CSCI316 ASSIGNMENT

Data Exploration

OUR DATA

The dataset we will for our analysis is being shared by Dubai Land Department.

The dataset used for our analysis consists of 70,741 rows and 16 columns, capturing comprehensive details on real estate valuations, including transaction history, property characteristics, and pricing insights.

Initial Data columns

- procedure_id
- procedure_name_ar
- procedure_name_en
- procedure_year
- procedure_number
- instance_date
- actual_worth
- row_status_code
- procedure_area
- property_type_id
- property_type_ar
- property_type_en
- property_sub_type_id
- property_sub_type_ar
- property_sub_type_en
- area_id
- area_name_ar
- area_name_en
- actual_area
- property_total_value

OUR PROBLEM

Accurately predicting property prices is a critical challenge in real estate investment, banking, and urban planning.

Traditional valuation methods like Comparative Market Analysis (CMA) and other pricing models often fail to capture complex relationships between property features and market dynamics.

- Non-linearity: Property values depend on multiple interacting factors (location, size, amenities, demand).
- Market Fluctuations: Prices change due to economic conditions, supply-demand shifts, and policy changes.
- Data Limitations: Missing values, outliers, and incorrect pricing make predictions less reliable.

Our Goal

To build a more accurate model using ensemble learning techniques to provide better real estate predictions.

DATA PREPROCESSING

Data Exploration

Our initial dataset contained columns which were the Arabic counterparts to some of the columns. As such, we chose to work with the English columns and dropped any columns that were in Arabic.

```
# Identify Arabic columns (those ending with '_ar')
arabic_columns = [col_name for col_name in df.columns if col_name.endswith("_ar")]

# Drop Arabic columns
df_cleaned = df.drop(*arabic_columns)
```

Arabic Columns

- procedure_name_ar
- property_type_ar
- property_sub_type_ar
- area_name_ar

Data Cleaning

Removed Columns:

- procedure_id, procedure_name_en, procedure_number

  They do not contribute meaningful information to property valuation.

- property_type_id

  This is a numerical ID that can be replaced with property_type_en, which is more interpretable.

- instance_date

  This may not be relevant for valuation predictions unless time-based trends are analyzed separately.

Handling Missing Values

Initial Data Exploration & Identifying Missing Values

- Loaded the dataset into PySpark for analysis.
- Missing data found in key columns like instance_date, procedure_area, and property_sub_type_en.

Handling Missing Values

- procedure_area: Replaced with the median value to maintain data distribution.
- property_sub_type_en: Missing and "Unknown" values replaced with the most common sub-type.
- Ensured data completeness for accurate analysis.

```
if all(col in df.columns for col in ["property_type_en", "property_sub_type_en", "area_name_en"]):
    df.fillna({"property_type_en": "Unknown", "property_sub_type_en": "Unknown", "area_name_en": "Unknown"}, inplace=True)
    df[["property_type_en", "property_sub_type_en", "area_name_en"]] = df[["property_type_en", "property_sub_type_en", "area_name_en"]].astype(str)
df = pd.get_dummies(df, columns=["property_type_en", "property_sub_type_en", "area_name_en"], drop_first=True)
```

Outlier Detection & Removal

Outlier Detection & Removal

- Identified outliers in the actual_worth column using the Interquartile Range (IQR) method.
- IQR Calculation:
  - Q1 (25th percentile) and Q3 (75th percentile) were computed.
  - Outlier Range: Below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR.
- Rows with values outside this range were removed to prevent skewed analysis.

Why Use the IQR Method?

- More robust than mean-based methods like Z-score, as it is resistant to extreme values.
- Focuses on the middle 50% of data, ensuring a cleaner and more reliable dataset.
- Simple to implement and widely used in statistical data cleaning.

Feature Engineering

Feature Engineering Steps:

- Dropped Irrelevant Columns: procedure_id, procedure_name_en, procedure_number, property_type_id, instance_date.
- Categorical Feature Encoding:
  - Used LabelEncoder() for: row_status_code, property_type_en, area_name_en.
  - Converted categorical columns into numerical format.
- One-Hot Encoding:
- Applied pd.get_dummies() to: property_type_en, property_sub_type_en, area_name_en.
- Dropped first category to avoid redundancy.

Data Scaling

Purpose of Scaling:

- Ensures numerical features are on the same scale.
- Improves model performance by preventing dominance of large values.

