INDIAN INSTITUTE OF
INFORMATION TECHNOLOGY, ALLAHABAD

Text Classification
Using
Support Vector Machine

Ayush Agnihotri	IIM2015004
Nidheesh Pandey	IIM2015501
Abhishek Pasi	ICM2015002
Shreyansh Gupta	IIM2015001
Vishal Kumar Singh	IIT2015141

UNDER SUPERVISION
of
DR. K P SINGH

DECLARATION

We hereby declare that the work presented in this project report entitled “ ***Text Classification Using Support Vector Machine*** ”, submitted as End Semester Report of 5th Semester B.Tech. (IT) at **Indian Institute of Information Technology, Allahabad**, is an authenticated record of our original work carried out from August 2017 to November 2017 under the guidance of **Dr. K P Singh**. Due acknowledgements have been made in the text to all other material used. The project was done in full compliance with the requirements and constraints of the prescribed curriculum.

Signature :

Date :

Contents

List of Figures

List of Tables

TEXT CLASSIFICATION
using
SUPPORT VECTOR MACHINE

Abstract

In this report, we explore Machine Learning and Natural Langauage Processing to categorize given set of text(article, e-mail,etc).We have used 80% of dataset (Reuters 21578) for training our model and 20% for texting. For training, dataset is parsed, cleaned (removing stop words, eliminating similar words by stemming) using various NLP techniques. The resultant words is vectorized into sparse matrix and then we use tf-idf technique to extract features for classification. Further, we reduce no.of features(which does not affect our decision making process) using singular value decomposition (SVD). At last we feed resultant data to Support Vector Machine(rbf kernel) for classification. A detailed analysis of accuracy and performance of model is done by varying various parameters of SVM (penalty parameter(C), kernel function, kernel coefficient).

Introduction

Document Classification is an important task in machine learnin g.Here, we categorize a text/document into a single or multiple classes/categories. By assigning categories to various documents (like customer reviews, emails, articles) we can use it in various application like spam detection, sentiment analysis, News categorisation, genre classification, etc. Through our model, we aim to categorize documents algorithmically using machine learning and NLP.We have trained our model with dataset (Reuters 21578)

Methodology

I. **Getting the dataset :**
The Reuters 21578 dataset is loaded and then sent for parsing to get it into a useble format.

II. **Parsing :**
In this step we convert the dataset into a usable format which can be classified for our experiment. This conversion process is known as Parsing. Here we have made SGML parser class which overrides HTMLParser.

III. **Stop Words Removal :**
Stop words are set of commonly used words in any language. It is important for us to filter these words inorder to focus on more important words. For example : a, an, is, it, that, the, with, from, has, were, was, its, of, be, will, with, and, etc.

IV. **Stemming :**

It is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form.[inflect (dictionary meaning) :- change the form of (a word) to express a particular grammatical function or attribute]

Example :-

- banks and banking become bank
- investing and invested become invest

V. **Vectorisation (Feature Extraction) :**

- It is used to convert raw text into a numerical data representation which can be used for classification.
- Firstly we create tokens from string & assign integer identifier to list of tokens, which allows them to be listed.
- After list formation, we get the count of tokens within document. We normalise these tokens to de-emphasise tokens that appear frequently within a document. This process is known as the **Bag Of Words (BoW)**.
- BoW allows a vector to be associated with each document, each component of which is real-valued and represents the importance of tokens (i.e. "words") appearing within that document.
- The entire dataset can be represented as a large matrix, each row representing one of the documents and each column representing token occurance in that document. This is the process of **vectorisation**.

VI. **tf–idf (Term-Frequency Inverse Document-Frequency) :**

TF-IDF is used here to get a better weighting scheme of the tokens used for classifying the documents.

The TF(term frequency) part increses the weight for a token with increase in frequency of word in the document but after that the IDF(inverse document frequency) part normalizes weight according to its frquency in the entire corpus. So the importance of the words which appear a lot in entire corpus reduces significantly than the words appearing a lot within a particular document.

We get a high TF-IDF value if its frequency is high in that document and low frequency in collection of all documents.

The formula for calculating TF-IDF value is as follows:-

$$\boxed{w_{i,j} = tf_{i,j}\ \mathbf{x}\ \log(\frac{N}{df_i})}$$

where:

$tf_{i,j}$ = number of occurences of i in j

df_i = number of documents containing i

$\mathbf{N}$ = total number of documents

VII. **Singular Value Decomposition (SVD) :**

We have used SVD here to reduce the number of features. SVD achieves this task by creating new features which are linear combination of the existing ones.
Singular value decomposition reduces a matrix of R rank to a matrix of K rank. It means that we can take a list of R unique vectors, and approximate them as a linear combination of K unique vectors.
It uses the TF-IDF matrix to do this.
Let A be a rank R matrix with m rows and n columns. SVD tells us :

$$\boxed{\mathrm{A}\ =\ \mathbf{U}\Sigma V^T}$$

where:
$\mathbf{U}$ is a square orthonormal m x r matrix. Its columns are the left singular vectors.

Σ is a diagonal r x r matrix with r positive values starting from the top left. These are singular values.

V^T is a square orthonormal r x n matrix. The rows are the right singular vectors.
In our context,
$\mathbf{U}$ = no. of documents * no. of new features

Σ = strength of new features in increasing order (positive)

V^T = old features * new features.

VIII. **Support Vector Machine(SVM):**

Support Vector Machine(SVM) is a supervised machine learning model which can be used for both classification (Support Vector Classification[SVC]) and regression (Support Vector Regression[SVR]).

SVM as a Classifier (SVC):

It is a discriminative and non-probabilistic classifier which can be used for classifying both the Linearly Separable Dataset and also the Non-Linearly Separable Dataset.

It partitions a feature space into different groups, which in our case means separating a collection of articles into different categories.

SVM achieves this by finding an optimal means of separating such groups based on their known categories. It takes the data points and outputs the decision boundary that best separates the categories.

This best decision boundary is the one that maximizes the margins between any two categories. In other words, the decision boundary whose distance to the nearest element of each category is the largest. This best decision boundary is called the **Maximum Margin Classifier** and the nearest points which help in finding this best decision boundary are called the **Support Vectors**. This is the reason that this model(SVM) is called Support Vector Machine.

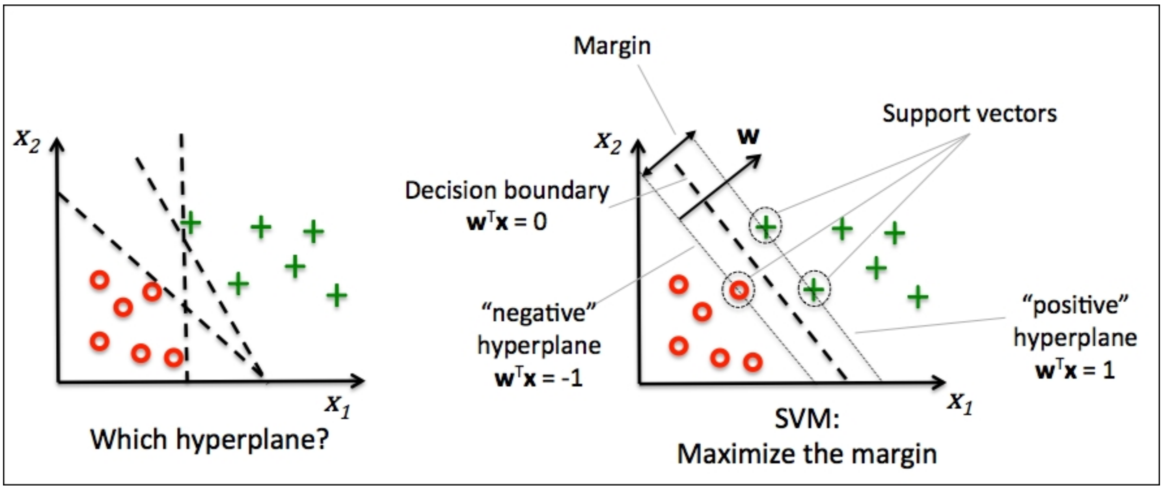


Fig - 1: *Maximum Margin Classifier and Support Vectors in SVM*

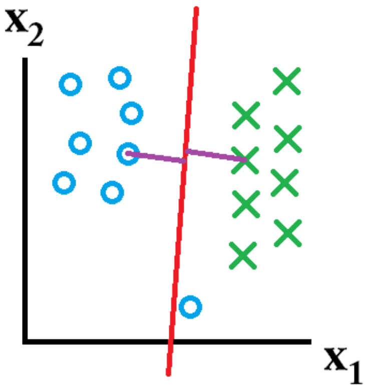


Fig - 2: *Outlier ignored in general SVM*

Only these **Support Vectors** contribute in finding the decision boundaries for the different categories, all other points/vectors don't contribute anything to the model.

