Subspace Classifier

Final Report

Vicky Wu
1st year Master In Data Science
Graduate Center, CUNY
bwu1@gradcenter.cuny.edu

Álvaro Faúndez
1st year Master In Data Science
Graduate Center, CUNY
afaundezreyes@gradcenter.cuny.edu

***Abstract*—In this report, we show the construction, training, testing, and the experimentation of a subspace classifier with Mahalanobis distances. The main methods applied in the building include essential feature selection, principal components analysis, and classification with Mahalanobis distances. Each process is implemented in combination with a parameter to control the desired output so that the classifier can be optimized. The Introduction and Technical section of the report explains the mathematical theories guiding the subspace selection and classification process. The Procedure section demonstrates the step by step computation for essential feature selection, subspace selection, feature projection onto the subspace and classification. The experimentation section reports the classifier's performance when tested on three different real-world data sets.**

***Index Terms*—machine learning, classifier, subspace, Mahalanobis, projection**

I. Introduction

Subspace classification is a technique for pattern identification that considers a N-dimensional space and a multi-classification domain. It creates a subspace for each class, based on the eigenvectors of the original space, and assigns a class to a measurement measuring the distance between the measurement to the mean of each subspace.

This report details the design and implementation of a classifier under the concept of subspaces and Mahalanobis distances. The training of the classifier includes the selection of essential features for each class and projecting measurements onto each class's subspace.

The classifier defines three parameters to be experimented with. The parameter $/theta$ sets the threshold of the probability of correct classification using a single feature classifier during essential feature selection. The parameter f determines the number of principal components kept for each class during finding subspace by determining the number of eigenvectors used. The parameter p_0 determines whether the distance from a data point to a subspace distribution is small enough for a data point to be considered for a class.

With three real-world data sets, we conducted experiments at the end of the report to show how different combinations of the three parameters could have interacting effects on classification accuracy for each class in each data set.

II. Technical

Given a space of all classes $C = \{c_1, \dots, c_K\}$, a classifier is a function $f : \mathbb{R}^N \to C$ that maps a measurement $\vec{x} \in \mathbb{R}^N$ to a class $c \in C$ [Equation 1].

$$\begin{aligned} f \colon \mathbb{R}^N &\to C \\ \vec{x} &\mapsto c \end{aligned} \tag{1}$$

A data set (X, y) is a sequence of Z tuples of measurements $\vec{x} \in \mathbb{R}^N$ and classes $c \in C$. The sequenced set X contains the measurements, and the corresponding sequenced set y contains the true class labels [Equation 2].

$$\begin{aligned} X &= \{\vec{x}_1, \dots, \vec{x}_Z\}, \, \vec{x}_z \in \mathbb{R}^N \\ y &= \{y_1, \dots, y_Z\}, y_i \in C \\ |X| &= |y| = Z \end{aligned} \tag{2}$$

The classifier trains on a training set (X^{train}, y^{train}) tests on a testing set (X^{test}, y^{test}). The test set must be independent of the training set.

A. Essential features

A feature, or column, represents a measurable piece of data that can be used for analysis. We want to select essential features for each class to help optimize the classifier's performance. In this process, we consider features that have a strong association with a particular class as essential. The idea is to modify the training set by filtering only essential features for each class by finding a way to measure each feature's association with each class and comparing the degree of association with a user-defined threshold.

Given a measurement space with N features $I = \{1, \dots, N\}$. Each measurement $\vec{x}$ is composed of N dimensions/features. Given a data set (X, y) and a feature $iinI$, the function ϕ transforms the measurements X into a measurements containing only the i-th feature. The data set $(\phi(X, i), y)$ is defined as a single feature data set [Equation].

$$\begin{aligned} \phi \colon \mathbb{R}^{N \times Z} \times I &\to \mathbb{R}^{Z \times 1} \\ (X, i) &\mapsto \{x_i \mid \vec{x} = (x_1, \dots, x_n), \forall \, \vec{x} \in X\} \end{aligned} \tag{3}$$

To determine the degree of association between a feature with index $i \in I$ and a class $c \in C$, a classifier f_i is trained with the single feature data set $(\phi(X^{train}, i), y^{train})$ so that it predicts a class label for each measurement in $\phi(X^{test}, i))$.

Using the predictions y^{pred} and the true classes y^{test}, the accuracy of the single feature classifier f_i over each pair of

(c, i) is calculated and stored in the matrix $A^{K \times N}$ [Equation].

$$A[c,i] = \frac{|\{y_z^{pred} \mid y_z^{pred} = y_z^{test} = c, z = \{1, \ldots, Z\}\}|}{|\{y_z^{test} \mid y_z^{test} = c, z = \{1, \ldots, Z\}\}|} \quad (4)$$

Feature i is defined an essential feature for class c when the probability of correct classification $A[c, i]$ is above the threshold θ. Applying this definition, every class $c \in C$ from the data set is eventually assigned a set of essential features indexes Q_c. [Equation 5].

$$Q_c = \{i \mid A[c,i] > \theta \forall i \in I\} \quad (5)$$

B. Class sets

Given a set of measurement tuples X and their corresponding classes y, the classifier splits the data set into subsets of each individual class $X_c = \{\vec{x_z} \in X \mid y_z = c\}$ for each $c \in C$.

C. Projection and principal components

Selecting principal components is a further step to reduce data dimension to assure that there is little or no redundant information in our data in the classification step. Unlike selecting essential features, in principal component analysis, we do not preserve the features themselves but projecting the features onto a selected subspace with fewer dimensions.

Given the measurement tuples of a data set X, we can calculate the mean vector $\vec{\mu}$ and the covariance matrix $\Sigma^{N \times N}$ of X [Equation 6].

$$\begin{aligned} \vec{\mu} &= \frac{1}{Z} \sum_{i=1}^{Z} \vec{x}_i \\ \Sigma^{N \times N} &= \frac{1}{Z-1} \sum_{i=1}^{Z} (\vec{x}_i - \vec{\mu})(\vec{x}_- \vec{\mu})^t \end{aligned} \quad (6)$$

The covariance matrix is factorized into its eigenvalues and eigenvectors $(\lambda_1, \vec{v}_1), \ldots, (\lambda_N, \vec{v}_N)$ where $\lambda_i \geqslant \lambda_{i+1}$, $1 \leqslant i < N$

$$\Sigma = V \Lambda \, V^{-1} \quad (7)$$

The matrix $\Lambda^{N \times N}$ is a diagonal matrix where each value on the diagonal is an eigenvalue.

$$\Lambda^{N \times N} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \quad (8)$$

The matrix $V^{N \times N}$ is formed with each eigenvector as a column.

$$V^{N \times N} = \begin{bmatrix} \vec{v}_1 & \ldots & \vec{v}_N \end{bmatrix} \quad (9)$$

The projection subspace $S^{N \times E}$ is defined by selecting the eigenvectors associated with the minimum amount of the eigenvalues that could explain a percentage of the variance of the covariance matrix. The eigenvalues are sorted in descending order. We add up each eigenvalue one by one until the sum of the selection over the sum of all eigenvalues is above a threshold f.

$$\begin{aligned} S^{N \times E} &= \begin{bmatrix} \vec{v}_1 & \ldots & \vec{v}_E \end{bmatrix} \\ &\sum_{i=1}^{E} \lambda_i \geqslant f \end{aligned} \quad (10)$$

The new subspace $\pi(X)$ is the result of projecting X onto S^T.

$$\pi(X)^{Z \times E} = S^t X \quad (11)$$

The principal component analysis helps to further reduce the noise in our data so that the transformed measurement tuples are more likely to be assigned to their true classes.

