Application Screening for Crowdfunding Projects using A.I.

A project report submitted to

Osmania University

In Partial fulfilment of the requirements
for the award of the degree of

Bachelor of Engineering
In
Computer Science and Engineering

Submitted by
B.E VIII semester

Md. Safi Ur Rahman Khan **(1608-17-733-062)**

Mohammed Ashfaq **(1608-17-733-102)**

Ahmed Rizwan **(1608-16-733-084)**

Under the Esteemed Guidance of
Dr. G. Shyama Chandraprasad
Assoc. Prof, Head Dept of IT

Department of Computer Science and Engineering
(Accredited by NBA)
Matrusri Engineering College

(Affiliated to Osmania University, Approved by AICTE)
Saidabad, Hyderabad-500059, Telangana, India)

(2020-21)

Department of Computer Science and Engineering

Matrusri Engineering College

(Affiliated to Osmania University, Approved by AICTE)
Saidabad, Hyderabad-500059
(2020-2021)

CERTIFICATE

This is to Certify that A Project report entitled **"Application Screening for Crowdfunding Projects using A.I.**" is being submitted by Md. Safi Ur Rahman Khan **(**1608-17-733-062**),** Mohammed Ashfaq **(**1608-17-733-102**),** Ahmed Rizwan **(**1608-16-733-084**)** in partial fulfilment of the requirement of the award for the degree of Bachelor of Engineering in "Computer Science and Engineering" **O.U**., Hyderabad during the year 2020-2021 is a record of bonafide work carried out by them under my guidance. The results presented in this thesis have been verified and are found to be satisfactory.

Dr.G.Shyama Chandra
Associate Prof.& Head,
Dept. of I.T.
Project Guide

Dr. P. Vijaya Pal Reddy
Professor & Head,
Dept. of C.S.E.
H.O.D

External Examiner(s)`

Department of Computer Science and Engineering
Matrusri Engineering College

(Affiliated to Osmania University, Approved by AICTE)

Saidabad, Hyderabad, T.S-500059

(2019-2020)

DECLARATION

We, **Mr. Md. Safi Ur Rahman Khan, Mr. Mohammed Ashfaq** and **Mr. Ahmed Rizwan** bearing **H.T. No. 1608-17-733-062, 1608-17-733-102** and **1608-16-733-084,** hereby certify that the project report entitled **"Application Screening for Crowdfunding Projects using A.I."** is submitted in the partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Computer Science and Engineering**.**

This is a record of the bonafide work carried out by us under the guidance of

Dr. G. Shyama Chandra Prasad, Professor, Matrusri Engineering College, Saidabad, Hyderabad. The Results embodied in this report have not been reproduced/copied from any source. The results embodied in this report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Md. Safi Ur Rahman Khan (1608-17-733-062)

Mohammed Ashfaq (1608-17-733-102)

Ahmed Rizwan (1608-16-733-084)

Dept. of CSE, MECS

ACKNOWLEDGEMENT

This project consumed huge amount of work, research and dedication. Still implementation would not have been possible if we did not have support of my Project Guide, Project Coordinator, Head of the Department and Principal. Therefore, we like to extend our sincere gratitude to all of them.

We are grateful to our project guide **Dr. G. Shyama Chandra Prasad,** Assoc. Professor, for provision of expertise, technical support and guidance in the implementation.

We wish to express our gratitude to project coordinators for their indefatigable inspiration, constructive criticisms and encouragement throughout this dissertation work.

We would like to express our sincere thanks to the Professor and Head of the Department, **Dr. P. Vijaya Pal Reddy**, for permitting us to do this project.

We would like to express our gratitude to **Dr. D. Hanumantha Rao**, principal of Matrusri Engineering College who permitted to carry out this project as per the academics.

We would like to thank CSE Department for providing us this opportunity to share and contribute our part of work to accomplish the project in time and all the teaching and support staff for their steadfast support and encouragement.

Nevertheless, we express our gratitude towards our families and colleagues for their kind cooperation and encouragement which helped us in completion of this project.

ABSTRACT

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects curated by the teachers and coordinator which are in need of funding.

As the organization’s reach scaled, a large number of volunteers are required to manually screen each submission before it's approved.

The problem we have to solve is How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible. We aim to use state of the art machine learning techniques in order to train our model to automatically predict the acceptance or rejection of the project proposals thereby reducing the amount of human work employed. Apart from that we also aim to perform various data analysis tasks to find out relevant insights and gain good business outcomes from it.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1. INTRODUCTION

1.1 PURPOSE OF PROJECT

The title “Application screening for crowd funding projects”. Crowdfunding is a way to raise money for an individual or organization by collecting donations through family, friends, friends of friends, strangers, businesses, and more. By using social media to spread awareness, people can reach more potential donors than traditional forms of fundraising.

Now even though, popular crowd funding websites dont let their data out in the open, hence we use the data from donorschoose.org

DonorsChoose.org is an organization that receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 1,000,000 project proposals. As a result, there are three main problems they need to solve:

1. How to scale current manual processes and resources to screen 1,000,000 projects so that they can be posted as quickly and as efficiently as possible
2. How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
3. How to focus volunteer time on the applications that need the most assistance

The goal of this project is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

With an algorithm to pre-screen applications, DonorsChoose.org can auto-approve some applications quickly so that volunteers can spend their time on more nuanced and detailed project vetting processes, including doing more to help teachers develop projects that qualify for specific funding opportunities.

Our machine learning algorithm can help more teachers get funded more quickly, and with less cost to DonorsChoose.org, allowing them to channel even more funding directly to

classrooms across the country.

1.2. PROBLEMS IN CURRENT APPROACH

As mentioned in the previous section, donorschoose.org hires a large number of volunteers (more than hundred volunteers) each year to help screen the thousands of project proposals that are submitted by teachers all over the USA. When we observe the form that the teachers have to fill out in order to get their project proposals onto the donorschoose.org website, we see that the various fields are Project title, brief project summary, upto 4 essays that describe various aspects of the project as shown below:

Prior to May 17, 2016, the prompts for the essays were as follows:

- project_essay_1: "Introduce us to your classroom"
- project_essay_2: "Tell us more about your students"
- project_essay_3: "Describe how your students will use the materials you're requesting"
- project_essay_4: "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- project_essay_1: "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- project_essay_2: "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

Now by observing the data, we can see that humongous amounts of data is to be read, analyzed and then background checks have to be performed in order to check whether the project proposal is really valid and not just a ruse to extract money from the good Samaritans

Now this whole process is time and resource consuming. Each volunteer would spend anywhere from 5 hours to a whole day on just one project. This is a huge waste of useful human energy and time. This time spent in reviewing the projects can be spent in doing more useful tasks like guiding teachers in preparing good project proposals.

Another grave problem that we observe in the original method is human bias. Humans have a tendency to be biased towards the things they are more passionate about or hold close to their hearts. For example, a volunteer from the state of Texas may be more inclined towards the project proposals that originate from Texas. Also, if a relative or friend of a volunteer submits a project proposal, his proposal is more likely to be chosen by the volunteer.

1.3. SOLUTION TO THE PROBLEM STATEMENT

We are proposing a more of an automated system of getting the project proposals approved. We use popular and state of the art Artificial Intelligence techniques especially Machine Learning algorithms. Using the data already available to us, we train the models using various algorithms. Now this model has learnt all the necessary features from the previous data and now can be used to predict the result for new data points.

Now the benefits of our method are quite evident. Since our model automatically analyzes the data and predicts whether the proposal should be accepted or not, the usage of volunteers reduces. Hence, a lot of human effort and cost is saved. This saved time can be used to guide the teachers on how to upload a proper application that is more likely to be accepted.

Another evident advantage is that bias is completely removed. This statement may seem quite obvious but it has some falsity. This highly depends on how unbiased is our training data. If our training data is unbiased, then the model surely eliminates all the bias.

When it comes to the part of background verification, our base paper comes in. The base paper “When words sweat” proves that when people write essays, those essays and any writing can be easily used to derive necessary conclusions regarding the mentality of the person and whether or not the persons statement is valid/true or not.

2. LITERATURE SURVEY

The base paper that we have chosen is “When Words Sweat: Identifying Signals for Loan Default in the Text of Loan Applications” written by Oded Netzer, Alain Lemaire, and Michal Herzenstein.

Although this paper is not directly related to our project, it provides us deep insights and has helped us in making various presumptions regarding the application screening process.

The authors present empirical evidence that borrowers, consciously or not, leave traces of their intentions, circumstances, and personality traits in the text they write when applying for a loan. This textual information has a substantial and significant ability to predict whether borrowers will pay back the loan above and beyond the financial and demographic variables commonly used in models predicting default. The authors use text-mining and machine learning tools to automatically process and analyze the raw text in over 120,000 loan requests from Prosper, an online crowdfunding platform. Including in the predictive model the textual information in the loan significantly helps predict loan default and can have substantial financial implications. The authors find that loan requests written by defaulting borrowers are more likely to include words related to their family, mentions of God, the borrower’s financial and general hardship, pleading lenders for help, and short-term-focused words. The authors further observe that defaulting loan requests are written in a manner consistent with the writing styles of extroverts and liars.

Imagine that you are considering lending $2,000 to one of two borrowers on a crowdfunding website. The borrowers are identical in terms of their demographic and financial characteristics, the amount of money they wish to borrow, and the reason for borrowing the money.

However, the text they provided when applying for a loan differs: Borrower#1 writes, “I am a hard working person, married for 25 years, and have two wonderful boys. Please let me explain why I need help. I would use the $2,000 loan to fix our roof. Thank you, God bless you, and I promise to pay you back.” Borrower #2 writes, “While the past year in our new place has been more than great, the roof is now leaking and I need to borrow $2,000 to cover the cost of the repair. I pay all bills (e.g., car loans, cable, utilities) on time.” Which borrower is more likely to default? This question is at the center of this research, as the authors investigated the power of words in predicting loan default. They claimed and show that the text borrowers write at loan origination provides valuable information that cannot be otherwise extracted from the typical data lenders have on borrowers (which mostly include financial and demographic data), and that additional information is crucial to predictions of default.

For example consider the word cloud in fig 1.1

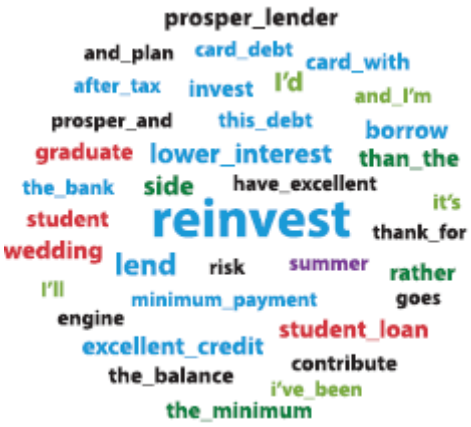


FIG: 1.1

The research showed that these are the most frequently used words that borrowers, who dont default typically use. The most commonly used words by such borrowers are relative words, financial literacy words, words related to a brighter financial future, "I" words, and time-related words.

Contrastingly, the word cloud in fig 1.2 shows the most frequently used words that are used by borrowers who are more likely to default the loans

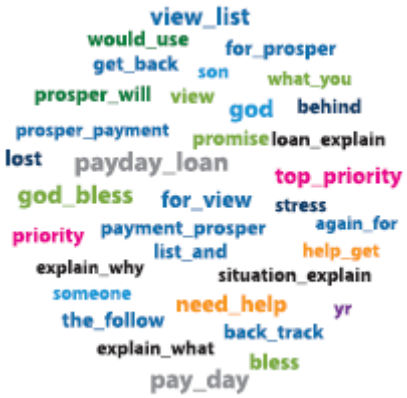


FIG: 1.2

The most commonly used words by defaulters are words related to explanations, external influence words and others, future-tense words, time-related words, work-related words, extremity words, words appealing to lenders, words relating to financial hardship, words relating to general hardship, and desperation/plea words.

Now this loan default prediction is quite similar to application screening process. In loan default prediction, the writeups are used to predict whether the writer is likely to default the

loan or not and accordingly granting the loans. Similarly, in application screening process, the essays are analyzed and is predicted whether the proposal should be accepted or not and if the project proposal is really valid or just a hoax.

Apart from this the various papers that we have considered are the papers that have introduced the various machine learning algorithms that we have used.

The paper "**Discriminatory analysis Non parametric discrimination : consistency properties**" presented by **Evelyn Fix** and **Joseph Hodges** of the **University of California, Berkeley** , introduced the concept of K Nearest Neighbors which is one of the simplest algorithms that we have used.

Further, the paper "**Support-Vector Networks**" published by Corinna Cortes and Vladimir Vapnik of AT&T Bell labs introduces us to the world of Support Vectors and how SVM can be used for classification purposes. Later, Support Vector Regressor was introduced that helped the world see the application of SVM in regression tasks.

Similarly, there were numerous papers and blogs that have helped us immensely in gaining a perspective and choosing relevant algorithms and coming up with respective models. Further, they helped us in understanding the methodologies of improving our accuracy

3. SYSTEM ANALYSIS

3.1 About Donors Choose

Founded in 2000 by a high school teacher in the Bronx, DonorsChoose empowers public school teachers from across the country to request much-needed materials for their students. The site has full transparency and you can see how the money donated is being used to fund project.

