ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

# Assignment 2

Text Analytics

*Submitted by*:

- Panagiotis Antoniozas
- Spyros Mastrodimitris Gounaropoulos
- Panagiota Tavoularea

➢ **You can access our Our Google Colab link here**

## *Introduction*

For this assignment we chose the “movie reviews” text, provided by the sklearn library. In the following tables we present some useful metrics for better understanding the data set.

| ***Number of Total Reviews*** | ***Number of Positive Reviews*** | ***Number of Negative Reviews*** |
|---|---|---|
| 2000 | 1000 | 1000 |

| ***Avg. Number of Characters per Review*** | ***Avg. Number of Characters per Positive Review*** | ***Avg. Number of Characters per Negative Review*** |
|---|---|---|
| 3893 | 3662 | 4124 |

## *Text Preprocessing*

After downloading the data set, we insert our data to the variables X,Y. Variable x is a list and contains all the reviews.Y is an array containing the corresponding annotations for each review (Y=0 : negative review , Y=1 : positive review).

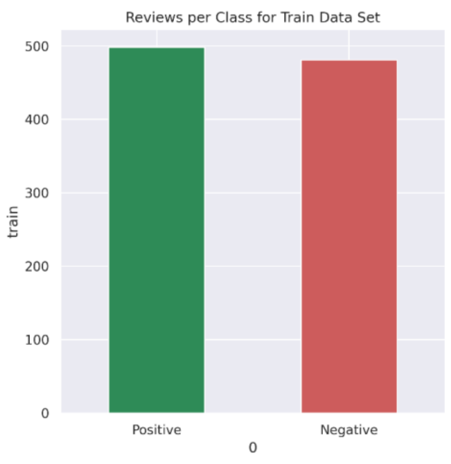
For the text preprocessing we removed all special and single characters. Also, removed all digits, converted the text to lowercase and performed lemmatization (process of reducing words to their root form).

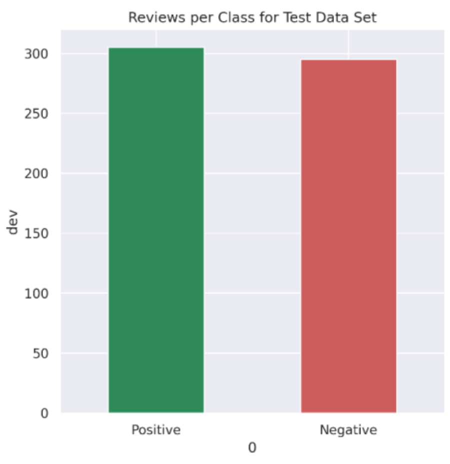
Below, we present an example of sentence for a specific review, before and after the text preprocessing

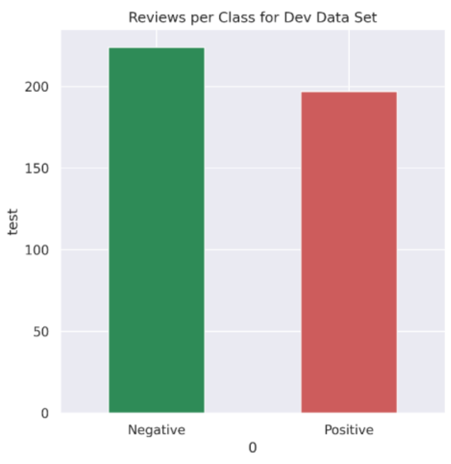
| | |
|---|---|
| ***Before the Text Preprocessing*** | *arnold schwarzenegger has been an icon for action enthusiasts , since the late 80's , but lately his films have been very sloppy and the one-liners are getting worse .* |
| ***After the Text Preprocessing*** | *arnold schwarzenegger ha been an icon for action enthusiast since the late but lately his film have been very sloppy and the one liner are getting worse* |

## *Splitting Data to Train, Test and Dev*

Afterwards, we split our data set to train and test, utilising the 70/30 rule. Then, on the already created train set we also use the 70/30 rule to split them into train and dev sets. In the following charts we present the reviews frequencies per class and per data set.







As we can see, there is class imbalance for each data set, especially for the dev set. Thus, we have to keep in mind that accuracy is not a valid metric to use for the evaluation of our classifiers.

