iulsegboe

April 17, 2025

[118]:
```
from snowflake.snowpark import Session
import pandas as pd
from snowflake.connector.pandas_tools import write_pandas
```

[75]:
```
connection_parameters = {
    "account": "XOB39151",
    "user": "CHIPMUNK",
    "password": "m65LM2RqSfp696Y",
    "role": "TRAINING_ROLE",
    "warehouse": "ANIMAL_TASK_WH",
    "database": "NETFLIX_PROJECT_DB",
    "schema": "NETFLIX_ANALYTICS"
}
```

[77]:
```
session = Session.builder.configs(connection_parameters).create()
```

[11]:
```
global_alltime = session.table("CLEAN_GLOBAL_ALLTIME").to_pandas()
global_alltime.head()
```

[11]:
```
          SHOW_TITLE         CATEGORY  RANK  HOURS_VIEWED_FIRST_91_DAYS  \
0         red notice  films (english)     1                   454200000
1      don't look up  films (english)     2                   408600000
2           carry-on  films (english)     3                   340800000
3  the adam project  films (english)     4                   281000000
4           bird box  films (english)     5                   325300000

   VIEWS_FIRST_91_DAYS  RUNTIME SEASON_TITLE
0            230900000        2          N/A
1            171400000        2          N/A
2            170400000        2          N/A
3            157600000        2          N/A
4            157400000        2          N/A
```

[15]:
```
# Add a format column
global_alltime['FORMAT'] = global_alltime['CATEGORY'].str.contains('series',␣
 ↪case=False).astype(int)
```

[21]:
```python
global_alltime.columns = global_alltime.columns.str.lower()
```

[23]:
```python
global_alltime.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40 entries, 0 to 39
Data columns (total 8 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   show_title                   40 non-null     object
 1   category                     40 non-null     object
 2   rank                         40 non-null     int8
 3   hours_viewed_first_91_days  40 non-null     int64
 4   views_first_91_days          40 non-null     int32
 5   runtime                      40 non-null     int8
 6   season_title                 40 non-null     object
 7   format                       40 non-null     int64
dtypes: int32(1), int64(2), int8(2), object(3)
memory usage: 1.9+ KB
```

[25]:
```python
# Drop rows with nulls in key features
global_alltime = global_alltime.dropna(subset=['runtime',␣
 ↪'views_first_91_days'])

# Ensure numeric types
global_alltime['runtime'] = global_alltime['runtime'].astype(float)
global_alltime['views_first_91_days'] = global_alltime['views_first_91_days'].
 ↪astype(float)
```

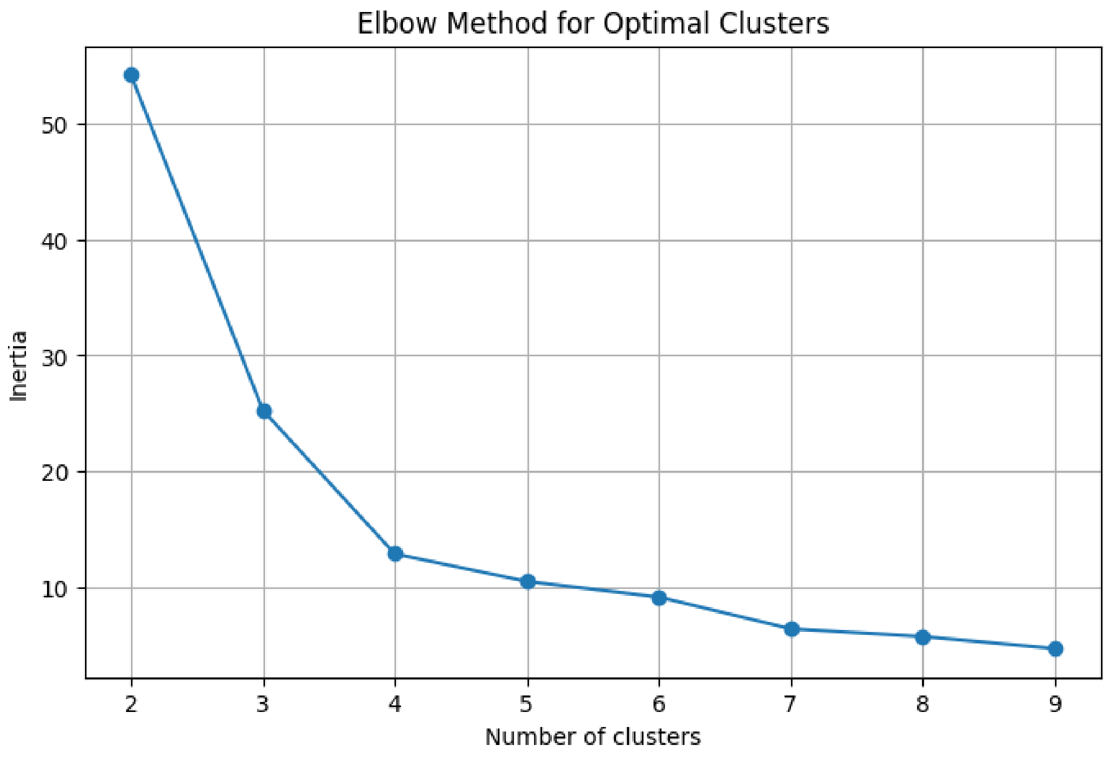
[27]:
```python
from sklearn.preprocessing import StandardScaler

features = global_alltime[['runtime', 'views_first_91_days', 'format']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)
```

[37]:
```python
inertias = []
for k in range(2, 10):
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X_scaled)
    inertias.append(km.inertia_)

# Plot Elbow Curve
import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
plt.plot(range(2,10), inertias, marker='o')
plt.title("Elbow Method for Optimal Clusters")
plt.xlabel("Number of clusters")
plt.ylabel("Inertia")
plt.grid(True)
plt.show()
```

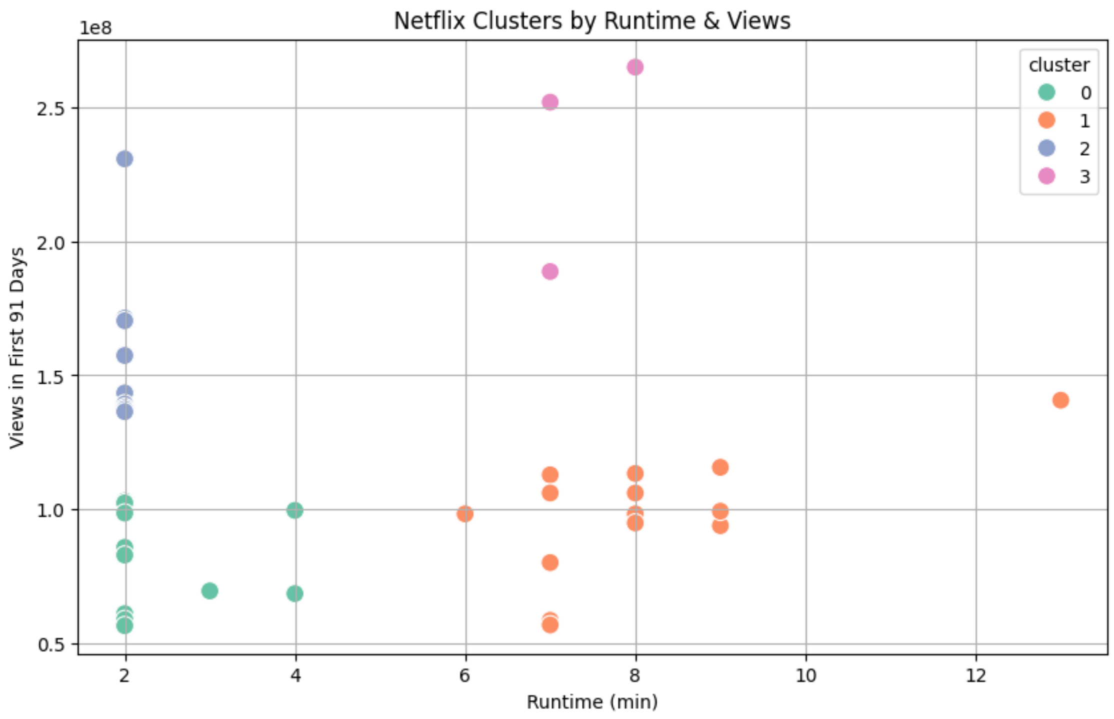


[29]:
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=4, random_state=42)
global_alltime['cluster'] = kmeans.fit_predict(X_scaled)
```

[31]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.scatterplot(
    x=global_alltime['runtime'],
    y=global_alltime['views_first_91_days'],
    hue=global_alltime['cluster'],
    palette='Set2',
    s=100
)
plt.title("Netflix Clusters by Runtime & Views")
plt.xlabel("Runtime (min)")
plt.ylabel("Views in First 91 Days")
plt.grid(True)
plt.show()
```



[33]:
```
from sklearn.metrics import silhouette_score

score = silhouette_score(X_scaled, global_alltime['cluster'])
print("Silhouette Score:", round(score, 3))
```

```
Silhouette Score: 0.599
```

[35]:
```
session.write_pandas(
    global_alltime[['show_title', 'runtime', 'views_first_91_days', 'format',␣
 ↪'cluster']],
    "CLUSTERED_CONTENT",
    auto_create_table=True,
    overwrite=True
)
```

[35]:
```
<snowflake.snowpark.table.Table at 0x13ff1a450>
```

[49]:
```
global_alltime_regression = global_alltime.copy()
```

```python
# Format
global_alltime_regression['format'] = global_alltime_regression['category'].str.
 ↪lower().str.contains('series').astype(int)

# Encode category
global_alltime_regression['category_encoded'] =␣
 ↪global_alltime_regression['category'].astype('category').cat.codes

# Convert runtime from hours to minutes
global_alltime['runtime'] = global_alltime['runtime'].astype(float) * 60
```

[51]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np

# Features and target
X = global_alltime_regression[['runtime', 'format', 'category_encoded']]
y = global_alltime_regression['views_first_91_days']

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

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print("R² Score:", round(r2_score(y_test, y_pred), 3))
