Legal Document Similarity Analysis

Corresponding Author / Report Owner: S Navin Sunder

Aayushi J Tripathy,
Department of Computer Science and Engineering,
Amrita School of Computing,
Bengaluru, India
BL.EN.U4CSE21003@bl.students.amrita.edu

S Navin Sunder,
Department of Computer Science and Engineering,
Amrita School of Computing,
Bengaluru, India
BL.EN.U4CSE21176@bl.students.amrita.edu

Siddhant Ashwani
Department Computer Science and Engineering,
Amrita School of Computing,
Bengaluru, India
BL.EN.U4CSE21189@bl.students.amrita.edu

***Abstract*: The similarity analysis of legal documents is done with the intention of improving document retrieval, grouping, and legal understanding. It uses ideas from text mining, machine learning, natural language processing, and information retrieval techniques. The main goal of the comparison is to compare the legal document's specifications, which will be vectorized, with the rest of the vectorized documents using text analysis. From there, a similarity score will be determined. The document that has the highest document score is assumed to be closely linked to the current document and will thus be shown. The article compares the accuracy of the models with those of the others and covers the various techniques currently employed for document similarity analysis.**

Keywords: legal document similarity, document matching, text similarity, common law system, machine learning, Natural language processing, Information retrieval

II. INTRODUCTION

A fundamental component of the legal profession, legal papers are the main method used to record and uphold laws, contracts, and agreements. These records come in a variety of formats, such as contracts, laws, rules, judgments, and legal opinions. The administration and analysis of these papers are of utmost importance since the legal profession strongly depends on their accurate interpretation and use. The research necessary for each legal matter is a crucial part of the preparatory process. These records are frequently voluminous and conceal crucial details of the case.

The legal documents are classified into two main categories: (I) previous court decisions and (ii) statutes, which are the written versions of a jurisdiction's laws that specify and punish certain crimes. The Common Law System places a great deal of weight on precedents, or earlier decisions that are comparable to the one at hand. Therefore, in order to comprehend and defend various legal features of a specific case, legal professionals must find numerous case papers that are comparable to it. Law professionals and academics need automated systems for searching and suggesting similar instances for a specific case since there are so many previous cases. The paralegals and solicitors find it challenging to draw out the aforementioned features as a result.

For it, a deep analysis of the various legal documents is required, and for doing so, there are a set of methods and algorithms that have been in current use. The methods currently in use broadly classify themselves into two categories: (1) network-based methods depend on the citations from the old documents, and (2) text-based methods make use of the textual information. Among these two categories, we have some methods, namely Text similarity, which compares the set of documents using techniques like word overlap, stemming, etc. Document structure, which compares structure of the documents like the order of the paragraphs, structure of matter, etc., and legal concept similarity, which compares the legal concepts that are discussed in the documents

This report finds that its goal is to compare the new cases with the ones that have already been resolved, focusing on the ones that have the greatest relative similarity in terms of case domain and issue for effective document clustering and semantic understanding in the legal domain. This is done in order for the paralegals to relate to the old cases and be prepared for court proceedings, taking into account the laws and sections of IPC used in the previous cases to be solved.

III. LITERATURE SURVEY

Definition: Legal document similarity analysis is a technique used to measure the similarity or relatedness between two or more legal documents, such as contracts, statutes, case opinions, or legal texts. The primary objective of this analysis is to assess how closely related or similar these documents are in terms of their content, structure, or legal concept

Key concepts and aspects of legal document similarity analysis:

Textual Similarity: At its core, legal document similarity analysis involves comparing the textual content of legal documents. This comparison can be based on various NLP techniques, including but not limited to:

·**Cosine Similarity**: This measures the cosine of the angle between the vector representations of documents. Documents with a smaller angle (a cosine value close to 1) are considered more similar.

Jaccard Similarity: This calculates the size of the intersection of terms in two documents divided by the size of their union. It is often used for comparing sets of words or phrases.

Word Embeddings: Techniques like Word2Vec or GloVe can be used to represent words and phrases in high-dimensional vector space, and document similarity is calculated based on the similarity between these vector representations.

i) Legal Ontologies:

To enhance the analysis, legal ontologies or knowledge graphs may be used. These ontologies capture legal concepts, relationships, and terminologies. By incorporating legal ontologies, the analysis can consider the semantic relationships between legal terms and concepts.

ii) **Document Matching:**

It is the process of identifying and pairing similar or related documents from a corpus of legal texts. This task is crucial in the legal field for various purposes,such as finding precedents, identifying potential plagiarism, or locating relevant documents for legal cases.

iii) Text Similarity:

It is a fundamental concept that helps measure the degree of resemblance between pieces of text, such as legal documents, contacts, case law, and regulations. Legal professionals often use text similarity techniques to identify similar documents, precedents, or relevant passages for various purposes.

iv) **Natural Language Processing (NLP):**

It is a subfield of Artificial Intelligence that focuses on the interaction between computers and human languages. It aims to enable computers to understand, interpret, and generate human language in a valuable way.

v) Information Retrieval:

It is the process of assigning labels or categories to documents based on their content. Finding relevant documents or passages from a large corpus of text in response to user queries

vi) Machine Learning:

It is a technique to implement Artificial Intelligence that can learn from data without being explicitly programmed.

vii) Named Entity Recognition (NER):

Identifying and comparing entities such as names of individuals, organizations, dates, and monetary amounts within legal documents can contribute to similarity analysis.

viii) Application Areas:

- **Legal Research:** Researchers and legal professionals can use similarity analysis to find relevant cases, statutes, or legal documents related to a specific topic.
- **Contract Analysis:** Businesses can analyze contracts to identify similarities or divergences between different versions or templates.
- **Plagiarism Detection**: In academia or legal practice, similarity analysis helps detect cases of plagiarism or unauthorized copying of legal documents.

