Università
Ca'Foscari
Venezia

ARTIFICIAL INTELLIGENCE: KNOWLEDGE REPRESENTATION AND PLANNING

CM 0472-1

ASSIGNMENT 2

Student:

Giosuè Zannini 873810

Academic year 2021/2022

Contents

1 Introduction **1**

1.1 Problem 1

1.2 Idea of implementation 2

2 Support Vector Machine **4**

2.1 Description of method 4

2.2 Build angular kernels 10

2.3 Advantages and Disadvantages 12

2.4 Performance 12

2.4.1 Linear kernel 13

2.4.2 Polynomial kernel 15

2.4.3 RBF kernel 17

3 Naive Bayes **19**

3.1 Description of method 19

3.2 Advantages and Disadvantages 20

3.3 Performance 21

4 K-Nearest Neighbors **23**

4.1 Description of method 23

4.2 Advantages and Disadvantages 25

4.3 Performance 26

5 Conclusions **28**

Chapter 1

Introduction

1.1 Problem

In this assignment I have to write a spam filter using discriminative and generative classifiers. Using 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. About dataset the first 54 features correspond to word/symbols frequencies, I ignored features 55-57 and feature 58 is the class label (1 spam/0 ham). I had to implement and compare the following methods over the TF/IDF representation:

- SVM classification using linear, polynomial of degree 2, and RBF kernels.
- Naive Bayes classifier for continuous inputs, modelling each feature with a Gaussian distribution, resulting in the following model:

$$\begin{aligned} &\mathbb{P}(y=k) = a_k \\ &\mathbb{P}(x|y=k) = \prod_{i=1}^{D} \left[(2\pi\sigma_{ki}^2)^{-\frac{1}{2}} \exp^{-\frac{1}{2\sigma_{ki}^2}(x_i-\mu_{ki})^2} \right] \end{aligned} \tag{1.1}$$

- K-NN classifier.

In addition to these implementations I had to answer the following question which is answered in section 2.2 :

- Can you transform the kernels to make use of angular information only (i.e., no length)? Are they still positive definite kernels?

1.2 Idea of implementation

In this assignment I decided first to carry out the TF/IDF transformation and after that I took a look at the dataset by printing a summary, displaying the distribution of each feature available and verifying the correlation between them. After this first phase I wanted to make a consideration, as trying one of the required methods I noticed from the confusion matrix a high number of false positives (a non spam mail is identified as spam).

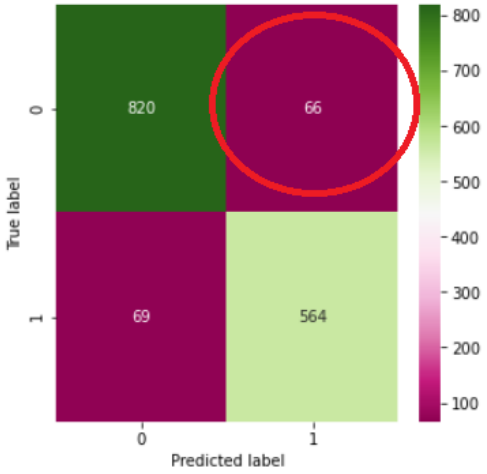


FIGURE 1.1: Confusion matrix of method K-NN with average between F_1 and accuracy

I found this number too high as having a false positive is more risky than a false negative as the average user goes to free the spam folder without paying too much attention to the content, and this could lead to a loss of important mails. To evaluate the quality of the algorithms I decided to make a tuning of the parameters using the 10 way cross validation and I chose the parameters that went to maximize the average between the accuracy and F-score with $\beta = 0.2$, in this way I have given more emphasis to the precision rather than the

recall in order to reduce the false positives but maintaining in consideration the accuracy.

The general formulas of F-score and accuracy are:

$$F_\beta = \frac{(1+\beta^2)\cdot TP}{(1+\beta^2)\cdot TP + \beta^2 \cdot FN + FP} \tag{1.2}$$

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.3}$$

Chapter 2

Support Vector Machine

2.1 Description of method

A Support Vector Machine (SVM) is a supervised machine learning algorithm and discriminative classifier. It can be defined as systems which use hypothesis space of a linear functions in a high dimensional feature space.

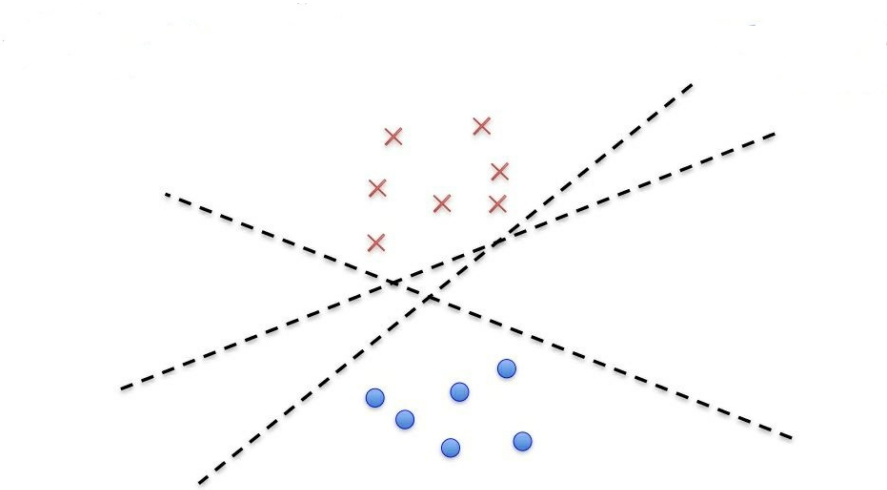


FIGURE 2.1: Example of possible lines separator

From above illustration, there are many linear classifiers (hyper-planes) that separate the data, however only one of these achieves maximum separation. The reason I need it is because if I use a hyper plane to classify, it might end up closer to one set of datasets compared to others and I do not want this to happen

and thus you see that the concept of maximum margin classifier or hyper plane as an apparent solution.

Here I assume the data is linearly separable, meaning that I can draw a line on a graph of x_1 vs x_2 separating the two classes when $D = 2$ and a hyperplane on graphs of $x_1, x_2, \ldots, x_D$ for when $D > 2$. This hyperplane can be described by $wx + b = 0$ where:

- w is normal to the hyperplane.
- $\frac{b}{||w||}$ is the perpendicular distance from the hyperplane to the origin.

Support Vectors are the examples closest to the separating hyperplane and the aim of it is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes.

