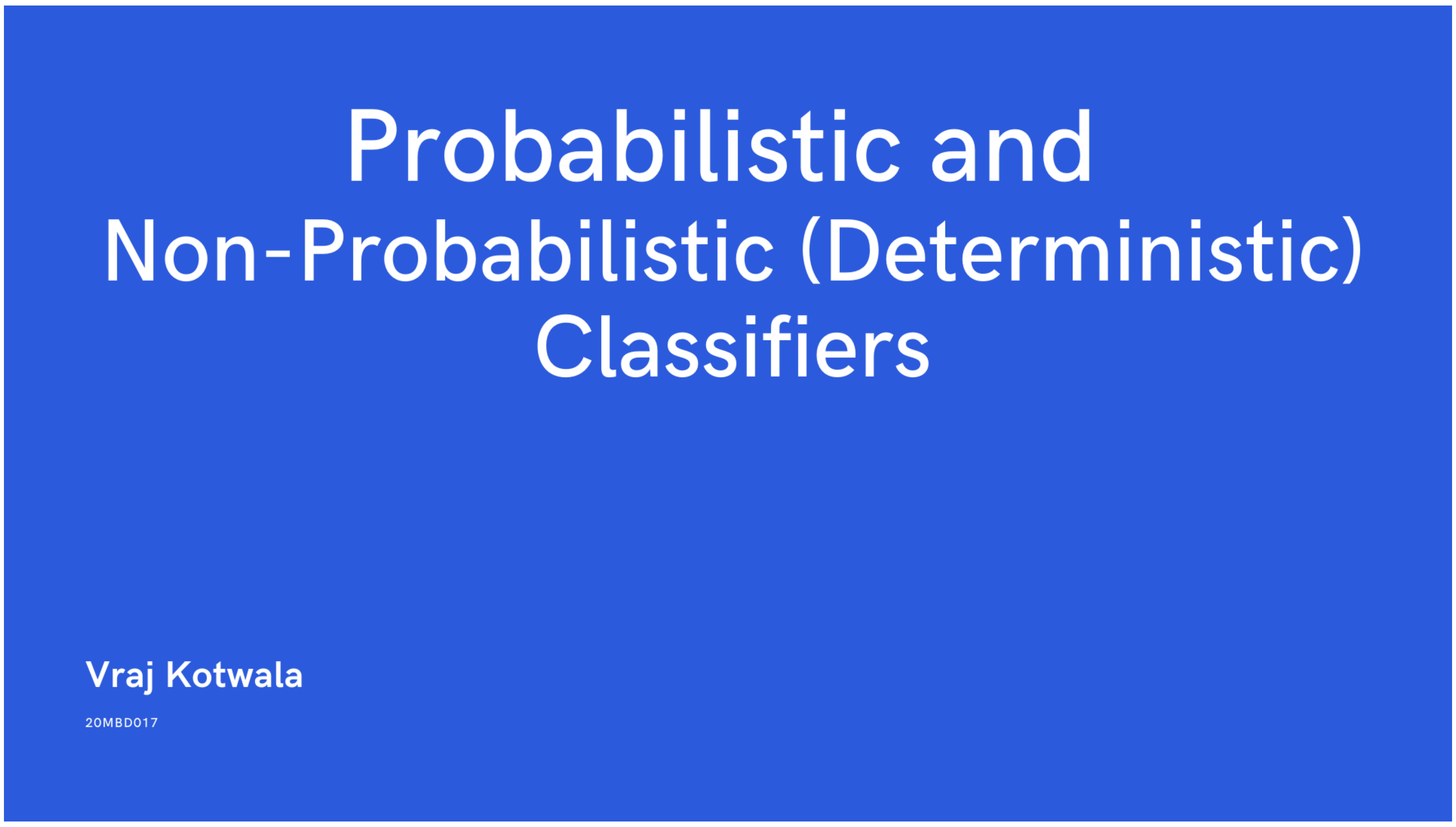