## *TF-IDF Creation / Feature Selection*

For the TF IDF features creation, we implemented the TfidfVectorizer() on all data sets. Also, for our vocabulary, we kept only the top 5000 features, ordered by term frequency across the corpus and ignored the stopwords.

Then, in the feature selection procedure, we used the mutual information criterion in order to identify the most informative words concerning the class (positive or negative). The mutual information is based on entropy estimation from k-nearest neighbours distances.

After calculating the mutual information scores we kept only the 2500 most informative features and these consist of our final vocabulary.

## *Baseline Model*

For the baseline model creation, we trained the algorithm on our train set. The algorithm simply classifies each data point to the class with the most reviews. We utilised such a naive model, in order to function as a yardstick to measure the others’ models' performance.

## *Logistic Regression*

For Logistic regression model we fit the train data set and also performed hyperparameter tuning on dev set  for the below parameters :

**Solver:** solvers are algorithms used to find the optimal parameters for a logistic regression model.These solvers can use different optimization techniques and their main goal is to minimise the cost function and find the best coefficients for the model.

**C:** The parameter "C" is a regularisation hyperparameter. It determines the trade-off between achieving a low training error and a low testing error. A smaller value of C creates a wider margin, which means the model is more forgiving of errors and may have a lower training error but may not generalise well to new data. A larger value of C, creates a narrower margin and increases the emphasis on classifying the training samples correctly, potentially leading to overfitting.

**max iterations:** A higher number of iterations increases the chances of finding the global minimum of the cost function, but also increases the computational time required to train the model. Conversely, a lower number of iterations may result in a faster training process, but may not converge to the optimal solution.

**L1_ratio:** The "l1 ratio" in logistic regression refers to the mixing parameter between L1 and L2 regularisation. It determines the relative weighting between the L1 penalty and the L2 penalty in the cost function. L1 regularisation can lead to sparse solutions, where some of the coefficients are exactly zero, which can be useful for feature selection. On the other hand, L2 regularisation can lead to solutions where the coefficients are small but non-zero, which can help improve the interpretability of the model.

After the parameter tuning process the f1-score on the dev set has a small increase from  84% to 85%

***In conclusion, use F1-score when the balance between precision and recall is more important, otherwise, use the Precision-Recall AUC.***

## *KNN Classifier*

For KNN model we fit the train data set and also performed hyperparameter tuning the :

**K-Number of neighbours:** the "k" refers to the number of nearest neighbours to consider when making a prediction. It determines how many of the closest training samples will be used to make a prediction for a new sample

**weights:** weights refer to the way in which the contribution of each of the k nearest neighbours is determined when making a prediction.

- Uniform weighting: In this scheme, each of the k nearest neighbours contributes equally to the prediction, regardless of their distance from the test sample
  .
- Distance weighting: In this scheme, the contribution of each neighbour is proportional to its inverse distance from the test sample. Closer neighbours have a higher weight, while more distant neighbours have a lower weight.

After the parameter tuning process the f1-score on the dev set has a quite significant increase from 69% to 74%.

## *Evaluation of the Models*

In the following tables we present precision, recall, F1, precision-recall AUC scores, for each class per classifier. We also present in the same tables, the same scores' macro averages per classifier.

### Train Set

| ***Classifier*** | ***Precision*** | | ***Recall*** | | ***F1*** | | ***Precision - Recall AUC*** | | ***Macro-Averages*** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Precision*** | ***Recall*** | ***F1*** | ***Pr -AUC*** |
| **Baseline** | 0.52 | 0 | 1 | 0 | 0.67 | 0 | 0.75 | 0.74 | 0.25 | 0.50 | 0.34 | 0.75 |
| **Logistic Regression** | 0.97 | 0.99 | 0.99 | 0.97 | 0.98 | 0.98 | 0.99 | 0.99 | 0.98 | 0.98 | 0.98 | 0.99 |
| **KNN** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### Test Set

| ***Classifier*** | ***Precision*** | | ***Recall*** | | ***F1*** | | ***Precision - Recall AUC*** | | ***Macro-Averages*** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Precision*** | ***Recall*** | ***F1*** | ***Pr -AUC*** |
| **Baseline** | 0.51 | 0 | 1 | 0 | 0.67 | 0 | 0.75 | 0.74 | 0.25 | 0.50 | **0.34** | **0.75** |
| **Logistic Regression** | 0.83 | 0.87 | 0.88 | 0.81 | 0.85 | 0.84 | 0.93 | 0.92 | 0.85 | 0.84 | **0.84** | **0.93** |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **KNN** | 0.72 | 0.81 | 0.85 | 0.65 | 0.78 | 0.72 | 0.85 | 0.82 | 0.76 | 0.75 | **0.75** | **0.84** |

