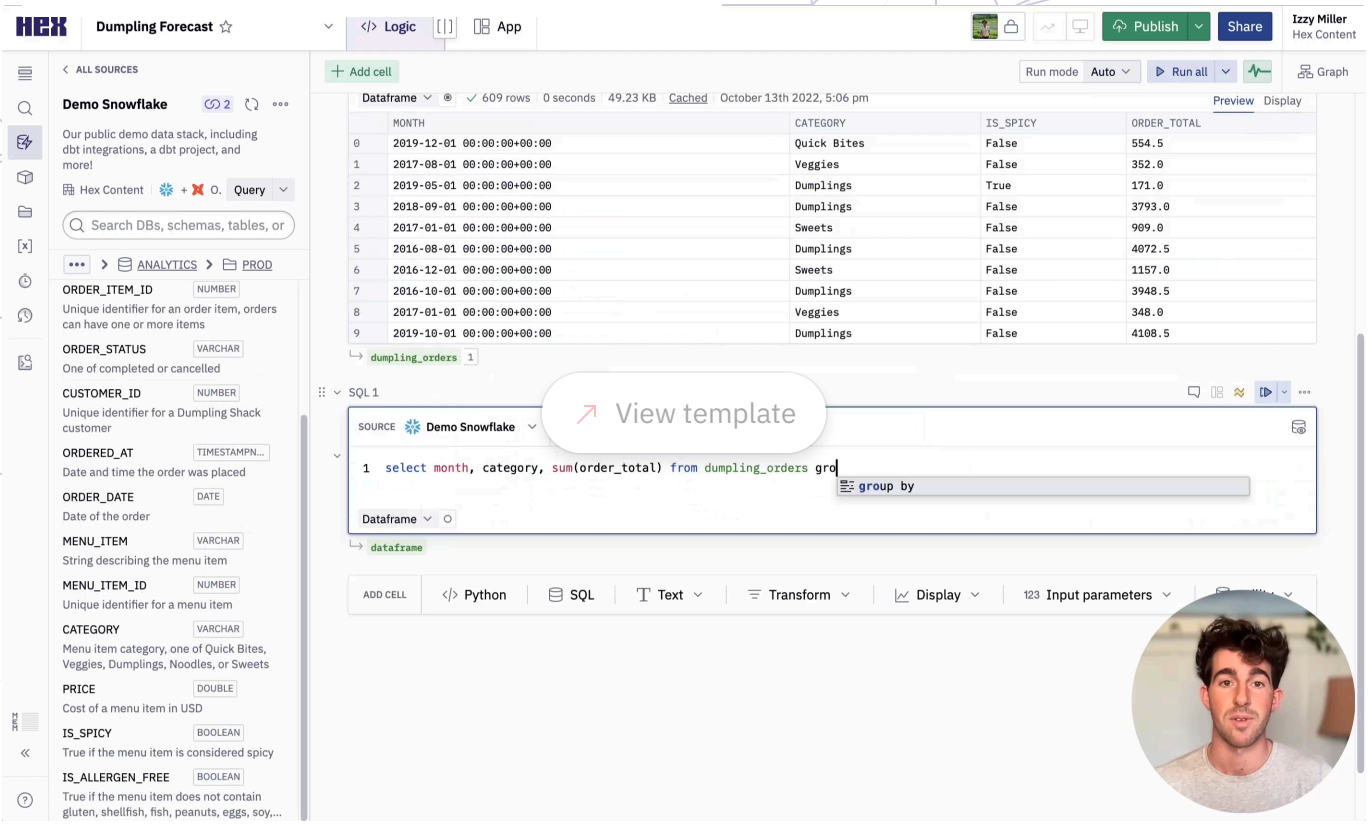
Click below to copy this free template to your workspace.

Copy this template

SHARE:



Click below to copy this free template to your workspace.

Copy this template

vs.

customers, understand their

es, leading to improved satisfaction

- **Finance:** Track market sentiment towards specific stocks or industries, informing investment decisions and risk management.
- **Politics:** Analyze public opinion on political candidates, policies, and events, providing valuable insights for campaigns and governance.
- **Healthcare:** Understand patient sentiment towards treatments, doctors, and healthcare experiences, improving patient care and communication.

