Artificial Intelligence: knowledge representation and planning

Year: 2018/2019

Assignment 2

SpamFilter

Author: **Cabrera Christian Bernabe**

26th April 2019

Index

1 Assignment goal

Write a spam filter using **discriminative** and **generative** classifiers. Use the Spambase dataset which already represents spam/ham messages through a bag-of-words representations through a dictionary of 48 highly discriminative words and 6 characters. The first 54 features correspond to word/symbols frequencies; ignore features 55-57; feature 58 is the class label (1 spam/0 ham).

- Perform SVM classification using linear, polynomial of degree 2, and RBF kernels over the TF-IDF representation. Can you transform the kernels to make use of angular information only (i.e., no length)? Are they still positive definite kernels?
- Classify the same data also through a Naive Bayes classifier for continuous inputs, modeling each feature with a Gaussian distribution, resulting in the following model:

$$p(y = k) = \alpha_k$$

$$p(x|y = k) = \prod_{i=1}^{D} \left[(2\pi\sigma_{ki}^2)^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2\sigma_{ki}^2}(x_i - \mu_{ki})^2 \right\} \right]$$

where α_k is the frequency of class k, and μ_{ki}, σ_{ki}^2 are the means and variances of feature i given that the data is in class k.
Provide the code, the models on the training set, and the respective performances in 10 way cross validation. Explain the differences between the two models.

2 Language and structure

The task has been done in Python. For every algorithm there is a single file. Specifically we have used the python library **sklearn**, that has the SVM classifier already developed.
To do this assignment both of us worked together. The reports have been written separately.

3 Introduction to the problem

The machine learning algorithms are usually divided into categories according to their purpose. The main categories are **Supervised Learning**, **Unsupervised Learning** and **Reinforcement Learning**.
In **Supervised learning** some examples of inputs and the corresponding outputs are given to the algorithm. The main goal is then to learn a general formula that maches inputs into outputs. More formally, it takes in input a set of observations X and its labels Y, and learns a model considering the pairs (x_i, y_i).
In **Unsupervised learning** algorithms, no labeled data are given and the algorithm try to learn a model for prediction. They can be useful to find hidden patterns.
Finally, **reinforcement learning** algorithms continuously learn from the environment in an iterative way. An agent learns from its experiences until it covers all the possible states. This kind of algorithms aim to using the observations collected from the interaction to make decisions that try to minimize the risk of maximize the reward.

In machine learning and statistics one of the most common problem is represented by **classification**. When talking about classification problems, we mean checking if an object belongs to a category. Given one or more inputs a classification model will try to predict the value of one or more outcomes. Basically classification either predicts categorical class labels or classifies data (construct a model) based on the training set and the values (class labels) in classifying attributes and uses it in classifying new data. In the terminology of machine learning, classification is considered an instance of supervised learning, i.e., learning where a training set of correctly identified observations is available. For this assignment the labels are two: **1** denotes that the email was considered Spam and **0** denotes that the email was considered Ham. In other words, "non-spam".

If we let $X = (X_1, ..., X_m) \in \mathcal{X}$ to be the input of our problem and let $Y \in \mathcal{Y}$ the label variable of the class of the observations we can define the classification problem as the function $\phi : \mathcal{X} \to \mathcal{Y}$ from a training set.

The task of this assignment is to make train the algorithm in order to recognize if an email is spam or not. We can then say that it belongs to the Supervised Learning category.
Supervised Learning can be also grouped into Classification and Regression problems and the this assignment is falls into classification of emails.

We can as well divide classifiers into **Generative** and **Discriminative**:

- A generative classifier learns the joint probability distribution $p(x, y)$ and then it predicts the conditional probability with the help of the Bayes Theorem.
- Discriminative classifiers model the posterior probability $p(y|x)$ directly, or learn a direct map from inputs x to the labels y.

The two tasks of this assignment are to use SVM classification using linear, polynomial and RBF kernels and to use also the Naive Bayes classifier. The first one is a type of discriminative model while the Naive Bayes is a generative type of model.

