Labor vs Liberal – A Comparative Analysis of Political Strategies in Facebook

DATA7201 Project Report

Semester 1, 2025

Branden Lee 44789813

Abstract

Introduction: This report analyses the digital advertising strategies deployed by Australia's Labor and Liberal parties on the Facebook platform, focusing on the period leading up to the 2022 Federal election. The study aims to uncover patterns and effectiveness in their political strategies by investigating trends in various features in their ads.

Tools and Methods: Utilising a large dataset of sponsored posts from the Facebook Ads Library API, the analysis involved preprocessing roughly 40 million records with 26 columns using PySpark and Hadoop Distributed File System. We investigated the ad creation volume, audience reach, demographic, and the content and sentiment of the political messages. Sentiment analysis was performed using VADER and data was visualised using Seaborn and Matplotlib.

Results: Both parties significantly increased the number of ads before the 2022 election. The Labor party consistently deployed higher ad volumes, which directly correlated to higher ad impressions. For demographics, the Labor party's age distribution skewed towards the younger age groups while the Liberal party's age distribution was more evenly spread. Despite the differing distributions, ads consistently achieved its highest impressions among the 25-34 age group. Content analysis highlighted the distinct strategies where Labor focused on public assistance and used broader vocabulary while Liberal focused on economic issues. Sentiment analysis highlighted both parties were using mostly positive sentiments, but Labor exhibited a wider range, including more negative sentiments.

Conclusion: The findings highlight the critical role of having strong presence in digital platforms for political campaigns. Parties can significantly maximise their reach through higher ad volume and prioritising the 25-34 age demographic. The study also finds that deploying techniques in the content of the ads may also positively impact reach.

Table of Contents

Word count: 1483 (excluding abstract, table of contents, and appendix)

Introduction

The age of social media and digital platforms has led to an outbreak of immense, complex datasets, often referred to as big data. These datasets are so big that traditional, single-machine systems cannot store or analyse them. Big data analytics aims to solve these challenges by introducing various infrastructures to store, process and analyse big data. For example, an e-commerce platform may want to track and analyse their users' behaviour and shopping habits to deliver highly personalised product recommendations. This will quickly result in overwhelming processing times and storage requirements on traditional systems. These challenges require a big data analytics solution like distributed systems, which breaks down large tasks across interconnected machines to achieve parallel processing and enables timely, scalable analysis of big data. This report explores the use of distributed systems in another example: analysing political advertising strategies on Facebook deployed by the Australian Labor Party and the Liberal Party of Australia. More specifically, we will investigate the use of big data analytical tools like PySpark and analyse trends in ad creation, spending, audience reach, and the content of political messages leading up to the 2022 Federal election. By the end, we should be able to understand their strategies and its effectiveness, potentially understanding the reasoning behind the Labor party's victory.

Dataset Analytics

Dataset and Tools

The dataset consists of sponsored political posts targeting Australian users on the Facebook platform. In its original form, it spans a four-year period from March 2020 to February 2024 and encompasses significant Australian political events like the lead-up to the May 2022 Federal election and the October 2023 Voice referendum, which may explain any trends found in the data. The Facebook Ads Library API provides the dataset in JSON file format, with each record representing a request for an active ad campaign performed approximately every 12 hours. The dataset comprises roughly 40 million records with 26 columns. This necessitates proper preprocessing due to the data collection process and its format.

PySpark is the main data analytics tool used due to its ability to handle very large datasets and big data processing tasks, where traditional tools would struggle because of memory and processing constraints. Furthermore, PySpark integrates seamlessly with other tools like Hadoop Distributed File System (HDFS), which is an ideal storage solution as it can distribute and manage large volumes of data. Thus, the data is stored on HDFS, pre-processed using PySpark, and visualised using Python libraries such as Seaborn and Matplotlib.

Preprocessing

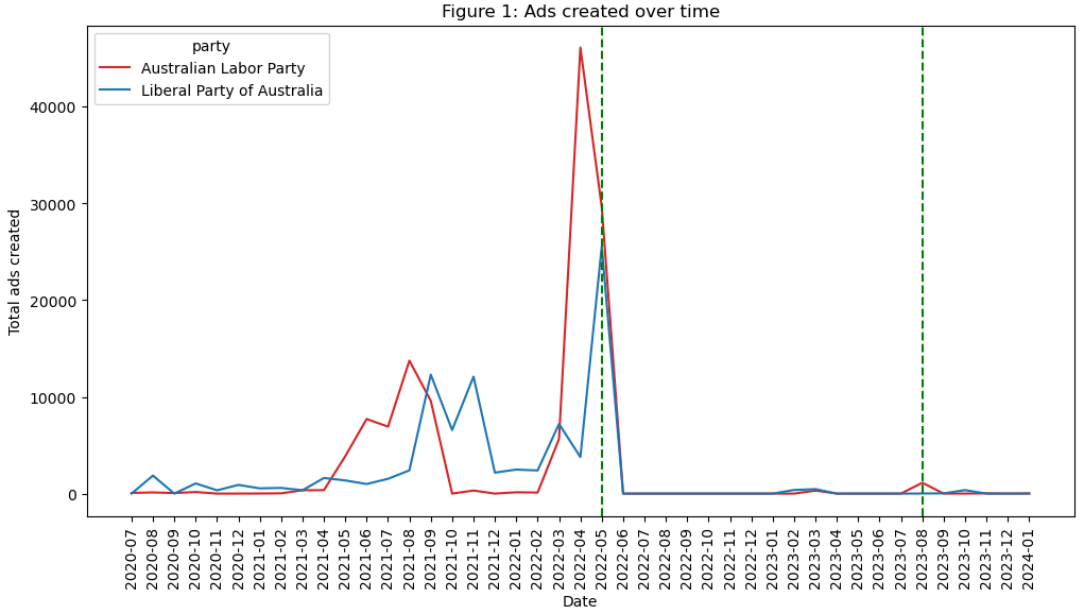
To perform the various analytics, the dataset first needs preprocessing. Firstly, each record represents an active ad campaign request, but it can be performed multiple times a day, resulting in duplicates. PySpark's built-in functions were used to handle these duplicates, halving the total number of records from 39,584,139 to 20,694,375. Secondly, specific columns require type casting for proper aggregation. For instance, the "ad_creation_time" column, which represents when an advertisement was created, was casted to a date format. Finally, unnecessary data was filtered out to retain only the most relevant data for various stages of the analysis.