How do we unlock this power of sentiment analysis? Python is to go-to tool for building a own sentiment analysis engine. Its ecosystem of libraries and frameworks like NLTK, TextBlob, and spaCy make it ideal for natural language processing tasks.

Here, we want to show you how to use Python and Hex to perform sentiment analysis. We'll cover everything from the foundational concepts to the practical

Click below to copy this free template to your workspace.

Copy this template

kages like NLTK, TF-IDF, SpaCy, or
ase statements in SQL directly

p review data. The data is stored in the Snowflake warehouse that can be accessed within the Hex environment. We'll focus on two main aspects of the Yelp review data:

1. **Narrative Structure:** Analyzing the narrative structure of a review, and its correlation with sentiment. We hypothesize that reviews often tell a story about the reviewer's experience and that this narrative structure might be associated with the sentiment of the review.
2. **Cultural Trends:** Analyzing all reviews over time to uncover cultural trends. We hypothesize that certain types of food or dining experiences might be associated with more positive reviews during their peak in popularity (ex: avocado toast).

Let's start with downloading the necessary dependencies for sentiment analysis.

need to install some additional d
downloaded with the help of the fo

sm

Next, we need to load all the dependencies that are going to be used in this article.

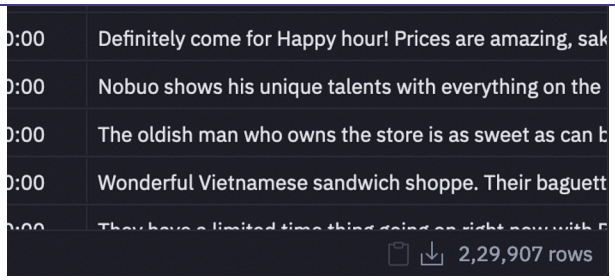
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
import nltk
import re
```

```
cuda/cudart_stub.cc:28] Could not find cuc
/platform/cpu_feature_guard.cc:182] This T
X512F FMA, in other operations, rebuild Te
iler/tf2tensorrt/utils/py_utils.cc:38] TF-
e/hexuser/nltk_data...

ome/hexuser/nltk_data...
eptron_tagger to

eptron_tagger.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/hexuser/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

As you can see in the above code, a lot of text preprocessing, modeling, and visualization libraries are imported. Then the additional data such as `punkt`, `wo rdnet`, `averaged_perceptron_tagger`, and `stopwords` is downloaded from the NLTK library. Finally, the tokenizer, tagger, parser, NER, and word vectors are loaded from the spacy library which will be used further for sentiment analysis.

Data Loading and Preprocessing

Now, we need to load the dataset from the warehouse and process it. We need to perform the text cleaning (removing punctuation, converting to lowercase, etc.) and calculate the sentiment scores for each review as part of the data



As you can see in the above image, the dataset consists of columns like `review_id`, `stars`, `date`, `text`, and so on.

Sentiment Analysis

Once the dataset is loaded, we need to start parsing and classifying the reviews in this dataset. For this, we need to use the TextBlob library of Python that returns two metrics for every text that is passed to it as input. The first one is Polarity, which indicates the positivity/negativity in the sentiment of the text. The second one is subjectivity which refers to objective info/facts versus personal opinions or emotions.

You simply need to load the textblob library and pass your text as input to get the polarity and subjectivity as follows:

				iew	9yKzy9PApeiPPOUJEtnvkg	0.402469	0.659112
			give bad review...	iew	ZRJwVLyzEJq1VAihDhYiow	0.229773	0.638485
IESLBzqUCLdSzSqm0eCSxQ	4	2012-06-14	love the gyro plate. Rice is so good and I als...	review	6oRAC4uyJCsJl1X0WZpVSA	0.566667	0.733333
G-WvGaISbqqaMHlNnByodA	5	2010-05-27	Rosie, Dakota, and I LOVE Chaparral Dog Park!!...	review	_1QQZuf4zZOyFCvXc0o6Vg	0.608646	0.700000

As you can see in the above code, the `get_sentiment()` method returns the polarity and subjectivity scores, Then we store these scores for all the reviews in the `polarity` and `subjectivity` columns of our original dataframe.

