M.Sc. in Data Science

Text Analytics

Assignment 2 - Multilayer Perceptrons

Grammatikopoulou Maria - f3352310
Phevos A. Margonis - f3352317
Moniaki Melina - f3352321

Instructor: Ion Androutsopoulos
Grader: Foivos Charalampakos

February 12, 2024

Abstract

This paper investigates sentiment analysis and part-of-speech tagging using Multi-Layer Perceptron (MLP) classifiers. Employing the IMDB Dataset for sentiment analysis and the English-GUM dataset from Universal Dependencies treebanks for part-of-speech tagging, the study demonstrates significant advancements over baseline models. Through meticulous data preprocessing and hyperparameter tuning, the MLP models achieved notable accuracy improvements.

Contents

1 Exercise 9

1.1 Introduction

Text classification is a fundamental task in natural language processing with applications ranging from sentiment analysis to topic categorization. In this report, we perform a sentiment analysis using Multi-Layer Perceptron (MLP) classifiers implemented in deep learning frameworks such as Keras/TensorFlow or PyTorch. The objective is to build and evaluate an MLP classifier, exploring various hyperparameters, and comparing its performance against baseline models.

1.2 Dataset

We conducted our analysis on the IMDB Dataset, consisting of 50,000 movie reviews, each annotated with sentiment labels (positive or negative). Our initial exploration revealed a balanced dataset, evenly distributed with 25,000 positive and 25,000 negative reviews.

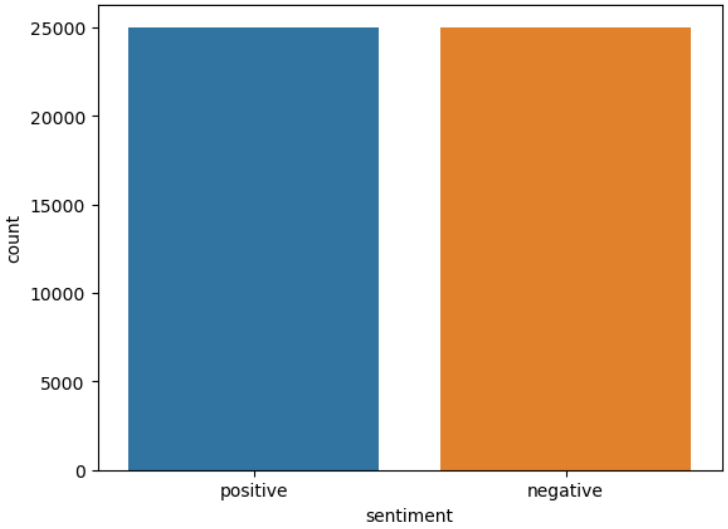

Figure 1: Number of Reviews per Class.

1.3 Data Preprocessing

Initially, we removed HTML tags from our dataset that were in the form of ¡br /¿.All text was converted to lowercase to establish uniformity and simplify subsequent processing. Following this, we proceeded to remove punctuation marks to enhance text consistency, as well as numerical digits and any single characters that remained after the removal of apostrophes. Multiple spaces and common stop-words were also removed from the dataset to eliminate non-essential words. Moving forward, we transformed sentiment labels into binary values. Positive sentiment was encoded as 1, while negative sentiment was encoded as 0. Then, we split the data set into training (35000 samples), validation (7500 samples) and test set (7500 samples). The Spacy library was used for sentence splitting and tokenization. For each tokenized word we generated TF-IDF features and then we employed SVD for dimensionality reduction, which reduced the feature space from 5000 dimensions to 500 dimensions.

1.4 Baseline Classifiers

Majority Classifier: We implemented a Majority Baseline Classifier using the Dummy Classifier from scikit-learn. This classifier predicts the most frequent class within the training set. The subsequent classification reports offer a comprehensive summary of precision, recall, and F1 score for each class across the training, validation, and test sets correspondingly. Moreover, the Precision-Recall AUC (PR-AUC) scores provide valuable insights into the classifier's capability to distinguish between classes in terms of precision and recall.

