A Project report

on

Data Analytics on Stackoverflow data

Submitted By

Kummara Bhargavi 18BEC024

Mohammed Safwan 18BEC028

Parvati Jayakumar 18BEC036

P. Chethan Krishna 18BEC040

Under the guidance of

Dr. Uma Sheshadri

Professor, Head of CSE Dept.

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

Certificate

This is to certify that the work contained in the project report titled **'Data Analytics on Stackoverflow data'** by **Kummara Bhargavi, Mohammed Safwan, Parvati Jayakumar,** and **P. Chethan Krishna** was completed during the VII semester - IV Year as a Minor Project under the guidance of **Dr Uma Sheshadri**, Professor, Head of CSE dept.

Signature of the supervisor

Dr. Uma Sheshadri,

Professor, Head of CSE dept

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will cause disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Kummara Bhargavi (18BEC024)

Mohammed Safwan (18BEC028)

Parvati Jayakumar (18BEC036)

P. Chethan Krishna (18BEC040)

Approval Sheet

This project report entitled **'Data Analytics on Stackoverflow data'** by **Kummara Bhargavi (18BEC024), Mohammed Safwan (18BEC028), Parvati Jayakumar (18BEC036)** and **P. Chethan Krishna (18BEC040)** of Indian Institute of Information Technology, Dharwad is approved for the degree of Bachelor of Technology in Electronics and Communication Engineering.

Supervisor

Dr Uma Sheshadri,
Professor, Head of CSE dept.
Computer Science and Engineering
IIIT Dharwad

Head of Department

Dr Deepak K T,
Assistant Professor,
Electronics and Communication Engineering
IIIT Dharwad

Examiner

Dr. Uma Sheshadri,
Professor, Head of CSE dept.
Computer Science and Engineering
IIIT Dharwad

Abstract

Stackoverflow is the most popular Q&A website for programmers as it provides useful information to many million programmers across the globe with its database of questions and answers. This provides techies a good opportunity to explore the trends and gather insights from data. Data Analytics refers to the process of examining available datasets to draw conclusions about the information they contain. This enables us to take raw data and discover patterns to extract valuable insights from it. Thus, we performed descriptive and exploratory data analytics on Stackoverflow to discover useful information, analyse trends and draw conclusions. Stackoverflow maintains a page with the survey data files year wise for the people who wish to perform data analysis and other things. We aimed to collect data using Web Scraping instead of using survey data which is already available. Success of any data analysis depends heavily on the quality of data. Finally performed both descriptive and exploratory analysis on data collected through Web Scarping and interpreted results by creating visualizations and various graphs.

Keywords*: Stackoverflow, Data-Analytics, Requests-HTML, Python, Tableau, EDA*

Table of Contents

List of Figures

Layout of the Project Report

Chapter 1

This project analyzes the question and answer site stackoverflow which helps the performance of Q&A systems for technical domains. We are performing the data analysis on stackoverflow by the web scraping technique. This chapter gives an introduction about our project and explains the scope.

Chapter 2

This chapter describes the Literature review that we have done during the course of our project. We have explored more about the various methods with which we can collect data and analyse it.

Chapter 3

Basically the data collection can be done in many ways like there are many open API keys which are available online. With the help of API keys, we can get the data and another method is Web scraping. It helps to get the information provided in the particular website based on the url and the tags.

Chapter 4

Now-a-days data is present everywhere in the world. We are getting a huge amount of data everyday . In order to visualize the data we need some tools to convert the data to a meaningful and understandable format.We can use both tableau and power-bi tools to visualize the data

Chapter 5

In this we are going to analyze the data and display the results.We are analyzing the data in stackoverflow based on the recent activity tag and selecting the particular keywords and getting the information of the questions in the .csv format and with the help of matplotlib and tableau we are analyzing and representing data based on the keywords.

Chapter 1: Introduction

The objective of this report is to show how to perform descriptive and exploratory analytics on Stackoverflow questions data. Stackoverflow is the most used website by many programmers. Therefore, we made an attempt to perform descriptive Analytics on Stackoverflow data and finally to observe trends and insights from the analysis. For this, we mainly invested quality time in extracting data properly from the site. Collecting data manually is a tedious task. To automate the data collection, we performed Web Scraping using python. To perform web scraping, basic knowledge of HTMl, CSS and Javascript is necessary. While scraping 'requests' and 'requests-html' dependencies should be installed. These are important for making requests to a web page in a way it is like asking permission for extracting HTML content from that web page. Communication between web servers and client computers is done by sending HTTP requests and HTTP responses. Overall, this HyperText Transfer Protocol follows a request-response mechanism. Everytime we load a web page, we are making a request to the server to view that web page. Similarly, we are making a request to a web page from our python script itself using the above mentioned libraries. And then after our request is successful, we get the complete HTML code that's been written for designing the web page. Then by inspecting the code, we can extract data using classes and CSS selectors. After collecting data with certain attributes, we need to clean or manipulate data by removing unnecessary things. Then store that data in a structured format like tables. Data extraction and cleaning part is complete. Now store that structured data in a csv format by creating data frames and now we can download this .csv file and perform analysis and create visualizations out of it. For creating visualizations we used an open source software called tableau and created plots and graphs. We performed both descriptive and exploratory analysis. We used matplotlib and seaborn library. Using those libraries we created plots showing the most used tags, least used tags, relation between number of views and answers, correlation between various tags and many more.

Chapter 2: Review of Literature

The pandemic period was a big barrier for the whole world .economy ,technology and science everything got a halt in its progress. People started working from home ,every crucial meeting started happening virtually .The demand for improved technology has arised . how people are interested on technical knowledge has changed, no one could live without getting assisted by technology now ,even kindergarten classes have scheduled their classes through virtual meets .so there is a need to get analysed on what are the interested topic for techies during pandemic ,what change have occurred, what are the current trends . to find a good solution for these questions. We have worked on this project of data analysis on stackoverflow using python.

The first ever challenge was getting the data to do analysis . datas on most can only be viewed , they don't offer the provision of downloading or the functionality or api to save the data for personal use, the only feasible way is to manually copy and paste, but it isn't possible with big datas analysis . . The search to get api keys of the data and api's like selenium lead us to the world of web scraping.. Though web scraping is a very effective technique in collection of large data, there may arise controversies or questions related to the copyright and terms of service(ToS). But a web scraper is free to copy any data in the form of figures or tables which won't be suspected to be copyright infringement because it's difficult to prove copyright over such data since only a specific arrangement or particular section of data is legally protected.

Web scraping is also known as web extraction or web harvesting. It is a technique to extract data from the world wide web (WWW), for later retrieval or analysis,[stinger international publishing, bo zhao]. We have explicitly described the method of web scraping using the python libraries " requests-html" and pandas . Using this algorithm anyone can scrape data from multiple pages of a particular website. The scraped data is further performed with several analysis to lead to our conclusion. Two ways of analysis are done to ensure the authenticity of the work, analysis using tableau is used to get a visual structure and analysis using python describes the work in detail.

Here we have performed web scraping on stackoverflow data which can be easily done using python with the help of various libraries like requests_html, beautiful soup etc. The decision of choosing this method (using library requests-html) was seen as convenient throughout the work. The research went through all the accessible open study websites like github and stackoverflow to learn what the process is. Later the search was focused on the particular method(using request-html library) to complete the web scraping .

The scraped data is framed and stored to the database for further analysis .We have performed two types of analysis on the data: descriptive and exploratory . Descriptive analysis is one of the easiest ways of getting insights about data. Upon doing this analysis, we will be left with a summarized description of the numerical variables present. i.e., to get a summarized count, maximum value, minimum value, percentile, mean etc. from the data collected (1)[IJERT]. Exploratory Data Analysis, popularly referred to as EDA by Data Analysts is a common process with which we can investigate the dataset to discover patterns, outliers (anomalies) and form assumptions (hypothesis) based on our understanding. It mainly involves generating numerical statistics and creating visualizations (2)[IJITEE].

The stackoverflow datas like “questions”, “votes”, “views” etc have been analysed using python tools and libraries like seaborn. Analysis using python has given the work a complete picture of what are the current trends ,and popular tags people use, relationship between tags and views (pointing to the interests) such questions are answered scientifically with the help of plots and statistical approaches. Tableau gave the analysis a reader friendly face to the work where we can play with the variables using Bar charts, Pie charts, Bubble charts, Heat map etc to understand more about the data.

Chapter 3: Data collection method

To collect data from the Stackoverflow website, we performed Web Scraping using python. Web scraping is the term for using a program to download and process data from the web. Several modules are available to make it easy for scraping web pages using python.
The following are the steps to perform web scraping using python:

3.1 Install and Import required dependencies

```
pip install requests requests-html pandas
```

The *requests* module allows us to download files easily from the web without any issues such as connection problems, network errors and data compression. This requests module doesn't come by default with python, so we'll have to install it before importing.

To make sure the above modules are installed correctly, enter the following into the shell and see if any error shows up. If there is no error message, it means that the modules are successfully installed.

```
import requests
from requests_html import HTML
import pandas as pd
import time
```

The *requests* library allows us to exchange requests on any website. It has many essential methods to send HTTP requests for a webpage. Next *requests_html* library is an extension to the requests library and is used for parsing through the HTML page which we will get using the *requests* library. This *requests_html* library will allow us to use CSS selectors and HTML classes to extract data from the web page. Then the *pandas* library is used for performing analysis and to manipulate data for storing it in a structured format. Finally the *time* module allows us to handle various operations regarding time.

3.2 Inspecting the Stackoverflow web page

We'll need to look at the Stackoverflow web page to look into some aspects like the page url, the number of questions contained in each page, available filters to gather information, the HTML classes used in designing the web page etc. To do this, first open the Stackoverflow website by using the url https://stackoverflow.com/questions, we can look over the number of questions present per each page and the filters available as shown in figure 3.1 to see the questions we're looking for. While doing so, observe the change in url and note it. This is needed to frame the url in our python script.

