**1. Python Basics 🐍**

* **Why**: Python is the most widely used language for machine learning due to its simplicity and rich ecosystem of libraries like Pandas, NumPy, and Scikit-learn, which simplify data manipulation and model building.

**2. Mathematics for Machine Learning 📐**

* **Linear Algebra**: vector ,matrices
  + **Why**: Fundamental for operations like matrix multiplication, which are key to training models and handling large datasets in machine learning.
* **Calculus**:
  + **Why**: Helps in optimizing machine learning models using techniques like **gradient descent**, which adjust model parameters to minimize error.
* **Statistics**:
  + **Why**: Essential for understanding data distributions and relationships between variables, which directly influence model performance. It forms the foundation for models like **regression** and **probabilistic models**.
* **Probability Theory**:
  + **Why**: Core to understanding probabilistic models in machine learning, such as Naive Bayes, and concepts like conditional probability and likelihood.

**3. Regression (Under Statistics) 📊**

* **Why**: Regression is a **statistical method** used to model the relationship between dependent and independent variables. In machine learning, **regression** is used to predict **continuous outcomes** like price or sales, making it a key part of **supervised learning**.
  + **Linear Regression**:
    - **Why**: Models the linear relationship between input features and a continuous output (e.g., predicting price based on demand).
  + **Multiple Linear Regression**:
    - **Why**: Extends linear regression to multiple input variables, improving the model's ability to predict outcomes.
  + **Polynomial Regression**:
    - **Why**: Captures non-linear relationships between variables when the data shows more complexity.

**4. Data Handling and Manipulation (Tools: Pandas & NumPy) 📊**

* **Why**: Clean, structured, and well-understood data is the foundation for building accurate models. Tools like **Pandas** and **NumPy** are critical for handling and transforming data before feeding it into machine learning models.

**5. Supervised Learning: Regression & Classification 🔍**

* **Why**: Supervised learning involves learning from labeled data to make predictions. It includes:
  + **Regression Models**:
    - **Why**: Predict continuous outcomes (e.g., price prediction).
  + **Classification Models**:
    - **Why**: Predict categorical outcomes (e.g., whether a customer will buy an item or not).
    - **Examples**: Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVM).

**6. Unsupervised Learning (Clustering, Dimensionality Reduction) 🔄**

* **Why**: Unsupervised learning is used when you have no labeled data and need to find patterns or groupings. This is useful for tasks like **customer segmentation**.
  + **Clustering (K-Means)**:
    - **Why**: Groups similar data points together without pre-labeled data.
  + **Dimensionality Reduction (PCA)**:
    - **Why**: Simplifies data by reducing the number of features while preserving important patterns, helping with data visualization and speeding up training.

**7. Model Evaluation & Tuning 🎯**

* **Why**: Once models are built, they need to be evaluated using metrics like **accuracy**, **precision**, **mean squared error (MSE)**, and **R² score**. Model tuning ensures optimal performance through techniques like **cross-validation** and **hyperparameter tuning**.

**8. Neural Networks (Deep Learning) 🤖**

* **Why**: Neural networks, part of **deep learning**, are designed to handle more complex tasks like image recognition and sequential data. They outperform traditional models in handling vast amounts of data and non-linear relationships.
  + **Artificial Neural Networks (ANN)**:
    - **Why**: Solve problems where traditional machine learning methods struggle with complexity.
  + **Convolutional Neural Networks (CNN)**:
    - **Why**: Highly effective for image recognition tasks.
  + **Recurrent Neural Networks (RNN)**:
    - **Why**: Used for sequential data, like time-series forecasting or text processing.

**9. Model Deployment & Integration 🌐**

* **Why**: Once trained, models need to be deployed in real-world applications. Using tools like **Flask** or **FastAPI**, you can expose your model as a web API, allowing your applications (e.g., a Java-based app) to interact with it in real time.
  + .

### ****1. Learn Statistics & Probability (with Regression Focus)**** 📐

Now, here’s where **regression** comes in. Understanding **statistics** and **probability** is the foundation of **machine learning**, especially for **supervised learning** techniques like **regression**.

#### ****Key Concepts****:

* **Descriptive Statistics**: Mean, median, standard deviation.
* **Probability**: Basic probability, distributions (normal, binomial).
* **Linear Relationships**: Correlation and covariance.

