2nd Assignment: Sentiment Classifier for Movie Reviews

CONTRIBUTORS: DIMITRA TSAKIRI (f3352123), PANTELEIMON SFAKIANAKIS(f3352121), DIMITRIOS GKAVERAS(f3352104), ALEXANDROS SKONDRAS (f3352119)

The link for our Google Colab notebook can be found here.

Exercise 15

For the development of the sentiment classifier for this exercise, movie reviews were used from the Cornell Movie Review Data. More specifically, the dataset polarity dataset v1.1 was used, which contains 1386 movie reviews with average document length of 3562 characters, which are annotated into two mutually exclusive classes as either positive (1) or negative (0).

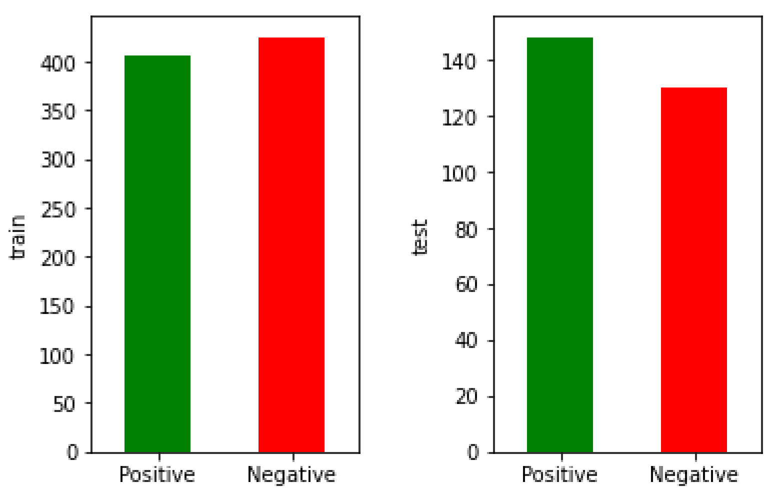
After downloading the files, the preprocessing of the reviews took place. Single characters and multiple spaces, numbers as well as non-word characters were all removed from the reviews. In addition, lemmatization was used in order to group the various inflected forms that a word might have as a single item. At this point, some statistics of the documents are shown in the below table.

Characters per Document		Characters per Negative Class			Characters per Positive Class		
Average	Vocabulary	Average	Min	Max	Average	Min	Max
3562	1382	3348	382	10111	3774	478	9921

The documents were randomly split into train, dev and test sets for the training of our classifiers, as shown below along with the class distribution:

Train	Test	Dev
831	278	277

Class balance of the Train and Test sets



Then, the features from the texts were extracted, using all the single words and bigrams and taking their TF-IDF scores. Stopwords were removed and the 5000 best features were selected according to their TF-IDF scores. Through the stated procedure, the most relevant words and bigrams (with the highest TF-IDF) score will be kept. Consequently, feature selection was applied keeping eventually only 100 features, because the train set consists of only 831 reviews and keeping more features would potentially lead to overfitting. These features where selected based on the dependency between the words.

After the preprocessing phase, a baseline had to be set. For this, the Dummy Classifier was used with 'most frequent' as strategy, which returns the most frequent class label as prediction for every test set item-review.

At this point, since a baseline was set, we chose to run our data through 3 classifiers of different approaches, Logistic Regressor Classifier, K-Nearest Neighbors and MultinomialNB Classifier.

Logistic Regression Classifier:

For the Logistic Regression Classifier, we fine-tuned some parameters of particular interest, solver = ['liblinear', 'lbfgs', 'saga'], C = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.], max_iter = [25, 50, 100, 200, 300], tol = [1e-4,1e-2,1e-6]. The parameter solver defines the algorithm to use in the optimization problem, while the parameter C is the Inverse of regularization strength. Like in support vector machines, smaller values specify stronger regularization. The parameter max_iter shows the maximum number of iterations taken for the solvers to converge, whereas the parameter tol defines the tolerance for stopping criteria. The final parameter values decided upon were solver = "liblinear", C = 1, max_iter = 100, tol=0.0001.

