Assignment 2: Text Classification

Authors: Jaka Vodlan, Mark Mihelič, Mark Tič

TABLE OF CONTENTS:

DATA PREPROCESSING:

Our initial thoughts were to train the models on short descriptions alone. The file we were given was quite large (aprox. 150k links to different articles). We decided to split the dataset and run it through BERT model that we were focusing on in this assignment. Despite the fact that BERT language model is one of if not the most used language model in the world right now, it still could not get us the desired results we were looking to get. We also could not train the model on all of the 150k cases as this was a very complex task that required a lot of computational power which we did not have.

Our number one priority was to make changes so that the model would run faster and then increase the dataset to get better final results, ideally making out our own cleaned and normalized dataset with scraped text.

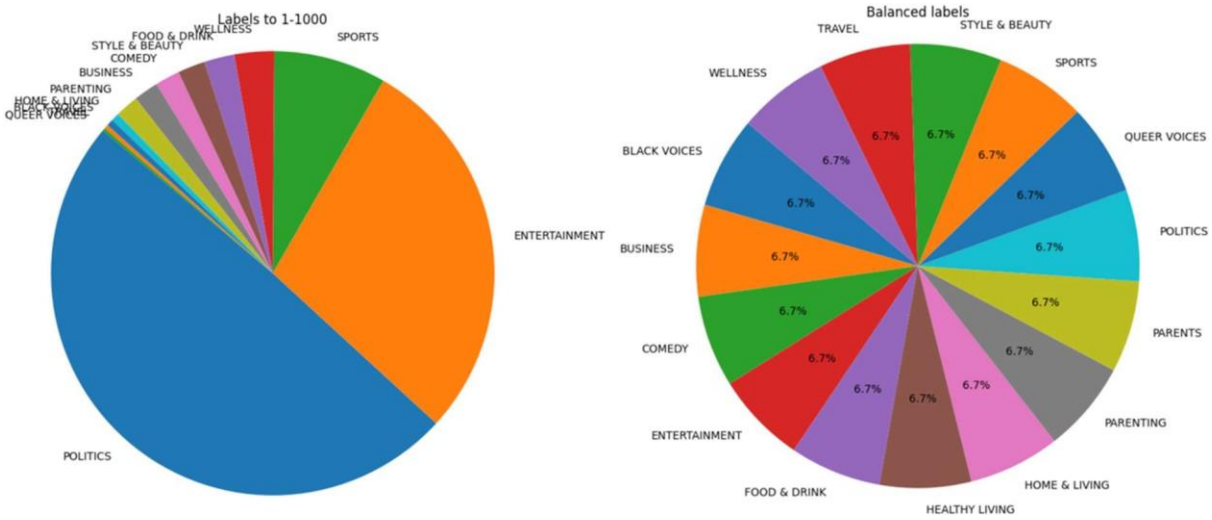
Upon taking a close look at the short descriptions of the given dataset we saw that some of the short descriptions were really lacking in context and even we could hardly guess the category. So we came up with a solution to scrape the data from these websites and train the model on that data which will give us way better results. With the great support for web scraping that Python has, we were able to write some scripts using Beautiful Soup4 (BS4) and extract the relevant data without the headers and other noise. Because scraping with BS4 can be quite time-consuming as it can only focus on one link at a time we also took a look at the python project »Scrapy«. This is a more complex and highly efficient tool that makes web scraping multiple times faster than BS4 but it comes with a number of problems including the need to set up a proxy because of too frequent requests and other stuff.

Our focus was to create a new dataset that would include every category that existed in the initial file and for each and every category we would have approx. the same number of scraped websites. After a few problems like some links being inaccessible and some websites having only two pictures and no text, we managed to put together a well built dataset to train our BERT model on.

To our big disappointment, after all this work we figured out that we do not have the computational power to train our model on that much text. We were experiencing crashes, very long training times and when we lowered the number of cases for each category the accuracy was not as good as we hoped for, in fact, it was worse than when we trained it on short descriptions.

Still, we decided to improve our initial dataset, first by adding the article's headline to the short description and then use this to train our models. The headline provided a few additional words as well as repeat the most important words from the article/short description. Another improvement we had to implement was to balance out the different categories in the dataset. While evaluating BERT we noticed that the confusion matrix does not show the diagonal of »true positives« like we expected. To even out the training dataset, we took an equal number of short descriptions belonging to each category. More on that in the »Results« section.

Down below are the graphs that show how we balanced out the differences in frequencies of the different categories.



In order to clean the data, we removed all punctuation and special signs from the text, such as dots and commas. We also removed some words which are most common in the English language e.g. “don’t”, “I”, “in”, “the”, … This way the model can learn to only take into account words which are connected to the article and therefore the category itself.

To analyze our dataset a bit further here is the word cloud of all words in the dataset after we removed the punctuation and unnecessary words.

With a word cloud we can easily see what words are common in a dataset, but to see what words come up most often in articles of different categories we created word clouds for each of them.

Words in the word clouds are a great representation of what makes the model choose one category over the other.

DETAILS OF TEXT CLASSIFICATION MODELS:

The primary model we were focusing on, is a BERT-based model that we employed using the `bert-base-uncased` pre-trained model. This model consists of a tokenizer, BERT model for sequence classification, and additional layers for customization. Besides BERT model we also implemented the next 7 models.

IMPLEMENTED MODELS:

1. **Logistic Regression Model**: Logistic Regression is a statistical model used for binary classification problems, where the output variable is categorical and has two classes (e.g., 0 or 1, true or false, positive or negative). Despite its name, it is used for classification rather than regression.

   Best hyperparameters: “C = 10” - "C" is the inverse of the regularization strength. Regularization is a technique used to prevent overfitting by penalizing large coefficients in the model.
2. **Random Forest**: is an ensemble learning method that is used for both classification and regression tasks. It builds multiple decision trees during training and outputs the average prediction (regression) or the mode of the predictions (classification) of the individual trees.

   Best hyperparameters: “max_depth': 50, 'n_estimators': 150” - n_estimators parameter represents the number of decision trees that will be built in the ensemble.
3. **K-Nearest Neighbors (KNN):** is a simple and intuitive machine learning algorithm used for both classification and regression tasks. It is a type of instance-based learning, where the model makes predictions based on the majority class or average of the k-nearest neighbors in the feature space.

