SOCIAL MEDIA HANDS-ON SESSION

By Kathakoli Sengupta

1

Today's Plan

- Social Media Survey Data Analysis
- Social Media Data Sentiment Analysis

2

Social Media Survey Data Analysis

Social Media Data can be collected in form of Survey.

- Frequently asked Question:
  - Do you ever use [platform]? 1:Yes, 2:No, 99:No Answer
  - How often do you get news from [source]? 1:Often, 2:Sometimes, 3:Rarely, 4:Never
  - Age, Gender, Education are some participant details collected for analysis.
- Survey can be represented in .sav format.
  - It is the native data format for SPSS (Statistical Package for the Social Sciences), IBM's commercial statistics software.
  - It is like an Excel file but designed specifically for survey data.
  - It stores not just the numbers but also variable labels (what each column means), value labels (what each coded number represents, like 1 = Male, 2 = Female), and missing value definitions.
- PEW Datasets have survey data: https://www.pewresearch.org/dataset/

3

Now let's explore Social Media Survey Data

- Registration Instructions:
  - Go to https://www.pewresearch.org/profile/registration/
  - Fill out Name, Email, Organization(use your university), Job Title(student), Click Submit.
  - Log in at https://www.pewresearch.org/profile/ with the credentials created.
  - Search Wave150 (Direct Link: https://www.pewresearch.org/dataset/american-trends-panel-wave-150/).
- Download Wave 150 Dataset.

4

Let's start Coding

- Steps of Social Media Survey Data Analysis
  - Loading and exploring the data.
  - Cleaning and Encoding the data.
  - Analysis
  - Plotting (for better understanding)
- Before we start
  - Take 5 mins and read the Readme.txt file given with your dataset for brief understanding of the dataset.
- Now, I will show you few analysis code and explain how I am using the data.
- Download the python file from https://drive.google.com/file/d/1_fpULYlSsXgcekKcs-k0bzm_oWnxS8F0/view?usp=sharing and copy paste exactly the sections I am showing and run with me.
- Then, I will ask you to code some analysis from the same dataset.

5

Loading and Exploring Data

- Importing Data

```
import os
print(os.getcwd())
os.chdir('')
SAV_FILE = "ATP W150.sav"
df, meta = pyreadstat.read_sav(SAV_FILE)
```

- Shape and columns

```
print(f"Dataset shape: {df.shape[0]} rows × {df.shape[1]} columns")
print(f"\n5column names:\n{list(df.columns[:5])}")
```

f "{}": Whatever is present inside {}, that value will be printed instead of the string.

6

Loading and Exploring Data

- Metadata
  - # meta.column_names → list of all variable names
  - # meta.column_labels → dict: variable name → description
  - # meta.variable_value_labels → dict: variable name → {code: label}

- Exploring columns

```
print(f"\nVariable label for SMUSE_a:")
idx = meta.column_names.index("SMUSE_a_W150")
print(meta.column_labels[idx])
print(f"\nValue labels for SMUSE_a:")
for code, label in meta.variable_value_labels['SMUSE_a_W150'].items():
   print(f"  {code} = {label}")
```

7

Data Details(Revealed from previous analysis)

```
Pew ATP datasets have TWO types of columns:

A. SURVEY QUESTIONS (specific to each wave):
   These are the questions asked in this particular survey.
   For Wave 150 (social media), the key variables are:

   SMUSE_a through SMUSE_n — "Do you ever use [platform]?"
     SMUSE_a = Facebook          1=Yes, 2=No, 99=No answer
     SMUSE_b = YouTube
     SMUSE_c = X (Twitter)
     SMUSE_d = Instagram
     SMUSE_e = Snapchat
     SMUSE_f = WhatsApp
     SMUSE_g = LinkedIn
     SMUSE_i = TikTok            (note: no _h)
     SMUSE_j = Reddit
     SMUSE_l = Nextdoor          (note: no _k)
     SMUSE_m = Rumble
     SMUSE_n = Truth Social
```