4 Support Vector Machine

The first model that we are going to analyze is the **Support Vector Machine** (SVM) classifier. It is based on the concept of decision planes that define decision boundaries. A decision plane(hyperplane) is one that separates between a set of objects having different classes.
We can formally define the SVM linear classifier in this way:

$$h_{w,b}(x) = g(w^T x + b)$$

$$g(z) = \begin{cases} 1 & \text{if} \quad z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Where $w^T + b$ represents a **separating hyperplane** for the two classes, and $g(z)$ is the decision function which returns the predicted class for the observation x.
We define the probability of classification given an observation x as:

$$p(y = 1|x; w, b)$$

In our problem the classifier would then predict 1 (spam) if $h_{w,b}(x) \geq 0$, which is equivalent of considering $w^T x + b \geq 0$, and -1 (ham) otherwise.

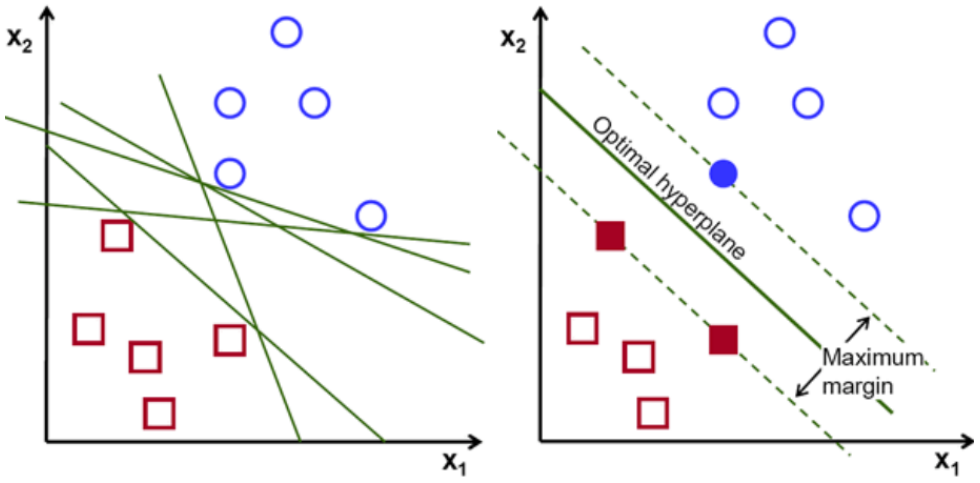

Figure 1: Support Vector Machines classifier

The figure on the left introduce the problem of which one is the right hyperplane. We can have several possible decision boundaries.
The SVM in particular defines the criterion to be looking for a decision surface that is maximally far away from any data point. This distance from the decision surface to the closest data point determines the margin of the classifier.

Functional margin can be seen as a scoring function, where the score becomes increasing positive as you move from the decision boundary in the direction of the y = 1 class, and increasing negative as you move in the other direction from the boundary. Formally speaking is defined as:

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

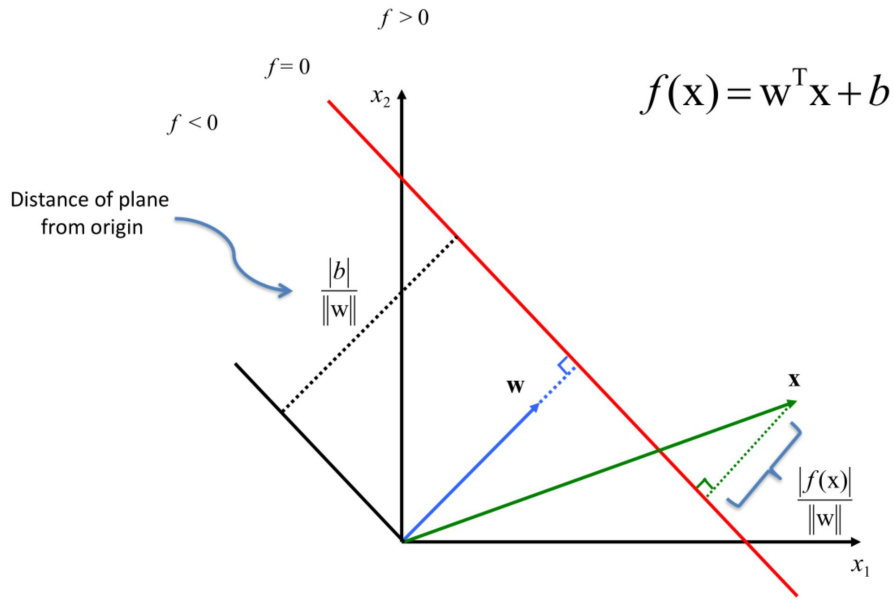

Figure 2: Geometric facts

Geometric margin simply represents the euclidean distance between a point and the decision boundary.
We can define the geometrical margin γ_i as the length of the segment.

The geometric margin is related with the functional margin by: $\gamma = \frac{\hat{\gamma}}{||w||}$.

The main idea of an SVM classifier is to find a separating hyperplane that maximizes the geometric margin. Among other choices, we could use unit vectors by requiring that $||w|| = 1$ since this would have the effect of making the geometric margin the same as the functional margin.
Learning the SVM can be formulated as an optimization problem:

$$\begin{aligned} \max \quad & \frac{\hat{\gamma}}{||w||} \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq \hat{\gamma} \qquad i = 1, \ldots, N \end{aligned}$$

Or equivalently imposing the scaling constraint $\hat{\gamma} = 1$, can be seen as a minimization problem of $||w||^2$:

$$\begin{aligned} \min \quad & \frac{1}{2}||w||^2 \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 \qquad i = 1, \ldots, N \end{aligned}$$

This is a (convex) quadratic optimization problem subject to linear constraints and there is a unique minimum. The points with the minimum margin from the decision boundary are called **support vectors**. The advantage of SVM is that instead of considering all the possible points SV from the training set it considers only these points to determine the decision boundary. To solve this constrained optimization problem, we introduce the Lagrange function with N Lagrange multipliers $(\lambda_1, \ldots, \lambda_m)$ and considering the dual problem, the maximum margin

separating hyperplane can be found by solving the following optimization problem::

$$\begin{aligned} \max \quad & L_D(w, b, \Lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j \\ \text{s.t.} \quad & \sum_{i=1}^{N} \lambda_i y_i = 0 \qquad \lambda_i \geq 0, \forall i = 1, \ldots, N \\ \text{where} \quad & \Lambda = (\lambda_1, \ldots, \lambda_N) \text{ is the vector of Lagrange multipliers.} \end{aligned}$$

This is a standard type of optimization problem called a convex quadratic programming problem for which there are well-known and efficient algorithms. Through solving this optimization, the following equations are satisfied, which provide the equation for the hyperplane given by:

$$\sum_{i=1}^{N} y_i \lambda_i x_i^T x + b = 0$$

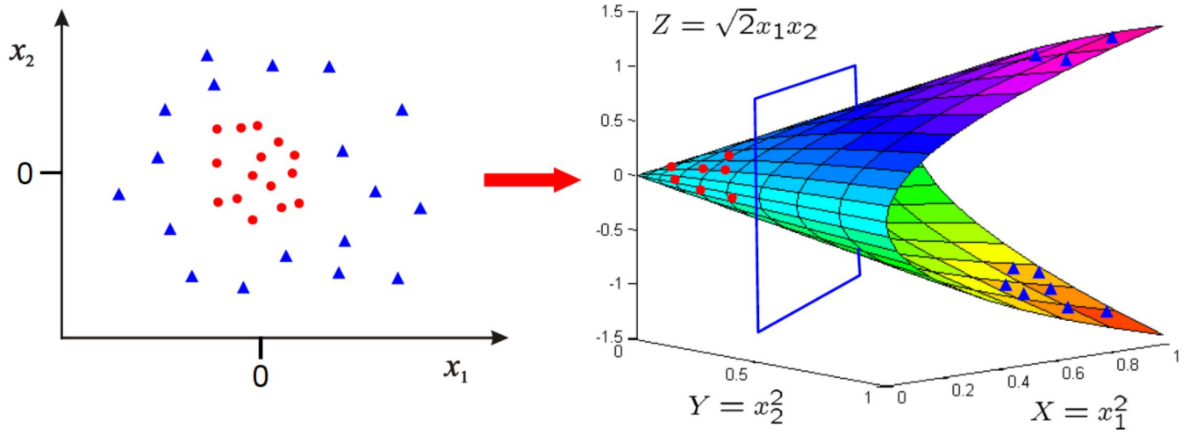

Figure 3: Data is still linearly separable in 3D

SVM origin arises from the idea of mapping the feature vectors in a nonlinear way to a high dimensional space and then utilize linear classifiers in this new space. Specifically we let H to be the new feature space and let ϕ denote the mapping such that: $\mathrm{X} \rightarrow \phi(x)$. This is also called **Kernel Trick**. SVMs are a special case of generalized kernel methods where a a kernel (or window) function is nothing else that an inner product between feature mappings of ϕ:

$$K(x, y) = \phi(x)^T \phi(y)$$

Since a kernel corresponds to a legal definition of a dot product, we require it to be valid, in other words symmetric and positive definite on X.

A function is a valid kernel function if it is a real-valued positive definite function, whose definition is recalled below:

Definition A symmetric function $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be **positive definite kernel** if:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0 \qquad c_i, c_j \in \mathbb{R}$$

In this assignment we are asked to perform the operations using the following kernels:

- **Linear**: $K(x_i, x_j) = x_i^T x_j$
- **Polynomial of degree 2**: $K(x_i, x_j) = (1 + x_i^T x_j)^2$
- **Gaussian Radial Basis Function** (RBF): $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

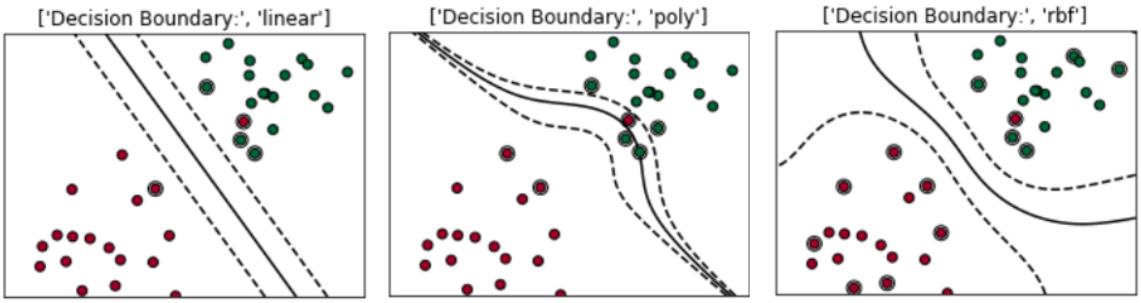

Figure 4: SVM: Linear Kernel, Polynomial Kernel, RBF Kernel

4.1 Angular kernels

In this assignment we are also asked to build new kernels that make use of angular information only. This because the given kernels for this assignment are affected by the length of the vectors. As a consequence of that, then two documents with very similar content can have a significant vector difference simply because one is much longer than the other. To compensate for the effect of document length, the standard way of quantifying the similarity between two documents d_1 and d_2 is to compute the cosine similarity of their vector representations $\mathrm{V}(d_1)$ and $\mathrm{V}(d_2)$. The effect of the denominator of Equation is thus to length-normalize **lenght-normalize** the vectors as:

$$\phi : \mathbb{R}^m \to \mathbb{R}^m \qquad \phi(x) = \frac{x}{||x||}$$

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = \frac{x_i}{||x_i||} \frac{x_j}{||x_j||} = \cos(x_i, x_j)$$

We can see that this kernel is not affected by the length of the two vector but it only considers the *cosine* of the angle create by two vectors x_i, x_j.

With this simple trick we can consider the angular information.