Narrative Structure

Click below to copy this free template to your workspace.

Copy this template

```
entiment arc of a review

tences
rity for s in sentences]
    return polarities

# Calculate the sentiment arcs for the first 100 reviews
arcs = reviews['text'].head(100).map(get_sentiment_arc)

# Plot the sentiment arcs
plt.figure(figsize=(10, 6))
for arc in arcs:
    plt.plot(np.linspace(0, 1, len(arc)), arc, alpha=0.1, color='blu
e')
plt.xlabel('Position in review')
plt.ylabel('Sentiment polarity')
plt.title('Sentiment Arcs of Yelp Reviews')
plt.show()
```

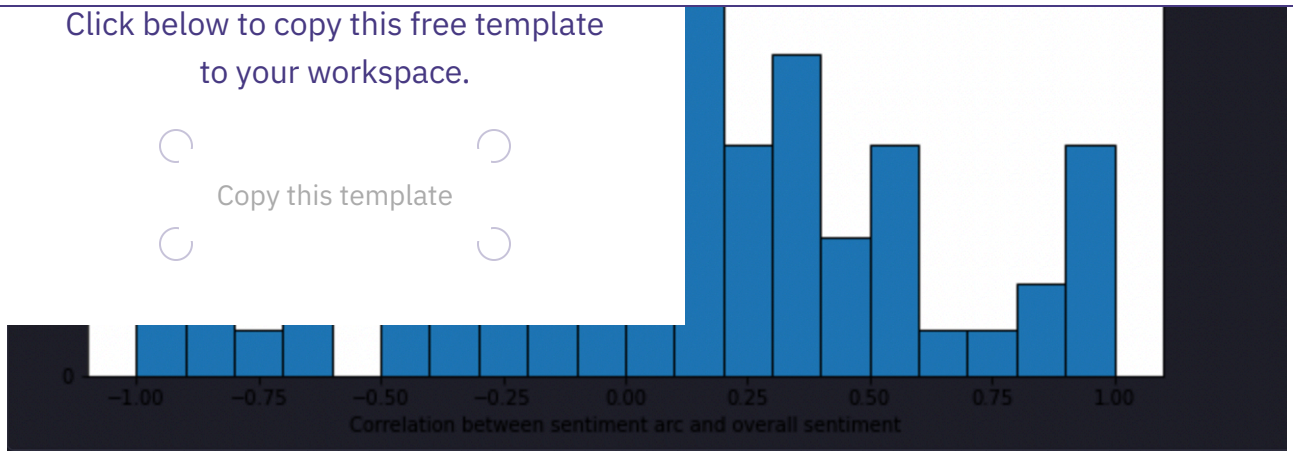
ısing the `get_sentiment_arc()`

t `100` reviews. Then we created a

between the sentiment arc and the overall sentiment, this can be done with the help of the following code:

```
# Define a function to calculate the correlation between the sentiment a
rc and the overall sentiment
def get_correlation(arc, overall_sentiment):
        return np.corrcoef(arc, np.linspace(0, 1, len(arc)) * overall_se
ntiment)[0, 1]

# Calculate the correlations for the first 100 reviews
correlations = [get_correlation(arc, sentiment) for arc, sentiment in zi
p(arcs, reviews['polarity'].head(100))]
```

In the above code, we are simply iterating over the sentiment arc and po larity score (sentiment) of each review to calculate a correlation coeff icient value. We are storing the correlation into a list named correlati ons. Next, you can plot a simple bar graph to visualize these correlatio ns as follows:



Upon inspection, the sentiment arcs of reviews are diverse and vary widely. There's a weak positive correlation between the sentiment arc and an overall positive sentiment— meaning that for overall positive reviews, the positivity of the review increases from the beginning to the end of the review and vice versa for negative ones.

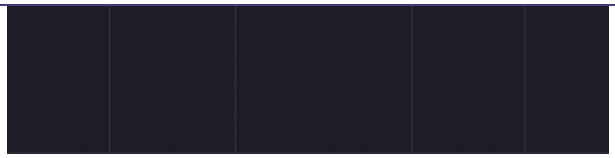
However, it's a pretty weak effect, and the wide distribution means there are probably other factors at play in determining the overall sentiment of a review.

