CaseCraft: The Analytics Sprint – Project 16

Influencer Impact on Brand Perception

Subheading: Mapping influencer sentiment and audience engagement to brand perception using Flair NLP and Sankey flows.

Project Goals

- Simulate influencer posts and engagement metrics
- Apply Flair sentiment tagging for nuanced tone detection
- Visualize influencer-brand flow using Sankey diagrams
- Use treemaps to show audience engagement clusters
- Predict brand perception score from influencer mix
- Summarize insights for influencer strategy optimization

```
!pip install flair
```

```
Collecting flair
  Downloading flair-0.15.1-py3-none-any.whl.metadata (12 kB)
Collecting boto3>=1.20.27 (from flair)
  Downloading boto3-1.40.16-py3-none-any.whl.metadata (6.7 kB)
Collecting conllu<5.0.0,>=4.0 (from flair)
  Downloading conllu-4.5.3-py2.py3-none-any.whl.metadata (19 kB)
Collecting deprecated>=1.2.13 (from flair)
  Downloading Deprecated-1.2.18-py2.py3-none-any.whl.metadata (5.7 kB)
Collecting ftfy>=6.1.0 (from flair)
  Downloading ftfy-6.3.1-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: gdown>=4.4.0 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: huggingface-hub>=0.10.0 in /usr/local/lib/pythor
Collecting langdetect>=1.0.9 (from flair)
  Downloading langdetect-1.0.9.tar.gz (981 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 981.5/981.5 kB 43.6 MB/s eta
  Preparing metadata (setup.py) ... done
Requirement already satisfied: lxml>=4.8.0 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: matplotlib>=2.2.3 in /usr/local/lib/python3.12/d
Requirement already satisfied: more-itertools>=8.13.0 in /usr/local/lib/python3
Collecting mpld3>=0.3 (from flair)
  Downloading mpld3-0.5.11-py3-none-any.whl.metadata (5.3 kB)
Collecting pptree>=3.1 (from flair)
  Downloading pptree-3.1.tar.gz (3.0 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3
Collecting pytorch-revgrad>=0.2.0 (from flair)
  Downloading pytorch_revgrad-0.2.0-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.12/di
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.12
Collecting segtok>=1.5.11 (from flair)
```

```
      Downloading segtok-1.5.11-py3-none-any.whl.metadata (9.0 kB)
    Collecting sqlitedict>=2.0.0 (from flair)
      Downloading sqlitedict-2.1.0.tar.gz (21 kB)
      Preparing metadata (setup.py) ... done
    Requirement already satisfied: tabulate>=0.8.10 in /usr/local/lib/python3.12/di
    Requirement already satisfied: torch>=1.13.1 in /usr/local/lib/python3.12/dist-
    Requirement already satisfied: tqdm>=4.63.0 in /usr/local/lib/python3.12/dist-p
    Collecting transformer-smaller-training-vocab>=0.2.3 (from flair)
      Downloading transformer_smaller_training_vocab-0.4.2-py3-none-any.whl.metadat
    Requirement already satisfied: transformers<5.0.0,>=4.25.0 in /usr/local/lib/py
    Collecting wikipedia-api>=0.5.7 (from flair)
      Downloading wikipedia_api-0.8.1.tar.gz (19 kB)
      Preparing metadata (setup.py) ... done
    Collecting bioc<3.0.0,>=2.0.0 (from flair)
      Downloading bioc-2.1-py3-none-any.whl.metadata (4.6 kB)
    Collecting jsonlines>=1.2.0 (from bioc<3.0.0,>=2.0.0->flair)
      Downloading jsonlines-4.0.0-py3-none-any.whl.metadata (1.6 kB)
    Collecting intervaltree (from bioc<3.0.0,>=2.0.0->flair)
      Downloading intervaltree-3.1.0.tar.gz (32 kB)
      Preparing metadata (setup.py) ... done
    Collecting docopt (from bioc<3.0.0,>=2.0.0->flair)
      Downloading docopt-0.6.2.tar.gz (25 kB)
      Preparing metadata (setup.py) ... done
    Collecting botocore<1.41.0,>=1.40.16 (from boto3>=1.20.27->flair)
      Downloading botocore-1.40.16-py3-none-any.whl.metadata (5.7 kB)
    Collecting jmespath<2.0.0,>=0.7.1 (from boto3>=1.20.27->flair)
      Downloading jmespath-1.0.1-py3-none-any.whl.metadata (7.6 kB)
    Collecting s3transfer<0.14.0,>=0.13.0 (from boto3>=1.20.27->flair)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from flair.models import TextClassifier
from flair.data import Sentence

np.random.seed(42)

influencers = ['@stylequeen', '@techguru', '@fitlife', '@foodie', '@traveljunkie']
brands = ['Zara', 'Apple', 'Nike', 'Starbucks', 'Airbnb']
n_posts = 500

df = pd.DataFrame({
    'influencer': np.random.choice(influencers, n_posts),
    'brand': np.random.choice(brands, n_posts),
    'post': np.random.choice([
        "Absolutely love this product!", "Not impressed at all.", "Decent experience.",
        "Highly recommend!", "Wouldn't buy again.", "Great value for money."
    ], n_posts),
    'likes': np.random.randint(100, 5000, n_posts),
    'comments': np.random.randint(10, 500, n_posts)
})

classifier = TextClassifier.load('en-sentiment')

def flair_sentiment(text):
    sentence = Sentence(text)
    classifier.predict(sentence)
```

```
    return sentence.labels[0].value

df['sentiment'] = df['post'].apply(flair_sentiment)
df['engagement'] = df['likes'] + df['comments']
```

```
2025-08-24 14:46:07,108 https://nlp.informatik.hu-berlin.de/resources/models/sen
100%|██████████| 253M/253M [00:11<00:00, 23.6MB/s]2025-08-24 14:46:18,702 copy

2025-08-24 14:46:19,140 removing temp file /tmp/tmpkk_ehdm7
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 2.84kB/s]
config.json: 100%  483/483 [00:00<00:00, 43.4kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 5.70MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 29.4MB/s]
```

```
df.head(10)
```

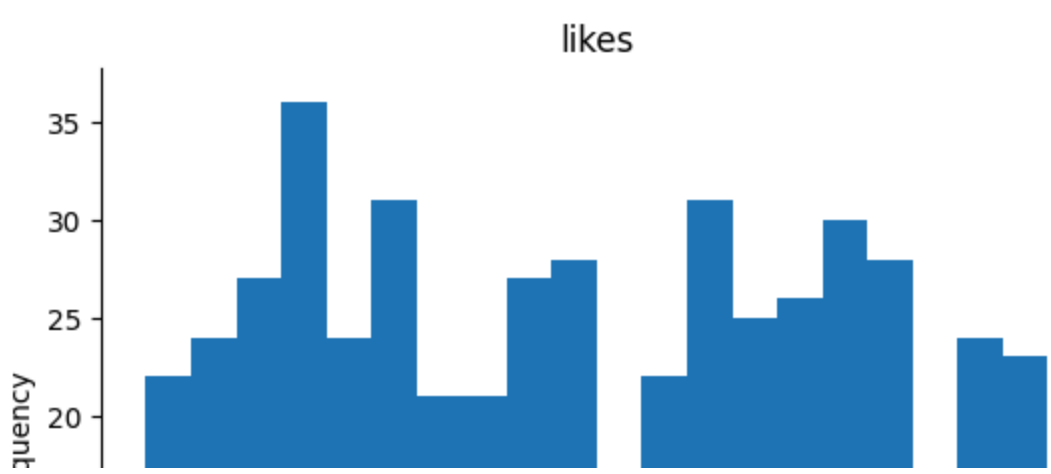
	influencer	brand	post	likes	comments	sentiment	engagement
0	@foodie	Starbucks	Highly recommend!	4235	333	POSITIVE	4568
1	@traveljunkie	Zara	Highly recommend!	1660	286	POSITIVE	1946
2	@fitlife	Airbnb	Highly recommend!	1518	150	POSITIVE	1668
3	@traveljunkie	Nike	Highly recommend!	1054	479	POSITIVE	1533
4	@traveljunkie	Nike	Wouldn't buy again.	1083	284	NEGATIVE	1367
5	@techguru	Zara	Wouldn't buy again.	1074	299	NEGATIVE	1373
6	@fitlife	Starbucks	Not impressed at all.	4233	343	NEGATIVE	4576
7	@fitlife	Starbucks	Absolutely love this product!	4072	390	POSITIVE	4462
8	@fitlife	Airbnb	Highly recommend!	1013	461	POSITIVE	1474
9	@traveljunkie	Zara	Not impressed at all.	4649	382	NEGATIVE	5031

Influencer vs brand

```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['brand'].value_counts()
    for x_label, grp in df.groupby('influencer')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('influencer')
_ = plt.ylabel('brand')
```

