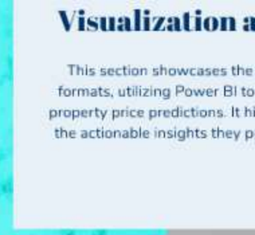
Bayut Project: Predicting Property Prices

Data Scraping, Machine Learning, and Visualization with Power BI



Introduction to the Bayut Project

The Bayut Project involves scraping data from the Bayut real estate website, cleaning the data, developing a machine learning model to predict property prices, and creating a visualization dashboard in Power BI. This project highlights the transformative power of data in understanding real estate market dynamics.

Project Overview

The Bayut Project focuses on utilizing web scraping techniques to gather extensive data from the Bayut platform, which lists various real estate properties. By systematically cleaning and analyzing this data, insights into property price predictions were facilitated using machine learning methodologies.

Objectives

The primary objective of the project is to build a predictive model for property prices based on comprehensive data analysis. By leveraging machine learning, the goal is to enhance decision-making for buyers, sellers, and real estate professionals through accurate price forecasts.

Importance of Price Prediction

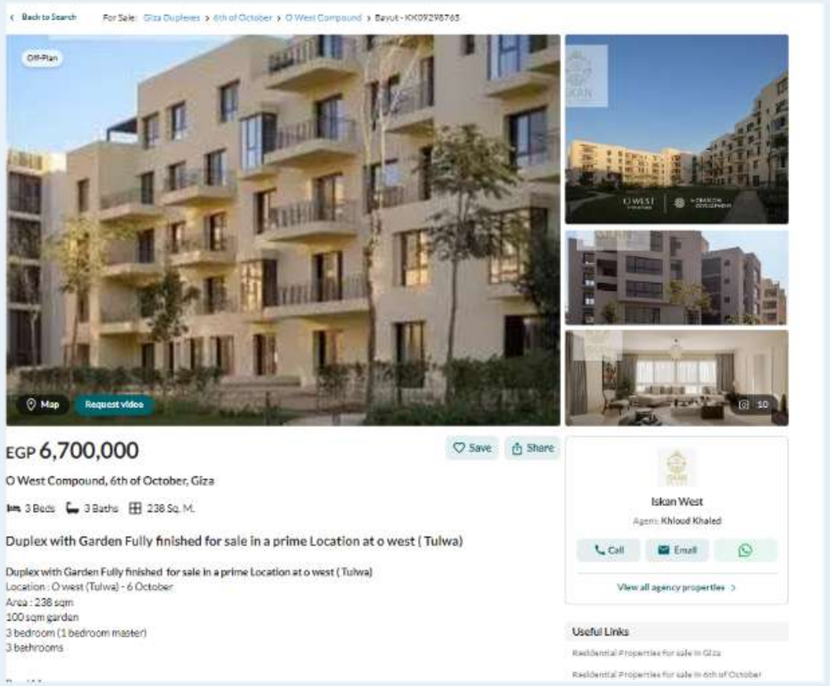
Accurate price prediction in real estate is crucial for effective investment strategies and market analysis. It enables stakeholders to make informed decisions, adapt to market fluctuations, and understand property valuation trends, thus enhancing investment security and profitability.

Data Collection Techniques for Bayut

The data collection phase involved scraping data from the Bayut website, a comprehensive online real estate platform in Egypt. Rigorous techniques were employed to ensure accurate and high-quality data extraction from diverse property listings.

Overview of the Bayut Website

Bayut is a leading real estate portal in the UAE, featuring thousands of listings for residential and commercial properties. The platform provides detailed property information, including prices, features, and proximity to amenities, making it a valuable resource for market analysis.



Data Scraping Techniques

Web scraping involved extracting data using automated scripts to gather listings, prices, and metadata from Bayut's pages. Techniques included using libraries such as BeautifulSoup for HTML parsing, requests for fetching the web pages, and ThreadPoolExecutor for concurrent execution to speed up the scraping process. The data was then organized and saved using pandas.

