**Machine Learning**

Week 1

**What is Machine Learning?**

Teaching computers to learn from patterns from data and make decisions and predictions based on that learning.

Understand trends and patterns in a dataset.

The machine itself is not explicitly programmed for all the scenarios, but it rather learns the patterns and rules from the data and makes relevant predictions and decisions on its own.

**Types:**

1. **Supervised Learning:**

The machine is trained on labeled data, which contains both the inputs and an output. Each instance in the dataset comes with an input and a corresponding output value. The machine learns the trend based on the inputs and outputs both and maps the inputs to the outputs after studying and generalizing the trend of the dataset.

**Process:**

* Provide a dataset containing input-output pairs
* Model Learns from the dataset
* After training, we can give new inputs, and the model will predict the output.

**Real Life Examples:**

1. Email Spam Detection
2. Credit Card Fraud Detection
3. Loan Approval

**Types:**

1. **Classification**

The output label belongs to a category or a class.

**Examples:**

* + Spam/Not Spam
  + Sick/Healthy
  + Red/Blue/Green

**Techniques:**

* Logistic Regression
* Decision Trees
* Random Forest
* K-Nearest Neighbors (KNN)
* Naïve Bayes
* Support Vector Machine (SVM)
* Neural Networks

1. **Regression**

The output label is numerical.

**Examples:**

* House Price= $250,000
* Exam Score= 87
* Temperature= 32°C

**Techniques:**

* Linear Regression
* Ridge/Lasso Regression
* Decision Tree Regressor
* Random Forest Regressor
* Support Vector Regression (SVR)
* Neural Networks

1. **Unsupervised Learning:**

The machine is provided with unlabeled data and hence the data itself doesn’t tell the machine what the correct output is. Each instance of the data only consists of the input features and no corresponding output value. The machine will analyze the structure and pattern within the data and identify similarities and form clusters/groups based on trends that were not obvious.

**Process:**

* Provide a dataset containing only inputs.
* Model analyzes the data to find patterns, similarities or structures.
* It groups/organizes the data into clusters, or lower dimensions
* After learning, the model can assign groupings or highlight patterns in new data.

**Real Life Examples:**

1. Friend Suggestions on Social Media
2. Fraud Detection
3. Music Recommendations

**Types:**

1. **Clustering**

Groups similar data points together.

**Examples:**

* Customer Segmentation
* Social Media Friend Groups
* Grouping Similar Products

**Techniques:**

* + - K-Means
    - DBSCAN
    - Hierarchical Clustering

1. **Dimensionality Reduction**

Reduces the number of input features while retaining important info.

**Examples:**

* + - Visualizing data in 2D/3D
    - Removing noise/redundant features

**Techniques:**

* Principle Component Analysis (PCA)
* T-SNE (for visualization)
* Autoencoders (neural network-based)

1. **Anomaly Detection**

Finds rare/unusual data points.

**Examples:**

* + Fraud Detection
  + Network Security Threats

**Techniques:**

* One-Class SVM
* Isolation SVM
* DBSCAN (can help in detecting outliers)

**Machine Learning Workflow**

Includes the step-by-step process used to build, train, test and deploy a machine learning model.

Transforms raw data into a working model that can make accurate predictions and provide us with insights/trends about the data itself.

**Steps:**

1. **Problem definition**

Clearly identify the problem that needs to be solved.

Classify the type of problem; supervised/unsupervised.

**Example:** Build a model that can predict whether a customer’s loan application should be approved or denied.

1. **Data Collection**

Gather the data that will be used to train the model. Acquire datasets from different sources.

**Example:** Banks database of past loan applications.

1. **Preprocessing**

Clean the anomalies in the data and prepare it for the model.

**Example:** Cater to missing values, noisy data, outliers, remove unnecessary features, splitting the dataset etc.

1. **Model Training**

Train multiple machine learning models based on the input dataset and the required output.

**Example:** Use 80% of the dataset to train a model like Logistic Regression and Random Forest.

1. **Evaluation**

Test the model’s performance using data new data.

**Example:** Usethe remaining 20% of the data and choose the model that yields the best evaluating metrics (Accuracy, Precision, Recall, F1 score, etc.).

1. **Deployment**

Put the model into a real-world system so that it is user friendly and usable.

**Example:** integrate the model into the bank’s loan application software where the user can input their details and the model evaluates their data and provides an outcome.

**Bias Variance Tradeoff**

Concept in machine learning that explains the balance between two sources of error (bias and variance) that affect a model’s ability to generalize to new data.