Given a data set X and y, we split X by y and project each X_c onto its subspace S_c.

$$X_c = \{\vec{x}_z \in X \mid y_z = c\} \quad (12)$$

$$\pi(X_c) = S_c^t X_c \quad (13)$$

D. Distance

We use Mahalanobis distance to measure how close one measurement tuple $\vec{x}$ is to the distribution of a set of measurement tuples X with mean of $\vec{mu}$ and covariance matrix Σ [Equation 14].

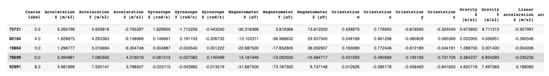
$$d_X(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu}) \Sigma^{-1} (\vec{x} - \vec{\mu})} \quad (14)$$

III. Assumptions

- Dimension of tuples should be between 10 and 20
- Number of classes should be between 4 and 10
- Values of features are real continuous numbers
- Number of tuples in training set for each class is no less than ten times of parameters needed by the classifier

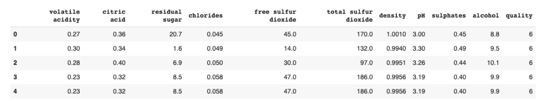
IV. Problems encountered and Solutions

- Implementation problems
  1) Class names do not match class indexes: this will raise index error when we iterate through classes.
  2) Dimensional error during single feature selection because we decided to use Mahalanobis distance classifier for single feature selection purposes as well.
- Special cases with the data
  1) No variance between classes: the values of the feature are the same for all the classes. In this case, we should delete this feature from the data set because the feature does not help to separate different classes.
  2) No variance within classes: the values of the feature are the same within one class but different for different classes. In this case, we should keep this feature because it is essential in separating



	Coarse label	Acceleration X [m/s2]	Acceleration Y [m/s2]	Acceleration Z [m/s2]	Gyroscope X [rad/s]	Gyroscope Y [rad/s]	Gyroscope Z [rad/s]	Magnetometer X [μT]	Magnetometer Y [μT]	Magnetometer Z [μT]	Orientation w	Orientation x	Orientation y	Orientation z	Gravity X [m/s2]	Gravity Y [m/s2]	Linear acceleration X [m/s2]
75721	0.0	5.265799	5.929916	-2.765397	1.629955	-1.712259	-0.443250	-30.318366	3.818366	13.812500	0.439975	0.176955	-0.818595	-0.324045	5.973850	6.711313	-0.307997
80184	4.0	1.928875	4.263383	-9.158968	0.146661	0.191743	-0.206732	-12.102311	-36.988632	29.937500	0.249166	0.961298	-0.080828	0.085369	2.002905	4.556951	-0.390546
19864	0.0	1.296777	3.016694	-9.304748	0.004887	-0.003540	0.001222	-22.687500	-17.652805	38.932907	0.163089	0.772446	-0.612189	-0.044191	1.288700	3.001400	-0.004006
76699	0.0	5.094861	7.080592	4.316018	-0.061015	-0.027382	0.140499	14.181946	-13.062500	-15.584717	0.431093	0.490569	0.195195	0.731709	5.384237	6.954293	-0.035235
92991	8.0	4.981699	7.550141	3.788507	-0.023115	-0.043982	-0.013219	-31.687500	-73.187500	6.157148	0.012626	-0.285178	-0.458463	-0.841620	4.820718	7.497563	0.188080

Fig. 1. The SHL Dataset: 5 samples snapshot.



	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

Fig. 2. White Wine data set: 5 samples snapshot.

classes. However, the inverse covariance matrix of this feature with no variance will be infinite, and the Mahalanobis distances will be Nan during the single feature selection, so we came up with the solution to force the Mahalanobis distances to be 0 when we encounter such a situation.

V. Procedure

A. Data

Three data sets were used to test the performance of the classifier. A summary of each data set is provided below:

1) The University of Sussex-Huawei Locomotion-Transportation (SHL) Dataset [3]. This data set has 9 classifications, 19 features and 100000 samples [Figure 1].
2) White Wine Dataset [5]. This data set has 6 classifications, 10 features and 4898 samples [Figure 2].
3) Avila Bible Dataset [2]. This data set has 12 classifications, 9 features and 10430 samples [Figure 3]

Each data set is randomly shuffled before split into training and testing sets of equal sizes [Table I].

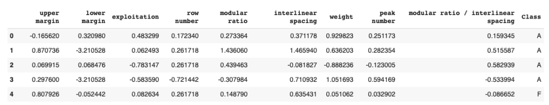
The training set's measurements are stored in X^{train}, and the classifications are stored in y^{train}. The testing set's measurements are stored in X^{test} and class tags are stored in y^{test}.

Data sets	Set samples		
	Train	*Test*	*Total*
SHL Dataset	50,000	50,000	100,000
White wine	2,449	2,449	4,898
Avila	5,215	5,215	10,430

TABLE I
Training and test sizes for each data set

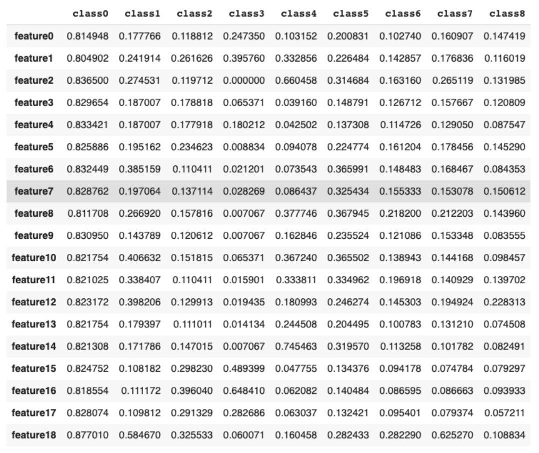
B. Single feature classification

For each feature i in I in the data set, we transform the original data set into a single feature data set $(\phi(X,i),y)$ [Equation 3].