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE:", round(rmse, 2))
```

```
R² Score: -0.602
RMSE: 39958539.57
```

[61]:
```python
# Input values
runtime = 100
format = 1
category_encoded = 3

# Simulate
new_data = pd.DataFrame([[runtime, format, category_encoded]], columns=X.
 ↪columns)
predicted_views = model.predict(new_data)[0]
```

```
print("Predicted Views in 91 Days:", round(predicted_views))
```

```
Predicted Views in 91 Days: 687775229
```

[67]:
```
simulated_inputs = pd.DataFrame([
    [40, 0, 1],
    [90, 1, 2],
    [120, 0, 3]
], columns=['runtime', 'format', 'category_encoded'])

# Add predicted views
simulated_inputs['predicted_views'] = model.predict(simulated_inputs)

# Add scenario names
simulated_inputs['scenario'] = ['Short Film', '90-min Series', 'Epic Action␣
↪Film']

# Reorder columns
simulated = simulated_inputs[['scenario', 'runtime', 'format',␣
↪'category_encoded', 'predicted_views']]
```

[69]:
```
session.write_pandas(
    simulated,
    "WHAT_IF_PREDICTIONS",
    auto_create_table=True,
    overwrite=True
)
```

[69]:
```
<snowflake.snowpark.table.Table at 0x168d3a3c0>
```

[79]:
```
# Load table
country_weekly = session.table("CLEAN_COUNTRY_WEEKLY").to_pandas()

# lowercasing + fixing column names
country_weekly.columns = country_weekly.columns.str.lower().str.replace(' ',␣
↪'_')

country_weekly.head()
```

[79]:
```
  country_name country_iso2        week category  weekly_rank  \
0    Argentina           AR  2025-03-02    films            1
1    Argentina           AR  2025-03-02    films            2
2    Argentina           AR  2025-03-02    films            3
3    Argentina           AR  2025-03-02    films            4
4    Argentina           AR  2025-03-02    films            5

                 show_title season_title  cumulative_weeks_in_top_10
```

```
0           counterattack          N/A                            1
1               uncharted          N/A                            2
2   a copenhagen love story        N/A                            1
3        ticket to paradise        N/A                            2
4            despicable me 3       N/A                            7
```

[81]:
```
# Make sure all column names are clean
country_weekly.columns = country_weekly.columns.str.lower().str.replace(' ',␣
↪'_')