Comprehensive Research on the papers:

1. *Legal case document similarity*:

This study examines how two court case papers compare to one another. Citation network-based and text-based approaches are also employed for this purpose.

Only past cases—also known as precedents—are taken into account by citation network-based systems (PCNet). Hier-SPCNet, which extends PCNet with a heterogeneous network of statutes, has been suggested by researchers. The dataset used in the study comes from the Indian Supreme Court, and legal professionals from two different reputable law schools in India—the Rajiv Gandhi School of Intellectual Property Law (RGSOIPL) and the West Bengal National University of Juridical Sciences (WBNUJS)—have annotated the document pairs' similarities. The first dataset serves as the validation set for the similarity estimation techniques' various hyper-parameters, while the second dataset serves as the test set to evaluate the performance of the methods [3]

2. *Legal document similarity: a multi-criteria decision-making perspective*

LIR tries to retrieve legal information items that are pertinent to a user's search. Legal information items are a variety of documents produced throughout a legal process, including court transcripts, verdicts, laws, and judgments. A lawyer needs these papers in order to make decisions and arguments since they are the main sources for judicial interpretations of the law. The four primary technologies of artificial intelligence (AI), network analysis, machine learning, and NLP have come together to create the methods and techniques employed in LIR. [4]

3. *Methods for Computing Legal Document Similarity: A Comparative Study*

This paper talks about the existing automatic techniques for finding similarity in documents that are network-based and text-based. It focuses on comparing all the existing methods of finding legal document similarity in order to ensure a fair comparison. The dataset contains 47 pairs of Indian Supreme Court case documents, where the similarity between each pair of documents is annotated on a scale from 0 to 10 by law experts. [5]

4. *Analysis of Law Document based on Word2vec*:

As the title suggests, this research paper talks about using the Word2Vec model for conducting document analysis of legal documents. The Word2vec model has the advantage that it can improve the accuracy of analysis by '0.20' when compared with the 'BOW model'. To summarize the index terms, they are: Learning, intelligence, Word2Vec, and similarity analysis. [6]

5. *Similarity Analysis of Legal Documents Using Content and a Network-Based Approach*:

This paper has proposed 'CaseRex,' which is a case recommendation system. It uses both content-based and network-based similarity measures for document similarity. The CaseRex algorithm is divided into two parts: Document Modeling and Relevant Document Identification. This algorithm structures the database of documents in a way that preserves their inherent relationships via the citations. Keywords used: natural language processing (NLP), Document Handling, and text analysis. [7]

6. *An Intelligent approach towards legal text document retrieval:*

The main focus of this paper is to retrieve relevant documents for particular Supreme Court cases in India from a set of prior case documents. For each distinct document, the current system lists a set of documents with their ranking scores using N_similarity, Fastt-Tex, and 'BERT'. The system returns the relevant document based on their ranking score. Database used: Document from Supreme Court of India Index terms used: information Retrieval, Legal Document, Natural Language Processing (NLP), Document Similarity [8]

7. *JPReg, a novel method to identify paragraph regularities in legal case judgments:*

Legal experts have to prepare a lot of legal documents, which is time- consuming. Paragraphs of existing documents that can be used, known as paragraph irregularities, need to be identified. JPReg is a method that can be adopted by legal experts as it adopts a two-step approach: in the first step, all similar documents are clustered according to semantic content, and in the second step, regularities are identified in the paragraphs for each cluster. Text embedding methods are adopted to represent a numerical nearest neighbor search method that is used to retrieve most similar paragraphs with respect to target document. This method has been found to be very effective in several experiments performed on real-world datasets. [9]

8. *Clustering legal documents using Topic Modeling*

The amount of information available for the justice system is voluminous, and it is extremely difficult for legal experts to sieve through so much of legal document information that is available. Legal documents, if managed properly as nuggets of knowledge, would help legal experts. If the legal documents are in a knowledge repository that is properly organized and if the legal experts can use a search engine on this knowledge repository for retrieval,analysis, and presentation, it is a very useful service. The clustering of similar documents can be done using topic modeling. [10]

9. *Exploring and Inferring Precedent Citations in Legal Documents Using a Web Visual System:*

The number of legal cases is ever increasing, and if binding precedents can be created for the lower courts by the Supreme Court, it will help in faster resolution of cases. A database of frequently cited binding precedents can be created, and this can be made available in a web-based visual analytics system to support analysis of legal documents that cite a binding precedent. Binding precedents can be created. The visual system can have three interactive components: the first showing an overview of data showing temporal patterns; the second grouping relevant documents by topic; and the third pointing to paragraphs that are likely to mention the binding citation.

