Practical 2025: Classifying Cable News Clips

Due 11:59 PM on Wednesday, April 30th, 2025

This assignment must be completed in groups of 2 to 3 students. You can seek partners via Ed, and course staff will release a team-finding spreadsheet. Section 2 of this document describes the classification problem, and Section 1 includes submission logistics and FAQs. In this same directory, you will find a file called `practical-template.tex`. Use it as a LaTeX template for your writeup, and be sure to read it for grading details.

Harvard College Honor Code: Members of the Harvard College community commit themselves to producing academic work of integrity – that is, work that adheres to the scholarly and intellectual standards of accurate attribution of sources, appropriate collection and use of data, and transparent acknowledgement of the contribution of others to their ideas, discoveries, interpretations, and conclusions. As such, **you may not share your code, or have it viewed by any other student outside your group.**

1 Logistics

1.1 Implementation Details

The train and validation datasets have been included in the repo with you, along with some (fairly sparse) starter code. You will be required to submit all of your implementation code to Gradescope. This assignment can be completed on either your own computer or Google Colab, if you are having issues locally.

1.2 Submission Details

The main deliverable of this practical is a 3-4 page typewritten document in PDF format to Gradescope. The document must follow the `practical-template.tex` file in this directory *and* follow all instructions outlined in Section 3. All relevant text—including your discussions of why you tried various things and why the results came out the way they did—must be included in the maximum of 4 pages. If you need more space than 4 pages for tables, plots, or figures, that is okay.

You should also submit your code as either a .py or .ipynb file on Gradescope. Make sure that the code is neatly organized to reflect which part is being answered. Before submitting, rerun the entire notebook and display all relevant results.

1.3 Grading

Our grading focus will be on your ability to clearly and concisely describe what you did, present the results, and most importantly *discuss* how and why different choices affected performance. Try to have a model that has at least 60% validation accuracy at the end, although you can still earn full credit without reaching this mark (provided that your methodology and writeup are sound).

Parts A, B1, B2 are each graded on a check, check-minus, and minus basis. A check is provided for successfully and thoughtfully completing the section. A check-minus is provided for completing parts of the section and providing little interpretation. Lastly, a minus is provided for providing little to no work. Part C is for optional exploration, for students who wish to explore the problem further.

See `practical-template.tex` for our desired submission format and more tips for what a full-credit submission looks like. All team members will receive the same practical grade.

1.4 Optional: Extra Credit

We will also be awarding 5 percentage points of extra credit on this assignment to the 5 teams with the best models. This will be measured by performance on a test set of cable news clips, meaning you will only have the text snips for these clips (no channel labels provided). If you are interested in this extra credit opportunity, you will use your final model to make predictions on this test set and submit those predictions to our Kaggle competition for this assignment. The results will be announced after the late deadline of May 2nd.

(Note: The Kaggle link and test set have not been set up yet; we will provide these shortly!)

2 Problem Background

For this practical, you will classify cable news clips from various television programs. Specifically, you will predict the channel each news clip originates from based solely on the raw transcript text.

The dataset was collected by scraping broadcast television content archived on the Internet Archive (`https://archive.org/details/tv`). Each clip was identified using a search query that filtered for news segments mentioning the keywords "ml" or "ai". Data were collected systematically across all of 2024.

The raw scraped dataset contains metadata including a unique identifier for each clip, the program title, broadcast channel, air time, and the corresponding runtime. Transcripts were extracted from the closed-caption (CC) text provided by the archive. During preprocessing, duplicates were removed based on their unique identifiers, and additional metadata (such as channel name, air time, and program details) were parsed from the title field.

For modeling purposes, the data are divided into training and validation splits based on time: the earlier months (January through October) form the training set, while clips from the later months (November and Ceember) are reserved for validation. This temporal split helps to simulate a realistic prediction scenario.

In this task, your only feature is the raw transcript text from each news clip. The target is the channel name. The additional metadata features such as the program name and runtime have been removed. You should carefully inspect the training dataset before modeling, particularly paying attention to class distribution of the target, as there may be significant class imbalance across different channels. Addressing or accounting for this imbalance may be necessary for effective classification.

3 Your Task and Deliverables

Below in Parts A-C we list three concrete explorations to complete in the context of this task. Through this process of guided exploration, you will be expected to think critically about how you execute and iterate your approach and describe your solution.

You are welcome to use whatever libraries, online resources, tools, and implementations that help you get the job done. Note, however, that you will be expected to *understand* everything you do, even if you do not implement the low-level code yourself. It is your responsibility to make it clear in your writeup that you did not simply download and run code.

3.1 Evaluation Metrics

In this practical, we would like you to primarily focus on optimizing for accuracy:

$$\text{Classification Accuracy} = \frac{\text{Number Correctly Classified Examples}}{\text{Total Number of Examples}}.$$

In Parts A - C, you will be asked to train several different classification models. **For each model you train, calculate the model's overall and per-class classification accuracies for the train and validation accuracies** and include these results in your write-up.

Even better, try to run several seeds for each model in a given experiment and report mean and standard deviation of the evaluation metric. This helps you see which variability is due to chance vs. which is attributable to your experimental choices.

3.2 Part A: Feature Engineering and Baseline Models

First, we will be looking at a relatively simple linear model. Your task is to train a **logistic regression on two feature representations of your choice**. For instance, you can use

`CountVectorizer` and `TfidfVectorizer` from `sklearn.feature_extraction.text`. Do not perform any hyperparameter tuning—this will be done in part B2.