	upper margin	lower margin	exploitation	row number	modular ratio	interlinear spacing	weight	peak number	modular ratio / interlinear spacing	Class
0	-0.165620	0.320980	0.483299	0.172340	0.273364	0.371178	0.929823	0.251173	0.159345	A
1	0.870736	-3.210528	0.062493	0.261718	1.436060	1.465940	0.636203	0.282354	0.515587	A
2	0.069915	0.068476	-0.783147	0.261718	0.439463	-0.081827	-0.888236	-0.123005	0.582939	A
3	0.297600	-3.210528	-0.583590	-0.721442	-0.307984	0.710932	1.051693	0.594169	-0.533994	A
4	0.807926	-0.052442	0.082634	0.261718	0.148790	0.635431	0.051062	0.032902	-0.086652	F

Fig. 3. Avila Bible data set: 5 samples snapshot.



	class0	class1	class2	class3	class4	class5	class6	class7	class8
feature0	0.814948	0.177766	0.118812	0.247350	0.103152	0.200831	0.102740	0.160907	0.147419
feature1	0.804902	0.241914	0.261626	0.395760	0.332856	0.226484	0.142857	0.176836	0.116019
feature2	0.836500	0.274531	0.119712	0.000000	0.660458	0.314684	0.163160	0.265119	0.131985
feature3	0.829654	0.187007	0.178818	0.065371	0.039160	0.148791	0.126712	0.157667	0.120809
feature4	0.833421	0.187007	0.177918	0.180212	0.042502	0.137308	0.114726	0.129050	0.087547
feature5	0.825886	0.195162	0.234623	0.008834	0.094078	0.224774	0.161204	0.178456	0.145290
feature6	0.832449	0.385159	0.110411	0.021201	0.073543	0.365991	0.148483	0.168467	0.084353
feature7	0.828762	0.197064	0.137114	0.028269	0.086437	0.325434	0.155333	0.153078	0.150612
feature8	0.811708	0.266920	0.157816	0.007067	0.377746	0.367945	0.218200	0.212203	0.143960
feature9	0.830950	0.143789	0.120612	0.007067	0.162846	0.235524	0.121086	0.153348	0.083555
feature10	0.821754	0.406632	0.151815	0.065371	0.367240	0.365502	0.138943	0.144168	0.098457
feature11	0.821025	0.338407	0.110411	0.015901	0.333811	0.334962	0.196918	0.140929	0.139702
feature12	0.823172	0.398206	0.129913	0.019435	0.180993	0.246274	0.145303	0.194924	0.228313
feature13	0.821754	0.179397	0.111011	0.014134	0.244508	0.204495	0.100783	0.131210	0.074508
feature14	0.821308	0.171786	0.147015	0.007067	0.745463	0.319570	0.113258	0.101782	0.082491
feature15	0.824752	0.108182	0.298230	0.489399	0.047755	0.134376	0.094178	0.074784	0.079297
feature16	0.818554	0.111172	0.396040	0.648410	0.062082	0.140484	0.086595	0.086663	0.093933
feature17	0.828074	0.109812	0.291329	0.282686	0.063037	0.132421	0.095401	0.079374	0.057211
feature18	0.877010	0.584670	0.325533	0.060071	0.160458	0.282433	0.282290	0.625270	0.108834

Fig. 4. Matrix A with single feature accuracies for SHL data set

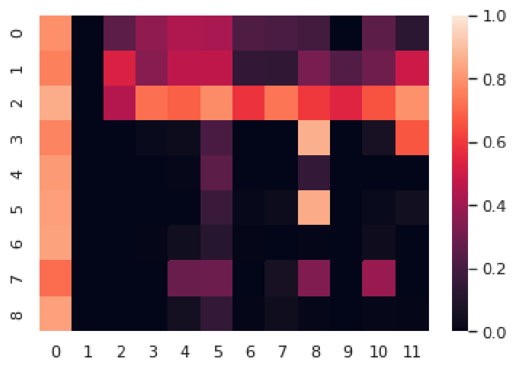


Fig. 5. Heatmap A with single feature accuracies for SHL data set

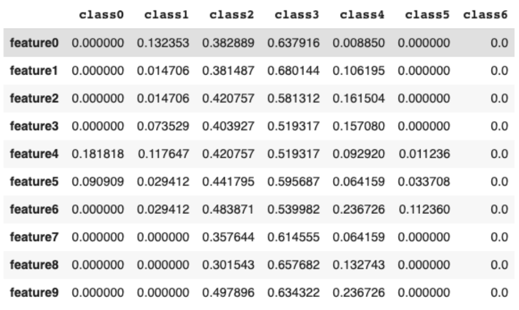
With the single feature data sets, we initialize a K-Nearest Neighbor (KNN) classifier named d_i using the SciKit-Learn package to train on all the samples $\phi(X^{train},i)$ and all the class tags y^{train}. In the KNN classifier, a number of neighbors are set at 5.

After training, we use $\phi(X^{test},i)$ to calculate the probability of correct classification for each classification c. The resulting matrix $A^{K\times N}$ [Figures 4, 6 and 8] stores all the probabilities for correct classification for each true class c when using only feature i, denoted as $A(c,|\,i)$ [Equation 15].

$$A[c,i]=\frac{|\{y_z^{pred}\mid y_z^{pred}=y_z^{test}=c,z=\{1,\ldots,Z\}\}|}{|\{y_z^{test}\mid y_z^{test}=c,z=\{1,\ldots,Z\}\}|} \tag{15}$$

See below charts and tables for A matrices for the 3 datasets:

1) Table A for SHL dataset: [Fig 4]
2) Visualized in a heat-map (lighter color represents higher probability): [Fig 5]
3) Table A for White Wine dataset: [Fig 6]
4) Visualized in a heat-map (lighter color represents higher probability): [Fig 7]
5) Table A for Avila dataset: [Fig 8]



	class0	class1	class2	class3	class4	class5	class6
feature0	0.000000	0.132353	0.382889	0.637916	0.008850	0.000000	0.0
feature1	0.000000	0.014706	0.381487	0.680144	0.106195	0.000000	0.0
feature2	0.000000	0.014706	0.420757	0.581312	0.161504	0.000000	0.0
feature3	0.000000	0.073529	0.403927	0.519317	0.157080	0.000000	0.0
feature4	0.181818	0.117647	0.420757	0.519317	0.092920	0.011236	0.0
feature5	0.090909	0.029412	0.441795	0.595687	0.064159	0.033708	0.0
feature6	0.000000	0.029412	0.483871	0.539982	0.236726	0.112360	0.0
feature7	0.000000	0.000000	0.357644	0.614555	0.064159	0.000000	0.0
feature8	0.000000	0.000000	0.301543	0.657682	0.132743	0.000000	0.0
feature9	0.000000	0.000000	0.497896	0.634322	0.236726	0.000000	0.0