5 SVM Implementation

The assignment requires to apply the kernels over the TF-IDF representation. However the dataset given to us is presented as a tf*100. The first thing to do was to perform the transformation. We have:

$$tf - idf_{t,d} = tf_{t,d} \times idf_t$$

$$\text{score}(q, d) = \sum tf - idf_{t,d}$$

$$idf_t = \ln \frac{N}{|df_t}$$

Where:

- $tf_{t,d}$ number of occurrences of the term t in the document d.
- idf_t is the inverse document frequency of the term t which measures its relevance.Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low.
- $tf - idf_{t,d}$ is a wieghting scheme that assigns to term t a wight in a document d, that is highest when t occurs many times within a small number of documents, ower when the term occurs fewer times in a document, or occurs in many or lowest when the term occurs in virtually all documents.
- score(t, d) is the score of the term t inside the document d.
- D collection of documents.

The second step is to implement the kernels which is straightforward using the class **sklearn.svm.SVC** (C-Support Vector Classification). This class supports different kernel types to be used in the algorithm. The supported kernels are 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used.

To perform the respective performances in 10 way cross validation as required by the assignment the library provides a helper function to use it, called **cross val score**.
Cross validation is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. This function uses the basic approach known as k-fold Cross Validation were the training set is split into k smaller sets. The following steps are followed for each of the k "folds":

- A model is trained using k-1 of the folds as training data;
- The resulting model is validated on the remaining part of the data.

The performance measure reported by k-fold Cross Validation is then the average of the values computed in the loop. This may be computationally expensive but does not waste too much data, which is a major advantage when the number of samples is very small. Here we can see the results obtained:

Kernel Type	**Mean score**	**Min score**	**Max score**	**MSE**
Linear	0.6079	0.6035	0.6174	0.3921
RBF	0.6059	0.6052	0.6065	0.3940
Polynomial degree: 2	0.6059	0.6052	0.6065	0.3940

As we can see none of the kernels seem to be more accurate than the others. This is because they are considering the vectors length and as we said previously this is generally not such a good idea.
Let's consider know the normalized dataset results.

Kernel Type	**Mean score**	**Min score**	**Max score**	**MSE**
Linear	0.9232	0.8409	0.9717	0.0768
RBF	0.9121	0.8214	0.9456	0.0879
Polynomial degree: 2	0.6059	0.6052	0.6065	0.3940

From this test we can express some considerations. Generally speaking we can notice that both Linear and RBF kernels did in fact benefit from using the angular information. Also the mean squared error for the first ones is quiet high, indicating that those models aren't very good. Overall for this dataset the Linear kernel seems to provide good results even better than rbf.

To perform some tests I decided to split the dataset into two parts. I opted for 75% of the dataset to be used for training purposes, while the remaining 25% for the test section. The split is being done randomly at each execution of the program. The goal of the svm is to find the optimal hyperplane f(x) denoted as $w^T x + b$. In order to optimize it we need to choose the best values for w and for b. By executing the code 20 times I found out a model with around 0.9365768 of accuracy. The best parameter for b was -1.0690701168413104 and w equal to:

-0.23	0.06	-0.14	1.98
1.44	0.93	2.38	1.58
1.30	1.00	-0.20	-0.74
-0.28	0.34	1.02	1.96
1.65	0.88	0.06	1.62
1.11	1.80	1.86	2.16
-2.29	-0.75	-2.20	0.25
-0.99	-0.40	-0.66	-0.63
-0.18	-0.02	-0.68	1.29
-0.07	0.23	-0.53	0.26
-0.62	-0.78	-0.63	-1.17
-0.95	-1.57	-0.44	-0.46
-1.55	-0.64	-0.35	2.39
4.12	1.58		

At the end of the training set I ended up with 735 support vectors divided as 370 for 0 and 365 for 1.

