Classification and Regression Techniques on Bag of Words Model

Abdullah Danyal Saeed (601250), Aisha Saeed (601263)
School of Science, Aalto University

1. Abstract

This report includes the learning background, the methods applied and consequent tasks of classification (such as Multivariate discrete distribution, Multivariate Gaussian Distribution and Logistic Regression) and regression methods (such as Multivariate regression and K-D trees) for prediction analysis. This report contains the description and implementation of the methods mentioned above, starting from analysing the given data, reporting the results and calculating the errors occurred. Thus, it is concluded with descriptions of the gathered results, some learning outcomes and the possible improvement scenarios.

2. Introduction

Yelp.com, a social networking website, assists the users to submit the reviews from 1 to 5 stars, for the local businesses such as schools and restaurants for their different products and services. It helps people to find out the great local businesses. 3 stars or better rating was received by 78% of the businesses listed on this website. If the review is useful to user, he can give that review a "thumbs-up". Thus, the votes to the review tell how much it was useful.

In this project, our aim is to classify if the user has given more than 3 star rating and to determine how many people find it useful. We are provided with 6000 Yelp user reviews and 50 features for the 50 words that were selected by using bag-of-words model (as they appeared most frequently). By using machine learning techniques, the task we perform is to train our method by using 5000 samples (known outputs) and then use remaining 1000 reviews in order to validate the chosen method. We also report the prediction performance by using appropriate methods of machine learning. We examined the errors that occurred and explained them thoroughly. Thus, this project involves learning of the machine learning techniques for data analysis.

3. Methods

In this project we use Multivariate discrete distribution (modify data for Bernoulli approach), Multivariate Gaussian Distribution and Logistic Regression for classification and compare their results. For Regression, we used Multivariate regression for different degree of polynomials and also k-d trees. In machine learning, the pre-processing task has very much importance which involves exploratory data analysis, feature selection and feature extraction. Pre-processing helps to improve the efficiency of algorithm and also gives better result sometimes. So, before performing the actual methods of classification and regression, we did pre-processing of data such as Andrew plots, scatterplot, component neighbourhood analysis and principal component analysis. We also did K-fold cross and LOO as a cross fold validation techniques for model assessment. After applying the techniques, we also estimate errors and accuracies and use confusion matrix to show the performance of the system.

3.1 Pre-processing of Data

3.1.1 Feature Selection

3.1.1.1 Neighbourhood Component Analysis

Neighbourhood component analysis (NCA) is a non-parametric method used for selecting features by maximizing prediction accuracy of regression and classification algorithms.

3.1.1.2 Forward Selection

In forward selection we add each variable one at a time and check the P-value, if it is below some pre-set level then the feature is selected. Initially the set is empty and then add the features. In backward selection, we remove the features and check P-value.

3.1.2 Feature Extraction

3.1.2.1 Principal Component Analysis

It is the most common technique for dimensionality reduction, it uses orthogonal transformation to convert a set of observations of correlated variables into linearly uncorrelated variables called principal components.

3.2 Classification

3.2.1 Multivariate Discrete Distributions

3.2.1.1 Bernoulli Approach

Let us assume that features X_j are binary(Bernoulli), where

$$P_{ij} = P(x_j = 1 | C_i)$$

Now draw another assumption that features are independent binary variables, then we have

$$P(x|C_i) = \Omega_{j=1-d} (P_{ij}^{xj}(1-P_{ij})^{(1-xj)})$$

This is just a naïve bayes classifier where $P(x|C_i)$ are Bernoulli. The discriminant function would be

$$g_i(x) = \log p(x|C_i) + \log P(C_i)$$

$$=\sum_j [x_j \log p_{ij} +(1-x_j) \log (1-p_{ij})] + \log P(C_i)$$

The estimator for $p_{ij} = (\sum_t x_j^t r_i^t)/(\sum_t r_i^t)$

To use this approach on yelp reviews data set, we modify bag of words model in such a way that if a particular word has frequency greater than zero, we set it to equal to one otherwise it is remain zero. In this way each feature is Bernoulli variable.

3.2.2 Logistic Regression

The goal of logistic regression is to find the best fitting model to describe the relationship between response and a set of features. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a *logit transformation* of the probability of presence of the characteristic of interest.

$$logit(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_kX_k$$

Where p is the probability of presence of the characteristic of interest. The logit transformation is defined as the logged odds:

$$odds = \frac{p}{1-p} = \frac{probability\ of\ presence\ of\ characteristic}{probability\ of\ absence\ of\ characteristic}$$

$$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

3.2.3 Multivariate Gaussian Distribution

In parametric method, we always assume that data belongs to some distribution. In this case we assume that data comes from Gaussian distribution. Multivariate Gaussian distribution is a generalization of univariate normal distribution to higher dimension. The multivariate normal distribution is used to define correlated random variables. The multivariate normal distribution for a k dimensional vector x would be

$$\begin{aligned} f_{\mathbf{x}}(x_1,\ldots,x_k) &= \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} \\ &= \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{|2\pi\boldsymbol{\Sigma}|}}, \end{aligned}$$

3.3 Regression

3.3.1 Multivariate Regression using polynomials of various degrees

It is a technique which estimates a single regression model having more than one feature. We use regression to predict responses on the basis of predictors (more than one predictor) to define how a response depends on predictor.

Regression deals with the polynomial (linear or nonlinear) equations, and the goal is to find the co-efficient which when applied on inputs give the good approximate of response. Let suppose X is the input vector and Y is response:

$$X_1, X_2, \ldots, X_k \Rightarrow Y$$

In the multiples linear regression model, response has normal distribution with mean:

$$Yfit = b_0 + b_1X_1 + \ldots + b\rho X\rho$$

The co-efficient $b_0, b_1 \ldots b_n$ computed in such a way to minimize the difference between actual value of response and predicted response i-e Yfit. This problem of regression is handled by least square error technique, the goal is to minimize the empirical error E (W|D).

$$\mathbf{w}_{opt} = \underset{\mathbf{w}}{\operatorname{argmin}}\, E(\mathbf{w}|\mathcal{D}) = \sum_{t=1}^{N}(r^t - \mathbf{w}^T\mathbf{x}^t)^2$$

We choose those co-efficient which has less value of W (Co-efficient). If we take the derivative of the error and set it equal to zero then we will least square error solution of multivariate regression which will be like this:

$$\mathbf{X}^T\mathbf{X}\mathbf{w}_{\text{opt}} = \mathbf{X}^T\mathbf{r}$$

3.3.2 K-d Trees

KD-Trees or Decision trees is a non-parametric method of regression and classification. As with any non-parametric method the main concept of KD-Trees is that similar data yields similar output. We divide the data into regions based on different decision boundaries. The decision of dividing a region into sub regions is taken based on the value of gain achieved. We keep dividing the regions in to sub regions until we find that the gain we are getting after the division is not that much or we reach a threshold value of max number of leaf nodes in a region.

3.4 Model Assessment

It is one of the most important factors in the accuracy of machine learning algorithm. Our main task in machine learning is to minimize generalization error which is the error on future observations. In training phase we have a number of hypothesis (models) and our goal is to select the model which gives good performance on testing datasets. There are many techniques of model assessment such as validation, cross validation, 0-1 loss and prediction accuracy, hold-out method, bias and variance. We used cross validation for model assessment.

3.4.1 Cross Validation

Cross Validation (CV) is a method used for validating model to check that how training results will generalize to an independent data sets. This technique is used when one wants to know how accurately a predictive model will perform in practice. The whole training set is split into two partitions namely validation set and training set. The model is selected according to the error on validation set. There are different cross validation techniques which we used in the project for classification and regression tasks.

3.4.1.1 The hold out Method

The data set split into two sets, called the training set and the testing set. The function approximator fits a function using the training set only. Then the function approximator is asked to predict the output values for the data in the testing set. The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model.

3.4.1.2 K-Fold Cross Validation

It is one way to improve over the holdout method. The data set is divided into *k* subsets, and the holdout method is repeated *k* times. Each time, one of the *k* subsets is used as the test set and the other *k-1* subsets are put together to form a training set. Then the average error across all *k* trials is computed.

3.4.1.3 Leave One Out CV

It is K-fold cross validation taken to its logical extreme, with K equal to N, the number of data points in the set. That means that N separate times, the function approximator is trained on all the data except for one point and a prediction is made for that point. As before the average error is computed and used to evaluate the model.