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

3.2 About the Dataset

A **data set** (or **dataset**) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables. In the open data discipline, data set is the unit to measure the information released in a public open data repository

3.3 Features of dataset

Features of a dataset are also referred as variables or attributes of the dataset. These have a measurable value for each entry. Each attribute gives us detailed information about the nature of project, the people involved in the project and various other details that help us decide if we want to fund the project or not

The features of dataset used in donorschoose can be listed as follows

1. **Project ID** : a unique identifier given to each entry or request that is made . each project can be recognized by this unique id. No 2 projects will have the same id and its easier to locate the projects based on this since many projects can have similar names. Eg : p036502

2. **Project Title** : Every project must have a title that is given by the person proposing the project to be funded. The title must be simple and convey the goals of the project accurately while being brief enough. It should also be in a manner that attracts the donors attention at a glance and make them want to investigate what the project is
3. **Project Grade** : Donors Choose divides proposed projects into several categories based on what grade for which it is being conducted

   The grades fall into one of these groups

   Grade preK-2

   Grade 3-5

   Grade 6-8

   Grade 9-12

   Based on this information the donor can also filter who they want to donate to as per their choice

4. **Project Subject** : all projects submitted are listed under various subjects and fields . The donor gets to choose a subject the have an affinity towards to donate their money. A filter lets them choose from the list available

5. **School State** : State , City and postal code are provided. Donors can choose to donate closer to their homes or even their old schools

6. **Resource Summary** : A short essay written by the teacher who is presenting the request for donation listing in a brief but detailed manner what the students will need in order to undertake the proposed project . also points out the aims and goals of the project

7. **Project Datetime** : Denotes when the project request was submitted
   EG: 2019-05-24 12:43:55.245

8. **Teacher ID** : a unique identifier that is assigned to a teacher when they propose a project
   EG: bdf8baa8fedef6bfeec7ae4ff1c15c56

9. **Teacher's Prefix** : Displays the teacher's title
   EG: Dr, Mr, Mrs, Ms etc.

10. **Number of projects submitted** : this keeps track of how many projects a teacher has submitted. It uses the Teacher ID to keep track of this

11. **Estimated Cost** : after a bit of research the teacher can calculate and submit an estimated amount required to fully fund the project.

3.4 Artificial intelligence

Artificial Intelligence refers to the phenomenon where a machine acts as a blueprint of the human mind, by being able to understand, analyze, and learn from data through specially designed algorithms. Our goal is to use this to help automate the screening process of applications submitted to Donors Choose.

3.5 Data Science

Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information, and make business decisions. Data science uses complex machine learning algorithms to build predictive models.The data used for analysis can be from multiple sources and present in various formats.

3.6 Machine Learning and Deep Learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial **intelligence** based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

Deep learning is a subset of **machine learning** in artificial intelligence that has networks capable of **learning** unsupervised from data that is unstructured or unlabeled. Also known as **deep** neural **learning** or **deep neural network**

The key differences between the two: **Machine learning** uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned. **Deep**

learning structures algorithms in layers to create an "artificial **neural network**" that can learn and make intelligent decisions on its own.

3.7 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. **Python's** simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.It supports modules and packages, which encourages program modularity and code reuse.

3.8 Libraries used

Python has several libraries that are used to help us develop machine learning and deep learning applications and programs. The main ones used in the project are

3.8.1. Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem, and is typically included in every Python distribution, from those that come with your operating system to commercial vendor distributions like ActiveState's ActivePython.

3.8.2 NumPy

NumPy is a library for the Python programming language, adding support for large,

multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays it's faster than regular Python arrays, which lack numpy's optimized and pre-compiled C code that does all the heavy lifting.

Another reason is that numpy arrays and operations are vectorized, which means they lack explicit looping or indexing in the code. This makes the code not only more readable, but also more similar to standard mathematical notation.

Do import it every time you want to use it in your Python IDE (Integrated Development Environment) like Jupyter Notebook or Spyder (both of them come with Anaconda by default). As a reminder, importing a library means loading it into the memory and then it's there for you to work with.

3.8.3 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython.

It is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs:

- The pyplot API is a hierarchy of Python code objects topped by *matplotlib.pyplot*
- An OO (Object-Oriented) API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib's backend layers.

3.8.4 Seaborn

Seaborn is a **Python** data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. It builds on top of matplotlib and integrates closely with pandas data structures.

It gives us the capability to create amplified data visuals. This helps us understand the data by displaying it in a visual context to unearth any hidden correlations between variables or trends that might not be obvious initially. Seaborn has a high-level interface as compared to the low level of Matplotlib.

There are essentially a couple of (big) limitations in matplotlib that Seaborn fixes:

1. Seaborn comes with a large number of high-level interfaces and customized themes that matplotlib lacks as it's not easy to figure out the settings that make plots attractive
2. Matplotlib functions don't work well with dataframes, whereas seaborn does

That second point stands out in data science since we work quite a lot with dataframes.

3.8.5 Scikit-learn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free

software machine learning library for the Python programming language.[3] It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Algorithmic decision-making methods, including:

- **Classification:** identifying and categorizing data based on patterns.
- **Regression**: predicting or projecting data values based on the average mean of existing and planned data.
- **Clustering**: automatic grouping of similar data into datasets.

ML algorithm basic concepts:

- **Representation** – is a way to configure data such that it can be assessed. Examples include decision trees, sets of rules, instances, graphical models, neural networks, support vector machines, model ensembles and others.
- **Evaluation** – given a hypothesis, evaluation is a way of assessing its validity. Examples include accuracy, prediction and recall, squared error, likelihood, posterior probability, cost, margin, entropy k-L divergence and others.
- **Optimization** – the process of adjusting hyperparameters in order to minimize model errors by using techniques like combinatorial optimization, convex optimization, constrained optimization, etc.

3.8.6 Tenserflow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.Tensorflow is a symbolic math library based on dataflow and differentiable programming.

TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. However, it proved to be very useful for deep learning development as well, and therefore Google open-sourced it.

TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data.

TensorFlow works on the basis of data flow graphs that have nodes and edges. As the execution mechanism is in the form of graphs, it is much easier to execute TensorFlow code in a distributed manner across a cluster of computers while using GPUs. Tensorflow supports both CPU and GPU applications.

3.8.7 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

The core data structures of Keras are layers and models. The simplest type of model is the Sequential model, a linear stack of layers. For more complex architectures, you should use the Keras functional API, which allows to build arbitrary graphs of layers, or write models entirely from scratch via subclasssing.

3.8.8 Pretty Table

PrettyTable is a **Python** library for generating simple ASCII **tables**. It was inspired by the ASCII **tables** used in the PostgreSQL shell psql. We can control many aspects of a **table**, such as the width of the column padding, the alignment of text, or the **table** border.

We can also choose which columns and rows are going to be displayed in the final output. PrettyTable can read data from CSV, HTML, or database cursor and output data in ASCII or HTML.

3.9 System Requirements

Software requirement

- Operating system : MAC OS, Linux , Windows 8 or higher
- IDE : Jupyter Notebook

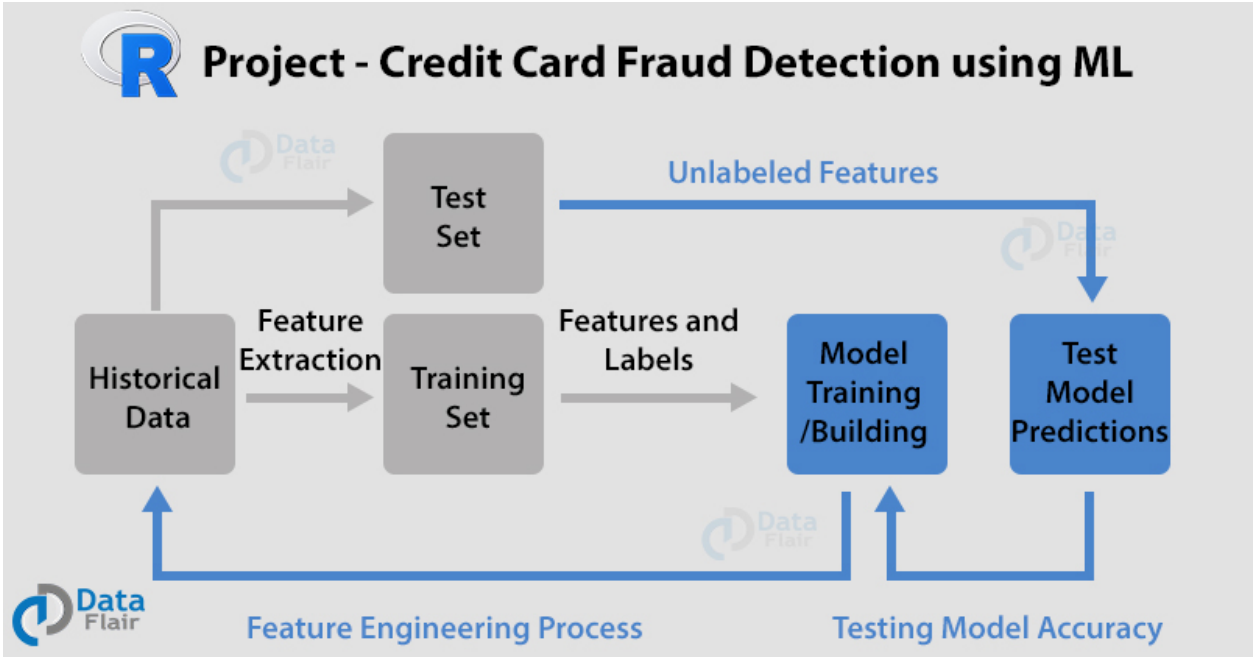
Hardware requirement

- At Least 8GB RAM
- At Least 20GB additional disk space
- Processor : Intel core i3 or higher / AMD Ryzen 5 or higher
- GPU : Nvidia 1650 or higher

4. SYSTEM DESIGN

4.1 Generic Solution

The following diagram represents the generic development cycle and architecture that is used in any machine learning project.

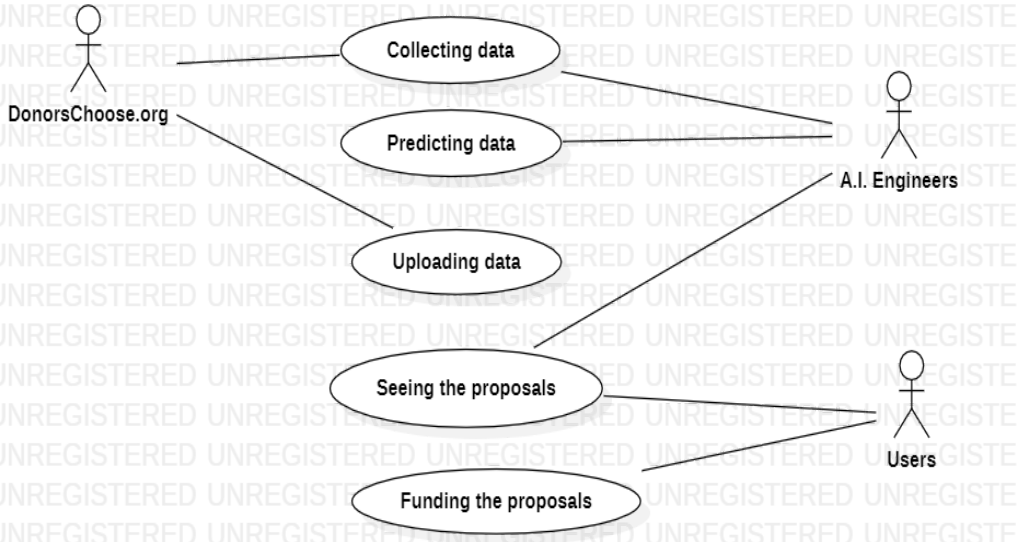


Key Steps in the project are :

- Gathering raw data
- Extraction of features
- Dividing data into training and testing sets
- Test model prediction
- Model training and building
- Repeating these in a cyclic manner till we reach expected accuracy

4.2 UML Diagram

A diagrammatic representation of the steps and characters involved in the processing of requests



This relational diagram depicts the processes as well as who is executing them. All members do not have access to all functions of the program.

4.3 Architecture Pipeline

The following diagram represents the development cycle and the pipeline that is followed in this project

Data collection: We are going to use the data provided by donorschoose.org as part of their free training and research. Alternatively, we can even use their API to scrape the data.

Data Preprocessing : This involves steps like data cleaning, normalization, basic featurization of data.

EDA :Here we do data analysis. We try to find data patterns that help us in efficient feature engineering and modelling

Modelling : Here we do actual feature engineering and data modelling. Some of the models include Logistic Regression, SVM, Decision trees and various ensembles.

Testing :This is the final stage before deployment. Here we test our model with completely new data points and evaluate it

5.IMPLEMENTATION

5.1 Reading Data

The dataset is presented to us by Donorschoose.org.

```
1. project_data = pd.read_csv('train_data.csv')
2. resource_data = pd.read_csv('resources.csv')
```

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.head(2)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

The code utilizes the panda library to read the data from the csv file to store the training data to make a model. The resources.csv has the data points specifying the cost of the project and other attributes with the respect to projects being submitted.

The data read from the csv file is not ready for a pre-processing as we convert some of the data types to the appropriate format which will help us during pre-processing.