Analysis

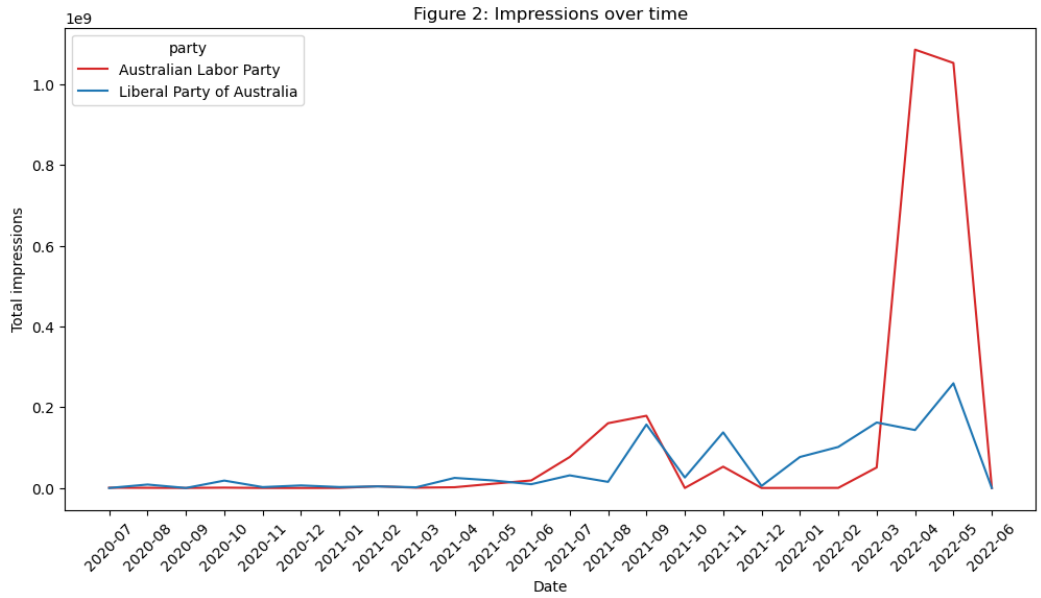
Ad campaign volume and impressions

After preprocessing, we first examined the number of ad campaigns created over time for each party, primarily using the “ad_creation_time” and “funding_entity” columns. As per the Facebook API documentation, the “funding_entity” column represents the entity funding the post, with many posts being funded by Australia’s two main political parties: Labor and Liberal. For the purpose of this report, the data was filtered to include only posts created by these two parties, resulting in Figure 1. The plot illustrates when the political parties were most active, with the dotted green lines indicating key political events. Evidently, most of the volume and variation occurs on and before the first dotted line, which signifies the 2022 Federal election. Clearly, the Labor party created a significantly greater number of ads than the Liberal party, though both exhibited large spikes in the months leading up to this event. Given this concentration, the subsequent analysis focused on data preceding this critical election.

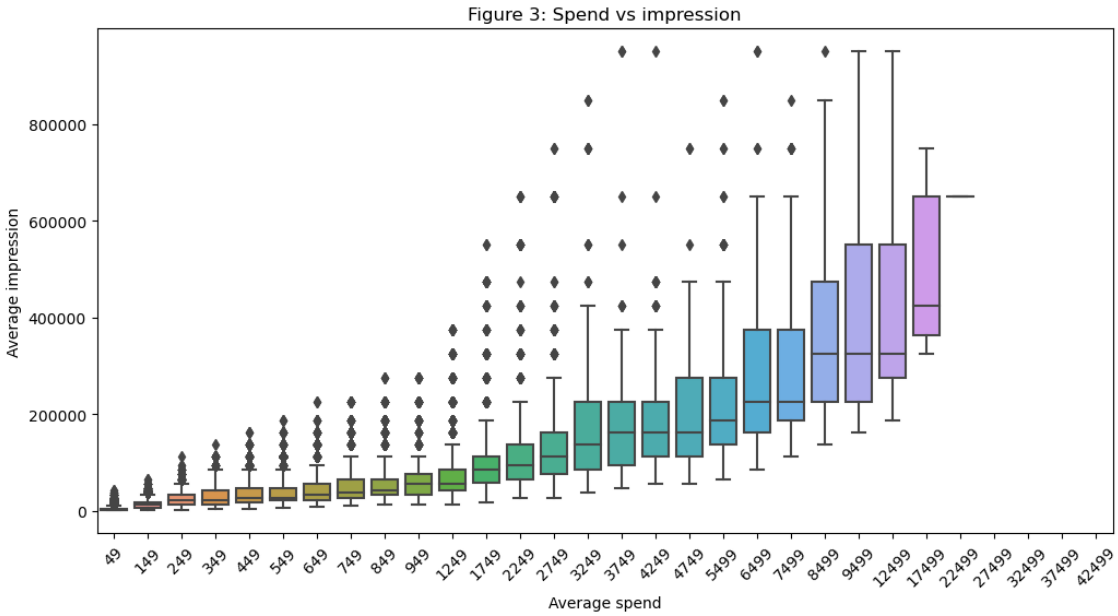
Figure 1: Ads created over time



Further exploration investigated impressions for each post, defined by Facebook as an instance of a post being displayed to a user. We used the midpoint of the estimated impression ranges to serve as the average impression per post. As seen in Figure 2, the plots mirrored Figure 1, suggesting that the number of posts directly impacts the number of people who could see them. Similarly, the “spend” column, representing the estimated AUD amount spent on each post, showed very comparable results (see Appendix A: Figure 1). This led us to investigate the correlation between spend and impression, seeking to determine if increased spending yielded more impressions.



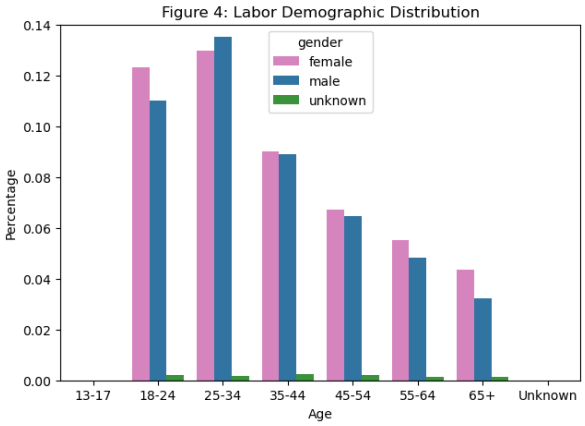
Since the dataset provides estimated ranges from a discrete set, spend values can have multiple different impression values. Consequently, a boxplot was chosen to visualise the average impression and range a party could expect for specific spend amounts. We found a fair positive correlation of 0.8097, where being closer to 1 is considered highly correlated. While Figure 3 does present an upwards curve, supporting the positive correlation, less data is available as the spend value increases. This is clear at the higher spend values where no box plots are present, suggesting insufficient data to draw firm conclusions at higher expenditure levels.

