ME308

IEOR Course Project

Project Management Using ML and OR

Team Code: 15

Arpit Tiwari	190110011
Sumit Chandrakant Bhong	190100121
Vipul Kumar Singh	190100137
Mudit Rathore	190100078

Instructors :

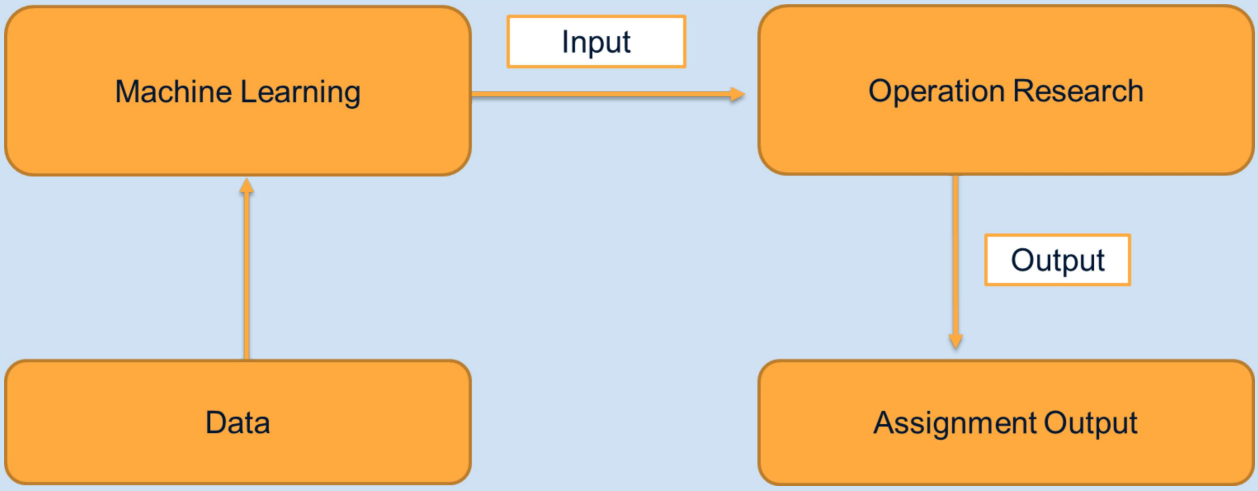
Prof. Makarand Kulkarni

Prof. Avinash Bhardwaj

Introduction /Problem Statement:

In any major software firm today, manual allotment of software issues/bugs to the employees is not a feasible solution. The allotment task should be done successfully and effectively in terms of time and also maximizing the quality.
We are dealing with the Project management issue allocation problem to the various employees in a software firm. And to facilitate and augment the quality of the Project management practice, we propose a hybrid approach that builds on the synergy between contemporary ML and OR techniques.

Basic Approach:



Data Attributes:

We are interested in only **Assignee**, **Summary** and **Description**. We will use this data to apply NLP (Natural Language Processing) to get the input for Optimizing the bug allotment.

Attribute	Description	Values	Attribute type
Id	The identifier of the issue	*e.g. {1345}*	Integer
Key	The textual identifier of the issue	*e.g. {HADOOP-1345}*	String
Labels	The labels of the issue	*e.g. {KeyStore, security, tpm}*	String
Assignee	The assignee of the issue	*e.g. {john_doe, jane_doe}*	String
Status	The project's current status	*{Close, In Progress, Open, Patch Available, Reopened, Resolved}*	String
Components	The parental architectural components of the module that concerns the issue	*e.g. {Back-end, Front-end, Main-Framework}*	String
Description	The description of the issue	*Unstructured text*	String
Summary	The title of the issue	*Unstructured text*	String
Reporter	The reporter of the issue	*e.g. {john_doe, jane_doe}*	String
Resolution Date	The resolution date of the issue	e.g. {1560771216}	Timestamp
Created at	The date the issue has been created	*e.g. {1560771216}*	Timestamp

Example Issue:

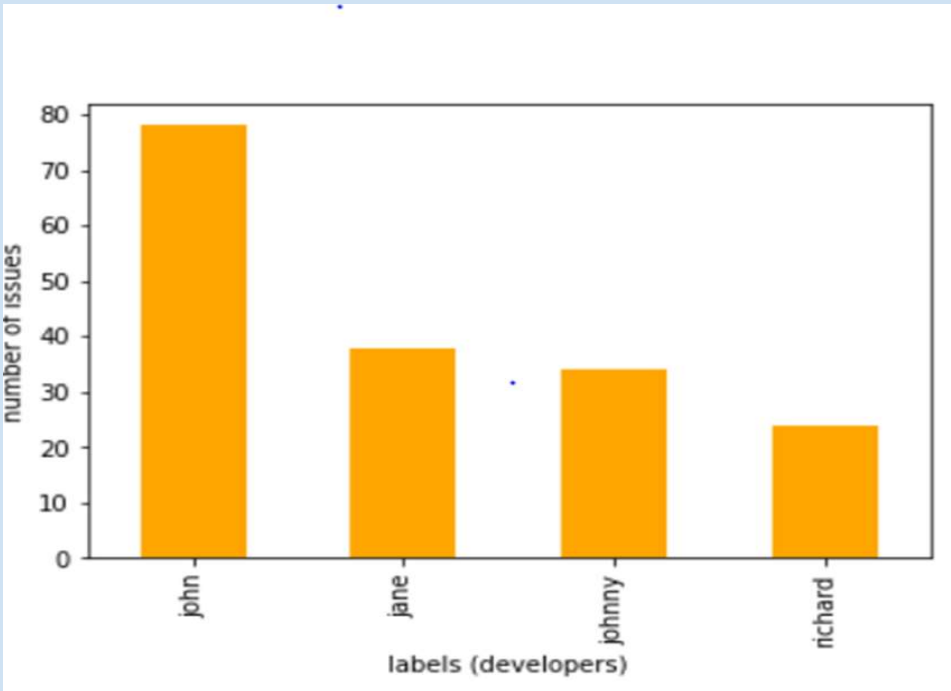
- "assignee": {
- "self": "https://issues.apache.org/jira/rest/api/2/user?username=Jack-Lee",
- "name": "Jack-Lee",
- "key": "jack-lee", }

