BANK OF ENGLAND PROJECT REPORT

The Alchemists

9th March 2025

Background

As part of its function to regulate and supervise financial services firms through the Prudential Regulation Authority (PRA), the Bank of England (BoE) closely and continuously monitors the decisions and performance of over 1,500 financial services companies[1]. An integral part of this process is the analysis of text-based financial information such as those found around quarterly result announcements – a traditionally work-intensive affair. Using natural language process and large language models, this project aims to streamline this process, allowing the Bank of England to more efficiently and accurately assess risk levels of global systemically important banks (G-SIBs).

Earnings calls are typically one of the most analysed text-based financial resource[2] and since the Q&A section specifically has less (although not zero) oversight from the banks themselves, these parts of the transcripts have been selected for analysis. This project endeavours to create a product that BoE analysts can use to extract key trends, talking points and create financial predictions based on the data contained in these Q&A sessions without having to comb through transcripts manually.

Data Selection and Processing

A selection of the largest banks in European and US market were chosen[3] to maintain BoE-specificity. A javascript scraper was used to extract text and metadata from financial services research platform Seeking Alpha, which upon compiling resulted in a workable table of raw data. An initial round data cleaning included ensuring consistency with person names, and the correct labelling of roles.

This table was then joined with information containing which company each analyst present on the Q&A calls belonged to, to allow for an extra avenue for analysis.

---

[1] https://www.gov.uk/government/publications/recommendations-for-the-prudential-regulation-committee-november-2024#:~:text=The%20Prudential%20Regulation%20Committee%20(PRC)%20is%20responsible%20for%20the%20exercise,insurers%20and%20major%20investment%20firms.

[2] https://corporatefinanceinstitute.com/resources/valuation/earnings-call/#:~:text=3.,their%20answers%20for%20certain%20questions.

[3] https://www.fsb.org/2024/11/2024-list-of-global-systemically-important-banks-g-sibs/

	person	role	position	text	ticker	call_name	call_date
0	Operator	operator	Unknown	[Operator Instructions]. Our first question to...	BCS	Q1 2023	27-Apr-2023
1	Omar Keenan	questioner	Credit Suisse	Good morning everybody. Congratulations on a g...	BCS	Q1 2023	27-Apr-2023
2	Omar Keenan	questioner	Credit Suisse	And my next question is just on capital genera...	BCS	Q1 2023	27-Apr-2023
3	Venkatakrishnan	answerer	Group Chief Executive	Thanks, Omar. It's Venkat. So let me answer th...	BCS	Q1 2023	27-Apr-2023
4	Anna Cross	answerer	Group Finance Director	Yes, sure. Omar, you're right. We said that in...	BCS	Q1 2023	27-Apr-2023

Figure 1: Raw table of Earnings Calls Q&A data from Seeking Alpha

To add a method of evaluating performance of NLP tools later in the project, stock price data was inputted using an API from Alpha Vantage. Three performance metrics were defined: `call_day_price_change`, `earnings_period_price_change`, and `quarter_price_change`, referring to the change in share price on the day of the call, in a ten-day window, and in a quarter-length window.

```
Data columns (total 16 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   person                        13229 non-null  object
 1   role                          13229 non-null  object
 2   position                      13229 non-null  object
 3   text                          13229 non-null  object
 4   ticker                        13229 non-null  object
 5   call_name                     13229 non-null  object
 6   call_date                     13229 non-null  datetime64[ns]
 7   company                       13157 non-null  object
 8   pre_earnings_price            13229 non-null  float64
 9   post_earnings_price           13229 non-null  float64
 10  earnings_period_price_change  13229 non-null  float64
 11  post_quarter_price            13229 non-null  float64
 12  quarter_price_change          13229 non-null  float64
 13  call_day_open                 13229 non-null  float64
 14  call_day_close                13229 non-null  float64
 15  call_day_price_change         13229 non-null  float64
```

Figure 2: Table structure after inputting company information and stock price data

To preprocess data in the `text` columns, stopwords were removed from the text columns using the Natural Language ToolKit (NLTK), as well as manually, along with lemmatisation, and other standard preprocessing techniques in data science.

Initial Data Exploration & BERTopic

To further understanding of the data, wordclouds and bar charts for the most popular words were created. Instantly these provide insight into key inter-bank differences in topics discussed, for example Credit Suisse calls mentioned the APAC market more prevalently than others, as seen in Figure 3.