   Best hyperparameters: “n_neighbors = 10” determines the number of neighbors used for making predictions for a given data point

4. **Support Vector Machine (SVM):** is a supervised machine learning algorithm used for both classification and regression tasks. SVM is particularly effective in high-dimensional spaces and is well-suited for tasks where there is a clear margin of separation between classes.

   Best hyperparameters: "C = 1, kernel = linear" best hyperparameters for this SVM model indicate that it is using a linear kernel ('kernel': 'linear') and a regularization parameter C set to 1 ('C': 1). This suggests that the model is giving equal importance to achieving a precise fit and maintaining a margin of separation between classes.

5. **The Naive Bayes model** is a probabilistic machine learning algorithm based on Bayes' theorem, which calculates the probability of a hypothesis given observed evidence. Despite its simplicity, Naive Bayes is effective for text classification and is commonly used in spam filtering, sentiment analysis, and document categorization.

   Best hyperparameters: "alpha = 0,1" In this case, the best hyperparameter suggests that Laplace smoothing with a relatively small value of alpha (0.1) resulted in improved performance during model training

6. **A Decision Tree** is a versatile and intuitive machine learning model that is used for both classification and regression tasks. It makes decisions based on a series of hierarchical, tree-like structures where each node represents a decision based on a feature, and each leaf node represents the output or prediction.

   Best hyperparameters: "max_depth = 20, min_samples_leaf = 1, min_samples_split = 10" the best hyperparameters suggest a decision tree with a maximum depth of 20, allowing for a relatively deep tree, and relatively small values for the minimum samples in a leaf and minimum samples to split, potentially indicating an effort to capture detailed patterns in the training data without overfitting.

7. **The Majority Classifier Model**, often referred to as the baseline or naive classifier, is one of the simplest models used in machine learning. It predicts the majority class for all instances, making it a straightforward benchmark for comparison with more sophisticated models.

For hyperparameter tuning we used GridSearchCV method that performs an exhaustive search over a specified parameter grid to find the best hyperparameters for a machine learning model. This method tries all combinations of hyperparameters specified in the grid and uses k-fold cross validation to evaluate the model's performance.

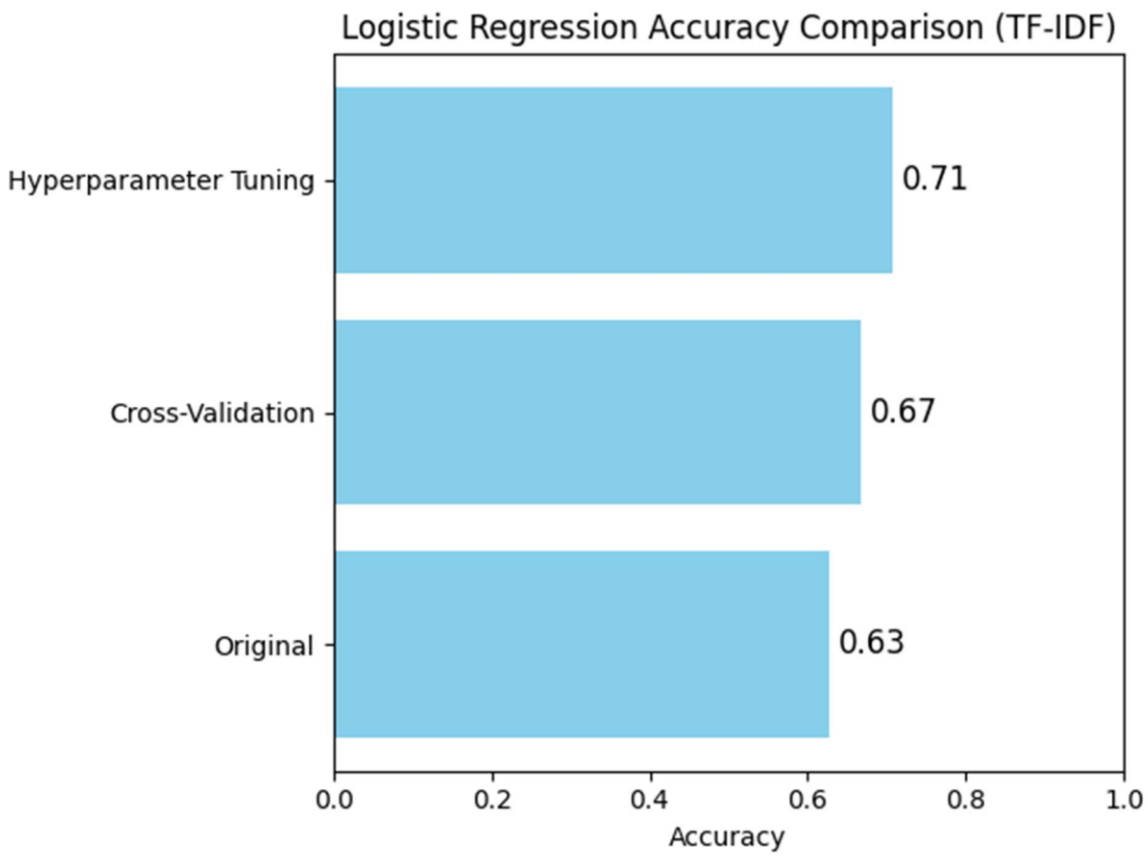
We also tested each of this models using two different vectorizers to see which performs better.

TF-IDF Vectorizer: TF-IDF is a widely used vectorization technique that considers the importance of a term not just within a document but across the entire corpus. It helps in identifying words that are significant to a specific document but not overly common in the entire dataset.

