DATA REDUCTION :

Chart, histogram

Description automatically generated

There are around 150 words on average in every document. We restricted the word count to 150 in each of the documents. We then took a subset of 20 % of the overall data randomly while still preserving the distribution of the classes.

Graphical user interface, application, Word

Description automatically generated Graphical user interface, text, application

Description automatically generated

Original distribution Sampled distribution

We then grouped the minority classes with less than 3% of the data into a class called “Other”.

SVD for Data Decomposition :

1. SVD can help reduce the dimensionality of data by decomposing the original matrix into three separate matrices: a left singular matrix, a diagonal matrix, and a right singular matrix.
2. The diagonal matrix contains the singular values of the original matrix, which can be thought of as a measure of the importance of each dimension in the data. By retaining only the top-k singular values, where k is a smaller number than the total number of singular values, we can reduce the dimensionality of the data while preserving most of its original information. Here we used the matrix obtained from the “Abstract” column to reduce the data. This is used for modelling to avoid overfitting.

Lemmatization :

First, a text lemmatizer was used to extract the core lemma from all the words. Lemmatizer was preferred over stemming as it preserves the actual meaning of the word. The lemmatizer was built on a random sample of the data eliminating all bias, that is discussed above.

Graphical user interface, text, application

Description automatically generated

SENTIMENT ANALYSIS:

Initially, the data was used to create distinct sentiment classes

Chart, bar chart

Description automatically generated Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated Chart, bar chart, waterfall chart

Description automatically generated

The positive sentiment is the majority class in all these analyses. A Vader pretrained model was used to mine all these sentiments in the abstracts.

For example, take the last 2 records:

Graphical user interface, application

Description automatically generated

We indeed find that it is a positive sentiment talking about what the paper has developed better than the existing literature. Also, it’s a strong argument that’s also very technical and involved the use of some jargon. So, our model did a decent job of mining the sentiment classes.

We also identify the technical sentiment through this word cloud visualization:

A picture containing text, newspaper

Description automatically generated

All the top terms used for the technical class talk about the technical advancements and nuances of the paper.

Topic Analysis:

We extracted the top 5 topics from the classes overall:

Graphical user interface, text, application, email

Description automatically generated

Here we see that the top 5 topics are all containing technical words that fall under the positive class which is the majority class at around 80 %. This again shows that most papers are highly technical and talk about the advancements and positive improvements brought about by the introduced paper.

Classification model – base:

Graphical user interface, text, application

Description automatically generated

We further went on to build a basic random forest classification model to predict the types of papers in the positive sentiment classes that are in the majority. We achieve barely acceptable accuracy. This shows that our model may be overfitting. So, we need to introduce more data for better generalization. So, for our overall modeling process, we use both positive and negative data.

Named Entity Recognition:

For our NER process, we found that the abstract text contained no location-based data. We were unable to extract geographic-tagged information. This again makes sense because there is very little geo-info when it comes to highly technical papers.

We were however able to extract the top nouns, verbs, and adjectives from the data as shown below. This again consolidates what we found earlier about the type of data in the majority classes. These nouns, verbs, and adjectives all convey a technical, positive (talks about the advancements of the paper), argumentative (existing methods minuses vs introduced methods positives) approach.

Chart, bar chart

Description automatically generated Chart, bar chart, histogram

Description automatically generated

Chart, bar chart, histogram

Description automatically generated

**Data Splitting:**

The data was split into 3 mutually exclusive and cumulatively exhaustive datasets namely:

1. \*Training data\*: This data comprised 60% of the entire dataset. This was used for training the model.
2. \*Validation data\*: This was 20% of the entire dataset. This was used to compare models.
3. \*Test Data\*: This was again 20% of the entire dataset. This was used to check what the accuracy of the final model will be on the unseen data.

Since the dataset size was reasonably high, we went ahead with a 60-20-20 split. Even though we trained only on 60% of the entire dataset, it gave us a good chunk of data to train on and provided us with the flexibility to keep a reasonable number of records for validation and test respectively.

