PARAMA: PREDICTIVE ANALYTICAL MODEL FOR ACADEMIC AND

RESEARCH ACTIVITIES

**Abstract**: We aim to automate the management of research materials, including publications like papers, patents and journals from our college faculty. This will be achieved through the integration of technologies such as HTML, CSS, JavaScript, ReactJS, MongoDB, NodeJS, and Express JS. Our application offers effortless exploration of research papers relevant to the intended topic of study. Users can seamlessly search for pertinent materials using keywords, enhancing the research process. We use Natural Language Processing and Machine Learning Algorithms for retrieving the records from the database based on user search.

**I.Introduction**

The academic world is characterised by a never-ending search for knowledge, and at the centre of this endeavour is a wealth of research resources, including articles, patents, books, and journals. Academic development is fundamentally driven by college instructors and researchers' ability to efficiently handle and access this plethora of knowledge. It is not just an issue of convenience. Seeing this need, we set out to create a state-of-the-art application that will revolutionise the handling and availability of research resources.

In a time when technology permeates every part of our existence, we must take advantage of its potential to efficiently automate the management of research materials. Modern technologies like HTML, CSS, JavaScript, ReactJS, MongoDB, and ExpressJS will be integrated to make this project a reality. These technologies will collectively serve as the foundation for our creative solution. Through the simplification of the process of organising, finding, and retrieving materials, this application aims to empower researchers and faculty members.

A strong search function is one of our application's main features. With the use of pertinent terms that are directly related to their job, users will be able to search for research papers and journals with this functionality. The days of searching through innumerable documents or having trouble locating particular study resources in a jumbled digital environment are long gone. Rather, customers may easily peruse an extensive library of scholarly materials, greatly improving the effectiveness of their research procedure.

Our application provides an advanced search option in addition to keyword-based search. With the use of this function, individuals can look up research articles by using the information contained in their own study abstracts. Finding resources that support a user's research goals adds a personalised touch to the academic process and increases focus and productivity.

Furthermore, by utilising data analytics, our application seeks to provide a comprehensive perspective on faculty research activities. Comprehending the academic research environment is essential for well-informed decisionmaking, resource distribution, and long-term planning. Our system will generate a culture of research excellence by gathering and evaluating data on research activities, giving important insights into the faculty's research areas of strength and improvement.

Our solution's core leverages the power of Natural Language Processing (NLP) and Machine Learning (ML) algorithms, going beyond simple technological integration. These state-of-the-art tools are essential for efficiently retrieving research records from our database. The system is able to suggest and rank items that most closely match the user's demands since the algorithms are able to understand the context and content of each document. All things considered, this technology-based strategy greatly improves both the user experience and the calibre of search results.

In summary, our project to create an automated research material management programme sits at the nexus of academics and technology. It has the potential to completely change how academic resources are accessed and used by college professors and scholars. Our approach, which combines HTML, CSS, JavaScript, ReactJS, MongoDB, ExpressJS, NLP, and ML, has the potential to significantly increase research efficiency and effectiveness by making important academic materials easily accessible.

**II.LITERATURE REVIEW**

**RONAN R. K. ANDO AND T. ZHANG [1]** The paper "Natural Language Processing (almost) from Scratch" authored by Ronan R. K. Ando and T. Zhang and published in the Journal of Machine Learning Research (JMLR) in November 2005, is a pivotal work that has significantly shaped the field of Natural Language Processing (NLP). This research introduces a groundbreaking framework that enables the development of NLP systems with minimal reliance on prior knowledge or extensive feature engineering. One of the key innovations in this paper is its emphasis on leveraging unlabeled data, a fundamental concept in unsupervised learning, to construct NLP models. This approach revolutionizes the conventional paradigm by allowing the system to draw knowledge from vast pools of unannotated text data, reducing the need for manual feature engineering and human domain expertise. Furthermore, the paper introduces the concept of transfer learning to NLP, demonstrating how knowledge acquired from one task can be effectively transferred to another related task. This idea has profound implications for NLP, as it enables the pre-training of models on large, general-purpose datasets and fine-tuning them for specific NLP tasks, resulting in significant performance improvements. The paper's contributions extend well beyond its initial publication, as the framework and ideas it presents have inspired subsequent research, leading to the development of highly effective NLP models and systems .Published in the esteemed JMLR, this paper has had a lasting impact on the machine learning and NLP communities, serving as a cornerstone for research and development in the field. Its focus on reducing the barriers to entry for NLP system development and improving the use of unllabeled data and transfer learning has paved the way for the development of more sophisticated and accessible NLP solutions. goal of the Intrusion Detection System (IDS) is to automate the process of filtering and blocking abnormal traffic into our network. The anomaly detection process is considered as one of the NP-Hard problems. The process of discriminating the network traffic as good traffic or bad traffic is crucial because the detection system must be equipped with suitable knowledge with the help of a knowledge base. Unfortunately, the research contributions published so far, on anomaly detection are still incomplete because of the continuous challenges coining from rapid technological changes. Section 2.2 of this chapter presents research contributions in the literature that motivated the present study, to carry out this research. Section 2.3 then, presents different machine learning techniques such as, supervised and unsupervised that are employed in the design of intrusion detection systems and pays attention to different measures that are employed. Section 2.4 of this chapter outlines different similarity measures used in the existing research literature for the identification of anomalies.

**Ben Wellner [2]** The literature survey titled "Automatically Identifying the Arguments in Discourse Connectives" by Ben Wellner, affiliated with The Mitre Corporation in Bedford, MA, USA, addresses the challenge of automatically identifying the arguments of discourse connectives in the Penn Discourse Tree Bank (PDTB). Rather than attempting to identify the full extents of these arguments as annotated in the PDTB, the paper reframes the problem by focusing on identifying the argument heads, effectively circumventing the complexity of discourse segmentation. The authors showcase substantial improvements in their approach by incorporating features extracted from a dependency parse representation, which outperforms those derived from a constituent-based tree parse. Additionally, the paper highlights the significance of capturing inter-argument dependencies through a log-linear re-ranking model, achieving a high accuracy rate in correctly identifying both arguments for over 74% of the connectives on held-out test data with gold-standard parses. This literature survey contributes to the field of computational linguistics and discourse analysis by presenting an innovative approach to handling the challenge of identifying arguments within discourse connectives. The paper's emphasis on dependency parsing and its effectiveness in improving accuracy is noteworthy. By sidestepping the intricacies of discourse segmentation, this work simplifies and enhances the identification of discourse connective arguments. It offers valuable insights into the practical applications of natural language processing and discourse analysis in various domains, including machine learning and language understanding. The authors' approach, which combines the benefits of dependency parsing and a log-linear re-ranking model, presents a promising avenue for future research in discourse analysis and automated text understanding. This paper's findings and methodologies have the potential to impact the development of more sophisticated and accurate NLP systems and language understanding tools.

**Hinton [3]** The abstract describes a methodology that leverages "complementary priors" to address challenges in densely connected belief networks with multiple hidden layers. By using these priors, the paper introduces a fast and efficient algorithm for training deep, directed belief networks one layer at a time. This method is particularly effective when the top two layers of the network create an undirected associative memory. The fast, greedy algorithm serves as an initialization step for a slower learning process that refines the network's weights using a contrastive version of the wake-sleep algorithm.

The outcomes of this research are noteworthy. After the fine-tuning process, a network with three hidden layers can effectively model the joint distribution of handwritten digit images and their corresponding labels, serving as a powerful generative model. This generative model surpasses the performance of discriminative learning algorithms in digit classification tasks, highlighting the potential of generative models. The abstract also suggests that the network can represent low-dimensional patterns, akin to "ravines" in a free-energy landscape, within the top-level associative memory. This representation allows for the exploration of the patterns and insights contained in the associative memory, offering a unique perspective on the information encoded in the network.

**III.Existing System**

The existing research material management system lacked accurate keyword search functionality, leading to inefficiencies as users struggled to locate relevant materials. Additionally, the absence of data visualizations like statistical analysis hindered users' ability to derive meaningful insights from research trends. In response, our initiative aims to revolutionize the system by leveraging advanced technologies such as Natural Language Processing and Machine Learning algorithms. Through these innovations, users can expect improved search accuracy, minimizing time spent on manual searches. Moreover, our system will introduce robust data visualization features, enabling users to glean valuable insights from research trends and patterns through intuitive visualizations. By addressing these deficiencies head-on, we seek to enhance the efficiency, effectiveness, and user experience of research material management within the college faculty, ultimately facilitating informed decision-making and driving meaningful discoveries.

**IV.Proposed System**

Users can search papers aligned with their research by inputting their abstracts, enhancing relevancy and efficiency. Our app offers advanced search, enabling easy discovery of research papers and journals relevant to user's work via keyword-based searches. In addition, users can utilize their research abstracts to conduct searches for research papers that are closely related to their own areas of investigation.