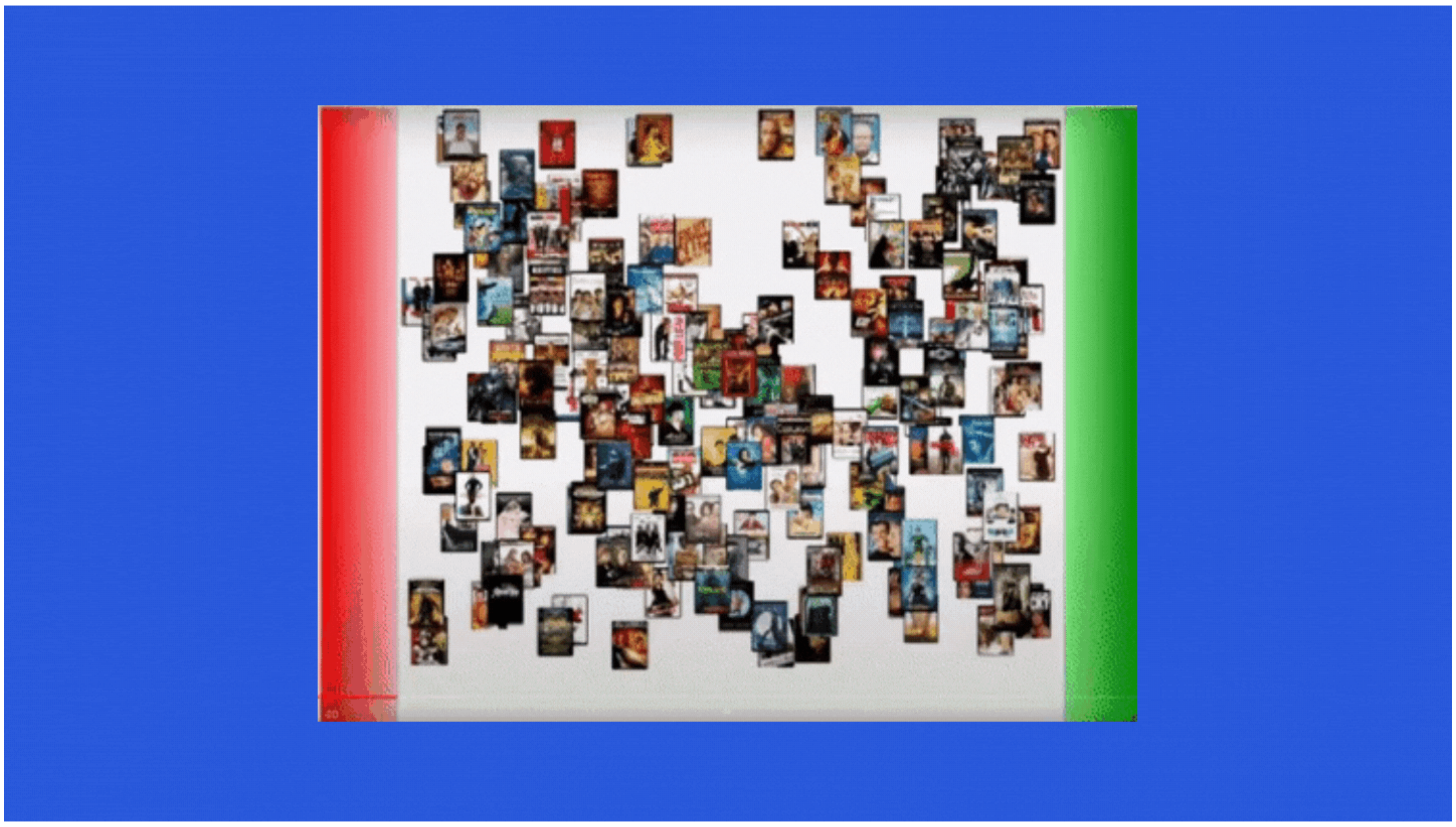
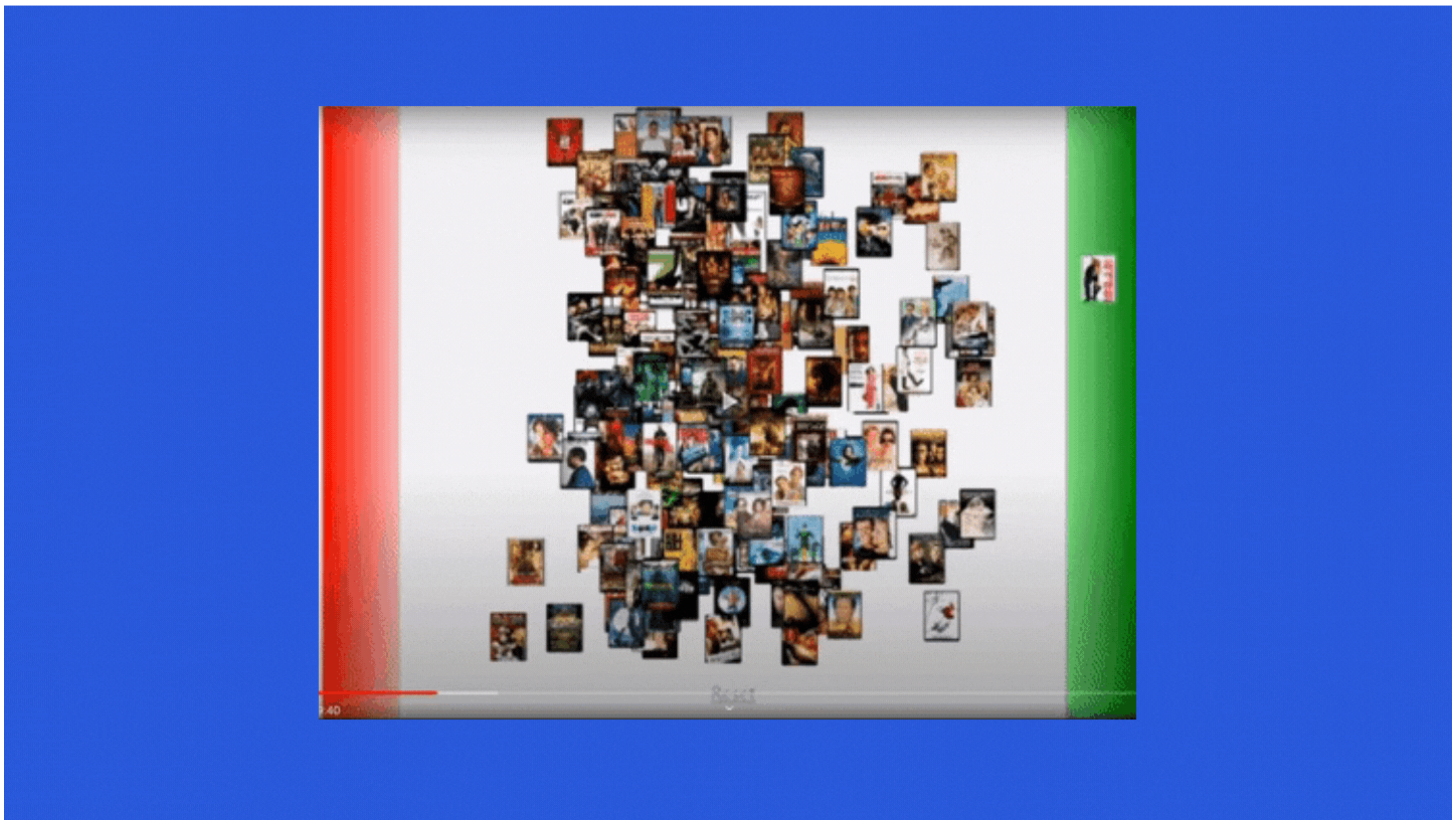