This has been successfully implemented in the Brazilian judicial system with an accuracy of 96%. [11]

10. *Clustering and Summarization of Legal Documents*

The increase in legal documents available led to digitization of the legal documents and the need for knowledge management of all these digitized documents. There is a need for an automated tool for organizing, analyzing, retrieving, and displaying relevant content. A search engine can then be used to access the system of digitized documents. This can be very beneficial if automated summary of legal documents is also available, as that can be easily understood by common man. As an added feature, the system can display recommendations from well-known lawyers. [12]

IV. METHODOLOGY

A1. Evaluate the intraclass spread and interclass distances between the classes in your dataset. If your data deals with multiple classes, you can take any two classes.

Ans. In this question, we evaluate the mean, standard deviation, and interclass distance using the "Pandas" library. As the very first step, we call the library Pandas and read the dataset from the Excel sheet to the dataframe using the command read.excel() and store it in a variable. This command will print the dataset on which we have to perform our operations.

The commands mean() and std() will calculate the mean and standard deviations of the entire dataset.

```
Mean of each class:
embed_0       0.008983
embed_1      -0.025840
embed_2       0.026057
embed_3       0.044410
embed_4       0.000759
                 ...
embed_763     0.037724
embed_764     0.020917
embed_765     0.012222
embed_766    -0.018116
embed_767    -0.020047
Length: 768, dtype: float64
```

Fig. 1: The mean of each class of the dataset

```
Standard Deviation of each class:
embed_0       0.013080
embed_1       0.014345
embed_2       0.012180
embed_3       0.012642
embed_4       0.012814
                 ...
embed_763     0.010076
embed_764     0.010277
embed_765     0.011059
embed_766     0.011709
embed_767     0.010896
Length: 768, dtype: float64
```

Fig. 2: The Standard Deviation of each class of the dataset

To calculate the distance between vectors, select any two columns using "data['column']" to select a class to perform the operation.

```
-0.037839825529663355
-0.03292605200497847
The distance between mean vectors between classes:  -0.004913773524684882
```

Fig. 3: The distance between two vectors

A2. Take any feature from your dataset. Observe the density pattern for that feature by plotting the histogram. Use buckets (data in ranges) for histogram generation and study.

Ans. Select any class from the dataset using the command "data[column]d using the "matplot.pyplot" library using the plot command to obtain the required histogram.

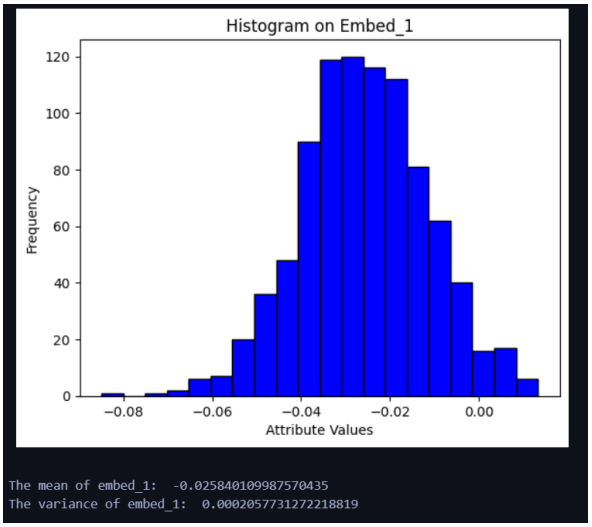

Fig. 4: Required Histogram

A3. Take any two feature vectors from your dataset. Calculate the Mikwoski distance with r ranging from 1 to 10. Make a plot of the distance and observe the nature of this graph.

Ans. Extracting the columns of a dataset to an array Set the r-values range from 1 to 11, run a for loop with the formula "dist = np.power(np.sum(np.abs(vector1 - vector2) ** r),1.0 / r)," and append the "dist" values into the list created. Using the pyplot library, plot the graph.

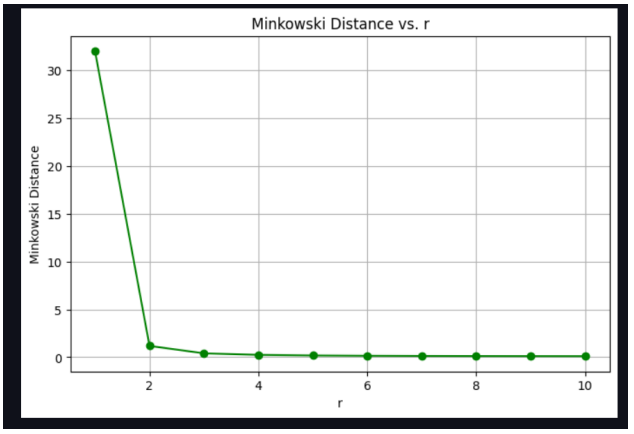

Fig. 5: Mikowski distance plot

A4. Divide the dataset in your project into two parts: training and implementation, and the testing set of the data.

Ans. Initializing the data into the array using the numpy library is the first step. Then split the dataset into features (X) and labels (y) and consider the last column as the target variable. Using train_test_split(),” the dataset of the project will be divided into two parts: the “train” set and the “test” set

A5. Train a kNN classifier (k = 3) using the training set obtained from the above exercise.

Ans. In this methodology, the kNN algorithm with k = 3 has been used to construct a classification model using the training dataset from the previous exercise. The scikit-learn library has been utilized for implementation, and the KNeighborsClassifier class has been imported for classification tasks.