Table 1: Majority Classifier: Performance Metrics by Class across Data Subsets

Class	Train				Validation				Test			
	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC
Negative	0.50	1.00	0.67	0.75	0.50	1.00	0.66	0.75	0.49	1.00	0.66	0.75
Positive	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.75
Macro Avg	0.25	0.50	0.33	0.75	0.25	0.50	0.33	0.75	0.25	0.50	0.33	0.75

Logistic Regression Classifier: The Logistic Regression model serves as an additional baseline for our text classification task. After training the logistic regression model, we observe the following result for the training, validation and test set subsequently.

Table 2: Logistic Classifier: Performance Metrics by Class across Data Subsets

Class	Training				Development				Test			
	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC
Negative	0.89	0.87	0.88	0.96	0.89	0.86	0.87	0.95	0.89	0.87	0.88	0.95
Positive	0.87	0.90	0.88	0.95	0.87	0.875	0.88	0.95	0.87	0.89	0.88	0.95
Macro Avg	0.88	0.88	0.88	0.95	0.88	0.88	0.88	0.95	0.88	0.88	0.88	0.95

1.5 Multi-Layer Perceptron (MLP) Classifier:

A MLP classifier is implemented to improve the results of the above baseline models.

Model Architecture The MLP classifier utilizes a Sequential model with multiple layers. The architecture includes a densely connected layer with 384 units and Rectified Linear Unit (ReLU) activation. A dropout layer with a rate of 0.2 is incorporated to mitigate overfitting. The final layer consists of a single unit with a sigmoid activation function, suitable for binary classification. The hyperparameters were derived using the library keras-tuner, which essentially implements a Random Search approach for the tunning.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 384)               192384

 activation (Activation)     (None, 384)               0

 dropout (Dropout)           (None, 384)               0

 dense_1 (Dense)             (None, 1)                 385

=================================================================
Total params: 192769 (753.00 KB)
Trainable params: 192769 (753.00 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 2: MLP Architecture

Model Training The model is trained using binary cross-entropy loss and the Adam optimizer with a learning rate of 0.001. The training process spans 100 epochs, with a batch size of 128, and employs early stopping to prevent overfitting. This ensures that the model generalizes well to unseen data.

Model Evaluation Evaluation of the MLP classifier involves assessing its performance on the training, validation, and test sets. Classification reports are generated, providing a comprehensive overview of precision, recall, and F1 score for each class. Precision-Recall AUC scores are computed

to evaluate the model's precision and recall trade-offs. This analysis is particularly relevant for imbalanced datasets, providing a nuanced understanding of the model's performance beyond accuracy.

Table 3: MLP Classifier: Performance Metrics by Class across Data Subsets

Class	Training				Development				Test			
	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC
Negative	0.99	0.99	0.99	1.00	0.89	0.87	0.88	0.95	0.89	0.86	0.88	0.95
Positive	0.99	0.99	0.99	1.00	0.87	0.90	0.88	0.95	0.87	0.90	0.88	0.95
Macro Avg	0.99	0.99	0.99	1.00	0.88	0.88	0.88	0.95	0.88	0.88	0.88	0.95

Learning Curves Learning curves illustrate the model's accuracy (see Figure 3) and loss (see Figure 4) over epochs for both the training and validation sets. These curves offer insights into the model's convergence and potential overfitting or underfitting issues.

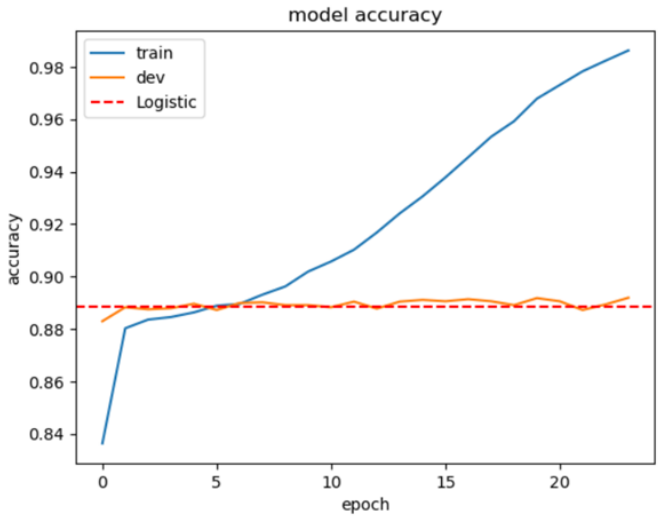

Figure 3: Accuracy Curves for Training and Validation Data as a Function of Training Epoch