# Preview columns
print(country_weekly.columns)

# Optional: Rename to make it more standard
country_weekly.rename(columns={
    'country_name': 'country',
    'show_title': 'title',
    'week': 'date',
    'weekly_rank': 'rank'
}, inplace=True)

# Check for nulls or duplicates
print(country_weekly.isnull().sum())
print(f"Duplicates: {country_weekly.duplicated().sum()}")
```

```
Index(['country_name', 'country_iso2', 'week', 'category', 'weekly_rank',
       'show_title', 'season_title', 'cumulative_weeks_in_top_10'],
      dtype='object')
country                       0
country_iso2                  0
date                          0
category                      0
rank                          0
title                         0
season_title                  0
cumulative_weeks_in_top_10    0
dtype: int64
Duplicates: 0
```

[83]:
```
# Keep only Top 10 ranks
country_weekly = country_weekly[country_weekly['rank'] <= 10].copy()
```

[85]:
```
country_weekly['date'] = pd.to_datetime(country_weekly['date'])
```

[91]:
```
selected_countries = ['India', 'United States', 'Brazil', 'Morocco', 'United␣
↪Arab Emirates']
```

```
country_filtered = country_weekly[country_weekly['country'].
  ↪isin(selected_countries)]
```

```
[93]: country_filtered = country_filtered.sort_values(by=['country', 'title', 'date'])
      country_filtered.reset_index(drop=True, inplace=True)
```

```
[103]: country_filtered.tail()
```

```
[103]:              country country_iso2       date category  rank  \
       18200  United States           US 2023-10-22       tv     1
       18201  United States           US 2023-10-29       tv     3
       18202  United States           US 2023-11-05       tv     8
       18203  United States           US 2023-08-13    films     6
       18204  United States           US 2021-09-19    films     9
       …                …          …          …      …   …
       19195  United States           US 2025-03-02       tv     2
       19196  United States           US 2023-06-11    films     5
       19197  United States           US 2023-06-18    films     8
       19198  United States           US 2023-05-14    films     6
       19199  United States           US 2023-05-21    films     7

                                     title  \
       18200  the fall of the house of usher
       18201  the fall of the house of usher
       18202  the fall of the house of usher
       18203        the fast and the furious
       18204  the father who moves mountains
       …                                …
       19195                        zero day
       19196                       zookeeper
       19197                       zookeeper
       19198                ¡que viva méxico!
       19199                ¡que viva méxico!

                                           season_title  \
       18200  The Fall of the House of Usher: Limited Series
       18201  The Fall of the House of Usher: Limited Series
       18202  The Fall of the House of Usher: Limited Series
       18203                                            N/A
       18204                                            N/A
       …                                              …
       19195                        Zero Day: Limited Series
       19196                                            N/A
       19197                                            N/A
       19198                                            N/A
       19199                                            N/A
       cumulative_weeks_in_top_10
18200                           2
18201                           3
18202                           4
18203                           1
18204                           1
…                             …
19195                           2
19196                           1
19197                           2
19198                           1
19199                           2

[1000 rows x 8 columns]
```

[105]:
```python
from prophet import Prophet
import pandas as pd
from tqdm import tqdm
```

[ ]:
```python
forecast_results = []

# Group the dataset
grouped = country_weekly.groupby(['country', 'title'])

for (country, title), group in tqdm(grouped, total=len(grouped)):
    try:
        # Must have at least 4 records for Prophet to work
        if group.shape[0] < 4:
            continue

        # Prepare time series format
        ts = group[['date', 'rank']].rename(columns={'date': 'ds', 'rank': 'y'})
        ts = ts.sort_values('ds')
        ts['y'] = 11 - ts['y']  # Reverse rank: higher = better

        # Fit the model
        model = Prophet()
        model.fit(ts)

        # Forecast 1 week ahead
        future = model.make_future_dataframe(periods=1, freq='W')
        forecast = model.predict(future)