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_subject_categories	project_subje
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2	Math & Science	Applied S
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5	Special Needs	

5.2 Pre-Processing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

5.2.1 Blank Spaces

We utilize the natural langauge processing techniques and remove the dirty data such as special characters and blank spaces. It is very common to find whitespace at the beginning, the end, or the inside of a string, whether it's data in a CSV file or data from another source.

5.2.2 Text Processing

Data in real world are rarely clean and homogeneous. Data can either be missing during data extraction or collection. Missing values need to be handled because they reduce the quality for any of our performance metric. It can also lead to wrong prediction or classification and can also cause a high bias for any given model being used.Depending on data sources, missing data are identified differently. Pandas always identify missing values as **NaN**. However, unless the data has been pre-processed to a degree that an analyst will encounter missing values as **NaN**.

Missing values can appear as a question mark (?) or a zero (0) or minus one (-1) or a blank. As a result, it is always important that a data scientist always perform exploratory data analysis(EDA) first before writing any machine learning algorithm. EDA is simply a litmus for understanding and knowing the behaviour of our data

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
# specific
phrase=re.sub(r"won't","will not",phrase)
phrase=re.sub(r"can\'t","can not",phrase)

# general
phrase=re.sub(r"n\'t"," not",phrase)
phrase=re.sub(r"\'re"," are",phrase)
phrase=re.sub(r"\'s"," is",phrase)
phrase=re.sub(r"\'d"," would",phrase)
```

```
    phrase=re.sub(r"\'ll"," will",phrase)
    phrase=re.sub(r"\'t"," not",phrase)
    phrase=re.sub(r"\'ve"," have",phrase)
    phrase=re.sub(r"\'m"," am",phrase)
    return  phrase
```

```
# after preprocesing
preprocessed_essays[20000]
```

```
'students come difficult family lives not let stop built community classroom allows student comfortable even though diverse
school everyone feels included high hispanic population 90 students free reduced price lunch students living single parent
parents work full time although many parents eager help way know important get kids moving want classroom place students ac
tive phyically mentally requested items allow students move day sitting chair movement limited kindergarten students hard t
ime sitting still long periods time would much rather bounce stability ball wiggle cushion sit hard chair choices classroom
allow students active learn time choices classroom also build greater bond students learn choose seat best fits learning st
yle hopefully able help classmates find seat works students move around room able work everyone instead one group day nanna
n'
```

5.2.3 Categorical data

Categorical variables are known to hide and mask lots of interesting information in a data set. It's crucial to learn the methods of dealing with such variables. If you won't, many a times, you'd miss out on finding the most important variables in a model.

5.2.3.1 Vectorizing Categorical data

Machine learning and deep learning models, like those in Keras, require all input and output variables to be numeric.This means that if your data contains categorical data, you must encode it to numbers before you can fit and evaluate a model.
The two most popular techniques are an **integer encoding** and a **one hot encoding**, although a newer technique called **learned embedding** may provide a useful middle ground between these two methods.

5.2.3.2 Vectorizing Text data

(a) Bag of words

The bag-of-words model is a simplifying representation used in natural language processing and

information retrieval. In this model, a text is represented as the bag of its words, disregarding grammar and even word order but keeping multiplicity. The bag-of-words model has also been used for computer vision.

```
1. vectorizer=CountVectorizer(min_df=10)
2. text_bow=vectorizer.fit_transform(preprocessed_essays)
3. print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109245, 16512)
```

```
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("shape of matrix after bow ",title_bow.shape)
```

```
shape of matrix after bow  (109245, 3222)
```

(b) TFIDF Vectorizer

TF-IDF is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

```
1. fromsklearn.feature_extraction.textimportTfidfVectorizer
2. vectorizer=TfidfVectorizer(min_df=10)
3. text_tfidf=vectorizer.fit_transform(preprocessed_essays)
4. print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109245, 16512)
```

```
vectorizer = TfidfVectorizer(min_df=10)
title_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after tfidf ",title_tfidf.shape)
```

```
Shape of matrix after tfidf  (109245, 3222)
```

(c) Using Pretrained Models: TFIDF weighted W2V

In this method first, we will calculate tfidf value of each word. than follow the same approach as above section by multiplying tfidf value with the corresponding word and then divided the sum by sum tfidf value.

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)

dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

```
tfidf_w2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split
())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_title.append(vector)

print(len(tfidf_w2v_title))
print(len(tfidf_w2v_title[0]))
```

```
100%|██████████| 109245/109245 [00:03<00:00, 28026.28it/s]
```

```
109245
300
```

5.2.4 Vectorizing Numerical features

Many machine learning algorithms perform better when numerical input variables are scaled to a standard range.This includes algorithms that use a weighted sum of the input, like linear regression, and algorithms that use distance measures, like k-nearest neighbors.

The two most popular techniques for scaling numerical data prior to modeling are normalization and standardization. **Normalization** scales each input variable separately to the range 0-1, which is the range for floating-point values where we have the most precision. **Standardization** scales each input variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one.

Standardize features by removing the mean and scaling to unit variance

The standard score of a sample x is calculated as: $z = (x - u) / s$

where u is the mean of the training samples or zero if with_mean=False, and s is the standard deviation of the training samples or one if with_std=False.

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using **transform**.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

```
1. From sklearn.preprocessing importStandardScaler
2. price_scalar=StandardScaler()
3. price_scalar.fit(project_data['price'].values.reshape(-1,1))# finding the
   mean and standard deviation of this data
4. print(f"Mean : {price_scalar.mean_[0]}, Standard deviation :
   {np.sqrt(price_scalar.var_[0])}")

5. # Now standardize the data with above maen and variance.
6. price_standardized=price_scalar.transform(project_data['price'].values.reshape
   (-1,1))
```

```
Mean : 298.1152448166964, Standard deviation : 367.49642545627506
```

Merging all the above features

We need to merge all the numerical vectors i.e categorical, text, numerical vectors.

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(school_state_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(grade_category_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
print(num_pro_standardized.shape)
```

```
(109245, 9)
(109245, 30)
(109245, 51)
(109245, 5)
(109245, 4)
(109245, 16512)
(109245, 1)
(109245, 1)
```

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

```
(109245, 16552)
```

5.3 Exploratory Data Analysis

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. Originally developed by American mathematician John Tukey in the 1970s, EDA techniques continue to be a widely used method in the data discovery process today.

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions. EDA can help answer questions about standard deviations, categorical variables, and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modeling, including machine learning.

a) Probability Density Function

Probability density is the relationship between observations and their probability.Some outcomes of a random variable will have low probability density and other outcomes will have a high probability density. The overall shape of the probability density is referred to as a probability distribution, and the calculation of probabilities for specific outcomes of a random

variable is performed by a probability density function, or PDF for short.

It is useful to know the probability density function for a sample of data in order to know whether a given observation is unlikely, or so unlikely as to be considered an outlier or anomaly and whether it should be removed. It is also helpful in order to choose appropriate learning methods that require input data to have a specific probability distribution.

It is unlikely that the probability density function for a random sample of data is known. As such, the probability density must be approximated using a process known as probability density estimation.

To determine the distribution of a discrete random variable we can either provide its PMF or CDF. For continuous random variables, the CDF is well-defined so we can provide the CDF.

However, the PMF does not work for continuous random variables, because for a continuous random variable $P(X=x)=0$ for all $x \in Rx \in R$. Instead, we can usually define the **probability density function (PDF)**.

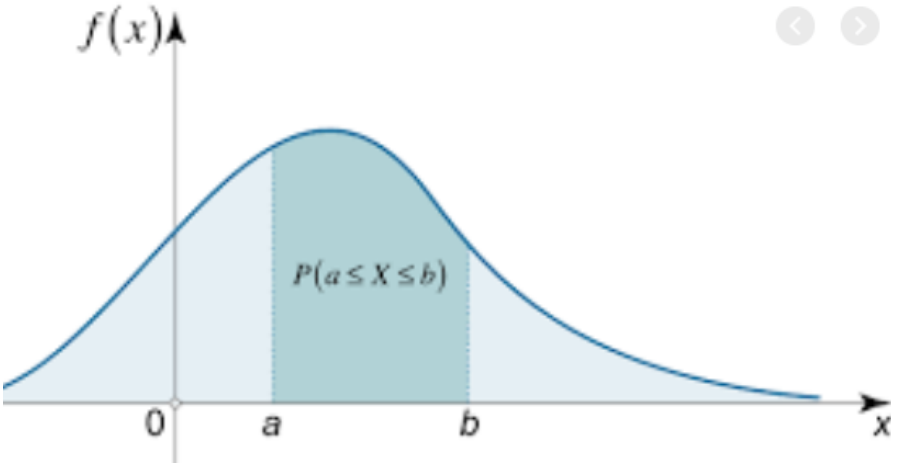
The PDF is the **density** of probability rather than the probability mass. The concept is very similar to mass density in physics: its unit is probability per unit length. To get a feeling for PDF, consider a continuous random variable X and define the function $fX(x)$ as follows (wherever the limit exists):

$$f_X(x) = \lim_{\Delta \to 0^+} P(x < X \leq x + \Delta)\Delta.$$

The function $f_X(x)$ gives us the probability density at point xx. It is the limit of the probability of the interval $(x, x+\Delta]$ divided by the length of the interval as the length of the interval goes to 00. Remember *that* $P(x<X \leq x+\Delta) = FX(x+\Delta) - FX(x)$

$$f_X(x) = \lim_{\Delta \to 0} \frac{F_X(x+\Delta) - F_X(x)}{\Delta}$$

$$= \frac{dF_X(x)}{dx} = F'_X(x), \qquad \text{if } F_X(x) \text{ is differentiable at } x.$$



Thus, we have definition for the PDF of continuous random variables.

b). Bar Plots

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

Bar plots need not be based on counts or frequencies. You can create bar plots that represent means, medians, standard deviations, etc. Use the aggregate( ) function and pass the results to the barplot( ) function.

Creating a bar plot

The **matplotlib** API in Python provides the bar() function which can be used in

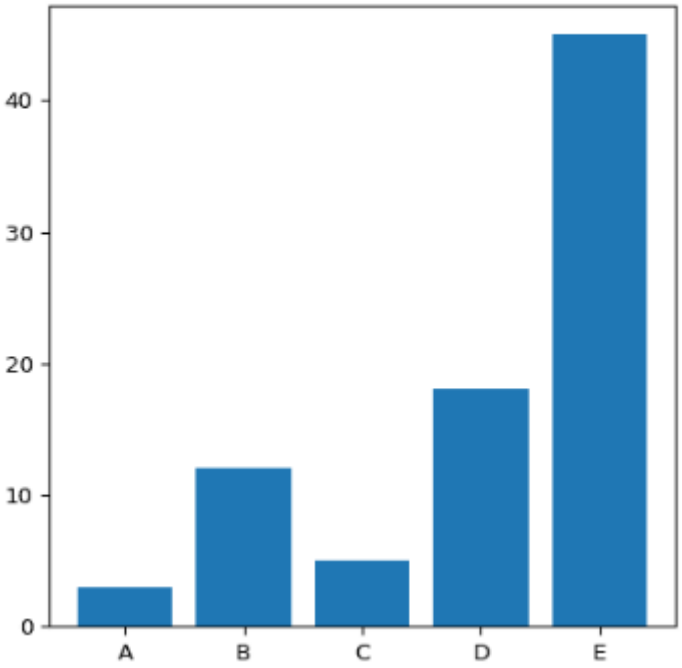
MATLAB style use or as an object-oriented API. The syntax of the bar() function to be used with the axes is as follows:-

plt.bar(x, height, width, bottom, align)

The function creates a bar plot bounded with a rectangle depending on the given parameters. Following is a simple example of the bar plot, which represents the number of students enrolled in different courses of an institute.

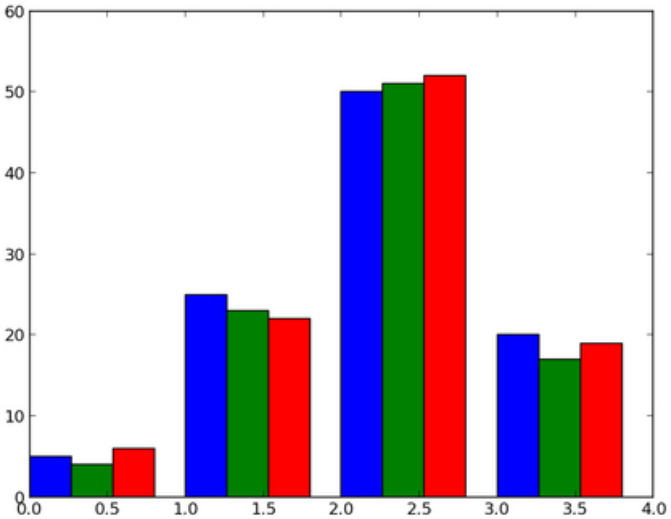
The code to generate a simple bar plot using the matplotlib library can be found below

```
1. import numpy as np
2. import matplotlib.pyplot as plt
3. # Make a random dataset:
4. height = [3, 12, 5, 18, 45]
5. bars = ('A', 'B', 'C', 'D', 'E')
6. y_pos = np.arange(len(bars))
7. # Create bars
8. plt.bar(y_pos, height)
9. # Create names on the x-axis
10.plt.xticks(y_pos, bars)
11.# Show graphic
12.plt.show()
```



Multiple bar plots

Multiple bar plots are used when comparison among the data set is to be done when one variable is changing. We can easily convert it as a stacked area bar chart, where each subgroup is displayed by one on top of the others. It can be plotted by varying the thickness and position of the bars.



c). Donut Plot

The Donut plot is similar to a pie chart, except it has a hole in the middle similar to a donut. In this article, I will walk you through how to create a Donut Plot with the Python programming language.