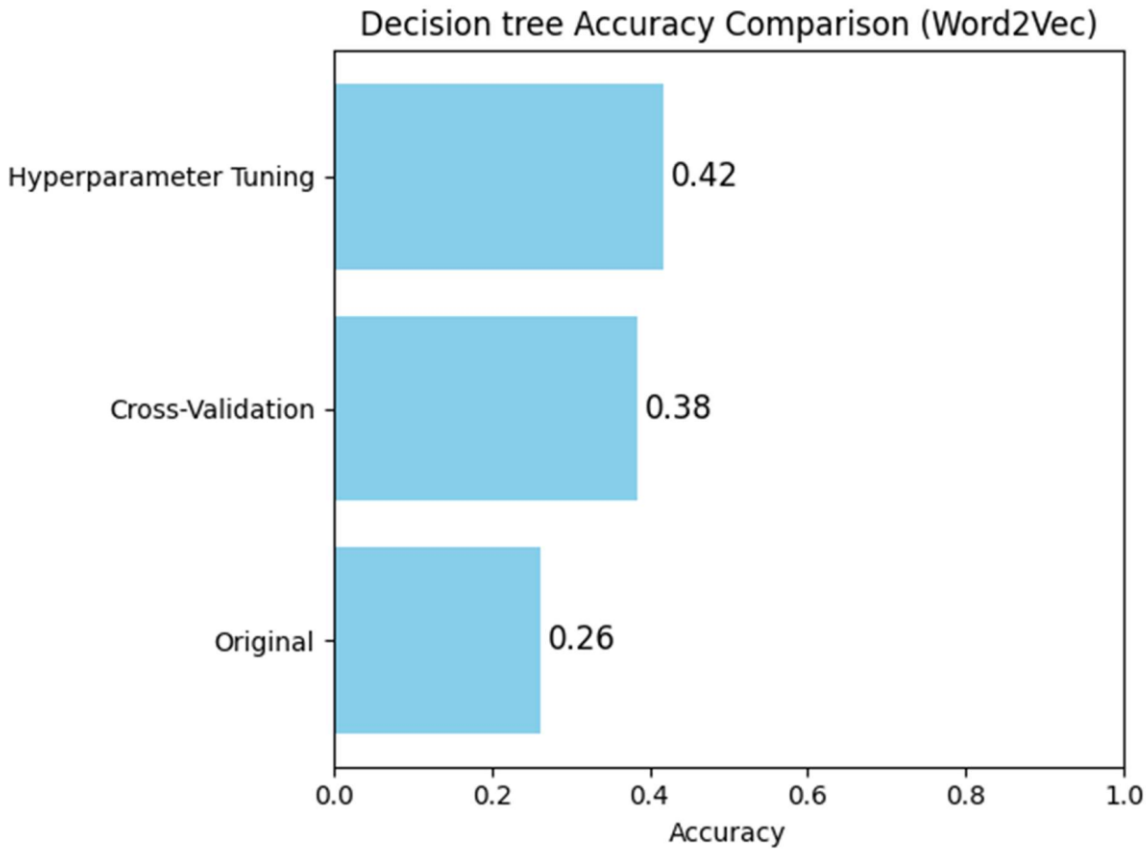
Word2Vec Vectorizer: Word2Vec captures semantic relationships between words and is used to convert words into numerical vectors. The main idea behind Word2Vec is to learn vector representations of words based on their context in a large corpus of text.

RESULTS:

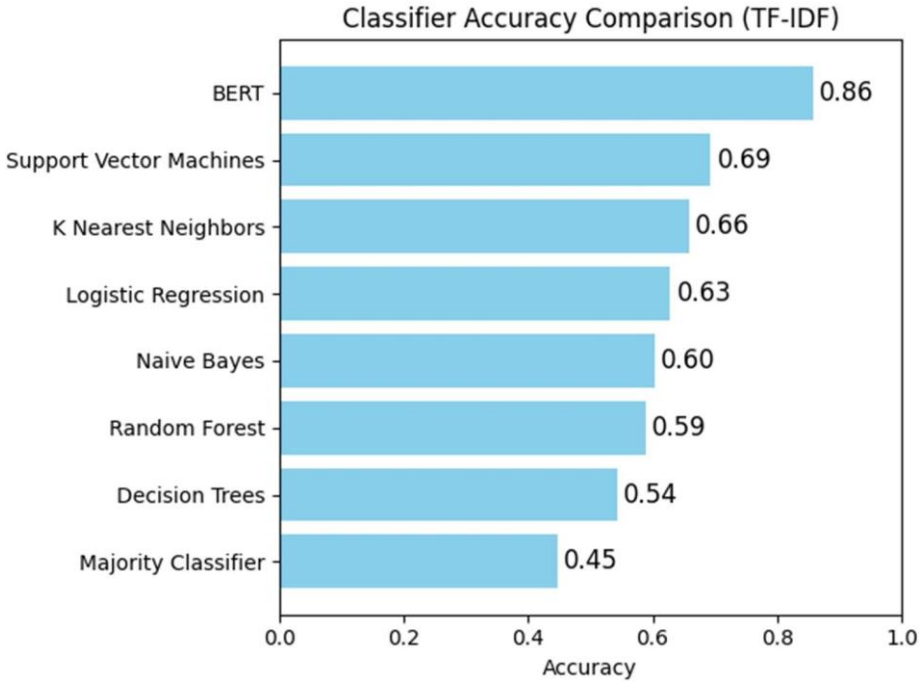
To get a really good view on how good the accuracy is for each model we compared the original accuracy, the accuracy after cross validation and the accuracy after hyperparameter tuning. For example the graph below shows how the accuracy first increases after cross validation and then again after hyperparameter tuning for the Logistic Regression model that used TD-IDF Vectorizer.

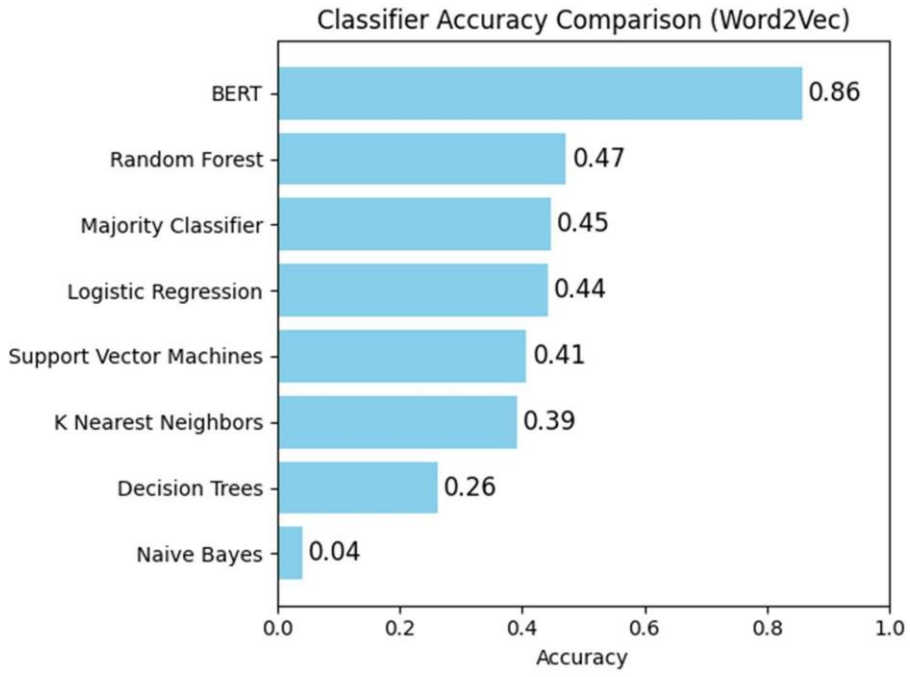


Here is another example of accuracy comparison for the Decision tree model that used Word2Vec Vectorizer.