**4.1 Objective of the proposed system:**

1. User will able to upload publication file into database through CSV format.
2. User can retrieve data belongs to existing publications.
3. User can retrieve research papers by applying filters on publishing date , keywords , authors names , title , publication type and indexing

* 1. **Data Collection:**

Data collection is the first and crucial in building the model. We collected different faculty of various departments details to complete our data set. It consists of various columns in which various information about the publications is mentioned consisting of name, author, Citations, year of publication, links to be redirected, abstract, keywords, etc. Data can be classified into different types of publications, including patents, journals, book chapters, copyrighted works, and conference papers. This classified data also consists of similar columns as of the initial data. The is completely filled and modified according to the technology we used. We filled abstract and keywords which are important for searching the publications in the data.

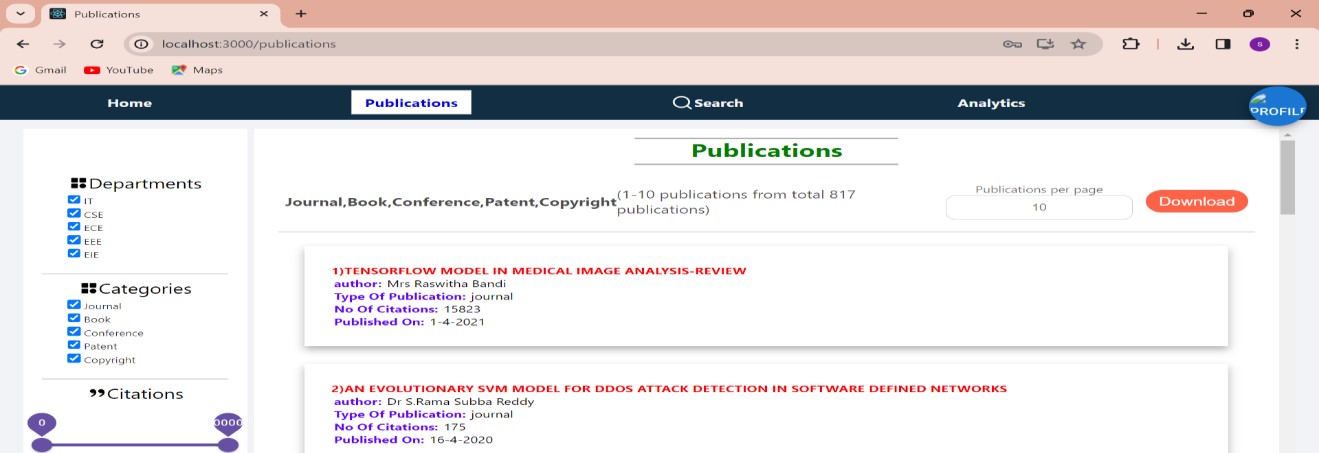
* 1. **Data Deployment:**

The express JS server is created with an API for deploying the data, when the API is called the CSV file (publication details) is sent in the request. At the backend each row of the CSV is converted into a Json object and all the Json objects are inserted into the array. The array is deployed onto mongo DB using mongoose module.

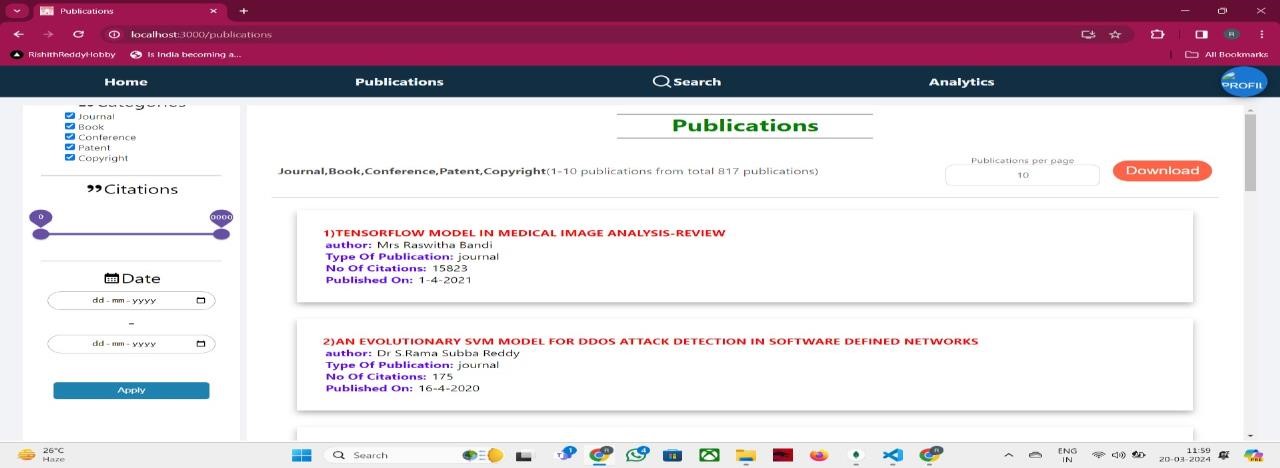
* 1. **Filter API:**

The API is created for the data retrieval. Data is retrieved based on the keywords, citations, date and type of publication (book chapter, patent,..). When the API for the data retrieval is called data for applying filters will be sent as query parameter with URL. Query parameters are used in the query (find query) for data on database. The retrieved data from the database is sent as a response from the server.

In the UI by default all the publications are shown when the user applies filter a request will be sent to the backend server as mentioned above and the data came as a response from the server is displayed in the UI.



The above figure illustrates the total number of publications. Users have the option to download these publications.



The above figure illustrates the total number of publications. Users have the option to download these publications.

* + 1. **Date filter:**

The published date of publication is filled in the format of ‘yyyy-mm-dd’ in the publication details csv file. In the backend while deploying data , a JS Date object is created using published date later the getTime function is called on that object which returns number of milli seconds (from 1st jan 1970 till the published date). This milliseconds is stored in the database.

When the user applies filter on date (from date – to date) as mentioned above the milliseconds are calculated for the from date and to date, these milliseconds calculated are compared to the milliseconds stored in the database (>=(from date milliseconds) and <=(to date milliseconds)). The date is retrieved based on this process.

* + 1. **Search:**

When the user searches in the search box the query is compared with different fields in the publication details data deployed in the database. Atlas search is used for search operation, which is service provided by MongoDB. The basic working of search operation is mentioned below.

Atlas Search is a full-text search service provided by MongoDB, designed to perform advanced search operations on data stored in MongoDB databases. Now lets see the process step by step

1. **Index Creation:**

The first step in using Atlas Search is to create a search index on the collection that contains the data you want to search. This index defines which fields to index and how to analyze them during search operations. For my website we created index over keywords,abstract and title fields

1. **Data Ingestion:**

After creating indexes as mentioned above we deployed around 820 documents on to the database.

1. **Query Parsing:**

When user search something using search feature In the UI that search will be sent as query using $search query operator for using the atlas search to the database from backend server,

At the MongoDB server Search parses the query to understand the user's intent and requirements. This parsing involves breaking down the query into tokens, identifying keywords, and potentially extracting entities or features from the query.

1. **NLP Processing:**

Natural Language Processing (NLP) algorithms come into play here to enhance the search experience. These algorithms include techniques such as tokenization, Stemming, part-of-speech tagging, named entity recognition (NER), sentiment analysis, and semantic analysis.

* Tokenization: Breaking down the query into individual words or tokens. (A sentence is breaked into words based on space)
* Stemming:In Search, stemming is a crucial part of the text analysis phase, which is essential for improving search accuracy by reducing words to their root or base form. Our search primarily uses the Snowball stemming algorithm, specifically designed for various languages. Let's dive into the Snowball stemming algorithm and its clear explanation:

**Snowball Stemming Algorithm:**

Snowball stemming algorithm, also known as the Porter2 stemming algorithm, is a widely-used and effective approach for stemming in natural language processing. It's designed to handle stemming for multiple languages, making it versatile for Atlas Search's diverse user base.

**Explanation of the Snowball Stemming Algorithm:**

**1. Initialization:**

* Snowball algorithm initializes with the original word.
* It sets up a pointer to the last character of the word.

**2. Stemming Rules:**

* Snowball algorithm applies a set of predefined rules to manipulate the word's suffixes and prefixes to find the root form.
* These rules are carefully crafted based on linguistic principles and common word variations observed in the language.

**3. Rule Application:**

* Snowball applies rules sequentially until a matching rule is found or until it exhausts the list of rules.
* If a rule matches, it modifies the word according to the rule's instructions.

