AI Roadmap / House Prices

(Extra notes)

* Correlation is only calculated between numeric columns. Heatmap to spot this.
* Linear Regression tries to find the best straight line that predicts a number (value) based on input features.
* 🔸 Real-life example:
* You want to predict a house price based on its size (in square feet).
* If you plot:
* X-axis: House size
* Y-axis: Price
* Linear regression will try to draw a straight line that best fits the data points — so you can predict the price for any size.
* What is the model learning?
* # It learns the best values for the weights (w) and bias (b) so that the predictions are as close as possible to the real values — using something called a loss function.

**MSE (Mean Squared Error)**

🟡 It tells how wrong the predictions are, on average. 🟡 Our MSE is 1.47 billion, which sounds big — but that’s because house prices are big numbers. 🟢 Lower MSE = better model.

**🔹 R² (R-squared Score)**

🟢 Tells how well the model explains the data. 🟢 0.8075 means the model explains 81% of the pattern in house prices.

* What scikit-learn does:
* # Provides ML Algorithms
* # Classification (e.g., spam detection)
* # Regression (e.g., predicting prices)
* # Clustering (e.g., customer segmentation)
* # Data Preprocessing
* # Scaling, encoding, missing value handling
* # Model Selection
* # Train/test split, cross-validation, hyperparameter tuning
* # Evaluation
* # Accuracy, confusion matrix, precision/recall, etc.
* # Pipelines
* # Combine preprocessing + model into one workflow
* scikit-learn: Great for small to medium datasets, and classical ML (SVM, Random Forests, etc.)
* # XGBoost/LightGBM/CatBoost: Better for structured/tabular data, for gradient boosting
* # TensorFlow/PyTorch: Best for deep learning (images, NLP, etc.)
* # Statsmodels: Use when you need p-values, confidence intervals, etc.
* Gradient Boosting is a machine learning technique used for classification and regression. It builds a strong model by combining many weak models (usually decision trees), where each new model corrects the errors of the previous ones using gradient descent. This process helps minimize prediction errors step by step. It's highly effective for structured data and is used in popular libraries like XGBoost, LightGBM, CatBoost, and scikit-learn. Gradient Boosting is widely used in tasks like fraud detection, credit scoring, and Kaggle competitions.
* # Gradient Boosting models (like decision trees) are very good at capturing patterns in data with clearly defined features and rows — like tables (CSV, Excel, databases).
* # They don’t need features to be on the same scale or to be transformed heavily (like neural networks often do).
* # They can handle missing values and categorical data (especially libraries like CatBoost).
* # Tabular data is common in business and finance (e.g., customer data, transaction logs), so these models perform very well there.
* **What is a Loss Function?**
* A **loss function** measures **how wrong** your machine learning model’s predictions are compared to the actual target values.
* It gives a number (called the **loss**) that tells you the **error** for a single prediction or for the whole dataset.
* **Why is it important?**
* The goal of training a model is to **minimize the loss**.
* The lower the loss, the better the model is performing.
* list of common loss functions with a plain, intuitive explanation of **how they calculate the loss**:
* **1. Mean Squared Error (MSE)**
* **Used for:** Regression
* **Logic:** Calculate the difference between the predicted and actual values, square these differences (to penalize big errors more), and then average them over all data points.
* **Why?** Penalizes large errors heavily, encouraging the model to be accurate everywhere.
* **2. Mean Absolute Error (MAE)**
* **Used for:** Regression
* **Logic:** Calculate the absolute difference between predicted and actual values and average it.
* **Why?** Treats all errors equally, more robust to outliers than MSE.
* **3. Huber Loss**
* **Used for:** Regression
* **Logic:** Combines MSE and MAE — behaves like MSE for small errors and like MAE for big errors.
* **Why?** Provides a balance; less sensitive to outliers but still penalizes small errors well.
* **4. Log Loss (Cross-Entropy Loss)**
* **Used for:** Classification (especially binary)
* **Logic:** Measures how close the predicted probabilities are to the actual class (0 or 1). It penalizes wrong predictions more when the model is confident but wrong.
* **Why?** Encourages the model to give probabilities close to 1 for the correct class.
* **5. Categorical Cross-Entropy**
* **Used for:** Multi-class classification
* **Logic:** Similar to Log Loss, but works when there are more than two classes. It measures how close the predicted probability distribution is to the actual class distribution.
* **Why?** Helps the model predict the right class with high confidence.
* **6. Hinge Loss**
* **Used for:** Support Vector Machines (SVMs) in classification
* **Logic:** Penalizes predictions that are on the wrong side of the decision boundary or too close to it.
* **Why?** Encourages a margin of safety between classes.
* **7. KL Divergence (Kullback-Leibler Divergence)**
* **Used for:** Comparing two probability distributions
* **Logic:** Measures how one probability distribution diverges from a reference distribution (usually the true distribution).
* **Why?** Used in models where you want predicted probabilities to closely match true distributions.

### **SVM (Support Vector Machine)**

SVM is a machine learning algorithm used mostly for **classification**. It works by finding the **best boundary (hyperplane)** that separates classes with the **widest possible margin**. The closest data points to this boundary are called **support vectors**. If data isn’t linearly separable, SVM uses a **kernel** to map it into a higher dimension where it can be separated.

* **Support Vectors**
* These are the **data points closest to the boundary** (the line or hyperplane that separates the classes).
* They are the most **important** points because:
* **They define the position and direction of the boundary.**
* If you remove them, the boundary might change.
* **✅ Kernel**
* A **kernel** is a mathematical trick SVM uses when the data **can’t be separated with a straight line**.
* It **transforms the data** into a higher dimension where it **can** be separated.
* You don’t have to manually transform the data — the kernel does it for you.
* 🧠 Example: Imagine drawing a circle around data in 2D. It’s hard with a straight line — but if you go to 3D, a flat plane can separate it easily.
* **🔁 In short:**

| * **Term** | * **Simple Meaning** |
| --- | --- |
| * **Support Vectors** | * The edge cases — closest points to the decision boundary |
| * **Kernel** | * A trick to handle complex, non-linear data by mapping it into higher dimensions |