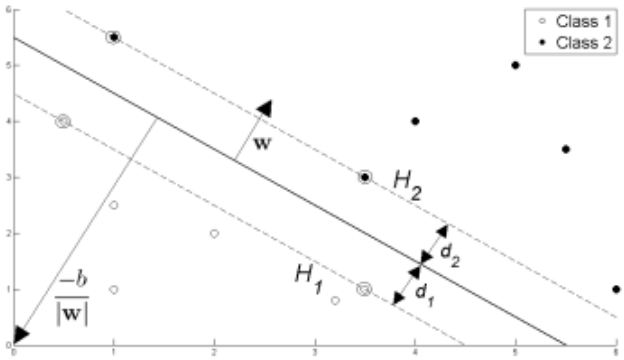


FIGURE 2.2: Example of hyperplane through two linearly separable classes

Referring to Figure 2.2, implementing a SVM boils down to selecting the variables w and b so that a training data can be described by:

$$\begin{aligned} x_i w + b &\geq 1 \quad \text{for} \quad y_i = 1 \\ x_i w + b &\leq -1 \quad \text{for} \quad y_i = -1 \end{aligned} \tag{2.1}$$

These equations can be combined into:

$$y_i(x_i w + b) - 1 \geq 0 \quad \forall_i \tag{2.2}$$

If I now just consider the points that lie closest to the separating hyperplane, then the two planes H_1 and H_2 that these points lie on can be described by:

$$\begin{aligned} x_i w + b &= 1 \quad \text{for} \quad H_1 \\ x_i w + b &= -1 \quad \text{for} \quad H_2 \end{aligned} \tag{2.3}$$

Referring to Figure 2.2, I define d_1 as being the distance from H_1 to the hyperplane and d_2 from H_2 to it. The hyperplane's equidistant from H_1 and H_2 means that $d1 = d2 - a$ quantity known as the SVM's margin. In order to orientate the hyperplane to be as far from the Support Vectors as possible,I need to maximize this margin. Simple vector geometry shows that the margin is equal to $\frac{1}{||w||}$ and maximizing it subject to the constraint in (2.2) is equivalent to finding:

$$\min ||w|| \quad \text{such that} \quad y_i(x_i w + b) - 1 \geq 0 \quad \forall_i \tag{2.4}$$

Minimizing $||w||$ is equivalent to minimizing $\frac{1}{2}||w||^2$ and the use of this term makes it possible to perform Quadratic Programming (QP) optimization later on. I therefore need to find:

$$\min \frac{1}{2}||w||^2 \quad \text{s.t.} \quad y_i(x_i w + b) - 1 \geq 0 \ \forall_i \tag{2.5}$$

In order to cater for the constraints in this minimization, I need to allocate them Lagrange multipliers α, where $\alpha_i \geq 0 \ \ \forall_i$:

$$\begin{aligned} L_P &\equiv \frac{1}{2}||w||^2 - \alpha[y_i(x_i w + b) - 1 \ \ \forall_i] \\ &\equiv \frac{1}{2}||w||^2 - \sum_{i=1}^{L} \alpha_i[y_i(x_i w + b) - 1] \\ &\equiv \frac{1}{2}||w||^2 - \sum_{i=1}^{L} \alpha_i y_i(x_i w + b) + \sum_{i=1}^{L} \alpha_i \end{aligned} \tag{2.6}$$

I wish to find the w and b which minimizes, and the α which maximizes (2.6). I can do this by differentiating L_P with respect to w and b and setting the derivatives to zero:

$$\frac{\partial L_P}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{L} \alpha_i y_i x_i \tag{2.7}$$

$$\frac{\partial L_P}{\partial b} = 0 \Rightarrow \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{2.8}$$

Substituting (2.7) and (2.8) into (2.6) gives a new formulation which, being dependent on α, I need to maximize:

$$\begin{aligned} L_D &\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i x_j \ \ \text{s.t.} \ \ \alpha_i \geq 0 \ \forall_i, \ \ \sum_{i=1}^{L} \alpha_i y_i = 0 \\ &\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i H_{ij} \alpha_j \ \ \text{where} \ \ H_{ij} \equiv y_i y_j x_i x_j \\ &\equiv \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha \ \ \text{s.t.} \ \ \alpha_i \geq 0 \ \forall_i, \ \ \sum_{i=1}^{L} \alpha_i y_i = 0 \end{aligned} \tag{2.9}$$

This new formulation L_D is referred to as the Dual form of the Primary L_P. It is worth noting that the Dual form requires only the dot product of each input vector x_i to be calculated. Having moved from minimizing LP to maximizing L_D, I need to find:

$$\max_{\alpha} \left[\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \alpha^T H \alpha \right] \quad \text{s.t.} \quad \alpha_i \geq 0 \ \forall_i \quad \text{and} \quad \sum_{i=1}^{L} \alpha_i y_i = 0 \tag{2.10}$$

This is a convex quadratic optimization problem, and we run a QP solver which will return α and from (2.7) will give us w. What remains is to calculate b.
Any data point satisfying (2.8) which is a Support Vector x_s will have the form:

$$y_s(x_s w + b) = 1 \tag{2.11}$$

Substituting in (2.7):

$$y_s \left(\sum_{m \in S} \alpha_m y_m x S_m x_s + b \right) = 1 \tag{2.12}$$

Where S denotes the set of indices of the Support Vectors. S is determined by finding the indices i where $\alpha_i > 0$. Multiplying through by y_s and then using $y_s^2 = 1$ from (2.1):

$$\begin{aligned} y_s^2 \left(\sum_{m \in S} \alpha_m y_m x S_m x_s + b \right) &= y_s \\ b = y_s - \sum_{m \in S} \alpha_m y_m x S_m x_s & \end{aligned} \tag{2.13}$$

Instead of using an arbitrary Support Vector x_s, it is better to take an average over all of the Support Vectors in S:

$$b = \frac{1}{N_s} \sum_{s \in S} \left(y_s - \sum_{m \in S} \alpha_m y_m x_m x_s \right) \tag{2.14}$$

I now have the variables w and b that define our separating hyperplane's optimal orientation and hence our Support Vector Machine.
Until now I worked with the assumption that the space is linearly separable, but SVM allows a strategy called kernel trick for learning a possible separating

hyperplane in a new space. In some cases could be interesting or evaluating not the original space but a transformation of it, for instance when in the original space data points are not linearly separable. I define a function $\phi(x)$ as the function that applies a mapping of a feature vector to another one. The SVM algorithm now instead of considering the vector x learns using the resulting vector of $\phi(x)$. A kernel function is nothing else that an inner product between feature mappings of ϕ:

$$K(x, z) = \phi(x)^T \phi(z) \tag{2.15}$$

Since the SVM algorithm works with only inner products between input vectors (x, z), then we can replace the simple inner product with a kernel function in order to learns in a different feature space that in some cases could be more efficient. But there is a restriction on the function K, it must satisfy the following property in order to be considered a valid kernel:

$$K(x, z) = \phi(x)^T \phi(z) \quad \forall x, z \in S \tag{2.16}$$

Another interesting property is the positivity of a kernel, so a symmetric function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is said to be positive definite kernel if:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0 \quad c_i, c_j \in \mathbb{R} \tag{2.17}$$