The validation and test datasets kept at 20% each enabled us to maintain a proper distribution of classes in each dataset. This would help us capture all the nuances of the dataset in each validation and testing dataset.

This provided us the ability to check for the accuracy of the model with decent confidence. We also measured the percentage of output classes in each of the datasets and it remains uniform.

The following graphs compare the share of each class in training, validation and test dataset respectively.

**\*Training data\* :**

Chart, bar chart, histogram

Description automatically generated

**\*Validation data\* :**

Chart, bar chart, histogram

Description automatically generated

**Test Data**

**Chart, bar chart, histogram

Description automatically generated**

## Select Modeling Techniques

Since the goal of this project is to perform classification of research papers into categories, we would want to perform multi-class classification. Multiple approaches will be tried to get the optimal model.

While selecting the modeling techniques, we want to have an exhaustive set of models with different capabilities.

We will choose to build the following classifier models:

* Logistic Regression: A linear model. This would help us capture the patterns in the data if it is linearly dependent on the output class. This model has assumptions of linearity of log-odds and no multicollinearity.
* Decision Tree: Would be able to capture the nuances and non-linear nature of the dataset. This is a robust model that has fewer assumptions.
* Random Forest: A cluster of decision trees. Typically works better than a single decision tree. This also has fewer assumptions on the training data.
* Support Vector Machine: Transforms the data into higher dimension to easily separate them out. The underlying assumption is that the support vectors, or the points that are used to demarcate the boundary, are separable in higher dimensions.

We were also aligned to using Naïve Bayes Models, since they work the well with text dataset. But since this dataset has been transformed by using Singular Value Decomposition (SVD), we would not be able to use Naïve Bayes Model.

The abstract column containing text was used for the overall modelling process. This abstract column was fed into a TF-IDF vectorizer. TF-IDF, short for term frequency-inverse document frequency, is a widely used statistical measure that assesses the relevance of a word to a document within a collection of documents. It works by calculating two metrics for each word in a document: the term frequency, which is the number of times a word appears in the document, and the inverse document frequency, which measures how often the word appears across all documents in the collection.

We used TF-IDF to select the most important terms to include in the models, thus improving its accuracy and performance. This has been further explained in the modelling stage below.

Code :

**#Building the TF-IDF Matrix**

**features = out['Abstract\_v2']**

**vectorizer = TfidfVectorizer(max\_features=2500, min\_df=10, max\_df=0.8)**

processed\_features = vectorizer.fit\_transform(features).toarray()

To present an idea, this is a snapshot of the term-by-document matrix :

Graphical user interface

Description automatically generated with medium confidence

Our TF-IDF method gives more weightage to the words appearing less frequently in the term document matrix (Inverse document frequency) and thus selects those terms that are not ubiquitous or commonplace. Thus we ensure proper generalization of our models.

Now, coming to the models we talked about above, they would get us the best of all worlds: linearity and non-linearity. They are also easy to model, and easier to interpret, allowing for better generalization and explainability from a business standpoint.

**Build The Models**

The text dataset was first processed with 2 types of feature engineering techniques:

* One hot encoding or Bag of Words
* TF-IDF Vectorization

There were multiple models that were trained to both of these features respectively:

* Random Forest
* Decision Tree
* Support Vector Machine
* Logistic regression.

The inherent problem in the dataset was class imbalance, meaning that some classes had very few samples compared to others. Hence, the models were processed for class-imbalance using a parameter called 'class\_weight' inside the machine learning model itself. For example, the random forest classifier was initialized with 'balanced' class\_weight, as shown in the code snippet.

Following code was used:

```Python

rf = RandomForestClassifier(class\_weight='balanced')

rf.fit(X\_train\_reduced, y\_train)

y\_pred\_rf = rf.predict(X\_valid\_reduced)

accuracy\_rf = accuracy\_score(y\_valid, y\_pred)

```

Logistic regression model was not able to converge even after iterating over the dataset multiple times.

We found that random forest on TF-IDF vectors outperformed the other models, providing us with 65% accuracy.

The following are confusion matrices and classification reports for each of the models:

**Decision Tree:**

A screenshot of a computer

Description automatically generated with medium confidence

Random Forest

Graphical user interface

Description automatically generated

Support Vector Machine:

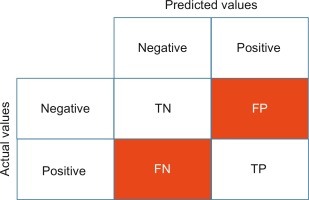
Graphical user interface

Description automatically generated

Since there are 10 classes in our model, the interpretation of precision, recall and f1-score becomes a bit tedious. However, accuracy can be used to measure the performance of each model and compare them side-by-side.

We can see that the accuracy of decision tree is 49%, random forest is 65% and that of support vector machine is 63%.

The way to interpret accuracy is the number of correctly predicted instances of the classes out of all instances. The following diagram is a simple illustration of accuracy for just two classes : Positive and Negative.



Accuracy = True Positive + True Negative / True Positive + True Negative + False Positive + False Negative

Here the accuracy of 65 % in our random forest model means that our model can predict the correct category 65 out of 100 times. So we chose the model that gives us the best predictability towards our 10 classes.

A screenshot of a computer

Description automatically generated

Moreover, we can look at how good the model is in predicting each class individually. The F-1 score (A weighted combination of both precision and recall) is decent across the classes.

To present a better understanding of the model, we can take a look at the ROC Curve :

Chart

Description automatically generated

This curve plots the True Positive Rate against the False Positive Rate. The best model would be something that maximizes the Area Under the Curve. Here we have 10 curves for 10 classes. The Area Under these Curves all appears decent for all classes.

ASSESS THE MODELS:

You must assess the models based on objective or quantifiable means (i.e. compare R-Square values, accuracy assessments, etc.) as well as subjectively based on your goals and objectives. Your objective and subjective judgments may come up with separate models. Explain the strengths and weaknesses of your models. Justify the selected model of choice. Thus,

* Explain your chosen assessment(s) and why you chose to employ it (them). Provide the results of your assessment(s). (6 pts.)
* Discuss the strengths and weaknesses of each model. List out the strengths and weaknesses (6 pts.)
* Justify the choice of your final model(s) (6 pts.)

| **Model** | **Strengths** | **Weaknesses** |
| --- | --- | --- |
| Logistic Regression | Simple and fast to train, making it suitable for large datasets. It can be easily interpreted, making it useful for understanding the relationships between features and the target variable. It can handle both binary and multi-class classification problems. | It may not perform well when the data is highly nonlinear or when there are many features with complex interactions. It assumes a linear relationship between the features and the target variable, which may not hold in some cases. It may be sensitive to outliers and missing data. |
| Decision Tree | Easy to interpret and visualize, making it useful for understanding the decision-making process. It can handle both categorical and continuous features. It can capture non-linear relationships between features and the target variable. | It is prone to overfitting, especially when the tree is too deep or the data is noisy. It may not generalize well to new data if the training set is not representative of the population. It may not perform well when there are many features with complex interactions. |
| Random Forest | Less prone to overfitting than decision trees, thanks to the ensemble learning approach that combines multiple trees. It can handle both categorical and continuous features. It can capture non-linear relationships between features and the target variable. | It is more complex and slower to train than decision trees, especially for large datasets. It may not perform well when there are many features with complex interactions. It may be less interpretable than decision trees because of the ensemble approach. |
| Support Vector Machine | Effective for high-dimensional data, such as text data, because it uses a kernel trick to map the data to a higher-dimensional space. It can handle both binary and multi-class classification problems. It is less prone to overfitting than some other models because of the regularization parameter. | It may be slower to train and less scalable than other models, especially for large datasets. It may be sensitive to the choice of kernel function and other hyperparameters, which may require careful tuning. It may be less interpretable than some other models, such as decision trees. |