1.6 Conclusions

As a result of the MLP implementation, we observe that the model achieved impressive validation and test accuracies, reaching 89%. Additionally, the Precision-Recall Area Under the Curve (AUC) approached 96%, indicating a strong ability to balance precision and recall. Furthermore, we observe that the model achieves high accuracy from the very first epoch, as we can see from the accuracy plot. For this reason, the model cannot improve the validation accuracy past the 14^{th} epoch, and thus the early stopping terminates the training phase at the 24^{th} epoch. Comparing the MLP classifier with the baseline models, it outperformed the Majority Classifier significantly. However, its performance closely resembled that of the Logistic Regression baseline.

2 Exercise 10

2.1 Introduction

Part-of-speech (POS) tagging is the process of assigning a part-of-speech label, such as noun, verb, adjective, etc., to each word in a given text, based on both its definition and its context. Context is essential for accurately classifying a token's part-of-speech because it clarifies the word's specific meaning and grammatical role, given its potential to vary based on surrounding words.

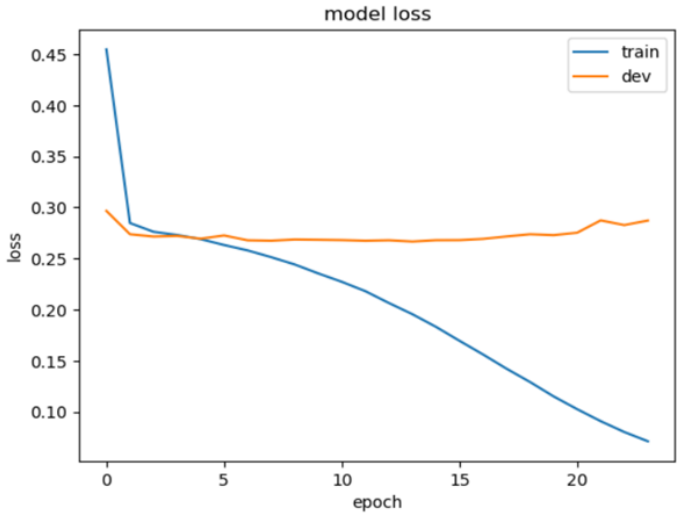

Figure 4: Loss Curves for Training and Validation Data as a Function of Training Epoch

2.2 Dataset

The language of choice for this exercise is English. A collection of annotated text corpora across multiple languages, standardized to represent grammatical information and syntactic relations, is provided by the Universal Dependencies treebanks. From this collection, the English corpus that will serve as the foundation for this study, provided by Georgetown University Multilayer, can be found here. This corpus contains 10,761 sentences, 184,373 tokens[1], and 187,417 syntactic words. More details about the dataset's intricacies can be further explored here.

Universal Dependencies have standardized CONLLU as their file format of choice. The main difference from a plain TEXT file is the inclusion of metadata for each token like POS Tags, Features, Relations, etc., and for that reason, the dedicated Python library conllu is required to parse them.

While parsing the data, the features that are extracted are (a) the token to be classified, (b) its preceding and subsequent words, forming a context window of $n = 3$, and (c) the token's UPOS label. The UPOS labels denote 17 distinct classes (i.e., ADJ, ADP, ADV, AUX). To account for the first and last word in a sentence, and by extension their context window, a pseudo-token *PAD* is prepended and appended accordingly. Hence, a DataFrame is created where each row holds the above information for each available token. Lastly, the DataFrame is stored as CSV to streamline the following steps. This process is repeated for all three available Train-Validation-Test datasets.

