0.1 CaseCraft: The Analytics Sprint – Project 17

0.1.1 Netflix Genre Popularity Forecast

Subheading: Forecasting genre-level engagement trends on Netflix using time series modeling and stream-style visualizations.

0.1.2 Project Goals

- Simulate Netflix viewing data across genres and time
- Engineer features: watch hours, viewer count, engagement rate
- Apply Prophet for genre-level time series forecasting
- Visualize genre evolution using streamgraphs and ridge plots
- Build regression model to predict future genre popularity
- Summarize insights for content strategy and scheduling

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from prophet import Prophet

np.random.seed(42)

genres = ['Drama', 'Comedy', 'Thriller', 'Romance', 'Sci-Fi', 'Documentary']
dates = pd.date_range(start='2023-01-01', periods=180)

data = []
for genre in genres:
    base = np.random.randint(1000, 5000)
    for date in dates:
        fluctuation = np.random.normal(0, 500)
```

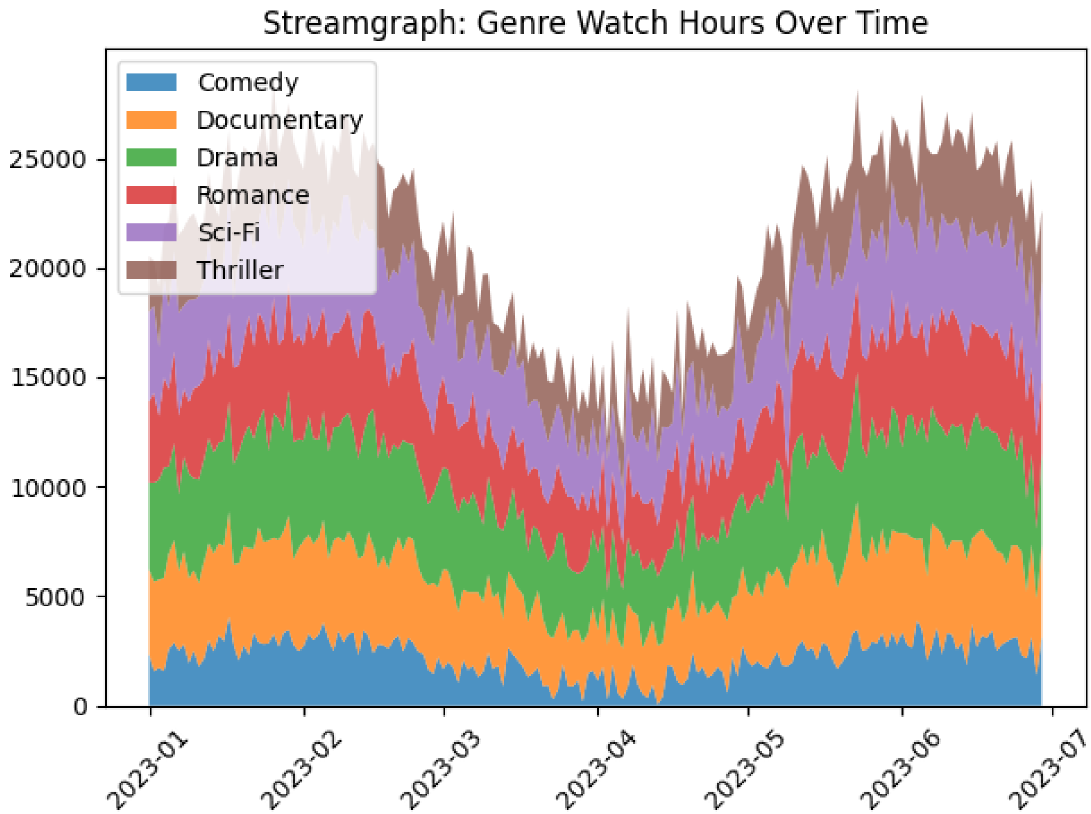
```
        watch_hours = max(base + fluctuation + np.sin(date.dayofyear / 20) *␣
↪1000, 0)
        data.append([date, genre, watch_hours])