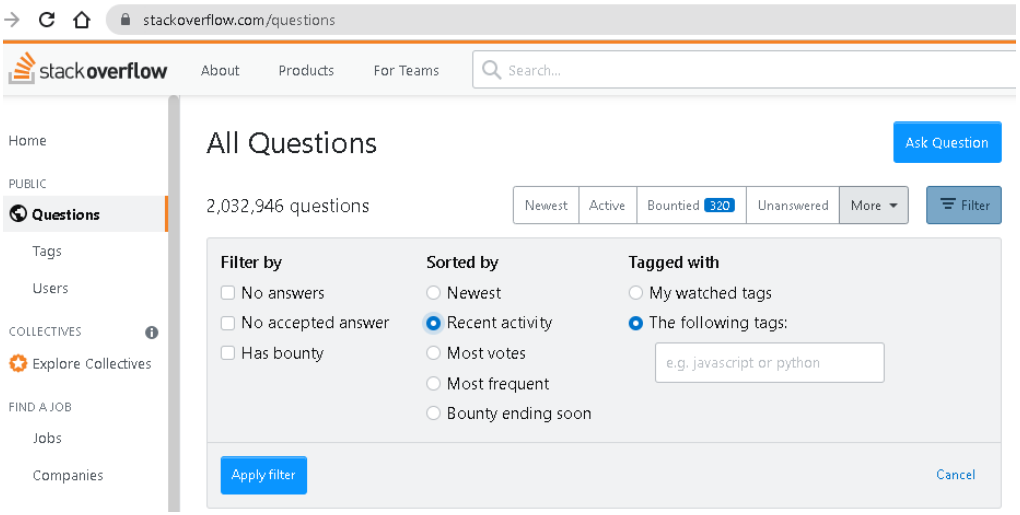

Figure 3.1 - Stackoverflow questions web page

We also recommend looking into HTML source code of the page that is going to be scrapped because it gives us the idea of what classes, ids and code is used to design the page. To do this click Ctrl + U anywhere on that web page. This shows the source code like shown below in figure 3.2

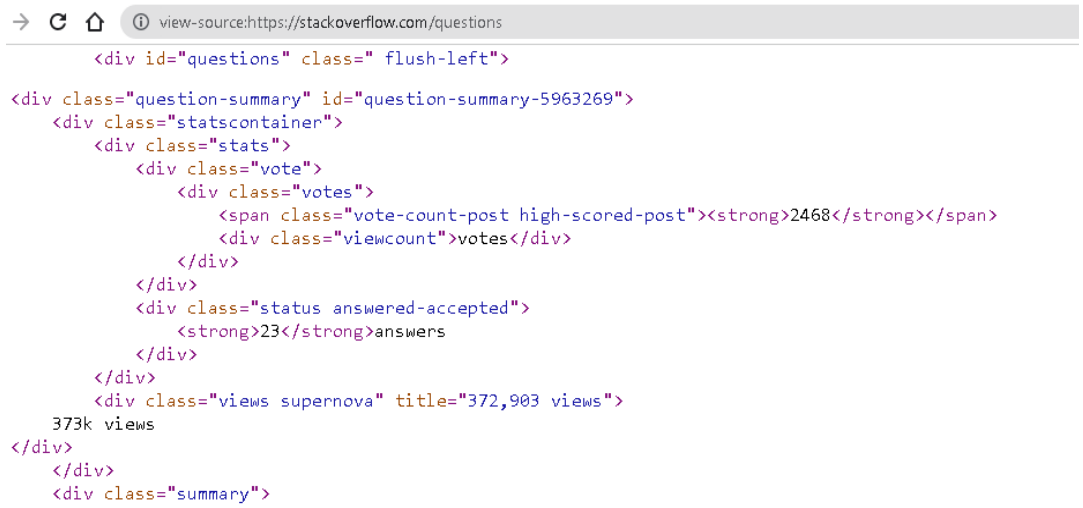

Figure 3.2 - Viewing the source of a web page

3.3 Function for downloading web page from the url

```
def extract_data_from_url(url):
  res = requests.get(url)
  if res.status_code not in range(200, 299):
     print("HTML response failure, 404 error - Page Not Found")
     return []
  html_str = res.text
  html = HTML(html=html_str)
  datas = parse_tagged_page(html)
  return datas
```

We use *requests.get()* to download the main web page from the Stackoverflow questions website. Whenever we make a request to a url it returns a response object, if the status code of that response object is not in the range of 200 to 299 then it means that the *requests.get()* operation is failed and displays the statement that is inside *print()* function. Then we are using the *HTML* function of *requests_html()* library to parse through the downloaded HTML source code, as this library can support javascript and can also parse through CSS selectors. The highlighted line in the code is another function that is defined for parsing the web page and for selecting the necessary attributes that are to be analysed.

3.4 Function for Parsing the downloaded web page

```
def parse_tagged_page(html):
  question_summaries = html.find(".question-summary")
  key_names = ['question', 'votes', 'views', 'num_answers', 'tags',   'user_name', 'date']
  classes_names = ['.question-hyperlink', '.vote', '.views', '.status', '.tags', '.user-details',
'.relativetime']
  datas = []
  for el in question_summaries:
     question_data = {}
     for i, _class in enumerate(classes_names):
        sub_el = el.find(_class, first=True)
        keyname = key_names[i]
        question_data[keyname] = clean_scraped_data(sub_el.text, keyname=keyname)
        question_data[keyname] = evaluate(question_data[keyname], keyname=keyname)

     datas.append(question_data)
  return datas
```

All the questions in the stackoverflow website page can be gathered using *.question summary* class. The attributes we considered for our analysis are questions, votes, views, number of answers, user name and date. So, for each question we are collecting data of all those attributes. Then cleaning data using *clean_scraped_data* function and for the attributes which contain numerical data like 10k views, 1m views we need to convert k and m to thousand and million. That way we are manipulating and cleaning the scraped data from each question and appending the modified data to the *datas*. The highlighted functions *clean_scraped_data* and *evaluate* are already defined in our python script.

3.5 Functions for Data Cleaning and Manipulation

```
def clean_scraped_data(text, keyname=None):
  if keyname == 'user_name':
    return text.split("\n")[0]
  if keyname == 'votes':
```

```
    if text != '1\nvote':
      return text.replace('\nvotes', '')
    else:
      return text.replace('\nvote', '')
  if keyname == 'views':
    return text.replace(' views', '')
  if keyname == 'num_answers':
    if text != '1answer':
      return text.replace('answers', '')
    else:
      return text.replace('answer', '')
  return text
```

```
def evaluate(text, keyname=None):
  if keyname == 'votes' or keyname == 'views' or keyname == 'num_answers':
    if text[-1] == 'k':
      text = pd.eval(text.replace('k', ' * 10**3',1))
    elif text[-1] == 'm':
      text = pd.eval(text.replace('m', ' * 10**6',1))
  return text
```

Both functions are used for cleaning and manipulating scraped data as mentioned earlier.

3.6 Main function for scraping data

```
def scrape_tag(tag = None, query_filter = None, max_pages=20):
  base_url = 'https://stackoverflow.com/questions/tagged/'
  datas = []
  for p in range(max_pages):
    page_num = p + 1
    url = f"{base_url}{tag}?sort={query_filter}&page={page_num}"
    print(url)
    datas += extract_data_from_url(url)
    time.sleep(1.2)
  df = pd.DataFrame(datas)
  df['votes'] = df['votes'].astype(int)
```

```
df['num_answers'] = df['num_answers'].astype(int)
df['views'] = df['views'].astype(int)
df.to_csv(tag+"_scraped.csv", index=False)
return df
```

The arguments defined in the main function *scrape_tag* are *tag*, *query_filter* and *max_pages*. Next we defined an empty list *datas* to store data while functioning. The *base_url* is framed by seeing the website url.

We wrote a for loop to extract data from each page upto the *max_pages* number we defined. After extracting data from each page, we are letting our python script stop running for 1.2 seconds of time. This is because extracting data continuously from the web page can cause serious problems like crashing of websites etc. Therefore, we need to give some time after extracting data from each page. After extracting data from all pages, for loop ends. Then we are creating a data frame and converting the data collected into structured format using the *pandas* library. We are converting some attributes which contain numerical data into integer type and finally we get a *.csv* file.

Chapter 4: Tableau Visualization

Tableau is a free open source visualization software which can be used in analyzing the data in the meaningful format .The main use of the tableau is it helps in data-storytelling techniques which can be easily understood by anyone.We all know that the data science is a booming field data is present in almost all aspects of life.We can visualize the huge amount of data in a simple and understandable way and we can present data of the 3 or 4 tags easily at a time.No coding is required for learning the data visualization both technical and non technical.It connects and extracts the data stored in various places.

The pulled data can be either connected live or extracted to the Tableau desktop.This is where the Data analysts and data engineer work with the pulled data and develop the visualizations.The created dashboards are shared with the users and it can be accessed and viewed easily.This is an enterprise platform where the collaboration , distribution features are provided.With the help of tableau software the users have the better experience of viewing and accessing the files.

The best features of the tableau are :

1. Data visualization and Data Blending
2. Real time data analysis
3. Imports the large size of the data
4. Creation of no code queries
5. Data profiling
6. Intuitive Dashboard creation
7. Connect to variety of data sources

Figure 4.1

4.1 Bar chart

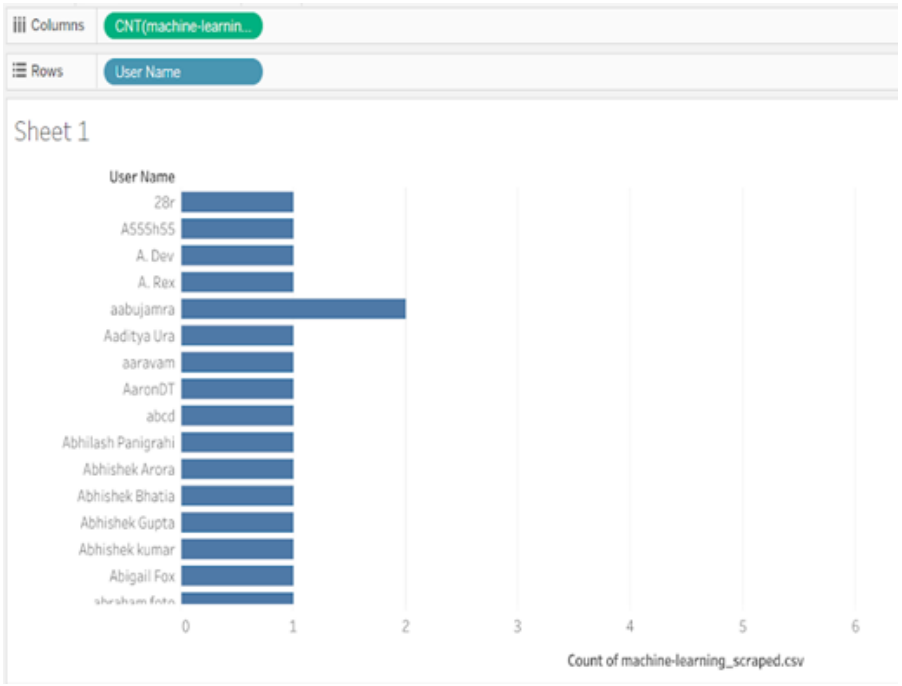

Figure 4.2

The above image (Figure 4.2) is the analyzing and visualization of the data of machine-learning_scraped.csv file and we have used the bar graph to plot the result using only one tag called usernames with respect to the rows and the main file .csv into columns side .