Frequency of likes

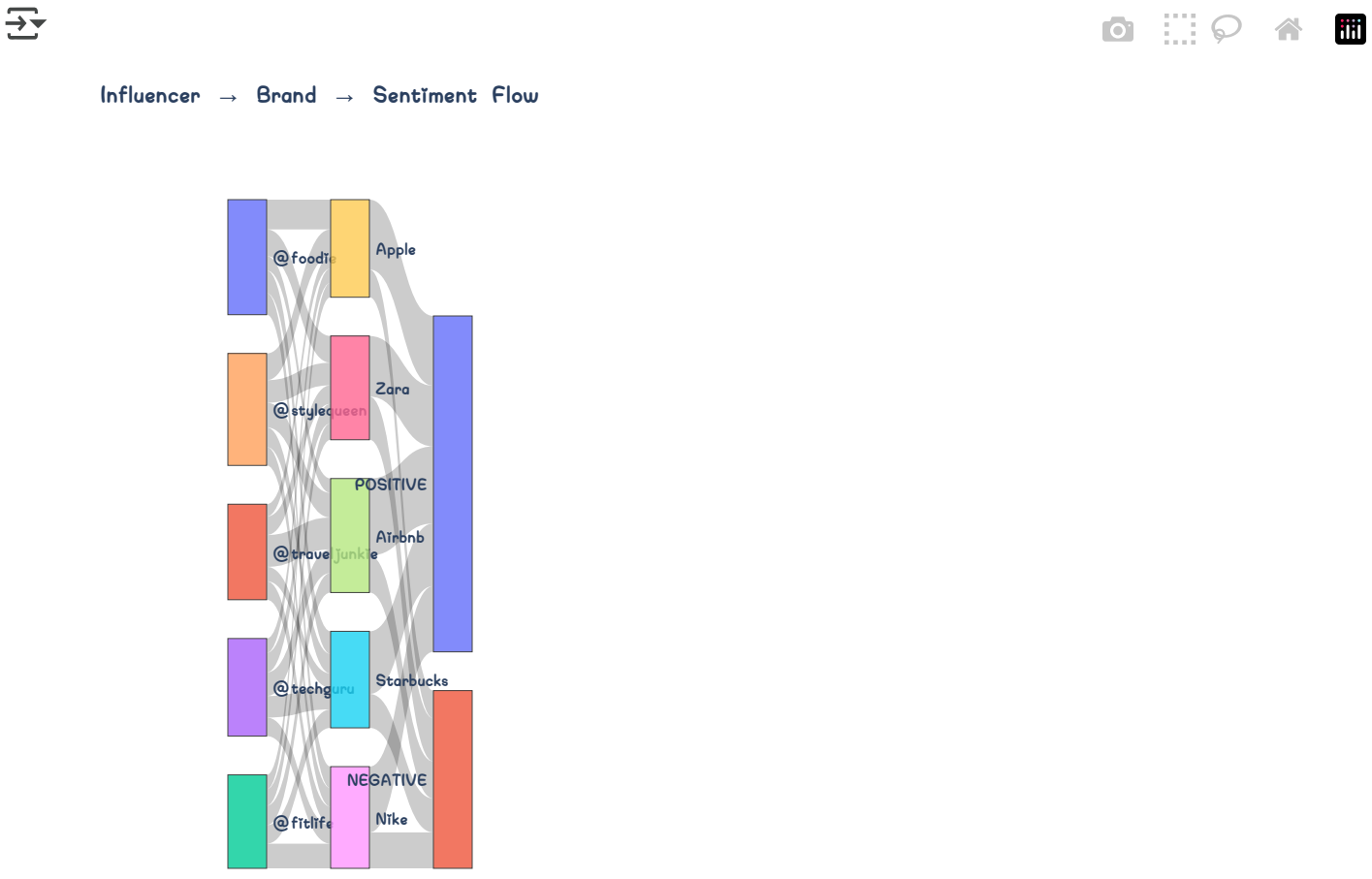
```
from matplotlib import pyplot as plt
df['likes'].plot(kind='hist', bins=20, title='likes')
plt.gca().spines[['top', 'right',]].set_visible(False)
```



