

01 - DATA Processing

```
In [2]: import pandas as pd
import numpy as np
import os
import json
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
```

```
In [3]: # Directories
os.makedirs('data/processed', exist_ok=True)
os.makedirs('data/final', exist_ok=True)
os.makedirs('results', exist_ok=True)
```

```
In [6]: # Load Data
districts = pd.read_csv('data/raw/districts.csv')
plots = pd.read_csv('data/raw/plots.csv')

# Show top 5 rows of each
print("Top 5 rows of districts:")
display(districts.head())

print("Top 5 rows of plots:")
display(plots.head())
```

Top 5 rows of districts:

	district_id	year	month	Type	AverageMeterPrice
0	21001145	2024	12	أرض سكني	5538
1	21001145	2024	12	فلل	10033
2	21001145	2024	12	مبنى تجاري سكني	21384
3	21001145	2025	1	أرض سكني	4951
4	21001145	2025	1	فلل	9841

Top 5 rows of plots:

	district_id	plot_id	Type	lat	lon	MeterPrice	month	year
0	21001294	21001353230	شقق	21.757199	39.173688	2691	12	2024
1	21001294	21001353230	شقق	21.757199	39.173688	2691	12	2024
2	21001293	210012112	شقق	21.541943	39.196728	2596	9	2024
3	21001293	21001475155	شقق	21.533378	39.190417	3237	9	2024
4	21001293	210011989	شقق	21.541744	39.197044	2480	9	2024

```
In [5]: # Clean Columns
districts.columns = districts.columns.str.strip()
plots.columns = plots.columns.str.strip()
```

```
In [7]: # Remove Duplicates
plots.drop_duplicates(inplace=True)
districts.drop_duplicates(inplace=True)
```

```
In [8]: # Map Property Types
type_map = {
    'شقق': 'apartment',
    'أرض سكني': 'residential_land',
    'فلل': 'villa',
}
```

```
'أرض تجاري سكني': 'mixed_commercial_residential_land',
'مبنى تجاري سكني': 'commercial_residential_building',
'أرض تجاري': 'commercial_land',
'مصانع ومستودعات': 'industrial_warehouse',
'زراعي': 'agricultural'
}
ptype_col = [c for c in plots.columns if 'type' in c.lower()][0]
plots['property_type_en'] = plots[ptype_col].map(type_map).fillna('other')
```

```
In [9]: # Remove Outliers from Price
price_col = [c for c in plots.columns if 'price' in c.lower() or 'meter' in c.lower()][0]
Q1, Q3 = plots[price_col].quantile([0.25, 0.75])
IQR = Q3 - Q1
low, high = Q1 - 2*IQR, Q3 + 2*IQR
plots = plots[(plots[price_col] >= low) & (plots[price_col] <= high)]
```

```
In [10]: # Feature Engineering
if 'lat' in plots.columns and 'lon' in plots.columns:
    lat, lon = 21.5433, 39.1728 # center point of Jeddah
    plots['distance_from_center'] = np.sqrt((plots['lat'] - lat)**2 + (plots['lon'] - lon)**2) * 111
    coords = plots[['lat', 'lon']]
    plots['location_cluster'] = KMeans(n_clusters=10, n_init=10).fit_predict(coords)

if 'month' in plots.columns and 'year' in plots.columns:
    plots['quarter'] = ((plots['month'] - 1) // 3) + 1
    plots['is_year_end'] = (plots['month'] == 12).astype(int)
```

```
In [11]: # Encode Property Types
le = LabelEncoder()
plots['property_type_encoded'] = le.fit_transform(plots['property_type_en'])
```

```
In [12]: display(plots.head())
```