K-Nearest Neighbors:

The parameters fine-tuned in this K-Nearest Neighbors Classifier were n_neighbors, weights and algorithm with Tested Values [5,6,….,59,60], ['uniform','distance'],['auto', 'ball_tree', 'kd_tree', 'brute'] respectively. The parameter n_neighbors, determines the number of neighbors used for the classification, weights defines the weight function used in prediction, and algorithm shows the algorithm used for the computation of the k nearest neighbors.

The best parameter values chosen were n_neighbors=54, weights='uniform', algorithm='auto'.

MultinomialNB Classifier (Naïve Bayes)**:**

In this classifier case, we fine-tuned the parameter alpha with 50 possible values between 0 and 5. Alpha shows Additive Laplace-Lidstone smoothing parameter. We concluded in alpha = 0.204 as the most efficient value.

It is significant to mention, that in all 3 classifier cases F1 score was used as a metric, for the hyperparameter tuning. Also, a condition was applied in the process of parameter hyper-tuning for the kNN classifier, according to which only the parameter combinations that would output a difference of no more than 5% in the F1 score were considered as feasible, so the possibility of overfitting in our model would be significantly reduced.

Learning Curves

For the learning curves visualization, we used the macro averaged F1 score, which is the unweighted mean of the two f1 scores (per class).