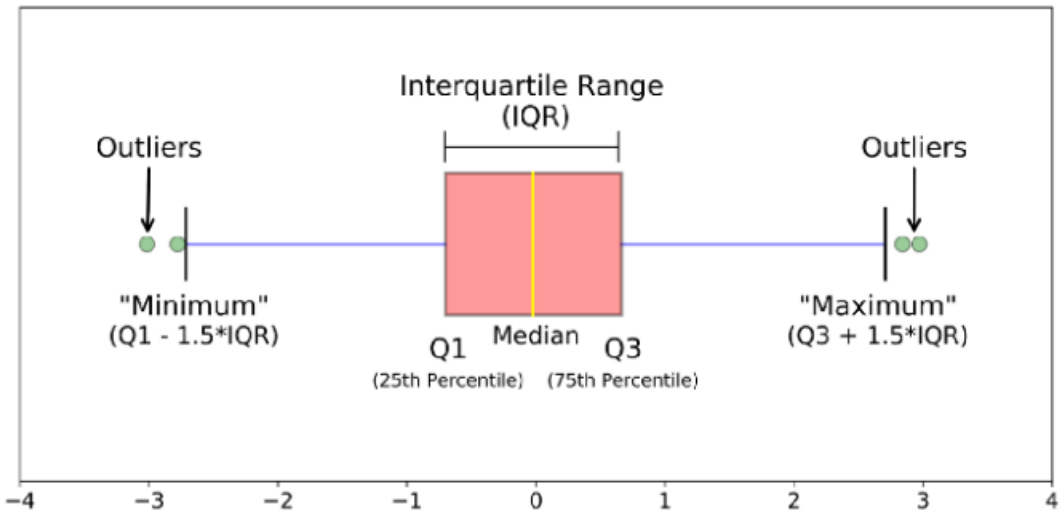
Matplotlib allows to build a pie-chart easily thanks to its pie() function. We can use the exact same principle and add a circle to the center thanks to thecircle() function and get a donut chart.

```
1. # library
2. import matplotlib.pyplot as plt

3. # create data
4. size_of_groups=[12,11,3,30]

5. # Create a pieplot
6. plt.pie(size_of_groups)

7. # add a circle at the center to transform it in a donut chart
8. my_circle=plt.Circle( (0,0), 0.7, color='white')
9. p=plt.gcf()
10.   p.gca().add_artist(my_circle)

11.   plt.show()
```

d). Boxplot

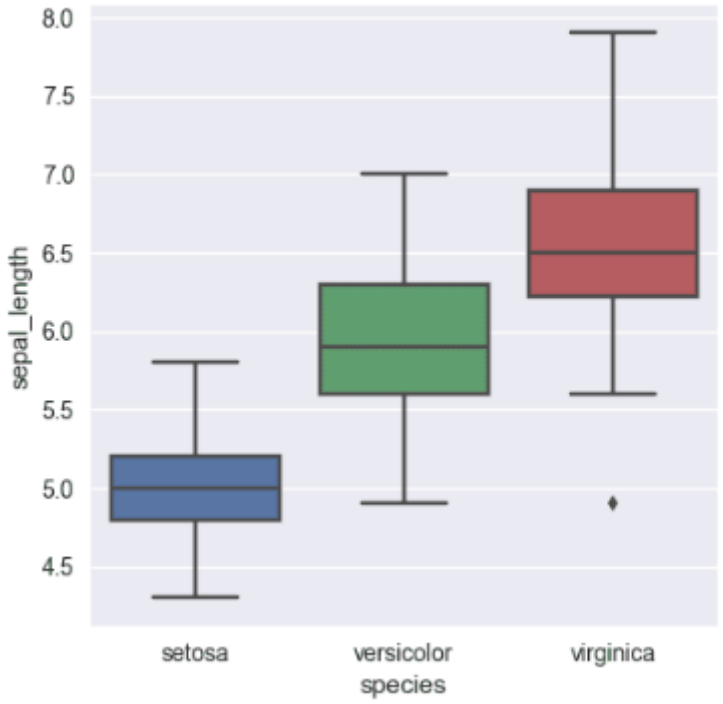
A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum"). It can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.

A boxplot is a graph that gives you a good indication of how the values in the data are spread out. Although boxplots may seem primitive in comparison to a histogram or density plot, they have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets.



Different parts of a boxplot

```
1. # library & dataset
2. import seaborn as sns
3. df = sns.load_dataset('iris')

4. sns.boxplot( x=df["species"], y=df["sepal_length"] )
```



e). Violin Plot

A **violin plot** is a method of plotting numeric data. It is similar to a box plot, with the addition of a rotated kernel density plot on each side.

Violin plots are similar to box plots, except that they also show the probability density of the data at different values, usually smoothed by a kernel density estimator. Typically a violin plot will include all the data that is in a box plot: a marker for the median of the data; a box or marker indicating the interquartile range; and possibly all sample points, if the number of samples is not too high.

Recently it was shown that the violin plot called Mirrored Density plot (MD plot) outperforms conventional violin plots in terms of the identification of interesting structures in data in the programming languages of R and Python. Violin plots are available as extensions to a number of software packages such as Data Visualization on CRAN and the md-plot package on PyPI.

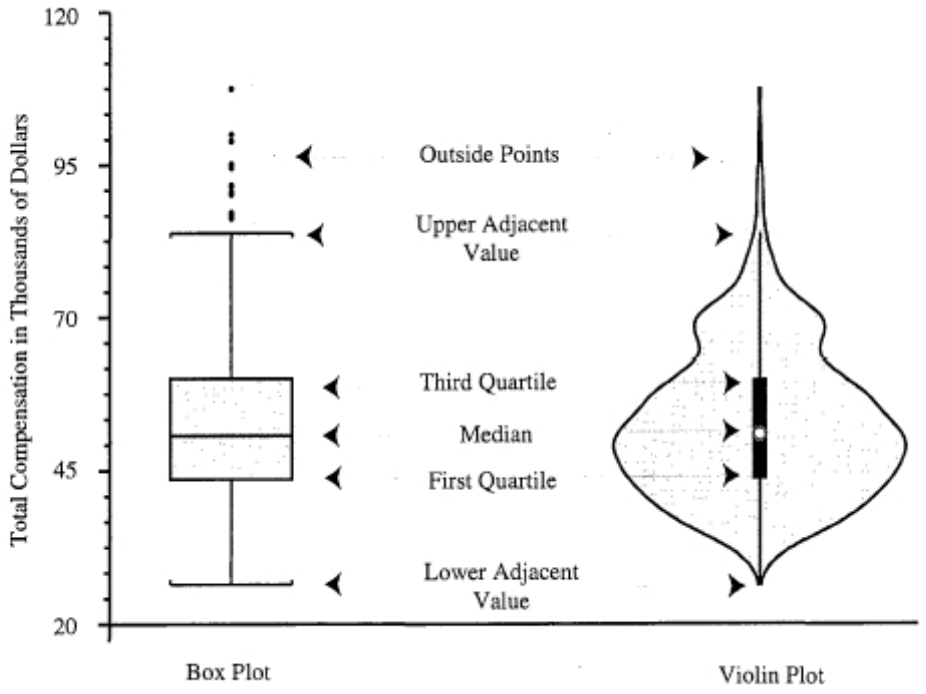
A violin plot is more informative than a plain box plot. While a box plot only shows

summary statistics such as mean/median and inter-quartile ranges, the violin plot shows the full distribution of the data. The difference is particularly useful when the data distribution is multimodal (more than one peak). In this case a violin plot shows the presence of different peaks, their position and relative amplitude.

Like box plots, violin plots are used to represent comparison of a variable distribution (or sample distribution) across different "categories" (for example, temperature distribution compared between day and night, or distribution of car prices compared across different car makers).

A violin plot can have multiple layers. For instance, the outer shape represents all possible results. The next layer inside might represent the values that occur 95% of the time. The next layer (if it exists) inside might represent the values that occur 50% of the time.

In general, violin plots are a method of plotting numeric data and can be considered a combination of the box plot with a kernel density plot. In the violin plot, we can find the same information as in the box plots



f). Word Cloud

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.

For generating word cloud in Python, modules needed are – matplotlib, pandas and wordcloud.

The wordcloud library is here to help you build a wordcloud in minutes. Here is a basic code snippets using the WordCloud() function to get you started.

```
1. # Libraries
2. from wordcloud import WordCloud
3. import matplotlib.pyplot as plt

4. # Create a list of word
5. text=("Python Python Python Matplotlib")

6. # Create the wordcloud object
7. wordcloud = WordCloud(width=480, height=480,
   margin=0).generate(text)

8. # Display the generated image:
9. plt.imshow(wordcloud, interpolation='bilinear')
10.  plt.axis("off")
11.  plt.margins(x=0, y=0)
12.  plt.show()
```

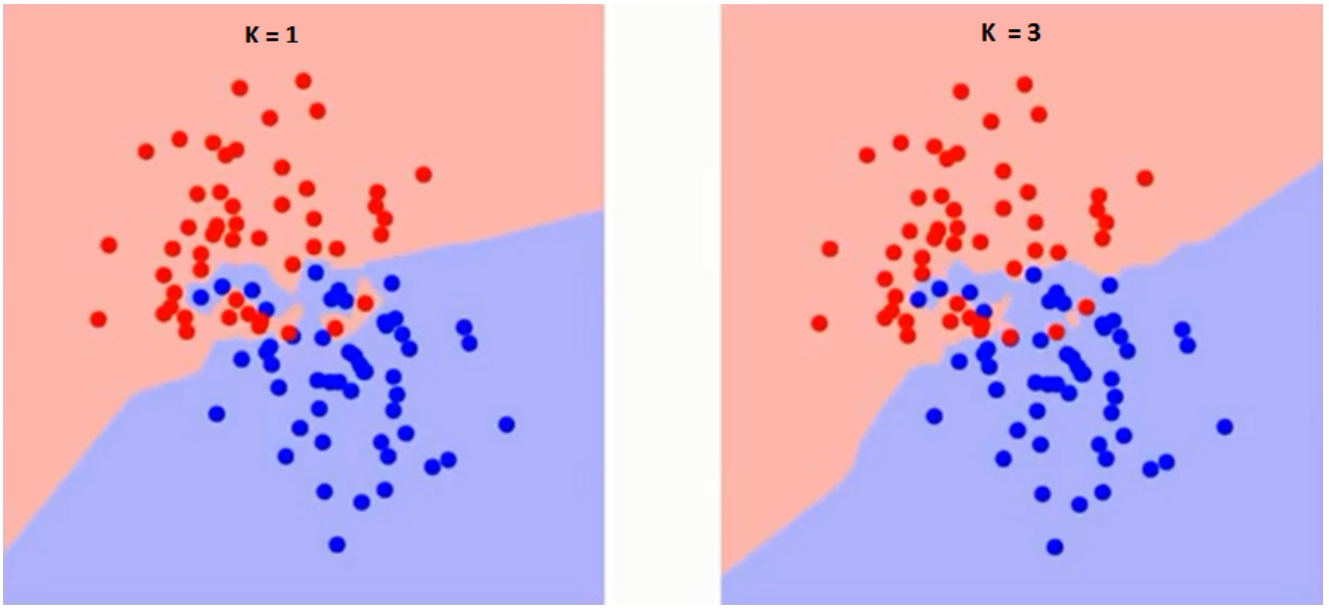
5.4 Training Models:

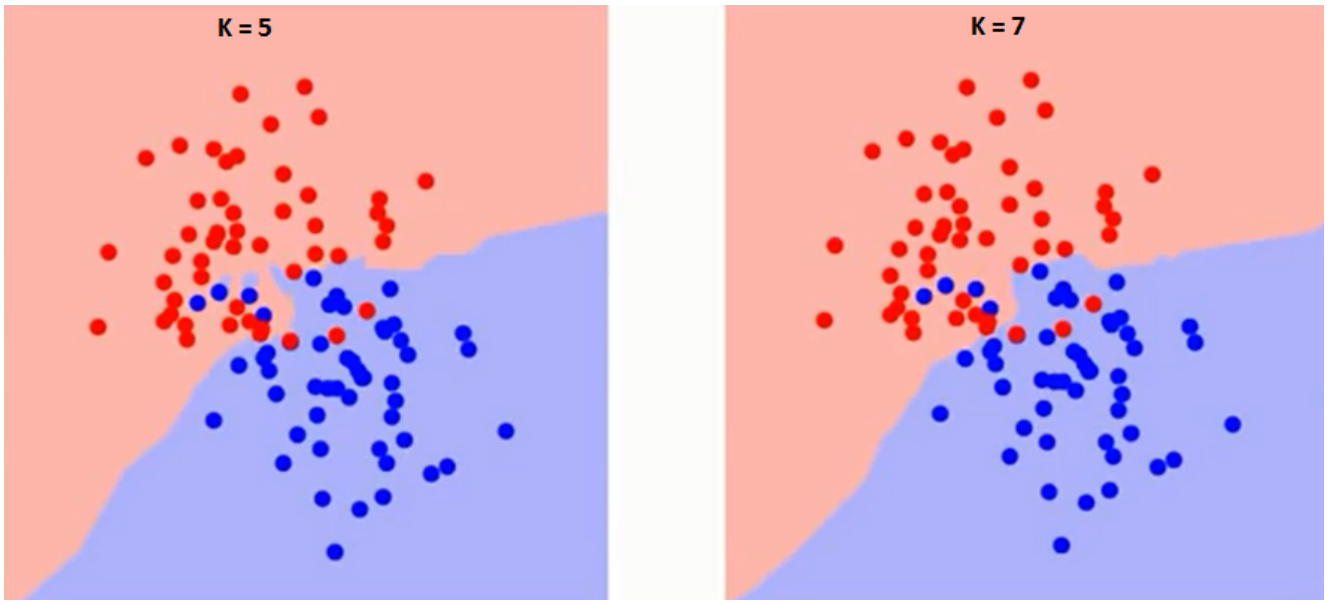
5.4.1 KNN

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output
2. Calculation time
3. Predictive Power

