

FACULTY EVALUATION SYSTEM

A PROJECT REPORT

Submitted by

Mr. PRANJAL SONAWANE (FS18IF032)

in partial fulfillment for the award

Diploma

IN

INFORMATION TECHNOLOGY



DEPARTMENT OF INFORMATION TECHNOLOGY

GOVERNMENT POLYTECHNIC MUMBAI.

JUNE 2021

GOVERNMENT POLYTECHNIC MUMBAI

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled **“FACULTY EVALUATION SYSTEM**
” is the bonafide work of **“ Mr. Pranjal Sonawane (FS18IF032) ”** ,
submitted in partial fulfillment of the requirements for the award of Diploma in
Information Technology of Government Polytechnic Mumbai.

Dr. R. A. Patil
HEAD OF THE DEPARTMENT

Prof. Namrata A. Wankhade
PROJECT GUIDE
Lecturer in Information Technology

EXTERNAL EXAMINER

Mrs. Swati. D. Deshpande
PRINCIPAL

DECLARATION

We hereby declare that the project entitled “**FACULTY EVALUATION SYSTEM** ” being submitted by us towards the partial fulfillment of the requirements for the award of Diploma in Information Technology is a project work carried by us under the supervision of **Prof NAMRATA A. WANKHADE** and have not been submitted anywhere else. We will be solely responsible if any kind of plagiarism is found.

Date :- 25/06/2021

Pranjal Sonawane
(FS18IF032)

ACKNOWLEDGEMENT

We like to share our sincere gratitude to all those who help us in completion of this project. During the work we faced many challenges due to our lack of knowledge and experience but these people help us to get over from all the difficulties and in final compilation of our idea to a shaped sculpture.

We would like to thank **Prof Namrata A. Wankhade** Mam for her governance and guidance, because of which our whole team was able to learn the minute aspects of a project work.

We would also like to show our gratitude to our Project Coordinators **Dr. R. A Patil** Sir and **Ms. Dipali Gosavi** Mam for their continuous help and monitoring during the project work.

All of our team is thankful to all the Faculties and Staff of Department of Information Technology , Government Polytechnic Mumbai , for their help and support towards this project and our team.

We are also thankful to our friends and most of all to our parents who have inspired us to face all the challenges and win all the hurdles in life

Thank you All.

TEAM MEMBERS

Pranjal Sonawane (FS18IF032)

Abstract

Natural Language Processing is a vital field of research having applications in different subjects. NLP is important for scientific, economic, social, and cultural reasons. NLP is experiencing rapid growth as its theories and methods are deployed in a variety of new language technologies. Generally Faculty Feedback are evaluated manually we can NLP to evaluate faculty feedback .Feedback review can be classify in to positive , negative and neutral sentiments. Text Classification is a part of NLP where the text is converted into a machine-readable formy by performing various methods. Tokenizing, part-of-speech tagging, stemming, chunking are some of the text classification methods. Implementing these methods on our data gives us a classified data on which we will train the model to detect sentiment of messages using Scikit-Learn Classifiers. We proposed a model to solve the issue of classifying messages as positive , negative and neutral by experimenting and analyzing the relative strengths of several machine learning algorithms such as K-Nearest Neighbors (KNN), Decision Tree Classifier, Random Forest Classifier, Logistic Regression, SGD Classifier, Multinomial Naive Bayes(NB), Support Vector Machine(SVM) to have a logical comparison of the performance measures of the methods we utilized in this research. The algorithm we proposed achieved an average accuracy of 98.49% with SVM model on ‘RateMyprofessor’ dataset. Using that model we can predict any sentiment and Evaluate any Faculty .

TABLE OF CONTENTS

Sr No	Title	PAGE
1	CERTIFICATE	I
2	DECLARATION	II
3	ACKNOWLEDGEMENT	III
4	ABSTRACT	IV
5	CHAPTER 1 :- INTRODUCTION	1
	Introduction	2
6	CHAPTER 2 :- LITERATURE REVIEW	4
	Literature Review	5
7	CHAPTER 3 :- NLP	7
	3.1 Creating Dataset	8
	3.2 Preprocessing	9
	3.3 Label Encoding	10
	3.4 Tokenizing Text	10
	3.5 Stopword Removal	11
	3.6 Stemming	11
	3.7 Feature Generation	12
	3.8 Vectorizing	12
8	CHAPTER 4 :- ML Algorithms	14
	4.1 Selecting ML model	15
	4.2 Pickling ML model	18
9	CHAPTER 5 :- WEB Application	19

	5.1 Creating Flask app	20
	5.2 Predicting Sentiments	21
	5.3 Exploratory Data Analysis	23
	5.4 Data Visualization	25
	5.5 Creating Document	30
	5.6 Deploying on cloud	31
10	RESULT	33
11	CONCLUSION	38
12	FUTURE SCOPE	39
13	REFERENCES	40

LIST OF ABBREVIATIONS

NLP: Natural Language Processing

FES: Faculty Evaluation System

SVM: Support Vector Machine

NB: Naive Bayes

RF: Random Forest

KNN: Nearest Neighbors

TF-IDF: Term Frequency-Inverse Document Frequency

EDA: Exploratory data analysis

CHAPTER 1

INTRODUCTION

INTRODUCTION

An automatic system to analyze the textual feedbacks of faculty members teaching in any institute is proposed. The proposed system extracts all the important aspects from the feedback and then the sentiment score of each aspect for each faculty is calculated using machine learning algorithms. Our system, Faculty Evaluation System (FES) identifies strengths and weaknesses of teachers on all those aspects which are important to students. This information may also be used by higher authorities of the institute to form appropriate teams of faculty members for different academic and administrative activities of the institute.

