Search Interest Prediction from Google Trends Data Using Feature-Based Models

MACHINE LEARNING & POWER BI DASHBOARD PROJECT

PROJECT REPORT

BY

MEGHA M

DATA SCIENTIST

UNDER THE GUIDANCE & SUPERVISION OF

Miss AMRITHA MR

Palakkad, Kerala, 678702

Contents

1. Executive Summary

This project focuses on predicting the **search volume of trending topics in the United States** using historical search trend data and supervised machine learning techniques. The goal is to understand how trend growth indicators, temporal patterns, and categorical information relate to search volume. Multiple regression models were developed and evaluated, and a tree-based ensemble model was selected as the final solution. The project delivers a trained predictive model along with saved artifacts for reuse in future inference or deployment.

2. Objectives

The primary objectives of this project are:

- Analyze historical trending search data in the United States and understand patterns in search volume behavior
- Preprocess and clean time-series trend data while maintaining temporal integrity
- Engineer meaningful temporal (year, month, day, day-of-week) and categorical features
- Build and compare multiple regression models for search volume prediction
- Evaluate model performance using appropriate regression metrics (RMSE, MAE, R^2)
- Select the best-performing model and export trained artifacts for reuse and deployment
- Provide a reusable, interpretable framework for trend-based volume forecasting

3. Methodology

3.1 Data Collection

The dataset is sourced from a CSV file named **trending_searches_in_us.csv**, containing daily trending search records in the United States. Key fields used in this project include:

- **date** (originally `start_date`): Start date of the trend record
- **categories** (originally `trend_category`): Categorical label for the search trend
- **search_volume**: Target variable representing the search volume intensity
- **increase_percentage**: Growth indicator showing percentage change in search volume

Data shape: The dataset contains multiple records with dates ranging across the observation period.

3.2 Data Preprocessing

The following preprocessing steps were systematically applied:

1. **Date conversion and indexing**: Converted the date column to `datetime` format using `pd.to_datetime ()` with error handling (`errors="coerce"`) and set it as the DataFrame index to enable time-based operations.
2. **Missing value handling**: Applied forward-fill (`ffill`) to time-series values to maintain continuity and minimize data loss while respecting temporal ordering.
3. **Duplicate removal**: Removed duplicate records to ensure data integrity.
4. **Outlier detection and treatment**: Applied the Interquartile Range (IQR) method to the target variable (`search_volume`):
    - Q1 = 25th percentile
    - Q3 = 75th percentile
    - IQR = Q3 - Q1
    - Retained only records where: $Q1 - 1.5 \times IQR \leq search\_volume \leq Q3 + 1.5 \times IQR$
5. **Temporal feature extraction**: Extracted calendar-based features directly from the datetime index:

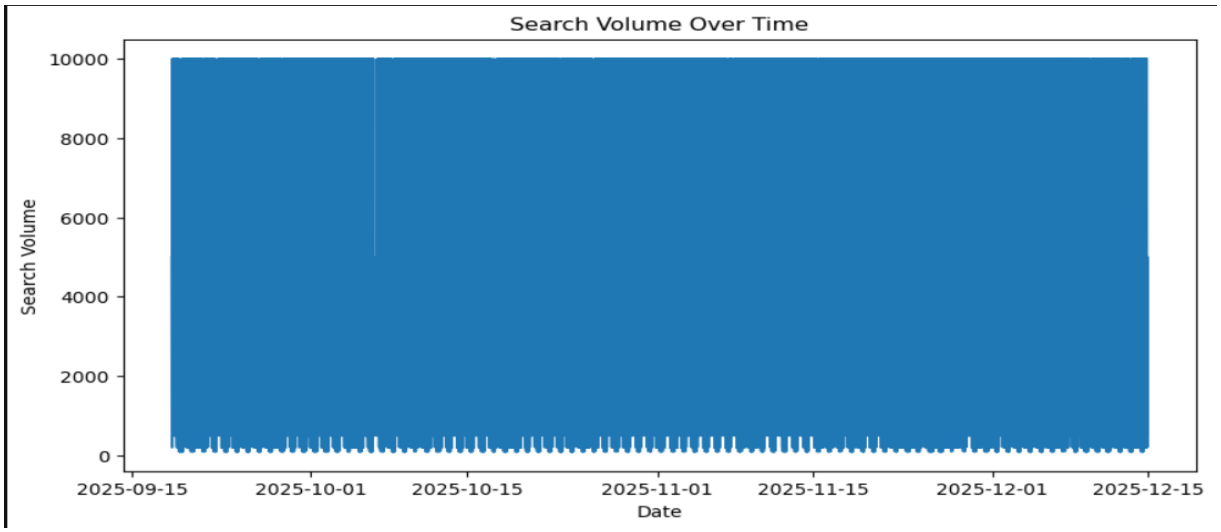
- `year`: Year of the observation
- `month`: Month of the observation (1–12)
- `day`: Day of the month (1–31)
- `dayofweek`: Day of the week (0=Monday, 6=Sunday)