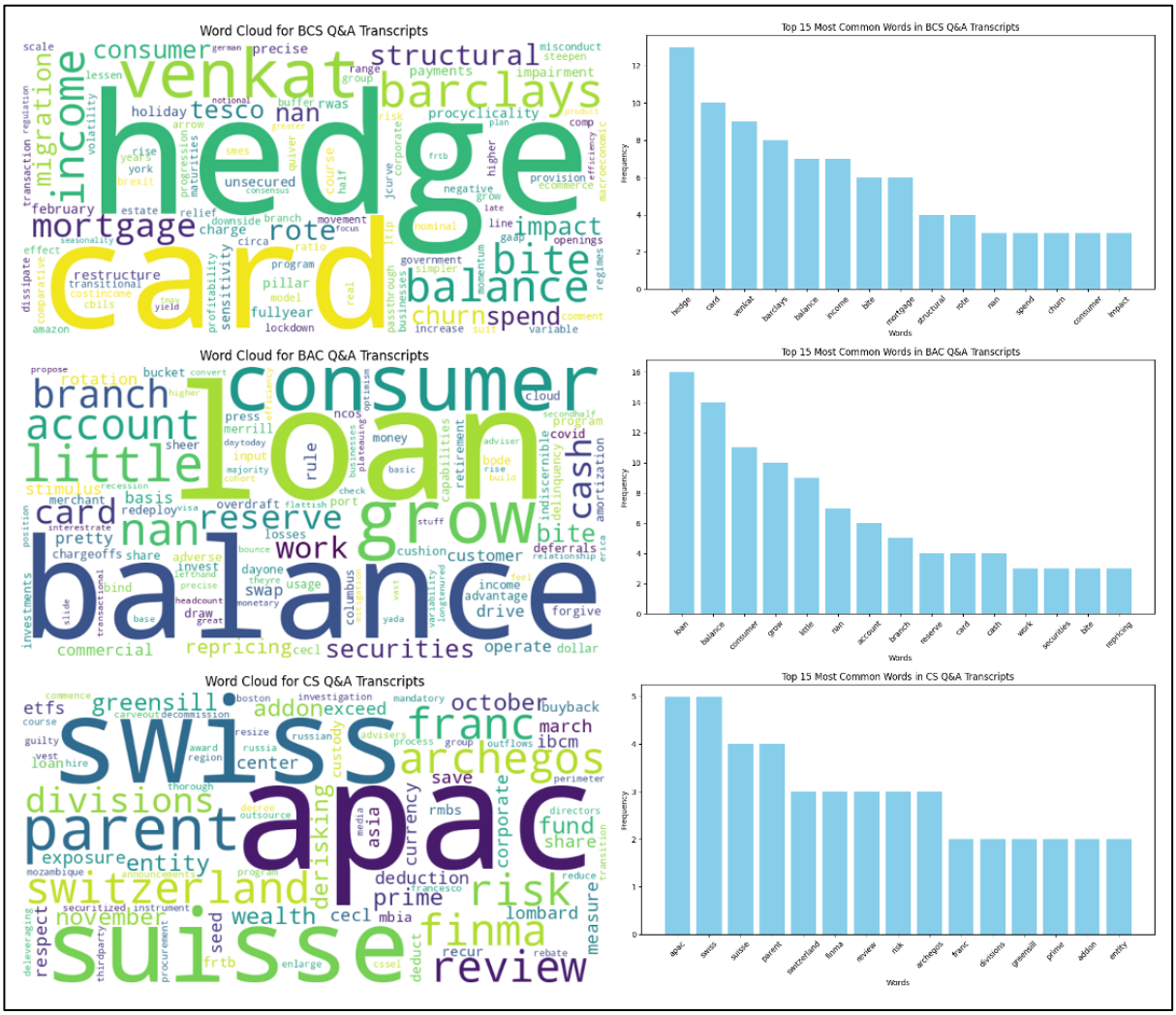


Figure 3: Wordclouds and most common words for Barclays, Bank of America, and Credit Suisse

BERTopic in conjunction with HSBSCAN was applied on the full corpus for topic discovery. The results which showed distinct topics pertaining to Hong Kong markets and hedge strategies were of note and confirmed that topic analysis should be part of the final product.

FinBERT for Sentiment Analysis

FinBERT was used to assign each comment as having a positive, neutral or negative sentiment, alongside a sentiment score, these were then averaged out for each call to provide an average sentiment score.

