Web Scraping and Analysis of Dubizzle Egypt Property Listings

1. Introduction

This project extracts, processes, analyzes, and visualizes real estate advertisement data from Dubizzle Egypt (https://www.dubizzle.com.eg/en/properties/apartments-duplex-for-sale/), a leading online marketplace for property listings in Egypt. The goal is to provide comprehensive insights into the Egyptian real estate market by collecting data on apartments and duplexes, including attributes like price, location, area, bedrooms, and amenities. The project employs web scraping, data cleaning with regular expressions, statistical analysis, visualization, and database storage, culminating in an interactive Streamlit web application to present the findings dynamically.

This project aligns with the DS Tools course objectives by demonstrating proficiency in Python libraries for web scraping (`requests`, `BeautifulSoup`), data processing (`pandas`, `re`), analysis (`numpy`, `sklearn`), visualization (`matplotlib`, `seaborn`, `plotly`), and database management (`pymongo`). The results offer valuable market insights for buyers, investors, and researchers.

2. Project Objectives

- **Data Extraction**: Scrape property listing data from Dubizzle Egypt and store it in a structured CSV file.
- **Data Cleaning and Processing**: Standardize and clean data using regular expressions to ensure quality.
- **Data Analysis**: Compute statistics, identify trends, and explore relationships between variables.
- **Data Visualization**: Create diverse visualizations (e.g., bar charts, histograms, heatmaps) to present findings.
- **Data Storage**: Store processed data in a MongoDB database.
- **Bonus Task**: Deploy an interactive Streamlit web app for dynamic exploration of the data.

3. Methodology

3.1 Data Extraction

- **Target Website**: Dubizzle Egypt, specifically the apartments and duplexes for sale section.
- **Tools**: `requests`, `BeautifulSoup`.
- **Process**:
  - Scraped paginated listing pages (`?page={}`) to collect unique advertisement URLs from `<div class="_637fa00f">` elements containing `<a>` tags.
  - Removed duplicates using `set` to ensure uniqueness.
  - For each URL, sent HTTP requests with a `User-Agent` header (`Mozilla/5.0`) and 10-second timeout, parsing HTML with `BeautifulSoup`.
  - Extracted 20 data points per listing: advertisement link, title, price, down payment, location, creation date, property type, ownership, area, bedrooms, bathrooms, furnished status, level, payment option, completion status, amenities, ad ID, seller type, seller name, and seller member since.
  - Handled errors by assigning `Not Available` for missing fields and `Error` for extraction failures, with checks for HTTP 403 errors to halt scraping if blocked.
  - Implemented a 1-second delay between requests to avoid website blocking.
  - Saved backups every 5 entries to `backup.csv` to protect against interruptions.
- **Output**: `Dubizzle_properties.csv` with 4966 listings and 20 columns, encoded in `utf-8-sig` to support Arabic text.

3.2 Data Cleaning and Processing

- **Tools**: `pandas`, `numpy`, `re`.
- **Steps**:
  - Replaced `Not Available` and `Error` with `pd.NA`.
  - Dropped rows with all columns (except the first) missing.
  - Used regular expressions:
    - Price: `r'EGP\s*([\d,]+)'` to extract numbers (e.g., `EGP 1,000,000` → `1000000`).
    - Down Payment: `r'[^\d]'` to extract numbers or `0` for `0%` cases.
    - Area: `r'(\d+)\s*m²'` to extract numbers (e.g., `120 m²` → `120`).
    - Creation Date: Parsed relative dates (e.g., `2 days ago`) using `re.search(r'(\d+)', str(date_str))` and converted to actual dates.
    - Location: Split into `area_name` and `city` using `r'^(.*?),\s*(.*)$'` (e.g., `Madinaty, Cairo` → `Madinaty`, `Cairo`).
    - Seller Member Since: Extracted year with `r'\b(\d{4})\b'` (e.g., `Member since Feb 2025` → `2025`).
  - Handled missing values:
    - `down_payment`: Filled with `0`.
    - `area`: Filled with mean area per property type.
    - `furnished`: Filled with mode (most common value).
    - `payment_option`: Inferred from `down_payment` (Cash if `0`, Installment if > `0`, else Cash or Installment).
  - Converted columns to appropriate types: `price`, `down_payment`, `area`, `bedrooms`, `bathrooms` to numeric; `creation_date` to datetime.