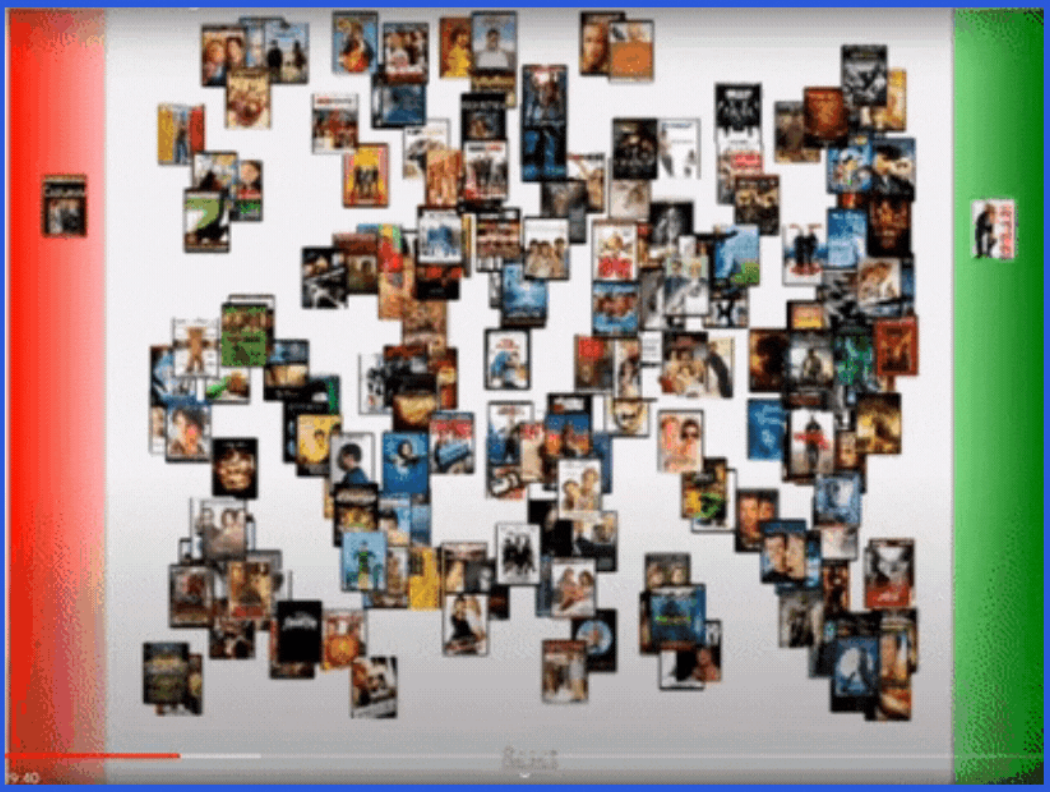