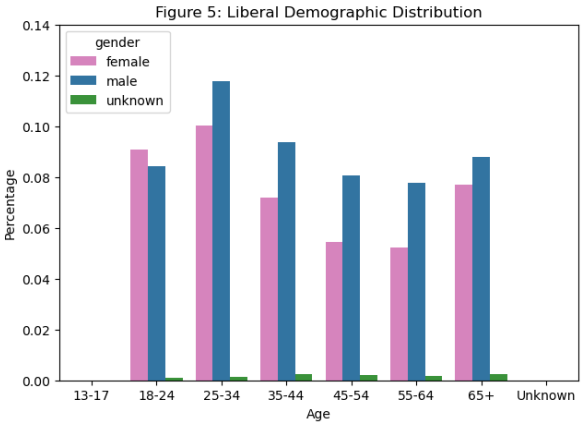


Demographic distribution

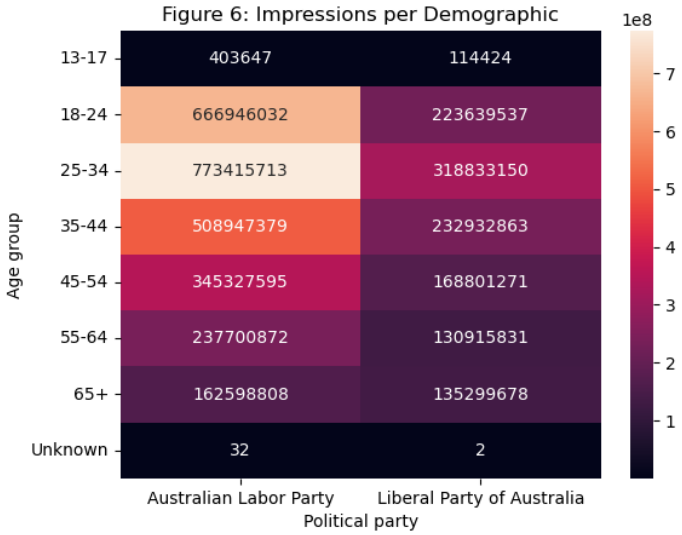
The analysis then shifted focus from financial investment to the demographics. The "demographic_distribution" column provides percentages for each demographic group reached by a post. By comparing Figure 4 and Figure 5 directly, we observed both similarities and clear differences. As expected, both parties did not reach underaged users. On the other hand, one key difference is

the fact that the distribution for the Labor party is skewed to the left, while the distribution for the Liberal party is relatively even across the age groups.





The impact of this can be seen in the heatmap showing impression values by age group. For both parties, the impression value was highest for the 25-34 age group, significantly surpassing older age groups. This is held true despite the Liberal party's somewhat evenly distributed age demographic, implying that focusing on younger audiences, particularly those aged 25-34, yields the most effective reach.



Ad content and sentiment

In this final segment, we analysed the content of the posts from the "ad_creative_body" column. Duplicate posts, arising from regional replication, were removed. First, we identified the top 200 words used by each party and created Word Clouds. Comparing these Word Clouds revealed distinct messaging strategies. On one hand, the Labor party seems to emphasise helping the public and used a larger variety of words. On the other hand, the Liberal party emphasises economic terms. Notably, both parties frequently mention the opposing party and their leaders.

Figure 7: Labor Word Cloud

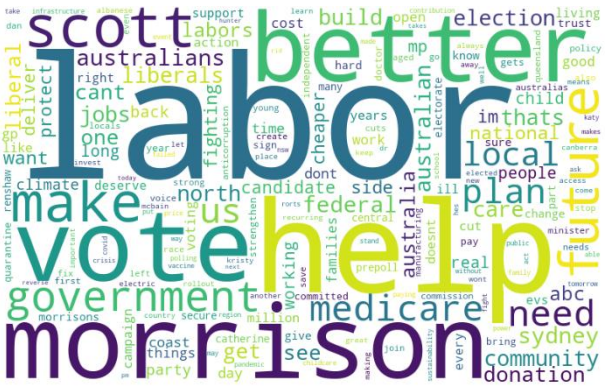
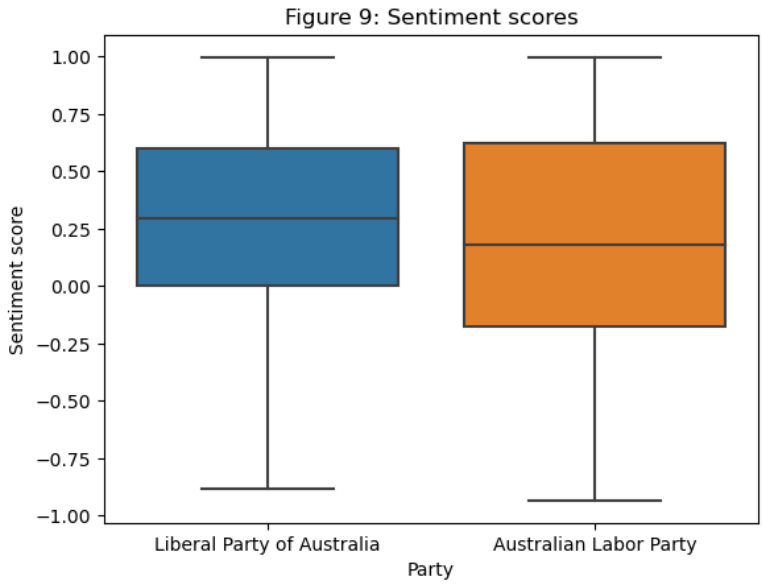


Figure 8: Liberal Word Cloud

Further text analysis involved sentiment analysis using VADER to determine the emotional tone of the posts by scoring them. A score higher than 0.05 is labelled as positive while a score lower than -0.05 is labelled negative. Anywhere in between is considered neutral. The boxplot shows that both parties generally used positive sentiment. However, the Labor party exhibits a larger sentiment range, suggesting they incorporated more negative tones than the Liberal party. This finding should be contextualised by the fact that Scott Morrison, from the Liberal party was the Australian Prime Minister before the election.