2.3 Pre-processing

Every successful model requires a rigorous and iterative exploratory process. The initial finding of this process was a discrepancy in the formulation of the CONLLU files. Every contraction in this format is stored three times: once to indicate the full word without metadata, once for the auxiliary word, and once for its subsequent part. Every line in the DataFrame containing the sum of these parts has, therefore, no label (due to missing metadata) and as a result, the action of choice is their removal. Consequently, the final length of the Training dataset is 147,950; of Validation, 19,653; and of the Test, 19,917 (which roughly equates to 80%-10%-10%).

Following this, every word needs to be translated into a machine-readable format. Since this study is centered around the role of each individual token, the chosen embedding strategy will follow the use of pre-trained *Word2Vec* (abbreviated as w2v) embeddings from the word2vec-google-news-300 model. To account for the pseudo-token *PAD*, a unique embedding will be assigned manually and is designed to resemble those of w2v, signifying the start/end of a sentence. Further exploration proved that every token belonging to the class *PUNCT* and *SYM* were not present in the vocabulary of w2v and, as a result, had no available embeddings. These tokens are also known as Out-Of-Vocabulary tokens. Thus, they were treated in the same way as *PAD*. A special case is the embedding of the dash (-) character. This token is context-specific and can be found both as *PUNCT* and *SYM*, so the embedding of choice is the centroid of the two candidate classes, in

[1] https://www.datacamp.com/blog/what-is-tokenization

order to keep them close in the feature space. Likewise, numbers (*NUM*) over 9 have no embedding. Consequently, every OOV number receives the embedding of the number *2*. This convention is followed because the magnitude of a number is independent of its class. Similarly, w2v seems to have omitted some stop-words in its model and thus, words belonging to the populous class *ADP* (like *of* and *to*) are OOV. Lastly, every other OOV word receives as embedding a vector of zeros. In conclusion, this preprocessing step is essential, as it will increase the overall accuracy by as much as 10%.

The process continues with the concatenation of each embedding of the row to create the context window. This approach ensures that the word order is preserved. Ultimately, the target labels are encoded, from their original *word* format, to integer labels, and then to one-hot vectors. The former are useful for several scikit modules, the latter are essential to the Multilayer Perceptrons (MLP) that will follow.

2.4 Baselines

Before proceeding to more complicated classifiers, two simpler ones are created that will later act as baselines. Firstly, a weak baseline is created using a *Majority* classifier, that assigns any new data point to the most common class that was found within the training set. The accuracy achieved using this approach, in the validation set, is only 17%, and the Precision-Recall AUC is only 52%. This is expected in a case of 17 different classes. A more detailed view is presented in Table 4.

Table 4: Majority Classifier: Performance Metrics by Class across Data Subsets

Class	Train				Validation				Test			
	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC
ADJ	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.53
ADP	0.00	0.00	0.00	0.54	0.00	0.00	0.00	0.54	0.00	0.00	0.00	0.55
ADV	0.00	0.00	0.00	0.52	0.00	0.00	0.00	0.52	0.00	0.00	0.00	0.52
AUX	0.00	0.00	0.00	0.52	0.00	0.00	0.00	0.52	0.00	0.00	0.00	0.52
CCONJ	0.00	0.00	0.00	0.51	0.00	0.00	0.00	0.51	0.00	0.00	0.00	0.51
DET	0.00	0.00	0.00	0.54	0.00	0.00	0.00	0.54	0.00	0.00	0.00	0.54
INTJ	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50
NOUN	0.17	1.00	0.29	0.58	0.17	1.00	0.29	0.58	0.18	1.00	0.30	0.58
NUM	0.00	0.00	0.00	0.51	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.51
PART	0.00	0.00	0.00	0.51	0.00	0.00	0.00	0.51	0.00	0.00	0.00	0.51
PRON	0.00	0.00	0.00	0.54	0.00	0.00	0.00	0.54	0.00	0.00	0.00	0.53
PROPN	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.52	0.00	0.00	0.00	0.53
PUNCT	0.00	0.00	0.00	0.56	0.00	0.00	0.00	0.56	0.00	0.00	0.00	0.56
SCONJ	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50
SYM	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50
VERB	0.00	0.00	0.00	0.55	0.00	0.00	0.00	0.55	0.00	0.00	0.00	0.55
X	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.50
Macro Avg	0.01	0.06	0.02	0.52	0.01	0.06	0.02	0.52	0.01	0.06	0.02	0.52