Probabilistic and Non-Probabilistic (Deterministic) Classifiers

Vraj Kotwala

20MBD017



Some examples for probabilistic models are Logistic Regression, Bayesian Classifiers, Nearest Neighbors, Hidden Markov Models, and Neural Networks (with a Softmax output layer).

An example to better understand probabilistic classifiers. Take the task of classifying an image of an animal into five classes — {Dog, Cat, Deer, Lion, Rabbit} as the problem. As input, we have an image (of a dog). For this example, let’s consider that the classifier works well and provides correct/ acceptable results for the particular input we are discussing. When the image is provided as the input to the probabilistic classifier, it will provide an output such as (Dog (0.6), Cat (0.2), Deer(0.1), Lion(0.04), Rabbit(0.06)). But, if the classifier is non-probabilistic, it will only output “Dog”.

One of the major advantages of probabilistic models is that they provide an idea about the uncertainty associated with predictions. In other words, we can get an idea of how confident a machine learning model is on its prediction. If we consider the above example, if the probabilistic classifier assigns a probability of 0.9 for ‘Dog’ class instead of 0.6, it means the classifier is more confident that the animal in the image is a dog. These concepts related to uncertainty and confidence are extremely useful when it comes to critical machine learning applications such as disease diagnosis and autonomous driving.

If the model is Non-Probabilistic (Deterministic), it will usually output only the most likely class that the input data instance belongs to. “Support Vector Machines” is a popular non-probabilistic classifier.

A deterministic model is a mathematical model in which the outcomes are determined through known relationships, without a room for random variation. In such models, a given input will always produce the same output, such as known chemical reaction.

Whereas, the probabilistic model includes both a deterministic component and a random error component.

Objective Functions

In order to identify whether a particular model is probabilistic or not, we can look at its Objective Function. In machine learning, we aim to optimize a model to excel at a particular task. The aim of having an objective function is to provide a value based on the model's outputs, so optimization can be done by either maximizing or minimizing the particular value.

If we take a basic machine learning model such as Linear Regression, the objective function is based on the squared error.

When it comes to Support Vector Machines, the objective is to maximize the margins or the distance between support vectors.

As you can see, in both Linear Regression and Support Vector Machines, the objective functions are not based on probabilities. So, they can be considered as non-probabilistic models.

The generally for ML researchers and practitioners in almost all cases, the SVM is better than the Naive Bayes. Because rather than taking a probabilistic approach SVM works on the geometric interpretation of the problems.

But, from a theoretical point of view, it is a little bit hard to compare the two methods.

For classifying Emails into spam or hams (not spam), or tweets into different polarities like Positive, Negative we can use different classification algorithms like Naïve Bayes, SVM, or even Neural Networks can be used.

Importing & Cleaninig data

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
sms_data = pd.read_csv("spam.csv", encoding='latin-1')
```

In [3]:
```
sms_data
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

In [4]:
```python
sms_data.drop(labels = ['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis = 1,inplace = True)
sms_data.head()
```

Out[4]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

1. Lemmatization / Stemming: Shorten the words to their root stems
2. Removing Stop words: Removing punctuations, or unwanted token that doesn't carry a lot of meaning based on the context
3. Lowercasing the words also helps
4. Count up all the tokens (which is known as term frequency)
5. The more the frequent a word, the more important it might be
6. Can be a great way to determine the significant words in a text

In [5]:
```python
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

In [6]:
```python
stemming = PorterStemmer()
corpus = []

for i in range (0,len(sms_data)):
    s1 = re.sub('[^a-zA-Z]',repl = ' ',string = sms_data['v2'][i])
    s1.lower()
    s1 = s1.split()
    s1 = [stemming.stem(word) for word in s1 if word not in set(stopwords.words('english'))]
    s1 = ' '.join(s1)
    corpus.append(s1)
```

In [7]:
```python
corpus
```

Out[7]:
```
['go jurong point crazi avail bugi n great world la e buffet cine got amor wat',
 'ok lar joke wif u oni',
 'free entri wkli comp win fa cup final tkt st may text fa receiv entri question std txt rate t c appli',
 'u dun say earli hor u c alreadi say',
 'nah i think goe usf live around though',
 'freemsg hey darl week word back i like fun still tb ok xxx std chg send rcv',
 'even brother like speak they treat like aid patent',
 'as per request mell mell oru minnaminungint nurungu vettam set callertun caller press copi friend callertun',
 'winner as valu network custom select receivea prize reward to claim call claim code kl valid hour',
 'had mobil month u r entitl updat latest colour mobil camera free call the mobil updat co free',
 'i gonna home soon want talk stuff anymor tonight k i cri enough today',
 'six chanc win cash from pound txt csh send cost p day day tsandc appli repli hl info',
 'urgent you week free membership prize jackpot txt word claim no t c www dbuk net lccltd pobox ldnw a rw',
 'i search right word thank breather i promis wont take help grant fulfil promis you wonder bless time',
 'i have a date on sunday with will',
 'xxxmobilemovieclub to use credit click wap link next txt messag click http wap xxxmobilemovieclub com n qjkgighjjgc
bl',
 'oh k watch',
 'eh u rememb spell name ye he v naughti make v wet',
```

By using this Count-Vectorizer we'll tokenize a collection of text documents and built a vocabulary, this vocabulary is also used to encode new documents.

This vector represents the length of the entire vocabulary and the count for the number of times each word appeared in the document.

In [8]:
```python
from sklearn.feature_extraction.text import CountVectorizer
countvectorizer = CountVectorizer()
```

In [9]:
```python
x = countvectorizer.fit_transform(corpus).toarray()
```

In [10]:
```python
x
```

Out[10]:
```
array([[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, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [11]:
```
y = sms_data['v1'].values
print(y)
```

```
['ham' 'ham' 'spam' ... 'ham' 'ham' 'ham']
```

Train – Test Split

In [12]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.3, stratify=y,random_state=100)
```

Naive Bayes

Naive Bayes is a supervised machine learning algorithm used for classification problems. It is built on Bayes Theorem. It is called Naïve because of its Naive assumption of Conditional Independence among predictors.

Bayes Theorem:

$$P\left(\frac{B}{A}\right) = \frac{\mathrm{P}(B) * P\left(\frac{A}{B}\right)}{\mathrm{P(A)}}$$

Naive Bayes:

$$P\left(\frac{Y}{X1,X2}\right) = \frac{\mathrm{P}(Y) * P\left(\frac{X1}{Y}\right) * P\left(\frac{X2}{Y}\right)}{\mathrm{P(X1,X2)}}$$

In [13]:
```
from sklearn.naive_bayes import MultinomialNB
multinomialnb = MultinomialNB()
multinomialnb.fit(x_train,y_train)
```

Out[13]:
```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [14]:
```
y_pred_NB = multinomialnb.predict(x_test)
```

