Cal Capstone Project: Detect Comm

By Brad Brown

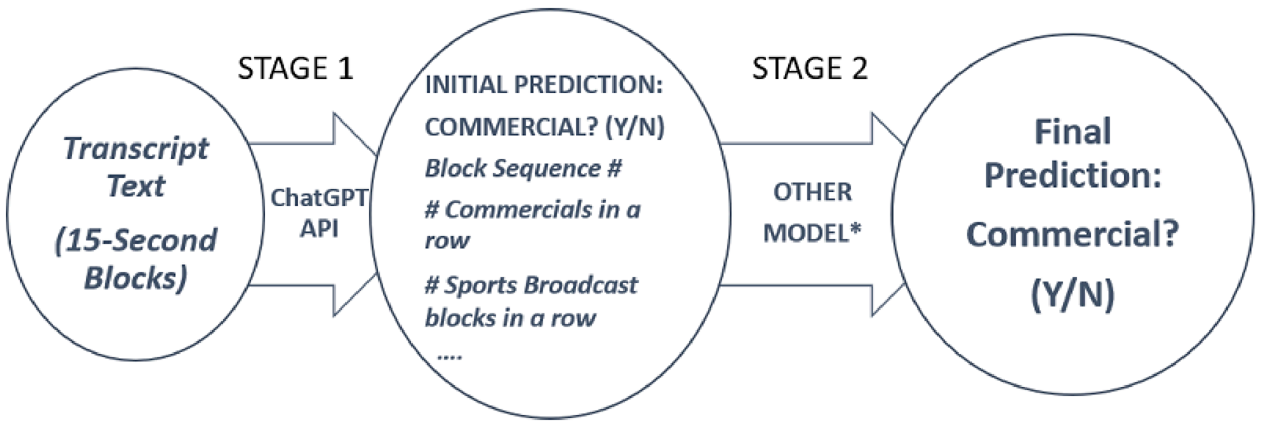
Executive summary

(Note: Instructions to run the project's ipython notebooks are at the end of this README file)

Goal

Using my previous research project and paper ("Helping an LLM improve its detection of TV Commercials in Sports Broadcasts") as a starting point, can I improve on my initial research to achieve higher accuracy and more granular **detection of commercials within a television sports broadcast?**

The previous research project and paper was completed in March of 2024. The project had two stages. Stage 1 fed 15-second 'chunks' of text from 722 minutes of transcripts of the audio of TV sports broadcasts to ChatGPT via a python api, instructing it to categorize each chunk as a commercial or a sports broadcast. In Stage 2, those answers along with other statistics about each chunk were then used to train a logistic regression model to try to improve the accuracy of the initial predictions.



*OTHER MODELS TRIED TO DATE ARE:

- Logistic Regression
- XGBoost

The business goal of the Cal Capstone project is to enable a better and new way to detect commercials and improve the prediction accuracy of the previous effort. The Capstone project will be particularly focused on improving the Stage 2 accuracy:

1. Rework stage 1 and stage 2 model processing and metrics to **predict whether each sentence is a commercial or not**
2. **Use feature engineering** to improve accuracy and quality of the stage 2 model
3. Use the **XGBoost model rather than logistic regression model**

See the section '*How we measured success*' below for detailed specific objectives.

How to get to the goal

1. We will understand and explore our data from stage 1 and its outputs. The outputs of stage 1 are the inputs to stage 2. Look for data quality errors and insights about potential feature engineering and prioritization hints.

2. We will decide a data split strategy to help us for feature engineering and the modeling process.

3. Will engineer features based on typical feature typical creation techniques and subject matter expertise. As an integral part of the modeling process, we will try nonlinear variants of our input data and then use feature importance tools to narrow down to the most promising feature set.

4. With a set of features defined, we will tune the parameters of the XGBoost and the Logistic Regression models, searching for the best hyper-parameters of each to tune each model.

5. We will make a final model choice, tune it and then analyze how it performs on our held back test set.

6. We will then summarize our results, draw conclusions and define next steps

Why it matters

There are potentially millions of people that would benefit from the ability to seamless detect commercials while watching sporting events. Sporting events are very popular but commercials are the opposite - very few people desire to see them. If the transitions to and from commercials can be detected reliably, then the possibility of providing interesting content to consumers during commercials via non-TV devices is feasible. For example, short inspirational or educational snippets of video could be triggered on a consumer's phone during a commercial on TV device. Therefore, every improvement possible to make the detection reliable matters to those consumers. Current techniques use visual image and audio signal processing modeling techniques. This project represents one of the first times large language models and other ML modeling techniques on that LLM's output are used to solve the commercial detection problem. It has promise to be a more real-time practical solution or potentially be part of a more effective and efficient ensemble of modeling approach.

With the initial research project, F1 macro score of individual chunk predictions improved from Stage 1 (ChatGPT predictions) of 82% to 89% in Stage 2 (Logistic Regression predictions using stage 1 outcomes as input). A 7% improvement is good; however, 15-second chunk granularity reduces its potential for practical use. Consumers care about accuracy around these transition points from mainly primary content to mainly commercial content and vice versa. If detection of a chunk is wrong, it wastes 15-seconds of consumer attention. Wrong twice in a row and it wastes 30-seconds - this result could mean the entire solution gets rejected by end users.

Therefore, **this project moves to a more granular sentence-based approach rather than a '15-second chunk' based approach.** While sentence block prediction is a more challenging task, if successful, it makes it a more useful result, especially if by using feature engineering and XGBoost, the system can produce an end result that is still 89% F1 macro score or higher.

Findings Summary

A straight forward 4% (F1 macro) overall improvement is possible in the Cal Capstone project approach.

- Most of the overall 4% improvement is due to feature engineering.

An additional 5% improvement (total 9%) is possible in the Cal Capstone project approach but it is likely difficult to achieve this gain in practice

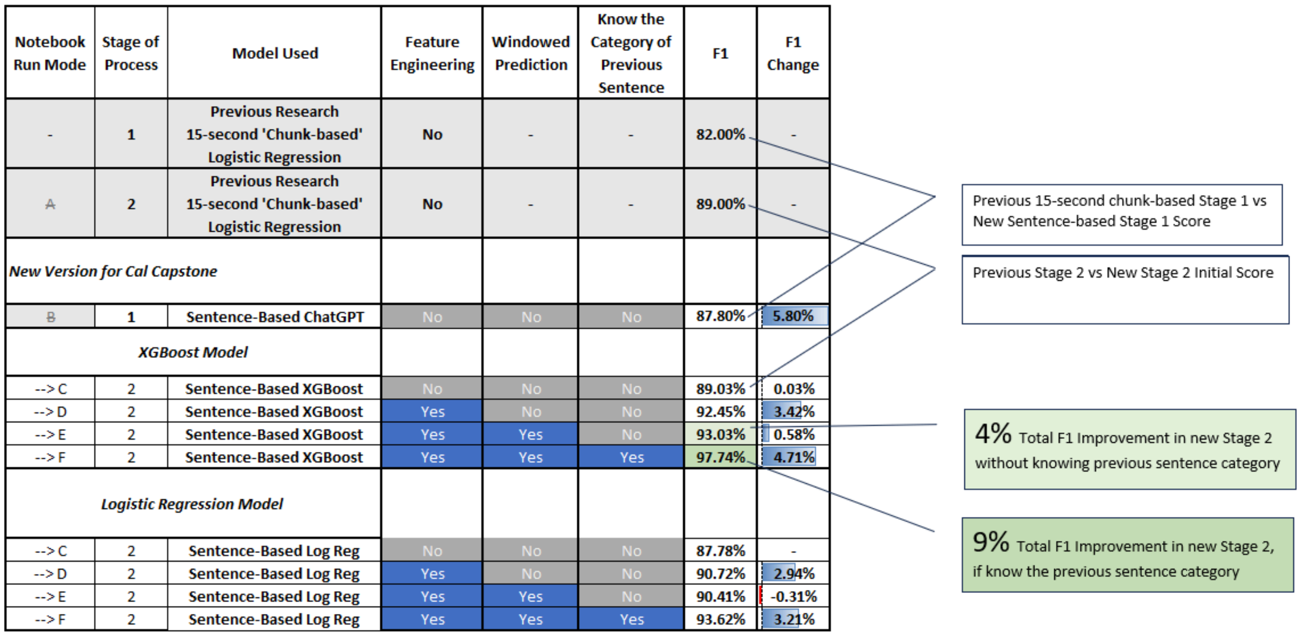
- If the training data exposes the actual ground truth category of the previous sentence and its relevant derived columns (e.g.,'Number of Sentence Since Sports Broadcast') to the current sentence prediction, then the F1 score jumps to 98% from 93%.
- However, in a real-time system the corrected label feedback loop would not be that immediate - could not get the actual previous sentence category in under a second.