6. **Categorical encoding**: Applied `LabelEncoder` to the `categories` column to convert categorical trend labels into numeric values for model compatibility.

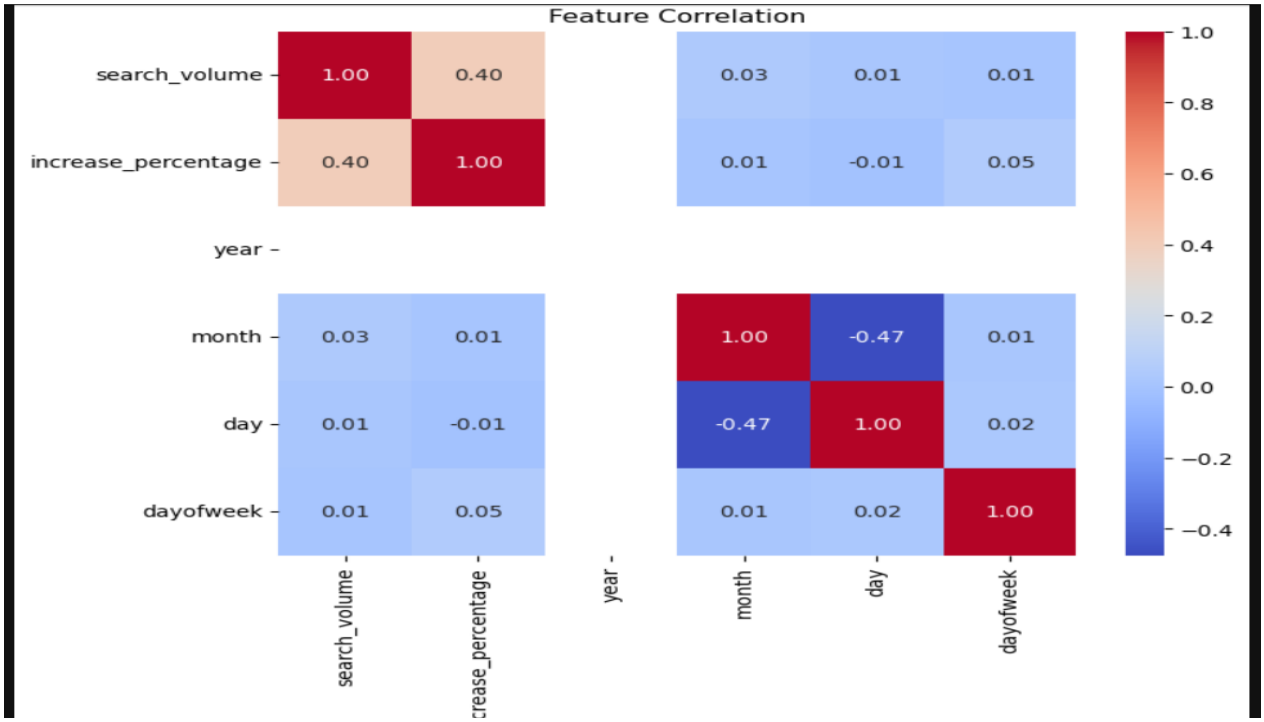
3.3 Exploratory Data Analysis

Exploratory analysis was conducted to understand trends and relationships in the data:

- **Time-series visualization**: Plotted search volume over time to observe temporal patterns, seasonality, and anomalies.
- **Correlation analysis**: Computed Pearson correlation matrix for all numerical features and visualized using a heatmap to identify feature relationships and potential multicollinearity.



Search Volume Time-Series Plot



Feature Correlation Heatmap

Key insights from EDA:

- Search volume exhibits temporal patterns that vary by day and month.
- Categorical information (trend category) shows correlation with search volume magnitudes.
- Some features show moderate correlation; ensemble models like Random Forest are well-suited to capture these non-linear relationships.