SVM looks at the very extreme case for finding the best boundaries, that is why SVM is very special and very different than most of the other Machine Learning models.

Note that the general SVM does not include **outliers** inside the decision boundary of its category.

But we can change the parameters of SVM to change the way a decision boundary is selected and can find the parameters which are best suited for our application.

Linearly Separable Dataset:

Linearly Separable dataset can be easily classified using the above approach.

Non-Linearly Separable Dataset:

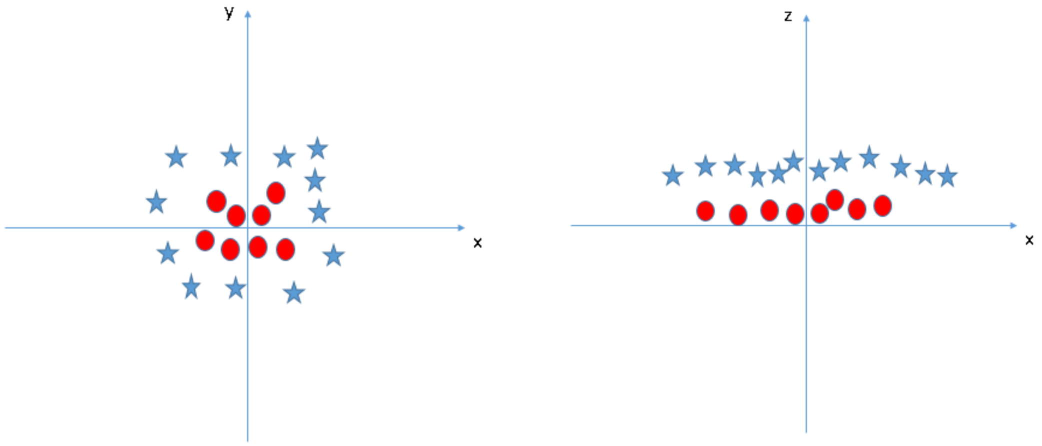
For Non-Linearly Separable dataset, we cannot just simply separate the different categories using a simple line or hyperplane.

We are required to take our non-linearly separable dataset, map it to a higher dimension to make it linearly separable, build decision boundaries using SVM and then project all of that back into our original dimensions.

But calculating the transformations to a higher dimensional space can be highly compute intensive and might require a lot of computation and processing power.

So, to overcome this, we use a trick known as the **kernel trick**:

SVM does not need the actual vectors to work on it, it can do it only with the **dot products** between them. This dot product is called a **kernel function**.



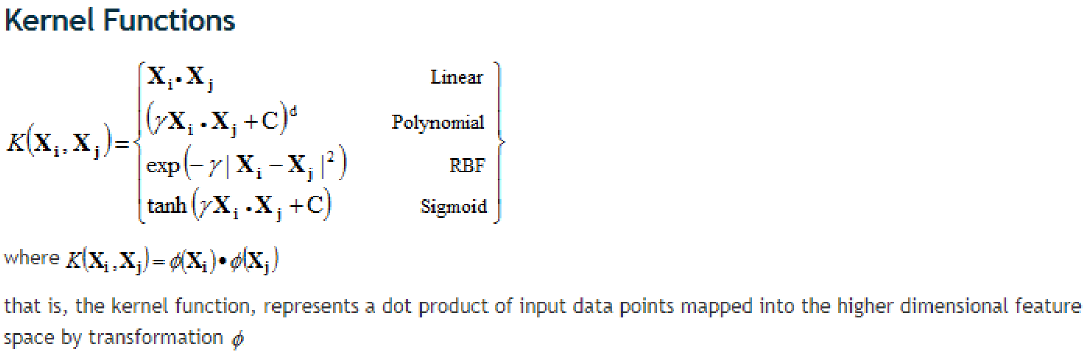
Non-linearly separable data

Mapping into higher dimension (adding z-axis component) Z = $x^2 + y^2$

This function will save us a lot of expensive calculations.

There are many kernel functions(kernels), some commonly used kernels are:

- Gaussian RBF kernel
- Linear kernel
- Sigmoid kernel
- Polynomial kernel



Kernel Functions

$$K(\mathbf{X}_i, \mathbf{X}_j) = \begin{cases} \mathbf{X}_i \cdot \mathbf{X}_j & \text{Linear} \\ (\gamma \mathbf{X}_i \cdot \mathbf{X}_j + \mathrm{C})^d & \text{Polynomial} \\ \exp(-\gamma \,|\, \mathbf{X}_i - \mathbf{X}_j \,|^2) & \text{RBF} \\ \tanh(\gamma \mathbf{X}_i \cdot \mathbf{X}_j + \mathrm{C}) & \text{Sigmoid} \end{cases}$$

where $K(\mathbf{X}_i, \mathbf{X}_j) = \phi(\mathbf{X}_i) \bullet \phi(\mathbf{X}_j)$

that is, the kernel function, represents a dot product of input data points mapped into the higher dimensional feature space by transformation ϕ

Fig - 3: *Different Types of Kernel functions*

Sources:

Fig 1: `https://www.packtpub.com/graphics/9781783555130/graphics/3547_03_07.jpg`

Fig 2: `https://stackoverflow.com/`

Fig 3: `https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_8.png`

Fig 4: https://www.analyticsvidhya.com/wp-content/uploads/2015/10/SVM_9.png

Fig 5: http://www.statsoft.com/textbook/support-vector-machines

Studying SVM parameters C and gamma

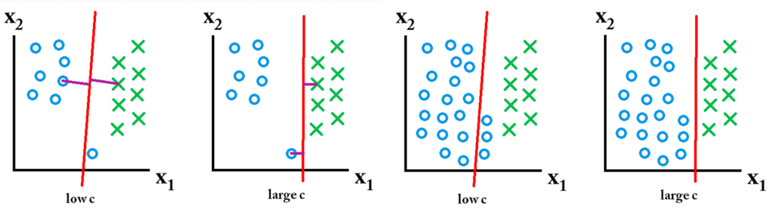
1. **The SVM penality parameter C :**

   In simple terms, a Support vector machine (for classification) mainly chooses a set of hyperplanes which classify the data into desired categories. SVM finds the maximum margin hyperplanes using the nearest points of different categories. However, perfect classification is present only in the ideal case, most of the time it is not possible to perfectly separate all points.

   In many cases we need to relax our maximum margin concept to classify most of the points correctly, these are known as soft margins. The role of penalty parameter of SVC class (sci-kit-learn python) becomes extremely important in such cases.

   Low C value means our separating hyperplane is close to maximum margin hyperplane. Low C is helpful in ignoring the outliers. However, sometimes low C values give us poor accuracy, In our experiments, C = 1 gave as low as 53.61% accuracy. Situations like these truly depend on the dataset, low C values provide room for diversity in a test set from a training set.In our case, it seems like data is under-fitted.

   High C value means our separating hyperplane is at some distance from maximum margin hyperplane. High C value provides great fitting on the training set. However, sometimes higher C values may lead to overfitting, In our experiments, it was found that increasing C value drastically increases accuracy and then it slowly reduces on any further increment. Observations made are a clear indication that higher C value leads to overfitting the training data.
   We changed values of the penalty parameter C and gamma (argument of an rbf kernel function) to observe the accuracy. It was observed that C values close to 1000 in case of an rbf kernel and 10-100 in case of a linear kernel. Note that these observations were only on a specific test data (generated randomly). Since, a relationship between the value of parameter C and Accuracy is highly dependent on the test set, we conducted More