XGBoost vs Logistic Regression: XGB better

- XGBoost is 2.62% better

Sentence-based Prediction is much better than 15-second chunk-based prediction

- In Stage 1, the sentence-based approach is 5.80% more accurate. Perhaps more importantly, from a practical user experience perspective, sentence-based prediction is much better. The shorter duration(elapsed time) of a sentence means that an incorrect sentence wastes less end-user time than an error categorizing a 15-second chunk of text.



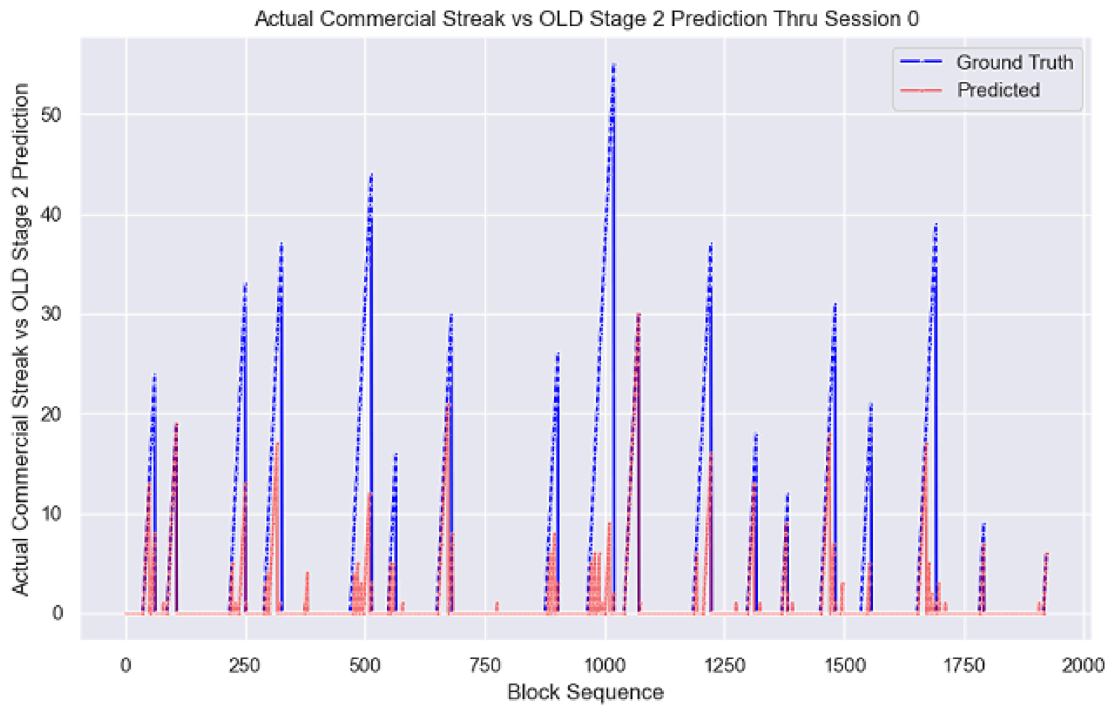
Notebook Run Mode	Stage of Process	Model Used	Feature Engineering	Windowed Prediction	Know the Category of Previous Sentence	F1	F1 Change
-	1	Previous Research 15-second 'Chunk-based' Logistic Regression	No	-	-	82.00%	-
~~A~~	2	Previous Research 15-second 'Chunk-based' Logistic Regression	No	-	-	89.00%	-
New Version for Cal Capstone							
~~B~~	1	Sentence-Based ChatGPT	No	No	No	87.80%	5.80%
		XGBoost Model					
--> C	2	Sentence-Based XGBoost	No	No	No	89.03%	0.03%
--> D	2	Sentence-Based XGBoost	Yes	No	No	92.45%	3.42%
--> E	2	Sentence-Based XGBoost	Yes	Yes	No	93.03%	0.58%
--> F	2	Sentence-Based XGBoost	Yes	Yes	Yes	97.74%	4.71%
		Logistic Regression Model					
--> C	2	Sentence-Based Log Reg	No	No	No	87.78%	-
--> D	2	Sentence-Based Log Reg	Yes	No	No	90.72%	2.94%
--> E	2	Sentence-Based Log Reg	Yes	Yes	No	90.41%	-0.31%
--> F	2	Sentence-Based Log Reg	Yes	Yes	Yes	93.62%	3.21%

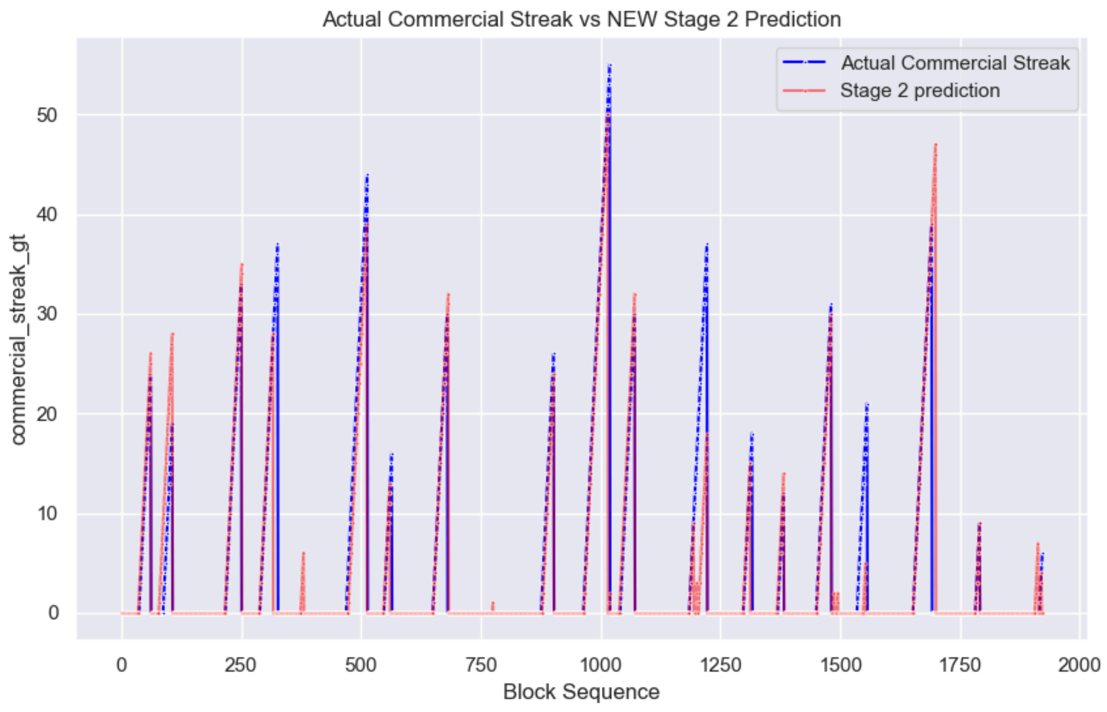
Previous 15-second chunk-based Stage 1 vs New Sentence-based Stage 1 Score

Previous Stage 2 vs New Stage 2 Initial Score

4% Total F1 Improvement in new Stage 2 without knowing previous sentence category

9% Total F1 Improvement in new Stage 2, if know the previous sentence category





Note above how the red lines more closely track the blue lines in the second chart.