8

```
 SOCIALNEWS2_a through _n — "Do you
regularly get NEWS on [platform]?"
     Same platform mapping as SMUSE
     1=Yes regularly get news, 2=No

 NEWSPLAT_a through _d — "How often do
you get news from [source]?"
     NEWSPLAT_a = Television
     NEWSPLAT_b = Radio
     NEWSPLAT_c = Print
     NEWSPLAT_d = Digital device
     Values: 1=Often, 2=Sometimes,
3=Rarely, 4=Never
```

```
B. DEMOGRAPHIC FRAME VARIABLES (same across
all waves):
   These start with "F_" and contain
demographics:

     F_AGECAT      → Age group (1=18-29,
2=30-49, 3=50-64, 4=65+)
     F_GENDER      → Gender (1=Man, 2=Woman,
3=Other)
     F_EDUCCAT     → Education (1=College+,
2=Some college, 3=HS or less)
     F_RACETHNMOD  → Race/ethnicity
     F_PARTY_FINAL → Party ID (1=Republican,
2=Democrat, 3=Independent, 4=Other)
     F_PARTYSUM_FINAL → Party with leaners
(1=Rep/Lean Rep, 2=Dem/Lean Dem)
     F_INC_SDT1    → Family income bracket
     F_METRO       → Metro area
(1=Metropolitan, 2=Non-metropolitan)
     F_CREGION     → Census region (1=NE,
2=MW, 3=South, 4=West)
   WEIGHT        → Survey weight (ALWAYS USE
THIS for representative results)
```

9

Cleaning and Encoding Data

- Encoding platform usage into useful labels
  - Recode platform usage: 1=uses, 0=does not use
  - In the raw data: 1=Yes, 2=No, 99=No answer
  - We recode to binary: 1=Yes, 0=No (treating 99 as No)

```
platform_map = {
   "SMUSE_a_W150": "Facebook",
   "SMUSE_b_W150": "YouTube",
   "SMUSE_c_W150": "X (Twitter)",
   "SMUSE_d_W150": "Instagram",
   "SMUSE_e_W150": "Snapchat",
   "SMUSE_f_W150": "WhatsApp",
   "SMUSE_g_W150": "LinkedIn",
   "SMUSE_i_W150": "TikTok",
   "SMUSE_j_W150": "Reddit",
   "SMUSE_l_W150": "Nextdoor",
   "SMUSE_m_W150": "Rumble",
   "SMUSE_n_W150": "Truth Social",
}
```

```
for var, name in platform_map.items():
   if var in df.columns:
      # 1=Yes → 1, everything else → 0
      df[f"uses_{name}"] = (df[var] == 1).astype(int)
   else:
      print(f"  Warning: {var} not found in dataset (may be a
different wave)")
```

10

Cleaning and Encoding Data

- Encoding news consumption into useful labels

```
news_map = {
    "SOCIALNEWS2_a_W150": "Facebook",
    "SOCIALNEWS2_b_W150": "YouTube",
    "SOCIALNEWS2_c_W150": "X (Twitter)",
    "SOCIALNEWS2_d_W150": "Instagram",
    "SOCIALNEWS2_e_W150": "Snapchat",
    "SOCIALNEWS2_f_W150": "WhatsApp",
    "SOCIALNEWS2_g_W150": "LinkedIn",
    "SOCIALNEWS2_i_W150": "TikTok",
    "SOCIALNEWS2_j_W150": "Reddit",
    "SOCIALNEWS2_l_W150": "Nextdoor",
    "SOCIALNEWS2_m_W150": "Rumble",
    "SOCIALNEWS2_n_W150": "Truth Social",
}
```

```
for var, name in news_map.items():
    if var in df.columns:
        df[f"news_{name}"] = (df[var] == 1).astype(int)
```