Figure 9: Sentiment scores



Discussion and Conclusions

Our analysis reveals clear distinctions in the digital advertising strategies employed by Australia's major political parties. The Labor party consistently deployed a significantly higher volume of ads, a strategy that directly translated to greater surges in ad impressions compared to the Liberal party. Demographically, Labors ads notably skewed towards younger age groups that contrasts with the Liberal party's more evenly distributed reach across age demographics. However, regardless of demographic targeting for both parties, the 25-34 age group critically demonstrated higher ad impressions, highlighting this as the prime demographic for maximum

reach on this platform. Content-wise, Labor's messaging emphasised public assistance and exhibited broader vocabulary, while the Liberal party focused predominantly on economic issues. Although both maintained a mostly positive tone, sentiment analysis further differentiated their approaches. Labor's ads exhibited a wider emotional range, including more negative sentiment. This could potentially be criticism of the opposing party or the leader, which could be attributed to their role as the opposing party criticising the then-elected Liberal leader.

In conclusion, these insights highlight several strategic lessons: the undeniable importance of a strong digital presence in a pre-election phase, the impact of ad volume and funding on reach, and the specific demographic sweet spot for maximising engagement. Moreover, the distinct messaging strategies and sentiment usage by each party highlights how content can be tailored to political objectives. Future political campaign strategies on digital platforms like Facebook can leverage these findings to optimise ad placement and refine messages to engage and emphasise with a younger audience.

Appendix A

Figure 1: Spend plot

The plot below shows the spend values over times. Evidently, the Labor party spent a significant amount more than the Liberal party. Interestingly, the trends in the plot are similar to the impression plots.

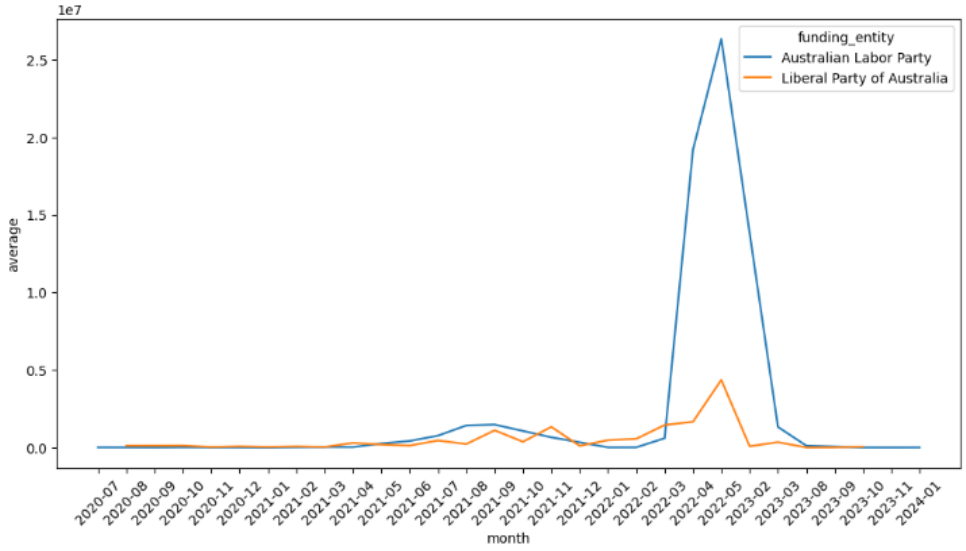
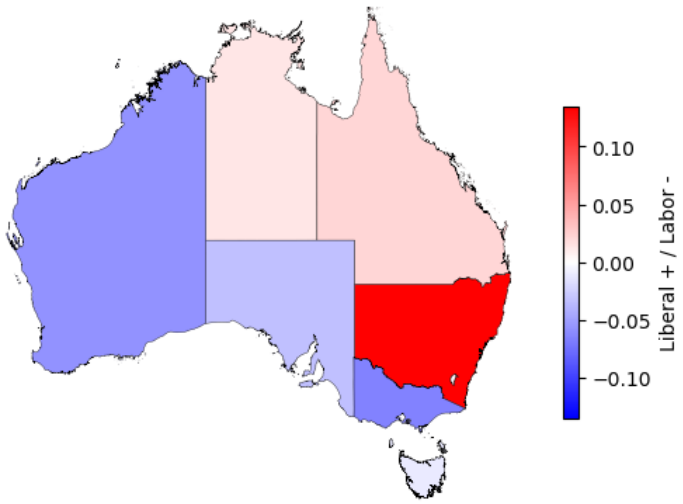


Figure 2: Regional distribution

The distributions across the different states and territories were also investigated. Here, we assume the “region_distribution” column represents the areas where the ads were delivered. Due to the difference in the number of ads posted by each party, we found the proportions instead and directly compared them, resulting in the figure below. Darker shades suggest more dominance for the parties, while lighter shades suggest that there was little dominance.



Appendix B – PySpark Code

PySpark Setup

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("project") \
    .getOrCreate()

df = spark.read.json("/data/ProjectDatasetFacebookAU/fbadsAU")
# df.show(5)


from pyspark.sql.functions import *
from pyspark.sql.types import *
```

Preprocessing

1) Drop duplicate records

```
# drop duplicate tuples
distinct_df = df.dropDuplicates()

# number of records in the original
df.count()
```

```
39584139
```

```
# number of records after removing duplicates
distinct_df.count()
```

```
20694375
```

2) Change column types

Change ad creation time from String to Date

```
from pyspark.sql.functions import to_date, col

# convert date columns from String to Date
distinct_df = distinct_df.withColumn("ad_creation_time", to_date("ad_creation_time"))

distinct_df.printSchema()
```

```
root
 |-- ad_creation_time: date (nullable = true)
 |-- ad_creative_bodies: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- ad_creative_body: string (nullable = true)
 |-- ad_creative_link_caption: string (nullable = true)
 |-- ad_creative_link_captions: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- ad_creative_link_description: string (nullable = true)
 |-- ad_creative_link_descriptions: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- ad_creative_link_title: string (nullable = true)
 |-- ad_creative_link_titles: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- ad_delivery_start_time: string (nullable = true)
 |-- ad_delivery_stop_time: string (nullable = true)
 |-- ad_snapshot_url: string (nullable = true)
 |-- bylines: string (nullable = true)
 |-- currency: string (nullable = true)
 |-- delivery_by_region: array (nullable = true)
 |    |-- element: struct (containsNull = true)
```