Interpreting Scaled Values:

- Mean-centered: Values are adjusted to have a mean of 0.
- Standardized spread: Each value represents standard deviations from the mean.
- Positive values indicate above-average data points; negative values indicate below-average.

	actual_worth	actual_area	property_total_value
0	0.095771	-0.118303	0.091028
1	2.958169	0.035429	2.879571
2	-0.450424	0.207514	-0.441075
3	-0.545415	-0.057902	-0.533615
4	1.659126	0.243423	1.614045

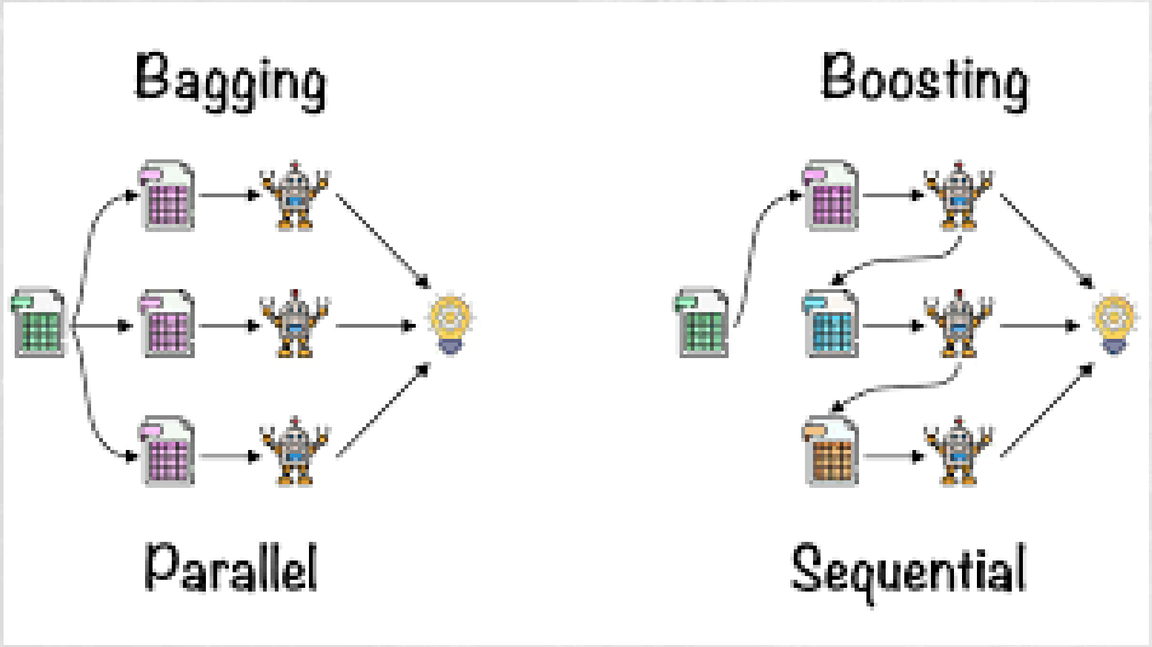
Ensemble Learning

Ensemble Learning is a technique that combines multiple weak models (base learners) to create a stronger, more accurate model.

It helps reduce errors, improve generalization, and increase stability in predictions.

Two common ensemble methods

- Bagging (Bootstrap Aggregating) → Reduces variance
- Boosting → Reduces bias



Data Splitting & Validation

Splitting Strategy:

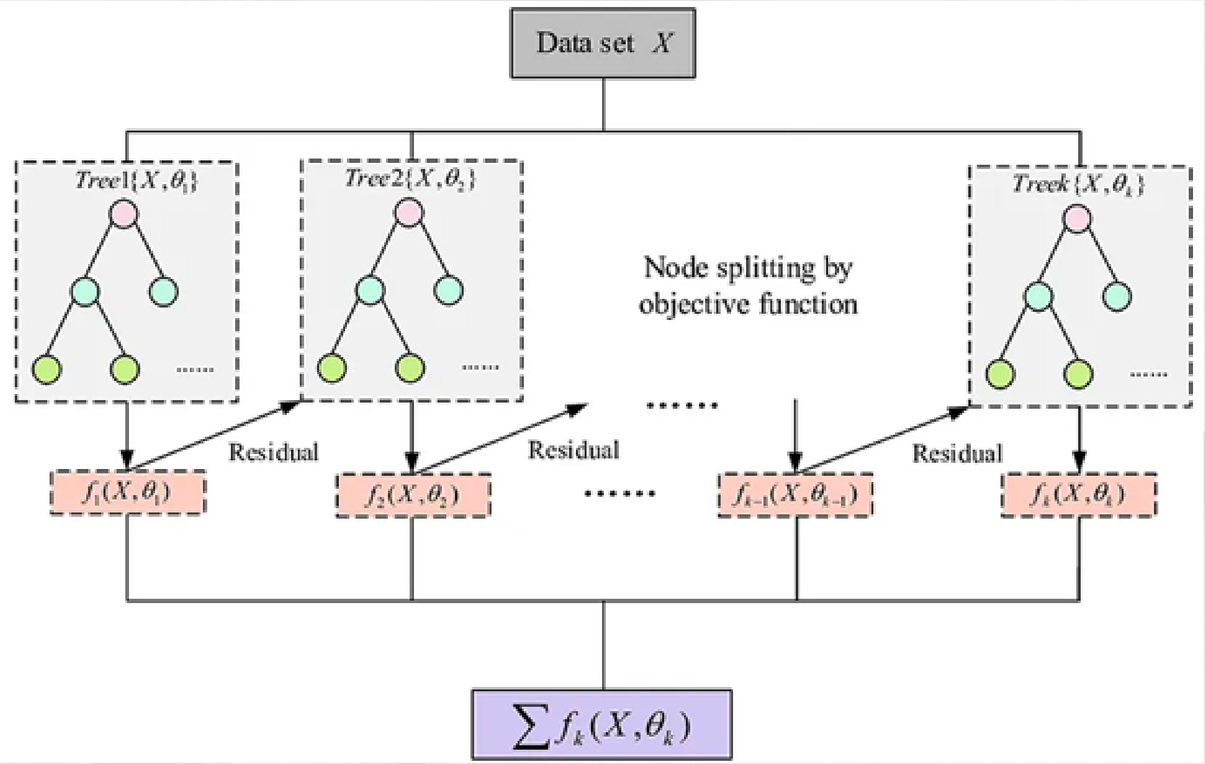
- 80% Training, 10% Validation, 10% Testing.
- train_test_split() used with random_state=42 for reproducibility.