**Development  Set**

| ***Classifier*** | ***Precision*** | | ***Recall*** | | ***F1*** | | ***Precision - Recall AUC*** | | ***Macro-Averages*** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Pos*** | ***Neg*** | ***Precision*** | ***Recall*** | ***F1*** | ***Pr -AUC*** |
| **Baseline** | 0.47 | 0 | 1 | 0 | 0.64 | 0 | 0.7 | 0.76 | 0.23 | 0.5 | 0.32 | 0.75 |
| **Logistic Regression** | 0.82 | 0.87 | 0.86 | 0.83 | 0.84 | 0.85 | 0.92 | 0.93 | 0.85 | 0.85 | 0.85 | 0.93 |
| **KNN** | 0.68 | 0.82 | 0.84 | 0.66 | 0.75 | 0.73 | 0.8 | 0.84 | 0.75 | 0.75 | 0.74 | 0.82 |

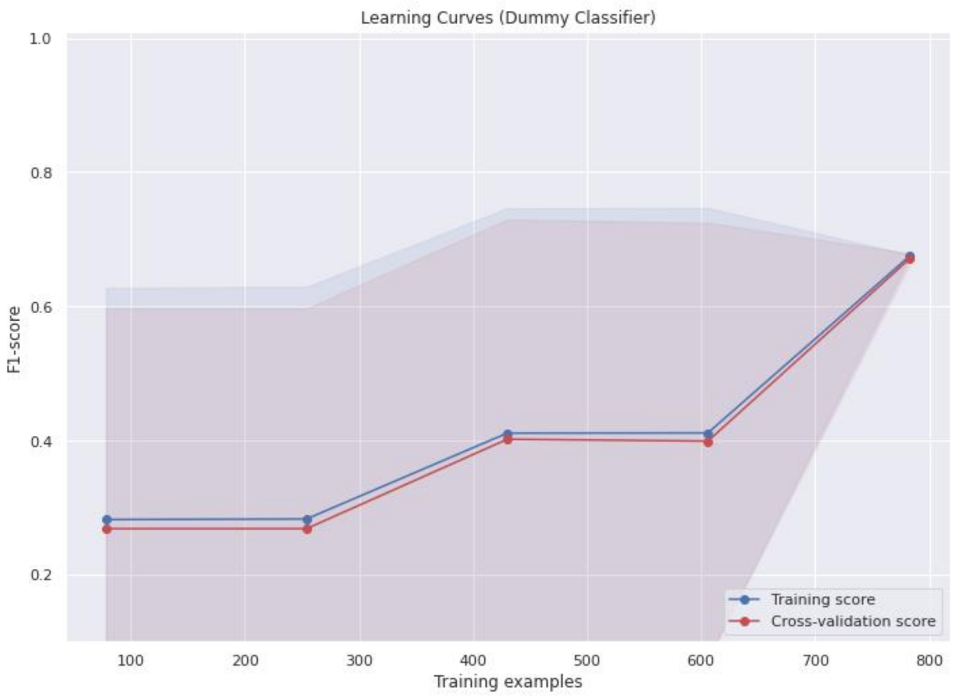
We omit references to the dummy classifier as it is only there to set a baseline. The logistic regression classifier shows in general promising results. Its macro average F1 is around 85% for the test and dev sets and up to 98% for the training. The model is proficient in learning the training data  in detail and is capable of classifying successfully unknown examples with high accuracy for both classes. Although the fact that the two F1 scores differ to such a degree gives a hint of possible overfitting something that is analysed further in the next section.

The KNN model lacks both precision and recall in both classes and that is further spotted on the macro average F1. The difference between the training and test results are further exaggerated than the logistic regression so it is highly probable that the model suffers from overfitting. In conclusion every aspect of the logistic regression is superior to the KNN.

