**IML Project Report**

**Importing libraries**

We have imported some important libraries needed for the project such as pandas, numpy, matplotlib, seaborn, NLTK(Natural Language Toolkit), sklearn, tensorflow.

**Why are these libraries needed?**

1. **Pandas** = Used to load datasets from CSV, perform data cleaning and manipulate the dataset to prepare it for model training and evaluation.
2. **Matplotlib** = Used to create plots for data visualization of model performance.
3. **Numpy** = Used for data transformation and calculations
4. **Seaborn** = Also used for data visualization. It is a library built on top of matplotlib that provides a high-level interface for creating graphs.
5. **Sklearn** = Used for implementation of machine learning models like logistic regression,etc
6. **Tensorflow** = Used for deep learning models and neural network structures
7. **NLTK** = Used for removing stopwords

pip : It is a package installer for python.

setuptools : It is a python library used to package python projects.

Since our dataset for both train and test data was given in a .tsv format we first converted it into a .csv format using the following code.



**Data loading and visualization.**

We use pandas function **read\_csv** to read our csv file named **sampleSubmission.csv** and the dataframe with the name ‘s’.

For the training file we use train.tsv and save the data with the name ‘s1’. The s1 has 156060 data samples and 4 features. The important features are ‘phrase’ and ‘sentiment’. The phrase feature are the movie reviews with noise and sentiment is similar to class labels. We have a total of 3 class label names ‘1’,’2’ and ‘3’.each represent positive , negative and neutral feelings towards the phrase.

The feature phrases contain many noises in the form of upper cases, punctuation,tags and some words which have no meaning itself such as helping verbs.we have done the following process to reduce the noise to increase our model performance.

1. Lowercasing the strings

We have change the strings in lower case by using the **str.lower function**.it will help

1. Removing HTML tags

We removed the HTML tags from the review phrases, if any are present using the **remove\_html\_tags**.

1. Removing URL

We removed any present URL using **remove\_url** from these reviews because it does not contribute to determining the sentiment of the review and will pose a problem in later stages.

1. Removing punctuation marks

We also removed punctuation marks using **remove\_punc1** as although these help a lot in delivering the gist of the statement when read by a human but not when we’ll apply ML models, so removing any and all not important components are prioritized.

1. Removing stop words

Stop words are common words that often appear in text but carry little meaningful information for certain text processing tasks. E.g the, is, in, and

We removed stop words using NLTK. **stopwords.words(‘english’)** contains a predefined list of English stop words.

**Converting Categorical Data into Numerical Data**

1. **Bag of Words**

**How it works?**

For the document containing the phrases “The cat sat on the mat” and “The dog sat on the log”, it creates a vocabulary of all unique words present in the phrases.

Here these words are ["the", "cat", "sat", "on", "mat", "dog", "log"]

Now each phrase is then represented as a vector, where each element corresponds to the frequency of a word from the vocabulary in that document.

Using the example phrases:

Phrase 1: "The cat sat on the mat."

* + Vector: [2, 1, 1, 1, 1, 0, 0] (since "the" appears twice, "cat" once, etc.)

Phrase 2: "The dog sat on the log."

* + Vector: [2, 0, 1, 1, 0, 1, 1]

Each document is represented as a vector, resulting in a matrix where rows correspond to documents and columns to words in the vocabulary. This matrix is often sparse (many zeros) since most documents will contain only a small subset of the words in the full vocabulary.

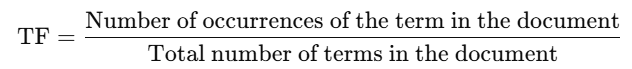
**Limitations**

1. BoW treats the text as a "bag" of words, ignoring grammar and word order, which can lose important context. For example, "not good" and "good" would be represented similarly in BoW.
2. The resulting document-term matrix is often sparse (many zeros), which can be computationally inefficient and require special handling.

2. **TF-IDF(Term Frequency - Inverse Document Frequency )**

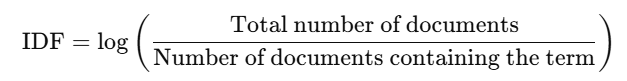
It comprises of two terms TF(Term Frequency) and Inverse Document Frequency(IDF)

1. TF measures how frequently a word appears in a document.



TF captures the importance of a term in a specific document. However, words that are common across documents (like "the" or "and") will have high TF values everywhere, which may not provide meaningful insights.

1. IDF measures how unique or rare a term is across the entire document.



IDF reduces the weight of terms that are common across many documents, like "the" and "is." This helps focus on terms that are unique or specific to individual documents, making them more valuable in differentiating content.

1. The final **TF-IDF score** for a term in a document is obtained by multiplying its TF and IDF values:

TF-IDF = TF x IDF

Each word in each document gets a TF-IDF score, where higher values indicate greater importance of the term in that document relative to the rest of the corpus.

Corpus can be understood simply as a collection of documents.

1. Example of TF-IDF Calculation

Suppose we have three documents:

1. "The cat sat on the mat."
2. "The dog sat on the log."
3. "The cat and dog play together."

Let's focus on calculating TF-IDF for the term "cat" in Document 1.

1. **TF (for "cat" in Document 1)**:
   * "cat" appears once in Document 1, which has 6 words.
   * TF = 1/6 ≈ 0.167
2. **IDF (for "cat")**:
   * "cat" appears in 2 out of the 3 documents.
   * IDF = log⁡(3/2) ≈ 0.176
3. **TF-IDF**:
   * TF-IDF for "cat" in Document 1 = TF × IDF = 0.167×0.176 ≈ 0.0290

Limitation = Like BoW, TF-IDF treats documents as collections of individual terms, disregarding word order and context.