In this assignment I have to deal with the following positive definite kernels:

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

2.2 Build angular kernels

The kernels defined until now (linear, polynomial and Gaussian RBF) are affected by the length of each vector, so that two vectors with the same direction (angle) but with different length results not so similar. Now if I want to consider only the direction of each vector is necessary to normalize vector such that at the end $||x||^2 = 1$. I know that a kernel is defined as:

$$\begin{aligned} \phi : \mathbb{R}^m \to \mathbb{R}^m \qquad \phi(x) = \frac{x}{||x||} \\ K(x_i, x_j) = \phi(x_i)^T \phi(x_j) = \frac{x_i}{||x_i||} \frac{x_j}{||x_j||} = \cos(x_i, x_j) \end{aligned} \tag{2.18}$$

I 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.
So I have seen that it is possible to consider only angular information, But now I have to check if kernels are still positive definite, and to verify it I have to prove that:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0 \quad c_i, c_j \in \mathbb{R} \tag{2.19}$$

so:

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(x_i, x_j) \quad \text{n number of points} \\
=&\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \langle \phi(x_i), \phi(x_j) \rangle \\
=&\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \sum_{k=1}^{m} \frac{x_{ik}}{||x_{ik}||}\frac{x_{jk}}{||x_{jk}||} \quad \text{m number of features} \\
=&\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} c_i \frac{x_{ik}}{||x_{ik}||} c_j \frac{x_{jk}}{||x_{jk}||} \\
=&\sum_{k=1}^{m}\left(\sum_{i=1}^{n} c_i \frac{x_{ik}}{||x_{ik}||}\right)\left(\sum_{j=1}^{n} c_j \frac{x_{jk}}{||x_{jk}||}\right) \\
=&\sum_{k=1}^{m}\left(\sum_{i=1}^{n} c_i \frac{x_{ik}}{||x_{ik}||}\right)^2 \geq 0
\end{aligned} \tag{2.20}$$

Regarding the other two kernels, radial basis and polynomial of degree p, I can write their kernel in this way:

$$\begin{aligned}
K_p(x_i, x_j) &= \phi_p(x_i)^T \phi_p(x_j) \\
K_{rb}(x_i, x_j) &= \phi_{rb}(x_i)^T \phi_{rb}(x_j)
\end{aligned} \tag{2.21}$$

From the theory I know that these kernels are positive definite. Now to apply these two kernels considering only the angular information, I have to update them in this way:

$$\begin{aligned}
\phi_p^{'}(x) &= \phi_p(\frac{x}{||x||}) \\
K_p^{'}(x_i, x_j) = \phi_p^{'}(x_i)^T \phi_p^{'}(x_j) &= \phi_p(\frac{x_i}{||x_i||})^T \phi_p(\frac{x_j}{||x_j||}) = K_p(\frac{x_i}{||x_i||}, \frac{x_j}{||x_j||}) \geq 0
\end{aligned} \tag{2.22}$$

$$\phi'_{bf}(x) = \phi_{bf}(\frac{x}{||x||})$$

$$K'_{bf}(x_i, x_j) = \phi'_{bf}(x_i)^T \phi'_{bf}(x_j) = \phi_{bf}(\frac{x_i}{||x_i||})^T \phi_{bf}(\frac{x_j}{||x_j||}) = K_{bf}(\frac{x_i}{||x_i||}, \frac{x_j}{||x_j||}) \geq 0 \tag{2.23}$$

Since K_p and K_{bf} are positive definite kernel so the property is satisfied also for the two new kernels: K'_p, K'_{bf}. With these proves I have seen that the three new kernel are valid positive definite kernels.

2.3 Advantages and Disadvantages

This algorithm has a lot of advantages and disadvantages, such as:

- Advantages:
  1. The algorithm works relatively well when there is a clear margin of separation between classes.
  2. SVM is relatively memory efficient.
- Disadvantages:
  1. This algorithm is not suitable for large data sets.
  2. It does not perform very well when the data set has more noise.

2.4 Performance

For this algorithm I decided to make three different section, one for each kernel.

2.4.1 Linear kernel

For this algorithm I decided to perform the tuning of the parameter C which controls the influence of each individual support vector. Large C values give solutions with less erroneous classification errors but a smaller margin, instead small C values give solutions with greater margin and more classification errors.