4.2 Heat map

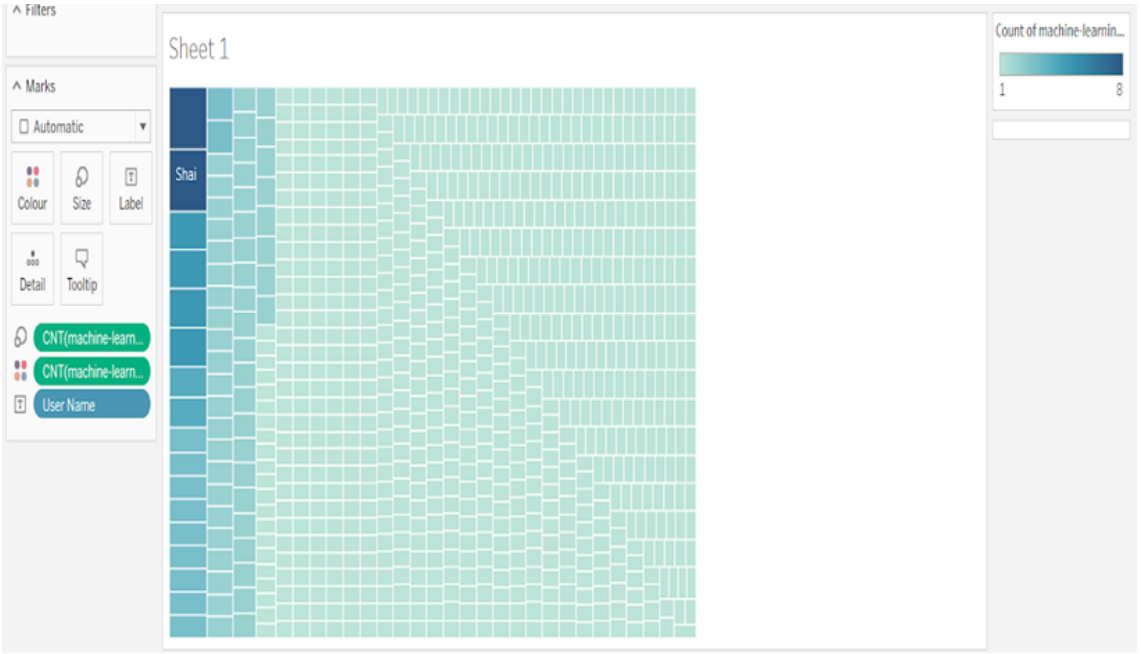

Figure 4.3

The above plot (Figure 4.3) is defined on tag username and defines most of the answered questions with the dark blue and medium number of answers with the light blue and the others are with an equal number of questions answered.

4.3 Pie chart

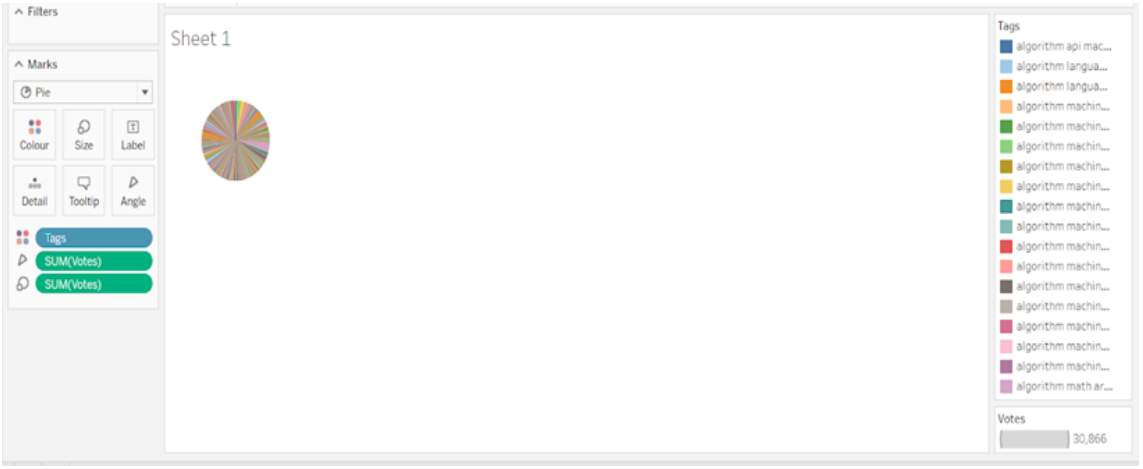

Figure 4.4

This pie chart (Figure 4.4) explains about the votes with each number of the questions and the total votes for all the questions is 30,866 and each question is defined as different colours and displayed according to the votes from high to low.

4.4 Bubble chart

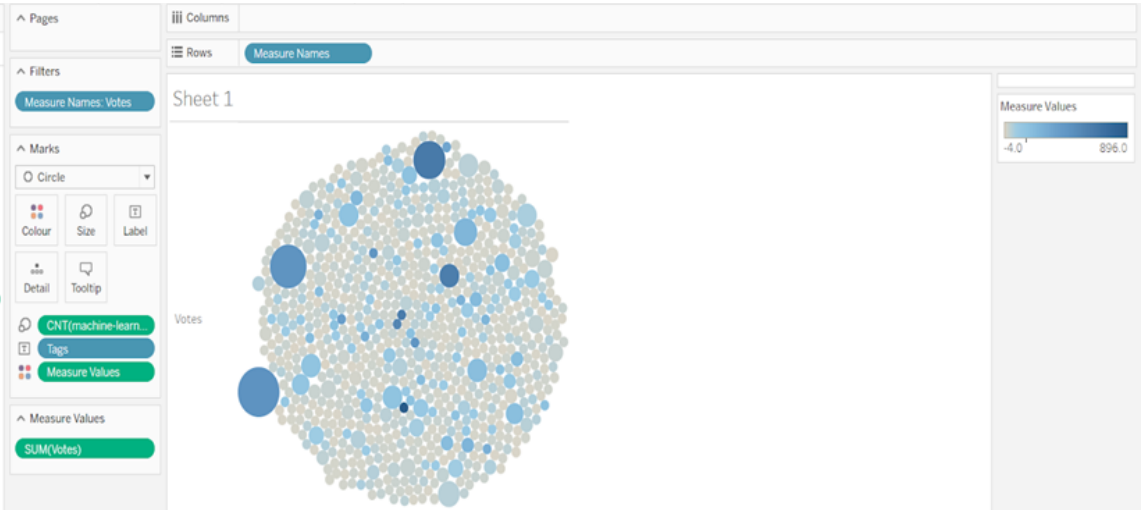

Figure 4.5

This figure (Figure 4.5) explains that with the multiple tags like the question tags and the number of votes to that questions will be displayed and the question tags with more votes will be in dark blue colour [max votes is 896]and the question vil minimum of votes will be displayed in the light colours [min votes is 4].

4.5 Visualization of all tags

Figure 4.6

This explains the whole tags with the all the tags and the main.csv file which defines the all the date,questions,tags,votes and the colours represent counts,num answers and votes and questions asked and date where mentioned in the above image also.

Chapter 5: Analysis and Results

By definition, Data Analysis is the process of capturing useful information by cleaning, transforming and modelling a dataset.

In general, there are four types of data analysis methods:

1. Descriptive Analysis
2. Exploratory Data Analysis
3. Predictive Analysis
4. Inferential Analysis

In this project, we have performed both Descriptive as well as Exploratory Data Analysis.

```
Which analysis would you like to perform?
1.Analyse Votes
2.Analyse Number of Answers
3.Analyse Views
4.Are the tags mentioned in questions similar to each other?
5.Which tags are popularly used by Stackoverflow community in this domain?
6.How many tags are present per question?
7.Are the number of views of question statistically related to number of answers?
8.Are the tags related to Views?
9.Analyse Votes, Number of Answers, Views
10.Exit
```

Figure 5.1

5.1 Descriptive Analysis

Descriptive analysis is one of the easiest ways of getting insights about data. Upon doing this analysis, we will be left with a summarized description of the numerical variables present. I.e., to get a summarized count, maximum value, minimum value, percentile, mean etc. from the data, we use this type of analysis. By doing so, we will get high-level information from the data present in the dataset. Additionally, we can use bar graphs, line charts etc. to perform this.

Let us assume that the user wants to get insights from the machine learning domain questions when the ‘Most Frequent’ sort method was used. A total of 1000 questions were collected.

5.1.1 Relationship between views and number of answers

In our project we have tried to analyse if the views present in questions are related to the number of answers in that question.

.describe() function of the *pandas* library was used to find this insight.

```
view_q = df[['views','num_answers']].drop_duplicates().reset_index(drop=True)
display(view_q.astype(int).describe())
```

	views	num_answers
count	440.000000	440.000000
mean	48529.188636	4.784091
std	60049.578218	4.113561
min	42.000000	0.000000
25%	10000.000000	2.000000
50%	30000.000000	4.000000
75%	60000.000000	6.000000
max	386000.000000	28.000000

Figure 5.2

From the output received (figure 5.2), we can understand that:

- There were many duplicate rows having the same number of views and answers. From 1000 questions, only 440 unique counts are present.
- A particular question took a maximum of 386000 views to obtain 28 answers.
- Whereas, there was even a question which failed to get any accepted answers despite 42 views.
- On an average, the questions had 48529 views and 4-5 answers which varied by a factor of 60049.57 and 4.11 respectively.
- 25% of the questions took around 1000 views to obtain 2 answers.
- 50% of the questions took around 3000 views to obtain 4 answers.
- And, 75% of the questions took 6000 views to obtain 6 answers.

5.1.2 Popularly used tags

Let us now understand which tags are popularly used by Stack overflow community in a mentioned domain.

```
tags_list = list(df['tags'].unique())
all_tags = []
for tag in tqdm_notebook(tags_list):
    all_tags += tag.split(' ')

count_tags = Counter(all_tags)
tags_list = pd.DataFrame([list(count_tags.keys()), list(count_tags.values())])
tags_list = tags_list.transpose()
tags_list.columns = ['tags', 'counts']
tags_list = tags_list.sort_values(by='counts', ascending=False)
plt.barh(list(tags_list['tags'])[:25], list(tags_list['counts'])[:25])
```

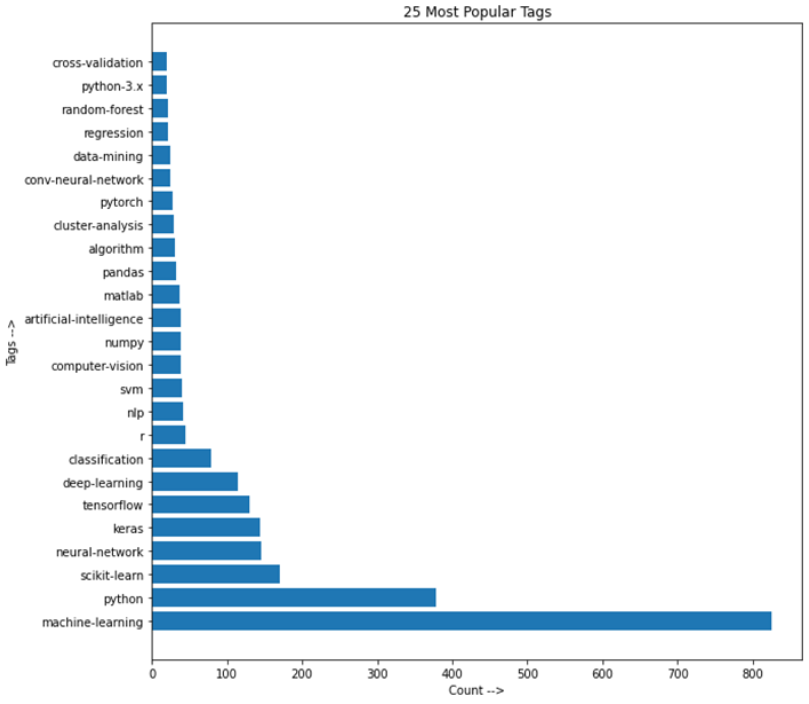

Figure 5.3

By seeing the above plotted bar graph (figure 5.3), we can understand that in the machine learning domain of stack overflow websites, the community is by far most interested in tags – python, scikit-learn, neural-networks, keras and tensorflow. Tags like cross-validation,

python-3.x, random-forest, regression etc. have comparatively lesser popularity among the users when they add a question.