When we train a machine learning model, we want it to perform well not just on the training data but also on new, unseen data.

**Bias (Too Simple, Underfitting)**

Bias is high when the model is too simple and does not recognize the important patterns in the data. It makes wrong assumptions, resulting in insufficient learning and poor performance on both training and testing data. High error rate on the training data itself.

**Variance (Too Complex, Overfitting)**

Variance is high when the model is too complicated and (sort of) memorizes the training data instead of forming/understand general trends/patterns in it. As a result, it performs well on the training data but poorly on testing/unseen/new data. Low error rate on training data, but high error rate on testing/unseen data.

Concept of tradeoff between bias and variance basically implies that we need our model to generalize well the patterns and insights from the training data so that it works nicely on both training and testing data.

**Real Life Example:**

* **High Bias (Underfitting)**

You use a simple model like a straight line to predict house prices using only one feature, e.g. house size. However, the price of a house depends on many other features as well like the location, condition, number of bedrooms etc.

This model will miss the important patterns and provide us with bad predictions on both training and testing data.

* **High Variance (Overfitting)**

You use a very complex model (e.g. a deep decision tree) that learns every small detail in the training data, including noise.

Such a data would work great on training data but fail to predict prices on new data because it did not learn the general trends.

* **Balanced Model (Optimal)**

A medium-complexity model (e.g. Random Forest or a linear model) that captures the important patterns without ‘memorizing’ all the details.

This will perform well on both the training data as well as the testing data.

**Exploratory Data Analysis (EDA)**

Process of exploring, understanding and summarizing the data before building a machine learning model.

The process of getting to ‘know’ your data.

Data first, Model later.

Cleaning, understanding, and visualizing data.

Helps make informed decisions before modeling.

**It helps us to:**

* Understand the structure,
* Find patters/trends,
* Detect missing values or outliers,
* Decide which features are/are not important.

**Why is it important?**

Primary goal of building a good machine learning model is for it to be accurate and efficient. This cannot be achieved with data that is messy and contains anomalies and errors. EDA will help us to identify all sorts of anomalies, errors, irrelevancies, distributions etc. in the data.

**Steps:**

* **Understanding the Dataset**
* Check rows/Columns in the dataset. Understand their purpose and the information they are providing.
* Identify types of columns, i.e. Categorical/Numerical
* **Summary Statistics**
  + Numerical Data: Mean, Median, Min, Max, Standard Deviation
  + Categorical Data: Unique Values, Frequency (Mode)
* **Checking for Missing/Duplicate Data**
  + Finding and handling number of null values
  + Count and handle duplicates
* **Univariate Analysis**

Analyze one column/feature at a time to understand its distribution, central tendency, spread, unusual values.

**For Numerical Data:**

* + Histogram
  + Box Plot
  + Descriptive Stats: Mean, Median, Min, Max, Standard Deviation

**For Categorical Data:**

* Count Plot
* Bar Chart
* Value Counts
* **Multivariate Analysis**

Analyze the relationship between two or more variables at the same time.

**Numerical vs Numerical:**

* + Scatter Plot
  + Correlation Matrix

**Categorical vs Numerical:**

* + Box Plot
  + Violin Plot
  + Group Means

**Categorical vs Categorical:**

* + Grouped Bar Chart
  + Heatmap
* **Outlier Detection**

Find unusually high or low values.

* + Box Plots (IQR method)
  + Z-Score method
  + Visual inspection

**Variance Inflation Factor (VIF)**

VIF tells us if any of the input variables in a regression model are too like each other, meaning that they are both contributing the same kind/amount of information towards the output variable. When two variables are very similar, strong correlated, the model gets confused about which one is responsible for the change in the output. This makes the results less reliable.

**Real Life Example:**

Suppose we are trying to guess someone’s height using two features, shoe size and leg length. These two are closely related. Using both might confuse the model, because they both contribute the same kind of information. This is also known as multicollinearity.

**How to deal with VIF**

Typically, VIF above and equal to 5 suggests multicollinearity. When it does, we need to start by analyzing all the features with high VIF and see if they are correlated with each other or if they describe similar characteristics. Once we do, here are the common techniques to handle VIF.

* Drop one of the features having high VIF.

Correlation Matrix can be used to decide which variables are similar

* Combine correlated features

An average can be computed as well as applying PCA

* Regularization
  + Lasso Regression (L1)
  + Ridge Regression (L2)

**Gradient Descents**

This is an algorithm used to find the best solution by gradually improving it. In machine learning, it is mostly used to minimize a loss (error) function, finding the best model parameters (like weights in linear regression) that make predictions as accurate as possible.

Gradient descent helps our model learn by tuning its parameters step-by-step to reduce prediction errors. It’s like climbing down a hill to find the lowest point.

**Working:**

Let’s assume we want to train a model with parameters (weight) w and minimize a loss function L(w)

* We start by randomly initializing the weights
* Compute the gradient. This will tell us which direction the slope goes (how much the error will change if we change weights).
* Update the weights

α is the **learning rate**, it controls how big your steps are.

wnew​=wold​− α⋅ ∂L/∂w

* Repeat until the change in loss is very small.

**Types**

* **Batch GD**

This uses the entire dataset to compute gradient. This is more accurate but also slow since it is making use of the entire dataset.

* **Stochastic GD (SGD)**

Uses only one training example at a time to update the weights. It is fast and can get noisy and the convergence is less stable.

* **Mini Batch GD**

Uses a small batch of data for each step. This is balanced, faster than batch and more stable than SGD. It works well with hardware accelerators (e.g. GPUs) and is often a default choice in deep learning.

**Underfitting vs Overfitting**

The goal of nay machine learning model is to learn a pattern from data that works well on both training and testing/unseen data. Sometimes this is not achieved very well.

**Underfitting**

Underfitting is when the model is too simple to learn the underlying pattern in the dataset. It performs poorly on training data and consequently also yields bad results on testing data.

**Causes**

* Model is too simple (linear instead of polynomial)
* Not enough training time
* Features are not very informative/relevant

**Symptoms**

* High training error
* High testing error

**Solution**

Use a more complex model and add better/more features. Train the data for a longer period.

**Overfitting**

Overfitting is when the model is too complex and ‘memorizes’ the training data instead of learning the general pattern. This will yield a very low error on training data but will be unable to accurately predict values on testing/unseen data.

**Causes**

* Model is too complex (too many parameters)
* Not enough data
* No regularization

**Symptoms**

* Low training error
* High testing error

**Solution**

Simplify the model. Use more data and apply regularization.

**Evaluating Metrics**

Evaluating our models in machine learning is crucial because it tells how good or bad our model is and whether its reliable enough for the real world or not.

It answers the fundamental question: ‘Is the model really working?’.

We can’t blindly rely on our model but rather need statistical proof that it is accurate and functions properly. It helps us recognize any overfitting or underfitting and make it easier for us to decide what to do next to make our model better. These metrics also tell us which model performs the best on our dataset. Moreover, evaluation metrics allow us to compare different models objectively, helping us determine which one performs best on our dataset.

**Types**

* **Classification Metrics**

Used when the model predicts categories (e.g. spam/not spam, yes/no)

**Types:**

1. **Confusion Matrix**

A table that shows how many predictions were correct and where the model got confused.

1. **Accuracy**

Proportion of total correct predictions.

Good when the classes are balanced. Bad when the data is imbalanced.

1. **Precision**

Of all the predicted positives, how many were correct.

This is useful when the false positives are costly.

1. **Recall (True Positive Rate/Sensitivity)**

Of all the actual positives, how many did the model catch.

This is good when the false negatives are costly.

1. **F1 Score**

Harmonic mean of precision and recall.

Best when we want a balance between precision and recall. Needs both precision and recall being calculated.

1. **ROC-AUC Score**

Measures models’ ability to distinguish between classes. AUC = 1.0 (Perfect classifier), AUC = 0.5 (Random Guessing)

Works well with probabilistic models.

* **Regression Metrics**

Used when predicting numbers (e.g. price, temperature, score, etc.)

**Types:**

* 1. **Mean Absolute Error (MAE)**

Average of the absolute differences between predicted and actual values.

It is easy to interpret but doesn’t penalize large errors heavily.

* 1. **Mean Squared Error**

Average of the squared differences between predicted and actual.

This penalizes larger errors more but is harder to interpret due to squaring.

* 1. **Root Mean Squared Error**

Square root MSE.

It is more interpretable that MSE and penalizes large errors.

* 1. **R-squared**

How well the model explains the variability in the output.

Tells us how useful our model is.

| **Task** | **Recommended Metrics** |
| --- | --- |
| **Balanced Classification** | Accuracy, F1 Score |
| **Imbalanced Classification** | Precision, Recall, F1, ROC-AUC |
| **Regression (general)** | MAE, RMSE, R² |
| **Regression (outliers’ matter)** | RMSE |
| **Regression (robust to outliers)** | MAE |