CHAPTER - 1

INTRODUCTION

Recruiters receive a massive influx of resumes when they post a job opening online. This means employers should quickly and efficiently evaluate resumes to find the right among them. One way organizations can do this is by using resume screening tools.

Resume screening is an automated process to scan through applications or CVs with keywords related to the job description requirements and then determine if applicants meet those qualifications from their responses on their documents. Companies often use applicant tracking systems (ATS), including explicitly designed algorithms. By doing so, employers can easily filter out unqualified or underqualified candidates without manually reading each CV.

The ATS looks for things like specific degrees, certifications, programming languages, or soft skills that match what the company needs in its roles, such as leadership qualities or communication ability – allowing recruiters/hiring managers to focus time spent reviewing only qualified ones during the shortlisting stage.

We know that key plays a vital role in any encryption processes and security of cipher depend on the quality of the key. For that new key generation algorithm for image encryption .If we design the key to be applied on an image that will largely decrease the correlation among the image elements, and then the generated cipher image will be more protected. It uses Block based transformation algorithm using chaos mapping, where the mapping is derived by applying one important artificial intelligence procedure Genetic Algorithm and the algorithm hides the original image through simple permutation of the pixel location.
Resume screening is the initial step in the hiring process where recruiters or hiring managers review job applications to determine which candidates possess the qualifications and experience necessary for the position. It involves quickly scanning resumes to identify key skills, experience, education, and other relevant factors that match the job requirements.

The goal of resume screening is to efficiently narrow down the pool of applicants to a manageable number for further consideration, typically through interviews. This process helps organizations save time and resources by focusing on candidates who are the best fit for the role. Automated resume screening software is often used to streamline this process, using algorithms to filter resumes based on predefined criteria. However, manual screening by human recruiters is also common, particularly for roles requiring nuanced judgment or subjective qualifications. Effective resume screening involves balancing speed with accuracy, ensuring that qualified candidates are not overlooked while also efficiently eliminating those who do not meet the minimum requirements. It requires a thorough understanding of the job description and the ability to quickly assess candidate qualifications based on their resumes.

CHAPTER -2

LITERATURE SURVEY

Literature surveys are mainly carried out in order to analyse the background of the current proj ect, which helps to find out flaws in the existing system and guides on which unsolved proble ms we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated us to propose solutions and work on project.

So, first let's talk about the importance of screening of resume in selection process as it exclud es all the irrelevant and unqualified candidates. Ugbah and Majors in 1992 reported that when selecting a fresher for a job, recruiters believed that applicant have great paper credentials suc h as resume, relevant work experience and right education followed by applicant's behaviour d uring interview and when using applicants' resumes screening medium, recruiters thinks that i nformation provided on the resume is linked to important, job-relevant attributes or personalit y characteristics. Because applicants typically mail or send their resumes electronically, if resu me information were linked to these important applicant constructs, substantial savings could accrue to an organization before investing in more-expensive, time-consuming selection techni ques requiring on-site applicant presence. [1]

The linkage between quality of hiring and organizational outcomes is well established (Boudre au & Ramstad, 1996; Erickson, Lamoureux, & Moulton, 2014, La2014. Lawler Organizations can reasonably expect that greater levels of hiring success will contribute to improved organiz ation-level outcomes such as profitability (Erickson et al., 2014). A variety of approaches may be taken by employers in order to improve their hiring success including improvements in app licant sources, recruiting practices, selection methods, selection criteria, and onboarding (Adki ns, 1995; Carr, Pearson, Vest, & Boyar, 2006; Rynes & Cable, 2003; Russell, 2007). Sackett a nd Lievens (2008) identified five strategies that may be utilized by organizations to improve hi ring success through selection system enhancements (i.e., enhancements of criteria and metho ds): (a) measure the same construct (e.g., personality) with another selection method, (b) impr ove construct measurement, (c) improve contextualization of measurement (e.g., ensure that 1 5 scales are work-specific), (d) reduce response distortion when using self-report instruments, and (e) impose a greater level of structure in the use of existing selection methods. This study provides an initial examination of resume screening processes and criteria for managerial appli cants that could be used to implement the strategies outlined by Sackett & Lievens (2008)[3].

The first resume parsers were born within the late '90s to produce an information structuring te chnology to HR software companies that are searching for a stand-alone packaged solution to c oncentrate on their core business. A number of these first-mover solutions are: Sovren (1996) TextKernel (2001) Daxtra (2002) How Daxtra , Sovren , Hireability, Textkernel and Segmentr (by Riminder) do at this task? Building a general and reliable parser requires many building bl ocks. For instance, the system should be ready to handle: complex layouts (ex: multi-column r esumes, pictures with backgrounds, etc.) ambiguous entities (ex: Facebook, as a former emplo yer vs. a social media skill) different media formats (PDF, Word, Image, etc.) Multiple langua ges etc.[4].

Author of [1] discusses Web application for resume screening. Use of NLP pipeline. Text Extr action is done using sections based segmentation. Semi supervised learning to train machine le arning model. The system has some drawbacks.

The web application shows results at the recruiter side in the form of ranking. One of the featu re can be described as Resume is only matched to those job openings which they are interested in and have applied for. The author of [2] makes use of OCR (Optical Character Recognition) Feature Extraction of Principal Component Analysis. Decision tree classification algorithm is used. But the system works only on Urdu Text.

Paper [3] mentions that Conventional SVM can be optimized for text classification. The conv entional SVM is optimized by selecting features using the entropy. The author of [4] compares three classification algorithms. Results show that support vector classifiers with the TF/IDF fe ature shows more accuracy than naive bayes and KNN. The author of [5] presents the web doc ument classification based on fuzzy k-NN network, in the process of classification, TF/IDF is a dopted for selecting features of document. The results show that classification performance is better than k-NN and SVM, but the speed of classification is bit slow than KNN.

CHAPTER-3

KNN Algorithm

The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

While the KNN algorithm can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered "plurality voting", the term, "majority vote" is more commonly used in literature. The distinction between these terminologies is that "majority voting" technically requires a majority of greater than 50%, which primarily works when there are only two categories. When you have multiple classes—e.g. four categories, you don't necessarily need 50% of the vote to make a conclusion about a class; you could assign a class label with a vote of greater than 25%. The University of Wisconsin-Madison summarizes this well with an example here (link resides outside ibm.com).

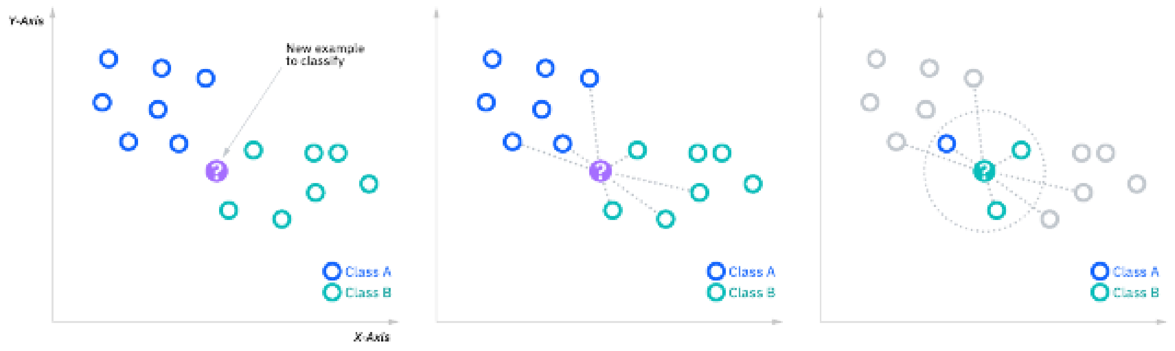


Fig1: Illustration of a graph that represents the K-Nearest Neighbors Algorithm

KNN diagram

Regression problems use a similar concept as classification problem, but in this case, the average the k nearest neighbors is taken to make a prediction about a classification. The main distinction here is that classification is used for discrete values, whereas regression is used with continuous ones. However, before a classification can be made, the distance must be defined. Euclidean distance is most commonly used, which we'll delve into more below.

It's also worth noting that the KNN algorithm is also part of a family of "lazy learning" models, meaning that it only stores a training dataset versus undergoing a training stage. This also means that all the computation occurs when a classification or prediction is being made. Since it heavily relies on memory to store all its training data, it is also referred to as an instance-based or memory-based learning method.