To measure the effectiveness of a teacher to the satisfaction of every student, we are proposing an aspect based sentiment evaluation system. In the proposed system, we collect student feedback in the form of running text. Using linguistic features and machine learning techniques, in the first phase, important aspects of teachers are identified from the collected feedbacks and then for those aspects, the students' views are processed to find out the sentiments of students as either "positive", "neutral" or "negative" about every teacher for all the important aspects extracted from the data itself. This evaluation model overcomes the problems of traditional feedback system and it also allows the administration to make use of the information about each teacher on a variety of aspects for assigning important responsibilities to them. During the aspect identification process, the system chooses the vital aspects by grading each aspect on the basis of feedback of students. Aspect detection and evaluation of each aspect for every faculty allows an institute to use this information for effective utilization of its people leading to better growth.

It is achieved using analyzing sentiments. Sentiment analysis can be used to study the opinion, views, feelings and attitudes from the documents, expressions, comments, tweets using NLP. It is also known as opinion mining. SA divides sentiments into types like Negative. Polarity is given "Positive", "negative" or "Neutral" to each sentiment which is a floating -point number. Its value lies between -1 to +1. Based on polarity, we can define the sentiment of the statement.

Sentiment analysis is done at three different levels: Aspect Based, Document Based and Sentence based. Various approaches are used to do a sentiment analysis like Lexicon based, Machine learning and Hybrid approach . In this project supervised machine learning algorithms are used for sentiment analysis like Support Vector Machine (SVM), Naïve Bayes and Random Forest. Performance of these algorithms is measured by finding accuracy, recall and precision.

CHAPTER 2

LITERATURE REVIEW

LITERATURE REVIEW

Sentimental classifiers can be developed using two machine learning approaches named Supervised and Unsupervised learning [50]. The most famous approach to build a classifier is supervised learning. In a supervised approach, the classifier requires labeled training data. Yet the training data is to be annotated using manual or automatic approaches on the basis of some predefined rules. Using predefined rules the citation sentences are annotated as positive, negative and neutral. For the purpose of annotations, human annotators are required. While in the case of an unsupervised approach, there is no need for labeled training data. Instead, there is a need for sentiment lexicon to assign polarities to citation sentences. This approach is very difficult because it requires different varieties of a lexicon for different genres. From the literature review .One such work is presented researcher in which expansion towards the lexicon is considered by using the concept of text position. They used the position of text in order to expand the lexicon to get better-classified results. They used the concept of assigning weights to the parts of the text. They tend to assign more weight to the more subject-oriented part while less weight to the less subject-oriented part. This method achieved an accuracy score of 65%, later they found that this technique is not as efficient as they expected.

Authors used machine algorithms for sentiment analysis of product reviews taken from Amazon about different products likes Laptops, Tablets, Camera, TVs and mobile phones etc. They applied two machine learning algorithms: SVM and Naïve Bayes. They have done dictionary based SA and got an accuracy of 98.17% by Naïve Bayes and got an accuracy of 93.54% by SVM for Camera reviews .we found that many researchers used supervised while others rely on unsupervised approach.

So we decide to use supervised learning model .One of serious problem was dataset there is no dataset present for faculty review so we decided to scrap the data from ratemyprofessor.com to create dataset . To scrape dataset we wrote python script which retrieve rating and review of teacher from website . Due to script we can scale dataset to huge amount .

We tried various Machine learning algorithm like Nearest Neighbors (KNN), Decision Tree Classifier, Random Forest Classifier, Logistic Regression, SGD Classifier, Multinomial Naive Bayes (NB), Support Vector Machine (SVM) to train and test the data in order to find to best model with highest accuracy .

To make it easily usable we need better gui so , we have also focused on making the gui as we know that the web applications are good and faster than the traditional gui they are more responsive than the traditional gui so we have taken that into consideration and created the web based application in python using cross platforming this functionality provided by the flask framework and it provide better user experience and make it easy to access . We deploy our ML model in cloud using Heroku services so it is easily available and everyone one can able to use it .

CHAPTER 3
NLP
(Natural Language Processing)

3.1. Creating Dataset

To Train ML mode we need Dataset and in this time we scrape data from Ratemyprofessor.com and create a Dataset

```
import requests

with open('rmp.csv', 'w') as f:
    f.write('tid,comment,quality\n')
    tids = [45881, 601915, 71357, 277558, 103621, 1307460, 1141164,
158088,1195829,2547960,765147,2618238,235,219884,260217,376177,1257758,360897,6234
46,1823468,2146337,1692336,1692333,184352,144352,1443522,]
    for tid in tids:
        remaining, page = 1, 1
        while remaining > 0:
            url =
'http://www.ratemyprofessors.com/paginate/professors/ratings?tid={0}&page={1}'.format(tid,
page)
            lm_json = requests.get(url).json()
            comments = []
            qualities = []
            for rating in lm_json['ratings']:
                comments.append(rating['rComments'])
                qualities.append(rating['quality'])
            for comment, quality in zip(comments, qualities):
                f.write('{0},{1},{2}\n'.format(tid, comment.replace('"', '\\"'), quality))
            remaining = lm_json['remaining']
            page += 1
```

The script create one csv file which include facultyid , review or comment on teacher and rating of teacher .

tid	comment	quality
45881	I struggle with getting my point across and was planning on trying my best to improve. Not only did I receive 3 straight D's - he also tore apart my papers over the smallest mistakes and gave no constructive feedback. I dropped the class and learned later that he had to give a flat 5% extra credit to all students' papers by the dean.	awful
45881	This guy is the worst professor I have ever had. Over 5 10+ page papers for entry level class. Does not care about his students. Does not post anything on blackboard, which makes you go to the lectures where you will learn nothing about english.	awful
45881	Avoid his class like the plague. He grades as if you were supposed to be an absolutely perfect writer. He even said himself that he's a perfect writer, and that he makes absolutely zero mistakes.	awful
45881	Time in class is useless. His teachings are unoriginal as he relies heavily on the textbooks for information. You will read chapters and then he will restate the information you already covered. He believes his writing is perfect and that yours must match it. If you have him, use his feedback and apply it to your next papers.	awful
45881	This class was a waste of time. The only way you would get above a 70% on a paper is if you were buddy-buddy with Rand. He favors kids that cause the most problems but put on a good poker face. The papers are useless and you will learn nothing. He doesn't teach, you have to teach yourself out of the TSIS book. Many complaints were sent to the dean.	awful
45881	He is an exceptionally harsh teacher, and hes an awful person as well.	awful
45881	Avoid Steven Rand's class like the plague. He is unnecessarily difficult for a 100 level class. You'd learn more about english if you were in a coma on Mars.	awful
45881	He grades as if we were English majors and not just taking this for our English credit. You also have to be prepared to spend more on this class than any other and still not get a good grade.	awful

