NLP - Project Report

Paul Engelmann, Johanna Dahlke, Dmitry Degtyar, Daniel Neufeld

1 Chosen Problem

Our goal was to compare different models on their ability to detect emotion in tweets. This was done by comparing different choices of parameters and drawing conclusions from the chosen parameters, such as different features, labels and classifiers.

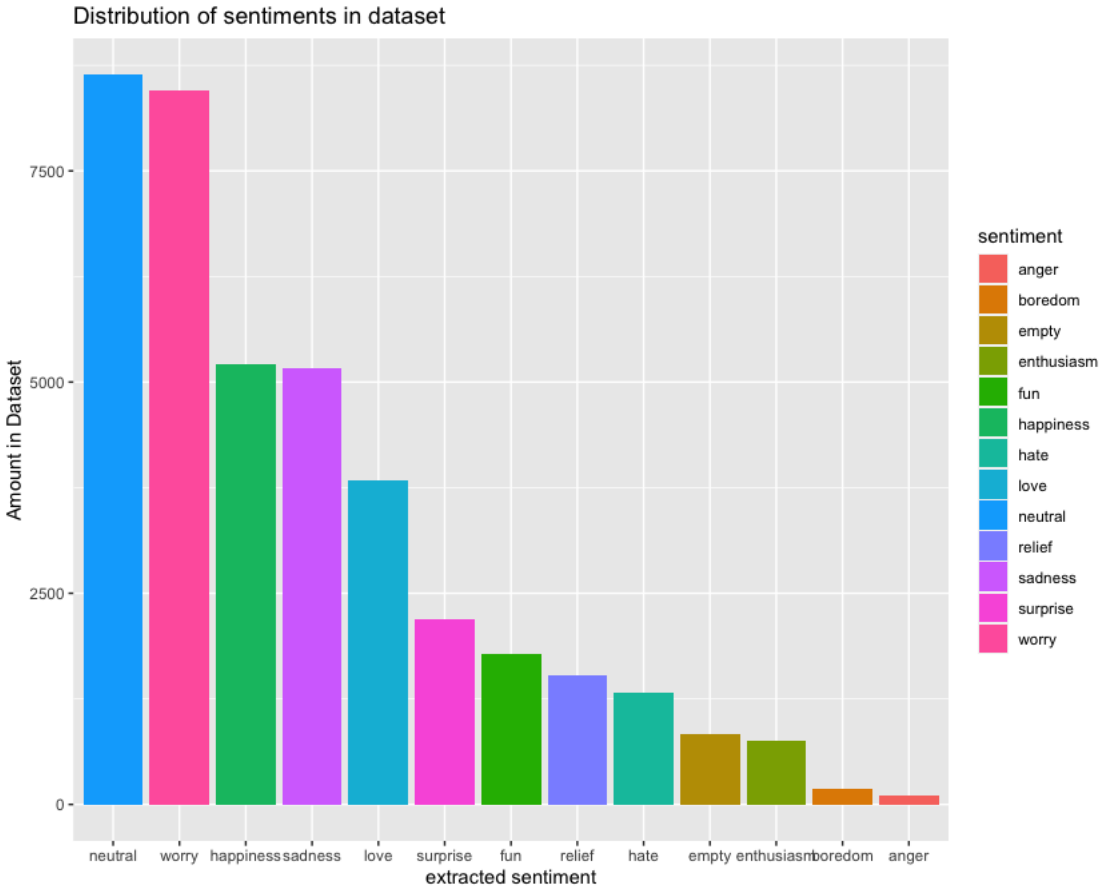
2 Dataset

The dataset we chose is an emotion detection dataset [1].
The data has been collected and probably labelled by hand, based on what type of emotion the tweet most likely would be associated with.
It contains roughly 40000 unique values with 3 columns each:

- **tweet_id** describes the id of the extracted tweet
- **sentiment** is the associated emotion with the tweet
- **content** represents the actual tweet text.

The class labels possess the following 13 different, unique values:
Anger, boredom, empty, enthusiasm, fun, happiness, hate, love, relief, sadness, surprise, worry, neutral.

[1] `https://www.kaggle.com/datasets/pashupatigupta/emotion-detection-from-text`



The chosen dataset possesses quite the imbalance in terms of labels, particularly the two emotions "neutral" and "worry" outweigh the rest. To be able to introduce a balance into the dataset, we've decided to split the dataset into three different categories

- Two Classes: Positive and Negative
- Three Classes: Positive, Negative and Neutral
- All Classes: Complete, unchanged dataset

These categories will help us balance the dataset as well as give us multiple different datasets to compare the models with.

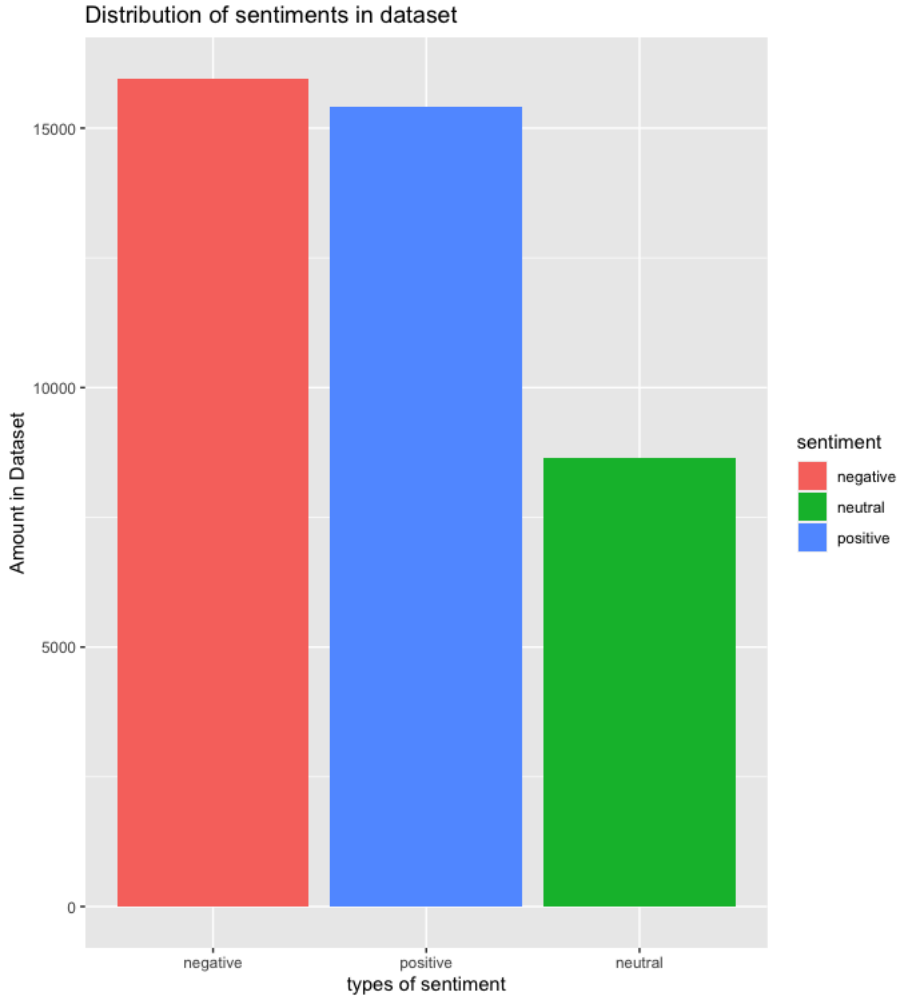
2.1 Binary Dataset

To create the dataset with two classes, we have chosen the following emotions to be included in the **positive** class: happiness, love, surprise, fun, relief and enthusiasm.

Conversely, for the **negative** class we have chosen the labels worry, sadness, hate, empty, boredom and anger.
Additionally, we have distributed the neutral dataset among the positive and negative class through relabeling of the dataset, as we have realised that many tweets in the neutral set do not exhibit a neutral emotion and can be reclassified to better label the emotions in that set.

2.2 Ternary Dataset

Similarly to the binary dataset, we have labeled the previously chosen labels in the same positive and negative classes.
In addition to that, we have also included the neutral dataset as the third and final class for this dataset, to see how well our redistribution worked between the binary and ternary dataset.



As we can see, if we categorize all emotions into the previously explained categories, we can see that labelling the emotions into positive and negative classes equalises the dataset quite well and allows models to be able to train with them quite well in theory.
With the further relabelling of the neutral dataset, we've tried to equally distribute the data onto both negative and positive classes.

2.3 Complete Dataset

To be able to see how well our models perform not only on the reduced class label amount but also on a higher amount, we've decided to also include the complete dataset. This allows us to establish a baseline on what essentially would be the worst case scenario and see how well we can perform with further improvements.

2.4 Further Improvements

When we first tried to train an example model on the binary dataset, we've noticed that the accuracy and f1-scores were quite low for the the relatively simple task of classifying tweets into positive and negative ones.
This suggested, that the labelling of the tweets wasn't particularly accurate and might need a relabelling to be able to compare models better and have a higher baseline in terms of scores. Since our dataset possesses 40000 datapoints, labelling the entire dataset by hand was unfortunately not possible, such that we've decided to use the text processing library TextBlob (footnote) together with spaCy to relabel both binary and ternary datasets.
This was done by extracting the polarity of a particular tweet with TextBlob and then label tweets as positive if their polarity was positive, negative if their polarity was negative. For the binary dataset, we have also included the neutrals in the dataset by relabelling them as either positive or negative. We've decided to relabel neutral tweets by hand, as we've realised that most of the tweets do not express a neutral sentiment. After we have finished our relabelling process, we have roughly kept the same distribution as before, while also increasing our scores to be able to compare models better later on.

3 Data Cleaning

To be able to handle the dataset, we had to clean and prepare the data, such that our models can focus on learning the important things from a sentence and not overfit on irrelevant parts.
Our data cleaning process followed the following steps:

- Remove Usernames / Mentions
- Remove URLs
- Remove Hashtags
- Remove Punctuation
- Remove Numbers
- Translate Slang
- Remove Stopwords
- Lemmatization

Mentions, URLs, Hashtags, Numbers and Punctuations seemed to add no value to our emotion detection and were removed to be able to reduce the feature count later on, as well as make it easier to identify patterns for the models.
Translating slang and lemmatization was also done to reduce the amount of different words in tweets and bringing them to their base form, further allowing us to reduce the amount of features to be created later on.
There are some arguments about removing stopwords or leaving them in, especially since stopwords can influence the sentiment of a tweet and further increase or decrease the polarity, depending on the context. Nonetheless, we've decided to remove stopwords, as we expected our features to be quite large in size and thus tried to reduce the dimensions as much as possible.

4 Features

Once we've cleaned the data, we are going to use two different algorithms to compare our models with:

- TF-IDF
- spaCy word embeddings

TF-IDF uses the method of counting the words and weighing them by their frequency in the entire corpus. If a word appears less frequent than others, then TF-IDF treats it as a more informative word and ranks it higher than others that appear more often.

In addition to TF-IDF we'll be using the spaCy word embeddings, extracted from the large model provided by spaCy. These word embeddings transform the meaning of the word into a vector and place them into a vector space. If two words are similar in meaning to each other, then they are closer in the space than words that are not.

With these two quite different algorithms we can try to understand, which of them perform better with our models and our dataset and conclude, why they might perform that way later on.

5 Classifiers

We have decided to use the following models to compare them to each:

- Support Vector Machine (SVM)
- Random Forest (RF)
- Multi Layer Perceptron (MLP)

These were chosen deliberately to compare models with different learning methods, particularly comparing the use of classical algorithms compared to neural networks such as MLP. We've decided to deviate from our previous selection of models mentioned in the Problem Formulation, as we've saw that Naive Bayes performs very similarly to our Support Vector Machine choice here. To diversify our models a bit, we've since replaced Naive Bayes with Random Forest.

6 Metrics

With our models chosen, we decided on the following metrics to evaluate our models and compare them:

- Accuracy
- Precision / Recall / F1-Score
- Confusion Matrix
- Learning Curve
- Matthews Correlation Coefficient

Accuracy and F1 Score gives us a good baseline to see how well our models perform and might give us early indications that models might have trouble with certain feature selection methods.

The Matthews Correlation Coefficient gives us a metric that allows us to see, how well a model classifies the labels. The metric is between 1 and -1, with 1 signifying a perfect classification, 0 signifying essentially a random guess from the model and -1 signifying an inverse classification, such that we could take the opposite of the models prediction and get a better classification.

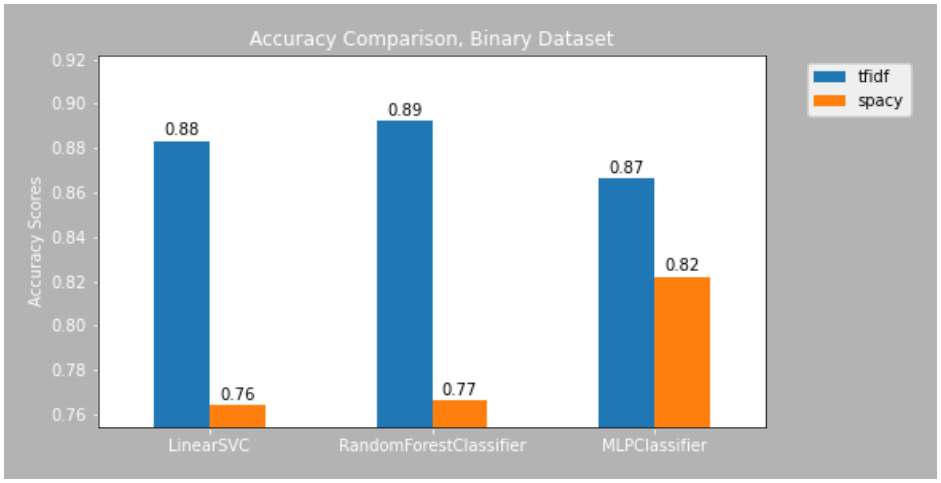
The Confusion Matrix gives us a nice visualisation of how well a classifier can predict labels and displays them based on the false and true classified ones. With this, we can see and establish, with what labels the model is struggling and which ones are performing well.

The Learning Curve gives us a metric, that allows us to compare the learning behaviour of a model to its cross validation behaviour through a plot. This allows us to see, how well a model is learning and how well the model can generalize from learning sentiment to applying them to unseen tweets.