```
|    |    |-- percentage: string (nullable = true)
|    |    |-- region: string (nullable = true)
|-- demographic_distribution: array (nullable = true)
|    |-- element: struct (containsNull = true)
|    |    |-- age: string (nullable = true)
|    |    |-- gender: string (nullable = true)
|    |    |-- percentage: string (nullable = true)
|-- estimated_audience_size: struct (nullable = true)
|    |-- lower_bound: string (nullable = true)
|    |-- upper_bound: string (nullable = true)
|-- funding_entity: string (nullable = true)
|-- id: string (nullable = true)
|-- impressions: struct (nullable = true)
|    |-- lower_bound: string (nullable = true)
|    |-- upper_bound: string (nullable = true)
|-- languages: array (nullable = true)
|    |-- element: string (containsNull = true)
|-- page_id: string (nullable = true)
|-- page_name: string (nullable = true)
|-- publisher_platforms: array (nullable = true)
|    |-- element: string (containsNull = true)
|-- region_distribution: array (nullable = true)
|    |-- element: struct (containsNull = true)
|    |    |-- percentage: string (nullable = true)
|    |    |-- region: string (nullable = true)
|-- spend: struct (nullable = true)
|    |-- lower_bound: string (nullable = true)
|    |-- upper_bound: string (nullable = true)
```

3) Filter data

Find all ad campaigns that were funded by the two main political parties

```
# filter so it only contains ads funded by the select political parties
parties = ["Liberal Party of Australia", "Australian Labor Party"]
distinct_df = distinct_df.filter(col("funding_entity").isin(parties))
```

Convert date format to year and month for time series analysis

```
distinct_df = distinct_df.withColumn("month", date_format("ad_creation_time", "yyyy-MM"))
```

Analysis

Ads created over time

In some months, the parties may not create any new ads.

So first create a data frame that contains the months for each party

```
from pyspark.sql.functions import *
from pyspark.sql.types import *

start_date = "2020-07-01"
end_date = "2024-01-01"

# dataframe containing every month between 2020-07-01 and 2024-01-01
date_range_df = spark.sql(f"""
    SELECT explode(sequence(to_date('{start_date}'), to_date('{end_date}'), interval 1 month)) as
month
""")

# convert to monthly format
months_df = date_range_df.select(date_format("month", "yyyy-MM").alias("date"))

# get the two parties
# parties = ["Liberal Party of Australia", "Australian Labor Party"]
# parties_df = distinct_df.filter(col("funding_entity").isin(parties))
parties_df = distinct_df.select(col("funding_entity").alias("party"))
parties_df = parties_df.dropDuplicates()

# cross join parties and dates
full_grid = months_df.crossJoin(parties_df)
```

```
full_grid.show()
full_grid.count()
```

```
+-------+--------------------+
|   date|               party|
+-------+--------------------+
|2020-07|Liberal Party of ...|
|2020-08|Liberal Party of ...|
|2020-09|Liberal Party of ...|
|2020-10|Liberal Party of ...|
|2020-11|Liberal Party of ...|
|2020-12|Liberal Party of ...|
|2021-01|Liberal Party of ...|
|2021-02|Liberal Party of ...|
|2021-03|Liberal Party of ...|
|2021-04|Liberal Party of ...|
|2021-05|Liberal Party of ...|
|2021-06|Liberal Party of ...|
|2021-07|Liberal Party of ...|
|2021-08|Liberal Party of ...|
|2021-09|Liberal Party of ...|
|2021-10|Liberal Party of ...|
|2021-11|Liberal Party of ...|
|2021-12|Liberal Party of ...|
|2022-01|Liberal Party of ...|
|2022-02|Liberal Party of ...|
+-------+--------------------+
only showing top 20 rows


86
```

Then find the monthly counts for each party

```
ad_count_df = distinct_df

# select date and party columns
ad_count_df = ad_count_df.select(
    col("month").alias("date"),
    col("funding_entity").alias("party"),
)

ad_count_df.na.drop("any")

# find monthly ads released by each party
ad_count_df = ad_count_df.groupBy(["date", "party"]).count().orderBy("date")

# join on the cross join performed earlier
# any empty dates will have 0 counts
result = full_grid.join(ad_count_df, on=["date", "party"], how="left").fillna({"count": 0})

# order by date
result = result.orderBy("date")

# convert to pandas dataframe
result_pandas = result.toPandas()

print(result_pandas)
```

```
       date                       party  count
0   2020-07      Australian Labor Party     70
1   2020-07  Liberal Party of Australia      0
2   2020-08      Australian Labor Party    119
3   2020-08  Liberal Party of Australia   1861
4   2020-09      Australian Labor Party     55
..      ...                         ...    ...
81  2023-11  Liberal Party of Australia      0
82  2023-12      Australian Labor Party      0
83  2023-12  Liberal Party of Australia      0
84  2024-01      Australian Labor Party     18
85  2024-01  Liberal Party of Australia      0
```

```
[86 rows x 3 columns]
```

```
# save to a local CSV file for visualisations
result_pandas.to_csv("counts.csv")
```

Spend vs Impressions

From the section above, it was found that little to no ad campaigns were created after the Federal election (any dates past May 2022).

Since the focus of the report has been narrowed down, the data should be filtered out first.

```
filtered_df = distinct_df.filter(col("ad_creation_time") < "2022-06-01")
```

Finding the impressions requires converting the columns to appropriate formats. As the impressions are provided as a range, we need to compute the average, or the midpoint of the range.