11

Cleaning and Encoding Data

- Recode demographics into readable labels

```
age_labels = {1: "18-29", 2: "30-49", 3: "50-64", 4: "65+"}
df["age_group"] = df["F_AGECAT"].map(age_labels)

gender_labels = {1: "Man", 2: "Woman", 3: "In some other way"}
df["gender"] = df["F_GENDER"].map(gender_labels)

edu_labels = {1: "College graduate+", 2: "Some College", 3: "H.S. graduate or less"}
df["education"] = df["F_EDUCCAT"].map(edu_labels)

party_labels = {1: "Rep/Lean Rep", 2: "Dem/Lean Dem"}
if "F_PARTYSUM_FINAL" in df.columns:
    df["party"] = df["F_PARTYSUM_FINAL"].map(party_labels)

metro_labels = {1: "Metropolitan", 2: "Non-metropolitan"}
if "F_METRO" in df.columns:
    df["metro"] = df["F_METRO"].map(metro_labels)

print("\nRecoding complete")
print(f"Usable rows: {df['age_group'].notna().sum()}")
```

12

Platform Adoption Rates (Analysis 1)

- Platform Adoption Rates gives how widely each social media platform is used in your survey population i.e., the percentage of people who use each platform.

```
print("Platform Adoption Rates (Weighted)")
platforms_available = [name for var, name in
platform_map.items() if var in df.columns]

weight_col = "WEIGHT_W150"

if weight_col is not None:
    print(f"\nUsing survey weights: {weight_col}\n")

    for name in platforms_available:
        col = f"uses_{name}"
        valid = df[col].notna() & df[weight_col].notna()

        weighted_pct = np.average(df.loc[valid, col],
weights=df.loc[valid, weight_col]) * 100
        unweighted_pct = df.loc[valid, col].mean() * 100
        print(f"  {name:15s}  Weighted:
{weighted_pct:.1f}%   Unweighted: {unweighted_pct:.1f}%")
else:
    print("\nNo suitable weight column found.")
    print("Found:", [c for c in df.columns if "WEIGHT" in
c.upper()])
    print("Using unweighted means for now.\n")

    for name in platforms_available:
        col = f"uses_{name}"
        pct = df[col].mean() * 100
        print(f"  {name:15s}  {pct:.1f}%")
```

13

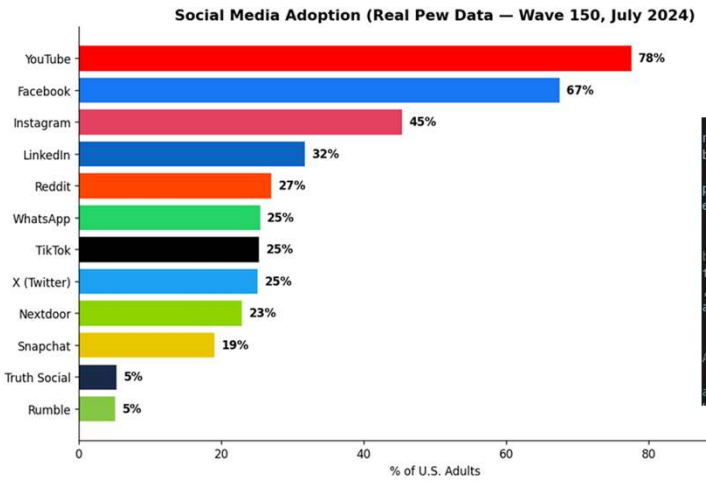
- Plotting

```
print("Generating Adoption Chart")

plt.rcParams.update({
    "figure.dpi": 120,
    "axes.spines.top": False,
    "axes.spines.right": False
})

COLORS = {
    "YouTube": "#FF0000", "Facebook": "#1877F2",
"Instagram": "#E4405F",
    "TikTok": "#010101", "Snapchat": "#E8C700", "X
(Twitter)": "#1DA1F2",
    "LinkedIn": "#0A66C2", "Reddit": "#FF4500",
"WhatsApp": "#25D366",
    "Pinterest": "#E60023", "Nextdoor": "#8ED500",
"Rumble": "#85C742",
    "Truth Social": "#1A2B4A",
}
```

```
# Map column → platform name
col_to_name = {f"uses_{name}": name for name in
platforms_available}
use_cols = [f"uses_{name}" for name in
platforms_available]

# Compute adoption
adoption = df[use_cols].mean() * 100
adoption = adoption.rename(index=col_to_name)
adoption = adoption.sort_values(ascending=True)
```