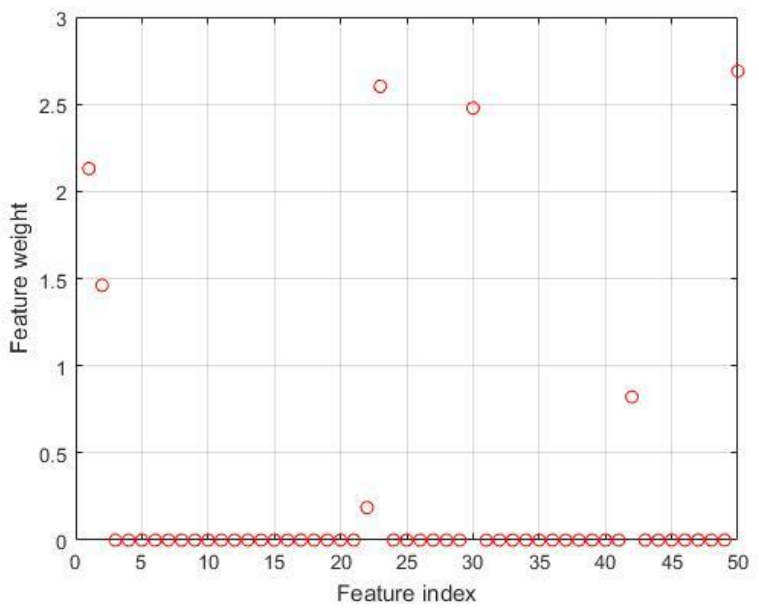
4. Experiments

4.1 Pre-Processing

4.1.1 Feature Selection

4.1.1.1 Neighbourhood Component Analysis (NCA)

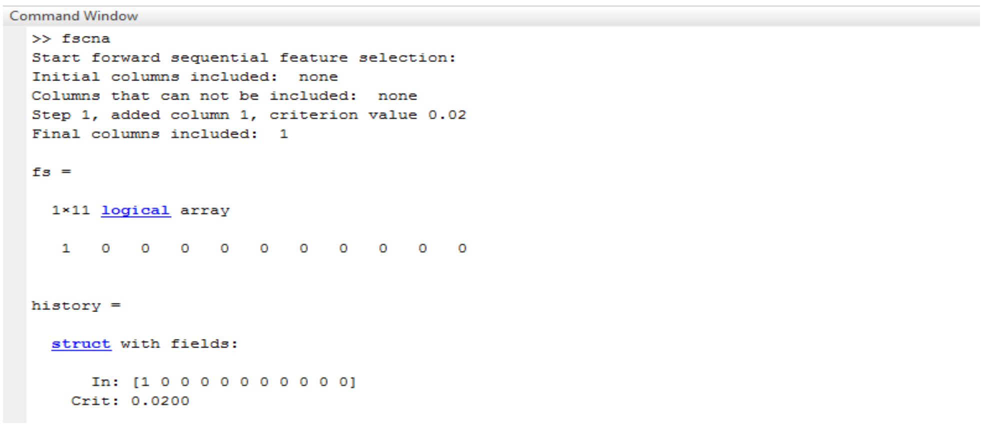
After applying NCA on the given data set, the results are as follow. We see that in the figure (4.1.1.1) given below, there is a plot against Feature weight and Feature index. Those features which have higher weight indicates that they should be selected for classification and regression tasks. As the plot depicts, feature 1,2,21,22,30,42 and 50 have higher weights than other feature. It means that they have much importance in determining the given class as compare to other features.
We did classification and regression in both ways that is with whole data and with only those features. We will see in later part that there is not a big difference in results if you select only these features or whole data depicting that we gain same results with less computations hence we can say that it improves our machine learning algorithm efficiency.



Fig(4.1.1.1)

4.1.1.2 Forward Search

We also select features by forward search and check against neighbourhood component analysis. As mentioned before, forward search always starts with an empty array and add features according to the criterion value. We used k-cross validation with k=10. We check with different criterion values to select features.



```
>> fscna
Start forward sequential feature selection:
Initial columns included:  none
Columns that can not be included:  none
Step 1, added column 1, criterion value 0.02
Final columns included:  1

fs =

  1×11 logical array

   1   0   0   0   0   0   0   0   0   0   0


history =

  struct with fields:

      In: [1 0 0 0 0 0 0 0 0 0 0]
    Crit: 0.0200
```

Fig(4.1.1.2) with criterion value 0.02, only one feature added

We see that, with criterion 0.02, it selects only 1st feature. We also check it with other criterions but results are not very significant.

4.1.2 Feature Extraction

4.1.2.1 PCA

To improve the efficiency of our machine learning algorithm, we also test our methods of classifications and regression with less number of dimensions. Principal component analysis plays a major role in feature extraction, it give us lee number of features as compared to original vector.

The following figures shows the scatter plot of PCA with different features, hence give us the necessary information about the principal components and their impact.

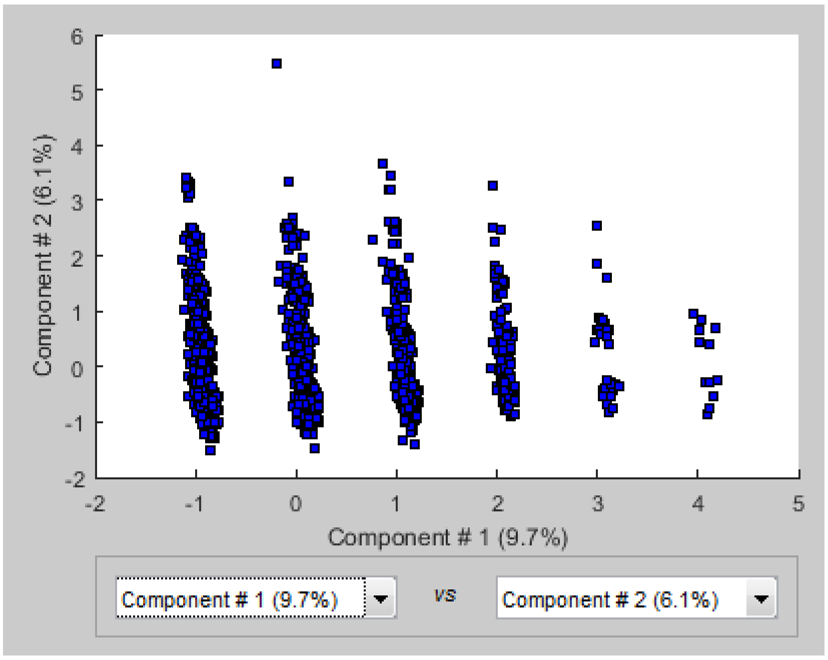


Fig (4.1.2.1a) PCA plot of component#1 against component #2, we see in the plot that component #1 has 9.7% while component#2 has 6.1%. As we mentioned before that 1st principal component has more value than second and so on.

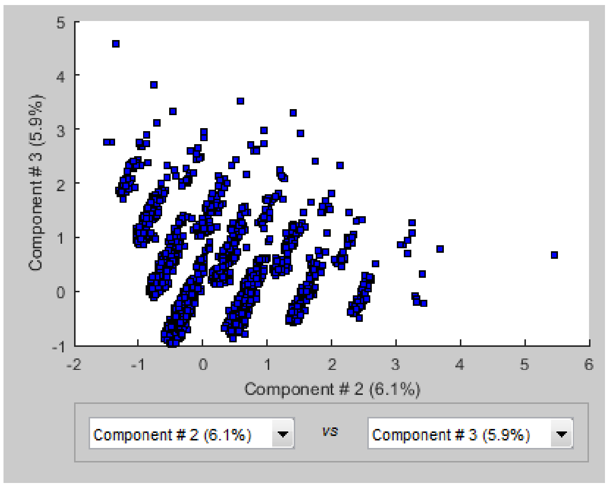


Fig (4.1.2.1b) PCA plot of component#2(6.1%) against component #3(5.9%)

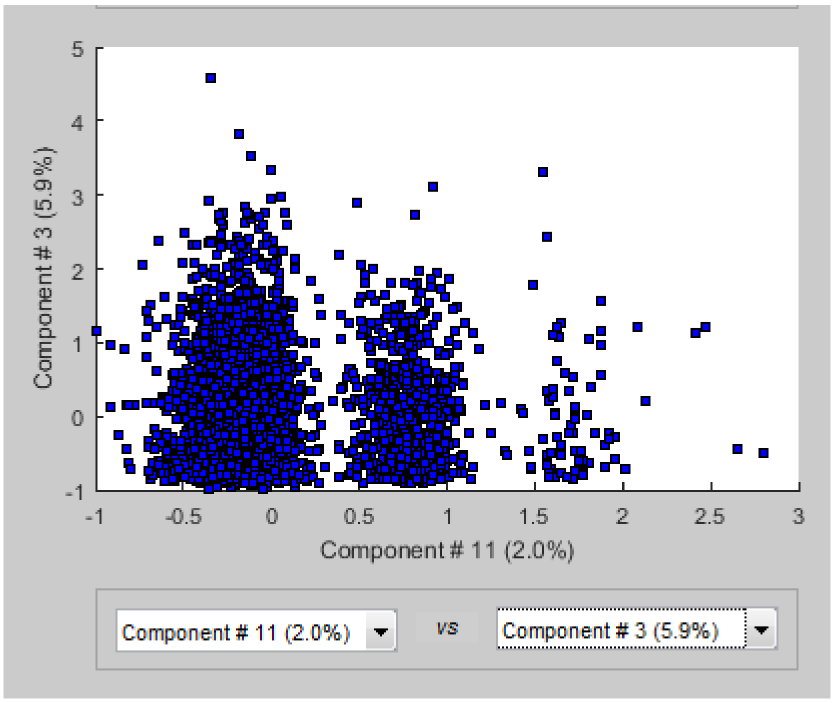


Fig (4.1.2.1c) PCA plot of component#11(2.0%) against component #3(5.9%)

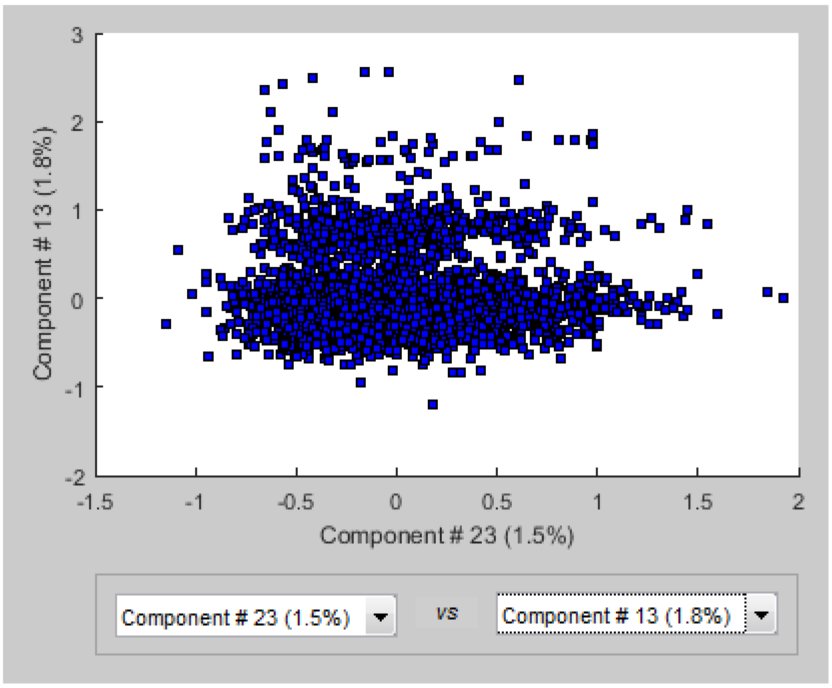


Fig (4.1.2.1d) PCA plot of component#11(2.0%) against component #3(5.9%)