Evelyn Fix and Joseph Hodges are credited with the initial ideas around the KNN model in this 1951 paper (link resides outside ibm.com) while Thomas Cover expands on their concept in his research (link resides outside ibm.com), "Nearest Neighbor Pattern Classification." While it's not as popular as it once was, it is still one of the first algorithms one learns in data science due to its simplicity and accuracy. However, as a dataset grows, KNN becomes increasingly inefficient, compromising overall model performance. It is commonly used for simple recommendation systems, pattern recognition, data mining, financial market predictions, intrusion detection, and more.

Ebook

Build responsible AI workflows with AI governance

Learn the building blocks and best practices to help your teams accelerate responsible AI.

Related content

Register for the white paper on AI governance

Compute KNN: distance metrics

To recap, the goal of the k-nearest neighbor algorithm is to identify the nearest neighbors of a given query point, so that we can assign a class label to that point. In order to do this, KNN has a few requirements:

Determine your distance metrics

In order to determine which data points are closest to a given query point, the distance between the query point and the other data points will need to be calculated. These distance metrics help to form decision boundaries, which partitions query points into different regions. You commonly will see decision boundaries visualized with Voronoi diagrams.

While there are several distance measures that you can choose from, this article will only cover the following:

Euclidean distance (p=2): This is the most commonly used distance measure, and it is limited to real-valued vectors. Using the below formula, it measures a straight line between the query point and the other point being measured.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Fig 2:Euclidean distance formula

Manhattan distance (p=1): This is also another popular distance metric, which measures the absolute value between two points. It is also referred to as taxicab distance or city block distance as it is commonly visualized with a grid, illustrating how one might navigate from one address to another via city streets.

$$\text{Manhattan Distance} = d(x,y) = \left(\sum_{i=1}^{m} \left|x_i - y_i\right|\right)$$

Fig 3 :Manhattan distance formula

Manhattan distance formula

Minkowski distance: This distance measure is the generalized form of Euclidean and Manhattan distance metrics. The parameter, p, in the formula below, allows for the creation of other distance metrics. Euclidean distance is represented by this formula when p is equal to two, and Manhattan distance is denoted with p equal to one.

$$\text{Minkowski Distance} = \left(\sum_{i=1}^{n} \left|x_i - y_i\right|\right)^{1/p}$$

Fig 4: Minkowski distance formula

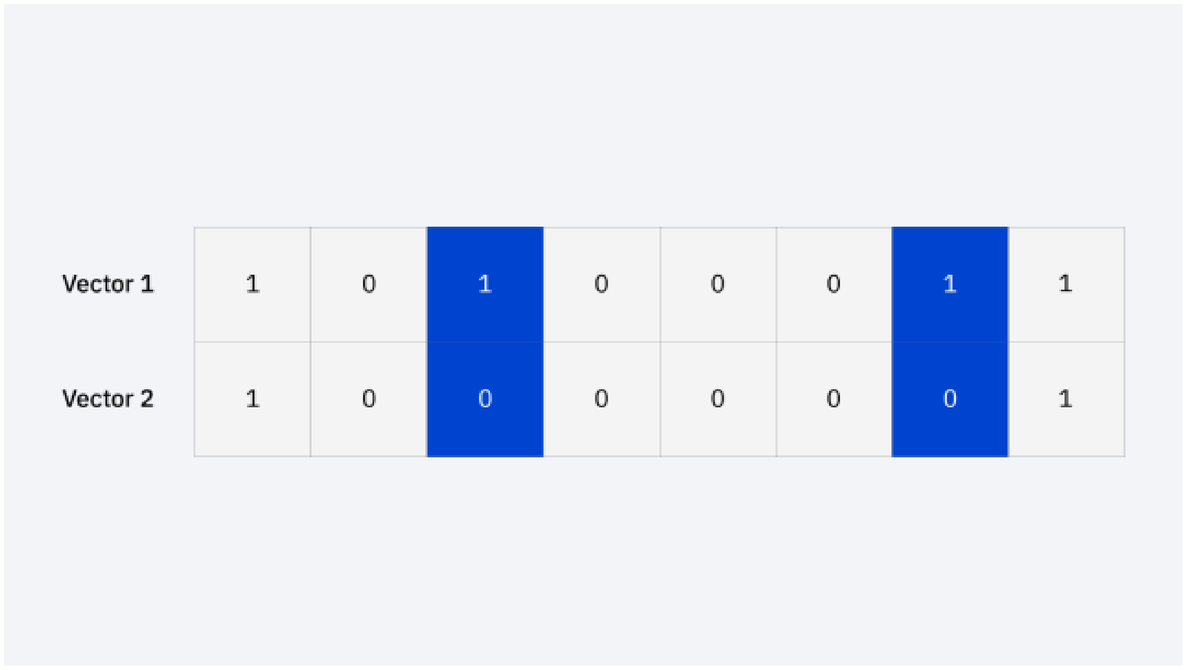
Hamming distance: This technique is used typically used with Boolean or string vectors, identifying the points where the vectors do not match. As a result, it has also been referred to as the overlap metric. This can be represented with the following formula:

$$\text{Hamming Distance} = D_H = \left(\sum_{i=1}^{k} |x_i - y_i| \right)$$

$$x = y \qquad D = 0$$
$$x \neq y \qquad D \neq 1$$

Fig 4:Hamming distance formula

As an example, if you had the following strings, the hamming distance would be 2 since only two of the values differ.



Vector 1	1	0	1	0	0	0	1	1
Vector 2	1	0	0	0	0	0	0	1

Fig 5:Hamming distance example

Compute KNN: defining k

The k value in the k-NN algorithm defines how many neighbors will be checked to determine the classification of a specific query point. For example, if k=1, the instance will be assigned to the same class as its single nearest neighbor. Defining k can be a balancing act as different values can lead to overfitting or underfitting. Lower values of k can have high variance, but low bias, and larger values of k may lead to high bias and lower variance. The choice of k will largely depend on the input data as data with more outliers or noise will likely perform better with higher values of k. Overall, it is recommended to have an odd number for k to avoid ties in classification, and cross-validation tactics can help you choose the optimal k for your dataset.

k-nearest neighbors and python

To delve deeper, you can learn more about the k-NN algorithm by using Python and scikit-learn (also known as sklearn). Our tutorial in Watson Studio helps you learn the basic syntax from this library, which also contains other popular libraries, like NumPy, pandas, and Matplotlib. The following code is an example of how to create and predict with a KNN model:

```
from sklearn.neighbors import KNeighborsClassifier

model_name = ‘K-Nearest Neighbor Classifier’

knnClassifier = KNeighborsClassifier(n_neighbors = 5, metric = ‘minkowski’, p=2)

knn_model = Pipeline(steps=[(‘preprocessor’, preprocessorForFeatures), (‘classifier’ , knnClassifier)])

knn_model.fit(X_train, y_train)

y_pred = knn_model.predict(X_test)
```

Applications of k-NN in machine learning

The k-NN algorithm has been utilized within a variety of applications, largely within classification. Some of these use cases include:

- Data preprocessing: Datasets frequently have missing values, but the KNN algorithm can estimate for those values in a process known as missing data imputation.

- Recommendation Engines: Using clickstream data from websites, the KNN algorithm has been used to provide automatic recommendations to users on additional content. This research (link resides outside ibm.com) shows that the a user is assigned to a particular group, and based on that group's user behavior, they are given a recommendation. However, given the scaling issues with KNN, this approach may not be optimal for larger datasets.

- Finance: It has also been used in a variety of finance and economic use cases. For example, one paper (link resides outside ibm.com) shows how using KNN on credit data can help banks assess risk of a loan to an organization or individual. It is used to determine the credit-worthiness of a loan applicant. Another journal (link resides outside ibm.com) highlights its use in stock market forecasting, currency exchange rates, trading futures, and money laundering analyses.

- Healthcare: KNN has also had application within the healthcare industry, making predictions on the risk of heart attacks and prostate cancer. The algorithm works by calculating the most likely gene expressions.

- Pattern Recognition: KNN has also assisted in identifying patterns, such as in text and digit classification (link resides outside ibm.com). This has been particularly helpful in identifying handwritten numbers that you might find on forms or mailing envelopes.

Advantages and disadvantages of the KNN algorithm

Just like any machine learning algorithm, k-NN has its strengths and weaknesses. Depending on the project and application, it may or may not be the right choice.

Advantages

- Easy to implement: Given the algorithm's simplicity and accuracy, it is one of the first classifiers that a new data scientist will learn.

- Adapts easily: As new training samples are added, the algorithm adjusts to account for any new data since all training data is stored into memory.