The kNN classifier was trained with the training data, where X_Train is for feature inputs and Y_Train is for target labels. The three nearest neighbors were used to make predictions by the algorithm, which was learned from the training examples. To convert categorical labels to integer values, which is a step needed for classification tasks, we have assumed the use of LabelEncoder somewhere in the code.

```
▾        KNeighborsRegressor
KNeighborsRegressor(n_neighbors=3)
```

Fig. 6: Conversion of the data into the classifier model

A6. Test the accuracy of the kNN using the test set obtained from above exercise.

Ans. In this methodology, to measure the accuracy of kNN using the test dataset from the previous exercise, score() method, a function inherent to kNN classifiers, has been used. The score() method compares predictions against actual target values in the test dataset. The accuracy score result is in the range from 0 to 1, quantifying the proportion of correct predictions. The real world applicability of the model can be assessed using this methodology, as the computed accuracy score determines how the model will behave with unseen and novel data.

```
Accuracy: 0.269100030526334 36
```

Fig. 7: The accuracy value of the k-NN classifier

A7. Use the predict() function to study the prediction behavior of the classifier for test vectors.

Ans. In this methodology, to evaluate the performance of the kNN classifier, the predict() function has been used to make predictions on a set of test vectors (X Test). The predict() function generates class labels, which provide insights on how the kNN classifier classifies unseen data points based on learnings from the training data. Every element of the p array maps to a predicted class label for a specific test vector, which is key information to assess the effectiveness of the classifier. By comparing predicted labels with true labels, accuracy can be computed. The code snippet provided is critical for this methodology as it explains the application of the classifier and the basis for evaluating performance.

```
Predicted labels:
[-0.02083319 -0.02055517 -0.02692434 -0.03663652 -0.02665348 -0.02630889
 -0.03127179 -0.01004201 -0.01080981 -0.01224249 -0.0180552  -0.01980415
 -0.02773031 -0.01622231 -0.03794583 -0.0375789  -0.01918284 -0.02391689
 -0.01400778 -0.01325827 -0.02686621 -0.02127205 -0.0201616  -0.02370403
 -0.03240223 -0.00565731 -0.01337921 -0.01231107 -0.02498185 -0.01835036
 -0.02822533 -0.02101326 -0.02598897 -0.02598968 -0.02212124 -0.02289729
 -0.01672267 -0.01178713 -0.0357108  -0.02996049 -0.00518574 -0.0324085
 -0.03476314 -0.01670494 -0.01515334 -0.0153668  -0.02318467 -0.02329467
 -0.02458811 -0.00999943 -0.02055735 -0.01124649 -0.02062439 -0.02391827
 -0.02446232 -0.01981818 -0.01817206 -0.0212317  -0.02425308 -0.02189546
 -0.00634815 -0.01925654 -0.02527598 -0.01221924 -0.02167488 -0.02978044
 -0.00989201 -0.02836932 -0.02984229 -0.01280012 -0.01885084 -0.00579298
 -0.01237001 -0.03677778 -0.0227156  -0.03034946 -0.0168973  -0.02402442
 -0.01597065 -0.01436569 -0.02515982 -0.01505829 -0.01624077 -0.01173173
 -0.0235668  -0.00753194 -0.02346256 -0.0280609  -0.0257658  -0.01453262
 -0.01267023 -0.01746248 -0.02510263 -0.013773   -0.01687999 -0.03263339
 -0.0135416  -0.02578479 -0.0129264  -0.01873224 -0.01595846 -0.01843206
 -0.03103559 -0.0318237  -0.0219124  -0.02466763 -0.01564206 -0.02402119
 -0.01301699 -0.03193134 -0.01513044 -0.02546488 -0.01280012 -0.02358413
 -0.02157627 -0.0248011  -0.01391989 -0.02526246 -0.01424354 -0.02744392
 -0.02784042 -0.0289116  -0.02444321 -0.01416598 -0.02298612 -0.00917125
 -0.01981087 -0.01296527 -0.01296863 -0.02545284 -0.01839262 -0.02780673
 -0.02246169 -0.00510416 -0.0172633  -0.00519816 -0.0106855  -0.01687387
 -0.02891714 -0.02360236 -0.01705293 -0.01730287 -0.02227096 -0.02791076
...
 -0.0156517  -0.02878358 -0.0116262  -0.02051494 -0.0075837  -0.02715314
 -0.01978894 -0.02264387 -0.02209517 -0.02734255 -0.01931683 -0.02222461
 -0.02227436 -0.03714085 -0.02327848 -0.02707521 -0.02827807 -0.01828801
 -0.01746248 -0.01576231 -0.0070214  -0.01585967 -0.02588421 -0.01622486]
```

Fig. 8: Printing the predicted values over the testing dataset

A8. Make k = 1 to implement the NN classifier and compare the results with kNN (k = 3). Vary k from 1 to 11 and make an accuracy plot.

Ans. In this methodology, two nearest neighbor-based regression models were implemented and compared. scikit-learn's KNeighborsRegressor class was utilized to create the models with values of k = 1 and 3, and two key metrics were chosen to assess their performance.

1. Mean Squared Error (MSE) – MSE measures the average squared difference between predicted and actual target values. The lower the value, the better the accuracy. The MSE metric is used to determine the accuracy of the model.

2. R Squared Score (R^2) – R^2 computes proportion of variance in the target variable. Closer to 1 indicates better fit, and further indicates a bad fit. R^2 metric is used to determine how good the model is.