14

```
# Plot
fig, ax = plt.subplots(figsize=(10, 6))

bars = ax.barh(
    adoption.index,
    adoption.values,
    color=[COLORS.get(p, "#888") for p in adoption.index],
    edgecolor="white",
    linewidth=0.5
)

for bar, val in zip(bars, adoption.values):
    ax.text(bar.get_width() + 1,
            bar.get_y() + bar.get_height() / 2,
            f"{val:.0f}%",
            va="center",
            fontweight="bold",
            fontsize=10)

ax.set_xlabel("% of U.S. Adults")
ax.set_title("Social Media Adoption (Real Pew Data — Wave 150, July 2024)",
             fontweight="bold", fontsize=13)
ax.set_xlim(0, 100)

plt.tight_layout()
plt.savefig("real_pew_chart1_adoption.png", bbox_inches="tight")
plt.close()

print("Saved: real_pew_chart1_adoption.png")
```

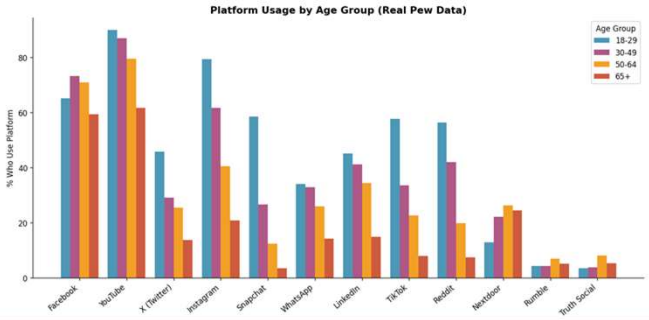




```
Using survey weights: WEIGHT_W150

  Facebook          Weighted: 65.6%   Unweighted: 67.5%
  YouTube           Weighted: 77.6%   Unweighted: 77.6%
  X (Twitter)       Weighted: 25.9%   Unweighted: 25.1%
  Instagram         Weighted: 46.8%   Unweighted: 45.4%
  Snapchat          Weighted: 23.7%   Unweighted: 19.0%
  WhatsApp          Weighted: 25.4%   Unweighted: 25.4%
  LinkedIn          Weighted: 27.5%   Unweighted: 31.7%
  TikTok            Weighted: 29.5%   Unweighted: 25.3%
  Reddit            Weighted: 28.2%   Unweighted: 27.1%
  Nextdoor          Weighted: 19.1%   Unweighted: 22.9%
  Rumble            Weighted: 5.3%   Unweighted: 5.1%
  Truth Social      Weighted: 5.5%   Unweighted: 5.3%
```

Platform Usage by Age Group (Analysis 2)

- Which age groups use which platforms, and how much?
- Refer to the python code for plotting.

```
use_cols = [f"uses_{name}" for name in
platforms_available]
age_table =
df.groupby("age_group")[use_cols].mea
n() * 100

age_table.columns =
platforms_available
print("\n",
age_table.round(1).to_string())
```




```
Platform Usage by Age Group

           Facebook  YouTube  X (Twitter)  Instagram  Snapchat  WhatsApp  LinkedIn  TikTok  Reddit  Nextdoor  Rumble  Truth Social
age_group
18-29          65.2     89.9         45.8       79.3      58.5      34.1      45.1    57.6    56.4      12.9     4.2           3.4
30-49          73.3     86.9         29.0       61.7      26.6      32.9      41.2    33.5    42.0      22.1     4.2           3.7
50-64          71.0     79.5         25.5       40.4      12.3      25.9      34.4    22.6    19.8      26.3     6.8           8.0
65+            59.3     61.6         13.7       20.8       3.4      14.1      14.9     7.8     7.3      24.5     5.0           5.3
```