Cultural Trends

Click below to copy this free template to your workspace.

Copy this template

```
ch term

|.str.contains(term, case=False)

# Display the first few rows of the dataframe
reviews.head()
```



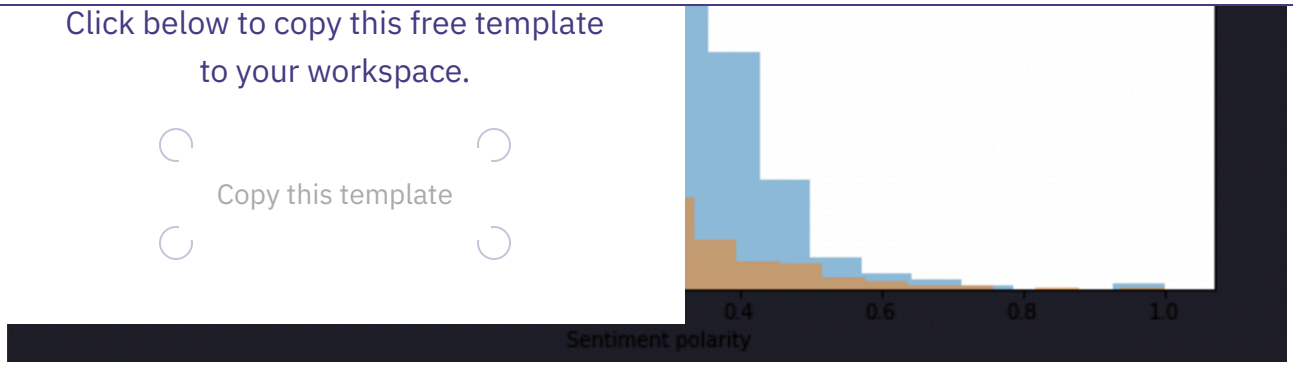
:e a new column in the dataset that
ns appeared. Then we will calculate an aggregated mean of the polarity score for the review containing these terms. Finally, we will create a line chart for visualizing the average sentiment of Yelp reviews mentioning cultural terms.

```
# Convert the date column to datetime
reviews['date'] = pd.to_datetime(reviews['date'])

# Extract the year from the date
reviews['year'] = reviews['date'].dt.year

# Calculate the average sentiment for each term for each year
average_sentiments = {}
for term in terms:
        average_sentiments[term] = reviews[reviews[term]].groupby('year'
)['polarity'].mean()

# Plot the average sentiment over time
plt.figure(figsize=(10, 6))
```

As you can see, the sentiment of the reviews containing the term `avocado` is increasing with time while with the `kale` term, it is decreasing.

you can also visualize the sentiment polarity for each term with the help of the following code:

```
# Calculate the sentiment polarity for each term
sentiment_polarities = {}
for term in terms:
        sentiment_polarities[term] = reviews[reviews[term]]['polarity']

plt.figure(figsize=(10, 6))
for term, polarity in sentiment_polarities.items():
        plt.hist(polarity, bins=20, alpha=0.5, label=term)
```



Review Ratings

Finally, we'll quickly look at the reviews-of-reviews given by other readers on whether a review was useful, funny, or cool. Do these correlate to sentiment—or to one another? To do so, you can simply use the `corr()` method to calculate the correlation among features like `polarity`, `useful`, `funny`, and `cool`.

```
reviews[['funny', 'useful', 'cool']] = pd.DataFrame(reviews['votes'].tolist())

# Calculate the correlation between the sentiment polarity and the number of votes
correlations = reviews[['polarity', 'useful', 'funny', 'cool']].corr()
```

ation between `polarity` and `stars`

```
e sentiment polarity and the numbe

ars']].corr()
```

```
# Display the correlation
correlation
```

	polarity	stars
polarity	1.000000	0.496123
stars	0.496123	1.000000

2 rows

As you can observe, these two features are correlated which makes sense as sentiment reviews directly reflect the ratings that users provide.

This is it! now you have your own Yelp data sentiment analysis system.

Python Packages for Sentiment Analysis

Click below to copy this free template to your workspace.