        # Get last predicted value
        latest = forecast.tail(1)

        forecast_results.append({
            'country': country,
            'title': title,
            'forecast_week': latest['ds'].values[0],
            'predicted_score': latest['yhat'].values[0],
            'confidence_low': latest['yhat_lower'].values[0],
            'confidence_high': latest['yhat_upper'].values[0]
        })

    except Exception as e:
        print(f"Error for {country} - {title}: {e}")
        continue
```

```
  0%|                                      | 1/123788 [00:00<19:16:31,
1.78it/s]18:13:26 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
  0%|                                       | 6/123788 [00:01<5:49:33,
5.90it/s]18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
  0%|                                      | 22/123788 [00:01<1:21:12,
25.40it/s]18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
  0%|                                      | 30/123788 [00:01<1:11:05,
29.01it/s]18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
  0%|                                        | 40/123788 [00:01<55:30,
37.16it/s]18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
  0%|                                        | 61/123788 [00:01<31:11,
66.11it/s]18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
18:13:27 - cmdstanpy - INFO - Chain [1] start processing
18:13:27 - cmdstanpy - INFO - Chain [1] done processing
  0%|                                        | 72/123788 [00:02<37:56,
18:54:10 - cmdstanpy - INFO - Chain [1] done processing
18:54:10 - cmdstanpy - INFO - Chain [1] start processing
18:54:10 - cmdstanpy - INFO - Chain [1] done processing
 96%|                    | 118551/123788 [40:44<01:01,
85.48it/s]18:54:10 - cmdstanpy - INFO - Chain [1] start processing
18:54:10 - cmdstanpy - INFO - Chain [1] done processing
18:54:10 - cmdstanpy - INFO - Chain [1] start processing
18:54:10 - cmdstanpy - INFO - Chain [1] done processing
18:54:10 - cmdstanpy - INFO - Chain [1] start processing
18:54:10 - cmdstanpy - INFO - Chain [1] done processing
```

[112]:
```
forecast_df = pd.DataFrame(forecast_results)
forecast_df['predicted_rank'] = 11 - forecast_df['predicted_score']
forecast_df = forecast_df.sort_values(['country', 'predicted_rank'])
```

[114]:
```
forecast_df.head()
```

[114]:
```
       country                                title forecast_week  \
182  Argentina                       the equalizer 2    2025-01-19
189  Argentina                     the lincoln lawyer    2024-11-03
60   Argentina                        emily in paris    2024-10-06
93   Argentina  john wick: chapter 3 - parabellum    2024-01-28
156  Argentina                          sex education    2023-10-29

     predicted_score  confidence_low  confidence_high  predicted_rank
182        28.534371       28.508582        28.559863      -17.534371
189        26.791828       26.784899        26.798756      -15.791828
60         17.378035       15.997608        18.736663       -6.378035
93         11.326164       11.326162        11.326165       -0.326164
156        10.977782        7.824085        14.257803        0.022218
```

[122]:
```
forecast_df['forecast_week'] = forecast_df['forecast_week'].dt.
  ↪strftime('%Y-%m-%d')
```

[124]:
```
session.write_pandas(
    forecast_df,
    table_name="PREDICTED_TOP_SHOWS_NEXT_WEEK",
    database="NETFLIX_PROJECT_DB",
    schema="NETFLIX_ANALYTICS",
    auto_create_table=True,
    overwrite=True
)
```

```
/opt/anaconda3/lib/python3.12/site-packages/snowflake/snowpark/session.py:3132:
UserWarning: Pandas Dataframe has non-standard index of type <class
'pandas.core.indexes.base.Index'> which will not be written. Consider changing
the index to pd.RangeIndex(start=0,…,step=1) or call reset_index() to keep
index as column(s)
```

```
 success, _, _, ci_output = write_pandas(
```

[124]:
```
<snowflake.snowpark.table.Table at 0x166ca4980>
```

[128]:
```python
# Load CLEAN_COUNTRY_WEEKLY from Snowflake
country_weekly2 = session.table("CLEAN_COUNTRY_WEEKLY").to_pandas()
```

[130]:
```python
country_weekly2.head()
```

[130]:
```
  COUNTRY_NAME COUNTRY_ISO2       WEEK CATEGORY  WEEKLY_RANK  \
0    Argentina           AR 2025-03-02    films            1
1    Argentina           AR 2025-03-02    films            2
2    Argentina           AR 2025-03-02    films            3
3    Argentina           AR 2025-03-02    films            4
4    Argentina           AR 2025-03-02    films            5

