Final Submission Paper

Topic:

Classification of the Dataset: One Million Post Corpus

Course:

Natural Language Processing

Department of Mathematics and Computer Science

At Phillips Marburg University

Submitted by Team10:

Tim Luckhardt,	*Mtr. 3142907*
Samuel Becker,	*Mtr. 2847416*
Mehmed Yilmaz,	*Mtr. 3476521*

Lecturers:

Prof. Dr. Lucie Flek

Vahid Sadiri Javadi

Paper submission date

08.08.2022

I. Introduction

For the following Group Project in the Course Natural Language Processing we decided to train a classification Model on the Dataset One Million Post Corpus.
This Dataset contains annotated user posts from the Austrian Newspaper Website "Der Standard".

The objective for this dataset is to train classifiers for each of the nine given labels ArgumentsUsed, Discriminating, Inappropriate, OffTopic, PersonalStories, PossiblyFeedback, SentimentNegative, SentimentNeutral and SentimentPositive.
The text we evaluate is in each case a comment posted by a user on an article in the newspaper, with possibly some other information, described in further detail in chapter 1.1.
These labels are quite diverse: For example, the label SentimentPositive is very imbalanced. The label OffTopic on the other hand needs not only information from the comment, but probably from the newspaper article as well. And the three labels SentimentNegative, SentimentNeutral and SentimentPositive have the property that they're disjunct from each other, i.e. any comment has exactly one of the three labels.

To do this, we first do a correlation analysis between the labels, and try a variety of models for the labels. For example, for the imbalanced labels, we try under- and oversampling methods, for the label OffTopic we try incorporating the article into the model, and in general we try both dense and sparse semantic word embeddings.

All used code and results can be found in the Git under the URL:
https://github.com/TLuckh/GermanArticlesNLP.

Table Of Content

1 Dataset Description

The “One Million Posts” corpus is an annotated data set consisting of user comments posted to the Austrian newspaper website “Der Standard” (in German language).

Der Standard is an Austrian daily broadsheet newspaper. On the newspaper’s website, there is a discussion section below each news article where readers engage in online discussions. The data set contains a selection of user comments from the 12 month time span from 2015-06-01 to 2016-05-31. There are 11,773 labeled and 1,000,000 unlabeled comments in the data set. Only 3599 comments were labeled with all categories, and a total of around 40,000 comments have at least some categories set. The labeled comments were annotated by professional forum moderators employed by the newspaper.

1.1 Dataset Structure

1.1.1 Post Structure

Each comment is structured like the following, the Article ID is a foreign key:

- Post ID
  → 206
- Article ID
  → 4
- Headline (max. 250 characters)
  → “Ja aber: Was hat der Stance….”
- Main Body (max. 750 characters)
  → “Ich bin Rechtshänder, fahre Goofy und kicke mit Rechts. Sie sind ..”
- User ID (the user names used by the website have been re-mapped to new numeric IDs)
  → 10467
- Time stamp
  → “2014-08-13 08:00:52.207”
- Parent post (replies give rise to tree-like discussion thread structures)
  → 203
- Status (online or deleted by a moderator)
  → “online”
- Number of positive votes by other community members
  → 0

- Number of negative votes by other community members
  → 0

1.1.2 Article Structure

For each article, the data set contains the following data:

- Article ID
  → 4
- Publishing date
  → "2014-08-13 05:30:00.00"
- Topic Path (e.g.: Newsroom / Sports / Motorsports / Formula 1)
  → "Newsroom/User/mitmachen/Mitreden"
- Title
  → "Welche Erfahrungen haben Sie als Linkshänder gemacht?"
- Body
  → "<div class="section" … Wie sieht Ihr Alltag als Linkshänder aus? Erledigen Sie manche Arbeiten mit rechts statt mit links? ..."

1.1.3 Data Structure

As the data is from an online newspaper website, in which each article has a comment sections, each comment is related to an article. Furthermore, one can comment either on the article directly, or on another comment.

Therefore, the comments themselves give rise to a tree-like structure.

1.2 Dataset Annotations for user generated content

1.2.1 Potentially undesirable content

- Sentiment (negative/neutral/positive)
  An important goal is to detect changes in the prevalent sentiment in a discussion, e.g., the location within the fora and the point in time where a turn from positive/neutral sentiment to negative sentiment takes place.
- Off-Topic (yes/no)
  Posts which digress too far from the topic of the corresponding article.
- Inappropriate (yes/no)
  Swearwords, suggestive and obscene language, insults, threats etc.