- Few hyperparameters: KNN only requires a k value and a distance metric, which is low when compared to other machine learning algorithms.

Disadvantages

- Does not scale well: Since KNN is a lazy algorithm, it takes up more memory and data storage compared to other classifiers. This can be costly from both a time and money perspective. More memory and storage will drive up business expenses and more data can take longer to compute. While different data structures, such as Ball-Tree, have been created to address the computational inefficiencies, a different classifier may be ideal depending on the business problem.

- Curse of dimensionality: The KNN algorithm tends to fall victim to the curse of dimensionality, which means that it doesn't perform well with high-dimensional data inputs. This is sometimes also referred to as the peaking phenomenon, where after the algorithm attains the optimal number of features, additional features increases the amount of classification errors, especially when the sample size is smaller.

- Prone to overfitting: Due to the “curse of dimensionality”, KNN is also more prone to overfitting. While feature selection and dimensionality reduction techniques are leveraged to prevent this from occurring, the value of k can also impact the model’s behavior. Lower values of k can overfit the data, whereas higher values of k tend to “smooth out” the prediction values since it is averaging the values over a greater area, or neighborhood. However, if the value of k is too high, then it can underfit the data.

CHAPTER -4
Multi Class Classification

4.1. What is Multi-class Classification?

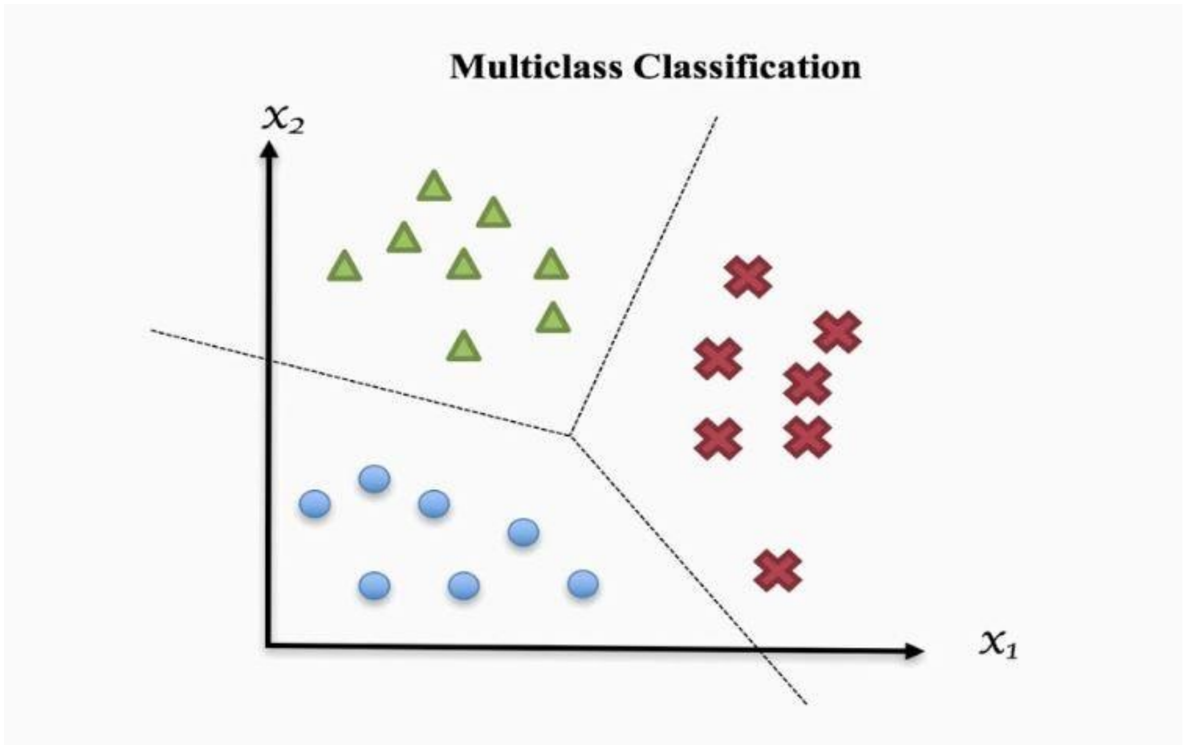


Figure 6: Multiclass classification

When we solve a classification problem having only two class labels, then it becomes easy for us to filter the data, apply any classification algorithm, train the model with filtered data, and predict the outcomes. But when we have more than two class instances in input train data, then it might get complex to analyze the data, train the model, and predict relatively accurate results. To handle these multiple class instances, we use multi-class classification.

Multi-class classification is the classification technique that allows us to categorize the test data into multiple class labels present in trained data as a model prediction.

There are mainly two types of multi-class classification techniques:-

- **One vs. All (one-vs-rest)**
- **One vs. One**

4.2. Binary classification vs. Multi-class classification

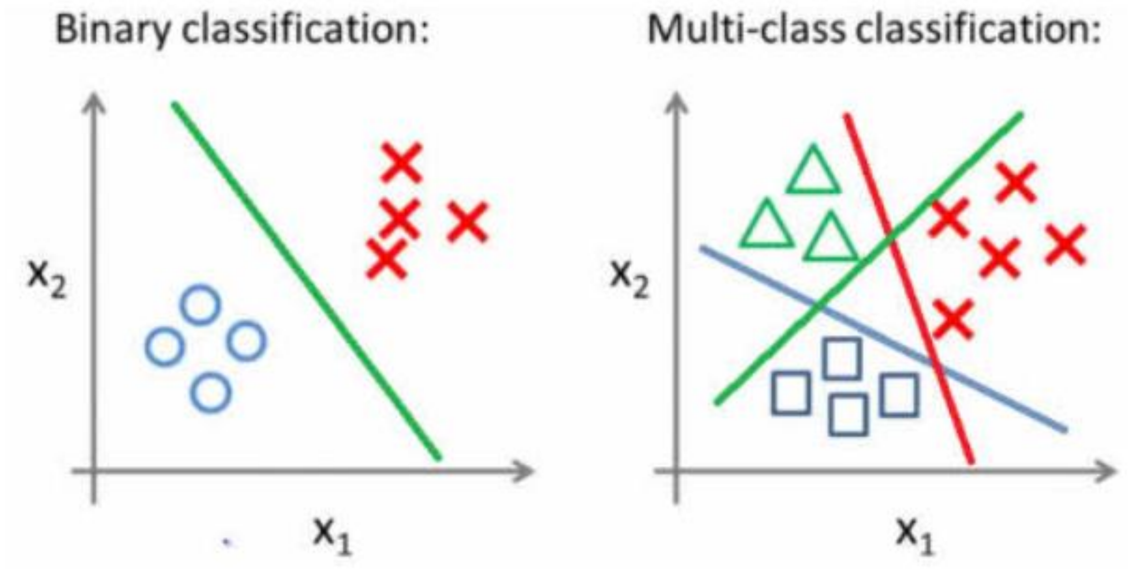


Figure 7: Binary classification vs multiclass classification

Binary Classification

- Only two class instances are present in the dataset.
- It requires only one classifier model.
- Confusion Matrix is easy to derive and understand.
- Example:- Check email is spam or not, predicting gender based on height and weight.

Multi-class Classification

- Multiple class labels are present in the dataset.

- The number of classifier models depends on the classification technique we are applying to.
- One vs. All:- **N-class instances** then **N binary classifier models**
- One vs. One:- **N-class instances** then **N* (N-1)/2 binary classifier models**
- The Confusion matrix is easy to derive but complex to understand.
- Example:- Check whether the fruit is apple, banana, or orange.

4.3. One vs. All (One-vs-Rest)

In one-vs-All classification, for the N-class instances dataset, we have to generate the N-binary classifier models. The number of class labels present in the dataset and the number of generated binary classifiers must be the same.