- Dropped `level` column as it was not critical for analysis.
- Added derived columns:
  - `amenities_count_full`: Number of amenities per listing.
  - Boolean columns: `has_garden`, `has_security`, `has_pool`, `has_balcony`, `has_parking`, `has_elevator`.
  - `price_per_sqm`: Price per square meter (`price / area`).

- **Output**: Cleaned DataFrame with standardized data, saved for analysis and storage.

3.3 Data Analysis

- **Tools**: `pandas`, `numpy`, `sklearn`.
- **Descriptive Statistics**:
  - 4966 listings with 21 columns (after adding derived columns).
  - Unique values: 1440 prices, 556 locations, 8 property types (Apartment: most common, ~80%), 4 ownership types (Primary: ~60%).
  - Most frequent location: Madinaty, Cairo (322 listings).
  - Most common creation date: Listings from 2 days ago (high recency).
- **Trends and Patterns**:
  - Price distribution is right-skewed, with outliers at high prices (e.g., luxury apartments).
  - High-demand areas: Madinaty, New Cairo, 6th of October.
  - Cash payments dominate (~50%), followed by installments.
  - Ready properties (~70%) are more common than under-construction.
- **Relationships**:
  - Strong correlation between price and area (~0.7).
  - Moderate correlation between price and bedrooms/bathrooms (~0.4–0.5).
  - Amenities like security and parking increase prices, especially for apartments.
  - Feature importance (via `RandomForestRegressor`): Area, bedrooms, and location are top price drivers.
- **Output**: Statistical summaries, correlation matrices, and feature importance rankings.

3.4 Data Visualization

- **Tools**: `matplotlib`, `seaborn`, `plotly`, `wordcloud`.
- **Visualizations** (implemented in both analysis script and Streamlit app):
  - **Price Analysis**:
    - Histogram and boxplot of price distribution.
    - Scatter plots: Price vs. down payment, price vs. area.
  - **Location Analysis**:
    - Bar charts: Top 10 most/least expensive areas, top 10 areas by listing count.
  - **Property Features**:
    - Boxplots: Price by property type, bedrooms, amenities count, completion status, payment option, ownership.
    - Scatter plots: Bedrooms vs. area by property type.
  - **Market Trends**:

        - Line plots: Listings per day, average price over time.
    - **Additional Insights**:
        - Pie charts: Ownership type, furnished status.
        - Bar charts: Property type, completion status.
        - Heatmaps: Correlations, bedrooms/bathrooms vs. property type, amenities impact on price.
        - Word cloud: Seller names (with Arabic text support via `arabic_reshaper`).
        - Bar chart: Feature importance for price prediction.
    - **Streamlit Enhancements**:
        - Interactive Plotly charts with zoom and hover.
        - Downloadable PNG files for each visualization.
- **Output**: Over 20 visualizations saved as images and embedded in the Streamlit app.

3.5 Data Storage

- **Database**: MongoDB.
- **Tools**: `pymongo`.
- **Process**:
    - Connected to a local MongoDB instance (`mongodb://localhost:27017/`).
    - Converted cleaned DataFrame to a list of dictionaries.
    - Inserted records into `dubizzle_db.properties` collection.
    - Verified storage by retrieving and printing 5 sample records.
- **Output**: MongoDB collection with 4966 documents, each representing a property listing.

3.6 Streamlit Web Application (Bonus Task)

- **Tools**: `streamlit`, `plotly`, `sklearn`.
- **Features**:
    - **Filters**: Search by area/title, select city, property type, price range, bedrooms, bathrooms, furnished status, payment option, amenities, and date range.
    - **Key Market Metrics**: Displayed in styled cards (average price, total listings, price per m², average area) with comparisons to overall data.
    - **Price Prediction**: `RandomForestRegressor` model to estimate prices based on city, area, bedrooms, and bathrooms.
    - **Market Insights**:
        - Key Findings: Most expensive/cheapest areas, average price per m², most common property type, average bedrooms.
        - Market Trends: Price trend (increasing/decreasing), listings growth, average days on market.
        - Fun Facts: Most expensive area, cheapest area, most common amenity (e.g., security).
    - **Visualizations**: Interactive charts across tabs (Price Analysis, Location Analysis, Property Features, Market Trends, Additional Insights).
    - **Data Table**: Filtered listings with sorting and CSV download.

- **Deployment**: Runs locally via `streamlit run app.py`, with caching (`@st.cache_data`) for performance.
- **Output**: Fully interactive web app with user-friendly interface and exportable results.

