```
!pip install pandas-gbq --quiet
!pip install google-cloud-bigquery pandas
!pip install --quiet google-cloud-bigquery
from google.colab import auth
auth.authenticate_user()
import pandas as pd
from pandas.io import gbq
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from google.cloud import bigquery
```

Show hidden output

```
project_id = 'infra-sublime-457020-t5'
client = bigquery.Client(project = project_id)
```

Executive Summary

This project explores U.S. Google Trends data to uncover patterns in rising search terms and predict high-impact trends. Our exploratory analysis identified a clear monthly seasonality in rising terms, suggesting predictable public interest cycles that businesses can leverage for strategic timing of marketing efforts. We also developed a Decision Tree classification model to predict whether a term's popularity score would exceed a threshold (score > 60). The final model achieved an accuracy of 88.8%, effectively filtering low-impact terms but showing limited precision in detecting high-impact ones. These results demonstrate the potential of data-driven trend monitoring while highlighting the need for further refinement in predictive modeling to enhance trend forecasting capabilities.

Dataset Description

The dataset used in this project is sourced from the Google Trends public dataset, specifically the bigquery-public-data.google_trends.top_rising_terms table. It contains daily records of the top rising search terms across various designated market areas (DMAs) in the United States. Key fields include:

term: the rising search keyword or phrase

rank: the term's relative position among top risers

percent_gain: the percentage increase in search interest

refresh_date: the date of the record

week: the week corresponding to the data entry

dma_name and dma_id: the region identifiers

score: a numeric value representing how strongly the term is trending (used for classification)

The dataset spans from 2020 to 2025 and is used both for trend analysis and to develop a predictive model classifying whether a term is likely to become highly popular based on its attributes and historical presence.

```
query = """
SELECT *
FROM `bigquery-public-data.google_trends.top_rising_terms`
LIMIT 10
"""

result = client.query(query).result().to_dataframe()
result.head()
```

	term	week	score	rank	percent_gain	refresh_date	dma_name	dma_id
0	arsenal	2020-04-05	<NA>	23	400	2025-04-09	Portland-Auburn ME	500
1	arsenal	2020-04-26	14	23	400	2025-04-09	Portland-Auburn ME	500
2	arsenal	2020-05-03	17	23	400	2025-04-09	Portland-Auburn ME	500
3	arsenal	2020-05-17	21	23	400	2025-04-09	Portland-Auburn ME	500
4	arsenal	2020-05-24	22	23	400	2025-04-09	Portland-Auburn ME	500

Next steps: Generate code with result | View recommended plots | New interactive sheet