```
# set new individual columns for impressions: lower bound, upper bound, average
impressions_df = filtered_df.withColumn("impression_lower",
col("impressions.lower_bound").cast("int"))
impressions_df = impressions_df.withColumn("impression_upper",
col("impressions.upper_bound").cast("int"))
impressions_df = impressions_df.withColumn("impression_avg", ((col("impression_lower") +
col("impression_upper")) / 2).cast("int"))

# find monthly values of impressions for each party
impressions_df = impressions_df.groupBy(["month", "funding_entity"]).agg(
    sum("impression_lower").alias("lower_bound"),
    sum("impression_upper").alias("upper_bound"),
    sum("impression_avg").alias("average")
).orderBy("month")

# rename columns for joining in next step
impressions_df = impressions_df.withColumnRenamed("funding_entity", "party")
impressions_df = impressions_df.withColumnRenamed("month", "date")

# set months with no ads created to 0
full_grid_filtered = full_grid.filter(col("date") < "2022-06-01")
impressions_df = full_grid_filtered.join(impressions_df, on=["date", "party"],
how="left").fillna({"average": 0})
impressions_df = impressions_df.orderBy(["date", "party"])

# convert to pandas dataframe
impressions_df_pandas = impressions_df.toPandas()

print(impressions_df_pandas)

impressions_df_pandas.to_csv("impressions.csv")
```

The spend values are also provided as a range, so the same process is applied here.

```
# set new individual columns for impressions: lower bound, upper bound, average
spend_df = filtered_df.withColumn("spend_lower", col("spend.lower_bound").cast("int"))
spend_df = spend_df.withColumn("spend_upper", col("spend.upper_bound").cast("int"))
spend_df = spend_df.withColumn("spend_avg", ((col("spend_lower") + col("spend_upper")) /
2).cast("int"))

# find monthly values of impressions for each party
spend_df = spend_df.groupBy(["month", "funding_entity"]).agg(
    sum("spend_lower").alias("lower_bound"),
    sum("spend_upper").alias("upper_bound"),
    sum("spend_avg").alias("average")
).orderBy("month")

# rename columns for joining in next step
spend_df = spend_df.withColumnRenamed("funding_entity", "party")
spend_df = spend_df.withColumnRenamed("month", "date")

# set months with no ads created to 0
```

```
spend_df = full_grid_filtered.join(spend_df, on=["date", "party"], how="left").fillna({"average": 0})

spend_df.toPandas()  # convert to pandas dataframe

print(spend_df_pandas)

spend_df_pandas.to_csv("spend.csv")
```

Looking at the impressions and the spends on their own does not show much. We may want to see if spend is correlated to impressions.

Repeat the previous process for a new dataframe

```
spend_impressions_df = filtered_df.withColumn("spend_lower", col("spend.lower_bound").cast("int"))
spend_impressions_df = spend_impressions_df.withColumn("spend_upper",
col("spend.upper_bound").cast("int"))
spend_impressions_df = spend_impressions_df.withColumn("spend_avg", ((col("spend_lower") +
col("spend_upper")) / 2).cast("int"))

spend_impressions_df = spend_impressions_df.withColumn("impression_lower",
col("impressions.lower_bound").cast("int"))
spend_impressions_df = spend_impressions_df.withColumn("impression_upper",
col("impressions.upper_bound").cast("int"))
spend_impressions_df = spend_impressions_df.withColumn("impression_avg", ((col("impression_lower") +
col("impression_upper")) / 2).cast("int"))

spend_impressions_pandas = spend_impressions_df.select("spend_avg", "impression_avg").toPandas()



spend_impressions_pandas.to_csv("spend_impressions.csv")
```

However, the values for spend and impressions are from a set of predefined values.

```
aggregated_df = spend_impressions_df.groupBy("spend_avg") \
                                    .agg(
                                        count("*").alias("ad_count"),
                                        avg("impression_avg").alias("mean_impressions")
                                    ) \
                                    .orderBy("spend_avg")

aggregated_df.show()



aggregated_pandas = aggregated_df.toPandas()
aggregated_pandas.to_csv("agg_spend_impressions.csv")
```

Demographic Distribution

Find the age groups for each party

```
df_exp = filtered_df.select(
    "funding_entity",
    explode("demographic_distribution").alias("demographic")
)

df_exp = df_exp.select(
    col("funding_entity").alias("party"),
    col("demographic.age").alias("age"),
    col("demographic.gender").alias("gender"),
    col("demographic.percentage").cast("float").alias("percentage")
)

# sums percentages for each age/gender group
age_distribution = df_exp.groupBy("party", "age", "gender") \
    .agg(sum("percentage").alias("total_percentage") ) \
    .orderBy("party", desc("age"), "gender")

# sum of total percentages for each party
```

```
party_totals = age_distribution.groupBy("party").agg(sum("total_percentage").alias("party_total"))

# normalised percentages for each party
age_distribution = age_distribution.join(party_totals, on="party") \
    .withColumn("normalised_percentage", col("total_percentage") / col("party_total"))

# age_distribution.filter(col("party")=="Australian Labor Party").show()

# check that sum of normalised percentages add to 1
age_distribution.filter(col("party")=="Australian Labor
Party").agg(sum("normalised_percentage")).show()
```

Labor Party

```
# separate dataframe for Labor party
labor_age_dist = age_distribution.filter(col("party")=="Australian Labor Party").orderBy("age")

# convert to pandas dataframe
labor_age_dist_pandas = labor_age_dist.toPandas()

# save to csv file
labor_age_dist_pandas.to_csv("labor_age_distribution.csv")
```

Liberal Party

```
# separate dataframe for Liberal party
liberal_age_dist = age_distribution.filter(col("party")=="Liberal Party of Australia").orderBy("age")

# convert to pandas dataframe
liberal_age_dist_pandas = liberal_age_dist.toPandas()

# save to csv file
liberal_age_dist_pandas.to_csv("liberal_age_distribution.csv")
```

Impression per Age Demographic

Repeat the same process as above but also include impression

```
df_exp = filtered_df.select(
    "funding_entity",
    col("impressions.lower_bound").cast("int").alias("impression_lower"),
    col("impressions.upper_bound").cast("int").alias("impression_upper"),
    explode("demographic_distribution").alias("demographic")
)

df_exp = df_exp.select(
    col("funding_entity").alias("party"),
    "impression_lower",
    "impression_upper",
    col("demographic.age").alias("age"),
    col("demographic.gender").alias("gender"),
    col("demographic.percentage").cast("float").alias("percentage")
)

df_exp = df_exp.withColumn("est_impressions", (col("impression_lower") + col("impression_upper")) / 2
* col("percentage"))

# drop any null values
df_exp.na.drop("any")

# sums percentages for each age/gender group
age_distribution = df_exp.groupBy("party", "age", "gender") \
    .agg(sum("percentage").alias("total_percentage") ) \
    .orderBy("party", desc("age"), "gender")

