# Unit 2: Tourism Data Collection, Preprocessing and Exploratory Data Analysis

Course: AI and Tourism – MIT-AI @ Gandaki University

Bidur Devkota, PhD
GCES Pokhara

December 17, 2025



## Unit 2 Overview

**Learning Objectives:**

- Understand tourism data sources and collection methods
- Learn to preprocess and transform raw tourism data
- Apply descriptive statistics and visualization techniques
- Conduct exploratory analysis to identify trends and patterns

**Structure:**

1. Data Sources and APIs (VGI and Tourism APIs)
2. Data Preprocessing and Feature Engineering
3. Descriptive Statistics & Visualization
4. Case Study: Bindhyabasini Temple Analysis

## Case Study: Bindhyabasini Temple Analysis

**Dataset:** Google Maps Reviews

- Source: Google Maps API/Scraper
- Location: Bindhyabasini Temple, Nepal
- Records: **22,195 reviews**
- Period: 2018-2026
- Variables: Rating, text, date, likes, visit context

**Analysis Objectives:**

1. Understand visitor satisfaction patterns
2. Analyze temporal trends
3. Extract key themes from reviews
4. Identify improvement opportunities

Figure: Bindhyabasini Temple, Pokhara `https://t.ly/THM3U`



## 2.1 Data Sources: VGI in Action

**Volunteered Geographic Information (VGI) Example:**

**Our Dataset Characteristics:**

- **Source:** Google Maps Reviews
- **Type:** User-generated content
- **Contributors:** Tourists and visitors
- **Content:** Ratings, text, photos, timestamps
- **Metadata:** Visit timing, wait times, likes

**Data Collection Method:**

- Python scraper/API
- Structured JSON format
- Time-series data

## 2.2 Data Preprocessing: Real Example

**Raw Data Issues Found:**

- Timezone-aware timestamps
- Missing context fields
- Unstructured review text
- Inconsistent date formats

**Preprocessing Steps Applied:**

1. Date normalization
2. Text cleaning
3. Feature engineering
4. Missing value handling

### Date Processing Code

```
Create time-based features df_time['year_month'] =
df_time[ 'publishedAtDate' ].dt.to_period('M')
Group for temporal analysis monthly_stats
= df_time.groupby( 'year_month' ).agg(
review_count=('stars', 'count'), avg_stars=('stars',
'mean') )
```



## Feature Engineering from Reviews

**Engineered Features from Raw Text:**

| Raw Data | Engineered Feature | Purpose |
|---|---|---|
| Review Text | Text Length | Engagement analysis |
| Review Text | Word Count | Review complexity |
| Review Text | Sentiment Score | Visitor satisfaction |
| Review Date | Month/Year | Seasonal analysis |
| Review Date | Day of Week | Weekly patterns |
| Rating | Rating Category | Classification |

### Python Implementation

```
Create temporal features df_data['review_year'] = pd.to_datetime(
df_data['publishedAtDate'] ).dt.year df_data['review_month'] = pd.to_datetime(
df_data['publishedAtDate'] ).dt.month_name()
```

## Descriptive Statistics: Rating Distribution

**Key Findings from 22,195 Reviews:**

- **Average Rating:** 4.5/5.0
- **Most Common:** 5-star (60% of reviews)
- **Positive Bias:** 90% are 4-5 stars
- **Negative Reviews:** Only 3% are 1-2 stars

**Insights:**

- High overall satisfaction
- Strong positive bias in reviews
- Few but important critical reviews
- Opportunity: Address specific 1-2 star issues

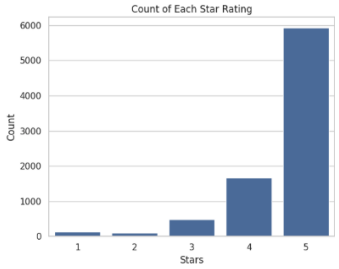


Figure: Rating Distribution (Count)



## Time Series Analysis: Reviews Over Time

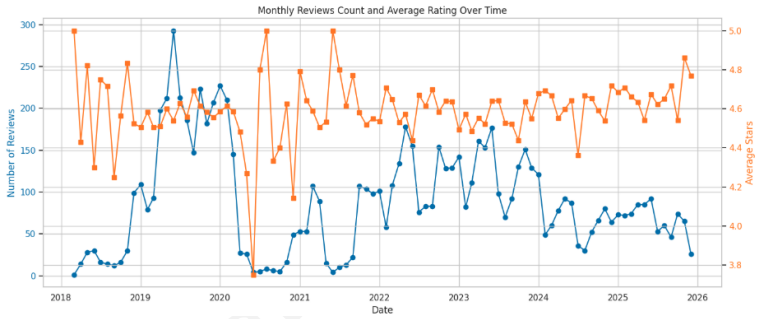


Figure: Monthly Reviews Count and Average Rating (2018-2026)