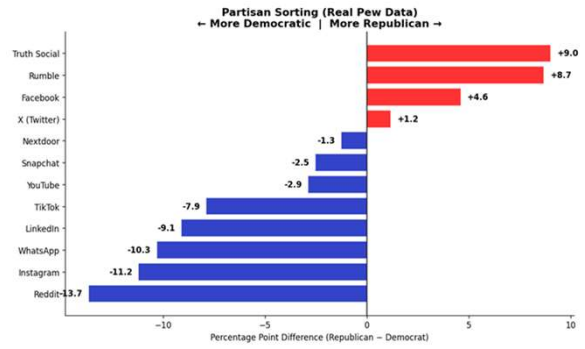
17

Partisan Platform Sorting(Analysis 3)

- How social media platform usage differs by political party?
- Refer to the python code for plotting.

```
if "party" in df.columns:
    party_table = df.groupby("party")[use_cols].mean() * 100
    party_table.columns = platforms_available

    if "Rep/Lean Rep" in party_table.index and "Dem/Lean Dem" in party_table.index:
        gap = party_table.loc["Rep/Lean Rep"] - party_table.loc["Dem/Lean Dem"]
        print("\nPartisan gap (Rep − Dem):")
        for name, g in gap.sort_values().items():
            lean = "R-leaning" if g > 0 else "D-leaning"
            print(f"  {name:15s}  {g:+.1f} pp  ({lean})")
```

18




```
Partisan gap (Republican - Democrat) in usage:
  Reddit           -13.7 pp  (D-leaning)
  Instagram        -11.2 pp  (D-leaning)
  WhatsApp         -10.3 pp  (D-leaning)
  LinkedIn         -9.1 pp  (D-leaning)
  TikTok           -7.9 pp  (D-leaning)
  YouTube          -2.9 pp  (D-leaning)
  Snapchat         -2.5 pp  (D-leaning)
  Nextdoor         -1.3 pp  (D-leaning)
  X (Twitter)      +1.2 pp  (R-leaning)
  Facebook         +4.6 pp  (R-leaning)
  Rumble           +8.7 pp  (R-leaning)
  Truth Social     +9.0 pp  (R-leaning)
```

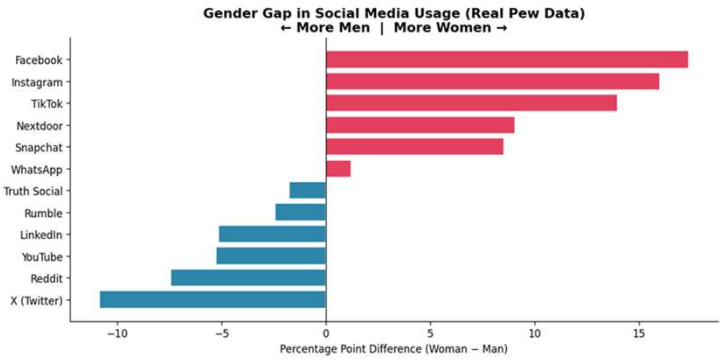
19

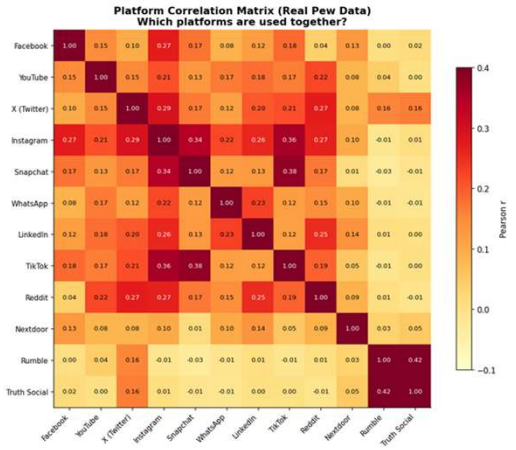
News Consumption Among Platform Users (Analysis 4) – Try this Yourself

- Among people who use a platform, how many actually use it to get news?
- Try to figure out some reasonable plot to show this analysis.

