***Unsupervised Learning***

*Definition:*

* Unsupervised Learning is a type of machine learning where the model learns patterns without labeled data. The model discovers the structure in the input data on its own.
* The goal is to explore the data and find hidden patterns, groupings, or relationships on its own without explicit guidance or prior knowledge of outcomes.

*Types of Unsupervised Learning:*

* **Pattern Mining :**
* Pattern Mining is about discovering interesting, frequent, or significant patterns in large datasets. It is widely used in market analysis, web usage mining, bioinformatics, and more.
* It works by identifying similarities, differences, or dependencies, or dependences among structures or instances.
* Frequent Itemsets:
* Items that occur together frequently in transactions.
* Example: People who buy bread and butter often buy milk too.
* Support, Confidence, Lift
* Support: How frequently an itemset appears in the dataset.
* Confidence: How often the rule has been found to be true.
* Lift: Measures how much more likely the consequent is given the antecedent.
* Commonly used algorithms are Apriori and FP – Growth.
* Apriori: Uses a bottom-up approach.
* Starts with single items and finds those that are frequent (based on a support threshold).
* Then creates larger itemsets by combining the frequent ones.
* Prunes the itemsets that have any infrequent subset.
* Repeats until no more frequent subsets can be found.
* Association Rules are generated based on the frequent subsets found.
* **Advantages & Disadvantages:**
* Efficiency: The apriori property allows the algorithm to efficiently prune infrequent itemsets, reducing the computational costs.
* Simplicity: The algorithm is relatively easy to understand and implement.
* The algorithm can be computationally expensive for very large datasets.
* Needs multiple scans of the dataset.
* Generates a lot of candidate itemsets, many of which may be useless.
* FP-Growth: This avoids generating candidates like apriori. Instead, it builds a compact tree structure:
* First scans the data to find frequent items.
* Builds an FP-Tree: Transactions are added to the prefix tree, shared paths are merged.
* Mines the tree recursively: Breaks it into smaller conditional trees and mines those.
* **Advantages & Disadvantages:**
* Much faster than apriori on large datasets.
* No candidate generation, fewer scans.
* More scalable for real-world applications.
* Tree structure can be memory-intensive if the dataset is huge and has overlap in items.
* More complex to implement and understand at first.

*Examples:*

A screenshot of a computer

AI-generated content may be incorrect.

A graph with blue and white bars

AI-generated content may be incorrect.

* Applied apriori to a dataset consisting of 9836 rows of data to discover frequent item combinations in grocery transactions. The top association rules (based on lift) revealed meaningful product affinities:
* Curd 🡪 Yogurt, Whole Milk
* Customers buying curd tend to also purchase yogurt and whole milk.
* Lift = 3.37 which means the combination is 3.37x more likely to occur together by random chance.
* (Citrus Fruit, Other Vegetables) 🡪 Root Vegetables
* Indicates customers shopping for healthy or fresh produce also frequently pick up root vegetables.
* Beef 🡪 Root Vegetables
* This rule may reflect a cooking preference or meal-planning pattern.
* Lift Values > 3 across these rules indicate strong positive correlations among items – valuable for product bundling or layout strategy in stores.

A screenshot of a computer

AI-generated content may be incorrect.

A graph of a graph with text

AI-generated content may be incorrect.