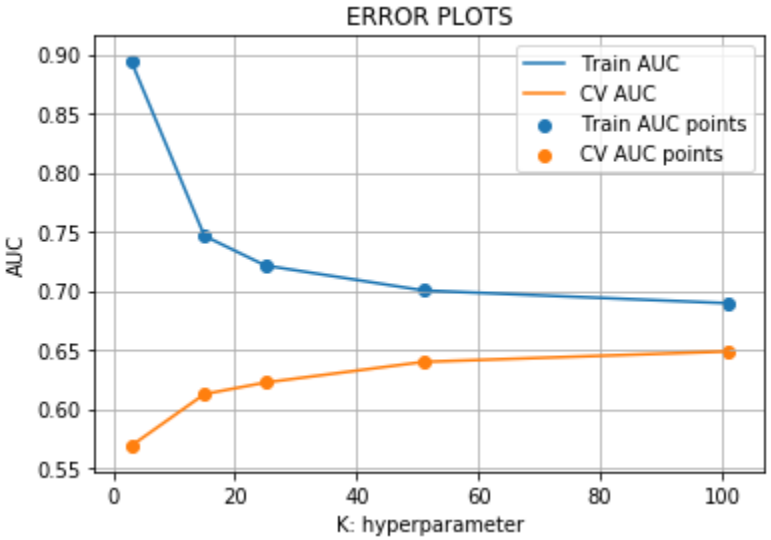
First let us try to understand what exactly does K influence in the algorithm. If we see the last example, given that all the 6 training observation remain constant, with a given K value we can make boundaries of each class. These boundaries will segregate RC from GS. In the same way, let's try to see the effect of value "K" on the class boundaries. The following are the different boundaries separating the two classes with different values of K.





In machine learning, **hyperparameter optimization** or **tuning** is the problem of choosing a set of optimal **hyperparameters** for a learning algorithm. A **hyperparameter** is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

```
1. def batch_predict(clf, data):
2.     # roc_auc_score(y_true, y_score) the 2nd
   parameter should be probability estimates of the
   positive class
3.     # not the predicted outputs
4.
5.     y_data_pred = []
6.     tr_loop = data.shape[0] - data.shape[0]%1000
7.     # consider you X_tr shape is 49041, then your
   tr_loop will be 49041 - 49041%1000 = 49000
8.     # in this for loop we will iterate unti the last
   1000 multiplier
9.     for i in range(0, tr_loop, 1000):
10.
   y_data_pred.extend(clf.predict_proba(data[i:i+1000])
   [:,1])
11.       # we will be predicting for the last data
   points
12.       if data.shape[0]%1000 !=0:
13.
   y_data_pred.extend(clf.predict_proba(data[tr_loop:])
   [:,1])
14.
15.       return y_data_pred
```



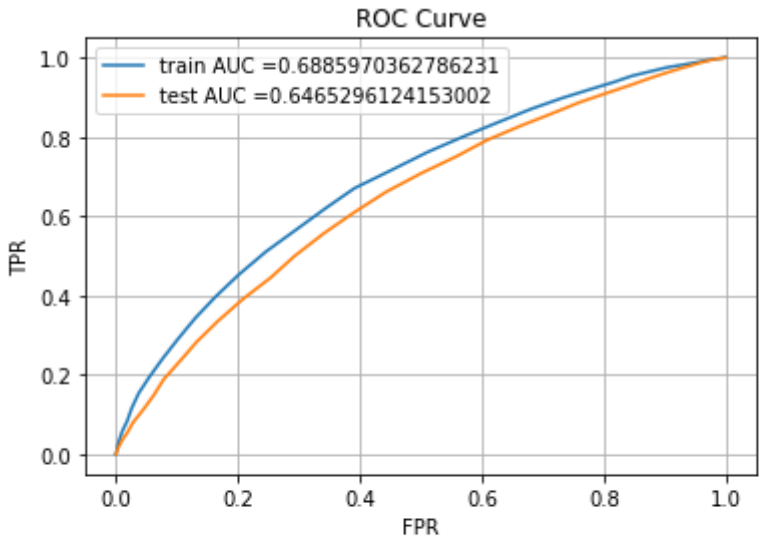
Analysis: By observing the above AUC vs K plot we can observe that the curve for CV data is not tending ti increase considerably after k=80. So since the value of AUC must be high. Therefore choosing k to be 90.

K = 95

Testing the performance

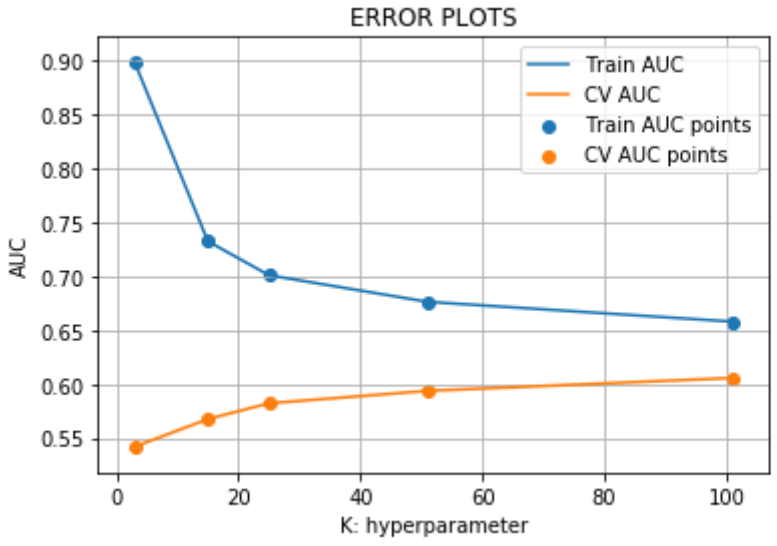
```
from sklearn.metrics import roc_curve, auc
from sklearn.neighbors import KNeighborsClassifier

best_k = 95
neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(new_Xtrain, y_train)

y_train_pred = batch_predict(neigh, new_Xtrain)
y_test_pred = batch_predict(neigh, new_Xtest)

 train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
 test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

 plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
```

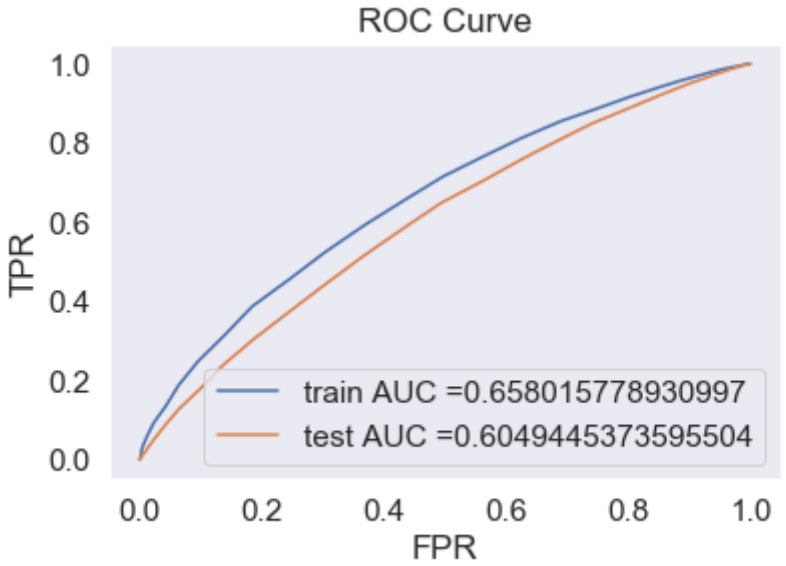
```
15.  plt.plot(test_fpr, test_tpr, label="test AUC
  ="+str(auc(test_fpr, test_tpr)))
16.  plt.legend()
17.  plt.xlabel("FPR")
18.  plt.ylabel("TPR")
19.  plt.title("ROC Curve")
20.  plt.grid()
21.  plt.show()
```



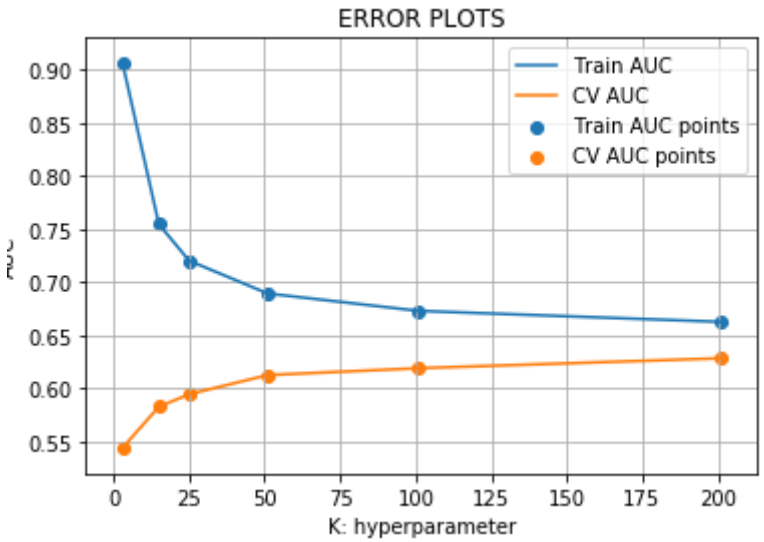
The AUC score for test data is 0.64 which is better than 0.5 i.e score for a random model.

Applying KNN brute force on TFIDF





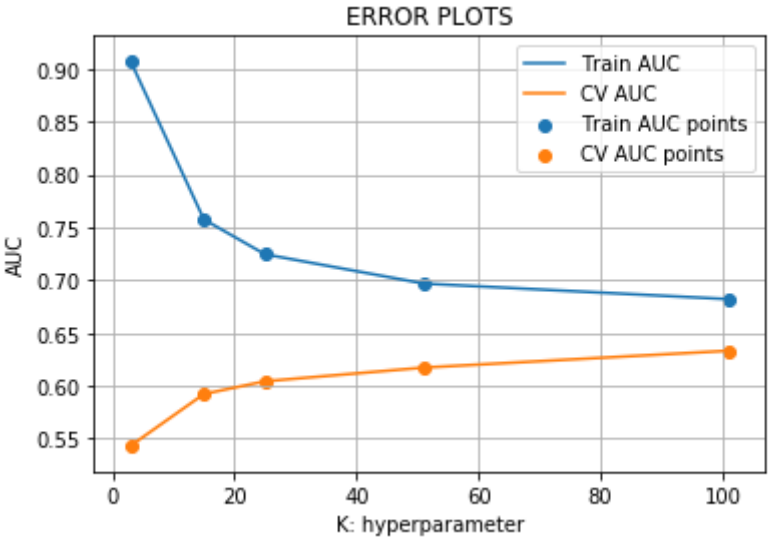
Applying KNN brute force on AVG W2V

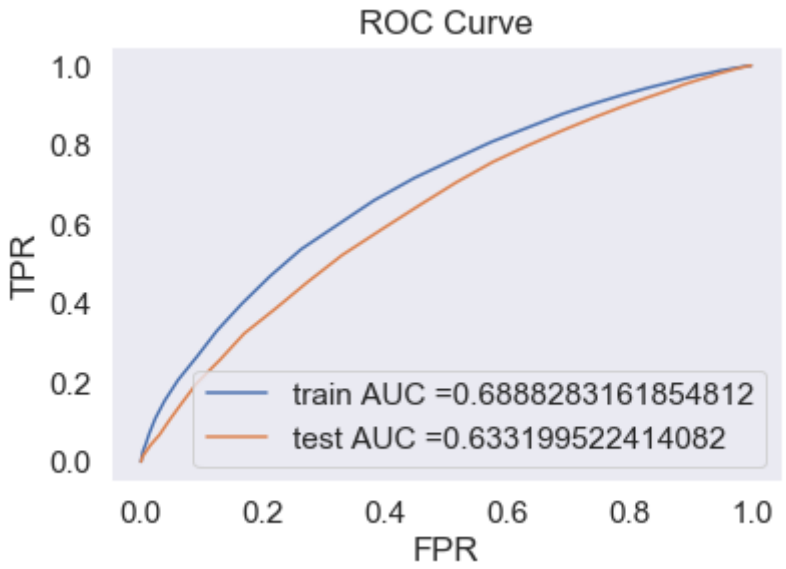


Analysis: Since from the above plot we can observe that there is not much change in the AUC values from 100 to 200. So to reduce the time of execution taking K as 101.