7 Results

7.1 Binary Dataset

7.1.1 Scores



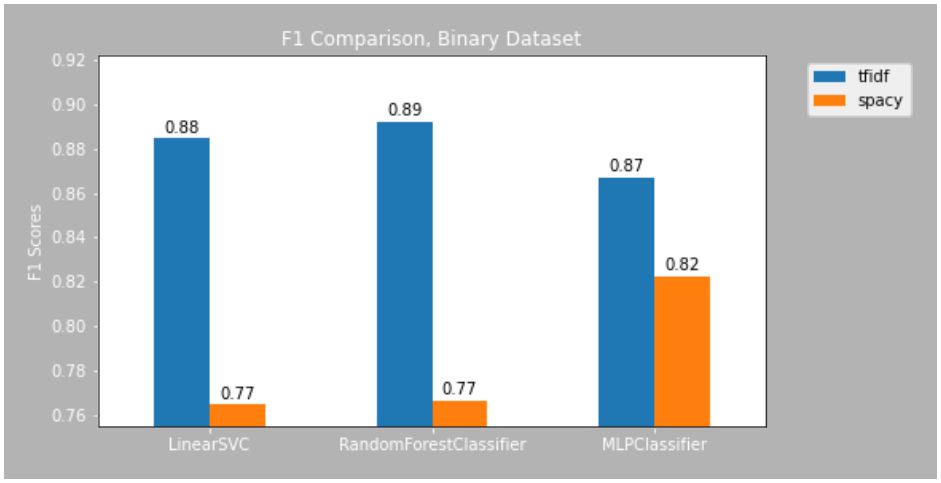


Figure 1: Accuracy and F1 comparison of the binary dataset

Models seem to perform quite similar to each other with TF-IDF, as seen by the accuracy and f1 scores. All of them are within in the range of 0.89 to 0.87, which indicates a strong performance when it comes to classifiying the binary

	SVM	RF	MLP
Accuracy	0.88	0.89	0.87
Precision	0.89	0.89	0.87
Recall	0.88	0.89	0.87
F1	0.88	0.89	0.87

	SVM	RF	MLP
Accuracy	0.76	0.77	0.82
Precision	0.77	0.78	0.82
Recall	0.77	0.77	0.82
F1	0.77	0.77	0.82

Accuracy, Precision, Recall and F1 Scores, Binary Dataset

	SVM	RF	MLP
tfidf	0.77	0.78	0.73
spacy	0.53	0.55	0.64

Matthews Correlation Coefficient, Binary Dataset

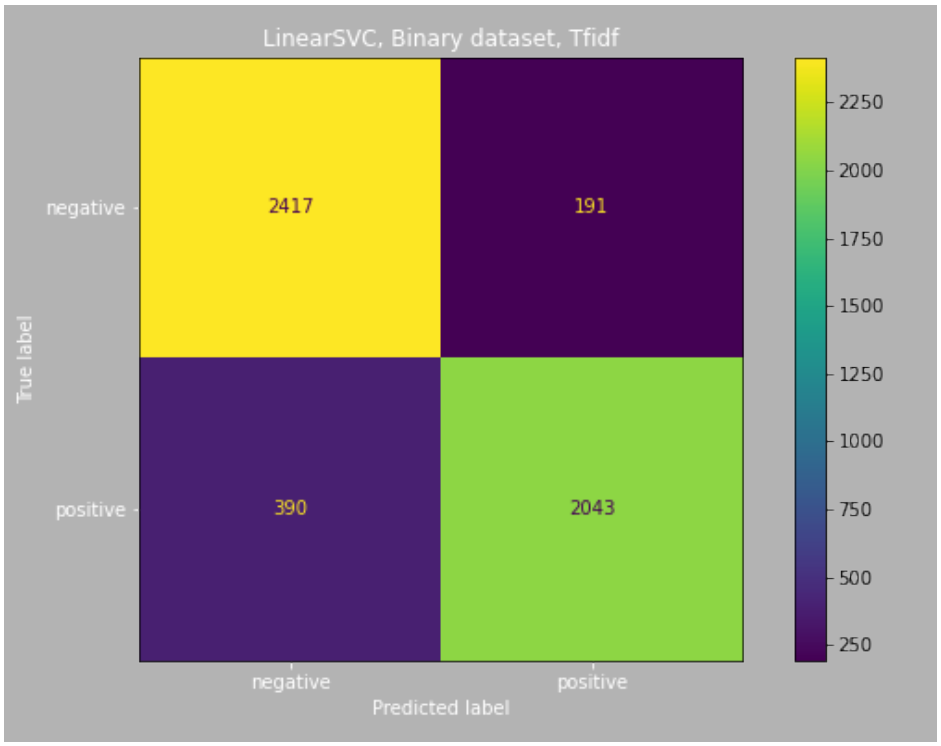
dataset.
With spaCy however, we can see that the scores drop quite dramatically, particularly with SVM and Random Forest. MLP on the other side seems to be able to predict better than the others, which seems unexpected.

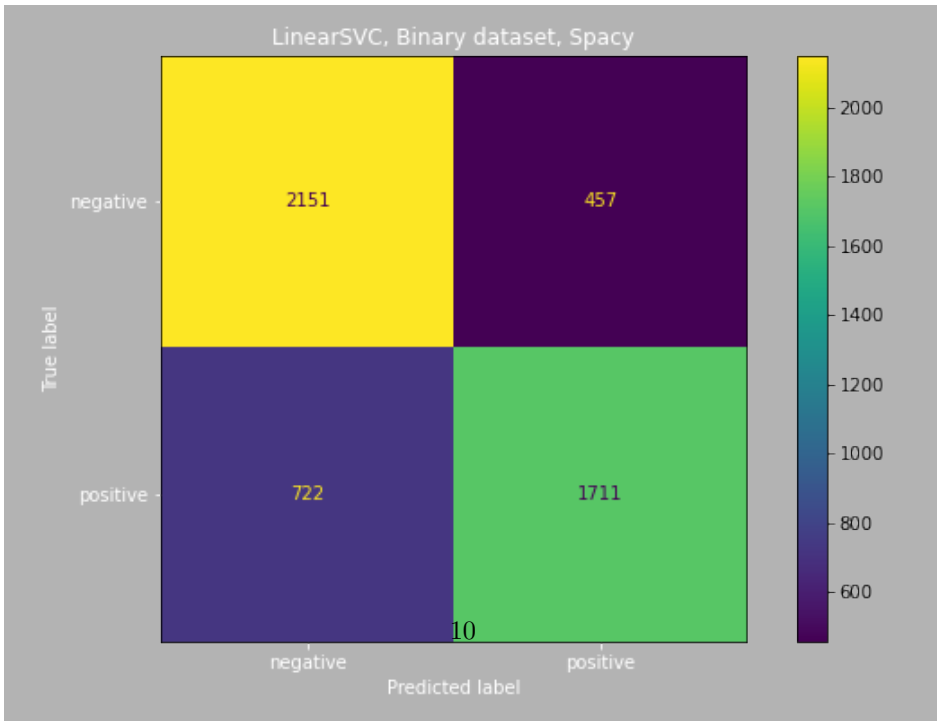
The Matthews Correlation Coefficient paints a similar picture to the accuracy scores and f1 scores.
We can see that with TF-IDF, all models seem to perform decently well.
The MCC score for MLP seems to be a bit lower, which might indicate that MLP can not perform quite as well as the others on our dataset with TF-IDF.

Essentially the opposite is true for spaCy however, SVM and RF perform worse when trying to classify our dataset, whereas MLP can deal with spaCy word embeddings better and possesses a higher MCC score than the others.

Overall however we can still see a strong ability to classify labels correctly. Models are able to correctly predict labels and possess a higher score than simply just guessing labels.

7.1.2 Confusion Matrix SVM





As we can see for SVM, TF-IDF performs well when it comes to classifying data. When we look at the Confusion Matrix for the spaCy version, we can see that SVM has trouble identifying the true classes and generates many more false positives and false negatives compared to the TF-IDF one.

7.1.3 Confusion Matrix RF

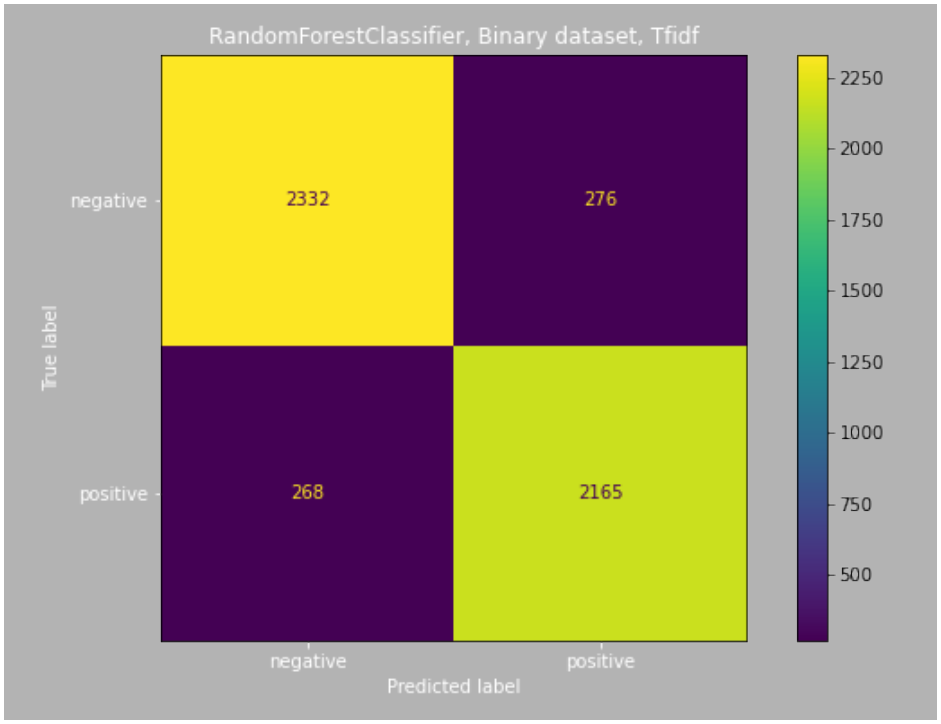


Figure 2: Random Forest together with TF-IDF on the binary dataset

With Random Forest we can see quite the similar picture to TF-IDF, with the model being able to perform similarly well, keeping the false negatives and false positives low. With spaCy however, the model has quite a lot of trouble particularly classifying true positives as such and resorts to classifying them as negative quite often.

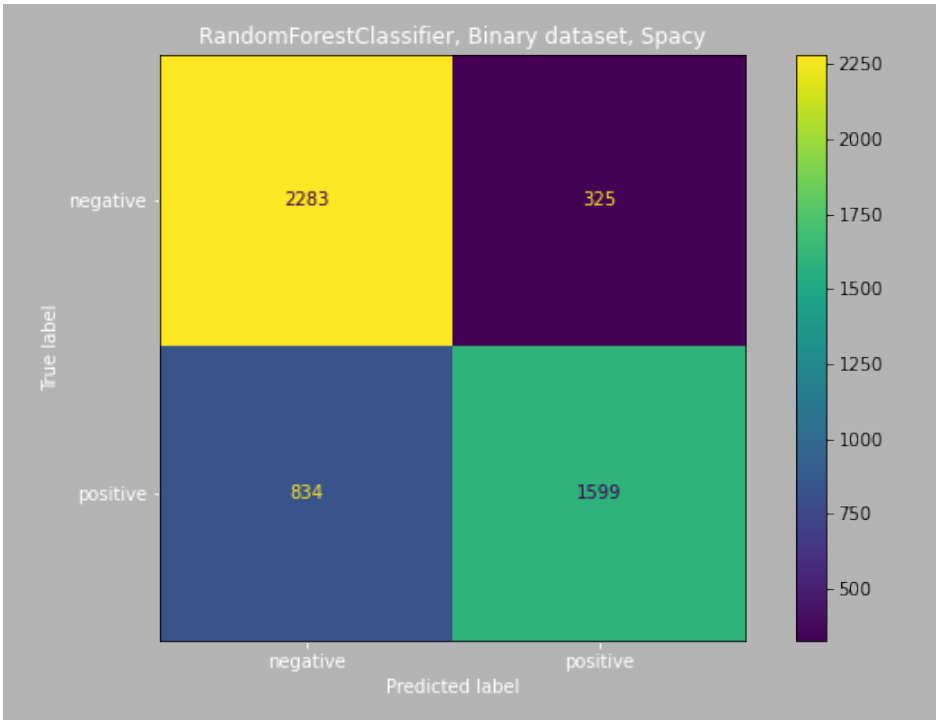


Figure 3: Random Forest together with spaCy on the binary dataset

7.1.4 Confusion Matrix MLP

If we compare the previous results with RF and SVM to the MLP one however, we can see that, while MLP has more false positive and negatives in the TF-IDF case, MLP can classify with spaCy better than the others and does not have as many problems with predicting true positive labels.

