1 Data representation

The dataset used for this task contained text files organized into five directories, representing distinct classes: business, sport, tech, entertainment, and politics. It was used in its original form.

2 Classification experiment

In order to find the best model for this task, selected classifiers and methods of tokenization and text vectorization were tested. Used functions come from the Python *scikit learn* module. The selected classifiers include:

1. `MultinomialNB()` – Naive Bayes classifier that assumes the features are conditionally independent given the class.
2. `DecisionTreeClassifier()` – classifier that builds a tree-like model to make predictions by evaluating decisions and their potential outcomes, using the values of the features and traversing the corresponding branches in the tree structure.
3. `KNeighborsClassifier()` – classifier that assigns a class to a data point based on the classes of its nearest neighbors, using a predefined number of neighbors to determine the class.
4. `SVC()` – C–Support Vector classifier that finds the optimal hyperplane to separate different classes in a high-dimensional space by maximizing the margin between the classes.

As for the selected tokenization, and vectorization methods, the following were selected:

1. `CountVectorizer()`— is a function that transforms text data into a "bag of words" representation, disregarding grammar and word order. It counts the occurrences of each word and builds a numerical matrix for further analysis and machine-learning tasks.
2. `TfidfVectorizer()` – is a function that converts text data into a numerical representation using the TF-IDF (Term Frequency-Inverse Document Frequency). It creates a matrix where each row represents a document and each column represents a term, capturing the importance of each term in the document collection.

Using the `GridSearchCV()` function, the individual models were cross–validated and their best hyperparameters were selected. The table 1 presents the best parameters for individual classifiers.

Classifier Vectorizer Parameters	Vectorizer	Vectorizer Parameters	Classifier Parameters
MultinomialNB	CountVectorizer	ngram_range: (1, 1), stop_words: 'english'	alpha: 0.1
	TfidfVectorizer	ngram_range: (1, 1), stop_words: 'english'	alpha: 0.1
DecisionTreeClassifier	CountVectorizer	ngram_range: (1, 1), stop_words: 'english'	max_depth: None
	TfidfVectorizer	ngram_range: (1, 1), stop_words: 'english'	max_depth: None
KNeighborsClassifier	CountVectorizer	ngram_range: (1, 1), stop_words: None	n_neighbors: 3
	TfidfVectorizer	ngram_range: (1, 2), stop_words: 'english'	n_neighbors: 5
SVC	CountVectorizer	ngram_range: (1, 2), stop_words: 'english'	C: 1, gamma: 'scale', kernel': 'linear'
	TfidfVectorizer	ngram_range: (1, 2), stop_words: 'english'	C: 25, gamma: 'scale', kernel': 'sigmoid'

Tabel 1: Results of hyperparameters tuning.

Each of the applied vectorizers converts the text to lowercase by default. Setting the `lowercase=False` made the average cross-validation results worse. It seems that for most models the result is positively affected by removing stop words such as "*the*", "*is*", "*and*" by setting the `stop_words='english'` as vectorizers parameters. The table 2 presents the final average cross–validation results for individual models after the selection of hyperparameters.

Model Name	Average Cross–Validation Score
CountVectorizer + MultinomialNB	0.9781
TF-IDF + MultinomialNB	0.9792
CountVectorizer + DecisionTreeClassifier	0.8348
TF-IDF + DecisionTreeClassifier	0.8236
CountVectorizer + KNeighborsClassifier	0.7646
TF-IDF + KNeighborsClassifier	0.9478
CountVectorizer + SVC	0.9719
TF-IDF + SVC	0.9837

Tabel 2: Cross–Validation Scores.

3 Final results

Based on the obtained results, the best model for this task turned out to be the `SVC()` classifier with the use of `TfidfVectorizer()`. Its accuracy was approximately 98%. Then, a test dataset was created to test the best model. The data provided for the task that was used for the training was part of the BBC Dataset which can be found at this link. In order to obtain a test set, the common part of the training dataset and the BBC Dataset was removed. In this way, new texts were obtained that could be classified by the best model. The number of test files for each class is shown in fig. 3.1.

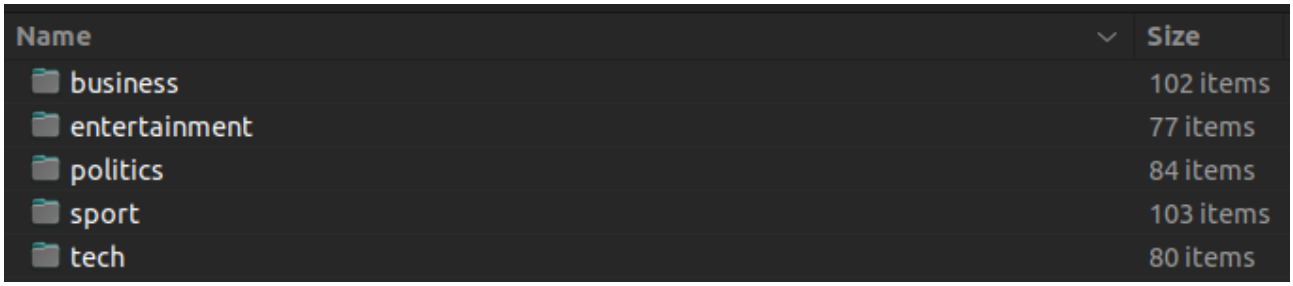

Figure 3.1: Test Dataset Directory.

```
Accuracy: 0.9798206278026906
Classification Report:
               precision    recall  f1-score   support

     business       0.97      0.97      0.97       102
entertainment       1.00      0.96      0.98        77
     politics       0.95      0.99      0.97        84
        sport       0.99      1.00      1.00       103
         tech       0.99      0.97      0.98        80

     accuracy                           0.98       446
    macro avg       0.98      0.98      0.98       446
 weighted avg       0.98      0.98      0.98       446
```

Figure 3.2: Classification Report.

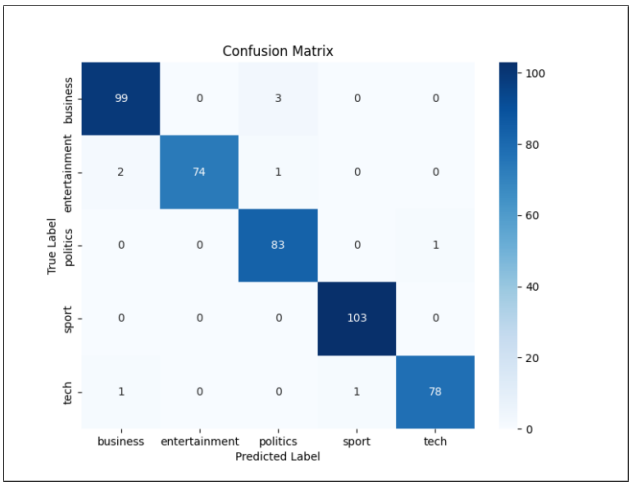

Figure 3.3: Confusion Matrix.

Fig. 3.2 and 3.3 show the results of model prediction on the test dataset. Based on the high accuracy (97.98%), precision, recall, and F1-scores observed across all classes, a conclusion can be drawn that the saved model exhibits strong generalization capabilities on new data and effectively classifies text into the business, entertainment, politics, sport, and tech categories.