**Simple Notes Of Chapter 2**

**1. Machine Learning Project Lifecycle**

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1. **Frame the problem** | Regression task (predicting continuous values). Target variable: MedHouseVal. |
| 2. **Get the data** | Used the built-in dataset from fetch\_california\_housing() in scikit-learn. |
| 3. **Explore the data** | Visualized distributions, relationships, outliers, and correlations. |
| 4. **Prepare the data** | Cleaned missing values, scaled features, added new features (e.g., rooms per household). |
| 5. **Select and train models** | Compared Linear Regression, Decision Tree, Random Forest, and SVR. |
| 6. **Evaluate models** | Used RMSE and MAE metrics. |
| 7. **Tune hyperparameters** | (Optional: GridSearchCV or RandomizedSearchCV). |
| 8. **Deploy the solution** | Built an interactive Streamlit app to predict prices using sliders. |

**2. Data Exploration**

**➤ Dataset Overview:**

* Loaded using:

python

CopyEdit

from sklearn.datasets import fetch\_california\_housing

housing = fetch\_california\_housing(as\_frame=True).frame

**➤ Key Features:**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| MedInc | Median income in district |
| HouseAge | Median house age |
| AveRooms | Average rooms per household |
| AveBedrms | Average bedrooms per household |
| Population | District population |
| AveOccup | Average occupancy |
| Latitude, Longitude | Geo-coordinates |

**➤ Visualizations:**

* Histograms
* Correlation heatmap
* Scatter plots (MedInc vs MedHouseVal)
* Geographical price heatmap (Longitude, Latitude, colored by MedHouseVal)
* Scatter matrix for top correlated features

**3. Data Preprocessing**

**➤ Key Preprocessing Techniques:**

|  |  |
| --- | --- |
| **Step** | **Method** |
| **Missing values** | SimpleImputer(strategy='median') |
| **Feature Scaling** | StandardScaler() (removes mean, scales to unit variance) |
| **Feature Engineering** | Added ratios like: bedrooms\_per\_room, income\_per\_room |
| **Train-test split** | train\_test\_split(test\_size=0.2) to evaluate on unseen data |

**Pipeline Explained:**

A **Pipeline** in scikit-learn is a tool to bundle preprocessing and modeling steps so you can treat them as a single object.

Example:

python

CopyEdit

from sklearn.pipeline import Pipeline

pipeline = Pipeline([

("imputer", SimpleImputer(strategy="median")),

("scaler", StandardScaler())

])

This ensures **consistent transformation** for both training and testing data.

**4. Model Training**

**➤ Trained Models:**

|  |  |  |
| --- | --- | --- |
| **Model** | **Purpose** | **Code** |
| Linear Regression | Simple linear model | LinearRegression() |
| Decision Tree Regressor | Rule-based, non-linear splits | DecisionTreeRegressor() |
| Random Forest Regressor | Ensemble of trees | RandomForestRegressor() |
| SVR | Kernel-based regression | SVR(kernel="rbf", C=100, gamma=0.1) |

**5. Model Evaluation**

**➤ Metrics:**

| **Metric** | **Use** |
| --- | --- |
| **RMSE (Root Mean Squared Error)** | Penalizes large errors. Good for most regression problems. |
| **MAE (Mean Absolute Error)** | Measures average error. Robust to outliers. |

python

CopyEdit

rmse = np.sqrt(mean\_squared\_error(y\_test, predictions))

mae = mean\_absolute\_error(y\_test, predictions)

**➤ Comparison Table Example:**

|  |  |  |
| --- | --- | --- |
| **Model** | **RMSE** | **MAE** |
| Random Forest | 0.492 | 0.328 |
| Support Vector Regressor | 0.518 | 0.365 |
| Linear Regression | 0.735 | 0.532 |
| Decision Tree | 0.754 | 0.497 |

**6. Deployment with Streamlit**

**➤ Purpose:**

To create a live interactive web app where users can input values using sliders and get instant predictions.

**➤ Features:**

* Beautiful background image with dark overlay
* Color-coded buttons and sliders (yellow contrast on dark)
* Sidebar with "About Project" section
* Real-time predictions

**➤ Example:**

bash

CopyEdit

streamlit run streamlit\_app/app.py

**7. Project Deliverables**

|  |  |
| --- | --- |
| **Item** | **Description** |
| 📄 Chapter 2 exercises | SVR, RandomizedSearchCV, full pipeline, feature selection |
| 📊 Data visualizations | Histograms, heatmap, scatter plots |
| 📋 Model table | RMSE + MAE of all models |
| 🖼️ Streamlit UI | Functional and styled UI with inputs and prediction |
| 🔗 GitHub repo | Includes all .py, .ipynb, and requirements.txt |
| 📁 requirements.txt | Contains all dependencies |
| ✅ Notebooks | Clean and modular Jupyter files for training and testing |

**Concepts That I Learned:**

* Pipeline creation and execution
* Data transformation with scikit-learn
* Feature engineering basics
* Model evaluation metrics
* GridSearchCV & RandomizedSearchCV
* Streamlit UI and deployment
* Correlation & distribution visualization