In [15]:
```
y_pred_NB
```

Out[15]:
```
array(['ham', 'ham', 'ham', ..., 'spam', 'ham', 'ham'], dtype='<U4')
```

In [16]:
```
multinomialnb.predict_proba(x_test)
```

Out[16]:
```
array([[9.99972983e-01, 2.70168380e-05],
       [9.99999999e-01, 6.43484365e-10],
       [9.98447409e-01, 1.55259143e-03],
       ...,
       [2.61467010e-08, 9.99999974e-01],
       [9.99983819e-01, 1.61812956e-05],
       [9.99822462e-01, 1.77537930e-04]])
```

probability of occurance of 0 (ham) is 9.99972983 and 1 (spam) is 2.70168380

In [17]:
```
print(multinomialnb.predict_proba(x_test)[:,1])
```

```
[2.70168380e-05 6.43484365e-10 1.55259143e-03 ... 9.99999974e-01
 1.61812956e-05 1.77537930e-04]
```

SVM

It is a supervised machine learning algorithm by which we can perform Regression and Classification.

In SVM, data points are plotted in n-dimensional space where n is the number of features. Then the classification is done by selecting a suitable hyper-plane that differentiates two classes.

SVM:

$$\beta 0 + \beta 1X1 + \beta 2X2 + \cdots + \beta nXn = \beta 0 + \sum_{i=1}^{n} \beta iXi$$

In [18]:
```
from sklearn.svm import LinearSVC
linearsvc = LinearSVC()
linearsvc.fit(x_train,y_train)
```

Out[18]:
```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
```

In [19]:
```
y_pred_SVM = linearsvc.predict(x_test)
```

In [20]:
```
y_pred_SVM
```

Out[20]:
```
array(['ham', 'ham', 'ham', ..., 'spam', 'ham', 'ham'], dtype=object)
```

In [21]:
```
# linearsvc.predict_proba(x_test)
```

Results

In [22]:
```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
```

In [23]:
```
print(classification_report(y_test,y_pred_NB))
print("MultinomialNB: ", accuracy_score(y_test,y_pred_NB))
```

```
              precision    recall  f1-score   support

         ham       0.99      0.98      0.99      1448
        spam       0.91      0.95      0.93       224

    accuracy                           0.98      1672
   macro avg       0.95      0.97      0.96      1672
weighted avg       0.98      0.98      0.98      1672

MultinomialNB:  0.9802631578947368
```

In [24]:
```
# GaussianNB: 0.8696172248803827
# MultinomialNB: 0.9802631578947368
```