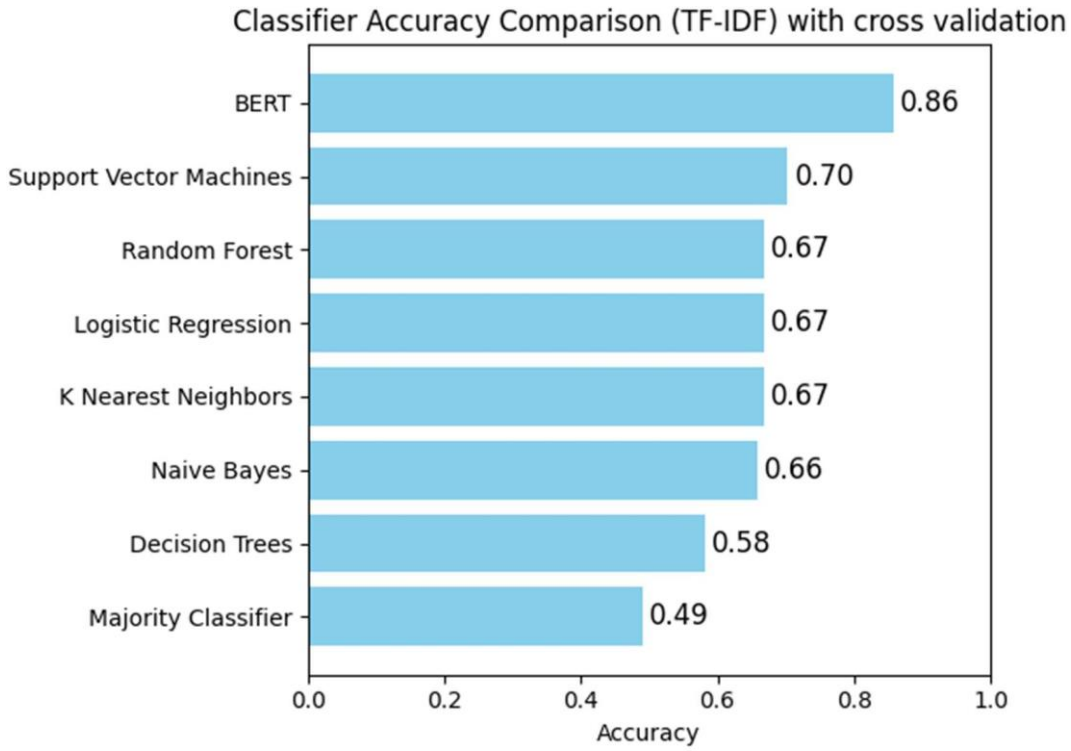


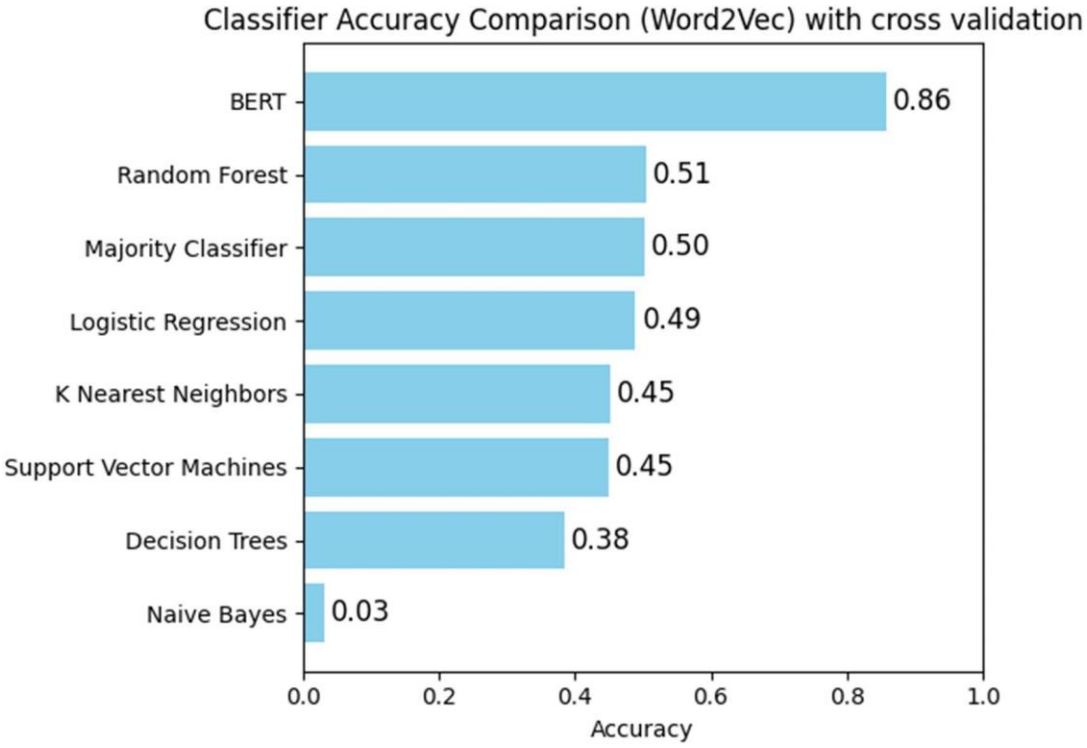
Then we compared accuracies of all the models using 1000 data inputs to see which one is the best. The first image down below shows the original accuracies of the models using TD-IDF Vectorizer in comparison to BERT model and the second one shows the original accuracies of the models using Word2Vec Vectorizer in comparison to BERT model.