#### ****Important Regression Topics**** (Focus Here):

1. **Linear Regression**:
   * **What It Is**: A method to model the relationship between a dependent variable (**e.g., price**) and one or more independent variables (**e.g., demand, stock**).
   * **Why It's Important**: In your case, you'll use linear regression to **predict the price** of an item based on factors like demand and stock levels.
   * **Math Behind It**: Simple formula: Y = aX + b, where:
     + **Y**: The output (predicted price).
     + **X**: The input variable (e.g., demand).
     + **a**: The slope (how much Y changes for a unit change in X).
     + **b**: The intercept.
2. **Multiple Linear Regression**:
   * **What It Is**: An extension of linear regression where multiple variables (demand, stock, season, etc.) predict the outcome.
   * **Math Behind It**: Y = a1X1 + a2X2 + ... + b.
3. **Polynomial Regression** (Optional, but useful):

This allows you to fit non-linear data using a polynomial equation. Useful if your pricing model shows non-linear patterns

**1.2 Python Basics 🐍**

* **Why**: Python is the most widely used language for machine learning due to its simplicity and rich ecosystem of libraries like Pandas, NumPy, and Scikit-learn, which simplify data manipulation and model building.

### Key Differences

| **Feature** | **NumPy** | **Pandas** |
| --- | --- | --- |
| **Data Structures** | Arrays (ndarray) | Series (1D), DataFrame (2D) |
| **Data Handling** | Best for numerical data | Best for labeled and structured data |
| **Indexing** | Numeric indexing | Labeled indexing (can use strings) |
| **Functionality** | Mathematical functions, linear algebra, random number generation | Data manipulation, filtering, grouping, merging, and joining |
| **Performance** | Generally faster for numerical operations | Slightly slower due to additional features but optimized for data analysis |
| **Use Cases** | Scientific computations, numerical analysis | Data analysis, data cleaning, data visualization |

### 2.Supervised Learning: An Introduction

Supervised learning is a fundamental concept in machine learning where the model learns from labeled data to make predictions. Let's break down the key components, focusing on regression and classification.

#### What is Supervised Learning?

* **Definition**: Supervised learning is a type of machine learning where the model is trained on a labeled dataset. This means that each training example is paired with an output label.
* **Goal**: The main goal is to learn a mapping from inputs (features) to outputs (labels) so that the model can make accurate predictions on unseen data.

#### Types of Supervised Learning

Supervised learning can be broadly categorized into two types: **Regression** and **Classification**.

### 1. Regression Models

* **Purpose**: Regression models are used to predict continuous outcomes. For example, predicting the price of a house based on its features (size, location, number of bedrooms).
* **Common Regression Algorithms**:
  + **Linear Regression**: Models the relationship between the dependent variable and one or more independent variables using a straight line.
  + **Polynomial Regression**: Extends linear regression by fitting a polynomial equation to the data.
  + **Ridge and Lasso Regression**: These are regularized versions of linear regression that help prevent overfitting.
* **Example**: If you want to predict the price of a car based on its age and mileage, you would use regression analysis.

### 2. Classification Models

* **Purpose**: Classification models are used to predict categorical outcomes. For example, determining whether an email is spam or not, or whether a customer will buy a product.
* **Common Classification Algorithms**:
  + **Logistic Regression**: Used for binary classification problems. It predicts the probability that a given input belongs to a particular category.
  + **Decision Trees**: A tree-like model that splits data into branches to make decisions based on feature values.
  + **Random Forests**: An ensemble of decision trees that improves accuracy and controls overfitting by averaging the results of multiple trees.
  + **Support Vector Machines (SVM)**: Finds the hyperplane that best separates different classes in the feature space.
* **Example**: If you want to classify emails as "spam" or "not spam," you would use a classification algorithm.

### 3.Unsupervised Learning: An Introduction for Beginners

Unsupervised learning is a key concept in machine learning that deals with datasets that do not have labeled outputs. The goal is to identify patterns or groupings within the data, making it particularly useful for exploratory data analysis. Let's dive into the two main types of unsupervised learning: clustering and dimensionality reduction.

#### What is Unsupervised Learning?

* **Definition**: Unsupervised learning is a type of machine learning where the model is trained on data without labeled outputs. Instead of predicting a known label, the model tries to learn the underlying structure or distribution of the data.
* **Goal**: The primary aim is to explore the data and find hidden patterns or intrinsic groupings without prior knowledge of the outcomes.

#### Types of Unsupervised Learning

Unsupervised learning can be broadly categorized into two types: **Clustering** and **Dimensionality Reduction**.

### 1. Clustering

* **Purpose**: Clustering is used to group similar data points together. It helps in identifying natural groupings within the data, which can be useful for tasks like customer segmentation or market research.
* **Common Clustering Algorithms**:
  + **K-Means Clustering**: This algorithm partitions the data into **K** distinct clusters based on feature similarity. It works by assigning data points to the nearest cluster centroid and then updating the centroids based on the assigned points.
  + **Hierarchical Clustering**: This method builds a hierarchy of clusters either by merging smaller clusters into larger ones or by splitting larger clusters into smaller ones.
  + **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: This algorithm groups together points that are closely packed together while marking points in low-density regions as outliers.
* **Example**: If you have customer data and want to segment customers into groups based on purchasing behavior, K-Means can be applied to identify these segments.