Fig. 6. Matrix A with single feature accuracies for White Wine data set

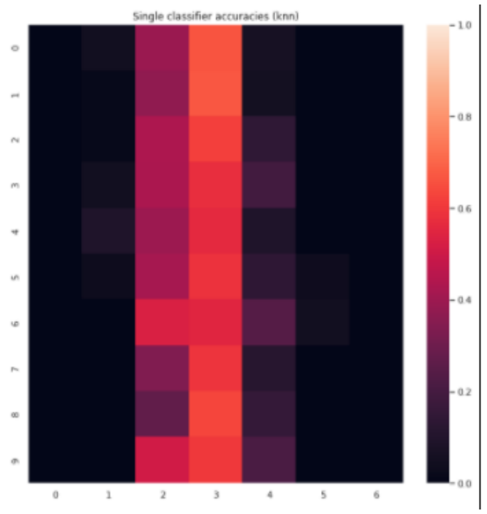


Fig. 7. Heatmap A with single feature accuracies for White Wine data set

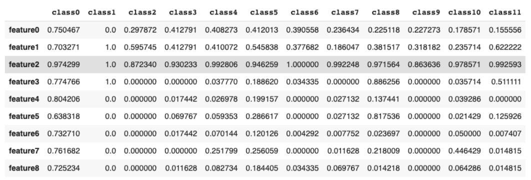
6) Visualized in a heat-map (lighter color represents higher probability): [Fig 9]

C. Essential feature selection

After obtaining the A table for probabilities of correct classification using single features, we move onto the selection of essential features for each class based on the first parameter θ. The selection returns a set Q_c for each class c that includes the indexes of the essential features that we selected for class c. [Equation 16]

$$Q_c = \{i \in I | A(i|c) > \theta\} \tag{16}$$

For class c, we determine if feature i goes into Q_c by comparing the parameter θ and the probability of correct



	class0	class1	class2	class3	class4	class5	class6	class7	class8	class9	class10	class11
feature0	0.750467	0.0	0.297872	0.412791	0.408273	0.412013	0.390558	0.236434	0.225118	0.227273	0.178571	0.155556
feature1	0.703271	1.0	0.595745	0.412791	0.410072	0.545838	0.377682	0.186047	0.381517	0.318182	0.235714	0.622222
feature2	0.974299	1.0	0.872340	0.930233	0.992806	0.946259	1.000000	0.992248	0.971564	0.863636	0.978571	0.992593
feature3	0.774766	1.0	0.000000	0.000000	0.037770	0.188620	0.034335	0.000000	0.886256	0.000000	0.035714	0.511111
feature4	0.804206	0.0	0.000000	0.017442	0.026978	0.199157	0.000000	0.027132	0.137441	0.000000	0.039286	0.000000
feature5	0.638318	0.0	0.000000	0.069767	0.059353	0.286617	0.000000	0.027132	0.817536	0.000000	0.021429	0.125926
feature6	0.732710	0.0	0.000000	0.017442	0.070144	0.120126	0.004292	0.007752	0.023697	0.000000	0.050000	0.007407
feature7	0.761682	0.0	0.000000	0.000000	0.251799	0.256059	0.000000	0.011628	0.218009	0.000000	0.446429	0.014815
feature8	0.725234	0.0	0.000000	0.011628	0.082734	0.184405	0.034335	0.069767	0.014218	0.000000	0.064286	0.014815

Fig. 8. Matrix A with single feature accuracies for Avila data set

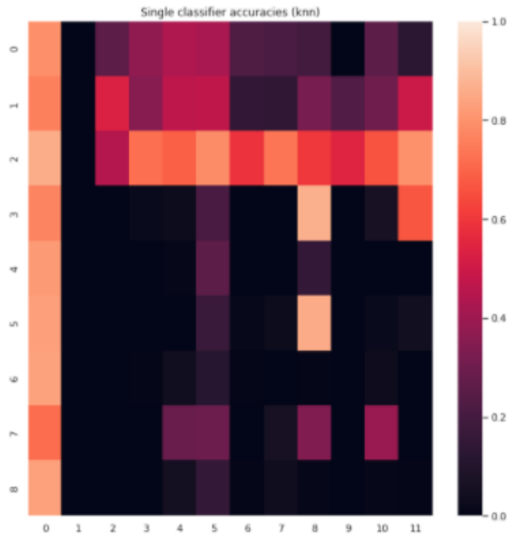


Fig. 9. Heatmap A with single feature accuracies for Avila dataset

assignment $A(i \mid c)$ in the A table. If $A(i \mid c) > \theta$, we append the feature i to the essential feature index set Q_c. We repeat this process for each classification c in the data set so that each c has a set of essential features independently from the rest. We added the possibility of choosing a different θ value for each classification, making the input of the step and array of θ values of size K.

For demonstration purposes, we set $\theta = 0.2$ for all three data sets and for all their classes. The output index set Q for each data set and for each class is shown below:

1) SHL data set
Q_0: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
Q_1: [1, 2, 6, 8, 10, 11, 12, 18]
Q_2: [1, 5, 15, 16, 17, 18]
Q_3: [0, 1, 15, 16, 17]
Q_4: [1, 2, 8, 10, 11, 13, 14]
Q_5: [0, 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 18]
Q_6: [8, 18]
Q_7: [2, 8, 18]
Q_8: [12]

2) White wine data set
Q_0: [] (no feature selected)
Q_1: [] (no feature selected)
Q_2: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Q_3: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Q_4: [6, 9]
Q_5: [] (no feature selected)
Q_6: [] (no feature selected)

3) Avila data set
Q_0: [0, 1, 2, 3, 4, 5, 6, 7, 8]
Q_1: [1, 2, 3]
Q_2: [0, 1, 2]

Q_3: [0, 1, 2]
Q_4: [0, 1, 2, 7]
Q_5: [0, 1, 2, 5, 7]
Q_6: [0, 1, 2]
Q_7: [0, 2]
Q_8: [0, 1, 2, 3, 5, 7]
Q_9: [0, 1, 2]
Q_10: [1, 2, 7]
Q_11: [1, 2, 3]

With an essential feature index set Q available for each classification, we transform the training set measurements by looking at their true class c and selecting only the essential features indexed in Q_c.