These metrics assess the efficacy of the regression model in predicting continuous target variables and, hence, provide insights into its performance.

```
Results for NN (k=1):
MSE: 0.00012546454023971257
R-squared score: -0.23194785411400365

Results for kNN (k=3):
MSE: 7.732513036150924e-05
R-squared score: 0.24073743676147763
```

Fig. 9: MSE and R-Squared values for different values of k

A9. Please evaluate the confusion matrix for your classification problem. From the confusion matrix, the other performance metrics such as precision, recall, and F1-Score are measured for both training and test data. Based on your observations, infer the model's learning outcome (underfit / regularfit, / overfit).

Ans, In this methodology, we utilize the kNN algorithm to assess performance of classification problems. The dataset is prepared initially, and the target variable is transformed into binary labels based on a threshold of 5. The dataset is then divided into two equal halves: the training dataset and test dataset. The kNN classifier is used for predictions, and is converted to discrete values by comparing them to a threshold of 0.2, followed by a confusion matrix to evaluate the performance of the model. The metrics that are monitored from the confusion matrix are accuracy, precision, and a few others, providing a comprehensive assessment of the model's ability to classify data.

```
The accuracy value is:  0.4533333333333333
The precision value is:  0.45112781954887216
The recall value is:  0.8695652173913043
The f1Score value is 0.594059405940594
```

Fig. 10: Assessment of the model's ability with respect to various parameters.

V. RESULT ANALYSIS

- Separation of the classes

Separation of classes is required for the visualization of the data, and along with that, it is required for studying the relationship between the feature vectors. In the scatter plot figure, we can see that the data points of 'embed_10' and 'embed_14' plotted are all clustered together, and no separation line is possible for both of the column variable vectors. Thus, after analyzing the scatter plot, we can say that the classes in our dataset are not well separated.

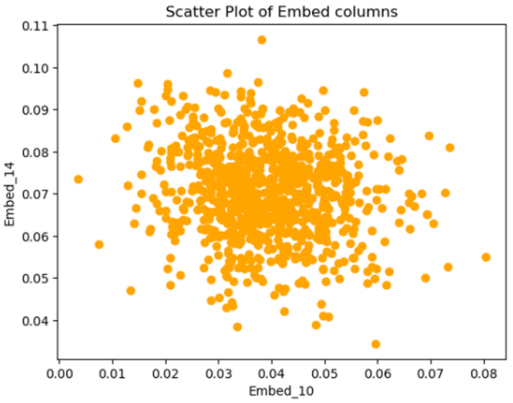

Fig. 11: Scatter plot of 'embed_10' and 'embed_14' for analyzing class separability.

- Usefulness of class centroid method

The class centroid in k means clustering; points that represent the clusters of the data are calculated as the mean of the data points belonging to the cluster. In this case, as all the data point values lie close to the other class's clusters, finding out the class centroid method is less useful for analyzing the class separability. Even the values of the data in one single class are varying to an extent, or in simple terms, there are a large number of outlier values, making it an inefficient method for data analysis.

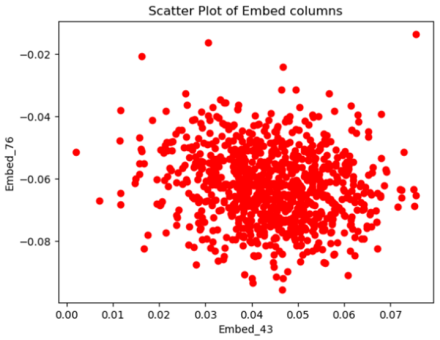

Fig. 12: Scatter plot for 'embed_76' and 'embed_43' columns

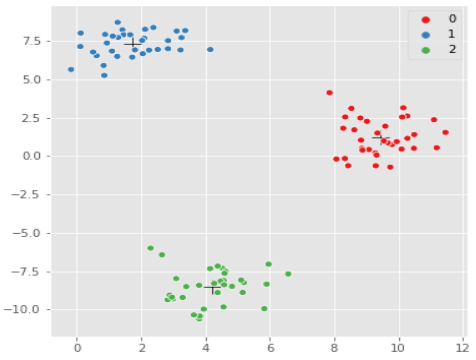

Fig. 13: An illustration where the class centroid method works well. [13]

In the above figure, we can see that the three classes have values such that they are separated as clusters by themselves, and we can have a proper class separation in them.

- Behavior Of KNN Classifier with 'k'

When k is small, the prediction is based on a very localized group of neighbors. This model tends to capture noise in the data and can lead to erratic predictions. Hence, this model may not be a good prediction as it is influenced by individual data points and may not represent the underlying pattern of data.

There is an optimal value of k that provides the right prediction by capturing the underlying pattern of data well. The predictions are accurate and stable. The optimal value of k needs to be determined through techniques such as cross-validation.

When k is large, the prediction is based on a broad group of neighbors. The prediction is not sensitive to individual data points, and the data is more representative. However, this can go wrong if the dataset has a complex structure, as a very large k value can lead to overgeneralization, and the model can miss important details in the data.