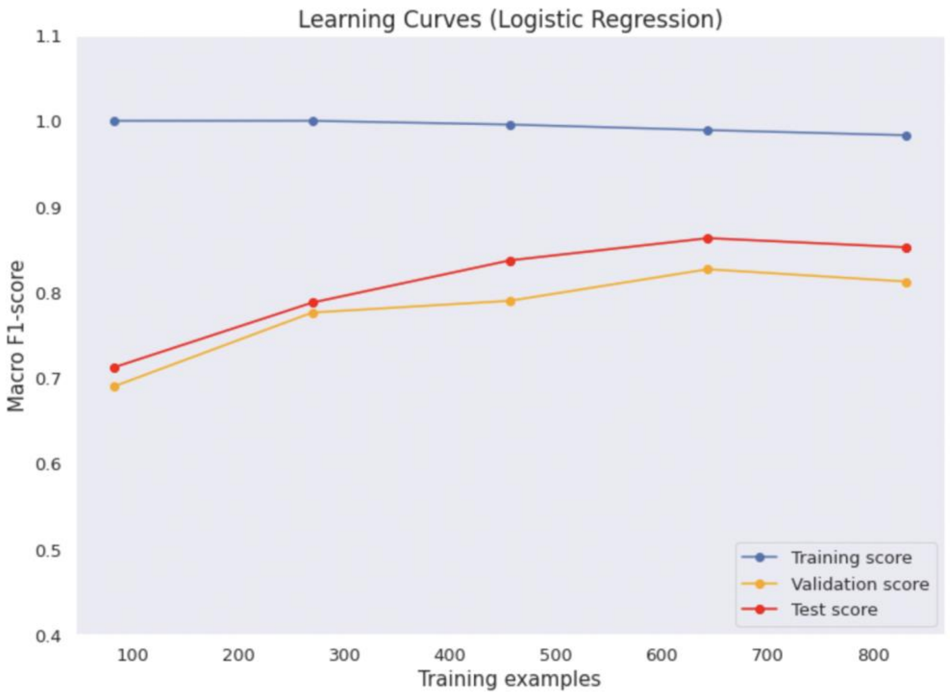


Figure 1

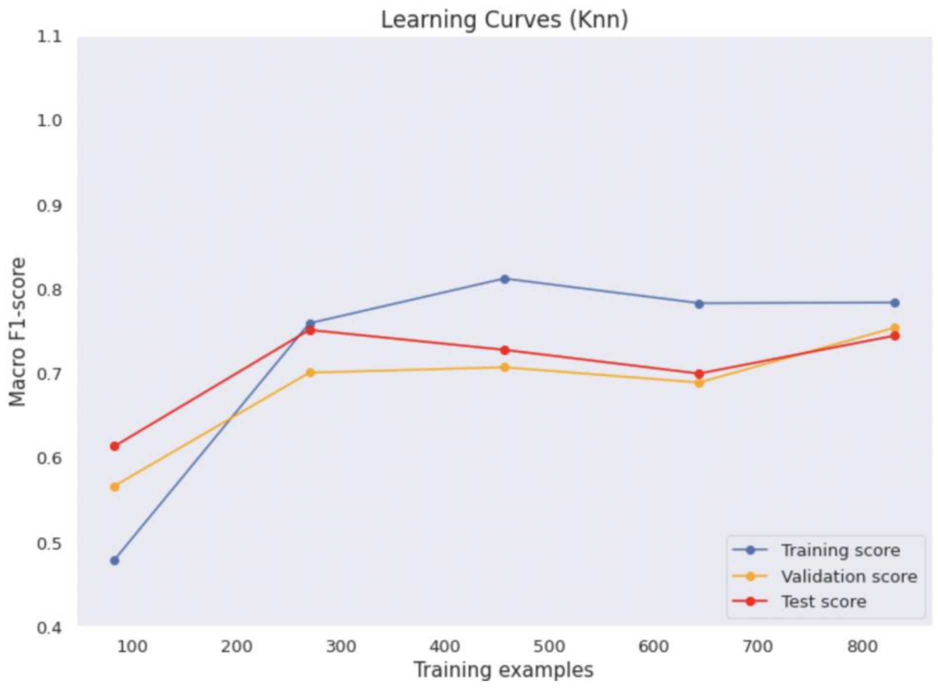


Figure 2

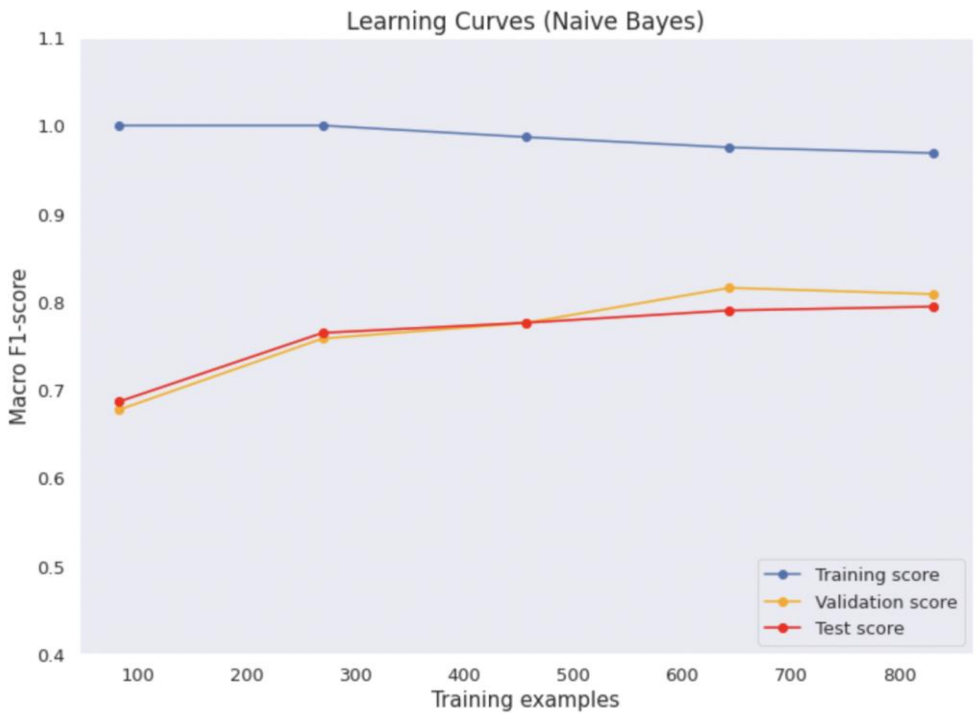


Figure 3

In the learning curves of Logistic Regression and Naive Bayes (figure 1, 3 respectively), it is evident that the blue curve, which shows the performance of the training set is relatively stable and also at the top of the graph. At the same time, the performance of the validation and test sets with yellow and red color curves, respectively, ameliorates as the training examples increase, but does not lead to convergence of the blue curve, which is an indication of overfitting. Now, for the learning curve of kNN (figure 2), it is apparent that the performance of the training set is significantly low for few training examples and incrases rapidly as new training examples are added. This is expected due to the nature of the algorithm, because very few training examples can easily mislead the classifier’s prediction. Despite its lower scores on the validation and test sets, compared to the other classifiers, for this specific fixed number of training examples, 831, the kNN classifier indicates curve convergence, so in this case overfitting is definitely avoided. Finally, it is worth-mentioning that the test curve (red) in the kNN and Logistic Regression classifiers is above the validation one (yellow), which can relate to the fact that cross-validation was not implemented during the learning curves depiction and the hypertuning of the models.

Evaluation of the Classifiers