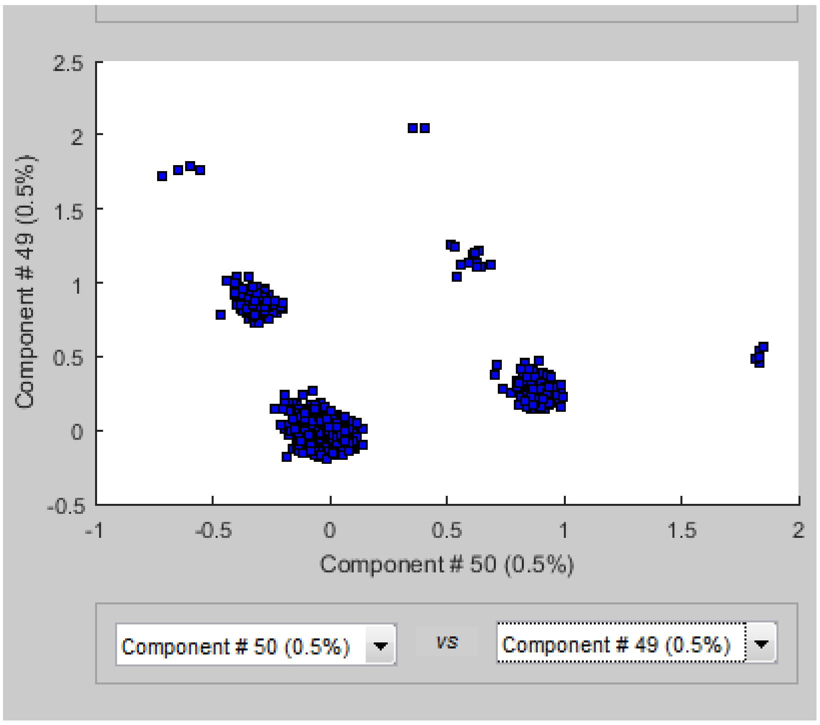


Fig (4.1.2.1e) PCA plot of component#11(2.0%) against component #3(5.9%)

4.2 Classification

4.2.1 Discrete Features

We used Bernoulli approach as described above. We also mentioned that we pre-process the dataset (only in this approach not for all other techniques) in such a way that if some word has frequency greater than zero we set them to one otherwise it remains zero. So, this kind of bag of words model tell us that whether in a given review a word has been occur or not. After this pre-processing step, we separate the data on the basis of classes, 0 and 1. Then we estimate Pij for both classes. After this we calculate the discriminant function against test data and check the accuracy of the system.

- Pij, calculated by sum the column and divide it by total total number of rows.
- Hence, there are 100 pij values for both classes as data features are 50.
- After estimating pij values, apply the discriminant function on 1000 data items from test set.
- Discriminant function is:

$$g_i(x) = \log p(x|C_i) + \log P(C_i)$$
$$\sum_j [x_j \log p_{ij} +(1-x_j) \log (1-p_{ij})] + \log P(C_i)$$

4.2.2 Logistic Regression

We find co-efficient for logistic regression by logit function in MATLAB. After it we apply these co-efficient on test data (by logit function). We also did model assessment and check it with different values of k in k-cross validation. We also applied leave one out cross validation technique on it. The figures given below shows the effect of value of k in k-cross validation on accuracy:

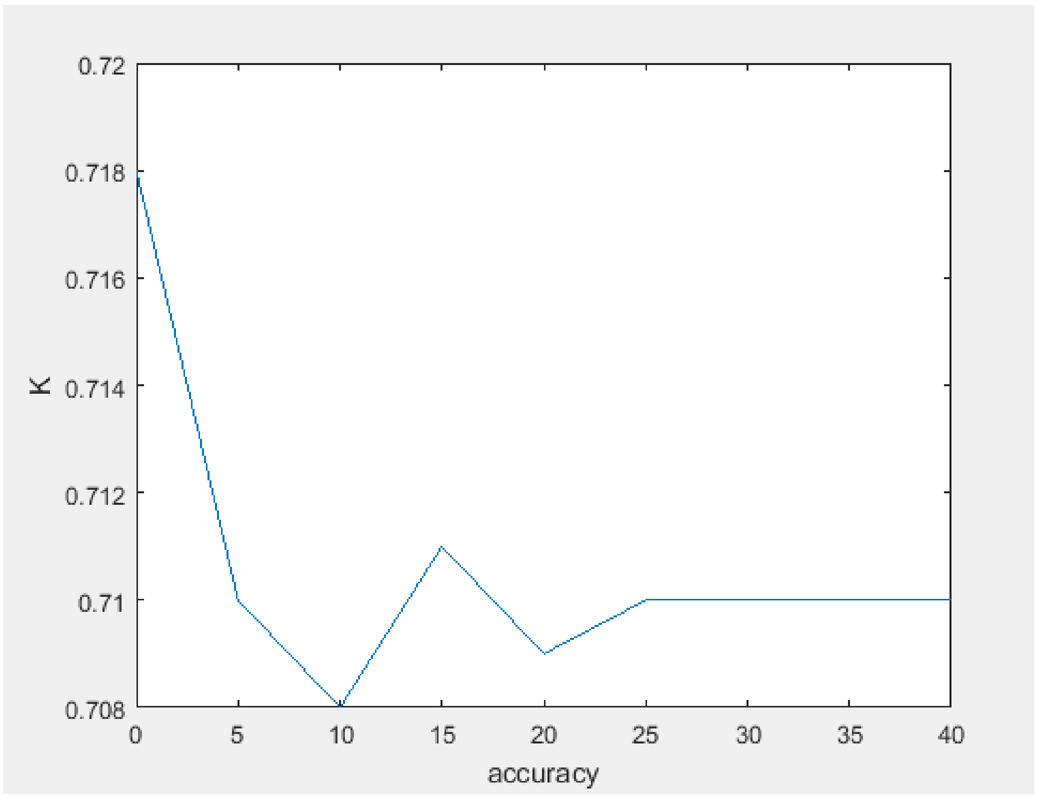


Fig (4.2.2a) K against accuracy

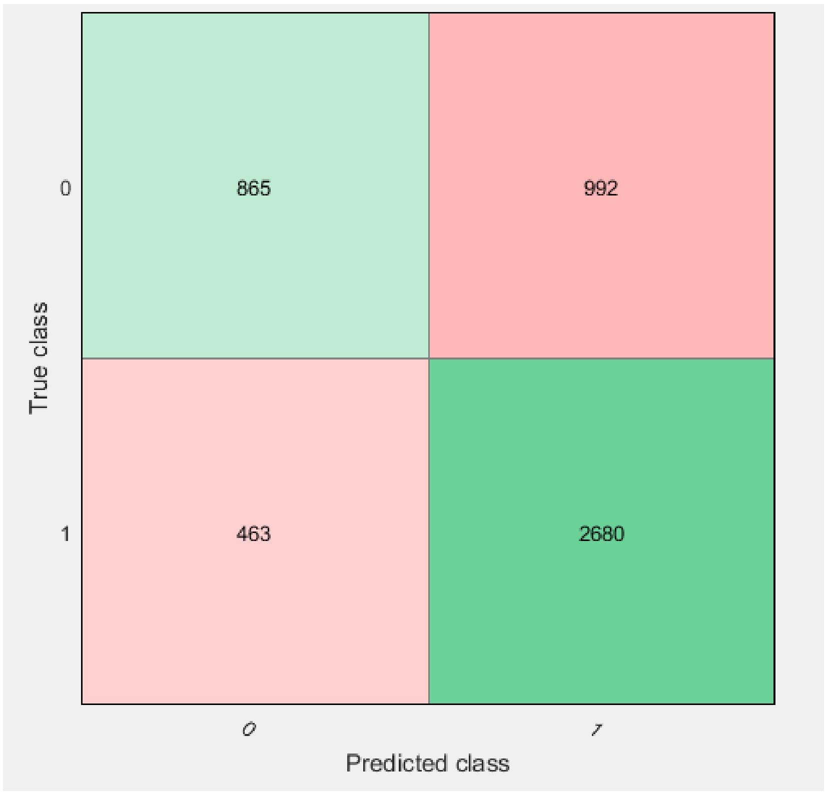


Fig (4.2.2b) Confusion Matrix with k-fold cross validation, K=10

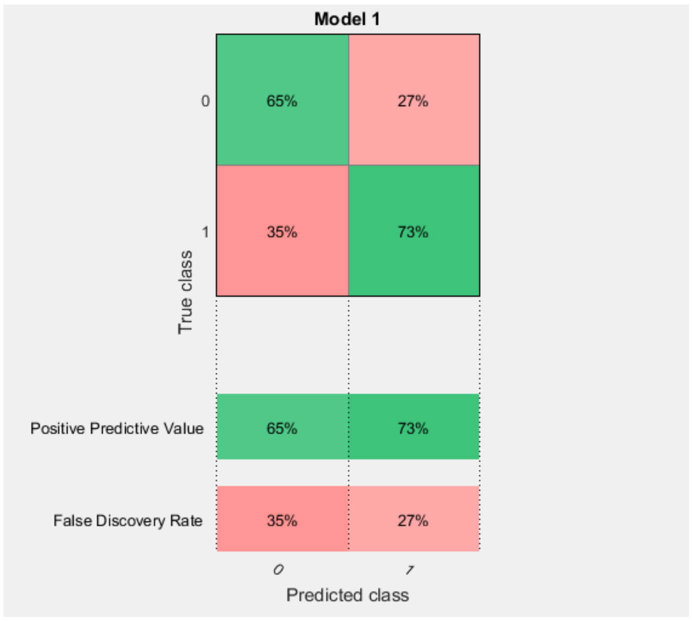
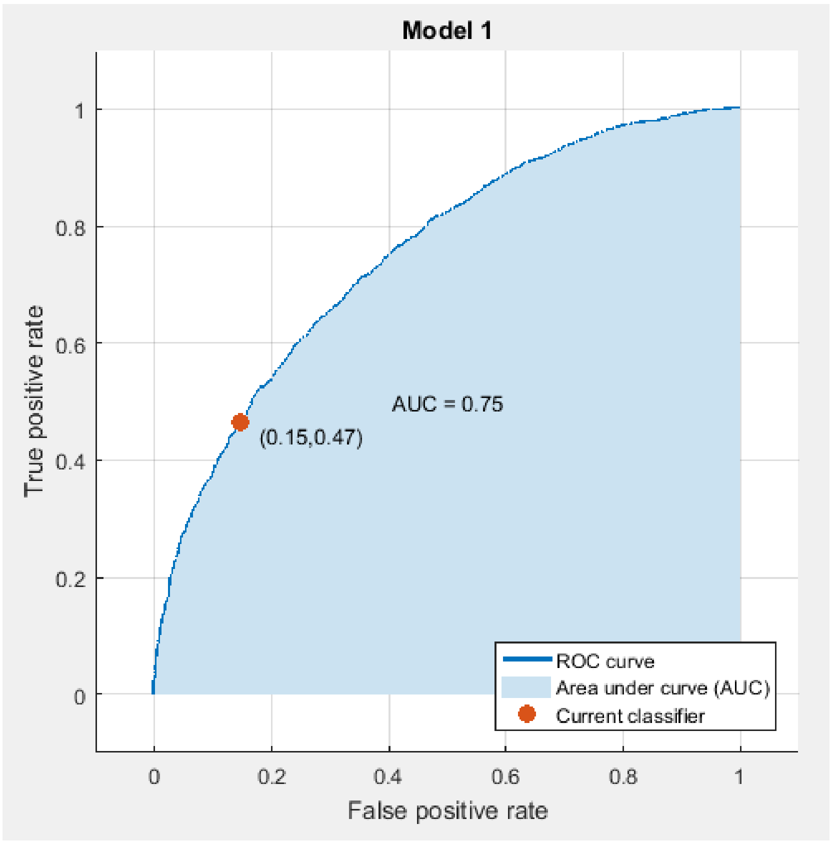
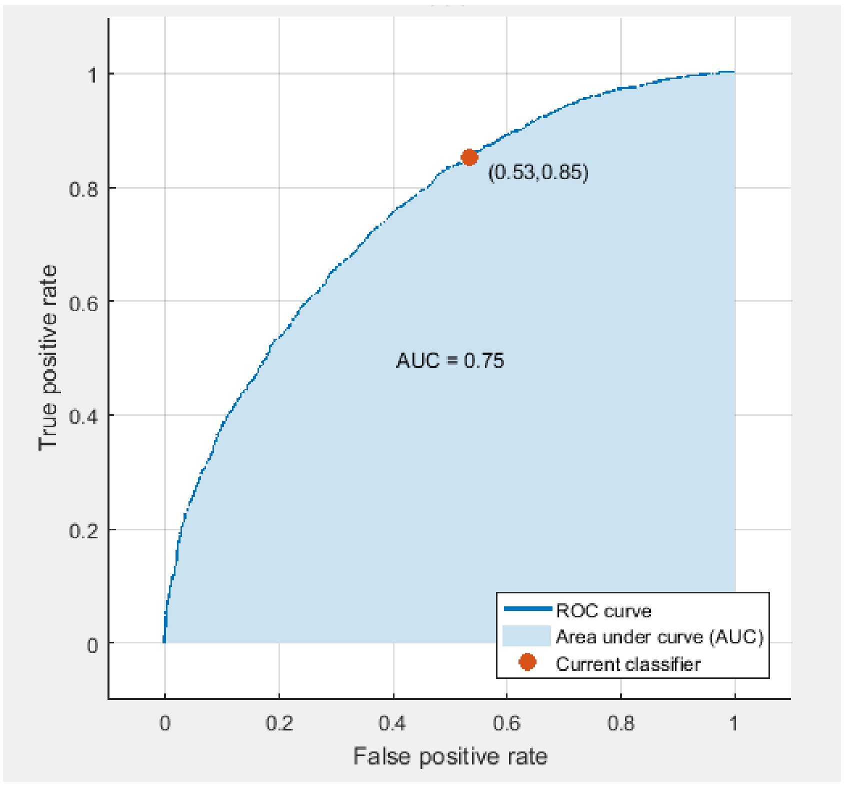


Fig (4.2.2c) Confusion Matrix from another perspective (False Discovery Rate)



Fig(4.2.2d) ROC curve visualization with K=10 and positive class = 0



Fig(4.2.2e) ROC curve with K=10 and positive class=1

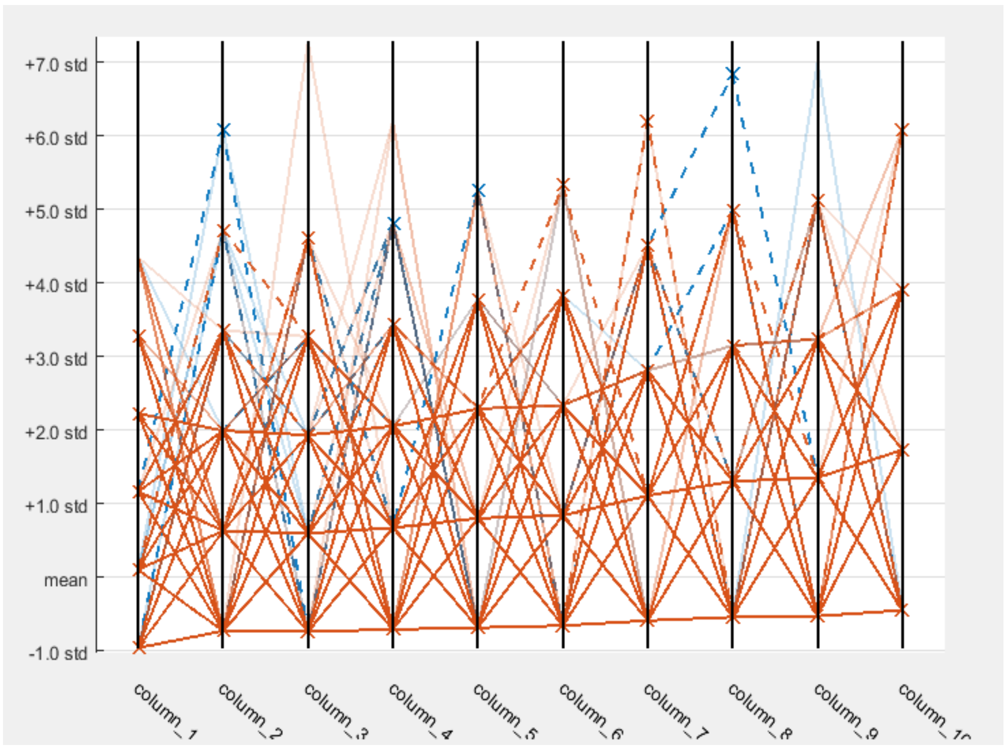


Fig (4.2.2f) Parallel coordinate plot for logistic regression model

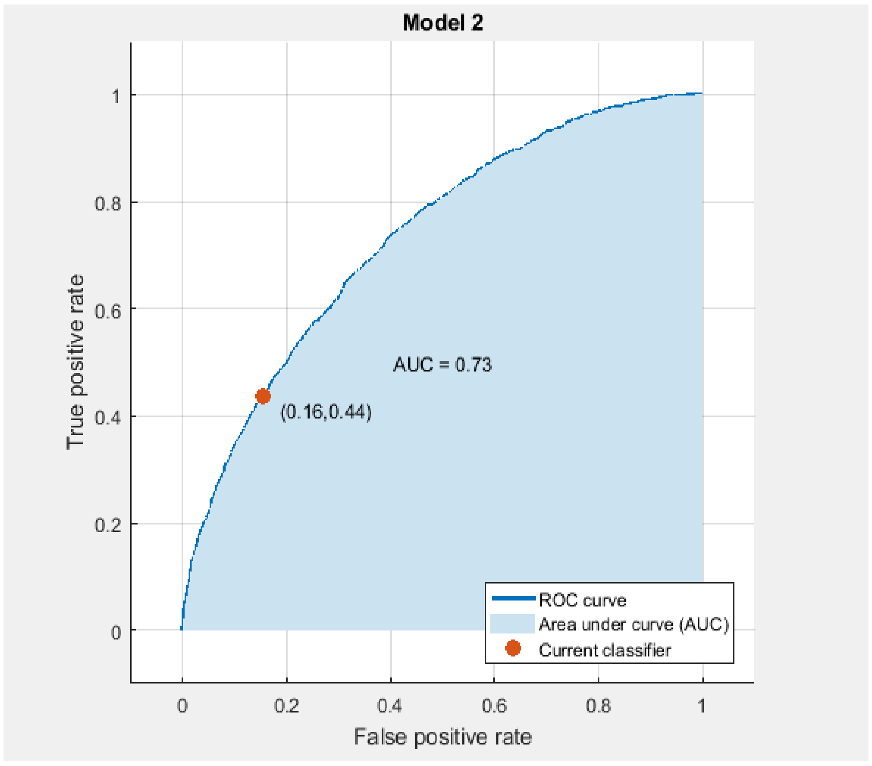
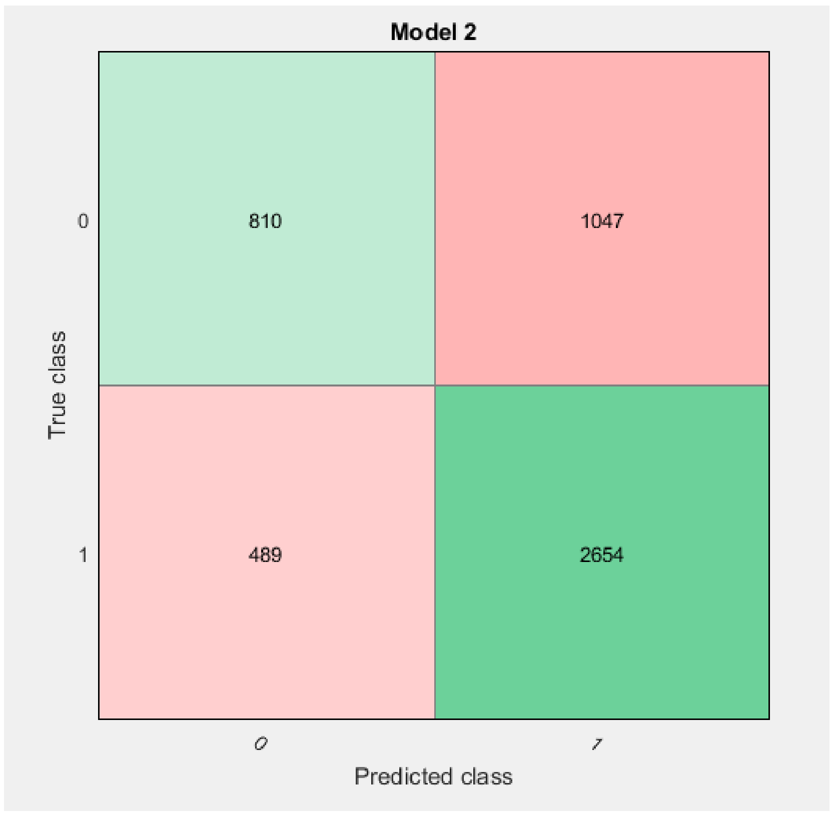