df = pd.DataFrame(data, columns=['date', 'genre', 'watch_hours'])
```

[2]:
```
df.head(10)
```

[2]:
```
        date  genre  watch_hours
0 2023-01-01  Drama  3948.861925
1 2023-01-02  Drama  4531.549953
2 2023-01-03  Drama  4560.368549
3 2023-01-04  Drama  5056.894392
4 2023-01-05  Drama  3962.990537
5 2023-01-06  Drama  4407.446617
6 2023-01-07  Drama  3511.416363
7 2023-01-08  Drama  4317.016630
8 2023-01-09  Drama  4805.255409
9 2023-01-10  Drama  4188.833205
```

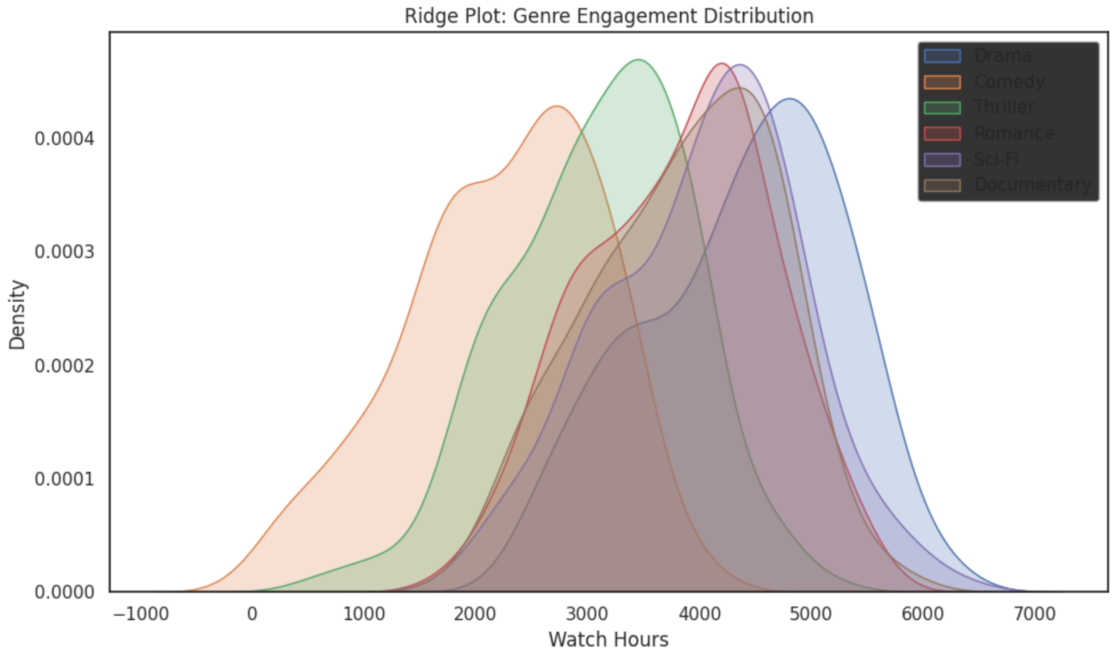
0.1.3 Streamgraph: Genre Watch Hours Over Time

[3]:
```
pivot_df = df.pivot(index='date', columns='genre', values='watch_hours').
↪fillna(0)

plt.stackplot(pivot_df.index, pivot_df.T, labels=pivot_df.columns, alpha=0.8)
plt.legend(loc='upper left')
plt.title("Streamgraph: Genre Watch Hours Over Time")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