### 2. Dimensionality Reduction

* **Purpose**: Dimensionality reduction techniques are used to reduce the number of features in a dataset while preserving important patterns and relationships. This simplification can help with data visualization and can speed up the training of machine learning models.
* **Common Dimensionality Reduction Techniques**:
  + **Principal Component Analysis (PCA)**: PCA transforms the data into a lower-dimensional space by finding the directions (principal components) that maximize the variance in the data. It helps to reduce noise and redundancy.
  + **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: This technique is particularly useful for visualizing high-dimensional data in two or three dimensions by preserving local structures.
  + **Autoencoders**: These are neural networks designed to learn efficient representations of data, often used for compressing data.
* **Example**: If you have a dataset with hundreds of features, PCA can be used to reduce it to a few principal components, making it easier to visualize and analyze.

# DATE – 23/10/2024

**Why Multiply Matrices: A Simple Real-Life Example**

Let’s imagine you are running a small shop, and you sell **3 types of products**: apples, bananas, and oranges. You also have **2 types of customers**: wholesale buyers and retail customers. Now, you want to figure out how much each customer type buys of each product.

**Step 1: Create Two Matrices**

1. **Customer Purchases** (Matrix A): Each row represents a customer type (wholesale, retail), and each column represents a product (apples, bananas, oranges). Each entry in the matrix represents how many units of each product the customers bought:

A=(203050(wholesale)5810(retail))A = \begin{pmatrix} 20 & 30 & 50 \\ \text{(wholesale)} \\ 5 & 8 & 10 \\ \text{(retail)} \end{pmatrix}A=​20(wholesale)5(retail)​308​5010​​

This means, for example, that wholesale customers bought 20 apples, 30 bananas, and 50 oranges, while retail customers bought 5 apples, 8 bananas, and 10 oranges.

1. **Product Prices** (Matrix B): Each column represents the price of each product:

B=(2(apples)1.5(bananas)3(oranges))B = \begin{pmatrix} 2 \\ \text{(apples)} \\ 1.5 \\ \text{(bananas)} \\ 3 \\ \text{(oranges)} \end{pmatrix}B=​2(apples)1.5(bananas)3(oranges)​​

Apples cost $2 per unit, bananas cost $1.50, and oranges cost $3.

**Step 2: Multiply the Matrices**

Now, you want to know how much revenue each customer type is generating for your store. This is where matrix multiplication comes in! By multiplying the two matrices, you combine the purchase information with the prices to calculate the **total revenue from each customer type**:

Revenue=A×B\text{Revenue} = A \times BRevenue=A×B

Performing the multiplication:

Revenue=(203050(wholesale)5810(retail))×(2(apples)1.5(bananas)3(oranges))\text{Revenue} = \begin{pmatrix} 20 & 30 & 50 \\ \text{(wholesale)} \\ 5 & 8 & 10 \\ \text{(retail)} \end{pmatrix} \times \begin{pmatrix} 2 \\ \text{(apples)} \\ 1.5 \\ \text{(bananas)} \\ 3 \\ \text{(oranges)} \end{pmatrix}Revenue=​20(wholesale)5(retail)​308​5010​​×​2(apples)1.5(bananas)3(oranges)​​

For wholesale customers:

(20×2)+(30×1.5)+(50×3)=40+45+150=235(20 \times 2) + (30 \times 1.5) + (50 \times 3) = 40 + 45 + 150 = 235(20×2)+(30×1.5)+(50×3)=40+45+150=235

For retail customers:

(5×2)+(8×1.5)+(10×3)=10+12+30=52(5 \times 2) + (8 \times 1.5) + (10 \times 3) = 10 + 12 + 30 = 52(5×2)+(8×1.5)+(10×3)=10+12+30=52

So, your total revenue from wholesale customers is **$235**, and from retail customers, it’s **$52**.

**Why This Example Relates to Machine Learning**

In this example:

* **Customer Purchases (Matrix A)** is like your **input data** (features).
* **Product Prices (Matrix B)** are like **weights** in a machine learning model.
* The result (Revenue) is like your **output**.

**What Are "Weights" in Machine Learning?**

In machine learning models, **weights** are numbers (like the prices in the example) that tell the model how important each feature (product) is in making predictions.

* **Input features** could be anything—age, height, or temperature, depending on the problem you're solving.
* **Weights** represent how strongly each feature affects the outcome. For example, in predicting house prices, features like the number of bedrooms or location are important. The model learns how much each feature should contribute (weights) to the final price.

**Step 1: Understanding Weights in House Price Prediction**

Imagine you are trying to predict the price of a house based on these **features**:

* **Number of bedrooms**
* **Square feet**
* **Location (city/town)**
* **House color**

Let’s assume a machine learning model is trained to predict house prices based on these features. After training, the model assigns **weights** to each feature, which tell us how much each feature contributes to the final price prediction.