20

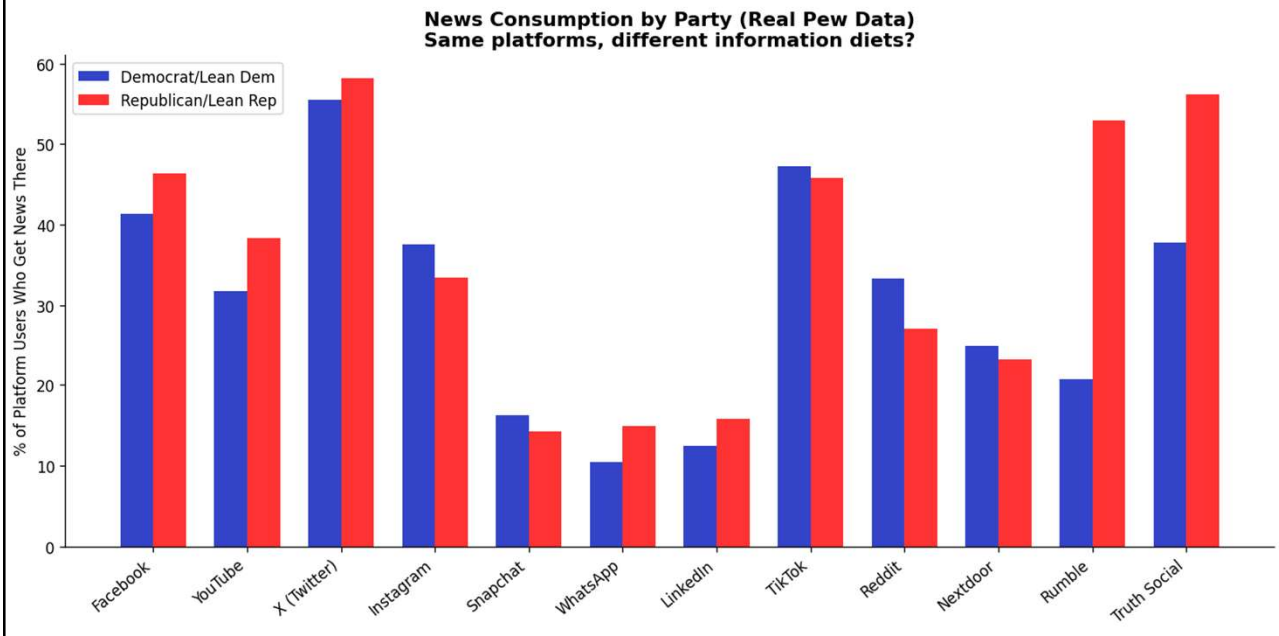
Other Interesting Analysis – Try at Home