* FP-Growth was run on a larger transaction dataset (38,000 values approx.) with higher diversity. Initially, using default thresholds produced no strong rules. After reducing minimum confidence to 0.1, the top rules (by confidence) showed consistent buying patterns involving whole milk:
* Yogurt 🡪 Whole Milk
* Rolls/Buns 🡪 Whole Milk
* Other Vegetables 🡪 Whole Milk
* Soda 🡪 Whole Milk
* While confidence values ranged around 12-13%, the lift values were < 1, suggesting these associations are not statistically stronger than random co-occurrences. These are still useful for understanding popular product combinations, even if not highly predictive.
* Support measures the frequency of an item or itemset, confidence indicates the likelihood of a rule’s consequent given its antecedent, and lift assesses the strength of the association between items, comparing observed confidence to expected confidence.
* **Clustering:** It is the task of grouping a set of data points into clusters, such that:
* Points within a cluster are similar to each other.
* Points in different clusters are dissimilar.
* There are no labels – the algorithm tries to find structure in the data by itself.
* It is a technique that groups unlabeled data points based on similarity, aiming to uncover hidden patterns and structures without prior knowledge of the data’s meaning.
* **Types:**
* Centroid-Based Clustering: Clusters are defined by a central point (Centroid)
* It groups data into clusters by identifying a central point (centroid) for each cluster and minimizing the distance between the data points and their centroid.
* K-Means Clustering:
* Choose the number of clusters k.
* Randomly initialize k centroids.
* Assign each data point to the nearest centroid.
* Recalculate centroids as the mean of all assigned points.
* Repeat steps 3 & 4 until centroids stop changing (or a set number of iterations is reached).
* **Advantages & Disadvantages:**
* Fast and efficient on large datasets.
* Easy to interpret and implement.
* Works well when clusters are aspherical and equally sized.
* You have to specify k manually.
* Sensitive to outliers and initial centroid positions.
* Struggles with non-globular clusters (like crescent or spiral shapes).
* Can get stuck in local minima – results may vary depending on initialization

A screenshot of a document

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A graph of a number of clusters

AI-generated content may be incorrect.

* **Observations:**
* Dataset Insights: A dataset with 53,503 records with 20 columns.
* Useful Metric Columns:
* Age
* Income Level
* Coverage Amount
* Premium Amount
* The Elbow plot is most noticeable at k = 4, X-axis = number of clusters and Y-axis – inertia (within-cluster sum of squares) 🡪 lower the better.
* Goal is to find the “elbow” – the point after which the rate of decrease sharply slows down.
* From the plot we can see that, from k = 1 to k = 4 inertia drops sharply which implies that the model is getting much better.
* After k = 4, the curve flattens – inertia still drops, but not significantly.
* The flattening is the “elbow” – it tells that adding more clusters beyond this point gives minimal gain in compactness.
* Not always exact, so must check visually.

A diagram of a number of colored dots

AI-generated content may be incorrect.

* Cluster customers based on financial features using the k-means algorithm with k = 4, as suggested by the Elbow Method.
* Features Used:
* Income level
* Coverage Amount
* Plot shows customers segmented into 4 distinct clusters based on income and coverage patterns.
* Color-coded clusters suggest the level of structure:
* High-income clusters with high coverage fall into one cluster
* Mid-range and low-income customers are grouped based on similar insurance behavior.
* While clusters overlap a bit, the segmentation reflects patterns In financial risk or customer value.
* Density-Based Clustering (DBSCAN): It groups data points by density.
* Works by partitioning data into clusters based on their distance to other points.
* Rather than partitioning data into fixed shapes (like K-Means does), DBSCAN groups dense regions of points together and treats sparse regions as noise or outliers.
* **Key Concepts:**
* Core Point: A point at least min\_samples neighbors within distance eps.
* Border Point: A point that’s not a core but lies within eps of a core point.
* Noise Point: Not a core, not reachable from a core 🡪 treated as outlier.
* Eps: Max Distance for a point to be considered a neighbor.
* Min\_samples: Min number of points to form a dense region (a cluster).
* **Advantages & Disadvantages:**
* No need to specify number of clusters in advance.
* Can find arbitrarily shaped clusters (not just circles).
* Identifies outliers as noise.
* Choosing good eps and min\_samples is tricky.
* Struggles with varying density in different parts of the data.
* Sensitive to scaling, so feature normalization is important.

A blue square with orange dots

AI-generated content may be incorrect.