- "summary": "Upgrade Jackson2 to the latest version",

- "description": "Now Jackson 2.9.5 is used and it is vulnerable (CVE-2018-11307). Let's upgrade to 2.9.6 or upper.",

Data Processing:

We had an Initial Data size of **1000 Issues.** Of which we removed the ones, which did not have an assignee, giving an intermediate data size of 647 issues. We then only choose statistically significant data,i.e., taking issues for only the top 4 assignee(s), giving a final data size of **174 issues.**



Natural Language Processing (NLP)

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

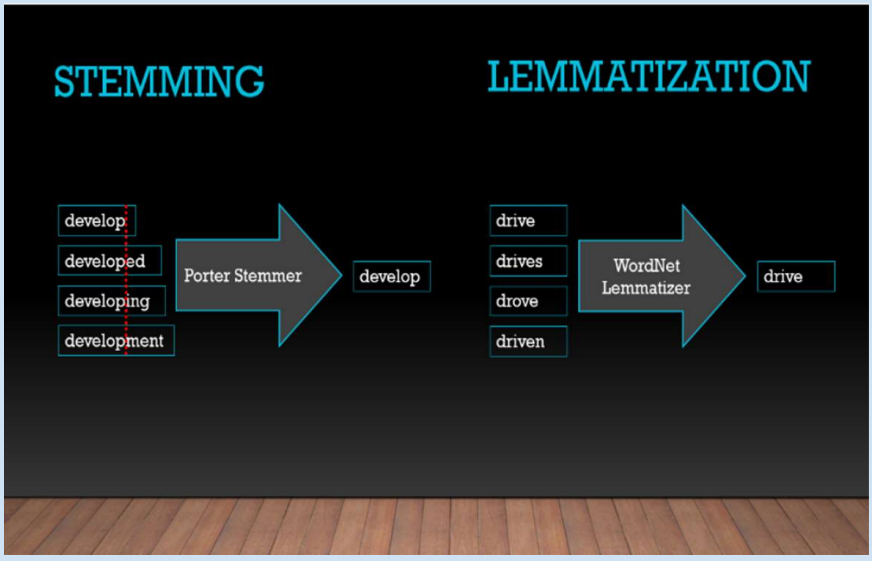
NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real-time. There's a good chance you've interacted with NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

Using NLP in our Model:

To extract features a variety of NLP techniques, including feature generation and text normalization, are applied to the 'summary' and 'description' given in data by:

- **Lemmatization:** In NLP linguistics is the process of grouping together the inflected forms of a word so they can be analyzed as a single item, identified by the word's lemma, or dictionary form
- **Stemming:** Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the



roots of words known as a lemma.

- **Removal of stop words:** the process of removing commonly used words, which usually add noise to the ML models
- **Removal of frequently occurred words:** the process of removing insignificant words from a text, taking into consideration the document frequency value of each word
- **Tokenization:** the process of generating tokens from unstructured text

Machine Learning Models

Naïve Bayes

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

Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using the Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is, the presence of one particular feature does not affect the other. Hence it is called naive.

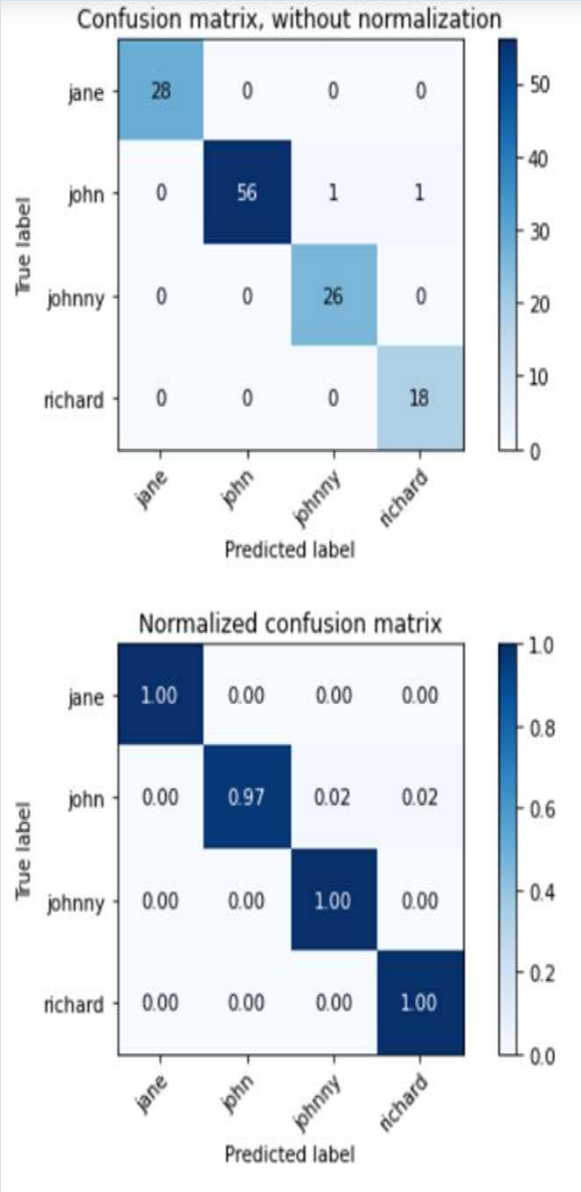
Types of Naive Bayes Classifier:

Multinomial Naive Bayes:
This is mostly used for document classification problems, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

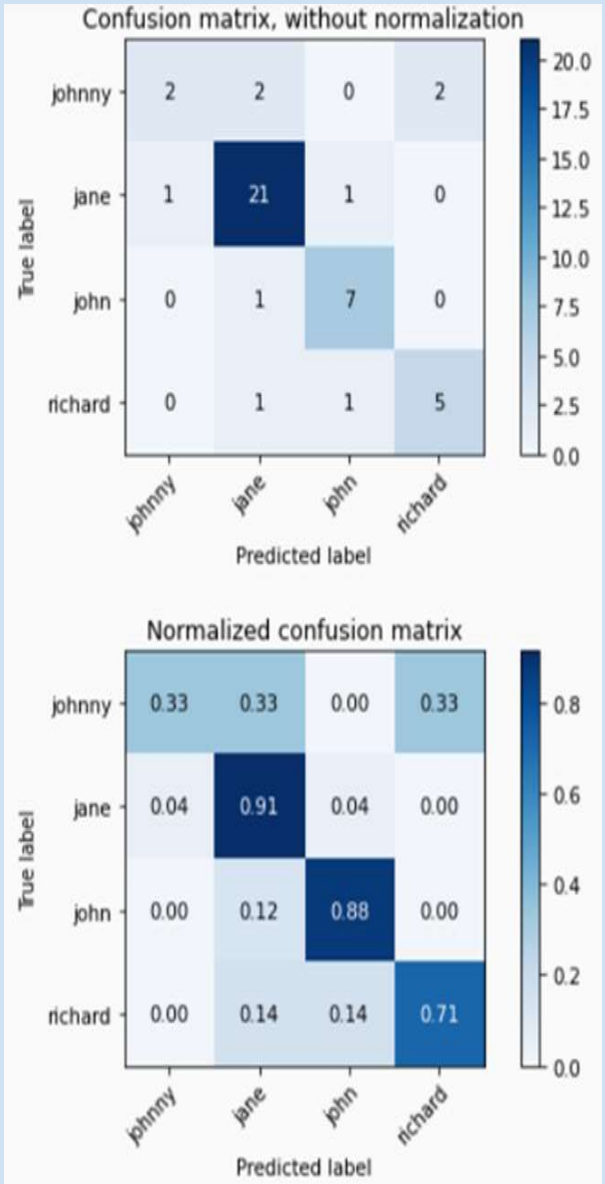
Bernoulli Naive Bayes:
This is similar to the multinomial naïve Bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values of yes or no, for example, if a word occurs in the text or not.

Gaussian Naive Bayes:
When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.

Prediction and Results Using Naive Bayes



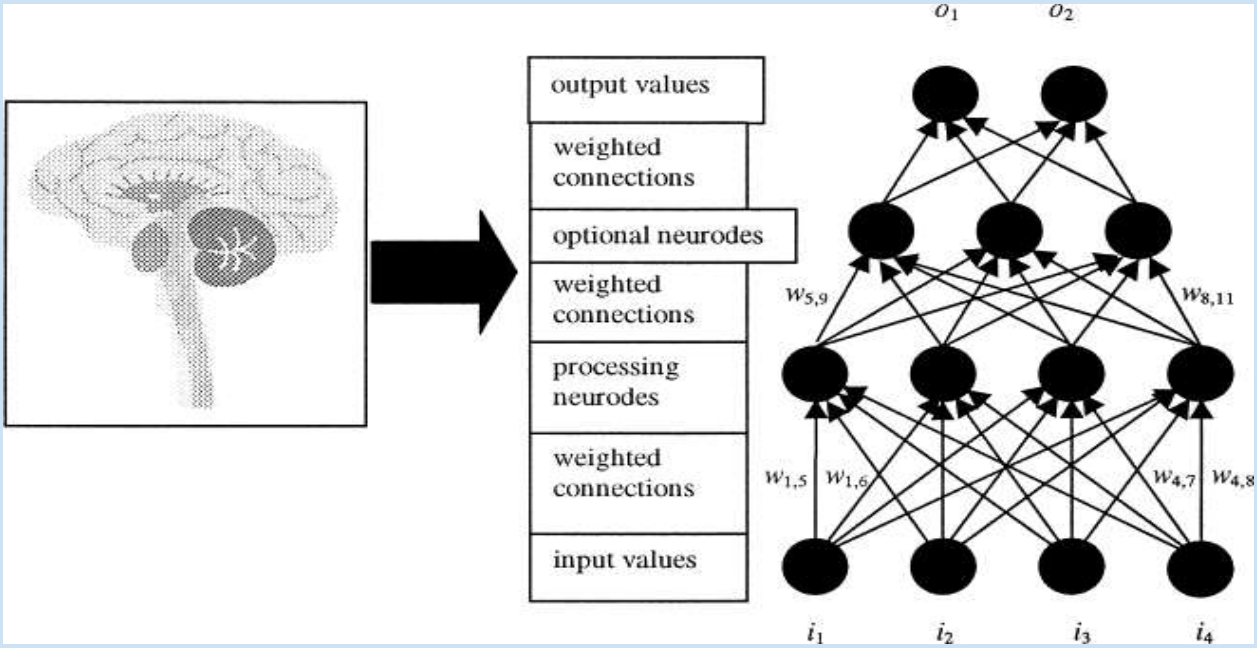
Training Data



Test Data

```
Accuracy on the Test Dataset is: 0.7954545454545454
Accuracy on the Train Dataset is: 0.9692307692307692
```