(a) Training

(b) After testing

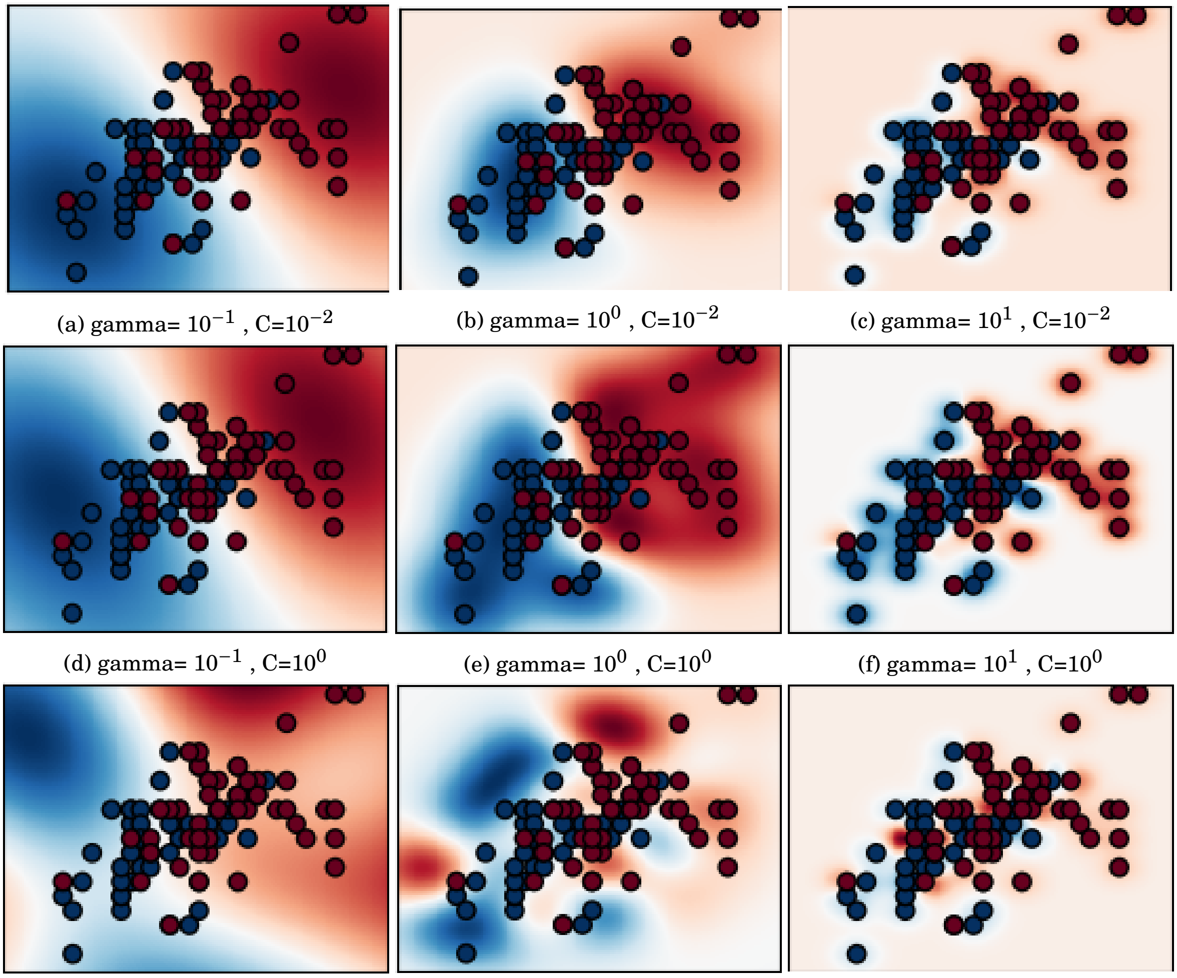
Fig - 4.4: low C and high C
Source: `https://stackoverflow.com/`

experiments (on many random test-train splits of our data) to determine the average accuracy of our model which will be discussed in subsequent sections of this project report.

2. **Importance of Gamma in case of non linear kernels:**

   Gamma parameter is important only in case of non-linear kernels. A Linear kernel is independent of the value of Gamma. We have verified this point by doing experiments and can be seen in our plots for a linear kernel.Intuitively, high gamma makes the separator more non-linear and is better suited for a case when training data is nonlinearly separable.

   Gamma parameter determines the role of a single support vector in determining the category of a point. High gamma value means a support vector's center of influence is small and low gamma values imply big center of influence.



(a) gamma= 10^{-1} , C=10^{-2}

(b) gamma= 10^{0} , C=10^{-2}

(c) gamma= 10^{1} , C=10^{-2}

(d) gamma= 10^{-1} , C=10^{0}

(e) gamma= 10^{0} , C=10^{0}

(f) gamma= 10^{1} , C=10^{0}

(g) gamma= 10^{-1} , C=10^{2}

(h) gamma= 10^{0} , C=10^{2}

(i) gamma= 10^{1} , C=10^{2}

Fig - 4.5: Gamma and C
Source : Sci-kit learn documentation

Quoting Scikit-learn documentation - " *The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors*"

It is observed that when gamma is very high the center of influence is diminished to the support vector itself.No value of C can be found which will reduce the overfitting of this model.If the test set has different characteristics than the training set then this model produces very bad accuracy. On the other hand, if a value of gamma is chosen to be low, model fails to capture the unique details of data in case of non-linear kernels.

C and Gamma

From the above findings, it is clear the value of C and Gamma depends on our dataset.

The image above from sci-kit learn documentation depicts the combined effect of C and Gamma on a simple classification problem.