Future Work and Development

We learned that XGBoost, sentence-based analysis, and feature engineering will improve accuracy at a more helpful granular level. We also learned that if we could get actual categorization for previous sentences, the accuracy improves significantly. Therefore, the most logical directions for future work would fall into two areas:

1. Quantitative work to try additional models such as a **a long short-term memory model (LSTM) neural network**
2. Technical and business process design work to **get more real-time feedback of the actual categorization of past sentences** (during the game)

Because ChatGPT and LLM capabilities are advancing so quickly, a **3rd area to explore is to use newer models of LLMs** - it's possible that Stage 1 will become so effective that there is no need for Stage 2, in particular the trends around sound and video capabilities of LLMs may greatly improve Stage 1.

Another 4th promising direction is to supplement the system with a retrieval augmented generation (RAG) approach in Stage 1. A vector-database of past sentences or groups of sentences could give the LLM more context to determine that the current sentence is a commercial or not. The total population of commercials is relatively limited and advertisers repeat the same commercials, often several times in the same sports broadcast.

Next Steps and Recommendations

The success with granular sentence-based categorization and the improvement in accuracy from feature engineering and XGBoost indicates that a more robust pilot version of the system should be created using these techniques and tested on larger data sets ASAP (We also may want to reduce the number of features used for interpretability).

As developers design the next version, they should keep an eye towards eventually creating a fast feedback loop so that the system could have the actual correct history of previous sentence categories available. But before developers implement, researchers should quickly test how much improvement can be gained, if there is a lag of 10+ sentences before ground truth categorizations are available to the current sentence prediction. We now know that a one sentence lag is very valuable (5% accuracy boost) but the usefulness of a larger lag may decrease rapidly.

If quantitative resources are available, we recommend to start a RAG based research effort. RAG may have the highest marginal return on investment (ROI) by improving Stage 1 accuracy using existing LLM technology and a vector-database of commercials.

Leadership should keep a close eye on the trends of improvements in the capabilities of LLMs and consider investing in prototypes using advanced LLMs.

Feature engineering gave a major boost in this research project. However, engineered features are derived from base features. In our case, the base feature is an estimate from Stage 1. Trying to extract even more accuracy from derivations on a previous sentence category estimate is likely to have a low marginal ROI. Improving Stage 1 estimates or creating a fast feedback loop to get ground truth labels of past sentences may provide a bigger impact.

How we measured success

The previous project relied on Confusion Matrices as well as F1, Precision, and Recall accuracy metrics to judge commercial detection effectiveness.

> The essence of the F1 Score lies in its ability to capture the trade-off between the two critical components of a model's performance: the precision, which measures how many of the items identified as positive are actually positive, and recall, which measures how many of the actual positives the model identifies. From: DeepGram , Feb 2024

The 'positive' in our case is a 'commercial detected'. This makes the F1 a useful measure for our case because our end-users want us to avoid sports broadcasts being labeled as commercials as well as not misclassify commercials as sports broadcasts. In other words, although precision is slightly more important, the recall also matters to our end-users, therefore, F1 captures these two metrics in one number.

The choice of F1 *Macro average* rather than F1 *Micro average* is because the data population is NOT particularly imbalanced: The data suggests that generally there are roughly twice the number of sports broadcasting sentences compared to commercials. Since end-users value the minority class (commercial detection) somewhat more, a macro-averaged measure is more appropriate (Micro average works well with heavily imbalanced data sets).

The confusion matrix is an excellent mechanism to visualize the distribution of the data into 4 intuitive boxes. We can easily see how many true positive and true negatives there are, but also see determine whether or not our misclassifications tend towards false positives or tend towards false negatives.

For these reasons in the Cal Capstone project, we again used these measures. Note that there are changes being made to the data structures and our analysis of the business needs also inspired changes in our prediction approach: sentence-based and windowed predictions.

Questions to answer as we change the design:

Data structure change: Sentence-based:

- If the project de-emphasized the contiguous 15-second chunks of text and instead relied on single sentences, would the Stage 1 CM and F1 accuracy worsen, stay the same, or improve?

Stage 2 modeling changes: XGBoost and feature engineering:

- Can we train an XGBoost Decision Tree model on the LLM answers and metadata about each sentence to improve the accuracy more than the previous use of a Logistic Regression model?
- Can we use feature engineering such as adding features that take the square of existing features to improve accuracy?
- If the model can rely on past history of correct labels (rather than only the history of Stage 1 estimates), will it help it predict current sentence label better?

Windowed, rolling, or 'smoothed' predictions:

- Adopting 'windowed' predictions in Stage 1, means three of the same prediction in a row are needed to change the current prediction.
- In a sentence-based model, will we experience 'flip flopping' where the incorrect categorizations of one sentence flips the user to the wrong context and then the next sentence flips them back and then the cycle repeats? If so, this cycle might be very annoying to end users.
- Are the errors that cause a flip-flop between correct and incorrect predictions sentence by sentence worse than a consistent set of incorrect predictions that are slower to transition to the correct answer after a shift to or from commercials?

In summary, the metrics we used are F1 Macro, Recall and Precision as well as Confusion Matrices. They were applied in multiple contexts:

BASELINE:

- A) Previous research baseline values

Cal Capstone:

- B) Stage 1 using new sentence-based approach
- **For XGBoost model and Logistic Regression Models:**
    - C) Stage 2 using new sentence-based approach WITHOUT feature engineering and WITHOUT Windowed Predictions
    - D) Stage 2 using new sentence-based approach WITH feature engineering but WITHOUT Windowed Predictions
    - E) Stage 2 using new sentence-based approach WITH feature engineering and WITH Windowed Predictions
    - F) Stage 2 like Stage E except we make the assumption that we know the actual (ground truth) category for the previous sentence

Data Exploration and Insights

Data ultimately comes from 7+ hours of video including commercials. Here is link to one now, if you want to review a few seconds of one.

One of 4 videos in data set: Warriors Vs Nets NBA Basketball Video

As part of my project, I had the 7 hours of video transcribed to raw text files. Here is a sample of the format. The bracket characters [] hold the time stamp relative to the start of the recording. Timestamps are in format [hour:minute:second.microsecond]. Snippet of transcript below.

> **[00:35:30.286]**

> Nets lead by three after one.

> Shout out to formerly the starters, no dunks crew, and to Ian Eagle himself. The bird loves wedgies. A good start to this night here at Barclays. **[00:35:45.332]**

> Can't be a pop star? Mmm, doubt it.

> Chauncey!

> This is rough, but it's also finished.

> I know what you want to do. Yeah, don't touch... Please don't touch that. **[00:36:00.149]**

> Get ready, America.

> Good morning, with Dulcolax.

> Good, good, good morning. Yeah.

> Try Dulcolax Chewy Fruit Bites for fast and gentle constipation relief in as little as 30 minutes. Making **[00:36:15.716]**

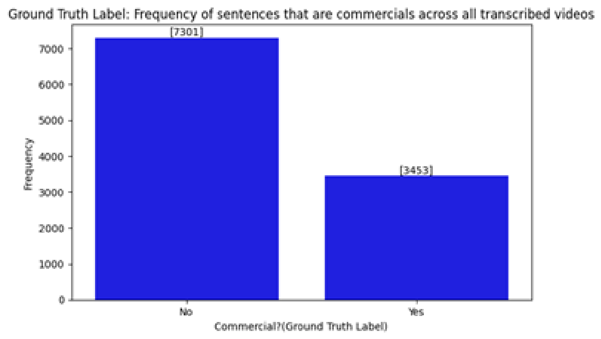
The most central table in the project is the llm_predictions table. It holds the prediction output from chatgpt API about whether the text is a commercial or sports broadcast.

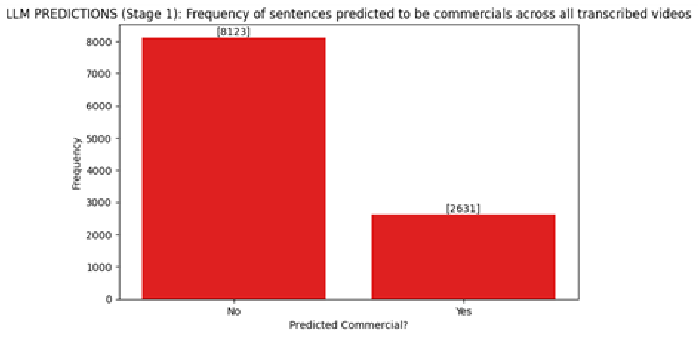
It also holds the statistics about the text block that we use to try to improve the chatgpt prediction using other ML Modeling techniques. The most important columns are:

- response_classification_c: Stage 1 prediction - values 0 for 'Sports Broadcast' and 1 for 'Commercial'
- num_blocks_since_sb - Number of sentences since seen a sports broadcast
- num_blocks_since_c - Number of block since seen a commercial
- sb_blocks_so_far - total number of sport broadcast sentences seen so far in the game
- c_blocks_so_far - total number of commercial sentences seen so far in the game
- sports_broadcasting_streak - number of sports broadcast sentences seen in a row (including the given sentence of this row)
- commercial_streak - number of commercials sentences seen in a row (including the given sentence of this row)

See the notebook [insert link] for a deeper dive into the data tables and data model and how the data was generated.

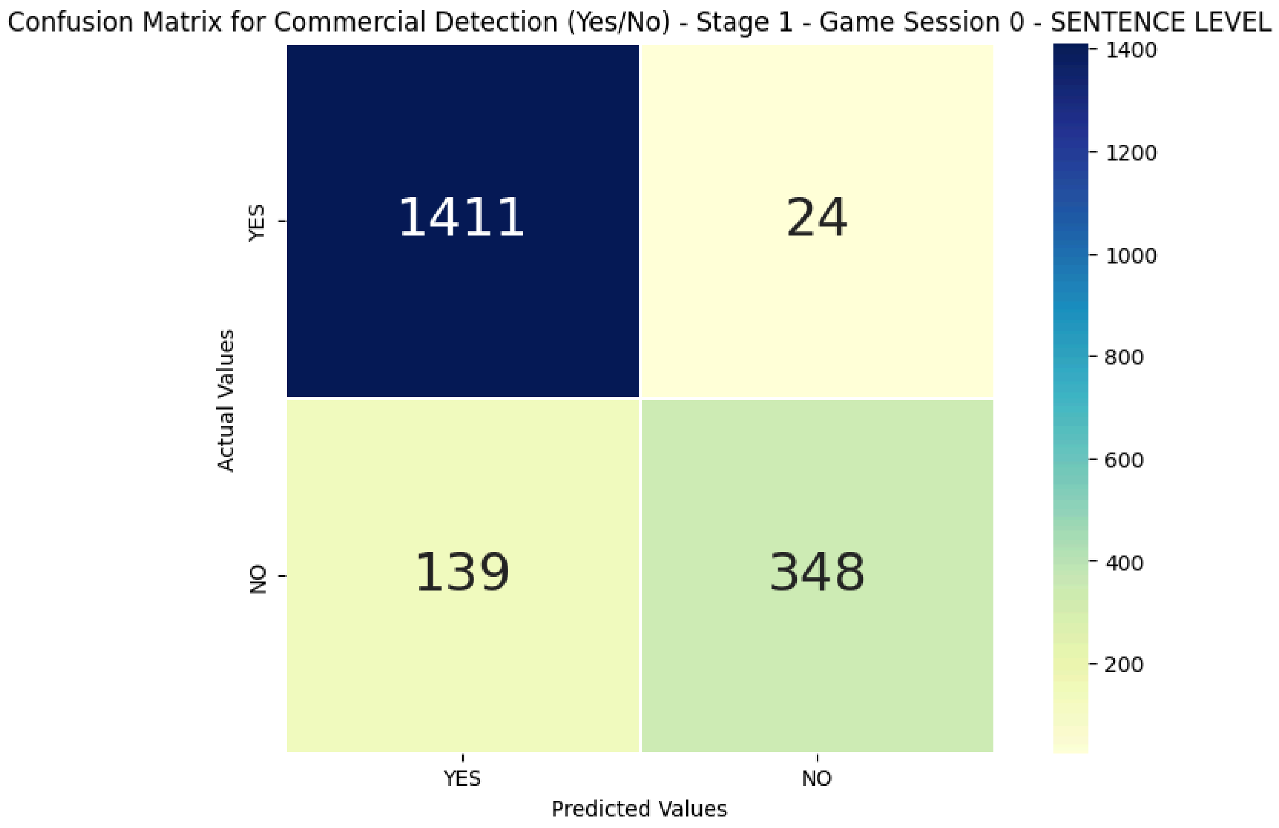
Let's look at the actual number of sentences that are commercials in our full data set and how frequently STAGE 1 predicts commercials.





So we see that the new Stage 1 predicts 802 fewer commercial sentences than actually are in the 722 minutes of video (3,453 actual - 2,631 predicted).

Another and better baseline stat is the confusion matrix from stage 1. The metrics from this baseline test dataset will be used to compare to stage 2 results.



```
recall_score_macro_Stage 1 - Game Session 0 - SENTENCE LEVEL : 0.8489
precision_score_macro_Stage 1 - Game Session 0 - SENTENCE LEVEL : 0.9229
f1_score_macro_Stage 1 - Game Session 0 - SENTENCE LEVEL : 0.8778
bal_acc_score_Stage 1 - Game Session 0 - SENTENCE LEVEL : 0.8489
```

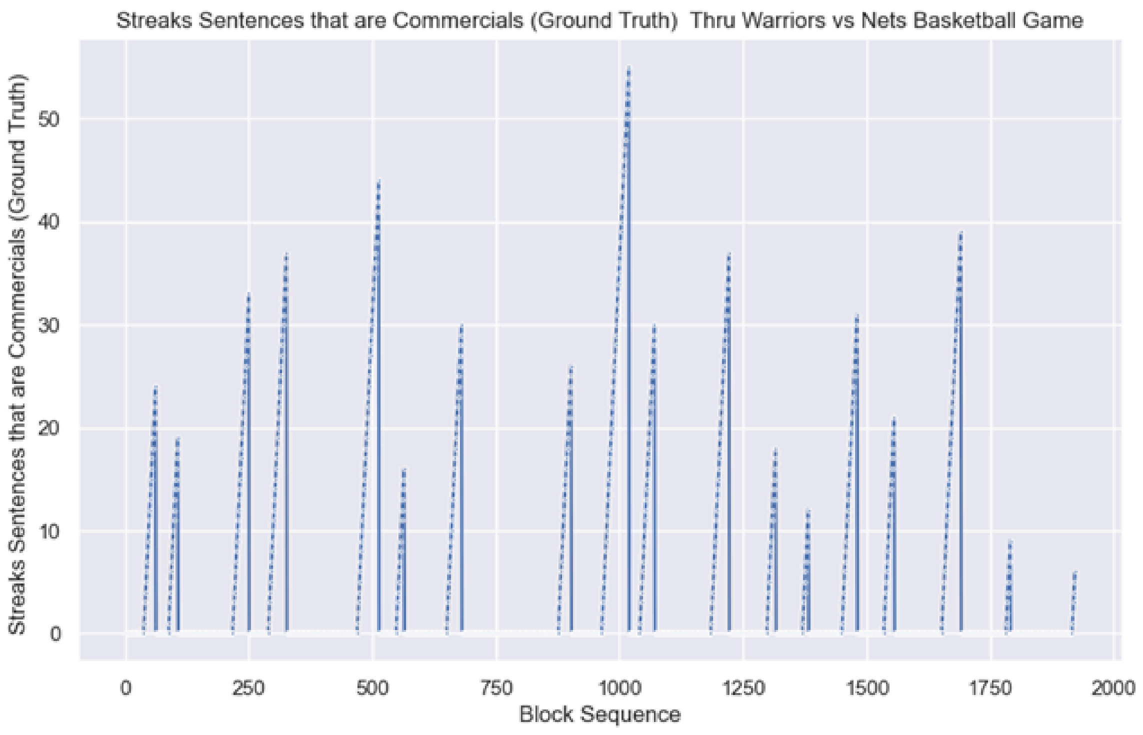
Let's explore the most interesting feature fields we can use for stage 2.

First, let's look at the commercial_streak field:

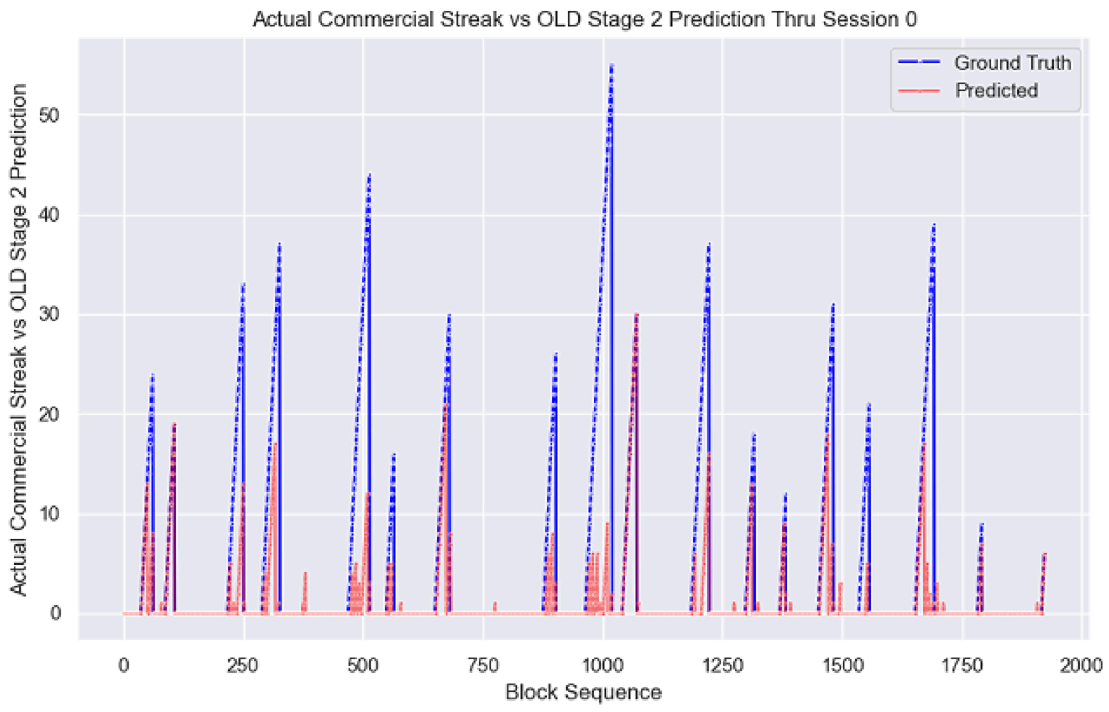
My theory is that commercials run in bunches of contiguous groups and thus a long string of sequential sentences that are all commercials.

I think we can use the pattern that the last X sentences were predicted to be commercials to inform the stage 2 model about whether the current text block is a commercial or not.

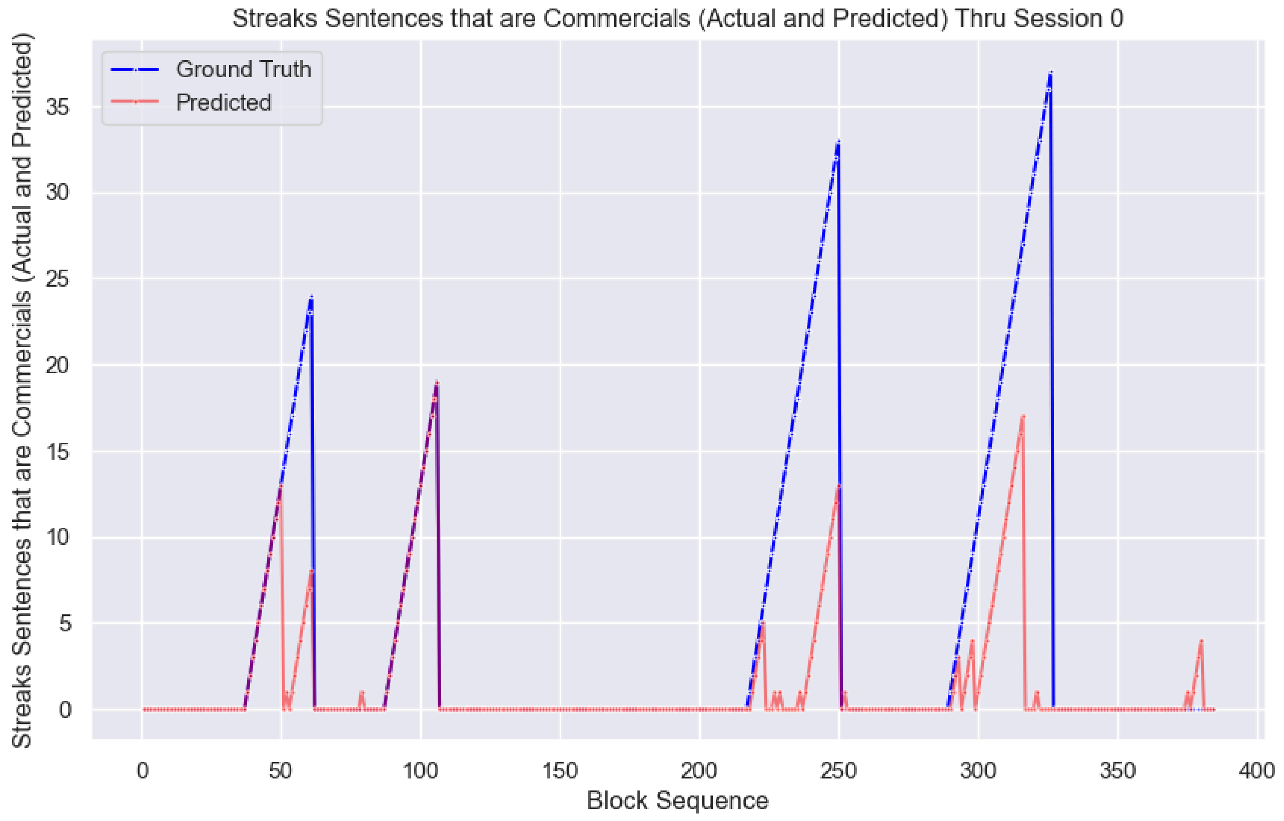
Look at this chart of the actual commercial streaks:



Yet, when we look at what Stage 1 predicted, the streaks are not consistently increasing. They rise and fall back 0 and start again multiple times inside the ground truth streaks triangles.



Zooming in on first 400 sentences...



We will aim to get the red triangles to map closely to the blue triangles

2. Data Split Strategy

In machine learning, data splitting divides your dataset into subsets for training, validation, and testing to prevent the model from simply memorizing the training data and ensures it generalizes well to unseen examples.

Since we are analyzing a time-series data set, we won't use random splits because we need to be very careful to avoid data leakage. In time series data, where order matters, randomly splitting the data for training and testing can be risky. If future values leak into the training set, the model might learn patterns specific to that future data. This can lead to overly optimistic performance that doesn't reflect the model's ability to predict on truly unseen future data.

Therefore, we will split our data by session (or game). We have 4 games in our 10,000+ sentences data set and we will hold back one game (session_id=0) as the test set. The rest will be used for training. Session 0 holds 18% of the sentences.

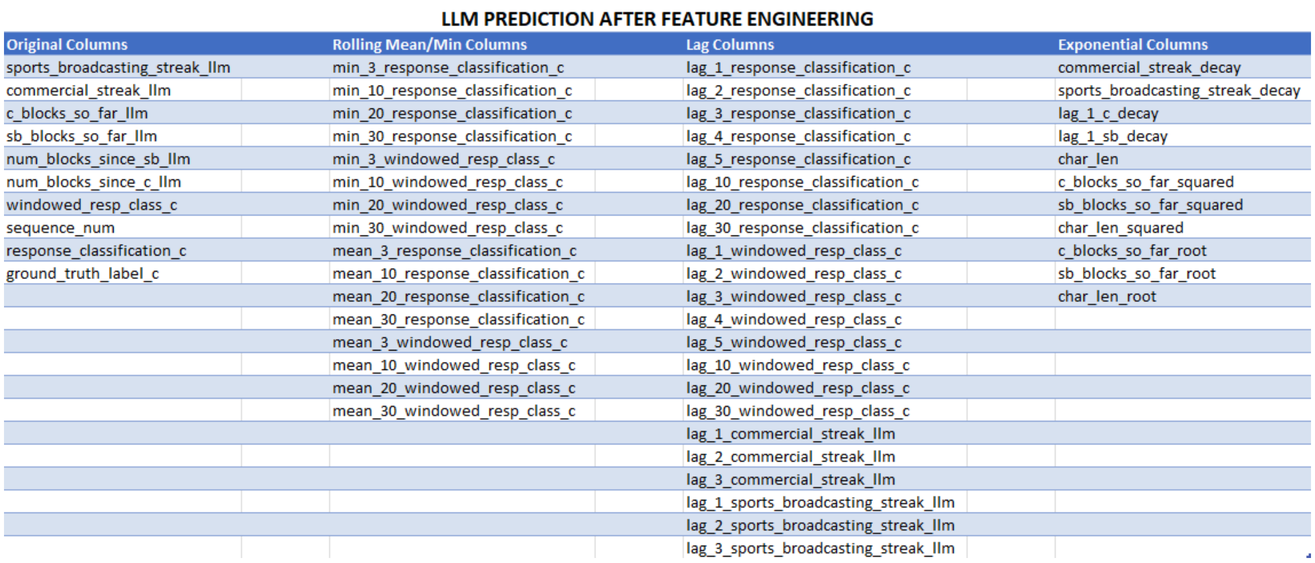
Game ID	Game Name	Total Sentences
0	Warriors vs Nets	1922
2	Warriors vs Celtics	1637
5	49ers vs Packers	3089
7	Superbowl	4106

3. Feature Engineering

For a time-series based analysis it is often useful to create Lag and Rolling features. Will create several of each for the **target value llm_prediction.response_classification_c**

We also create several features based on hunches, visualization observations, and subject matter expertise.

Examples are: exponential decay feature, length of sentence, square and square root of blocks seen so far, etc



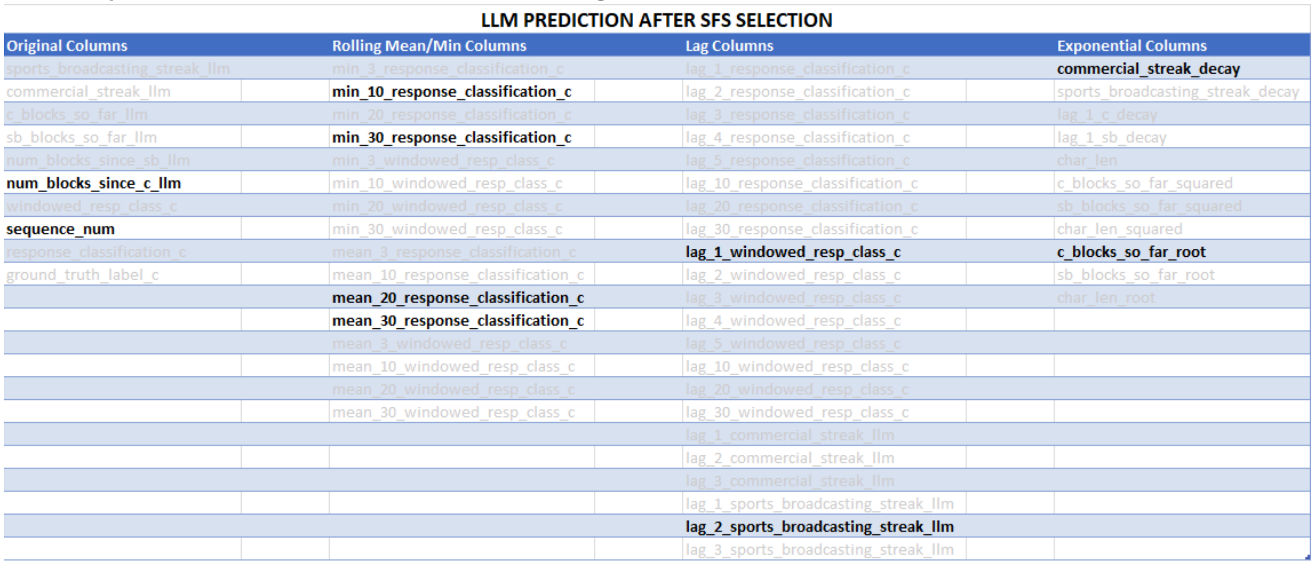
LLM PREDICTION AFTER FEATURE ENGINEERING

Original Columns	Rolling Mean/Min Columns	Lag Columns	Exponential Columns
sports_broadcasting_streak_llm	min_3_response_classification_c	lag_1_response_classification_c	commercial_streak_decay
commercial_streak_llm	min_10_response_classification_c	lag_2_response_classification_c	sports_broadcasting_streak_decay
c_blocks_so_far_llm	min_20_response_classification_c	lag_3_response_classification_c	lag_1_c_decay
sb_blocks_so_far_llm	min_30_response_classification_c	lag_4_response_classification_c	lag_1_sb_decay
num_blocks_since_sb_llm	min_3_windowed_resp_class_c	lag_5_response_classification_c	char_len
num_blocks_since_c_llm	min_10_windowed_resp_class_c	lag_10_response_classification_c	c_blocks_so_far_squared
windowed_resp_class_c	min_20_windowed_resp_class_c	lag_20_response_classification_c	sb_blocks_so_far_squared
sequence_num	min_30_windowed_resp_class_c	lag_30_response_classification_c	char_len_squared
response_classification_c	mean_3_response_classification_c	lag_1_windowed_resp_class_c	c_blocks_so_far_root
ground_truth_label_c	mean_10_response_classification_c	lag_2_windowed_resp_class_c	sb_blocks_so_far_root
	mean_20_response_classification_c	lag_3_windowed_resp_class_c	char_len_root
	mean_30_response_classification_c	lag_4_windowed_resp_class_c	
	mean_3_windowed_resp_class_c	lag_5_windowed_resp_class_c	
	mean_10_windowed_resp_class_c	lag_10_windowed_resp_class_c	
	mean_20_windowed_resp_class_c	lag_20_windowed_resp_class_c	
	mean_30_windowed_resp_class_c	lag_30_windowed_resp_class_c	
		lag_1_commercial_streak_llm	
		lag_2_commercial_streak_llm	
		lag_3_commercial_streak_llm	
		lag_1_sports_broadcasting_streak_llm	
		lag_2_sports_broadcasting_streak_llm	
		lag_3_sports_broadcasting_streak_llm	

We then do Feature Selection using Sequential Feature Selection (SFS) from sklearn.

SFS "backward" selection removes features in what is called a 'greedy' algorithm. A greedy algorithm tackles problems by making the locally optimal choice at each step, hoping it leads to a globally optimal solution. In this case, at each step, the process selects the optimal feature to remove based on the cross-validation F1 score.

After experimentation, I decided on narrowing down to the 10 best features.



LLM PREDICTION AFTER SFS SELECTION

Original Columns	Rolling Mean/Min Columns	Lag Columns	Exponential Columns
sports_broadcasting_streak_llm	min_3_response_classification_c	lag_1_response_classification_c	**commercial_streak_decay**
commercial_streak_llm	**min_10_response_classification_c**	lag_2_response_classification_c	sports_broadcasting_streak_decay
c_blocks_so_far_llm	min_20_response_classification_c	lag_3_response_classification_c	lag_1_c_decay
sb_blocks_so_far_llm	**min_30_response_classification_c**	lag_4_response_classification_c	lag_1_sb_decay
num_blocks_since_sb_llm	min_3_windowed_resp_class_c	lag_5_response_classification_c	char_len
num_blocks_since_c_llm	min_10_windowed_resp_class_c	lag_10_response_classification_c	c_blocks_so_far_squared
windowed_resp_class_c	min_20_windowed_resp_class_c	lag_20_response_classification_c	sb_blocks_so_far_squared
sequence_num	min_30_windowed_resp_class_c	lag_30_response_classification_c	char_len_squared
response_classification_c	mean_3_response_classification_c	**lag_1_windowed_resp_class_c**	**c_blocks_so_far_root**
ground_truth_label_c	mean_10_response_classification_c	lag_2_windowed_resp_class_c	sb_blocks_so_far_root
	mean_20_response_classification_c	lag_3_windowed_resp_class_c	char_len_root
	mean_30_response_classification_c	lag_4_windowed_resp_class_c	
	mean_3_windowed_resp_class_c	lag_5_windowed_resp_class_c	
	mean_10_windowed_resp_class_c	lag_10_windowed_resp_class_c	
	mean_20_windowed_resp_class_c	lag_20_windowed_resp_class_c	
	mean_30_windowed_resp_class_c	lag_30_windowed_resp_class_c	
		lag_1_commercial_streak_llm	
		lag_2_commercial_streak_llm	
		lag_3_commercial_streak_llm	
		lag_1_sports_broadcasting_streak_llm	
		lag_2_sports_broadcasting_streak_llm	
		lag_3_sports_broadcasting_streak_llm	

We checked for NaNs in our data AFTER feature engineering

Note that NaNs come from lag columns that we engineered because they can't get values prior to the beginning of the data sets.

NaN check AFTER FEATURE ENGINEERING

- Train set Nans: 87
- Train set Nans: 29
- % of training rows with at least one NaN: 0.01
- % of test rows with at least one NaN: 0.02

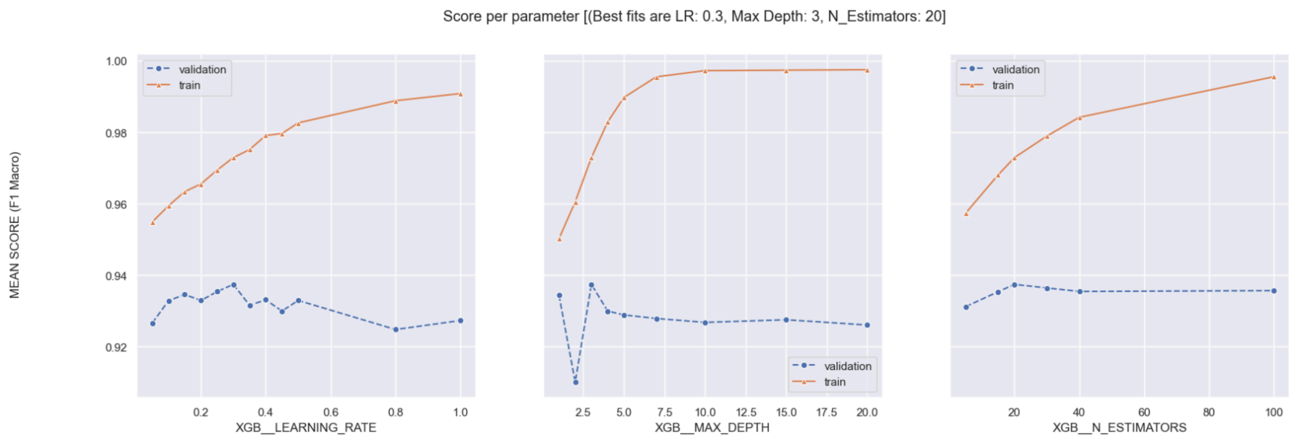
I Left the NaNs in the data sets because below NAN TOLERANCE level we set at 0.03 and XGBoost can handle them anyway

4. Tune The Model

Model Choice

With most optimal feature set defined, now tune the model using Grid Search CV

Note: I used a time series split CV function in cross validation to avoid data leakage



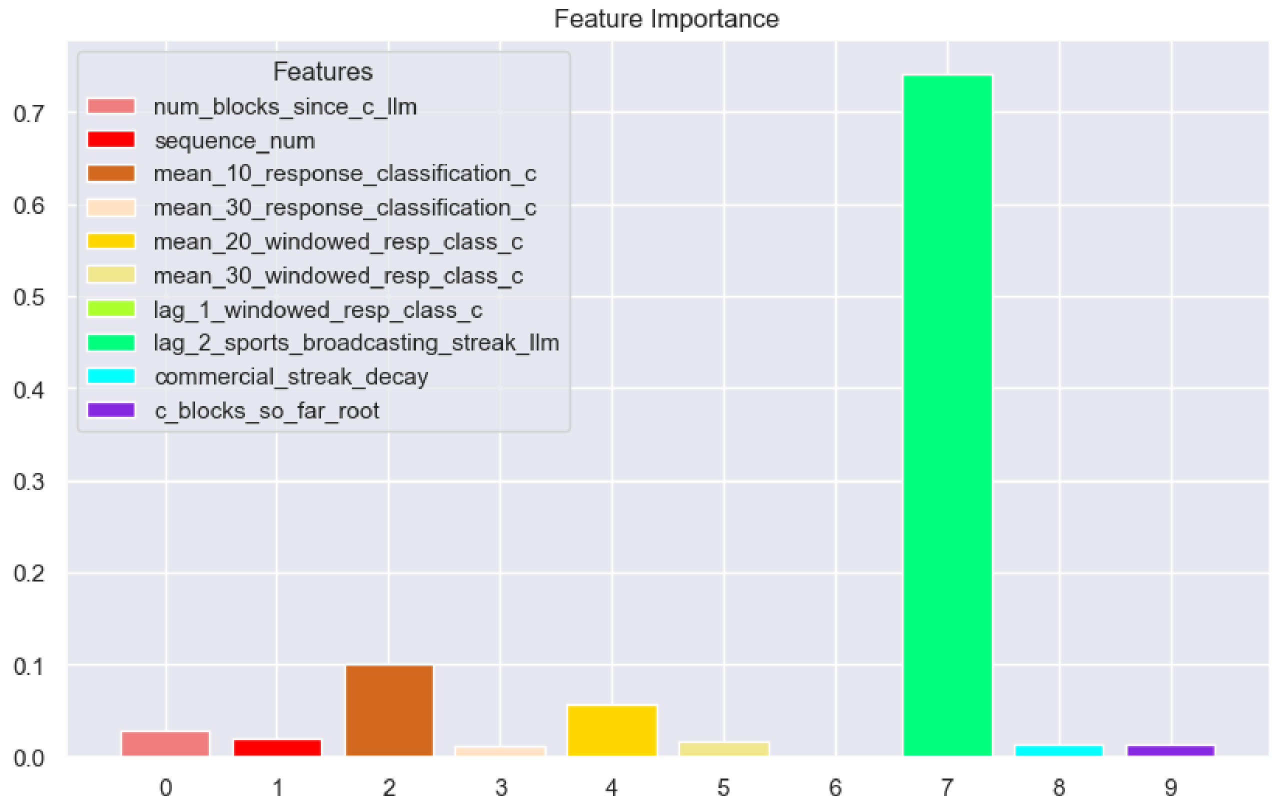
Grid Search helped us find parameters that optimize for validation sets.

Charts show F1 for a range of values of a given parameter (while keeping other parameters to their best fit value).

From charts, we see that we are NOT overfitting because Grid Search CV chose highest point of validation curve near the point of divergence from training curve

With these parameters, we can see the ranking of the 10 features by importance to the model prediction:

- *Lag_2_sports_broadcasting_streak_llm was the most important by far:*

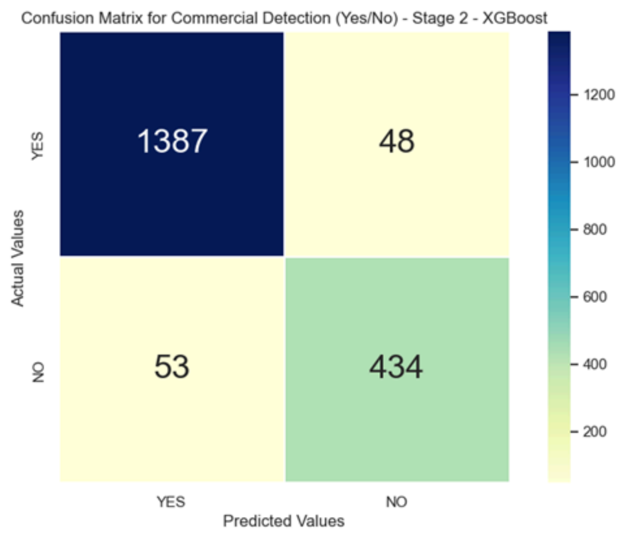


- Lag_2_sports_broadcasting_streak_llm means the model is looking at two rows(sentences) prior. For that row, it is seeing how many sports broadcast sentences had been seen in a row prior to it. This 'streak' is helping the model figure out whether the current sentence is a commercial or sports broadcast. The recent history of sentences seems to relate to the category of the current sentence.

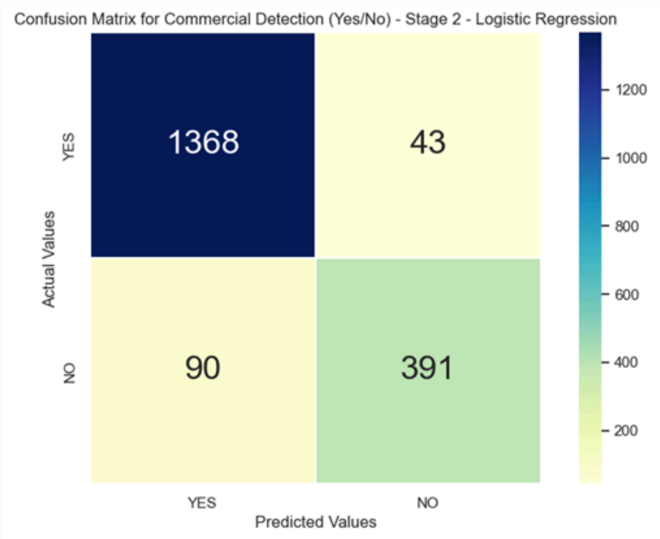
Additional Results and Analysis

The main findings are in the "Findings Summary" section of this document. Also please review "Next steps" section above.

Here are some more details about the results:



```
recall_score_macro_Stage 2 - XGBoost : 0.9289
precision_score_macro_Stage 2 - XGBoost : 0.9318
f1_score_macro_Stage 2 - XGBoost : 0.9303
bal_acc_score_Stage 2 - XGBoost : 0.9289
```



```
recall_score_macro_Stage 2 - Logistic Regression : 0.8912
precision_score_macro_Stage 2 - Logistic Regression : 0.9196
f1_score_macro_Stage 2 - Logistic Regression : 0.9041
bal_acc_score_Stage 2 - Logistic Regression : 0.8912
```

In github repository, see the notebook Modelue24.CapstoneEvaluation-BradBrown.ipynb for more detail

More information

This readme file, related notebook code and images are hosted on github here

If you have python 3.xx installed, you can clone the repo to your local drive to try it out. You can run in multiple models. See the "Notebook Configuration" section.

(Note: In addition to the Cal Capstone repo, I can provide Github source code access to the previously completed Stanford Research Project upon request, but to review the code, you can unzip a copy of the Stanford project source code in this cal capstone github - see the 'prev_research' folder)

Note to grading team:

This cal-cap Github is self-contained in terms of code, but I have included zip files of the original code for the Stanford project and my improvements to make Stage 1 sentence-based (See prev_research folder). The decryption password i will send via the support message

I believe these code zips are above and beyond the required scope of the Cal Capstone grading rubrick. I do not plan to make these githubs public.

But if you feel the cal-cap github is incomplete because its data relies on code from these other githubs, I can open it up to specific graders. Thank you!

Instructions

```
  - Clone the GitHub repository
  - Read this README.MD for non-technical overview
  - Please open the notebook Module24.CapstoneEvaluation-BradBrown.ipynb
```

```
- See 'Notebook Configuration and Settings'
------ You can set the RUN_MODE flag to values as described there
-------This will run the notebook with different assumptions about features available
-------You can set the AUTO_FEAT_SELECT to True if you want the SFS package to
------ to analyze the feature set and choose best available 10 features (Can run for 15 minutes
------ By default, it is set to False. When False, it uses a hard coded list
------ that was compiled by past runs of SFS in different Run_Modes

### Git cal-cap Directory Structure

cal-cap
├── db_export
|     ├── chunks.csv
|     ├── instruction_prompts.csv
|     ├── llm_predictions.csv
|     ├── sessions.csv
|     ├── sources.csv
|     ├── text_block_stat.csv
|     ├── text_blocks.csv
|     ├── transitions.csv
|     ├── updated_llm_predictions.csv
├── images
|     ├── (several png based on notebook runs)
├── output
|     ├── (several png based on notebook runs - recreated after each run of notebooks)
|     ├── (several saved 'print as pdf' of notebook runs in different Run_Modes)
├── Module20.1.InitialReportandEDA-BradBrown.ipynb
    (Only needed for grading Module 20)
├── Module24.CapstoneEvaluation-BradBrown.ipynb
|   README.MD
├── prev_research
|   ├── Two encrypted zip files of source of related project (password provided via support tic
        (optional)
├── scope
|     ├── Brad Brown Cal Capstone Problem Statement.pdf
|     ├── Brad Brown Cal Capstone Scope.pdf
|     ├── CS229_Stanford_Poster_-_Brad_Brown.pdf
        (optional reading)
```

Go to the main notebook: Module24.CapstoneEvaluation-BradBrown.ipynb

Contact and Further Information

Brad Brown Email: bradbrown@416.simplelogin.com