- Discriminating (yes/no)
  Racist, sexist, misogynistic, homophobic, antisemitic and other misanthropic content.

1.2.2 Neutral content that requires a reaction

- Feedback (yes/no)
  Sometimes users ask questions or give feedback to the author of the article or the newspaper in general, which may require a reply/reaction.

1.2.3 Potentially desirable content

- Personal Stories (yes/no)
  In certain fora, users are encouraged to share their personal stories, experiences, anecdotes etc. regarding the respective topic.
- Arguments Used (yes/no)
  It is desirable for users to back their statements with rational argumentation, reasoning and sources.

1.3 Dataset Issues

As the comments in the data comes from online posts, which are written by private persons, there are a number of possible peculiarities to be noted:

- The comments, as well as the articles may contain typos.
- Comments may contain correctly written, but semantically incorrect words, which most likely are the result of a type of auto-correct.
- Some comments contain creative ways of putting emphasis, e.g. Capslock or repeated vocals ("looooooooooooooong").
- Some users are sidestepping curse word filters by replacing letters, for example by stars.
- Comments may contain smileys as arrangement of ASCII-symbols.
  - However, the dataset seems to contain no exotic non-ASCII symbols.
- The articles have a complex HTML-structure. The comments have no HTML-Tags or more complicated structure, though maybe links.

The distribution of True-False instances for some labels is very imbalanced, with the most extreme example being SentimentPositive, as seen in the figure below.

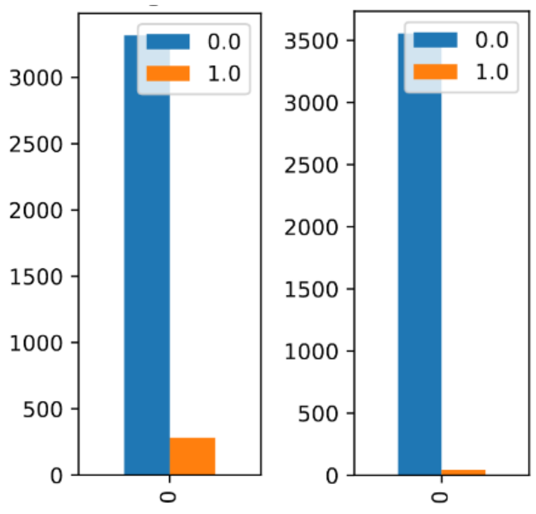
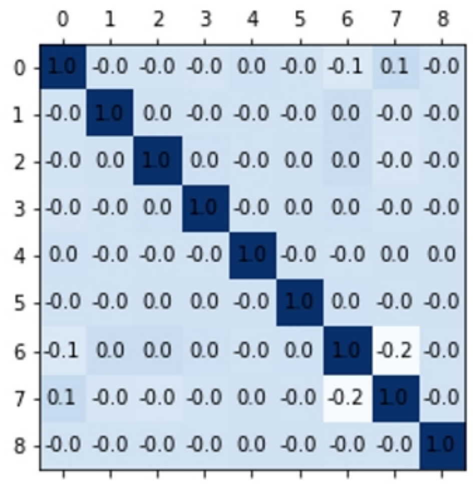


Figure 1 Label distribution – Discriminating, SentimentPositive

For most of these imbalanced labels it is necessary to look at special strategies like under- or oversampling to avoid skewed results in the classifiers.
For the shown extreme case of SentimentPositive however, it might prove impossible to build a working classifier at all, as it has only 43 positive instances.

2 Stochastic Analysis

The figure below shows the Correlation Matrix between features:



	0	1	2	3	4	5	6	7	8
0	1.0	-0.0	-0.0	-0.0	0.0	-0.0	-0.1	0.1	-0.0
1	-0.0	1.0	0.0	-0.0	-0.0	-0.0	0.0	-0.0	-0.0
2	-0.0	0.0	1.0	0.0	-0.0	0.0	0.0	-0.0	-0.0
3	-0.0	-0.0	0.0	1.0	-0.0	0.0	0.0	-0.0	-0.0
4	0.0	-0.0	-0.0	-0.0	1.0	-0.0	-0.0	0.0	0.0
5	-0.0	-0.0	0.0	0.0	-0.0	1.0	0.0	-0.0	-0.0
6	-0.1	0.0	0.0	0.0	-0.0	0.0	1.0	-0.2	-0.0
7	0.1	-0.0	-0.0	-0.0	0.0	-0.0	-0.2	1.0	-0.0
8	-0.0	-0.0	-0.0	-0.0	0.0	-0.0	-0.0	-0.0	1.0

0. ArgumentsUsed
1. Discriminating
2. Inappropriate
3. OffTopic
4. PersonalStories
5. PossiblyFeedback
6. SentimentNegative
7. SentimentNeutral
8. SentimentPositive

As one can see, there is little to no correlation between the features, with the exception of SentimentPositive and SentimentNeutral, in which case the shown negative correlation is expected, as they are disjunct labels.
One puzzling aspect in this matrix is that there should be a negative correlation between all three Sentiment-labels, which for some reason unknown to us is not given.