Neural Networks Based Approach



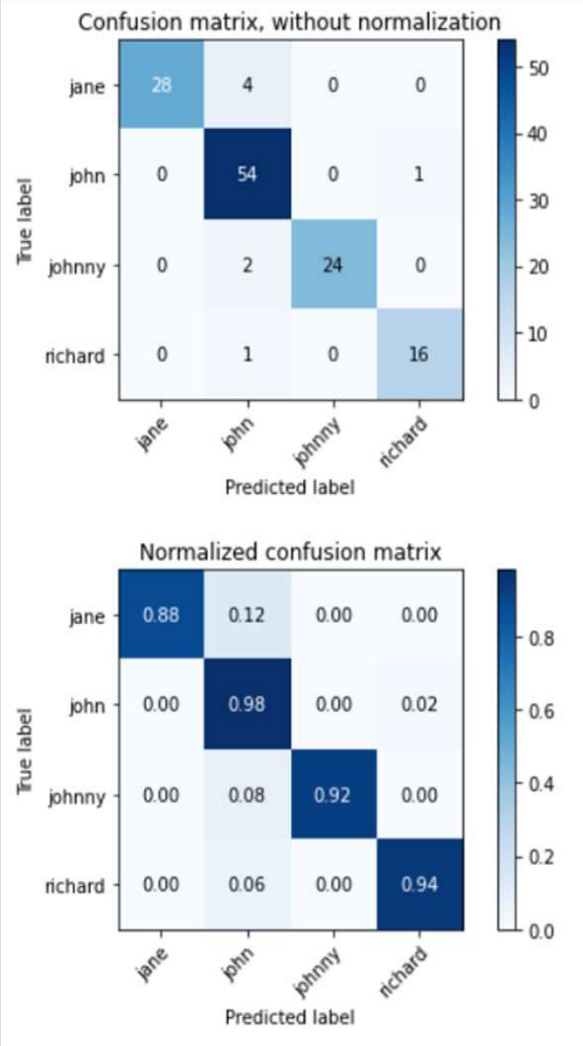
Our Model Architecture

Layer (type)	Output Shape	Param #
dense_22 (Dense)	(None, 512)	1230848
activation_16 (Activation)	(None, 512)	0
dropout_18 (Dropout)	(None, 512)	0
dense_23 (Dense)	(None, 512)	262656
activation_17 (Activation)	(None, 512)	0
dropout_19 (Dropout)	(None, 512)	0
dense_24 (Dense)	(None, 512)	262656
dropout_20 (Dropout)	(None, 512)	0
dense_25 (Dense)	(None, 512)	262656
dropout_21 (Dropout)	(None, 512)	0
dense_26 (Dense)	(None, 4)	2052
activation_18 (Activation)	(None, 4)	0

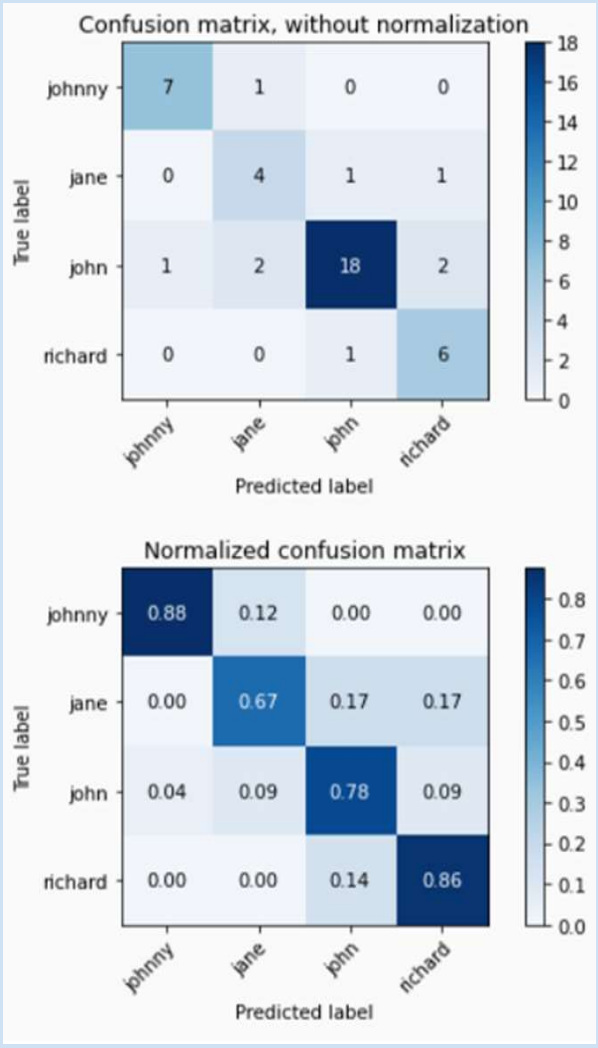
Total params: 2,020,868
Trainable params: 2,020,868
Non-trainable params: 0

Prediction and Results Using NN

Training Data



Test Data




```
Accuracy on the Test Dataset is: 0.7954545617103577
Accuracy on the Train Dataset is: 0.9384615421295166
```

SVM (Support Vector Machines)

We will be using SVM for multiclass classification to get the relevance matrix.