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0.38	0
0.34	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0.85
0	0.05	0	0.03
0	0		

Figure 5: example of support vector

6 Naive Bayes Classifier and implementation

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable and dependent feature vector x_1 through x_n.

$$p(y|x_1,\ldots,x_n) = \frac{p(x_1,\ldots,x_n)p(y)}{P(x_1,\ldots,x_n)}$$

We use the naive conditional independence assumption that is stated as

$$p(x_i|y,x_1,\ldots,x_{i-1},x_{i+1}\ldots,x_n) = P(x_i|y)$$

for every i and we can get:

$$P(x_i|y) = \prod_{i=1}^{n} P(x_i|y)$$

With the posterior probability can then simplify the relationship into:

$$p(y|x_1,\ldots,x_n) = \frac{p(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,\ldots,x_n)}$$

Since $P(x_1,\ldots,x_n)$ is constant given the input, we can use the classification rule. Also the Naive Bayes classification problem then becomes: for different class values of y, find the maximum of $P(y)\prod_{i=1}^{n} P(x_i|y)$. All of this can be formulated as follows:

$$\begin{aligned} P(y \mid x_1,\ldots,x_n) &\propto P(y)\prod_{i=1}^{n} P(x_i \mid y) \\ &\Downarrow \\ \hat{y} &= \arg\max_y P(y)\prod_{i=1}^{n} P(x_i \mid y), \end{aligned} \tag{1}$$

where $\propto$ means positive proportional to. We can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class in the training set.

However we still don't know the quantity of $p(x_i|y)$. Since in this assignment we assume that the classifier is a Guassian Naive Bayes, then the feature distributions on the classes (spam or ham) will be Gaussian distributions. So:

$$p(x|y=0) = N(x|\mu_0,\sigma_0) = (2\pi|\sigma_0|^2)^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(x-\mu_0)^2|\sigma_0|^{-2}\right\}$$

$$p(x|y=1) = N(x|\mu_1,\sigma_1) = (2\pi|\sigma_1|^2)^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(x-\mu_1)^2|\sigma_1|^{-2}\right\}$$

We are interested in calculating the posterior probability from the likelihood and prior probabilities. Pratically speaking, the likelihood $P(x_i|y)$ are usually modelled using the same class of probability distribution. The different Naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$.
In p dimensions the formula becomes:

$$p(x|y=0) = (2\pi|\Sigma_0|^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu_0)^T|\Sigma_0|^{-1}(x-\mu_0)\right\}$$
$$p(x|y=1) = (2\pi|\Sigma_1|^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu_1)^T|\Sigma_1|^{-1}(x-\mu_1)\right\}$$

So given a set of i.i.d. data $x_i = x_1, \ldots, x_n$ drawn from N(x;μ, Σ) we want to estimate (μ, Σ) by MLE. The log-likelihood function is

$$\ln p(x|\mu, \Sigma) = -\frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^T\Sigma^{-1}(x_n-\mu)$$

After some transformations and derivatives we obtain:

$$\mu = \frac{1}{N}\sum_{n=1}^{N}(x_n)$$

$$\Sigma_k = \frac{1}{n}\sum_{n=1}^{N}(x_i-\mu_k)(x_i-\mu_k)^T$$

The classification basically involves assigning the class C_k to the data point for which the value p(C_k|x) is the greatest.
Since determining the most likely class for a data point $x_i = x_1, \ldots, x_n$ consists of calculating the product of n+1 factors K times, the big O notation for the complexity of classification is O(nK). After having executed k-fold Cross Validation, we can notice (from the table below), that the results obtained through Naive Bayes are worse than the ones with the SVM obtained earlier.
After having split the data randomly at each execution (75% for training and 25% for test purposes) and run the prediction function on 30 different executions, I have never got a model with a prediction score higher than 0.82% with the best one having $p(ham) = 0.607$ and $p(spam) = 0.393$.

Classifier	**Mean score**	**Min score**	**Max score**	**MSE**
Naive Bayes	0.8149	0.6230	0.8847	0.1851

9.00E-06	2.54E-04	2.59E-05	1.01E-08
4.07E-05	5.28E-06	1.31E-06	6.33E-06
4.57E-06	4.66E-05	1.96E-06	1.00E-04
7.10E-06	1.31E-05	9.51E-07	4.59E-05
4.25E-06	1.59E-05	3.23E-04	1.16E-06
1.04E-04	3.86E-05	3.90E-07	4.93E-06
3.97E-04	1.26E-04	1.80E-03	3.94E-05
5.91E-05	3.38E-05	2.12E-05	1.71E-05
4.87E-05	1.72E-05	4.86E-05	2.31E-05
2.36E-05	9.47E-06	2.41E-05	1.87E-05
2.00E-05	1.03E-04	6.80E-06	7.00E-05
1.71E-04	1.47E-04	9.34E-07	1.37E-05
1.04E-05	7.14E-06	1.94E-06	8.37E-05
4.97E-07	7.61E-06		