5.1.3 Tags present per question

Now let us graphically visualize how many tags are present per question.

```
num_tags = []

for tag_ in tqdm_notebook(tags):
        num_tags.append(len(tag_.split(' ')))

num_tags = Counter(num_tags)

plt.bar(list(num_tags.keys()), list(num_tags.values()))
```

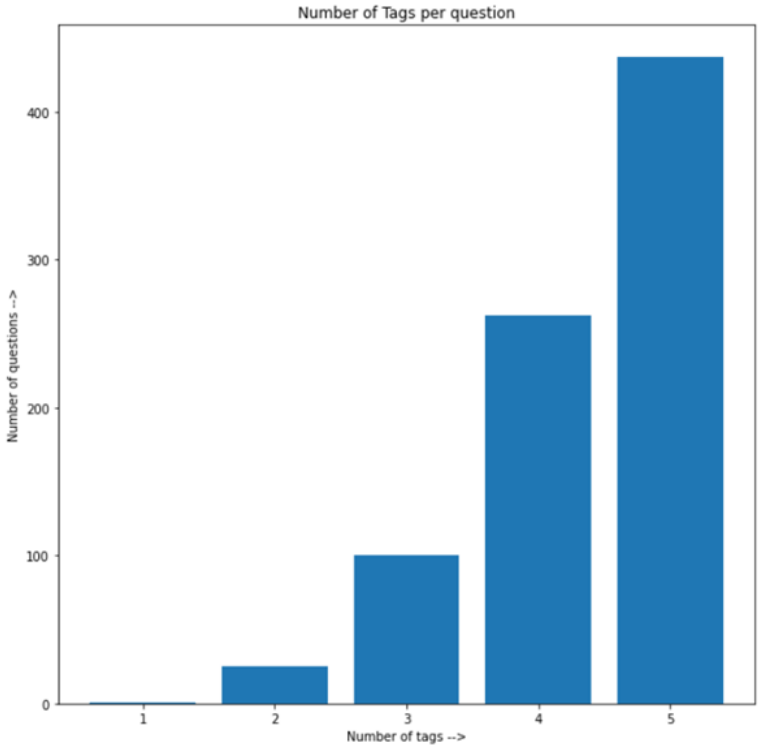

Figure 5.4

On the horizontal axis (figure 5.4) we have a scale ranging from 1 to 5 that denotes the minimum and maximum number of tags that are possible per question (i.e., while user enters a question, he/she should give minimum 1 tag and can give upto 5 tags maximum).

The above graph says that in the 1000 questions collected, around 450 questions had all 5 tags given, 250+ questions had 4 tags given, approximately 100 questions had 3 tags, less than 50 questions had 2 tags and very few (less than questions) had 1 tag attached to them (Which probably might be 'machine-learning', since this was a mandatory tag to choose the questions).

By seeing the above graph (figure 5.4), we can see that there are many questions having all 5 tags attached to them.

5.1.4 Relationship between views and individual tags

Let us see now if giving additional tags will have any impact on the number of views, i.e., in general let us see if the individual tags have any impact on the number of views each question got.

```
tag_views = df[['tags', 'views']].drop_duplicates().reset_index(drop = True)
tagView = {}
for i in tqdm_notebook(range(len(tag_views['tags']))):
    for tag in tag_views['tags'][i].split(' '):
        try:
            tagView[tag].append(tag_views['views'][i])
        except Exception:
            tagView[tag] = [tag_views['views'][i]]
tagViewAvg = {}
for key in list(tagView.keys()):
    tagViewAvg[key] = np.average(tagView[key])
tagViewAvg = pd.DataFrame([list(tagViewAvg.keys()), list(tagViewAvg. values())])
tagViewAvg = tagViewAvg.transpose()
tagViewAvg.columns = ['tags', 'views']
tagViewAvg = tagViewAvg.sort_values('views')
plt.barh(list(tagViewAvg['tags'])[-25:],list(tagViewAvg['views'])[-25:])
```

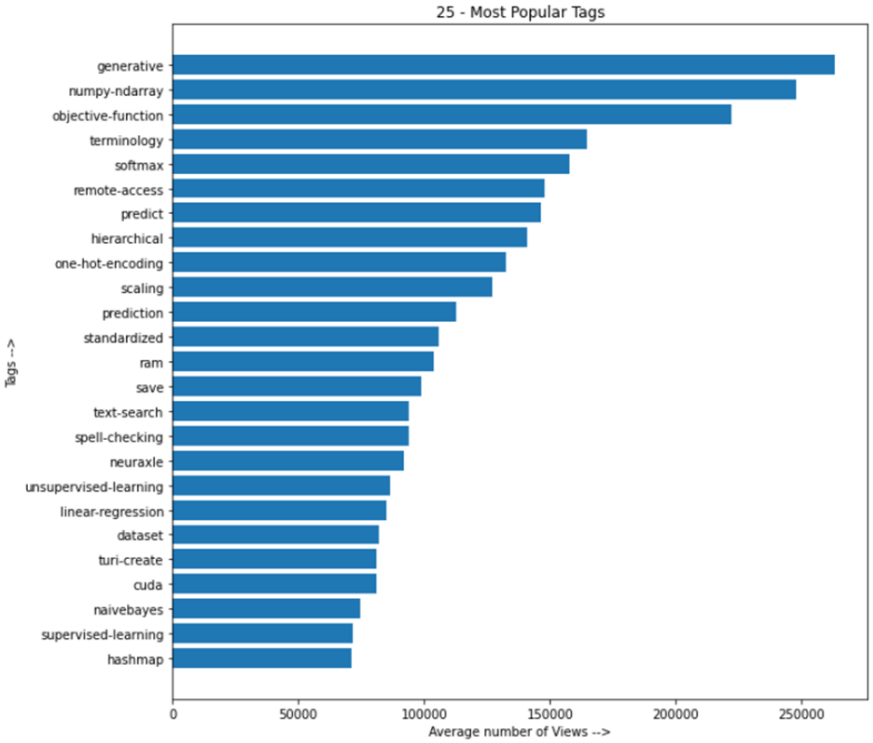

Figure 5.5

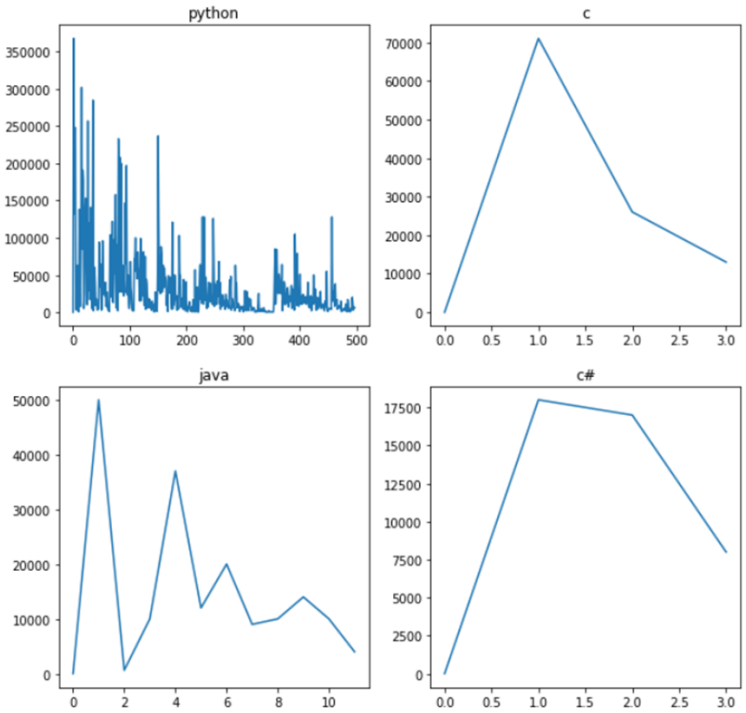

Figure 5.6

Figure 5.5 says that questions with tags generative, numpy-ndarray, objective-function, terminology, softmax have received more viewers than the questions with tags hashmap, supervised learning, naivebayes etc. This doesn't imply that the community is interested in these topics of machine-learning domain, instead it could mean that the mentioned topic is very small (thus got fewer questions with higher audience), or it could even mean that those topics are not that explored and isn't a trend in the community.

In Figure 5.6, we have plotted a line chart to compare how many views popular programming languages have received with respect to the number of questions. Here, we can see a comparison between python, c, c# and java. Python is a popular tag used in machine-learning questions by the users. Here, we can see that it has around 500 questions, one with a maximum view of 350000+. Java tag is used more than c, whereas c# has only 3 questions in the whole set.

5.1.5 Similarity between individual tags

Here, we will try to visualize a plot from which we can manually interpret if there are any similarities between the individual tags. For this, let us use the Word2Vec model.

The Word2Vec model is commonly used in Natural Language processing. It uses a two-layer neural network to process text by 'vectorizing' words. It is commonly used to learn word associations between the texts given as input. There are two architectures commonly used here, Skip-Gram (SG) model and Continuous Bag of Words (CBOW).

Here, we have used Skip-Gram architecture whereby we will give one text as input and get many vectors. A Gensim module was used to implement the same. Minimum word size to be counted as a vector is given by min_count = 1. Size of each vector is given by N=256.