The correlation matrix can be obtained by running the Jupyter notebook "OneMillionCorpusNLP_Correlation_Matrix.ipynb".

3 Evaluation methodology

To train the classifiers that can predict the categories of comments mentioned in Chapter 1.2, we generally split the dataset into training (70%) and test (30%) data.
Because of data imbalance, we will not use accuracy for performance measurement. Instead, we will use F1-Score as main evaluation criteria for the performance of a classifier.

Trained Models are further described in Chapter 3.

4 Preprocessing of comments and used features

Since both the headline of a comment, as well as the comment body both are highly relevant for the label, we fused headline and comment to a single attribute "comment".

We then transformed this attribute into a word embedding by taking for each word its word embedding, and averaging over all words.
We tried both the dense semantic word embeddings from spacy as well as a sparse word embedding obtained by applying tf-idf on the training data.

For the dense word embeddings, we mostly used the attribute comment as is, while for the tf-idf embedding we first lemmatized the words and removed punctuation.

5 Trained Classifiers and Results

We have used different classifiers to get a performance overview and choose the best classifier. Both the Support Vector Column classifier and the Feedforward Neural Network get their own chapter, as we employed many different hyperparameter combinations for the classifiers, as well as specifically compared the results of taking, or not taking, the data imbalance into account.
Afterwards, we give a chapter in which we discuss the different alternative features, how they compare to each other, and which feature set should be used.

5.1 Support Vector Column

We have tried a number of different kernels for the SVC classifier: Linear kernel with primal formulation, linear kernel with dual formulation, the RBF Kernel as well as a selection of polynomial kernels.
The training results, i.e. the best performing SVC classifier when using the en_core_web_md pipeline from spacy, and not accounting for data imbalance can be seen in the document "Training Results ..._md, unweighted SVC".
The document in turn is generated by the jupyter notebook "OneMillionCorpusNLP_only_SVC.ipynb".
We then tried the same classifiers again, trying instead the en_core_web_lg pipeline. However, even though the pipeline employs a much larger set of words for which dense emebddings are given, changing the pipeline didn't improve the classifiers performance. As such, we didn't generate a result document for this. If one wants to check the results anyway, one has to change the used pipeline in the last mentioned Jupyter notebook to "en_core_web_lg".

We then tried taking the imbalance into account, the results of which can be seen in the document "Training Results ..._lg, weighted SVC". For most imbalanced labels taking the imbalance into account significantly improved the classifier performance, e.g. for the ArgumentsUsed kernel when employing the RBF_Kernel, the F1-score went from 0.53 to 0.63.

In general, the SVC classifiers performed rather well (compared to the other classifiers we tested), with the RBF-Kernel being in most cases the most performant one.

5.2 Feed Forward Neural Network

All punctuation marks and numbers of the MLPClassifier (sklearn) input were removed. The result was then vectorized with the vectorizer from spacy. The "de_core_news_lg"-model was used for the vectorizer. To get a higher accuracy in some cases the "Status" and/or the "Number of positive votes" and "Number of negative votes" were also appended to the vector.
The iterations of the classifier were increased to 5000, because before the classifier needed in some cases more iterations for the best result. Changed to the stochastic gradient descent solver, because the result was better. The learning rate was set to "adaptive" and the "learning_rate_init", the "momentum", as well as the "n_iter_no_change" variable, were slightly adjusted. The best parameters were determined by testing and using the f1 score.

To avoid the imbalance of the data, under- and oversampling was used. However, this did not lead to improvements in all cases. Undersampling led most often to an improvement, in 5 of 9 cases. Oversampling was only an improvement in one case and for the rest the standard data set led to the best results.
The code to recreate the results can be found in the "OneMillionCorpusNLP_MLPClassifier.ipynb"-file. The parts "Create Posts_Annotated", "Create Posts_Annotated_combined" and "Create Dataframe for Test" are used to generate the input data for the classifier. In "Classifier Parameter Test" there are most of the test to get the best parameters. In the last part "Test Classification" is the code for testing different input vectors and under- as well as oversampling. If it's run without changes, the best results for each category will be generated.