* Number of Clusters found: 47
* Noise Points: 52,824
* Didn’t work as well with this dataset as 99% of the values were considered noise, might be due to dataset being spread evenly spread or high-dimensional for DBSCAN to find large clusters.
* Most of the data is too sparse to be grouped.
* DBSCAN is great for finding small groups, but it needs data with uneven density and clear gaps between clusters.
* Hierarchical Clustering: It groups data points into a hierarchy of clusters, represented by a tree-like structure called a dendogram , allowing for exploration of data relationships at multiple levels of granularity.
* Each data point starts in its own cluster.
* Clusters are repeatedly merged based on closeness until only one big cluster remains.
* **Two Approaches**: Agglomerative and Divisive.
* **Agglomerative**: This approach starts by treating each data point as its own cluster and then iteratively merges the closest clusters until a single cluster containing all data points is formed.
* **Divisive**: This approach starts with all data points in a single cluster and then recursively splits the clusters until each data point is its own cluster.
* **Key Terms:**
* Linkage: How distance between clusters is measured.
* Common Methods:
* Single (Closest Points)
* Complete (Farthest Points)
* Average (Mean of Distance)
* Ward (Minimizes variance within clusters - best for Euclidean distance)
* Dendrogram: A tree diagram that shows how clusters merge at each step.
* **Advantages & Disadvantages:**
* No need to pre-define the number of clusters.
* Can visually inspect and decide the number of clusters from the dendrogram.
* Good for small to medium datasets.
* Computationally expensive for large datasets.
* Sensitive to noise and scaling.
* Once merged/split, cannot undo decisions (not flexible like K-Means).

A graph showing a cluster of data

AI-generated content may be incorrect.

* Agglomerative Approach
* Same dataset as above, but had to set the sample size to 500 as computing 53,000 is too much!
* The dendrogram shows how clusters are merged step-by-step based on similarity.
* Cutting the dendrogram at a moderate height resulted in 4 primary clusters.
* This hierarchical structure can be helpful to understand how clusters relate to each other.
* Each Sampled Cluster us now labeled with a Hierarchical\_Cluster (0 to 3).
* **Principal Component Analysis (PCA)**
* PCA transforms a high-dimensional dataset into smaller number of dimensions (called principal components) that capture the most variance in the data.
* Used To:
* Reduce complexity while preserving patterns.
* To visualize high-dimensional data in 2D or 3D.
* As a preprocessing step before clustering or modeling.
* How it works?
* Finds the directions (axes) where the data varies the most.
* Rotates the data onto these new axes (Principal Components).
* Drops less informative components to reduce dimensions.

A blue dot diagram with white grid

AI-generated content may be incorrect.

* The objective was to reduce dimensionality of customer data from 4 numerical features to 2 principal components for visualization.
* Features Used:
* Age
* Income Level
* Coverage Amount
* Premium Amount
* The variance tells us that:
* PC1 caputres ~25.5% of the total variance.
* PC2 captures ~25.1%
* Combined: ~50.6% of the original data’s information is preserved in just 2 dimensions.
* Each point is a customer.
* The spread across both components shows how customers differ based on linear combinations of the original features.
* **Anomaly Detection:**
* It is the process of identifying rare, unusual, or suspicious data points that don’t conform to the general pattern of the dataset.
* **Types:**
* Point Anomaly: Single instance far from others.
* Contextual: Anomaly depending on the context.
* Collective: A group of related anomalies forming a pattern.
* Common Techniques:
* Z-Score
* Isolation Forest
* Local Outlier Factor
* One-Class SVM
* Isolation Forest:
* A tree-based method for unsupervised anomaly detection.
* Works by randomly partitioning the data into trees.
* Anomalies are easier to isolate, so they require fewer splits.
* Returns an anomaly score for each data point.
* Efficient and scalable for large datasets.
* Key Parameter:
* Contamination: Proportion of expected anomalies in the data.

A blue and red graph

AI-generated content may be incorrect.

* Normal Points: 52,967
* Anomalies: 536 (Contamination = 0.01% setting).
* Each dot is a customer.
* Blue = normal, Red = Anomaly
* Most Anomalies appear:
* Around a very low or very high-income levels.
* At extremes of coverage amounts.
* In less dense regions of the plot, as expected.
* This aligns with the idea that anomalies are “easier to isolate” – they exist in areas where fewer points lie.
* Local Outlier Factor: LOF is a density-based anomaly detection method. It works by comparing the local density of a data point to its neighbors.
* If a point is in a much sparser region than its neighbors, it’s considered an outlier.
* How it works:
* Neighborhood: For each point, LOF looks at its k nearest neighbors.
* Local Density Estimate: It calculates how close the point is to its neighbors – i.e., local reachability density.
* LOF Score: If the point has a much lower density than its neighbors 🡪 high LOF score 🡪 anomaly.
* **Key Parameters:**
* N\_neighbors: Number of neighbors to consider (default = 20)
* Contamination: Proportion of expected outliers
* Novelty: Only used for unseen data
* **Advantages & Disadvantages:**
* Great for local anomalies.
* Works well for clusters of varying densities.
* Can be sensitive to n\_neighbors.
* Not ideal for very large datasets.

A blue and red graph

AI-generated content may be incorrect.

* Normal (1): 52,967
* Anomalies (-1): 536
* LOF identified points with lower local density compared to their neighbors.
* Red dots appear around:
* Sparse regions
* Edges of Dense Clusters
* Possibly near extreme income or coverage values.
* LOF is more locally sensitive than Isolation Forest – meaning it can flag points that aren’t globally unusual but look weird compared to their neighbors.
* Local Outlier Factor detects anomalies by comparing the local density of each point to its neighbors. Points in sparser regions than their surroundings are flagged as outliers.
* It’s great for catching subtle, local anomalies – especially in clustered data.
* **Collaborative Filtering:**
* Collaborative filtering is a technique used in recommender systems to suggest items to a user based on preferences of similar users or similar items.
* **Types:**
* User-Based Collaborative Filtering:
* Recommends items that similar users liked.
* Example: “People who are similar to you also liked this movie.”
* Steps:
* Find users similar to the target user.
* Recommend items those users liked, but the target user hasn’t seen yet.
* Item-Based Collaborative Filtering:
* Recommends items that are similar to the ones the user already liked.
* Example: Because you liked Movie A, here’s another movie that is similar.”
* Steps:
* Find items similar to those that the user liked.
* Recommend these similar items.
* How it works?
* Create a user\_matrix
* (rows=users, columns=items, values=ratings or interactions)
* Compute Similarities
* User-based: Similarity between users (e.g., Cosine, Pearson)
* Item-based: Similarity between items.
* Use neighbors to predict unknown values in the matrix.

A close-up of a message

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

* Recommended movies to a user based on ratings from similar users.
* Recommended movies similar to what a user has already liked.

|  |  |  |
| --- | --- | --- |
| *Feature* | *User-Based CF* | *Item-Based CF* |
| Works well when | Users have a stable preference | Item relationships are stronger |
| Slower If | Many Users | Many Items |
| Scalable? | Less than item-based | More Scalable (fewer items) |
| Use Case Example | Netflix: similar users | Amazon: “similar items” |

* *Real Life Use-Cases for Everything:*
* **Pattern Mining:**
* Market Basket Analysis
* E-Commerce: Used for product bundling and cross-selling recommendations.
* Healthcare: Discover frequent symptom or medication combinations in patient data.
* **Clustering:**
* Customer Segmentation: Grouping customers based on purchase history, behavior, or demographics.
* Image Segmentation: Dividing an image into meaningful regions for object recognition or medical imaging.
* Document/News Clustering: Automatically organizing articles or posts by topic or item.
* **Principal Component Analysis:**
* Data Visualization: Projecting high-dimensional data into 2D or 3D for easier visual analysis.
* Noise Reduction: Filtering out unimportant features from data like images or signals.
* Model Optimization:Reducing dimensions to speed up training and prevent overfitting.
* **Anomaly Detection:**
* Fraud Detection: Detecting unusual credit card transactions or user activity.
* Network Security: Identifying suspicious behavior, like potential intrusions or malware.
* Manufacturing: Catching faulty or irregular machine behavior on production lines.
* **Collaborative Filtering:**
* Recommender Systems: Platforms like Netflix, Spotify, and YouTube suggest content based on your preferences or similar users/items.
* E-Commerce: Amazon shows “Customers who bought this also bought…” using item-based CF.
* Online Learning Platforms:Recommending courses, videos, pr resources based on similar learners’ interests.