**4. Example:**

* + Let's consider the word "running".
  + Snowball algorithm applies rules iteratively:
    - Rule 1: If the word ends with "ing", remove "ing". (Result: "runn")
    - Rule 2: No match.
    - Rule 3: No match.
  + Snowball stops as there's no more applicable rule.
  + The resulting stem is "runn".

**Clear Explanation:**

- **Incremental Rule-Based Approach:** Snowball stems words incrementally by applying a series of rules. Each rule targets specific suffixes or prefixes commonly found in words.

- **Language-Awareness:** Snowball algorithm is designed with language-specific rules, ensuring that stemming is sensitive to linguistic nuances and irregularities of different languages.

- **Rule Prioritization:** Rules are prioritized based on their effectiveness in reducing words to their root forms while minimizing over-stemming (reducing words excessively) and under-stemming (insufficient reduction).

- **Rule Composition:** Snowball stems words by composing simple, atomic rules into more complex transformations. This allows for a comprehensive coverage of various word variations.

- **Performance Considerations:** Snowball algorithm is optimized for performance, balancing between accuracy and computational efficiency to provide fast and reliable stemming for text processing tasks.

Overall, the Snowball stemming algorithm used in Search leverages a combination of linguistic rules and computational efficiency to produce accurate stem forms of words, enhancing the search capabilities.

* Stop word removal:Stop word removal is a fundamental step in text analysis, aiming to filter out common words that often occur but carry little semantic meaning, such as "the", "and", "is", etc. In Atlas Search, stop word removal is typically implemented using predefined lists of stop words for various languages. Here's a clear explanation of the stop word removal algorithm:

**Stop Word Removal Algorithm:**

**1. Initialization:**

* The algorithm begins with a piece of text or a document containing words.
* It also has access to a list of predefined stop words specific to the language being processed.

**2. Tokenization:**

* The input text is tokenized into individual words or tokens. This process separates the text into meaningful units, typically based on whitespace or punctuation.

**3. Stop Word Detection:**

* For each tokenized word, the algorithm checks whether it matches any word in the stop word list.
* Stop word detection is usually case-insensitive to ensure that words in different cases (e.g., "The" and "the") are both filtered out.

**4. Filtering:**

* If a token matches a stop word in the list, it is removed from further processing.
* Otherwise, if the token does not match any stop words, it is retained for subsequent analysis and indexing.

**5. Output:**

* The output of the stop word removal algorithm is a filtered list of words, excluding those identified as stop words.
* This filtered list forms the basis for subsequent text analysis steps, such as stemming, indexing, and search operations.

**Clear Explanation:**

- **Predefined Stop Word Lists:** Atlas Search utilizes predefined lists of stop words for different languages. These lists are curated based on common usage patterns and linguistic considerations.

- **Efficient Lookup:** Stop word removal involves efficient lookup operations to quickly determine whether a word matches any stop words in the predefined list.

- **Language Sensitivity:** Stop word removal is sensitive to the language of the text being processed. Different languages may have different sets of stop words, reflecting their unique grammatical structures and usage conventions.

- **Customization:** While Atlas Search provides predefined stop word lists, users may have the flexibility to customize or augment these lists based on their specific domain or application requirements.

- **Impact on Search Relevance:** By removing stop words, the algorithm improves the relevance of search results by focusing on content-bearing words that carry more semantic meaning. This helps to prioritize important keywords during search operations.

Overall, the stop word removal algorithm in Atlas Search efficiently filters out common and non-informative words from textual data, enhancing the quality and relevance of search results for MongoDB databases.

**5. Query Execution:**

After parsing and processing the query, Atlas Search executes the search operation using the index created earlier. It matches the query against the indexed fields and documents in the collection.

Atlas Search utilizes various search algorithms and techniques, including inverted indexes, TF-IDF (Term FrequencyInverse Document Frequency) algorithm.

**TF-IDF (Term Frequency-Inverse Document Frequency) algorithm:**

The Term Frequency-Inverse Document Frequency (TF-IDF) algorithm is a widely used technique in information retrieval and text mining for evaluating the importance of a term within a document relative to a collection of documents. TF-IDF is commonly used in search engines, including Atlas Search, to rank documents based on their relevance to a given query. Let's break down the TF-IDF algorithm:

1. **Term Frequency (TF):**

Term Frequency measures how frequently a term appears in a document. It indicates the importance of a term within the document.

Term Frequency for a term t in a document d is calculated as the ratio of the number of times term t appears in the document to the total number of terms in the document.

Mathematically, TF is calculated as:

TF(*t*,*d*)=Total number of terms in document *d/*Number of times term *t* appears in document *d*

1. **Inverse Document Frequency (IDF):**

* Inverse Document Frequency measures the importance of a term across a collection of documents. It helps to identify rare terms that may have more discriminatory power.
* IDF for a term t is calculated as the logarithm of the ratio of the total number of documents in the collection to the number of documents containing the term t.

Mathematically, IDF is calculated as:

IDF(*t*)=log(Number of documents containing term *t/*Total number of documents)

**3. TF-IDF Weighting:**

* TF-IDF is calculated by multiplying the Term Frequency (TF) of a term in a document by its Inverse Document Frequency (IDF) across the entire collection.
* The resulting TF-IDF score reflects the importance of a term within a document relative to its importance across the entire collection.

Mathematically, TF-IDF is calculated as: TF-IDF(*t*,*d*,*D*)=TF(*t*,*d*)×IDF(*t*) where D represents the collection of documents.

**4. Ranking Documents:**

* Once the TF-IDF scores are calculated for all terms in a document, documents are ranked based on their overall TF-IDF scores for the query terms.
* Documents with higher TF-IDF scores are considered more relevant to the query.

**6. Presentation of Results:**

Finally, the results are returned by MongoDB server to the website backend server and backend server will send the results to frontend where result publicaions will be displayed.

**Case study of search:**

**1.User Input:**

* The user enters the search text "machine learning algorithms" into the search bar on the website's UI.

**2.Query Processing:**

* The UI sends a request to the backend server with the search text.

**3. Query Construction:**

* The backend server receives the search text "machine learning algorithms" and constructs a search query to send to MongoDB Atlas Search.

**4. Text Analysis:**

* MongoDB Atlas Search tokenizes the search text into individual words: "machine", "learning", "algorithms".
* The text analyzer applies stemming to reduce words to their root forms:
* "machin", "learn", "algorithm".
* Stop-word removal is performed to filter out common and non-informative words:
* "machin", "learn", "algorithm".

**5. Index Lookup:**

* MongoDB Atlas Search looks up the indexed fields (e.g., `Title`, `Abstract`, `Keywords`) to find documents containing the search terms.

**6. Matching Documents:**

* Documents containing any of the search terms ("machin", "learn", "algorithm") in the indexed fields are identified.
* Let's say the search returns the following documents:
* Document 1: Title - "Introduction to Machine Learning", Abstract - "This paper introduces various machine learning algorithms..."
* Document 2: Title - "Advanced Algorithms for Machine Learning", Abstract - "This paper explores advanced algorithms used in machine learning..."

**7. Scoring and Ranking:**

* MongoDB Atlas Search scores the matching documents based on relevance to the search terms.
* Documents with higher relevance scores are ranked higher in the search results.
* For example, Document 2 might have a higher relevance score because it contains all three search terms in its title.

**8. Search Result Retrieval:**

* MongoDB Atlas Search retrieves the top-ranked documents that match the search query.
* In this case, both Document 1 and Document 2 are retrieved as search results.

**9. Result Presentation:**

* The backend server formats the search results and sends them back to the UI for display.
* The UI presents the search results to the user, showing relevant metadata such as title, authors, and publication date.
* Additionally, the sentiment analysis results (e.g., neutral sentiment) can be optionally displayed to the user, providing insights into the context of the search query.

**10. User Interaction:**

* The user interacts with the search results, clicking on individual items to view more details or continuing to refine the search query.

**4.5 Analytics:**

When home page is loaded a request is sent to the backend server and server sends three arrays as response to the frontend in which one array consists of total count of publications, sub counts of each type of publication (journal, patent, book chapter, conference, copyright), The second array consists of year wise publications count in the past 15 years and the last array consists of sub counts of each type of publications of every year for the past 10 years. Then we presented the data as follows using LineChart, PieChart, BarChart components of material UI.