The new training set is then: [Equation 17]

$$\{(\pi_{Qc}(I,x1), \pi_{Qc}(I,x2), ..., \pi_{Qc}(I,x1)), (c_1, c_2, ..., c_z)\} \tag{17}$$

D. Covariance matrix and decomposition

Having the new training set X_train_Q, we can calculate the covariance matrix Σ_c for each class of the new training set X_train_{Qc} [Equation 6], as well as the the inverse covariance matrix Σ_c^{-1}.

For each class c, We store the new training set X_train_{Qc} with its corresponding covariance matrix Σ_c and its inverse covariance matrix Σ_c^{-1}. Then we perform eigenvalue - eigenvector decomposition on each covariance matrix Σ_c to obtain its eigenvalues λ and eigenvectors v.

$$\Sigma_c = V_c \Lambda_c \, V_c^{-1} \tag{18}$$

For example, the covariance matrix of class 4 of the white wine dataset is:

$$\begin{bmatrix} 1.08969026e-04 & -1.00791473e-05 \\ -1.00791473e-05 & 9.36144279e-07 \end{bmatrix}$$

The inverse covariance matrix is:

$$\begin{bmatrix} 2.22087425e+062.39113982e+07 \\ 2.39113982e+072.58514109e+08 \end{bmatrix}$$

The eigenvalues are:
$[1.09901335e-04, 3.83544581e-09]$

The eigenvectors are:
$[0.99574925, 0.09210559]$ $[-0.09210559, 0.99574925]$

E. Principal Components

For each class c of the new training set, we order its eigenvalues and eigenvectors by sorting the eigenvalues from the largest to the smallest. We then select the number of principal components E_c for class c based on how much variance we want to keep in the data, which is decided by a parameter f. We choose the smallest integer E_c to satisfy the below condition:

$$f < \frac{\sum_{n=1}^{E_c} \lambda_n}{\sum_{n=1}^{N} \lambda_n} \tag{19}$$

Take the white wine dataset as an example, the number of principal components selected for each class when $\theta = 0.2, f = 0.95$ for all classes is:

1) Class 0: 0
2) Class 1: 0
3) Class 2: 3
4) Class 3: 3
5) Class 4: 1
6) Class 5: 0
7) Class 6: 0

After E_c is selected for class c, we define the subspace S_c for class c to be the span of the first E_c eigenvectors of covariance matrix Σ_c.

$$S_c = V_c[: E_c] \tag{20}$$

After obtaining S_c, We project each measurement tuple from the new training set of class X_{Qc} onto S_c. The projected new training set is then called Y_c

$$Y_c = S_c^t X_{Qc} \tag{21}$$

Again, we calculate the covariance matrix of the projected new training set for class c Y_c to be Σ_{Y_c} as well as its inverse $\Sigma_{Y_c}^{-1}$. We save the inverse covariance matrix $\Sigma_{Y_c}^{-1}$ to be used for calculating Mahalanobis distance in the next step.

F. Mahalanobis Distances

For each class c, using the new projected training set Y_c, we calculate the Mahalanobis distance for each $\vec{x}_i$ in Y_c to the mean of class c $\vec{\mu_c}$. We sort these distances from the smallest to the largest and store them in $d_{Y_c} = \{d_{Y_c}[1], \ldots, d_{Y_c}[Z_c]\}$ [14].

G. Subspace Classifier

Up to this point, for each class of the training data, we have generated:

- a set of essential features indexes Q_c
- a new training set for class c X_{Q_c} with only essentials features indicated in Q_c
- subspace for class c S_c
- projected new training set for class c Y_c
- mean vector of the projected new training set $\vec{\mu_c}$
- inverse covariance matrix of the projected new training set $\Sigma_{Y_c}^{-1}$
- a set of Mahalanobis distances d_{Y_c} for each measurement y in Y_c

Moving on to the test set, we want to assign a class for each $x_z \in X_test$. To do so, we need to calculate the Mahalanobis distance d_{x_z} of each vector $\vec{x}$ from the test set to the mean of each class of the training set and compare them with the training set distances d_{Y_c}.

Data Set	Accuracy
SHL	0.68
Avila Bible	0.03
White Wine	0.33

TABLE II
CLASSIFIER'S RESULTS WITHOUT FILTERING ESSENTIAL FEATURES ($\theta = -1.0$), SELECTING ALL THE PRINCIPAL COMPONENTS ($f = 1.0$) AND WITHOUT DISCARDING ANY p-VALUE ($p_0 = 1.0$)

Before calculating the distance d_{x_z}, we transform each x_z from the test set by filtering only the essential features indexed in Q_c and projecting the new x_z onto the subspace S_c, resulting in a new measurement vector $\vec{x}_z{}'$. Then, we calculate the Mahalanobis distance $d_{\vec{x}_i'}$ of $\vec{x}_i{}'$ to the mean of class c of the train set $\vec{\mu_c}$ using the inverse covariance matrix of the subspace $\Sigma_{Y_c}^{-1}$. We compare this distance with the set of distances previously stored in d_{Y_c}, and generate a p_{value} [22] based on the below functions.

$$p_{value}(\vec{x}) = \begin{cases} \frac{1}{Z_c} & d_{Y_c}(\vec{x}) \leqslant d_{Y_c}[1] \\ \frac{j-0.5}{Z_c} & d_{Y_c}[i-1] < d_{Y_c}(\vec{x}) \leqslant d_{Y_c}[i], i \in [2, Z_c] \\ 1 & d_{Y_c}[Z_c] < d_{Y_c}(\vec{x}) \end{cases} \tag{22}$$

The p_{value} calculated for each class is then stored in an array $p_{values} = \{p_{value}(c) \mid c \in C\}$.

Defining a parameter p_0, we discard the classes where its p_{value} is larger than p_0, then assign x_i to the class with the smallest p_{value}. [23]. Note that when setting p_0, our implementation allows a different p_0 for each class.

$$predict(\vec{x}) = \begin{cases} \textit{reserved class} & p_0 < \min_{c \in C} p_{value}(c) \\ \operatorname{argmin}_{c \in C} p_{value}(c) & \end{cases} \tag{23}$$

VI. EXPERIMENTAL RESULTS

We trained and tested the subspace classifier on the three datasets listed above and conducted experimentation with the parameters and scaling the datasets to achieve optimal accuracy.

A. Tuning parameters

We started with setting all three parameters to their baseline values where no filtering was performed, the classifier works like it was constructed without parameters. Classification accuracy from the three datasets were reported in [Table II].