Methodology Flow Diagram

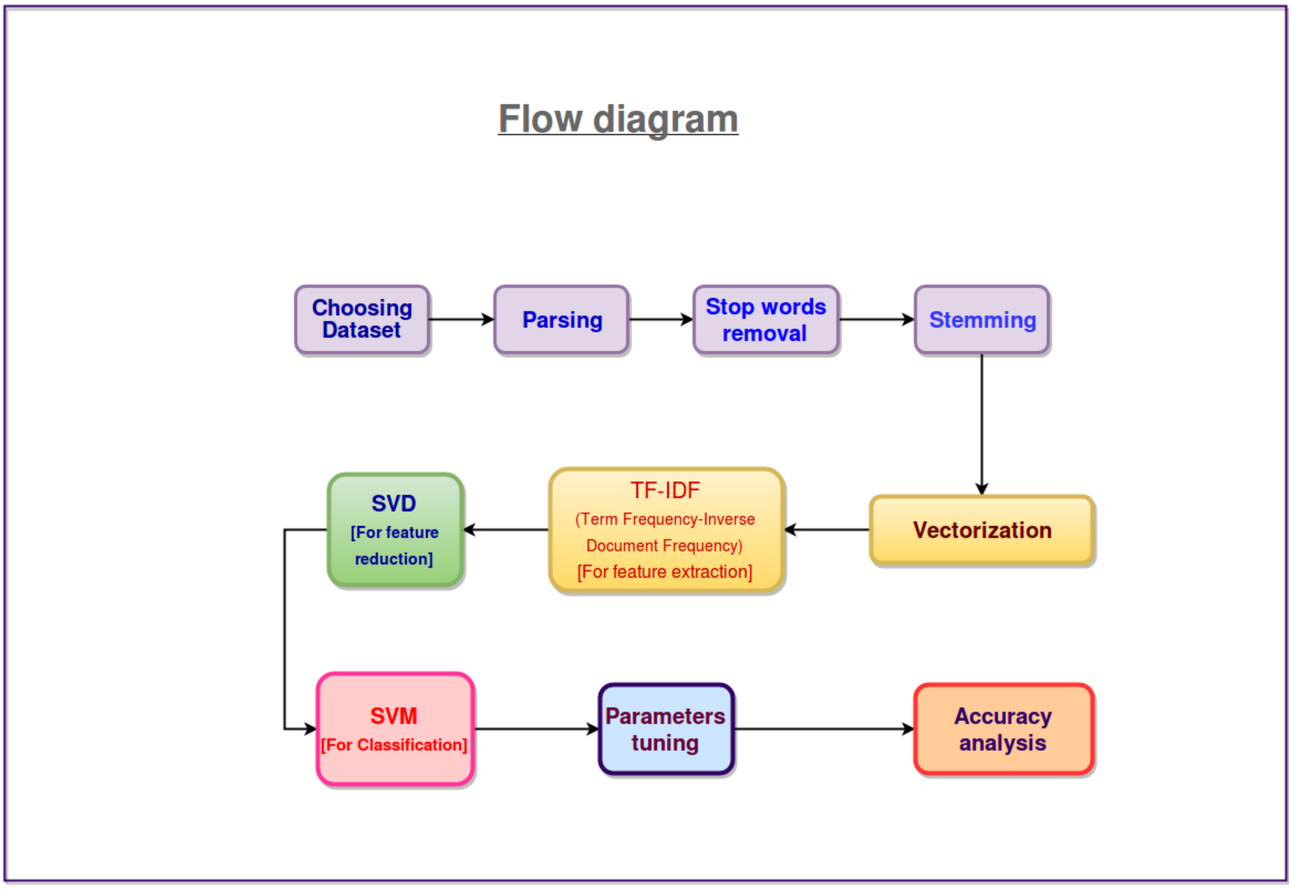


Fig - 5.1: Flow diagram

Literature survey

S.no.	Paper Title	Year	Journal/Conference	Abstract	Author
1.	Inductive learning algorithms and representations for text categorization [2]	1998	7th International Conference on Information and knowledge management1998-11-01	In this paper five different algorithms for text classification have been compared like find similar, decision trees, naive bayes,bayes nets and SVM.The best result were found using SVM . So it guided us to choose SVM for classification.	Susan Dumais,Mehran Sahami,John Platt,David Heckerman
2	A Statistical Learning Model of Text Classification forSupport Vector Machines [4]	2001	24th annual international ACM SIGIR conference on Research and development in information retrieval2001-09-01	In this paper the statistical properties of text-classification tasks is connecced with generalizaton performance of SVM. It explained why and when SVMs perform well for text classification.	Thorsten Joachims
3.	High-performing feature selection for text classification [7]	2002	11th International Conference on Information and knowledge management2002-11-04	This paper mainly focuses on the importance of feature selection and feature reduction in text classification. Here various techniques have been discussed on different machine learning algorithms for feature reduction.In this paper CHI_MAX and IG methods have been combined for giving weights. For reducing redundancy μ co-occurence method is used here.	Monica Rogati,Yiming Yang
4.	Transductive Inference for Text Classification using Support Vector Machines [3]	1999	16th International Conference on Machine Learning27-06-1999	In this paper the concept of Transductive SVM (TSVM) is introduced. It explains us the limitations of normal SVM in some cases where TSVM are better than SVM. The experiments here tells us about significant improvement of TSVM over inductive methods. TSVMs work very well in cases where there is smaller training dataset than the test set.	Thorsten Joachims
5.	An Optimal SVM-Based Text Classification Algorithm [8]	2006	International Conference on Machine Learning and Cybernetics13/09/2006	This paper describes new algorithms for feature selection which highly optimize the efficiency of classification. The new algorithm applies entropy weighing scheme for feature selection in a newer way. Also optimal parameter settings have been used to get better results.	Zi-Qiang Wang, Xia Sun, De-Xian Zhang, Xin Li

Table - 6.1: Literature survey

Experiments

S.No.	Kernel	Penalty Parameter (C)	Kernel Coefficient (gamma)	Training Time (in seconds)	Accuracy (% age)
1.1	Radial Basis Function (rbf)	1	0	14.7179185738	53.5163776493
1.2	Radial Basis Function (rbf)	1	5	20.2596582446	88.2947976879
1.3	Radial Basis Function (rbf)	1	10	49.7798475756	86.8015414258
2.1	Radial Basis Function (rbf)	10	0	10.9192837309	80.9730250482
2.2	Radial Basis Function (rbf)	10	5	19.5810598891	88.9691714836
2.3	Radial Basis Function (rbf)	10	10	50.6595395278	87.2350674374
3.1	Radial Basis Function (rbf)	100	0	7.2436635578	86.7052023121
3.2	Radial Basis Function (rbf)	100	5	19.2433398237	88.2947976879
3.3	Radial Basis Function (rbf)	100	10	50.6608579851	87.1868978805
4.1	Radial Basis Function (rbf)	1000	0	6.0687723209	88.1502890173
4.2	Radial Basis Function (rbf)	1000	5	19.1799743322	88.2947976879
4.3	Radial Basis Function (rbf)	1000	10	50.5428827899	87.1868978805
5.1	Radial Basis Function (rbf)	10000	0	6.2777937673	87.6685934489
5.2	Radial Basis Function (rbf)	10000	5	19.1768832492	88.2947976879
5.3	Radial Basis Function (rbf)	10000	10	50.9507252798	87.1868978805
6.1	Radial Basis Function (rbf)	100000	0	7.1739251665	87.5722543353
6.2	Radial Basis Function (rbf)	100000	5	19.1878773779	88.2947976879
6.3	Radial Basis Function (rbf)	100000	10	50.5921981372	87.1868978805
7.1	Radial Basis Function (rbf)	1000000	0	9.7328505405	87.1868978805
7.2	Radial Basis Function (rbf)	1000000	5	19.2118927153	88.2947976879
7.3	Radial Basis Function (rbf)	1000000	10	50.5413184316	87.1868978805

Table - 7.1: Change in **Accuracy** and **Training time** with change in SVC parameter **C** and **gamma** [for **'rbf'** kernel]

S.No.	Kernel	Penalty Parameter (C)	Training Time (in seconds)	Accuracy (% age)
1	Linear	1	5.7447666912	85.5973025048
2	Linear	10	4.4802451655	88.4393063584
3	Linear	100	4.4675985817	87.9094412331
4	Linear	1000	5.3948755933	87.4759152216
5	Linear	10000	10.4627913232	87.1868978805
6	Linear	100000	101.26222459789	87.0905587669

Table - 7.2: Change in **Accuracy** and **Training time** with change in SVC parameter **C** [for **'linear'** kernel]

Observations :

Using different values for the **Penalty Parameter (C)** and **Kernel coefficient (gamma)**, we found that:

For **‘rbf’** kernel:

- For a fixed **gamma** value,with the increase in the value of **C**, the accuracy of our predictions increased gradually, reached a certain maximum value and then decreased gradually or remained constant. This observation can be justified because as we increase the value of **C**, the generality of the classification model is also decreased and is somewhat lost with very large value of **C** as the model tries to include every outlier inside the correct decision boundary and results in overfitting the training data.

For **‘linear’** kernel:

- Training time increased with increasing value of **C**, and increases by a high amount after **C = 10000**.
- **gamma** parameter have NO significance for the **‘linear’** kernel function.

Best results :

The following are the best results we found from our experiments:

For **‘rbf’** kernel:

- C = 10.0
- gamma = 5.0
- Training Time taken = 19.581059889092217 seconds
- Accuracy on test dataset = 88.969171483622356 %

For **‘linear’** kernel:

- C = 10.0
- Training Time taken = 4.48024516547855 seconds
- Accuracy on test dataset = 88.439306358381498 %

After taking these observations, we then took the values of the svc parameters C and gamma of the 5 most accurate observations. We then used these 5 most accurate observations to computed the Average Accuracy and Average Training time with different random splits of training and test, for both the ‘rbf’ and the ‘linear’ kernel.

Use of random training and test dataset:

S.No.	Kernel	Penalty Parameter (C)	Kernel Coefficient (gamma)	Average Training Time (in seconds)	Average Accuracy (% age)	Minimum Accuracy (% age)	Maximum Accuracy (% age)
1	Radial Basis Function (rbf)	10	5	19.5975905119	87.591522158	86.8497109827	88.1984585742
2	Radial Basis Function (rbf)	1	5	19.8280292122	88.2369942197	88.0539499037	88.3429672447
3	Radial Basis Function (rbf)	100	5	19.1036437403	86.5317919075	85.9344894027	86.7052023121
4	Radial Basis Function (rbf)	1000	5	18.7571186214	87.9768786127	87.1868978805	88.1984585742
5	Radial Basis Function (rbf)	10000	5	20.2823468282	88.7379576108	87.8612716763	89.2581888247

Table - 7.3: Computation of **Average Accuracy** and **Average Training time** with different random splits of training and test data for fixed value of SVC parameters **C** and **gamma** [for **'rbf'** kernel]

S.No.	Kernel	Penalty Parameter (C)	Average Training Time (in seconds)	Average Accuracy (% age)	Minimum Accuracy (% age)	Maximum Accuracy (% age)
1	Linear	10	26.4541190135	86.3102119461	85.3082851638	87.6685934489
2	Linear	100	19.0949781875	87.3603082852	86.6570327553	88.8728323699
3	Linear	1000	26.5995590032	85.4624277457	84.4894026975	86.8978805395
4	Linear	10000	19.1290173923	87.2832369942	86.4161849711	88.5356454721
5	Linear	100000	22.8623470763	88.1695568401	87.4759152216	89.0655105973

Table - 7.4: Computation of **Average Accuracy** and **Average Training time** with different random splits of training and test data for fixed value of SVC parameter **C** [for **'linear kernel'**]

Observations :

Using different random splits of training and test datasets for fixed value of svc parameters, we found that:

For **'rbf'** kernel:

- All the top 5 most accurate observations were found having gamma = 5.0 .
- We found that the best result found on average basis is having different **C** parameter from the best result found previously. This is the result of the change in training and test dataset, which shows that the accuracy depends heavily on the training data, as well as the test data on which the model is to be tested of the same dataset. So, we cannot fix the value of the parameters **C** and **gamma** and say that these are the perfect parameter values for this dataset. The parameter values which are best for a specific split of training and test data may not be the best for any other split.
- So, we cannot find the perfect parameter values for the dataset if the training and test datasets are applicable to changes but we can find the perfect parameter values for a dataset with fixed training and test dataset.
- We even reached 89.25 % accuracy for some random splitting of out dataset in our last observation, which also turned out to be the best observation.

For **'linear'** kernel:

- In this case also we have different best results for the same reason as above.
- Here also we reached 89.06 % accuracy for some random splitting of out dataset in our last observation, which also turned out to be the best observation.

