	Kaiser	Broke	Condition
Banknote	1	3.5	3.5	2
Blood	1.5	3.5	3.5	1.5
Breast Cancer	1	2	4	3
Climate	1.5	3	4	1.5
Cryotherapy	1.5	3.5	3.5	1.5
Diabetic	1	4	3	2
Immunotherapy	1.5	3	4	1.5
Liver	2.5	4	1	2.5
Maledon	4	2	1	3
Musk	1	2	3	4
Plrx	3	4	2	1
Qsar	3	1.5	4	1.5
Sonar	2	4	3	1
Spect	2	3	4	1
Spectf	3	2	4	1
Vertebral	1.5	4	3	1.5

Table10. KNN: Average Based Ranking

Ranking based on p value

Databases	Full Dimension	Kaiser	Broken	Condition
Banknote	1	-1	-1	0
Blood	0	0	0	0
Breast Cancer	1	1	-1	0
Climate	1	0	-1	1
Cryotherapy	1	-1	-1	1
Diabetic	1	-1	-1	0
Immunotherapy	1	0	-1	1
Liver	0	0	0	0
Maledon	-1	0	1	-1
Musk	1	1	0	-1
Plrx	1	-1	1	1
Qsar	0	0	0	0
Sonar	0	-1	-1	1
Spect	1	1	-1	1
Spectf	-1	0	-1	1
Vertebral	1	-1	0	1

Table10. KNN: p Based Ranking

Where in p-based Ranking :

1. 1 means statistically insignificantly different from the best
2. 0 means statistically significantly different from the best and the worst
3. -1 means statistically insignificantly different from the worst

10. Conclusion

Performance measure using each classifier on all databases after applying reduction using these 3 heuristics approach demonstrated that conditional number was in all the 3. But based on this, we can't conclude that conditional number will always be our intrinsic dimension because we also found that when other 2 heuristic to retain principal components were bigger, classifiers abled to achieve highest balanced accuracy which means that. There is no universal answer to the question which of the retaining component is best, but most of the times in our study, conditional outperform the kaiser and broken stick rule.

10. References

James et al. 2013

<https://www.geo.fu-berlin.de/en/v/soga/Geodata-analysis/Principal-Component-Analysis/principal-components-basics/Choose-principal-components/index.html>

Jackson, D. A. (1993). Stopping rules in principal components analysis: a comparison of heuristically and statistical approaches. *Ecology* 74, 2204–2214.

Pacheco, Joaquín & Casado, Silvia & Porras, Santiago. (2013). Exact Methods for Variable Selection in Principal Component Analysis: Guide Functions and Pre-Selection. *Computational Statistics & Data Analysis*. 57. 95-111. 10.1016/j.csda.2012.06.014.

<https://medium.com/analytics-vidhya/the-math-of-principal-component-analysis-pca-bf7da48247fc>

https://docs.displayr.com/wiki/Kaiser_Rule#Kaiser_Rule

https://www.mohanwugupta.com/post/broken_stick/

<https://www.cs.huji.ac.il/w~csip/Fisher-LDA.pdf>

Berrar, Daniel. (2018). Cross-Validation. 10.1016/B978-0-12-809633-8.20349-X.

<https://towardsdatascience.com/what-is-stratified-cross-validation-in-machine-learning-8844f3e7ae8e>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.zscore.html>

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.LeaveOneOut.html

<https://github.com/Mirkes?tab=repositories>


https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html


https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

11. APPENDIX

LinkedIn Posts

Post 1st






Mohit Baliyan (He/Him) • You
Credit Risk Analyst
4d • 

...


Hello 1st, 2nd and 3rd+ connections,


I am really glad to announce that I have been working on one of the major problems in classical machine learning as my final project. Dimensionality Reduction is one of the major sub problems of Data Mining's 6 historical problems. Principal Component Analysis is one of the mathematical tools which is widely used to project high-dimensional data into lower-dimensional data. But, there is no validated choice for selecting the number of principal components to represent the original data.


I would like to thank Dr. [Evgeny Mirkes](#) and [University of Leicester](#) for giving me this opportunity to work and for their great support.


   Shubham Agarwal and 41 others


8 comments

 Like

 Comment


 Share


 Send

 2,532 impressions

[View analytics](#)

Post 2nd



Mohit Baliyan (He/Him) • You
Credit Risk Analyst
4d • 

...

