Qassim University جامعة القصيم

College of Computer | كلية الحاسب

PREDICTING LAND PRICES IN RIYADH CITY

Using Random Forest Regressor

Table of Contents

introduction

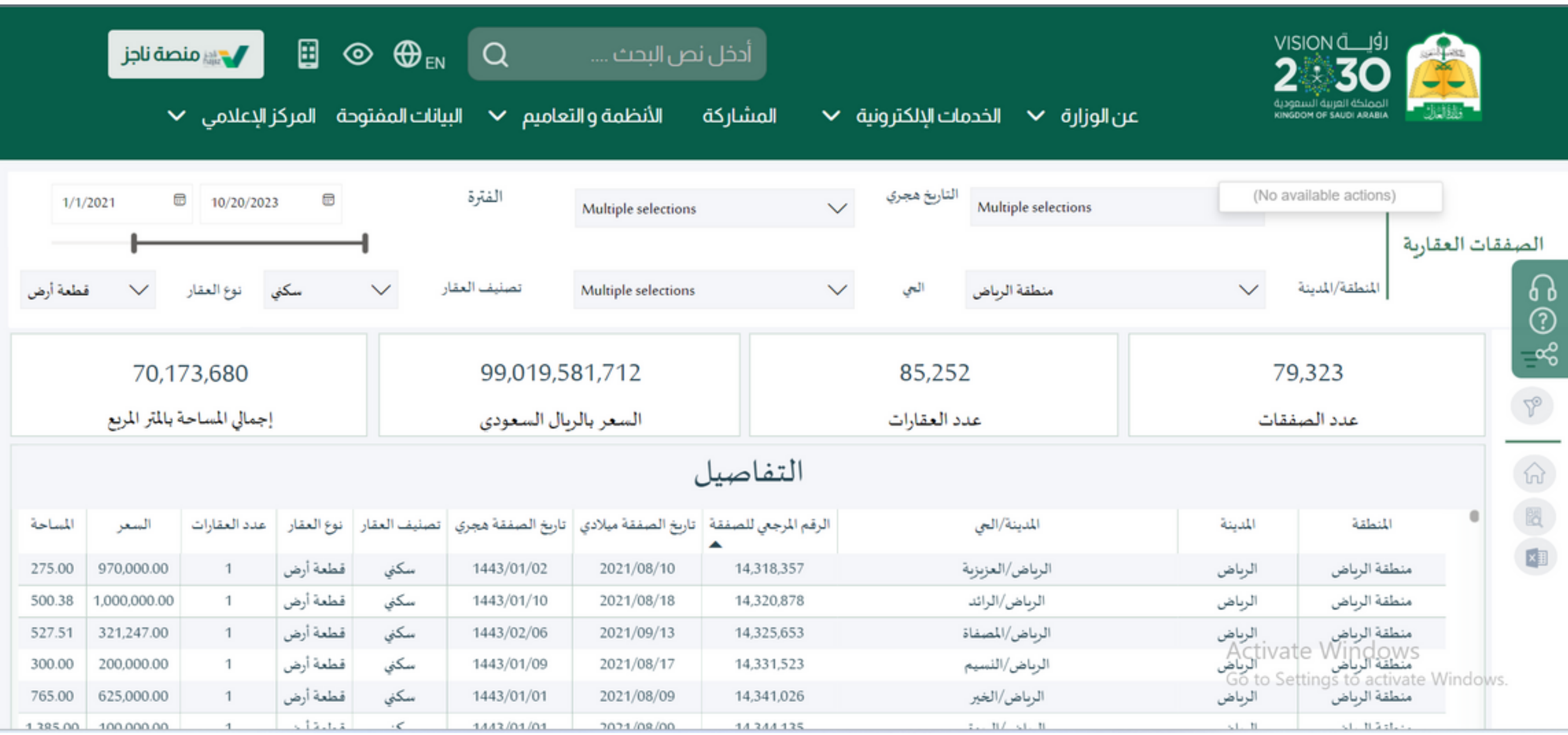
How did this idea come to be chosen for the project, and what were the compelling reasons behind its selection?