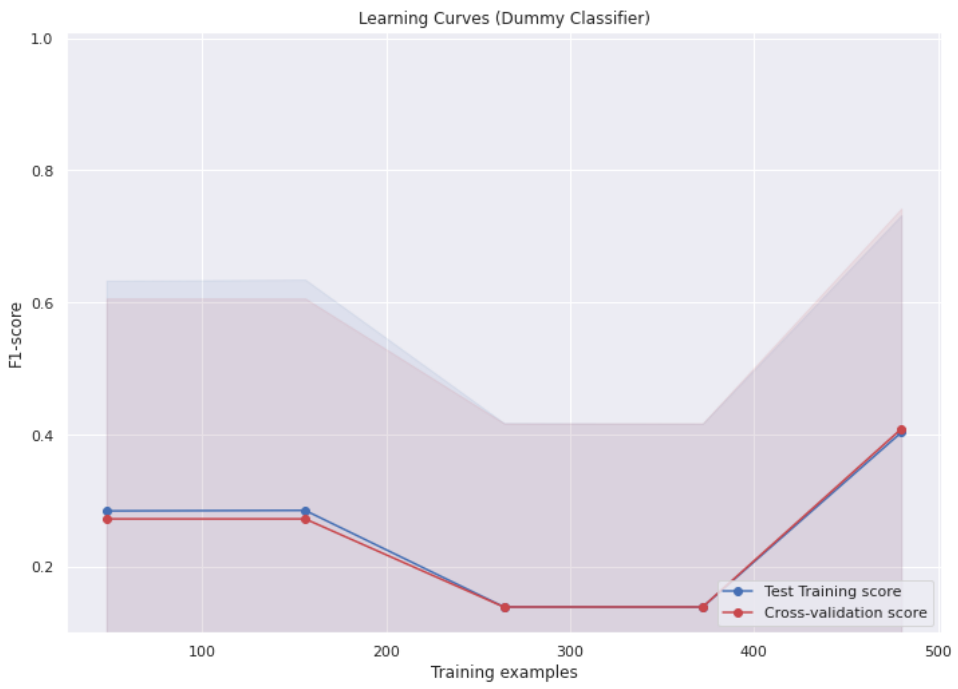
## *Evaluation of the Learning curves*

### *Baseline Model*

For the baseline model it is important to mention that as the classification depends solely on the most numerous data category the idea of a learning rate is not accurate. The perceived learning rate corresponds to the sequence and the percentage of the members of the most frequent class. For this reason the learning curves for the baseline classifier are unimportant compared to those of the other two classifiers. Although as it was asked for the exercise report below it is presented the “learning rate” of the dummy classifier for the train and test sets. The dev graph is available in the collab link.