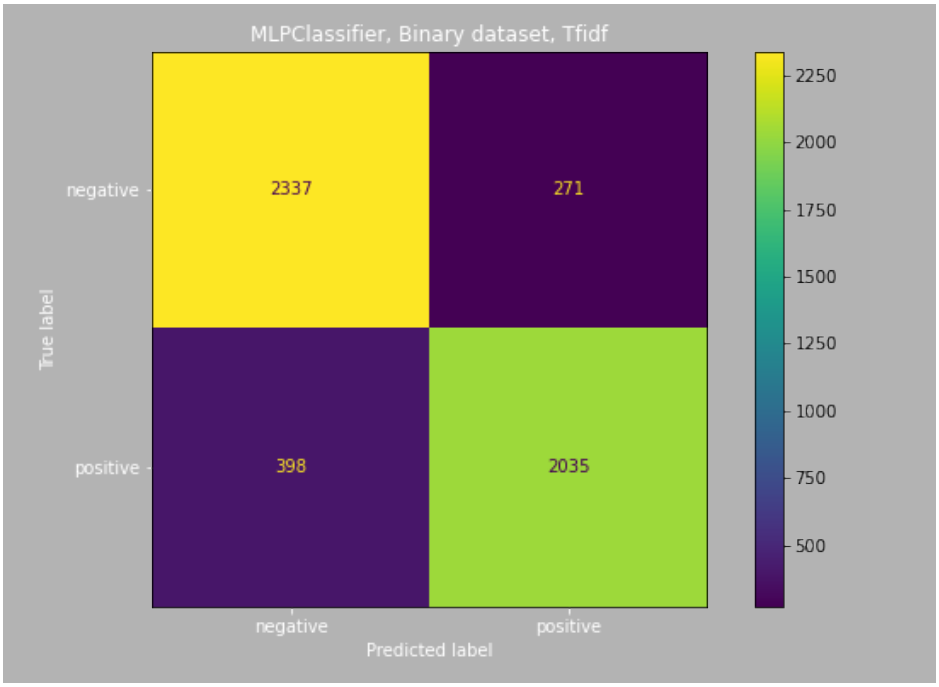


Figure 4: MLP together with TF-IDF on the binary dataset

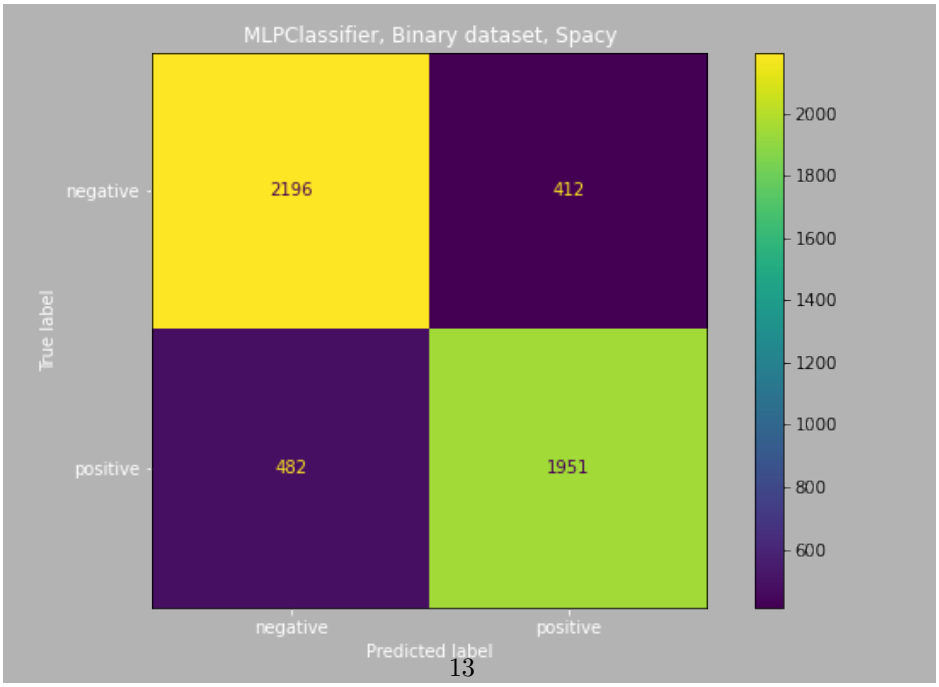
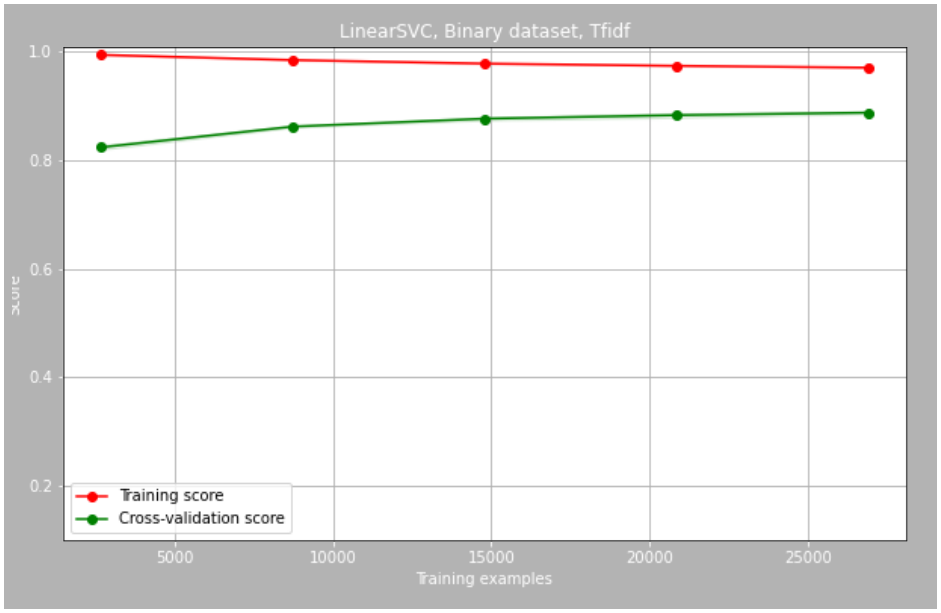


Figure 5: MLP together with spaCy on the binary dataset

7.1.5 Learning Curves TF-IDF



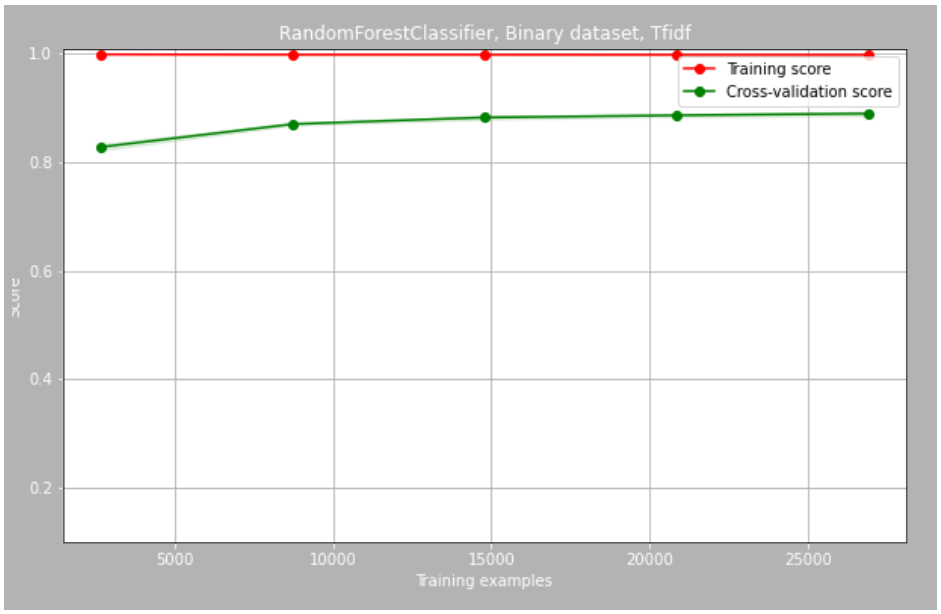


Figure 6: Learning Curves for SVM and RF with TF-IDF

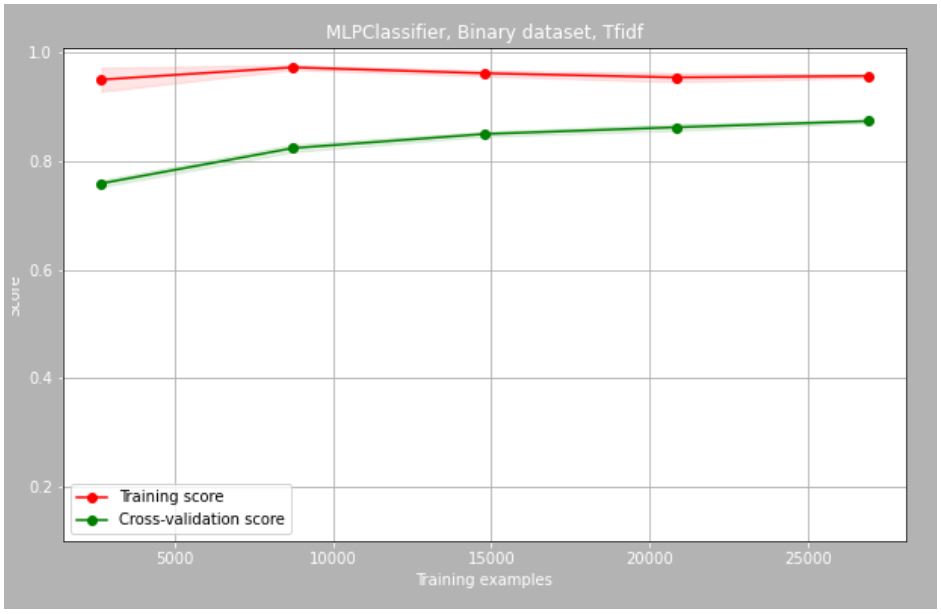
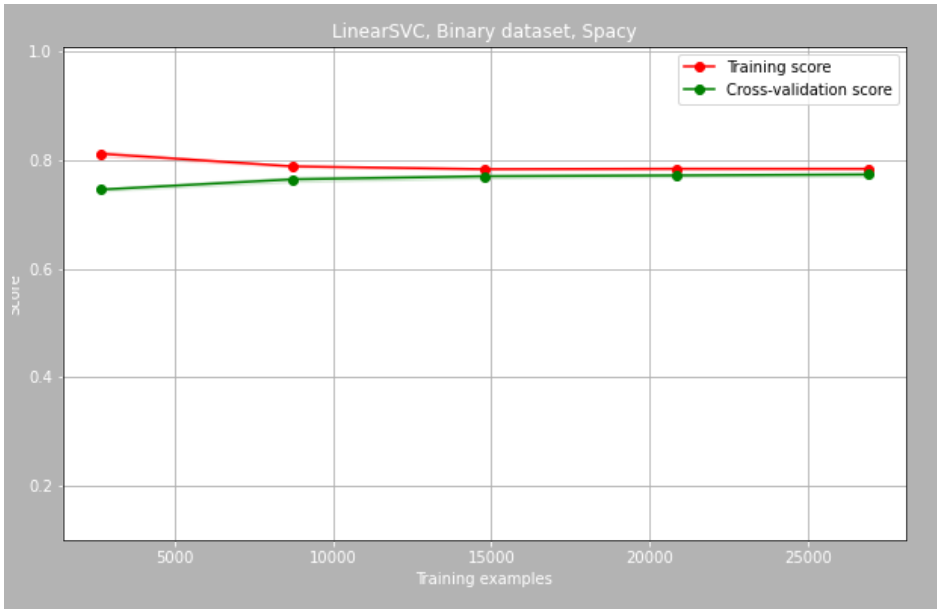


Figure 7: Learning Curve for MLP with TF-IDF

Through the learning curves we can see that all classifiers perform similarly well when learning with TF-IDF. Particularly Random Forest seems to learn the training data too well and may show signs of overfitting later on, if used on other datasets. For the binary dataset however, all models possess good scores for both the training and cross validation, which indicates that they learn and classify well enough.

7.1.6 Learning Curves spaCy



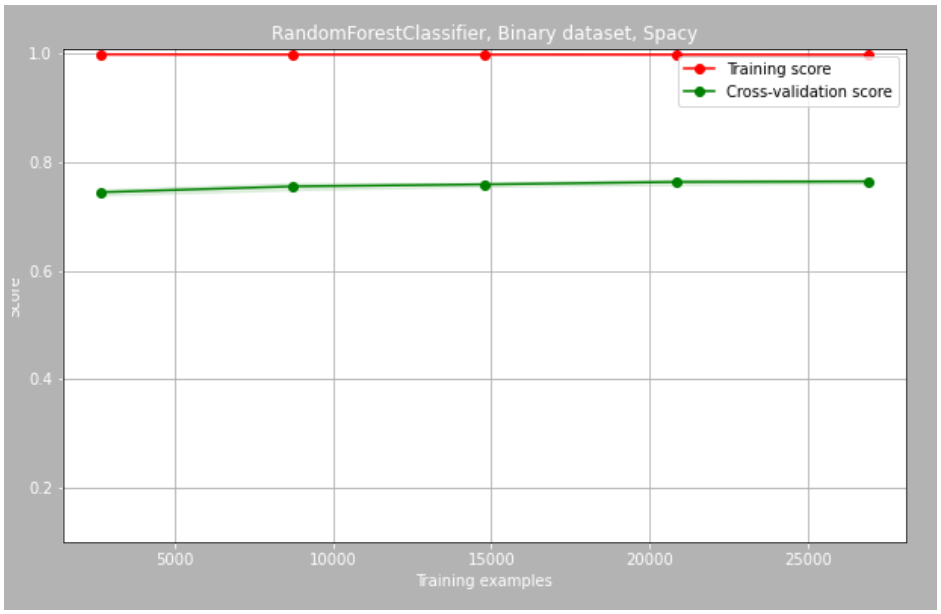


Figure 8: Learning Curves for SVM and RF with spaCy

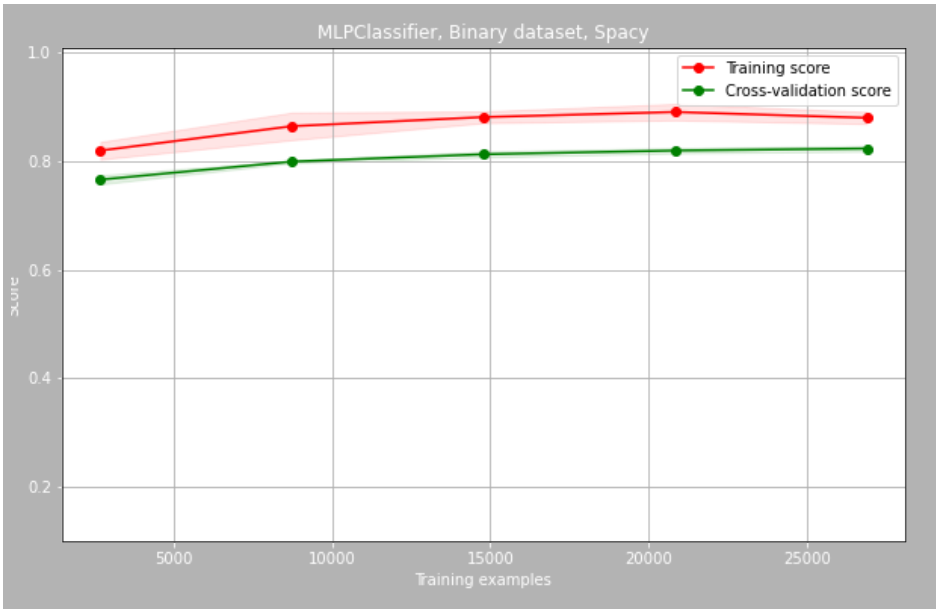
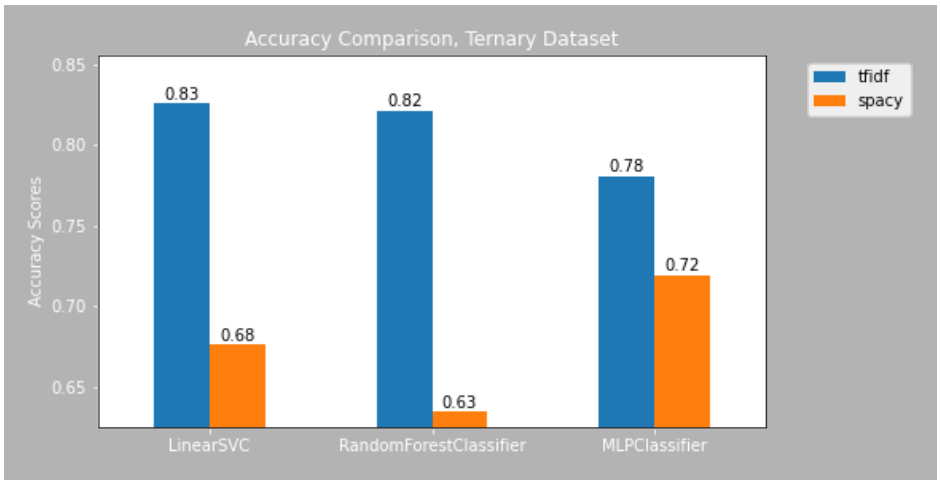