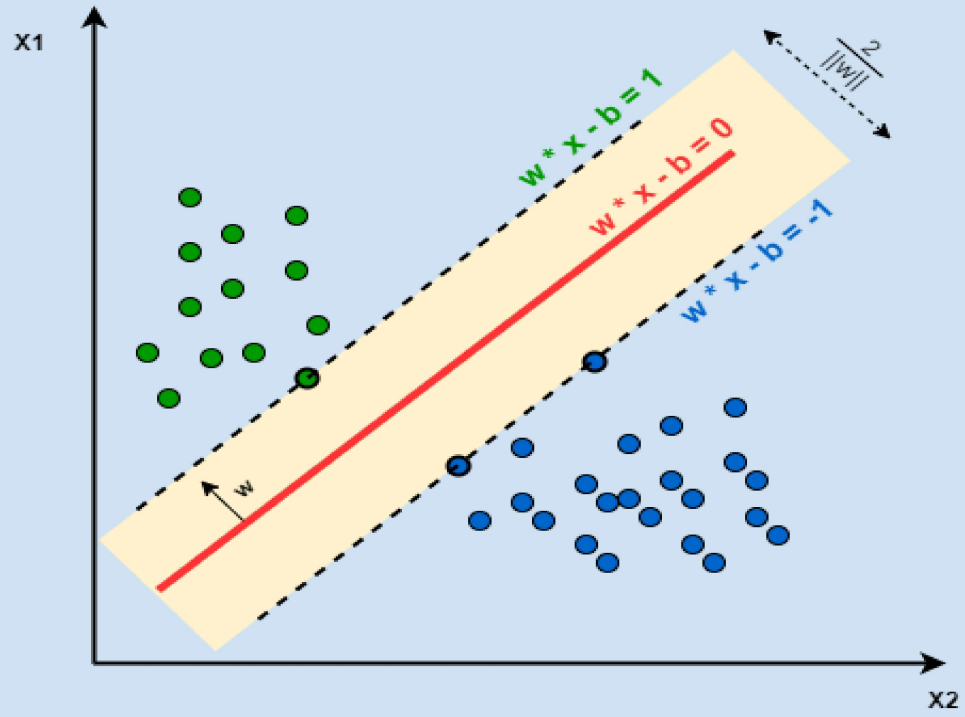
In this type, the machine should classify an instance as only one of three classes or more. The following are examples of multiclass classification:

- Classifying a text as positive, negative, or neutral
- Determining the dog breed in an image
- Categorizing a news article as sports, politics, economics, or social

How does it Work?

In the base form, linear separation, SVM tries to find a line that maximizes the separation between a two-class data set of 2-dimensional space points. To generalize, the objective is to find a hyperplane that maximizes the separation of the data points to their potential classes in an n-dimensional space. The data points with the minimum distance to the hyperplane (closest points) are called Support Vectors.

In the image below, the Support Vectors are the 3 points (2 blue and 1 green) laying on the scattered lines, and the separation hyperplane is the solid red line:



Multiclass Classification Using SVM

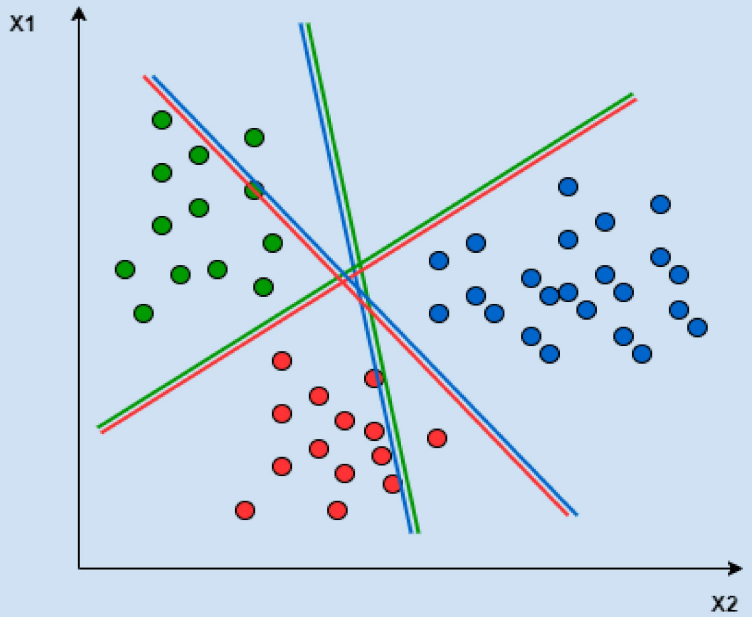
In its most simple type, SVM doesn't support multiclass classification natively. It supports binary classification and separating data points into two classes. For multiclass classification, the same principle is utilized after breaking down the multi-classification problem into multiple binary classification problems.

The idea is to map data points to high dimensional space to gain mutual linear separation between every two classes. **This is called a One-to-One approach, which breaks down the multiclass problem into multiple binary classification problems. A binary classifier per each pair of classes.**

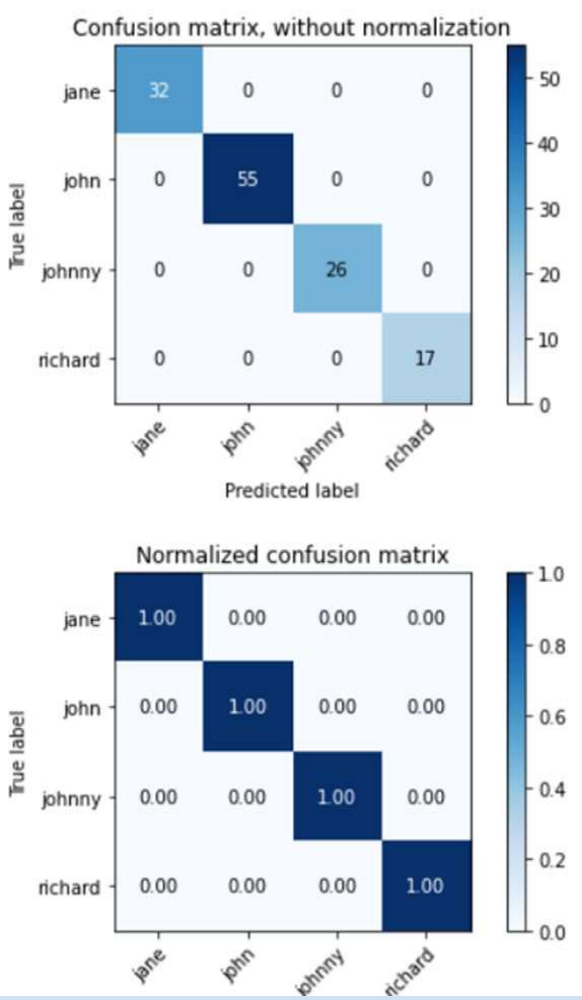
Another approach one can use is One-to-Rest. In that approach, the breakdown is set to a binary classifier per class.