# sum of total percentages for each party
party_totals = age_distribution.groupBy("party").agg(sum("total_percentage").alias("party_total"))
```

```
# normalised percentages for each party
age_distribution = age_distribution.join(party_totals, on="party") \
    .withColumn("normalised_percentage", col("total_percentage") / col("party_total"))

# age_distribution.filter(col("party")=="Australian Labor Party").show()

# find impressions based on demographic
impressions_per_demo = df_exp.groupBy("party", "age", "gender") \
    .agg(sum("est_impressions").alias("total_impressions") ) \
    .orderBy("party", desc("age"), "gender")

impressions_per_demo.show()



impressions_per_demo_pandas = impressions_per_demo.toPandas()
impressions_per_demo_pandas.to_csv("impressions_per_demo.csv")
```

Labor party

```
labor_impressions_per_demo = impressions_per_demo.filter(col("party")=="Australian Labor
Party").orderBy("age")
labor_impressions_per_demo_pandas = labor_impressions_per_demo.toPandas()
labor_impressions_per_demo_pandas.to_csv("labor_impressions_demo.csv")
```

Liberal party

```
liberal_impressions_per_demo = impressions_per_demo.filter(col("party")=="Liberal Party of
Australia").orderBy("age")
liberal_impressions_per_demo_pandas = liberal_impressions_per_demo.toPandas()
liberal_impressions_per_demo_pandas.to_csv("liberal_impressions_demo.csv")
```

Regional Distribution

Not used in the report as there were no major findings

```
valid_regions = [
    "New South Wales",
    "Queensland",
    "Victoria",
    "Western Australia",
    "South Australia",
    "Northern Territory",
    "Tasmania",
    "Australian Capital Territory"
]

region_df = filtered_df.withColumn("region_info", explode("region_distribution"))

region_df = region_df.withColumn("region", col("region_info.region"))
region_df = region_df.withColumn("percentage", col("region_info.percentage").cast("float"))

valid_regions = [
    "New South Wales",
    "Queensland",
    "Victoria",
    "Western Australia",
    "South Australia",
    "Northern Territory",
    "Tasmania",
    "Australian Capital Territory"
]
region_df = region_df.filter(col("region").isin(valid_regions))

region_distribution = region_df.groupBy("region",
"funding_entity").agg(sum("percentage").alias("total_percentage"))
# region_distribution.show()
```

```
# sum of total percentages for each party
party_region_totals =
region_distribution.groupBy("funding_entity").agg(sum("total_percentage").alias("party_total"))

# normalised percentages for each party
region_distribution = region_distribution.join(party_region_totals, on="funding_entity") \
    .withColumn("normalised_percentage", col("total_percentage") / col("party_total"))

# region_distribution.show()

region_distribution_pandas = region_distribution.toPandas()
region_distribution_pandas.to_csv("regions.csv")
```

Text analysis

Sentiment Analysis

```
text_df = filtered_df.select("funding_entity", "ad_creative_body", "ad_creation_time").dropna()

text_pandas = text_df.toPandas()
text_pandas.to_csv("text_data.csv")
```

Find the word frequencies for each party

```
text_df = filtered_df.select("funding_entity", "ad_creative_body").dropna()
# dropping duplicates
text_df = text_df.dropDuplicates()

# filter based on party
labor_text_df = text_df.filter(col("funding_entity")=="Australian Labor Party")
liberal_text_df = text_df.filter(col("funding_entity")=="Liberal Party of Australia")

# remove punctuation
labor_words_df = labor_text_df.select(explode(
    split(
        regexp_replace(lower("ad_creative_body"), "[^a-z\s]", ""), "\s+"
    )
).alias("word"))

liberal_words_df = liberal_text_df.select(explode(
    split(
        regexp_replace(lower("ad_creative_body"), "[^a-z\s]", ""), "\s+"
    )
).alias("word"))
```

Convert the words to a single array

```
labor_words_df = labor_words_df.withColumn("word", split("word", " "))
```

Remove stop words

```
from pyspark.ml.feature import StopWordsRemover

remover = StopWordsRemover(inputCol="word", outputCol="filtered_word")

labor_words_df = remover.transform(labor_words_df).select(explode("filtered_word").alias("word"))
```

Find the top 200 words for Word Cloud

```
labor_word_counts = labor_words_df.groupBy("word").count().orderBy("count", ascending=False)
labor_top_words = labor_word_counts.limit(200).toPandas()



labor_top_words.to_csv("labor_words.csv")
```

Repeat for Liberal party

```
liberal_words_df = liberal_words_df.withColumn("word", split("word", " "))
liberal_words_df = remover.transform(liberal_words_df).select(explode("filtered_word").alias("word"))
liberal_word_counts = liberal_words_df.groupBy("word").count().orderBy("count", ascending=False)
liberal_top_words = liberal_word_counts.limit(200).toPandas()
liberal_top_words.to_csv("liberal_words.csv")
```

Appendix C – Python Code

Figure 1 visualisation

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv("DATA7201/counts.csv")

party_colors = {
    "Australian Labor Party": "C3",
    "Liberal Party of Australia": "C0",
}

plt.figure(figsize=(12,6))
sns.lineplot(data=data, x="date", y="count", hue="party", palette=party_colors)
plt.axvline(x="2022-05", color="g", linestyle="--", label="Federal election")
plt.axvline(x="2023-08", color="g", linestyle="--", label="Referendum")
plt.xticks(rotation=90)
plt.xlabel("Date")
plt.ylabel("Total ads created")
plt.title("Figure 1: Ads created over time")
plt.show()
```

Figure 2 visualisation

```
impressions = pd.read_csv("DATA7201/impressions.csv")

plt.figure(figsize=(12,6))
sns.lineplot(data=impressions, x="date", y="average", hue="party", palette=party_colors)
plt.xticks(rotation=45)
plt.xlabel("Date")
plt.ylabel("Total impressions")
plt.title("Figure 2: Impressions over time")
plt.show()
```