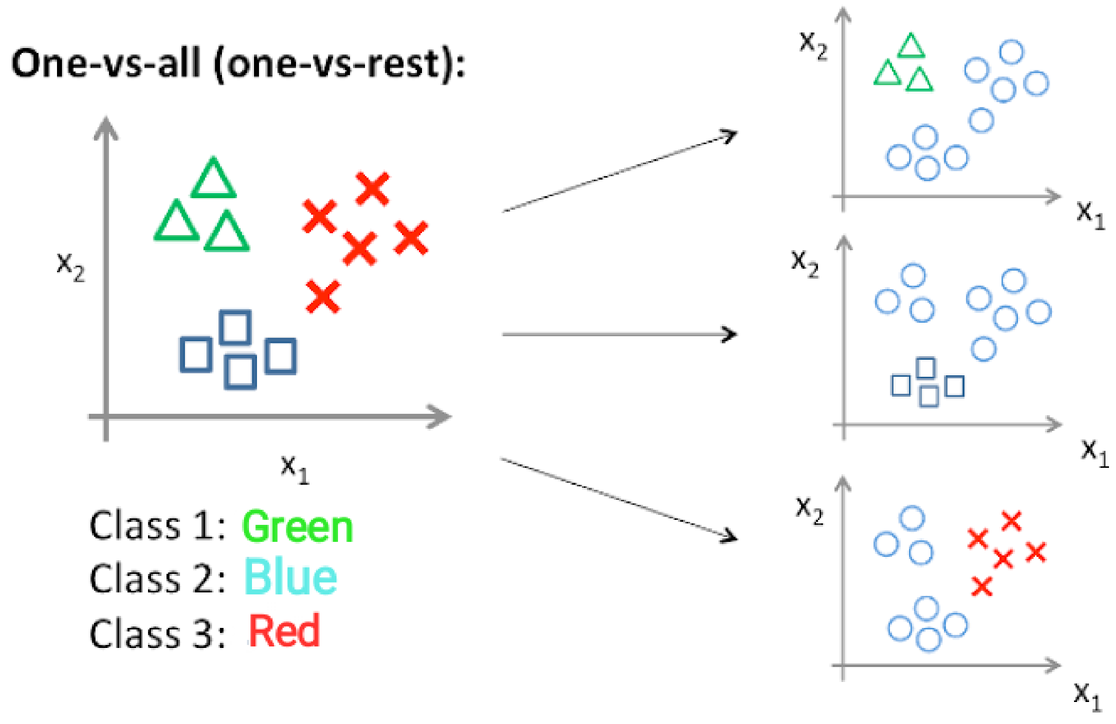


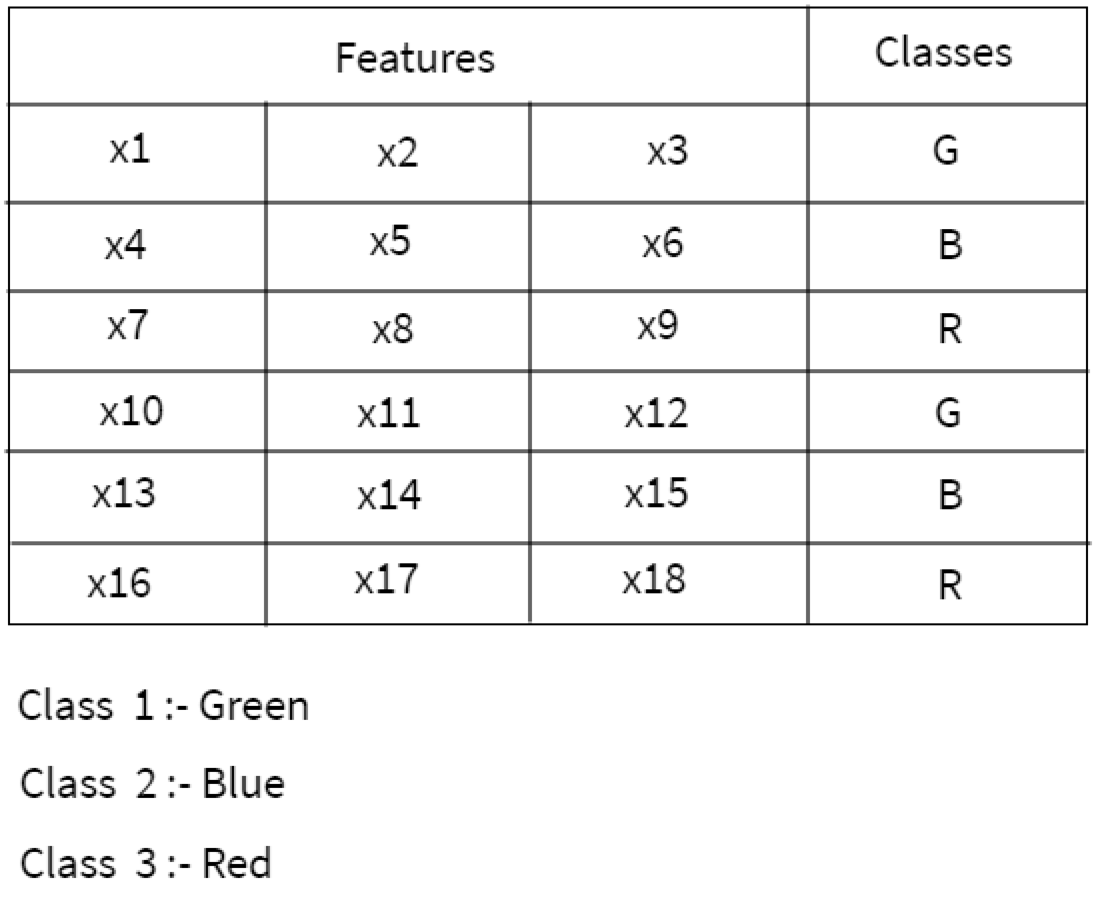
Figure 8: Photo via cc.gatech.edu

As shown in the above image, consider we have three classes, for example, type 1 for Green, type 2 for Blue, and type 3 for Red.

Now, as I told you earlier that we have to generate the same number of classifiers as the class labels are present in the dataset, So we have to create three classifiers here for three respective classes.

- **Classifier 1:- [Green] vs [Red, Blue]**
- **Classifier 2:- [Blue] vs [Green, Red]**
- **Classifier 3:- [Red] vs [Blue, Green]**

Now to train these three classifiers, we need to create three training datasets. So let's consider our primary dataset is as follows,



Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green

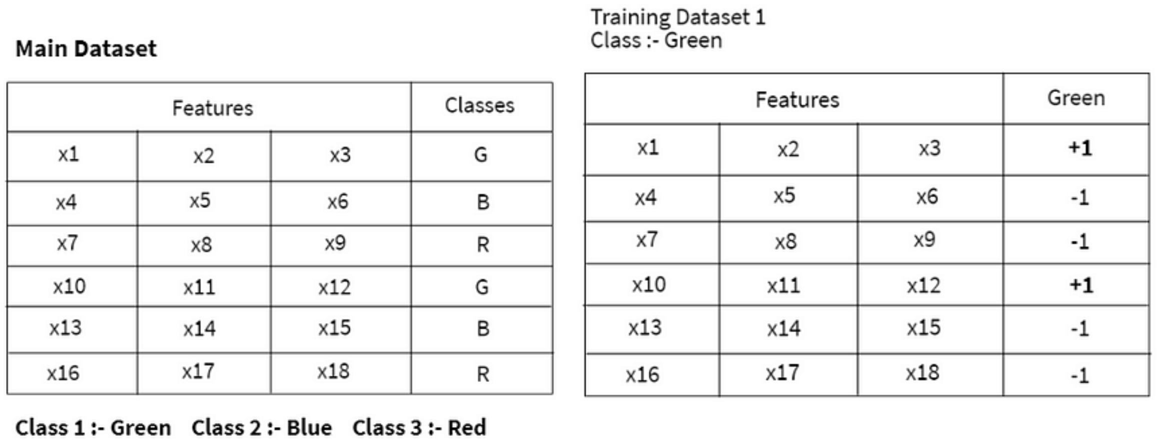
Class 2 :- Blue

Class 3 :- Red

Figure 9: **Primary Dataset**

You can see that there are three class labels **Green**, **Blue,** and **Red** present in the dataset. Now we have to create a training dataset for each class.

Here, we created the training datasets by putting +1 in the class column for that feature value, which is aligned to that particular class only. For the costs of the remaining features, we put -1 in the class column.