rmp.csv X		
601915	His tests are worth majority of your grade, yet he puts some questions that he's never gone over in class. Homework is worth lots of points but only for a fraction of your overall grade. Just overall frustration with this course because it is unfair. Plus he does not take timing into consideration for tests so its easy to run out out of time!	awful
601915	I took his Algebra class last semester and I am now taking his Applied Calculus. Algebra was only one hour a week and I didn't get much from him because it was mostly a self taught class. Calculus I now have 3 hours a week and his lectures just confuse me. I find myself finding outside resources for help because the class itself is hard regardless.	good
601915	Stuart is not good teacher. He doesn't ever actually answer your questions, he just answers everything BUT the question.	awful
601915	It is super important to go to his lectures because he talks through things well and there is tons of in class work. I struggled a lot in this class and should have failed but he told me because I went into his office hours so much to get help that he could see I was trying and bumped me to a C. He's so chill and the nicest guy ever	good
601915	He is a nice guy, don't get me wrong. He is a very caring person and he wants you to pass his class. We only had class once per week on Tuesday and the rest of the week was spent using an online tool called Aleks. That was my least favorite part of the class, it felt like I was taking an online course rather than a regular math class.	average
601915	Despite the reviews I read about Mr. Farm I believe he truly loves math and although not everyone in the room loves his humor. He genuinely wants you to succeed in his class. He always says his availability and will stick around after class to be available to his students.	awesome
601915	Stuart gets a bad rap, but he is so beyond willing to help if you ask for it. There's a fair amount of homework, and you're graded on in-class work, but there's no better way to learn math. There are 90 possible points on the tests, but they're only worth 80, so an A is feasible. Needing to show up to earn points doesn't make this a hard class.	awesome
601915	I love Stu because he is a really chill guy. He wants you to succeed and is fun in class. The only downside for me is that he tends to over explain answers which would confuse me. He is super lenient on giving out extensions and always answers his email. Pretty good at making a math class enjoyable and entertaining.	good

Various experiments were applied on the dataset which were based on Natural Language Processing(NLP) concepts like label encoding, tokenization, stemming, stop word removal , generating features and then applied ensemble method – voting classifier. All these experiments performed in the model classify the data set accurately.

3.2 Pre-Processing

The messages have to be pre-processed for the removal of unwanted punctuation, grammar, stop words etc.

```

nltk.download('stopwords')
nltk.download('punkt')
faculty_reviews['reviews_text_new'] = faculty_reviews['review'].str.lower()

```

```

spl_chars = faculty_reviews['reviews_text_new'].apply(lambda review:
                                                    [char for char in list(review) if not char.isalnum() and char != '
'])

## Getting list of list into a single list
flat_list = [item for sublist in spl_chars for item in sublist]

review_backup = faculty_reviews['reviews_text_new'].copy()
faculty_reviews['reviews_text_new'] =
faculty_reviews['reviews_text_new'].str.replace(r'^A-Za-z0-9 ]+', ' ')
stop_words = stopwords.words('english')
def stopwords_removal(stop_words, sentence):
    return [word for word in nltk.word_tokenize(sentence) if word not in stop_words]

faculty_reviews['reviews_text_nonstop'] = faculty_reviews['reviews_text_new'].apply(lambda
row: stopwords_removal(stop_words, row))
faculty_reviews[['reviews_text_new', 'reviews_text_nonstop']]

```

3.3 Label Encoding

Label Encoder encode labels with values “awesome”, “good”, “average”, “poor”, “awful” where represents as “positive”, “negative”, “netural” labels for the classes. In our experiment, we convert the class labels to “positive”, “negative”, “netural” .

```

faculty_reviews.loc[faculty_reviews.rating == "awesome", "Sentiment_rating"] = "positive"
faculty_reviews.loc[faculty_reviews.rating == "poor", "Sentiment_rating"] = "netural"
faculty_reviews.loc[faculty_reviews.rating == "awful", "Sentiment_rating"] = "negative"
faculty_reviews.loc[faculty_reviews.rating == "average", "Sentiment_rating"] = "netural"

```

3.4 Tokenizing Text

Tokenization is the first step in NLP. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

A word (Token) is the minimal unit that a machine can understand and process. So any text string cannot be further processed without going through tokenization. Tokenization is the process of splitting the raw string into meaningful tokens. The complexity of tokenization varies according to the need of the NLP application, and the complexity of the language itself. For example, in English it can be as simple as choosing only words and numbers through a regular expression.

Sentence tokenizer breaks text paragraph into sentence

Word tokenizer breaks text paragraph into words.

3.5 Stop Word Removal

When using Natural Language Processing(NLP), our goal is to perform some analysis or processing so that a computer can respond to text appropriately. A machine cannot understand the human readable form. So, data has to be pre-processed in order to make it machine-readable. This is “pre-processing” of which one of the major forms is to filter out useless data. This useless data (words) is generally referred as “stop words” in Natural Language Processing(NLP).

3.6 Stemming

Stemming is another pre-processing step that normalize sentences. Stemming is a way to account for the variations of words and sentences which often have a same meaning; furthermore, it will help us shorten the sentences and shorten our lookup. For example, consider the following sentence:

1. I was taking a ride on my horse.
2. I was riding my horse.

These sentences mean the same thing, as noted by the same tense in each sentence; however, that isn't intuitively understood by the computer. To account for all the variations of words in the English language, we can use the Porter stemmer, which has been around since 1979.