	district_id	plot_id	Type	lat	lon	MeterPrice	month	year	property_type_en	distance_from_center	location_cluster	quarter	is_year_end	property_type_encoded
0	21001294	21001353230	شقق	21.757199	39.173688	2691	12	2024	apartment	23.743010	9	4	1	0
2	21001293	210012112	شقق	21.541943	39.196728	2596	9	2024	apartment	2.660247	1	3	0	0
3	21001293	21001475155	شقق	21.533378	39.190417	3237	9	2024	apartment	2.244313	1	3	0	0
4	21001293	210011989	شقق	21.541744	39.197044	2480	9	2024	apartment	2.696660	1	3	0	0
5	21001293	210012108	شقق	21.541999	39.196990	3395	9	2024	apartment	2.688916	1	3	0	0

```
In [13]: # Merge with District Data
if 'district_id' in plots.columns and 'district_id' in districts.columns:
    districts_agg = districts.groupby('district_id').agg({'AverageMeterPrice': 'mean'}).reset_index()
    districts_agg.rename(columns={'AverageMeterPrice': 'price_per_meter_mean'}, inplace=True)
    plots = plots.merge(districts_agg, on='district_id', how='left')
```

```
In [14]: display(plots.head())
```

	district_id	plot_id	Type	lat	lon	MeterPrice	month	year	property_type_en	distance_from_center	location_cluster	quarter	is_year_end	property_type_encoded	price_per_meter_mean
0	21001294	21001353230	شقق	21.757199	39.173688	2691	12	2024	apartment	23.743010	9	4	1	0	1563.000000
1	21001293	210012112	شقق	21.541943	39.196728	2596	9	2024	apartment	2.660247	1	3	0	0	4875.392857
2	21001293	21001475155	شقق	21.533378	39.190417	3237	9	2024	apartment	2.244313	1	3	0	0	4875.392857
3	21001293	210011989	شقق	21.541744	39.197044	2480	9	2024	apartment	2.696660	1	3	0	0	4875.392857
4	21001293	210012108	شقق	21.541999	39.196990	3395	9	2024	apartment	2.688916	1	3	0	0	4875.392857

```
In [15]: # Final Data
features = ['lat', 'lon', 'distance_from_center', 'location_cluster',
            'month', 'year', 'quarter', 'is_year_end',
            'property_type_encoded', 'price_per_meter_mean', 'district_id']
features = [f for f in features if f in plots.columns]
final_data = plots[features + [price_col]].copy()
```

```
In [16]: display(final_data.head())
```

	lat	lon	distance_from_center	location_cluster	month	year	quarter	is_year_end	property_type_encoded	price_per_meter_mean	district_id	MeterPrice
0	21.757199	39.173688	23.743010	9	12	2024	4	1	0	1563.000000	21001294	2691
1	21.541943	39.196728	2.660247	1	9	2024	3	0	0	4875.392857	21001293	2596
2	21.533378	39.190417	2.244313	1	9	2024	3	0	0	4875.392857	21001293	3237
3	21.541744	39.197044	2.696660	1	9	2024	3	0	0	4875.392857	21001293	2480
4	21.541999	39.196990	2.688916	1	9	2024	3	0	0	4875.392857	21001293	3395

```
In [17]: # Save Files
plots.to_csv('data/processed/cleaned_data.csv', index=False)
final_data.to_csv('data/final/modeling_data.csv', index=False)
```

```
In [18]: # Save Summary
summary = {
    'rows': len(final_data),
    'features': features,
    'target': price_col
}
with open('results/summary.json', 'w') as f:
    json.dump(summary, f, indent=2)
```

```
In [23]: print(summary)

{'rows': 5550, 'features': ['lat', 'lon', 'distance_from_center', 'location_cluster', 'month', 'year', 'quarter', 'is_year_end', 'property_type_encoded', 'price_per_meter_mean', 'district_id'], 'target': 'MeterPrice'}
```

```
In [ ]:
```

```
In [10]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.pipeline import Pipeline
import joblib
import os
```

```
In [14]: # Load the type-aware processed data
data = pd.read_csv('data/final/modeling_data.csv')
print(f"✅ Loaded data: {len(data):,} rows, {len(data.columns)} columns")

display(data.head())
```