Sankey Diagram: Influencer → Brand → Sentiment Flow

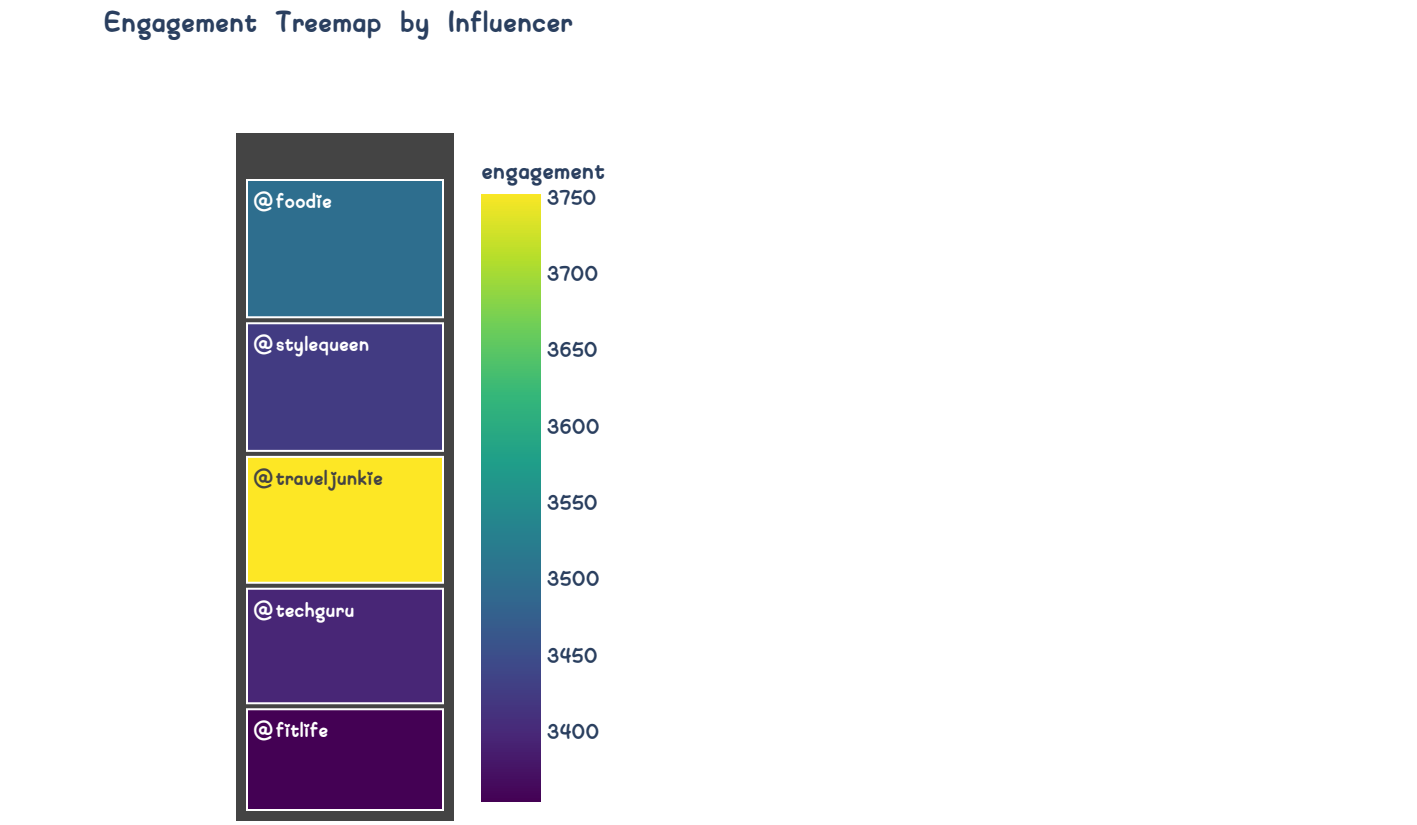
```
import plotly.graph_objects as go

source_labels = df['influencer'].unique().tolist()
target_labels = df['brand'].unique().tolist()
sentiment_labels = ['POSITIVE', 'NEGATIVE']

all_labels = source_labels + target_labels + sentiment_labels

source_indices = []
target_indices = []
```

```
values = []

for influencer in source_labels:
    for brand in target_labels:
        count = len(df[(df['influencer'] == influencer) & (df['brand'] == brand)])
        if count > 0:
            source_indices.append(all_labels.index(influencer))
            target_indices.append(all_labels.index(brand))
            values.append(count)

for brand in target_labels:
    for sentiment in sentiment_labels:
        count = len(df[(df['brand'] == brand) & (df['sentiment'] == sentiment)])
        if count > 0:
            source_indices.append(all_labels.index(brand))
            target_indices.append(all_labels.index(sentiment))
            values.append(count)

fig = go.Figure(data=[go.Sankey(
    node=dict(label=all_labels),
    link=dict(source=source_indices, target=target_indices, value=values)
)])
fig.update_layout(title_text="Influencer → Brand → Sentiment Flow", font_size=10)
fig.show()
```



Treemap: Engagement Clusters by Influencer

```
import plotly.express as px

fig = px.treemap(df, path=['influencer'], values='engagement', color='engagement',
                 color_continuous_scale='Viridis', title='Engagement Treemap by Influencer'
fig.show()
```



Brand Perception Score Modeling

- Define brand perception as weighted sentiment × engagement
- Train regression model to predict perception score

```
df['sentiment_score'] = df['sentiment'].map({'POSITIVE': 1, 'NEGATIVE': -1})
df['perception_score'] = df['sentiment_score'] * df['engagement']

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
X = pd.get_dummies(df[['influencer', 'brand']])
y = df['perception_score']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
score = model.score(X_test, y_test)
```

Summary Analysis