5.3 k-Nearest Neighbour

Due to its simplicity in training and its robustness to imbalanced data, we also tested the performance of the k-Nearest Neighbour classifier for our data.
The results of training it can be seen in "Training Results ..._md, k-Nearest-Neighbors", and be generated using the Jupyter notebook "OneMillionCorpusNLP_md_kNeighborsVote".

However, there was no category for which k-Nearest-Neighbour was competitive with the best results from the other classifiers.

5.4 Alternative/Improved Feature-Sets

As mentioned in chapter 4, we trained the classifiers once with dense word embeddings as given by the spacy pipeline, and once with the sparse word embeddings given by the tf-idf-vector of the training data split.
The performance for most used classifiers using this tf-idf-vector can be found in "Training Results .._md, partially weighted, tf-idf.pdf". The .pdf in turn can be produced by running the Jupyter notebook "OneMillionCorpusNLP_tf_idf.ipynb".
However, while the sparse word embedding in some cases, e.g. for the labels ArgumentsUsed and PersonalStories, didn't perform bad, the dense word embeddings as given by the spacy pipeline consistently outperformed the sparse word embeddings.
Therefore, from now on we generally use the dense word embeddings as part of the feature set.

For the label OffTopic, we tried adding a similarity measure between post and comment into the feature set. For this we first cleaned up the article by removing the HTML parts and turning the article into a series of paragraphs. We then calculated, for each paragraph of the article, the similarity of the paragraph to the comment (as angle of the word embeddings for each), and then added the highest computed similarity as a feature.
The effect of adding this feature on the classifier performance can be seen in the document “Best_lg_using_article_OffTopic”, which in turn can be generated using the Jupyter Notebook “OneMillionCorpusNLP_using_Article.ipynb”.
Doing so improved the F1-score from 0.29 to 0.33 compared to only using the dense word embeddings.

Finally, we tried taking a few of the meta-features into account, namely whether the post had been deleted, and the number of up- and downvotes of the post (as relative amount of the total number of votes).
The results of this can be seen in “Best_lg_with_Votes_PostStatus”, and can be generated by the Jupyter notebook “OneMillionCorpusNLP_preprocessed.ipynb”. However, take note that to succesfully run the notebook, one has to have run the notebook “NLP_Data_Preprocessor.ipynb” at least once.
Adding these meta-labels improved performance of the best classifier for most labels. For example, the F1-Score for the label Discriminating improved from 0.21 to 0.26.

5.5 Best performing Classifier per Label

When comparing the results of each classifier for each label we get following Table of best performing classifier per label. The Results can be viewed in “7.1 Best Results”. The following results all use as features the dense word embedding given by spacy together with the meta-label whether the post was deleted, as well as the up- and downvotes.

<u>**Label**</u>	<u>**Classifier**</u>	<u>**F1-Score**</u>
ArgumentUsed	SVC – RBF Kernel	0.648
Discriminating	Log. Regression – Balanced	0.262
Inappropriate	SVC – Primal	0.252
OffTopic	SVC – RBF Kernel	0.353
PersonalStories	MLP – Oversample	0.745
PossiblyFeedback	SVC – RBF Kernel	0.649
SentimentNegative	SVC – Dual	0.629

SentimentNeutral	Gradient Boosting Trees	0.647
SentimentPositive	SVC – RBF Kernel	0.108

SVCs perform best for 6 out of 9 labels, although it is not one specific SVC-Kernel but different SVC-Kernels. From these different SVC-Kernels, the RBF-Kernel accounts for 4 out of 6 labels.

6 Conclusion

Due to the large imbalance in the Dataset and the, depending on the category, quite diverse reasons for labelling individual posts as positive or negative instances, the parameter tweaks in the classifiers could not improve the results by a large margin.
Although our performance tweaks did not boost the results by large, we were able to achieve a F1-Score above 0.6 for 5 of 9 labels, where the other 3 labels were too imbalanced for any learning to take place. This separates labels in simple and difficult labels with the 3 difficult labels being “Discriminating”, “Inappropriate” and “OffTopic”.
For these labels the classifiers might benefit from knowing the full parent comment thread as well as the parent post. Especially the prediction of the label OffTopic slightly benefited from the inclusion of the parent comment thread. However, knowing the parent comment thread did not provide any significant improvement and we were unsure on how to use the tree structure to improve the classifier performance for the labels “Discriminating” and “Inappropriate”.

The hardest label for any classifier to predict is the label “Discriminating”, while the label “SentimentPositive” is hardly trainable at all given the lack of positive instances.

In conclusion there is not one single best performing classifier. While with sparse embeddings SVC with Primal and Dual Kernel gave best results, with dense embeddings SVC with RBF-Kernel and other classifiers gave best results

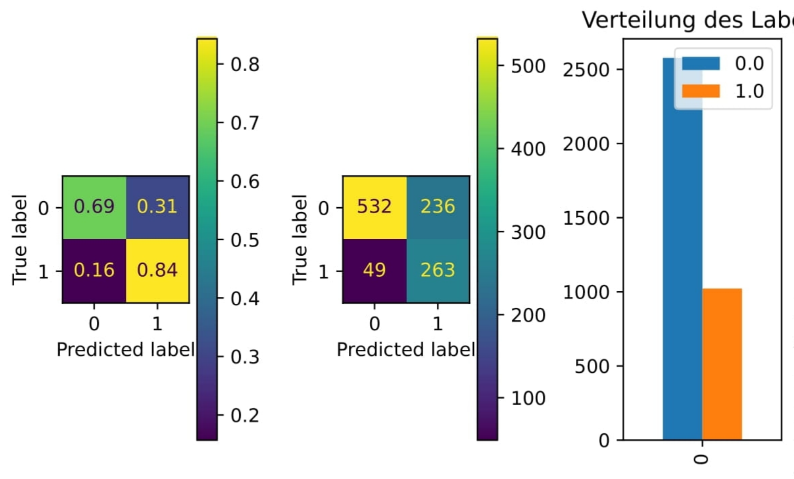
Finally, our results largely match with the findings of the analysis done by Schabus et al. in their paper “One Million Posts: A Data Set of German Online Discussions”.

7 Results

7.1 Best results

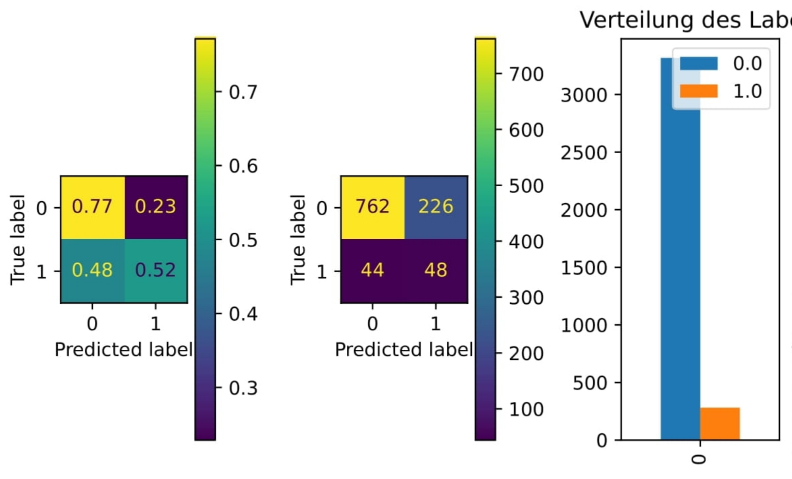
In the following are the best results we got with our classifier. The pictures are also available in the “Best_lg.pdf”-file.

Label: ArgumentsUsed ; Model: Support_Vector_Column_RBF_Kernel



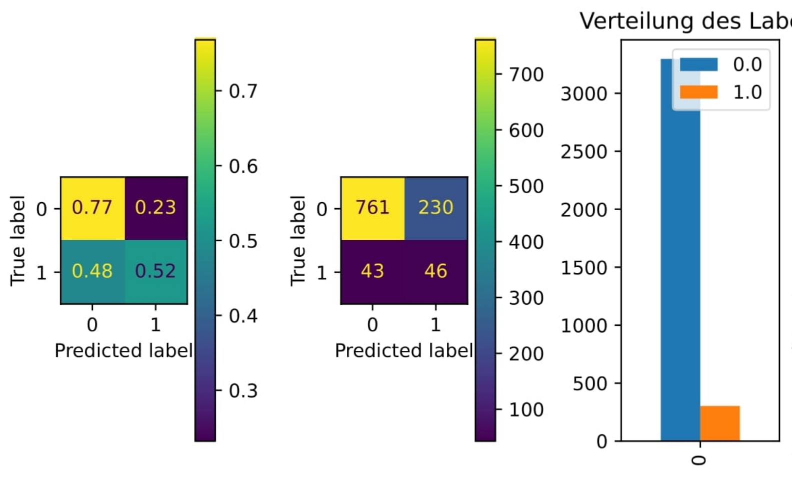
```
F1_Score: 0.6485819975339087
Accuracy: 0.7361111111111112
Precision: 0.5270541082164328
Recall    : 0.842948717948718
Instances of value 0: 2577
Instances of value 1: 1022
```

Label: Discriminating ; Model: Log_Regression_Balanced



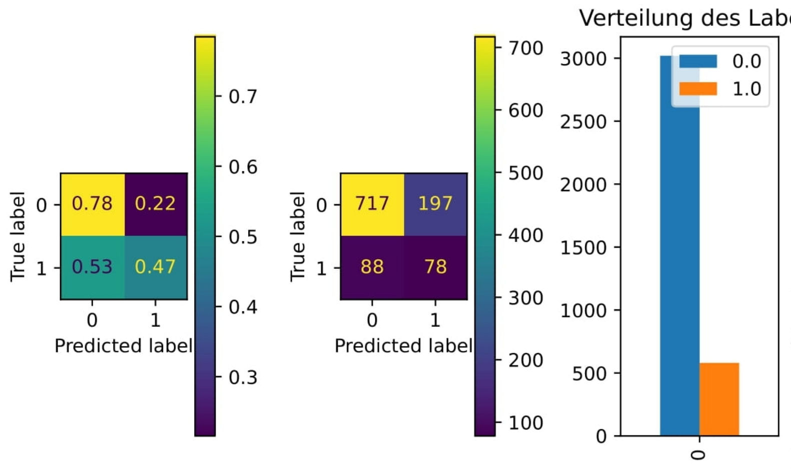
```
F1_Score: 0.26229508196721313
Accuracy: 0.75
Precision: 0.17518248175182483
Recall    : 0.5217391304347826
Instances of value 0: 3317
Instances of value 1: 282
```

Label: Inappropriate ; Model: Support_Vector_Column_Primal



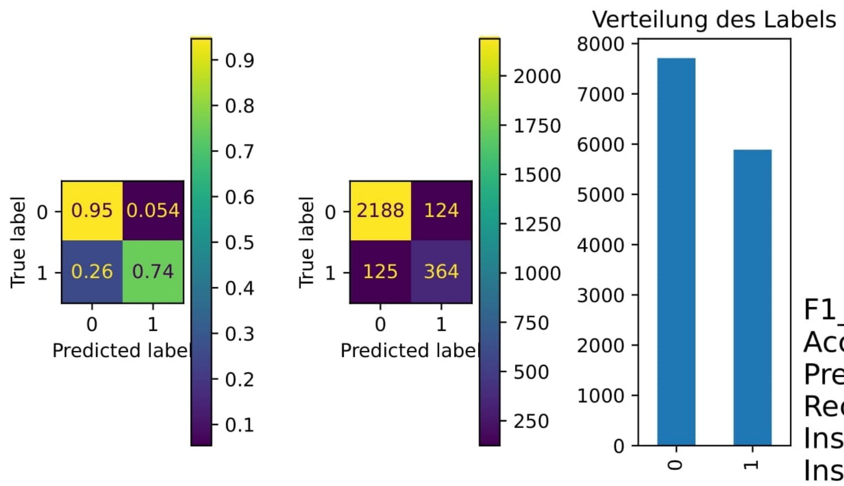
```
F1_Score: 0.25205479452054796
Accuracy: 0.7472222222222222
Precision: 0.16666666666666666
Recall    : 0.5168539325842697
Instances of value 0: 3296
Instances of value 1: 303
```

Label: OffTopic ; Model: Support_Vector_Column_RBF_Kernel



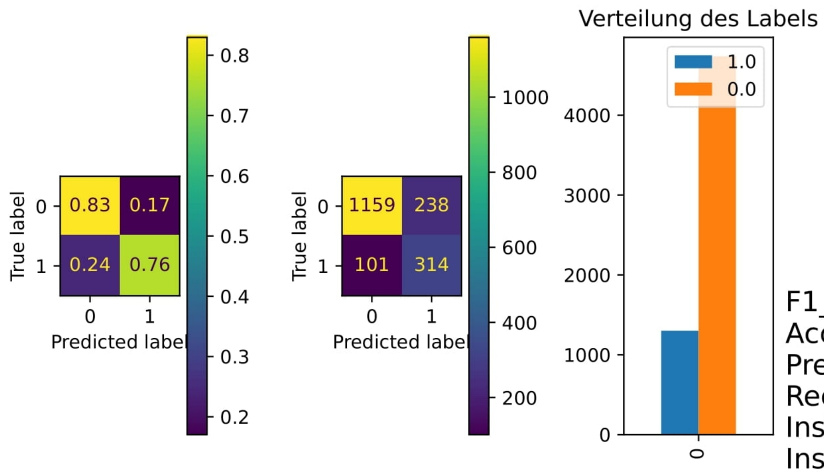
F1_Score: 0.35374149659863946
Accuracy: 0.7361111111111112
Precision: 0.28363636363636363
Recall : 0.46987951807228917
Instances of value 0: 3019
Instances of value 1: 580