We are then performing Principal Component Analysis to reduce the vector dimension to 2 to visualize the same on a scatter plot.

```
tags = []
for tag in list(df['tags']):
        tags.append(tag.split(' '))

embedding = Word2Vec(tags, sg=1, size=256, min_count=1)
X = embedding[embedding.wv.vocab]
pca = PCA(n_components=2)
result = pca.fit_transform(X)
```

```
fig, ax = plt.subplots()
ax.plot(result[:, 0], result[:, 1], 'o')

fig, ax = plt.subplots()
ax.scatter(result[:25, 0], result[:25, 1])
for j, txt in enumerate(list(embedding.wv.vocab)[:25]):
        ax.annotate(txt, result[j])
```

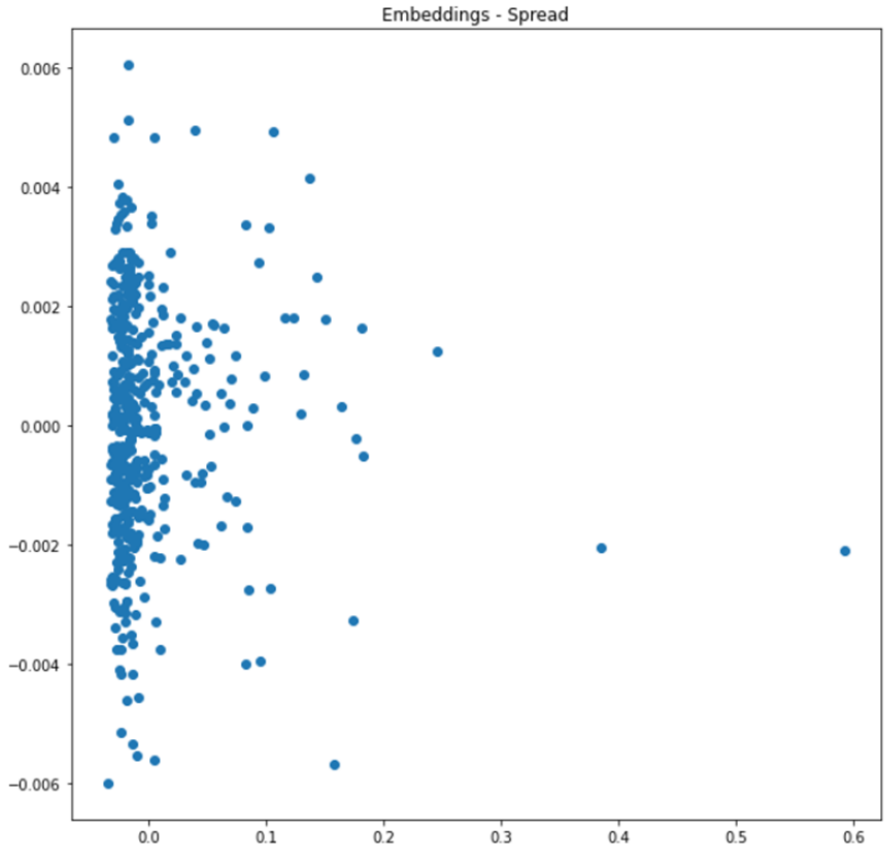

Figure 5.7

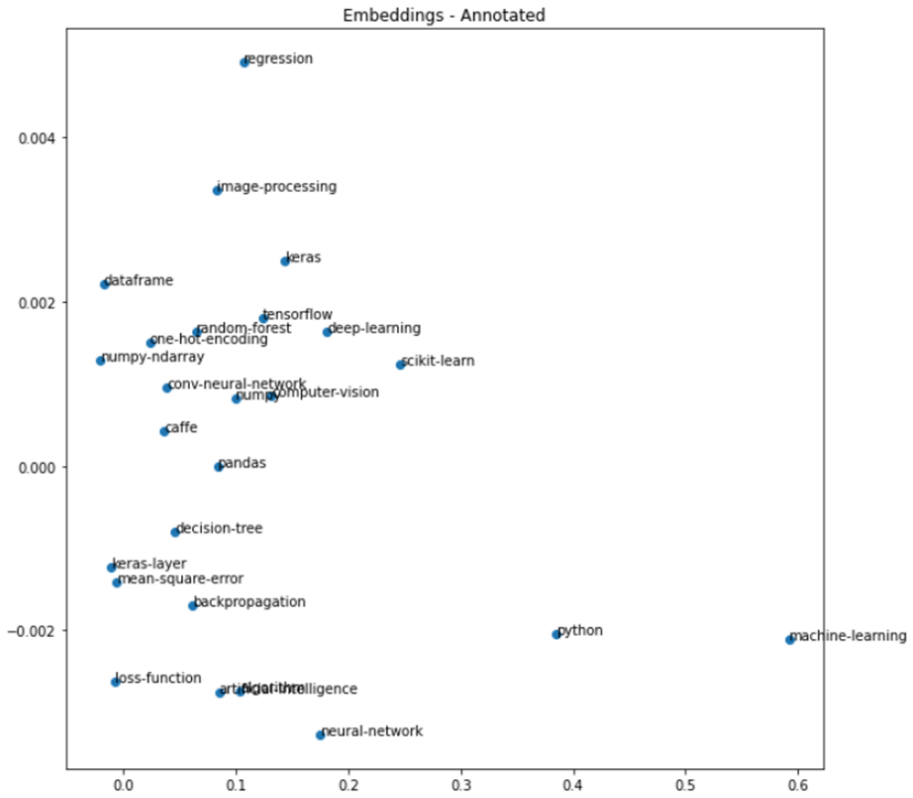

Figure 5.8

Figure 5.7 shows us all the vector points corresponding to the tags list present with us. To see their corresponding vocabulary, let us observe Figure 5.8. For simplicity we have chosen 25 tags to avoid crowding.

Assumptions for example, dataframe tag is visually closer to numpy-ndarray than loss-function tag. This means, dataframe is more similar or is closely related to numpy-ndarray then loss-function word. Similar predictions can be made for other words.

Let us now get into the Exploratory Data Analysis.

5.2 Exploratory Data Analysis

Exploratory Data Analysis, popularly referred to as EDA by Data Analysts is a common process with which we can investigate the dataset to discover patterns, outliers (anomalies) and form assumptions (hypothesis) based on our understanding. It mainly involves generating numerical statistics and creating visualizations.

The Seaborn library of Python was used throughout to do this Data Analysis method.

In EDA, we usually perform Univariate, Bivariate and Multivariate Analysis.

5.2.1 Univariate Analysis

Univariate Analysis is perhaps the simplest statistical analysis method where we will examine each variable separately (or individually).

To analyse the numerical columns (votes, views and num_answers) individually, we have used Box plot.

5.2.1.1 Box plot: Views

Box plot is very helpful for detecting outliers present in the data. Here, we can get the statistical distribution of the data comprising minimum, maximum, 1st quartile, 2nd quartile (median) and 3rd quartile.

For example, let us see the box plot of the column: 'views'.

```
sns.boxplot(df['views'])
```

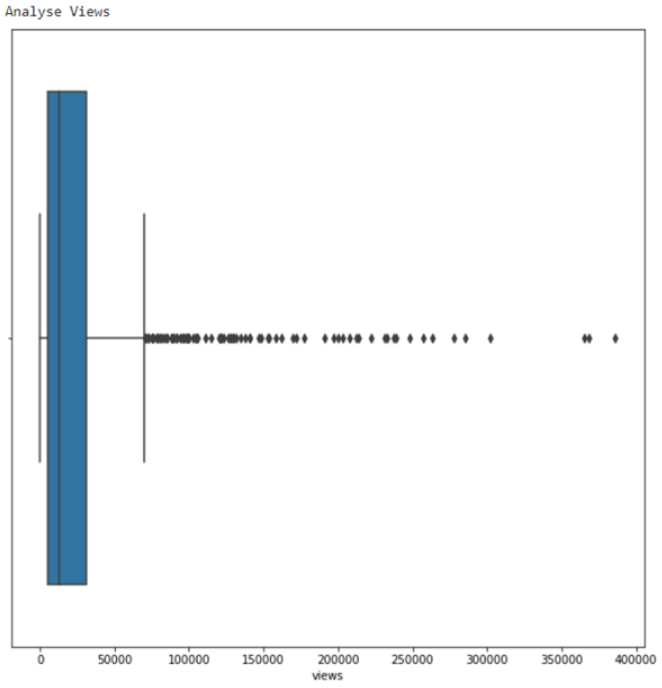

Figure 5.9

The blue portion visible in figure 5.9 is where the majority of the views lie. The 2nd interquartile region (middle line in the blue portion) tells us the median view value. Probably

the minimum value too lies in the interquartile region, but the scatter points after 7000views mark are the outliers (ie, anomalies) which includes the last 37k views value (maximum).

5.2.1.2 Box plot: Votes

For a large continuous value of votes and num_answers, we can plot box plot to get more insights on the distribution of the data points.

```
sns.boxplot(df['votes'])
```

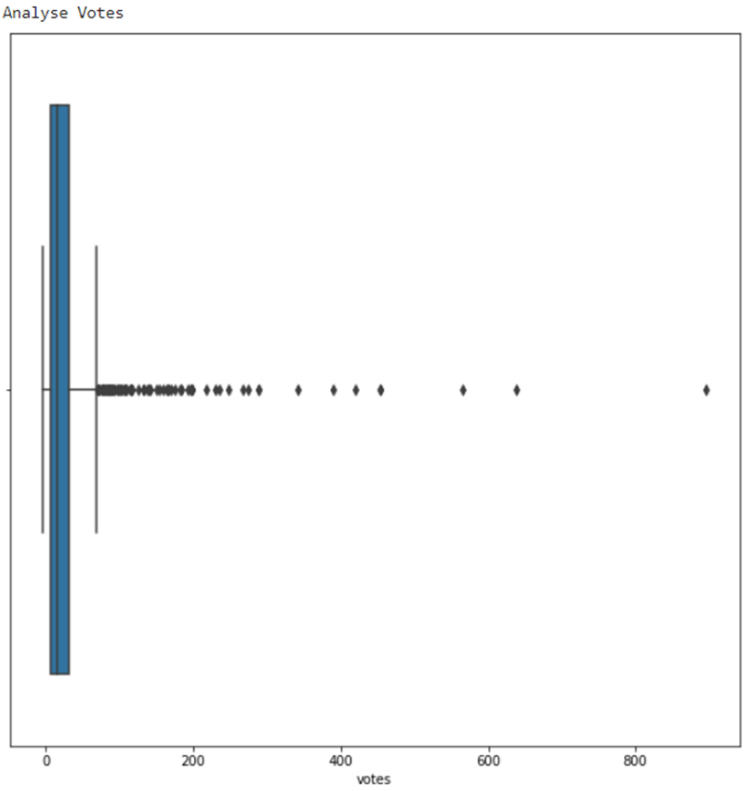

Figure 5.10

5.2.1.3 Box plot: Number of Answers

```
sns.boxplot(df['num_answers'])
```

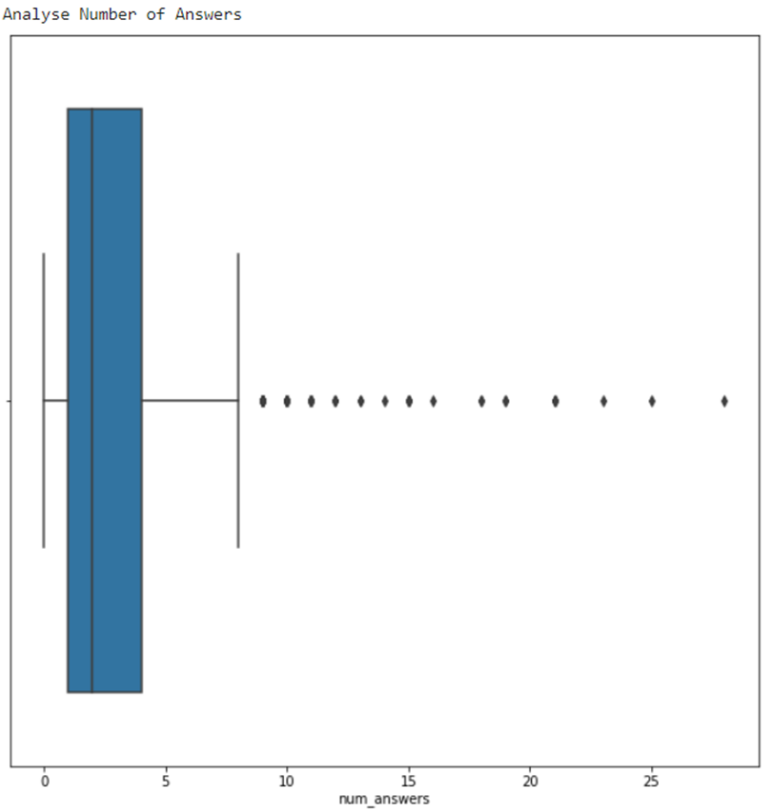

Figure 5.11

Similar predictions like how we did for views applies here in votes and num_answers too.

5.2.1.4 Count plot: Views

To get a visual count distribution of the categorical variables, we can use the Count plot. We can see the count of each categorical bin here. Since the values in columns ‘votes’, ‘views’ and ‘num_answers’ are distinct, we can use the Count plot on these columns.

```
df['views'].value_counts()/len(df)*100
sns.countplot(df['views'])
```

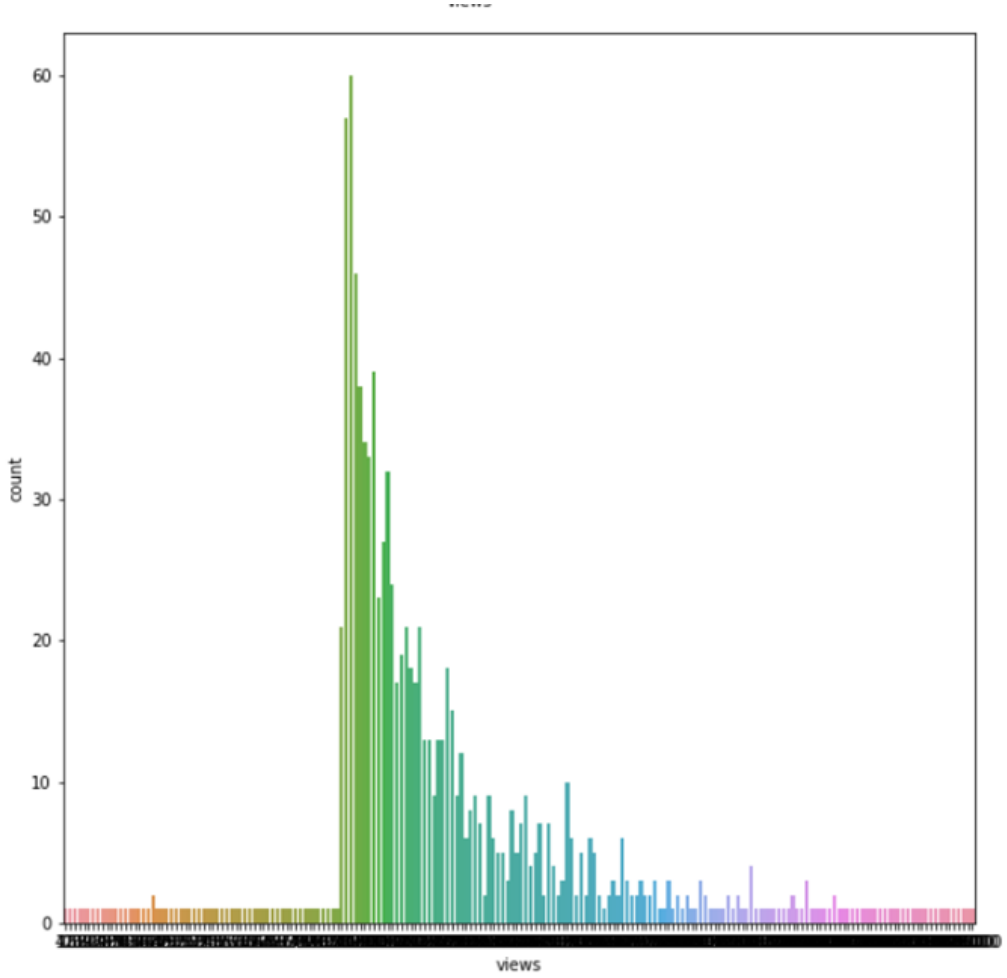

Figure 5.12

Here, for each view on the horizontal axis, the vertical axis shows the corresponding count. To get insights about the sort query filter “Most Frequent”, this graph will be very beneficial. Additionally, just to understand the distribution, we can use a count plot.

5.2.1.5 Count plot: Votes

Similar insights like we saw in ‘Views’ applies here too

```
df['votes'].value_counts()/len(df)*100
sns.countplot(df['votes'])
```

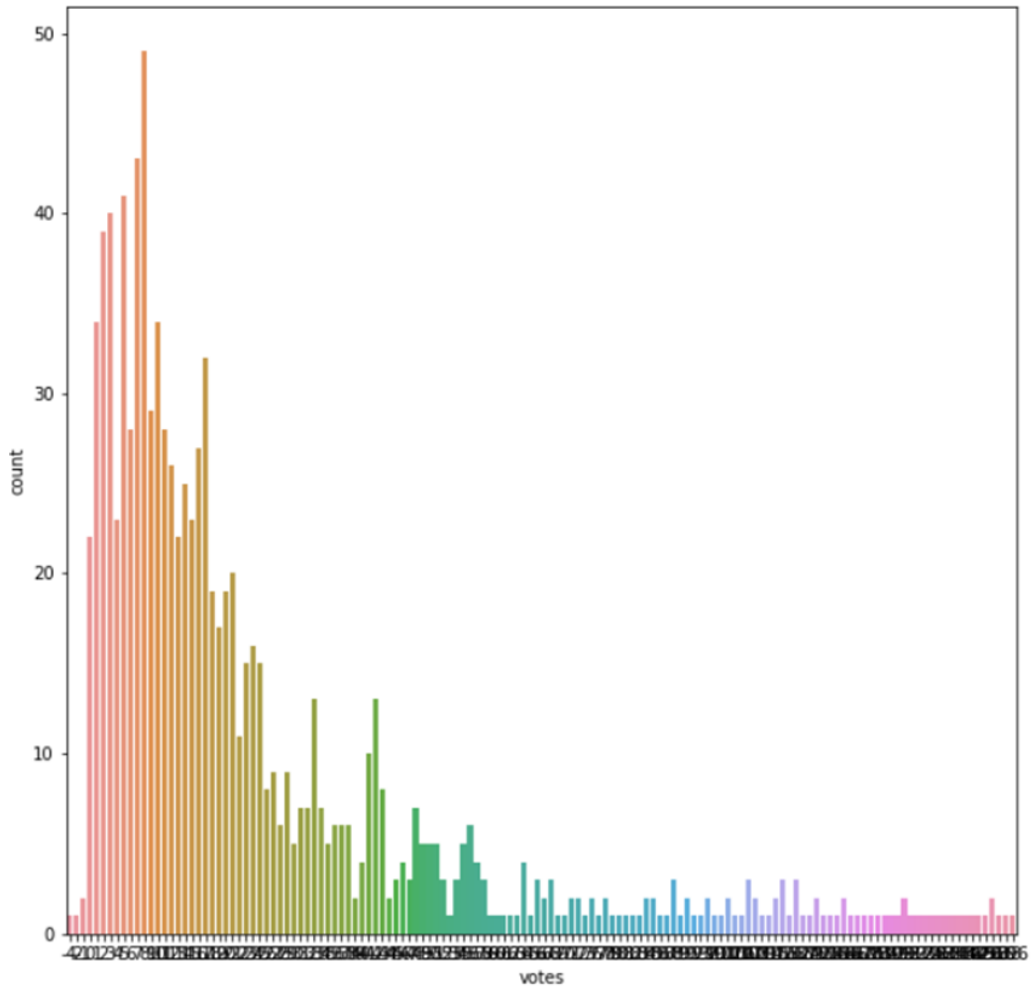

Figure 5.13

5.2.1.6 Count plot: Number of Answers

Similar insights like what we saw in Views and Votes applies here too.

```
df['votes'].value_counts()/len(df)*100
sns.countplot(df['votes'])
```

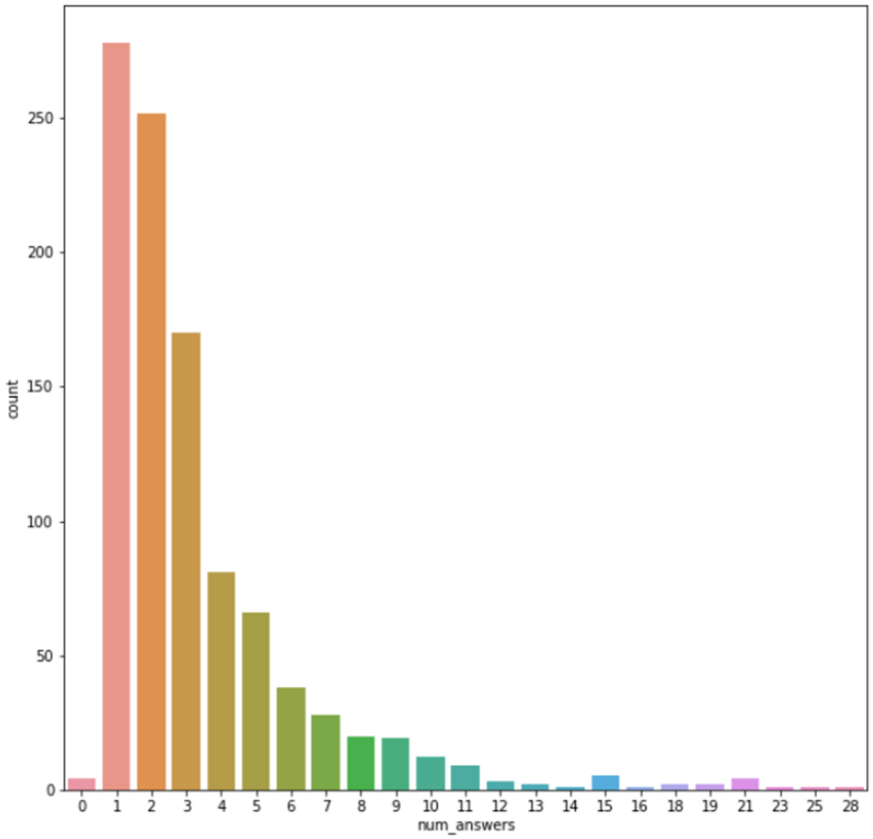

Figure 5.14

5.2.2 Bivariate Analysis

To get a relationship between two variables, we can use Bivariate Analysis. Common Seaborn plots like distplot, line plot etc. can be used for this analysis. Since we have computed the relation between variables in the Descriptive Analysis, we are not getting too deep into this 2nd method.

5.2.3 Multivariate Analysis

The final and very important method is Multivariate Analysis; where we can get the relationship between continuous multi variables at a time.

5.2.3.1 Scatter plot: Pair plot

Pairplot is used to plot all the variables as a scatter plot at a time taking one horizontal variable and one vertical variable.

```
sns.pairplot(df)
```

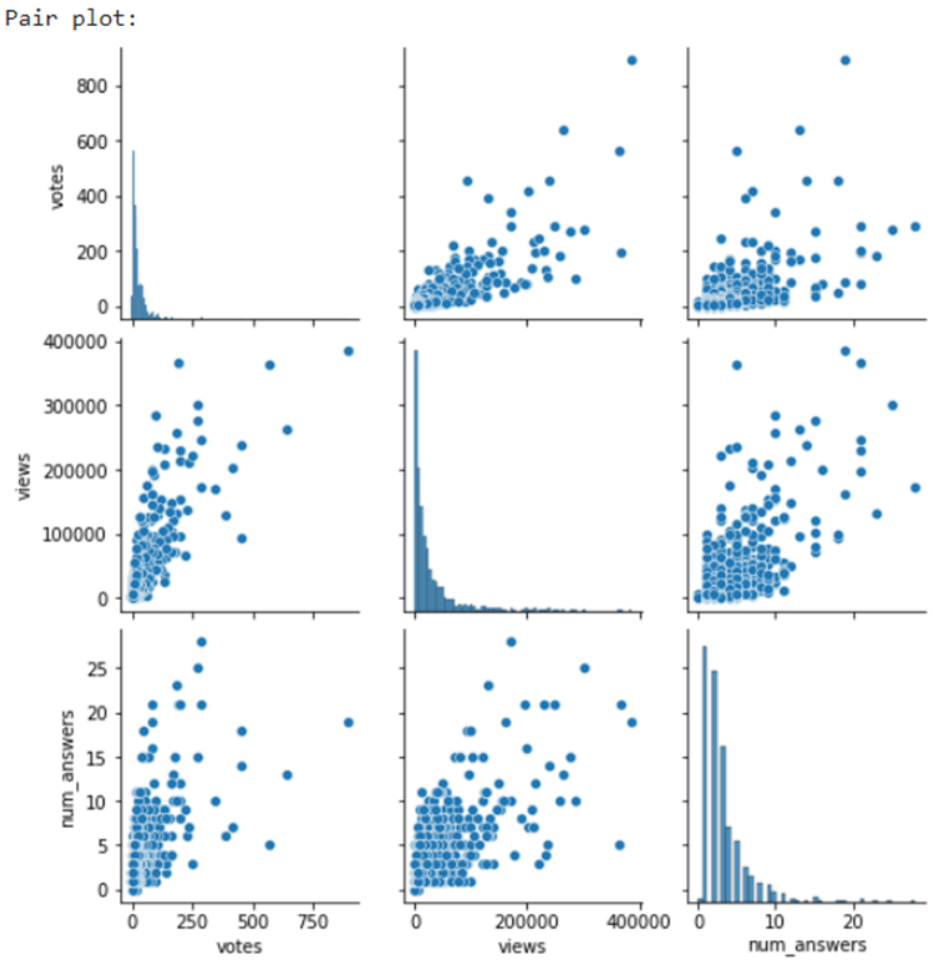

Figure 5.15

In Figure 5.15, we can see all the scatter plots plotted together (at the same time). We can get insights like how votes are related to num_answers, are the questions with higher numbers of answers having many views etc. Also, if you notice, you can see the same results as we got for countplot here in the diagonal plots too since we are comparing the same variable to the same variable.

5.2.3.2 Correlation: Heat map

To get a correlation value for each variable with respect to the other (i.e., to understand if there is any relationship between the variables), we can use seaborn's heatmap function. A highly correlated variable will have value 1 and variable with no correlation at all will have value 0.

```
sns.heatmap(df.corr())
```

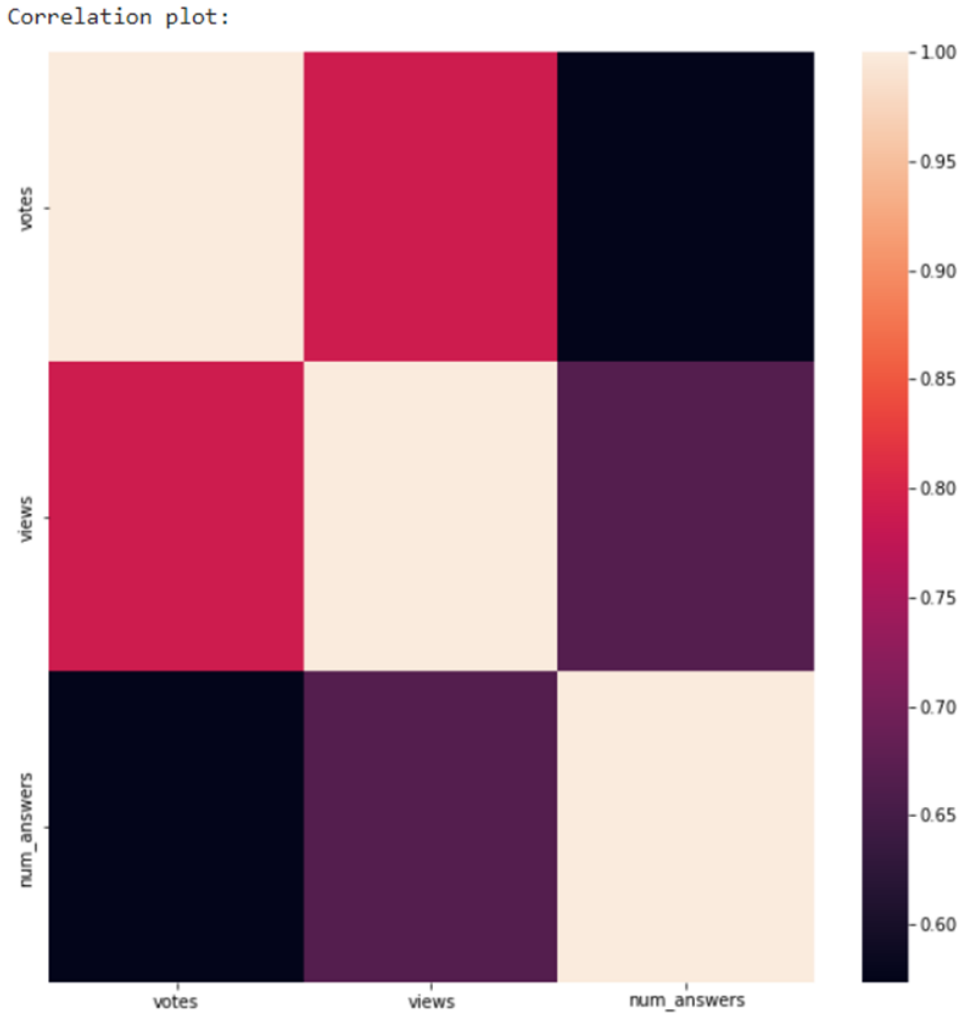

Figure 5.16

From figure 5.16, we can see that when we compare the same variable with each other, correlation value will be 1 (votes-votes, views-views, num_answers-num_answers). Votes-views have a correlation value of approximately 0.75, ie, they have 75% similarity to each other. Views and Number of Answers are 70% correlated to each other, whereas votes and Number of answers are only 60% correlated.

Summary

In this project, our goal was to collect the dataset, visualize and analyse it. We have chosen to work on Stackoverflow questions (https://stackoverflow.com/questions) dataset.

The data collection method was done manually using the Request-HTML web scraping method. We have collected the question summary consisting of the question, number of votes, total number of views on it, number of accepted answers for each question, additional tags attached, name of the user who asked and the date and time when the question was asked. We can collect as many questions as we need from the website. Just for simplicity, analysis was done on a total of 1000 questions. Users can themselves choose any domain they need to perform the analysis. They are also given the freedom to choose a search filter (Votes, Recent Activity, Most Frequent etc.).

The data visualization process was done on Tableau software. We have uploaded the collected data to the software allowing the software to make visualizations for us. We can play with the variables using Bar charts, Pie charts, Bubble charts, Heat map etc to understand more about the data.

Additionally we have performed two data analysis methods: Descriptive Analysis and Exploratory Data Analysis. Various Python libraries (including Seaborn) were used for this process. The following analysis can be done in our user interface.

- Analyse Votes.
- Analyse Number of Answers.
- Analyse Views.
- Are the tags mentioned in questions similar to each other?
- Which tags are popularly used by the Stackoverflow community in this domain?
- How many tags are present per question?
- Are the number of views of a question statistically related to the number of answers?
- Are the tags related to Views?
- Analyse Votes, Number of Answers, Views.

Conclusion

Web scraping is the process of collecting of raw data from the web and return the source code - HTML and sending the request of a specific URL and returns the code for that page and the collected data can be used as data source of any tag from the website and able to get the information and the information can be converted into .csv file which will be helpful in analyzing the data and it can be plotted through the python matplotlib function and helps in the identifying the trends which topic has more number of votes and people facing more problems in a specific topics it can be plotted.We can measure the data theoretically like finding the mean,standard deviation and visualize the data more clearly by using the Tableau and get the results.

Appendices

Appendix I: Data Collection from Stackoverflow website

```
#Scraping multiple pages:Functions for data cleaning and manupulation
def clean_scraped_data(text, keyname=None):
  if keyname == 'user_name':
    return text.split("\n")[0]
  if keyname == 'votes':
      if text != '1\nvote':
        return text.replace('\nvotes', '')
      else:
        return text.replace('\nvote', '')
  if keyname == 'views':
      return text.replace(' views', '')
  if keyname == 'num_answers':
    if text != '1answer':
      return text.replace('answers', '')
    else:
      return text.replace('answer', '')
  return text

def evaluate(text, keyname=None):
  if keyname == 'votes' or keyname == 'views' or keyname == 'num_answers':
    if text[-1] == 'k':
      text = pd.eval(text.replace('k', ' * 10**3',1))
    elif text[-1] == 'm':
      text = pd.eval(text.replace('m', ' * 10**6',1))
  return text

#Function for Parsing the Webpage
def parse_tagged_page(html):
  question_summaries = html.find(".question-summary")
  key_names = ['question', 'votes', 'views', 'num_answers', 'tags', 'user_name', 'date']
  classes_names = ['.question-hyperlink', '.vote', '.views', '.status', '.tags', '.user-details',
'.relativetime']
  datas = []
```

```
  for el in question_summaries:
      question_data = {}
      for i, _class in enumerate(classes_names):
          sub_el = el.find(_class, first=True)
          keyname = key_names[i]
          question_data[keyname] = clean_scraped_data(sub_el.text, keyname=keyname)
          question_data[keyname] = evaluate(question_data[keyname], keyname=keyname)

      datas.append(question_data)
  return datas


#Function for extracting the data from URL
def extract_data_from_url(url):
  res = requests.get(url)
  if res.status_code not in range(200, 299):
      print("HTML response failure, retrying..")
      return []
  html_str = res.text
  html = HTML(html=html_str)
  datas = parse_tagged_page(html)
  return datas


#Function that estimates the sort query filter
def filter_sort(filter = None):
  if filter == '1':
    return "Newest"
  elif filter == '2':
    return "RecentActivity"
  elif filter == '3':
    return "MostVotes"
  elif filter == '4':
    return "MostFrequent"
  elif filter == '5':
    return "BountyEndingSoon"
  else:
    return []
```

```
#Main function for scraping data
def scrape_tag(tag = None, query_filter = None, max_pages=20):
  base_url = 'https://stackoverflow.com/questions/tagged/'
  datas = []
  for p in range(max_pages):
      page_num = p + 1
      url = f"{base_url}{tag}?sort={query_filter}&page={page_num}"
      print(url)
      datas += extract_data_from_url(url)
      time.sleep(1.2)
  df = pd.DataFrame(datas)
  df['votes'] = df['votes'].astype(int)
  df['num_answers'] = df['num_answers'].astype(int)
  df['views'] = df['views'].astype(int)
  df.to_csv(tag+"_scraped.csv", index=False)
  return df
```

Appendix II: Data Analysis Functions

```
#Analyse votes
def ip_1(df=None):
  print("\nAnalyse Votes")
  sns.boxplot(df['votes'])
  plt.show()
  df['votes'].value_counts()/len(df)*100
  sns.countplot(df['votes'])
  plt.show()

#Analyse Number of Answers
def ip_2(df=None):
  print("\nAnalyse Number of Answers")
  sns.boxplot(df['num_answers'])
  plt.show()
  df['num_answers'].value_counts()/len(df)*100
  sns.countplot(df['num_answers'])
  plt.show()
```

```
#Analyse Views
def ip_3(df=None):
  print("\nAnalyse Views")
  sns.boxplot(df['views'])
  plt.show()
  df['views'].value_counts()/len(df)*100
  sns.countplot(df['views'])
  plt.show()


#Are the tags mentioned in questions similar to each other
def ip_4(df=None):
  print("\nAre the tags mentioned in questions similar to each other?")
  tags = []
  for tag in list(df['tags']):
    tags.append(tag.split(' '))

  embedding = Word2Vec(tags, sg=1, size=256, min_count=1)
  X = embedding[embedding.wv.vocab]
  pca = PCA(n_components=2)
  result = pca.fit_transform(X)

  fig, ax = plt.subplots()
  ax.plot(result[:, 0], result[:, 1], 'o')
  ax.set_title('Embeddings - Spread')

  fig, ax = plt.subplots()
  ax.set_title('Embeddings - Annotated')
  ax.scatter(result[:25, 0], result[:25, 1])
  for j, txt in enumerate(list(embedding.wv.vocab)[:25]):
    ax.annotate(txt, result[j])

  plt.show()


#Which tags are popularly used by Stackoverflow Community?
def ip_5(df=None):
  print("\nWhich tags are popularly used by Stackoverflow community in this domain?")
  tags_list = list(df['tags'].unique())
  all_tags = []
```

```
  for tag in tqdm_notebook(tags_list):
      all_tags += tag.split(' ')

  count_tags = Counter(all_tags)
  tags_list = pd.DataFrame([list(count_tags.keys()), list(count_tags.values())])
  tags_list = tags_list.transpose()
  tags_list.columns = ['tags', 'counts']
  tags_list = tags_list.sort_values(by='counts', ascending=False)

  plt.barh(list(tags_list['tags'])[:25], list(tags_list['counts'])[:25])
  plt.ylabel('Tags -->')
  plt.xlabel('Count -->')
  plt.title('25 Most Popular Tags')
  plt.show()


#How many tags are present per question?
def ip_6(df=None):
    print("\nHow many tags are present per question?")
    tags = list(df['tags'].unique())
    num_tags = []

    for tag_ in tqdm_notebook(tags):
        num_tags.append(len(tag_.split(' ')))

    num_tags = Counter(num_tags)

    plt.bar(list(num_tags.keys()), list(num_tags.values()))
    plt.xlabel('Number of tags -->')
    plt.ylabel('Number of questions -->')
    plt.title('Number of Tags per question')
    plt.show()


#Are the number of views of question statistically related to each other?
def ip_7(df=None):
  print("\nAre the number of views of question statistically related to number of answers?")
  view_q = df[['views', 'num_answers']].drop_duplicates().reset_index(drop=True)
  display(view_q.astype(int).describe())
```

```
#Are the tags related to Views?
def ip_8(df=None):
  print("\nAre the tags related to Views?")
  tag_views = df[['tags', 'views']].drop_duplicates().reset_index(drop=True)
  tagView = {}
  for i in tqdm_notebook(range(len(tag_views['tags']))):
      for tag in tag_views['tags'][i].split(' '):
          try:
              tagView[tag].append(tag_views['views'][i])
          except Exception:
              tagView[tag] = [tag_views['views'][i]]
  tagViewAvg = {}
  for key in list(tagView.keys()):
      tagViewAvg[key] = np.average(tagView[key])

  tagViewAvg = pd.DataFrame([list(tagViewAvg.keys()), list(tagViewAvg.values())])
  tagViewAvg = tagViewAvg.transpose()
  tagViewAvg.columns = ['tags', 'views']
  tagViewAvg = tagViewAvg.sort_values('views')

  plt.barh(list(tagViewAvg['tags'])[-25:], list(tagViewAvg['views'])[-25:])
  plt.xlabel('Average number of Views -->')
  plt.ylabel('Tags -->')
  plt.title('25 - Most Popular Tags')
  plt.show()

  fig, ax = plt.subplots(2, 2)

  ax[0][0].plot([0] + tagView['python'])
  ax[0][0].set_title('python')

  ax[0][1].plot([0] + tagView['c'])
  ax[0][1].set_title('c')

  ax[1][0].plot([0] + tagView['java'])
  ax[1][0].set_title('java')

  ax[1][1].plot([0] + tagView['c#'])
  ax[1][1].set_title('c#')
```

```
  plt.show()


#Analyse Votes, Number of Answers and Views
def ip_9(df=None):
  print("\nAnalyse Votes, Number of Answers, Views")
  print("\nPair plot: ")
  sns.pairplot(df)
  plt.show()
  print("\nCorrelation plot: ")
  sns.heatmap(df.corr())
  plt.show()
```

Appendix III: User Frontend

```
print("**********************************************************************
*** StackOverflow Data Analysis
**********************************************************************
**********************")

TAG = input("Hello!\nOn which domain would you like to perform the analysis?\nNote:
Enter domain names without blank space between them.\nFor example: Enter machine
learning as machine-learning\n")
filter = input("\nWhich filter would you like to apply?\n1.Newest\n2.Recent
activity\n3.Most votes\n4.Most frequent\n5.Bounty ending soon\n")
FILTER = filter_sort(filter)
print("\nCollecting data..")
df = scrape_tag(TAG,FILTER)
print("Data collection successful.")

ip = 0
while ip != '10':

  ip = input("\n\n\nWhich analysis would you like to perform?"
        "\n1.Analyse Votes"
        "\n2.Analyse Number of Answers"
        "\n3.Analyse Views"
        "\n4.Are the tags mentioned in questions similar to each other?"
```

```
        "\n5.Which tags are popularly used by Stackoverflow community in this domain?"
        "\n6.How many tags are present per question?"
        "\n7.Are the number of views of question statistically related to number of
answers?"
        "\n8.Are the tags related to Views?"
        "\n9.Analyse Votes, Number of Answers, Views"
        "\n10.Exit\n")

 if ip == '1':
   ip_1(df)

 elif ip == '2':
   ip_2(df)

 elif ip == '3':
   ip_3(df)

 elif ip == '4':
   ip_4(df)

 elif ip == '5':
   ip_5(df)

 elif ip == '6':
   ip_6(df)

 elif ip == '7':
   ip_7(df)

 elif ip == '8':
   ip_8(df)

 elif ip == '9':
   ip_9(df)
```

References

[1]Kabita Sahoo, Abhaya Kumar Samal, Jitendra Pramanik, Subhendu Kumar Pani (2019) Exploratory Data Analysis using Python, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, Issue-12

[2] Kiranbala Nongthombam, Deepika Sharma, Data Analysis using Python, (2021), International Journal of Engineering Research & Technology (IJERT) http://www.ijert.org ISSN: 2278-0181 IJERTV10IS070241 (This work is licensed under a Creative Commons Attribution 4.0 International License.) Published by : www.ijert.org Vol. 10 Issue 07

[3]Bho Zao, 2017, Web Scraping, Encyclopedia of Big Data (pp.1-3), Publisher: Springer International Publishing, Editors: Laurie A. Schintler, Connie L. McNeely

[4] Sandeep Mathur, David Mathew Thomas, 2019, . Data Analysis by Web Scraping using Python, 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)

[5] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, Jure Leskovec, Discovering Value from
Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow, Standford university, 2012

[6] J. Guo, S. Xu, S. Bao, and Y. Yu, tapping on the Potential of Q&A Community by Recommending
Answer Providers, 8th ACM conference on Recommender systems, 2014

[7] Jun Zhang, Mark S. Ackerman, Lada Adamic, Expertise Networks in Online Communities: Structure and Algorithms, Track: E*-Applications, 2007

[8] Jose San Pedro, Alexandros Karatzoglou, Question Recommendation for Collaborative Question
Answering Systems with RankSLDA, RecSys, 2014

[9] Kenneth Reitz: requests-HTML v0.3.4 documentation (python-requests.org)

[10]Saini: https://jovian.ai/saini-9/research-papers-web-scraping

Acknowledgement

It gives us immense pleasure and satisfaction in presenting this mini project **'Data Analytics on Stackoverflow data'**.

We would like to express our deep sense of gratitude towards **Dr.Uma Sheshadri** for her continuous guidance and assistance.

We also wish to express our heartfelt gratitude to all those who have played a crucial role in the research for this project, without their active cooperation the preparation of this project could not have been completed within the specified time limit.

Kummara Bhargavi (18BEC024)

Mohammed Safwan (18BEC028)

Parvati Jayakumar (18BEC036)

P. Chethan Krishna (18BEC040)