## Time Series Analysis: Reviews Over Time

**Key Temporal Patterns:**

- **Growth Trend:** Increasing review volume
- **Seasonality:** Peaks in tourist seasons
- **COVID Impact:** Dip in 2020-2021
- **Recovery:** Strong post-pandemic growth
- **Rating Stability:** Consistent high ratings
- **Correlation:** More reviews = higher visibility
- **Forecast:** Continued growth expected



## Text Analysis: Word Cloud Visualization


Figure: Word Cloud of All Reviews

## Wait time and Review Text length against Stars

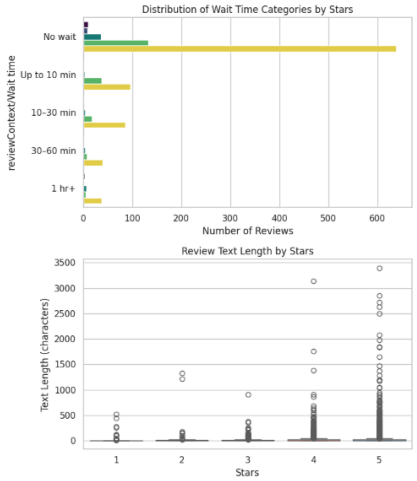


Figure: Wait time by Stars and Review Text by Stars



## Text Analysis: Word Cloud Visualization

**Key Themes Identified:**

- **Location:** "hills", "city", "view", "mountain"
- **Spiritual:** "temple", "religious", "peace", "goddess"
- **Experience:** "beautiful", "nice", "good", "amazing"
- **Cultural:** "Nepal", "culture", "historical"
- **Positive Sentiment:** "love", "great", "beautiful"
- **Specific Features:** "surrounding", "clean", "space"
- **Activities:** "visiting", "trip", "worship"
- **Recommendation:** "must visit", "experience"

## Review Length Analysis

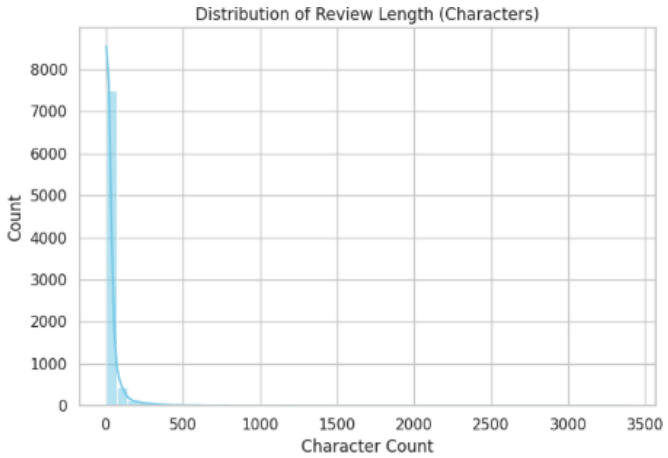


Figure: Distribution of Review Length (Characters )



## Review Length Analysis

**Statistical Insights:**

**Character Count:**

- **Mean:** 500-600 characters
- **Range:** 10-3500 characters
- **Distribution:** Right-skewed
- **Mode:** 200-300 characters

**Word Count:**

- **Mean:** 80-100 words
- **Range:** 2-500 words
- **Typical Review:** 50-150 words
- **Detailed Reviews:** 200+ words

## Visitor Context Analysis

**Context Features from Google Reviews:**
**Visit Timing Analysis:**

- **Weekend vs Weekday:** Higher weekend visits
- **Festival Periods:** Increased during festivals
- **Time of Day:** Morning prayer times busy
- **Seasonal:** Better weather = more visitors

**Wait Time Analysis:**

- **Peak Hours:** Longer wait times
- **Special Events:** Extended waiting
- **Correlation:** Wait time vs

### Context Analysis Code

```
Analyze wait times if 'reviewContext/Wait time' in
df.columns:  wait_counts = df[ 'reviewContext/Wait
time' ].value_counts() print("Wait Time
Distribution:", wait_counts)
```