Preprocessing Validation:

- Ensured categorical encoding and numerical scaling were applied correctly.
- Cross-validation ensures that the model generalizes well and prevents overfitting.
- Each fold in cross-validation provides insights into model performance across different data subsets.

XGBoost

- Extreme Gradient Boosting (XGBoost) is an advanced machine learning algorithm based on gradient boosting.
- It sequentially trains decision trees, where each tree corrects the errors of the previous ones.
- Used for structured data problems, such as real estate price prediction.



Bagging Ensemble Model

What is Bagging?

- Builds multiple decision trees in parallel to reduce variance.
- Predictions are aggregated to provide more stable results.

Advantages:

- Reduces overfitting compared to individual decision trees.
- Improves model stability by considering multiple perspectives.

```python
class BaggingRegressorScratch:
    def __init__(self, base_estimator=DecisionTreeRegressor, n_estimators=10, max_samples=0.8):
        self.base_estimator = base_estimator
        self.n_estimators = n_estimators
        self.max_samples = max_samples
        self.models = []

    def fit(self, X, y):
        np.random.seed(42)
        n_samples = int(self.max_samples * len(X))
        for _ in range(self.n_estimators):
            sample_indices = np.random.choice(len(X), n_samples, replace=True)
            X_sample, y_sample = X.iloc[sample_indices], y.iloc[sample_indices]
            model = self.base_estimator()
            model.fit(X_sample, y_sample)
            self.models.append(model)

    def predict(self, X):
        predictions = np.array([model.predict(X) for model in self.models])
        return np.mean(predictions, axis=0)
```

Cross-Validation Strategy

10-Fold Cross-Validation:

- Training set split into 10 subsets.
- Model trained on 9 subsets and validated on 1.
- Ensures robustness and avoids overfitting.

Evaluation Metrics Explained:

- Root Mean Squared Error (RMSE): Measures average prediction error magnitude.
- R-squared Score (R^2): Measures model's goodness-of-fit.
- Mean Absolute Error (MAE): Measures absolute differences between actual and predicted values.
- Median Absolute Error (MedAE): Robust to outliers.
- Explained Variance Score: Measures proportion of variance captured by the model.

```
Mean RMSE (Cross-Validation): 0.06
Standard Deviation of RMSE (Cross-Validation): 0.02
Mean R-squared (Cross-Validation): 1.00
Standard Deviation of R-squared (Cross-Validation): 0.00
Mean MAE (Cross-Validation): 0.01
Standard Deviation of MAE (Cross-Validation): 0.00
Mean MedAE (Cross-Validation): 0.01
Standard Deviation of MedAE (Cross-Validation): 0.00
Mean Explained Variance (Cross-Validation): 1.00
Standard Deviation of Explained Variance (Cross-Validation): 0.00
```

Results

Model	Result
RMSE on Test Set	0.05
R-squared on Test Set	1
MAE on Test Set	0.01
Mean Explained Variance on Test Set	1

Insights:

- High R^2 indicates excellent model fit.
- Low RMSE and MAE suggest minimal prediction error.

Conclusion

Key Takeaways:

- Successfully implemented XGBoost for real estate valuation predictions.
- Bagging ensemble method improved model stability and reduced variance.
- Cross-validation ensured robustness and prevented overfitting.

Final Model Performance:

- R^2 Score: 1.00 – Indicates near-perfect predictions.
- RMSE: 0.05, MAE: 0.01 – Minimal error, demonstrating high accuracy.

References

Lopez, F. (n.d.). Bagging and Boosting. Available at: https://towardsdatascience.com/ensemble-learning-bagging-boosting-3098079e5422/ [Accessed 7 Feb. 2025].

Omarzai, F. (2024) XGBoost Example, XGBoost Classification In Depth. Available at: https://medium.com/@fraidoonomarzai99/xgboost-classification-in-depth-979f11ef4bf9 (Accessed: 07 February 2025).