UTM UNIVERSITI TEKNOLOGI MALAYSIA

CAPSTONE PROJECT:

CLASSIFICATIONS OF PROPERTY TRANSACTIONS PRICE IN MELAKA AND JOHOR

Prepared by:

Mohammad Zaihan bin Shamsuddin

Supervisor:

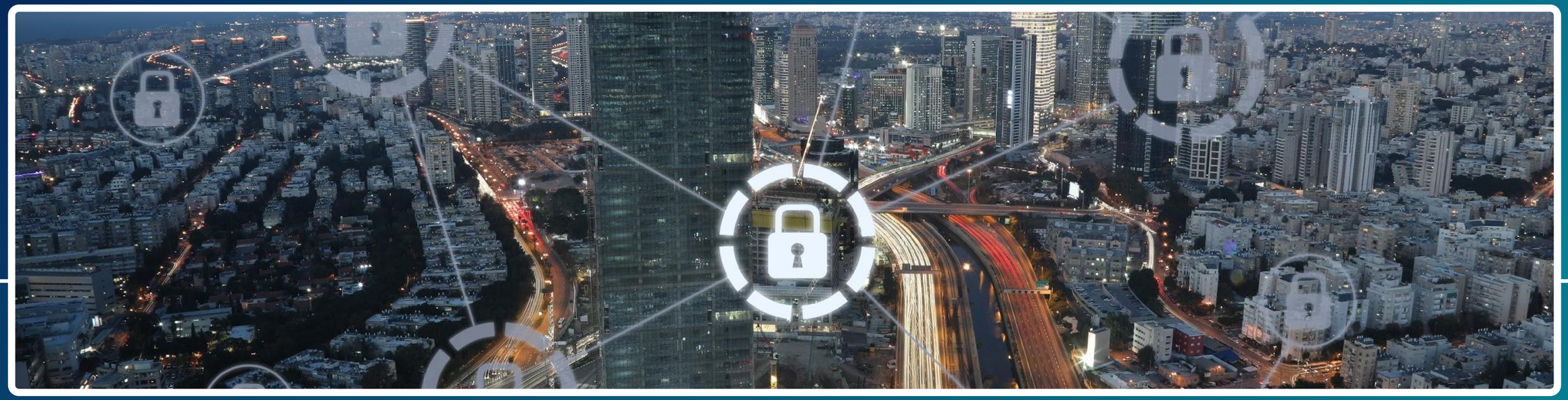
Dr. Chan Weng Hoe

CAPSTONE PROJECT BACKGROUND

- ✔ Title : Classification of Property Transactions Price
- ✔ Location : Melaka & Johor
- ✔ Type of Property : Single & Double Storey Terraced House
- ✔ Time Period : 2019 – 2023

CONTENT

- Problem Statement
- Data Description
- Data Cleaning
- Data Preparation
- Exploratory Data Analysis
- Machine Learning
- Dashboard



PROBLEM STATEMENT

The real estate markets in Melaka and Johor are diverse, with property prices varying based on several factors such as built-up area, location, property type, and market conditions. Stakeholders such as investors, developers, buyers, and policymakers require a reliable method to classify properties into different pricing tiers—low-tier, mid-tier, and high-tier—to make informed decisions. Currently, there is no efficient, data-driven method available for quickly assessing these price tiers across a large number of transactions.

Main Goals of Classifications

Categories:

Low tier

Mid tier

High tier

Model offer structured approach

Helps stakeholders and investors make faster and smart decisions

DATA DESCRIPTION

It captures a range of property attributes, including:

- Location:NEGERI (State): Indicates if the property is in Melaka or Johor.
- DAERAH1 (District): Specifies the district within each state, providing more granular location data.
- Property Characteristics:B_TINGKAT (Storey Type): Identifies the property as either single-storey or double-storey.
- LUAS_LOT (Lot Size): Measures the land area of the property.
- LUAS_LOT_BGN (Built-Up Area): Indicates the property's built-up space.
- UMUR_BGN (Building Age): Reflects the age of the property.
- BILIK_TDR (Number of Bedrooms): Counts bedrooms, which can impact property value.
- KEADAAN_BGN (Building Condition): Describes the overall condition (e.g., new, good, average).
- Transaction Details:HARGA_B (Transaction Price): The actual price for which the property was sold, serving as the target variable.
- KATEGORI_KAW (Area Category): Evaluates the area quality, such as good or very good.
- KLASIFIKASI_KAW (Area Classification): Specifies whether the property is in an urban, suburban, or rural area.TKH_NILAI (Transaction Year): The year of the transaction, allowing

DATA CLEANING

Handling missing values

Filtration and Dropping Irrelevant Columns

Handle Duplicates

Data Binning

```python
import pandas as pd

# Filter rows based on the conditions and exclude rows where LUAS_L  BGN > LUAS_LOT for B_TINGKAT == 1
Filter_Melaka = df_Melaka[
    (df_Melaka['SECTOR'] == 'Residential') &
    (df_Melaka['CATEGORY'] == 'Terraced House') &
    (df_Melaka['YEAR1'].isin([2019, 2020, 2021, 2022, 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_TINGK == 1
    ~((df_Melaka['B_TINGKAT'] == 1) & (df_Melaka['LUAS_LOT_BGN'] > df_Melak['LUAS_LOT']))
]
```

```python
Filter_Melaka.shape
```

```
(21227, 64)
```

```python

# List of columns to drop
columns_to_drop = [
    'KOD_CAW', 'KAIT', 'STATE', 'BAHAGIAN', 'BAHAGIAN1', 'DAERAH',
    'JENIS_LOT', 'LOT_PLOT', 'JENIS_HAKMILIK', 'NO_HAKMILIK', 'SECTOR',
    'TKH_NILAI', 'Year', 'HALF YEAR', 'Quarter', 'Month', 'STAT_FEROR',
    'STAT_FEREE', 'HALF YEAR1', 'QUARTER1', 'Expr1022', 'ALAMAT', 'MONTH1',
    'MUKBAN', 'CATEGORY', 'PRO_TYPE', 'SYER1', 'SYER2', 'UNIT_LUAS_LOT',
    'RANGE', 'PRICE RANGE', 'TKH_HANTAR', 'Expr1041', 'P_LIBAT', 'P_LIBAT2',
    'Expr1046', 'Expr1047', 'TRANSEFEROR', 'TRANSFEREE', 'NILAI_LPR',
    'Expr1051', 'PEGAWAI_KENDALI', 'NAMA_BGN', 'PAJAKAN', 'KLASIFIKASI_BGN',
    'KOD_JENIS_TNH_PTN', 'STATUS_LOT', 'JENIS_TNH_PTN'
]

# Drop these columns from the DataFrame
df_Melaka = Model_Melaka.drop(columns=columns_to_drop)

# Display the first few rows of the filtered dataset to verify
print(df_Melaka.head())

# Save the filtered DataFrame to a CSV file
df_Melaka.to_csv('Melaka_filtered_dataset.csv', index=False)
```

DATA CLEANING

Handling missing values

Dropping Irrelevant Columns

Handle Duplicates

Data Binning

2.1) Check Missing Value

```
1  df_Melaka. isna().sum()
```

	0
DAERAH1	0
YEAR1	0
PM_PERTAMA	0
SKIM	2
MUKIM1	0
B_TINGKAT	0
LUAS_LOT	0
LUAS_LOT_BGN	0
HARGA_B	0
BILIK_TDR	0
JENIS_PEGANGAN	0
JENIS_BINAAN	0
KEADAAN_BGN	4
KATEGORI_KAW	161
KLASIFIKASI_KAW	0

Remove Missing Value

```
1  df_Melaka.dropna(inplace=True)
```

```
[ ]  1  # Count the number of duplicate rows based on the specific columns
     2  duplicate_count = Filter_Melaka.duplicated(subset=columns_to_check, keep=False).
        sum()
     3
     4  # Print the count of duplicates
     5  (f"Number of duplicate rows: {duplicate_count}")
     6
```

```
'Number of duplicate rows: 0'
```

```
1  # Categorize HARGA_B into 'Low tier', 'Mid tier', 'High tier'
2  df_Melaka['PRICE_CLASS'] = pd.qcut(df_Melaka['HARGA_B'], q=3, labels=['Low
   tier', 'Mid tier', 'High tier'])
3  print(df_Melaka[['HARGA_B', 'PRICE_CLASS']].head())
```

```
    HARGA_B PRICE_CLASS
0  230000.0    Mid tier
6  650000.0   High tier
14 260000.0    Mid tier
18 200000.0    Low tier
24 340000.0   High tier
```

DATA PREPARATION

Encoding

```
from sklearn.preprocessing import LabelEncoder
import pickle

# Identify columns that are categorical and should be label encoded
categorical_columns = ['PM_PERTAMA', 'DAERAH1', 'MUKIM1', 'JENIS_BINAAN',
'JENIS_PEGANGAN',
                       'KEADAAN_BGN', 'KATEGORI_KAW', 'KLASIFIKASI_KAW',
                       'SKIM', 'PRICE_CLASS']

# Dictionary to store each LabelEncoder for later use
label_encoders = {}

# Apply Label Encoding to each categorical column
for col in categorical_columns:
    label_encoder = LabelEncoder()  # Create a new LabelEncoder for each column
    df_Melaka[col] = label_encoder.fit_transform(df_Melaka[col].astype(str))
    label_encoders[col] = label_encoder  # Save the encoder for this column

# Display the first few rows to check the encoded data
print(df_Melaka.head())

# Save each LabelEncoder for future use
for col, encoder in label_encoders.items():
    with open(f'{col} label encoder.pkl', 'wb') as file:
```

Normalization

```
from sklearn.preprocessing import StandardScaler

# Assuming your DataFrame is named 'df'
X = df_Melaka.drop(['HARGA_B','PRICE_CLASS'], axis=1)
y = df_Melaka['PRICE_CLASS']

scaler = StandardScaler()
Xscaled = scaler.fit_transform(X)
```

```
import pickle
from sklearn.preprocessing import StandardScaler

# Assuming you have a scaler object
scaler = StandardScaler()
# Fit and transform the scaler on your data (if needed)
# scaler.fit(your_data)

# Save the scaler to a file
with open('scaler.pkl', 'wb') as file:
    pickle.dump(scaler, file)

```

Split Features & Target Variables

Train-Test Data Splitting

DATA PREPARATION

Encoding

Normalization

Split Features & Target Variables

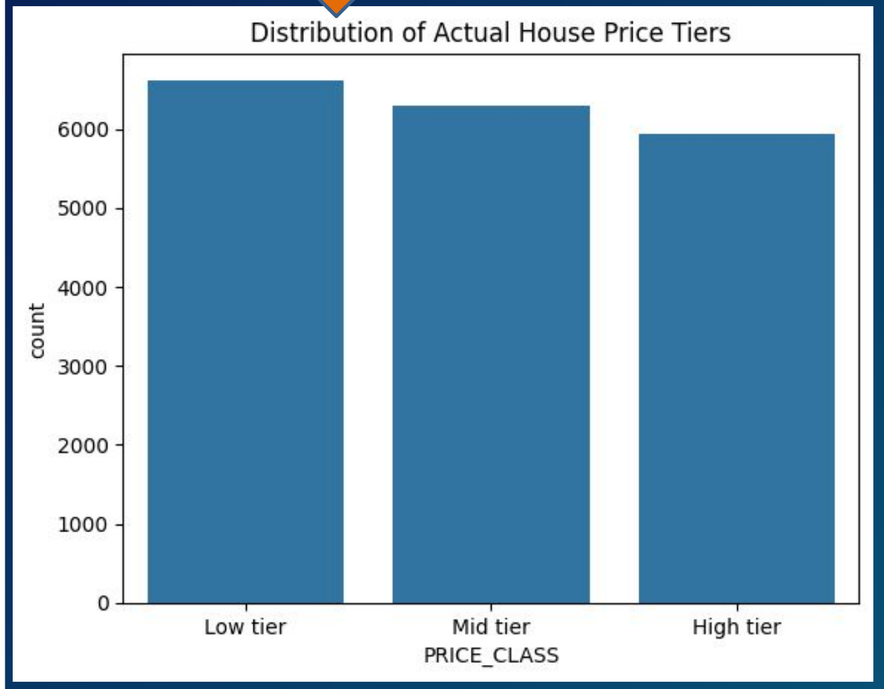
Train-Test Data Splitting

```
[ ]  1  from sklearn.preprocessing import StandardScaler
     2
     3  # Assuming your DataFrame is named 'df'
     4  X = df_Melaka.drop(['HARGA_B','PRICE_CLASS'], axis=1)
     5  y = df_Melaka['PRICE_CLASS']
     6
     7  scaler = StandardScaler()
     8  Xscaled = scaler.fit_transform(X)
```

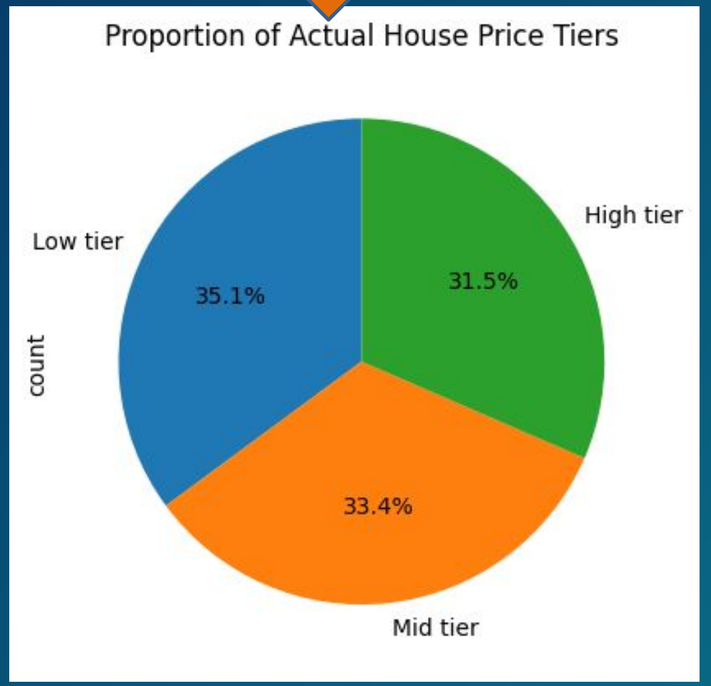
```
[ ]  1  from sklearn.model_selection import train_test_split
     2  X_train, X_test, y_train, y_test = train_test_split(Xscaled, y, test_size=0.2,
        random_state=42)
```

EXPLORATORY DATA ANALYSIS

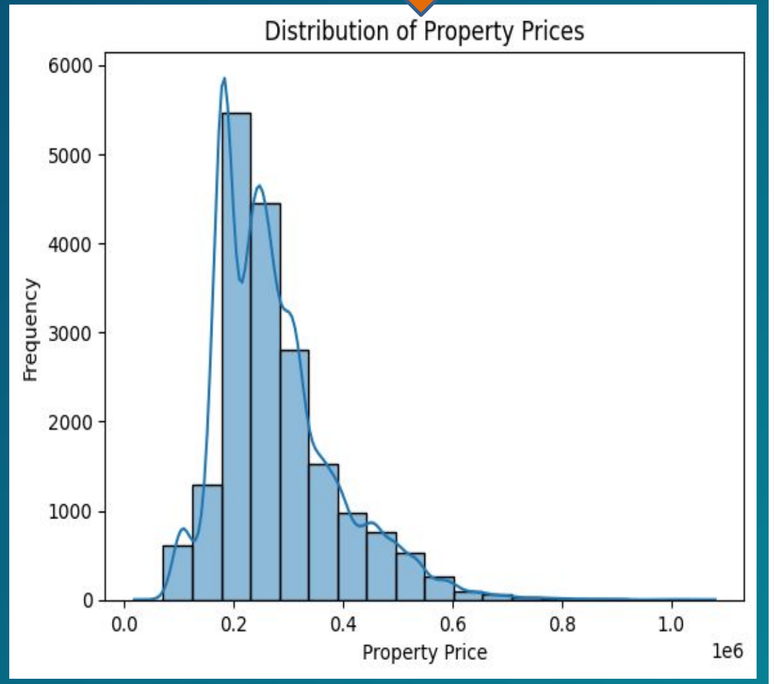
Bar Plot



Pie Chart

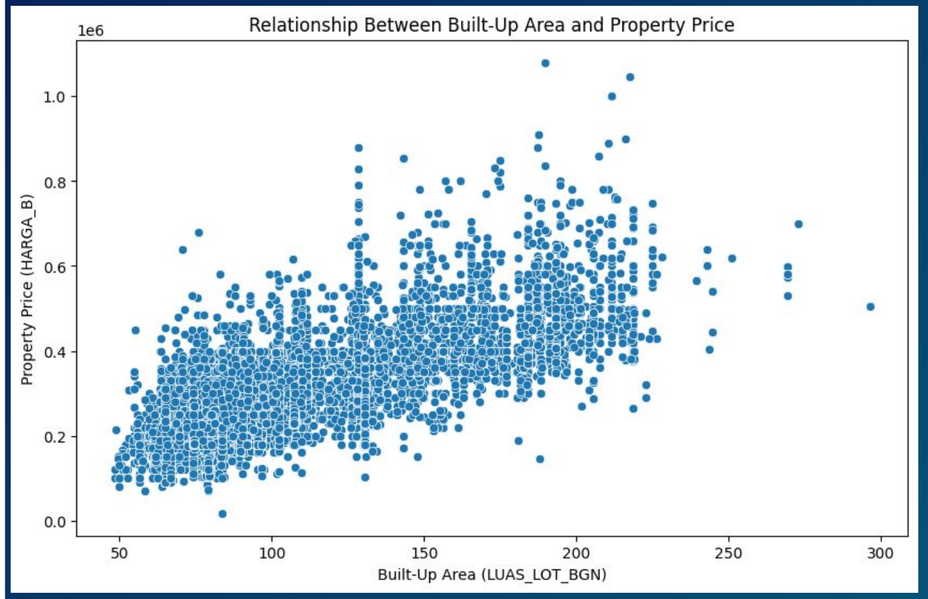


Histogram



EXPLORATORY DATA ANALYSIS

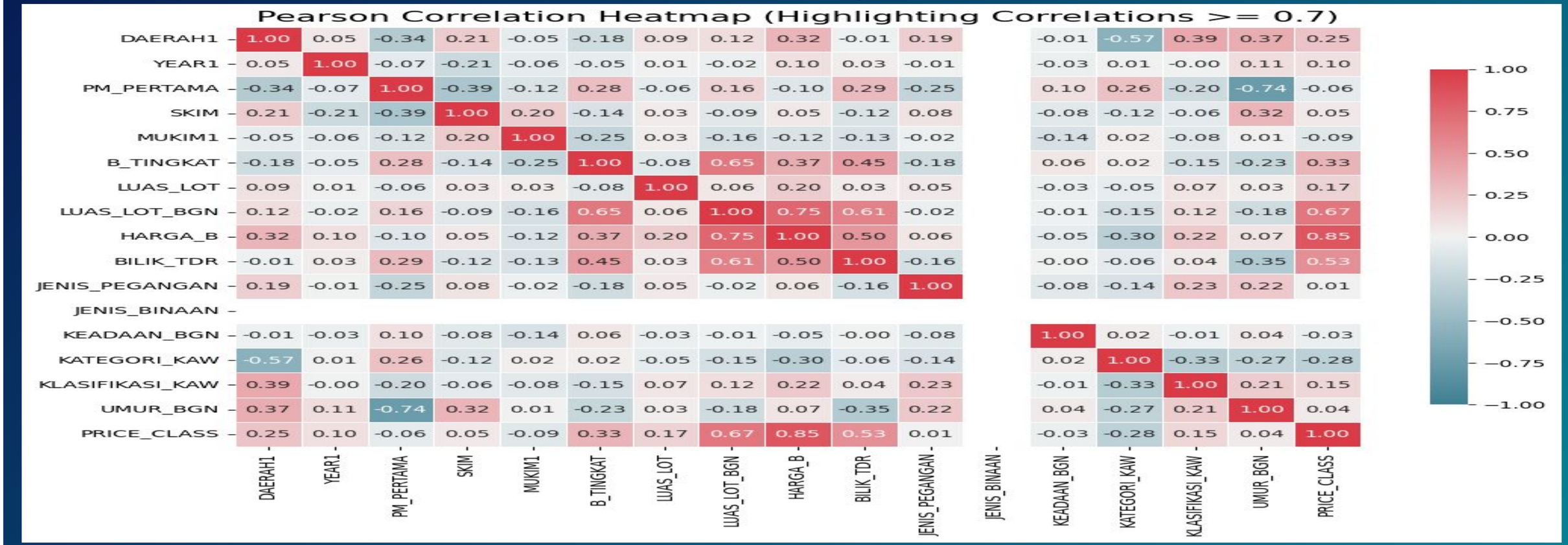
Scatter Plot



Stacked Bar

EXPLORATORY DATA ANALYSIS

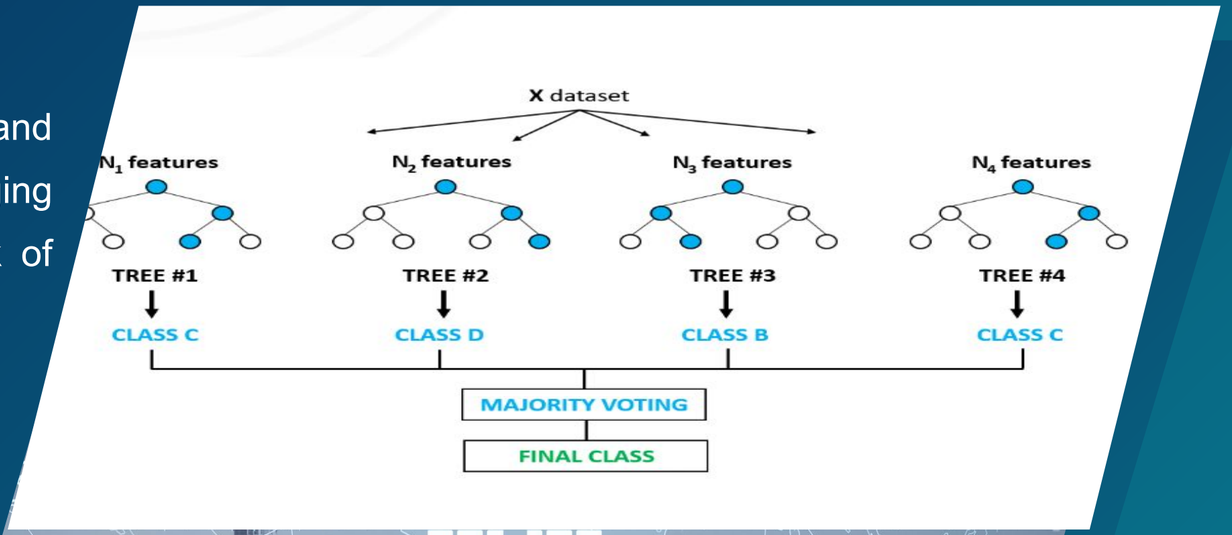
Pearson Correlation Heatmap



MACHINE LEARNING

Random Forest Model

The Random Forest model showed strong results, with high accuracy and balanced precision and recall across all property price categories. By averaging the results of multiple decision trees, Random Forest minimized the risk of overfitting and generalized well on the unseen test data.



Random Forest Process

Random Forest Classifier (sklearn.ensemble module)

Confusion Matrix

Training using (fit) method

Classification Report

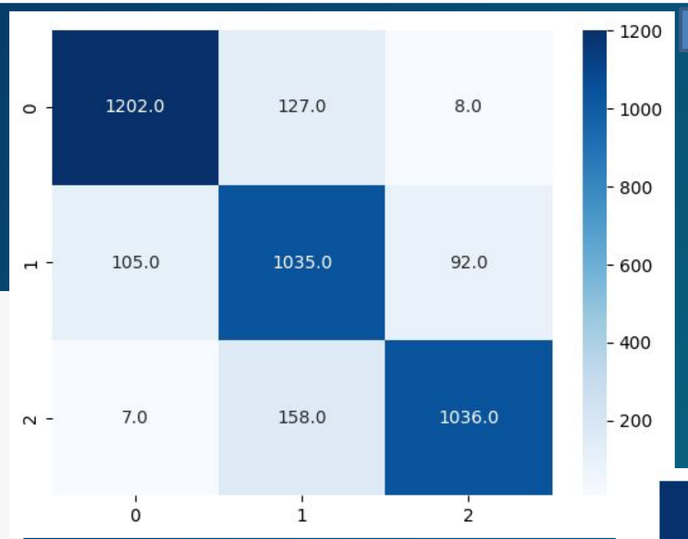
Random Forest Model Evaluation

Confusion Matrix

```python
from sklearn import metrics

# First prediction (use this throughout)
rf_prediction = rf_model.predict(X_test)

# Calculate confusion matrix and accuracy using rf_prediction
rf_confusion_matrix = metrics.confusion_matrix(y_test, rf_prediction)
rf_accuracy = metrics.accuracy_score(y_test, rf_prediction)
print(f'Confusion Matrix:\n{rf_confusion_matrix}')
print(f'Accuracy: {rf_accuracy * 100:.2f}%')
```

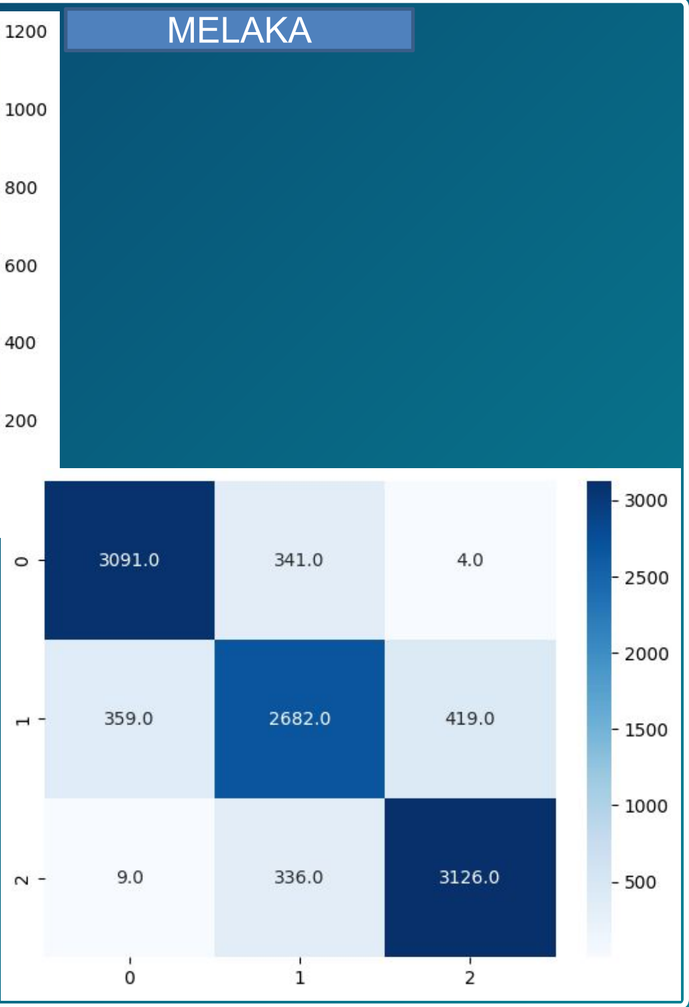
MELAKA

```
Confusion Matrix:
[[1202  127    8]
 [ 105 1035   92]
 [   7  158 1036]]
Accuracy: 86.82%
```



JOHOR

```
Confusion Matrix:
[[3091  341    4]
 [ 359 2682  419]
 [   9  336 3126]]
Accuracy: 85.84%
```



Random Forest Model Evaluation

Classification Report

```python
from sklearn.metrics import accuracy_score, classification_report

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Classification report for detailed performance metrics
print(classification_report(y_test, y_pred))

```

```
Accuracy: 0.8681697
              precision    recall  f1-score   support

           0       0.91      0.90      0.91      1337
           1       0.78      0.84      0.81      1232
           2       0.91      0.86      0.89      1201

    accuracy                           0.87      3770
   macro avg       0.87      0.87      0.87      3770
weighted avg       0.87      0.87      0.87      3770
```

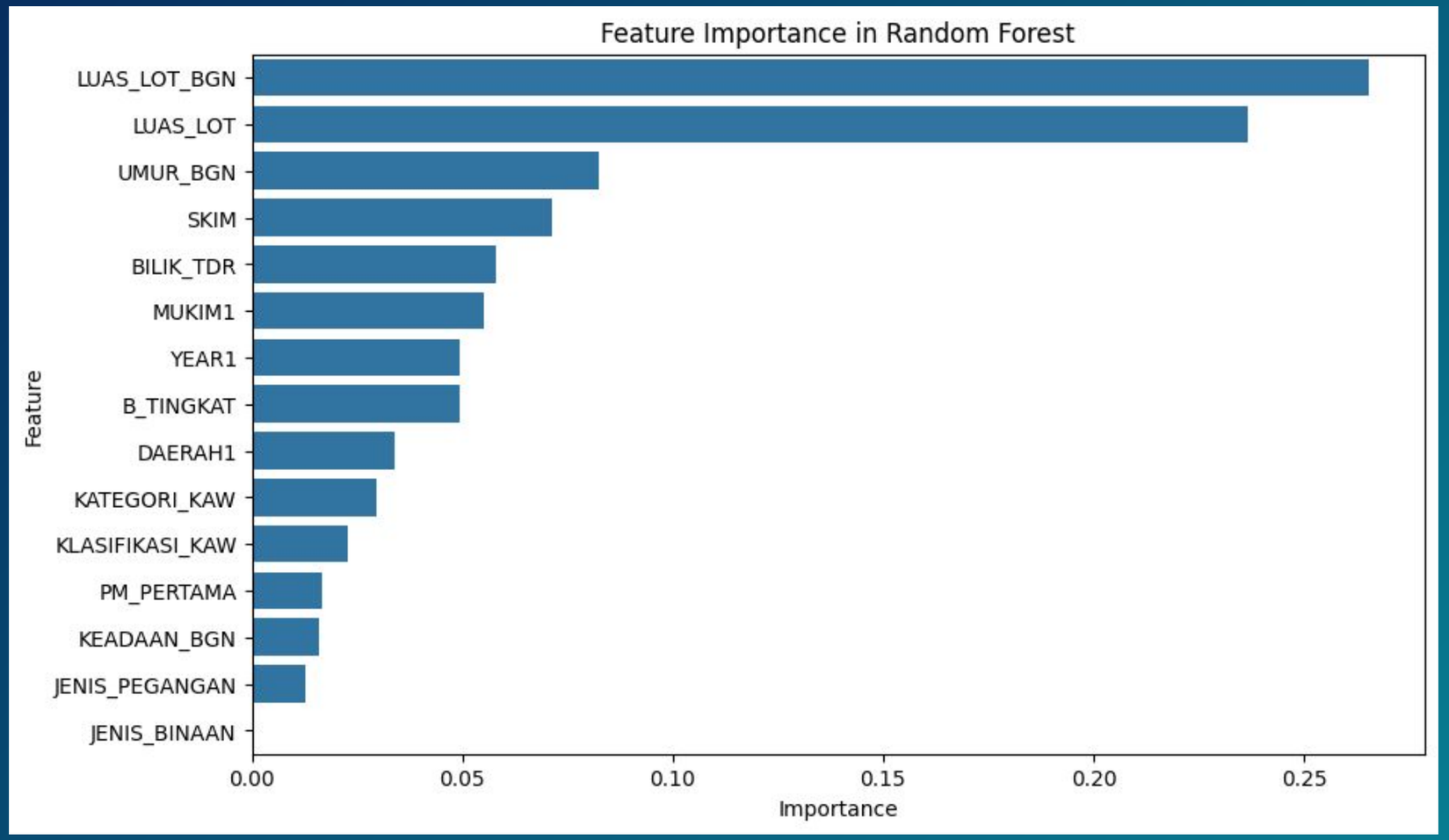
MELAKA

```
              precision    recall  f1-score   support

           0       0.89      0.90      0.90      3436
           1       0.80      0.78      0.79      3460
           2       0.88      0.90      0.89      3471

    accuracy                           0.86     10367
   macro avg       0.86      0.86      0.86     10367
weighted avg       0.86      0.86      0.86     10367
```

JOHOR

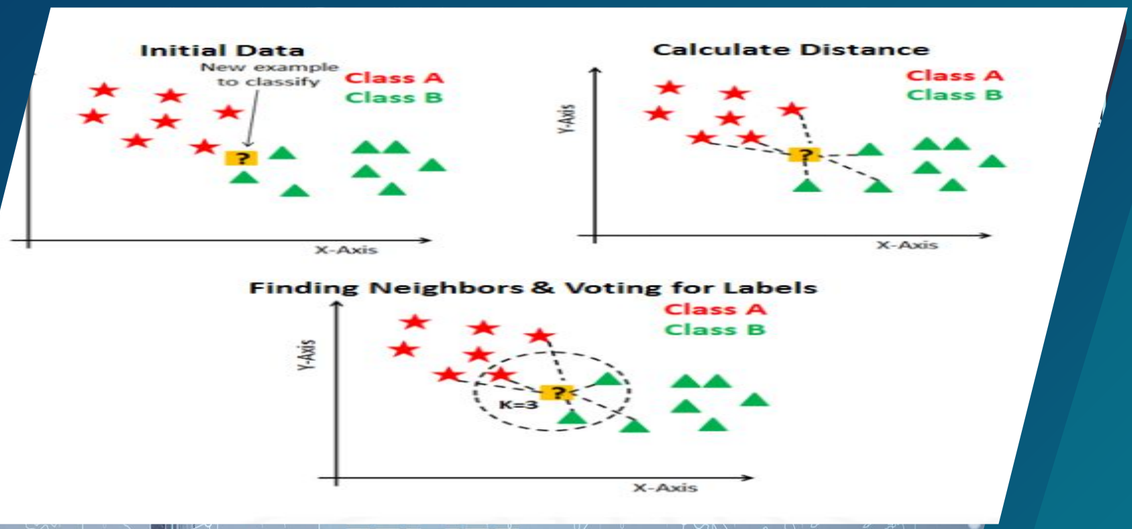
Random Forest Model Evaluation

Feature Performance



K-Nearest Neighbors (KNN) Model

KNN is a simple yet effective machine learning algorithm that works by finding the nearest neighbors of a data point and making predictions based on the majority class among those neighbors.



Random Forest Process

KNN Classifier

(sklearn.ensemble module)

Confusion Matrix

Training using (fit) method

Classification Report

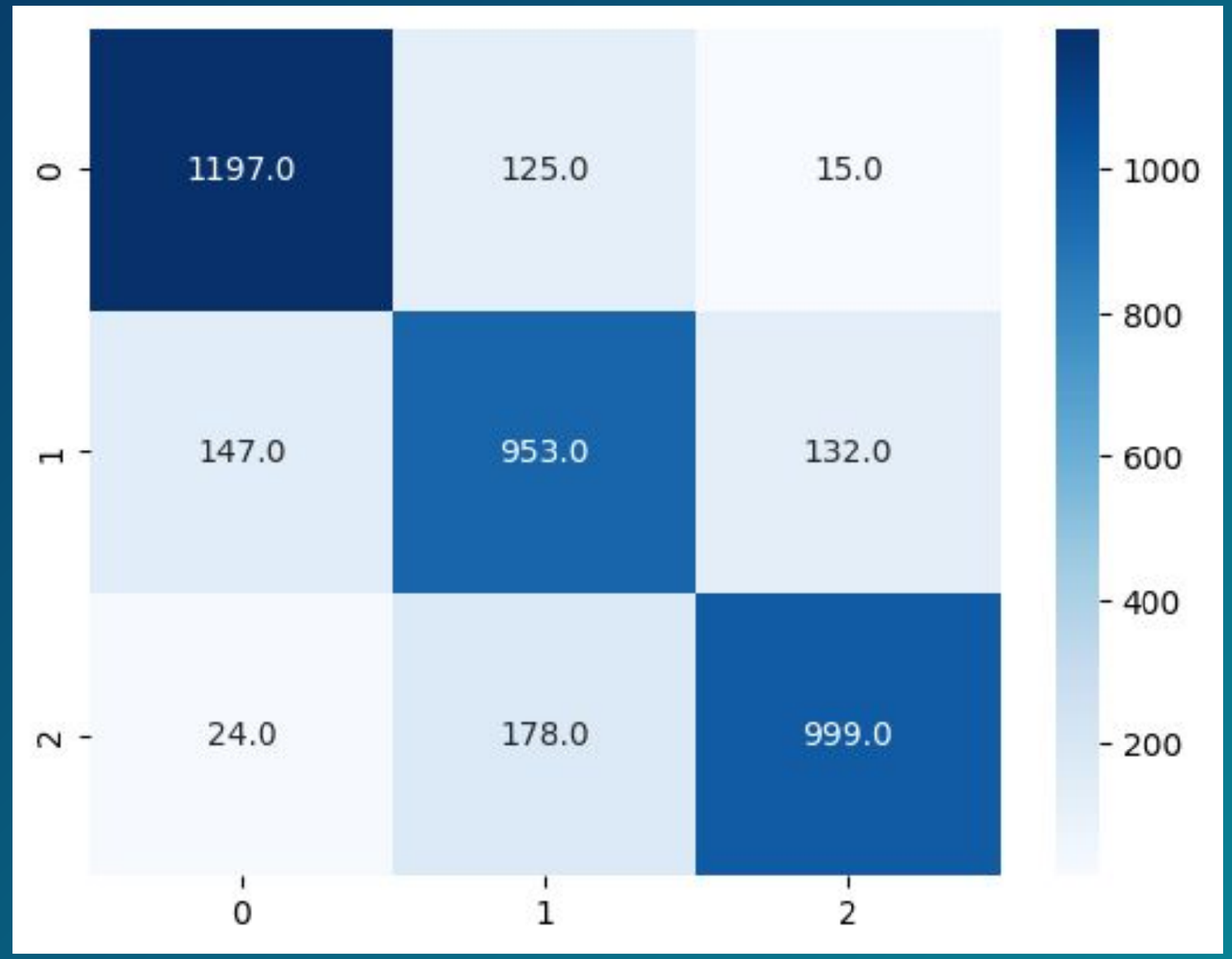
K-Nearest Neighbors(KNN) Model Evaluation

Confusion Matrix

```
[43]  1  knn_prediction = knn_model.predict(X_test)
```

```
1  # Evaluate the predictions using `metrics` module from sklearn
2  from sklearn import metrics
3
4  # use `confusion_matrix(prediction, target_test)` to generate the confusion
   matrix based on the prediction and real test labels
5  knn_confusion_matrix = metrics.confusion_matrix(y_test, knn_prediction)
6  print(f'Confusion Matrix:\n{knn_confusion_matrix}')
7
8  # `accuracy_score(prediction, target_test)` to get the accuracy of the
   predictions
9  knn_accuracy = metrics.accuracy_score(y_test, knn_prediction)
10 print(f'Accuracy: {knn_accuracy * 100:.2f}%')
```

```
Confusion Matrix:
[[1197  125   15]
 [ 147  953  132]
 [  24  178  999]]
Accuracy: 83.53%
```



K-Nearest Neighbors (KNN) Model Evaluation

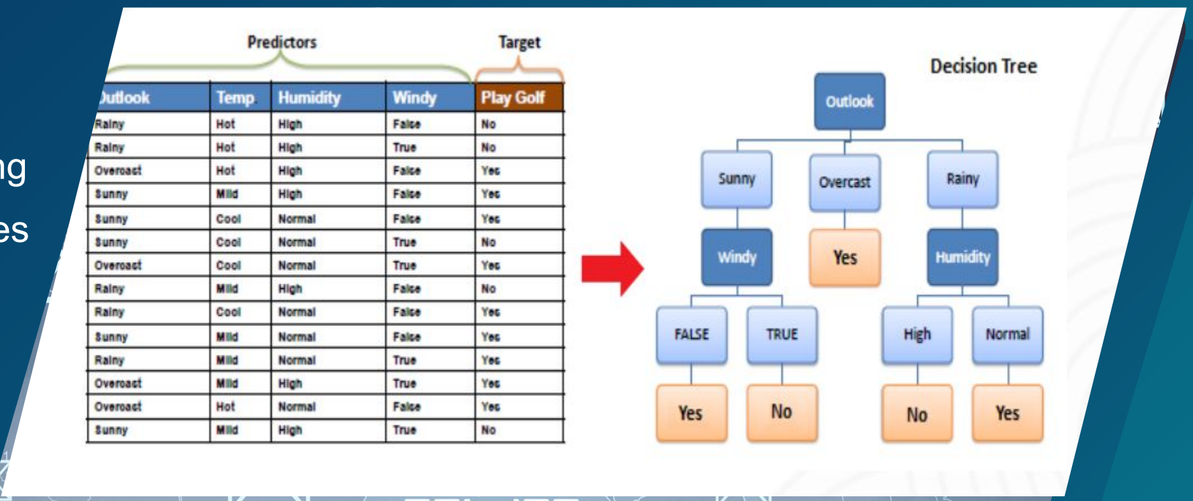
Classification Report

```python
from sklearn.metrics import accuracy_score, classification_report

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Classification report for detailed performance metrics
print(classification_report(y_test, y_pred))

```

```
Accuracy: 0.8681697612732096
              precision    recall  f1-score   support

           0       0.91      0.90      0.91      1337
           1       0.78      0.84      0.81      1232
           2       0.91      0.86      0.89      1201

    accuracy                           0.87      3770
   macro avg       0.87      0.87      0.87      3770
weighted avg       0.87      0.87      0.87      3770
```

Decision Tree Model

Decision Trees are a powerful and interpretable model, capable of handling both categorical and numerical data by learning a set of rules from the features to make predictions.



Random Forest Process

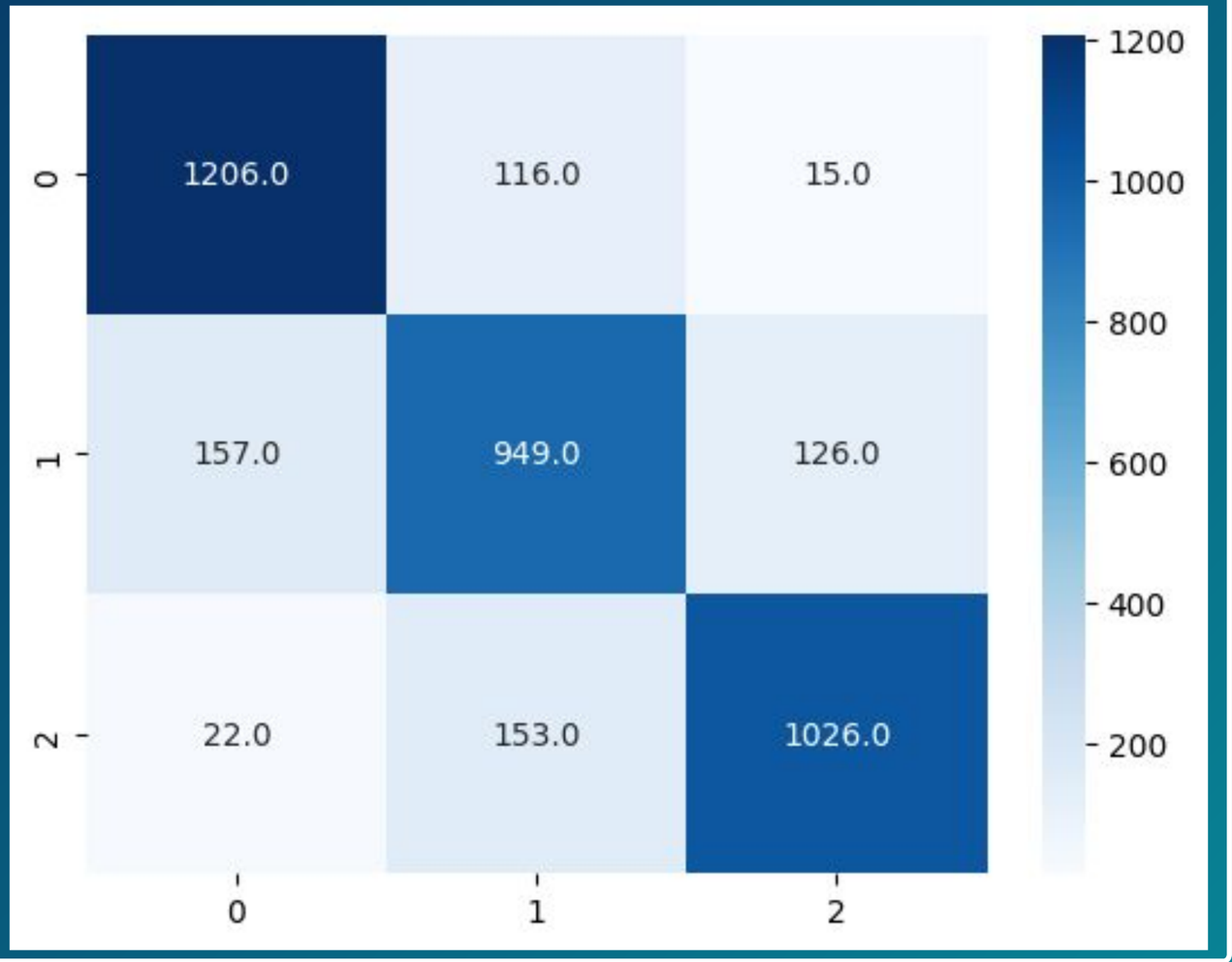
- Decision Tree Classifier (sklearn.tree module)
- Confusion Matrix
- Training using (fit) method
- Classification Report

Decision Tree Model Evaluation

Confusion Matrix

```python
from sklearn import metrics

dt_confusion_matrix = metrics.confusion_matrix(y_test, dt_prediction)
dt_accuracy = metrics.accuracy_score(y_test, dt_prediction)
print(f'Confusion Matrix:\n{dt_confusion_matrix}')
print(f'Accuracy: {dt_accuracy * 100:.2f}%')
```

```
Confusion Matrix:
[[1206  116   15]
 [ 157  949  126]
 [  22  153 1026]]
Accuracy: 84.38%
```



Decision Tree Model Evaluation

Classification Report

```
[ ]  1 dt_classification_report = metrics.classification_report(y_test, dt_prediction)
     2 print(f'Classification Report:\n{dt_classification_report}')
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.90      0.89      1337
           1       0.78      0.77      0.77      1232
           2       0.88      0.85      0.87      1201

    accuracy                           0.84      3770
   macro avg       0.84      0.84      0.84      3770
weighted avg       0.84      0.84      0.84      3770
```

COMPARISONS OF ACCURACY BETWEEN MELAKA AND JOHOR

Random Forest

KNN

Decision Tree

```
[ ]
1  import pandas as pd
2
3  results = [rf_accuracy,knn_accuracy, dt_accuracy, ]
4  methods = ['Random Forest', 'K-Nearest Neighbors', 'Decision Tree', ]
5
6  compare = {
7      'Methods': methods,
8      'Accuracy': results
9  }
10
11 comparison = pd.DataFrame(compare)
12 comparison.head()
```

	Methods	Accuracy
0	Random Forest	0.868170
1	K-Nearest Neighbors	0.835279
2	Decision Tree	0.843767

MELAKA

```
[ ]
1  import pandas as pd
2
3  results = [rf_accuracy,knn_accuracy, dt_accuracy, ]
4  methods = ['Random Forest', 'K-Nearest Neighbors', 'Decision Tree', ]
5
6  compare = {
7      'Methods': methods,
8      'Accuracy': results
9  }
10
11 comparison = pd.DataFrame(compare)
12 comparison.head()
```

	Methods	Accuracy
0	Random Forest	0.857915
1	K-Nearest Neighbors	0.806501
2	Decision Tree	0.830906

JOHOR

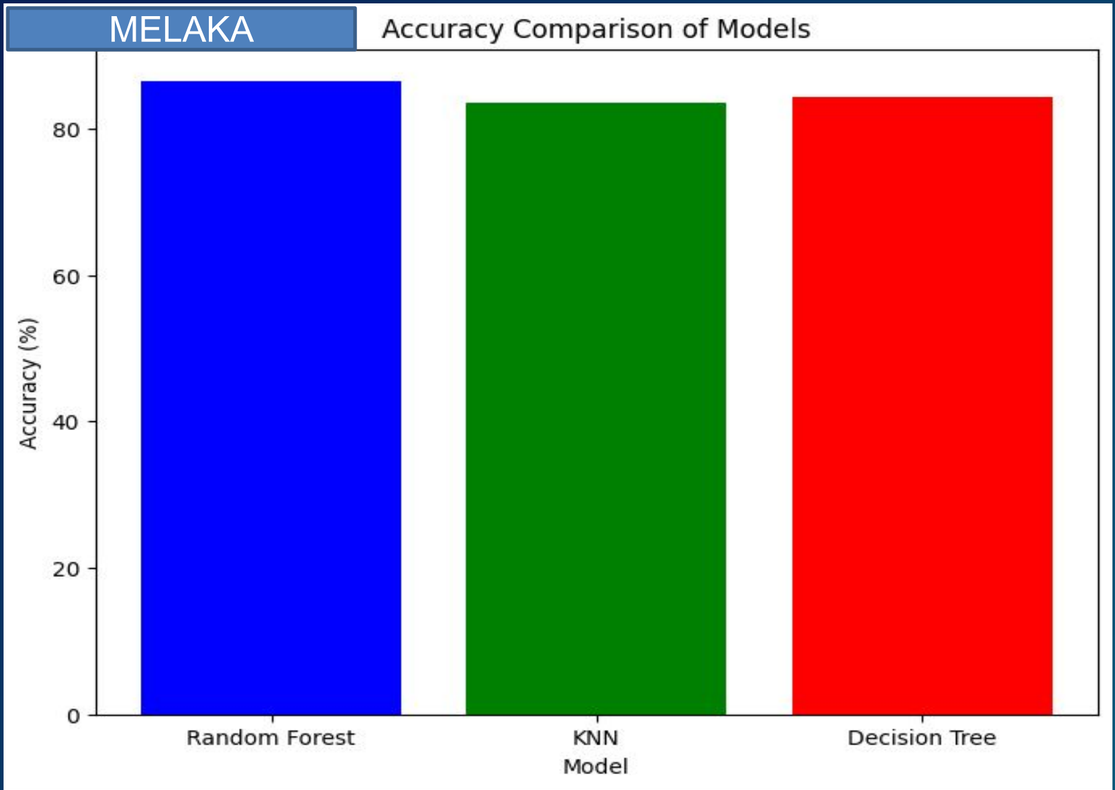
COMPARISONS OF ACCURACY BETWEEN MELAKA AND JOHOR

Random Forest

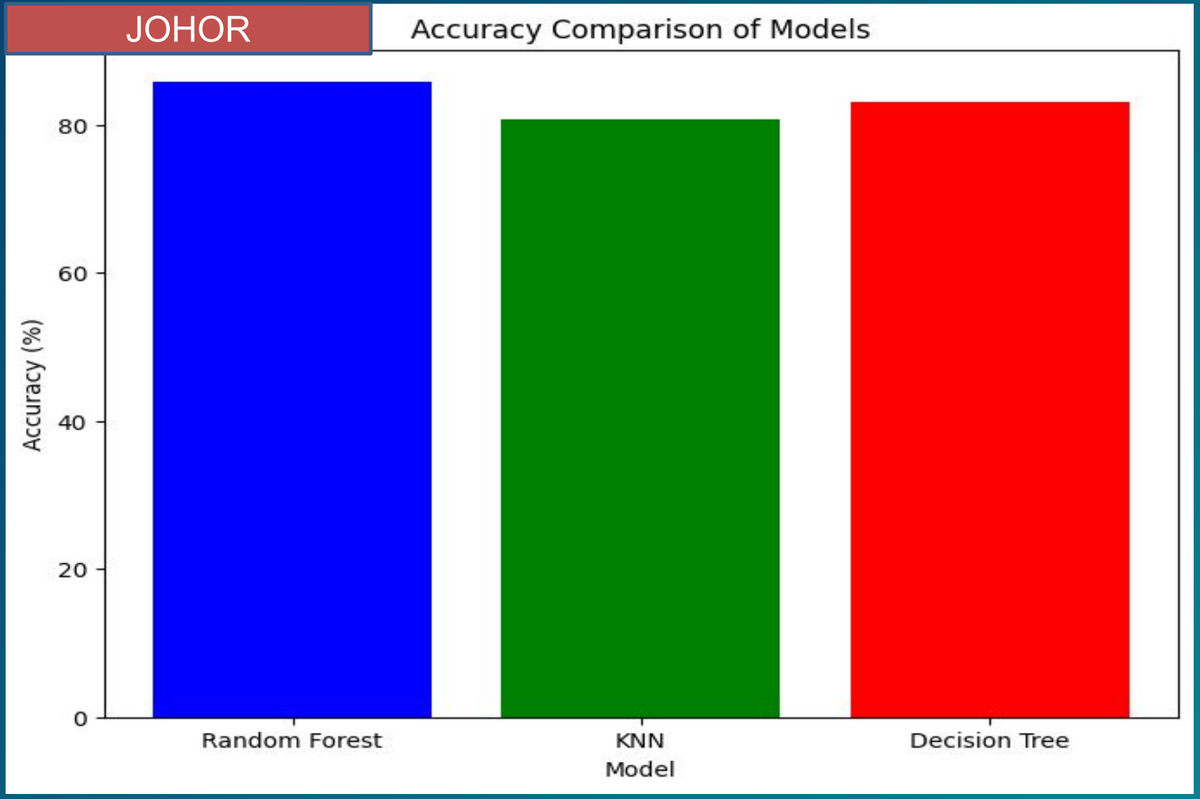
KNN

Decision Tree

MELAKA



JOHOR



DASHBOARD USING TABLEAU

DASHBOARD 1

OVERVIEW OF PROPERTY HOUSE TRANSACTIONS IN JOHOR AND MELAKA

Filters

State
All

District
All

Property Type
All

Year
All

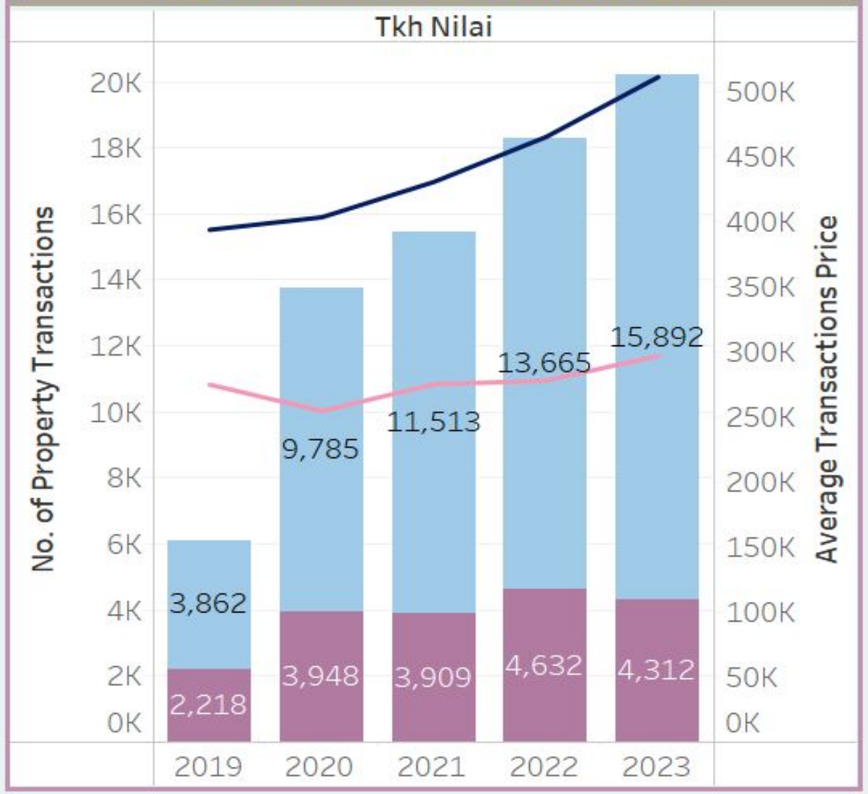
NUMBER OF TRANSACTIONS

Johor	54,717
Melaka	19,019

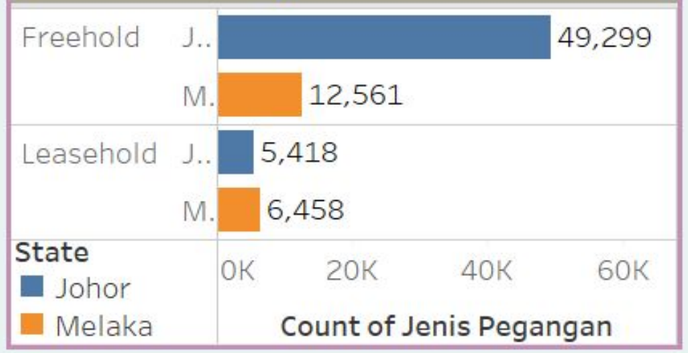
YEAR

2019 2020 2021 2022 2023

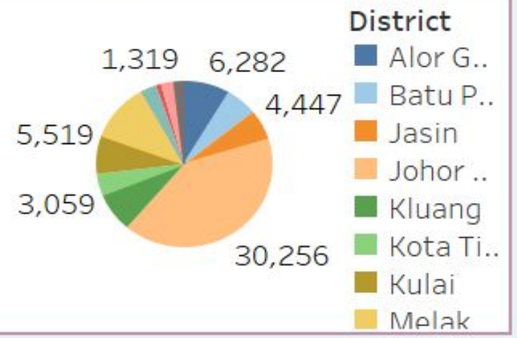
Yearly Volume of Property Transactions vs. Average Transactions Value Trends by State All



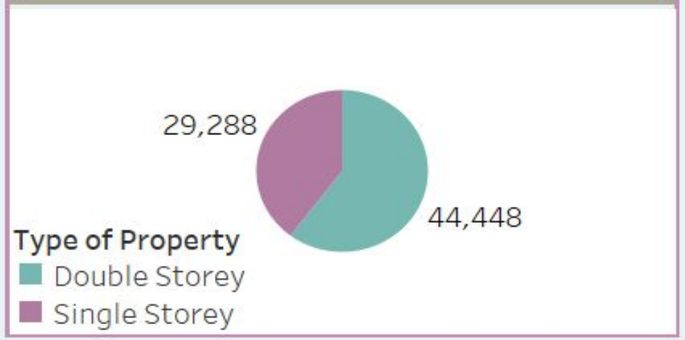
Distribution of Property Tenureship by State All, District All, Type None and Year None



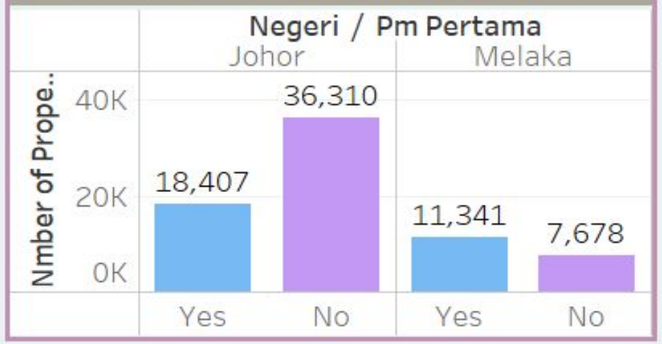
Terraced House Transactions Volume by State All, District All, Type None and Year None



Volume Type of Property Transactions by State All



Volume of First Transfer and Second Transfer by State All, District All, Type None and Year None



DASHBOARD 2

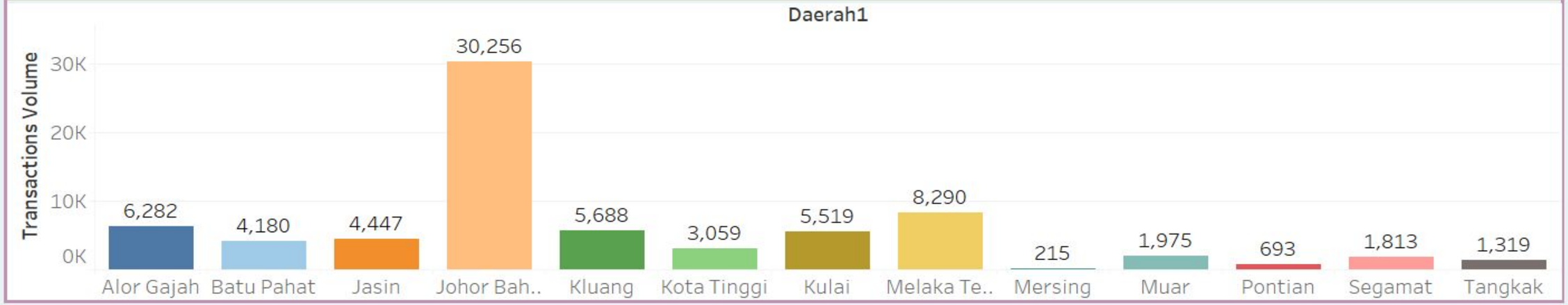
OVERVIEW OF PROPERTY HOUSE TRANSACTIONS IN JOHOR AND MELAKA

Filters

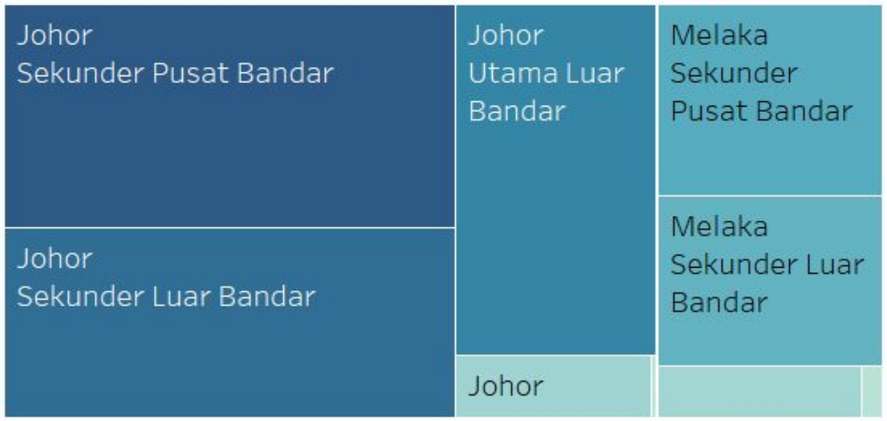
Negeri
All

District
All

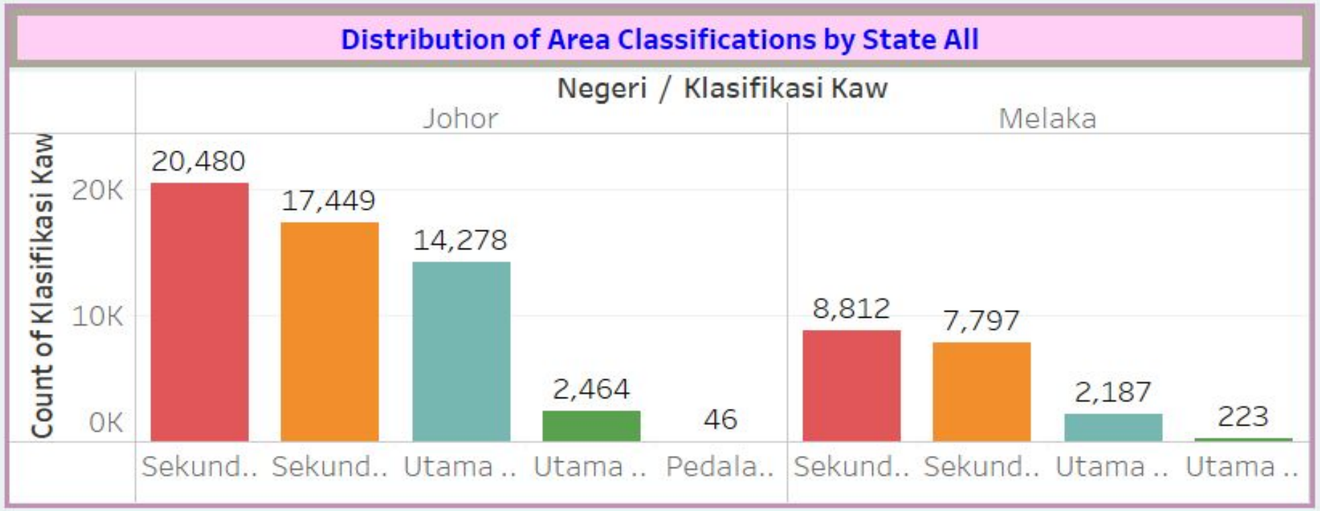
Terraced House Transactions Volume by District All,



Propotion of Area Classifications



Distribution of Area Classifications by State All



DASHBOARD 3

ACTUAL VS PREDICTED

Filters

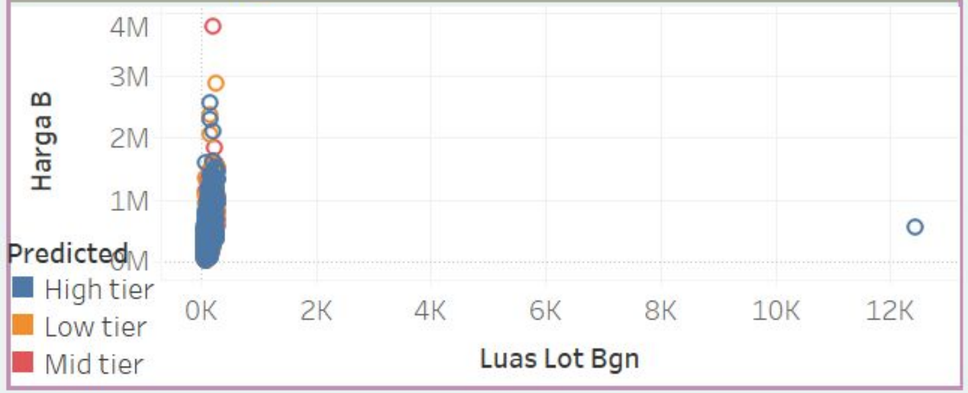
State
Melaka

Confusion Matrix Actual Price Class vs. Predicted Price Class

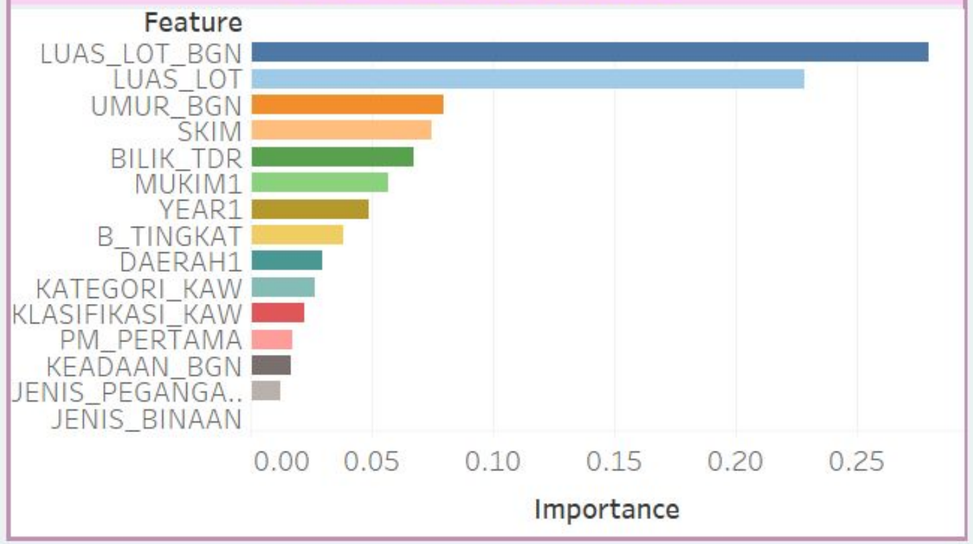
Predicted Price Class

Actual ..	Low tier	Mid tier	High tier
Low tier	1,389	154	7
Mid tier	121	1,206	103
High tier	9	176	1,182

Built Up Area vs. Predicted Price Class

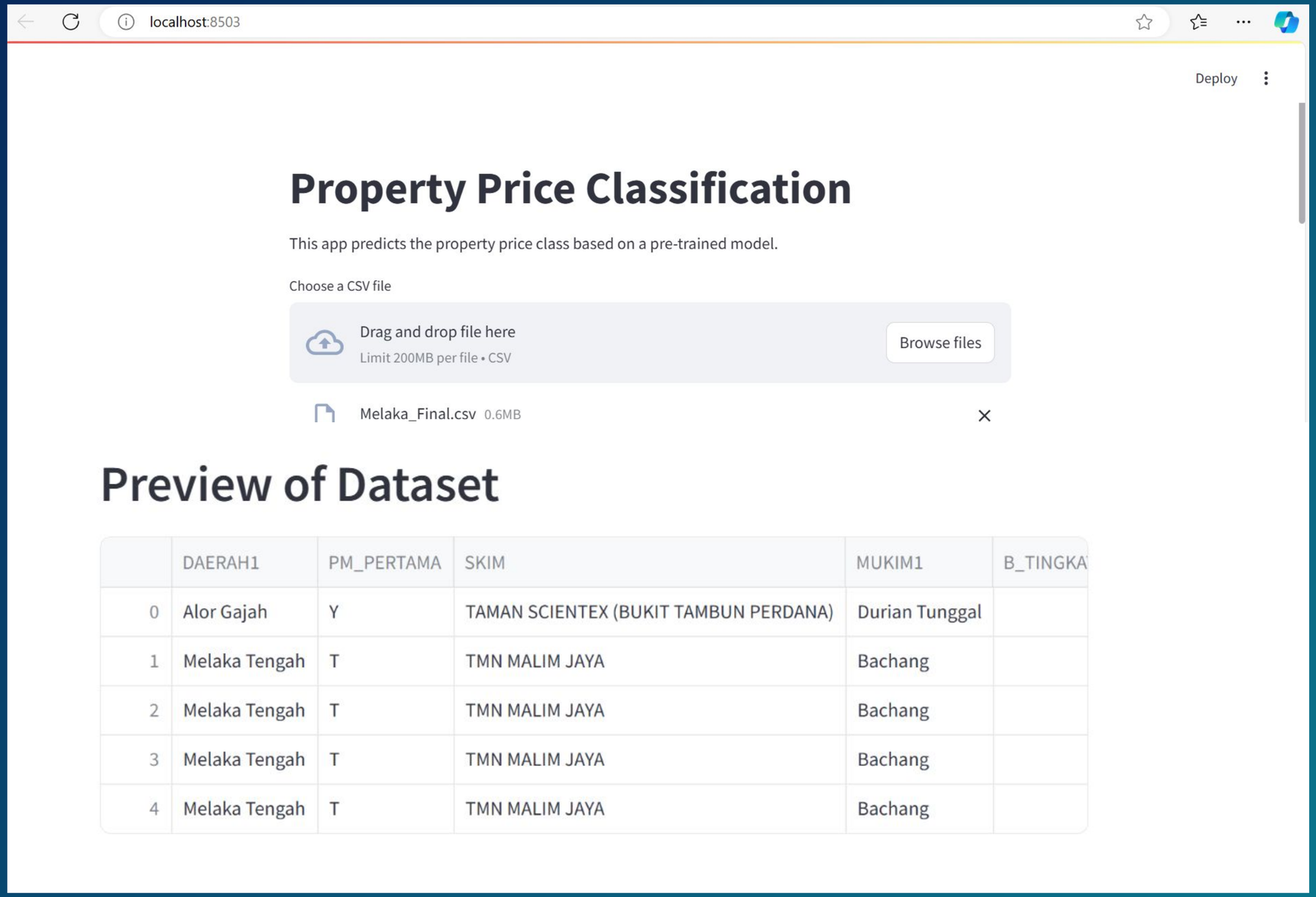


Feature Importance



DASHBOARD USING STREAMLIT

STREAMLIT DASHBOARD



	DAERAH1	PM_PERTAMA	SKIM	MUKIM1	B_TINGKA
0	Alor Gajah	Y	TAMAN SCIENTEX (BUKIT TAMBUN PERDANA)	Durian Tunggal	
1	Melaka Tengah	T	TMN MALIM JAYA	Bachang	
2	Melaka Tengah	T	TMN MALIM JAYA	Bachang	
3	Melaka Tengah	T	TMN MALIM JAYA	Bachang	
4	Melaka Tengah	T	TMN MALIM JAYA	Bachang	

STREAMLIT DASHBOARD

Predictions

	LUAS_LOT_BGN	LUAS_LOT	UMUR_BGN	BILIK_TDR	SKIM	MUKIM1	YEAR1	B_T	Predicted Price Class
17	82.68	65	28	3	TMN MALIM PERMA	Bachang	2,019		1
18	82.68	65	28	3	TMN MALIM PERMA	Bachang	2,020		1
19	70.72	65	35	2	TMN MALIM JAYA	Bachang	2,022		1
20	66.82	65	38	2	TMN MALIM JAYA	Bachang	2,024		2
21	66.82	65	38	2	TMN MALIM JAYA	Bachang	2,024		2
22	66.82	65	37	2	TMN MALIM JAYA	Bachang	2,023		2
23	66.82	65	36	2	TMN MALIM JAYA	Bachang	2,022		1
24	66.82	65	35	2	TMN MALIM JAYA	Bachang	2,021		1
25	66.82	65	34	2	TMN MALIM JAYA	Bachang	2,020		0
26	65.03	65	36	2	TMN MALIM JAYA	Bachang	2,020		
27	65.03	65	36	3	TMN MALIM JAYA	Bachang	2,020		

Thank You For Your Attention

End Of Presentation