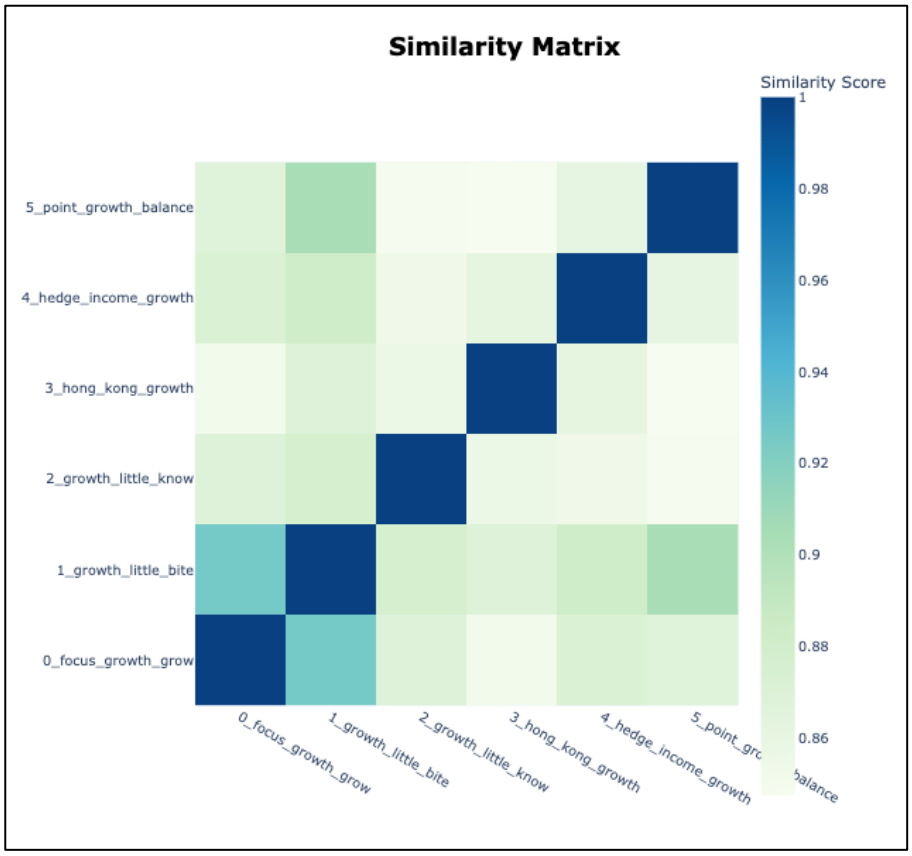


Figure 4: BERTopic detected six distinct topics covered in the corpus

FinBERT showed that the calls heavily skewed towards the neutral classification. Whilst inherently not a bad thing, having an overrepresented neutral bin is of limited use to analysts at the Bank of England and so to counter this, the FinBERT model was tuned using the `sbhatti/financial-sentiment-analysis` dataset containing financial sentences from two sub-datasets (FiQA, Financial PhraseBank) with sentiment labels. As well as improving the model, tuning had the desired effect of increasing the percentage of texts falling into positive or negative classification.

Plotting the percentage of negative and positive comments against stock price changes resulted in Credit Suisse being the clear outlier (Figure 6). Since Credit Suisse is indeed an outlier in the banks included for analysis in the sense it went through financial distress leading to failure, this suggests that there is merit in using NLP for analysis on textual documents. Most banks do not have a clear correlation between sentiment and change in stock price, implying that the financial health of banks is generally resilient to sentiment shown on earnings calls Q&As segments. However, in Credit Suisse's lone case, the apparent link between share price change

and call sentiment suggests that these two values may be linked in the case of a bank performing poorly. This hypothesis should be explored further with more case studies of failing banks.

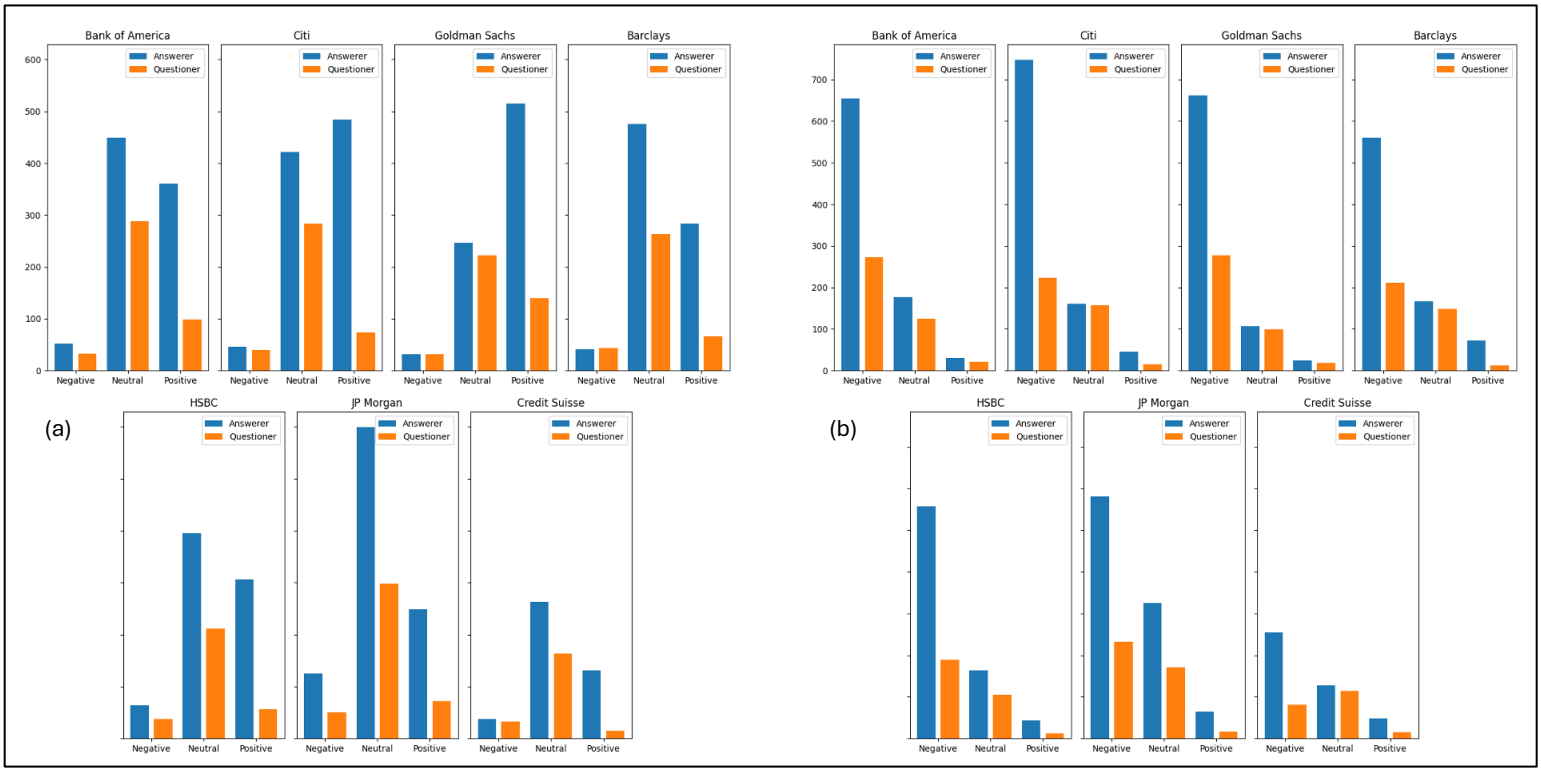


Figure 5:Untuned (a) and tuned (b) FinBERT sentiment classification for all comments in Q&A Earnings Calls

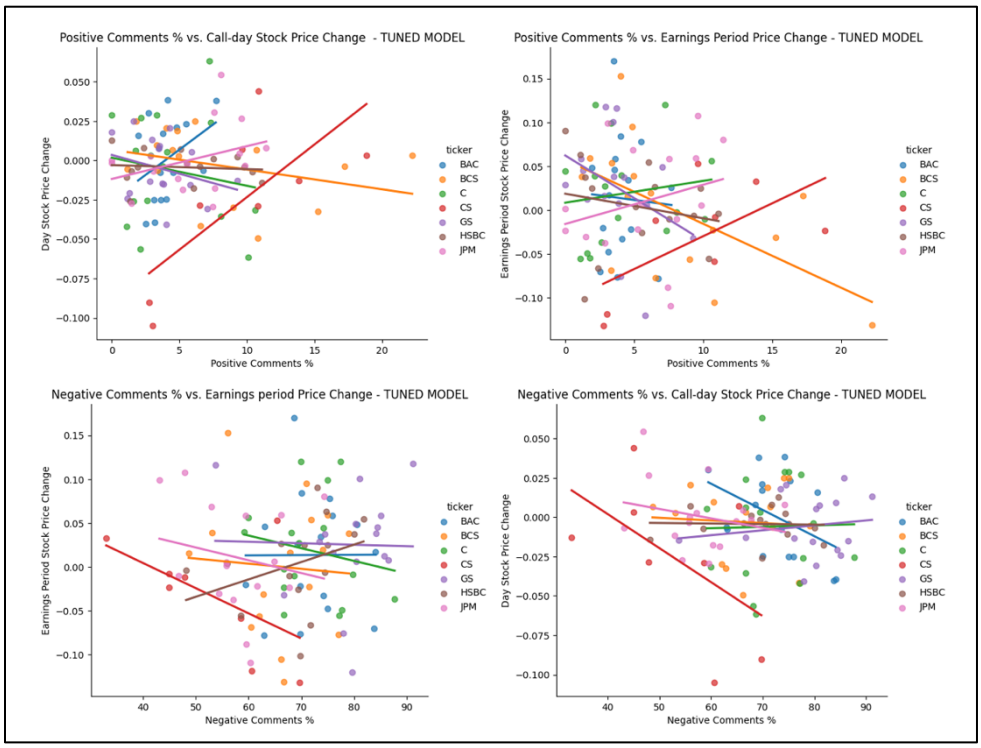


Figure 6: Stock price change against percentage of positive and negative comments (as evaluated by the tuned FinBERT model) present in the Q&A transcript depicting, the now failed, Credit Suisse as the clear outlier

The positivity, P, of a string of comment was introduced as a new metric, defined:

$$P = \begin{cases} S_p - S_n + 1, \text{if the sentiment is "positive"} \\ S_n - S_p + 1, \text{if the sentiment is "negative"} \\ 0.5, \text{if the sentiment is "neutral"} \end{cases}$$

Where S_p is the positive confidence score and S_n is the negative sentiment score.