Figure 9: Learning Curves for MLP with spaCy

With the spaCy curves we can see that SVM seems to be limited by the training data. Early on the training score plateaus and does not seem to be able to go above 0.8, which also limits the cross validation score and thus the model can not classify labels as well.
Random Forest on the other hand seems to overfit on the data and thus not be able to generalize as well on unseen data, as seen by the perfect training score and worse cross validation score.
MLP seems to be able to deal with the training data much better than the others, as it does not seem to overfit or plateau, but rather increase its cross validation score as more data gets added.

7.2 Ternary Dataset

7.2.1 Scores



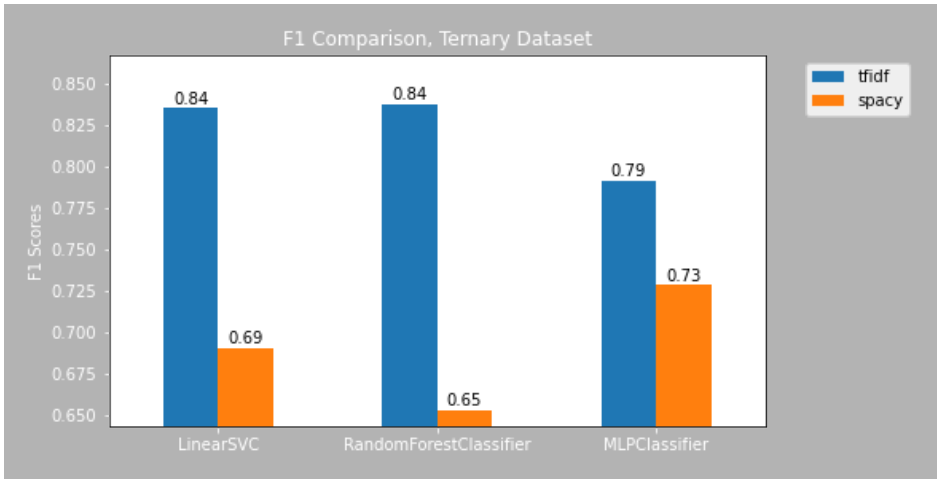


Figure 10: Accuracy and F1 comparison for the ternary dataset

As previously explained, the ternary dataset includes all the previously labeled datapoints with their positive or negative class. Unlike the binary dataset, this dataset includes the neutral class as well. Similarly to the results of the binary dataset, the ternary dataset seems to perform quite well with the classical algorithms and TF-IDF, as seen by the accuracy and f1 scores. MLP on the

	SVM	RF	MLP
Accuracy	0.83	0.82	0.78
Precision	0.84	0.85	0.79
Recall	0.84	0.84	0.79
F1	0.84	0.84	0.79

	SVM	RF	MLP
Accuracy	0.68	0.63	0.72
Precision	0.69	0.66	0.73
Recall	0.69	0.66	0.73
F1	0.69	0.65	0.73

Accuracy, Precision, Recall and F1 Scores, Ternary Dataset

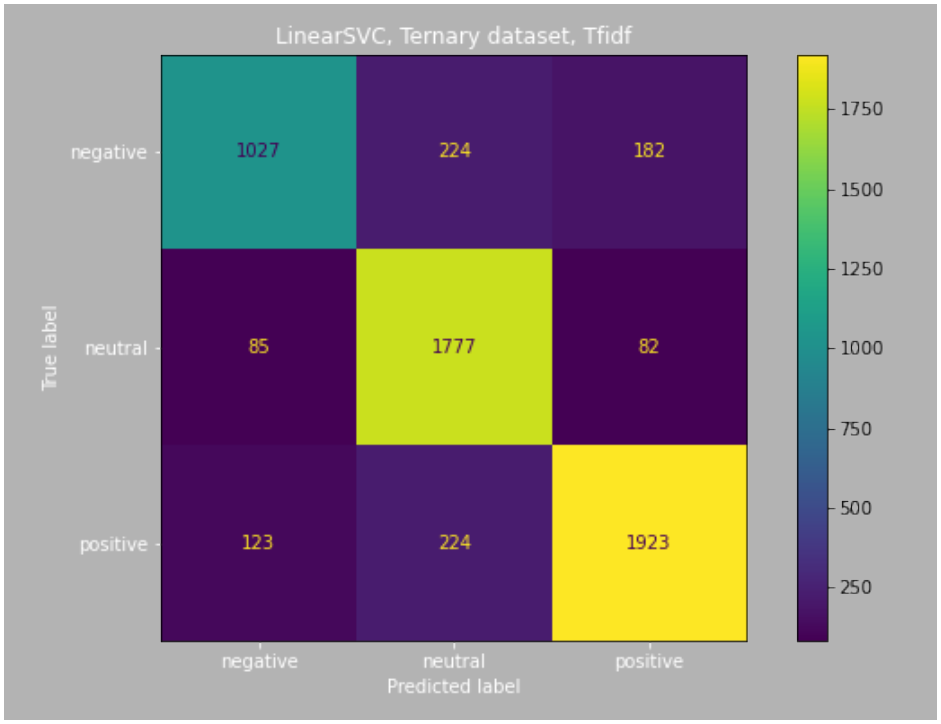
	SVM	RF	MLP
tfidf	0.75	0.76	0.68
spacy	0.53	0.48	0.59

Matthews Correlation Coefficient, Ternary Dataset

contrary performs quite a bit worse with TF-IDF compared to the other ones. spaCy gives us a similar picture to the binary dataset as well, as we can see that MLP performs the best in relation to the others with the spaCy word embeddings. Random Forest performs the worst out of all of them with spaCy at a low f1 score of 0.65 compared to MLPs 0.73, indicating that RF might have an issue dealing with spaCy embeddings.
If we look at the MCC score, we can see similarities to the binary dataset as well. MLP performs the worst for the TF-IDF vectorization and the best with the spaCy one. The scores themselves are a bit lower than the binary dataset in general, but overall indicate that the models themselves still have the ability to classify labels better than just randomly guessing.

7.2.2 Confusion Matrix SVM

For TF-IDF we can see that almost all labels are classified correctly with the SVM classifier. Compared to spaCy, we can see that many negative and neutral are misclassified more often. Particularly neutrals are quite often classified as positive ones, indicating that the neutral tweets might be similar to the positive tweets in the dataset.



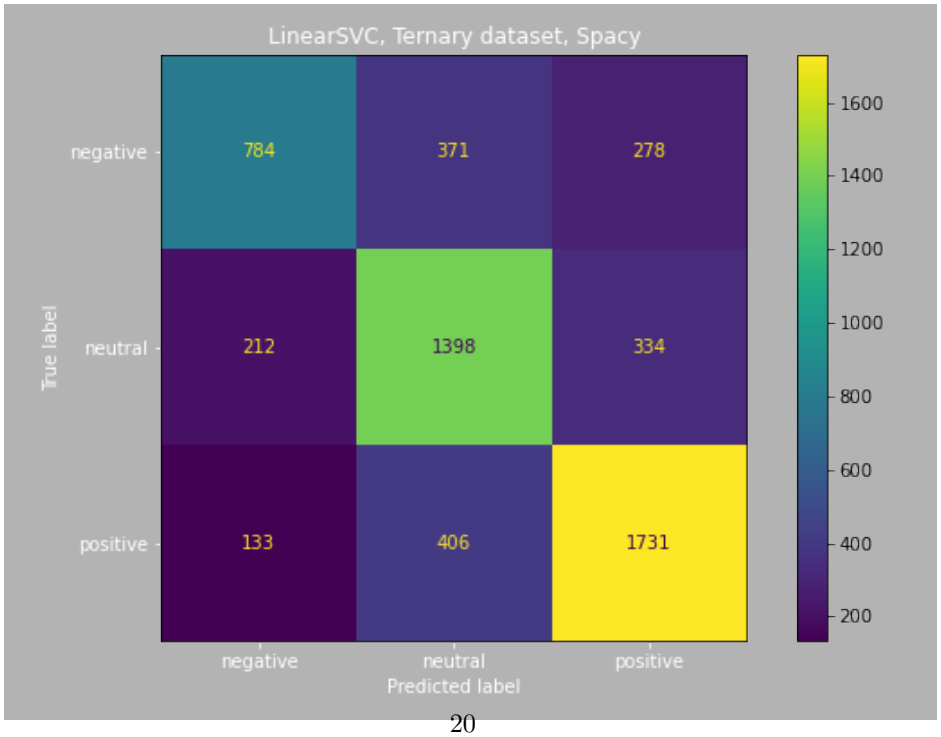


Figure 11: SVM Confusion Matrix with TF-IDF and spaCy on the ternary dataset

7.2.3 Confusion Matrix RF

Similarly to SVM, RF seems to perform quite well with TF-IDF. With spaCy however, we can see that RF struggles with classifying true negative ones correctly, essentially guessing which label they belong to and getting confused quite easily.