The Problem and Goal of the Project:

The aim of this data science project is to predict the land price in Riyadh City. With Riyadh becoming a prominent global destination and the expected influx of people to the city in the near future, it is crucial to forecast land prices accurately. By leveraging machine learning techniques, specifically the Random Forest Regressor, we seek to provide valuable insights into the future real estate market trends in Riyadh.

Dataset Description:

The dataset used in this project is sourced from the Ministry of Justice. It encompasses information about various neighborhoods in Riyadh City, including their names, corresponding land spaces, and prices. The dataset consists of multiple rows and columns, **6K+** providing a comprehensive view of the real estate market in Riyadh.



Data preprocessing and visualisation

+ Code + Text

First of all we need to ensure that there is any no missing values

```
missing_values=land.isnull().sum()
print(missing_values)
```

```
Region                              0.0
City                                0.0
City / neighborhood                 0.0
The reference number of the deal    0.0
The date of the deal AD             0.0
Hijri date of the deal              0.0
Real estate classification          0.0
Real estate                         0.0
The number of real estate           0.0
the price                           0.0
Space                               0.0
dtype: float64
```

Descriptive Statistics:

Calculate descriptive statistics for the price variable grouped by neighborhood. This will provide summary measures such as mean, median, minimum, maximum, and standard deviation for each neighborhood, giving us a quantitative understanding of how prices vary across different neighborhoods.

+ Code + Text

```
neigh_price_stats = land.groupby('City / neighborhood')['the price'].describe()
print(neigh_price_stats)
```

```
                       count          mean           std        min        25%  \
City / neighborhood
Riyadh/ 3419             1.0  4.800000e+05           NaN   480000.0   480000.0
Riyadh/ Al -Bayan       44.0  8.305235e+05  2.933652e+05   290000.0   699978.0
Riyadh/ Al -Raya        12.0  4.158333e+05  2.549762e+05   200000.0   320000.0
Riyadh/ Al -Shula       12.0  2.919167e+05  4.816158e+04   200000.0   267500.0
Riyadh/ Al -Shumaisi    12.0  2.662787e+05  3.300693e+05    25000.0    94416.5
...                      ...           ...           ...        ...        ...
Riyadh/resort           12.0  2.061248e+06  8.970619e+05   950000.0  1260000.0
Riyadh/roses            19.0  7.256034e+06  1.814782e+07   430667.0  1650000.0
Riyadh/square            7.0  2.263653e+06  2.836175e+06   500000.0   756286.0
Riyadh/symposium         4.0  1.237500e+06  3.772157e+05  1000000.0  1037500.0
Riyadh/wilderness        9.0  3.050000e+05  1.579161e+05   100000.0   200000.0

                            50%        75%        max
City / neighborhood
Riyadh/ 3419           480000.0   480000.0   480000.0
Riyadh/ Al -Bayan      761661.0   849477.5  1950000.0
Riyadh/ Al -Raya       347500.0   407500.0  1200000.0
Riyadh/ Al -Shula      292500.0   312000.0   400000.0
Riyadh/ Al -Shumaisi   190000.0   292500.0  1250000.0
...                         ...        ...        ...
```