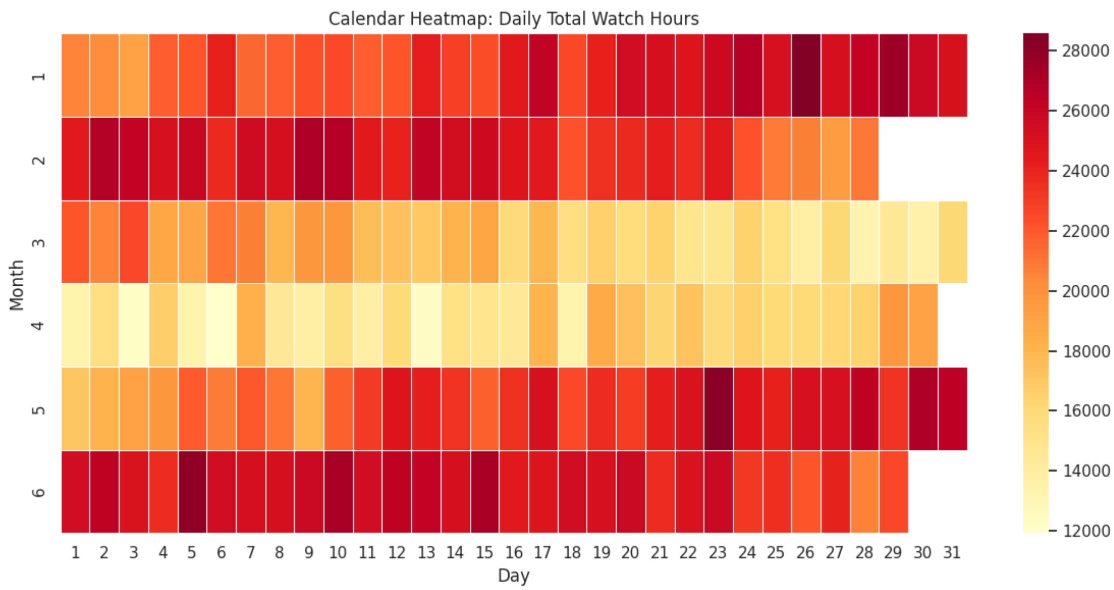
0.1.4 Ridge Plot: Genre Engagement Distribution

[4]:
```
plt.figure(figsize=(10, 6))
sns.set(style="white", rc={"axes.facecolor": (0, 0, 0, 0)})

for i, genre in enumerate(genres):
    subset = df[df['genre'] == genre]
    sns.kdeplot(subset['watch_hours'], bw_adjust=1, label=genre, fill=True)

plt.title("Ridge Plot: Genre Engagement Distribution")
plt.xlabel("Watch Hours")
plt.tight_layout()
plt.legend()
plt.show()
```



0.1.5 Calendar Heatmap: Daily Total Watch Hours

[7]:
```
daily_total = df.groupby('date')['watch_hours'].sum().reset_index()
daily_total['day'] = daily_total['date'].dt.day
daily_total['month'] = daily_total['date'].dt.month

heatmap_data = daily_total.pivot(index='month', columns='day',␣
 ↪values='watch_hours')

plt.figure(figsize=(12, 6))
sns.heatmap(heatmap_data, cmap='YlOrRd', linewidths=0.5)
plt.title("Calendar Heatmap: Daily Total Watch Hours")
plt.xlabel("Day")
plt.ylabel("Month")
plt.tight_layout()
plt.show()
```