For example:

* Number of bedrooms → **Weight = 50**
* Square feet → **Weight = 200**
* Location → **Weight = 300**
* House color → **Weight = 5**

These weights are multiplied by the feature values (e.g., 3 bedrooms, 2000 square feet) to make the price prediction.

**Step 2: How Does the Model Find the Weights?**

The process of finding the optimal weights for each feature happens during **training**, and it usually follows these steps:

1. **Start with Random Weights**: The model starts with random weights. Initially, it doesn't know which features are important, so it guesses.
2. **Make a Prediction**: The model makes a prediction using the current weights. For example, if the current weights are random, the prediction might be completely wrong.
3. **Calculate the Error (Loss)**: The model compares the predicted house price to the actual house price and calculates an **error**. This error tells the model how far off it is from the correct price.
4. **Adjust the Weights**: The model adjusts the weights slightly to reduce the error. This process is called **gradient descent**. It keeps adjusting the weights in the right direction (based on how wrong the prediction was) so that the next prediction is a little closer to the actual value.
5. **Repeat**: The model repeats this process over and over, adjusting the weights each time, until the predictions are very close to the actual values. At this point, the model has learned the optimal weights for each feature.

This process of finding the optimal weights is what enables the model to learn the relationship between the features (like number of bedrooms) and the target output (house price).

**Step 3: What Happens After Finding the Weights?**

Once the model has learned the optimal weights, it uses them to make predictions on **new data** (houses it hasn’t seen before). Here’s how:

1. **Take New Input Data**: Suppose you have a new house with the following features:
   * 3 bedrooms
   * 2000 square feet
   * Located in a popular area
   * House color is blue
2. **Multiply Each Feature by Its Weight**: The model will take the weights it learned during training and multiply them by the values of the new house's features:

Predicted Price=(3×50)+(2000×200)+(Location Weight×300)+(House Color Weight×5)\text{Predicted Price} = (3 \times 50) + (2000 \times 200) + (\text{Location Weight} \times 300) + (\text{House Color Weight} \times 5)Predicted Price=(3×50)+(2000×200)+(Location Weight×300)+(House Color Weight×5)

Let’s assume the location is a popular one, so the model assigns it a high weight (300).

The calculation would look like this:

Predicted Price=(3×50)+(2000×200)+(1×300)+(Blue Color Weight×5)\text{Predicted Price} = (3 \times 50) + (2000 \times 200) + (1 \times 300) + (\text{Blue Color Weight} \times 5)Predicted Price=(3×50)+(2000×200)+(1×300)+(Blue Color Weight×5)

Breaking it down:

* 3 bedrooms: 3×50=1503 \times 50 = 1503×50=150
* 2000 square feet: 2000×200=400,0002000 \times 200 = 400,0002000×200=400,000
* Location (popular area): 1×300=3001 \times 300 = 3001×300=300
* House color (blue might not be important): 1×5=51 \times 5 = 51×5=5

Adding these up:

Predicted Price=150+400,000+300+5=400,455\text{Predicted Price} = 150 + 400,000 + 300 + 5 = 400,455Predicted Price=150+400,000+300+5=400,455

So, the predicted price of the house would be **$400,455**.

## FLOW OF EXECUTION IN A MODEL

1)COVERT THE DATA INTO VECTORS

2)FIND DOT PRODUCTS TO KNOW WEIGHTS(APPROX)

3)FINDS THE NEAREST ACTUAL WEIGHTS USING TRAIL AND ERROR METHODS

4)WHEN IT GET NEW INPUT IT WILL CNVERT IT TO VECTOR AND MULTIPLES WITH ITS CORRESPONDING WEIGHTS VECTOR

# NEXT TOPIC

1)CALCULUS

## TODAY LEARNINGS –20/10/2024

1) analysis of python code for pricing in linear regression

2) discussed about machine learning roadmap

## TODAY LEARNINGS –22/10/2024

1) basics of linear algebra

2) basics of vectors – dot product, addition, norms

3) basic of matices - Matrix Addition , Matrix Multiplication , Matrix Transpose , Matrix Inverse

4) topics to learn in depth –a) what is dot product and why it is used?

b) what is Matrix Inverse and why it is used?

c) what is Identity Matrix and why it is used?

5) created a program in python for showing this operations of vectors and matrices

6) added a new file VectorAndMatrices.py and pushed to github

## TODAY LEARNINGS –23/10/2024

1) learned about weights \*\*\*

2) learned about vectors –why and how we do the dot , product of vectors

3) learned about matices – why we do Matrix Multiplication

## TODAY LEARNINGS –24/10/2024

1) learned about derivatives

2)learned about gradient descent

3)learned about loss function

4)learned about bias