Copy this template

)asic preprocessing steps where it cleaning, tokenization, and

ntial features from the text and converting it into a format suitable for analysis. Techniques like bag-of-words or TF-IDF (Term Frequency-Inverse Document Frequency) can be effortlessly implemented using NLTK's functionalities.

- Pre-trained models and classifiers in NLTK categorize text into positive, negative, or neutral sentiments. Custom classifiers can be trained as well.
- NLTK includes lexicons, such as the WordNet sentiment lexicon , which assigns sentiment scores to words. By utilizing these lexicons, NLTK enables a lexicon-based approach to sentiment analysis.
- It seamlessly integrates with machine learning models (e.g., Naive Bayes), allowing users to train models on labeled datasets for improved sentiment predictions.

ke breaking down sentences into

eatures from the text, providing

s.

- Its parsing capability helps understand how words in a sentence relate.
- Spacy identifies and categorizes entities (like names or locations), enhancing analysis with specific elements.

TextBlob

TextBlob is a simplified and user-friendly natural language processing library in Python. It incorporates pre-trained models and lexicons, making sentiment analysis accessible without the need for extensive configuration. A standard approach to perform the sentiment analysis with TextBlob is as follows:

- Begin by creating a TextBlob object with the text data you want to analyze. TextBlob automatically processes the text and prepares it for sentiment analysis.

ended for text found on social that it is sensitive to the strength and nt. This tool is a useful resource for atforms since it excels at handling encountered there. The steps of using Vader for sentiment analysis are listed below.

- Begin by installing the VADER sentiment analysis library in your Python environment. You can install it using a package manager like pip.
- Import the VADER module and initialize the sentiment analyzer.
- Utilize the VADER sentiment analyzer to score the sentiment of the text. VADER provides a compound score that represents the overall sentiment intensity.
- Interpret the compound score to understand the sentiment. Positive scores indicate positive sentiment, negative scores denote negativity, and scores around zero suggest neutrality.
- Optionally, fine-tune the analysis by adjusting parameters or thresholds based on specific requirements or domain-specific characteristics.

ures suitable for machine learning.

Scikit-Learn, such as Naive Bayes or entiment analysis.

ed dataset using Scikit-Learn's straightforward API, providing text features and sentiment labels.

- Assess the model's performance using Scikit-Learn's functions for metrics like accuracy, precision, recall, and F1 score.
- Once trained, use the model to predict sentiments for new text data, enabling efficient sentiment analysis.

Gensim

Gensim is used for document similarity analysis and topic modeling. Although Gensim's main focus is on unsupervised learning tasks such as topic modeling, it can also be used for sentiment analysis tasks because it has text processing and vector space modeling functionalities. Gensim is especially helpful in situations where sentiment analysis goals are aligned with a topic extraction

:ific techniques or models.

learning library TensorFlow. Tasks involving sentiment analysis that require the capability of deep learning architectures are a good fit for TensorFlow. You can build a simple deep-learning neural network for sentiment analysis as follows:

- Prepare a labeled dataset with text samples and corresponding sentiment labels (positive, negative, or neutral).
- Use TensorFlow's text preprocessing tools to convert the text into numerical representations. Common techniques include tokenization and padding.
- Design a neural network architecture suitable for sentiment analysis. This typically involves embedding layers, recurrent or convolutional layers, and output layers.
- Compile the model by specifying the optimizer, loss function, and metrics. This step configures the training process.

Click below to copy this free template to your workspace.

Copy this template

d corresponding sentiment labels.

onvert text into numerical

analysis, tailoring it to project requirements.

- Set up the sentiment analysis model using PyTorch's neural network modules.
- Train the model on the dataset using PyTorch's automatic differentiation for efficient backpropagation.
- Evaluate model performance on a validation set using metrics like accuracy, precision, recall, and F1 score.

Sentiment analysis is one of those techniques it is great to have in your toolbox as a data analyst. It allows you to understand how your business, product, or team are perceived by users and public and then act on that information. Given the wealth of tools available to you to build not just basic good/bad sentiment analysis but sophisticated sentiment engines, there is no excuse not to have this technique running on your data all the time.