Experiment with θ while fixing f and p_0 (each parameter is set to the same for all classes): [Fig 10]

Experiment with f while fixing θ and p_0 (each parameter is set to the same for all classes): [Fig 11]

Experiment with p_0 while fixing θ and f (each parameter is set to the same for all classes): [Fig 12]

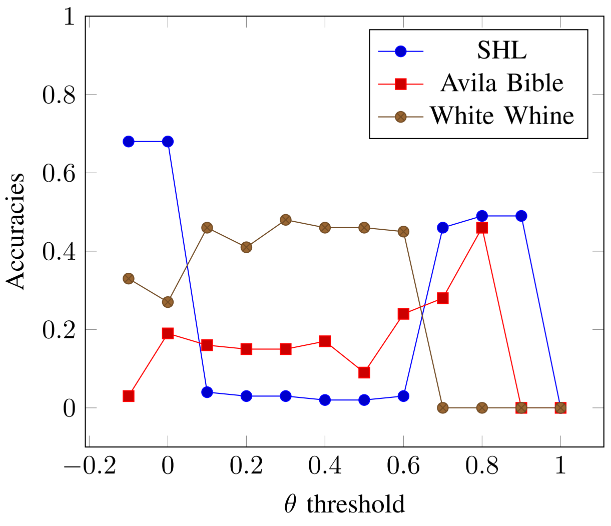


Fig. 10. Experiment with θ while fixing f and p_0

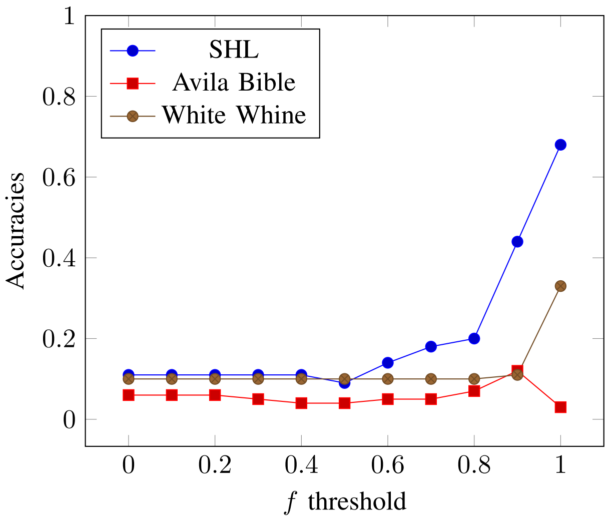


Fig. 11. Experiment with f while fixing θ and p_0

B. Data Scaling

Besides tweaking the parameters, we also tried using different data scalers from Python Scikit-learn packages to scale the original datasets. Here we show the results for experiments with the Avila Bible dataset [Figures 13, 14, and 15]. The results for the other datasets are in Appendix B.

Confusion Matrix for each dataset using that parameter combination that yields the highest accuracy [Figures 16, , and].

C. Conclusions

The three major procedures of constructing this classifier are selection of essential features, projection onto subspace, and classification with Mahalanobis distance. Each procedure introduces a different parameter: essential feature selection with parameter θ, subspace selection with parameter f, classification with parameter p_0.