✅ Loaded data: 5,550 rows, 12 columns

	lat	lon	distance_from_center	location_cluster	month	year	quarter	is_year_end	property_type_encoded	price_per_meter_mean	district_id	MeterPrice
0	21.757199	39.173688	23.743010	9	12	2024	4	1	0	1563.000000	21001294	2691
1	21.541943	39.196728	2.660247	1	9	2024	3	0	0	4875.392857	21001293	2596
2	21.533378	39.190417	2.244313	1	9	2024	3	0	0	4875.392857	21001293	3237
3	21.541744	39.197044	2.696660	1	9	2024	3	0	0	4875.392857	21001293	2480
4	21.541999	39.196990	2.688916	1	9	2024	3	0	0	4875.392857	21001293	3395

```
In [15]: # Load the full processed data for type information
full_data = pd.read_csv('data/processed/cleaned_data.csv')
print(f"✅ Loaded full data: {len(full_data):,} rows")
display(full_data.head())
```

✅ Loaded full data: 5,550 rows

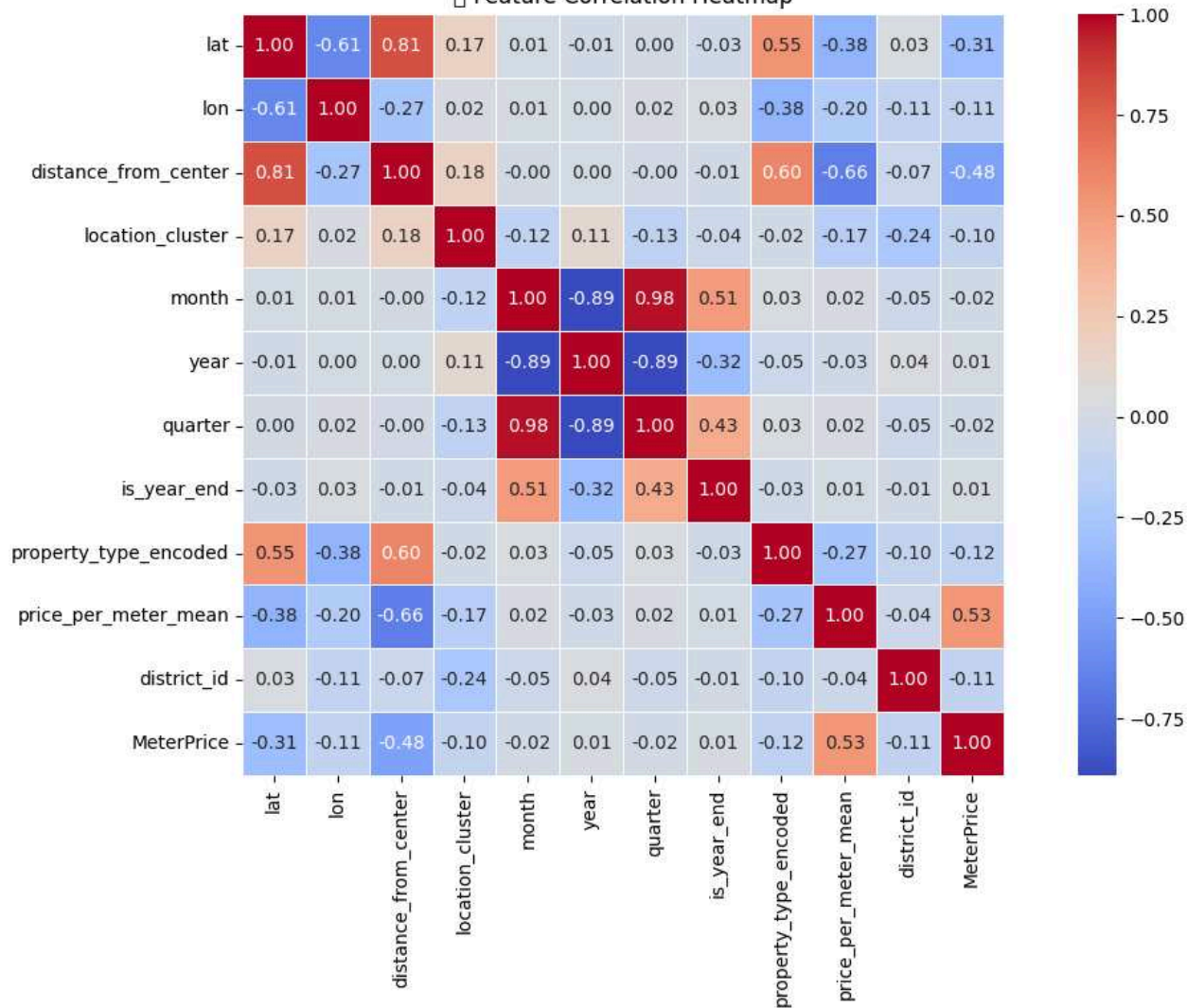
	district_id	plot_id	Type	lat	lon	MeterPrice	month	year	property_type_en	distance_from_center	location_cluster	quarter	is_year_end	property_type_encoded	price_per_meter_mean
0	21001294	21001353230	شقق	21.757199	39.173688	2691	12	2024	apartment	23.743010	9	4	1	0	1563.000000
1	21001293	210012112	شقق	21.541943	39.196728	2596	9	2024	apartment	2.660247	1	3	0	0	4875.392857
2	21001293	21001475155	شقق	21.533378	39.190417	3237	9	2024	apartment	2.244313	1	3	0	0	4875.392857
3	21001293	210011989	شقق	21.541744	39.197044	2480	9	2024	apartment	2.696660	1	3	0	0	4875.392857
4	21001293	210012108	شقق	21.541999	39.196990	3395	9	2024	apartment	2.688916	1	3	0	0	4875.392857

```
In [28]: # Only keep numeric columns for correlation
numeric_df = data.select_dtypes(include='number')

# Calculate correlation matrix
corr_matrix = numeric_df.corr()

# Draw heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", square=True, linewidths=0.5)
plt.title("🔗 Feature Correlation Heatmap")
plt.tight_layout()
plt.show()
```