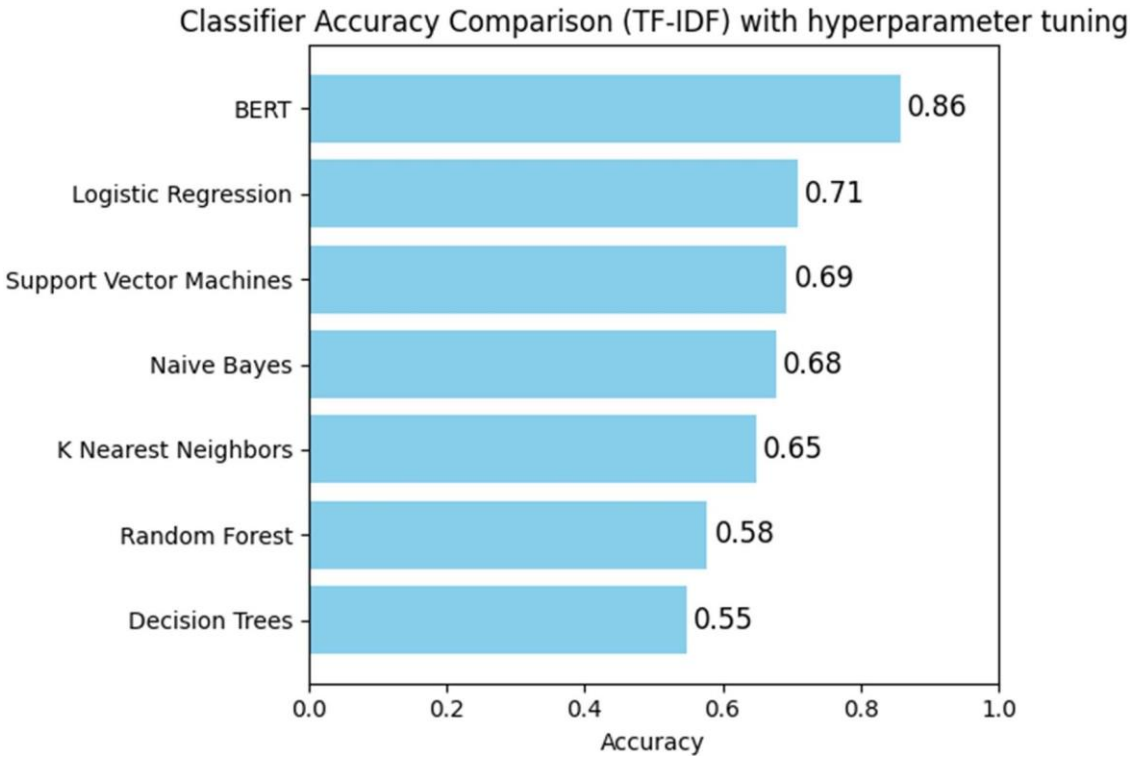


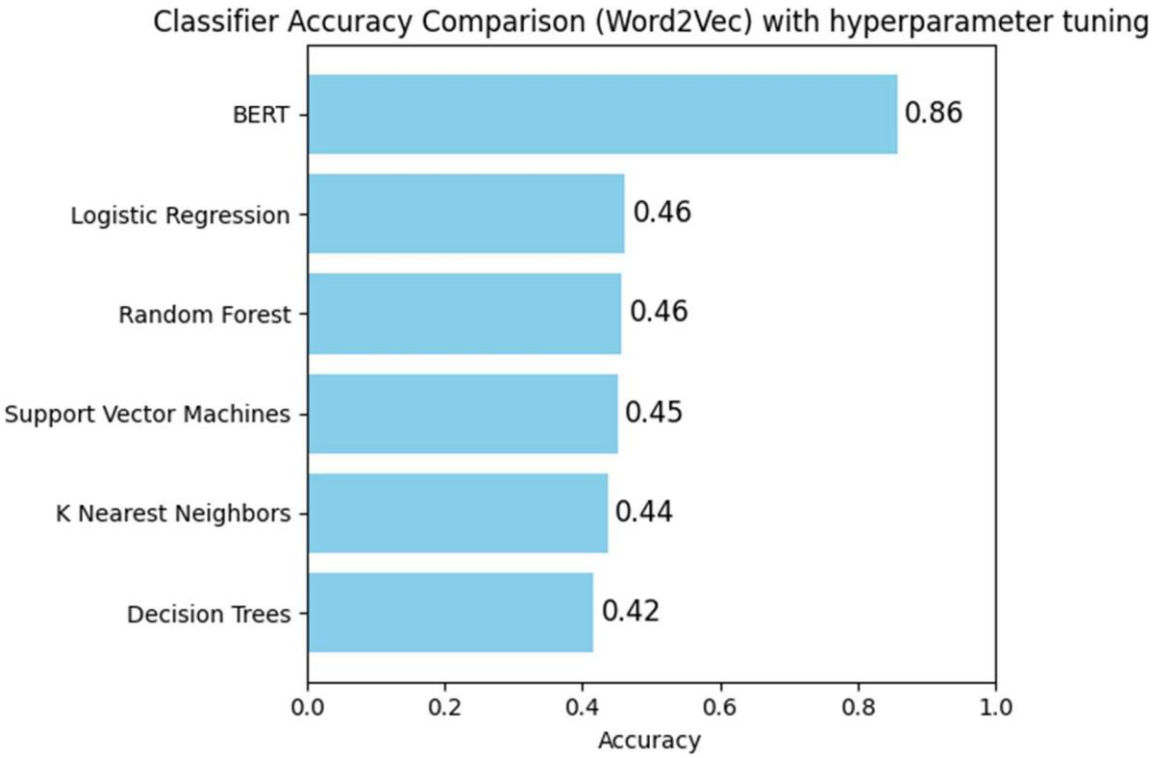
Here is the comparison of the models with implemented cross validation.