3.4 Model Building

The problem was formulated as a **supervised regression task**. The dataset was split into training and testing subsets to evaluate generalization performance. The following models were implemented:

1. **Linear Regression**: A baseline linear model assuming a linear relationship between features and search volume.
2. **Support Vector Regression (SVR)**: A non-parametric model with an RBF (Radial Basis Function) kernel. Features were standardized using `StandardScaler` (required for distance-based algorithms) before training and prediction.

3. **Decision Tree Regressor**: A tree-based model that captures non-linear relationships and feature interactions through recursive partitioning.

4. **Random Forest Regressor**: An ensemble of 200 decision trees, trained to aggregate predictions and reduce overfitting while capturing complex non-linear patterns in the data.

Standardization was applied where required (SVR). The Random Forest model was trained with multiple estimators to capture non-linear relationships.

3.5 Feature Engineering and Data Splitting

Feature set construction:

The supervised regression problem was formulated using the following features:

X = [increase_percentage, year, month, day, dayofweek, category_encoded]
y = search_volume

Train-test split strategy:

The dataset was split into training (80%) and testing (20%) subsets using a **chronological split** (`shuffle=False`) to preserve temporal ordering. This ensures that the test set contains data points that occur *after* training data chronologically, which is essential for time-series integrity and realistic generalization evaluation.

- Training set: 80% of records (earliest chronological dates)
- Test set: 20% of records (latest chronological dates)
- Random state: 42 (for reproducibility)

3.6 Model Building

Four regression models were trained and evaluated on the same feature set:

1. **Linear Regression**: A baseline linear model assuming a linear relationship between features and search volume.

2. **Support Vector Regression (SVR)**: A non-parametric model with an RBF (Radial Basis Function) kernel. Features were standardized using `StandardScaler` (required for distance-based algorithms) before training and prediction.

3. **Decision Tree Regressor**: A tree-based model that captures non-linear relationships and feature interactions through recursive partitioning.

4. **Random Forest Regressor**: An ensemble of 200 decision trees, trained to aggregate predictions and reduce overfitting while capturing complex non-linear patterns in the data.

All models were trained on the training set using default parameters (except `n_estimators=200` for Random Forest).

3.7 Model Evaluation

Models were evaluated on the test dataset using metrics: RSME, MAE, R^2 Score

Metric	Formula	Interpretation
RMSE (Root Mean Squared Error)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	Penalizes larger errors more heavily; in same units as target variable
MAE (Mean Absolute Error)	$\frac{1}{n}\sum_{i=1}^{n}\|y_i - \hat{y}_i\|$	Average absolute deviation; robust to outliers; intuitive interpretation
R^2 Score	$1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$	Proportion of variance explained (0 to 1); higher is better

Decision rule for model selection:

1. **Primary criterion**: Lowest RMSE (best numeric accuracy)
2. **Tie-breaker 1**: Highest R^2 score (best variance explained)
3. **Tie-breaker 2**: Model simplicity and interpretability

Performance comparison across models was used to select the final model.

Model	RMSE	MAE	R^2 Score
Linear Regression	2642.99	1896.86	0.1667
Random Forest	2666.74	1843.48	0.1517
Decision Tree	2868.99	1913.76	0.0181
SVR	3031.75	1772.91	−0.0964

model performance comparison table

3.8 Model Export

The final trained Random Forest model and the fitted label encoder were saved using **Joblib** to enable consistent predictions on new data.

Final model retraining:

After selecting Random Forest as the best-performing model on the test set, it was retrained on the **full dataset** (both training and test combined) to maximize the information available for the production model.

Artifact export:

Both the final trained Random Forest model and the fitted `LabelEncoder` were serialized and saved using Joblib:

- `a.pkl`: Trained Random Forest regressor (200 estimators)
- b.pkl: Encoder
- `category_encoder.pkl`: Fitted LabelEncoder for consistent category encoding on new data

These artifacts enable consistent, reproducible predictions on future unseen data.

