**INTERVIEW QUESTIONS AND ANSWERS:**

**Logistic regression Assignment:**

**1. What is the difference between precision and recall?**

Answer:

Precision and recall are performance metrics used to evaluate classification models, especially in imbalanced datasets:

* Precision measures how many of the predicted positive instances are actually positive.

**Precision=True Positives / True Positives + False Positives**

High precision means fewer false positives.

* Recall (also known as sensitivity or true positive rate) measures how many of the actual positive instances were correctly identified.

**Recall=True Positives /True Positives + False Negatives ​**

* + High recall means fewer false negatives.

Example:  
In a medical test for detecting cancer:

* Precision is important to ensure that people diagnosed as positive truly have cancer (avoiding false alarms).
* Recall is critical to catch as many actual cancer cases as possible (avoiding missed detections).

**2. What is cross-validation, and why is it important in binary classification?**

Answer:

Cross-validation is a resampling technique used to assess the generalizability of a machine learning model by splitting the dataset into multiple train-test subsets.

* The most common method is k-fold cross-validation, where the data is divided into *k* parts. The model is trained on *k-1* parts and tested on the remaining part. This is repeated *k* times, and the results are averaged.
* Importance in binary classification:
  + Helps to detect overfitting or underfitting.
  + Ensures that model performance is not biased by a particular train-test split.
  + Provides a more robust estimate of accuracy, precision, recall, and other metrics, especially on imbalanced datasets.
  + Useful when the dataset is small, allowing every data point to be used for both training and testing.

**Recommendation System:**

Q1. What is the difference between precision and recall?

* Precision is the proportion of true positive predictions among all positive predictions made by the model.  
  Precision = TP / (TP + FP)
* Recall is the proportion of true positive predictions among all actual positives.  
  Recall = TP / (TP + FN)
* Precision is useful when false positives are costly.
* Recall is critical when false negatives are more severe.

**Q2. What is cross-validation, and why is it important in binary classification?**

* Cross-validation is a technique where the data is split into multiple folds, and the model is trained and validated on different combinations of these folds.
* It helps assess model performance more reliably by reducing variance caused by a single train-test split.
* In binary classification, it ensures the model generalizes well to unseen data and helps avoid overfitting.

**Association Rule Mining**:

**1. What is Lift and why is it important in Association Rules?**

**Lift** is a metric used to evaluate the performance of an association rule in data mining. It tells us how much more likely the **consequent (Y)** is to occur **given the antecedent (X)**, compared to its usual occurrence.

**Formula: Lift(X⇒Y)= P(X∩Y)​ / P(X).P(Y)**

**Interpretation:**

* **Lift = 1**: No association between X and Y (they are independent).
* **Lift > 1**: Positive association – X and Y occur together more than expected.
* **Lift < 1**: Negative association – X and Y occur together less than expected.

**Why it’s important:**

Lift helps identify **truly interesting associations**. While confidence alone may be high simply because the consequent is common, lift filters out such cases and shows the **strength of dependency** between items.

**2. What is Support and Confidence? How do you calculate them?**

**Support:**

Support measures how frequently an itemset appears in the dataset.

**Support(X) = Number of transactions containing X / Total number of transactions**

Example: If 2 out of 10 transactions contain {milk, bread}, then:

**Support(milk,bread)= 2/10 =0.2**

**Confidence:**

Confidence measures the **reliability** of the inference made by a rule.

**Confidence(X⇒Y)=Support(X∪Y)/Support(X)**

**Confidence(milk⇒bread)= 2/4 ​=0.5**

**3. What are some limitations or challenges of Association Rule Mining?**

* Large number of rules: Can generate thousands of rules, making it hard to interpret or select relevant ones.
* Sparse data: Many item combinations may be rare, leading to low support.
* Lack of semantic meaning: The rules are statistical and may not always make business sense.
* Computationally expensive: Mining frequent itemsets can be very slow on large datasets.
* Confusing metrics: High confidence does not always mean a strong relationship (lift is often needed).
* Overfitting risk: Rules might capture noise or spurious patterns.

**Decision Trees and Encoding Techniques:**

**1. What are some common hyperparameters of decision tree models, and how do they affect the model's performance?**

Here are the key hyperparameters for decision trees (e.g., in sklearn.tree.DecisionTreeClassifier):

| **Hyperparameter** | **Description** | **Impact on Model** |
| --- | --- | --- |
| max\_depth | Maximum depth of the tree | Controls **overfitting**. Shallow trees may underfit, deeper trees may overfit |
| min\_samples\_split | Minimum number of samples required to split an internal node | Higher values reduce overfitting by requiring more samples per split |
| min\_samples\_leaf | Minimum number of samples required to be at a leaf node | Prevents overfitting by ensuring a minimum leaf size |
| max\_features | Number of features to consider when looking for the best split | Helps **reduce variance** and improve generalization |
| criterion | Function to measure the quality of a split ("gini" or "entropy") | Affects how splits are made; both work similarly but may yield different trees |
| splitter | Strategy to choose the split at each node ("best" vs "random") | "best" is deterministic; "random" introduces randomness (good for ensembles) |

**Tuning these hyperparameters helps in balancing bias and variance**, controlling tree complexity and improving generalization on test data.

**2. What is the difference between Label Encoding and One-Hot Encoding?**

| **Feature** | **Label Encoding** | **One-Hot Encoding** |
| --- | --- | --- |
| **Definition** | Assigns a unique integer to each category | Converts categories into binary columns |
| **Example** | Red=0, Green=1, Blue=2 | Red: [1,0,0], Green: [0,1,0], Blue: [0,0,1] |
| **Use Case** | Ordinal data (e.g., Education level) | Nominal data (e.g., City names) |
| **Risk** | Can mislead model into thinking categories have order | Avoids false ordinal relationships |
| **Resulting Columns** | 1 column | n columns (n = number of unique categories) |

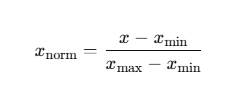
**Label Encoding is simple** but should only be used when categories have an **inherent order**.  
 **One-Hot Encoding is preferred for nominal categorical variables** to prevent misleading the model.

**MLR**

**1. What is Normalization & Standardization and how is it helpful?**

**Normalization**

* **Definition:** Rescaling data to a fixed range, typically [0, 1].
* **Formula:**



* **When to use:**
  + When the features do **not follow a Gaussian (normal) distribution**.
  + Especially useful for **distance-based algorithms** like KNN, K-Means, SVM with RBF kernel.

**Standardization**

* **Definition:** Transforms data to have a **mean of 0** and **standard deviation of 1**.
* **Formula:**



* **When to use:**
  + When features have different units/scales but are approximately **normally distributed**.
  + Preferred in **linear models**, **logistic regression**, **PCA**, etc.

**Why It's Helpful:**

* Avoids **feature dominance** due to scale differences.
* Improves convergence speed and stability for **gradient-based models**.
* Makes coefficients and interpretations **more meaningful**.

**2. What techniques can be used to address multicollinearity in multiple linear regression?**

**Techniques to address multicollinearity:**

1. **Remove Highly Correlated Predictors**
   * Use **correlation matrix** or **VIF** to identify features with high collinearity (VIF > 5 or10).
   * Drop one of the correlated features.
2. **Principal Component Analysis (PCA)**
   * Transforms correlated features into a smaller set of **uncorrelated components**.
   * Useful when you don’t want to lose information but reduce dimensionality.
3. **Combine Features**
   * Sometimes combining correlated features into a single **composite variable** (e.g., sum or average) can help.
4. **Regularization Techniques**
   * **Ridge Regression (L2 Regularization):** Shrinks coefficients, useful for handling multicollinearity.
   * **Lasso Regression (L1 Regularization):** Can **shrink some coefficients to zero**, thus performing feature selection.
   * **ElasticNet:** Combination of L1 and L2 penalties.
5. **Centering the Data**
   * Subtracting the mean from features reduces multicollinearity due to interaction terms or polynomial features.

**Summary**

| **Concept** | **Normalization** | **Standardization** | **Multicollinearity Fixes** |
| --- | --- | --- | --- |
| Use case | Range scaling | Gaussian-like data | Linear model stability |
| Technique | Min-Max Scaling | Z-score | Remove vars, PCA, Ridge/Lasso |
| Helps in | KNN, SVM, KMeans | Regression, PCA | Better model interpretability |