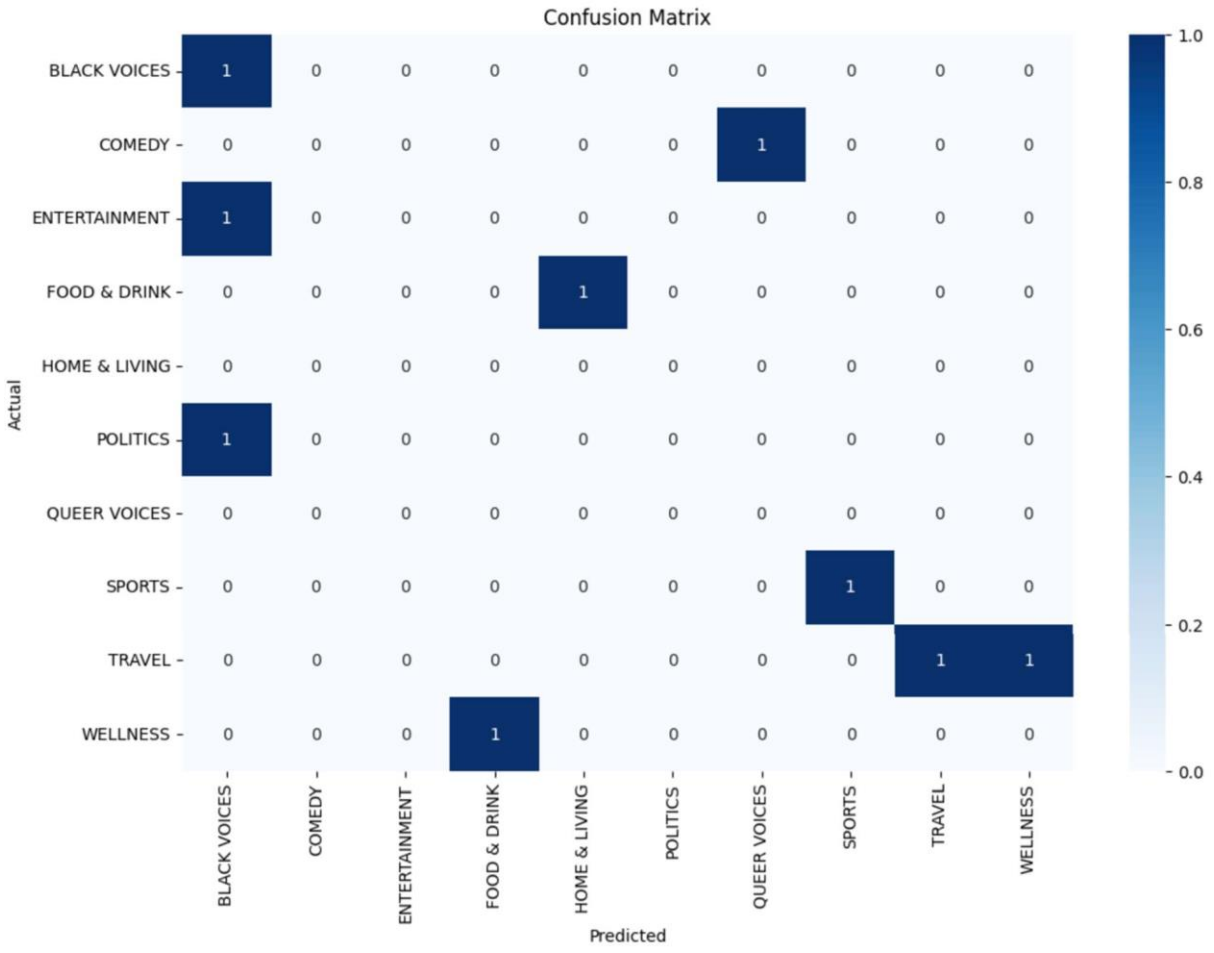


And here is the comparison of the models with implemented hyperparameter tuning.