Main Dataset

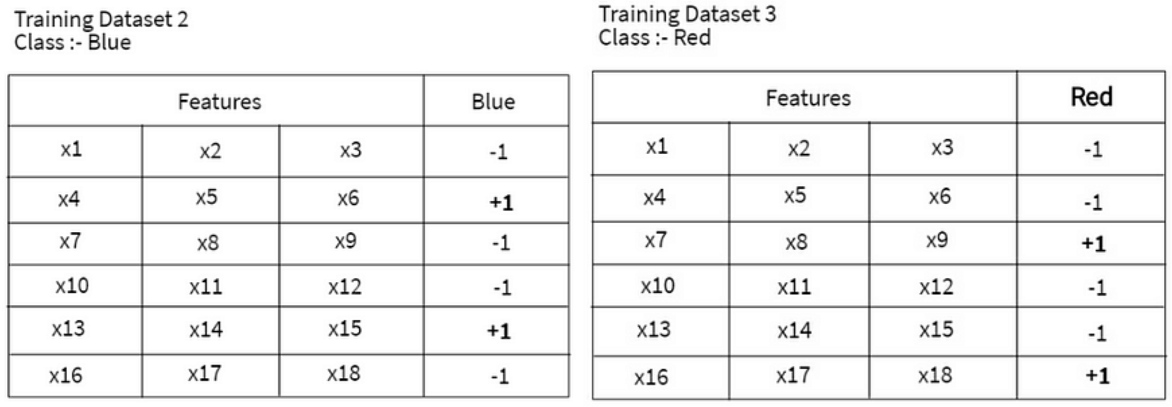
Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green Class 2 :- Blue Class 3 :- Red

Training Dataset 1
Class :- Green

Features			Green
x1	x2	x3	**+1**
x4	x5	x6	-1
x7	x8	x9	-1
x10	x11	x12	**+1**
x13	x14	x15	-1
x16	x17	x18	-1

Figure 10: Training dataset for **Green** class



Training Dataset 2
Class :- Blue

Features			Blue
x1	x2	x3	-1
x4	x5	x6	**+1**
x7	x8	x9	-1
x10	x11	x12	-1
x13	x14	x15	**+1**
x16	x17	x18	-1

Training Dataset 3
Class :- Red

Features			Red
x1	x2	x3	-1
x4	x5	x6	-1
x7	x8	x9	**+1**
x10	x11	x12	-1
x13	x14	x15	-1
x16	x17	x18	**+1**

Figure 11: Training dataset for **Blue** class and **Red** class

Let's understand it by an example,

- Consider the primary dataset, in the first row; we have x1, x2, x3 feature values, and the corresponding class value is G, which means these feature values belong to G class. So we put +1 value in the class column for the correspondence of green type. Then we applied the same for the x10, x11, x12 input train data.

- For the rest of the values of the features which are not in correspondence with the Green class, we put -1 in their class column.

I hope that you understood the creation of training datasets.

Now, after creating a training dataset for each classifier, we provide it to our classifier model and train the model by applying an algorithm.

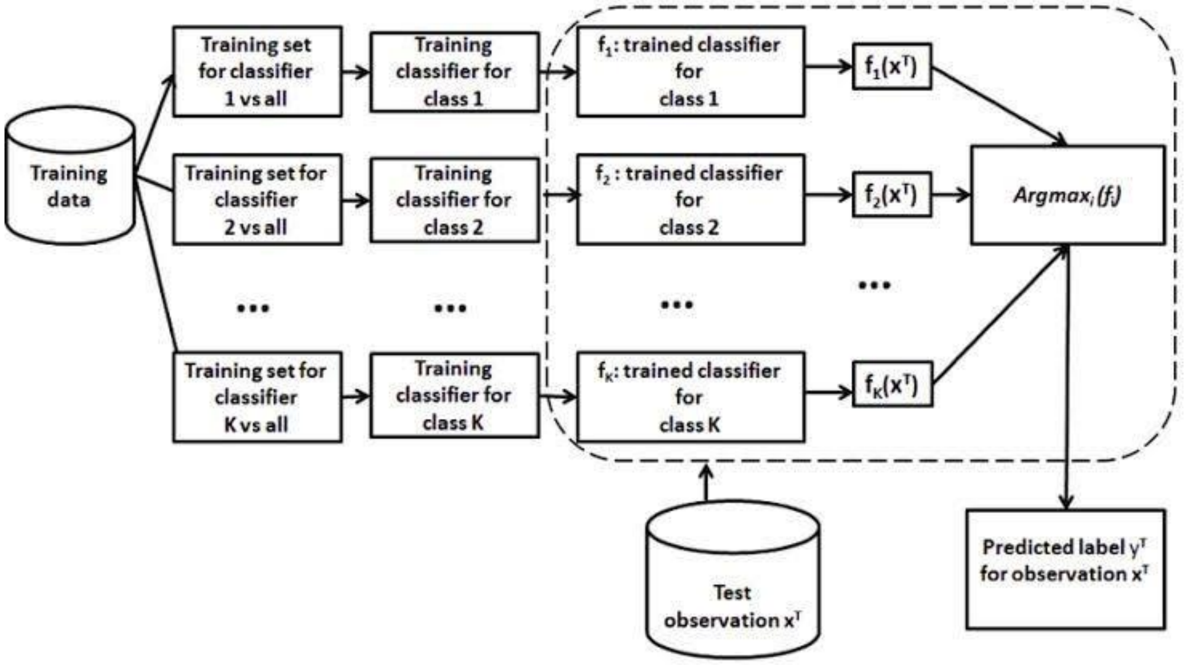

Training data
Training set for classifier 1 vs all
Training set for classifier 2 vs all
Training set for classifier K vs all
Training classifier for class 1
Training classifier for class 2
Training classifier for class K
f1: trained classifier for class 1
f2 : trained classifier for class 2
fK: trained classifier for class K
f1(xT)
f2(xT)
fK(xT)
Argmaxi (fi)
Test observation xT
Predicted label yT for observation xT

Figure 12: Classifier Model

After the training model, when we pass input test data to the model, then that data is considered as input for all generated classifiers. If there is any possibility that our input test data belongs to a particular class, then the classifier created for that class gives a positive response in the form of **+1**, and all other classifier models provide an adverse reaction in the way of **-1**. Similarly, binary classifier models predict the probability of correspondence with concerning classes.

By analyzing the probability scores, we predict the result as the class index having a maximum probability score.

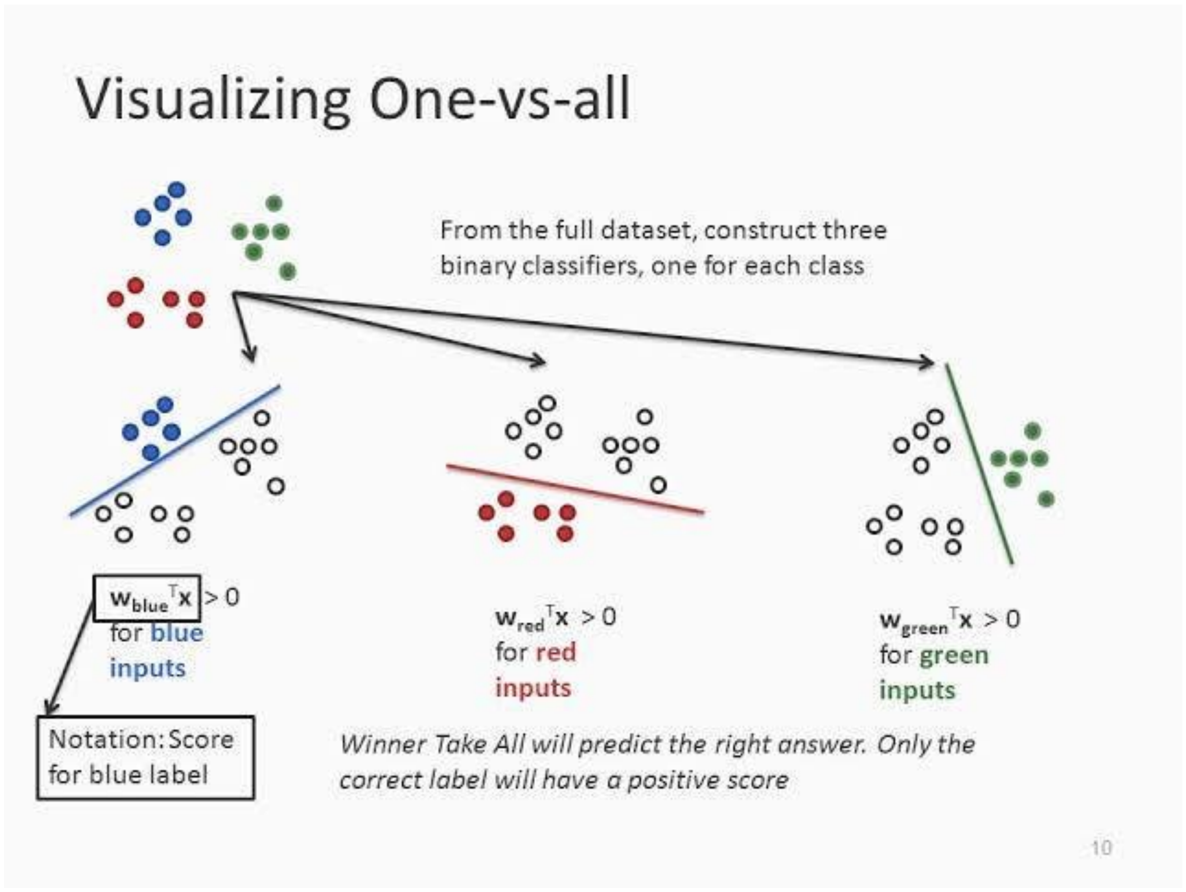

Visualizing One-vs-all
From the full dataset, construct three binary classifiers, one for each class
$w_{blue}^T x > 0$ for blue inputs
$w_{red}^T x > 0$ for red inputs
$w_{green}^T x > 0$ for green inputs
Notation: Score for blue label
Winner Take All will predict the right answer. Only the correct label will have a positive score
10