Figure 3 and 4 visualisation

```
labor_age = pd.read_csv("DATA7201/labor_age_distribution.csv")
labor_age = labor_age.sort_values(by=["age", "gender"])

liberal_age = pd.read_csv("DATA7201/liberal_age_distribution.csv")
liberal_age = liberal_age.sort_values(by=["age", "gender"])

gender_colors = {
    "male": "C0",
    "female": "C6",
    "unknown": "C2"
}

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(16, 5))

# plot Labor / left side
sns.barplot(
    data=labor_age,
    x="age",
    y="normalised_percentage",
    hue="gender",
    palette=gender_colors,
    ax=axes[0]
)
axes[0].set_xlabel("Age")
axes[0].set_ylabel("Percentage")
```

```
axes[0].set_ylim(0.00, 0.14)
axes[0].set_title("Figure 4: Labor Demographic Distribution")

# plot liberal / right side
sns.barplot(
    data=liberal_age,
    x="age",
    y="normalised_percentage",
    hue="gender",
    palette=gender_colors,
    ax=axes[1]
)
axes[1].set_xlabel("Age")
axes[1].set_ylabel("Percentage")
axes[1].set_ylim(0.00, 0.14)
axes[1].set_title("Figure 5: Liberal Demographic Distribution")
plt.show()
```

Figure 5 visualisation

```
spend_vs_impressions = pd.read_csv("DATA7201/spend_impressions.csv")

plt.figure(figsize=(12,6))
sns.boxplot(data=spend_vs_impressions, x="spend_avg", y="impression_avg")
# sns.regplot(data=spend_vs_impressions, x="spend_avg", y="impression_avg")
plt.xticks(rotation=45)
plt.title("Figure 3: Spend vs impression")
plt.xlabel("Average spend")
plt.ylabel("Average impression")
plt.show()
```

Spend vs Impression correlation

```
spend_vs_impressions.corr()
```

	Unnamed: 0	spend_avg	impression_avg
Unnamed: 0	1.000000	0.003827	0.002597
spend_avg	0.003827	1.000000	0.809697
impression_avg	0.002597	0.809697	1.000000

Figure 6 visualisation

```
pdf = pd.read_csv("DATA7201/impressions_per_demo.csv")
pivot = pdf.pivot_table(index="age", columns="party", values="total_impressions", aggfunc="sum")
sns.heatmap(pivot, annot=True, fmt=".0f")
plt.xlabel("Political party")
plt.ylabel("Age group")
plt.title("Figure 6: Impressions per Demographic")
plt.show()
```

Figure 7 visualisation

```
words_df = pd.read_csv("DATA7201/labor_words.csv")

words_df = words_df.dropna(subset=['word', 'count'])

word_freq = dict(zip(words_df["word"], words_df["count"]))

wordcloud = WordCloud(width=800, height=500,
background_color='white').generate_from_frequencies(word_freq)
wordcloud.to_image()
```

Figure 8 visualisation

```
liberal_words_df = pd.read_csv("DATA7201/liberal_words.csv")

liberal_words_df = liberal_words_df.dropna(subset=['word', 'count'])
```

```
word_freq = dict(zip(liberal_words_df["word"], liberal_words_df["count"]))

wordcloud = WordCloud(width=800, height=500,
background_color='white').generate_from_frequencies(word_freq)
wordcloud.to_image()
```

Figure 9 sentiment analysis

```
text_df = pd.read_csv("DATA7201/text_data.csv")

import re

def clean_text(text):
    # remove punctuation
    text = re.sub(r"[^\w\s]", "", text)
    text = text.lower().strip()
    return text

# drop null values
text_df = text_df.dropna()

text_df = text_df.reset_index(drop=True)

# clean the text data
text_df["cleaned"] = text_df["ad_creative_body"].apply(clean_text)

from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download("vader_lexicon")

sia = SentimentIntensityAnalyzer()

def get_sentiment_scores(text):
    return sia.polarity_scores(text)

sentiment_scores = text_df["cleaned"].apply(get_sentiment_scores)
sentiment_df = pd.json_normalize(sentiment_scores)

def classify_sentiment(score):
    if score >= 0.05:
        return "positive"
    elif score <= -0.05:
        return "negative"
    else:
        return "neutral"

sentiment_df["category"] = sentiment_df["compound"].apply(classify_sentiment)

text_df = pd.concat([text_df, sentiment_df], axis=1)

text_df = text_df.drop("Unnamed: 0", axis=1)
```

Figure 9 visualisation

```
test_df = text_df.drop_duplicates()

sns.boxplot(data=test_df, x="funding_entity", y="compound")
plt.xlabel("Party")
plt.ylabel("Sentiment score")
plt.title("Figure 10: Sentiment scores")
plt.show()
```

Spend data visualisation

```
spend = pd.read_csv("DATA7201/spend.csv")

plt.figure(figsize=(12,6))
sns.lineplot(data=spend, x="month", y="average", hue="funding_entity")
plt.xticks(rotation=45)
plt.show()
```

Regional distribution visualisation

```
regions_df = regions_df.sort_values(["region", "normalised_percentage"], ascending=[True, False])

def get_dominant(group):
    parties = group["funding_entity"].tolist()
    percentages = group["normalised_percentage"].tolist()

    dominant_party = parties[0]

    if dominant_party == "Australian Labor Party":
        margin = percentages[0] - percentages[1]
    else:
        margin = percentages[1] - percentages[0]

    return pd.Series({
        "dominant_party": dominant_party,
        "dominance_margin": margin
    })

dominant_df = regions_df.groupby("region").apply(get_dominant).reset_index()

import geopandas as gpd

gdf = gpd.read_file("DATA7201/map/gadm41_AUS_1.shp")

import matplotlib.colors as mcolors

australia_states_gdf = gdf.merge(dominant_df, how="left", left_on="NAME_1", right_on="region")
# drop smaller regions
australia_states_gdf = australia_states_gdf.dropna()

# set norm for map
max_abs_margin = max(abs(australia_states_gdf['dominance_margin'].min()),
                     abs(australia_states_gdf['dominance_margin'].max()))
norm = mcolors.TwoSlopeNorm(vmin=-max_abs_margin, vcenter=0, vmax=max_abs_margin)

# set cmap for map
cmap = plt.cm.bwr

australia_states_gdf.plot(column="dominance_margin",
                          cmap=cmap,
                          norm=norm,
                          legend=True,
                          legend_kwds={'label': "Liberal + / Labor -", 'shrink': 0.6},
                          linewidth=0.25,
                          edgecolor='black')

plt.xlim(110, 155)
plt.ylim(-45, -10)
plt.axis("off")
plt.title("Figue 7: Labor vs Liberal reach")
plt.show()
```