Label: PersonalStories | Status; Model: MLPClassifier | Oversample

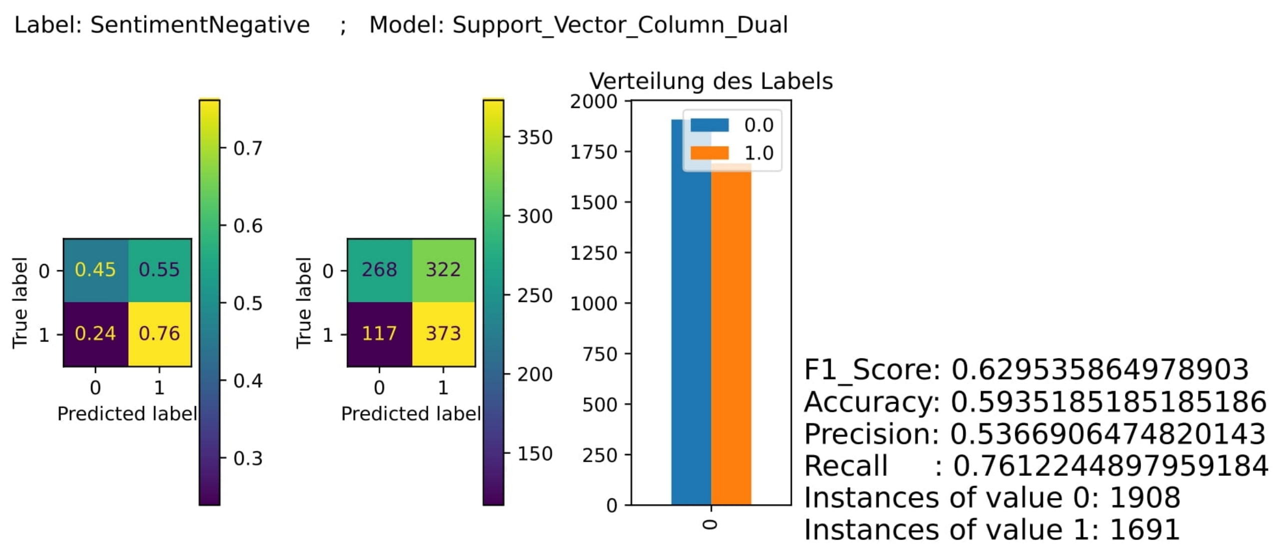


F1_Score: 0.7451381780962129
Accuracy: 0.9111031774366298
Precision: 0.7459016393442623
Recall: 0.7443762781186094
Instances of value 0: 7711
Instances of value 1: 5888