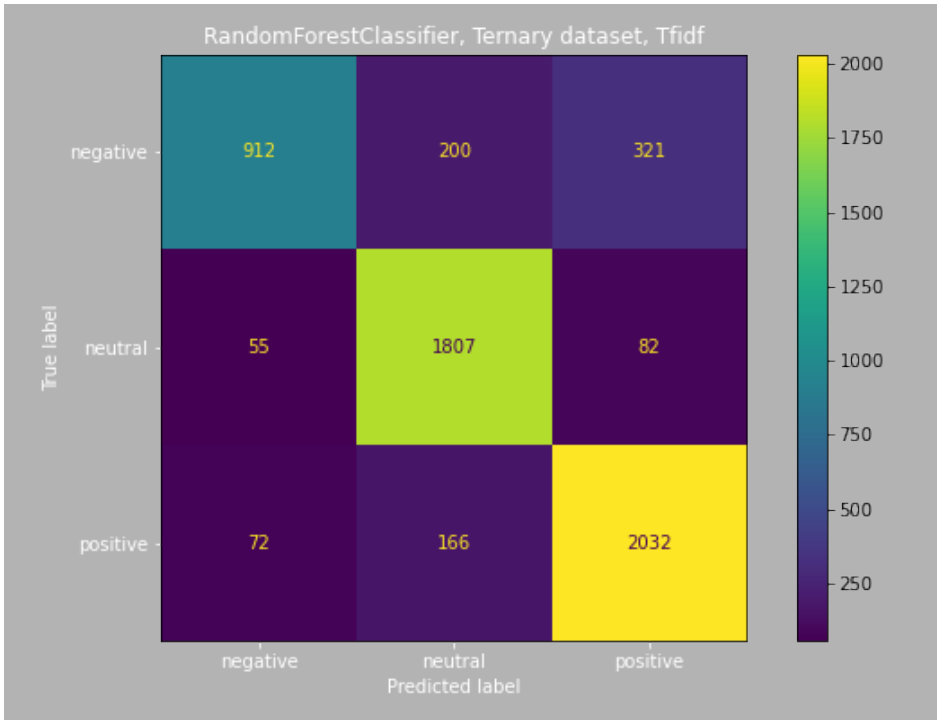


Figure 12: Confusion Matrix with TF-IDF on the ternary dataset

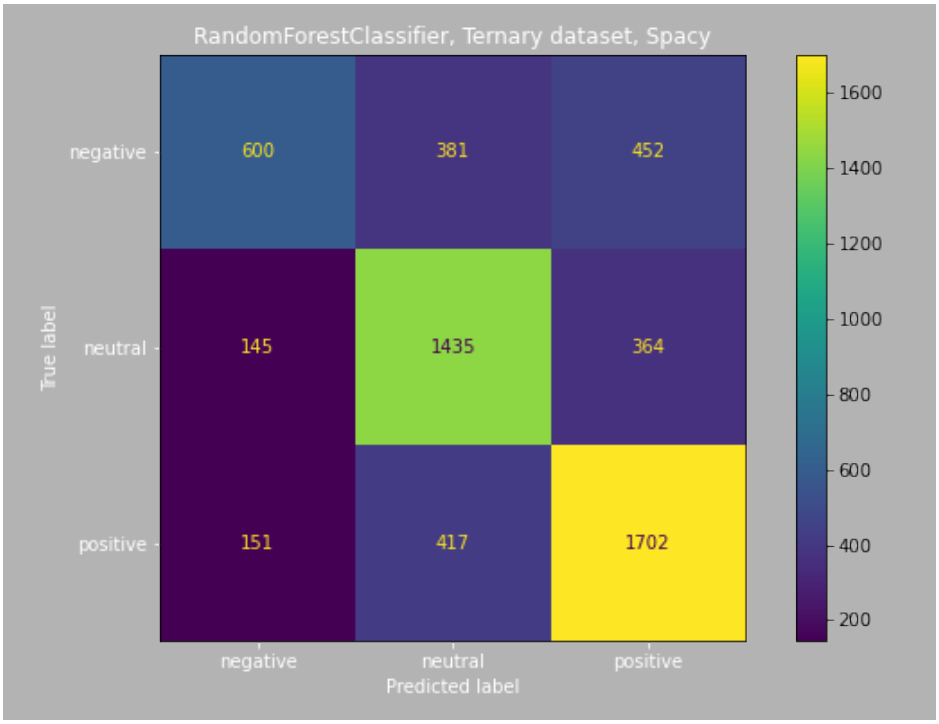


Figure 13: Random Forest Confusion Matrix with spaCy on the ternary dataset

7.2.4 Confusion Matrix MLP

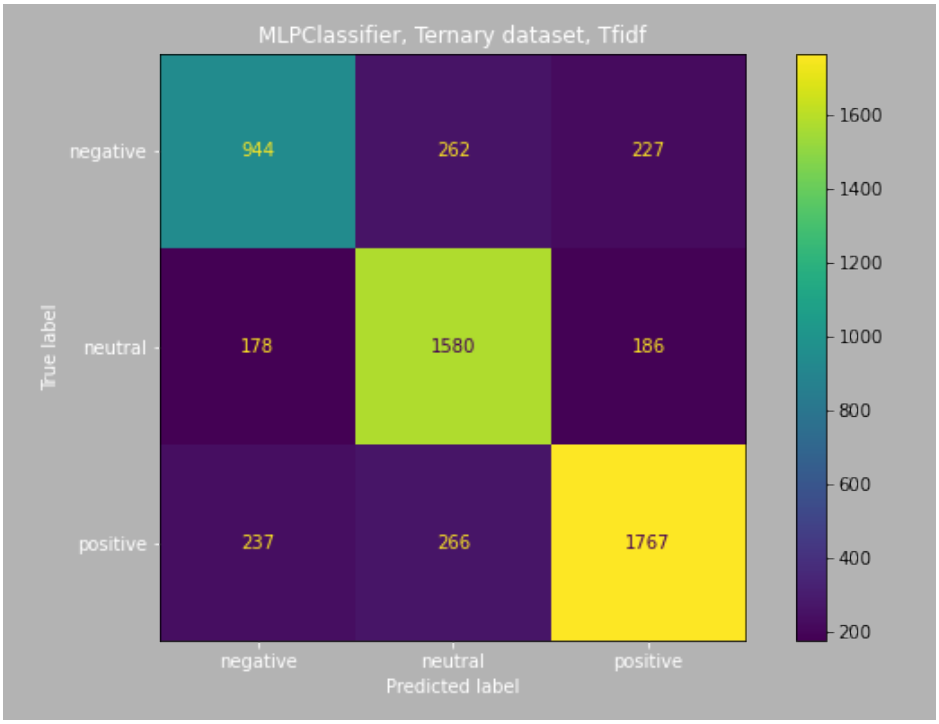


Figure 14: MLP Confusion Matrix with TF-IDF on the ternary dataset

Looking at the MLP confusion matrix, we see that, again with TF-IDF, MLP performs quite a bit worse than the other models, but seems to be able to classify data much better than the others with spaCy. Particularly towards the negative and neutral labels, MLP does not seem to be struggle as much as the others with labelling them them properly and such can classify them correctly more often than the others.

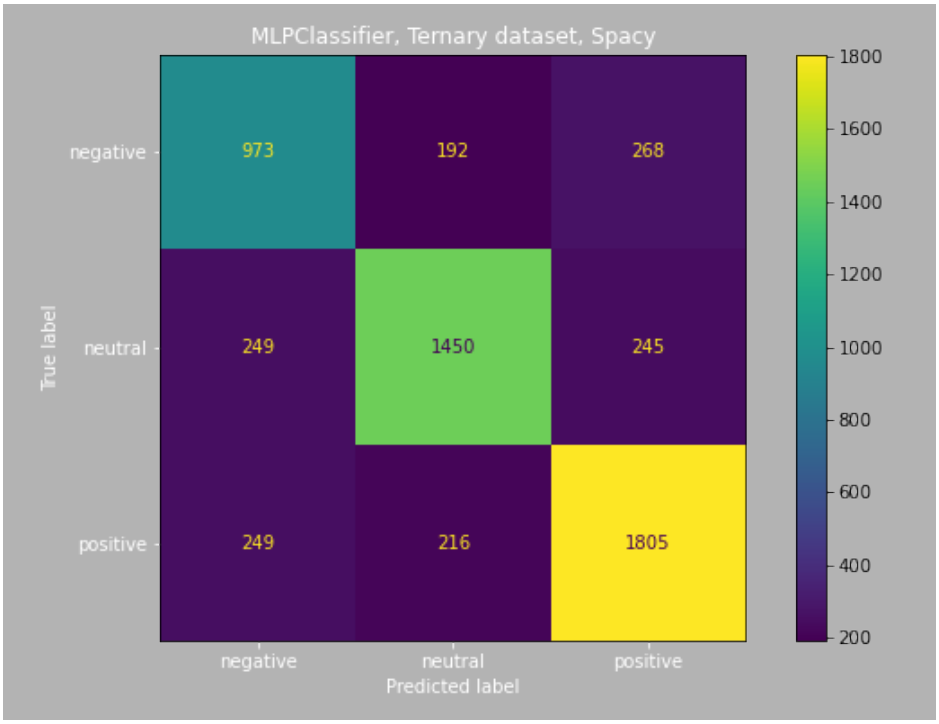
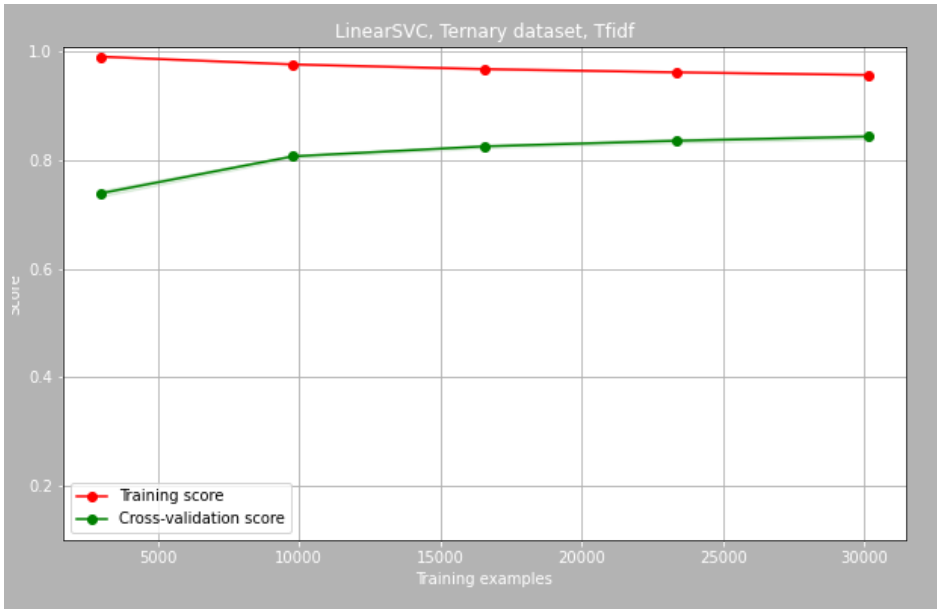


Figure 15: MLP Confusion Matrix with spaCy on the ternary dataset

7.2.5 Learning Curves TF-IDF



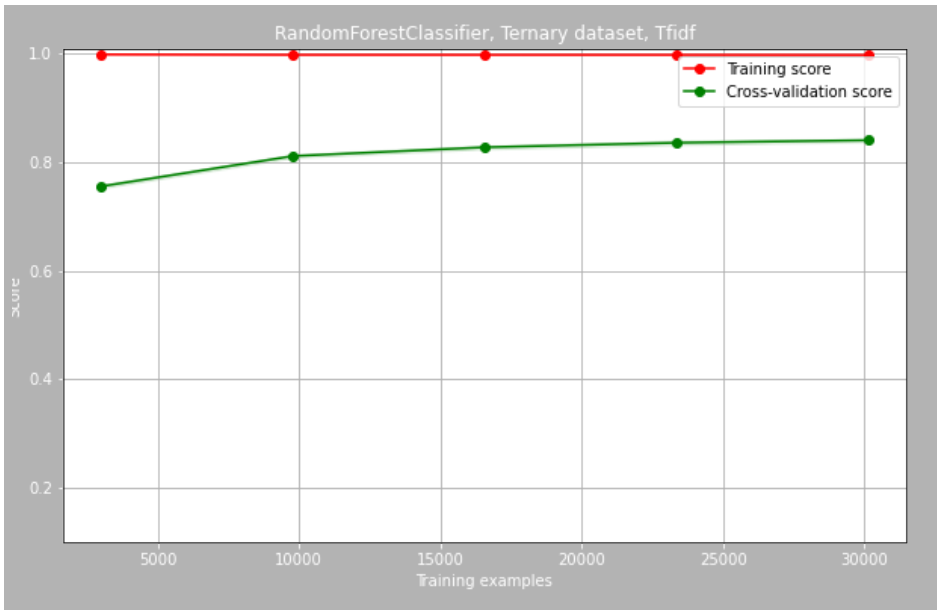


Figure 16: Learning Curves for SVM and RF with TF-IDF on the ternary dataset

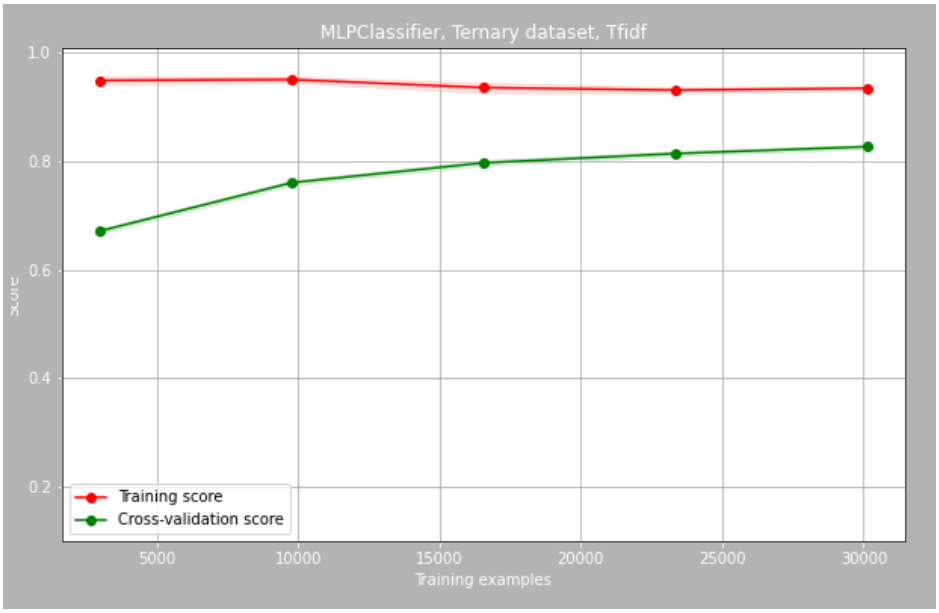
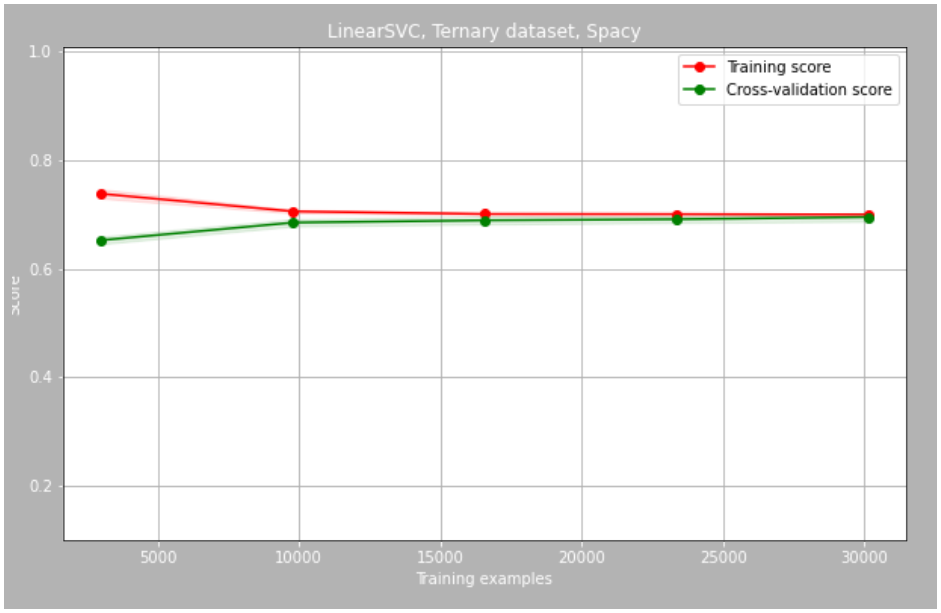


Figure 17: Learning Curves for MLP with TF-IDF on the ternary dataset

We can see similar issues to the binary dataset in the ternary dataset here as well. With TF-IDF, both MLP and SVM seem to learn reasonably well, although MLP learns a bit worse compared to SVM, and can apply the learned patterns to unseen data quite well, even if the overall cross validation score is lower than in the binary case.