A single SVM does binary classification and can differentiate between two classes. So that, according to the two breakdown approaches, to classify data points from classes data set:

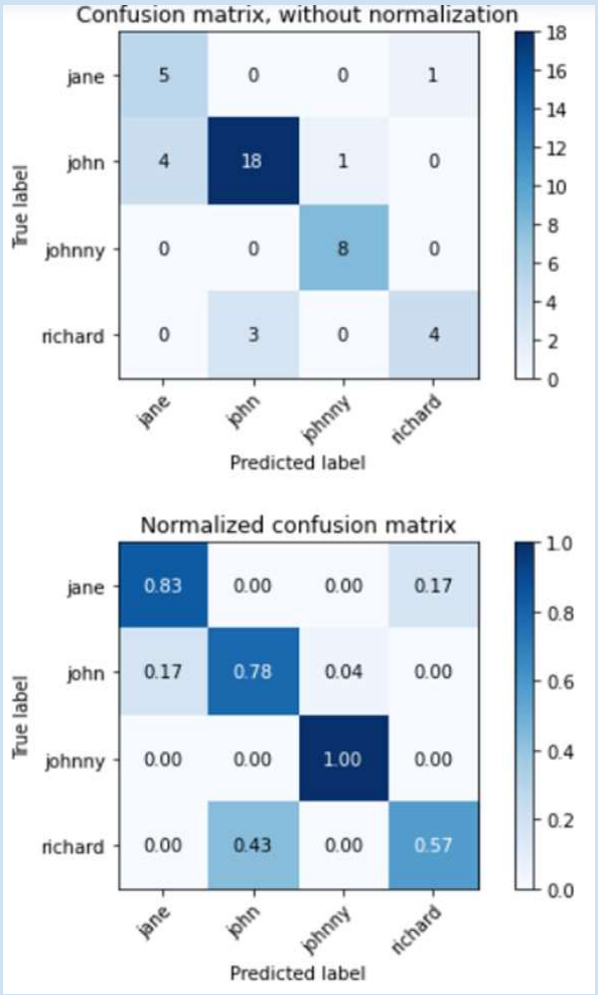
- In the One-to-Rest approach, the classifier can use SVMs. Each SVM would predict membership in one of the classes.
- In the One-to-One approach, the classifier can use SVMs.



Prediction and Results Using SVM



Training Data



Test Data

```
Accuracy on the Test Dataset is: 0.7954545454545454
Accuracy on the Train Dataset is: 0.9692307692307692
```

Final Relevance Matrix Using SVM

	Jane	John	Jhonny	Richard
0	105.000000	5.000000	5.000000	5.000000
1	104.999994	5.000000	5.000000	5.000006
2	105.000000	5.000000	5.000000	5.000000
3	92.402310	6.118143	5.001440	16.478107
4	52.629130	5.057578	5.000003	57.313288
5	70.498275	5.000011	39.501714	5.000000
6	5.000076	104.993259	5.006665	5.000000
7	15.436097	92.893406	6.668463	5.002034
8	104.999982	5.000003	5.000014	5.000000
9	104.982891	5.014169	5.000000	5.002940
10	5.000846	5.000000	104.999153	5.000001
11	10.123514	99.706409	5.170078	5.000000
12	5.061444	5.000000	104.938556	5.000000
13	104.891679	5.088243	5.020078	5.000000
14	104.998691	5.000000	5.000000	5.001309
15	5.023581	5.005677	104.970480	5.000262
16	104.998051	5.000234	5.000019	5.001696
17	5.000000	5.000000	5.000000	105.000000
18	103.696826	5.000000	6.303174	5.000000
19	104.999033	5.000000	5.000000	5.000967

Linear Programming Formulation:

- Sets and Indices:
  - I : set of Issues
    - $\{issue\_1, issue\_2, \ldots\ldots issue\_n\}$
  - J : set of Coders
    - $\{coder\_1, coder\_2, coder\_3, \ldots\ldots coder\_m\}$
  - i : Index in set I : $i \in I$
  - j : Index in set J : $j \in J$
- Parameters:
  - **Relevance Matrix : R_{ij} $\forall$ i,j $\in$ I, J**
    - This gives the relevance of coder j to solve issue i
  - **Capacity of coder: Capacity [j] $\forall$ j$\in$ J**
    - Maximum number of issues a coder j can solve
  - **Time required to solve an issue: time_issue[i] $\forall\ i \in I$**
    - We have randomly added time values within a certain range to model this complexity
  - **Threshold time: thresh_time[j] $\forall$ j$\in$J**
    - Depending on the skills of the coder we assign a threshold time for each coder
    - A coder will solve only those issue which requires more time than the threshold time
- Decision Variables :
  - **α_{ij} = Binary variable**
    - **1** if issue i is assigned to coder j
    - **0** if issue i is not assigned to coder j
- Objective Function :
  - Maximize the total relevance:

$$Max\left[\sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_{ij}R_{ij}\right]$$

- Constraints :
  - **Capacity constraint :** Total number of issues assigned to coder j must be less than capacity[j].