3.3 Part B: More Modeling

Now, you will experiment with more expressive nonlinear model classes to maximize accuracy on the news clip classification task. Examples of nonlinear models include random forests, KNN, and neural networks.

3.3.1 B1: First Step

First, you will train an untuned nonlinear model. Your task is to train at least one **nonlinear model on a feature representation of your choice (this can be the same as used for the linear model)**. For model classes with hyperparameters, select a hyperparameter value you intuitively think is appropriate. Do not perform any hyperparameter tuning—this will be done in the next part. Compare your results to the logistic regression models in Part A and discuss what your results imply about the task.

3.3.2 B2: More Complicated Models–Hyperparameter Tuning and Validation

In this section, you will explore hyperparameter tuning. Model hyperparameters such as network architecture or random forest maximum tree depth determine the expressivity of the model class. Training hyperparameters such as learning rate, weight decay, or regularization coefficients influence optimization and can encourage desirable properties (such as sparsity) in the final learned models.

Popular hyperparameter tuning techniques include random search, where you train a set of models with hyperparameters chosen uniformly at random from a set of possible values, [1] and grid search, where all possible parameter values are considered exhaustively [2].

Your task is to **perform a hyperparameter search to maximize predictive accuracy for two model classes of your choice**. You can choose which hyperparameters you search over (feel free to search over multiple simultaneously if you'd like!), but you must search over at least 5 possible values for at least 1 hyperparameter. Explore the changes in performance as you choose different hyperparameter values. In your writeup, discuss your validation strategy and your conclusions.

Note: Choose how to present your results of your hyperparameter search in a way that best reflects how to communicate your conclusions.

[1]https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf
[2]https://scikit-learn.org/stable/modules/grid_search.html

4 Final Write-up and Reflections

To wrap up all your exploratory research into applying different types of models and related training strategies to this use case, your final task is to document and summarize the steps you took in the ML research pipeline, and reflect on the choices that you made.

Your task: For each of the components below, include a paragraph (2-4 sentence) explanation and reflection of what you did for each step. Think through any trade-offs, domain-specific input, real-world considerations, and ethical decisions that you made along the way, in addition to any considerations made from a purely machine learning perspective.

Key Components

1. **Data Pipeline**: What representations and data processing techniques did you employ in your exploration? Which one stands out as most relevant?
2. **Model Selection**: What was your choice of model, what trade-offs did you make? Did you observe any trends or advantages/drawbacks of any of the methods that you tried?
3. **Model Tuning**: How did you tune or optimize your model? Discuss your thoughts on your choice of problem framing and optimization technique. Did you include additional strategies (validation, cross-validation, ensembling) to further improve your models?
4. **Bias-Variance Trade-off**: Are any of your models over/under-fit? Did you take steps to find an optimal trade-off? What techniques did you employ to minimize over/under-fitting? What decisions did I ultimately make?
5. **Evaluation**: What metrics did you use to evaluate your models' performance? What drawbacks or advantages did those choices hold? Are there any metrics that you considered?
6. **Domain-specific Evaluation**: Are there any insights about the domain or dataset that you used to make key choices in your exploration? Do a gut check—do the choices of problem framing and optimization make sense for this specific problem? Does your model (the resulting architecture, learned weights, etc.) make intuitive sense for making predictions on the data?
7. **Design Review**: Critique your designs and choices—is there anything that you would have done differently? Are there any foreseeable issues with my choices of dataset, models, optimization strategies, etc.?
8. **Execution & Implementation**: Are there are real-world considerations (deployment, training resources, data availability) that would be relevant to consider? Are

there any ethical considerations to be had about collecting data and deploying these models? Should I, or could I, deploy this model?

4.1 Optional Exploration, Part C: Explore some more!

This section is not required to receive a full-credit grade on the practical. See Section 1.3 for more details about practical grading.

Your task: Try any combination of the suggestions below, or come up with your own ideas to improve model training or expand your evaluation! In your write-up, discuss what you tried, what happened, and your conclusions from this exploration.

Some ideas:

- **Use a transformer**: BERT is a good starting point.
- **Explore other evaluation metrics**: You may also consider optimizing for or reporting other metrics e.g. precision or recall across various classes.
- **Address class imbalance**: Some classes are very infrequent in the training dataset. Popular techniques to address class imbalance are using a class-weighted loss function [3] or by upsampling infrequent classes during training [4].
- **Use a generative classifier:** You could build a model for the class-conditional distribution associated with each type of sound and compute the posterior probability for prediction.
- **Use a support vector machine:** If you prefer your objectives convex.
- **Go totally Bayesian:** Worried that you're not accounting for uncertainty? You could take a fully Bayesian approach to classification and marginalize out your uncertainty in a generative or discriminative model.

4.2 Other FAQs

What language should I code in? As you will be submitting your code, you should code in Python.

Can I use {scikit-learn | pylearn | torch | jax | other ML library}? You can use these tools, but not blindly. You are expected to show a deep understanding of the methods we study in the course, and your writeup will be where you demonstrate this.

[3] https://www.kdnuggets.com/2018/12/handling-imbalanced-datasets-deep-learning.html
[4] https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28

What do I submit? You will submit both your write-up on Gradescope and all of your practical code to a supplemental assignment on Gradescope.

Can I have an extension? Yes, your writeup can be turned in late according to the standard homework late day policy. You can use at most two late days on the practical.