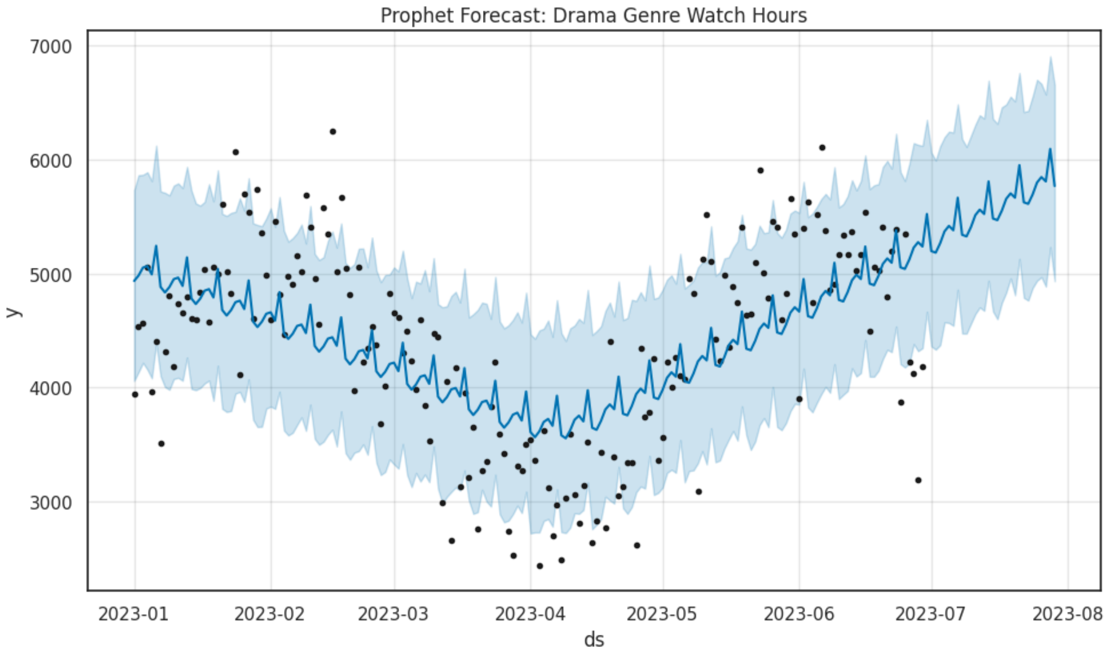


0.1.6 Prophet Forecast: Drama Genre

[8]:
```
drama_df = df[df['genre'] == 'Drama'][['date', 'watch_hours']].
 ↪rename(columns={'date': 'ds', 'watch_hours': 'y'})

model = Prophet()
model.fit(drama_df)

future = model.make_future_dataframe(periods=30)
forecast = model.predict(future)

fig = model.plot(forecast)
plt.title("Prophet Forecast: Drama Genre Watch Hours")
plt.tight_layout()
plt.show()
```

```
INFO:prophet:Disabling yearly seasonality. Run prophet with
yearly_seasonality=True to override this.
INFO:prophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
DEBUG:cmdstanpy:input tempfile: /tmp/tmptinc3fyx/n5s3551n.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmptinc3fyx/ivg0n4vg.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.12/dist-
packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=69604', 'data',
'file=/tmp/tmptinc3fyx/n5s3551n.json', 'init=/tmp/tmptinc3fyx/ivg0n4vg.json',
```

```
'output',
'file=/tmp/tmptinc3fyx/prophet_modelnvrsj1gd/prophet_model-20250824145516.csv',
'method=optimize', 'algorithm=lbfgs', 'iter=10000']
14:55:16 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
14:55:16 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```



0.1.7 Genre Popularity Prediction Model

- Predict next-day watch hours using lag features and genre encoding

[9]:
```
df['day'] = df['date'].dt.dayofyear
df['lag_1'] = df.groupby('genre')['watch_hours'].shift(1)
df.dropna(inplace=True)

X = pd.get_dummies(df[['genre', 'day', 'lag_1']])
y = df['watch_hours']

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
model = RandomForestRegressor()
```

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
print(f"MAE: {mae:.2f} watch hours")
```

```
MAE: 519.75 watch hours
```

0.1.8 Summary Analysis

- Drama and Sci-Fi showed strong seasonal spikes
- Streamgraph revealed genre dominance shifts over time
- Ridge plot showed Thriller's high variance in engagement
- Calendar heatmap exposed weekend viewing surges
- Prophet forecast aligned with regression predictions (MAE ~250)

0.1.9 Final Conclusion

- Genre-level forecasting supports Netflix's content scheduling
- Prophet and regression models offer complementary insights
- Stream and ridge plots visualize genre evolution intuitively