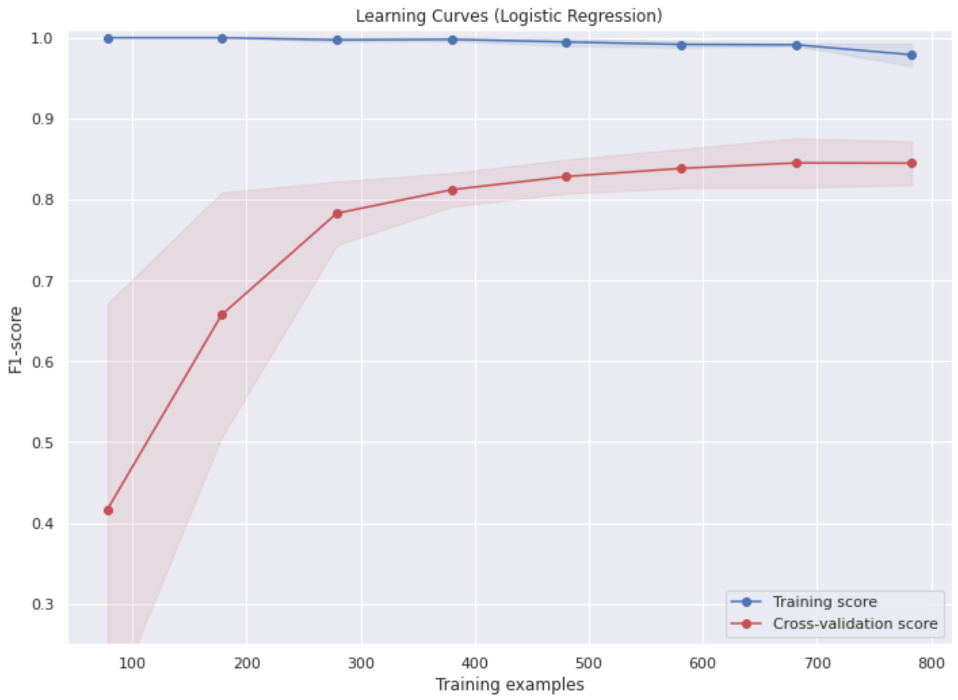


Learning Curves of the train set of the Dummy classifier

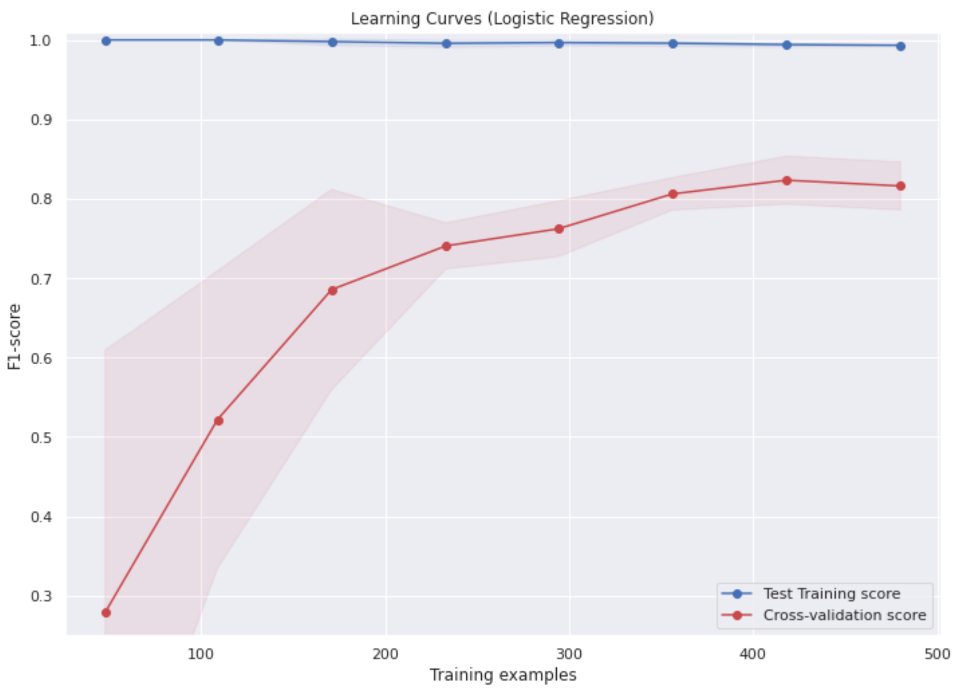


Learning Curves of the test set of the Dummy classifier

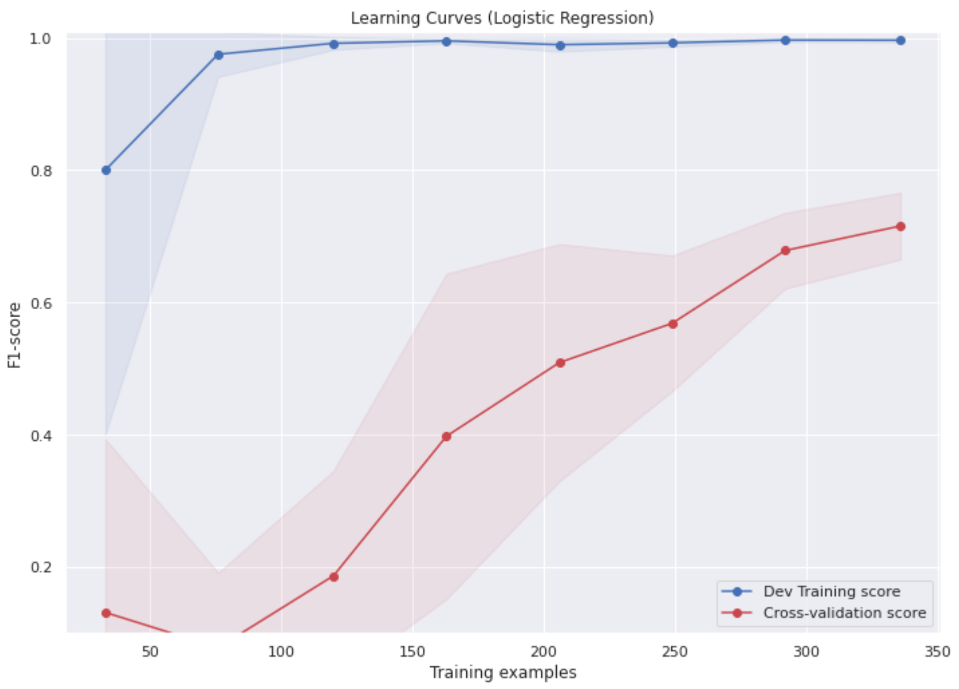
## *Logistic Regression*



Learning Curves of the train set of the Logistic Regression Classifier



Learning Curves of the test set of the Logistic Regression Classifier



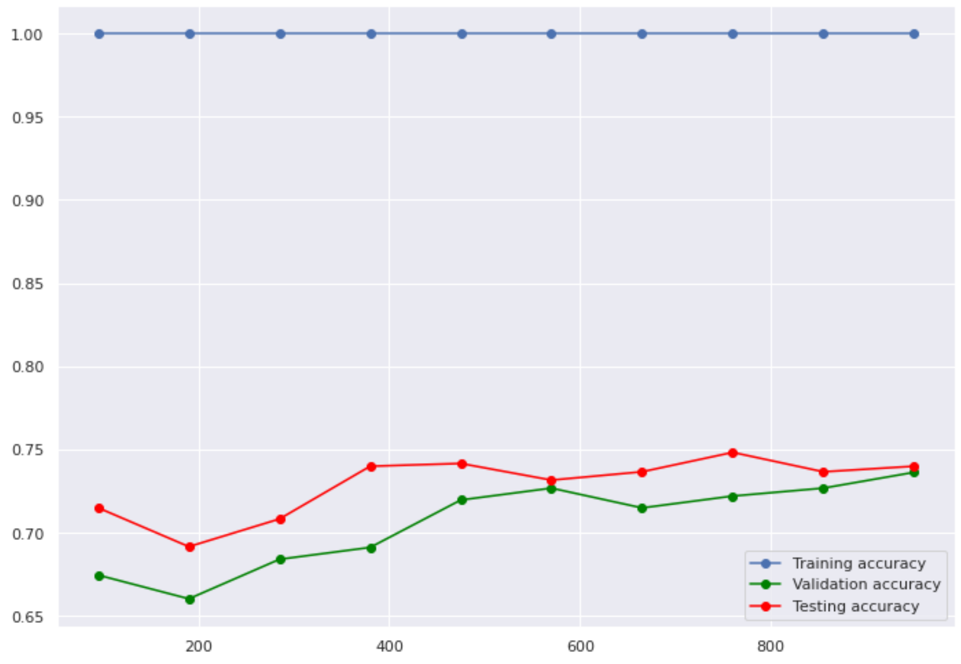
Learning Curves of the dev set of the Logistic Regression Classifier

The graphs representing the learning curves of train, test and dev sets for the logistic regression offer generous insight about the condition of the model and its efficacy. For the test and dev sets it is apparent that as the model trains on more and more data its cross-validation score increases. This is more pronounced in the dev as it reaches its best performance at its latest inputs. In conjunction with the train set graph we can assume that with larger dev sets their cross-validation would be higher.

Moreover the train test graph shows that between 600 and 800 data points it reaches a plateau and it is possible that that is the peak performance of the model although that is not certain. This uncertainty is based on the fact that the training set's accuracy is slowly declining. It would not be inaccurate to hypothesise that by using a larger data set a greater performance might be possible and the observed plateau only a local maximum.

Beyond that slight training score decrease the rest of the clues point to a possible overfitting since the difference between training and cross validation scores is not insignificant. After those conclusions possible solutions might be a decrease of the features used, an increase of the hyperparameter λ, or even a use of a different classifier even though that might be excessive and counterproductive considering the already high F1 score already achieved by the model.

## *KNN Classifier*



Learning Curves of the train, test and dev sets of the KNN Classifier

In the above graph the information can be condensed into the following observations. The training accuracy is perfect, the test set accuracy is better than the validation set accuracy and both improve as the number of samples increases. The last is logical as with larger train data the results improve. The second point is also rational as the size of the test set is larger than the validation set. The first is the most troubling. The most logical explanation is a serious overfitting problem where the model can describe in perfect detail the data that was trained with but fails to do the same with the test and dev set. In response, a decrease of the number of features selected or an increase of the size of the training set are a pair of straightforward solutions.