Figure 13: Visualizing One vs rest Classifier

- Let's understand with one example by taking three test features values as y1, y2, and y3, respectively.
- We passed test data to the classifier models. We got the outcome in the form of a positive rating derived from the **Green** class classifier with a probability score of (**0.9)**.
- Again We got a positive rating from the **Blue** class with a probability score of **(0.4)** along with a negative classification score from the remaining **Red** classifier.
- Hence, based on the positive responses and decisive probability score, we can say that our test input belongs to the **Green** class.

Look at the below example of fitting multi-class **Logistic Regression** model using a built-in **one vs. rest (OvR)** technique.

```
#Import LogisticRegression() model from scikit_learn
 from sklearn.datasets import make_classification
 from sklearn.linear_model import LogisticRegression#define dataset
 X_train, y_train = make_classification(n_samples=500, n_features=8, n_informative=5,
n_redundant=5, n_classes=4, random_state=1)#define classification model
 Multiclass_model = LogisticRegression(multi_class='ovr')#fit model
 Multiclass_model.fit(X_train, y_train)#make final predictions
 y_pred = model.predict(X_train)
```

4.4. One vs. One (OvO)

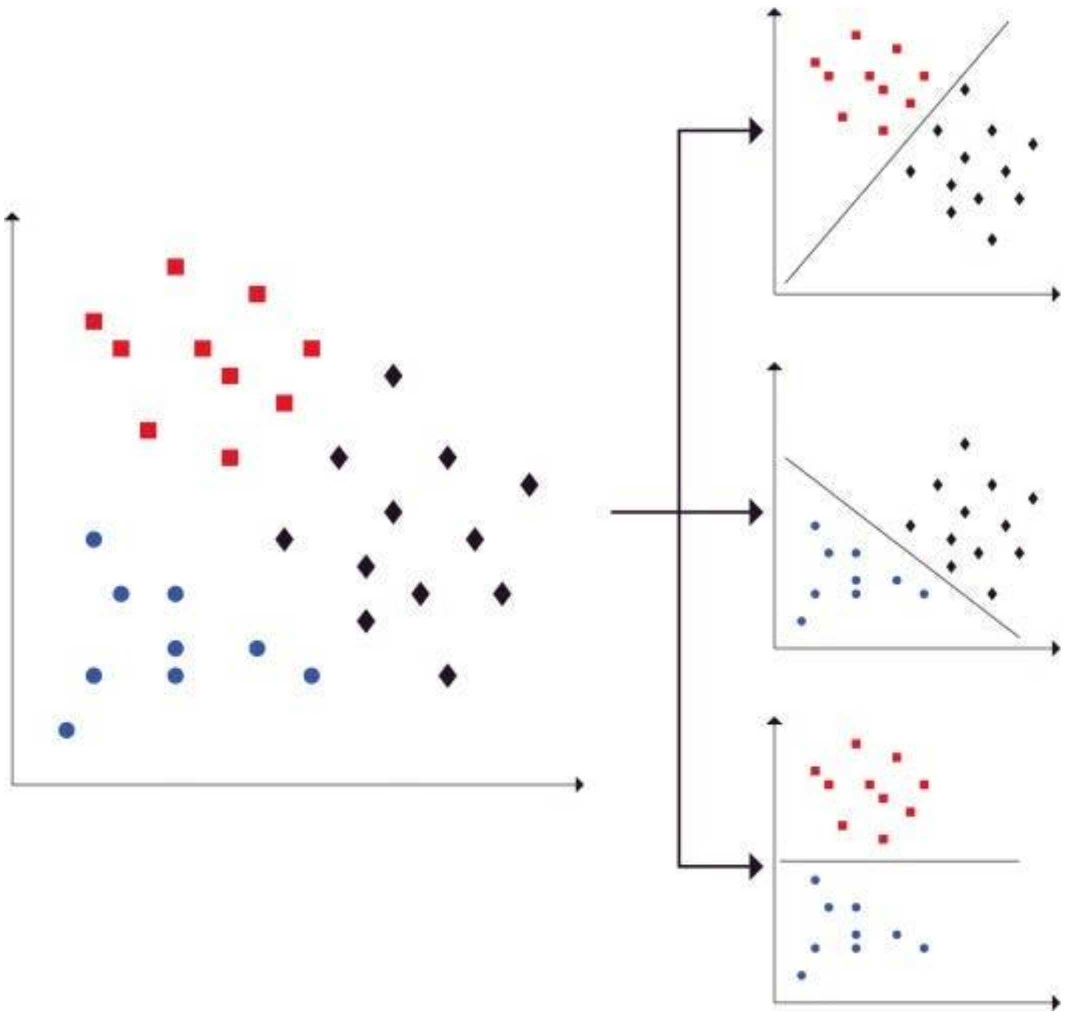


Figure 14: one vs one

In One-vs-One classification, for the **N-class** instances dataset, we have to generate the **N* (N-1)/2** binary classifier models. Using this classification approach, we split the primary dataset into one dataset for each class opposite to every other class.

Taking the above example, we have a classification problem having three types: **Green**, **Blue**, and **Red (N=3).**

We divide this problem into **N* (N-1)/2 = 3** binary classifier problems:

- Classifier 1: Green vs. Blue
- Classifier 2: Green vs. Red
- Classifier 3: Blue vs. Red

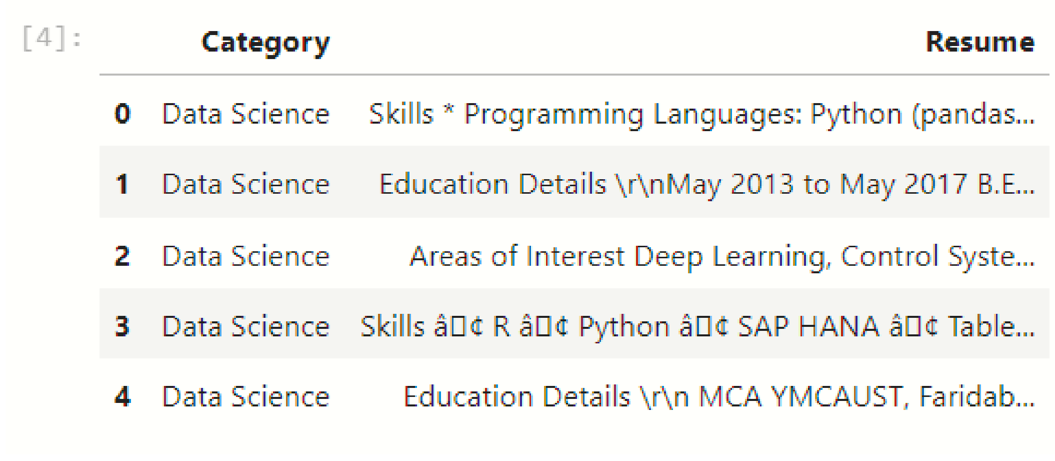
Each binary classifier predicts one class label. When we input the test data to the classifier, then the model with the majority counts is concluded as a result.

CHAPTER -5

PROPOSEDWORK

This is a project for resume screening and this is helpful for any hiring process as it can read through the whole resume and after reading the resume it can tell that the particular resume belongs to the particular category.

This projects dataset consists of total 23 different categories which covers mostly all the popular and some of the nonpopular job categories.