Feature Correlation Heatmap



```
In [16]: # Get price column and type info
price_col = 'MeterPrice'
type_info = full_data[['lat', 'lon', 'property_type_en']].drop_duplicates()
data_with_types = data.merge(type_info, on=['lat', 'lon'], how='left')

In [17]: # Prepare features and target
X = data.drop(columns=[price_col])
y = data[price_col]

In [18]: # Train-test split (random split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [21]: # Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
joblib.dump(scaler, 'models/scaler.pkl')
```

Out[21]: ['models/scaler.pkl']

```
In [20]: # Save scaler for later use
joblib.dump(scaler, 'models/feature_scaler.pkl')
```

Out[20]: ['models/feature_scaler.pkl']

```
In [24]: ### LINEAR REGRESSION ###
linear_model = LinearRegression()
linear_model.fit(X_train_scaled, y_train)
linear_pred = linear_model.predict(X_test_scaled)
linear_r2 = r2_score(y_test, linear_pred)
linear_rmse = np.sqrt(np.mean((y_test - linear_pred) ** 2))
print(f"Linear: R2={linear_r2:.3f}, RMSE={linear_rmse:.0f}")
joblib.dump(linear_model, 'models/linear_model.pkl')
```

Linear: R2=0.271, RMSE=1316

Out[24]: ['models/linear_model.pkl']

```
In [25]: ### RIDGE REGRESSION ###
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train_scaled, y_train)
ridge_pred = ridge_model.predict(X_test_scaled)
ridge_r2 = r2_score(y_test, ridge_pred)
ridge_rmse = np.sqrt(np.mean((y_test - ridge_pred) ** 2))
print(f"Ridge: R2={ridge_r2:.3f}, RMSE={ridge_rmse:.0f}")
joblib.dump(ridge_model, 'models/ridge_model.pkl')
```

Ridge: R2=0.271, RMSE=1316

Out[25]: ['models/ridge_model.pkl']

```
In [27]: ### LASSO REGRESSION ###
lasso_model = Lasso(alpha=1.0, max_iter=3000)
lasso_model.fit(X_train_scaled, y_train)
lasso_pred = lasso_model.predict(X_test_scaled)
lasso_r2 = r2_score(y_test, lasso_pred)
lasso_rmse = np.sqrt(np.mean((y_test - lasso_pred) ** 2))
print(f"Lasso: R2={lasso_r2:.3f}, RMSE={lasso_rmse:.0f}")
joblib.dump(lasso_model, 'models/lasso_model.pkl')
```

Lasso: R2=0.272, RMSE=1316

Out[27]: ['models/lasso_model.pkl']

In []: