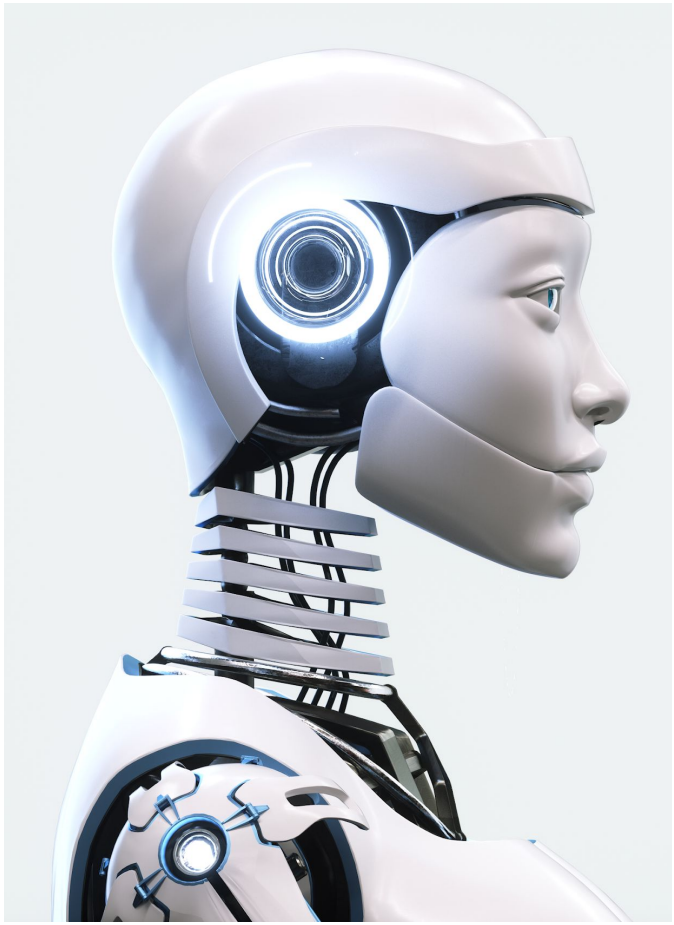
CAPSTONE PROJECT

HOUSE PRICE PREDICTION MODEL

PREPARED BY: ABU SALATHIN AKASHAH BIN KHAIRUDDIN

SUPERVISOR: TS. DR. CHAN WENG HOWE

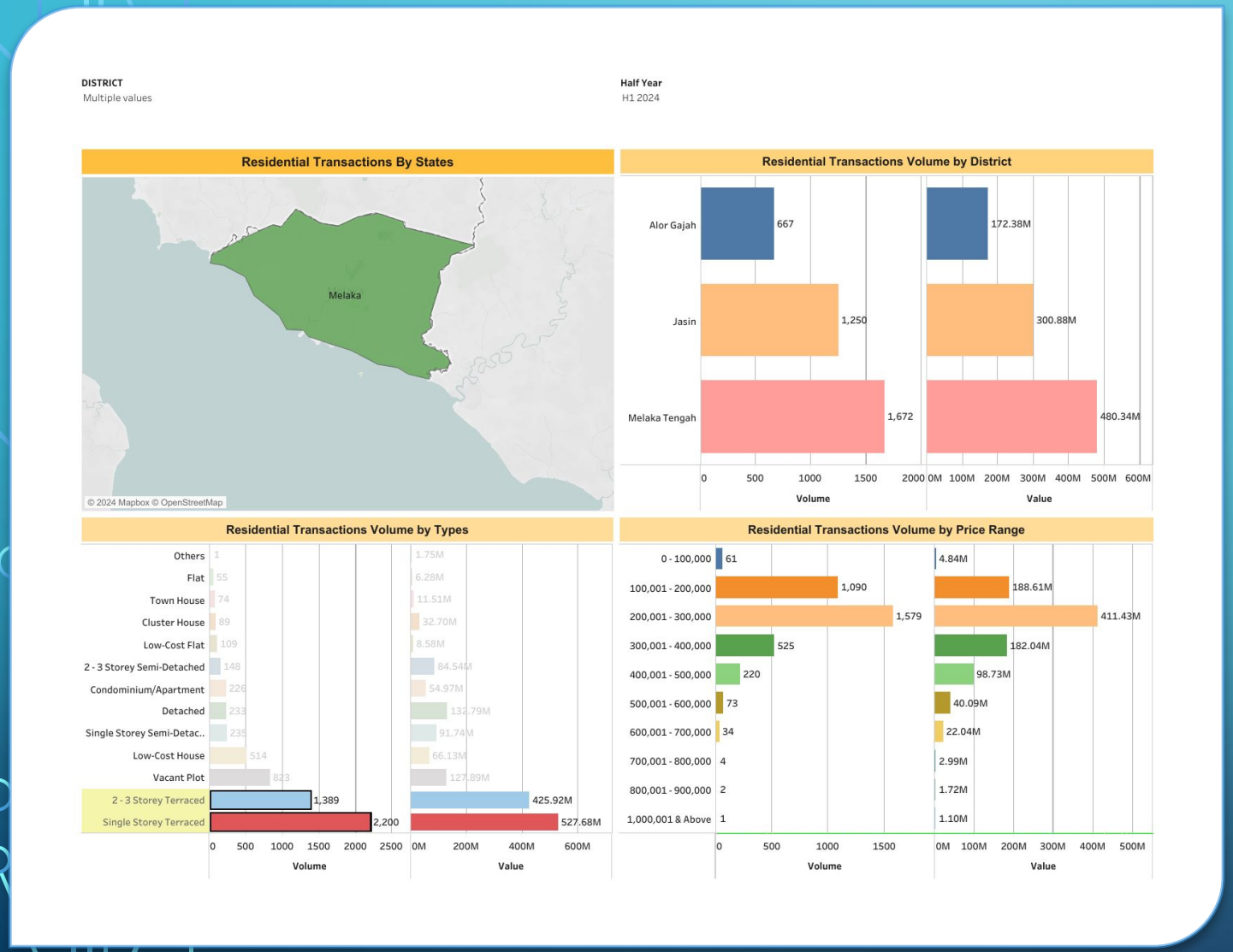
CONTENT



PROBLEM STATEMENT

- The problem this project addresses is the need for a reliable check-and-balance system to ensure the accuracy of national property reports. While property valuations by JPPH branches are generally reliable, there can be differences between the JPPH valuation and transaction price / market value. This project aims to apply machine learning models to property transaction data to cross-check these valuations. By identifying potential discrepancies, the machine learning approach helps ensure that NAPIC's national property reporting remains accurate and consistent, providing stakeholders with trustworthy information about the property market.

NAPIC PUBLICATION



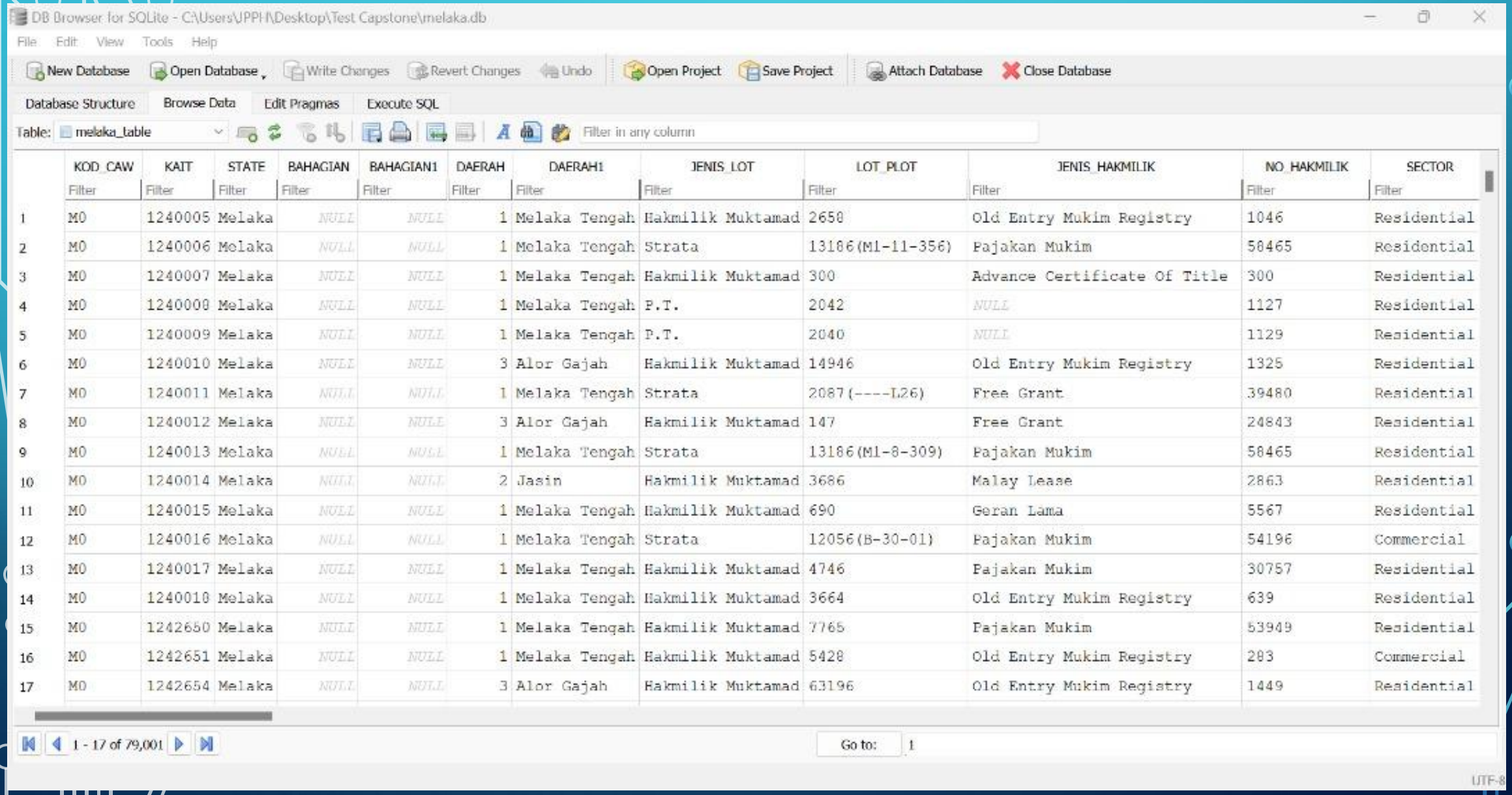
KOD_CAW	1	CATEGORY	PRO_TYPE	B_TINGKA	SYER1	SYER2	TKH_NILAI	LUAS_LOT	UNIT_LUA	RANGE	LUAS_LOT	HARGA_B	NILAI_LPR	PREDICTED VALUE
M0	bang	Terraced House	2 - 2 1/2 Storey Terraced	2	1	1	1/5/2023	143	mp	4	124.86	RM0.00	RM350,000.00	?
M0	ık	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	01/16/2023	143	mp	2	76.92	RM80,000.00	RM180,000.00	?
M0	Tengah	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	01/13/2023	143	mp	2	82.5	RM0.00	RM110,000.00	?
M0	il	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	1/9/2023	156	mp	4	111.48	RM0.00	RM380,000.00	?
M0	ak	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	1/11/2023	102	mp	2	65.03	RM110,000.00	RM140,000.00	?
M0	ndam	Terraced House	2 - 2 1/2 Storey Terraced	2	1	1	02/14/2023	104	mp	2	83.43	RM0.00	RM180,000.00	?
M0	ak	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	1/4/2023	114	mp	2	61.32	RM90,000.00	RM160,000.00	?
M0		Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	1/5/2023	111	mp	2	74.32	RM0.00	RM200,000.00	?
M0	Pindah	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	1/4/2023	111	mp	2	74.32	RM0.00	RM180,000.00	?
M0	laka	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	1/2/2023	188	mp	4	91.97	RM0.00	RM310,000.00	?
M0	il	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	3/1/2023	130	mp	3	74.88	RM230,000.00	RM230,000.00	?
M0	ndam	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	03/24/2023	112	mp	3	70.61	RM232,000.00	RM232,000.00	?
M0	ndam	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	03/20/2023	159	mp	3	67.91	RM210,000.00	RM210,000.00	?
M0	Pindah	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	03/17/2023	121	mp	3	74.32	RM250,000.00	RM250,000.00	?
M0		Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	03/20/2023	111	mp	3	67.89	RM212,000.00	RM212,000.00	?
M0	ndam	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	1/9/2023	184	mp	3	87.67	RM0.00	RM264,000.00	?
M0	anjang	Terraced House	1 - 1 1/2 Storey Terraced	1	1	1	03/14/2023	102	mp	2	63.17	RM90,000.00	RM180,000.00	?
												RM1,504,000.00	**RM3,768,000.00**	?

NAPIC SOURCE OF PUBLICATION

DATA DESCRIPTION

- This project useses property transaction dataset containing detailed information on property locations, characteristics, and transaction attributes. The dataset includes location identifiers such as district, mukim and skim, as well as property features including land area, building area, building age and the number of floors. Additional details cover the number of bedrooms, ownership type, and building condition. Area classification variables such as area category and area classification further enhance the context for each transaction. The dataset is organized by unique property identifiers, with data stored in SQLite Database which is a structured format suitable for regression analysis.

DATASET EXAMPLE


DB Browser for SQLite - C:\Users\UPPI\Desktop\Test Capstone\melaka.db

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Undo | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: melaka_table | Filter in any column

	KOD_CAW	KAIT	STATE	BAHAGIAN	BAHAGIAN1	DAERAH	DAERAH1	JENIS_LOT	LOT_PLOT	JENIS_HAKMILIK	NO_HAKMILIK	SECTOR
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	M0	1240005	Melaka	NULL	NULL	1	Melaka Tengah	Hakmilik Muktamad	2658	Old Entry Mukim Registry	1046	Residential
2	M0	1240006	Melaka	NULL	NULL	1	Melaka Tengah	Strata	13186(M1-11-356)	Pajakan Mukim	58465	Residential
3	M0	1240007	Melaka	NULL	NULL	1	Melaka Tengah	Hakmilik Muktamad	300	Advance Certificate Of Title	300	Residential
4	M0	1240008	Melaka	NULL	NULL	1	Melaka Tengah	P.T.	2042	NULL	1127	Residential
5	M0	1240009	Melaka	NULL	NULL	1	Melaka Tengah	P.T.	2040	NULL	1129	Residential
6	M0	1240010	Melaka	NULL	NULL	3	Alor Gajah	Hakmilik Muktamad	14946	Old Entry Mukim Registry	1325	Residential
7	M0	1240011	Melaka	NULL	NULL	1	Melaka Tengah	Strata	2087(----L26)	Free Grant	39480	Residential
8	M0	1240012	Melaka	NULL	NULL	3	Alor Gajah	Hakmilik Muktamad	147	Free Grant	24843	Residential
9	M0	1240013	Melaka	NULL	NULL	1	Melaka Tengah	Strata	13186(M1-8-309)	Pajakan Mukim	58465	Residential
10	M0	1240014	Melaka	NULL	NULL	2	Jasin	Hakmilik Muktamad	3686	Malay Lease	2863	Residential
11	M0	1240015	Melaka	NULL	NULL	1	Melaka Tengah	Hakmilik Muktamad	690	Geran Lama	5567	Residential
12	M0	1240016	Melaka	NULL	NULL	1	Melaka Tengah	Strata	12056(B-30-01)	Pajakan Mukim	54196	Commercial
13	M0	1240017	Melaka	NULL	NULL	1	Melaka Tengah	Hakmilik Muktamad	4746	Pajakan Mukim	30757	Residential
14	M0	1240018	Melaka	NULL	NULL	1	Melaka Tengah	Hakmilik Muktamad	3664	Old Entry Mukim Registry	639	Residential
15	M0	1242650	Melaka	NULL	NULL	1	Melaka Tengah	Hakmilik Muktamad	7765	Pajakan Mukim	53949	Residential
16	M0	1242651	Melaka	NULL	NULL	1	Melaka Tengah	Hakmilik Muktamad	5428	Old Entry Mukim Registry	283	Commercial
17	M0	1242654	Melaka	NULL	NULL	3	Alor Gajah	Hakmilik Muktamad	63196	Old Entry Mukim Registry	1449	Residential

1 - 17 of 79,001 | Go to: 1

UTF-8

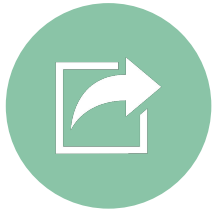
DATA PREPROCESSING

IMPORT DATASET

EXPLORE DATASET

DATA FILTRATION

DATA CLEANING



EXPORT

IMPORT DATASET

USE CASE: Predicting Melaka Terraced House

In this process, we will use a dataset of single and double storey terraced house in Melaka for the year of 2023 of around 5,000 housing transactions and the pricing. I will explore this dataset and build a regression model to predict the house price using the related information.

1. Import and load dataset

1. Read the file `melaka.db` using `sqlite3` library using the `pd.read_sql_query()`

```
[ ]
import pandas as pd
import gdown

# Google Drive URL
shared_url = 'https://drive.google.com/file/d/1pVYinnzh9PEld_bY1SnvM3qIc4-MSenV/view?usp=drive_link'
file_url = f'https://drive.google.com/uc?id={shared_url.split("/")[-2]}'  # Generate the correct download URL

# Output file name for the downloaded SQLite database
output = 'melaka.db'

# Download the file using gdown
gdown.download(file_url, output, quiet=False)

# Now you can proceed to connect to the SQLite database and create a DataFrame
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect(output)

# Load the data into a Pandas DataFrame
df_Melaka = pd.read_sql_query("SELECT * FROM melaka_table", conn)

# Display the first few rows of the DataFrame
df_Melaka.head()

# Close the connection after usage
conn.close()
```

```
Downloading...
From: https://drive.google.com/uc?id=1pVYinnzh9PEld_bY1SnvM3qIc4-MSenV
To: /content/melaka.db
100%|██████████| 39.4M/39.4M [00:00<00:00, 60.6MB/s]
```

EXPLORE DATASET

2. Exploring the dataset

Find out the following information about the dataset:

- size of the dataset (no. of rows, no. of columns)
- what is the features (or columns) available
- check the data types of all columns in the dataset

```
[ ]
print(f"Dataset size: {df_Melaka.shape[0]} rows and {df_Melaka.shape[1]} columns \n")

print(f"Features: \n {df_Melaka.info()}")
```

```
Dataset size: 79001 rows and 64 columns

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79001 entries, 0 to 79000
Data columns (total 64 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   KOD_CAW           79001 non-null  object
 1   KAIT              79001 non-null  int64
 2   STATE             79001 non-null  object
 3   BAHAGIAN          0 non-null      object
 4   BAHAGIAN1         0 non-null      object
 5   DAERAH            79001 non-null  int64
 6   DAERAH1           79001 non-null  object
 7   JENIS_LOT         79001 non-null  object
 8   LOT_PLOT          78632 non-null  object
 9   JENIS_HAKMILIK    77171 non-null  object
 10  NO_HAKMILIK       78993 non-null  object
 11  SECTOR            79001 non-null  object
 12  YEAR1             78356 non-null  float64
```

CLEANING DATASET

```
<ipython-input-4-716d61bbc6a3>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/in
  df_Melaka['UMUR_BGN'] = df_Melaka['YEAR1'] - df_Melaka['TKH_SIAP']
```

	KOD_CAW	KAIT	STATE	BAHAGIAN	BAHAGIAN1	DAERAH	DAERAH1	JENIS_LOT	LO
0	M0	1240005	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	
6	M0	1240011	Melaka	None	None	1	Melaka Tengah	Strata	2087(
7	M0	1240012	Melaka	None	None	3	Alor Gajah	Hakmilik Muktamad	
9	M0	1240014	Melaka	None	None	2	Jasin	Hakmilik Muktamad	
10	M0	1240015	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	

5 rows × 64 columns

```
[ ]  1  df_Melaka.shape
```

```
(5052, 64)
```

3. Cleaning the data

1. Filter data according to needs.
2. Create a new column from existing columns.
3. Drop column that is irrelevant.
4. Drop duplicate rows and retain only one row.
5. Check if there is any missing values. If yes, remove the missing values.

```
[ ]
# Assuming `df_Melaka` is your DataFrame
# Step 1: Remove rows based on the conditions and exclude rows where LUAS_LOT_BGN > LUAS_LOT for B_TINGKAT == 1
df_Melaka = df_Melaka[
    (df_Melaka['SECTOR'] == 'Residential') &
    (df_Melaka['CATEGORY'] == 'Terraced House') &
    (df_Melaka['YEAR1'] == 2023) &
    (df_Melaka['B_TINGKAT'].isin([1, 2])) &  # B_TINGKAT is either 1 or 2
    (df_Melaka['SYER2'] == 1) &
    (df_Melaka['UNIT_LUAS_LOT'] == 'mp') &
    (df_Melaka['LUAS_LOT'] >= 55) &          # LUAS_LOT is 55 and above
    (df_Melaka['LUAS_LOT_BGN'] >= 45) &      # LUAS_LOT_BGN is 45 and above
    (df_Melaka['BILIK_TDR'] > 1) &           # BILIK_TDR is more than 1
    (df_Melaka['JENIS_BINAAN'] == 'Kekal') & # JENIS_BINAAN is 'Kekal'

    # Exclude rows where LUAS_LOT_BGN is greater than LUAS_LOT when B_TINGKAT == 1
    ~((df_Melaka['B_TINGKAT'] == 1) & (df_Melaka['LUAS_LOT_BGN'] > df_Melaka['LUAS_LOT']))
]

# Step 2: Create the new column 'UMUR_BGN' by subtracting TKH_SIAP from YEAR1
df_Melaka['UMUR_BGN'] = df_Melaka['YEAR1'] - df_Melaka['TKH_SIAP']

# Step 3: Drop the 'TKH_SIAP' column
df_Melaka = df_Melaka.drop(columns=['TKH_SIAP'])

# Display the updated DataFrame
df_Melaka.head()
```

REMOVE DUPLICATES

```
[ ]
# Define the columns to check for duplicates
columns_to_check = ['LUAS_LOT', 'HARGA_B', 'MUKIM1', 'TKH_NILAI', 'SKIM',
'LOT_PLOT', 'TRANSEFEROR', 'TRANSFEREE']

# Count the number of duplicate rows based on the specific columns
duplicate_count = df_Melaka.duplicated(subset=columns_to_check, keep=False).sum
()

# Print the count of duplicates
(f"Number of duplicate rows: {duplicate_count}")

'Number of duplicate rows: 20'
```

```
[ ]
# Define the columns to check for duplicates
columns_to_check = ['LUAS_LOT', 'HARGA_B', 'MUKIM1', 'TKH_NILAI', 'SKIM',
'LOT_PLOT', 'TRANSEFEROR', 'TRANSFEREE']

# Remove duplicate rows based on these columns and keep only the first
occurrence
df_Melaka = df_Melaka.drop_duplicates(subset=columns_to_check, keep='first')

# Display the resulting DataFrame
df_Melaka.head()
```

```
[ ]
# Count the number of duplicate rows based on the specific columns
duplicate_count = df_Melaka.duplicated(subset=columns_to_check, keep=False).sum
()

# Print the count of duplicates
(f"Number of duplicate rows: {duplicate_count}")

'Number of duplicate rows: 0'
```

```
[ ]
df_Melaka.shape

(5042, 64)
```

REMOVE NULL OR BLANK VALUES

```
[ ]
# List of columns to check for null or blank values
columns_to_check = ['DAERAH1', 'PM_PERTAMA', 'SKIM', 'MUKIM1', 'B_TINGKAT',
'TKH_NILAI', 'LUAS_LOT',
                    'LUAS_LOT_BGN', 'HARGA_B', 'NILAI_LPR', 'UMUR_BGN',
                    'BILIK_TDR', 'JENIS_PEGANGAN',
                    'YEAR1', 'KEADAAN_BGN', 'KATEGORI_KAW', 'KLASIFIKASI_KAW']

# Replace blank strings with NaN to handle them as missing values
df_Melaka.replace(r'^\s*$', pd.NA, regex=True, inplace=True)

# Remove rows with null or blank values in any of the specified columns
df_Melaka = df_Melaka.dropna(subset=columns_to_check)

# Display the resulting DataFrame after dropping rows with null or blank values
df_Melaka.head()
```

	KOD_CAW	KAIT	STATE	BAHAGIAN	BAHAGIAN1	DAERAH	DAERAH1	JENIS_LOT	LOT_PLOT	JENIS_HAKMILIK
0	M0	1240005	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	2658	Old Entry Mukim Registry
6	M0	1240011	Melaka	None	None	1	Melaka Tengah	Strata	2087(----L26)	Free Grant
7	M0	1240012	Melaka	None	None	3	Alor Gajah	Hakmilik Muktamad	147	Free Grant
9	M0	1240014	Melaka	None	None	2	Jasin	Hakmilik Muktamad	3686	Malay Lease
10	M0	1240015	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	690	Geran Lama

5 rows × 64 columns

```
[ ]
df_Melaka.shape
```

```
(4970, 64)
```

REMOVE DATA (BELOW MARKET VALUE)

```
[ ]
# Filter the DataFrame where HARGA_B is equal to or greater than NILAI_LPR
df_Melaka = df_Melaka[(df_Melaka['HARGA_B'] == df_Melaka['NILAI_LPR']) |
                      (df_Melaka['NILAI_LPR'] < df_Melaka['HARGA_B'])]

# Display the filtered rows
df_Melaka.head()
```

	KOD_CAW	KAIT	STATE	BAHAGIAN	BAHAGIAN1	DAERAH	DAERAH1	JENIS_LOT	LOT_PLOT
0	M0	1240005	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	2658
6	M0	1240011	Melaka	None	None	1	Melaka Tengah	Strata	2087(----L26)
14	M0	1242650	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	7765
18	M0	1242686	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	5122
38	M0	3231653	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	9252

5 rows × 64 columns

```
[ ]
df_Melaka.shape
```

```
(4242, 64)
```

CHECK FOR MISSING VALUES

```
[ ]
# List of columns to check for missing values
columns_to_check = ['DAERAH1', 'PM_PERTAMA', 'SKIM', 'MUKIM1', 'B_TINGKAT',
'TKH_NILAI', 'LUAS_LOT',
                    'LUAS_LOT_BGN', 'HARGA_B', 'NILAI_LPR', 'UMUR_BGN',
                    'BILIK_TDR', 'JENIS_PEGANGAN',
                    'YEAR1', 'KEADAAN_BGN', 'KATEGORI_KAW', 'KLASIFIKASI_KAW']

# Check for missing values (NaN) in the specified columns
missing_values = df_Melaka[columns_to_check].isna().sum()

# Display the count of missing values for each column
(missing_values)
```

	0
DAERAH1	0
PM_PERTAMA	0
SKIM	0
MUKIM1	0
B_TINGKAT	0
TKH_NILAI	0
LUAS_LOT	0
LUAS_LOT_BGN	0
HARGA_B	0
NILAI_LPR	0
UMUR_BGN	0
BILIK_TDR	0
JENIS_PEGANGAN	0
YEAR1	0
KEADAAN_BGN	0
KATEGORI_KAW	0
KLASIFIKASI_KAW	0

dtype: int64

EXPLORATORY DATA ANALYSIS (EDA)

Objective	Data Segmentation	Data Visualization
Statistical analysis	Interpretation	Explore further

EXAMPLE OF EXPLORATORY DATA ANALYSIS

4. EDA (Exploratory Data Analysis)

- Descriptive statistics: Examine summary statistics like mean, median, standard deviation, and data distribution for each feature.
- Data visualization: Use visual tools (like histograms, scatter plots, and box plots) to explore relationships, detect patterns, and spot potential issues (e.g., skewness, outliers).
- Correlation analysis: Investigate the relationships between features and identify which features have the strongest relationship with the target variable.

```
[ ]
import matplotlib.pyplot as plt
import seaborn as sns

# Your original script to calculate and display descriptive statistics
central_tendency = df_Melaka['HARGA_B'].describe()
print(central_tendency)

# Add Histogram visualization
plt.figure(figsize=(10, 5))
sns.histplot(df_Melaka['HARGA_B'], bins=20, kde=True)
plt.title('Histogram of HARGA_B')
plt.xlabel('HARGA_B')
plt.ylabel('Frequency')
plt.show()
```

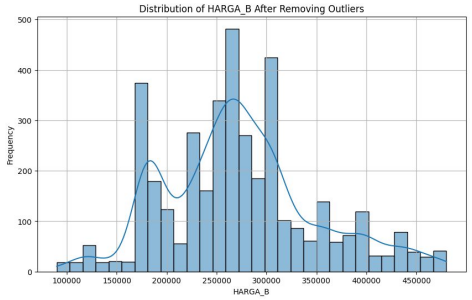
```
count       4242.000000
mean      295407.786893
std       108519.688009
min        70000.000000
25%       230000.000000
50%       270000.000000
75%       330000.000000
max      1000000.000000
Name: HARGA_B, dtype: float64
```

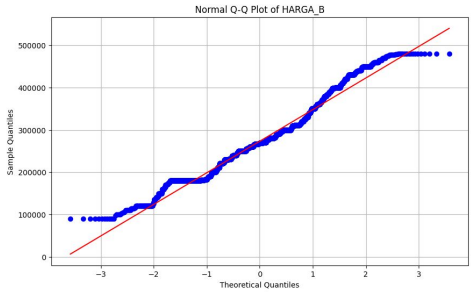
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Function to remove outliers based on IQR and return excluded values
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    # Define the lower and upper bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Get the values outside the bounds (outliers)
    excluded_values = df[(df[column] < lower_bound) | (df[column] > upper_bound)][column]

    # Filter out the outliers
    df_filtered = df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

    # Get values excluded for lower and upper bounds
    lower_excluded = df[df[column] < lower_bound][column]
    upper_excluded = df[df[column] > upper_bound][column]

    return df_filtered, lower_excluded, upper_excluded, lower_bound, upper_bound

# Apply the outlier removal function to 'HARGA_B' column
df_Melaka, lower_excluded, upper_excluded, lower_bound, upper_bound = remove_outliers_iqr(df_Melaka, 'HARGA_B')

# Print the excluded values and bounds for reference
print(f"Values excluded below the lower bound ({lower_bound}):\n{lower_excluded}")
print(f"Values excluded above the upper bound ({upper_bound}):\n{upper_excluded}")

# Plotting the histogram to visualize the distribution after removing outliers
plt.figure(figsize=(10, 6))
sns.histplot(df_Melaka['HARGA_B'], bins=30, kde=True)  # kde=True adds the Kernel Density Estimate line
plt.title('Distribution of HARGA_B After Removing Outliers')
plt.xlabel('HARGA_B')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

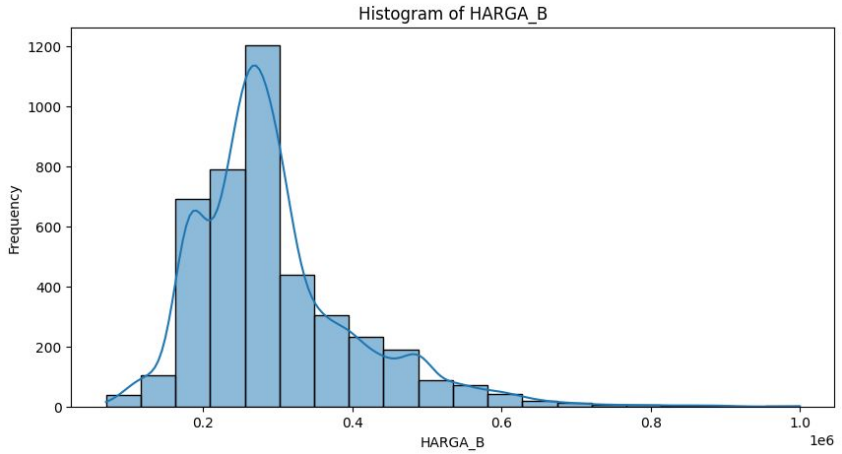
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Group the data by LUAS_LOT_BGN and calculate the average HARGA_B
df_grouped = df_Melaka.groupby('LUAS_LOT_BGN')['HARGA_B'].mean().reset_index()

# Plotting the relationship between LUAS_LOT_BGN and average HARGA_B
plt.figure(figsize=(10, 6))

# Scatter plot
sns.scatterplot(x='LUAS_LOT_BGN', y='HARGA_B', data=df_grouped)

# Adding a linear regression line
sns.regplot(x='LUAS_LOT_BGN', y='HARGA_B', data=df_grouped, scatter=False, color='red', ci=None)

plt.title('Average HARGA_B vs LUAS_LOT_BGN with Linear Regression Line')
plt.xlabel('LUAS_LOT_BGN (Building Area)')
plt.ylabel('Average HARGA_B (Price)')
plt.grid(True)
plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import pandas as pd



harga_b = df_Melaka['HARGA_B']
plt.figure(figsize=(10, 6))
stats.probplot(harga_b, dist="norm", plot=plt)
plt.title('Normal Q-Q Plot of HARGA_B')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')
plt.grid()
plt.show()
```

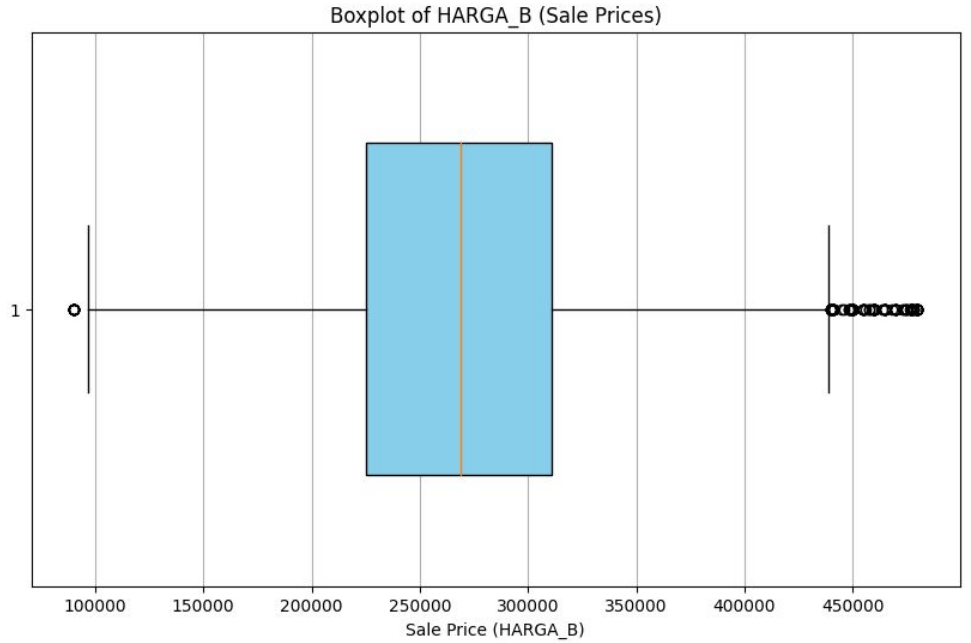
EXAMPLE OF EXPLORATORY DATA ANALYSIS



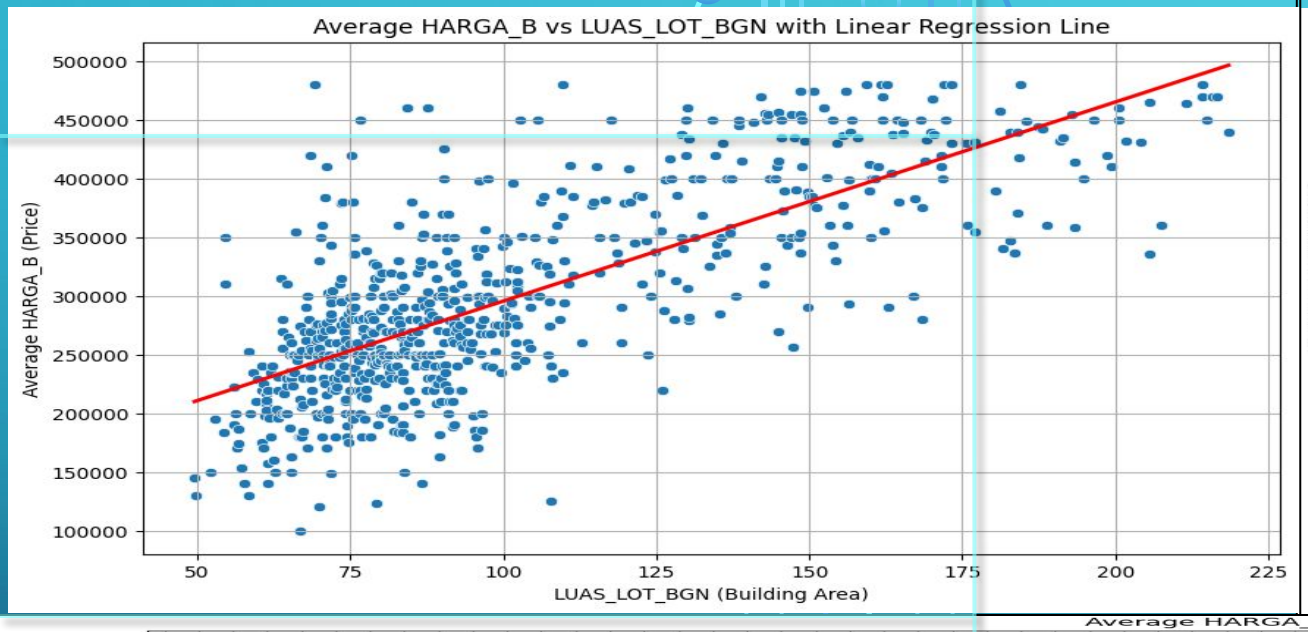


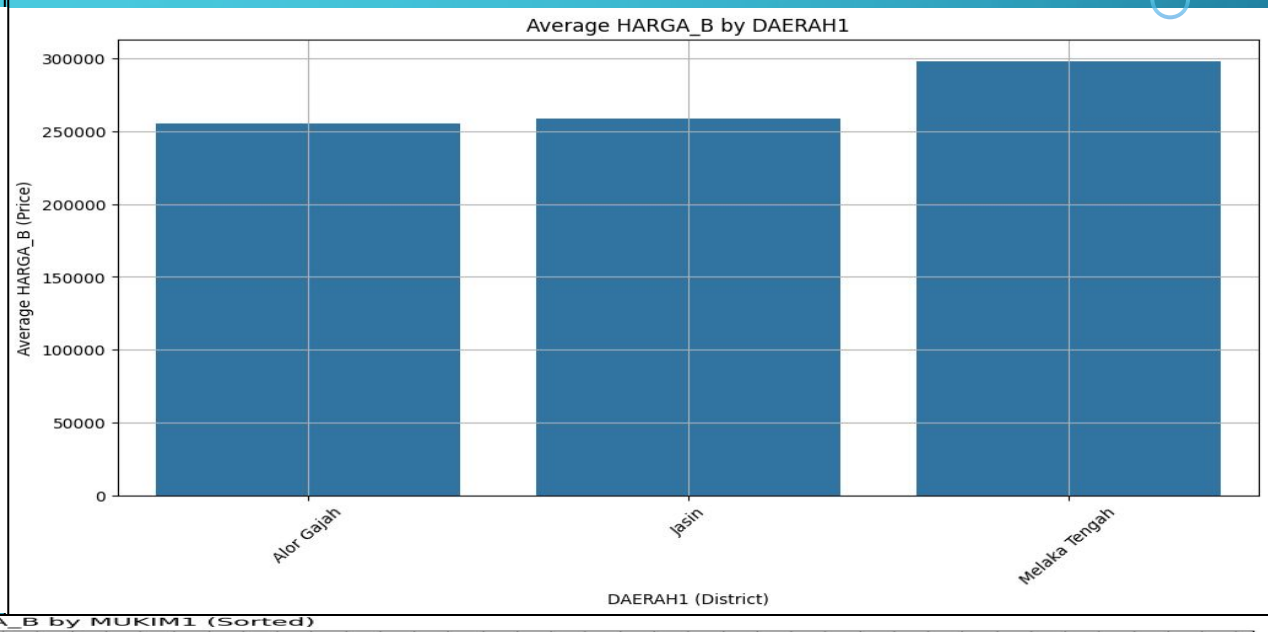


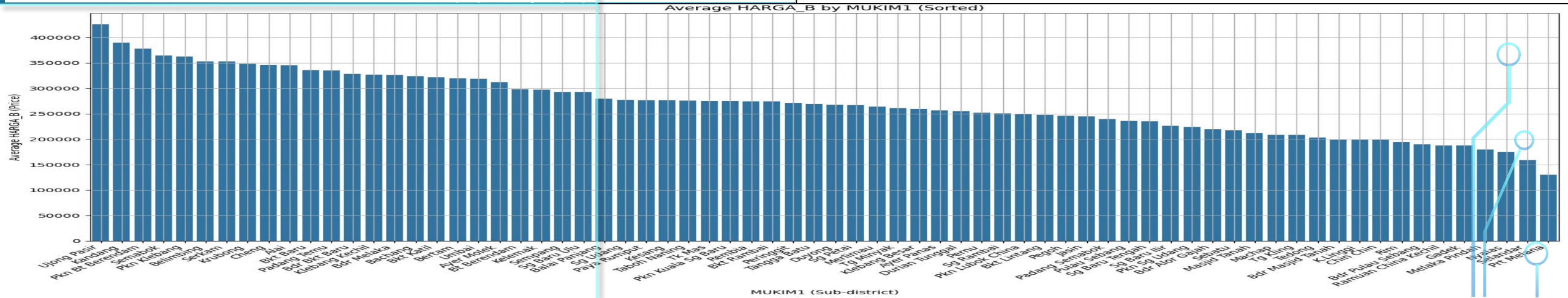
EXAMPLE OF EXPLORATORY DATA ANALYSIS



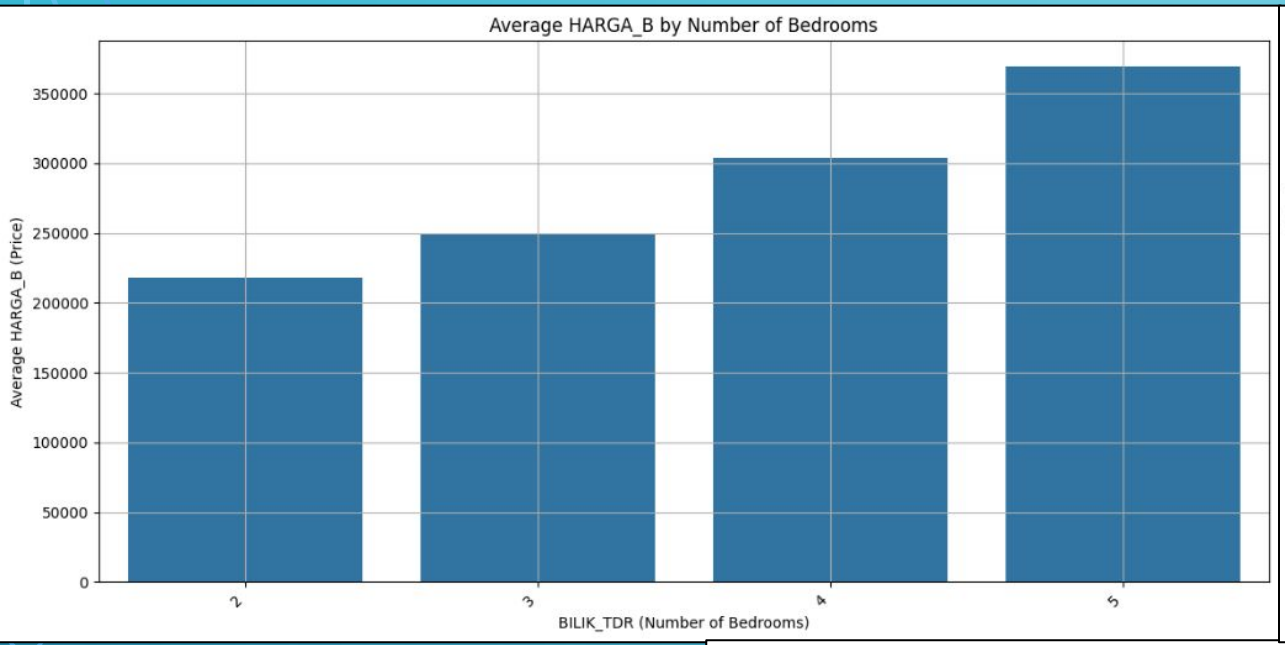
EXAMPLE OF EXPLORATORY DATA ANALYSIS

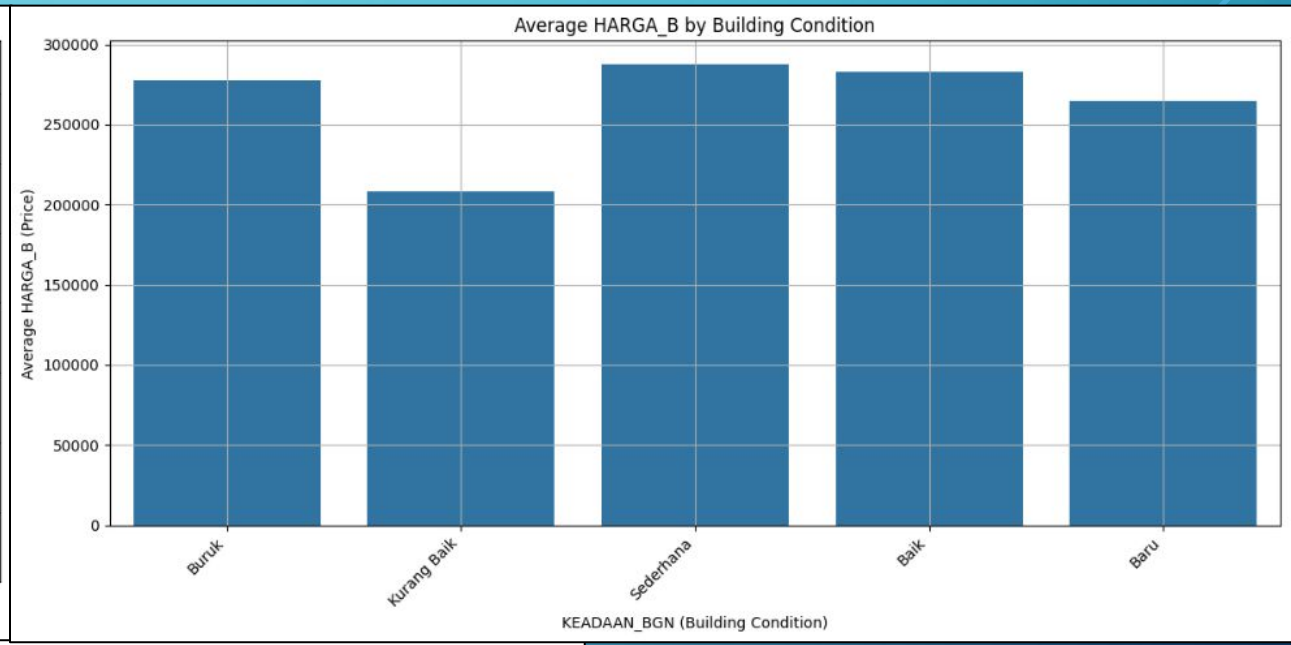


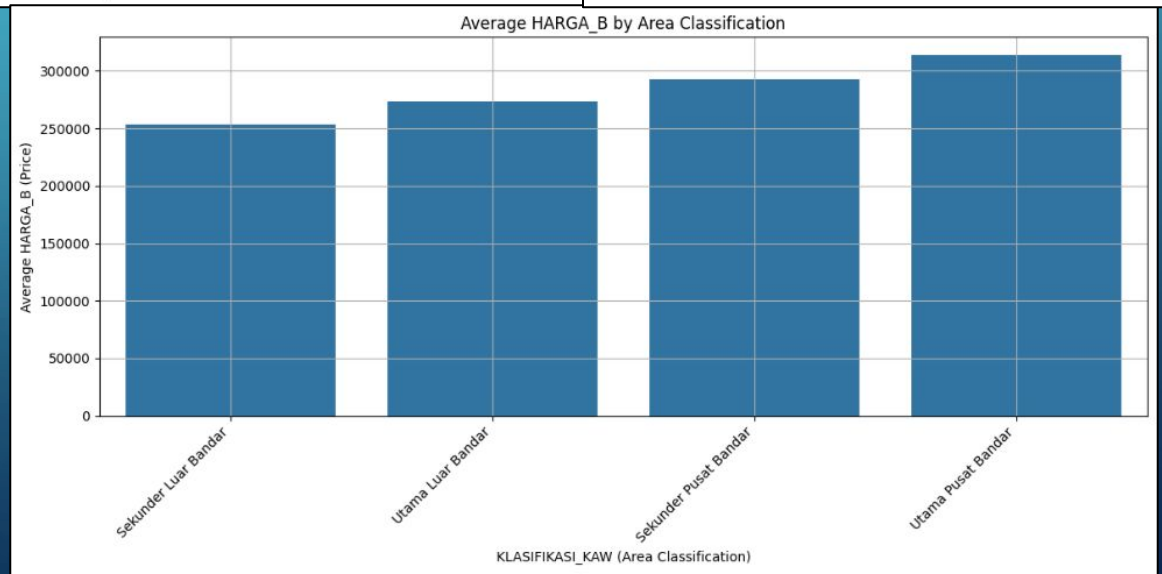




EXAMPLE OF EXPLORATORY DATA ANALYSIS








MACHINE LEARNING

ONE HOT ENCODER & LABEL ENCODER

5. Preparing the input data (Feature Engineering)

- Feature selection: Select the most relevant features (independent variables) for the model, potentially dropping features that add noise or do not contribute to predictions.
- Encoding categorical variables: Convert categorical data (e.g., city names) into numerical data using techniques like One-Hot Encoding or Label Encoding.
- Normalization/Standardization: Scale numeric features to ensure that they are on the same range, especially if your model is sensitive to feature magnitude (e.g., DT, RF, KNN, Neural Networks).

```
def get_one_hot(data, col):
    res = pd.get_dummies(data[col], dtype=int)
    res.astype(int)
    return res
```

```
categorical_columns = df_Melaka.dtypes[(df_Melaka.dtypes=='object')]
categorical_columns = categorical_columns.index.values
print(categorical_columns)
```

```
['KOD_CAW' 'STATE' 'BAHAGIAN' 'BAHAGIAN1' 'DAERAH1' 'JENIS_LOT' 'LOT_PLOT'
 'JENIS_HAKMILIK' 'NO_HAKMILIK' 'SECTOR' 'HALF YEAR' 'Quarter'
 'PM_PERTAMA' 'HALF YEAR1' 'QUARTER1' 'ALAMAT' 'SKIM' 'MUKIM1' 'CATEGORY'
 'PRO_TYPE' 'TKH_NILAI' 'UNIT_LUAS_LOT' 'PRICE RANGE' 'TKH_HANTAR'
 'P_LIBAT' 'P_LIBAT2' 'Expr1046' 'TRANSEFEROR' 'TRANSFEREE' 'Expr1051'
 'PEGAWAI_KENDALI' 'NAMA_BGN' 'JENIS_PEGANGAN' 'JENIS_BINAAN'
 'KEADAAN_BGN' 'KLASIFIKASI_BGN' 'KATEGORI_KAW' 'KLASIFIKASI_KAW'
 'STATUS_LOT' 'JENIS_TNH_PTN']
```

```
# One-hot encode the 'Category' column
one_hot_encoded = pd.get_dummies(df_Melaka[['PM_PERTAMA', 'JENIS_PEGANGAN']],
prefix=['PM_PERTAMA', 'JENIS_PEGANGAN'])
df_Melaka = pd.concat([df_Melaka, one_hot_encoded], axis=1)
df_Melaka.drop(['PM_PERTAMA', 'JENIS_PEGANGAN'], axis=1, inplace=True)

df_Melaka.head()
```

	KOD_CAW	KAIT	STATE	BAHAGIAN	BAHAGIAN1	DAERAH	DAERAH1	JENIS_LOT	LOT_PLOT	JENIS_HAKMILIK	...	KATEGORI_KAW	KLASIFIKASI_KAW	KOD_JENIS_TNH_PTN	STATUS_LOT	JENIS_TNH_PTN	UMUR_BGN	PM_PERTAMA_T	PM_PERTAMA_Y	JENIS_PEGANGAN_Kekal	JENIS_PEGANGAN_Pajakan
0	M0	1240005	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	2658	Old Entry Mukim Registry	...	Baik	Sekunder Pusat Bandar	NaN	N	None	32	True	False	False	True
14	M0	1242650	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	7765	Pajakan Mukim	...	Baik	Sekunder Pusat Bandar	NaN	N	None	11	True	False	False	True
18	M0	1242686	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	5122	Geran Mukim Peringkat Pertama	...	Baik	Sekunder Pusat Bandar	NaN	N	None	28	True	False	True	False
38	M0	3231653	Melaka	None	None	1	Melaka Tengah	Hakmilik Muktamad	9252	Pajakan Mukim	...	Baik	Sekunder Pusat Bandar	NaN	None	None	10	True	False	False	True

```
import pandas as pd
import pickle
from sklearn.preprocessing import LabelEncoder

# Copy the data to avoid modifying the original
df_Melaka_prep_le = df_Melaka_features.copy()

# Initialize LabelEncoder for nominal (non-ordinal) features
encoder = LabelEncoder()

# Define the ordinal mappings based on the provided features
ordinal_mappings = {
    'KEADAAN_BGN': {'Buruk': 0, 'Kurang Baik': 1, 'Sederhana': 2, 'Baik': 3,
    'Sangat Baik': 4, 'Baru': 5},
    'KATEGORI_KAW': {'Kurang Baik': 0, 'Sederhana': 1, 'Baik': 2, 'Sangat
    Baik': 3},
    'KLASIFIKASI_KAW': {'Pedalaman': 0, 'Sekunder Luar Bandar': 1, 'Utama Luar
    Bandar': 2, 'Sekunder Pusat Bandar': 3, 'Utama Pusat Bandar': 4}
}

# Strip whitespace and standardize case before mapping
df_Melaka_prep_le['KEADAAN_BGN'] = df_Melaka_prep_le['KEADAAN_BGN'].str.strip().
str.title()
df_Melaka_prep_le['KATEGORI_KAW'] = df_Melaka_prep_le['KATEGORI_KAW'].str.strip
().str.title()
df_Melaka_prep_le['KLASIFIKASI_KAW'] = df_Melaka_prep_le['KLASIFIKASI_KAW'].str.
strip().str.title()

# Apply ordinal mappings for the specified columns
for column, mapping in ordinal_mappings.items():
    df_Melaka_prep_le[column] = df_Melaka_prep_le[column].map(mapping)

# Save ordinal mappings for future use
with open('ordinal_mappings.pkl', 'wb') as file:
    pickle.dump(ordinal_mappings, file)

# Initialize a dictionary to save LabelEncoders for nominal columns
label_encoders = {}

# Apply LabelEncoder for nominal features (not ordinal) and save the encoders
for column in df_Melaka_prep_le.columns:
    if df_Melaka_prep_le[column].dtype == 'object':  # Only apply LabelEncoder
    to object (categorical) columns
```

```
[ ]
# Define the list of columns to use as features
selected_columns = ['DAERAH1', 'PM_PERTAMA_T', 'PM_PERTAMA_Y', 'SKIM',
'MUKIM1', 'B_TINGKAT', 'LUAS_LOT',
                    'LUAS_LOT_BGN', 'UMUR_BGN', 'BILIK_TDR',
                    'JENIS_PEGANGAN_Kekal', 'JENIS_PEGANGAN_Pajakan',
                    'KEADAAN_BGN', 'KATEGORI_KAW', 'KLASIFIKASI_KAW']

# Select the columns from Final_Model_prep to create the features DataFrame
df_Melaka_features = df_Melaka_prep[selected_columns]

# Display the selected features
df_Melaka_features.head()
```

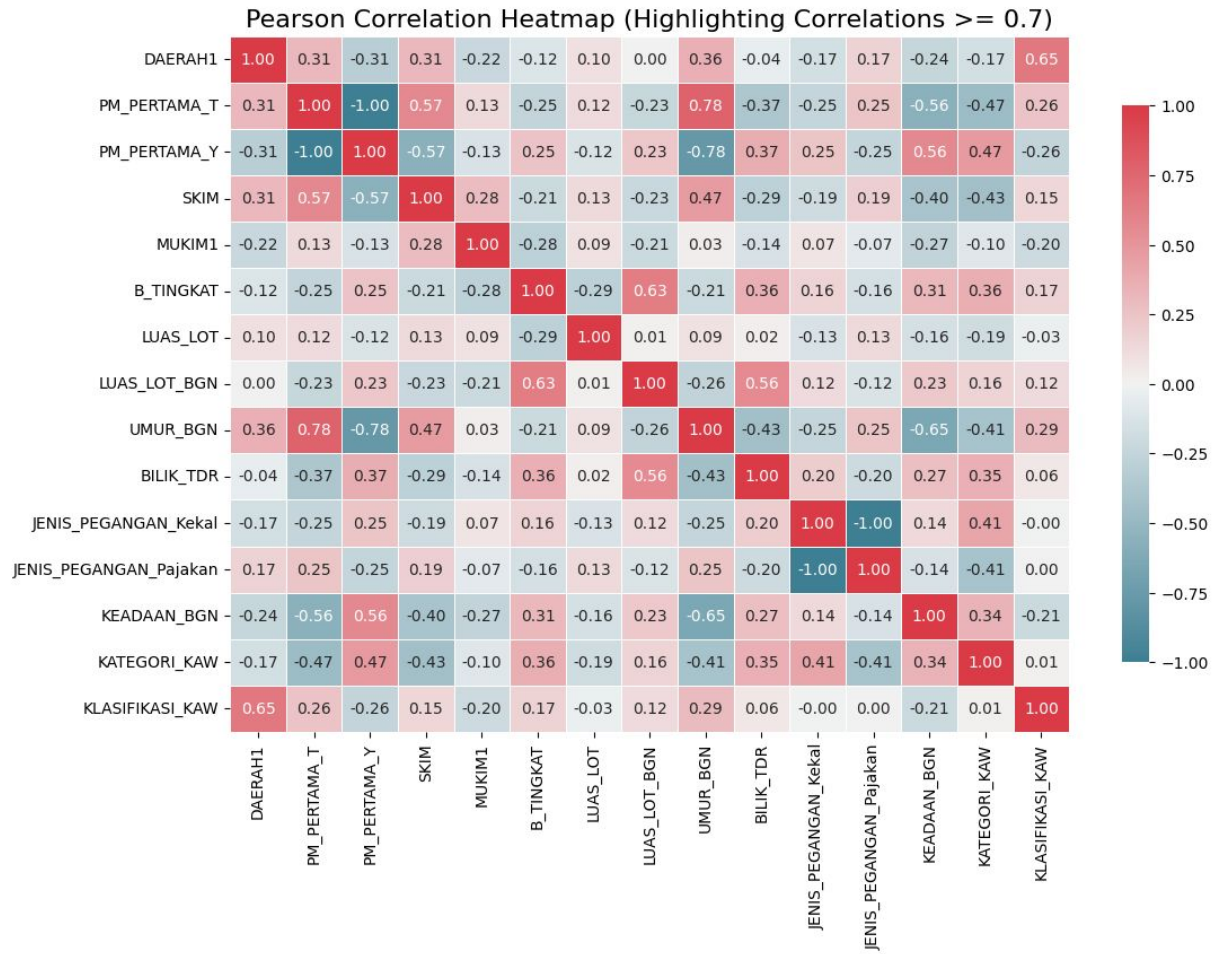
	DAERAH1	PM_PERTAMA_T	PM_PERTAMA_Y	SKIM	MUKIM1	B_TINGKAT	LUAS_LOT	LUAS_LOT_BGN	UMUR_BGN	BILIK_TDR	JENIS_PEGANGAN_Kekal	JENIS_PEGANGAN_Pajakan	KEADAAN_BGN	KATEGORI_KAW	KLASIFIKASI_KAW
0	Melaka Tengah	True	False	TMN BERTAM JAYA FASA 1	Cheng	1	107.000	69.49	32	3	False	True	Baik	Baik	Sekunder Pusat Bandar
14	Melaka Tengah	True	False	TMN RAMBAI IDAMAN	Bkt Rambai	1	130.000	83.80	11	3	False	True	Baik	Baik	Sekunder Pusat Bandar
18	Melaka Tengah	True	False	TMN SERI RAMBAI 1	Bkt Rambai	1	121.000	76.18	28	3	True	False	Baik	Baik	Sekunder Pusat Bandar
38	Melaka Tengah	True	False	TMN DESA TANJUNG MINYAK (SERI BERTAM)	Tg Minyak	1	130.000	74.30	10	3	False	True	Baik	Baik	Sekunder Pusat Bandar
75	Melaka Tengah	True	False	TMN SINN	Ujong Pasir	1	143.066	90.20	48	3	True	False	Baik	Sangat Baik	Utama Pusat Bandar

```
[ ]
df_Melaka_target = df_Melaka_prep['HARGA_B']
df_Melaka_target
```

	HARGA_B

IDENTIFY FEATURES AND TARGET

PEARSON CORRELATION HEATMAP



DATA NORMALIZATION

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
import pickle

# Step 1: Drop the 'UMUR_BGN' column before scaling
# -------------------------------------------------
df_Melaka_prep_le_dropped = df_Melaka_prep_le.drop(columns=['UMUR_BGN'])

# Step 2: Initialize the scaler
scaler = StandardScaler()

# Step 3: Fit the scaler to the data and transform the data
df_Melaka_scaled = scaler.fit_transform(df_Melaka_prep_le_dropped)

# Step 4: Save the scaler as a .pkl file for future use
with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)

# Optional: Convert scaled data back to a DataFrame (if you prefer working with DataFrames)
df_Melaka_scaled_df = pd.DataFrame(df_Melaka_scaled, columns=df_Melaka_prep_le_dropped.columns)

# Check the scaled data
print(df_Melaka_scaled_df.head())
```

```
    DAERAH1  PM_PERTAMA_T  PM_PERTAMA_Y      SKIM    MUKIM1  B_TINGKAT  \
0  1.101874      1.157377     -1.157377 -0.425286 -0.439931  -0.811369
1  1.101874      1.157377     -1.157377  1.226315 -0.530252  -0.811369
2  1.101874      1.157377     -1.157377  1.661334 -0.530252  -0.811369
3  1.101874      1.157377     -1.157377  0.098213  1.772935  -0.811369
4  1.101874      1.157377     -1.157377  1.867785  1.863256  -0.811369

   LUAS_LOT  LUAS_LOT_BGN  BILIK_TDR  JENIS_PEGANGAN_Kekal  \
0 -0.679459     -1.077576  -0.854077             -1.480538
1 -0.180470     -0.571182  -0.854077             -1.480538
2 -0.375726     -0.840834  -0.854077              0.675430
3 -0.180470     -0.907362  -0.854077             -1.480538
4  0.102999     -0.344702  -0.854077              0.675430

   JENIS_PEGANGAN_Pajakan  KEADAAN_BGN  KATEGORI_KAW  KLASIFIKASI_KAW
0                1.480538    -0.973182     -0.483216         1.031895
1                1.480538    -0.973182     -0.483216         1.031895
2               -0.675430    -0.973182     -0.483216         1.031895
3                1.480538    -0.973182     -0.483216         1.031895
4               -0.675430    -0.973182      0.954655         2.055928
```

DATA SPLITTING / TRAIN AND MODEL TESTING

6. Data Splitting

Before start the model training, we need to split our data into Training and Testing Set. Training set is the data that used by the algorithm to search for interesting patterns that able to be used to classify the differentypes of species.

Steps:

1. Load the `model_selection` module from the `scikit-learn` library
2. Then split the data into train and test set using the `train_test_split()` function.
3. Split the training and testing set with ratio 70:30

```
[ ]
from sklearn.model_selection import train_test_split

# Assuming 'HARGA_B' is your target variable (dependent variable)
# and all other columns are features (independent variables)
X = df_Melaka_scaled_df  # Features (scaled data)
y = df_Melaka_target  # Target variable (before scaling)

# Split the data: 70% training, 30% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Check the shapes of the resulting datasets
print(f"Training features: {X_train.shape}")
print(f"Testing features: {X_test.shape}")
print(f"Training target: {y_train.shape}")
print(f"Testing target: {y_test.shape}")
```

```
Training features: (2734, 14)
Testing features: (1173, 14)
Training target: (2734,)
Testing target: (1173,)
```

7. Train and Test Model

In the following sections, we will explore and use different machine learning algorithms in `python` through the `scikit-learn` library. Each algorithm requires to import specific modules within the library.

Here you are 4 models:

- Decision Tree
- Random Forrest
- K-Nearest Neighbors
- Tensorflow (Neural Networks)

```
[ ]
from sklearn.tree import DecisionTreeRegressor

dt_model = DecisionTreeRegressor()

dt_model.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor()
```

```
[ ]
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor()
```

```
[ ]
from sklearn.neighbors import KNeighborsRegressor

knn_model = KNeighborsRegressor()
knn_model.fit(X_train, y_train)
```

```
KNeighborsRegressor
KNeighborsRegressor()
```

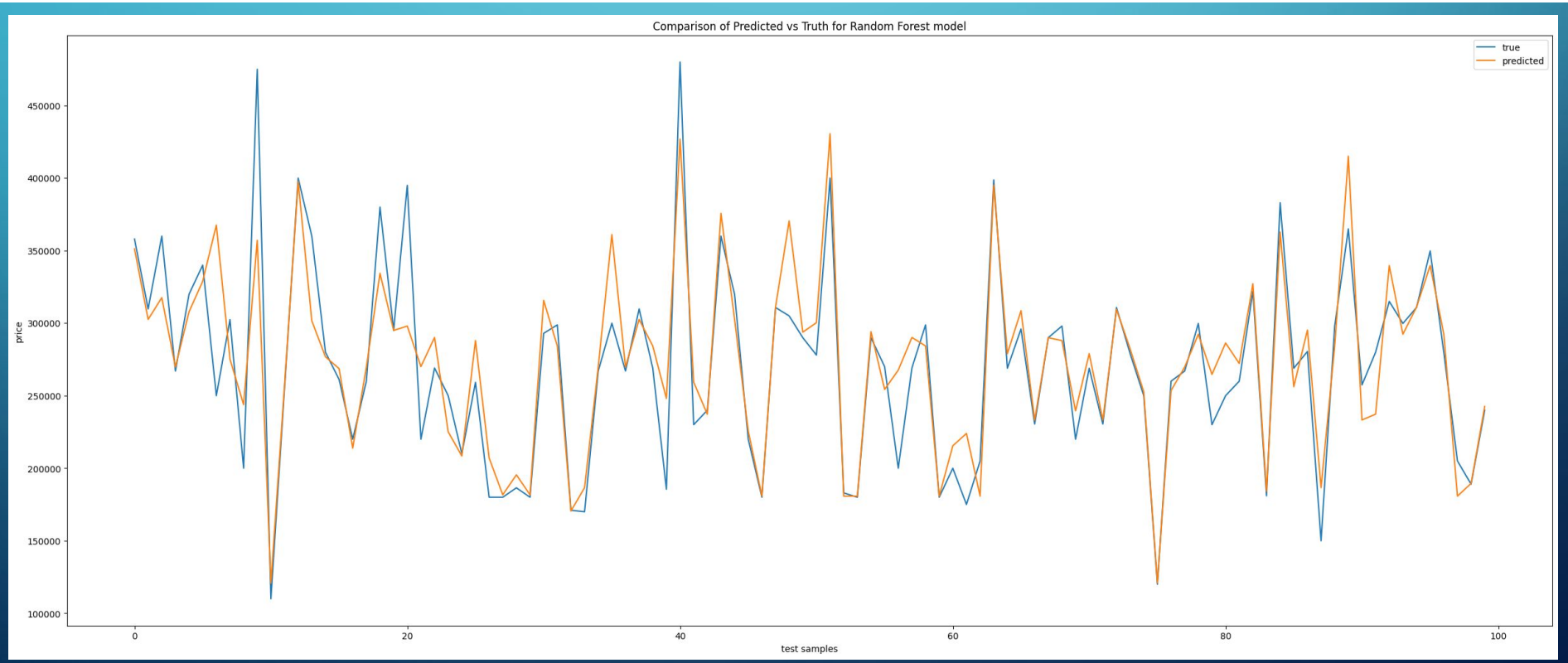
DATA SPLITTING / TRAIN AND MODEL TESTING

Simple Neural Network in TensorFlow using Keras

```
[ ]
import tensorflow as tf
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, mean_absolute_percentage_error
import numpy as np

# Create a simple neural network model using TensorFlow
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),  # Input layer and first hidden layer
    tf.keras.layers.Dense(64, activation='relu'),  # Second hidden layer
    tf.keras.layers.Dense(1)  # Output layer for regression (no activation)
])

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Fit the model to the training data
model.fit(X_train, y_train, epochs=100, batch_size=32, verbose=1)  # Train for 100 epochs

# Make predictions on the test set
tf_predictions = model.predict(X_test).flatten()  # Flatten the predictions

# Define a function to evaluate the model (same as above)
def evaluate_model(y_true, y_pred, model_name):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    mape = mean_absolute_percentage_error(y_true, y_pred) * 100  # Expressed as percentage

    # Print the evaluation metrics
    print(f"{model_name} - MSE: {mse:.2f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}, R²: {r2:.2f}, MAPE: {mape:.2f}%\n")

    return mse, rmse, mae, r2, mape

# Evaluate the TensorFlow model
tf_mse, tf_rmse, tf_mae, tf_r2, tf_mape = evaluate_model(y_test, tf_predictions, "TensorFlow Neural Network")
```

MODEL EVALUATION

```
Decision Tree Regressor - MSE: 1614272288.01, RMSE: 40178.01, MAE: 22923.97, R²: 0.71, MAPE: 8.50%
Random Forest Regressor - MSE: 939038895.21, RMSE: 30643.74, MAE: 18761.34, R²: 0.83, MAPE: 7.08%
K-Nearest Neighbors Regressor - MSE: 1356977698.13, RMSE: 36837.18, MAE: 23509.20, R²: 0.76, MAPE: 8.98%
TensorFlow Neural Network - MSE: 1902369961.59, RMSE: 43616.17, MAE: 32191.20, R²: 0.66, MAPE: 12.69%
```



HYPERPARAMETER TUNING (GRID SEARCH)

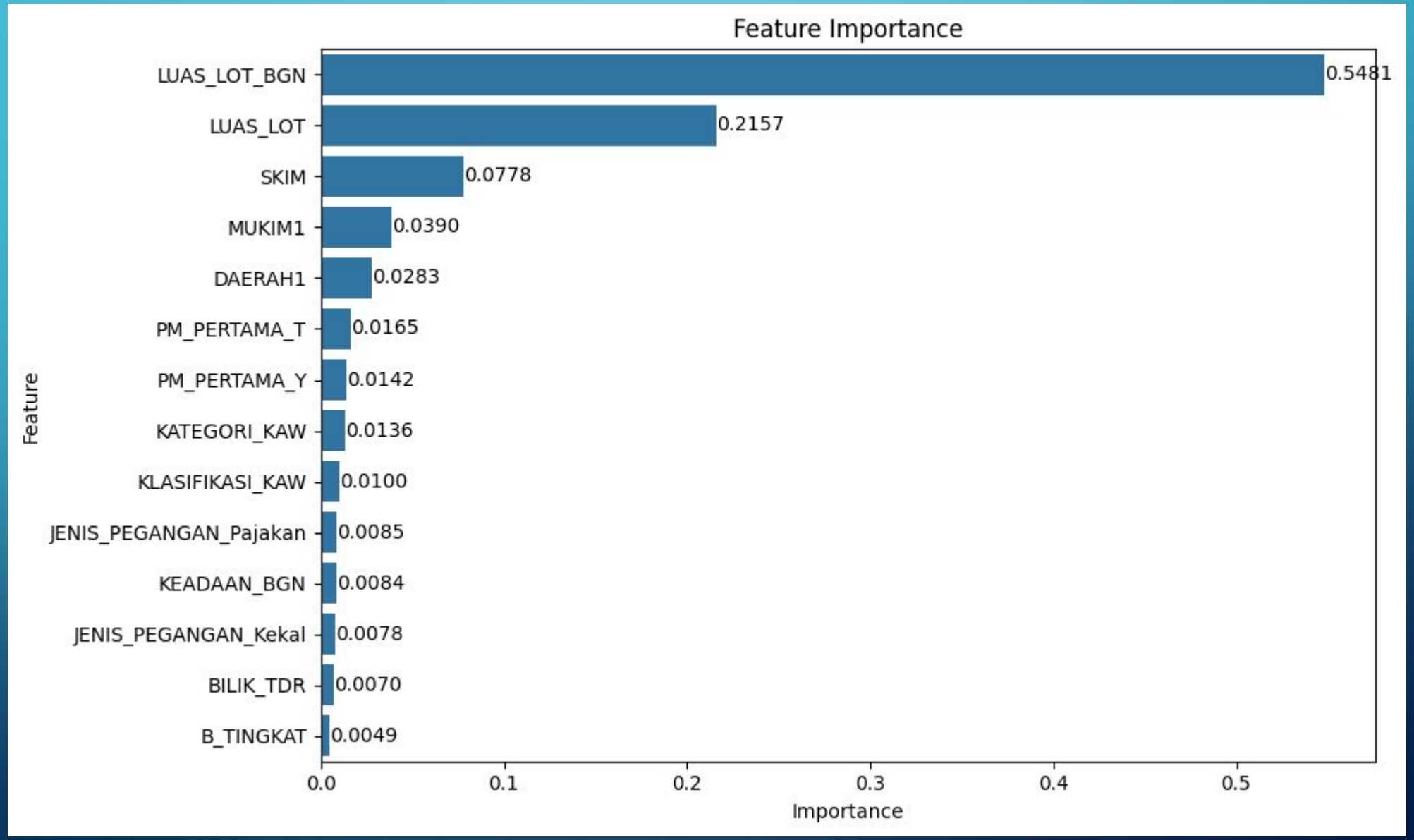
9. Model Selection

- *Choose the best model to run in prediction for new dataset.*

```
[ ]
from sklearn.model_selection import GridSearchCV

# Define the hyperparameter grid for Random Forest Regressor
param_grid = {
    'n_estimators': [100, 200, 300],  # Number of trees
    'max_depth': [None, 10, 20, 30],  # Depth of the trees
    'min_samples_split': [2, 5, 10],  # Minimum samples required to split an
    internal node
    'min_samples_leaf': [1, 2, 4],    # Minimum samples required at a leaf node
    'bootstrap': [True, False]        # Bootstrap samples when building trees
}

# Initialize RandomForestRegressor
rf_model = RandomForestRegressor(random_state=42)

# Use GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5,
n_jobs=-1, verbose=2, scoring='neg_mean_squared_error')

# Fit the grid search model
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_
print(f"Best hyperparameters: {best_params}")

# Use the best model to make predictions
best_rf_model = grid_search.best_estimator_

# Fit the best model on the training set
best_rf_model.fit(X_train, y_train)

# Make predictions using the best RandomForest model
best_rf_predictions = best_rf_model.predict(X_test)

# Re-evaluate the model using the best hyperparameters
rf_mse, rf_rmse, rf_mae, rf_r2, rf_mape = evaluate_model(y_test,
best_rf_predictions, "Best Random Forest Regressor")

```

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
Best hyperparameters: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimat
Best Random Forest Regressor - MSE: 929780524.76, RMSE: 30492.30, MAE: 18987.60, R²: 0.83, MAPE: 7.16%
```

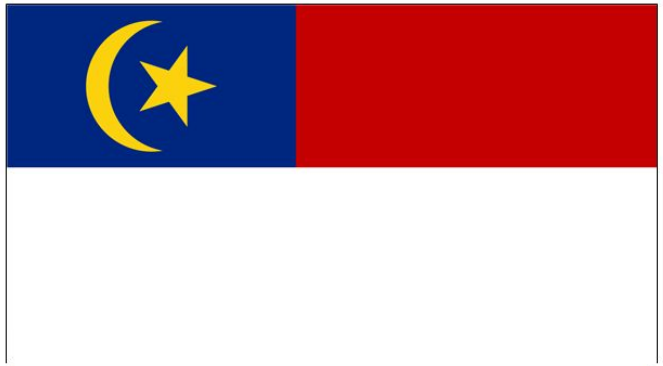
FEATURES IMPORTANCE



DASHBOARD USING TABLEAU

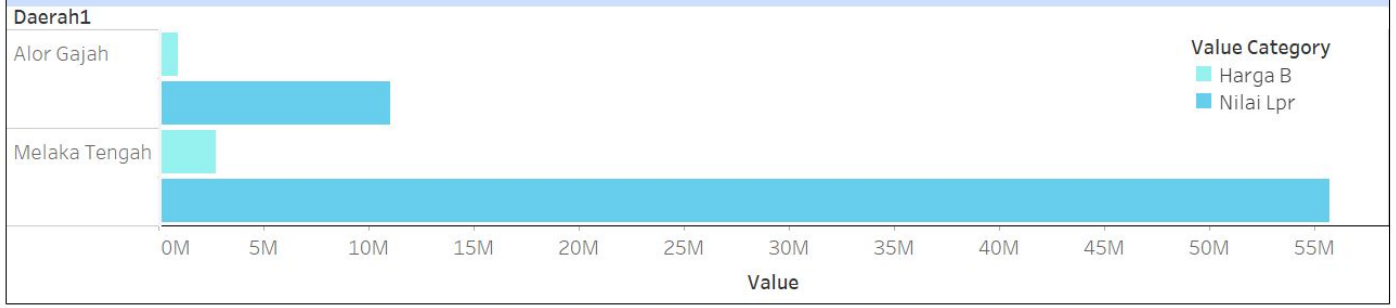
REVIEW OF VALUE OF TRANSACTIONS FOR NAPIC PUBLICATION

(SINGLE AND DOUBLE STOREY TERRACE HOUSE)

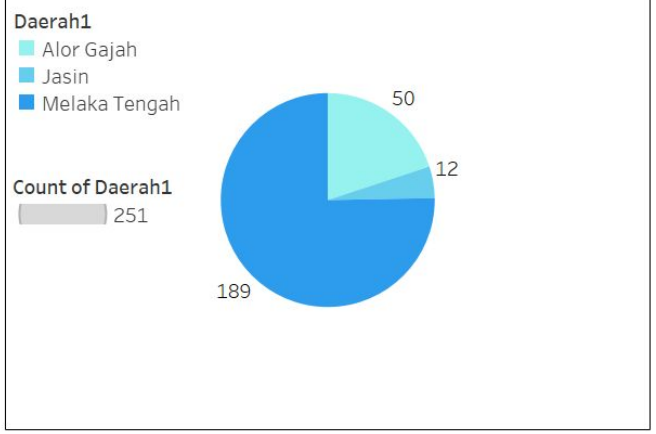


Year1
2024

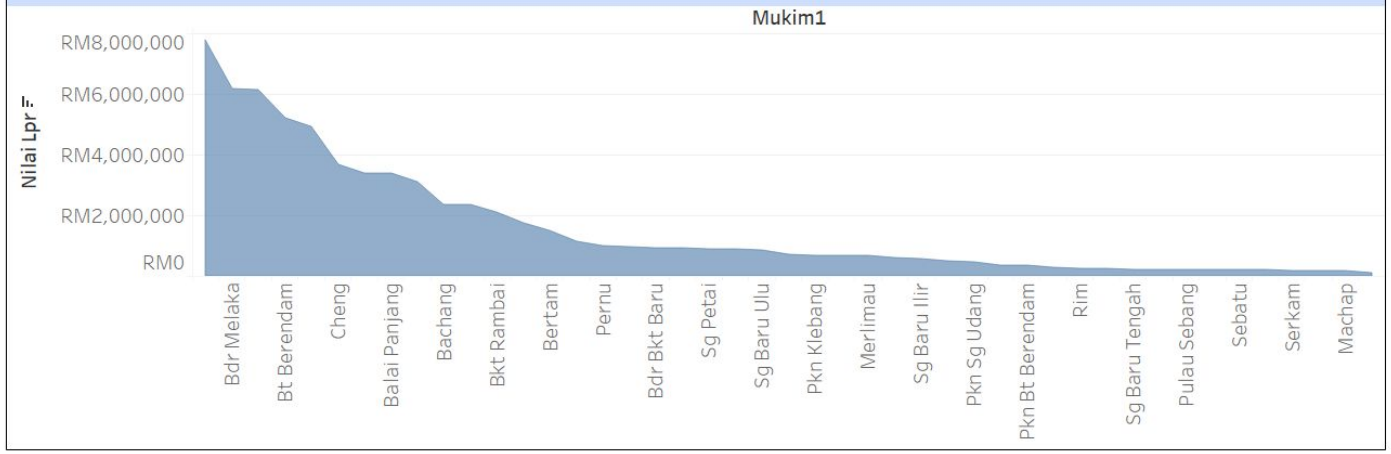
Value for Harga_b and Nilai_lpr by District



Volume of Transactions by District

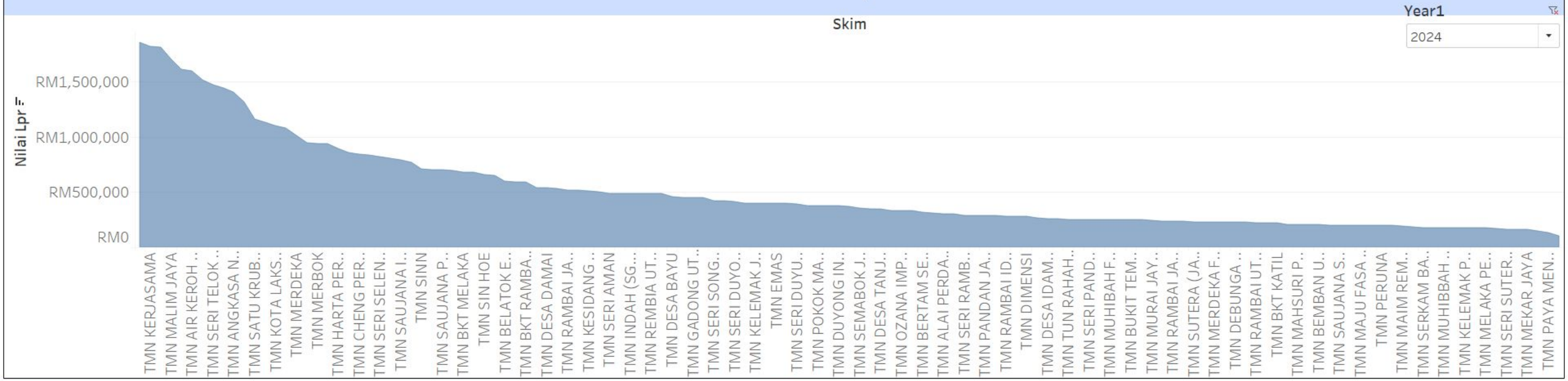


Value of Transactions by Mukim



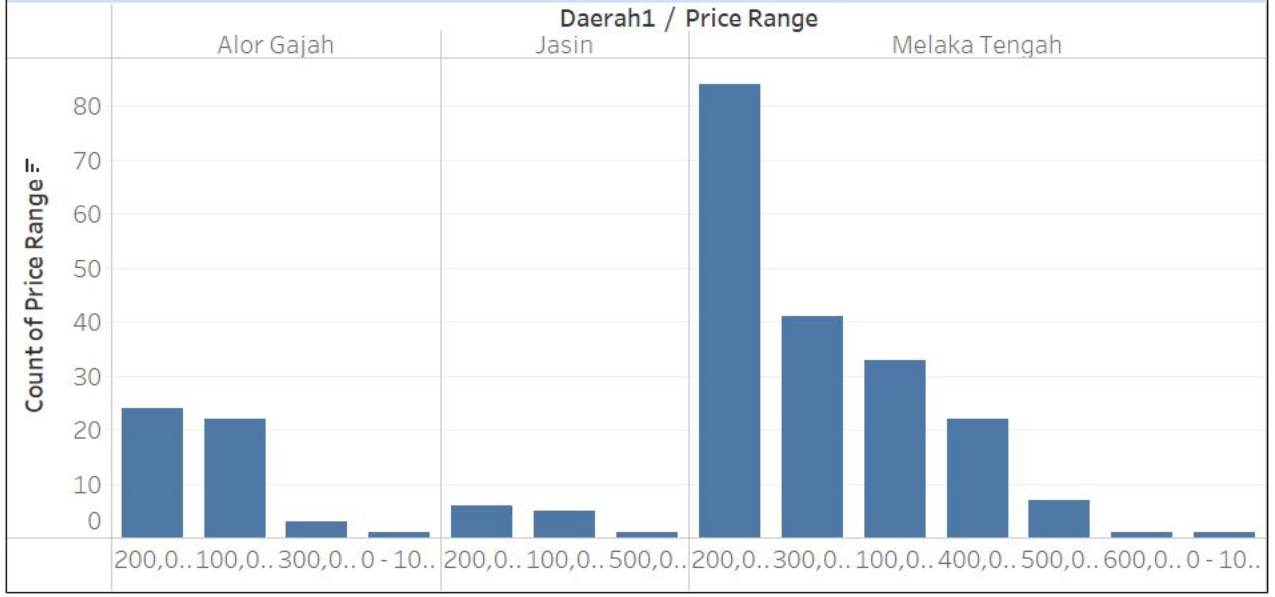
heet 6.2 | Sheet 7.3 | Sheet 8.3 | Sheet 9.3 | Sheet 10.3 | Sheet 11.3 | Sheet 12.3 | Sheet 13.4 | Sheet 14.4 | Sheet 15.4 | Sheet 16.4 | **Dashboard 1** | Dashboard 2 | Dashboard 3 | Dashboard 4

Value of Transactions by Skim



Volume of Transactions by Nilai Ipr Price Range

Value of Transactions by Price Range (District)



Comparison Between Nilai_lpr and Predicted Value

2024

NILAI_LPR

RM69,626,000

PREDICTED_VALUE

RM72,938,006

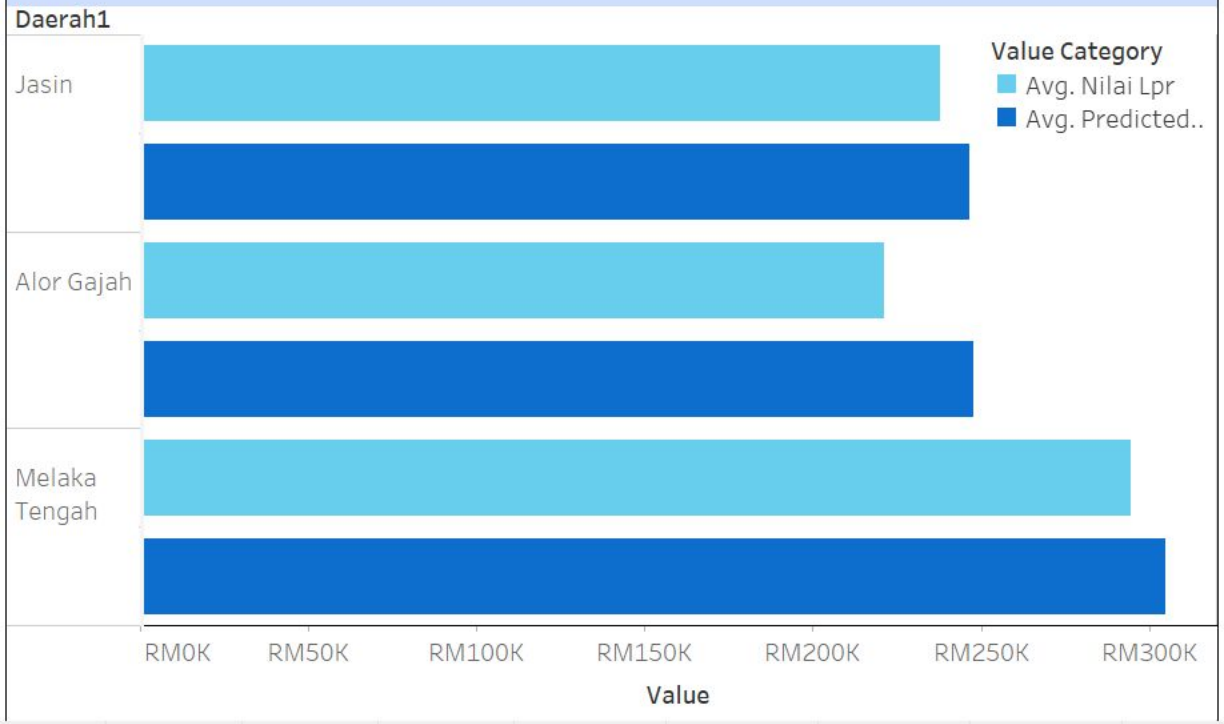
DIFFERENCE IN VALUE

▲3,312,006

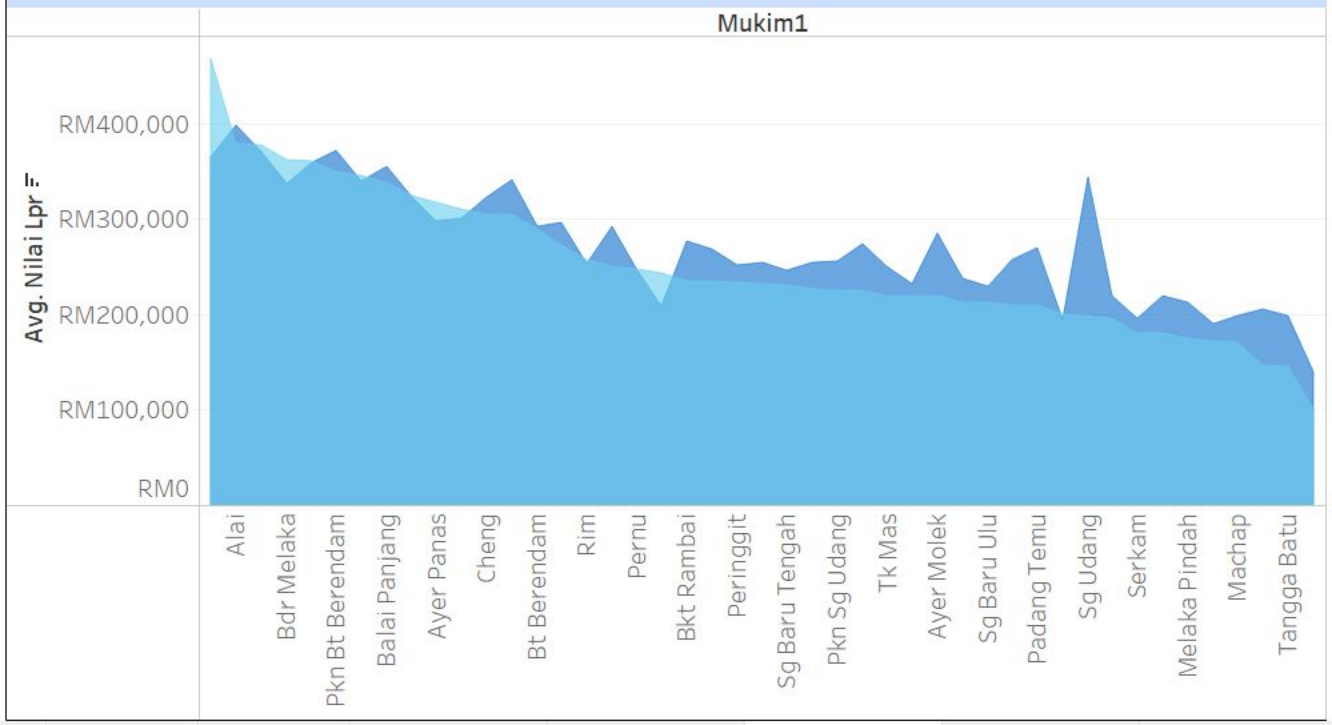
DIFFERENCE IN PERCENTAGE

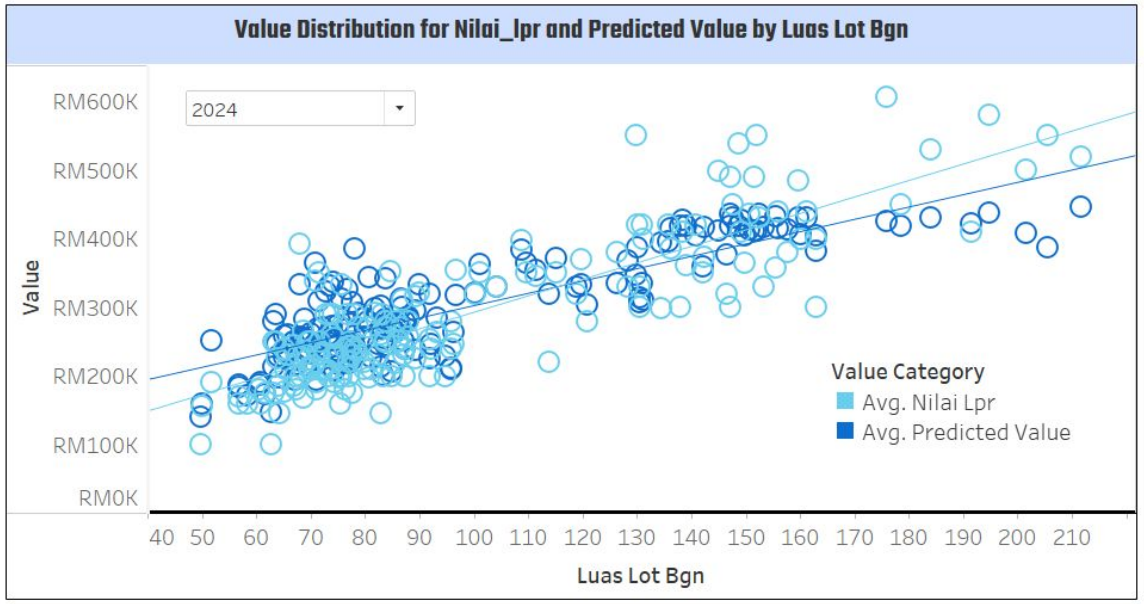
▲4.8%

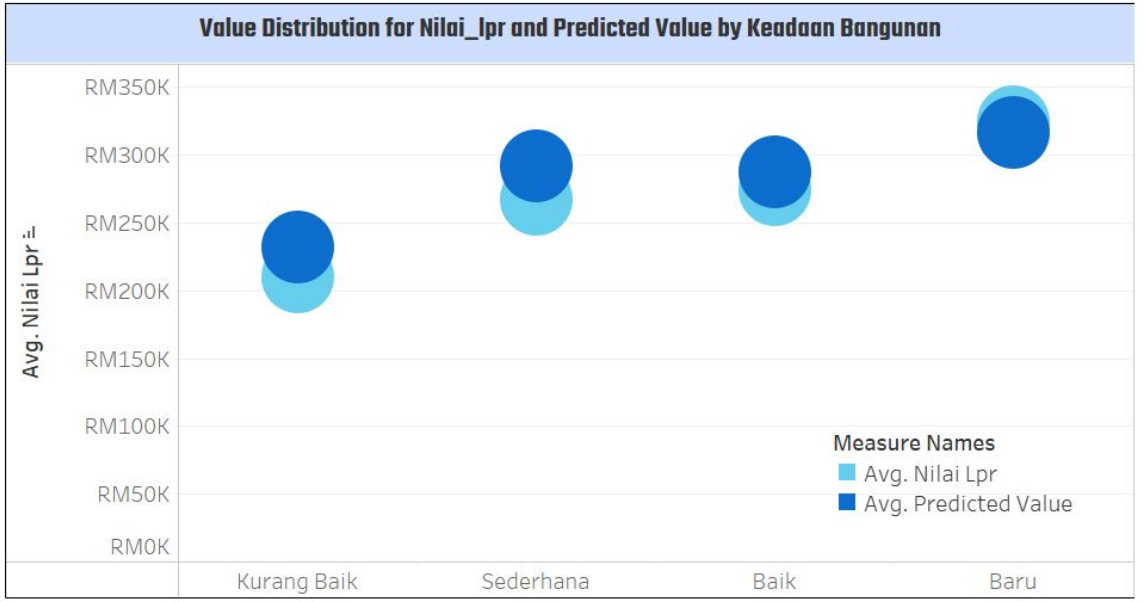
Value for Nilai_lpr and Predicted Value by District

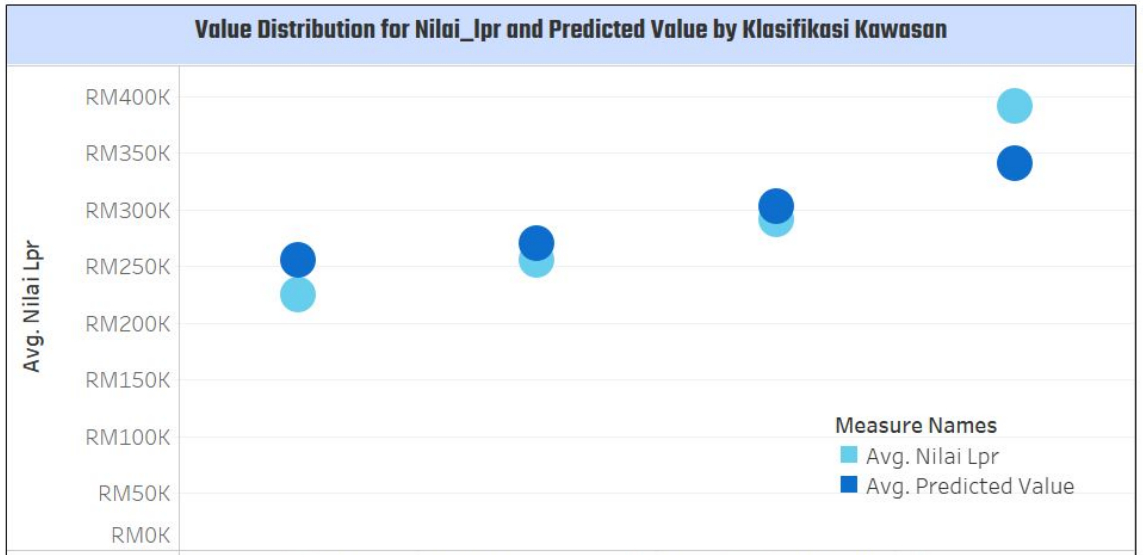


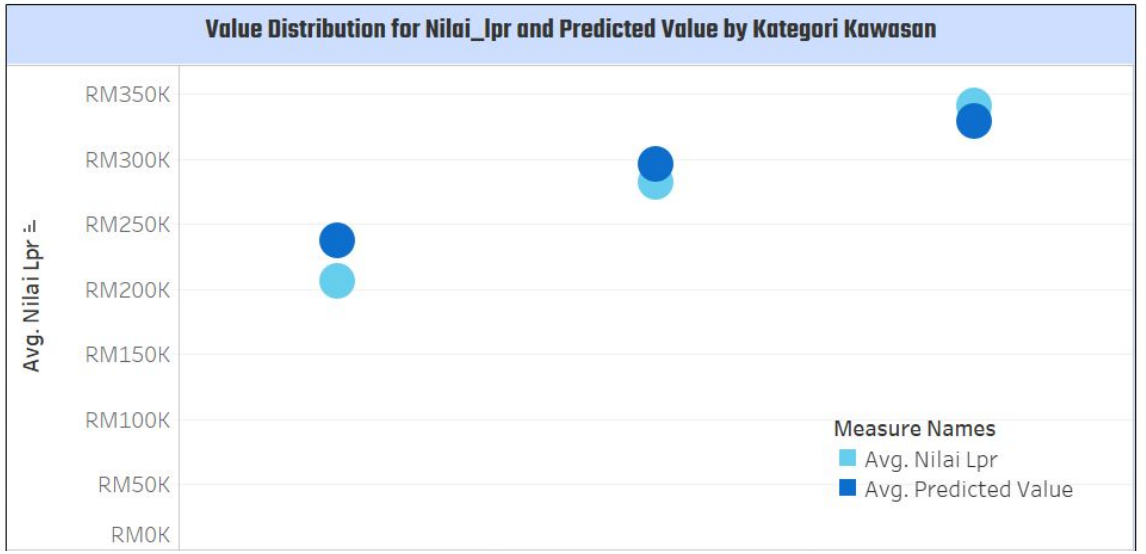
Value of Transactions by Mukim



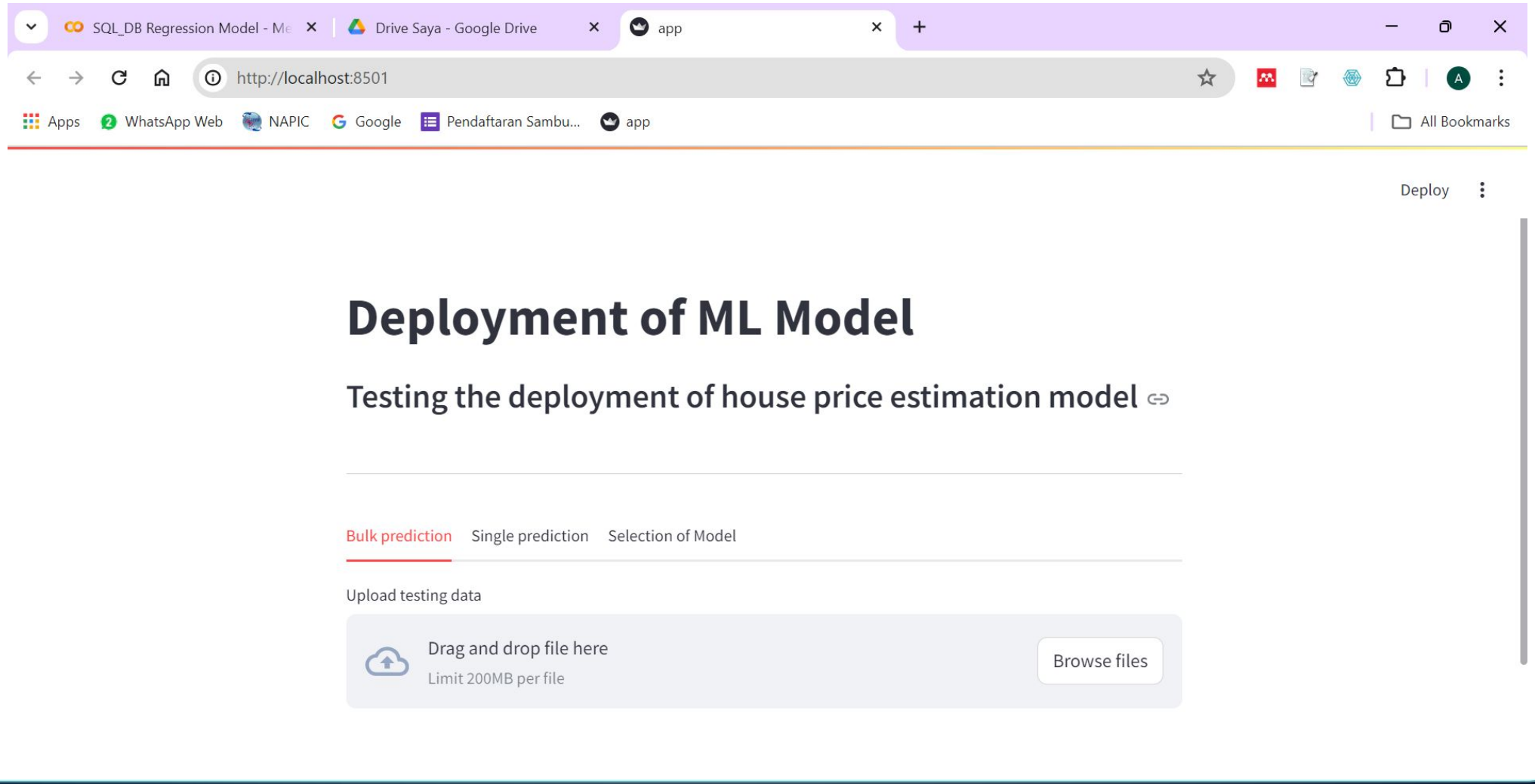

Value Distribution for Nilai_lpr and Predicted Value by Luas Lot Bgn
2024
RM600K
RM500K
RM400K
RM300K
RM200K
RM100K
RM0K
Value
40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210
Luas Lot Bgn
Value Category
Avg. Nilai Lpr
Avg. Predicted Value


Value Distribution for Nilai_lpr and Predicted Value by Keadaan Bangunan
RM350K
RM300K
RM250K
RM200K
RM150K
RM100K
RM50K
RM0K
Avg. Nilai Lpr
Kurang Baik
Sederhana
Baik
Baru
Measure Names
Avg. Nilai Lpr
Avg. Predicted Value


Value Distribution for Nilai_lpr and Predicted Value by Klasifikasi Kawasan
RM400K
RM350K
RM300K
RM250K
RM200K
RM150K
RM100K
RM50K
RM0K
Avg. Nilai Lpr
Measure Names
Avg. Nilai Lpr
Avg. Predicted Value


Value Distribution for Nilai_lpr and Predicted Value by Kategori Kawasan
RM350K