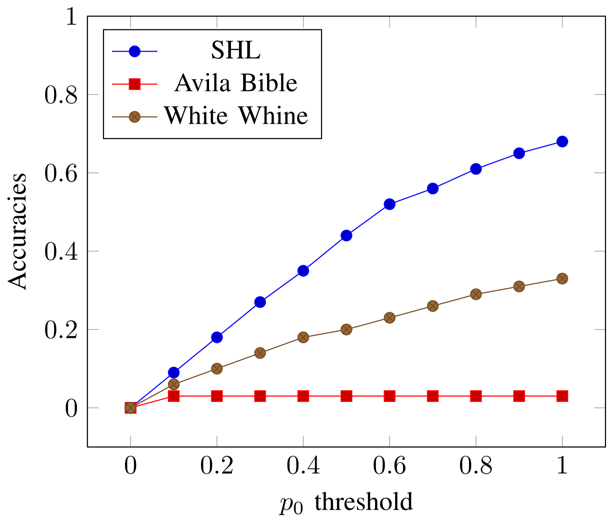


Fig. 12. Experiment with p_0 while fixing θ and f

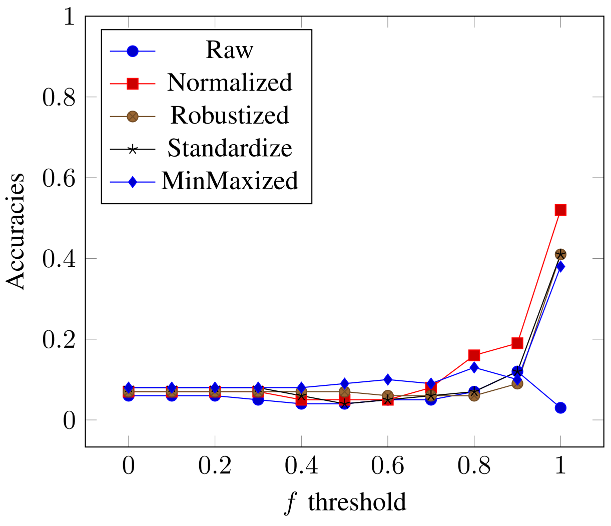


Fig. 14. Scaled Avila Bible dataset with f changed

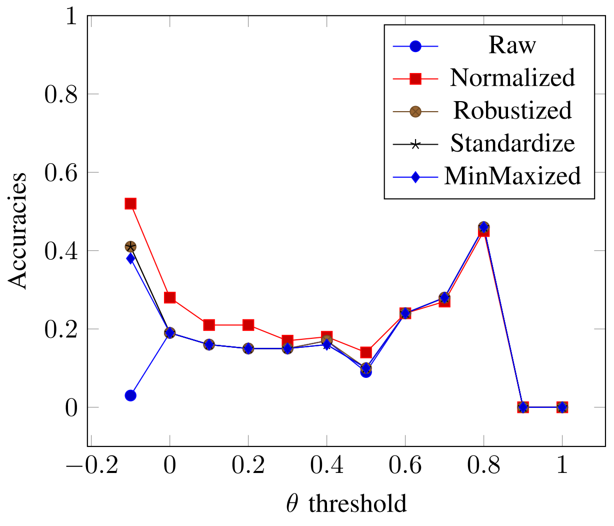


Fig. 13. Scaled Avila Bible dataset with theta changes

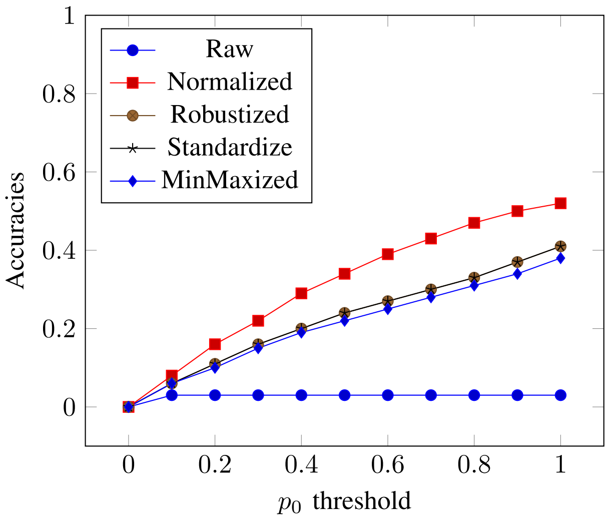


Fig. 15. Scaled Avila Bible dataset with p changes

During essential feature selection, we tried implementing different single feature classifiers, including KNN, KMeans, Decision Trees, Logistic Regression, as well as the Mahalanobis distance classifier itself. For simplicity, we only discussed using KNN in this report. We initially thought selecting features with higher values from the A table would help produce higher accuracy for classification. However, during experimentation, we discovered that using features with higher values from the A table does not necessarily yield higher accuracy, as almost always the datasets produce higher accuracy when θ is set to an extremely low value, selecting all the features for all the classes.

During principal component analysis, we discovered that our datasets always produce better accuracy when f threshold is set high or close to 1. We also came upon an interesting discovery that the class with the smallest number of principal components seem to get assigned more times than the classes with a larger number of principal components. We think the problem could be that the number of measurement tuples for each class is not balanced in our datasets. We researched and experimented with multiple scaling techniques like StandardScaling, Normalizing, MinMaxScaling and RobustScaling. We

$$\begin{bmatrix} 0 & 0 & 6 & 5 & 0 & 0 & 0 \\ 0 & 0 & 49 & 8 & 11 & 0 & 0 \\ 0 & 0 & 483 & 147 & 83 & 0 & 0 \\ 0 & 0 & 425 & 440 & 248 & 0 & 0 \\ 0 & 0 & 57 & 186 & 209 & 0 & 0 \\ 0 & 0 & 7 & 40 & 42 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 & 0 \end{bmatrix} \tag{24}$$

Fig. 16. Confusion matrix for White Wine dataset classification

$$\begin{bmatrix}
1359 & 0 & 19 & 246 & 169 & 280 & 35 & 5 & 7 & 0 & 8 & 12 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
15 & 0 & 5 & 5 & 7 & 14 & 0 & 0 & 1 & 0 & 0 & 0 \\
56 & 0 & 9 & 63 & 23 & 14 & 1 & 2 & 0 & 0 & 3 & 1 \\
164 & 0 & 21 & 51 & 214 & 70 & 5 & 15 & 1 & 1 & 10 & 4 \\
351 & 0 & 36 & 79 & 53 & 402 & 21 & 0 & 3 & 0 & 1 & 3 \\
71 & 0 & 6 & 6 & 14 & 36 & 100 & 0 & 0 & 0 & 0 & 0 \\
52 & 0 & 23 & 9 & 71 & 37 & 11 & 52 & 1 & 0 & 0 & 2 \\
25 & 0 & 3 & 10 & 11 & 20 & 4 & 0 & 336 & 0 & 4 & 9 \\
5 & 0 & 0 & 4 & 3 & 0 & 0 & 0 & 0 & 10 & 0 & 0 \\
57 & 0 & 3 & 10 & 83 & 4 & 0 & 0 & 14 & 0 & 89 & 20 \\
16 & 0 & 0 & 0 & 7 & 3 & 0 & 0 & 16 & 0 & 9 & 84
\end{bmatrix} \tag{25}$$

Fig. 17. Confusion matrix for Avila Bible dataset classification

$$\begin{bmatrix}
17139 & 20 & 3601 & 152 & 1988 & 300 & 621 & 126 & 738 \\
243 & 1249 & 1356 & 57 & 95 & 0 & 430 & 186 & 63 \\
90 & 0 & 3103 & 96 & 24 & 0 & 8 & 0 & 12 \\
0 & 0 & 15 & 550 & 0 & 0 & 1 & 0 & 0 \\
69 & 2 & 54 & 39 & 1921 & 0 & 1 & 0 & 8 \\
927 & 0 & 129 & 1 & 6 & 2282 & 501 & 81 & 166 \\
804 & 0 & 590 & 0 & 35 & 91 & 2473 & 32 & 63 \\
331 & 0 & 34 & 1 & 1 & 5 & 132 & 2948 & 252 \\
415 & 0 & 392 & 2 & 1 & 17 & 185 & 214 & 2532
\end{bmatrix} \tag{26}$$

Fig. 18. Confusion matrix for SHL dataset classification

also used the scalers in different parts of the process: after loading dataset, before projecting, after projecting. However, neither of these experiments saw a significant improvement.

During the final classification process, we observed that a higher value of p_0 usually generates a higher accuracy. We think this is because a higher p_0 produces less "reserved class", and a large number of "reserved class" could negatively impact the classification accuracy in theory.

We detected that datasets with highly correlated features cause singular covariance matrices, and we were unable to continue with the chance of inverting them. To fix this issue, just after pulling the dataset, we remove the columns with a correlation coefficient higher than .95.

Note that the conclusions provided here are confined to the three datasets provided in this report and not applicable to any other dataset. We also tested the classifier on smaller datasets and synthetically generated datasets and were able to obtain accuracy over .9 in several cases. We discovered that the classifier performs especially well on synthetically generated dataset when one class's distribution is far away from another.

References

[1] Robert M. Haralick, *Final Project 2020* `http://haralick.org/ML/final_project_2020.pdf`

[2] C. De Stefano, M. Maniaci, F. Fontanella, A. Scotto di Freca, *Reliable writer identification in medieval manuscripts through page layout features: The "Avila" Bible case* `Engineering Applications of Artificial Intelligence, Volume 72, 2018, pp. 99-110.`

[3] H. Gjoreski, M. Ciliberto, L. Wang, F. J. O. Morales, S. Mekki, S. Valentin, D. Roggen. *The University of Sussex-Huawei Locomotion and Transportation Dataset for Multimodal Analytics with Mobile Devices.* `IEEE Access 6 (2018): 42592-42604.`

[4] L. Wang, H. Gjoreski, M. Ciliberto, S. Mekki, S. Valentin, and D. Roggen *Enabling reproducible research in sensor-based transportation mode recognition with the Sussex-Huawei dataset* `IEEE Access 7 (2019): 10870-10891.`

[5] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. *Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems* `Elsevier, 47(4):547-553, 2009.`

Appendix A
Code

The code and experiments are available through the following link:
`https://colab.research.google.com/drive/18J4tp375uM305tfL6MYicGOQtDXroawu?usp=sharing`