K = 101

Applying KNN Brute force on TFIDF W2V





Conclusions :

Vectorizing Technique	Hyperparameter	AUC Score
BOW	95	0.64
TFIDF	101	0.604
Avg W2V	101	0.624
TFIDF W2V	105	0.633

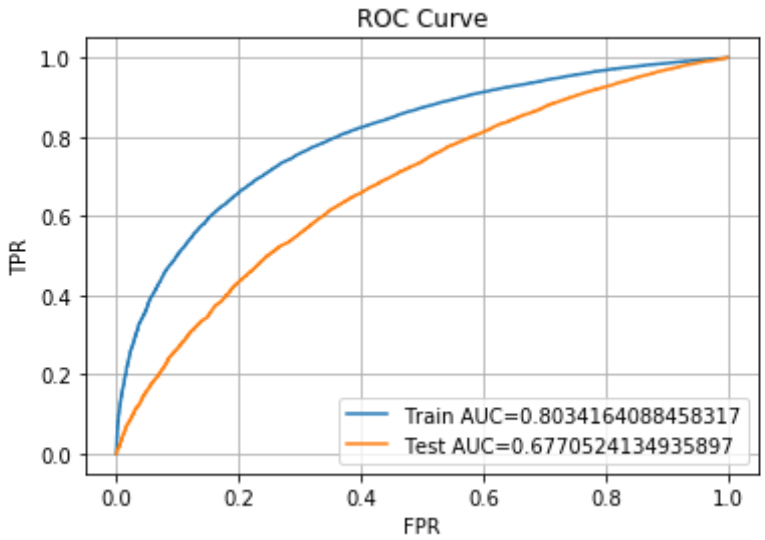
5.4.2 Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist.
In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

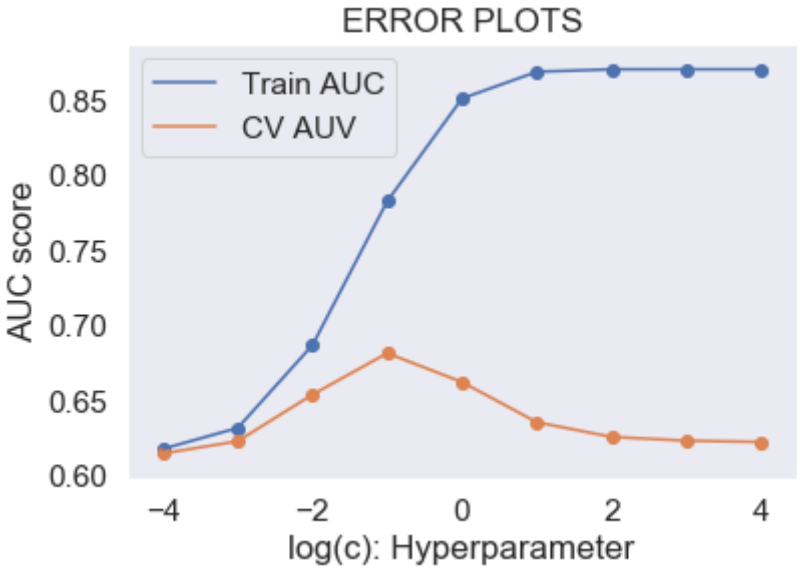
We will be applying similar rules and using it based on the same approach as we did for KNN model where we used different encoding and categorization techniques.

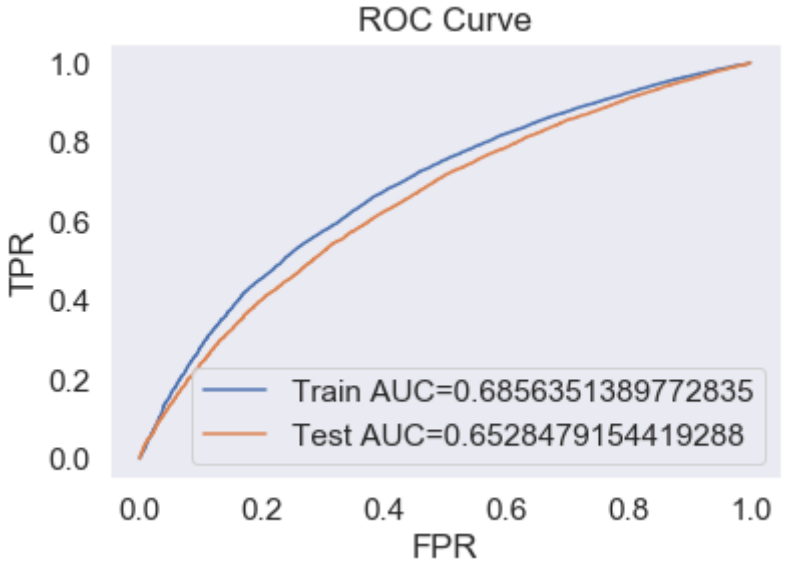
1. Bag Of Words
2. TFIDF
3. Avg W2V
4. TFIDF W2V
5. No text data

1. Bag of Words

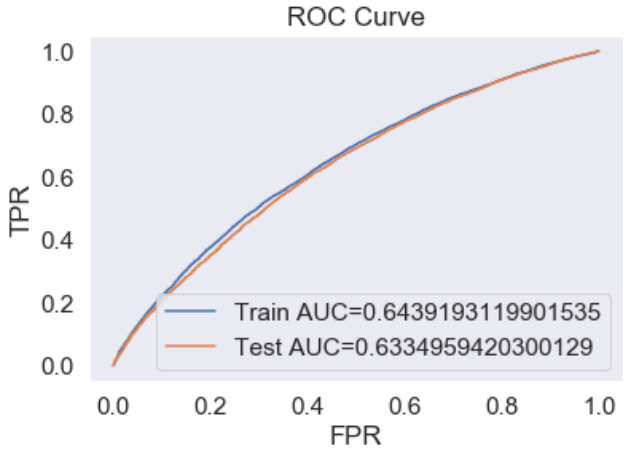


2. TFIDF


ERROR PLOTS
Train AUC
CV AUV
AUC score
0.85
0.80
0.75
0.70
0.65
0.60
−4
−2
0
2
4
log(c): Hyperparameter


ROC Curve
TPR
1.0
0.8
0.6
0.4
0.2
0.0
Train AUC=0.6856351389772835
Test AUC=0.6528479154419288
0.0
0.2
0.4
0.6
0.8
1.0
FPR

3. TFIDF W2V


ROC Curve
TPR
1.0
0.8
0.6
0.4
0.2
0.0
Train AUC=0.6439193119901535
Test AUC=0.6334959420300129
0.0
0.2
0.4
0.6
0.8
1.0
FPR

Conclusion

Vectorizing Technique	Hyperparameter	AUC Score
BOW	0.01	0.677
TFIDF	0.1	0.652
Avg W2V	1	0.687
TFIDF W2V	0.1	0.698
No Text data	0.001	0.633

5.4.3 SVM

S**upport-vector machines** (**SVMs**, also **support-vector networks**[]) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues (Boser et al., 1992, Guyon et al., 1993, Vapnik et al., 1997), SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik and Chervonenkis (1974) and Vapnik (1982, 1995). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The **support-vector clustering** algorithm, applies the statistics of support vectors, developed in the support vector machines algorithm, to categrorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

SVMs belong to a family of generalized linear classifiers and can be interpreted as an extension of the perceptron. They can also be considered a special case of Tikhonov regularization. A special property is that they simultaneously minimize the empirical *classification error* and maximize the *geometric margin*; hence they are also known as **maximum margin classifiers**.

Implementation

The parameters of the maximum-margin hyperplane are derived by solving the optimization. There exist several specialized algorithms for quickly solving the quadratic programming (QP) problem that arises from SVMs, mostly relying on heuristics for breaking the problem down into smaller, more manageable chunks.

Another approach is to use an interior-point method that uses Newton-like iterations to find a solution of the Karush–Kuhn–Tucker conditions of the primal and dual problems. Instead of solving a sequence of broken-down problems, this approach directly solves the problem altogether. To avoid solving a linear system involving the large kernel matrix, a low-rank approximation to the matrix is often used in the kernel trick.

Another common method is Platt's sequential minimal optimization (SMO) algorithm, which breaks the problem down into 2-dimensional sub-problems that are solved analytically, eliminating the need for a numerical optimization algorithm and matrix storage. This algorithm is conceptually simple, easy to implement, generally faster, and has better scaling properties for difficult SVM problems.

The special case of linear support-vector machines can be solved more efficiently by the same kind of algorithms used to optimize its close cousin, logistic regression; this class of algorithms includes sub-gradient descent (e.g., PEGASOS) and coordinate descent (e.g., LIBLINEAR). LIBLINEAR has some attractive training-time properties. Each convergence iteration takes time linear in the time taken to read the train data, and the iterations also have a Q-linear convergence property, making the algorithm extremely fast

.

The general kernel SVMs can also be solved more efficiently using sub-gradient descent (e.g. P-packSVM), especially when parallelization is allowed.
Kernel SVMs are available in many machine-learning toolkits, including LIBSVM, MATLAB, SAS,

. .

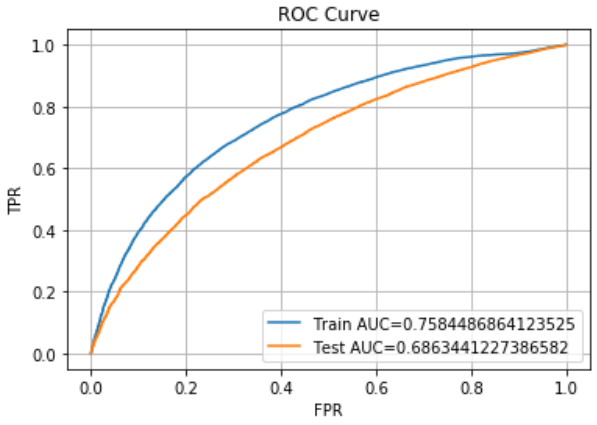
SVMlight, kernlab, scikit-learn, Shogun, Weka, Shark, JKernelMachines, OpenCV and others. Preprocessing of data (standardization) is highly recommended to enhance accuracy of classification.

There are a few methods of standardization, such as min-max, normalization by decimal scaling, Z-score. Subtraction of mean and division by variance of each feature is usually used for SVM.

We will be applying similar rules and using it based on the same approach as we did for KNN model where we used different encoding and categorization techniques.

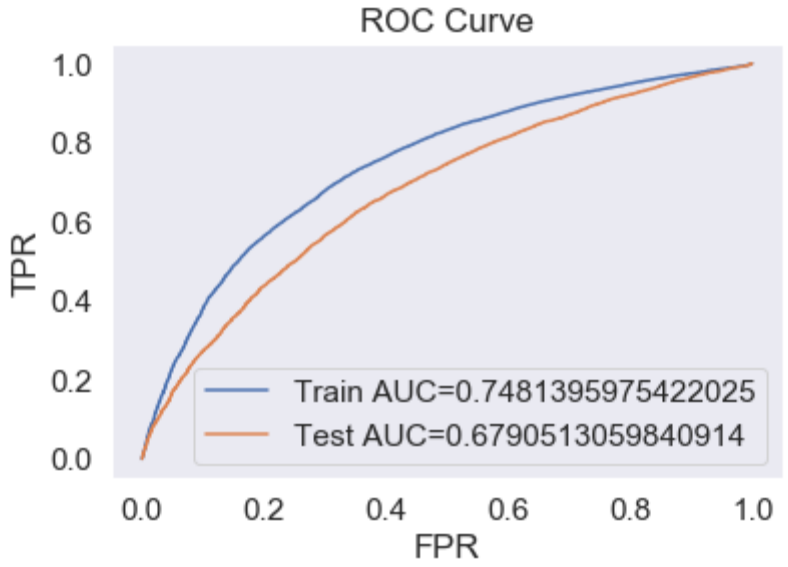
1. Bag Of Words
2. TFIDF
3. Avg W2V
4. TFIDF W2V
5. No text data

    1. Bag of words

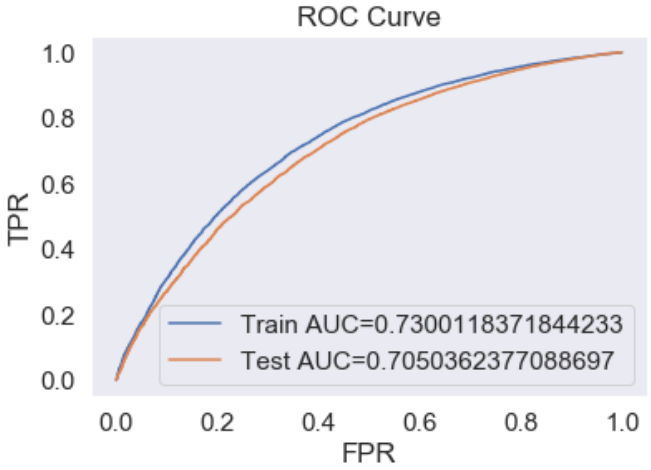


The AUC score obtained is 0.683

2. TFIDF



3. **Avg W2V**



As we can observe from the above plot that the AUC score for test data is 0.705 which is better than a random model.