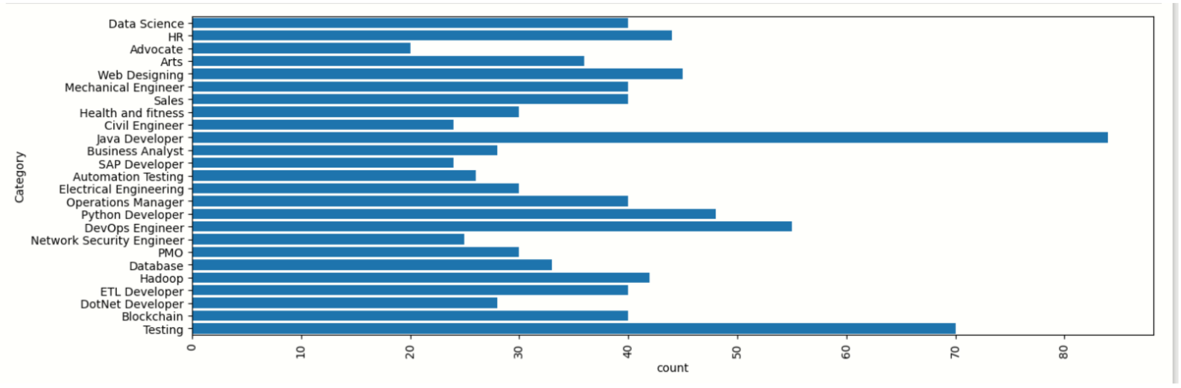


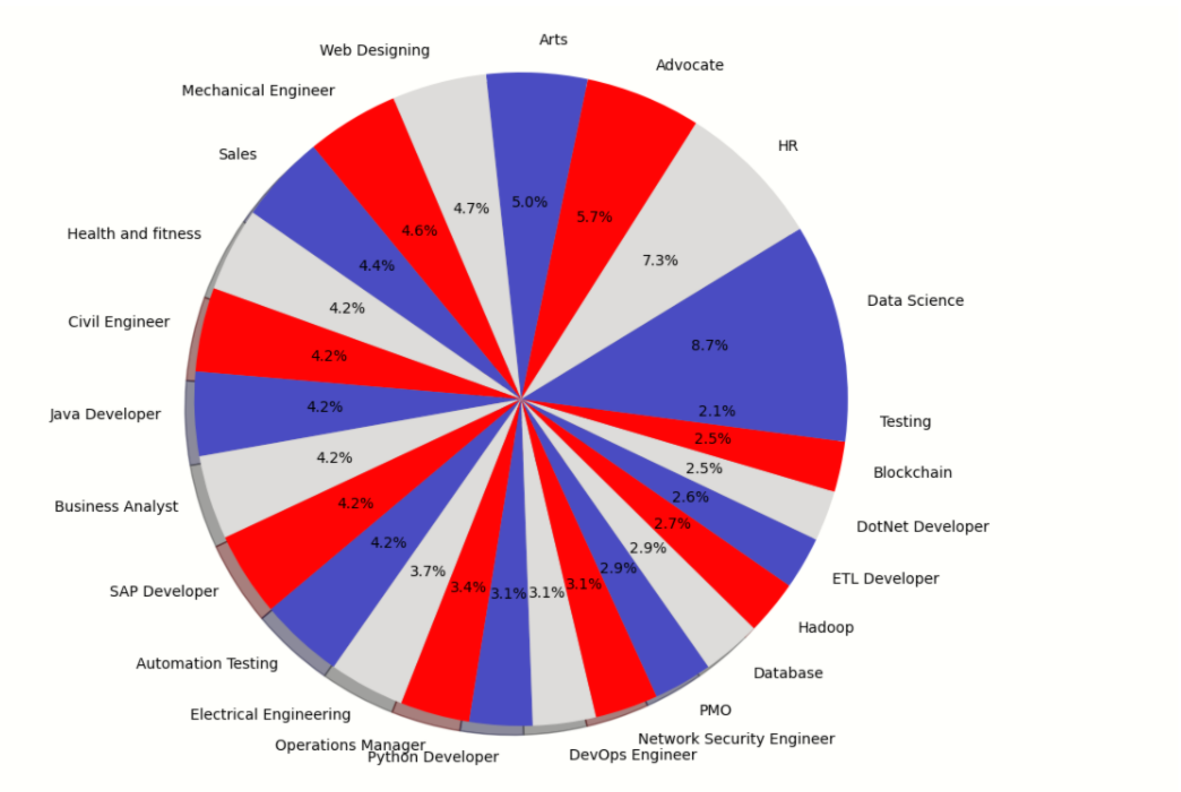
[4]:

	Category	Resume
0	Data Science	Skills * Programming Languages: Python (pandas...
1	Data Science	Education Details \r\nMay 2013 to May 2017 B.E...
2	Data Science	Areas of Interest Deep Learning, Control Syste...
3	Data Science	Skills â¢ R â¢ Python â¢ SAP HANA â¢ Table...
4	Data Science	Education Details \r\n MCA YMCAUST, Faridab...

In this project we have taken the dataset from Kaggle and the specific dataset consists of 2 columns and 962 data entries. The first column consists of the category of the resume and the second column consists of the data which is present in the particular resume. There are total 962 such resumes in the dataset of 23 different categories.

Firstly in this project we have explored the dataset that how many categories are there and which category has more number of resumes. Also we have visualized this data in diagrammatical format. For doing this we have used the pandas library in python to import dataset as it is in the csv format that is comma separated files. Also we have used matplotlib and seaborn library from python to visualize the data in different patterns one of them is bar graph and one of them is pie chart.





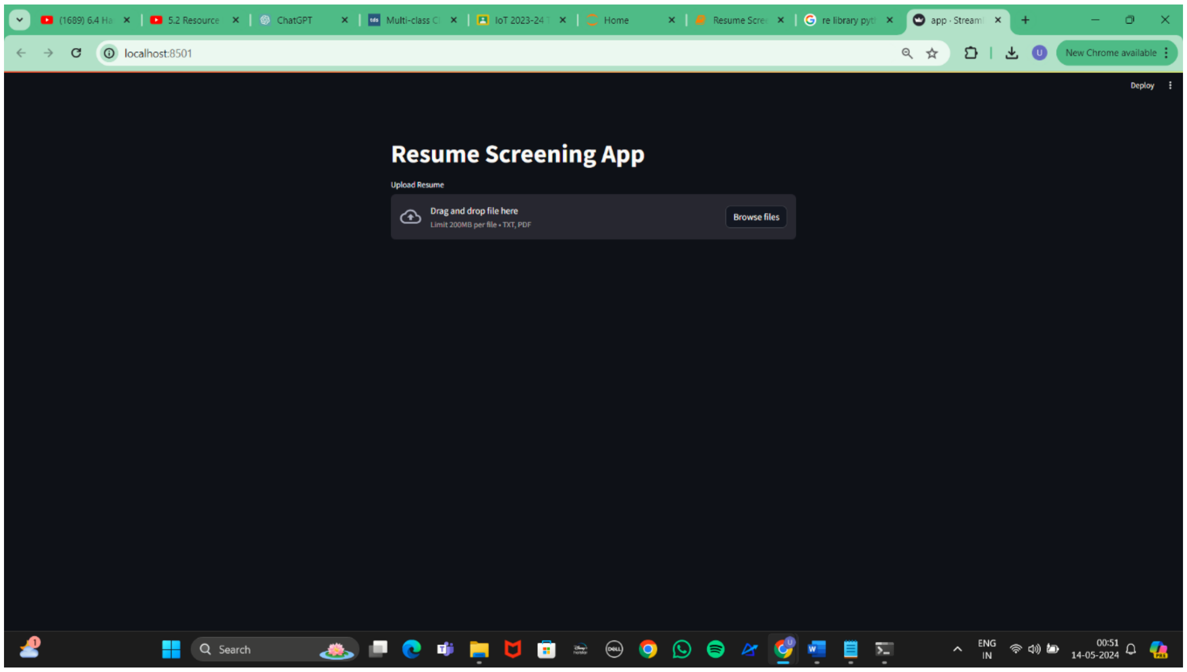
Then in the next part we have cleaned the dataset as it contained the some of the unwanted things like URLs, hashtags, mentions, special letters, punctuations etc. Also to clean the above mentioned things we have used the regular expression library from python to clean the data. For cleaning the data we have created a function which performs all the operations to clean the data.