All these are Process of Natural Language Processing

3.7 Feature Generation

Feature engineering is the process of constructing features for machine learning algorithms by using the knowledge of that specific domain. The words in each text message are the features on which the algorithm will predict the output. For this purpose, it will be necessary to tokenize each word. The most common words that are generated in feature generation will be used as our features. Then the data is split in training and testing datasets with a test size of 20%.

3.8 Vectorizing

The next step is to further transform the cleaned text into a form that the machine learning model can understand. This process is known as Vectorizing. One of Most efficient vectorizers is the TF-IDF vectorizer.

TF-IDF (term frequency-inverse document frequency) was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don't mean much to that document in particular.

TF-IDF for a word in a document is calculated by multiplying two different metrics:

- The term frequency of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.
- The inverse document frequency of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking

the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.

- So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

```
vectorizer = TfidfVectorizer (min_df=7, max_df=0.8,  
stop_words=stopwords.words('english'))  
  
processed_features =  
vectorizer.fit_transform(faculty_reviews['reviews_text_new'].astype(str)).toarray()
```

CHAPTER 4

MACHINE LEARNING ALGORITHM

4.1 Selecting ML model

After preprocessing and features selection the very next step is to apply classification algorithms. We need to import each algorithm from the scikit-learn library along with performance metrics. We require accuracy score and classification report metrics to predict the accuracy and give a classified report on the output. . We have used different algorithms of machine learning including Naïve-Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbor (KNN), and Random Forest (RF).

a) Naïve Bayes: Naïve- Bayes is the most popular classification algorithm due to its simplicity and effectiveness. This classifier works according to the concept of Bayes theorem. It's a kind of module classifier that follows the idea of probabilities for the purpose of classification. Bernoulli and multinomial are the models of naïve Bayes classifier .

b) Support Vector Machine: In the world of machine learning one such supervised learning algorithm that achieves enough improvements on a variety of tasks is a Support vector machine classifier . Particularly in the case of analyzing the sentiments, SVM has demonstrated good results . In-text classification the SVM contributes towards excellent precision scores while poor recall scores while adjusting the thresholds recall scores can be adjusted .

c) Decision Tree: In various fields of text classification the use of decision tree classifier can be seen and analyzed [. Its popularity is based on the nature of classification rules that make it interesting for NLP researchers . The decision is constructed by selecting the data from the data set randomly . The information gain is calculated for all values and the feature with the highest information gain value becomes the tree's root and the whole tree is constructed by finding the features for the next level again and again.

d) Random Forest: The importance of a random forest classifier and compared its performance with the other classifiers. Researchers claimed that the random forest algorithm provides efficient and discriminative classification, as a result, it is considered an interesting classifier. Some developers discuss the importance of random forest classifiers in the field of computer vision. Introduced class recognition based on random forest.

e) K-th Nearest Neighbour: KNN is a simple and efficient classifier. It is also called lazy learner because its training phase contains nothing but storing all the training examples as classifiers. KNN requires a lot of memory while storing the training values. The performance issue of KNN can also be solved by efficient estimations of parameters.

We tried all these different algorithms and found Random Forest provides high accuracy among all.

```
X_train, X_test, y_train, y_test = train_test_split(
    processed_features, # Features
    faculty_reviews['Sentiment_rating'],
    test_size=0.2,
    random_state=0
)
```

```
from sklearn.ensemble import RandomForestClassifier

text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
text_classifier.fit(X_train, y_train)
```

```
predictions = text_classifier.predict(X_test)
```



```
[ ] review_array = np.array(["faculty is very very rude , always comming late"])
print(type(review_array))
review_vector = vectorizer.transform(review_array).toarray()
print(text_classifier.predict(review_vector))
```

```
<class 'numpy.ndarray'>
['negative']
```

```
▶ review_array = np.array(["faculty is very don't help to anyone"])
print(type(review_array))
review_vector = vectorizer.transform(review_array).toarray()
print(text_classifier.predict(review_vector))
```

```
<class 'numpy.ndarray'>
['negative']
```

```
[ ] review_array = np.array(["His class was not too bad, but he tries very hard t
print(type(review_array))
review_vector = vectorizer.transform(review_array).toarray()
print(text_classifier.predict(review_vector))
```

```
<class 'numpy.ndarray'>
['netural']
```

4.2 Pickling ML Model

Pickle is a python module used to serialize a python object into a binary format and deserialize it back to the python object. In machine learning, while working with the scikit learn library, we need to save the trained models in a file and restore them in order to reuse it to compare the model with other models, to test the model on new data. The saving of data is called Serialization, while restoring the data is called Deserialization.

Also, we deal with different types and sizes of data. Some datasets are easily trained i.e- they take less time to train but the datasets whose size is large (more than 1GB) can take very large time to train on a local machine even with GPU. When we need the same trained data in some different project or later sometime, to avoid the wastage of the training time, store trained model so that it can be used anytime in the future.

Hence we pickel our ML model and vectorizer for further use .

```
import pickle
pickle.dump(vectorizer, open('tranform.pkl', 'wb'))
pickle.dump(text_classifier, open('nlp_model.pkl', 'wb'))
```

CHAPTER 5

WEB APPLICATION

5.1 Creating Flask Application

To use ML model and make applications we made web applications using a cross platform flask framework .

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.



The screenshot displays a web application interface for a 'Faculty Review System'. The browser's address bar shows 'nlp-deploy-v2.herokuapp.com'. The main heading is 'Faculty Review System'. Below it, there is a section for 'Upload CSV File' with a 'Choose file' button, a status indicator 'No file chosen', and an 'Upload' button. Further down, the 'Predict sentiment' section features a text input field with the placeholder 'Enter Your Message Here' and a 'predict' button. The background of the application has a subtle geometric pattern.

