

# 1. Exploratory Data Analysis (EDA)

**Definition:** Process of understanding your dataset before building models.

**Purpose:** - Check data distribution - Detect outliers - Find relationships between features - Detect missing values - Decide feature engineering steps

**Example:**

```
import pandas as pd
data = pd.read_csv('/kaggle/input/housedata/data.csv')
print(data.describe())
print(data.isnull().sum())
```

**Notes:** - Essential step before modeling - Helps to improve model accuracy

---

# 2. Histogram and Bins

**Definition:** Graphical representation of the distribution of numerical data.

- X-axis: range of values
- Y-axis: frequency (how many values fall in each range)
- Bins: intervals of data

**Importance of Bins:** - Too few bins: loss of detail - Too many bins: noisy plot

**Example:**

```
import matplotlib.pyplot as plt
plt.hist(data['price'], bins=50)
plt.title("Price Distribution")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```

**Notes:** - Each bin counts how many values fall within its range - Adjust bins to see clearer patterns

---

## 3. Boxplot

**Definition:** Visualizes distribution and outliers of numerical data.

- Box: 25th to 75th percentile (IQR)
- Line inside box: median (50th percentile)
- Whiskers: data range (usually  $1.5 \times \text{IQR}$ )
- Points outside whiskers: outliers

**Importance:** - Quickly identifies outliers - Shows spread and symmetry of data - Useful for scaling and cleaning

**Example:**

```
plt.boxplot(data['price'])
plt.title("Boxplot of House Prices")
plt.show()
```

---

## 4. Hyperparameter Tuning and GridSearchCV

**Hyperparameters:** Parameters set before training, e.g., number of trees, max depth.

**Goal:** Find the best combination to improve model performance.

**GridSearchCV:** - Tries all combinations of hyperparameters - Uses cross-validation to evaluate - Returns the best combination

**Example:**

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10]
}

rf = RandomForestRegressor(random_state=42)
grid = GridSearchCV(rf, param_grid, cv=5, scoring='r2')
grid.fit(X_train_scaled, Y_train)
```

```
print("Best Hyperparameters:", grid.best_params_)
print("Best R2 Score:", grid.best_score_)
```

**Explanation:** - `n_estimators=50,100,200` → number of trees - `max_depth=10,20,None` → depth of trees - `min_samples_split=2,5,10` → min samples to split node - `cv=5` → 5-fold cross-validation - `grid.best_params_` → best hyperparameter combination

**Importance:** - Default parameters may not be optimal - Proper tuning improves accuracy and generalization

## Summary Table

Concept	Purpose	Example / Key Notes
EDA	Understand data	Summary stats, plots, missing values
Histogram	Show distribution	<code>plt.hist(data['price'], bins=50)</code>
Bins	Grouping intervals in histogram	Adjust for clarity
Boxplot	Detect outliers & spread	<code>plt.boxplot(data['price'])</code>
Hyperparameters	Pre-set model settings	Number of trees, depth, learning rate
GridSearchCV	Find best hyperparameters	Tries all combinations with cross-validation