Secondly, a *Logistic* classifier is designed as a strong baseline. Utilizing the lbfgs solver, which can handle multi-class problems, and increasing the maximum iterations to 1000, allowing the model ample time to converge, the validation accuracy is 92% and the Precision-Recall AUC 91%. A more detailed view is presented in Table 5.

2.5 Multilayer Perceptron Model

Aiming to improve upon those baselines, a Multilayer Perceptron model is implemented. The design utilized the *Sequential* API to add:

- A hidden layer with 128 neurons and the *relu* activation function,
- A dropout layer with a probability of 0.4,
- A second hidden layer with 128 neurons and the *tanh* activation function,
- A dropout layer with a probability of 0.1,
- An output layer with 17 neurons, one for each class, and the *softmax* activation function.

Table 5: Logistic Classifier: Performance Metrics by Class across Data Subsets

Class	Train				Validation				Test			
	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC
ADJ	0.90	0.89	0.90	0.94	0.86	0.88	0.87	0.92	0.87	0.87	0.87	0.93
ADP	0.96	0.98	0.97	0.99	0.94	0.97	0.95	0.98	0.95	0.97	0.96	0.98
ADV	0.91	0.91	0.91	0.96	0.87	0.87	0.87	0.93	0.89	0.86	0.88	0.94
AUX	0.97	0.98	0.97	0.99	0.96	0.98	0.97	0.99	0.96	0.98	0.97	0.99
CCONJ	0.82	0.89	0.85	0.93	0.81	0.85	0.83	0.90	0.78	0.85	0.81	0.88
DET	0.95	0.96	0.96	0.99	0.94	0.95	0.95	0.99	0.93	0.95	0.94	0.98
INTJ	0.91	0.79	0.84	0.91	0.85	0.72	0.78	0.86	0.90	0.80	0.85	0.88
NOUN	0.90	0.93	0.92	0.96	0.91	0.92	0.92	0.96	0.88	0.92	0.90	0.95
NUM	0.98	0.96	0.97	0.99	0.98	0.93	0.96	0.98	0.98	0.96	0.97	0.98
PART	0.96	0.90	0.93	0.97	0.93	0.88	0.91	0.95	0.89	0.83	0.86	0.92
PRON	0.99	0.99	0.99	0.99	0.98	0.99	0.98	0.99	0.97	0.98	0.98	0.99
PROPN	0.78	0.73	0.75	0.83	0.73	0.69	0.71	0.79	0.77	0.69	0.73	0.81
PUNCT	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.99
SCONJ	0.91	0.82	0.86	0.93	0.83	0.74	0.78	0.86	0.83	0.79	0.81	0.87
SYM	0.94	0.75	0.84	0.88	1.00	0.57	0.73	0.98	0.76	0.65	0.70	0.78
VERB	0.94	0.94	0.94	0.97	0.93	0.91	0.92	0.96	0.93	0.90	0.92	0.96
X	0.88	0.34	0.49	0.60	0.50	0.33	0.40	0.36	0.87	0.46	0.60	0.54
Macro Avg	0.92	0.87	0.89	0.93	0.88	0.83	0.85	0.91	0.89	0.85	0.87	0.90

The above hyperparameters were derived using the library keras-tuner, which essentially implements a *RandomSearch* approach for tuning. The *loss* function for this multi-class problem is selected to be the categorical_crossentropy, which compares the output probabilities of the MLP with the one-hot vectors of the target class. Next, the Adam optimizer is used to handle the backpropagation that will compute the weights, and the metric by which the model will be assessed is set to accuracy. Following that, the ModelCheckpoint is used to monitor the accuracy on the validation set during training and store the model that scores the highest. During the training phase, the batch size is set to compute 256 training data simultaneously, and the maximum number of epochs is set to 100. At the end of each epoch, the class callback computes the metrics accuracy, precision, recall, and F1 score for both training and validation data. Then, it checks if the resulting model is better than the previous. Finally, if the model has not improved its validation accuracy during the last 10 epochs, the class early_stopping terminates the training phase prematurely.

As a result, the Multilayer Perceptron model achieves a validation accuracy of almost 94% and a Precision-Recall AUC close to 93%. A detailed view can be seen in Table 6. Furthermore,the model's accuracy (see Figure 5) and loss curves (see Figure 6) are plotted as a function of the training epochs. In both cases, the validation curve seems almost flat, achieving remarkable results from the very first epoch. This phenomenon can be attributed to the disproportionately large training set, which guided the validation set to a good solution during the evaluation of its mini-batches.

2.6 Conclusions

1. Data pre-processing is an essential step that can significantly enhance the final outcomes.
2. The Majority classifier is not a suitable baseline for this multinomial problem involving 17 classes.
3. The Logistic Regression classifier serves as a robust baseline, yielding impressive results.
4. The Multilayer Perceptron surpassed the robust baseline with a notable difference.
5. The final architecture of the MLP is relatively shallow, suggesting that the problem could be considered relatively straightforward.
6. The low Recall of class `X` could be improved with better preprocessing for the embedding of the `PAD` token, since the mislabeled tokens are actually numbers that are part of section titles, and thus are misclassified as numbers.

Table 6: MLP Classifier: Performance Metrics by Class across Data Subsets

Class	Train				Validation				Test			
	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC	Pre	Rec	F1	AUC
ADJ	0.96	0.98	0.97	0.99	0.91	0.93	0.92	0.97	0.88	0.92	0.90	0.96
ADP	0.98	0.99	0.99	0.99	0.95	0.97	0.96	0.99	0.95	0.98	0.97	0.99
ADV	0.98	0.94	0.96	0.99	0.94	0.87	0.91	0.96	0.96	0.87	0.91	0.97
AUX	0.99	0.99	0.99	0.99	0.98	0.97	0.98	0.99	0.98	0.97	0.98	0.99
CCONJ	0.93	0.94	0.93	0.98	0.83	0.87	0.85	0.91	0.79	0.84	0.81	0.89
DET	0.99	0.99	0.99	0.99	0.95	0.96	0.96	0.99	0.95	0.96	0.96	0.99
INTJ	0.89	0.87	0.88	0.93	0.82	0.76	0.79	0.89	0.88	0.81	0.85	0.89
NOUN	0.96	0.98	0.97	0.99	0.93	0.95	0.94	0.97	0.89	0.94	0.92	0.96
NUM	0.99	0.97	0.98	0.99	0.99	0.96	0.97	0.98	0.98	0.96	0.97	0.99
PART	0.99	0.97	0.98	0.99	0.93	0.90	0.92	0.96	0.92	0.83	0.87	0.94
PRON	0.99	1.00	0.99	1.00	0.99	0.99	0.99	0.99	0.98	0.99	0.98	0.99
PROPN	0.89	0.88	0.88	0.95	0.79	0.73	0.76	0.83	0.80	0.70	0.75	0.83
PUNCT	0.99	0.99	0.99	0.99	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.99
SCONJ	0.96	0.91	0.94	0.98	0.86	0.77	0.81	0.89	0.84	0.81	0.83	0.90
SYM	0.96	0.83	0.89	0.92	1.00	0.64	0.78	0.97	0.96	0.68	0.79	0.79
VERB	0.98	0.98	0.98	0.99	0.95	0.95	0.95	0.98	0.95	0.94	0.94	0.98
X	0.97	0.34	0.50	0.67	0.80	0.33	0.47	0.44	1.00	0.39	0.56	0.52
Macro Avg	0.96	0.91	0.93	0.96	0.92	0.86	0.88	0.92	0.92	0.86	0.88	0.92

Contributions

In this assignment, the work was divided between the members of our team. The first part of exercise 9, up to and including the baseline classifiers, was implemented by *Melina Moniaki*, while the rest was designed by *Maria Grammatikopoulou*. Lastly, exercise 10 was implemented by *Phevos A. Margonis*.