RM300K
RM250K
RM200K
RM150K
RM100K
RM50K
RM0K
Avg. Nilai Lpr
Measure Names
Avg. Nilai Lpr
Avg. Predicted Value

DASHBOARD USING STREAMLIT


SQL_DB Regression Model - Me
Drive Saya - Google Drive
app
http://localhost:8501
Apps
WhatsApp Web
NAPIC
Google
Pendaftaran Sambu...
app
All Bookmarks
Deploy
Deployment of ML Model
Testing the deployment of house price estimation model
Bulk prediction
Single prediction
Selection of Model
Upload testing data
Drag and drop file here
Limit 200MB per file
Browse files

Testing the deployment of house price estimation model

Bulk prediction | Single prediction | Selection of Model

House Information:

DAERAH1

Alor Gajah

MUKIM1

Alai

SKIM

BANDAR BOTANI PARKLAND

PM PERTAMA

T

Number of Floors (B_TINGKAT)

1

Lot Size (LUAS_LOT)

Building Size (LUAS_LOT_BGN)

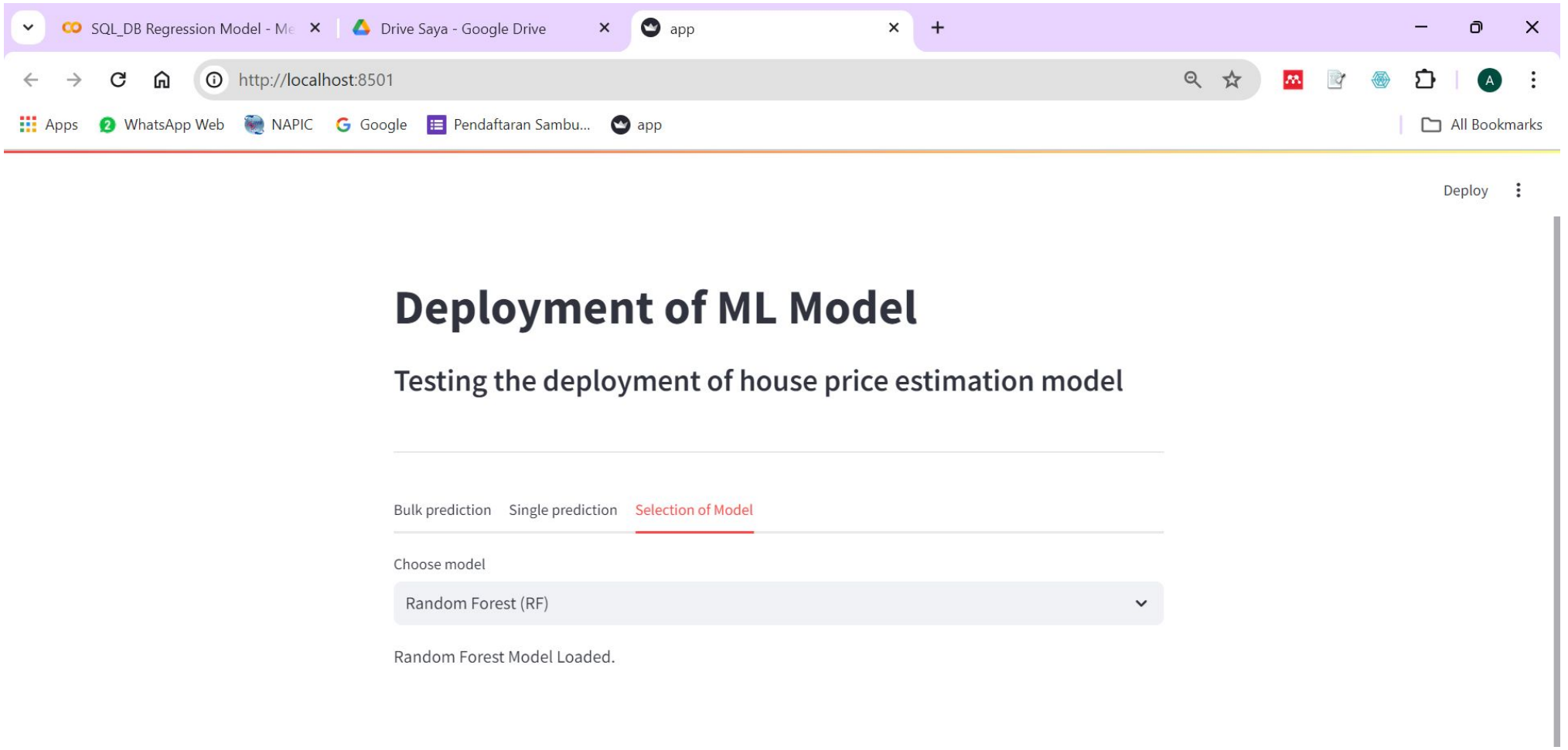
Number of Bedrooms (BILIK_TDR)

1

Jenis Pegangan (Kekal or Pajakan)

Kekal

KEADAAN BANGUNAN


SQL_DB Regression Model - Me
Drive Saya - Google Drive
app
http://localhost:8501
Apps
WhatsApp Web
NAPIC
Google
Pendaftaran Sambu...
app
All Bookmarks
Deploy
Deployment of ML Model
Testing the deployment of house price estimation model
Bulk prediction
Single prediction
Selection of Model
Choose model
Random Forest (RF)
Random Forest Model Loaded.

2023_2024_predictions.csv 479.7KB

Prediction Results:

	Sample_Number	NILAI_LPR	Predicted_Value	Percentage_Change
0	1	270,000	316,509.8693	17.2259
1	2	500,000	366,630.7581	-26.6738
2	3	360,000	412,946.3079	14.7073
3	4	314,000	302,449.3663	-3.6785
4	5	260,000	339,255.1953	30.4828
5	6	290,000	229,307.8782	-20.9283
6	7	550,000	432,211.7149	-21.4161
7	8	390,000	400,938.9228	2.8049
8	9	210,000	228,373.8041	8.7494
9	10	270,000	220,964.7274	-18.1612

Summary:

Total NILAI_LPR: 245399800

Total Predicted Value: 270818967.7822471

Overall Percentage Change: 10.36%

Deployment of ML Model

Testing the deployment of house price estimation model

Bulk prediction | Single prediction | Selection of Model

House Information:

DAERAH1: Jasin

MUKIM1: Merlimau

SKIM: TMN MERLIMAU JAYA

PM PERTAMA: T

Number of Floors (B_TINGKAT): 1

Lot Size (LUAS_LOT): 400

Building Size (LUAS_LOT_BGN): 200

Number of Bedrooms (BILIK_TDR): 1

Jenis Pegangan (Kekal or Pajakan): Kekal

KEADAAN BANGUNAN: Baik

KATEGORI KAWASAN: Baik

KLASIFIKASI KAWASAN: Sekunder Luar Bandar

Estimate Price

Predicted price: 419696.606725139


THANK YOU