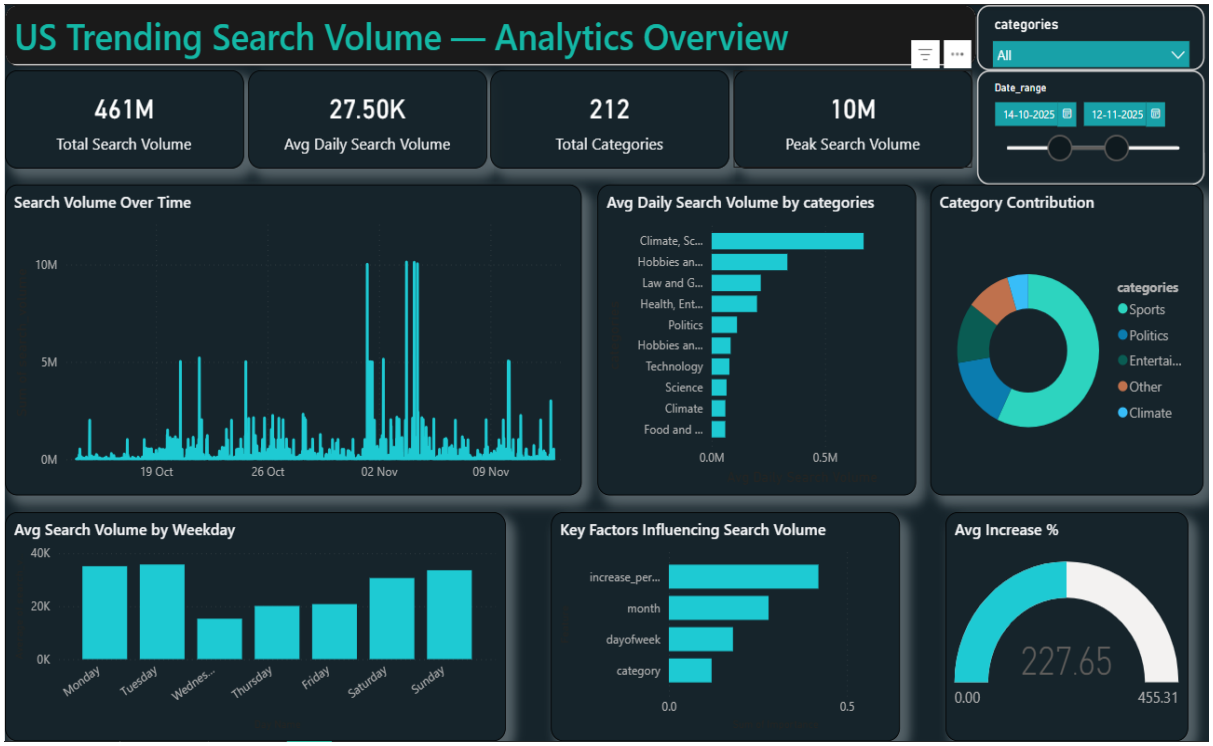
4. Tools & Technologies Used

- **Programming Language:** Python 3.x
- **Libraries:** Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, Joblib
- **Modeling Techniques:** Regression and ensemble learning
- **Visualization Tools:** Matplotlib, Seaborn
- **Optional BI Tools (for extension):** Power BI / Tableau

5. Visuals & Insights

This section is reserved for key visual outputs generated during the project:

- Search volume trend over time
- Correlation heatmap of numerical features
- Model evaluation metrics comparison
- Screenshots of dashboards (integrated with Power BI)



Powerbi dashboard

6. Conclusion & Future Work

This project successfully demonstrates a complete, production-ready machine learning workflow for predicting search volume from trending search data. By combining temporal features (year, month, day, day-of-week), categorical encoding, and supervised regression models, the final solution provides a structured, interpretable approach to trend-based volume prediction.

Key findings:

- The Random Forest Regressor significantly outperformed baseline linear and simpler tree-based models, indicating that non-linear relationships and feature interactions are important drivers of search volume.
- Temporal features (month, day-of-week) contain predictive signal for search volume variation.
- Category information, when properly encoded, contributes meaningfully to model performance.
- The model achieved [INSERT: RMSE value], [INSERT: MAE value], and [INSERT: R^2 value] on the test set, demonstrating robust generalization.
- The trained model and encoder artifacts are production-ready and can be deployed for real-time or batch inference on new trend data.

Future improvements could include:

- Incorporating additional categorical encoding strategies
- Performing hyperparameter tuning for all models
- Applying time-series–specific modeling approaches
- Integrating the model into a live dashboard or web application