The following tables show the precision, recall, F1 and precision-recall AUC scores, per class and classifier, separately for the training, development, and test sets along with the macro-averaged precision, recall, F1 and precision-recall AUC scores. Precision can be interpreted as the proportion of the data points our model says were relevant and actually were relevant. Recall can define the ability to find all relevant instances in a dataset. F1 score is the harmonic mean of precision and recall. Lastly, precision recall AUC scores are the area below the precision-recall curves, using various thresholds. For this exercise all these metrics were calculated for both the positive and the negative classes. It is worth noting that the higher these values are simultaneously, the better a model is.

TRAIN

Est.	Precision		Recall		F1		PR-AUC		Macro Average			
	N	P	N	P	N	P	N	P	Prec	Recall	F1	PR-AUC
Base	0.51	0.00	1.00	0.00	0.68	0.00	0.75	0.74	0.26	0.50	0.34	0.75
LR	0.98	0.99	0.99	0.98	0.98	0.98	0.99	0.99	0.98	0.98	0.98	0.99
KNN	0.82	0.75	0.74	0.83	0.78	0.79	0.86	0.86	0.79	0.78	0.78	0.86
NB	0.97	0.97	0.97	0.97	0.97	0.97	0.99	0.99	0.97	0.97	0.97	0.99

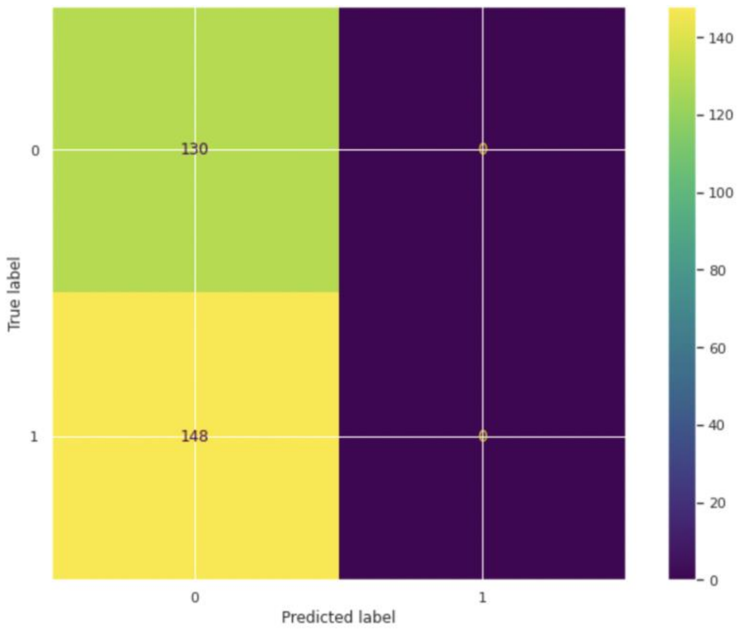
DEV

Est.	Precision		Recall		F1		PR-AUC		Macro Average			
	N	P	N	P	N	P	N	P	Prec	Recall	F1	PR-AUC
Base	0.49	0.00	1.00	0.00	0.66	0.00	0.74	0.75	0.25	0.50	0.33	0.75
LR	0.81	0.82	0.82	0.81	0.81	0.81	0.90	0.87	0.81	0.81	0.81	0.88
KNN	0.79	0.73	0.69	0.81	0.74	0.77	0.82	0.78	0.76	0.75	0.75	0.80
NB	0.79	0.83	0.84	0.78	0.81	0.80	0.85	0.85	0.81	0.81	0.81	0.85