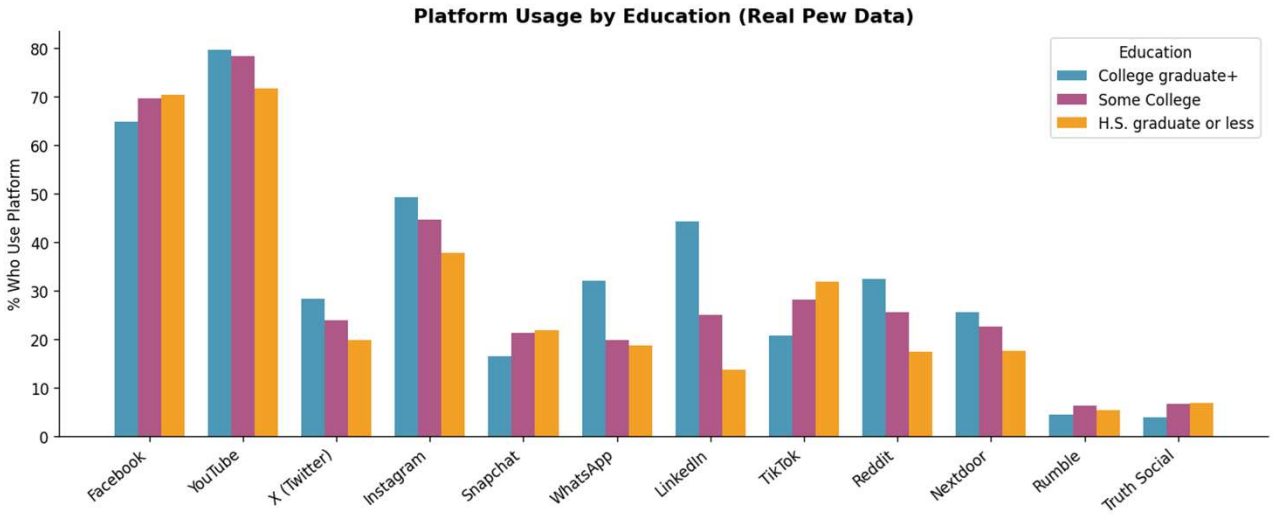
For each platform, what is the percentage point difference in usage between women and men?
Which platforms are usually used together? Hint: Correlation

21



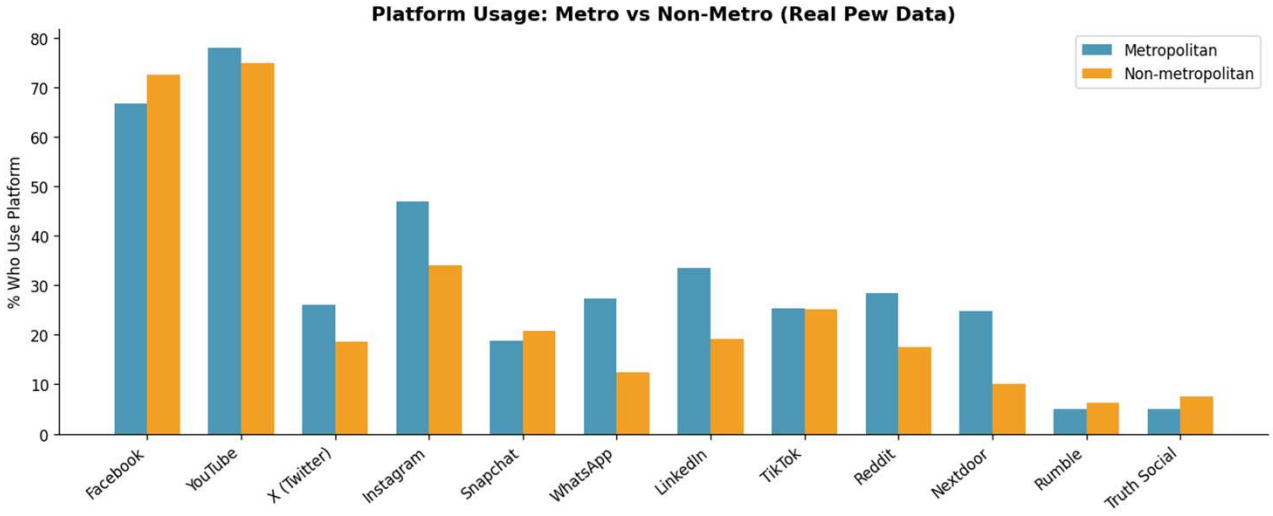
Among users of the same platform, do Democrats and Republicans differ in how much they use it for news?

22



Do people with different education levels use different social media platforms, and by how much?

23



Do people in metropolitan areas and non-metropolitan areas use different social media platforms, and by how much?

24

Sentiment Analysis on Social Media Data

- What is Sentiment Analysis?