Figure 6: Spam: mean value features of Σ_0

1.01E-05	1.30E-05	2.41E-05	5.00E-04
4.77E-05	1.04E-05	3.21E-05	3.29E-05
1.25E-05	3.50E-05	6.81E-06	4.27E-05
1.40E-05	1.11E-05	1.29E-05	8.47E-05
4.21E-05	4.93E-05	2.36E-04	4.45E-05
1.53E-04	1.83E-04	2.45E-05	2.11E-05
2.95E-06	1.05E-06	1.42E-07	1.20E-05
4.04E-09	1.05E-06	1.70E-07	1.63E-08
1.04E-06	1.71E-07	2.62E-07	1.93E-06
7.33E-06	2.86E-07	7.76E-07	2.68E-06
7.38E-10	8.08E-08	2.53E-07	4.65E-07
1.11E-05	1.56E-06	3.57E-08	7.86E-08
6.60E-07	1.00E-05	1.64E-07	4.70E-05
1.19E-05	4.76E-05		

Figure 7: Spam: mean value features of Σ_1

7.55E-04	2.34E-03	2.02E-03	5.02E-06
1.84E-03	4.60E-04	9.63E-05	3.68E-04
3.94E-04	1.69E-03	1.97E-04	5.43E-03
6.14E-04	4.45E-04	8.65E-05	7.72E-04
4.77E-04	9.77E-04	1.28E-02	9.09E-05
4.38E-03	4.95E-04	5.93E-05	1.47E-04
8.73E-03	4.36E-03	1.27E-02	1.94E-03
1.68E-03	1.73E-03	1.05E-03	7.86E-04
1.41E-03	7.91E-04	1.69E-03	1.41E-03
1.89E-03	2.05E-04	1.15E-03	8.60E-04
6.91E-04	2.19E-03	6.64E-04	1.26E-03
4.26E-03	2.98E-03	7.29E-05	5.00E-04
5.26E-04	1.59E-03	2.36E-04	1.19E-03
1.10E-04	2.31E-04		

Figure 8: Class spam: mean value features of μ_0

1.59E-03	1.69E-03	4.05E-03	1.48E-03
5.20E-03	1.74E-03	2.73E-03	2.13E-03
1.70E-03	3.41E-03	1.23E-03	5.57E-03
1.53E-03	8.80E-04	1.03E-03	4.93E-03
3.03E-03	3.25E-03	2.28E-02	2.00E-03
1.41E-02	2.19E-03	2.47E-03	1.96E-03
1.98E-04	1.01E-04	1.67E-05	2.12E-04
4.50E-06	5.86E-05	1.70E-05	3.46E-06
1.25E-04	2.25E-05	6.00E-05	2.56E-04
4.60E-04	4.83E-05	1.06E-04	3.96E-04
7.37E-07	2.51E-05	8.11E-05	7.32E-05
1.29E-03	1.24E-04	1.20E-05	2.03E-05
1.84E-04	1.12E-03	7.78E-05	5.01E-03
1.72E-03	8.46E-04		

Figure 9: Class spam: mean value features of μ_1

However we know that if the assumptions of Naive Bayes are satisfied (Conditional Indipendent and Multivariate Guassian), then it should perform extremely well. To execute the following tests I used r.

To test if conditionally independence is satisfied I simply executed a few times the chi-square independence test that is a procedure for testing if two categorical variables are related in some population. The null hypothesis for a chi-square independence test is that two variables are independent.
Since the p-value is less than the significance level, we cannot accept the null hypothesis. Thus we conclude that there is indeed a relationship between the various words. We can assume that basically all the other pairs of words are not independent. Thus the conditionally independence condition is not satisfied.