7.2.6 Learning Curves spaCy



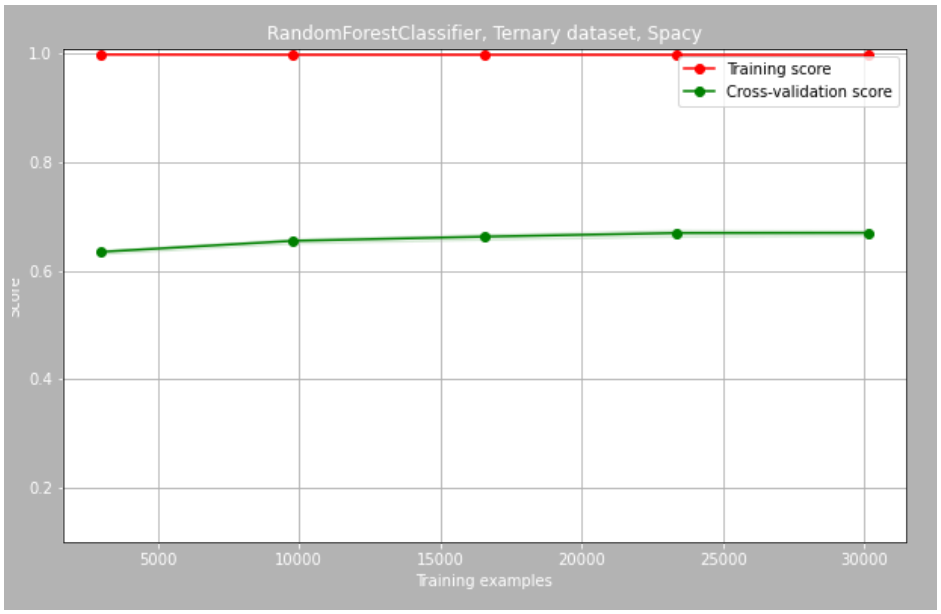


Figure 18: Learning Curves for SVM and RF with spaCy on the ternary dataset

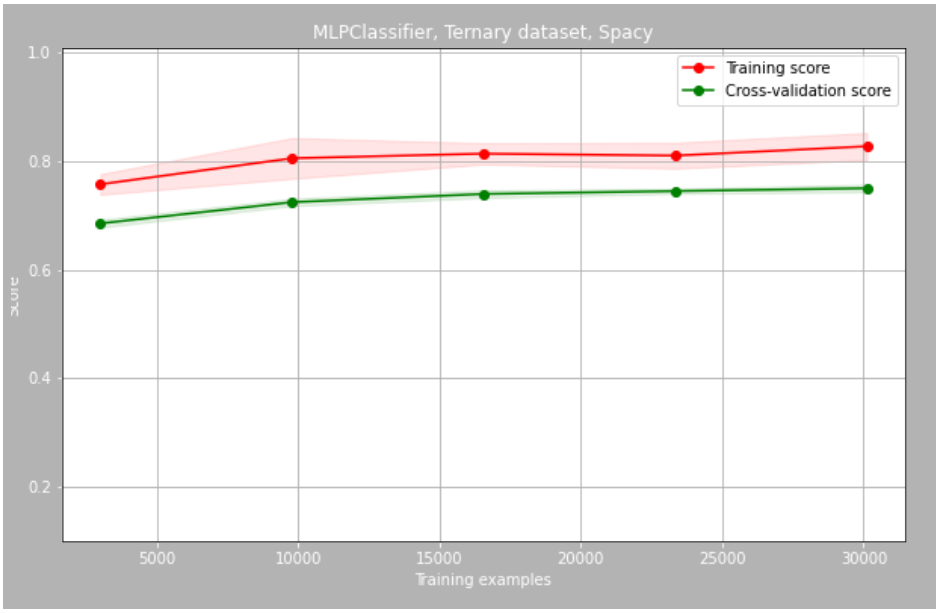


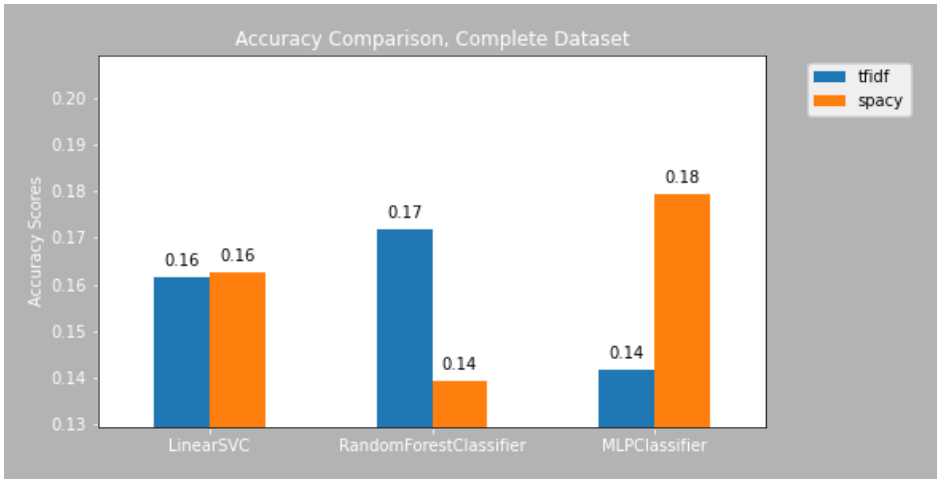
Figure 19: Learning Curves for MLP with spaCy on the ternary dataset

The spaCy learning curves show us, that the same issues seem to to persist in the ternary dataset in comparison to the binary dataset. SVM has trouble learning spaCy data, which indicates that SVM might not be able to correctly extract patterns from the training data and is thus quite limited when it comes to applying those learned patterns on new, unseen data.
Random Forest overfits the training data with a training score of 1, such that it can not generalize well enough for the new data to be classified correctly and thus can not perform well enough for classification.
MLP handles spaCy word embeddings better, can learn them better and thus apply them much easier as well. In particular we can see that the training curve has a positive slope almost everywhere, which might indicate that more training data could further improve the performance of the MLP model, which then would also increase the cross validation score and make the model perform better in general.

7.3 Complete Dataset

In the end, we compared our models to the complete dataset, to see how well it would perform and to draw conclusions about how much our split improved our models compared to just training them like this.

7.3.1 Scores



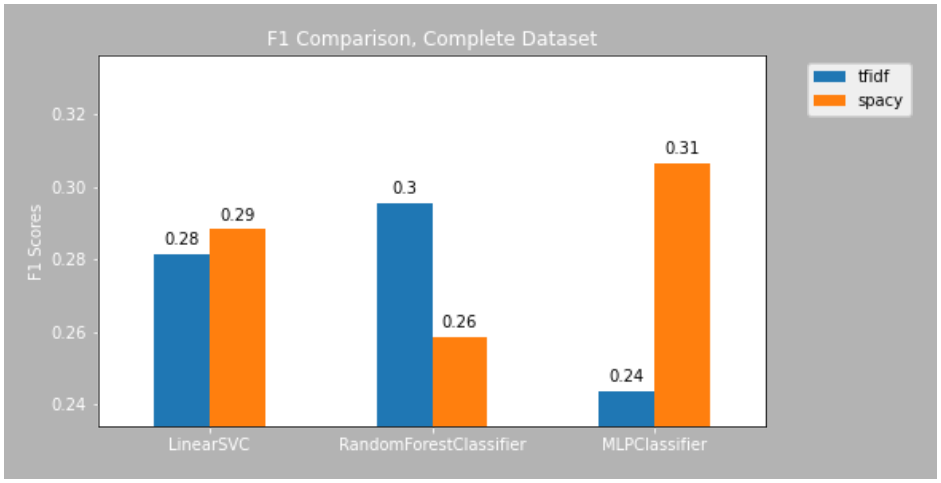


Figure 20: Accuracy and F1 comparison for the complete dataset

	SVM	RF	MLP
Accuracy	0.16	0.17	0.14
Precision	0.28	0.31	0.24
Recall	0.3	0.33	0.25
F1	0.28	0.3	0.24

	SVM	RF	MLP
Accuracy	0.16	0.14	0.18
Precision	0.32	0.29	0.3
Recall	0.33	0.32	0.34
F1	0.29	0.26	0.31

Accuracy, Precision, Recall and F1 Scores, Complete Dataset

	SVM	RF	MLP
tfidf	0.17	0.19	0.12
spacy	0.19	0.16	0.2

Matthews Correlation Coefficient, Complete Dataset

For the complete dataset we can see that the models can not handle all the classes quite as well. We can see for the accuracy scores that they are all within the range 0.14 to 0.18, indicating quite low classification ability. For F1 scores we see a similar picture, but in the range 0.24 to 0.31.

The differences between models show quite clearly here as well. SVM seems to be able to use TF-IDF as well as spaCy and doesn't show much difference between those two. RF shows quite the clear difference between TF-IDF and spaCy, with it performing better in accuracy and f1 for TF-IDF.

For MLP we see the opposite of RF, MLP has a much easier time classifying with spaCy compared to TF-IDF and outperforms the others with spaCy in both score types, even if barely.

If we look at the MCC score, MLP seems to beat the others with spaCy and lose out on TF-IDF, similar to the previous datasets. Nonetheless, all models are still better than just random guessing, even if barely so, indicating that they are not only guessing for all labels.

7.3.2 Confusion Matrix SVM

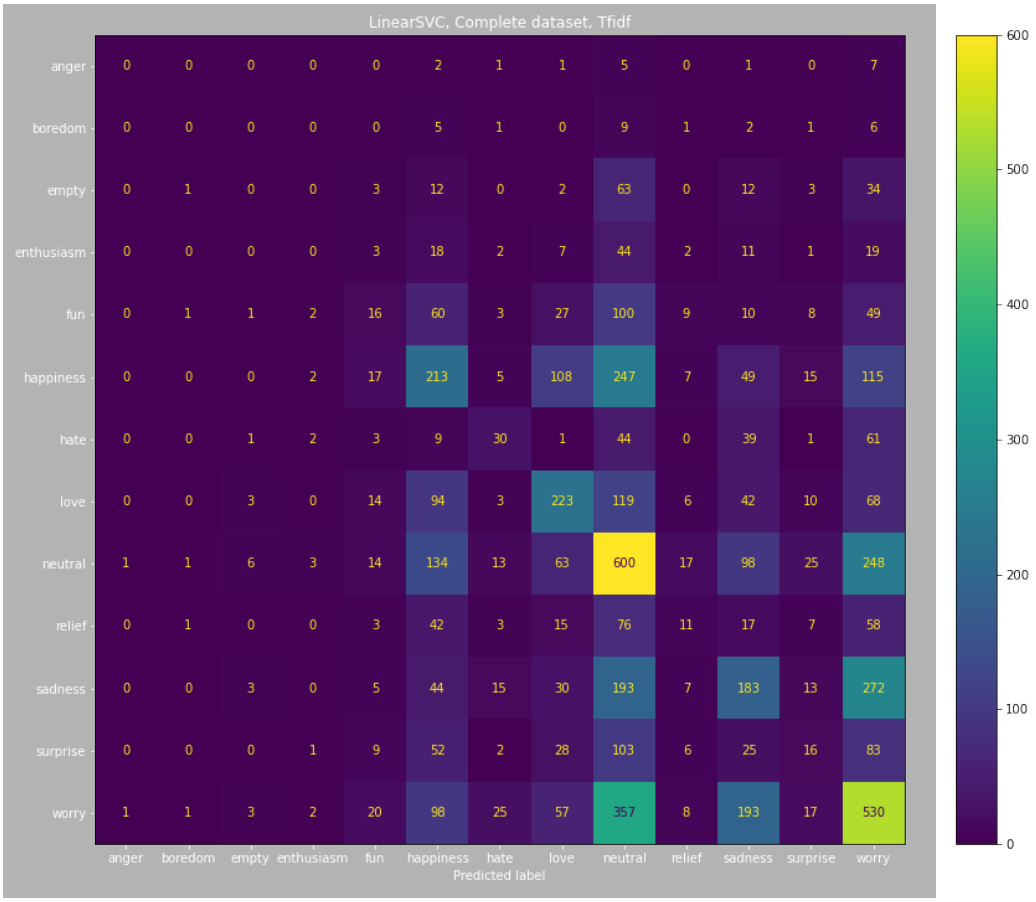


Figure 21: Support Vector Machine together with TF-IDF on complete dataset

With TF-IDF we can see that most labels are quite hard to classify correctly, mainly since the dataset is so unbalanced and relevant data might not appear in the test set, thus providing us with no way to verify if the data can be classified in the first place.
We can see, that most labels are being classified as either neutral or worry. This stems from the fact that most of the data belongs to both classes, such that the classifier can learn the data from these labels the best.
Similarly and unsurprisingly to TF-IDF, spaCy has the same issues stemming from the dataset, such that they perform equally as bad in classifying data correctly, althought we can see that spaCy can classify the label *worry* better than TF-IDF can.

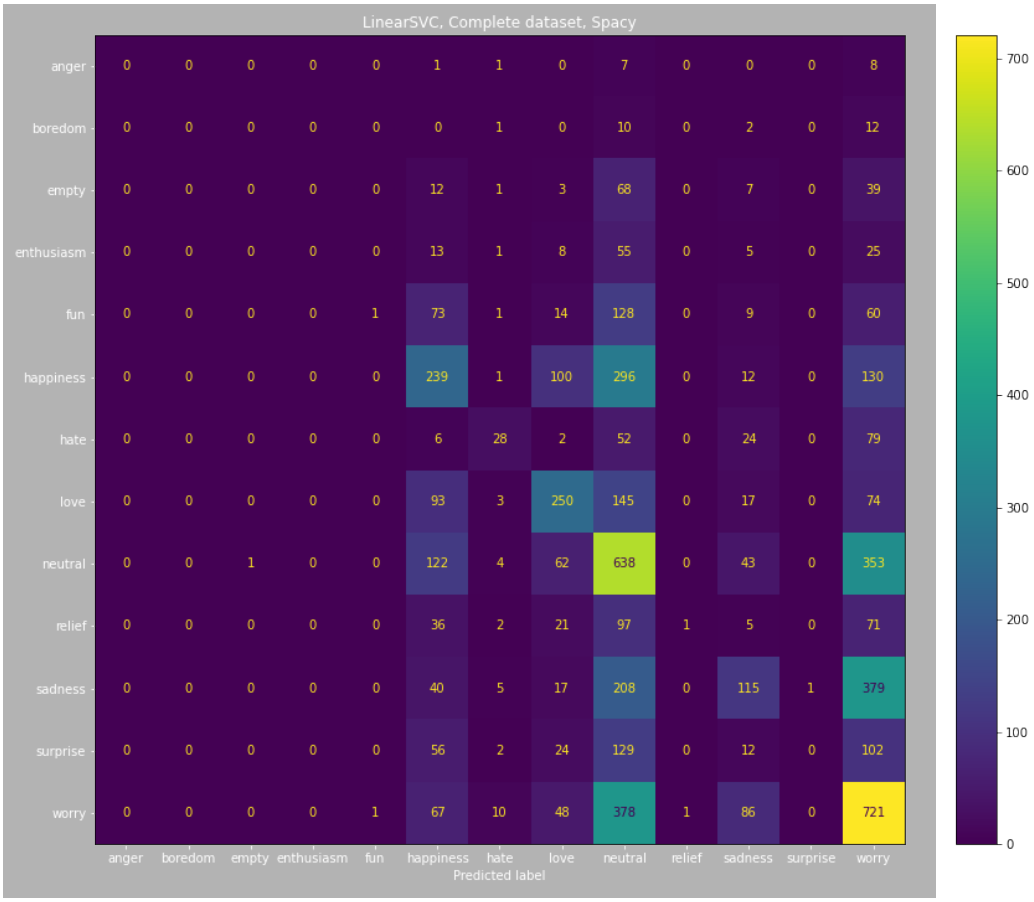


Figure 22: Support Vector Machine together with spaCy on complete dataset

7.3.3 Confusion Matrix RF

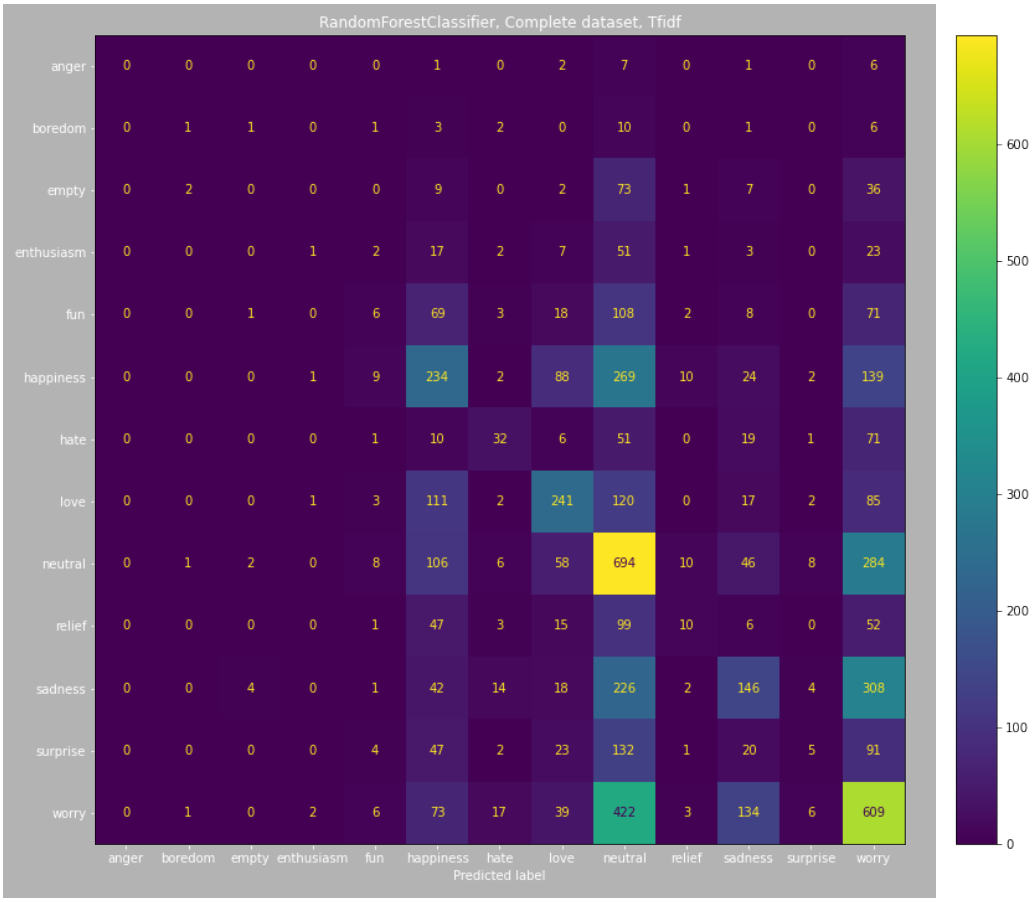


Figure 23: Random Forest together with TF-IDF on complete dataset

Similarly to SVM, RF has a better performance with TF-IDF. We can see that the ones that the model can classify, are classified decently. With spaCy however, we can see that the model only realy knows how to classify *worry* and *neutral*, essentially losing out on all the other classes. The biggest difference here is the label *sadness*, where with TF-IDF RF could still somewhat classify occurrences of the label, whereas with spaCy the model can not classify the label at all anymore.

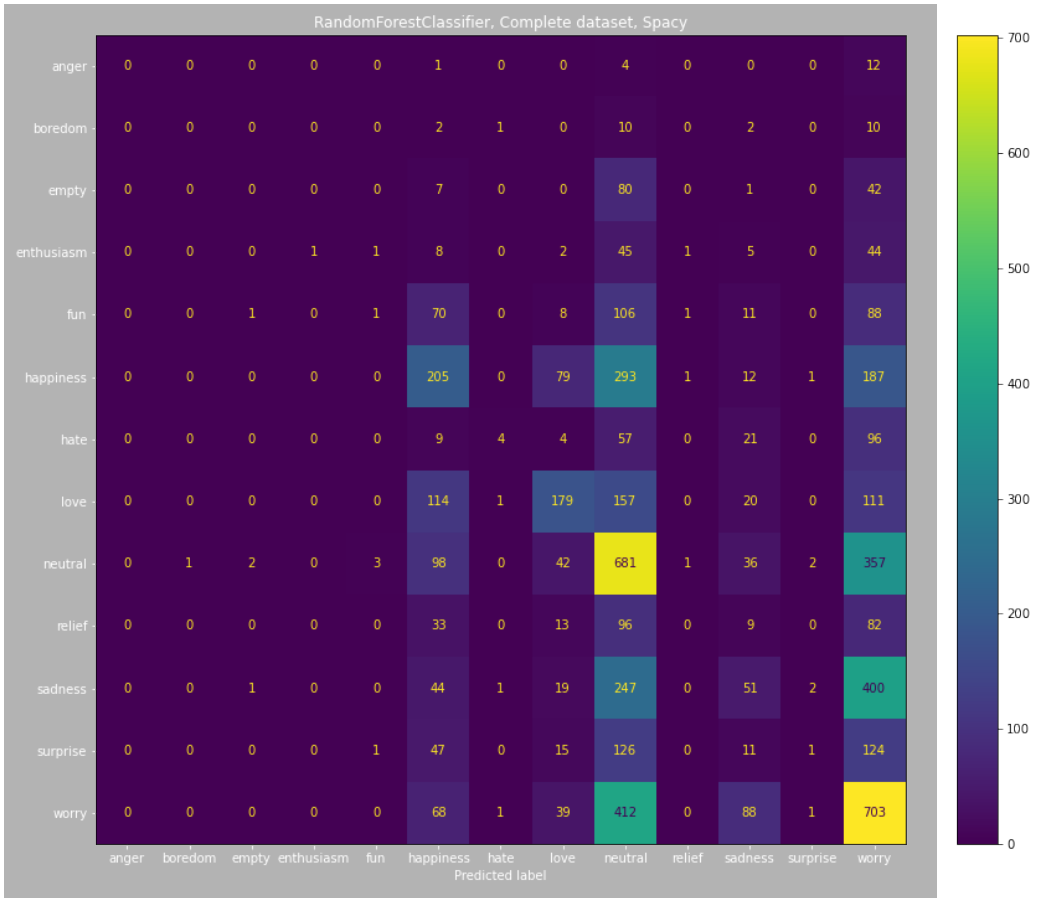


Figure 24: Random Forest together with spaCy on complete dataset

7.3.4 Confusion Matrix MLP

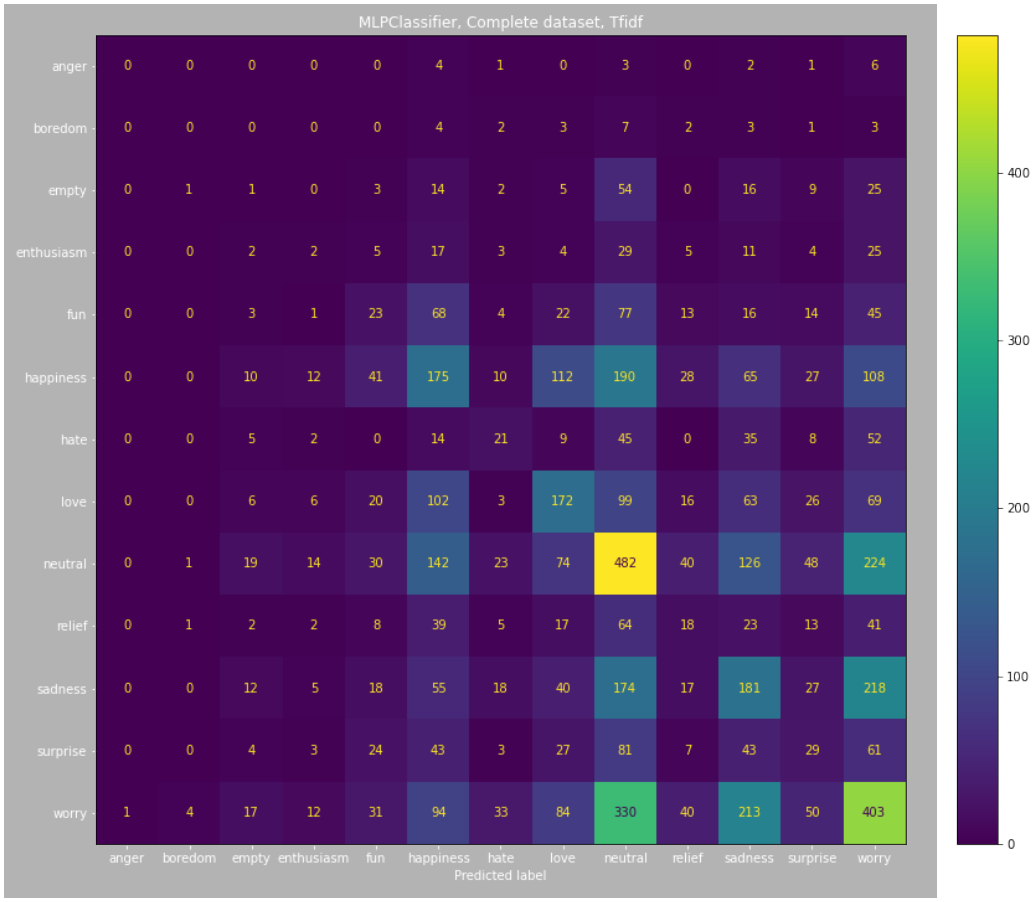


Figure 25: MLP together with TF-IDF on complete dataset

Look at the MLP confusion matrix, we see that with TF-IDF, *worry* is not classified as well as with other models. With spaCy we see the opposite, worry is classified well and others, such as neutral, are not classified as well anymore. This might hint towards spaCy being able to group sentences with the label worry better together than others. We can also observe quite a bit of overlap with neutral and worry, as both are quite frequently predicted as being the other class. This could hint towards the classes being quite close in meaning and thus being classified as the other.

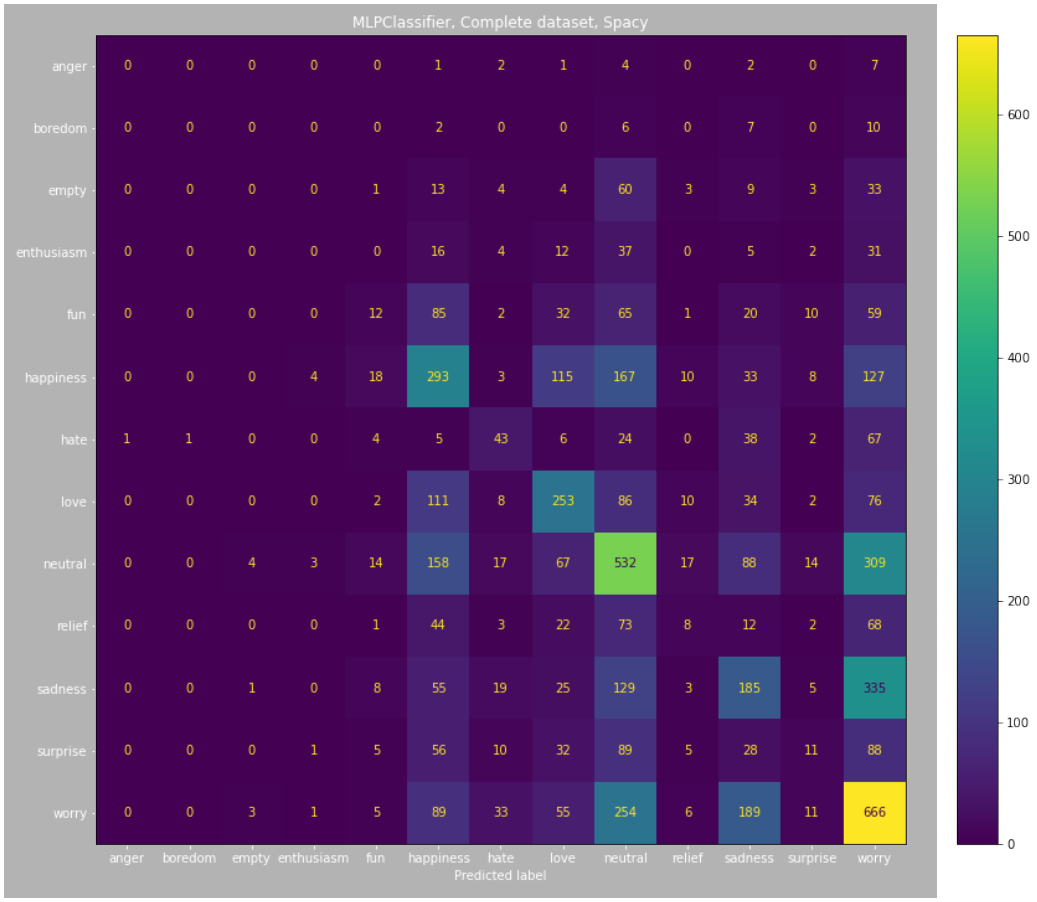
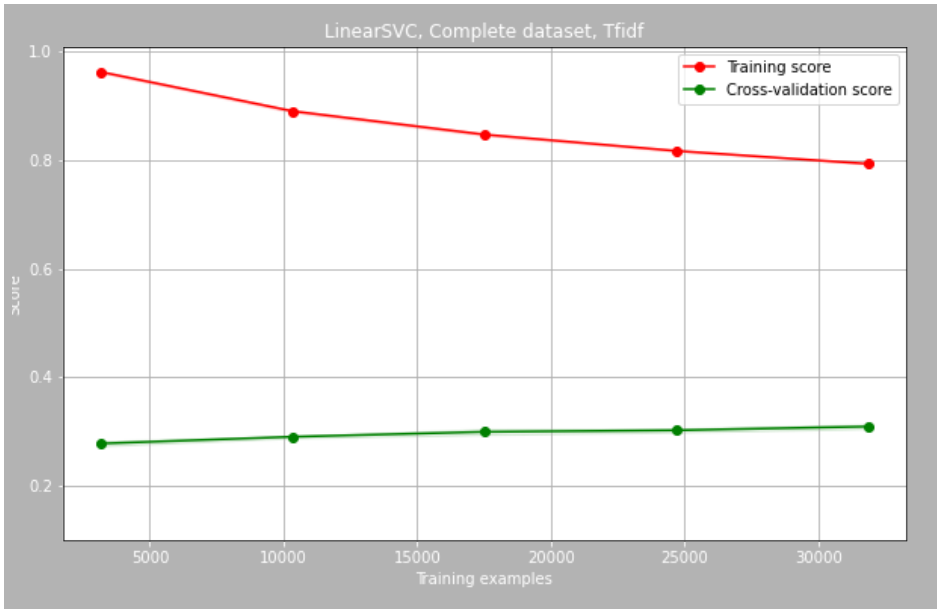


Figure 26: MLP together with spaCy on complete dataset

7.3.5 Learning Curves TF-IDF



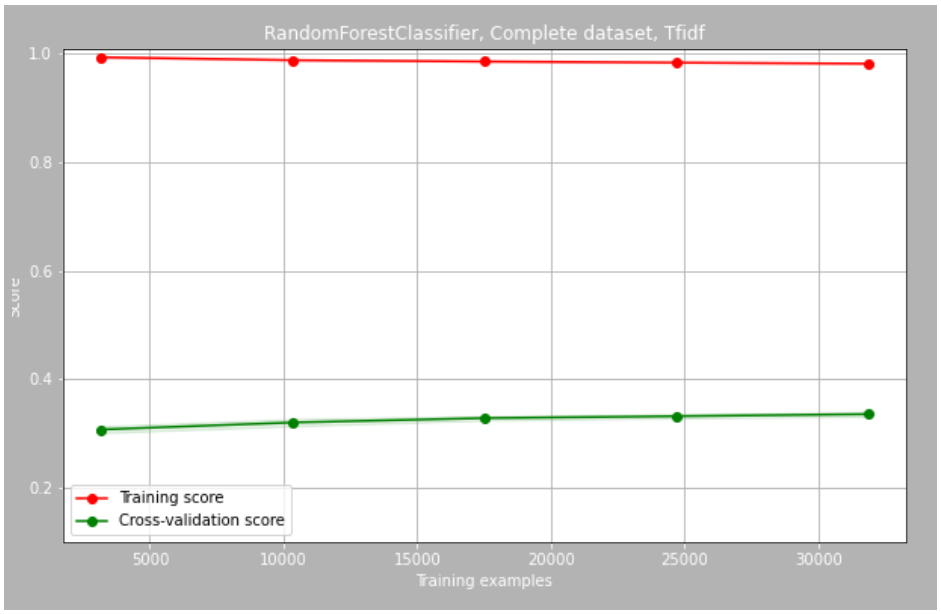


Figure 27: Learning Curves for SVM and RF with TF-IDF on the complete dataset

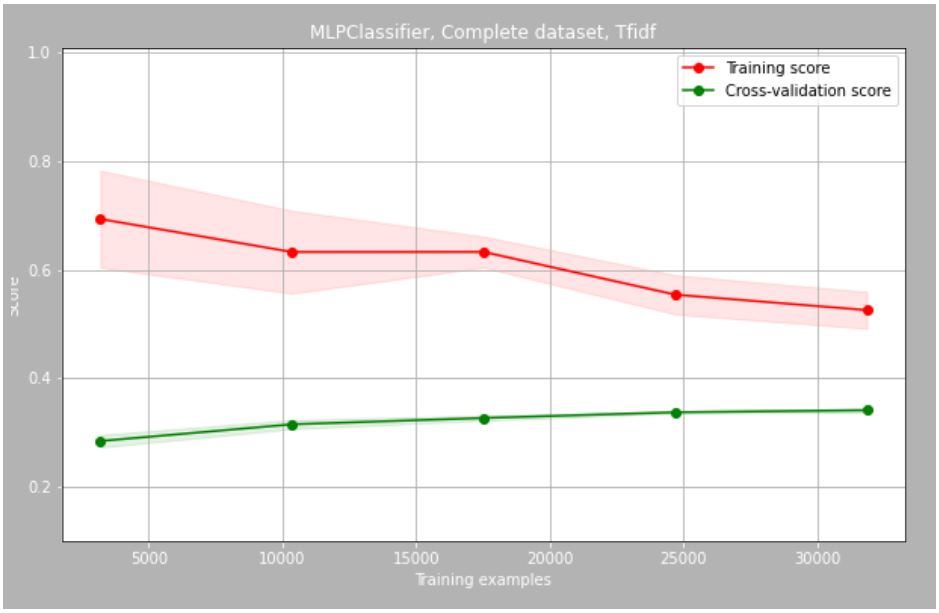
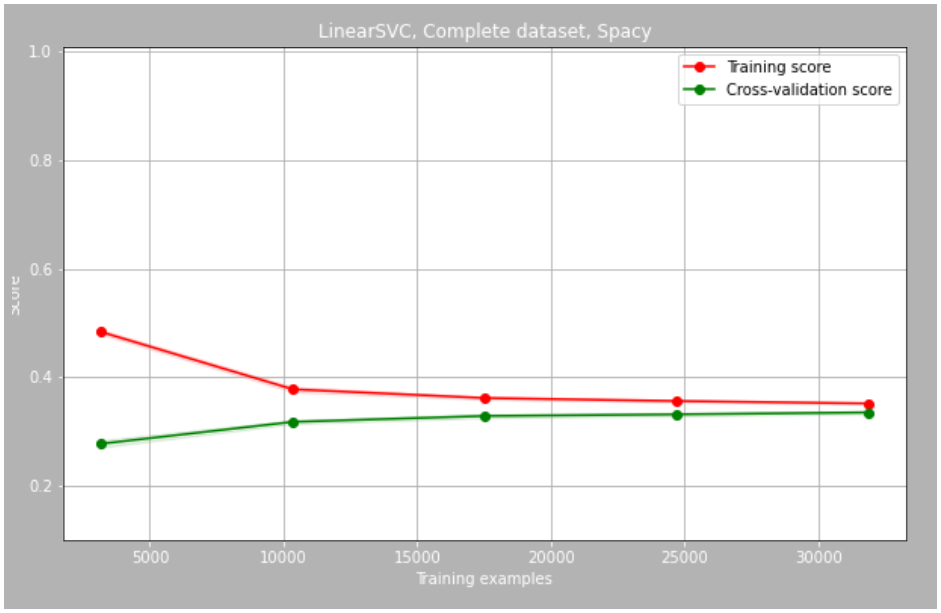


Figure 28: Learning Curves for MLP with TF-IDF on the complete dataset

Expectedly, the learning curves show the inability to learn the data properly with the class imbalance present in the data set. We can see that for both SVM and MLP, the increase in training examples did not help the training score but rather reduced it, signifying that the models had quite the issue with learning many classes at once.

7.3.6 Learning Curves spaCy



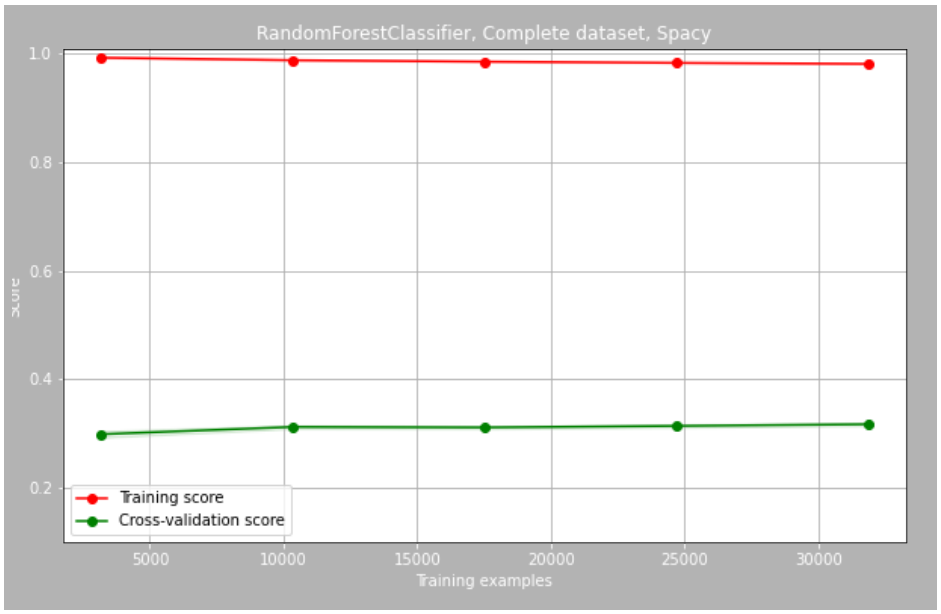


Figure 29: Learning Curves for SVM and RF with spaCy on the complete dataset

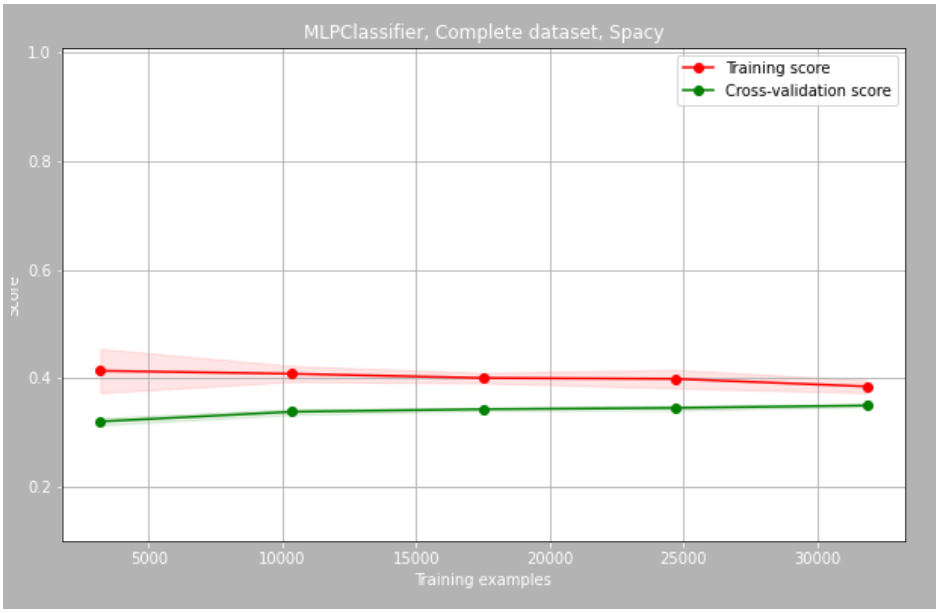


Figure 30: Learning Curves for MLP with spaCy on the complete dataset

Similarly to the previous datasets, we see the same issues arise with the learning ability with spaCy in the complete dataset as well. More data does not seem to increase the training score, as the training plateaus quite early, further proving the inability of the models to be able to adapt to the imbalance of the dataset.

8 Summary and Conclusion

We've seen that the different models can handle the vectorization methods and datasets quite differently. TF-IDF generally seems to perform better with the datasets with the classical algorithms SVM and RF, where as spaCy seems to perform better with MLP than with others, which is unexpected.

We see that spaCy in general performs worse with all models in the binary and ternary dataset, which indicates that spaCy might not be able to learn the data that we possess quite as well. Since spaCy tries to extract the meaning of a tweet, it could have trouble with particularly uncommon words and thus not be able to separate word meanings quite as clearly in the embedding, resulting in lower scores.

Since TF-IDF does not actually try to understand the words appearing but rather just count them, it performs better with our data and can thus just classify them better as well.

With our models we can see that SVM seems to learn the best when fewer data points are available and gradually gets worse when more data is added to the training.
RF seems to overfit on all of our data, whether it is with TF-IDF or spaCy. With TF-IDF we could see that it was able to generalize still quite well, but with spaCy we saw a drop in generalization ability, indicating that RF has trouble learning spaCy. This could be stemming from the fact that spaCy itself has trouble classifying data according to particular meanings as well as RF not learning the complete data but rather just a part of it and classifying based on that.
With MLP we see that it performs the best with spaCy across all datasets. We are not quite sure why specifically MLP is performing the best with spaCy. Through the learning curves we can see that MLP does not learn linearly but fluctuates in its learning, particularly showing that different amounts of data provide different amounts of learning ability. The ternary dataset together with spaCy and MLP show that an increase in data could potentially increase the training score further and allow it to better predict unseen data. The learning curves in the binary and complete dataset show however that this is not always true, as they possess a slight negative slope, indicating that they might be hit a limit on how well they can learn the provided data.

Overall we see that with for our dataset, TF-IDF performs better and stands out to be the better choice to be able to deal with twitter data, particularly if the data is expected to possess a lot of slang or unknown words. With our findings using TF-IDF together with a classical algorithm such as SVM or Random Forest seems to be the preferred choice as well, as these perform better with the TF-IDF vectors. If one is required to use word embeddings however, particularly the spaCy word embeddings, it would be advised to look into neural

networks, as these seem to handle word embeddings better than the classical algorithms.