Fig (4.2.2g) ROC curve with PCA



Model 2
True class
Predicted class

True class \ Predicted class	0	1
0	810	1047
1	489	2654

Fig (4.2.2h) Confusion matrix with PCA and K=10

4.2.3 Multivariate Gaussian Distribution

The last method we used for classification was multivariate Gaussian distribution. We assume that our data comes from Gaussian distribution with parameter mean and covariance. Then we used the maximum likelihood of parameters to find class specific means and covariance. The maximum likelihood estimate of the mean and covariance is:

$$\mathbf{m}_i = \frac{\sum_{t=1}^{N} r_i^t \mathbf{x}^t}{\sum_{t=1}^{N} r_i^t},$$

$$\mathbf{S}_i = \frac{\sum_{t=1}^{N} r_i^t (\mathbf{x}^t - \mathbf{m}_i)(\mathbf{x}^t - \mathbf{m}_i)^T}{\sum_{t=1}^{N} r_i^t}, \text{ and } \hat{P}(C_i) = (1/N)\sum_{t=1}^{N} r_i^t$$

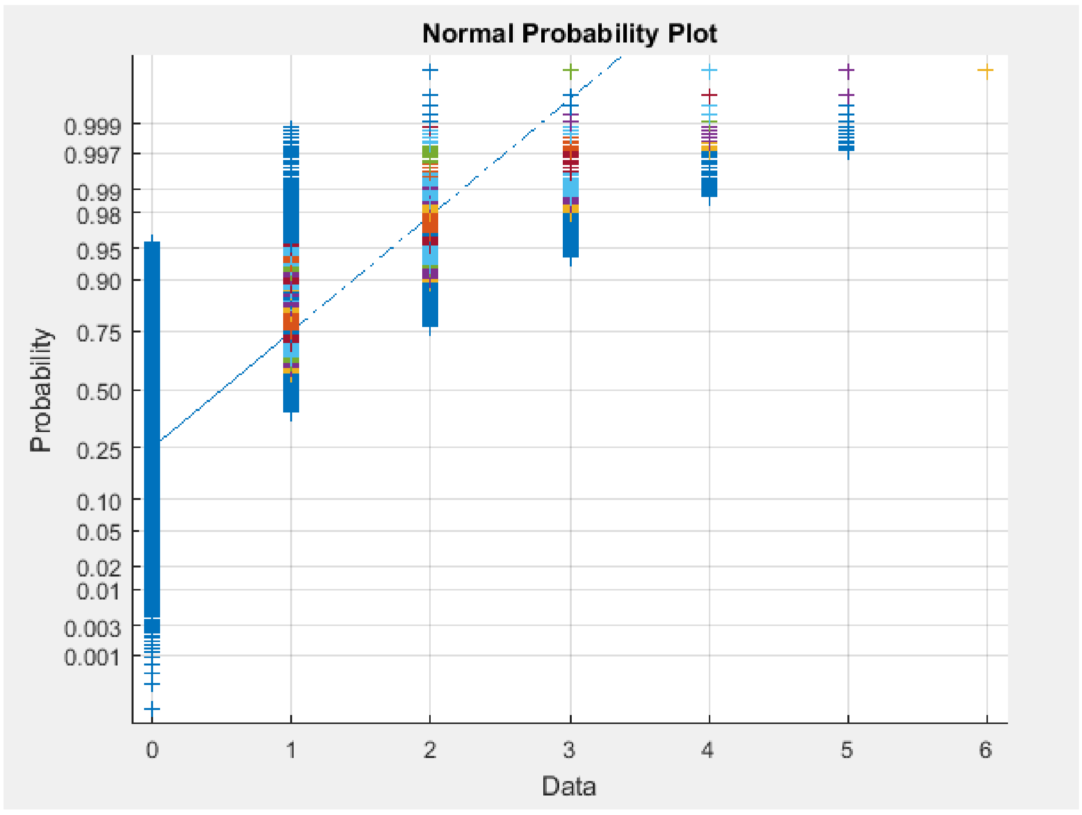
After finding maximum likelihood estimate of class specific mean and covariance, we are done with the training phase.

Then, next we calculate the posterior probability of test points, for each class and then we compare the posterior probability of both classes. We assign the class depends on higher posterior probability. The formula for posterior probability is given below:

$$P(C \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C) \times P(C)}{P(\mathbf{x})}$$

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}.$$

The accuracy with the training data set was 64.2% which was not a good result. Its mean that our assumption was not good, the data wasn't driven from Gaussian distribution. So, it's always better to first check whether the data belongs to normal distribution or any other. MATLAB provide us the tools to check the various distributions. Here, we check whether our data belongs to normal or not.



Here we see that, the data is deviated from the line, if it's around the line then we can say that it's from normal distribution. For better visualization we also plot with individual features and the results was same, the feature also not belongs to Gaussian distribution. Below plots will show normal probability plot for those features which we select by NCA.