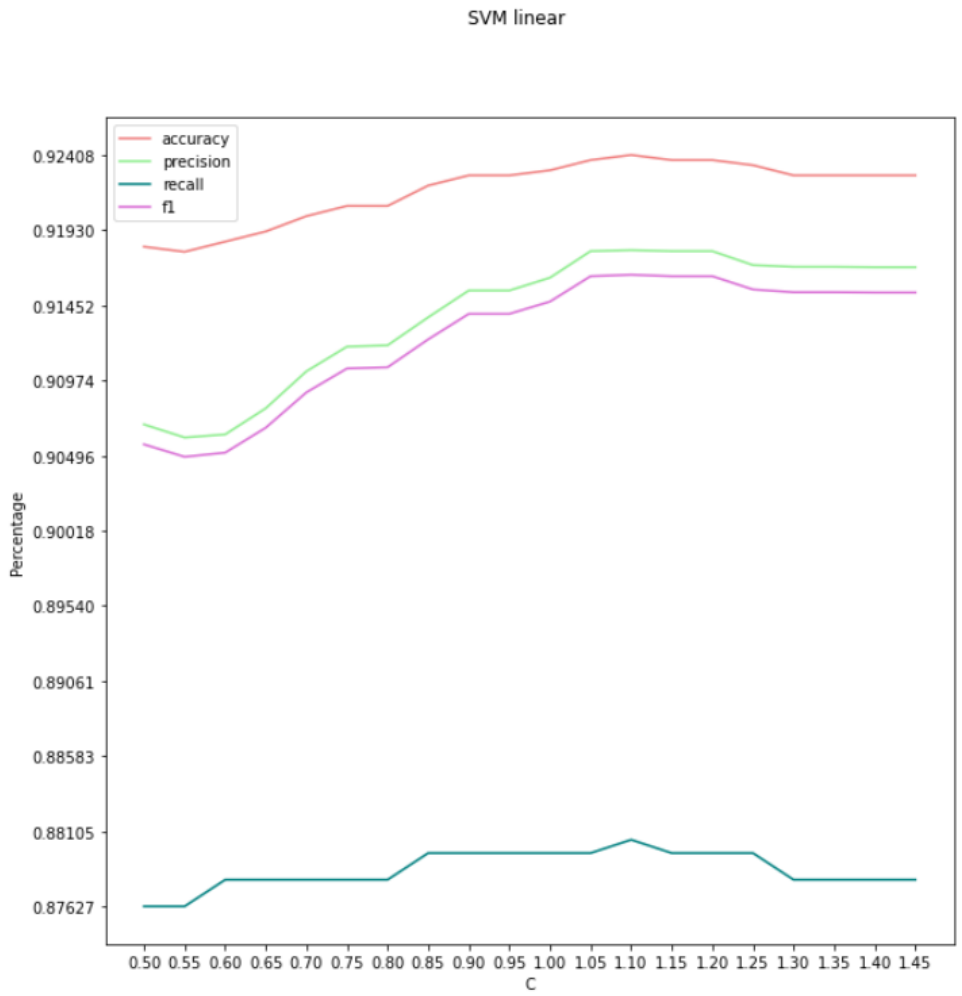


FIGURE 2.3: Tuning of SVM with linear kernel

From this graph I can see that the best hyper-parameter is the value 1.1. Thanks to it I got the following results in the test set:

Accuracy	Precision	Recall	F_1-score
0.918	0.923	0.877	0.899

And the associated confusion matrix is:

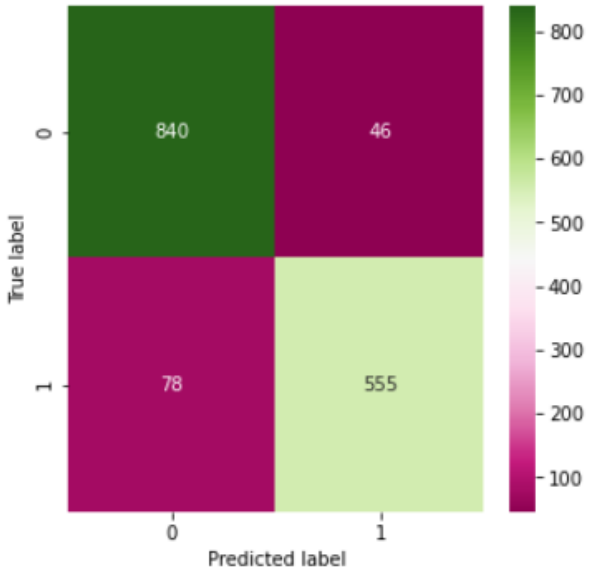
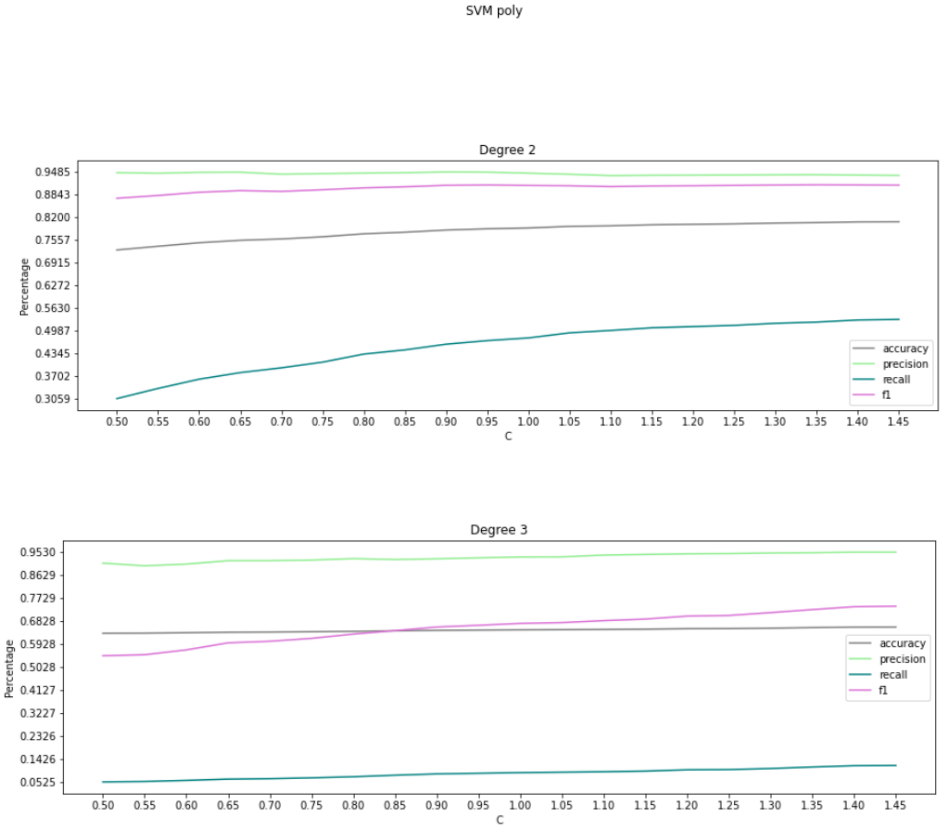


FIGURE 2.4: Test set confusion matrix of SVM with linear kernel

From the matrix note that I have few false negatives and few false positives. In fact this is reflected in the values of precision and recall.

2.4.2 Polynomial kernel

For this algorithm I decided to perform the tuning of the parameter C as before and the parameter degree whose indicates the degree of polynomial. The degree of the polynomial kernel control the flexibility of the resulting classifier. The lowest degree polynomial is the linear kernel, which is not sufficient when a nonlinear relationship between features exists.



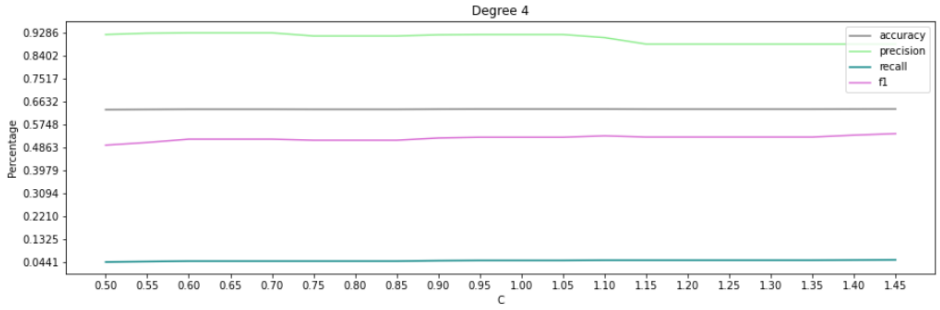


FIGURE 2.5: Tuning of SVM with polynomial kernel

From this graph I can see that the best hyper-parameter are $C = 1.4$ and degree equal 2. Thanks to them I got the following results in the test set:

Accuracy	Precision	Recall	F_1-score
0.789	0.944	0.528	0.677

And the associated confusion matrix is:

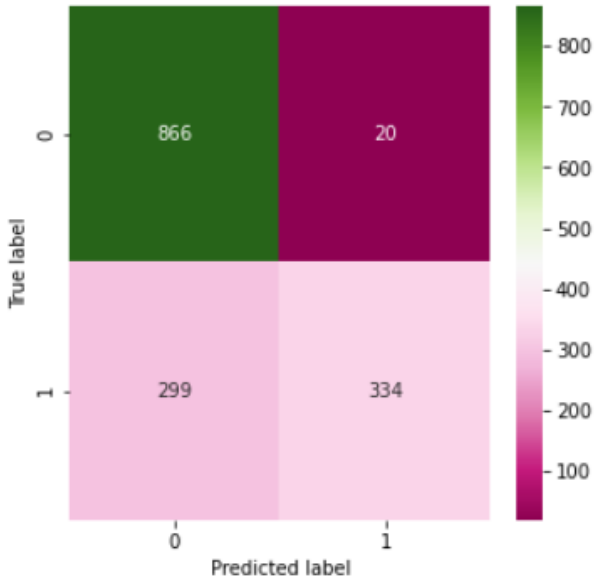


FIGURE 2.6: Test set confusion matrix of SVM with polynomial kernel

From the matrix note that I have very few false positives, but at the same time I have many false negatives. In fact the recall value is about 50%.

2.4.3 RBF kernel

For this algorithm I decided to perform the tuning of the parameter C as in the Linear kernel.

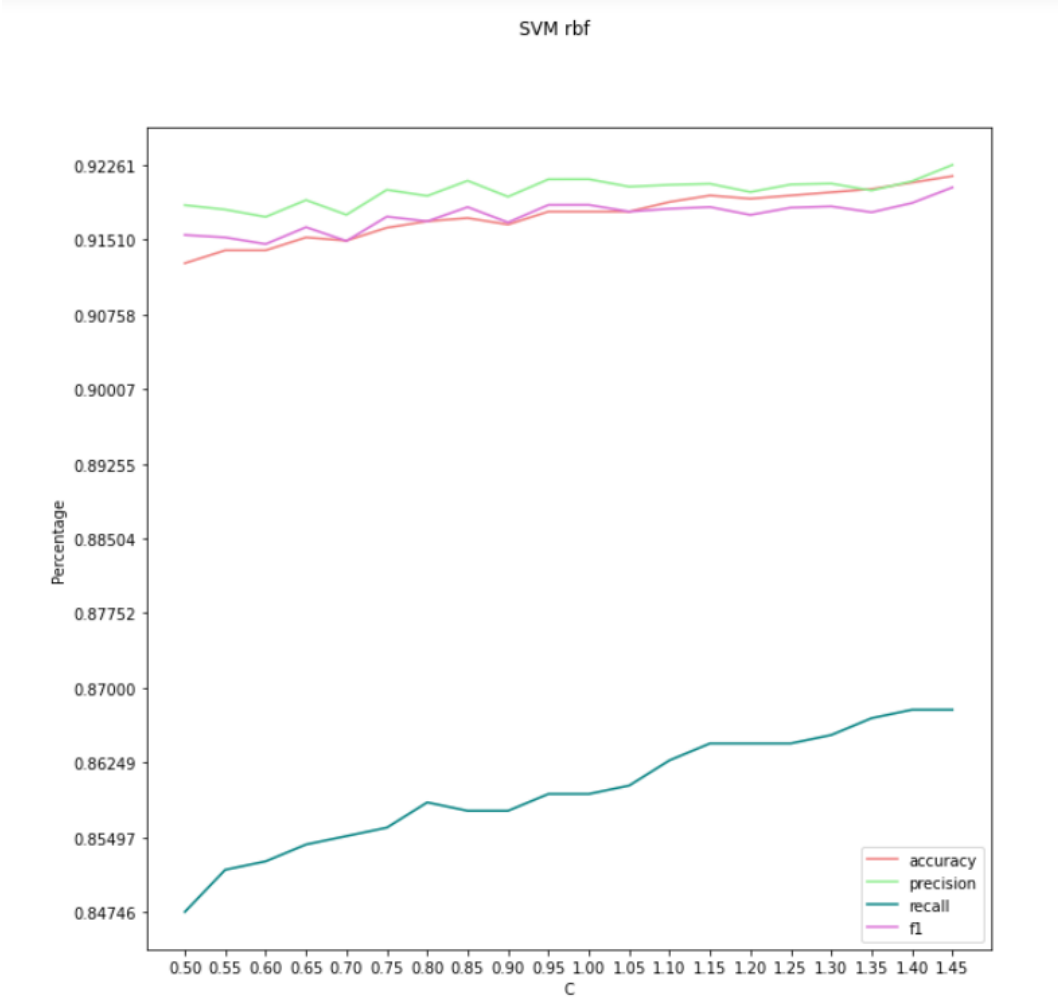


FIGURE 2.7: Tuning of SVM with RBF kernel

From this graph I can see that the best hyper-parameter is the value 1.45. Thanks to it I got the following results in the test set:

Accuracy	Precision	Recall	F_1-score
0.924	0.933	0.882	0.907

And the associated confusion matrix is:

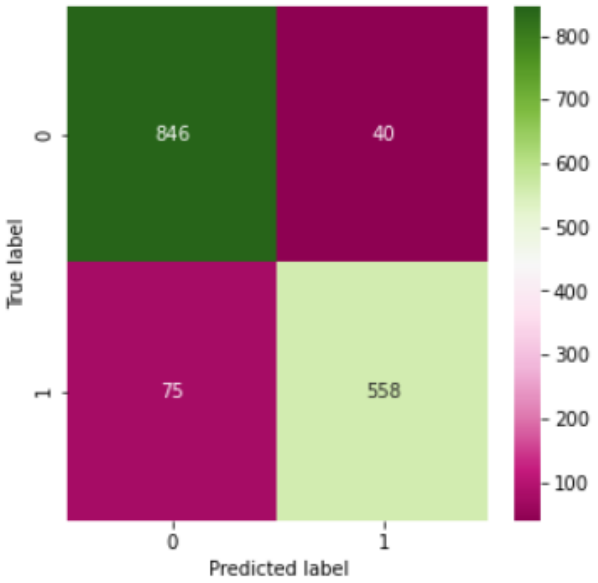


FIGURE 2.8: Test set confusion matrix of SVM with RBF kernel

From the matrix I know that I have good results in general, as I have high values regarding true positive and negative and low results for both false.