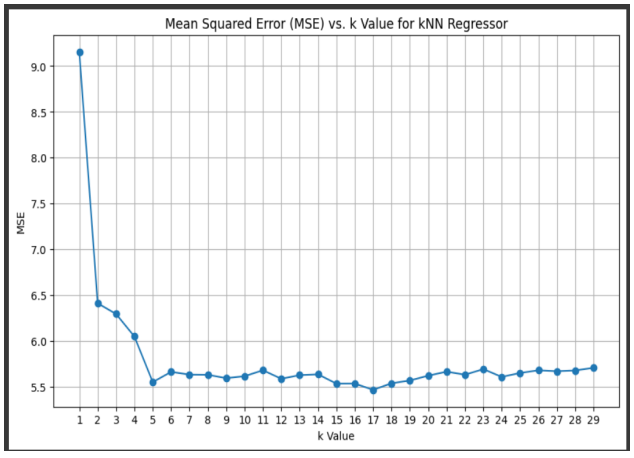

Fig. 14: Illustrating Mean Squared Error (MSE) vs. K value for a kNN Regressor

The above illustration shows a graph of Mean Square Error vs kNN Regressor. The optimal value of k is determined based on the lowest MSE value on the x-axis. We choose a value of k = 5 as the optimal value, even though the lowest value of MSE is at k=17, because a higher value of k can lead to underfitting as the decision boundaries become simpler and too smooth, not adapting well to the actual distribution of data points in the feature space.

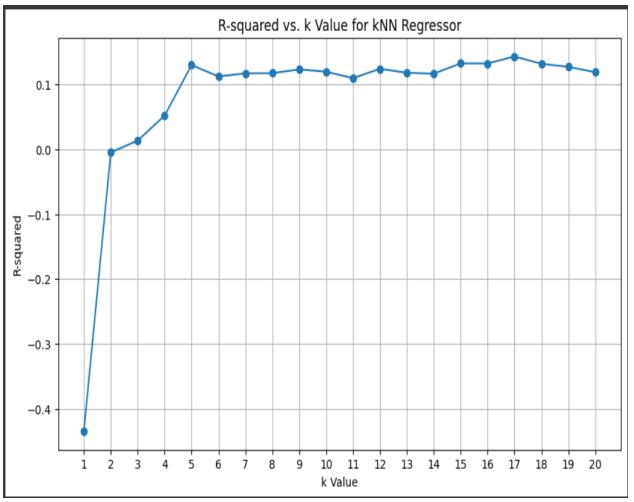

Fig. 15: Illustrating R-squared vs K value for a kNN Regressor

The above illustration shows a graph of R squared error vs. k value for kNN regressor. The optimal value for k is the one where the R square error is maximum. In this graph, we choose as the R is at its blue is maximum at the corresponding point. We also ensure that the k is not too high or too low to avoid overfitting and underfitting.

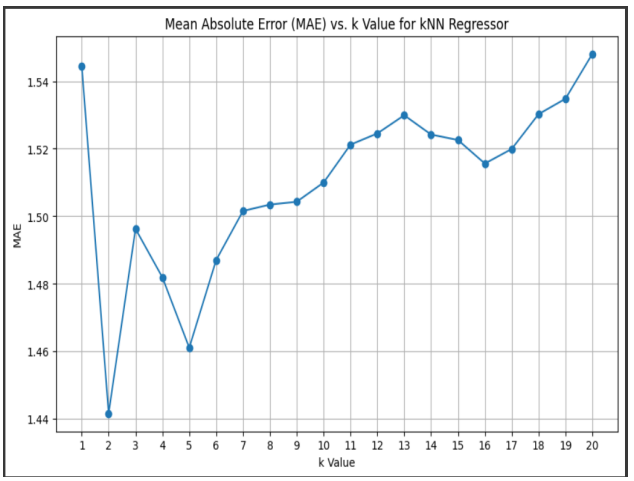

Fig. 16: Illustrating Mean Absolute Error (MAE) vs. K value for a kNN Regressor

The above illustration shows a graph of Mean Absolute Error vs k value for a kNN regressor. We choose an optimal value of k where the MAE is lowest. Here, we

Choose k = 2 as the MAE value that is lowest, corresponding to that k value x-axis.

Optimal reasoning for all metrics-

After evaluating the three metrics—MSE, MAE, and R-squared — we have determined that the optimal value for k is 5. This decision is based on the fact that two out of the three metrics pinpoint 5 as the optimal k value, striking a harmonious balance between both overfitting and underfitting.

Selecting a high k value could potentially lead to an under fitted model, as it might not sufficiently learn the intricate patterns present in the data.

Conversely, opting for a very small k value can cause the model to overfit, as it would over-learn the patterns in the training data, adapting too closely to the noise and fluctuations, which does not generalize well to unseen data.

Therefore, an optimal k should be a value that adeptly avoids both underfitting and overfitting, while also ensuring minimal error. In our case, we have chosen to proceed with k = 5 as it meets these criteria.

- Implementation of KNN Classifier

For both classification and regression problems, the k-Nearest Neighbors (KNN) classifier is a straightforward and efficient machine learning technique. The idea behind it is that similar data points usually belong to the same class or have comparable goal values. KNN classifier functions as follows:

1. Training:

The complete dataset and any associated class labels used for classification or target values used for regression are stored in the training phase of the KNN algorithm.

2. Predictions:

For classification: In order to classify a new data point, KNN uses a distance metric (such as Euclidean distance) to locate the k-nearest data points (neighbors) from the training set. The anticipated class for the new data point is the class label of the vast majority of these k neighbors.

For Regression: The average, also known as the weighted average of the target values of the k-nearest data points, is computed by KNN as the prediction for a new data point when it has to be used to forecast a target value for a new data point

The Hyperparameters (a machine learning parameter that is chosen before the training of a learning algorithm) of the KNN algorithm are:

1. K: ‘K’ determines the number of neighbors. The number of closest neighbors to take into account is one you set. The performance of the algorithm may be influenced by selecting a suitable value for k. While a big k can cause the model to become too smooth, a little k could make the model more susceptible to noise.