              SHOW_TITLE SEASON_TITLE  CUMULATIVE_WEEKS_IN_TOP_10
0          counterattack          N/A                           1
1              uncharted          N/A                           2
2  a copenhagen love story          N/A                           1
3      ticket to paradise          N/A                           2
4         despicable me 3          N/A                           7
```

[132]:
```python
# Rename and clean columns
country_weekly2.columns = country_weekly2.columns.str.lower().str.replace(" ",␣
  ↪"_")
country_weekly2 = country_weekly2.sort_values(by=['country_name', 'show_title',␣
  ↪'week'])
country_weekly2.rename(columns={
    'country_name': 'country',
    'show_title': 'title',
    'week': 'date',
    'weekly_rank': 'rank'
}, inplace=True)

# Convert to datetime
country_weekly2['date'] = pd.to_datetime(country_weekly2['date'])

# SHIFT: Get next week's rank to create target
country_weekly2['next_rank'] = country_weekly2.groupby(['country',␣
  ↪'title'])['rank'].shift(-1)
country_weekly2['is_rank1_next_week'] = (country_weekly2['next_rank'] == 1).
  ↪astype(int)

# Encode categorical fields
country_weekly2['country_encoded'] = country_weekly2['country'].
  ↪astype('category').cat.codes
```

```
country_weekly2['title_encoded'] = country_weekly2['title'].astype('category').
  ↪cat.codes

# Feature: Trend (change in rank)
country_weekly2['prev_rank'] = country_weekly2.groupby(['country',␣
  ↪'title'])['rank'].shift(1)
country_weekly2['rank_trend'] = country_weekly2['prev_rank'] -␣
  ↪country_weekly2['rank']

# Feature: Weeks in Top 10
country_weekly2['weeks_in_top10'] = country_weekly2.groupby(['country',␣
  ↪'title']).cumcount() + 1

# Drop rows with NaNs (from shifting)
country_weekly2.dropna(subset=['next_rank', 'prev_rank'], inplace=True)
```

[136]:
```
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

features = ['rank', 'rank_trend', 'weeks_in_top10', 'country_encoded',␣
  ↪'title_encoded']
X = country_weekly2[features]
y = country_weekly2['is_rank1_next_week']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
  ↪random_state=42)

model = XGBClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))

# Add predictions and probabilities
country_weekly2['rank1_prob'] = model.predict_proba(X)[:, 1]
```

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     28779
           1       0.71      0.23      0.35      2075

    accuracy                           0.94     30854
   macro avg       0.83      0.61      0.66     30854
weighted avg       0.93      0.94      0.93     30854
```

[138]:
```
latest_date = country_weekly2['date'].max()
latest = country_weekly2[country_weekly2['date'] == latest_date].copy()

# Top predicted show per country
top_preds = latest.sort_values(['country', 'rank1_prob'], ascending=[True,␣
 ↪False]) \
                  .groupby('country').first().reset_index()

# Clean result
top_preds = top_preds[['country', 'title', 'date', 'rank1_prob']]
top_preds.rename(columns={'date': 'forecast_week'}, inplace=True)

# Optional: round probability
top_preds['rank1_prob'] = top_preds['rank1_prob'].round(3)
```

[140]:
```
from snowflake.connector.pandas_tools import write_pandas

# Get low-level connection from session
conn = session._conn._conn

write_pandas(
    conn=conn,
    df=top_preds,
    table_name="PREDICTED_RANK1_NEXT_WEEK",
    database="NETFLIX_PROJECT_DB",
    schema="NETFLIX_ANALYTICS",
    auto_create_table=True,
    overwrite=True
)
```

[140]:
```
(True,
 1,
 16,
 [('kookxvqkdl/file0.txt', 'LOADED', 16, 16, 1, 0, None, None, None, None)])
```

[ ]:
```

```