Conclusion

Vectorizing Technique	Hyperparameter(alpha)	Regularizer	AUC Score
BOW	0.01	l2	0.683
TFIDF	0.0001	l1	0.679
Avg W2V	0.0001	l1	0.705
TFIDF W2V	0.001	l2	0.703
Set 5	0.0001	l1	0.676

5.4.4 Decision Tree

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called **classification trees**; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called **regression trees**.

Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making). This page deals with decision trees in data mining.

Decision tree learning is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables.

A decision tree is a simple representation for classifying examples. For this section, assume that all of the input features have finite discrete domains, and there is a single target feature called the "classification". Each element of the domain of the classification is called a *class*. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with an input feature are labeled with each of the possible values of the target feature or the arc leads to a subordinate decision

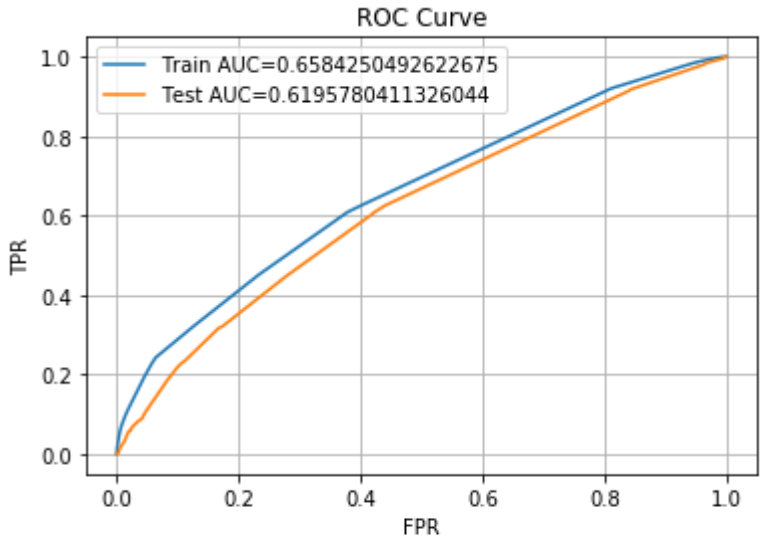
node on a different input feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes, signifying that the data set has been classified by the tree into either a specific class, or into a particular probability distribution (which, if the decision tree is well-constructed, is skewed towards certain subsets of classes).

A tree is built by splitting the source set, constituting the root node of the tree, into subsets—which constitute the successor children. The splitting is based on a set of splitting rules based on classification features. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node has all the same values of the target variable, or when splitting no longer adds value to the predictions. This process of *top-down induction of decision trees* (TDIDT) is an example of a greedy algorithm, and it is by far the most common strategy for learning decision trees from data.

In data mining, decision trees can be described also as the combination of mathematical and computational techniques to aid the description, categorization and generalization of a given set of data.

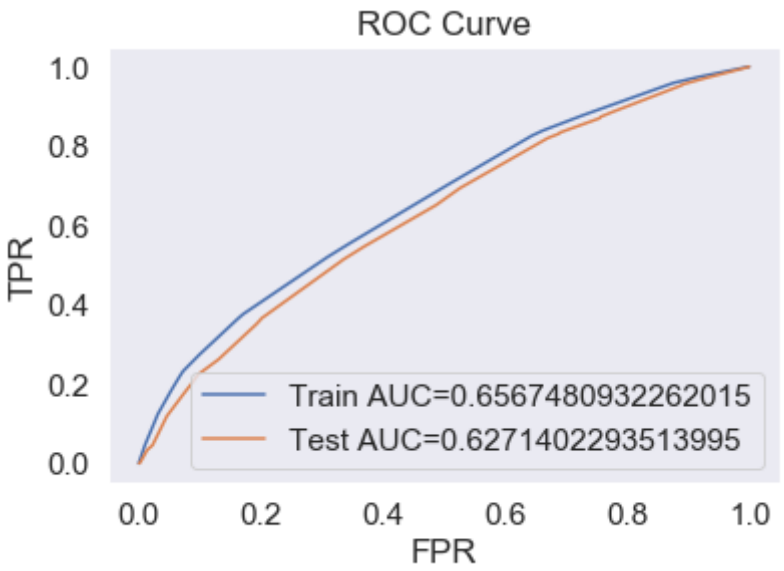
Appling Decision Tree on different kind of featurization

1. **TFIDF**



As we can observe from the above plot that the AUC score is 0.619

2. **TFIDF W2V**



Vectorizing Technique	Max Depth	Min Samples Split	AUC Score
TFIDF	10	500	0.6195
TFIDF W2V	5	500	0.6271
Set 5	10	500	0.624

5.4.5 Random Forest

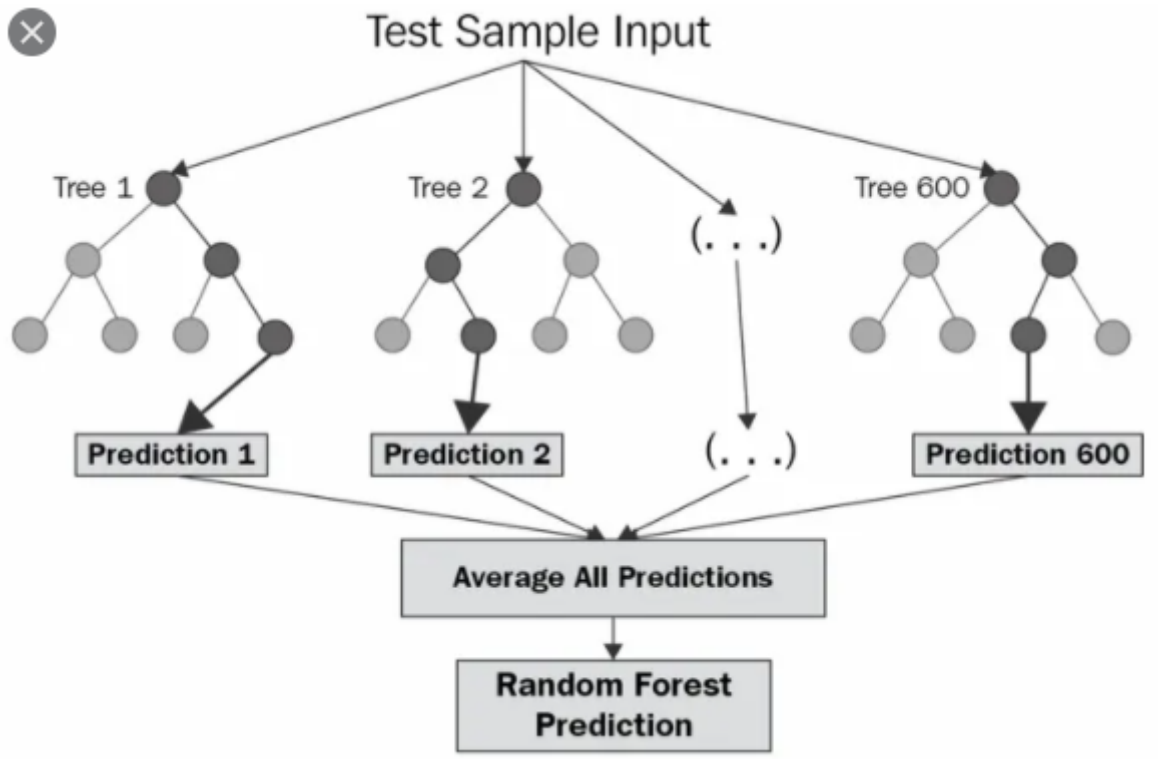
Random forests or **random decision forests** are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.). The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho and later independently by Amit and Geman in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.



Conclusion

Rand_Forest - MODEL	HyperparameterS		Train_AUC	Test_Auc
BOW-ENC-RF	Depth:10	Samp_Split:250	0.67	0.63
TFIDF-ENC-RF	Depth:10	Samp_Split:250	0.66	0.62
AvgW2V-ENC-RF	Depth:5	Samp_Split:500	0.64	0.6
Tf-Idf-ENC-RF	Depth:5	Samp_Split:250	0.65	0.61

5.4.6 Gradient Boosting Decision Tree

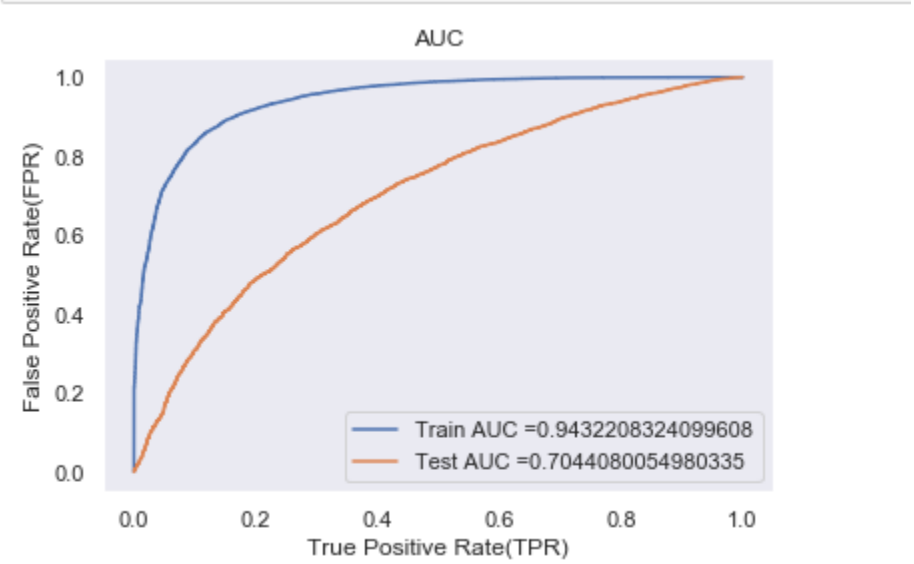
Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Implementation

Fitting the training set too closely can lead to degradation of the model's generalization ability. Several so-called regularization techniques reduce this overfitting effect by constraining the fitting procedure.

One natural regularization parameter is the number of gradient boosting iterations *M* (i.e. the number of trees in the model when the base learner is a decision tree). Increasing *M* reduces the error on training set, but setting it too high may lead to overfitting. An optimal value of *M* is often selected by monitoring prediction error on a separate validation data set. Besides controlling *M*, several other regularization techniques are used.

Another regularization parameter is the depth of the trees. The higher this value the more likely the model will overfit the training data.



Conclusion

GBDT - MODEL	HyperparameterS	Train_AUC	Test_Auc
BOW-ENC-GBDT	Depth:5 | Samp_Split:250	0.94	0.81
TFIDF-ENC-GBDT	Depth:10 | Samp_Split:250	0.92	0.78
AvgW2V-ENC-GBDT	Depth:5 | Samp_Split:50	0.88	0.77
Tf-Idf-ENC-GBDT	Depth:5 | Samp_Split:100	0.88	0.79

6. TESTING

6.1 INTRODUCTION TO VARIOUS METRICS

For evaluating the performance of our machine learning models, we make use of various metrics that give us an estimate of how well our model works both on the train data and test data. Some of the popularly used metrics are listed below

1. Classification accuracy
2. Logarithmic loss
3. Confusion Matrix
4. Area under Curve (AUC)
5. F1 score
6. Mean Absolute error
7. Mean Squared error

Evaluating your machine learning algorithm is an essential part of any project. Your model may give you satisfying results when evaluated using a metric say accuracy_score but may give poor results when evaluated against other metrics such as logarithmic_loss or any other such metric.

Therefore choosing an appropriate metric is quite important and is a very essential domain knowledge task. After having a look over all the metrics, we finally come to a conclusion of using a pair of metrics that is shown later.

Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples. It works well only if there are equal number of samples belonging to each class.

For example, consider that there are 98% samples of class A and 2% samples of class B in our training set. Then our model can easily get 98% training accuracy by simply predicting every training sample belonging to class A.

When the same model is tested on a test set with 60% samples of class A and 40% samples of class B, then the test accuracy would drop down to 60%. Classification Accuracy is great, but gives us the false sense of achieving high accuracy.

Logarithmic Loss or Log Loss, works by penalising the false classifications. It works well for multi-class classification. When working with Log Loss, the classifier must assign probability to each class for all the samples. Suppose, there are N samples belonging to M classes, then the Log Loss is calculated as below :

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$

Log Loss has no upper bound and it exists on the range [0, ∞). Log Loss nearer to 0 indicates higher accuracy, whereas if the Log Loss is away from 0 then it indicates lower accuracy.

Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model.

Lets assume we have a binary classification problem. We have some samples belonging to two classes : YES or NO. Also, we have our own classifier which predicts a class for a given input sample. On testing our model on 165 samples ,we get the following result.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

There are 4 important terms :

- **True Positives** : The cases in which we predicted YES and the actual output was also YES.
- **True Negatives** : The cases in which we predicted NO and the actual output was NO.
- **False Positives** : The cases in which we predicted YES and the actual output was NO.
- **False Negatives** : The cases in which we predicted NO and the actual output was YES.