2. Distance: Depending on the type of data and issue at hand, you can select from a variety of distance measures, including Manhattan distance, Euclidean distance, and others.

For the given dataset, the KNN classifier proved to be efficient, as the analysis could be performed on both classification- and regression-based datasets. Parameters such as Mean Squared Error (MSE) and R-Squared Error are used to perform the analysis after predicting the values for the given value of K.

Mean Squared Error (MSE): A standard indicator for assessing how well regression models perform. It calculates the mean of the squared discrepancies between the values in a dataset that are predicted and the actual (observed) values. In other words, it measures how well the predicted values agree with the actual values; greater values suggest a larger average squared difference, which denotes inferior performance. This can be used to assess the model’s accuracy in predicting continuous target variables, which in turn will determine the performance of the machine.

Formula:

$MSE = (1/n) * \Sigma(y\_actual - y\_predicted)^2$

where n: is the number of data points in the dataset

y_actual: represents the actual values.

y_predicted: represents the predicted values

Properties:

1. A lower MSE represents greater model performance, and it is always a non-negative value.

2. Smaller errors are less penalized by squaring the differences than larger ones.

3. Because huge errors will have a considerable impact on the total MSE, it is sensitive to outliers.

4. Direct interpretation of MSE can be difficult because it is expressed in the target variable's squared units.

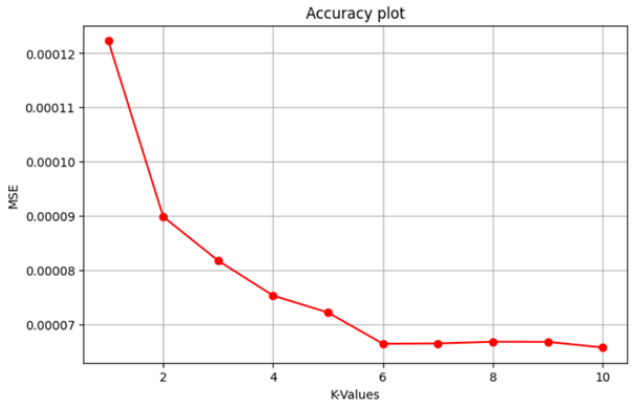

Fig. 17: Graph to show accuracy plot for Mean Squared Error and K-values

R-Squared Error (R2): A statistical indicator frequently used to assess the regression model's quality of fit. It reveals how effectively the model accounts for the variance in the dependent variable (the intended variable). A higher R2 number denotes a better fit of the model to the data. R-squared is a value between 0 and 1. The amount of the dependent variable's variance that the independent variables (features) in the model are responsible for explaining is expressed as a percentage by the R-squared. A value of 0 means the model does not explain any of the variance in the target variable, whereas a value of 1 means the model perfectly accounts for all the variance. The proportion of variance explained is measured between 0 and 1, with larger values denoting more explanatory power.

Formula:

R2 = 1 - (SSR / SST)

, where SSR (S(y_actualred Residuals)) is the sum of the squared discrepancies between the predicted values and the dependent variable mean

SST (Total Sum of squares) is the sum of the squared deviations between the actual values and the dependent variable's mean.

Properties:

1. R-squared can be understood as the percentage of the dependent variable's overall variation that the model can account for.

2. A better fit is indicated by higher values, which range from 0 to 1.

3. An R-squared number that is negative indicates that the model performs worse than a horizontal line (the dependent variable's mean).

4. R-squared should be taken into account along with other metrics and domain expertise, not as the exclusive standard for measuring a model's performance.

Accuracy Plot for R-Squared Error for the K-values from 1–11:

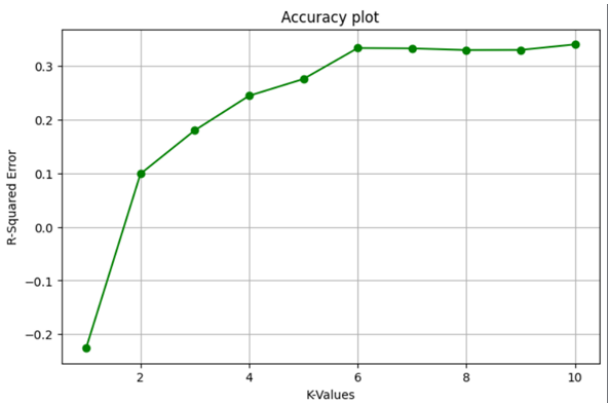

Fig. 18: Graph to display the accuracy plot for R-Squared Error and K-Values

To conclude the discussion, we can conclude that a KNN classifier can be a good classifier based on the observations made and mentioned in the above question.

- Regular Fitting of the dataset

A model with a regular fit, commonly referred to as a balanced fit, strikes a decent compromise between complexity and generality. Without being extremely simple or unduly complex, it reflects the underlying patterns in the data. A normal fit should have the following key features:

1. With a reasonable amount of training error, the model performs well on the training set of data without being overfit (low bias).

2. Good generalization: The model also performs satisfactorily on test or unknown data, proving its generalizability (low variance).

3. Appropriate complexity: A regular fit model has the correct number of parameters or complexity to accurately depict the true relationships in the data without fitting noise.