Appendix B
Experiments

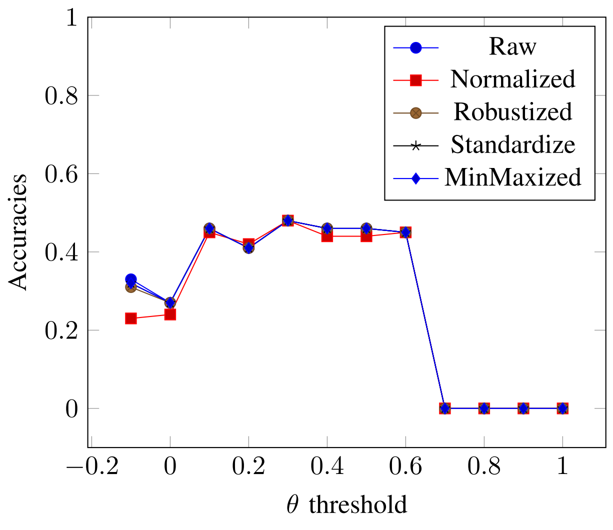


Fig. 19. Scaled White Wine dataset with theta changes

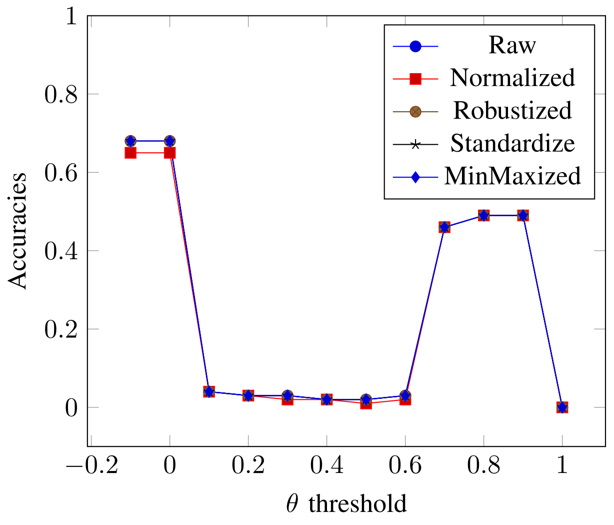


Fig. 20. Scaled SHL dataset with theta changes

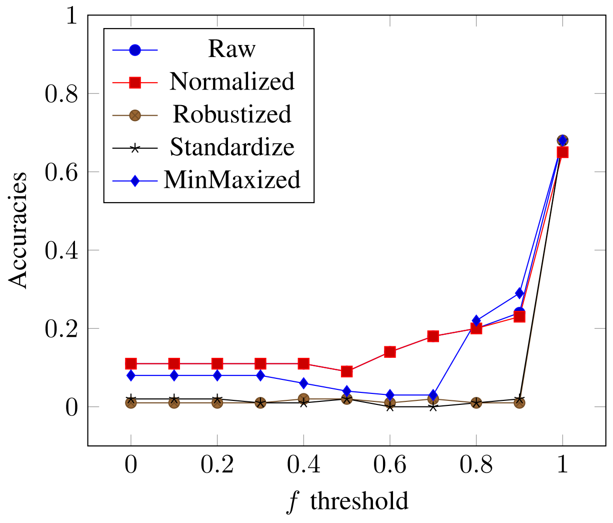


Fig. 22. Scaled SHL dataset with f changes

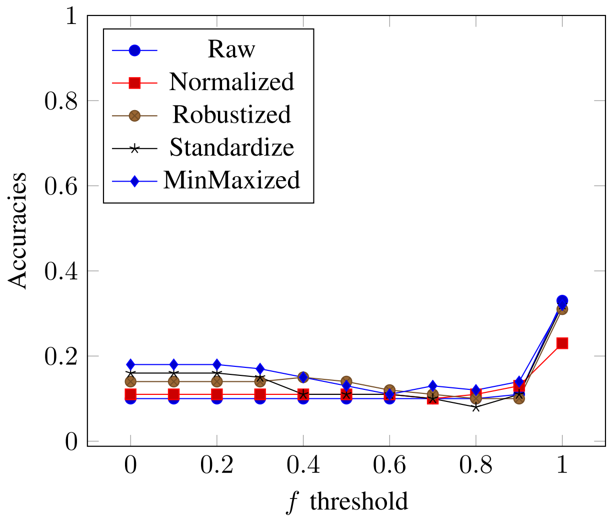


Fig. 21. Scaled White Wine dataset with f changes

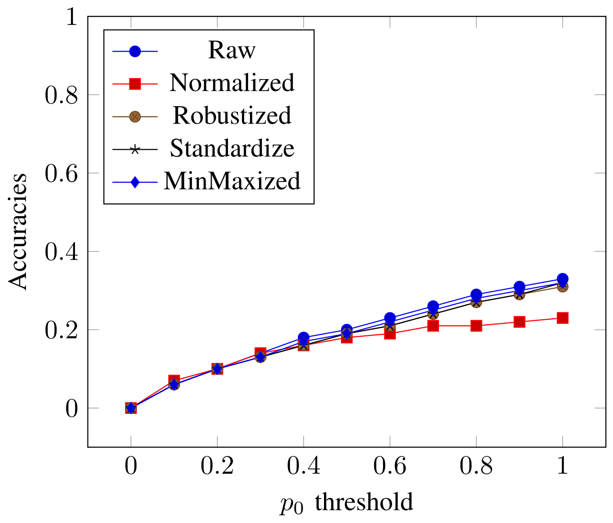


Fig. 23. Scaled White Wine dataset with p changes

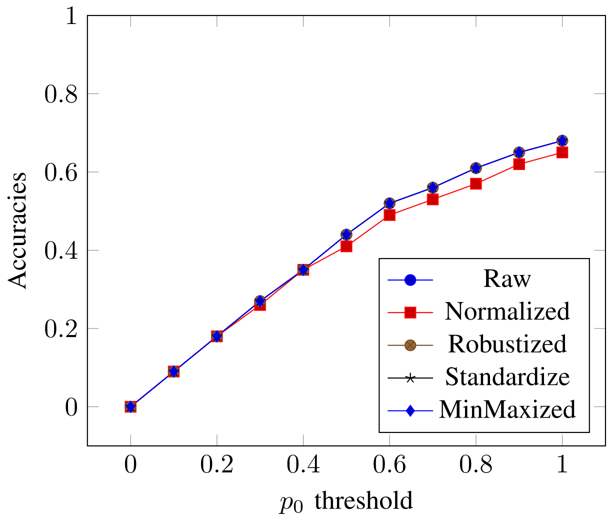


Fig. 24. Scaled SHL dataset with p changes