Hi connections,

It has been really fun working with data and it is cherry on top when you get to play with data where your interest lies. This is my update so far on project I have been working on as I mentioned in my previous post and this post I am giving updates what I have learnt so far.

There are three main heuristics to select number of informative (useful, meaningful, etc.) component namely Kaiser, Broken Stick and Conditional Number and used more than 20 databases in this study.

We used 3 classifiers for performance measurement:

1. KNN
2. Logistic Regression
3. Fisher Discriminant Analysis

For selecting K for each dataset, we used leave one out cross validation technique.


For logistic Regression, we use threshold selection technique to select the best decision boundary during prediction at each fold and each iteration.


Steps:


1. Define a set of unique values with all possible points.
2. Add all borders
3. Select threshold which gives the least error rate


For performance metrics, we used balanced accuracy, it was calculated for 10 times repeated 10 fold cross validation.


I would like to thank you [Evgeny Mirkes](#) for thier support throughout project.


 Dinesh Khatri and 18 others

 Like

 Comment


 Share

 Send

 1,662 impressions

[View analytics](#)

Post 3rd



Mohit Baliyan (He/Him) • You
 Credit Risk Analyst
 4d •

Hello Connections,

I am glad to announce that the project I have been working is finished. I have got some amazing results in the end of this project which I will be uploading as an open source link to have a look.

We used 2 ranking protocols to rank each dimensionality reduction method.

1. Average Ranking
2. Ranking based on test of statistical significance of differences of performances

For p-value based ranking :

- > 1 means statistically insignificantly different from the best
- > 0 means statistically significantly different from the best and the worst.
- > -1 means statistically insignificantly different from the worst

There is no one universal answer which of intrinsic dimension is better. In average, dimensionality reduction on base of conditional number is better.

In the end I would like to thank my professor for supporting through out. Please feel free to connect if any connection is interested in discussion about similar kind of project.

Databases	BA	BA-kaiser	BA-trunk	BA-cond	Ranking based on average				Ranking based on p value			
					Full dim	Kaiser	Broken	Condition	Full dim	Kaiser	Broken	Condition
Banknote.mat	0.9877	0.7848	0.7848	0.9137	1	3.5	3.5	2	1	-1	-1	0
breast.mat	0.9679	0.6017	0.6017	0.5075	3.5	1.5	1.5	3.5	-1	1	1	-1
breastCancer.mat	0.9587	0.9507	0.9293	0.9501	1	2	4	3	1	0	-1	0
climate.mat	0.8025	0.6354	0.499	0.8025	1.5	3	4	1.5	1	0	-1	1
Cryotherapy.mat	0.9025	0.785	0.805	0.9025	1.5	4	3	1.5	1	-1	0	1
diabetic.mat	0.7241	0.5866	0.5868	0.8123	1	4	3	2	1	-1	-1	0
Immunotherapy.mat	0.5393	0.5964	0.55	0.5393	3.5	1	2	3.5	-1	1	-1	-1
liver.mat	0.5525	0.5166	0.4975	0.5456	1	3	4	2	1	0	-1	1
malaren.mat	0.5558	0.5738	0.6135	0.57	4	2	1	3	-1	0	1	0
Musk.mat	0.7185	0.7218	0.6838	0.6874	2	1	4	3	1	1	-1	0
plrx.mat	0.5046	0.4923	0.51	0.4869	2	3	1	4	1	-1	1	-1
qm.mat	0.8401	0.7764	0.8022	0.7831	1	4	2	3	1	-1	0	-1
sonar.mat	0.6484	0.6139	0.6819	0.7067	4	3	2	1	-1	0	0	1
spect.mat	0.6883	0.6838	0.6722	0.6684	1	2	3	4	0	0	0	0
vehicle.mat	0.5914	0.6257	0.5424	0.6145	3	1	4	2	0	1	-1	1
vehicle.mat	0.7938	0.6405	0.6481	0.7938	1.5	4	3	1.5	1	-1	-1	1

Shubham Agarwal and 20 others

3 comments

Like Comment Share Send

1,559 impressions View analytics

GitHub username - mohit-baliyan

Repository - [dimensionality-reduction](#)

[Access - Professor Evgeny Mirkes is added as a collaborator](#)