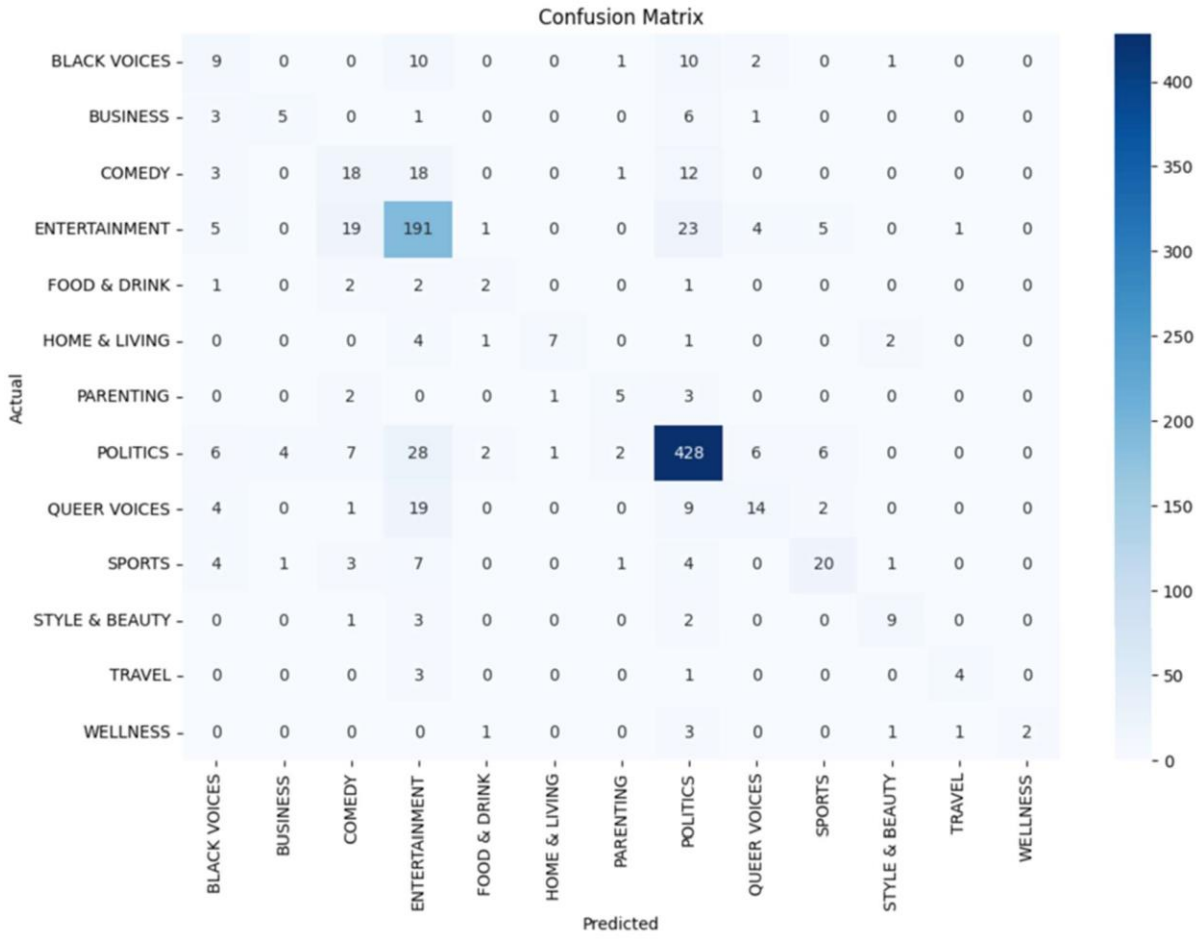


As expected, the BERT model turned out to be better than all the other models in every way. That is why we tested BERT model the most. We tested it on different datasets of different sizes. At first, we ran it through first 100 of the provided short descriptions in the JSON file. Of course, distributed among the train set (80%), evaluation set (10%) and test set (10%). Then we increased the size of the dataset to see how the model would improve. To our surprise the model performed worse on larger datasets, but we later realized why that was the case. Here is our first set of results:



Confusion matrix for BERT (100 samples)

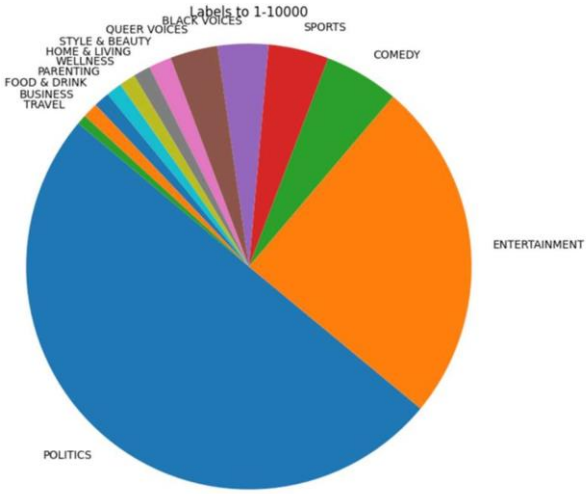
The confusion matrix is as expected. We ran it through 100 samples and the results seem rather random. We increased the number to 1.000 samples and then 10.000 samples.



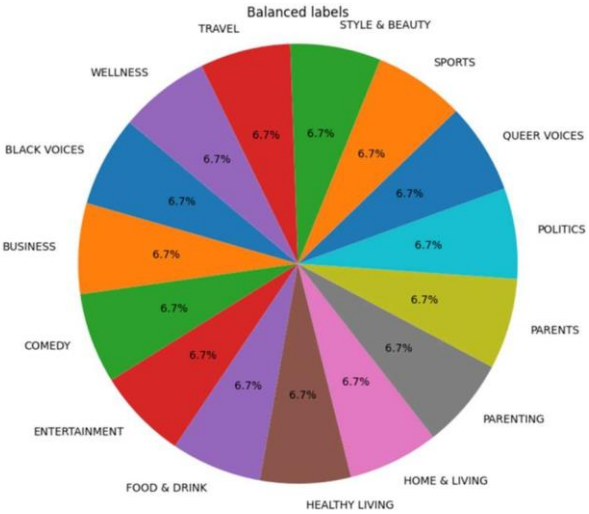
Confusion matrix for BERT (10.000 samples)

The confusion matrix for 10.000 samples was better. It was not random like the one before, but still the problem was that the diagonal, which normally means good results, was not visible.

That was because the “POLITICS” category was so common compared to the others, as seen in the pie charts, that it was all darkened out in the confusion matrix. This is where we got an idea to balance the different categories so that there would be an equal amount of all.