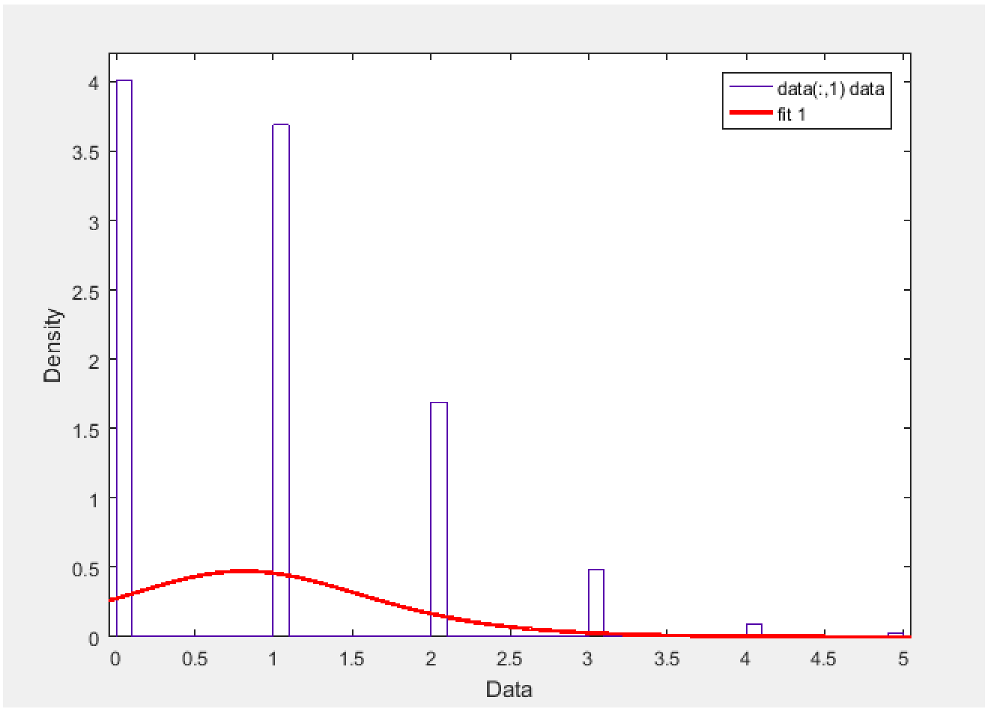


Fig (4.2.3a) Feature #1 ('but') normal probability distribution plot

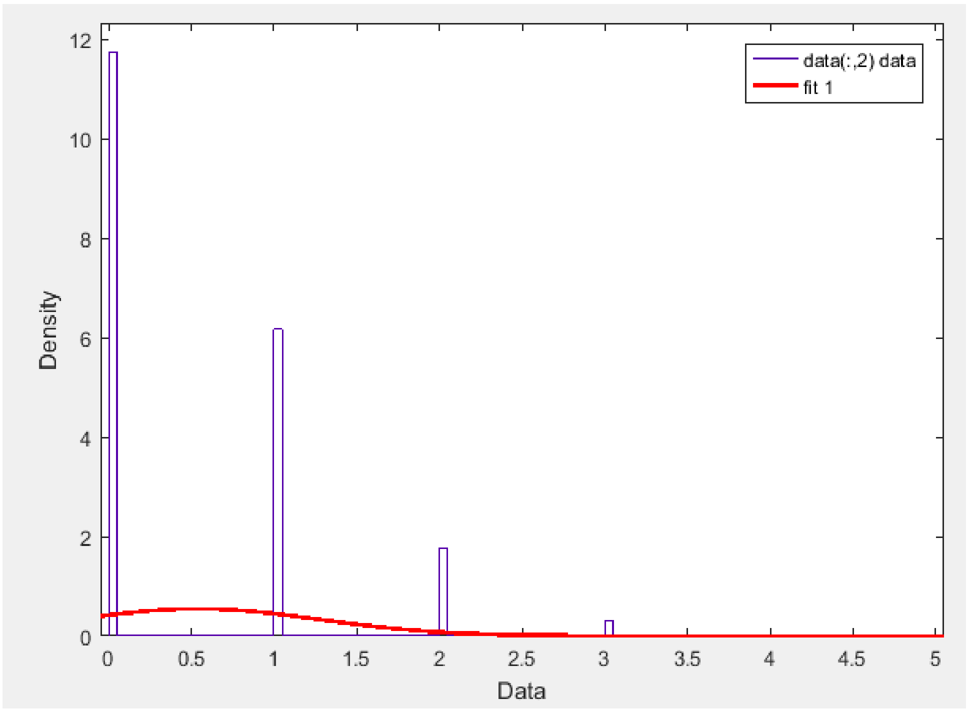


Fig (4.2.3b) Feature #2 ('good') normal probability distribution plot

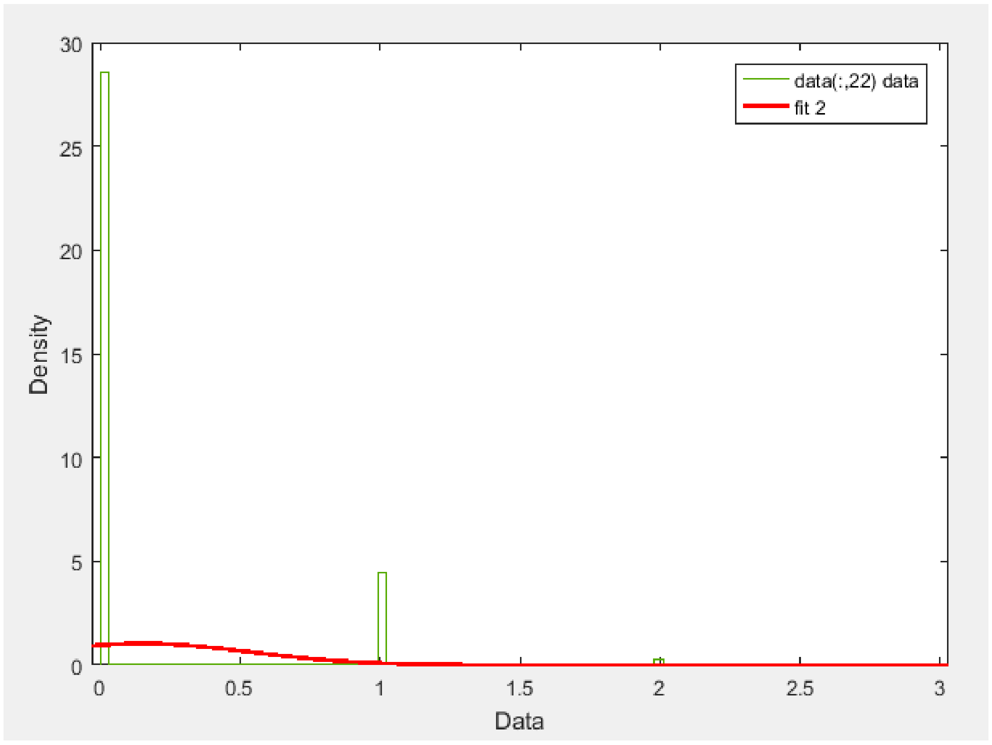


Fig (4.2.3c) Feature # 22 ('order') normal probability distribution plot

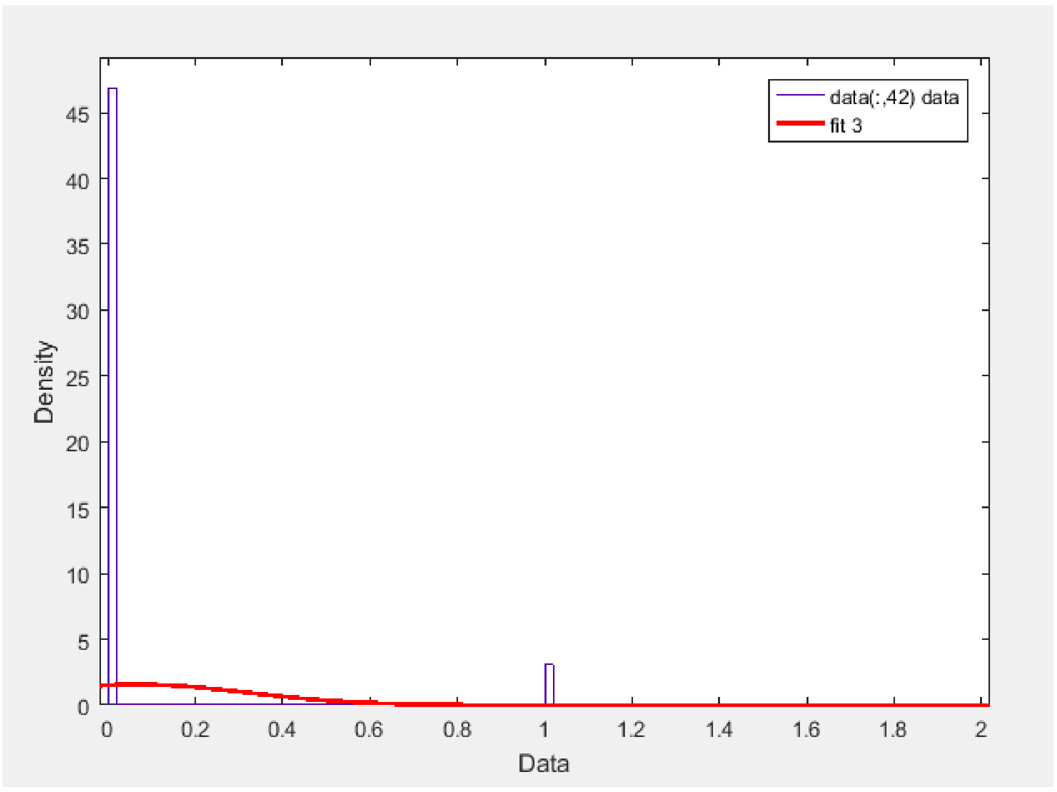


Fig (4.2.3d) Feature# 42 ('quality') normal probability distribution plot

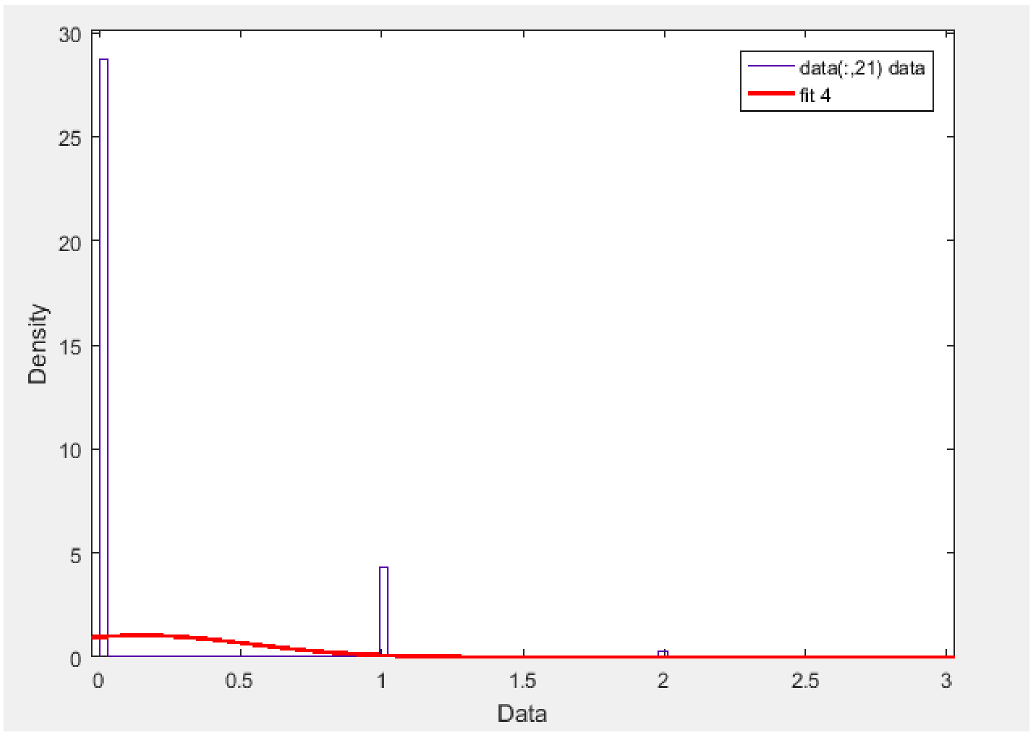


Fig (4.2.3e) Feature # 21 ('restaurant') normal probability distribution plot

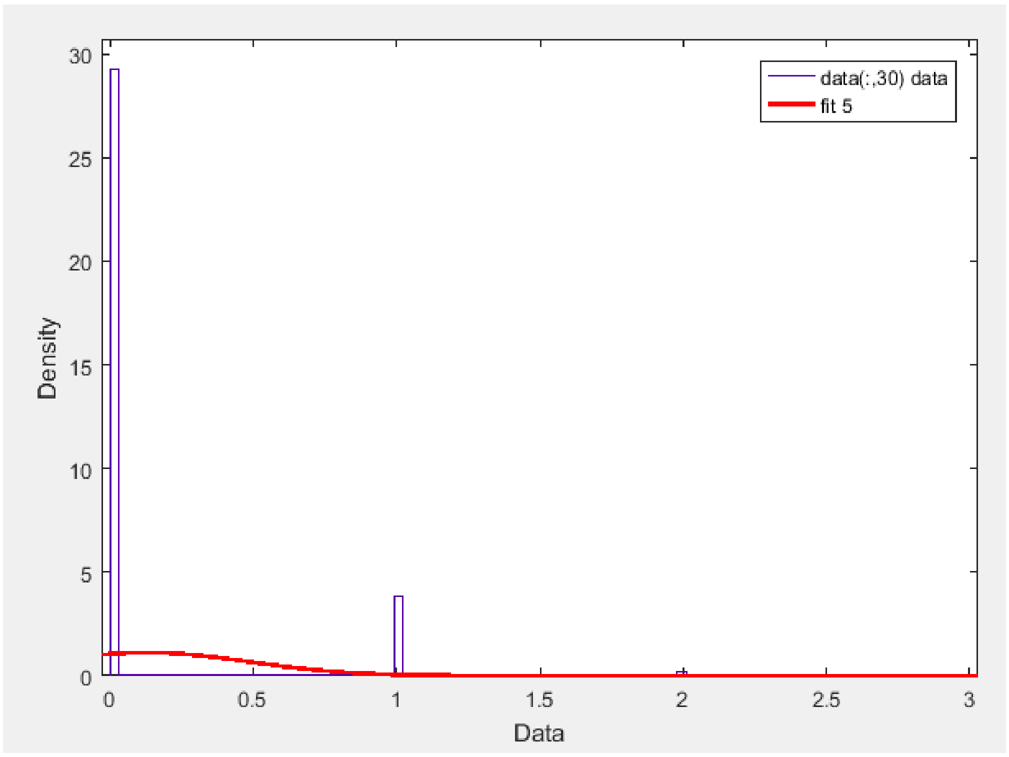


Fig (4.2.3f) Feature# 30 ('definitely') normal probability distribution plot

4.3 Regression

4.3.1 Multivariate Regression

We used multivariate regression with least squared error as criterion to check the validity of our model. For model assessment, we split the dataset in such a way that validation set has 30% of training data and we check our model against this validation data. The empirical error is given by:

$$\mathbf{w}_{opt} = \underset{\mathbf{w}}{\operatorname{argmin}}\, E(\mathbf{w}|\mathcal{D}) = \sum_{t=1}^{N} (r^t - \mathbf{w}^T \mathbf{x}^t)^2$$

The empirical error was 7% after model assessment and the validation error was 17%.

We increase the degree of polynomial and see that validation error decreases for higher order polynomial up to certain degree. We tested another cross validation technique called leave one out and it gives us more good results. We have chosen degree 3 after tried out both of these techniques.

4.3.2 K-d trees

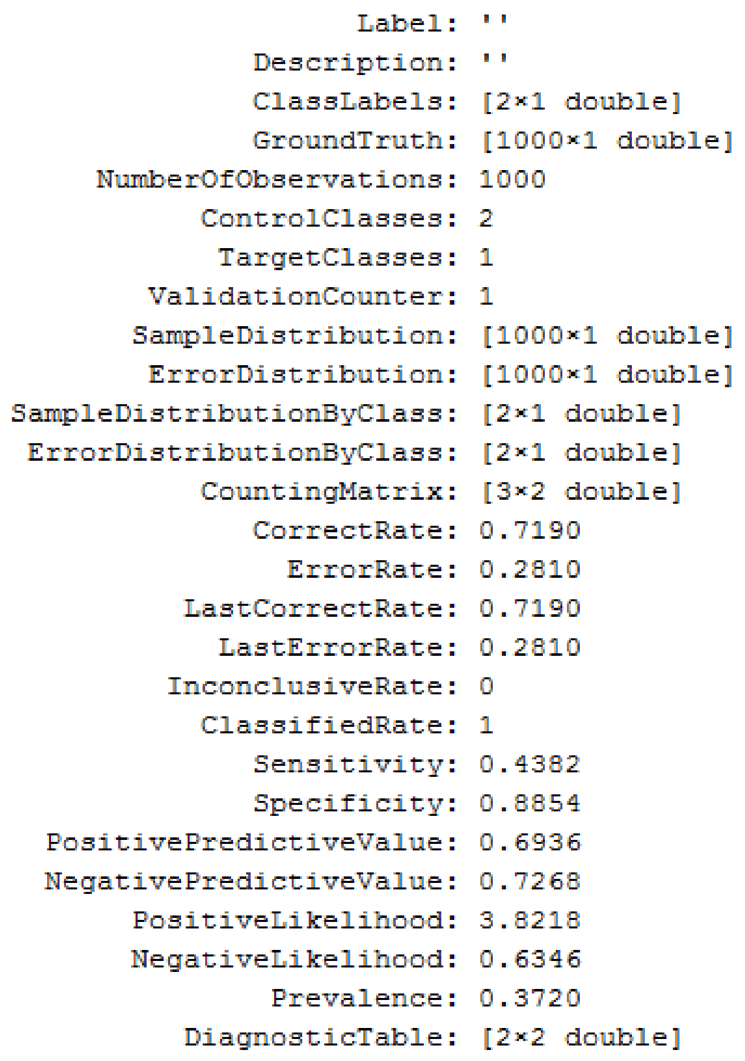
We used MATLAB’s own implementation of decision trees considering the fact that it provides a wide range of attributes and control over the parameters and validation options. The function we used was fittree for regression. We first trained the tree without using any validation and the results we obtained was far from accurate.

We then experimented with the different validation methods to minimize the generalization and the best we could get from it was 82% accuracy with KFolds cross validation.