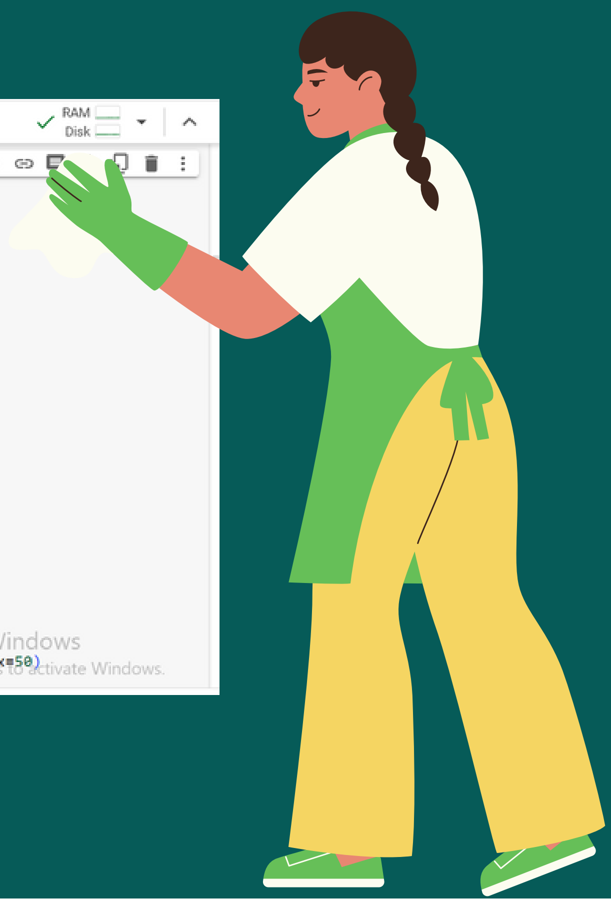
A hexbin plot, also known as a hexagonal binning plot, is a type of 2D scatter plot that represents the distribution of data points using hexagonal bins. It is particularly useful when dealing with a large number of data points and allows for better visualization of the data density.

Hexbin plots are beneficial for visualizing the relationship between two continuous variables and identifying patterns or clusters in the data. They provide a more concise representation of dense regions compared to traditional scatter plots.

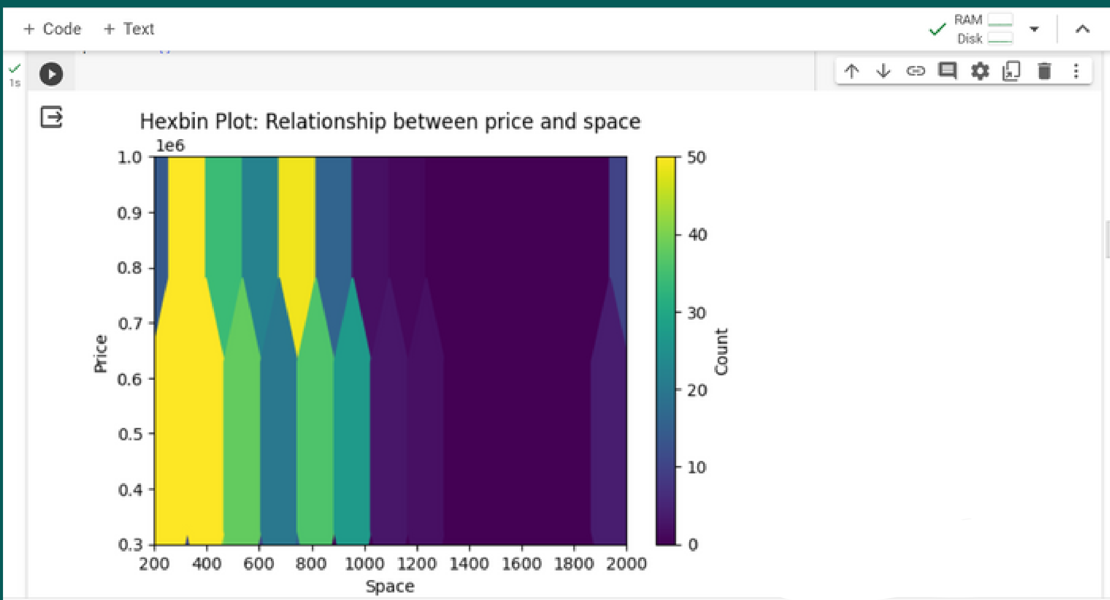
```
import matplotlib.pyplot as plt
import numpy as np
import random
import re

# 'land' is our DataFrame with 'the price' and 'Space' columns

# Function to clean and convert columns to float
def clean_and_convert(value):
    if isinstance(value, str):
        try:
            # Remove non-numeric characters and convert to float
            cleaned_value = float(re.sub(r'[^\d.]', '', value.replace(',', '.')
            return cleaned_value
        except ValueError:
            return np.nan  # Return NaN for values that couldn't be conver
    else:
        return value  # Return the original value for non-string elements

# Clean and convert columns
land['the price'] = land['the price'].apply(clean_and_convert)
land['Space'] = land['Space'].apply(clean_and_convert)
```