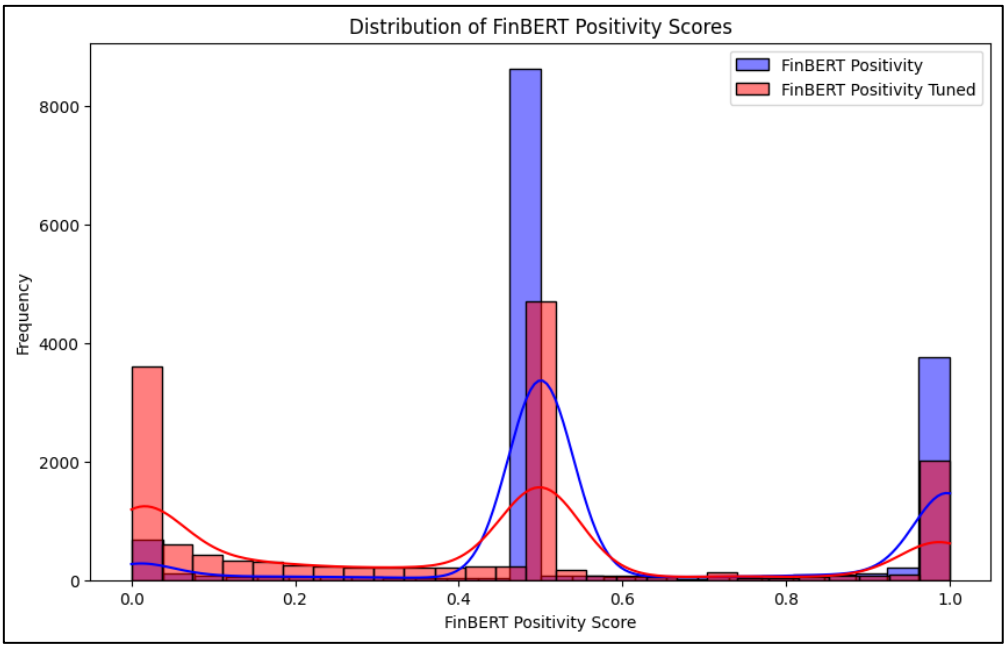


Figure 7: Distribution of positivity scores

This allowed for the creation of graphs present in Figure 8 which plot stock price change on the same time axis as average positivity, and Figure 9 which shows that the reactivity of stock price for failing banks (such as Credit Suisse and Silicon Valley Bank), with respect to sentiment is in contrast to the lower volatility over longer periods as the market comes to the realisation that the bank is failing.

More here around using a positivity-based metric if we end up putting them in the interface otherwise leave (you select a call on app.py, and it tells you the avg positivity score for comments on the call, tells you the avg positivity score for calls from that bank, tells you the avg positivity score for banks that quarter, and % above, below each)