5. Results

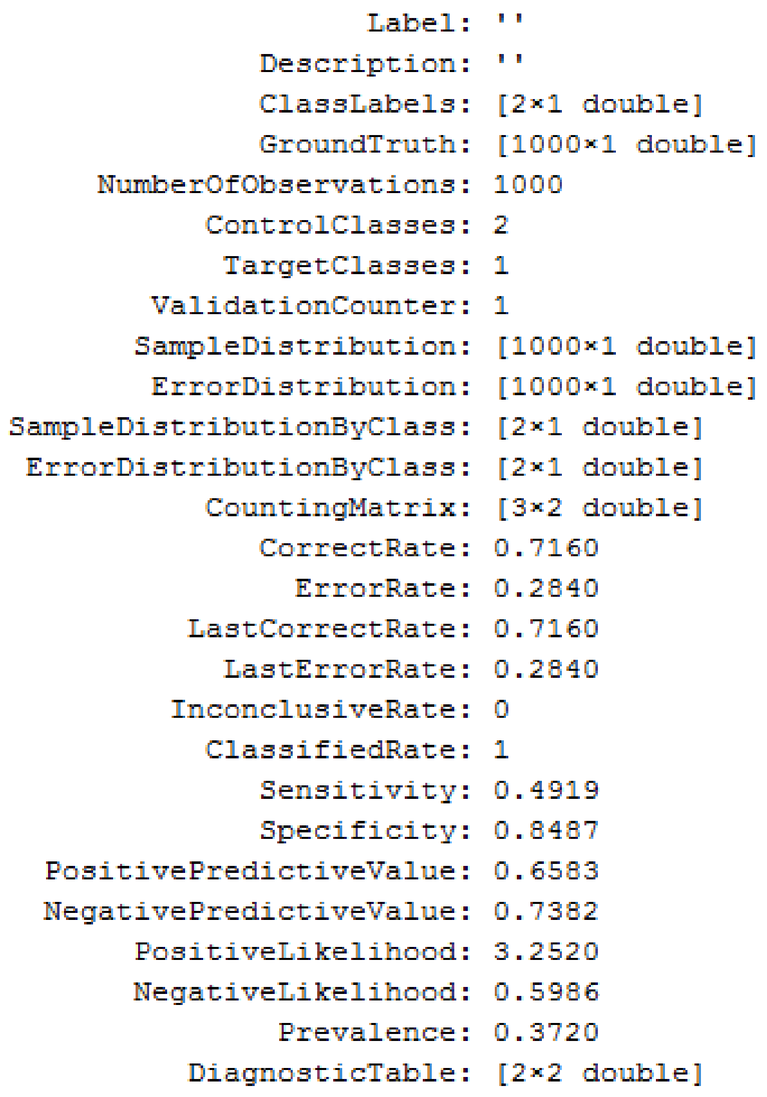
5.1 Classifications

5.1.1 Discrete Features



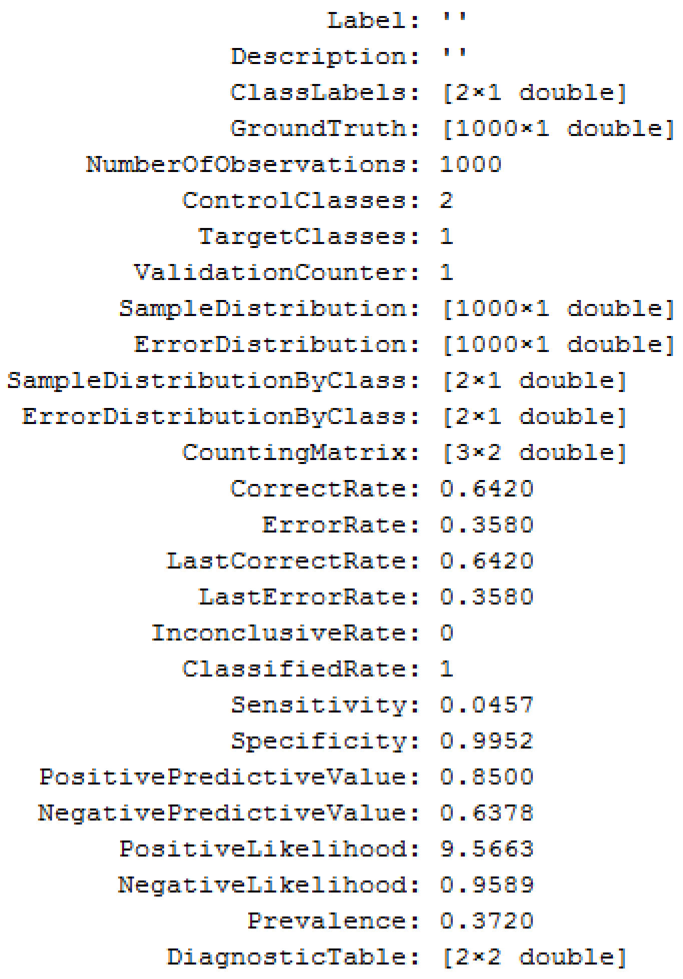
```
                   Label: ''
             Description: ''
             ClassLabels: [2×1 double]
             GroundTruth: [1000×1 double]
    NumberOfObservations: 1000
          ControlClasses: 2
           TargetClasses: 1
       ValidationCounter: 1
      SampleDistribution: [1000×1 double]
       ErrorDistribution: [1000×1 double]
SampleDistributionByClass: [2×1 double]
 ErrorDistributionByClass: [2×1 double]
          CountingMatrix: [3×2 double]
             CorrectRate: 0.7190
               ErrorRate: 0.2810
         LastCorrectRate: 0.7190
           LastErrorRate: 0.2810
        InconclusiveRate: 0
          ClassifiedRate: 1
             Sensitivity: 0.4382
             Specificity: 0.8854
 PositivePredictiveValue: 0.6936
 NegativePredictiveValue: 0.7268
      PositiveLikelihood: 3.8218
      NegativeLikelihood: 0.6346
              Prevalence: 0.3720
         DiagnosticTable: [2×2 double]
```

5.1.2 Logistic Regression



```
                    Label: ''
              Description: ''
              ClassLabels: [2×1 double]
              GroundTruth: [1000×1 double]
     NumberOfObservations: 1000
           ControlClasses: 2
            TargetClasses: 1
        ValidationCounter: 1
       SampleDistribution: [1000×1 double]
        ErrorDistribution: [1000×1 double]
SampleDistributionByClass: [2×1 double]
 ErrorDistributionByClass: [2×1 double]
           CountingMatrix: [3×2 double]
              CorrectRate: 0.7160
                ErrorRate: 0.2840
          LastCorrectRate: 0.7160
            LastErrorRate: 0.2840
         InconclusiveRate: 0
           ClassifiedRate: 1
              Sensitivity: 0.4919
              Specificity: 0.8487
  PositivePredictiveValue: 0.6583
  NegativePredictiveValue: 0.7382
       PositiveLikelihood: 3.2520
       NegativeLikelihood: 0.5986
               Prevalence: 0.3720
          DiagnosticTable: [2×2 double]
```

5.1.3 Multivariate Gaussian Distribution



```
                    Label: ''
              Description: ''
              ClassLabels: [2×1 double]
              GroundTruth: [1000×1 double]
     NumberOfObservations: 1000
           ControlClasses: 2
            TargetClasses: 1
        ValidationCounter: 1
       SampleDistribution: [1000×1 double]
        ErrorDistribution: [1000×1 double]
SampleDistributionByClass: [2×1 double]
 ErrorDistributionByClass: [2×1 double]
           CountingMatrix: [3×2 double]
              CorrectRate: 0.6420
                ErrorRate: 0.3580
          LastCorrectRate: 0.6420
            LastErrorRate: 0.3580
         InconclusiveRate: 0
           ClassifiedRate: 1
              Sensitivity: 0.0457
              Specificity: 0.9952
  PositivePredictiveValue: 0.8500
  NegativePredictiveValue: 0.6378
       PositiveLikelihood: 9.5663
       NegativeLikelihood: 0.9589
               Prevalence: 0.3720
          DiagnosticTable: [2×2 double]
```

5.2 Regression

5.2.1 Multivariate Regression

Accuracy on Test Data	Accuracy on Validation Set
83%	93%

5.2.2 K-D Trees

Accuracy on Test Data	Accuracy on Validation Set
74%	78%

6. Conclusion

When we use base learner method, we see that the majority class was ‘1’. If we assign 1 to all testing data the accuracy is 62%. We achieve more accuracy by applying discrete feature technique using Bernoulli approach and logistic regression. Hence, the modification in bag of words model will be good but not superb. With Gaussian distribution, the error was 35%, which is too much and we should not use the Gaussian distribution for prediction. So, its always better to test distribution of data before applying it. We can test many distributions such as Bernoulli, Gaussian, multinomial, dirichlet etc. SO, we should apply the probability distribution which give us the good result in testing process. For regression, multivariate regression is quite good as compared to k-d trees (in our case not always), but it all depends on data. We also see that by increasing the degree in polynomial fitting, we achieved less validation error up to certain degree and then it increases.

7. References

- Introduction to Machine Learning 2nd Edition, Ethem Alpaydin
- Pattern Recognition in Machine Learning by Christopher Bishop
- Lecture Notes by Andrew-Ng
- Statistics and Machine Learning Toolbox, MATLAB

8. Appendices

8.1 Source Code of Discrete Feature

```
%load data
data = csvread('classification_dataset_training.csv',1,1);
data_class0 = [];
data_class1 = [];
%pre-processing of data
for k=1:5000
    for l = 1:50
        if(data(k,l)>0)
            data(k,l)=1;
        end
    end
end
%split data
for i=1:5000
    if data(i,51)==0
        data_class0 = [data_class0; data(i,:)];
    else
        data_class1 = [data_class1; data(i,:)];
    end
end

%estimate Pij
pij0 = [];
pij1 = [];

for j=1:50
    pij_0 = sum(data_class0(:,j))/1857;
    pij_1 = sum(data_class1(:,j))/3143;
    pij0 = [pij0 pij_0];
    pij1 = [pij1 pij_1];
end

%check test case
test_data = csvread('classification_dataset_testing.csv',1,1);
%pre-process the test data
for k=1:1000
```

```
    for l = 1:50
        if(test_data(k,l)>0)
            test_data(k,l)=1;
        end
    end
end
test_result = zeros(1000,1);
%discriminінat function
for m = 1:1000
    sum0 = 0;
    sum1 = 0;
    for n=1:50
        sum0 = sum0 + (test_data(m,n)*log(pij0(1,n))) + ((1-
test_data(m,n))*log(1-pij0(1,n)));
        sum1 = sum1 + (test_data(m,n)*log(pij1(1,n))) + ((1-
test_data(m,n))*log(1-pij1(1,n)));
    end
    %calculate gix
    gix0 = sum0  + log((1857/5000));
    gix1 = sum1  + log((3143/5000));
    if (gix0<gix1)
        test_result(m,1) = 1;

    end
end
sol = csvread('classification_dataset_testing_solution.csv',1,1);
p = accuracy(sol,test_result)
```

8.2 Source Code of Logistic Regression

```
%logistic regression

%loading datasets
data = csvread('classification_dataset_training.csv',1,1);

%predictors
X = data(:,1:50);

%responses
Y = data(:,51);

%find-coefficients
b = glmfit(X,Y,'binomial','link','logit');

%load test dataset
testData = csvread('classification_dataset_testing.csv',1,1);

%split testData into predictors and responses
X_test = testData(:,1:50);

%apply logistic function on testData predictors
yfit = glmval(b,X_test,'logit');

%give classes
for i=1:1000
    if yfit(i,1)>=0.5
        yfit(i,1) = 1;
    else
        yfit(i,1) = 0;
    end
end

%check accuracy
solution = csvread('classification_dataset_testing_solution.csv',1,1);
p = accuracy(solution,yfit);
```

8.3 Source code for Multivariate Gaussian

```
%I assume my model is gaussian distribution, so first step is to find
parameters
%load data
data = csvread('classification_dataset_training.csv',1,1);
data_class0 = [];
data_class1 = [];
%split data
for i=1:5000
    if data(i,51)==0
        data_class0 = [data_class0; data(i,:)];
    else
        data_class1 = [data_class1; data(i,:)];
    end
end
%get mean vector
mean_class0 = [];
mean_class1 = [];

for i=1:50
    m0 = sum(data_class0(:,i)/1857);
    m1 = sum(data_class1(:,i)/3143);
    mean_class0 = [mean_class0 m0];
    mean_class1 = [mean_class1 m1];
end

%finding covariances
covariance_class0 = cov(data_class0(:,1:50));
covariance_class1 = cov(data_class1(:,1:50));
%calculate prior
p0 = 1857/5000;
p1 = 3143/5000;
%load test data
testData = csvread('classification_dataset_testing.csv',1,1);
testDataresult = zeros(1000,1);
for i=1:1000
    disc_class0 = mvnpdf(testData(i,:),mean_class0);
    posterior_class0 = p0 * disc_class0;
    disc_class1 = mvnpdf(testData(i,:),mean_class1);
    posterior_class1 = p1 * disc_class1;
    if (posterior_class0<posterior_class1)
        testDataresult(i,1) = 1;
    end


end
sol = csvread('classification_dataset_testing_solution.csv',1,1);
p = accuracy(sol,testDataresult)
```

8.4 Source Code of Feature Selection

```matlab
%nca feature selection
data = csvread('classification_dataset_training.csv',1,1);
X = data(:,1:50);
y = data(:,51);
mdl = fscnca(X,y)
figure()
plot(mdl.FeatureWeights,'ro')
grid on
xlabel('Feature index')
ylabel('Feature weight')

c = cvpartition(y,'k',100);
opts = statset('display','iter');
fun = @(XT,yT,Xt,yt)...
      (sum(~strcmp(yt,classify(Xt,XT,yT,'quadratic'))));

[fs,history] = sequentialfs(fun,data(:,40:50),y,'cv',c,'options',opts)

maxdev = chi2inv(.95,1);
opt = statset('display','iter',...
              'TolFun',maxdev,...
              'TolTypeFun','abs');

inmodel = sequentialfs(@critfun,X,y,...
                       'cv','none',...
                       'nullmodel',true,...
                       'options',opt,...
                       'direction','forward');
```