Conclusion = TF-IDF is somewhat better than BoW as unlike BoW, which counts all words equally, TF-IDF reduces the impact of frequent, less-informative words and boosts terms that are unique and relevant to individual documents.

3. **Word2Vec**

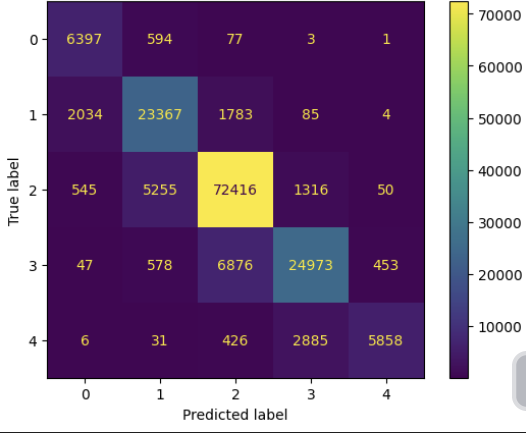
**How it works?**

Word2Vec uses a neural network to learn the vector representations of words in such a way that words used in similar contexts have similar vectors. It can be trained in two main ways:

1. **Continuous Bag of Words (CBOW)**:
   * In CBOW, the model tries to predict a target word based on its surrounding context words. For example, given the context words "The cat on the" the model would try to predict "mat."
   * CBOW is good for smaller datasets and is generally faster than Skip-gram.
2. **Skip-gram**:
   * Skip-gram works in the opposite way: given a target word, it tries to predict the surrounding context words. For example, if the target word is "cat," Skip-gram would try to predict words like "The," "sat," and "on" that appear nearby.
   * Skip-gram is slower but performs better with rare words and larger datasets.

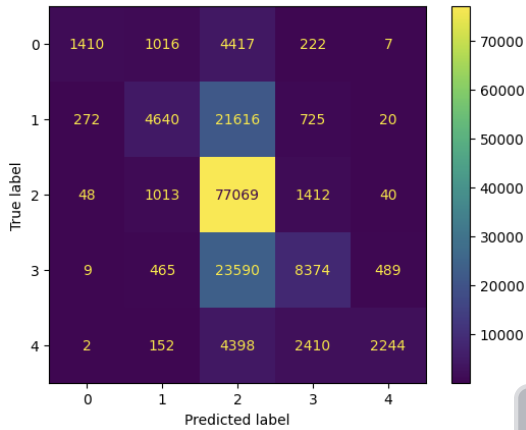
**Model training.**

We have used the following model on our training dataset.

1. Logistic regression- it predicts the probability of a binary event occurring based on a given dataset. As we have only two classes in our dataset we can train this model in our data.after training the model we get accuracy of 0.6822.
2. Decision tree - this model give us accuracy of 0.8523.

As we can see by using its confusion matrix it predicts most of the data correctly.it has predicted 6397 of 0 class data corrected out of total 0 class data. For ‘1’ class the number is 23367, for ‘2’ the number is 72416, for ‘3’ the number is 24973 and for ‘4’ class it is 5858 out of total data for each class label.

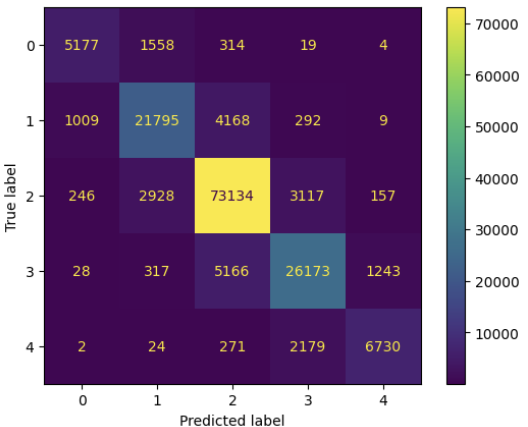
1. XGB classifier- this classifier is good at handling large datasets. The training accuracy by this classifier is 0.6006.

The confusion matrix-

As we can see by using its confusion matrix it predicts half of data correctly.it has predicted 1410 of ‘0’ class data corrected out of total ‘0’ class data. For ‘1’ class the number is 4640, for ‘2’ the number is 77069, for ‘3’ the number is 8374 and for ‘4’ class it is 2244 out of total data for each class label.

1. Random forest classifier-t is a supervised machine-learning algorithm made up of decision trees. It is used for both classification and regression problems.

Training accuracy:0.8522

Confusion matrix: 

As we can see by using its confusion matrix it predicts most of the data correctly.it has predicted 5177 of ‘0’ class data corrected out of total ‘0’ class data. For ‘1’ class the number is 21795, for ‘2’ the number is 73134, for ‘3’ the number is 26173 and for ‘4’ class it is 6730 out of total data for each class label.

5.SVM - it work by mapping data to a high-dimensional feature space so that data points can be categorized easily.

Accuracy: 0.6295

precision recall f1-score support

0 0.56 0.22 0.32 1416

1 0.52 0.40 0.45 5527

2 0.68 0.85 0.75 15639

3 0.56 0.50 0.53 6707

4 0.61 0.22 0.32 1923