Techniques like cross-validation, regularization, feature selection, and model complexity monitoring are frequently used to achieve a regular fit and make sure the model generalizes effectively without overfitting or underfitting.

We determine whether a model is a regular fit based on accuracy and Recall.

In terms of accuracy, both the training and validation datasets were very high and stable.

And in terms of recall, it requires both high and stable recall throughout the training and validation datasets.

	precision	recall	f1-score	support
0	0.94	0.86	0.90	237
1	0.38	0.64	0.48	33
accuracy			0.83	270
macro avg	0.66	0.75	0.69	270
weighted avg	0.88	0.83	0.85	270

Fig. 19: The values of Accuracy, Precision, Recall and F1 Score

From the above observation, we can witness that both Accuracy and Recall are high and consistent for both the test and training datasets. From this, we can conclude that the model is a Regular fit.

- Overfitting in the k-NN classifier

Overfitting occurs when the value of k is too small or when the model is too complex for the given dataset.

When k is small say 1 or 2, the model becomes sensitive to noise or outliers in the data. It tries to fit the training data too closely and handles minor variations which may not represent the underlying pattern of the data. In this scenario, the model performs well on training data but poorly on unseen or new data.

When data is complex, noisy and not balanced, a small value of k can lead to overfitting as the model will capture the noise or the specific characteristics too much and thus lose out on the true relationships in the data.

If the underlying data pattern is localized, small value of k will capture the localized patterns and fail to generalize well for overall data.

VII. CONCLUSION

In this paper, we implement the k-NN classifier technique to analyze the embedded dataset. We then ran the dataset through several accuracy testing methods, and we further found the best value for 'k' so that the k-NN model could be applied to it. Since legal papers come in a variety of formats and it might be difficult to sort through a lot of information in these documents, the necessity of analyzing legal documents for similarity is ultimately explored. Knowledge management is required in order to enable a search engine that makes it simple to find, analyze, and show related documents and paragraphs for use in currently written legal papers. This report uses two widely used methodologies, network-based and text-based and discusses a wide range of approaches and tools, including machine learning, that may be used for similarity analysis.

VII. ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the university, Amrita Vishwa Vidyapeetham, our lecturer, Dr. Peeta Basa Pati, and Ms. Roshni for assigning us the assignment and project and helping us with the insights through the project. Their knowledge of the subject assisted in the successful completion of this project report.

VIII. REFERENCES AND CITATIONS

[1] What is Overfitting? Overfitting in Machine Learning Explained (AWS)

https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20is%20an%20undesirable%20machine,on%20a%20known%20data%20set

[2] Regression vs Classification in Machine Learning: Javatpoint
https://wwsmall.atpoint.com/regression-vs-classification-in-machine-learning

[3] Paheli Bhattacharya, Kripabandhu Ghosh, Arindam Pal, and Saptarshi Ghosh, "Legal case document similarity": Volume 59, Issue 6, 2022, 103069

ISSN 0306-4573

https://www.sciencedirect.com/science/article/abs/pii/S0306457322001716

[4] Rupali S. Wagh and Deepa Anand, Published: 2020 Mar 23.PM CID: PMC7924540PMID: 33816914," Legal document similarity: a multi-criteria decision-making perspective"

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7924540/

[5] Paheli Bhattacharyaa, Kripabandhu Ghosh, Arindam Palcand, and Saptarshi Ghosha: Methods for Computing Legal Document Similarity:A Comparative Study, published: April 26, 2020

2004.12307.pdf (arxiv.org)

[6] C. Xia, T. He, W. Li, Z. Qin, and Z. Zou, "Similarity Analysis of Law Documents Based on Word2vec," 2019 IEEE 19th International Conference on Software Quality, https://ieeexplore.ieee.org/document/8859429

[7] S. Chavan, J. Balasubramanian, J. Puro, M. Naik and A. V. Nimkar, "Similarity Analysis of Legal Documents using Content and Network Based Approach," 2020 https://ieeexplore.ieee.org/document/9225586

[8] M. M. Rahman, Z. Azmaeen, M. Arman, M. L. Rahman, and M. M. Hoque, "An Intelligent Approach Towards Legal Text-Document Retrieval,"

https://ieeexplore.ieee.org/document/10054858

[9] De Martino, G.; Pio, G. (2022). Identification of Paragraph Regularities in Legal Judgements Through Clustering and Textual Embedding

https://link.springer.com/chapter/10.1007/978-3-031-16564-1_8

[10] Halgekar, A., Rao, A., Khankhoje, D., Khetan, I., and Bhowmick, K. (2023). Topic Modeling-Based Approach for Clustering Legal Documents, Springer, Singapore

https://link.springer.com/chapter/10.1007/978-981-19-0095-2_17

[11] L. E. Resck, J. R. Ponciano, L. G. Nonato and J. Poco, "LegalVis: Exploring and Inferring Precedent Citations in Legal Documents,"

https://ieeexplore.ieee.org/document/9716779

[12] A. Rao, A. Halgekar, D. Khankhoje, I. Khetan, and K. Bhowmick, "Legal Document Clustering and Summarization, pp. 1–4.

https://ieeexplore.ieee.org/document/10010585

[13] Stack Overflow (n.d.): Distance between nodes and the centroid in a KMeans cluster

https://stackoverflow.com/questions/54240144/distance-between-nodes-and-the-centroid-in-a-kmeans-cluster