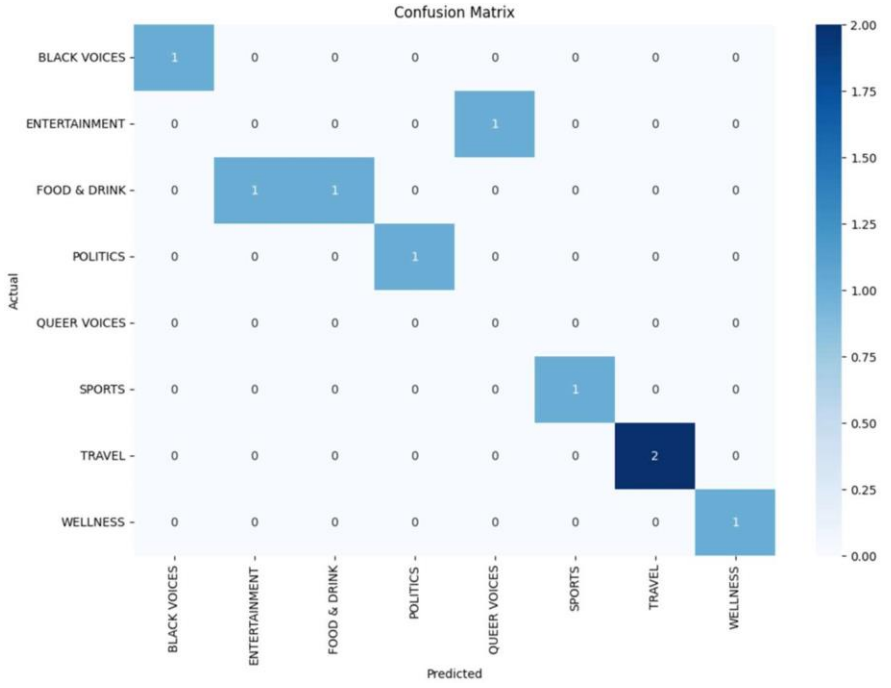


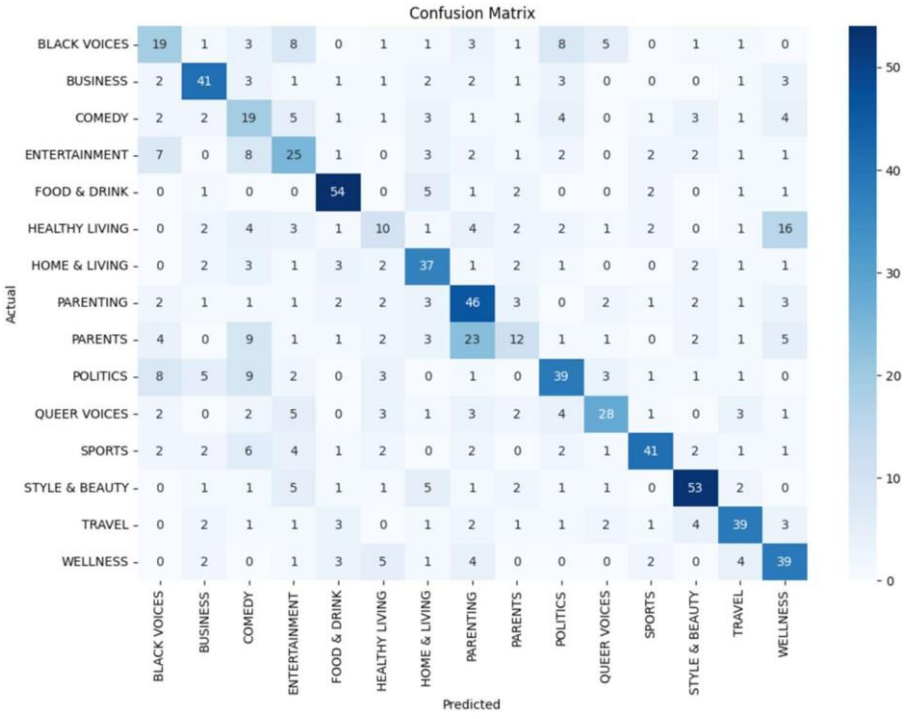
Unbalanced categories



Balanced categories

We ran our BERT model again over the improved dataset and the results were as expected the first time. Here are confusion matrices for 100 and 10.000 samples.





The **accuracy scores** we got, by using different datasets and samples sizes were as following:

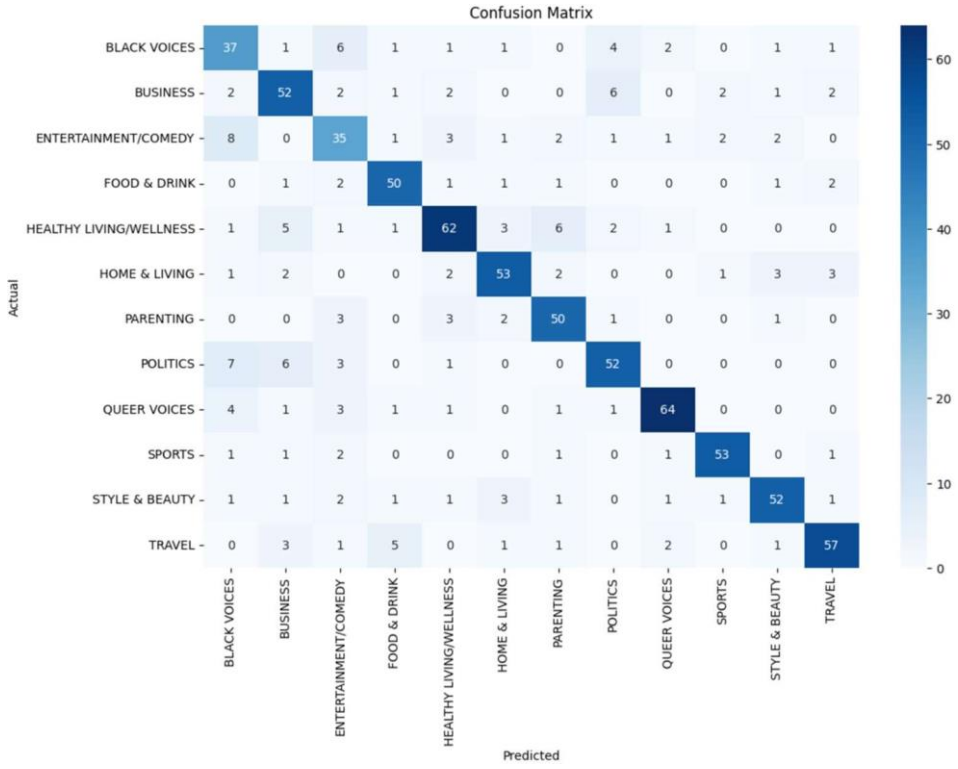
	100 samples	*1.000 samples*	*10.000 samples*
Unbalanced	12.50%	85.86%	73.87%
Balanced	00.00%	52.34%	63.51%

The results of different datasets are as expected. The imbalance of the categories caused the accuracy to increase drastically in the beginning, but then decrease on a larger dataset. It was the opposite on a balanced dataset, since the quantity of the data improves the model accuracy.

To improve our model even further, we decided to “cheat” by combining a few categories that were very similar:

1. PARENTING + PARENTS
2. COMEDY + ENTERTAINMENT
3. HEALTHY LIVING + WELLNESS

These categories seemed so similar, both to us and our BERT model. Even though this model does not officially count, since the categories are not the same, we wanted to see what accuracy we can get by removing some “false positives” from the results. The final accuracy score was 84.24%, which, considering we ran it over a balanced dataset, is a pretty good result. Of course, if we had better computational power and more time, we could test it on larger datasets which would give an even better accuracy score.



Confusion matrix for BERT – balanced dataset and combined categories (10.000 samples)

DISCUSSION:

We expected the models to perform the best when using cross validation and hyperparameter tuning. We also expected BERT to perform the best of them all, especially on larger datasets, since it is the most complex of all the models we implemented. The results more or less met our expectations, except for BERT. The model performed worse on larger datasets, which was strange at first. Later when analyzing the data we realized that certain categories were more common than others, therefore it needed a larger dataset size to get enough samples for less common categories as well.

From all this, we have learned that to train this model you need immense computational power and in our case quantity (more short descriptions) was more important for model tuning than the quality of the text.

In conclusion, our seminar task provided a comprehensive understanding of the critical steps involved in data preprocessing for textual data and introduced various text classification models. From traditional algorithms like Naive Bayes and Decision Trees to advanced models like BERT, we gained insights into how each approach tackles the challenges posed by text data. These lessons equip us with a valuable toolkit to approach real-world problems involving text classification and pave the way for informed decisions in model selection and data preparation.