Label: PossiblyFeedback ; Model: Support_Vector_Column_RBF_Kernel

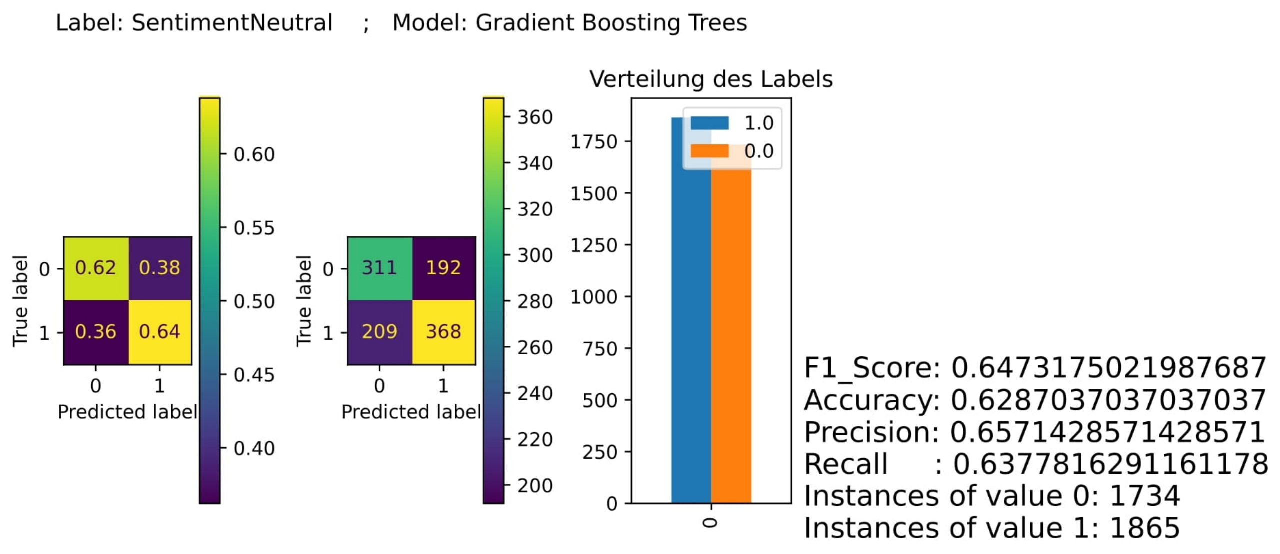


F1_Score: 0.6494312306101345
Accuracy: 0.8129139072847682
Precision: 0.5688405797101449
Recall : 0.7566265060240964
Instances of value 0: 4737
Instances of value 1: 1301



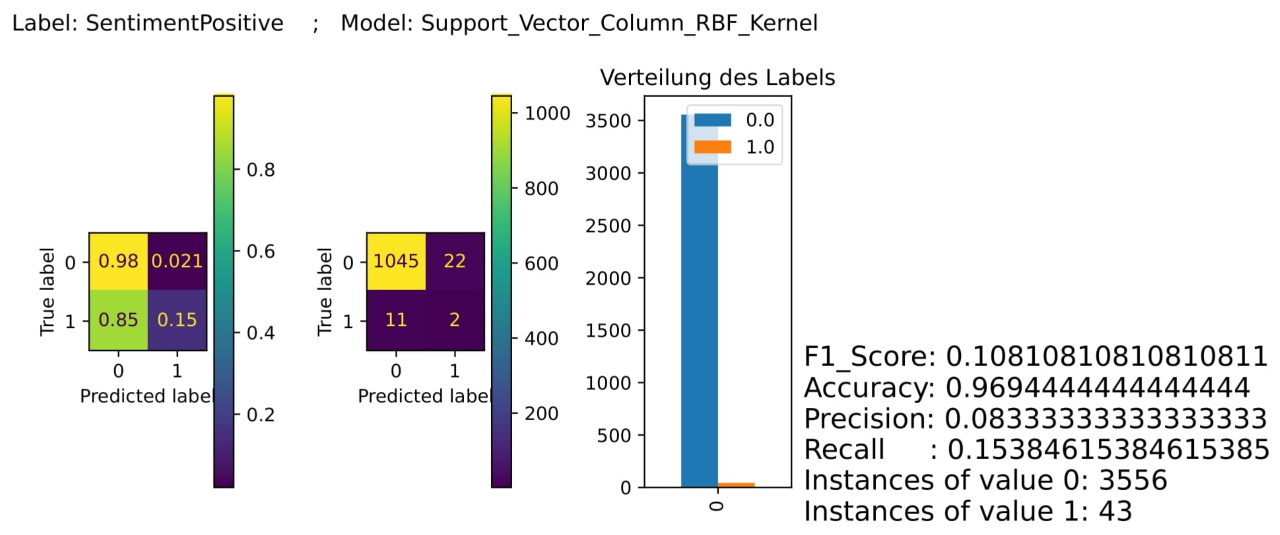
Label: SentimentNegative ; Model: Support_Vector_Column_Dual

```
F1_Score: 0.629535864978903
Accuracy: 0.5935185185185186
Precision: 0.5366906474820143
Recall    : 0.7612244897959184
Instances of value 0: 1908
Instances of value 1: 1691
```



Label: SentimentNeutral ; Model: Gradient Boosting Trees

```
F1_Score: 0.6473175021987687
Accuracy: 0.6287037037037037
Precision: 0.6571428571428571
Recall    : 0.6377816291161178
Instances of value 0: 1734
Instances of value 1: 1865
```



Label: SentimentPositive ; Model: Support_Vector_Column_RBF_Kernel

```
F1_Score: 0.10810810810810811
Accuracy: 0.9694444444444444
Precision: 0.08333333333333333
Recall    : 0.15384615384615385
Instances of value 0: 3556
Instances of value 1: 43
```

7.2 All Results

The results of all classifier for all categories can be found in the “Training Results XXX.pdf”-files. Where the XXX stands for some special settings and/or the classifier.