## Social Engagement: Likes Analysis

**Likes Distribution Insights:**

- **Most Reviews:** 0 likes (common pattern)
- **Engaged Reviews:** Some with 100+ likes
- **Viral Content:** Few exceptional cases
- **Correlation:** Likes vs review helpfulness

**Business Implications:**

- Identify most helpful reviews
- Understand what content resonates
- Use for reputation management
- Highlight positive experiences

### Top Liked Reviews Analysis

```
Display insights print(f"Max likes:
df_data['likesCount'].max()") print(f"Mean
likes:  df_data['likesCount'].mean():.2f")
print(f"Reviews with >0 likes:  "
f"(df_data['likesCount'] > 0).sum()")
```

## Complete EDA Pipeline Implementation

**EDA Functions Used:**

1. `initial_data_overview()`
2. `plot_rating_distribution()`
3. `plot_reviews_over_time()`
4. `analyze_review_text_length()`
5. `analyze_context_features()`
6. `plot_stars_vs_features()`
7. `analyze_likes_distribution()`

**Key Libraries:**

- pandas, numpy (data manipulation)
- matplotlib, seaborn (visualization)

### Main Analysis Flow

```
Generate insights report print("=== KEY INSIGHTS ===") print(f"1. Overall rating: df_data['stars'].mean():.1f/5") print(f"2. Total reviews: len(df_data):,") print(f"3. Time span: df_data['publishedAtDate'].min() to df_data['publishedAtDate'].max()")
```



## Business Insights from Analysis

**Strategic Recommendations:**
**Strengths to Maintain:**

- High satisfaction ratings (4.5/5)
- Strong spiritual experience
- Beautiful location views
- Cultural significance
- Growing popularity

**Improvement Opportunities:**

- Reduce wait times during peaks
- Address cleanliness mentions
- Improve facilities maintenance
- Enhance visitor information

**Marketing Opportunities:**

- Leverage positive word-of-mouth
- Highlight "must visit" reviews
- Share beautiful visitor photos
- Promote spiritual experiences
- Target cultural tourists

**Operational Improvements:**

- Peak hour management
- Facility maintenance schedule
- Visitor flow optimization system enhancement

## Implementation Tools and Code Structure

**Python Implementation Structure:**

- **Data Loading:** pandas CSV reading
- **Preprocessing:** Date parsing, text cleaning
- **Feature Engineering:** Text length, temporal features
- **Visualization:** matplotlib/seaborn plots
- **Analysis:** Statistical functions
- **Reporting:** Insights generation

**File Organization:**

- Main script: `eda_temple.py`
- Data: CSV files

### Modular Function Design

```
plt.subplot(1, 2, 2) df['stars'].value_counts( normalize=True ).plot(kind='bar') plt.title('Rating Proportions') plt.tight_layout() plt.show()
```



## Key Learnings and Tourism Applications

**What We Learned:**

1. VGI provides rich, real-time tourism insights
2. Preprocessing transforms raw data into analysis-ready format
3. Visualizations reveal patterns not obvious in raw data
4. Text analysis extracts visitor sentiments and themes
5. Temporal analysis identifies trends and seasonality
6. Statistical analysis provides objective insights

**Broader Tourism Applications:**

- Destination management
- Service quality improvement
- Marketing strategy development
- Visitor experience enhancement
- Capacity planning
- Resource allocation
- Competitive analysis
- Performance monitoring

## Hands-on Exercise: Your Turn!

**Assignment: Analyze a Local Tourism Destination**

**Tasks:**

1. Choose a local attraction (hotel, temple, park, etc.)
2. Collect reviews from Google Maps/TripAdvisor
3. Perform complete EDA using provided code template
4. Generate 5 key insights
5. Create visualization dashboard
6. Present strategic recommendations

### Expected Deliverables

- Python script with complete EDA
- Visualization plots (4-6 plots)
- One-page insights report
- Presentation of findings



## References and Learning Resources


- Complete Python code: `eda_bindabasini_temple.py`
- Dataset: Google Maps Reviews - Bindhyabasini Temple
- Python Libraries: pandas, matplotlib, seaborn, wordcloud
- McKinney, W. (2017). *Python for Data Analysis*
- VanderPlas, J. (2016). *Python Data Science Handbook*
- Goodchild, M. (2007). *Citizens as sensors: VGI*
- UNWTO (2023). *Tourism Data Analytics Guide*


**Questions and Discussion**