The second condition to verify was the multivariate normality test. We assumed that $p(x|y)$ was a multivariate Gaussian. To test this I used Mardia's test. This function performs multivariate skewness and kurtosis tests at the same time and combinestest results for multivariate normality. If both tests indicates multivariate normality, then datafollows a multivariate normality distri-

```
        Pearson's Chi-squared test

data:  spam$word_freq_free and spam$word_freq_business
X-squared = 53829, df = 42720, p-value < 2.2e-16


        Pearson's Chi-squared test

data:  spam$word_freq_address and spam$word_freq_internet
X-squared = 28967, df = 19152, p-value < 2.2e-16


        Pearson's Chi-squared test

data:  spam$word_freq_money and spam$word_freq_receive
X-squared = 34396, df = 13068, p-value < 2.2e-16
```

Figure 10: chi-squared independence test

bution at the 0.05 significance level. We can see from these results that pvalues are essentially 0, meaning that the hypothesis of multinomial distribution for both the classes is rejected from Mardia's test.

```
> mvnSpam$multivariateNormality
             Test        Statistic p value Result
1 Mardia Skewness 4158941.59531139       0     NO
2 Mardia Kurtosis 4395.78574614606       0     NO
3             MVN              <NA>    <NA>     NO
>

> rvmHam$multivariateNormality
             Test        Statistic p value Result
1 Mardia Skewness  5851056.1507152       0     NO
2 Mardia Kurtosis 5712.64349995361       0     NO
3             MVN              <NA>    <NA>     NO
>
```

Figure 11: Mardia multivariate normality test

As we can see the pvalues of either the skewness and the kurtosis are basically 0 and from the result column we can see that the the multinomial distribution didn't pass the Mardia's test. As a result we can conclude that both the assumptions for Naive Bayes are not satisfied. And this explains why it doesn't perform too well with the given dataset.

7 Final thoughts and brief comparison

For this assignment, we were asked to compare essentially two types of classifiers: SVM (with linear, rbf and polynomial kernels) and Naive Bayes. As I said previously SVM is a discriminative type of classifier while Naive Bayes is a generative one. In general, a discriminative model models the decision boundary between the classes while a generative model explicitly models the actual distribution of each class.
A Generative Model learns the joint probability distribution p(x,y) and it predicts the conditional probability with the help of Bayes Theorem.
A Discriminative model learns the conditional probability distribution p(y|x).
Discriminative and generative strategies can be adopted and considired in base of the given problem since each one has its own advantages and disadvantages.

Overall the biggest difference between the two kind of classifiers is that the Naive Bayes assumes that the features are independent, while SVMs look at the interactions between the features up to a certain degree, without providing a model of how points are actually generated or how are features related. The biggest weakness of Naive Bayes is indeed the assumption that makes it very simple such as the attributes are independent of each other; however, when the attributes are dependent, the large number of incorrect classifications occurs. As we can see, using a generative classifier is more complicated since is not limited on learning a function, but it requires non simple assumptions that need to be verified.
When the assumptions of Naive Bayes are not satisfied SVM tends to perform much better than the Naive Bayes classifier. However with small data set discriminative models might produce false results, since data can be influenced by noise.
In practice, discriminative classifiers could be better than generative classifiers in presence of a large set of data that does not preserve the required assumptions.

Another comparison is that Naive Bayes Classifier and Support Vector Machine have different options including the choice of kernel function for each as we have seen. They also are both sensitive to parameter optimization (i.e. different parameter selection can significantly change their output). So, if we had a result showing that Naive Bayes is performing better than SVM this is can be true only for the selected parameters. However, for another parameter selection, we might find SVM performing better. This was not the task of this assignment, but in real applications it should be taken under consideration.

Finally we have seen that SVMs with the given kernels initially didn't perform very well but after the normalization of the data, we had a performance boost specifically with the linear kernel and the rbf. The polynomial of second degree however didn't perform too well for this operation. Finally Naive Bayes did perform good enought, but still not as much as using SVMs, specifically because the assumptions of Naive Bayes were not satisfied. The best one for this task resulted to be the linear kernel.