Best results :

The following are the best results we found from our experiments: For **'rbf'** kernel:

- C = 10000.0
- gamma = 5.0
- Average Training Time taken = 20.28234682823986 seconds
- Average Accuracy on test dataset = 88.737957610789986 %

For **'linear'** kernel:

- C = 100000.0
- Average Training Time taken = 22.862347076279367 seconds
- Average Accuracy on test dataset = 88.169556840077068 %

Accuracy vs Penalty Parameter (C) graphs :-

For linear kernel :

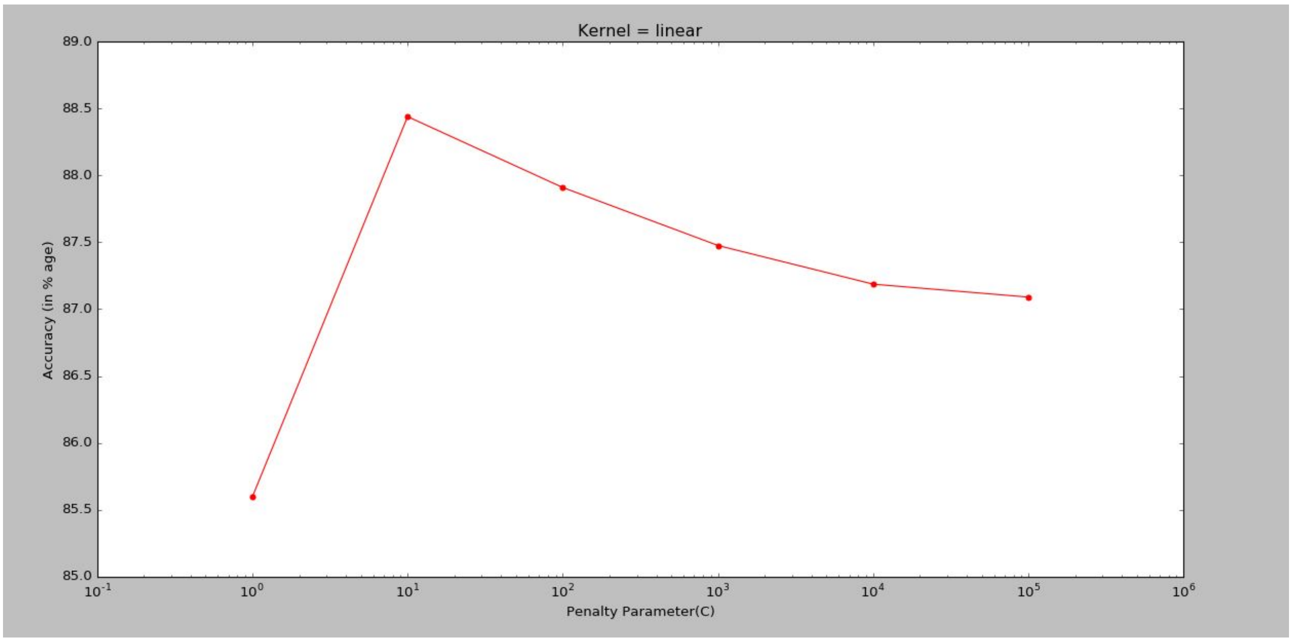


Fig - 7.1: **Accuracy** vs **Penalty Parameter (C)** (for **linear kernel**)
Standard deviation = 0.964835

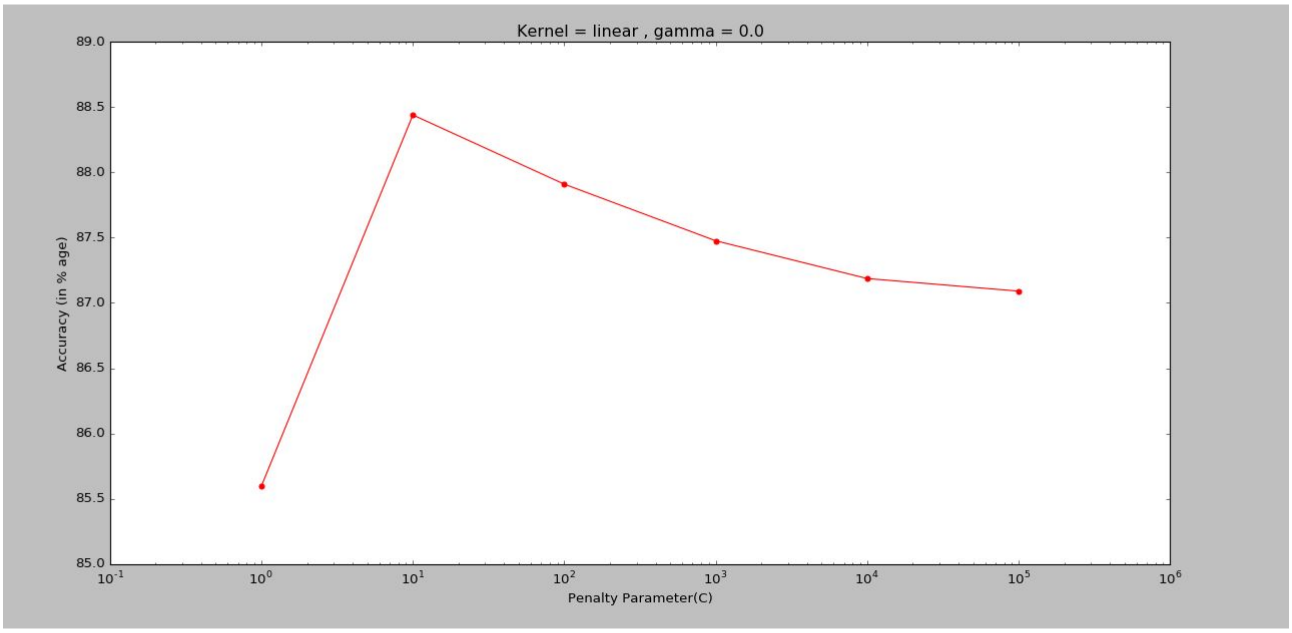


Fig - 7.2: **Accuracy** vs **Penalty Parameter (C)** (for **linear kernel** with **gamma** = 0.0)
Standard deviation = 0.964835

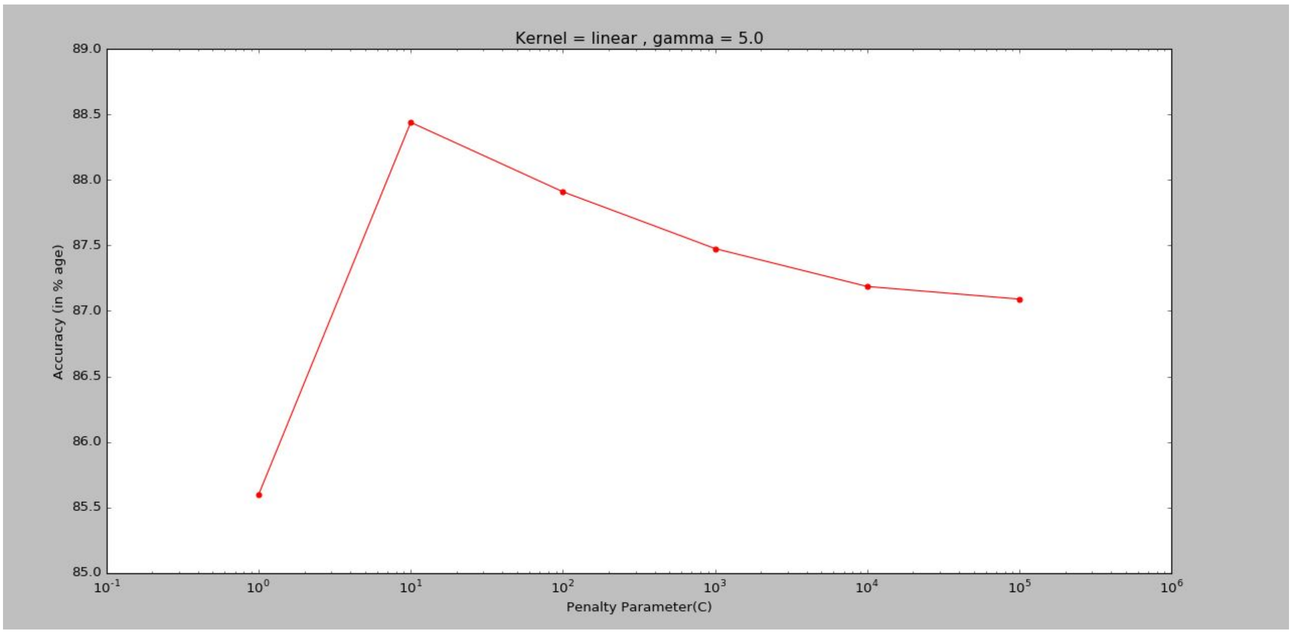


Fig - 7.3: **Accuracy** vs **Penalty Parameter (C)** (for **linear kernel** with **gamma** = 5.0)
Standard deviation = 0.964835