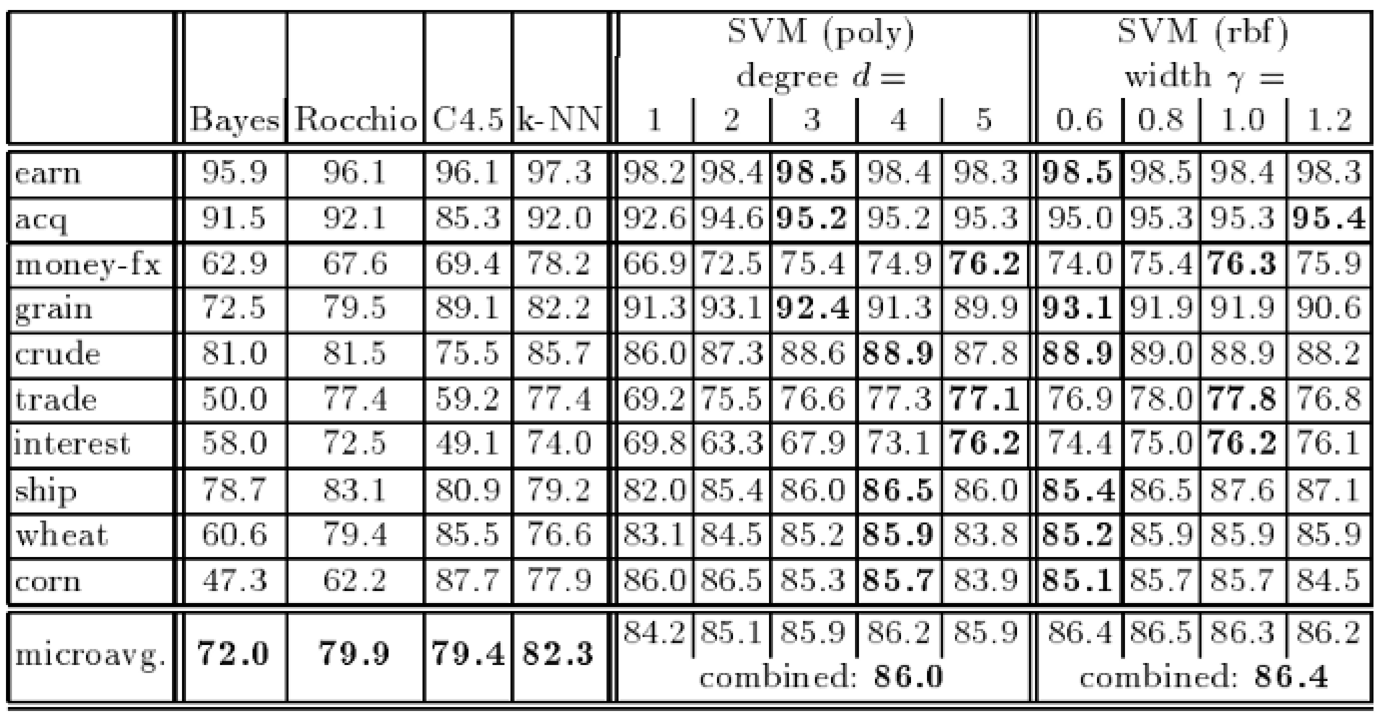
In [25]:
```
print(classification_report(y_test,y_pred_SVM))
print("SVM: ", accuracy_score(y_test,y_pred_SVM))
```

```
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99      1448
        spam       0.99      0.89      0.93       224

    accuracy                           0.98      1672
   macro avg       0.98      0.94      0.96      1672
weighted avg       0.98      0.98      0.98      1672

SVM:  0.9832535885167464
```



					SVM (poly) degree $d =$					SVM (rbf) width $\gamma =$			
	Bayes	Rocchio	C4.5	k-NN	1	2	3	4	5	0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

[http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf (http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf)](http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf)

In the end it comes down to performance on your problem - you basically want to choose the simplest method which will give good enough results for your problem and have a good enough performance. Spam detection has been famously solvable by just Naive Bayes, for example. Face recognition in images by a similar method enhanced with boosting etc.

Conclusion

Which algorithm is better for text classification, well there is no single answer and there will never be since it all comes down to use cases.

There is no single answer about which is the best classification method for a given dataset. Different kinds of classifiers should be always considered for a comparative study over a given dataset. Given the properties of the dataset, you might have some clues that may give preference to some methods. However, it would still be advisable to experiment with all, if possible.

Naive Bayes Classifier (NBC) and Support Vector Machine (SVM) have different options including the choice of kernel function for each. They are both sensitive to parameter optimization (i.e. different parameter selection can significantly change their output) . So, if you have a result showing that NBC is performing better than SVM. This is only true for the selected parameters. However, for another parameter selection, you might find SVM is performing better.

In general, if the assumption of independence in NBC is satisfied by the variables of your dataset and the degree of class overlapping is small (i.e. potential linear decision boundary), NBC would be expected to achieve good. For some datasets, with optimization using wrapper feature selection, for example, NBC may defeat other classifiers. Even if it achieves a comparable performance, NBC will be more desirable because of its high speed.

In summary, we should not prefer any classification method if it outperforms others in one context since it might fail severely in another one. (THIS IS NORMAL IN DATA MINING PROBLEMS).

No free lunch in search and optimization

In computational complexity and optimization the no free lunch theorem is a result that states that for certain types of mathematical problems, the computational cost of finding a solution, averaged over all problems in the class, is the same for any solution method. No solution therefore offers a "short cut".

Thank You