---

4. Findings and Insights

- **Market Overview** (based on 4966 listings):
  - Average price: ~EGP 7,562,613 .
  - Total listings: 4966, concentrated in Cairo (e.g., Madinaty: 322 listings).
  - Average price per m²: ~EGP 48,253.
  - Average area: ~163 m² .
- **Key Trends**:
  - Prices are higher in premium areas like New Cairo and Zamalek.
  - Listings growth: Positive, with new ads daily.
- **Fun Facts**:
  - Most expensive area: Marassi, ~EGP 65,875,000.
  - Cheapest area: Al Hanouvel, ~EGP 325,000.
  - Most common amenity: Security.
- **Key Relationships**:
  - Area is the strongest predictor of price, followed by bedrooms and location.
  - Amenities like pools and gardens add ~10–20% to apartment prices.
  - Cash payments are associated with lower prices compared to installments.
- **Market Insights**:
  - Apartments dominate the market (~80%), reflecting demand for affordable housing.
  - Ready properties are preferred, indicating buyer preference for immediate occupancy.
  - Madinaty and New Cairo are high-demand areas due to modern amenities and infrastructure.

---

5. Challenges and Solutions

- **JavaScript-Driven Content**:
  - **Challenge**: Listings page uses JavaScript, making static scraping incomplete.
  - **Solution**: Used a two-phase pipeline (scrape URLs, then details) instead of Selenium, which was slower.
- **Duplicate Links**:
  - **Challenge**: Pagination caused duplicate URLs.
  - **Solution**: Removed duplicates using `set`.
- **Network Interruptions**:
  - **Challenge**: Scraping was prone to timeouts.

    - **Solution**: Set 10-second timeout and saved backups every 5 entries.
- **Website Blocking**:
    - **Challenge**: Frequent requests risked blocking.
    - **Solution**: Added `User-Agent` and 1-second delay; monitored for 403 errors.
- **Dynamic HTML Structure**:
    - **Challenge**: Varying HTML classes broke selectors.
    - **Solution**: Used flexible selectors (e.g., `EGP` for price, dynamic class matching for seller name).
- **Missing Data**:
    - **Challenge**: Fields like down payment were often missing.
    - **Solution**: Assigned `Not Available`/`Error` during scraping; imputed logically during cleaning (e.g., `0` for down payment).
- **Arabic Text Handling**:
    - **Challenge**: Arabic text caused encoding issues.
    - **Solution**: Used `utf-8-sig` for CSV and ensured MongoDB Unicode support.
- **Streamlit Performance**:
    - **Challenge**: Rendering large datasets slowed the app.
    - **Solution**: Used caching (`@st.cache_data`) and optimized data filtering.

---

6. Conclusion

This project successfully built an end-to-end pipeline for analyzing Dubizzle Egypt's real estate listings. From scraping 4966 listings to cleaning, analyzing, and visualizing the data, the project uncovered key market trends, such as the dominance of apartments, high demand in areas like Madinaty, and the impact of area and amenities on pricing. The MongoDB database ensures data persistence, while the Streamlit app provides an intuitive interface for exploring insights and predicting prices. The project demonstrates proficiency in data science tools and offers a scalable framework for future real estate analyses.

Future enhancements could include:

- Real-time data updates via scheduled scraping.
- Expanding to other property types (e.g., villas).
- Deploying the Streamlit app on a public platform like Streamlit Cloud.

---

7. Deliverables

- **Code**:
    - `Dubizzle_Scrapping.ipynb`: Web scraping script.
    - `Dubizzel_Analysis.ipynb`: Data cleaning, analysis, and visualization script.
    - `app.py`: Streamlit web app.

- **Raw Data**: `Dubizzle_properties.csv` (4966 listings).
- **Processed Data**: MongoDB collection (`dubizzle_db.properties`).
- **Visualizations**: Over 20 charts (PNG files via analysis script and Streamlit downloads).
- **Report**: This document, summarizing methodology, findings, and insights.
- **Streamlit App**: Local app with interactive filters, charts, and price prediction.

---

8. References

- Dubizzle Egypt: https://www.dubizzle.com.eg
- Python Libraries: `requests`, `beautifulsoup4`, `pandas`, `numpy`, `matplotlib`, `seaborn`, `plotly`, `streamlit`, `pymongo`, `scikit-learn`, `wordcloud`, `arabic_reshaper`.
- MongoDB Documentation: https://www.mongodb.com/docs/
- Streamlit Documentation: https://docs.streamlit.io/