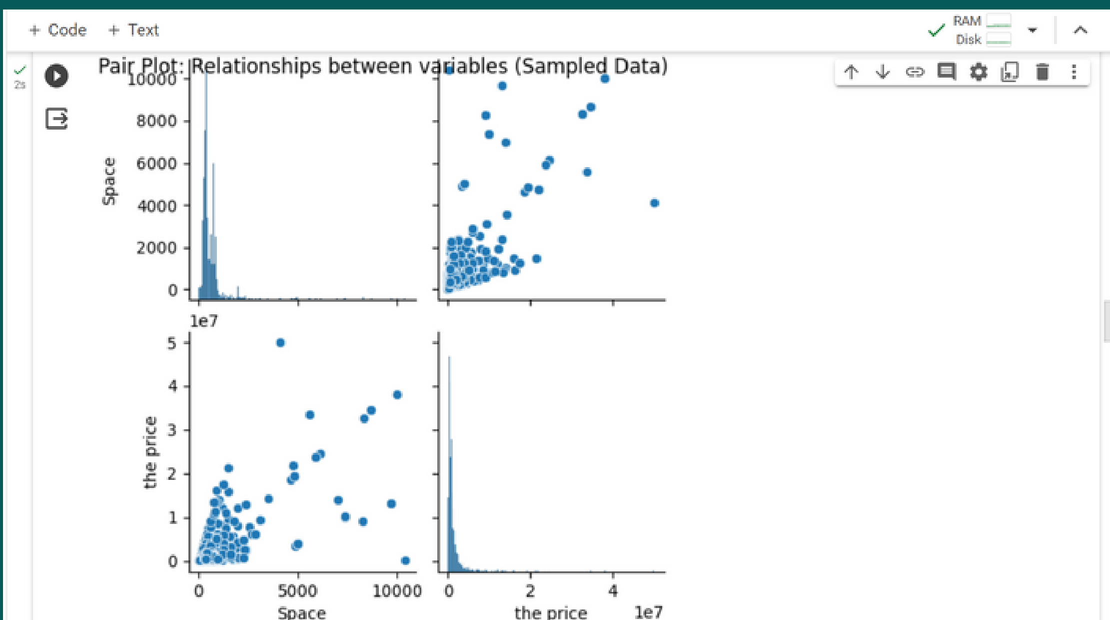
```
# Clean and convert columns
land['the price'] = land['the price'].apply(clean_and_convert)
land['Space'] = land['Space'].apply(clean_and_convert)

price = land['the price']
space = land['Space']

fig, ax = plt.subplots()
fig.subplots_adjust(bottom=0.2)

# Set reasonable limits for price values
price_min = 300000
price_max = 1000000

# Set reasonable limits for space values
space_min = 200
space_max = 2000

# Use random.sample on the DataFrame index
idx = random.sample(land.index.tolist(), 1000)

# Create a hexbin plot with adjusted gridsize and color scaling
hb = ax.hexbin(space.loc[idx], price.loc[idx], gridsize=100, cmap='viridis', vmin=0, vmax=50)
```

```python
# Use random.sample on the DataFrame index
idx = random.sample(land.index.tolist(), 1000)

# Create a hexbin plot with adjusted gridsize and color scaling
hb = ax.hexbin(space.loc[idx], price.loc[idx], gridsize=100, cmap='viridis', vmin=0, vmax=50)

plt.title('Hexbin Plot: Relationship between price and space')
plt.xlabel('Space')
plt.ylabel('Price')

# Add a colorbar
cbar = plt.colorbar(hb)
cbar.set_label('Count')

# Set limits for the x-axis (space) and y-axis (price)
plt.xlim(space_min, space_max)
plt.ylim(price_min, price_max)

plt.show()
```



By **creating a pair plot,** you can visually analyze the relationships and patterns between the 'Space' and 'the price' variables in the sampled data. The scatter plots show the pairwise relationships, while the histograms provide information about the distribution of each individual variable.