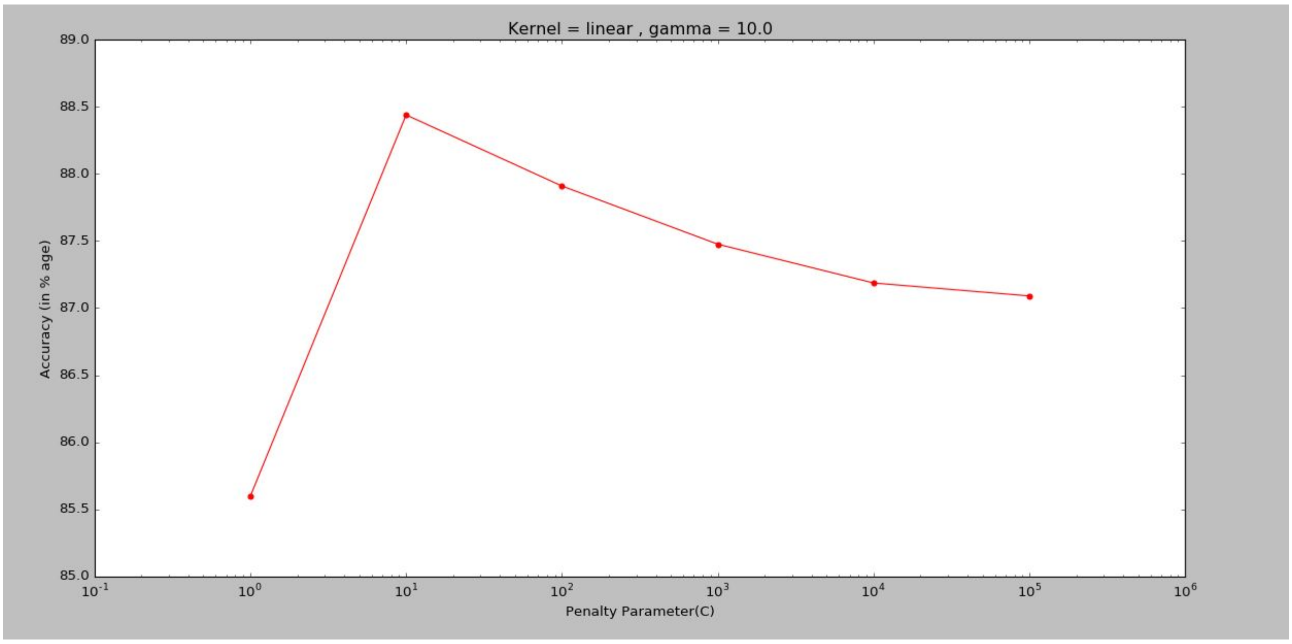


Fig - 7.4: **Accuracy** vs **Penalty Parameter (C)** (for **linear kernel** with **gamma** = 10.0)
Standard deviation = 0.964835

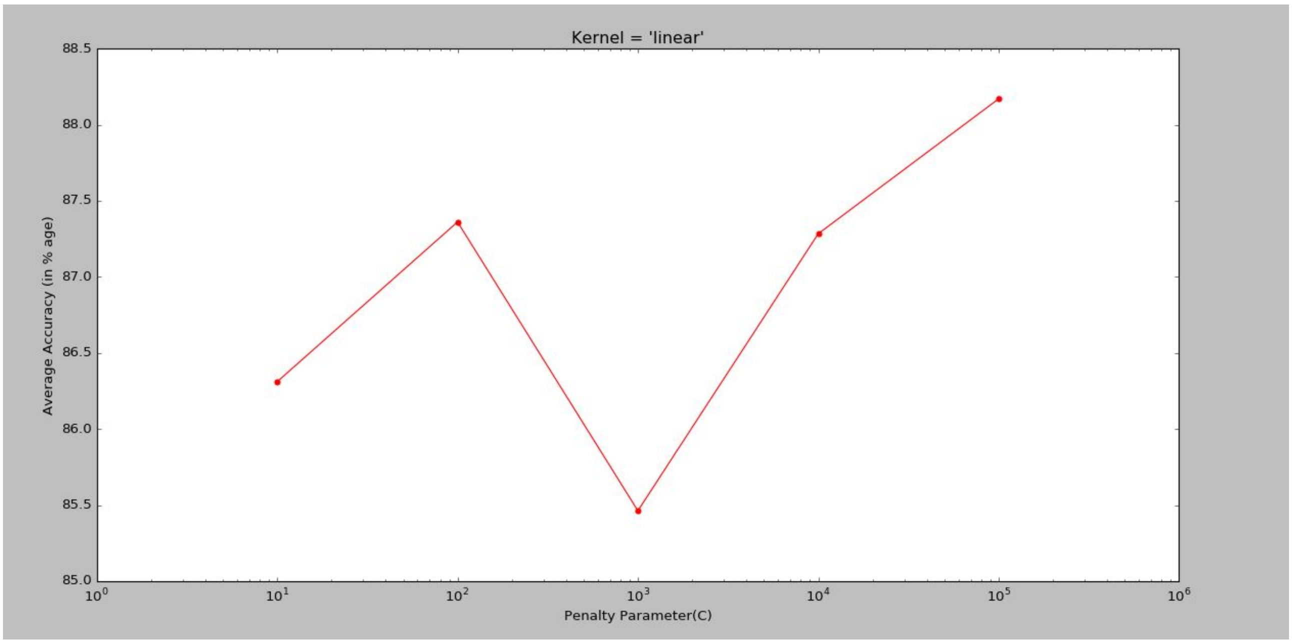


Fig - 7.5: **Accuracy** vs **Penalty Parameter (C)** (for **linear kernel** with different splits of training and test data)
Standard deviation = 1.046843

We can clearly see that the accuracy for the test data varies by a good amount for different splits of training and test data.

This shows that the accuracy depends heavily on the data on which the model is trained, as well as the future data on which it will be tested.

For rbf kernel :

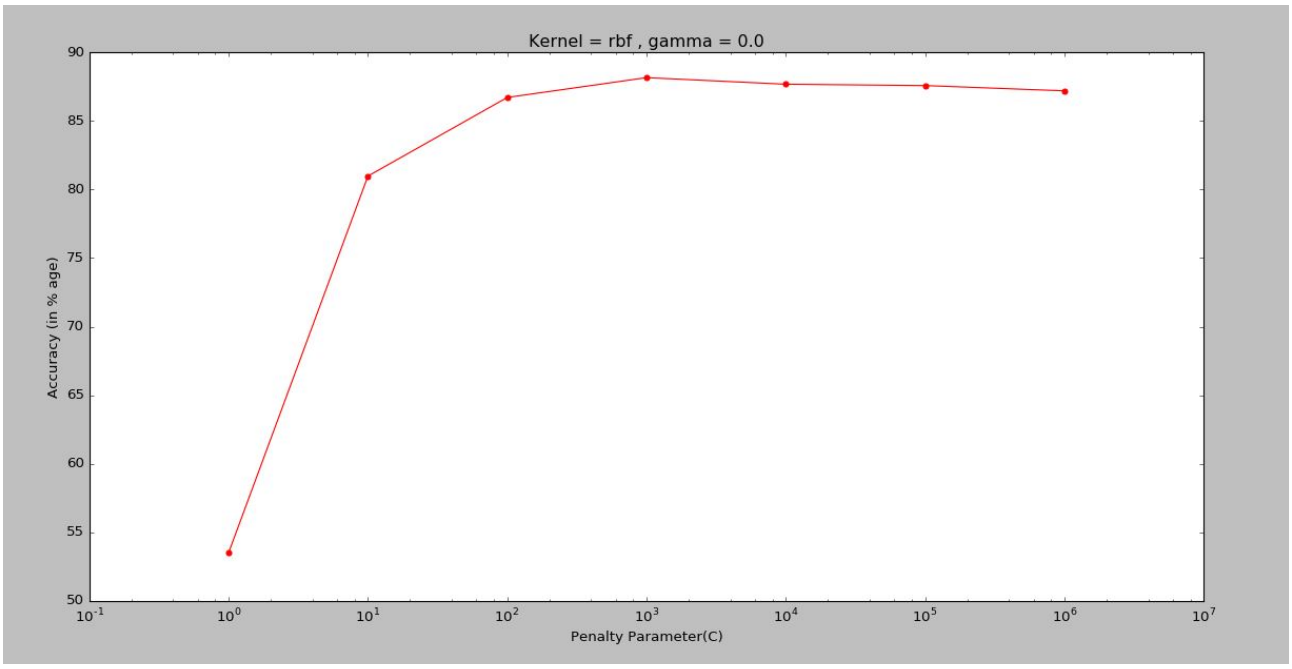


Fig - 7.6: **Accuracy** vs **Penalty Parameter (C)** (for **rbf kernel** with **gamma** = 0.0)
Standard deviation = 12.660401