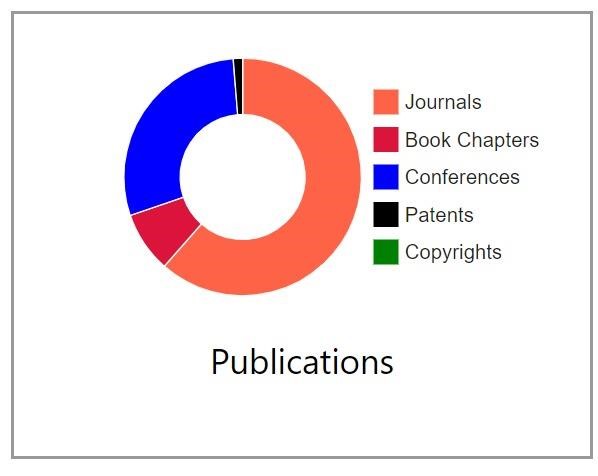
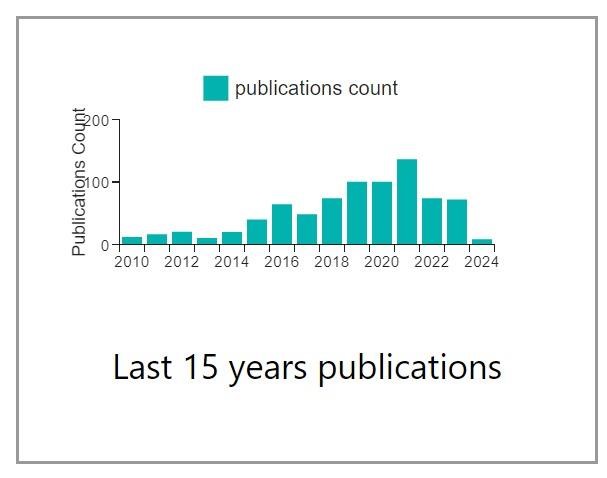


Fig: 1 Fig: 2

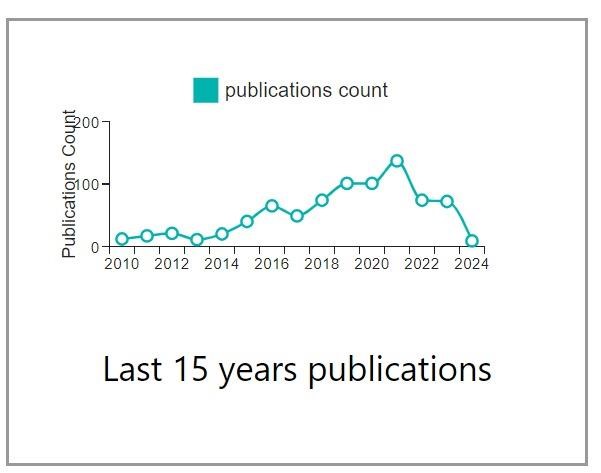


Fig: 4

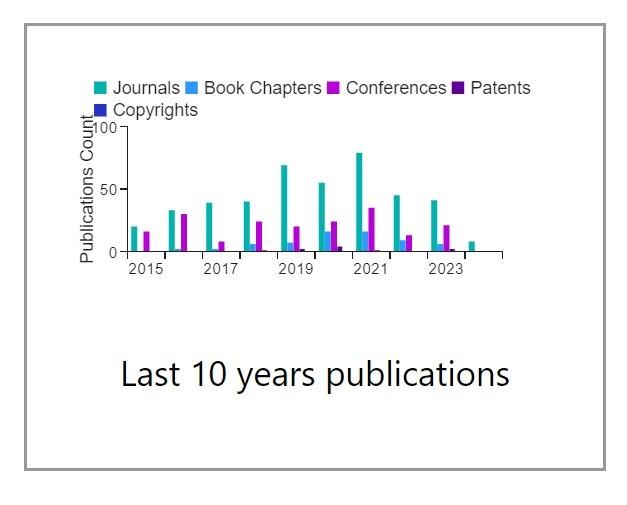


Fig: 3

Figure 1 Illustrates the overall total number of publications using pie charts, figure 2 displays last 15 analytics based on total count of publications, figure 3 years shows 10 years graph with respective to the type of category of publication and figure 4 shows the publication count for the last 15 years line graph respectively.

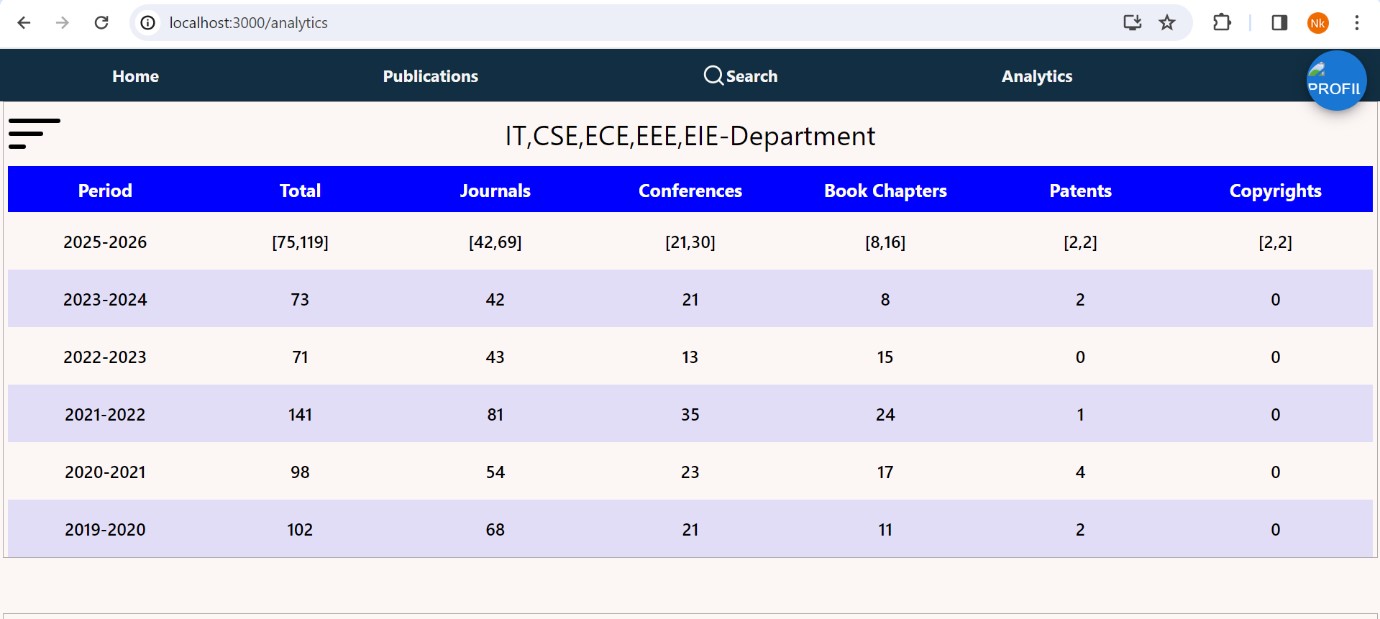


Fig: 5

Using date filter as mentioned above publications count are calculated for each year individually and stored in a object for each year. And the array of above objects is sent as response to frontend by the backend server, then the data is displayed in the UI as in figure 5.

**4.6 Prediction:**

We used machine learning algorithms (linear Regression, XG Boost) to predict the future results of how many publications in total for the future year and individual number of publications categorized into patents, Journals, copy rights, book chapters, conferences. The prediction is shown in a range of numbers in form of minimum number of publications possible to the maximum number of publications possible.

We processed the data in the database to get the required data to train a machine learning model to predict number of possible publications in the following year and the generated data is in the csv format as shown below in figure 6.

**4.6.1 Data for the Training Models:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **periodS** | **publications** | **journals** | **books** | **conferences** | **patents** | **copyrights** |
| 2023 | 73 | 42 | 8 | 21 | 2 | 0 |
| 2022 | 71 | 43 | 15 | 13 | 0 | 0 |
| 2021 | 141 | 81 | 24 | 35 | 1 | 0 |
| 2020 | 98 | 54 | 17 | 23 | 4 | 0 |
| 2019 | 102 | 68 | 11 | 21 | 2 | 0 |
| 2018 | 71 | 38 | 8 | 24 | 1 | 0 |
| 2017 | 52 | 42 | 2 | 8 | 0 | 0 |
| 2016 | 66 | 34 | 2 | 30 | 0 | 0 |
| 2015 | 39 | 19 | 5 | 15 | 0 | 0 |
| 2014 | 20 | 11 | 0 | 9 | 0 | 0 |
| 2013 | 9 | 4 | 0 | 5 | 0 | 0 |
| 2012 | 18 | 10 | 0 | 8 | 0 | 0 |
| 2011 | 17 | 11 | 2 | 4 | 0 | 0 |
| 2010 | 12 | 7 | 0 | 5 | 0 | 0 |
| 2009 | 2 | 0 | 0 | 2 | 0 | 0 |
| 2008 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2007 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2006 | 2 | 0 | 0 | 2 | 0 | 0 |

**4.6.2 Model Training:**

Since, there is very limited data we have tried using different algorithms for better predictions. The algorithms used are described below.