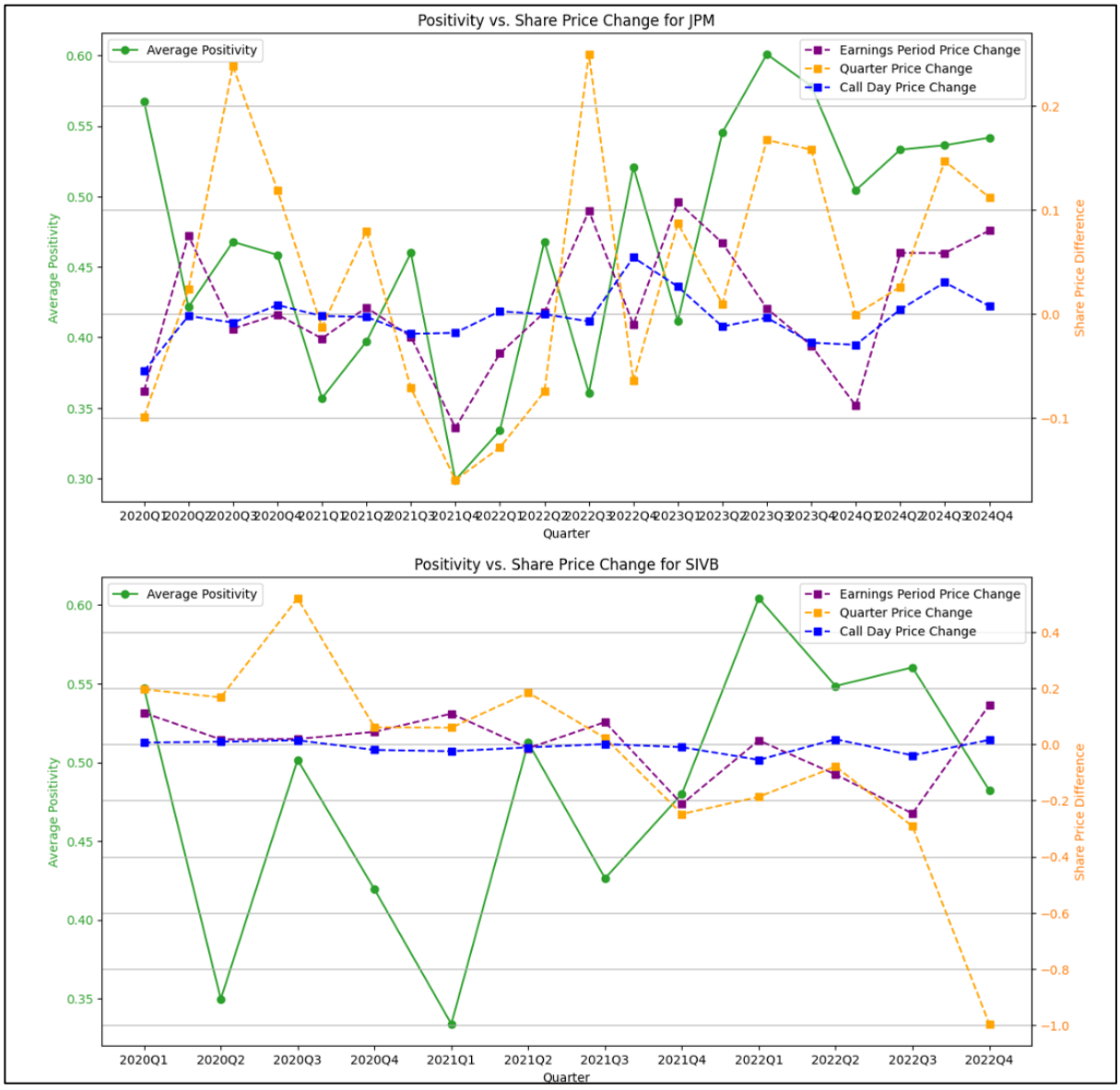


Figure 8: Positivity over time overlaid on stock price change around earnings calls

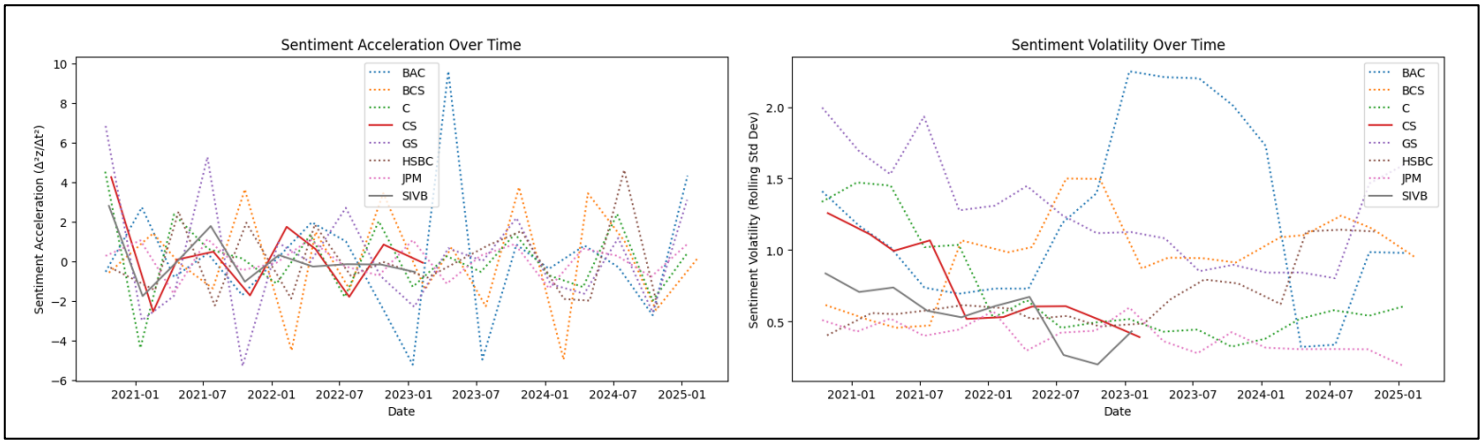


Figure 9: Sentiment acceleration and volatility over time, with Credit Suisse and Silicon Valley Bank highlighted