TEST

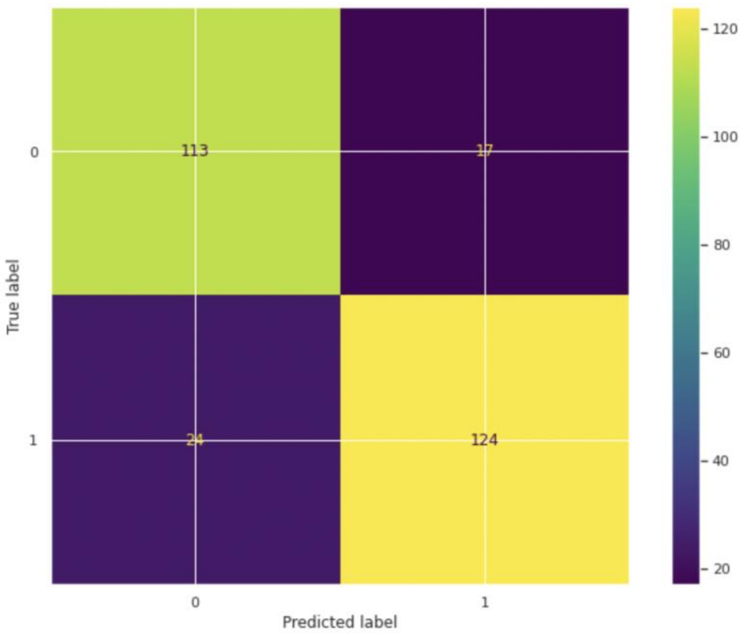
Est.	Precision		Recall		F1		PR-AUC		Macro Average			
	N	P	N	P	N	P	N	P	Prec	Recall	F1	PR-AUC
Base	0.47	0.00	1.00	0.00	0.64	0.00	0.73	0.76	0.23	0.50	0.32	0.75
LR	0.82	0.88	0.87	0.84	0.85	0.86	0.89	0.92	0.85	0.85	0.85	0.91
KNN	0.77	0.73	0.66	0.82	0.71	0.78	0.78	0.85	0.75	0.74	0.74	0.82
NB	0.77	0.81	0.79	0.80	0.78	0.81	0.83	0.88	0.79	0.79	0.79	0.86

BASELINE



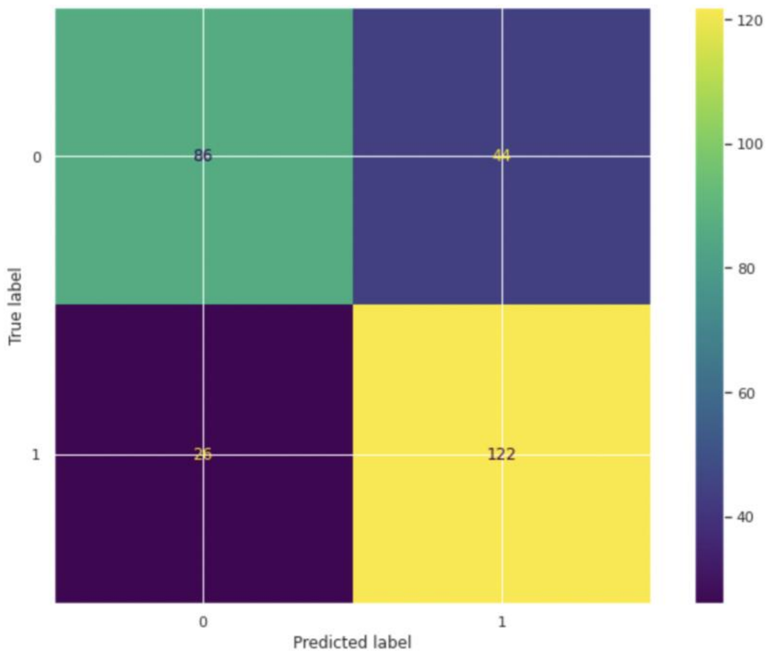
The confusion matrix plot of the classifier we used as baseline is depicted above and it is obvious that it predicts the most frequent class, as mentioned before. Considering the metrics of the 3 metrics tables(blue), it is evident that the classifier always gives the negative class as prediction (recall of the negative class equals to 1 and of the positive to 0).