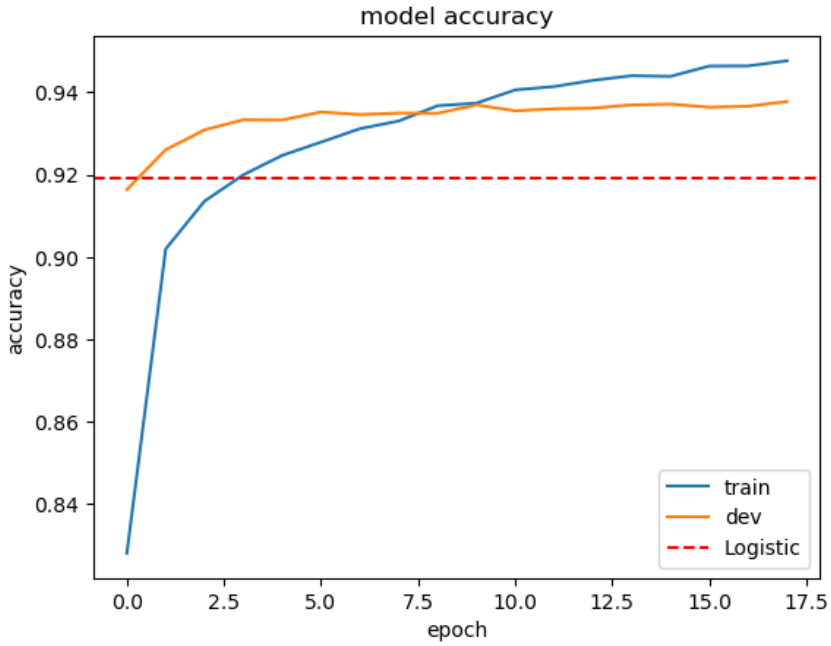

Figure 5: Accuracy Curves for Training and Validation Data as a Function of Training Epoch

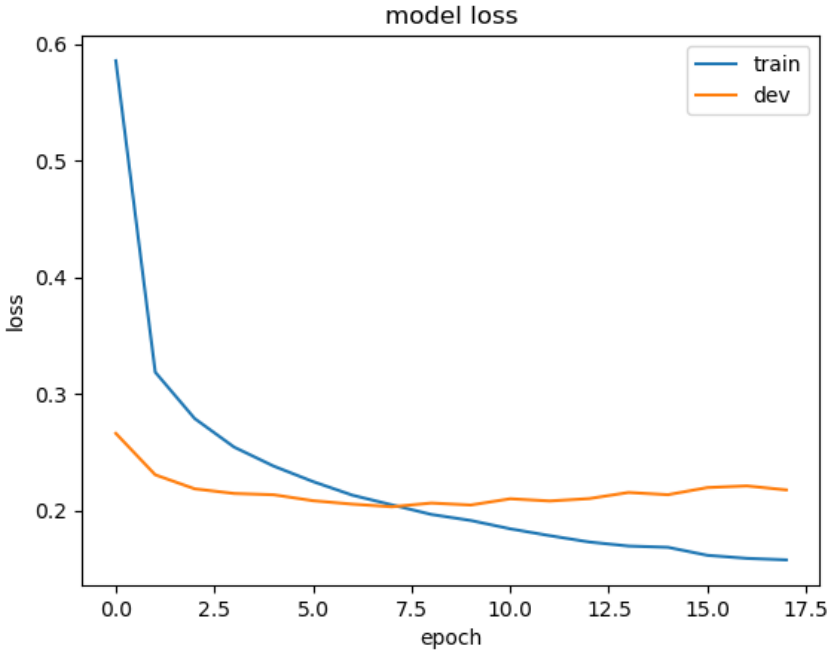

Figure 6: Loss Curves for Training and Validation Data as a Function of Training Epoch