Area Under Curve(AUC) is one of the most widely used metrics for evaluation. It is used for

binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. Before defining AUC, let us understand two basic terms :

True Positive Rate (Sensitivity) : True Positive Rate is defined as TP/ (FN+TP). True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.

$$TruePositiveRate = \frac{TruePositive}{FalseNegative + TruePositive}$$

True Negative Rate (Specificity) : True Negative Rate is defined as TN / (FP+TN). False Positive Rate corresponds to the proportion of negative data points that are correctly considered as negative, with respect to all negative data points.

$$TrueNegativeRate = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

False Positive Rate : False Positive Rate is defined as FP / (FP+TN). False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

$$FalsePositiveRate = \frac{FalsePositive}{TrueNegative + FalsePositive}$$

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances). High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model.

F1 Score tries to find the balance between precision and recall.

Mean Absolute Error is the average of the difference between the Original Values and the Predicted Values. It gives us the measure of how far the predictions were from the actual output. However, they don't gives us any idea of the direction of the error i.e. whether we are under

predicting the data or over predicting the data.

Mean Squared Error(MSE) is quite similar to Mean Absolute Error, the only difference being that MSE takes the average of the square of the difference between the original values and the predicted values. The advantage of MSE being that it is easier to compute the gradient, whereas Mean Absolute Error requires complicated linear programming tools to compute the gradient. As, we take square of the error, the effect of larger errors become more pronounced then smaller error, hence the model can now focus more on the larger errors.

Now after analyzing all the popular metrics, we came to a conclusion that the metrics that best satisfy the constraints of our problem are AUC score and confusion matrix.

6.2 TESTING IN VARIOUS MODELS

First, lets consider out base model i.e., K Nearest Neighbors.

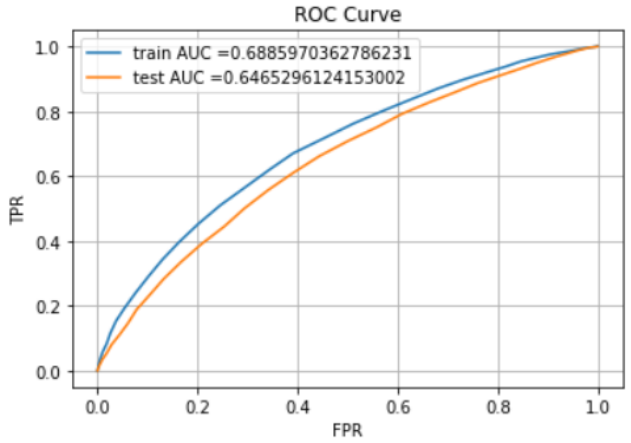
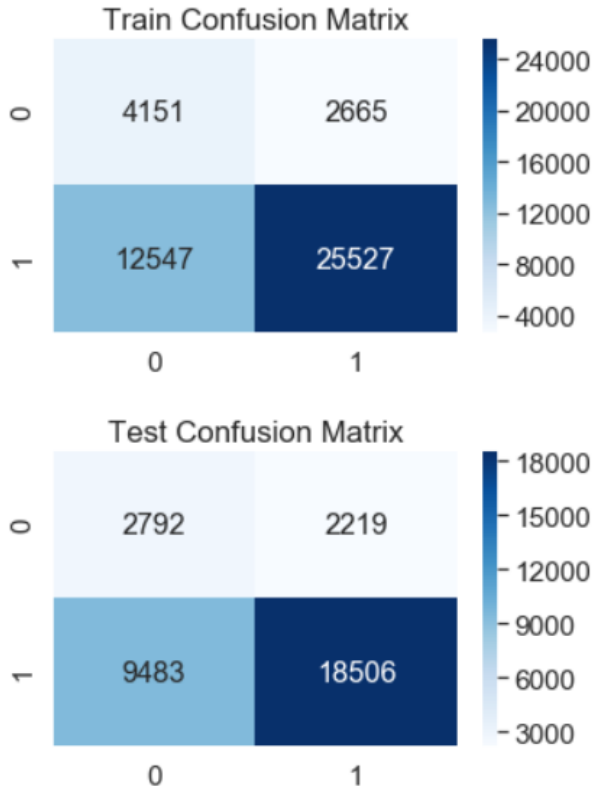


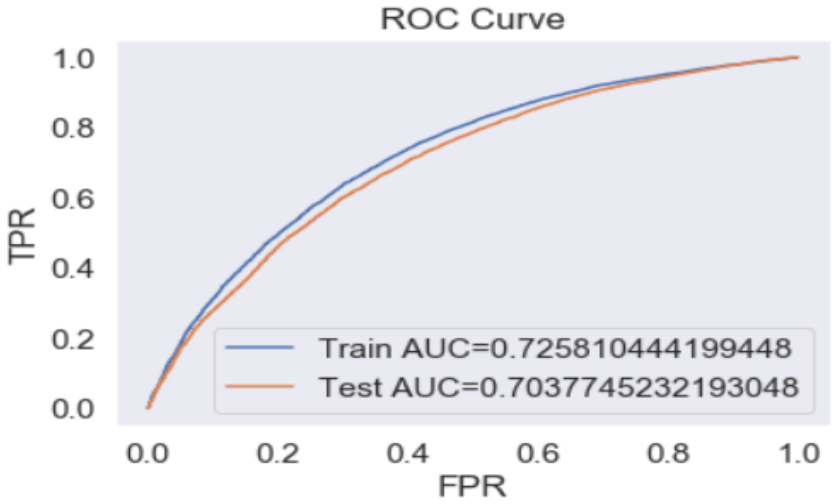
Figure x shows the ROC (receiver operating characteristic) curve for our KNN model. By observing, we can see that the train AUC score is 0.688 whereas the test AUC score is 0.646. These two values are greater than 0.5 and hence we can conclude that our model is performing better than a completely random model. Also, the two scores are considerably close and hence we can say that our model is neither overfitting nor under fitting.

The confusion matrix is shown in the below figure x.

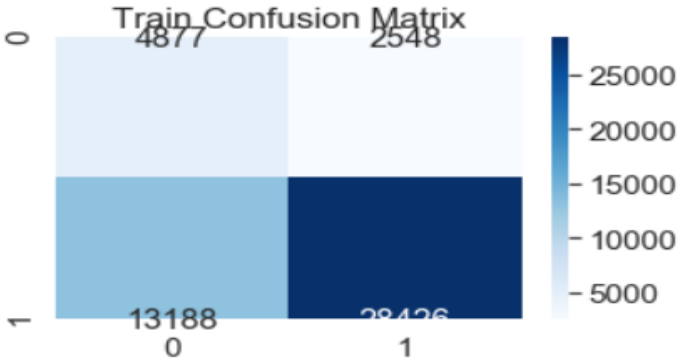


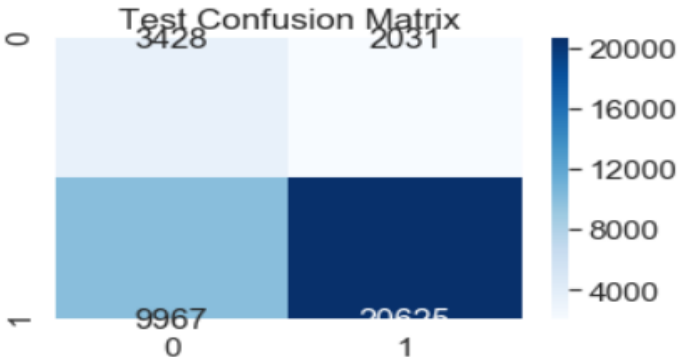
Since the confusion matrix is color coded, we can observe the various densities of points and can come to a conclusion on the precision and recall of the model. The box [1,1] is heavily dense. This means that our model is able to correctly predict the majority of the approved projects.

Similarly, below figures fig x and fig x show the ROC curve and confusion matrix of SVM model


ROC Curve
TPR
FPR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Train AUC=0.725810444199448
Test AUC=0.7037745232193048

\


Train Confusion Matrix
4877
2548
13188
0
1
0
1
25000
20000
15000
10000
5000


Test Confusion Matrix
3428
2031
9967
0
1
0
1
20000
16000
12000
8000
4000

7. RESULTS

After extensive training and performing hyperparameter tuning to improve the AUC score, we were finally able to arrive at the below best results

FINAL CONCLUSIONS

In [5]:
```
from prettytable import PrettyTable
TB = PrettyTable()

TB.field_names = ["MODEL", "ENCODING", "AUC-SCORE"]
TB.add_row(["K NEAREST NEIGH.", "TFIDF-W2V",0.633])
TB.add_row(["LOGISTIC REGRESSION", "TFIDF-W2V" ,0.698])
TB.add_row(["SUPPORT VECTOR MACHINE", "AVG-W2V",0.705])
TB.add_row(["DECISION TREES", "TFIDF-W2V", 0.627])
TB.add_row(["RANDOM FORESTS", "BOW", 0.63])
TB.add_row(["GRADIENT BOOSTED DECISION TREES", "BOW", 0.81])
print(TB)
```

```
+---------------------------------+-----------+-----------+
|              MODEL              |  ENCODING | AUC-SCORE |
+---------------------------------+-----------+-----------+
|         K NEAREST NEIGH.        | TFIDF-W2V |   0.633   |
|       LOGISTIC REGRESSION       | TFIDF-W2V |   0.698   |
|      SUPPORT VECTOR MACHINE     |  AVG-W2V  |   0.705   |
|          DECISION TREES         | TFIDF-W2V |   0.627   |
|          RANDOM FORESTS         |    BOW    |    0.63   |
| GRADIENT BOOSTED DECISION TREES |    BOW    |    0.81   |
+---------------------------------+-----------+-----------+
```

As we can observe from the above table, our initial base model didnt perform as expected and gave mediocre AUC scores for all kinds of encodings.
Further ahead, more complex models like SVM and Ensemble models like Gradient Boosted Decision Trees perform quite well giving an AUC Score of 0.81.

Hence we conclude that the model that can be used further ahead to deploy the model is a gradient boosted decision tree model.

8. FUTURE ENHANCEMENTS

This project has a great future scope. Various new features as well as new technologies can be used to further ease the work of the volunteers.

Right now, we are just analyzing the text data and achieving the AUC Score of nearly 0.81. Further, we can even train our model to use images that are submitted by the teacher in order to verify whether the pictures that are submitted by the teachers are indeed relevant to the project or not. Also, currently, the teachers directly give the resources list required by the students. Moving ahead, we can also create a transfer learning model that uses the existing image recognition models to verify whether the teacher has indeed procured those resources or not.

Also, moving further down the line, we can enable the teachers to submit a video presentation describing their project. These videos can be processed, separately along with the essays. State of the art deep learning techniques can be used for this video processing to determine whether the teacher is indeed tellin the truth or not. Facial expressions, environment and various other factors can be used for this purpose.

A.I. research is a continuous process and new algorithms, techniques and methodologies are being invented almost daily. Some of these techniques are state of the art and our model can be trained using these techniques to achieve an even higher accuracy.

9. REFERENCES

[1] Oded Netzer, Alain Lemaire, Michal Herzenstein, **"When Words Sweat: Identifying Signals for Loan Default in the Text of Loan Applications"** , Journal of Marketing Research (JMR), August 2019.

[2] Fix,Evelyn; Hodges, Joseph L. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties (Report). USAF School of Aviation Medicine, Randolph Field, Texas.

[3] Altman, Naomi S. (1992). "**An introduction to kernel and nearest-neighbor nonparametric regression**" (PDF). *The American Statistician*. **46** (3): 175–185.

[4] Cox, David R. (1958). "**The regression analysis of binary sequences (with discussion)**". *J R Stat Soc B*. **20** (2): 215–242. JSTOR 2983890

[5] Cramer, J. S. (2002). ***The origins of logistic regression*** (PDF)(Technical report). **119**. Tinbergen Institute. pp. 167–178. doi:10.2139/ssrn.360300.

[6] Cortes, Corinna; Vapnik, Vladimir N. (1995). "**Support-vector networks**" (PDF). *Machine Learning*. **20** (3): 273–297. CiteSeerX 10.1.1.15.9362. doi:10.1007/BF00994018. S2CID 206787478.

[7] Bennett, Kristin P.; Campbell, Colin (2000). **"Support Vector Machines: Hype or Hallelujah?"** (PDF). *SIGKDD Explorations*. **2** (2): 1–13. doi:10.1145/380995.380999. S2CID 207753020

[8] James, Gareth; Witten, Daniela; Hastie, Trevor; Tibshirani, Robert (2017). "**Tree-Based Methods**" ***An Introduction to Statistical Learning: with Applications in R***. New York: Springer. pp. 303–336. ISBN 978-1-4614-7137-0.

[9] Wu, Xindong; Kumar, Vipin; Ross Quinlan, J.; Ghosh, Joydeep; Yang, Qiang; Motoda, Hiroshi; McLachlan, Geoffrey J.; Ng, Angus; Liu, Bing; Yu, Philip S.; Zhou, Zhi-Hua (2008-01-01). "**Top 10 algorithms in data mining**". *Knowledge and Information Systems*.

[10] Ho, Tin Kam (1995). ***Random Decision Forests*** Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282

[11] Boehmke, Bradley; Greenwell, Brandon (2019). "**Gradient Boosting**". *Hands-On Machine Learning with R*. Chapman & Hall. pp. 221–245.

[12] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Lukasz; Polosukhin, Illia (2017-12-05). "**Attention Is All You Need**". arXiv:1706.03762