1. **Linear Regression:**

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data The goal is to minimize the difference between the observed value of the dependent variable on the predicted line. It assumes continuous relationships between variables and is commonly used for forecasting and statistical purposes in fields such as finance, economics, and the social sciences.

There's only one independent variable x in a simple linear regression model, the prediction is calculated as follows:

y = beta\_0 + beta\_1 x

Where:

-y is the predicted value of the dependent variable,

- beta\_0 is the intercept term,

- beta\_1 is the coefficient corresponding to the independent variable x.

The model computes the predicted value y by multiplying the value of the independent variable x by its coefficient beta\_1, and then adding the intercept term beta\_0.

1. **XG Boost:**

XG Boost (Extreme Gradient Boosting) is a powerful machine learning framework that uses gradient boosting algorithms. It in turn constructs multiple decision trees to minimize loss functions, and makes predictions by combining the results of these trees. It performs well, tends to outperform other algorithms in terms of prediction accuracy, and is widely used in competitive and real-world applications in a variety of industries.

XGBoost makes predictions by combining the outputs of multiple decision trees, which are constructed sequentially to correct the errors of the previous trees. Here's a simplified overview of how XGBoost works for prediction:

* 1. Initial Prediction: XGBoost starts with an initial prediction, typically the mean of the target variable for regression or the majority class for classification.
  2. Sequential Tree Construction: It builds decision trees one at a time, where each subsequent tree focuses on reducing the errors (residuals) made by the previous trees.
  3. Gradient Boosting: At each iteration, XGBoost calculates the gradient of the loss function with respect to the predicted values from the previous iteration. This gradient represents the direction and magnitude of the error for each data point.
  4. Tree Construction: XGBoost constructs a decision tree to predict the gradient values obtained in the previous step. This tree is trained to minimize the loss function using the gradient information.
  5. Tree Combination: The predictions from all the trees are combined to produce the final prediction. In regression, this might involve averaging the predictions of all trees, while in classification, it may involve a voting scheme.
  6. Regularization: XGBoost applies regularization techniques to prevent overfitting during tree construction, such as maximum depth limits, minimum child weight, and subsampling of data points and features.
  7. Prediction: Finally, to make a prediction for a new data point, XGBoost applies the sequence of decision trees to the features of the new data point and combines their predictions according to the model parameters.

Overall, XGBoost's prediction process utilizes an ensemble of decision trees trained in a gradient boosting framework, allowing it to capture complex patterns in the data and achieve high predictive accuracy.

1. **Support Vector Machine:**

In machine learning, support-vector machines, also known as support-vector networks are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. For aiming to obtain a higher accuracy, and to deal with the large number of attributes that contribute to the output, the Support Vector Machine algorithm has been used with a polynomial kernel.

1. **Random Forest:**

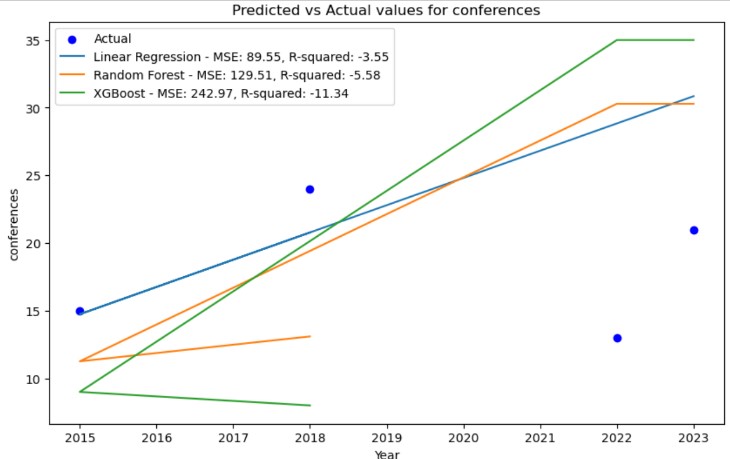
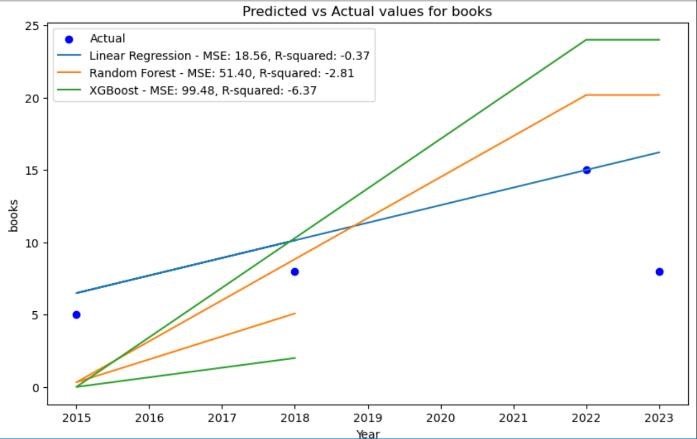
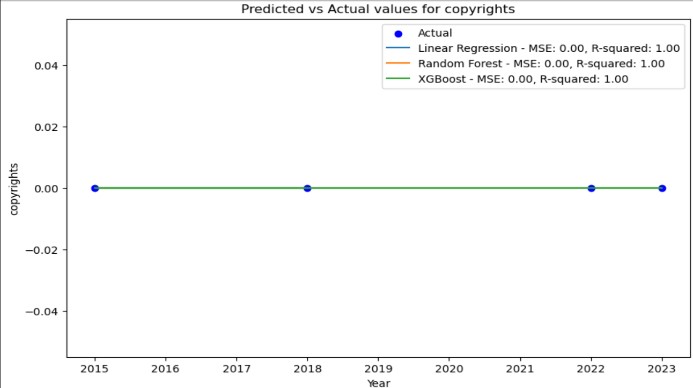
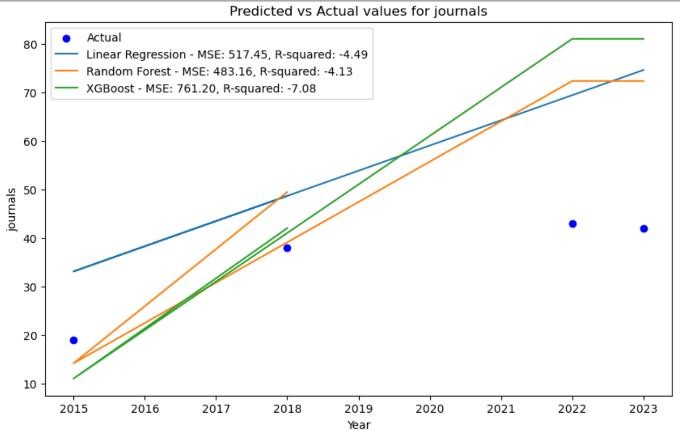
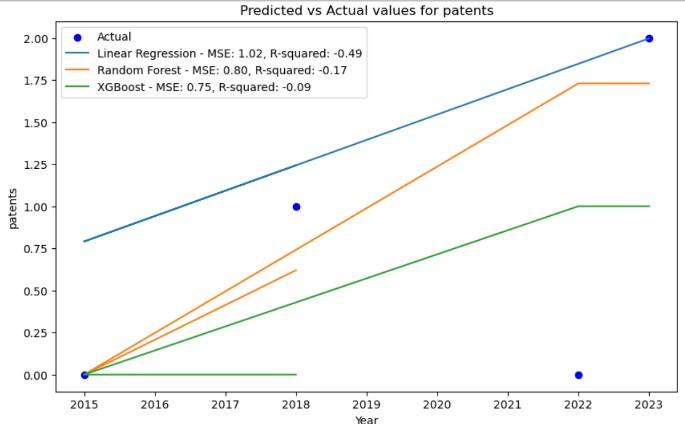
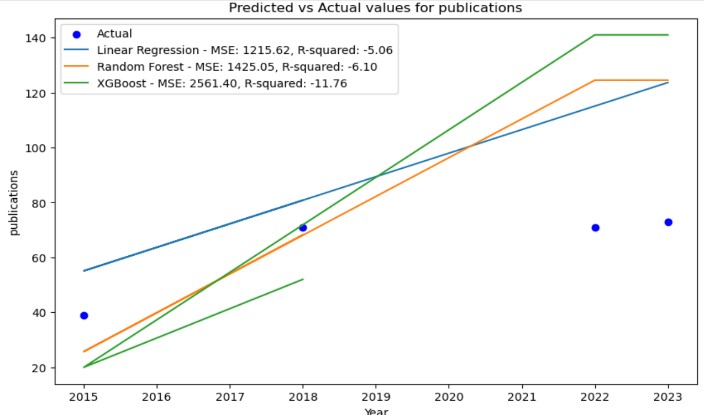
It is an ensemble learning technique that helps in classification and regression problems. It is a very useful algorithm for handling large datasets, as in this case. It integrates various decision trees to produce a model that is more reliable and accurate. A random subset of the input features and a subset of the training data are used to train each decision tree in the random forest. To provide a final prediction, the Random Forest Classifier builds a lot of decision trees and then combines their predictions. A random subset of the training data is used to train each decision tree in the forest, a process known as bagging or bootstrap aggregation.