Code Snipets

The code starts by defining a list of URLs to scrape data from the Bayut website. It uses the requests library to fetch the web pages and BeautifulSoup to parse the HTML and extract relevant data such as property details and prices. The ThreadPoolExecutor is employed to run multiple scraping tasks concurrently, speeding up the data collection process. Finally, the extracted data is organized into a pandas DataFrame and saved to a CSV file for further analysis.

```python
import requests
from bs4 import BeautifulSoup
from concurrent.futures import ThreadPoolExecutor

page_links = []

pages_links = ["https://www.bayut.eg/%D8%B9%D9%82%D8%A7%D8%B1%D8%A7%D8%AA-%D

for i in range(2, 2084):
    url = f"https://www.bayut.eg/صفحة-{i}/مصر/عقارات-للبيع/?gclid=CjwKCAjwgfm3Bh
    pages_links.append(url)

def scrape_page(url):
    try:
        response = requests.get(url )
        soup = BeautifulSoup(response.content, 'html.parser')

        # Extract property links
        ul_element = soup.find('ul', class_='e20beb46')
        if ul_element:
            li_elements = ul_element.find_all('li')
            for li in li_elements:
                a_tag = li.find('a', href=True)
                if a_tag:
                    full_url = "https://www.bayut.eg" + a_tag['href']
                    page_links.append(full_url)
    except Exception as e:
        print(f"Error scraping {url}: {e}")

with ThreadPoolExecutor(max_workers=10) as executor:
    executor.map(scrape_page, pages_links)
```



```python
cpan_text = soup.find('span', class_='cpan _41163454')
cpan_text = cpan_text.get_text(strip=True) if cpan_text else None

# Extract details in ul
ul_element = soup.find('ul', class_='_3dc8d08d')
details = {}
if ul_element:
    li_elements = ul_element.find_all('li')
    for idx, li in enumerate(li_elements):
        detail_label = li.get_text(strip=True)
        span = li.find('span', class_='_2fdf7fc5')
        detail_value = span.get_text(strip=True) if span else None
        if idx == 0:
            details['نوع العقار'] = detail_value
        elif idx == 1:
            details['نوع العرض'] = detail_value
        elif idx == 2:
            details['الرقم المرجعي'] = detail_value
        elif idx == 3:
            details['حالة البناء'] = detail_value
        elif idx == 4:
            details['التأثيث'] = detail_value
        elif idx == 5:
            details['تاريخ الإضافة'] = detail_value
        elif idx == 6:
            details['الملكية العقارية'] = detail_value

# Extract company name (Company providing the property)
company = soup.find('span', class_='_02db0128')
company_name = company.get_text(strip=True) if company else None

# Extract real estate agent (وكيل العقار)
agent = soup.find('span', class_='d8185451')
agent_name = agent.get_text(strip=True) if agent else None
```


```python
cpan_text = soup.find('span', class_='cpan _41163454')
cpan_text = cpan_text.get_text(strip=True) if cpan_text else None

# Extract details in ul
ul_element = soup.find('ul', class_='_3dc8d08d')
details = {}
if ul_element:
    li_elements = ul_element.find_all('li')
    for idx, li in enumerate(li_elements):
        detail_label = li.get_text(strip=True)
        span = li.find('span', class_='_2fdf7fc5')
        detail_value = span.get_text(strip=True) if span else None
        if idx == 0:
            details['نوع العقار'] = detail_value
        elif idx == 1:
            details['نوع العرض'] = detail_value
        elif idx == 2:
            details['الرقم المرجعي'] = detail_value
        elif idx == 3:
            details['حالة البناء'] = detail_value
        elif idx == 4:
            details['التأثيث'] = detail_value
        elif idx == 5:
            details['تاريخ الإضافة'] = detail_value
        elif idx == 6:
            details['الملكية العقارية'] = detail_value

# Extract company name (Company providing the property)
company = soup.find('span', class_='_02db0128')
company_name = company.get_text(strip=True) if company else None

# Extract real estate agent (وكيل العقار)
agent = soup.find('span', class_='d8185451')
agent_name = agent.get_text(strip=True) if agent else None
```

Data Cleaning

Data cleaning is a critical step that ensures accuracy and integrity in the predictive modeling process. By addressing inconsistencies and errors, the quality of the machine learning model improves significantly.

Raw Data Overview

The initial dataset obtained from Bayut comprised thousands of property listings, including features such as location, size, price, and amenities. However, this raw data presented challenges due to inconsistencies and inaccuracies requiring immediate attention to ensure reliable analysis.

Data Cleaning for Property Listings

Rooms

```
In [ ]: bayot['Rooms'] = bayot['Rooms'].str.replace('غرف','',regex=False)
        bayot['Rooms'] = bayot['Rooms'].str.replace('غرفة','',regex=False)
        bayot['Rooms'] = bayot['Rooms'].str.replace('١','',regex=False)
        bayot['Rooms'] = bayot['Rooms'].str.replace('1' ,'استوديو',regex=False)
        bayot['Rooms'] = bayot['Rooms'].str.replace('حمام', '',regex=False)
        bayot['Rooms'] = bayot['Rooms'].str.replace('حمامات', '',regex=False)
        bayot['Rooms'] = bayot['Rooms'].str.strip()

In [ ]: bayot = bayot[~bayot['Rooms'].str.contains('متر مربع', na=False)]

In [10]: bayot['Rooms'] = bayot['Rooms'].str.replace('ات', '', regex=False)

In [72]: bayot['Rooms'] = bayot['Rooms'].astype(str).str.strip().str.normalize('NFKC')
         bayot = bayot[bayot['Rooms'] != 'بيوت - QjOM2g-202697308']

In [11]: bayot['Rooms'] = bayot['Rooms'].astype('float')
```

Bathrooms

```
In [28]: bayot['Bathrooms'] = bayot['Bathrooms'].str.replace('94 2','متر مربع',regex=False)
         bayot['Bathrooms'] = bayot['Bathrooms'].str.replace('54 2','متر مربع',regex=False)

In [29]: bayot['Bathrooms'] = bayot['Bathrooms'].str.replace('حمام', '',regex=False)
         bayot['Bathrooms'] = bayot['Bathrooms'].str.replace('حمامات', '',regex=False)
         bayot['Bathrooms'] = bayot['Bathrooms'].str.strip()

In [35]: bayot['Bathrooms'] = bayot['Bathrooms'].str.replace(r'.*1' ,'متر مربع.*', regex=True)

In [31]: bayot['Bathrooms'] = bayot['Bathrooms'].str.replace('ات', '',regex=False)

In [68]: bayot['Bathrooms'] = bayot['Bathrooms'].astype('float')
```

Area

```
In [21]: bayot['Area'] = bayot['Area'].str.replace('متر مربع','',regex=False)

In [22]: bayot['Area'] = bayot['Area'].str.replace(',','',regex=False)

In [23]: bayot['Area'] = bayot['Area'].astype('float')

In [24]: median_area = bayot.Area.median()

In [25]: median_area
Out[25]: 165.0

In [26]: bayot.Area.fillna(median_area, inplace=True)
```

This code snippet focuses on cleaning and standardizing the data in the bayot DataFrame, specifically for the columns Price, Rooms, Bathrooms, and Area. It removes commas and specific words like “villa” and “apartment” from the Price values, and cleans the Rooms column by removing words such as “rooms”, “room”, and “bathroom”, replacing “studio” with “1”, and stripping any whitespace. For the Bathrooms column, it removes “bathroom” and “bathrooms”, strips whitespace, and corrects misplaced area values. The Area column is cleaned by removing “square meters”, converting values to float, and removing commas. This process ensures the data is consistent and ready for analysis and modeling.

Rename columns & Translate them

In [93]:
```
column_translation = {
    'نوع العقار': 'Property Type',
    'نوع العرض': 'Offer Type',
    'الرقم المرجعي': 'Reference Number',
    'حالة البناء': 'Building Status',
    'التأثيث': 'Furnishing',
    'تاريخ الإضافة': 'Date Added',
    'الملكية العقارية': 'Ownership'
}
```

In [94]:
```
bayot.rename(columns=column_translation, inplace=True)
```

In [12]:
```
threshold = len(bayot.columns) / 2
```

In [13]:
```
bayot = bayot.dropna(thresh=threshold)
```

In [23]:
```
def split_location(location):
    if pd.isna(location):  # Check if the value is NaN or missing
        return pd.Series([np.nan, np.nan, np.nan])

    location = str(location)
    parts = location.split('،')

    if len(parts) == 3:
        compound, state, city = parts[0].strip(), parts[1].strip(), parts[2].strip()
    elif len(parts) == 2:
        compound, city = parts[0].strip(), parts[1].strip()
        state = np.nan

    elif len(parts) == 1:
        compound = parts[0].strip()
        state = city = np.nan
    else:
        compound = state = city = np.nan

    return pd.Series([compound, state, city])

bayot[['Compound Name', 'State', 'City']] = bayot['Location'].apply(split_location)

bayot = bayot.drop(columns='Location')

print(bayot)
```

Renaming Columns and Translation & Handling Nulls

Normalization was applied to standardize the numerical ranges of features, ensuring that each one contributes equally to the model. This process is crucial for algorithms sensitive to the scale of input data, improving the model's effectiveness.