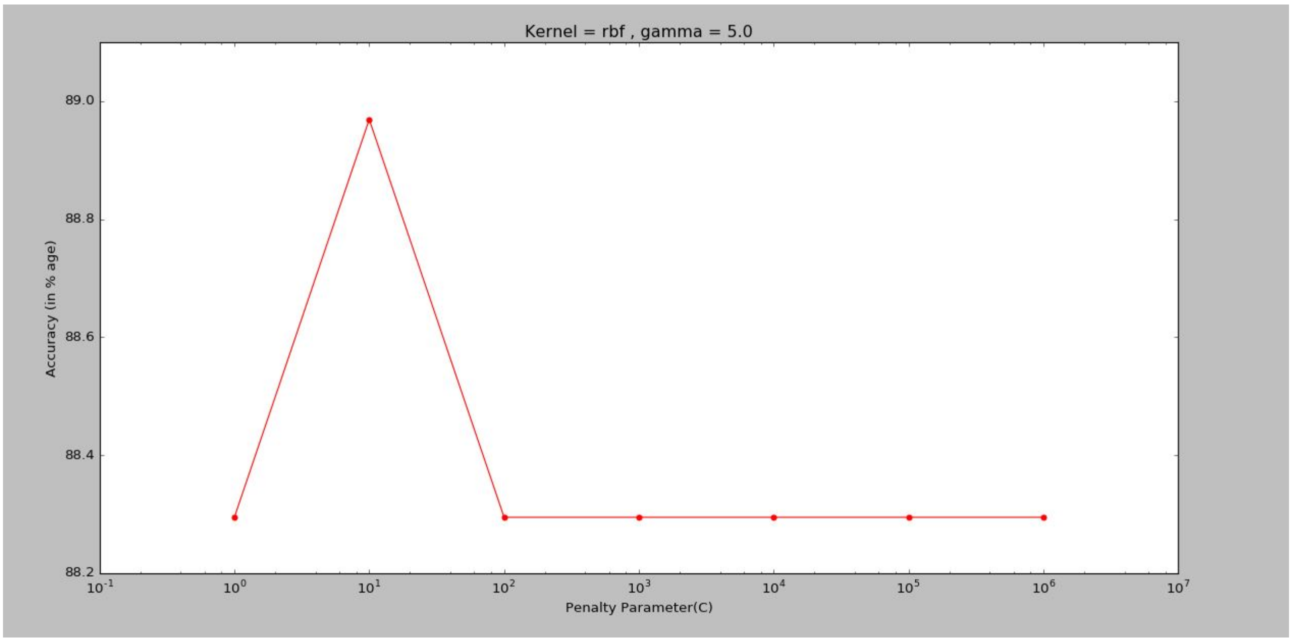


Fig - 7.7: **Accuracy** vs **Penalty Parameter (C)** (for **rbf kernel** with **gamma** = 5.0)
Standard deviation = 0.254889

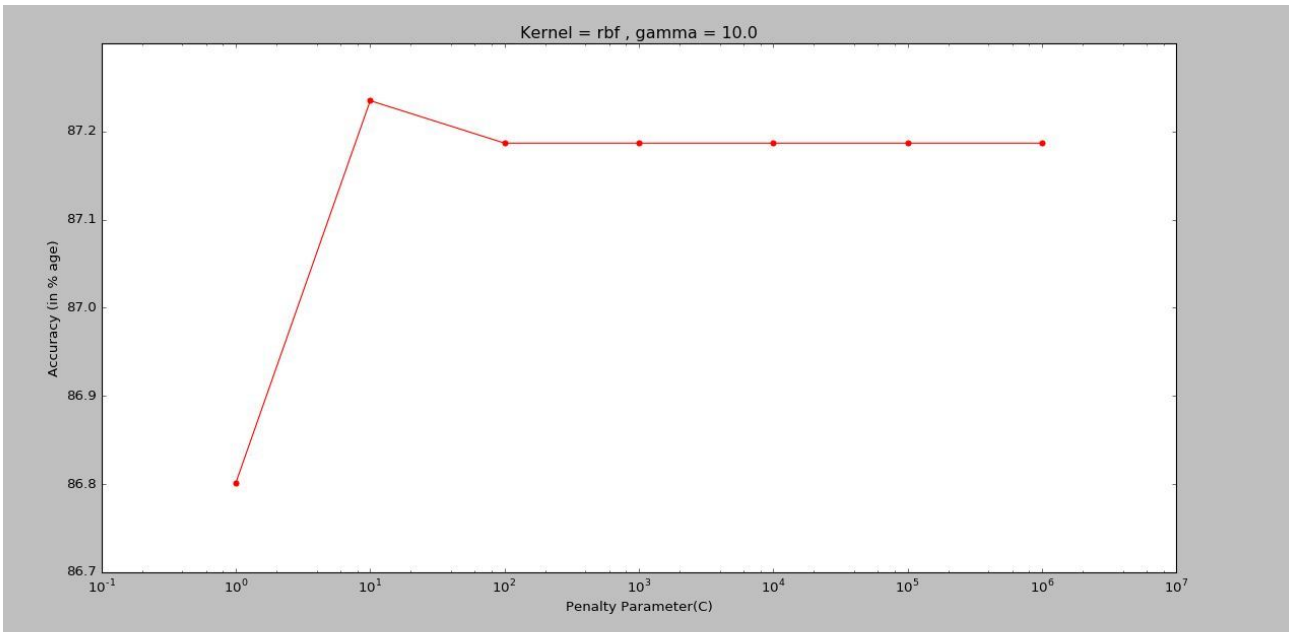


Fig - 7.8: **Accuracy** vs **Penalty Parameter (C)** (for **rbf kernel** with **gamma** = 10.0)
Standard deviation = 0.149765

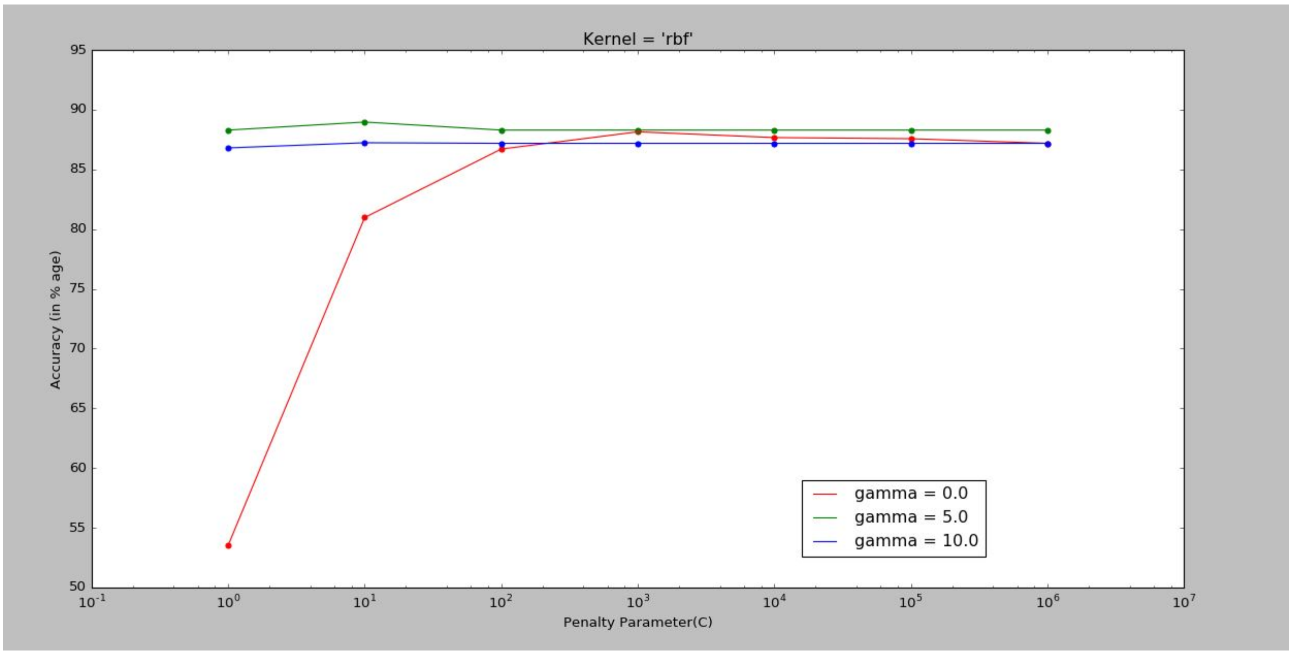


Fig - 7.9: **Accuracy** vs **Penalty Parameter (C)** (for **rbf kernel**)