LLMs for Call Summary and Key Points

The Perplexity API, built off Perplexity AI's Sonar LLM, was used to create a user-facing summarisation model based off the data contained in the scraped earnings calls data.

```python
def ask_llm_for_summary(text: str, api_key: str) -> str:
    """
    SUMMARY
    """
    user_message = (
        "Summarize the following bank earnings call transcript:\n\n"
        f"{text}\n\n"
        "**In your summary, be sure to cover:**\n"
        "- Overall financial performance (revenues, expenses, profit/loss, margins)\n"
        "- Key operational highlights (business segments, new initiatives, major deals)\n"
        "- Management's outlook or guidance for future quarters\n"
        "- Market conditions or external factors that impacted performance\n"
        "- Any risks, challenges, or concerns mentioned\n\n"
        "Keep your response concise, well-structured, and focused on the main points. "
        "Whenever possible, provide specific figures or examples mentioned in the transcript.\n"
    )
    return sonar_chat(SYSTEM_MESSAGE, user_message, api_key)
```

```python
def ask_llm_for_takeaways(text: str, api_key: str) -> str:
    """
    TOP 3 TAKEAWAYS
    """
    user_message = (
        "Please read the following bank earnings call transcript:\n\n"
        f"{text}\n\n"
        "Identify the **top 3 takeaways** and present them in bullet-point format.\n"
        "Use the exact format:\n\n"
        "TAKEAWAYS:\n"
        "1) [Takeaway One]\n"
        "2) [Takeaway Two]\n"
        "3) [Takeaway Three]\n\n"
        "For each takeaway:\n"
        "- Provide a concise 1-2 sentence explanation.\n\n"
        "Do not include any additional commentary outside of these bullet points.\n"
        "Ensure the final output appears exactly in the 'TAKEAWAYS:' format described."
    )
    return sonar_chat(SYSTEM_MESSAGE, user_message, api_key)
```

```python
def ask_llm_for_topics(text: str, api_key: str) -> str:
    """
    TOP 3 TOPICS
    """
    user_message = (
        "Please read the following bank earnings call transcript:\n\n"
        f"{text}\n\n"
        "Identify the **top 3 topics** discussed and provide them in a consistent format.\n"
        "Use bullet points, and label them clearly as follows:\n\n"
        "TOPICS:\n"
        "1) [Topic One]\n"
        "2) [Topic Two]\n"
        "3) [Topic Three]\n\n"
        "For each topic:\n"
        "- Provide a short title.\n"
        "- Include a concise 1-2 sentence description.\n\n"
        "Do not include any additional commentary outside of these bullet points.\n"
        "Ensure the final output appears exactly in the 'TOPICS:' format described."
    )
    return sonar_chat(SYSTEM_MESSAGE, user_message, api_key)
```

```python
def ask_llm_for_concerns(text: str, api_key: str) -> str:
    """
    TOP 3 CONCERNS
    """
    user_message = (
        "Please read the following bank earnings call transcript:\n\n"
        f"{text}\n\n"
        "Identify the **top 3 concerns or risks** discussed and present them in bullet-point format.\n"
        "Use the exact format:\n\n"
        "CONCERNS:\n"
        "1) [Concern One]\n"
        "2) [Concern Two]\n"
        "3) [Concern Three]\n\n"
        "For each concern:\n"
        "- Provide a concise 1-2 sentence explanation.\n\n"
        "Do not include any additional commentary outside of these bullet points.\n"
        "Ensure the final output appears exactly in the 'CONCERNS:' format described."
    )
    return sonar_chat(SYSTEM_MESSAGE, user_message, api_key)
```

```python
def ask_llm_for_score(text: str, api_key: str) -> str:
    """
    SCORE OUT OF 100
    """
    user_message = (
        "Please read the following bank earnings call transcript:\n\n"
        f"{text}\n\n"
        "Give an **overall performance score** for the quarter, on a scale of 0 to 100.\n"
        "Use the exact format:\n\n"
        "SCORE:\n"
        "[Score Value]\n\n"
        "REASON:\n"
        "[1-2 sentence reason explaining the assigned score]\n\n"
        "Do not include any additional commentary or text outside of the above two fields.\n"
        "Ensure the final output appears exactly as 'SCORE:' then 'REASON:'."
    )
    return sonar_chat(SYSTEM_MESSAGE, user_message, api_key)
```

Figure 10: Code snippet containing five defined LLM functions as read by the Perplexity API on earnings call data