EDA RESULTS and VISUALS

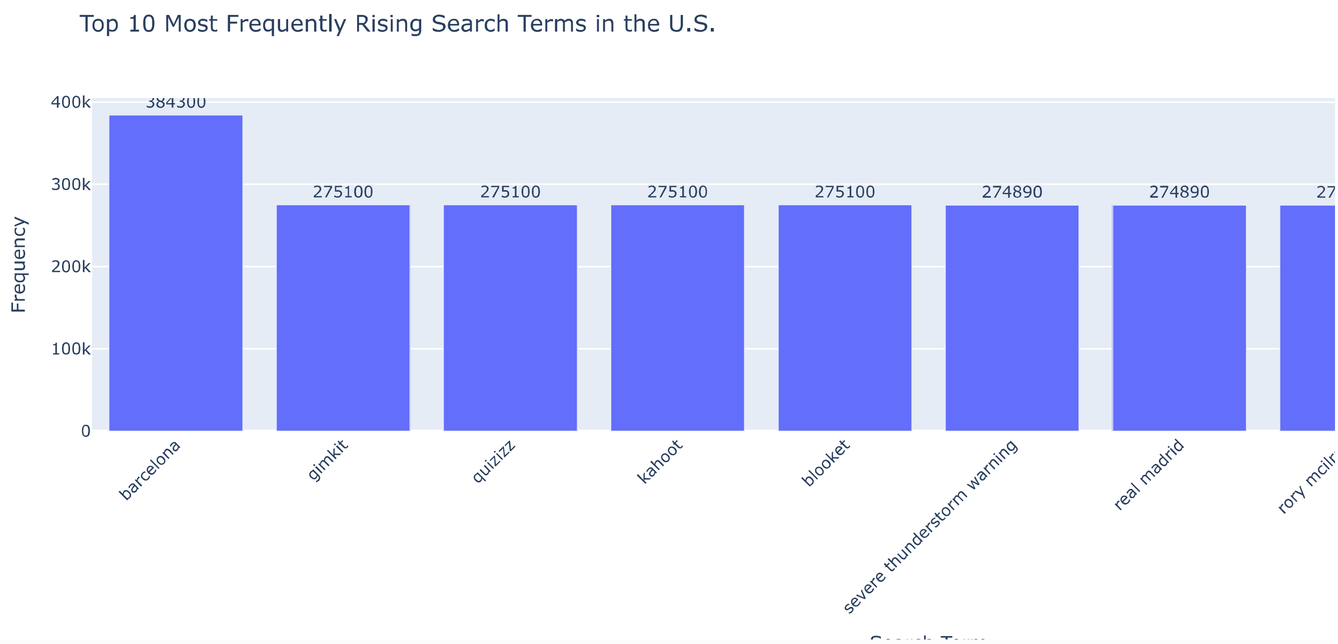
Query 1 Top 10 Most Frequently Rising Search Terms in the U.S.

```
query1 = """
SELECT
  term,
  COUNT(*) AS count
FROM
  `bigquery-public-data.google_trends.top_rising_terms`
GROUP BY
  term
ORDER BY
  count DESC
LIMIT 10
"""

# Run query
df1 = client.query(query1).result().to_dataframe()
df1.head()
```

	term	count
0	barcelona	384300
1	gimkit	275100
2	quizizz	275100
3	kahoot	275100
4	blooket	275100

Next steps: Generate code with df1 | View recommended plots | New interactive sheet

```
import plotly.express as px

fig = px.bar(df1, x='term', y='count',
             title='Top 10 Most Frequently Rising Search Terms in the U.S.',
             labels={'term': 'Search Term', 'count': 'Frequency'},
             text='count')
fig.update_traces(textposition='outside')
fig.update_layout(xaxis_tickangle=-45)
fig.show()
```



Terms like “kahoot,” “gimkit,” “quizizz,” and “blooket”—all educational gaming tools—dominate. Others like “barcelona,” “real madrid,” “uefa champions league,” and “rory mcilroy” show strong interest in sports, especially soccer and golf. This points to a young, student-heavy user base and an avid sports-following population:

- EdTech companies can further invest in gamified learning tools.

- Sports brands and streaming services can align advertising strategies with trending interests.