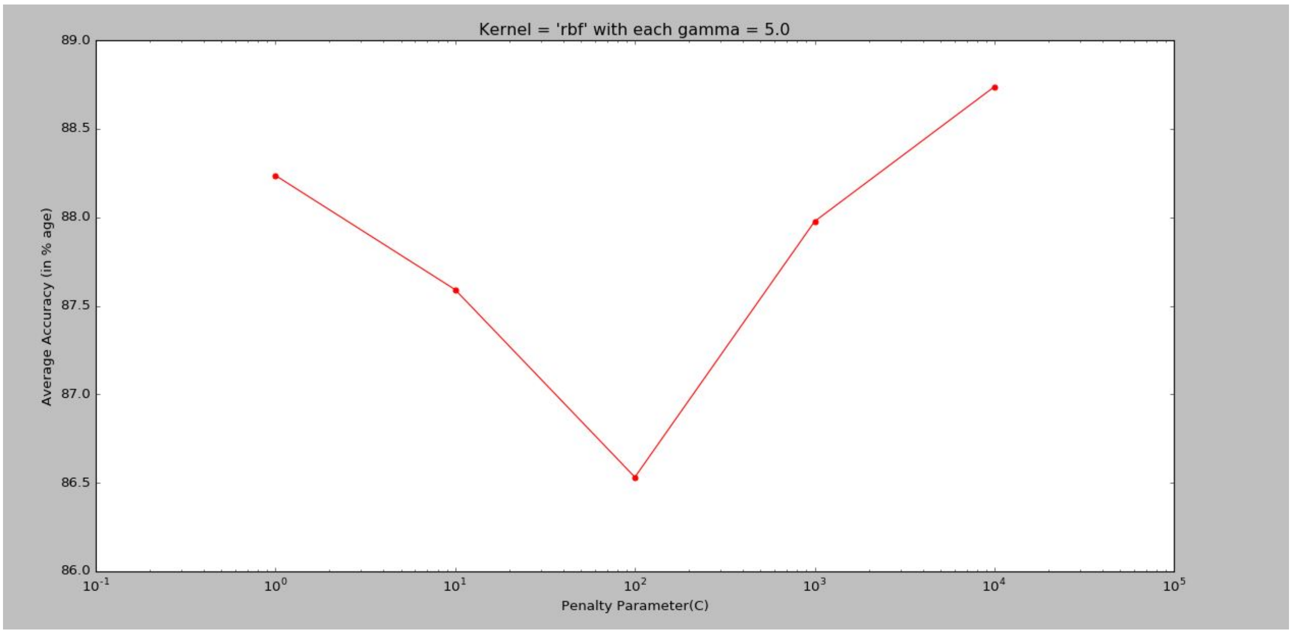


Fig - 7.10: **Accuracy** vs **Penalty Parameter (C)** (for **rbf kernel** with different splits of training and test data)
Standard deviation = 0.829563

Here also we can clearly see that the accuracy for the test data varies by a good amount for different splits of training and test data.

This shows that the accuracy depends heavily on the data on which the model is trained, as well as the future data on which it will be tested.

Generic Text Classifier

After training, testing and analysing various parameter of SVM function on various kernels for getting the best accuracy results on "Reuters Dataset".

- We have created a generic text classification model which can be used on many datasets by applying minor changes in the code and sometimes without any changes.
- We tested our text classification model on various small datasets like News headline classifier [6], amazon reviews [5], IMDb movie reviews [5], Yelp reviews [5] and SMS Spam Collection [1] etc.
- The tests we did was on the same dataset obtained by splitting datasets into training and testing parts

We did binary classification (positive or negative) on review datasets:-

Results Obtained:

The average accuracy obtained for various datasets:-

1. Amazon review = 75.74% (data size = 1000 , kernel = linear)

2. IMDb review = 73.83% (data size = 1000 , kernel = linear)

3. Yelp review = 77.03% (data size = 1000 , kernel = linear)

4. Spam Detection = 96.34% (best we obtained till now , data size = 5574 , kernel = linear)

5. News Headline = 90.71% (data size = 5977 , kernel = rbf)

All these were having 100 random states.

Amazon Online Review Binary-Classifier

For amazon review dataset we trained it and tested it on the splitted portion, Once this part was over, we modified our model and made it online ,

- now it takes review from user.
- extracts and reduces the features complete dataset (original+user input) using SVD.
- training is done only for our previous dataset.
- then the review is accordingly classified
- the user entered review is added to our dataset after classification

This added data helps us to improve accuracy if similar input is given again.

Future Work

Till now we have trained and tested our model on the same dataset. So there is a high chance of getting good accuracy, but our new challenge is to train our model on one dataset and using its extracted features, try to classify(test) different but related dataset.

For doing this , our instructor suggested us to use "Transfer Learning".

Transfer Learning

Transfer learning is an emerging field in Machine Learning.It involves using knowledge extracted while solving one problem to apply it to a related but different problem.For instance, we can use knowlegde extracted by base (bigger) dataset and then extract feature of the new (smaller) dataset and add these feature to previous features so that the combined knowledge gives us better reults without training our model again and again.

For instance, we could train our model on X news agency's(say FOX News) dataset and test on Y news agency's(say ABC news) dataset and analyse the features which lead to anomalous results. We could then adjust our model to these features, in order to develop a generic model which gives accurate results for most of the datasets in this domain(News).

During concluding part of our project,we worked on applying transfer learning in our model. We trained our model on News Articles Data(obtained from web scraping "www.webhose.io") dataset and tried to tested it on another dataset which contains News Headlines only but we faced various challenges.

So when we train our model with a larger dataset with 5000 features and we have to test on a smaller dataset say with 100 features, our objective is to extract features from the bigger dataset which are common in both the datasets.Finally , we need to scale features of both the datasets on a common scale. We tried to use SVD for feature reduction but it didn't yield required results as applying SVD individually gives different scales for the two sets of features. The basic requirement for a effective transfer learning is that no. of features in training dataset should be equivalent to features in testing dataset.

Softwares Used :

Spyder 3 (Scientific PYthon Development EnviRonment)

Language Used :

Python 3.5

Libraries Used :

- pandas (Python Data Analysis Library) for inputting the Dataset (.tsv file).
- The module pyplot of matplotlib library is used for graph plotting re library for the usage of Regular Expressions for text cleaning.
- Stopwords list from the 'corpus' package of nltk (Natural Language Processing Toolkit) data package.
- PorterStemmer algorithm from the package nltk.stem.porter for stemming of words.
- CountVectorizer class from sklearn.feature_extraction.text submodule for building the matrix (sparse) of word counts from text documents.
- TfidfTransformer class from sklearn.feature_extraction.text submodule to convert the count matrix (sparse) to a matrix of TF_IDF features.
- TruncatedSVD class from sklearn.decomposition module for dimensionality reduction of the feature space.
- train_test_split function from sklearn.cross_validation module for splitting the whole Dataset into random train and test sunsets.
- SVC class from sklearn.svm module for classification of the test set.

Bibliography

[1] Tiago A. Almeida and José María Gómez Hidalgo. Sms spam collection v. 1, 2011. http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/.

[2] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 148–155, New York, NY, USA, 1998. ACM. http://doi.acm.org/10.1145/288627.288651.

[3] Thorsten Joachims. Transductive inference for text classification using support vector machines.

[4] Thorsten Joachims. A statistical learning learning model of text classification for support vector machines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 128–136, New York, NY, USA, 2001. ACM. http://doi.acm.org/10.1145/383952.383974.

[5] M. Lichman. UCI machine learning repository, 2013. http://archive.ics.uci.edu/ml.

[6] Morgan Mazer. How news appears on social media, 2017. https://www.kaggle.com/socialmedianews/how-news-appears-on-social-media/data.

[7] Monica Rogati and Yiming Yang. High-performing feature selection for text classification. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 659–661, New York, NY, USA, 2002. ACM. http://doi.acm.org/10.1145/584792.584911.

[8] Zi-Qiang Wang, Xia Sun, De-Xian Zhang, and Xin Li. An optimal svm-based text classification algorithm. In *Machine Learning and Cybernetics, 2006 International Conference on*, pages 1378–1381. IEEE, 2006.