The five functions, as defined in figure 10, aim to

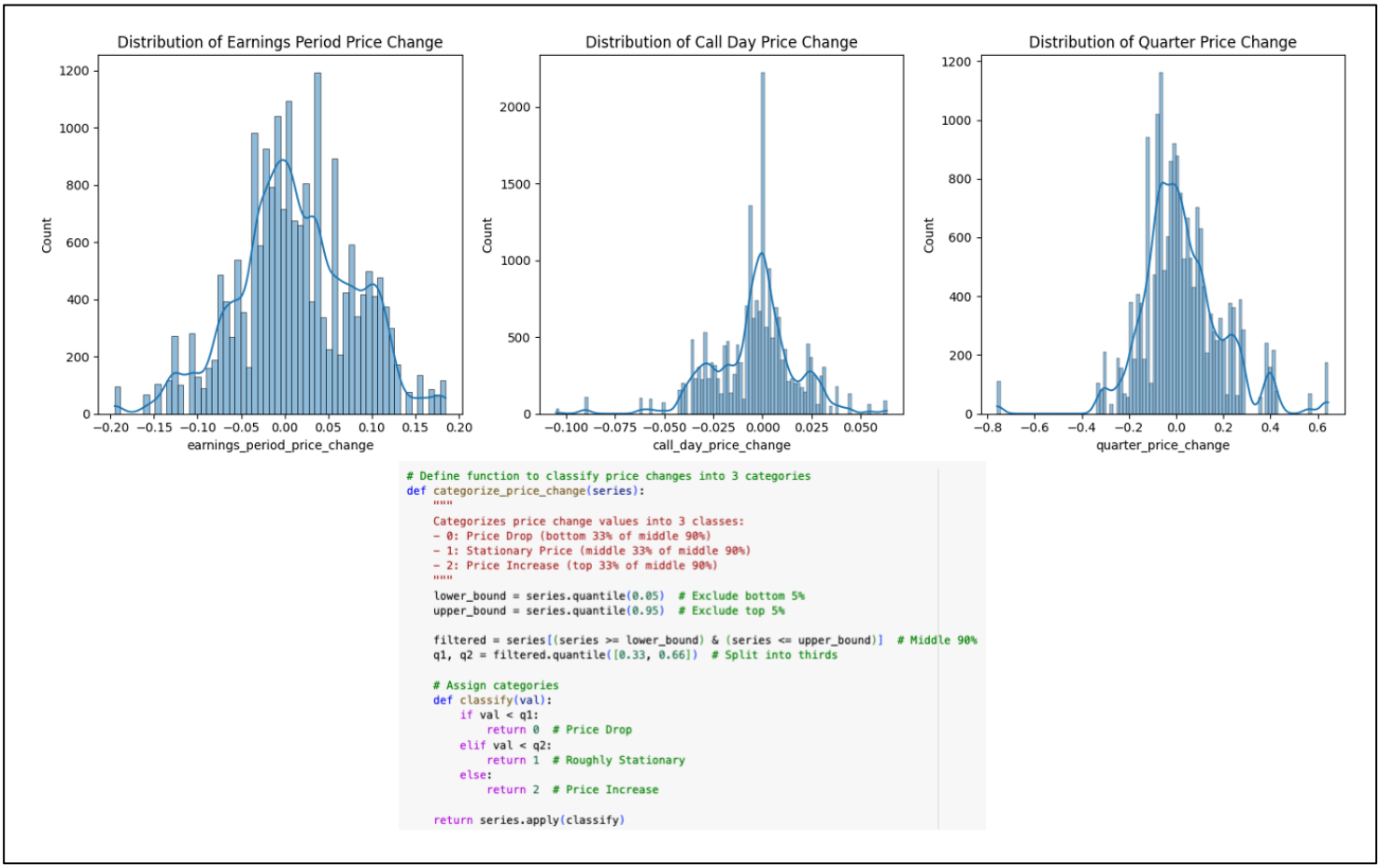
1. Summarise the full Q&A transcript
2. Outline three heavily discussed topics
3. Provide the BoE analyst with three takeaways for actioning or further analysis
4. Notify the BoE analyst of three potential causes of concern
5. Score the bank's performance in that financial quarter

This code's output is to serve as the body of the BoE-facing product. **Maybe a couple more lines here but code is pretty self-explanatory.**

XGBoost to Predict Price Change

With a view to provide the BoE with a prediction on a bank's financials in the aftermath of an earnings call, a neural network using eXtreme Gradient BOOSTing (XGBoost) was created, with text contained in the Q&A transcript vectorised using Term Frequency-Inverse Document Frequency (TF-IDF) acting as the sole input variable.

Creating an effective model necessitated introducing data from more calls and more banks, so a further seven banks (139 calls) were added to the dataset. The ideal output for this model was to inform an analyst how in which direction and how much stock price would change after an earnings call, based on a corpus of trained data from previous calls, however models using this approach were ineffective prompting a switch from an XGBoost for regression to an XGBoost for classification.




```python
# Define function to classify price changes into 3 categories
def categorize_price_change(series):
    """
    Categorizes price change values into 3 classes:
    - 0: Price Drop (bottom 33% of middle 90%)
    - 1: Stationary Price (middle 33% of middle 90%)
    - 2: Price Increase (top 33% of middle 90%)
    """
    lower_bound = series.quantile(0.05)  # Exclude bottom 5%
    upper_bound = series.quantile(0.95)  # Exclude top 5%

    filtered = series[(series >= lower_bound) & (series <= upper_bound)]  # Middle 90%
    q1, q2 = filtered.quantile([0.33, 0.66])  # Split into thirds

    # Assign categories
    def classify(val):
        if val < q1:
            return 0  # Price Drop
        elif val < q2:
            return 1  # Roughly Stationary
        else:
            return 2  # Price Increase

    return series.apply(classify)
```

Figure 11: Differences in distribution of stock price changes dependant on time window used

Since three metrics are stored for stock price change, categories had to be adjusted for the distribution of each. Thus three categories were defined: price drop, stationary price, and price increase relating to the bottom 33% (most negative),

middle 33% (most roughly stationary), and top 33% (mostly positive), when excluding 5% extremes on either end.

XGBoost models for all three target metrics were trained, with techniques including class weighting, Synthetic Minority Over-sample Technique (SMOTE), and hyperparameter tuning utilised to optimise results.

The quarter-period stock price change model produced by XGBoost resulted in a 58.3% accuracy – a **75.1% increase** from a 33% random selection (of price drop, price stationary, or price increase). This is an output to include in the client-facing product.

Results and Interface

Talk about streamlit, structure of the app (order big stuff up top, summarisation, supplementary graphs and how they may be used).

Conclusion

Achieve what we set out? How can we improve (more banks, looking into people and companies asking questions, as started in notebook but needs further fleshing out etc.). Definite scope to refine XGBoost model especially for earnings period