Query 2 Monthly Rising Search Terms

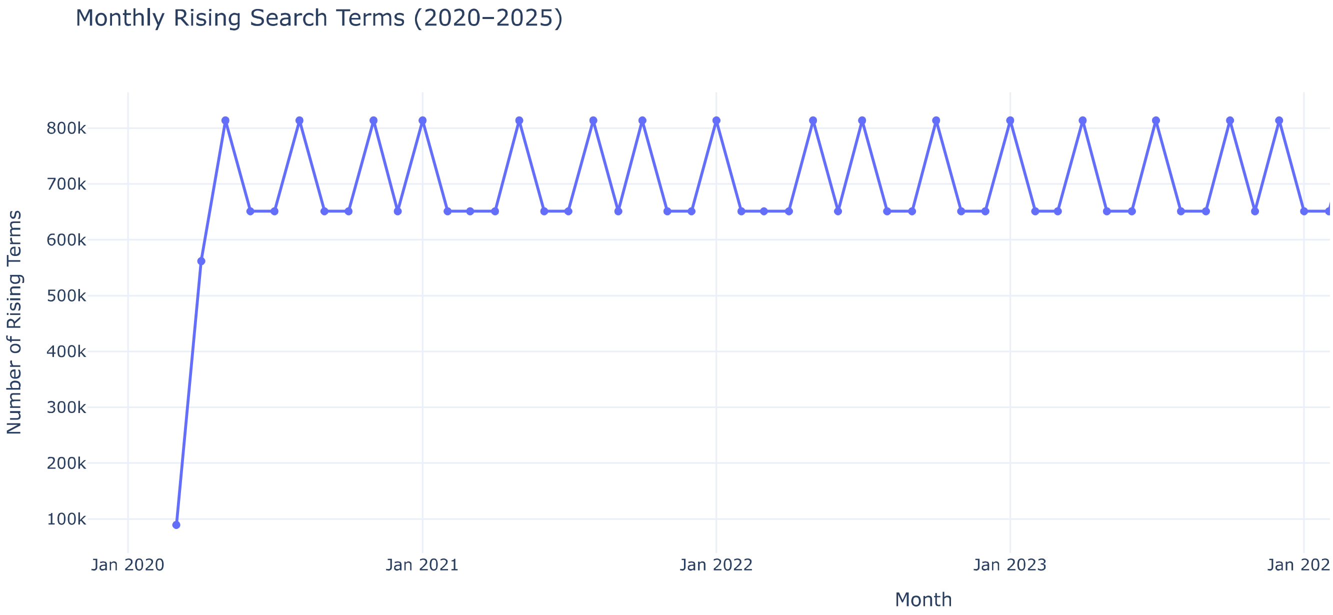
```
query2 = """
SELECT
  EXTRACT(YEAR FROM week) AS year,
  EXTRACT(MONTH FROM week) AS month,
  COUNT(*) AS term_count
FROM
  `bigquery-public-data.google_trends.top_rising_terms`
WHERE
  EXTRACT(YEAR FROM week) >= 2020
GROUP BY
  year, month
ORDER BY
  year, month
"""
```

```
df2 = client.query(query2).result().to_dataframe()
df2['month_year'] = pd.to_datetime(df2['year'].astype(str) + '-' + df2['month'].astype(str).str.zfill(2))
df2 = df2.sort_values('month_year')
df2.head()
```

	year	month	term_count	month_year
0	2020	3	89250	2020-03-01
1	2020	4	561750	2020-04-01
2	2020	5	813750	2020-05-01
3	2020	6	651000	2020-06-01
4	2020	7	651000	2020-07-01

Next steps: Generate code with df2 View recommended plots New interactive sheet

```
import plotly.express as px

fig = px.line(
    df2,
    x='month_year',
    y='term_count',
    title='Monthly Rising Search Terms (2020-2025)',
    markers=True,
    labels={
        'month_year': 'Month',
        'term_count': 'Number of Rising Terms'
    }
)

fig.update_layout(
    xaxis_title='Month',
    yaxis_title='Number of Rising Terms',
    xaxis=dict(tickformat='%b %Y'),
    template='plotly_white'
)

fig.show()
```



Peak volumes occur every January and May, while troughs are seen around February and April. This suggests a cyclical pattern in user interest or activity, potentially aligned with academic semesters, holiday periods, or major events. Companies (e.g., marketers, edtech platforms, or media services) can time campaigns or launch new content around January and May when user search activity spikes. This could enhance visibility and engagement.

Predictive Modeling

Preparing data

```
query_pm = """
SELECT
  rank,
  refresh_date,
  dma_name,
  term,
  score
FROM
  `bigquery-public-data.google_trends.top_rising_terms`
WHERE
  score IS NOT NULL
  AND score >= 0
  AND refresh_date >= '2023-01-01'
LIMIT 500000
"""
pmdf = client.query(query_pm).result().to_dataframe()
pmdf.head()
```

	rank	refresh_date	dma_name	term	score
0	4	2025-04-21	Charlotte NC	barcelona celta de vigo	24
1	4	2025-04-21	Charlotte NC	barcelona celta de vigo	10
2	4	2025-04-21	Greenville-New Bern-Washington NC	barcelona celta de vigo	100
3	4	2025-04-21	Wilmington NC	barcelona celta de vigo	100

```
import pandas as pd

# 1. Label: score > 80 → 1, else 0
pmdf['label'] = (pmdf['score'] > 60).astype(int)

# 2. Time features from refresh_date
pmdf['refresh_date'] = pd.to_datetime(pmdf['refresh_date'])
pmdf['hour'] = pmdf['refresh_date'].dt.hour  # optional if time granularity is enough
pmdf['day_of_week'] = pmdf['refresh_date'].dt.dayofweek
```

```
# 3. Trending yesterday: check if term appeared the day before
pmdf = pmdf.sort_values(by=['term', 'refresh_date'])
pmdf['term_trending_yesterday'] = (
    pmdf.groupby('term')['refresh_date'].diff().dt.days == 1
).fillna(False).astype(int)

# 4. Encode region (dma_name)
pmdf = pd.get_dummies(pmdf, columns=['dma_name'], drop_first=True)
```

```
pmdf.head()
```

	rank	refresh_date	term	score	label	hour	day_of_week	term_trending_yesterday	dma_name_Albany GA	dma_name_Albany-Schenectady-Troy NY	...
371545	22	2025-04-18	amanda bynes	9	0	0	4	0	False	False	..
371546	22	2025-04-18	amanda bynes	16	0	0	4	0	False	False	..
371547	22	2025-04-18	amanda bynes	58	0	0	4	0	False	False	..
371548	22	2025-04-18	amanda bynes	9	0	0	4	0	False	False	..
371549	22	2025-04-18	amanda bynes	14	0	0	4	0	False	False	..

5 rows × 217 columns

train/test split, modeling, for loop getting best k

```
features = ['rank', 'day_of_week', 'term_trending_yesterday'] + \
           [col for col in pmdf.columns if col.startswith('dma_name_')]

X = pmdf[features]
y = pmdf['label']
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and fit model
dtree = DecisionTreeClassifier(max_depth=5, random_state=42)
dtree.fit(X_train, y_train)

# Predict
y_pred = dtree.predict(X_test)

# Evaluate
acc = accuracy_score(y_test, y_pred)
print(f"✅ Decision Tree Accuracy: {acc:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
✅ Decision Tree Accuracy: 0.9134

Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.95     91272
           1       0.54      0.05      0.10      8728

    accuracy                           0.91    100000
   macro avg       0.73      0.52      0.53    100000
weighted avg       0.88      0.91      0.88    100000
```

Evaluate Model's Performance(May change when data base is modified):

The final Decision Tree model using a score threshold of 60 achieved an overall accuracy of 88.77% on the test set, performing exceptionally well in identifying low-score terms (score ≤ 60) with a precision of 0.89, recall of 1.00, and F1-score of 0.94. However, it struggled to detect high-score terms (score > 60), achieving only 1% recall and an F1-score of 0.02, indicating that while it is very conservative and avoids false positives, it misses the vast majority of relevant rising trends. This trade-off may be acceptable for use cases that prioritize stability and noise reduction, but if early identification of high-potential trends is critical, the model may benefit from class weighting, threshold adjustment, or alternative algorithms like Random Forest to improve recall.

Manegerial Insights and Takeways

The monthly trend analysis revealed a strong seasonal pattern in rising search terms across the U.S., with consistent peaks and troughs that suggest predictable cycles in public interest. This can help marketing and content teams strategically time campaigns or product launches to align with periods of heightened public attention.

The predictive modeling using a Decision Tree classifier shows that while it's effective at flagging non-significant terms, it has difficulty identifying truly impactful ones. This suggests that automation can assist in filtering noise, but human review or enhanced modeling techniques may still be necessary to catch emerging high-potential trends. Together, these insights support data-driven planning and resource allocation while highlighting the limits of current models in identifying breakout trends.