$$\sum_{i=1}^{n}\alpha_{ij} \leq Capacity\,[j]\ \forall\ j\ \in J$$

  - Each issue must be assigned to only one coder:

$$\sum_{i=1}^{m} \alpha_{ij} = 1 \; \forall \, i \; \in I$$

- Coder j must get only those issues, which require more time than thresh_time[j]

$$Time\ Issue\ [i] \; \geq \alpha_{ij} \; \times \; threshold\ time[j]$$

$$\forall \, i, j \; \in I, J \; respectively$$

AMPL code snippets :

Model:

```
set coders ordered;
set issues ordered;

param capacity{j in coders};
param thresh_time{ j in coders};

param time_issue{i in issues};

param relevance{i in issues, j in coders};

var alpha{i in issues, j in coders}  binary;

maximize totRel: sum{i in issues, j in coders} relevance[i,j]*alpha[i,j];

subject to con1{j in coders}: sum{i in issues} alpha[i,j]<=capacity[j];
subject to con2{i in issues}: sum{j in coders} alpha[i,j]=1;

subject to con3{i in issues, j in coders} : time_issue[i] >= alpha[i,j]*thresh_time[j];
```

Importing Data:

```
reset;
load amplcsv.dll;
model "C:\Users\ROG\OneDrive\Desktop\Final\model3.mod";

table Coders IN "amplcsv" "C:\Users\ROG\OneDrive\Desktop\Final\Coders.csv": coders <- [coders], capacity,thresh_time;
table Issues IN "amplcsv" "C:\Users\ROG\OneDrive\Desktop\Final\issues.csv": issues <- [issues], time_issue;
table Relevance IN "amplcsv" "C:\Users\ROG\OneDrive\Desktop\Final\Relevance_delta.csv": [issues, coders], relevance;


read table Coders;
read table Issues;
read table Relevance;

option solver "E:\Sem 6\ME308\Ampl\ampl.mswin64\cplex.exe";
solve;

table Results OUT "amplcsv" "C:\Users\ROG\OneDrive\Desktop\Final\result.csv":
      [issues,coders], alpha;

write table Results;

#display alpha;
```

Solution of Optimization:

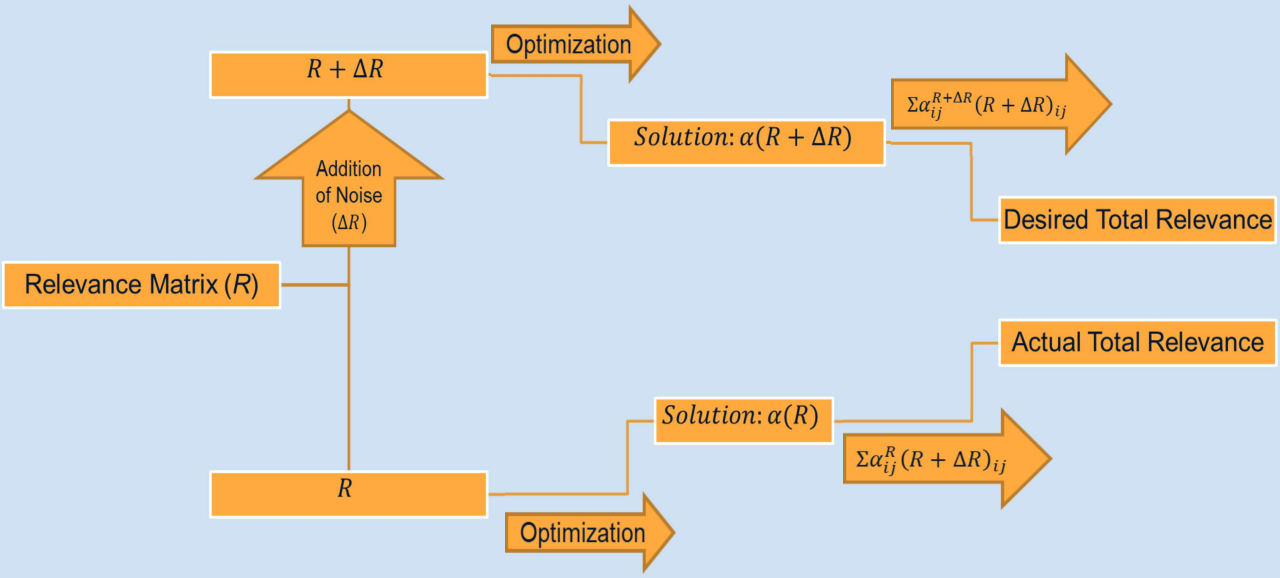
	Jane	John	Jhonny	Richard
0	0	1	0	0
1	0	1	0	0
2	0	1	0	0
3	1	0	0	0
4	1	0	0	0
5	0	0	0	1
6	0	0	1	0
7	0	0	1	0
8	0	1	0	0
9	0	1	0	0
10	0	0	0	1
11	0	0	1	0
12	0	0	0	1
13	0	0	0	1
14	0	1	0	0
15	0	0	0	1
16	0	1	0	0
17	1	0	0	0
18	0	0	0	1
19	0	1	0	0
20	1	0	0	0
21	0	0	0	1
22	0	0	1	0
23	0	0	0	1
24	0	0	1	0
25	0	0	0	1
26	1	0	0	0
27	0	0	0	1
28	0	0	1	0
29	0	1	0	0
30	0	1	0	0
31	0	1	0	0
32	0	0	1	0
33	0	1	0	0
34	0	1	0	0
35	1	0	0	0
36	0	1	0	0
37	1	0	0	0
38	0	0	1	0
39	0	0	0	1
40	0	0	1	0
41	0	0	0	1
42	0	1	0	0
43	0	0	1	0

```
ampl: include model3.run;
CPLEX 20.1.0.0: optimal integer solution; objective 2877.106307
```

Coders	Capacity
Jane	10
John	15
Johnny	10
Richard	20

	Jane	John	Johnny	Richard	Total
Number of issues assigned	7	15	10	12	44

Error Analysis:



We added some noise in the original relevance matrix to model uncertainty in the relevance matrix. Then we applied an optimization model to the new relevance matrix and found the solution. This optimization will give some optimal value of total relevance which we are calling Desired total relevance. Also, optimization on the original relevance matrix gives us some solution (α), using which we calculate total relevance by applying it to the new relevance matrix (R + ΔR) which is actually total relevance as shown in the above flow chart.