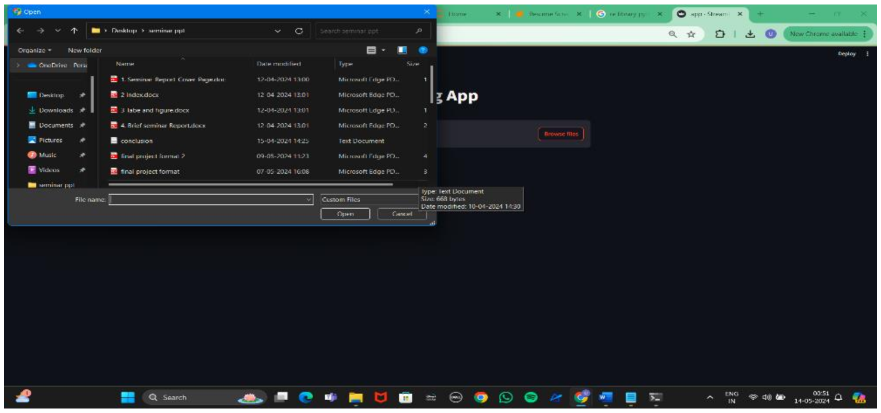
Also we have vectorized the textual data into numerical format to train the model because the model is trained in the numerical data format.

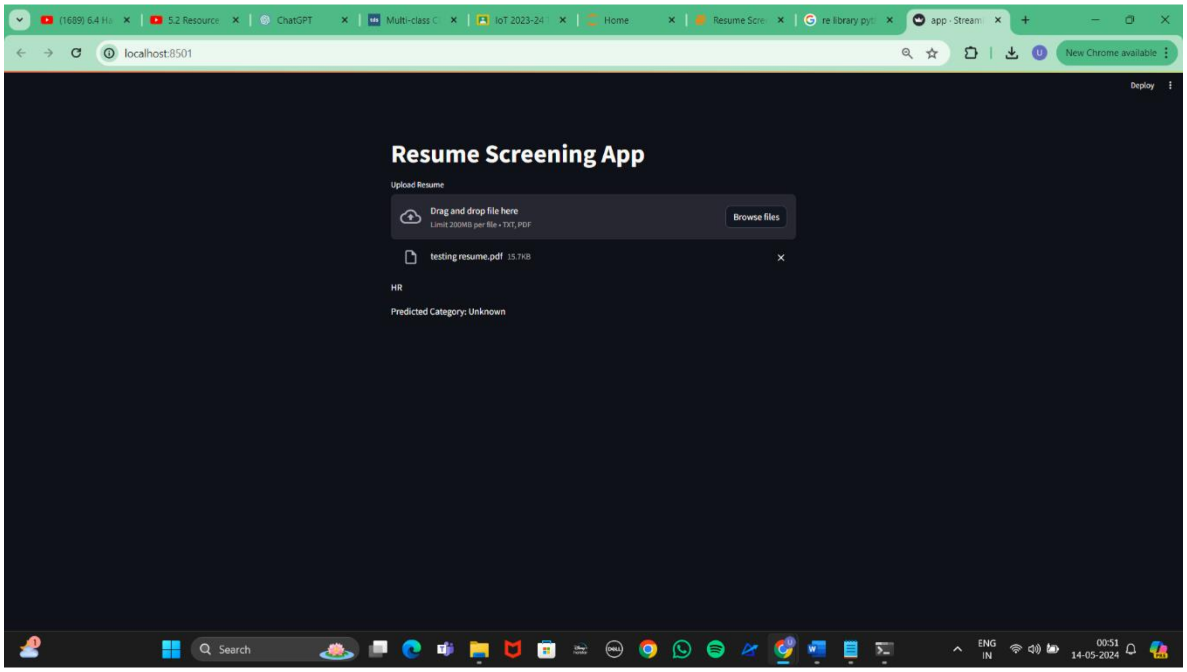
After the data is cleaned we have used the scikit-learn library form python to implement the KNN algorithm and ONEvsREST classifier to classify the category of the resume from 23 different categories. Also to evaluate the created model we have used the performance metrics from the scikit-learn library which is very useful and easy to implement.

After that everything has been done and in the last we have created a basic frontend using streamlit framework in python and we can give the input from framework using a text file or pdf file which will contain a resume and our model will give us a category by reading the resume. If it does not guess the category of the given resume it will give the category as unknown.

Chapter 6
RESULT







localhost:8501

Resume Screening App

Upload Resume

Drag and drop file here
Limit 200MB per file • TXT, PDF

testing resume.pdf 15.7KB

HR

Predicted Category: Unknown

Chapter-7

Future Scope

The future scope of the proposed smart resume screening algorithm is promising and encompasses several potential avenues for further enhancement and application. Firstly, while the current model focuses primarily on resumes written in English, there is a significant opportunity to extend its capabilities to handle resumes in other languages. By incorporating natural language processing (NLP) techniques and language translation algorithms, the model can effectively process and analyze resumes written in diverse languages, thereby broadening its applicability to a global audience.

Furthermore, the integration of advanced AI and ML techniques can enhance the model's accuracy and efficiency in screening resumes. Deep learning algorithms, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can be leveraged to extract deeper semantic meaning from resume content, enabling more precise matching of applicant profiles with organizational requirements. Additionally, the incorporation of unsupervised learning approaches, such as clustering and topic modeling, can facilitate the automatic identification of relevant keywords and skill sets, further streamlining the screening process.

CONCLUSION

Resume screening plays a vital role in the hiring process by efficiently identifying candidates who match the requirements of a job opening. Whether conducted manually by human recruiters or through automated software, the goal remains the same: to sift through a large pool of applicants and select those who are the best fit for further consideration.

By effectively screening resumes, organizations can save time and resources while ensuring that qualified candidates are not overlooked. However, it's essential to strike a balance between speed and accuracy, as rushing through the screening process may result in missing out on promising candidates or making poor hiring decisions.

Ultimately, a well-executed resume screening process sets the stage for successful hiring outcomes by providing a solid foundation for subsequent stages of the selection process, such as interviews and assessments. It empowers hiring teams to focus their efforts on evaluating candidates who have the greatest potential to excel in the role, contributing to the overall success of the organization.

REFERENCES

[1]. Sujit Amin , Nikita Jayakar , Sonia Sunny , Pheba Babu , M.Kiruthika, Ambarish Gurjar, Web Application for Screening Resume,2019 International Conference on Nascent Technologies in Engineering (ICNTE 2019),978-1-5386-9166-3/19/$31.00 ©2019 IEEE

[2]. K. Khan,R. Ullah khan, Ali Alkhalifah, N. Ahmad, Urdu Text Classification using Decision Tree,978-1-4673-9268-6/15/$31.00 ©2015 IEEE

[3]. ZI-QIANG WANG, XIA SUN, DE-XIAN ZHANG, XIN LI, AN OPTIMAL SVM-BASED TEXT CLASSIFICATION ALGORITHM, Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 13-16 August 2006 1-4244-0060-0/06/$20.00 ©2006 IEEE

[4]. Fang Miao, Pu Zhang, Libiao Jin*, Hongda Wu, Chinese News Text Classification Based on Machine learning algorithm,2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetic

[5]. Huabei Nie, Yi Niu, Juan Zhang, Web Document Classification Based on Fuzzy k-NN Algorithm, 2009 International Conference on Computational Intelligence and Security 978-0-7695-3931-7/09 $26.00 © 2009 IEEE

[6]. N. Monaikul, G. Castellucci, S. Filice, O. Rokhlenko,Continual learning for named entity recognition, InProceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, (2021)

[7]. Z. Liu, F. Jiang, Y. Hu,C. Shi, P. Fung,NER-BERT: A Pre-trained Model for LowResource Entity Tagging, arXiv preprint arXiv:2112.00405.,(2021)

[8]. J. Yu, B. Bohnet, M. Poesio,Named entity recognition as dependency parsing, arXiv preprint arXiv:2005.07150., (2020)

[9]. P. K. Roy, S. S. Chowdhary, R. Bhatia, A Machine Learning approach for automation of Resume Recommendation system, Procedia Computer Science, (2020)

[10]. A. Lad, S. Ghosalkar, B. Bane, K. Pagade and A. Chaurasia,Machine Learning Based Resume Recommendation System, International Journal of Modern Developments in Engineering and Science, (2022)

[11]. I. Ali, N. Mughal, Z. H, Khand, J. Ahmed, G. Mujtaba,Resume classification system using natural language processing and machine learning techniques, Mehran University Research Journal Of Engineering & Technology,(2022)

[12]. M. F. Mridha, R. Basri,M. M. Monowar, M. A. Hamid, A Machine Learning Approach for Screening Individual's Job Profile Using Convolutional Neural Network. IEEE International Conference on Science & Contemporary Technologies (ICSCT) (2021)