Random Forest Regressor

Random forest regression is a supervised learning algorithm and bagging technique that uses an ensemble learning method for regression in machine learning. The trees in random forests run in parallel, meaning there is no interaction between these trees while building the trees.

Why we choose it ?

- High Predictive Accuracy
- Deals with Noisy Data
- Handles Mixed Data Types
- Ease of use
- Parallelization (Faster Training Time)

Code Review

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer  # Import the imputer

# Load your dataset
land = pd.read_csv('land.csv')

land.head()
# Apply the cleaning function to 'the price' column
land['the price'] = land['the price'].apply(clean_and_convert)
land['Space'] = land['Space'].apply(clean_and_convert)  # Apply the appropriate cleaning function for 'space'

features = land.drop('the price', axis=1)
features['Space'] = land['Space']
target = land['the price']

# Convert categorical columns to numerical using Label Encoding
label_encoder = LabelEncoder()
for column in features.select_dtypes(include=['object']).columns:
    features[column] = label_encoder.fit_transform(features[column])

# Impute missing values using SimpleImputer
imputer = SimpleImputer(strategy='mean')
features_imputed = pd.DataFrame(imputer.fit_transform(features), columns=features.columns)
```

This code demonstrates the preprocessing steps for our model , we also made a function for replacing the Arabic comma since the dataset was Arabic.

```
# Function to clean and convert
def clean_and_convert(price_str):
    # Remove any non-numeric characters (including Arabic commas)
    cleaned_price = ''.join(c for c in price_str if c.isdigit() or c == '.')

    # Convert to float
    try:
        return float(cleaned_price)
    except ValueError:
        # Handle cases where the conversion fails
        return None  # or another appropriate value
```

Code Review

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_imputed, target, test_size=0.1, random_state=42)

# Create a Random Forest Regressor
regressor = RandomForestRegressor(random_state=42)

# Train the regressor on the training data
regressor.fit(X_train, y_train)

# Make predictions on the test set
predictions = regressor.predict(X_test)

# Evaluate model performance for regression
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

Apply The Random forest regressor on the dataset , Evaluate the model using ‘ Mean Squared Error , R squared ‘

Results

```
Mean Squared Error: 4392506126844.7803
R-squared: 0.5863619906552313
```

- Mean Squared Error (MSE): lower values indicate better predictive performance
- R-Squared : Higher R-squared values indicate a better fit.

The model's Mean Squared Error (MSE) is 4.3 trillion, indicating significant prediction deviations. The R-squared value of 0.58 suggests a moderate level of explanatory power. Improvement opportunities lie in refining features and addressing potential overfitting. Further model adjustments may enhance predictive accuracy.

Lesson Learned

1.Preprocessing Challenges:

Issue: Preprocessing the dataset posed challenges, especially in handling non-standard numerical representations (e.g., Arabic commas)
Lesson Learned: Establish clear preprocessing guidelines early on and create robust cleaning functions to address specific quirks in the data.

1.Translating the Dataset:

Issue: The need to translate the dataset from Arabic to English presented difficulties in maintaining data integrity and interpreting certain linguistic nuances.
Lesson Learned: Prioritize effective translation processes and ensuring accurate preservation of data semantics .

2. Model Fit:

Issue: Choosing an initially complex model resulted in poor fit for the dataset.
Lesson Learned: Prioritize model simplicity that aligns with dataset intricacies for better predictive accuracy.

4. Feature Engineering:

Issue : Insufficient feature engineering contributed to model misfit and suboptimal performance.
Lesson : Invest in feature selection and engineering to enhance the model's ability to capture relevant patterns in the data.

Thank You

resources:

the dataset :
https://www.moj.gov.sa/ar/opendata/Pages/reports.aspx

other :
forest-regression-model-d060706a5e7f

/easy-guide-data-preprocessing-python

https://builtin.com/data-science/random-forest-python

Represented By :
Lama alharbi
Raghad

Represented To
Dr.afaf

Scan to see the whole project