Nulss And Duplicates

```
In [12]: threshold = len(bayot.columns) / 2

In [13]: bayot = bayot.dropna(thresh=threshold)

In [27]: duplicates = bayot.duplicated(subset=['Unit Link'])
         duplicate_rows = bayot[duplicates]
         print(duplicate_rows)
```

```
                                                Unit Link  \
1219   https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
1239   https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
1591   https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
1598   https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
1909   https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
...                                                  ...
49425  https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
49446  https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
49447  https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
49563  https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...
49567  https://www.bayut.eg/%D8%AA%D9%81%D8%A7%D8%B5%...

                                                Pictures       Price  Roo
1219   ['https://bayut-eg-production.s3.amazonaws.com...  10000000.0    3
1239   ['https://bayut-eg-production.s3.amazonaws.com...  12800000.0    3
1591   ['https://bayut-eg-production.s3.amazonaws.com...  10000000.0    3
1598   ['https://bayut-eg-production.s3.amazonaws.com...   2000000.0    3
1909   ['https://bayut-eg-production.s3.amazonaws.com...  10700000.0    3
...                                                  ...         ...    .
49425  ['https://bayut-eg-production.s3.amazonaws.com...  11927000.0    4
49446  ['https://bayut-eg-production.s3.amazonaws.com...  13600000.0    2
49447  ['https://bayut-eg-production.s3.amazonaws.com...  12293000.0    2
49563  ['https://bayut-eg-production.s3.amazonaws.com...   5843000.0    2
49567  ['https://bayut-eg-production.s3.amazonaws.com...   2500000.0    1
```

```
In [28]: num_duplicates = duplicates.sum()
         print(num_duplicates)
```

```
952
```

```
In [34]: bayot = bayot.drop_duplicates(subset=['Unit Link'])
```

Handling Nulls & Duplicates

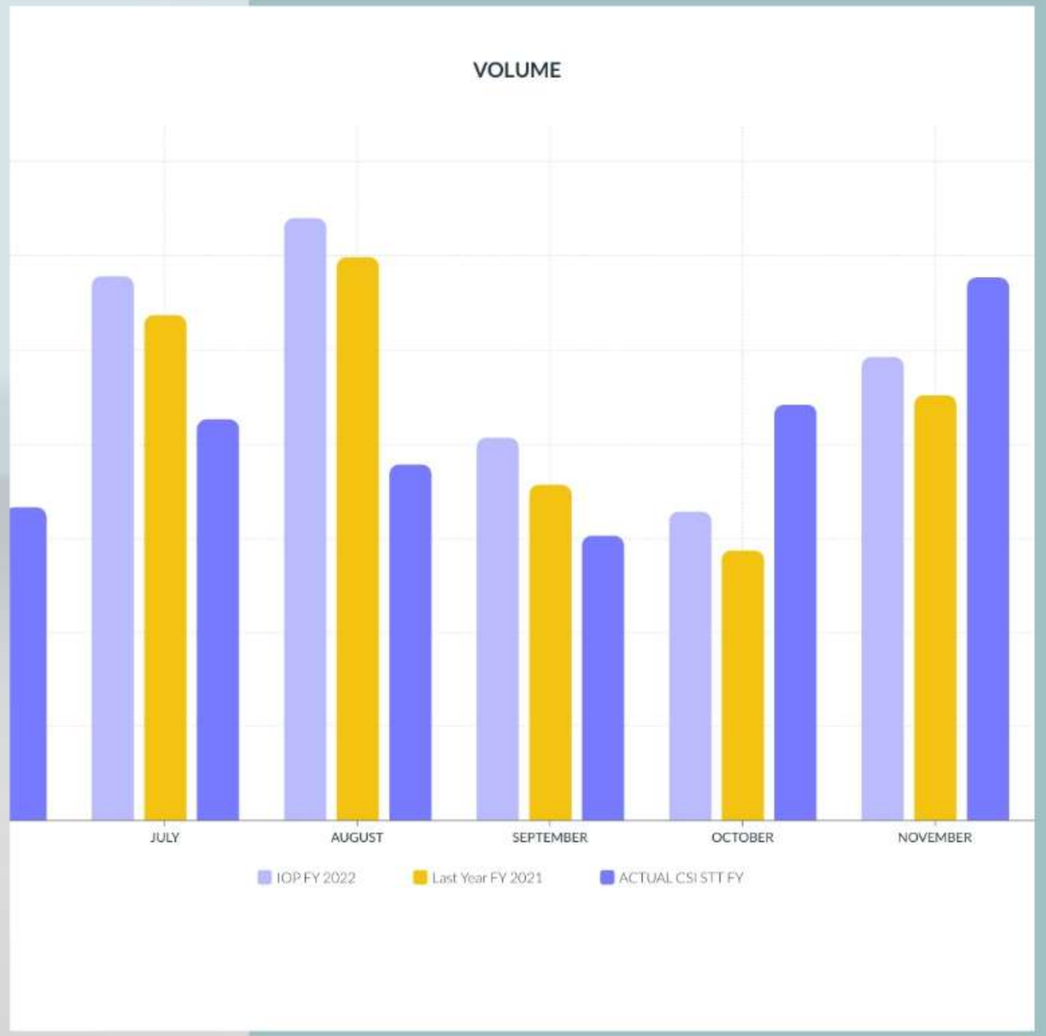
This code snippet first sets a threshold to drop rows with missing values in more than half of the columns in the bayot DataFrame. It then identifies duplicate rows based on the Unit Link column, printing these duplicates and the total number of duplicates found. Finally, it removes these duplicate rows from the DataFrame to ensure each property listing is unique.