1. **Logistic Regression:**

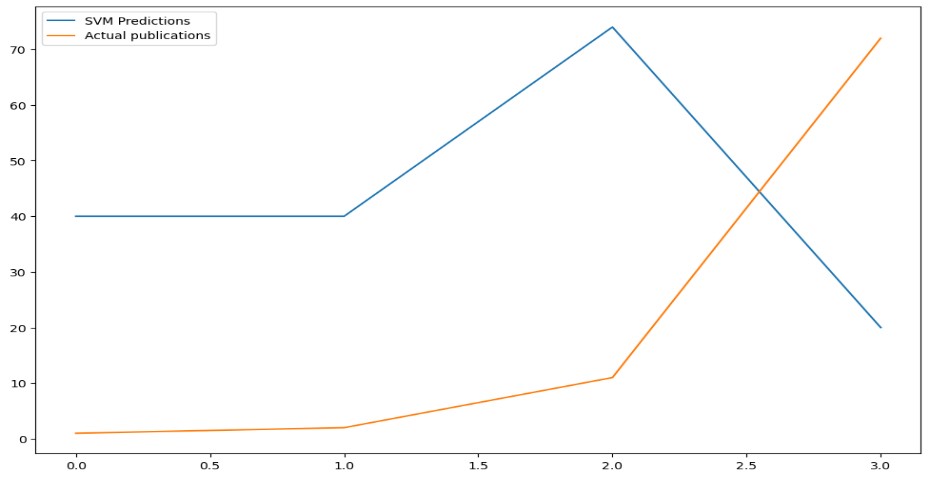
For binary classification issues, where the objective is to predict one of two possible outcomes, logistic regression is a statistical machine learning approach that is utilized. In many different applications, including fraud detection, picture classification, and medical diagnosis, it is frequently employed. Here, the target variable is a binary value, hence logistic regression can be well-suited for the dataset being used.

The graphs shown below shows the actual vs predicted values In which three algorithms are implemented together with mean squared error and R-squared value. The blue line shows Linear Regression Graph, Orange line shows the results of Random Forest, Green line shows the XG Boost graph finally the blue dots indicate the actual results.

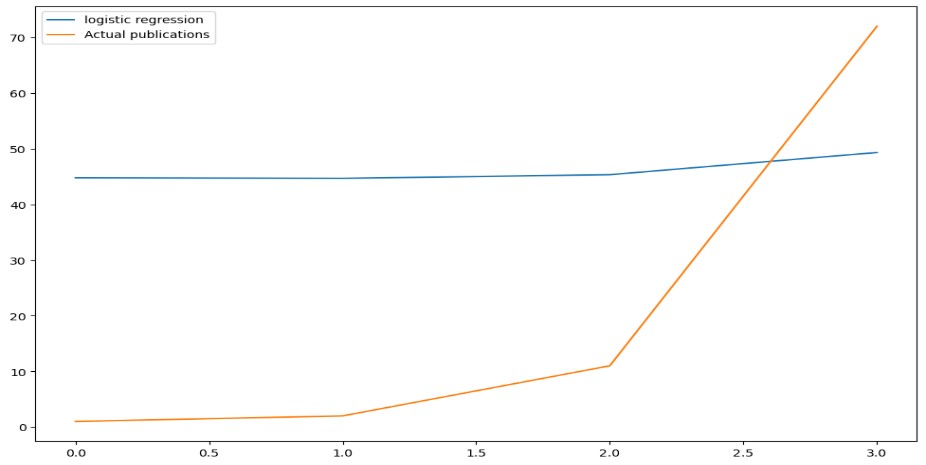
The below six graphs shows the prediction results for total publications and individual type of publications like Journals, patents, copy rights, book chapters and copy rights.



The below graph shows the actual vs predicted graph in which the blue line the prediction graph using SVM and the orange line shows the actual values plotted in a line graph.



The graph shown below represents the logistic regression graph. The orange line shows actual results graph and the blue line shows the logistic regression prediction.



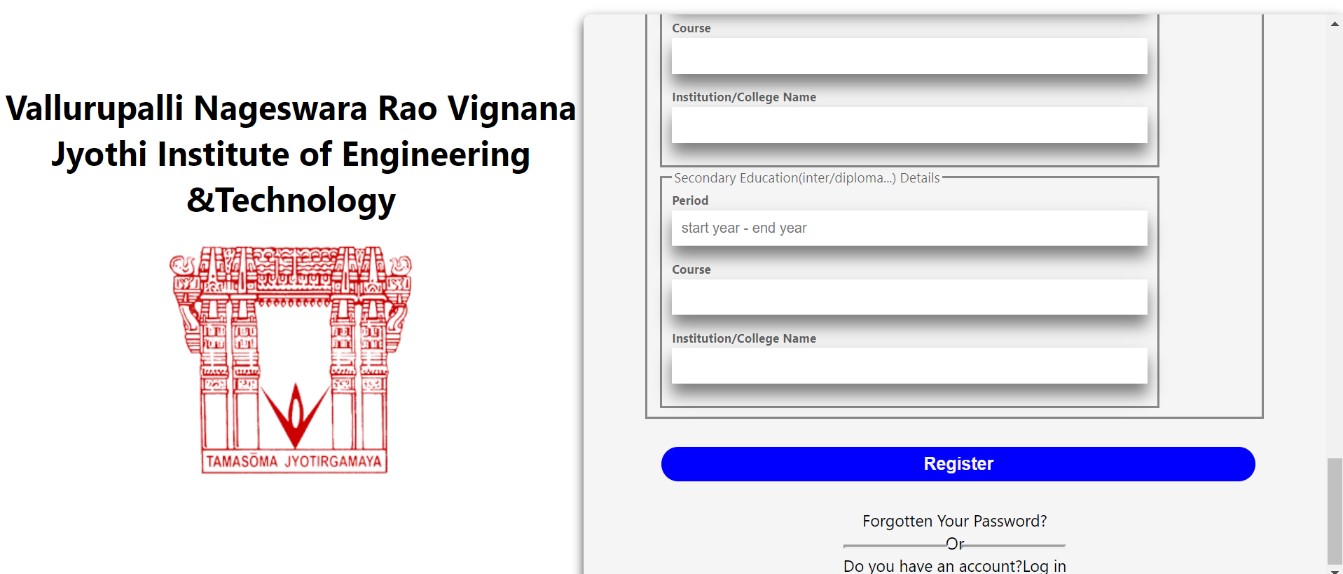
Among all these algorithms we used XG Boost for minimum value in range and Linear Regression for maximum value in the range. As these algorithms are best working for minimum data and the actual values are near to the predicted data by these algorithms.

**4.7 Register:**

To register as a user first step is to fill the details in the register page and click on register. When user clicks on register the details filled will be sent in Json format to the backend server, at the backend details will be saved to the database before saving password will get encrypted then a token will generated using jwttoken and the generated token will be added to the cookies in the browser using this cookie user is verified whether the user is logged in or not.