LOGISTIC REGRESSION



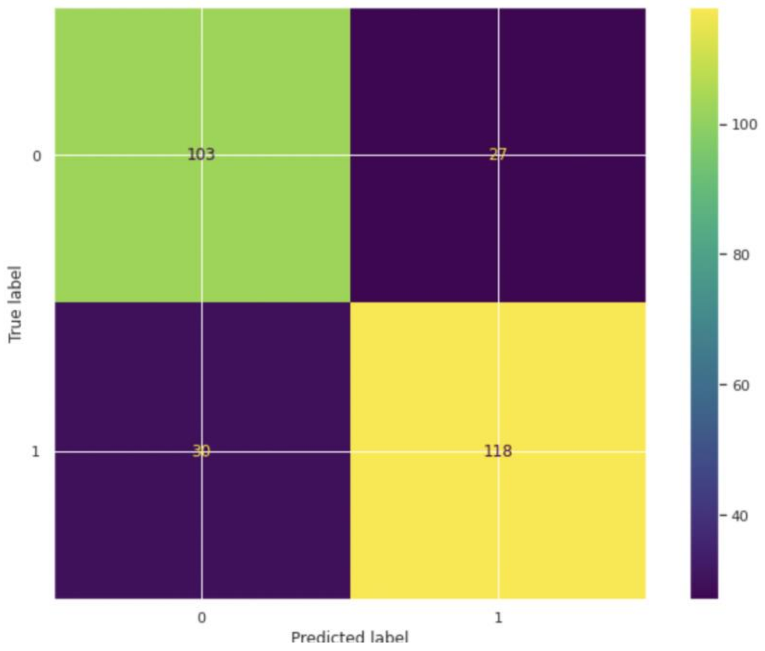
From the respective confusion matrix plot for the Logistic Regression classifier and the metric tables, it can be inferred that LR's metrics appear to be better than those of the baseline, as expected, but, of course, there is overfitting, since the train set's metrics are significantly larger than those of the validation and the test set, e.g. train macro-F1 score=0.98 but test macro-F1 score=0.85.

KNN



The K Nearest Neighbors classifier's metrics are similar for every set of our data (train, dev, test). Complementary to the metrics is the kNN confusion matrix, which shows that the predictions are relatively sufficient. In particular, the predictions for the positive reviews are close to the gold truth, but on the contrary, there are a lot of negative reviews, that are falsely predicted as positive. This is an indication that our model underfits.

NAIVE BAYES



The Naive Bayes classifier has similar metrics with the Logistic Regression classifier, which means that although the test set's scores are seemingly considerably high, they are significantly lower than those of the train set. This, as aforementioned, is an indication of overfitting.

Inference

Based on the evaluation of the classifiers, none of our classifiers are adequate enough. However, we decided that the best performing classifier out of the 3, is K Nearest Neighbor because there is no indication of overfitting and the positively labeled class is predicted significantly well, since its recall is quite high(0.82). Unfortunately, the same cannot be said for the negatively labeled reviews (recall=0.66).

Manner of working on the assignment:

The presented assignment was processed with the equal participation of all team members/contributors. The team worked together in group Discord calls, during which the understanding and structure of the assignment were discussed and cleared upon. Subsequently, more Discord calls took place with one member each time sharing the screen, during which the code was created with the simultaneous attention/participation of all members. The code was each day at night worked upon or modified by each member individually, to eventually reach its final form.

The report was written in a similar manner.