5.2 Predicting Sentiments

Generally Faculty Feedback is circulated using forms, website or google form . In all such cases we get Excel or csv files in which all reviews are present . We just need to upload the csv file and the ML model will predict the sentiment and classify into respective categories .

```
import pickle
clf=pickle.load(open("nlp_model.pkl",'rb'))
cv=pickle.load(open('transform.pkl','rb'))
def process_file(path, filename):
    file=filename
    # remove_watermark(path, filename)
    test = pd.read_csv(path)
    testing_review=test["rating"].tolist()
    test_data_features = cv.transform(testing_review)
    predictions_test = clf.predict(test_data_features)
    output = pd.DataFrame(data={"ID":test["id"],"rating":test["rating"],
    "prediction":predictions_test} )
    output.to_csv( app.config['DOWNLOAD_FOLDER']+filename,index=False)
```

Earlier we pickel our model so we don't need to train mode again and again . After predicting sentiments it saves output in the Download Folder with the same filename . We can further process this data to extract meaningful and useful information out of it.

A	B	C
ID	rating	prediction
1	nothing positive	negative
1	Ma'am is very good	positive
1	teacher is very supportive towards students	netural
1	The teacher has thorough knowledge of the subject	positive
1	taught us nicely	negative
1	lecture not on time	negative
1	taking efforts and taught them while sitting	positive
1	Teaching method is worst	negative
1	Steven was a nice guy, but he graded much	netural
1	Steven expects a lot from the student, who	netural
1	I am currently in this class. I took this,	negative
2	His class was not too bad, but he tries very	netural
2	Seems like a nice guy, but he expects you	netural
2	Annoying, not helpful. Incredibly unclear.	netural
2	Like others have said, he seems like a gr	netural

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 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.

Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. 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.

Some of the most common data science tools used to create an EDA include:

Python: An interpreted, object-oriented programming language with dynamic semantics. Its high-level, built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. Python and EDA can be used together to identify missing values in a data set, which is important so you can decide how to handle missing values for machine learning.

Pandas

Pandas is a data analysis package, and it is used for exactly that. It also finds use in the initial phases of Machine Learning, where there is a lot of exploratory data analysis, data cleaning and data wrangling. With a wide compatibility net, you can import data from popular application formats like XLS, XLSX(Excel), JSON, CSV (comma separated values), flat files and more. You can even read data from a database table into pandas data structures. There are many operations possible at a row-level, column level, or on the entire Data Frame or Series, like merging, re-shaping, scalar operations, and operations between Data Frames and Series

NumPy

NumPy stands for Numerical Python and is one of the most useful scientific libraries in Python programming. It provides support for large multidimensional array objects and various tools to work with them. Various other libraries like Pandas, Matplotlib, and Scikit-learn are built on top of this amazing library.

5.4 Data Visualization

"A picture is worth a thousand words". We are all familiar with this expression. It especially applies when trying to explain the insight obtained from the analysis of increasingly large data sets. Data visualization plays an essential role in the representation of both small and large scale data. One of the key skills of a data scientist is the ability to tell a compelling story, visualizing data and findings in an approachable and stimulating way. Learning how to leverage a software tool to visualize data will also enable you to extract information, better understand the data, and make more effective decisions. Various techniques have been developed for presenting data visually but in this course, we will be using several data visualization libraries in Python, namely Matplotlib, seaborn, and Folium.

Matplotlib

Matplotlib is a 2-D plotting library that helps in visualizing figures. Matplotlib emulates MATLAB like graphs and visualizations. MATLAB is not free, is difficult to scale and as a programming language is tedious. So, matplotlib in Python is used as it is a robust, free and easy library for data visualization.

Seaborn

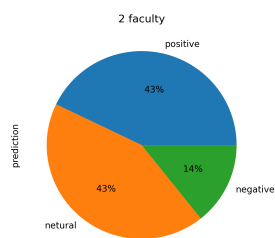
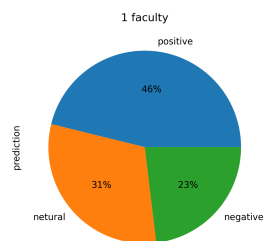
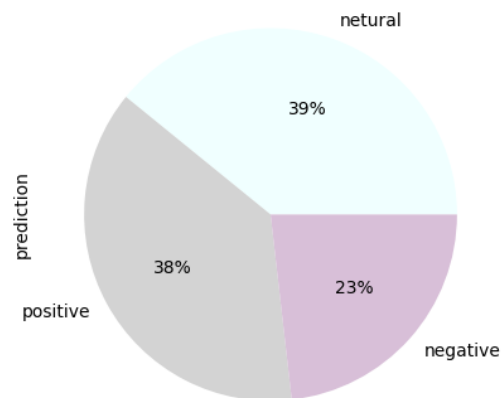
Seaborn is a Python data visualization library based on matplotlib. It offers a simpler interface, sensible defaults for plots needed for machine learning, and most importantly, the plots are aesthetically better looking than those in Matplotlib.

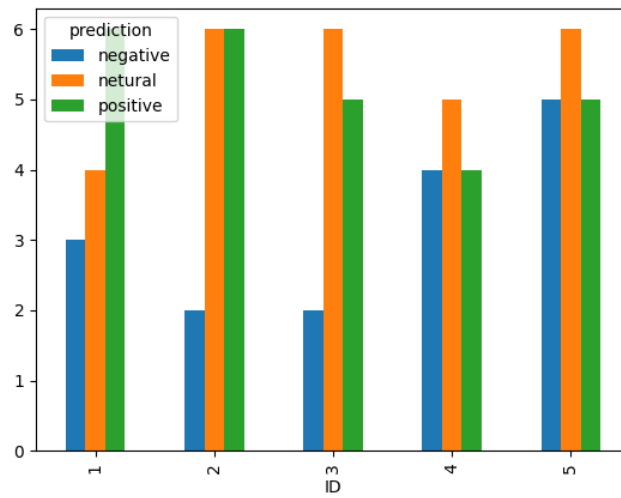
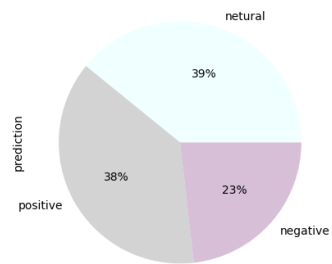
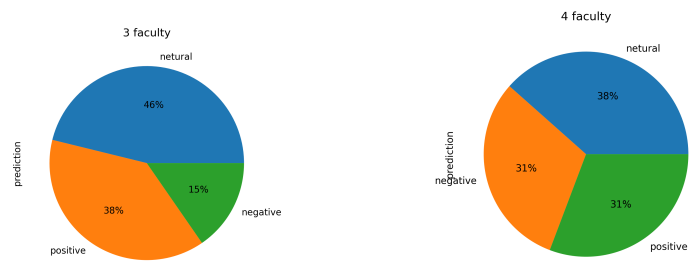
```
df = pd.read_csv( app.config['DOWNLOAD_FOLDER']+file)
df.prediction.value_counts().plot(kind='pie', autopct='%1.0f%%', colors=["azure",
"lightgrey", "thistle", "khaki", "pink"])
plt.savefig("totalpercent")
# Barplot of all teacher
faculty_sentiment = df.groupby(['ID', 'prediction']).prediction.count().unstack()
faculty_sentiment.plot(kind='bar')
plt.savefig("com_faculty")

convert_dict = {'ID':str }
```

```
df = df.astype(convert_dict)
faculty_ID=list(df['ID'].unique())

for i in faculty_ID:
    plt.figure()
    var = df.loc[(df['ID']==f'{i}')]
    var.prediction.value_counts().plot(kind='pie',autopct='%1.0f%%')
    plt.title(f'{i} faculty')
    plt.savefig(f'{i}',dpi=300)
    #plt.show()
```





Word Cloud

Word cloud is a simple yet powerful visualization technique. It is primarily used to highlight significant textual data points. Word cloud is also known as text cloud or tag cloud. Altogether, the more frequently a specific word appears in the textual data, the greater prominence is given to that word in the cloud. Hence, the bigness and boldness of a word depend on the frequency of that particular word in the document.

```
word_cloud_text = ".join(df['rating'])

wordcloud = WordCloud(max_font_size=100, # Maximum font size for the
largest word
                      max_words=100, # The maximum number of words
                      background_color="white", # Background color for the word cloud
image
                      scale = 10, # Scaling between computation and drawing
                      width=800, # Width of the canvas
                      height=400 # Height of the canvas
                      ).generate(word_cloud_text)

plt.figure()
plt.imshow(wordcloud,
           interpolation="bilinear") # to make the displayed image appear more
smoothly
plt.axis("off")
# plt.show()
wordcloud.to_file('features_sentiments.png')
```



5.5 Creating Document using Docx

We create word document to represent Exploratory Data Analysis and Data visualization in proper readable format . Word documents contain formatted text wrapped within three object levels. Lowest level- Run objects, Middle level- Paragraph objects and Highest level- Document object.

So, we cannot work with these documents using normal text editors. But, we can manipulate these word documents in python using the python-docx module.

```
from docx import Document
from docx.shared import Inches
from docx.shared import Pt
document = Document()

document.add_heading('Faculty Review Analysis ', 0)

document.add_heading('Distribution of predicted sentiments ', level=1)
document.add_picture('totalpercent.png', width=Inches(4))

lis = []
for i in faculty_ID:
    var = df.loc[(df['ID']== f'{i}')]
    print("name of faulty "+ i)
    a=(var['prediction'].value_counts())
    a=list(a)
    document.add_heading(f'Faculty ID {i}', level=1)
    document.add_paragraph(f'Number of positive feedback : {a[0]}', style='List Bullet')
    document.add_paragraph(f'Number of negative feedback : {a[1]}', style='List Bullet')
    # document.add_paragraph(f'Number of netural feedback : {a[2]}', style='List Bullet')
    document.add_paragraph("")
    document.add_picture(f'{i}.png', width=Inches(4))
    document.add_paragraph("")

document.add_heading('Students Sentiments in wordcloud ', level=1)
# document.add_picture('features_sentiments.png', width=Inches(5))
document.add_paragraph("")
document.add_heading('Comparasion of among all Faculty ', level=1)
document.add_picture('com_faculty.png', width=Inches(5))
document.save(app.config['DOWNLOAD_FOLDER']+file+'.docx')
```

5.6 Deploying on Cloud (Heroku)

Above we have created a Flask application `app.py` file is present , when we run that `python app.py` This launches a very simple built-in server `http://127.0.0.1:5000/` . We have to deploy this flask webapp on Heroku Cloud.

To deploy ML mode on Heroku we need to use additional packages and files .

- Install gunicorn

Gunicorn is a Python WSGI HTTP Server for UNIX. It allows you to run any Python application concurrently by running multiple Python processes within a single dyno. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resources, and fairly speedy.

```
pip install gunicorn
```

- Declare app dependencies

Create `requirements.txt` file in the root directory of the project by `pip freeze` command. The `requirements.txt` file lists all the app dependencies together. When an app is deployed, Heroku reads this file and installs the appropriate Python dependencies using the `pip install -r` command.

```
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy>=1.9.2
scipy>=0.15.1
scikit-learn>=0.18
matplotlib>=1.4.3
pandas>=0.19
nltk>=3.4.5
python-docx>=0.8.11
```

- Create Procfile

The Procfile is always a simple text file that is named Procfile without a file extension in the root directory of the project, to explicitly declare what command should be executed to start your app.

/python-projects/flask-projects/flask-app/Procfile

```
web: gunicorn app:app
```

A Heroku app's web process type is special: it's the only process type that can receive external HTTP traffic from Heroku's routers. If your app includes a web server, you should declare it as your app's web process.


The first app refers to the filename app.py. The second app refers to the instance of Flask which is inside app.py file.

- Create an account on Heroku & GitHub , Create Repository on github and upload all files on github .
- After login on Heroku click on create new App .Give appropriate name to application .
- Connect with github and select the repository and click on deploy to the main branch .
- Heroku will install all packages and deploy to cloud

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

Choose a branch to deploy

 main

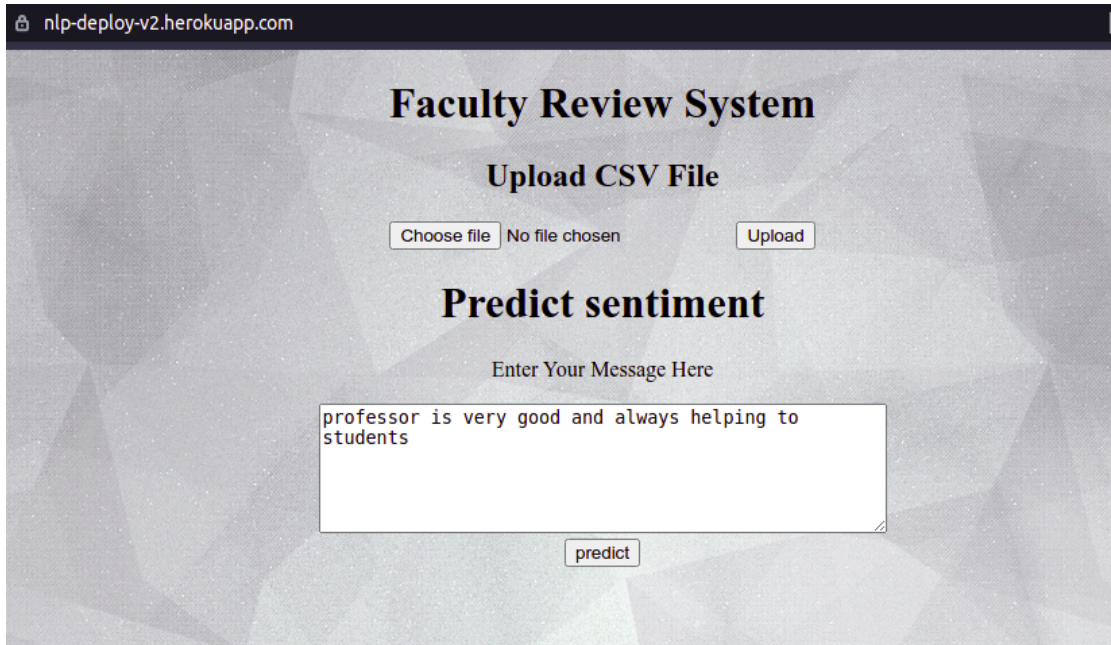
Deploy Branch

Receive code from GitHub	✓
Build main ecba1745	✓
Release phase	✓
Deploy to Heroku	✓

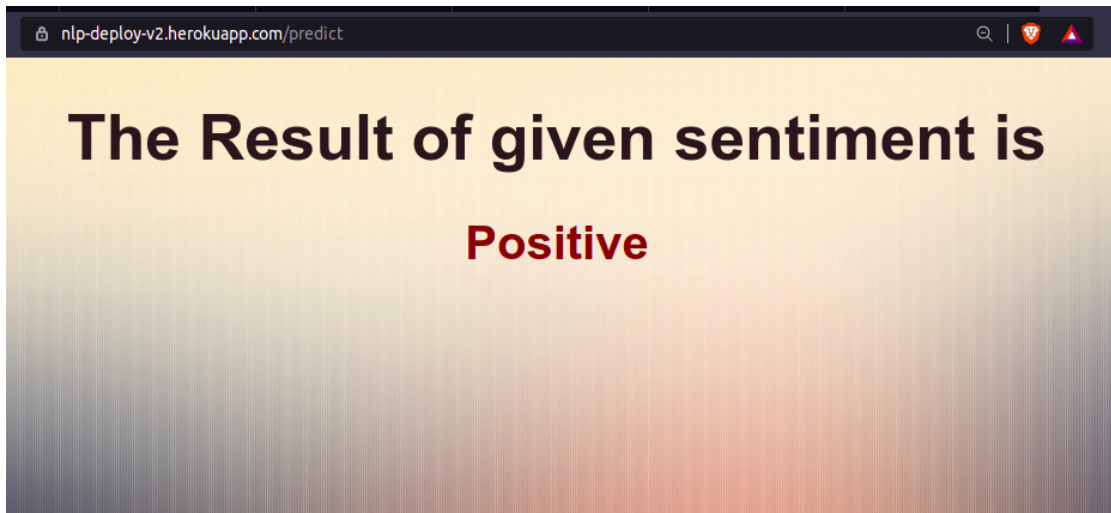
Your app was successfully deployed.

View

RESULTS

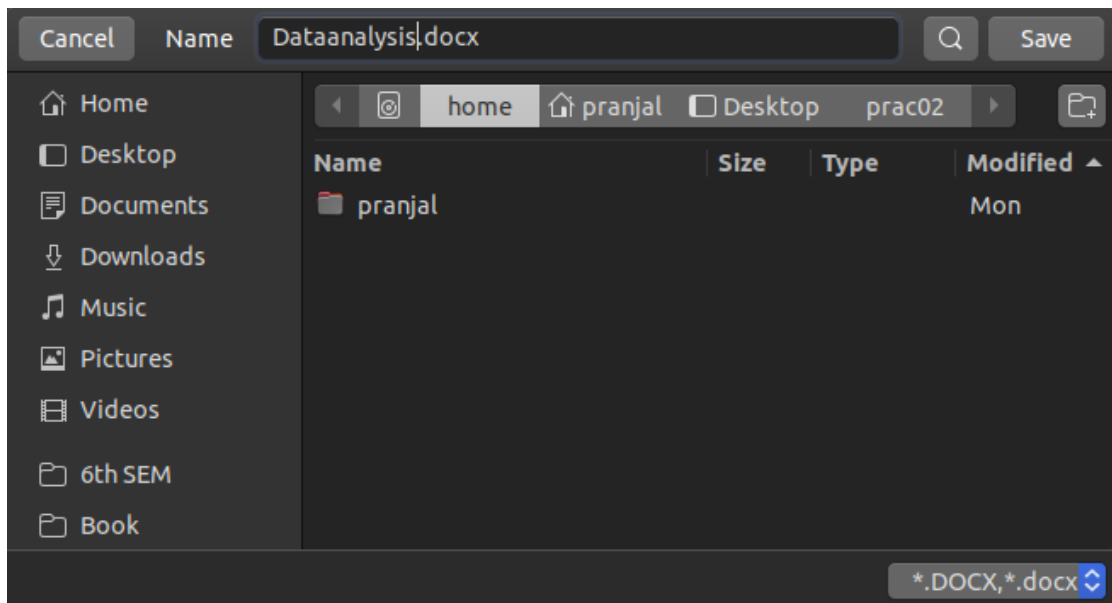


The screenshot shows a web browser window with the address bar displaying "nlp-deploy-v2.herokuapp.com". The page has a dark background with a geometric pattern. The main heading is "Faculty Review System" in a large, bold, black font. Below it is the sub-heading "Upload CSV File" in a smaller, bold, black font. There are two buttons: "Choose file" and "Upload". The "Choose file" button is disabled, and the text "No file chosen" is displayed next to it. Below the upload section is the heading "Predict sentiment" in a large, bold, black font. Underneath is the prompt "Enter Your Message Here" in a smaller, regular, black font. A text input field contains the message "professor is very good and always helping to students". Below the input field is a "predict" button.



The screenshot shows a web browser window with the address bar displaying "nlp-deploy-v2.herokuapp.com/predict". The page has a light yellow background with a gradient. The main text is "The Result of given sentiment is" in a large, bold, black font. Below it is the word "Positive" in a large, bold, red font.

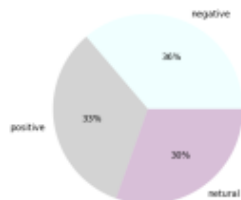
As we know review are present in csv or xlsx file we can upload that file and get visualized output of teacher performance .



It provide a document file where all analysis is present and visualized in a graphical manner .

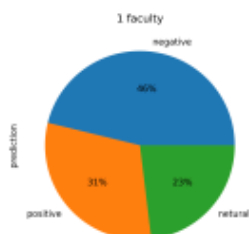
Faculty Review Analysis

Distribution of predicted sentiments



Faculty ID 1

- Number of positive feedback :6
- Number of negative feedback :4
- Number of netural feedback :3



Faculty ID 2

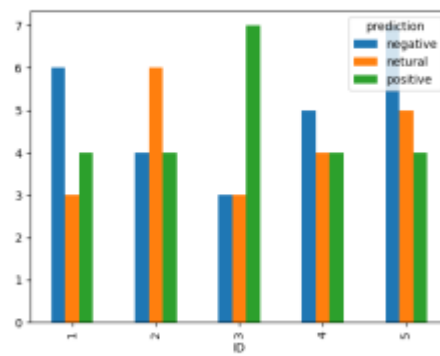
- Number of positive feedback :6
- Number of negative feedback :4
- Number of netural feedback :4



Faculty ID 3

- Number of positive feedback :7
- Number of negative feedback :3
- Number of netural feedback :3

Comparasion of among all Faculty



CONCLUSION

The Proposed Faculty Evaluation System (FES) is able to perform aspect-based sentiment analysis of textual feedback collected from the students. Proposed Faculty Evaluation System is more versatile and useful than the traditional questionnaire-based system that is very restricted; students can give their opinion (in form of marks or grade) and most of checking are done manually. Main computational modules of the faculty evaluation system are Feature Extractor, Sentiment Analyzer and Feature Sentiment Evaluator. To evaluate the performance of our system, we have divided the collected set into training and test sets. We get a pretty good amount of accuracy with random forest algorithm.

Pickel mode of random forest not only classifies faculty review but also classifies any general sentence into “positive” ,”negative ” and “ neutral “ . Also exploratory data analysis on ML mode output gives us understandable representation and visualization of Faculty .

FUTURE SCOPE

Natural Language Processing (NLP) is a field that provides machines with the ability to understand natural human language. Natural language processing is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human languages, in particular how to program computers to process and analyze large amounts of natural language data. Evaluating Faculty review using NLP is one use case and it can be used by many institutions . In the future better models can be created that this one with more accuracy and various features . Project is scalable and processes huge amounts of data .Various ML , AI , NLP techniques can be applied to make better models like use of neural networks , deep learning algorithms which lead to more accuracy .

REFERENCES

- Sentiment Analysis and Subjectivity, in Handbook of Natural Language Processing, Second Edition, N. Indurkha and F.J. Damerau, Editors.
- N. Magesh, P.Thangaraj, S. Sivagobika, S. Praba, R. Mohana Priya, Employee Performance Evaluation using Machine Learning Algorithm, International Journal Computer Communications and Networks
- <https://towardsdatascience.com>
- <https://machinelearningmastery.com>
- Natural Language Processing with Python by Steven Bird, Ewan Klein and Edward Loper.
- B Pang and L. Lee, "Opinion mining and sentiment analysis", *FoundTrends Inform Retrieval*, pp. 1-135, 2008.
- Medhat Walaa, Ahmed Hassan and Hoda Korashy, "Sentiment analysis algorithms and applications: A survey"
- Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems by Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta, Harshit Surana
- Natural Language Processing in Action: Understanding, analyzing, and generating text with Python by Hobson Lane, Hannes Hapke, Cole Howard
- Introduction to Natural Language Processing by Jacob Eisenstein
- Natural Language Processing in Action: Understanding, analyzing, and generating text with Python.
- <https://www.ibm.com/blogs/research/tag/natural-language-processing/>
- <https://www.analyticsvidhya.com/blog/category/nlp/>
- <http://www.marekrei.com/blog/>