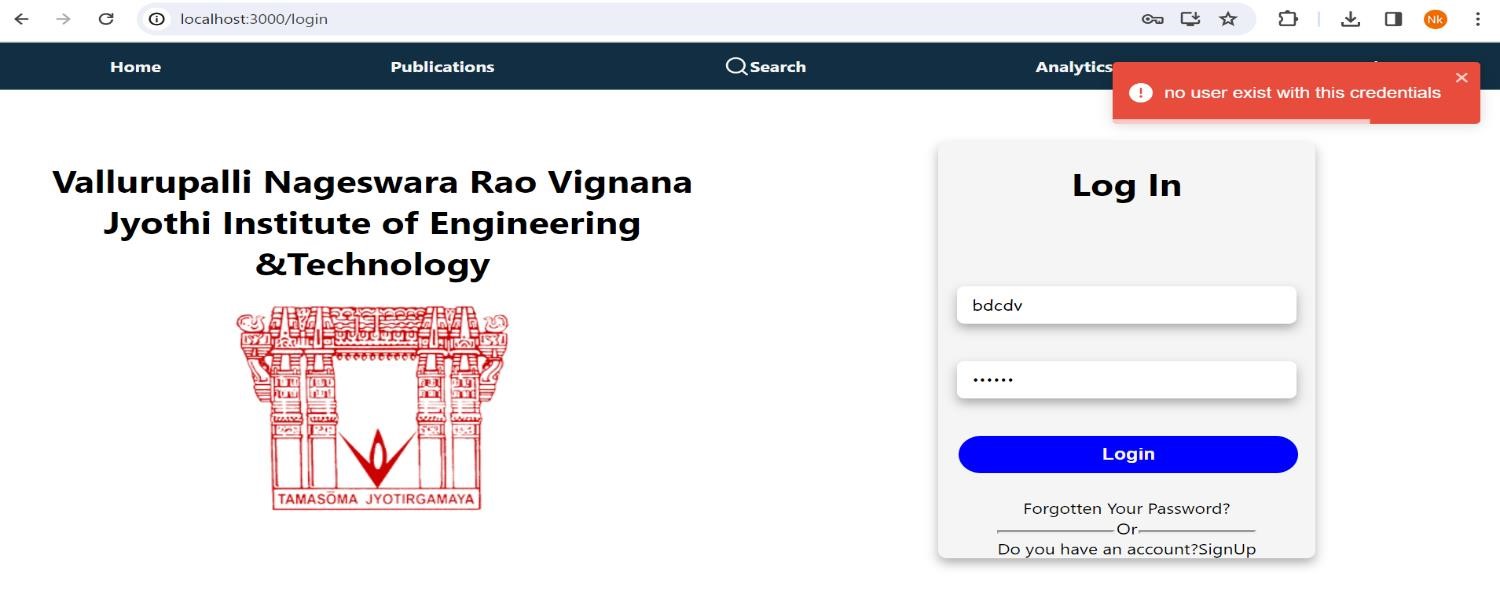
A screenshot of a computer

Description automatically generated



**4.8 Login:**

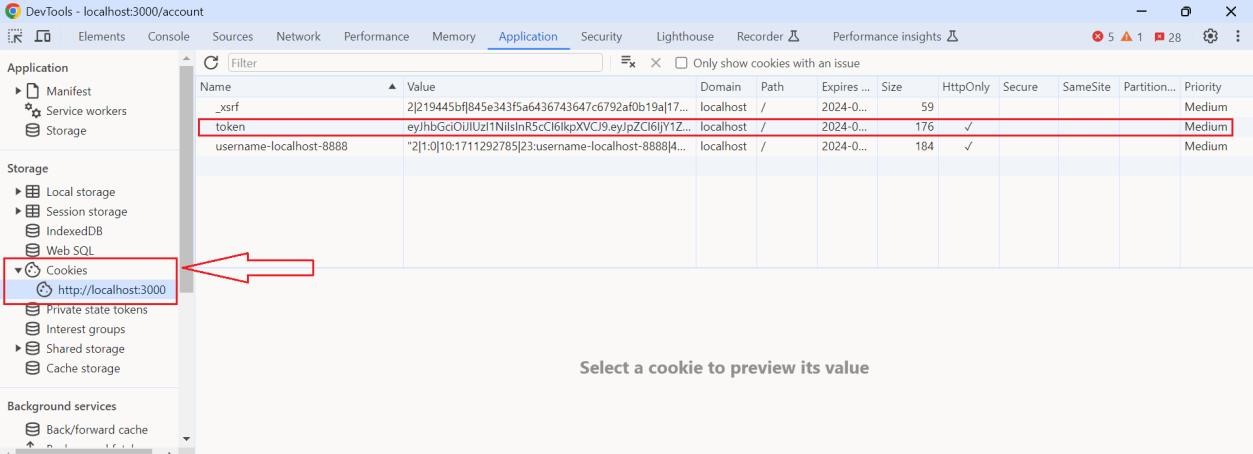
User have to login using their registered email and password. The login details are sent as a request to the backend server and the server verifies the details if the details are incorrect it responds with an error message, if the details are verified correctly a jwttoken is generated and added to the cookies. Whenever user tries to perform actions which are privileged only when user logged in into the account backend will verify whether the user is logged in or not using this cookie.



A screenshot of a computer

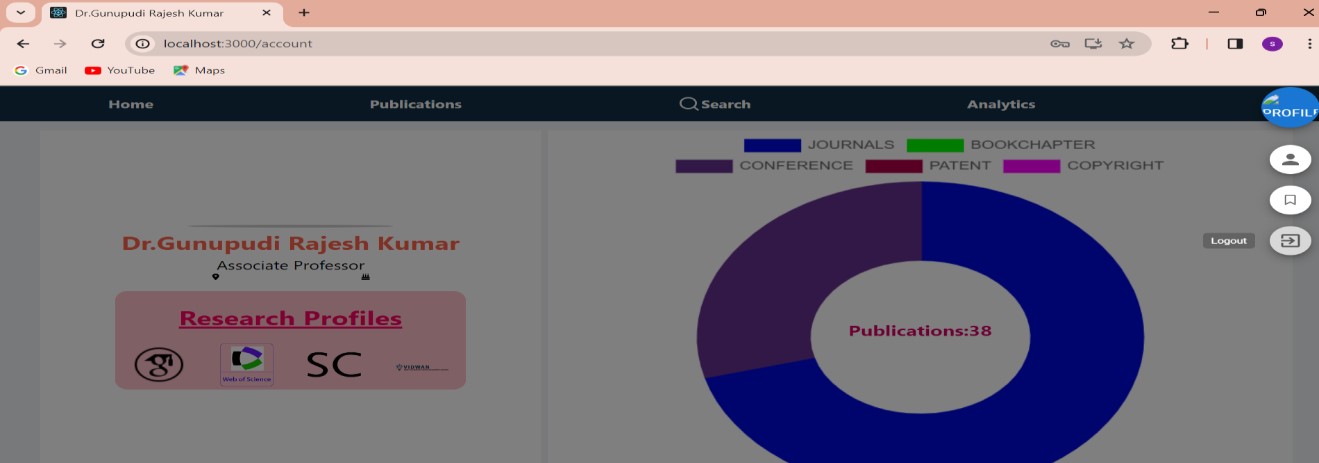
Description automatically generated

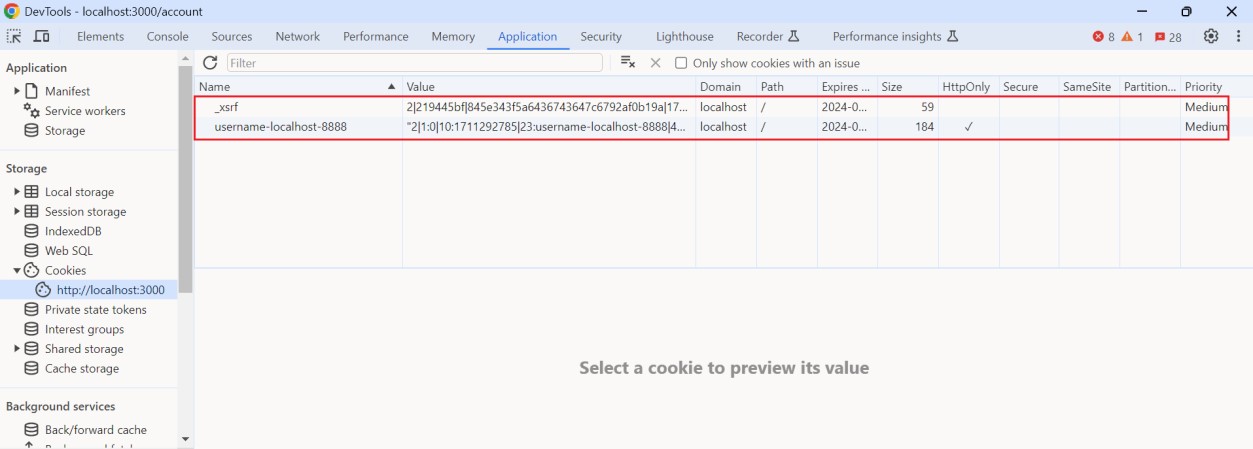
When user logged in into their account the token will be added into cookies as shown in the figure below.