Machine Learning Model

This section delves into the key processes involved in creating a machine learning model for predicting property prices using cleaned data from the Bayut website. Each component plays a vital role in ensuring accurate predictions.

Model Selection

The selection of the machine learning model is crucial for accurate predictions. Models like Linear Regression, Decision Trees, and Random Forest were considered, with a focus on those that can handle regression tasks effectively.



Feature Engineering

Feature engineering enhances model accuracy by transforming raw data into meaningful variables. Factors such as location, number of bedrooms, and unit size were extracted and refined for optimal performance in prediction.

Training and Testing the Model

The data was split into training and testing sets to ensure model generalization. A robust training process involved iterative learning where the model refined its parameters to minimize prediction errors.

Model Evaluation Metrics

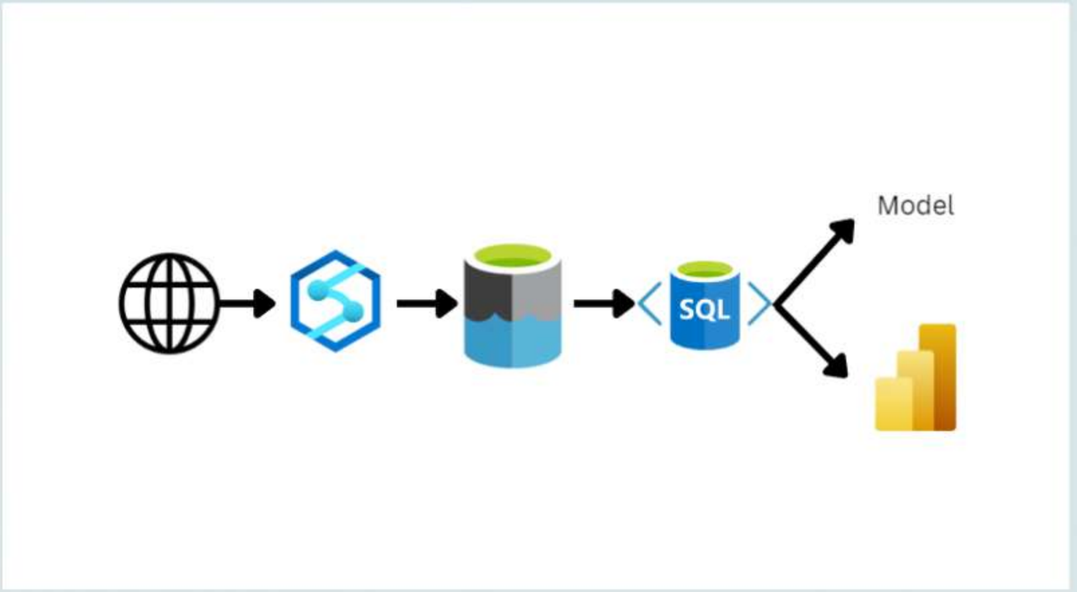
Model performance was assessed using metrics such as Mean Absolute Error (MAE) and R-squared values. These metrics provided insights into the model's accuracy and reliability in predicting unit prices.

Azure Cloud

Azure Cloud is where we handle data storage, cleaning, and processing. It enables us to securely manage large datasets, clean them, and prepare them for analysis and machine learning models.

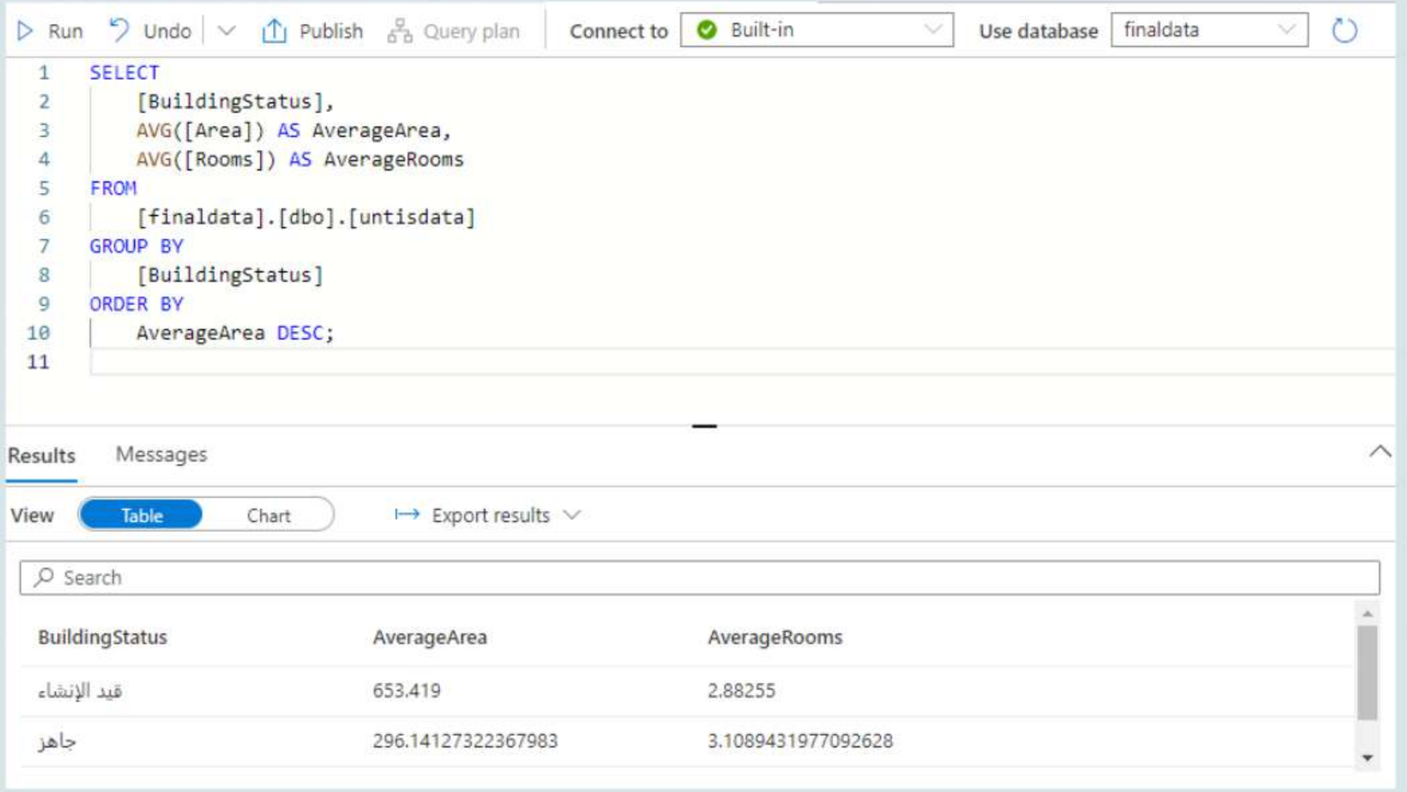
We start by scraping data from a website. This data is then sent to Azure Synapse for processing. In Synapse, we clean and transform the raw data to make it useful. After that, the cleaned data is stored in a serverless SQL database. From the database, we send the data in two directions:

To a machine learning model for prediction and insights.
To Power BI for further analysis and visualizations.



Average Area and Room Count by Building Status

This query calculates the average area and the average number of rooms for properties grouped by their building status (e.g., under construction, completed).



Total Listings by City and State

This query counts the total number of property listings by each city and state.

```
    [state],
    [city],
    COUNT([UnitLink]) AS TotalListings
FROM
    [finaldata].[dbo].[untisdata]
GROUP BY
    [state],
    [city]
```

Results | Messages

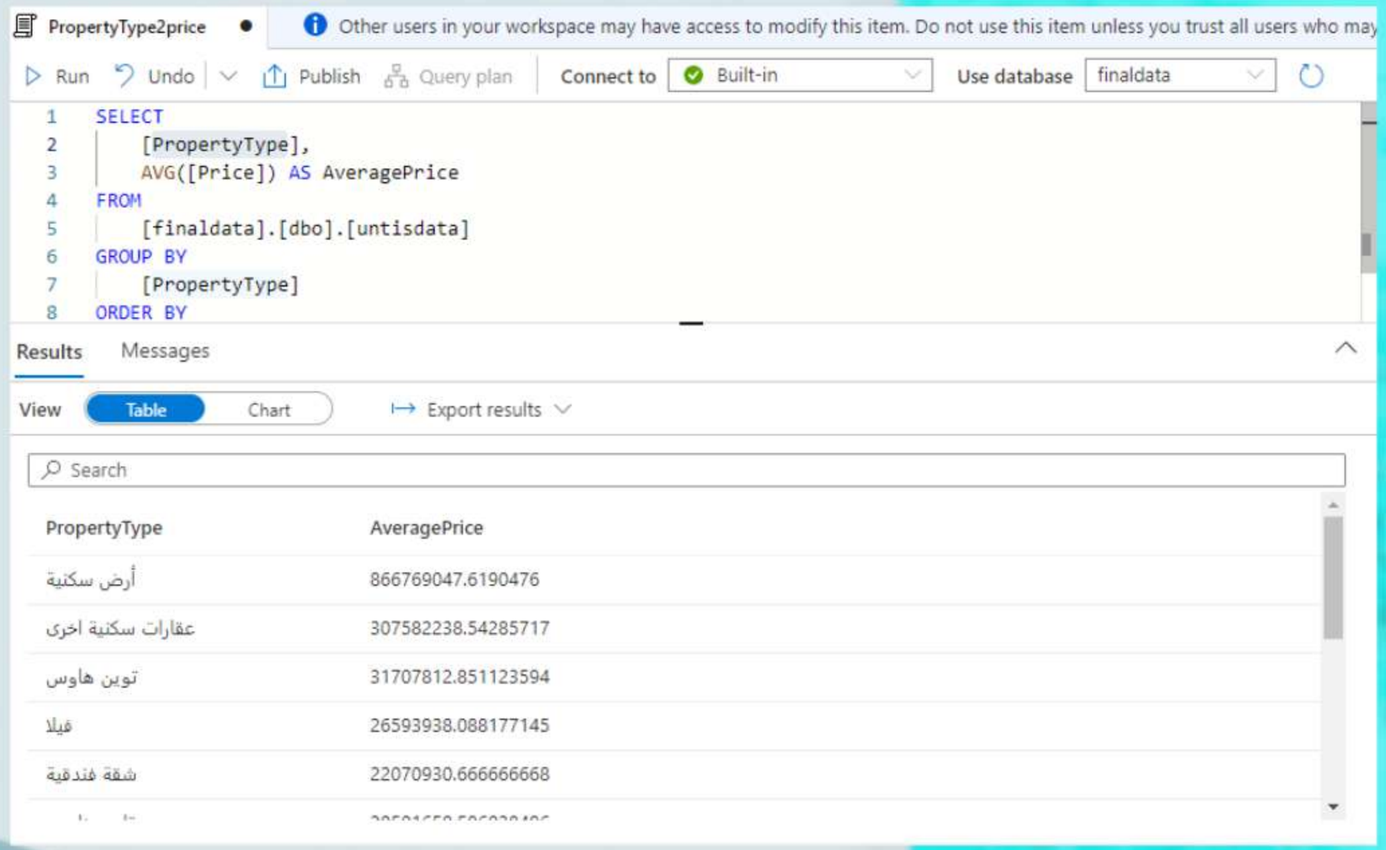
View: Table | Chart | Export results

Search

state	city	TotalListings
القاهرة الجديدة	القاهرة	14311
الساحل الشمالي	مطروح	6843
الشيخ زايد	الجيزة	4404
مدينة المستقبل	القاهرة	4387

Average Price by Property Type

This query calculates the average price of properties grouped by their type.

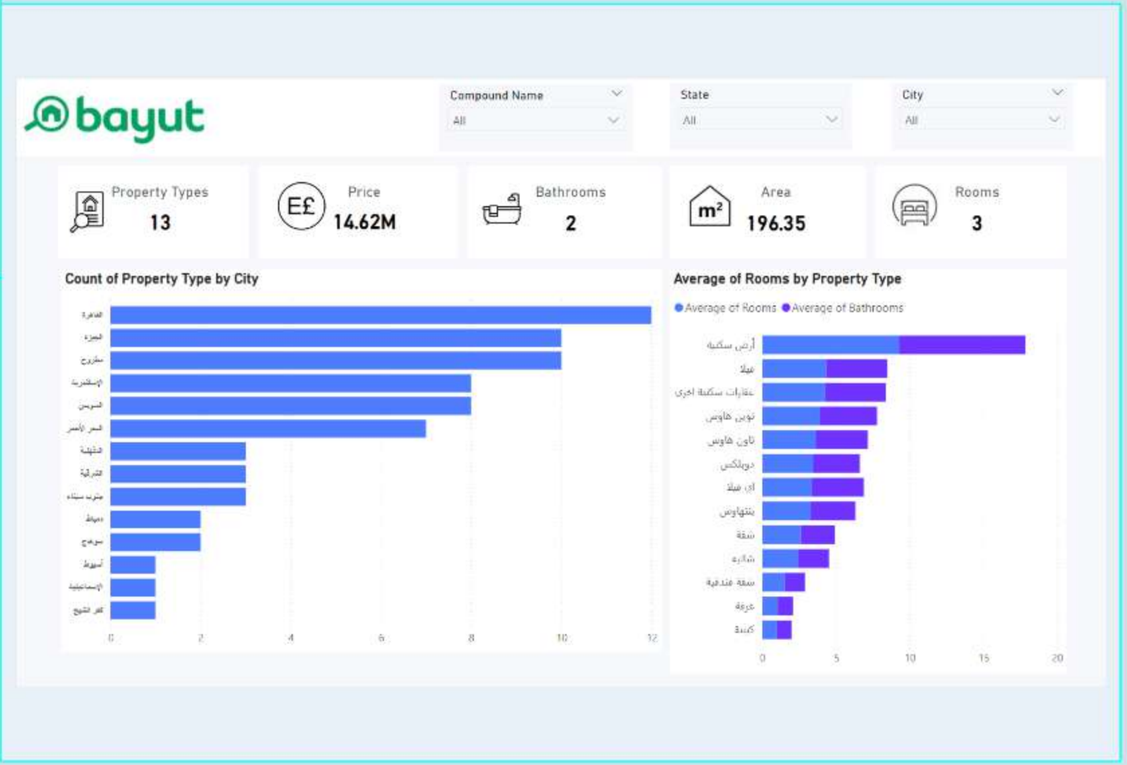


Visualization and Insights

This section showcases the synthesis of data into visual formats, utilizing Power BI to convey insights derived from property price predictions. It highlights key visualizations and the actionable insights they provide from the modeled data.

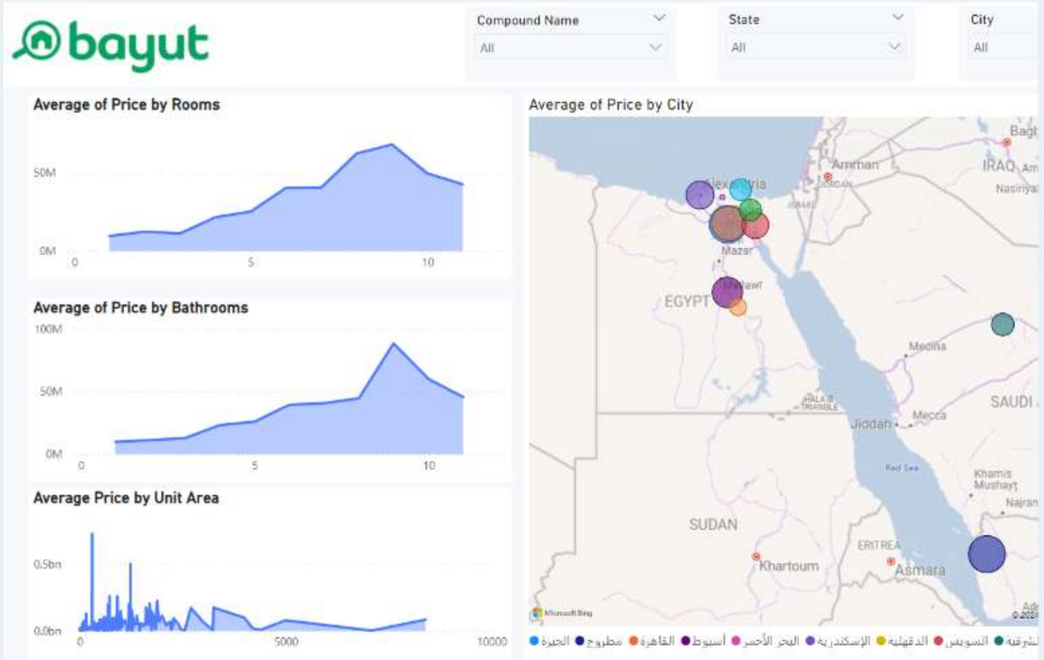
Power BI Dashboard Overview

The Power BI dashboard serves as an interactive visual interface to present real estate data effectively. It consolidates various metrics and KPIs, allowing stakeholders to quickly assess market trends and property values.



Key Visualizations

The dashboard highlights key visualizations such as the Count of Property Types by City, showing Cairo, Giza, and Matrouh as having the highest property type counts. The Average Rooms and Bathrooms by Property Type reveals that residential land has the highest average room count. Additionally, the Price Distribution by Rooms and Bathrooms shows prices rising with more rooms or bathrooms, while a map displays Average Price by City geographically. The Number of Units by Rooms and Bathrooms visualizes that most units have around four rooms and two bathrooms.



Insights Gained from the Data

The dashboard reveals that Cairo, Giza, and Matrouh have the most diverse property types, with larger properties commanding higher prices as rooms and bathrooms increase. Residential land has the most rooms on average, and major cities like Cairo and Alexandria have the highest property prices. Most units feature three rooms and two bathrooms, indicating demand for mid-sized housing.

Future Work and Recommendations

Future enhancements include refining the ML model for improved accuracy and incorporating additional data sources such as economic indicators. Regularly updating the Power BI dashboard will ensure timely insights for decision-makers.