- **Addition of Noise (ΔR):**

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

  - **Gaussian Function:**
    - We used a Gaussian function for the creation of a noise matrix.
    - By changing the values of σ *and* μ, we calculated the optimal solution (desired total relevance) for each R+ΔR relevance matrix.
    - Also calculated the optimal total relevance (actual total relevance) by using the original solution for matrix R and calculated % error in actual total relevance and desired total relevance.

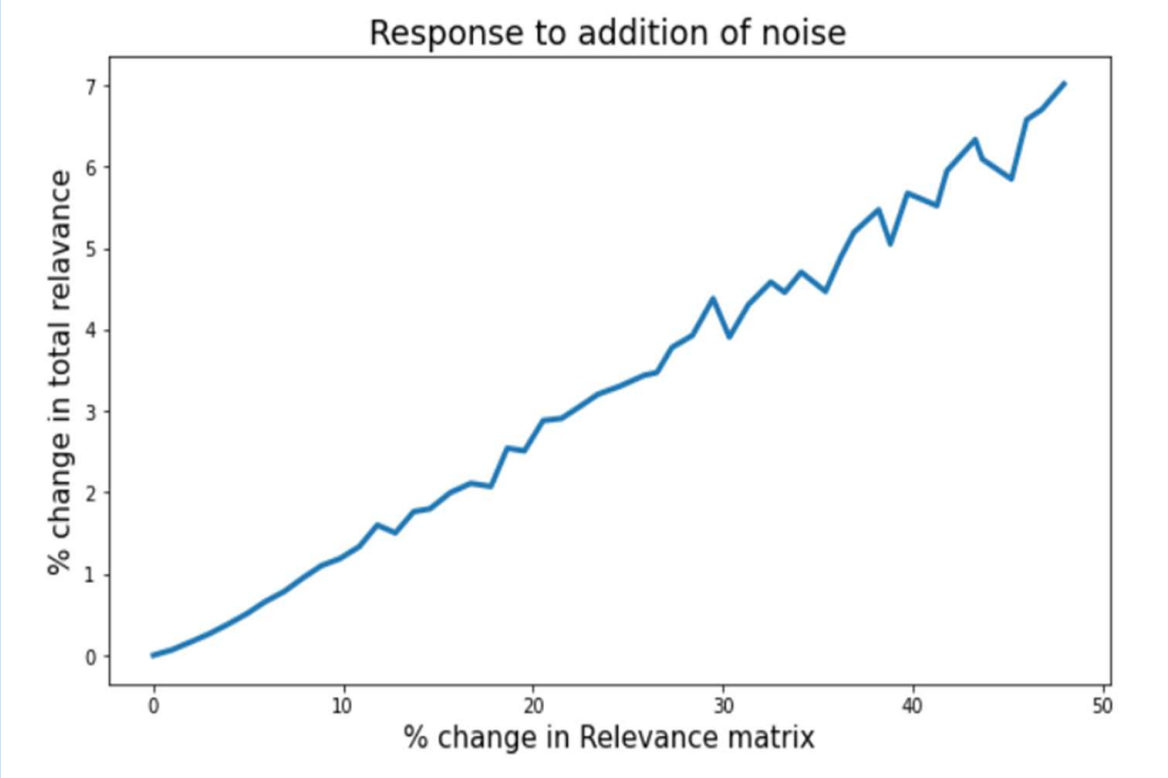
- **% Change in Relevance Matrix**:
  - We used the Frobenius norm to quantify the noise:

$$\frac{||\Delta R||}{||R||} \times 100$$

- % Change in Objective:

$$= \frac{Relevance_{desired} - Relevance_{actual}}{Relevance_{actual}} \times 100$$

- Analysis:
  - Below is the plot of % change in total relevance vs % change in relevance matrix.
  - As our ML model gave a relevance matrix with an accuracy of **80 %**, we can assume that uncertainty in the relevance matrix can be up to **20 %.**
  - From the plot, we can see that for **20 %** of uncertainty in the relevance matrix, our model gives only **3 %** of the change in objective.



Future Work:

1. To increase the accuracy of the Relevance matrix.
2. Auto Assignment Problem: Adding a dynamic element to our model to make it automated
    - If any coder is not available at a given time then we need to optimize the assignment by considering only available coders.

References:

- https://dspace.mit.edu/handle/1721.1/122099
- https://www.coursera.org
- https://www.analyticsvidhya.com/blog/2021/11/a-guide-to-building-an-end-to-end-multiclass-text-classification-model/
- https://www.baeldung.com/cs/svm-multiclass-classification
- **https://towardsdatascience.com/the-complete-guide-to-neural-networks-multinomial-classification-4fe88bde7839**
- ME308 lectures by Prof. Makarand Kulkarni and Prof. Avinash Bhardwaj

Github link:

https://github.com/arpittiwari11/Project-Management-using-NLP-and-Operation-Research

Libraries used :

```
import json
import pandas as pd


from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

#Libraries for NLP
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()

from nltk.stem import LancasterStemmer
stemmer = LancasterStemmer()
nltk.download('omw-1.4')

#Lime text explainer for explaining the model
from lime.lime_text import LimeTextExplainer

# For error
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

%matplotlib inline
# Operations Research Libraries
from ortools.graph import pywrapgraph
```