It is simply the process of working out (statistically) whether a piece of text is positive, negative or neutral.

- Python Package Used:

https://github.com/cjhutto/vaderSentiment

25

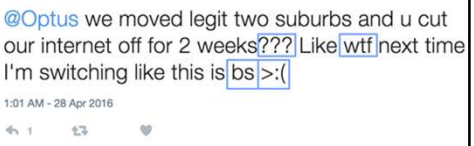
- VADER(Valence Aware Dictionary and sEntiment Reasoner) belongs to a type of sentiment analysis that is based on lexicons of sentiment-related words.
- Each of the words in the lexicon is rated as to whether it is positive or negative, and in many cases, *how* positive or negative.
- VADER analyses a piece of text to see if any of the words in the text are present in the lexicon. For example, the sentence "The food is good and the atmosphere is nice" has two words in the lexicon (good and nice) with ratings of 1.9 and 1.8 respectively.
- VADER produces four sentiment metrics from these word ratings.

Word	Sentiment rating
tragedy	-3.4
rejoiced	2.0
insane	-1.7
disaster	-3.1
great	3.1

26

- The first three, positive, neutral and negative, represent the proportion of the text that falls into those categories. The example sentence was rated as 45% positive, 55% neutral and 0% negative.
- The final metric, the compound score, is the sum of all of the lexicon ratings (1.9 and 1.8 in this case) which have been standardised to range between -1 and 1. In this case, our example sentence has a rating of 0.69, which is pretty strongly positive.
- Vader can handle informal writing - multiple punctuation marks, acronyms and an emoticon by including these sorts of terms in its lexicon.

Sentiment metric	Value
Positive	0.45
Neutral	0.55
Negative	0.00
Compound	0.69



27

How to use Vader?

```
!pip install vaderSentiment

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyser = SentimentIntensityAnalyzer()

def print_sentiment_scores(sentence):
    snt = analyser.polarity_scores(sentence)
    print("{:-<40} {}".format(sentence, str(snt)))
```

Link: https://drive.google.com/file/d/1WtmFodXIOLuDpf0O3om9aEA6VsiwRvb0/view?usp=sharing

28

Examples

print_sentiment_scores("I just got a call from my boss - does he realise it's Saturday?")

I just got a call from my boss - does he realise it's Saturday? {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}

print_sentiment_scores("I just got a call from my boss - does he realise it's Saturday? :(")

I just got a call from my boss - does he realise it's Saturday? :({'neg': 0.172, 'neu': 0.828, 'pos': 0.0, 'compound': -0.4404} (Negative score from emoji)

print_sentiment_scores("I just got a call from my boss - does he realise it's Saturday? smh :(")

I just got a call from my boss - does he realise it's Saturday? smh :({'neg': 0.271, 'neu': 0.729, 'pos': 0.0, 'compound': -0.6369} (Negative score increases from smh)

print_sentiment_scores("The food is good.")

The food is good.---------- {'neg': 0.0, 'neu': 0.508, 'pos': 0.492, 'compound': 0.4404}

29

print_sentiment_scores("The food is GOOD.")

The food is GOOD.---------- {'neg': 0.0, 'neu': 0.452, 'pos': 0.548, 'compound': 0.5622} (Positive score increases from capital letters.)

print_sentiment_scores("The food is GOOD!")

The food is GOOD!---------- {'neg': 0.0, 'neu': 0.433, 'pos': 0.567, 'compound': 0.6027} (Positive score increases from !.)

print_sentiment_scores("The food is really GOOD!")

The food is really GOOD!---------- {'neg': 0.0, 'neu': 0.487, 'pos': 0.513, 'compound': 0.6391} (Compound score increases from “really”.)

print_sentiment_scores("The food is really GOOD! But the service is dreadful.")

The food is really GOOD! But the service is dreadful. {'neg': 0.284, 'neu': 0.548, 'pos': 0.169, 'compound': -0.3977} (Negative score increases from “dreadful”.)

30