**4.9 Logout:**

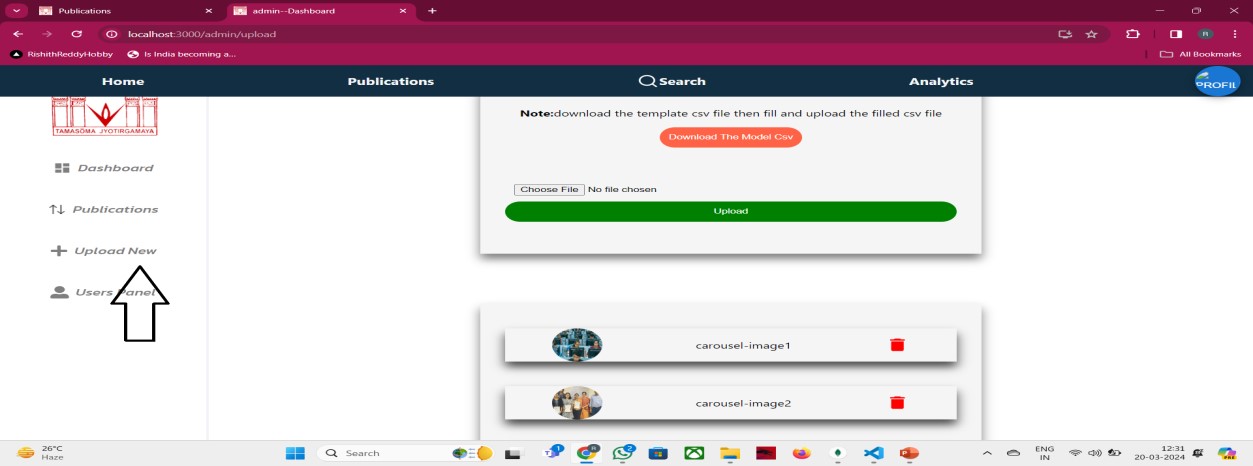
When user clicks on logout button backend server will remove the token from cookies.

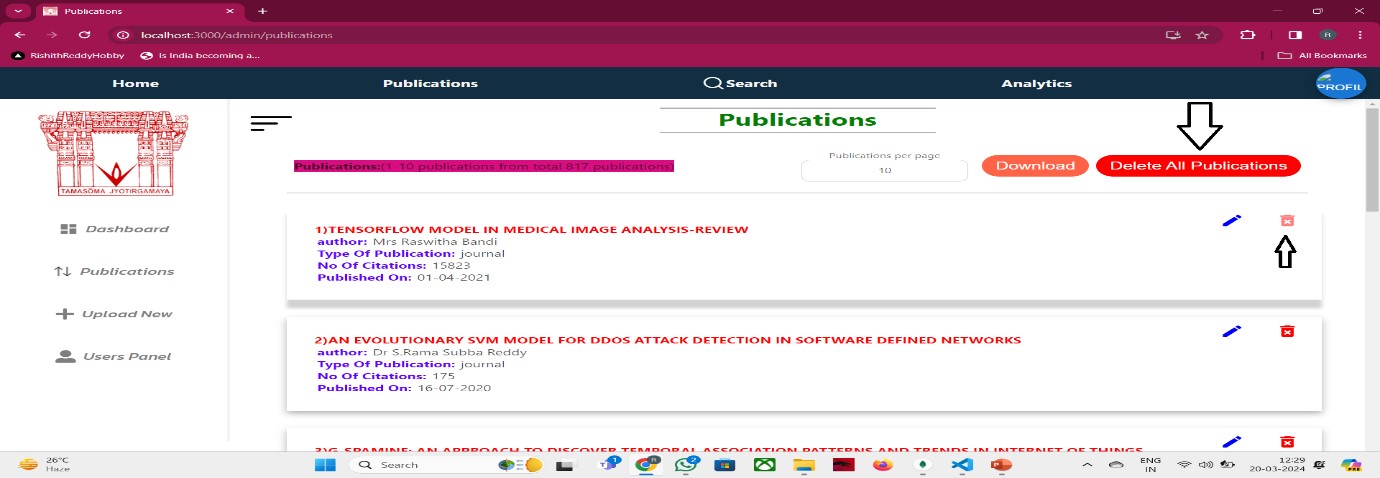


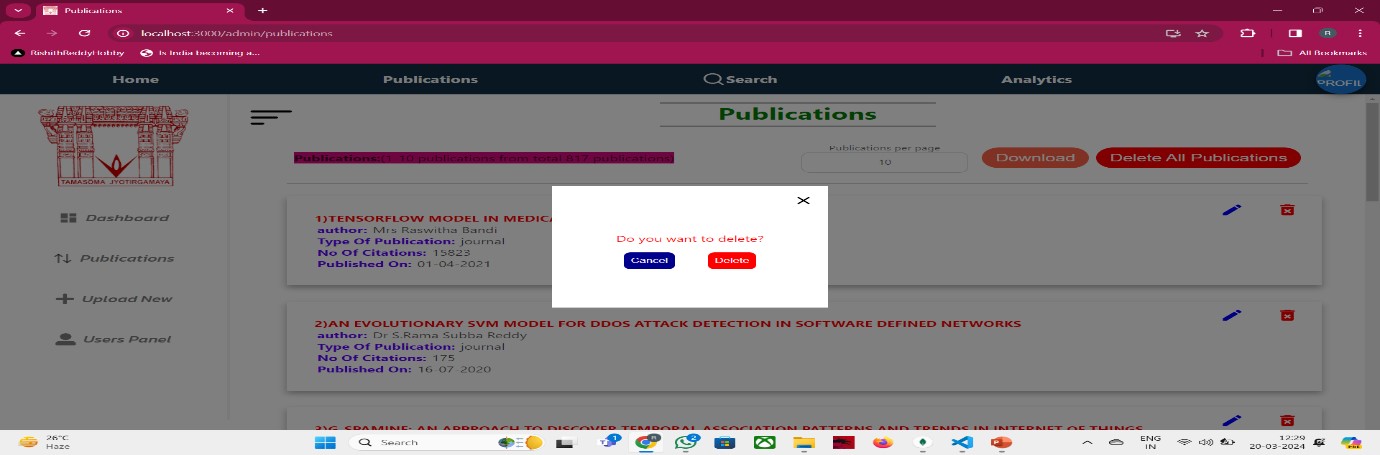


**4.10 Admin:**

For this application there are two types of roles one is admin and the other is user. The main differences between admin and user is that admin have access to all the user publications to update, delete and also can add new publications but the user have access only for to adding their own publication. The admin can also add and delete pictures to carousel, can add new users and can change their role to admin.







**V.Conclusion:**

To sum up, our project marks a significant advancement in the field of academia as we set out on a revolutionary quest to improve the efficacy and efficiency of research material management for academic researchers and college instructors. This project is our answer to the issues presented by the unparalleled era of information dissemination that the modern era has ushered in. We are dedicated to using cutting-edge technology, such as HTML, CSS, JavaScript, ReactJS, MongoDB, and ExpressJS, since we want to provide academics with a cutting-edge tool that makes organising, finding, and retrieving research materials more efficient.

Fundamentally, the goal of this application is to completely transform the way that research resources are accessed by making them simply accessible and discoverable. Users may easily explore a multitude of academic resources with keyword-based and advanced search options, cutting down on search time and improving the effectiveness of their research projects. Additionally, the research experience is made more personalised by the integration of Natural Language Processing (NLP) and Machine Learning (ML) algorithms, which provide customised recommendations based on users' unique study goals.

Beyond the efficiency benefits, our project aims to use data analytics to give academic institutions a complete picture of faculty research activity, enabling them to make well-informed decisions. We place a high priority on having an intuitive and responsive user interface so that users can easily access the programme on many platforms and that strong data security is maintained to protect confidential research materials.

In addition, we understand that continuous user education, assistance, and a sustainability strategy are critical to guaranteeing the application's durability and flexibility. This project is a major achievement at the nexus of technology and academia, not just a technological integration. It has the potential to increase research process productivity and efficacy, making it easier for academics and college instructors to move about the academic environment. As we go, our dedication to enabling academia with a technologically advanced, dynamic solution that maximises the potential of research materials and cultivates a culture of research excellence and knowledge progress does not waver.

**VI.FUTURE SCOPE**

We anticipate a state-of-the-art interface in the future that will seamlessly and effortlessly provide article searches, successfully addressing the shortcomings of the current system. By utilising the latest Machine Learning (ML) techniques along with Natural Language Processing (NLP) capabilities, our system will provide users with extremely precise and context-aware keyword-based and abstract-based searches, hence improving their overall experience. Future improvements will centre on sophisticated data visualisations, utilising machine learning algorithms to produce informative and interactive data representations, in addition to effective searching. These improvements will encourage a culture of research excellence and knowledge progress in the academic community by enabling users to easily locate research resources and to have a deeper grasp of the academic environment through dynamic and personalised data analytics.

**VII.References**

1. Natural Language Processing (almost) from Scratch Ronan R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. JMLR, 6:1817–1953, 11 2005
2. Ben Wellner,The Mitre Corporation 202 Burlington Road Bedford ,MA USA
3. James Pustejovsky ,Department of computer science,Brandeis University Waltham ,MA USA
4. Hinton, G.E.; Osindero, S.; The, Y.W. A fast learning algorithm for deep belief nets. Neural Comput. 2006, 18, 1527–1554. [Google Scholar] [CrossRef] [PubMed]
5. Vidwan Website-The display of the user publications and the graphs representations.
6. www.scopus.com,www.googlescholar.com,www.webofsciences.com-Publication details.