Chapter 3

Naive Bayes

3.1 Description of method

A Naive Bayes classifier is a probabilistic machine learning and generative model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

$$\begin{aligned} \mathbb{P}(y = k|X) &= \frac{\mathbb{P}(X|y = k)\mathbb{P}(y = k)}{\mathbb{P}(X)} \\ X &= (x_1, x_2, x_3, \ldots, x_n) \end{aligned} \tag{3.1}$$

Using Bayes theorem, I can find the probability of y happening, given that X has occurred. Here, X is a vector of features and $y = k$ is the hypothesis. The assumption made here is that the features are independent, that is presence of one particular feature does not affect the other. Hence it is called naive. With this approach the denominator remains static, therefore it can be removed and a proportionality can be introduced.

$$\mathbb{P}(y = k|x_1, \ldots, x_n) \propto \mathbb{P}(y = k) \prod_{i=1}^{n} \mathbb{P}(x_i|y = k) \tag{3.2}$$

So I need to find the class y with maximum probability

$$\hat{y} = \arg\max_k \mathbb{P}(y = k) \prod_{i=1}^{n} \mathbb{P}(x_i|y = k) \tag{3.3}$$

For this assignment I have to conclude that each feature leads a Gaussian distribution, so the conditional probability changes to:

$$\mathbb{P}(x|y=k) = \prod_{i=1}^{D} \left[\frac{1}{\sqrt{2\pi\sigma_{ki}^2}} \exp^{-\frac{(x_i-\mu_{ki})^2}{2\sigma_{ki}^2}} \right] \tag{3.4}$$

3.2 Advantages and Disadvantages

This algorithm has a lot of advantages and disadvantages, such as:

- Advantages:
    1. The algorithm is easy and fast to predict class of test data set. It also perform well in multi class prediction.
    2. When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- Disadvantages:
    1. This algorithm is also notorious as a lousy estimator. So, you shouldn't take the probability outputs of 'predict_proba' too seriously.
    2. It assumes that all the features are independent. While it might sound great in theory, in real life, you'll hardly find a set of independent features.

3.3 Performance

For this algorithm I decided to perform the tuning of the parameter regarding the variability of variance which increases the width of the curve of the Gaussian function for each feature, thus decreasing the probability of values close to the average and increasing the probability in the queues of the distribution.

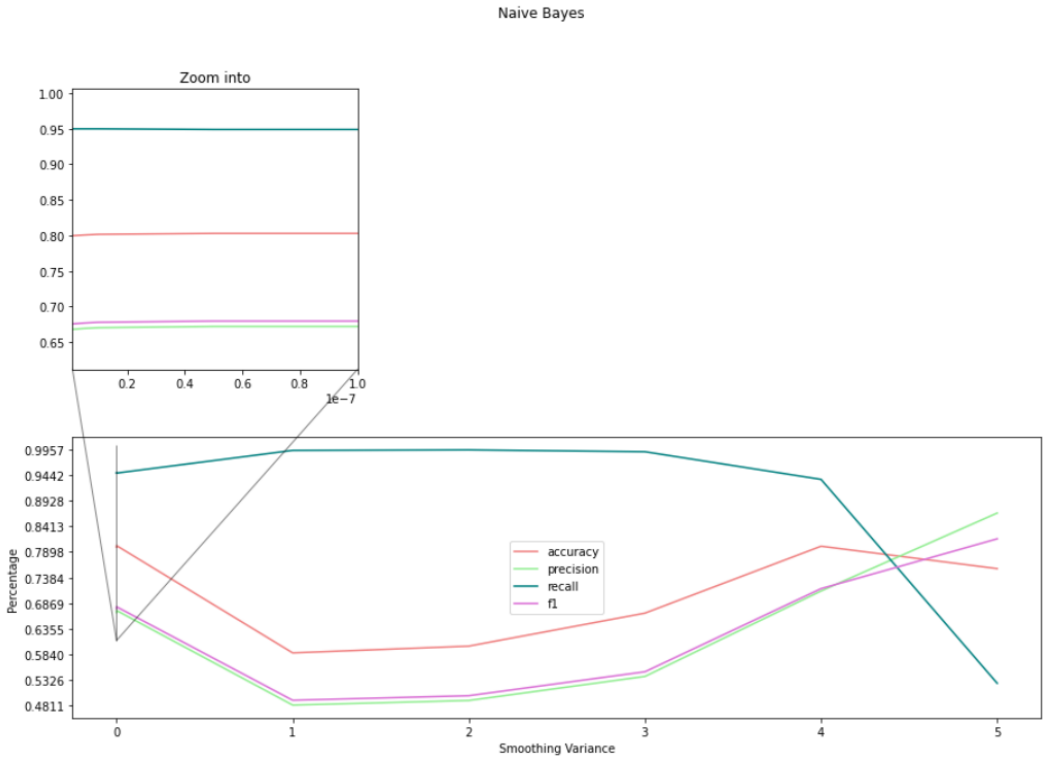


FIGURE 3.1: Tuning of Naive Bayes

From this graph I can see that the best hyper-parameter is the value 5. Thanks to it I got the following results in the test set:

Accuracy	Precision	Recall	F_1-score
0.785	0.927	0.524	0.670

And the associated confusion matrix is:

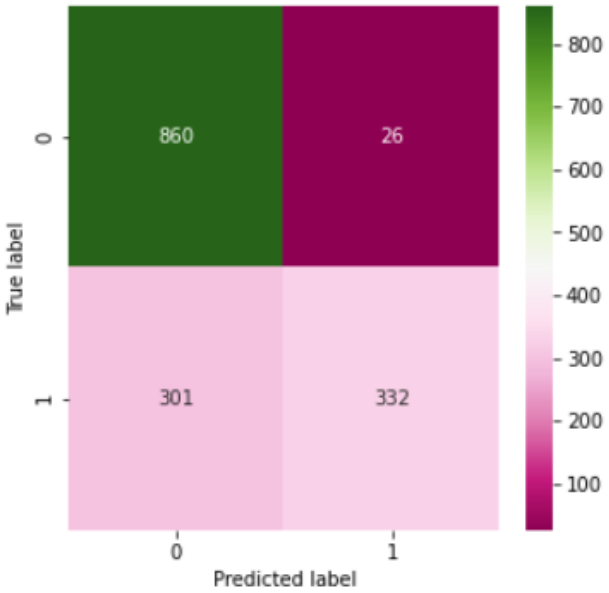


FIGURE 3.2: Test set confusion matrix of Naive Bayes

From the matrix note that I have few false positives but at the same time the true positives are too low. Also the false negative is very high and therefore there will be a lot of spam in the main box, indeed the recall has a very low value.

Chapter 4

K-Nearest Neighbors

4.1 Description of method

KNN is supervised machine learning algorithm and discriminative classifier also known as lazy learning algorithm which can be used for both classification and regression problems. It assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

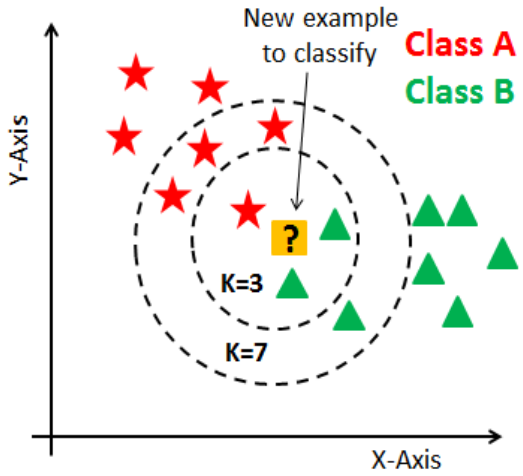


FIGURE 4.1: Example of KNN with different hyper parameter k

In this assignment I used the following distance:

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}} \tag{4.1}$$

It is called Minkowski distance and it can be applied only in a normed vector space. The p value in the formula can be manipulated to give different distances like:

- $p = 1$, is the Manhattan distance. This distance is also known as taxicab distance or city block distance, that is because the way this distance is calculated. The distance between two points is the sum of the absolute differences of their Cartesian coordinates.

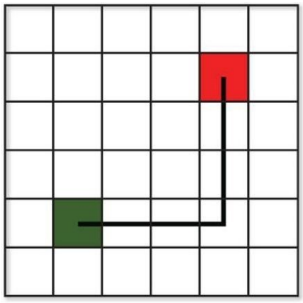


FIGURE 4.2: Example of Manhattan distance

- $p = 2$ This distance is the most widely used one. It is a measure of the true straight line distance between two points in Euclidean space.

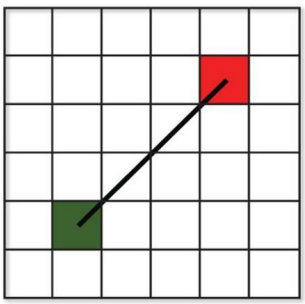


FIGURE 4.3: Example of Euclidean distance

4.2 Advantages and Disadvantages

This algorithm has a lot of advantages and disadvantages, such as:

- Advantages:
  1. The algorithm is simple and easy to implement.
  2. There's no need to build a model, tune several parameters, or make additional assumptions.
  3. The algorithm is versatile. It can be used both for classification and regression problems.
- Disadvantages:
  1. KNN might be very easy to implement but as dataset grows efficiency or speed of algorithm declines very fast.
  2. KNN works well with small number of input variables but as the numbers of variables grow this algorithm struggles to predict the output of new data point.
  3. One of the biggest issues is to choose the optimal number of neighbors (k) to be consider while classifying the new data entry.
  4. KNN algorithm is very sensitive to outliers as it simply chose the neighbors based on distance criteria.

4.3 Performance

For this algorithm I decided to perform the tuning of the parameters regarding the k and the metric to be used. The choice of k is crucial because with a k too small you are subject to outliers, instead with a k too high the response of the classifier will always be the same because it will always take the label more frequently within the dataset.

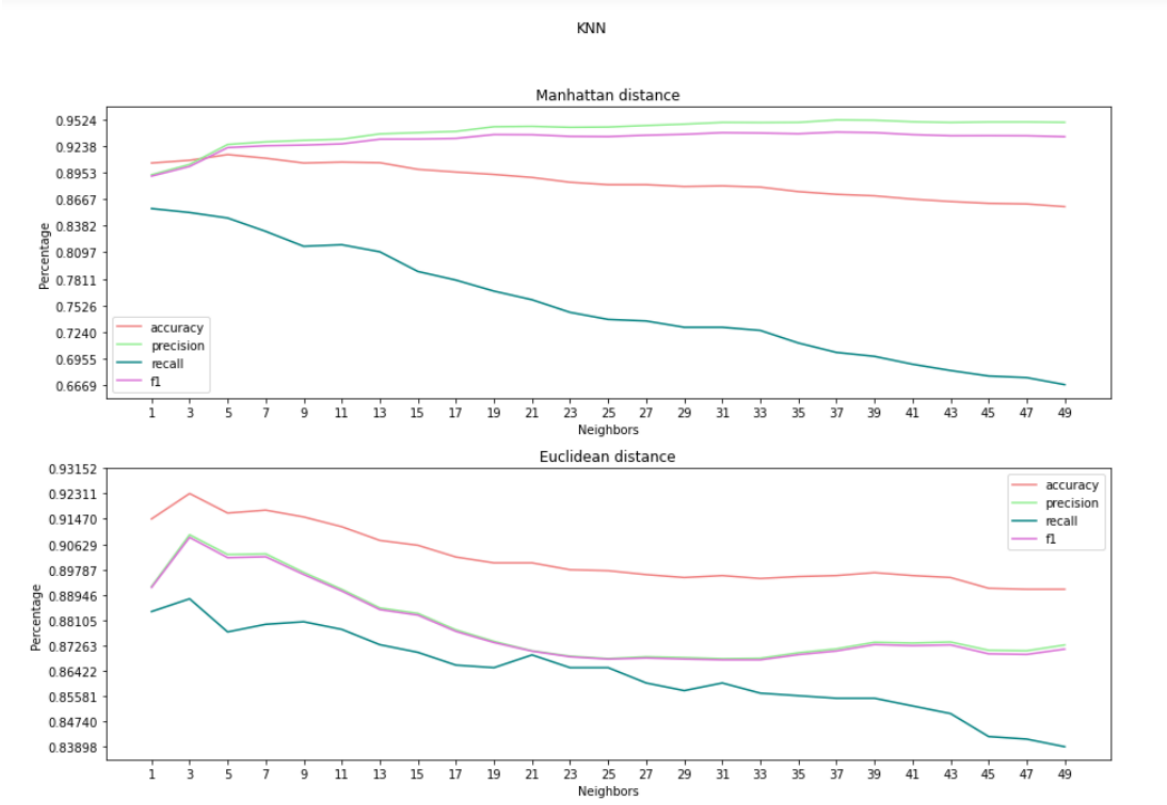


FIGURE 4.4: Tuning of K-NN

From this graph I can see that the best hyper-parameters are $k = 13$ and as metric is Manhattan distance. Thanks to them I got the following results in the test set:

Accuracy	Precision	Recall	F_1-score
0.903	0.948	0.812	0.874

And the associated confusion matrix is:

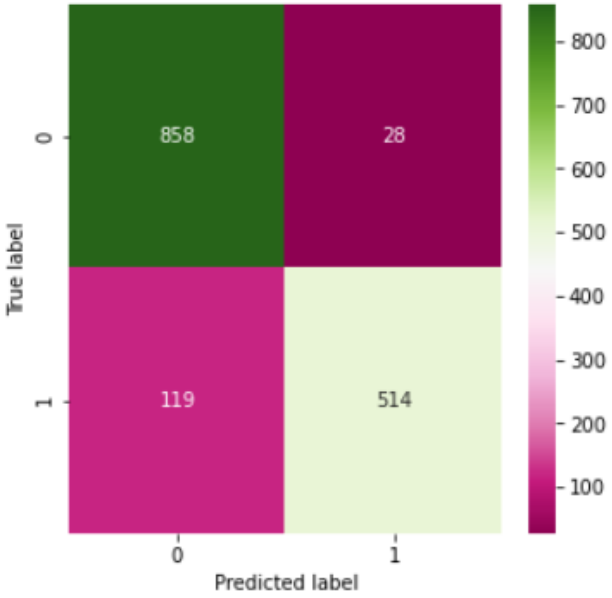


FIGURE 4.5: Test set confusion matrix of K-NN

From the matrix known that I have good results and false positives are drastically decreased compared to the matrix of confusion that can be noted in Chapter 1.2.

Chapter 5

Conclusions

In this assignment I had to compare three types of classifiers: K-NN, SVM and Naive Bayes; where the first two are discriminative classifiers and the last is generative classifier. Discriminative classifiers focus on learning a decision boundary $h(x)$ which separates as better as possible the considered class. They allow me to classify points, without providing a model of how the points are actually generated or how features are related to the correspondent class. Generative classifiers focus on learning the underlying data distribution, in particular try to learn the join probability $\mathbb{P}(X, Y)$ directly from data taking advantage of some possible assumptions. Both the two strategies are adopted depending on the dealing problem, since each of them has particular advantages and disadvantages that should be considered. As I could see the role of a generative classifier is more difficult, it is not limited on learning a function but probability distributions, and for this reason it requires some non simple assumptions. Surely two advantages are that knowing the joint probability distribution $\mathbb{P}(X, Y)$ it is possible to generate new possible pairs (X, Y), and that with them it is possible to identify possible outliers. Instead in discriminative classifier to find the probability, they directly assume some functional form for $\mathbb{P}(Y|X)$ and then estimate the parameters of $\mathbb{P}(Y|X)$ with the help of the training data. These type of classifiers separate classes instead of modeling the conditional probability and don't make any assumptions about the data points, in fact they are not capable of generating new data points. Therefore, the ultimate objective

of discriminative models is to separate one class form another. In summary a generative model focuses on explaining how the data was generated, while a discriminative model focuses on predicting the labels of the data. In mathematical terms, a discriminative machine learning trains a model which is done by learning parameters that maximize the conditional probability $\mathbb{P}(Y|X)$, while on the other hand, a generative model learns parameters by maximizing the joint probability of $\mathbb{P}(X, Y)$.

Now I want to give a bit of space about comparison among methods that I had to implements, First of all I want to ex plane the difference of time required for learning phase. SVM even if it considers only support vectors and not all the data points, requires more iterations to find the right parameters in order to find the best hyperplane. Instead the training phase of Naive Bayes requires less data and it is very simple since it is performed with a single iteration. And as last K-NN requires less time to train in fact it is even called lazy algorithm since all computational cost is shifted in prediction phase.

After that if the Naive Bayes assumptions are not satisfied, SVM and K-NN tend to outperform it as the size of the dataset increase. This result is given by the fact that dependency structure cannot be captured by the Naive Bayes classifier, instead discriminative classifiers are not affected by this structure and when the dataset increase the performance improve. Of course it can happen that with small data set discriminative models might consider false patterns, in fact data are influenced by the presence of noise. So discriminative classifiers could be better than generative classifiers in presence of a large set of data that does not preserve the required assumptions. In general a generative classifier works well and learn the underlying structure of the data if the model is right specified and assumptions are satisfied.

Indeed in this assignment I noticed that the discriminatory models tested work better than the generative model because they do not require any assumption,

in fact in Naive Bayes there are two strong assumptions: the independence between the features, and that all of them follow a Gaussian distribution. But as I can see these hypotheses are not real, in fact the performances of this method are not satisfied. In addition, the veracity of these two assumptions can be seen from the correlation matrix and feature histograms (visible overall within the .ipynb file) below.

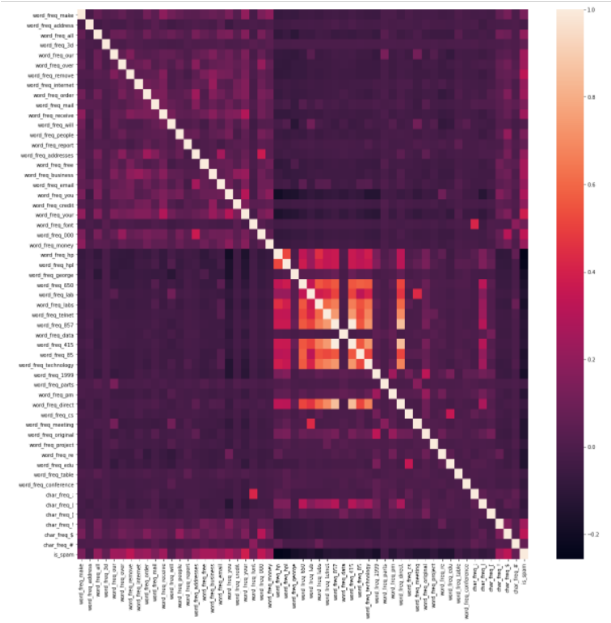


FIGURE 5.1: Correlation matrix among features

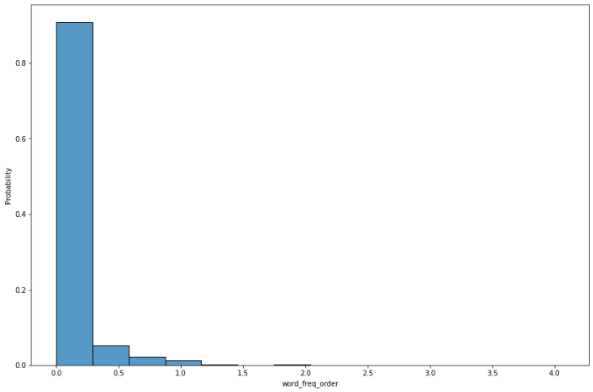


FIGURE 5.2: Example of probability distribution of the feature order

In the end the best method I used in this assignment is SVM with radial base function which has a very high performance.