I have a final practical examination on our Machine leaarning subject. And I have just 1-2 hour to prepare for it.

I have attached all the assignments that could be asked in the practical.

Give me all possible questions that could be asked by the professor during the viva with all the answers I should give with keyword highlighted. And I want it so that each answer explains the concept properly like a story like explaining the process of it and stuff. But also give me answers in a form that I can say to the proffessor with also keywords highlighted

Give me all the questioons and answers that could be asked and anything that is related to it.

I do have much time so this would be the only resource I would have time to read about. So give me a proper guide that if I read 2-3 times I'll get full marks.

Also make sure I'll only remember 10-20% of what I would read, so give me enough so that there will be no problem.

Also tell me main functions that will be used for each assignmets and why we use them with all the things about it that could be asked.

And tell me about all possible questions with also all possible related concepts about these assignments that could be asked. Give me everything

Based on the files you've uploaded, I've prepared a **comprehensive viva preparation guide** covering all your ML assignments. Since you only have 1-2 hours and will retain 10-20%, I've made this extra thorough. I've highlighted **keywords** and explained concepts like stories for the professor.

COMPLETE ML VIVA PREPARATION GUIDE

□ ASSIGNMENT 1: LINEAR REGRESSION

Q1: What is Linear Regression?

Answer: "Linear Regression is like **drawing the best-fitting straight line** through scattered data points. Imagine you have data about **study hours and exam scores**. Linear Regression helps us predict scores based on hours studied by finding the **line** that minimizes the distance between all points and itself. It uses the formula `y = mx + c` where `m` is the **slope** (how much Y changes when X increases) and `c` is the **intercept** (where the line crosses the Y-axis). It's a **supervised learning** algorithm for **regression** (predicting continuous values)."

Keywords: Supervised learning, regression, best-fit line, slope, intercept, minimize error

Q2: What are the evaluation metrics for Linear Regression?

Answer: "We use several metrics to check how good our model is:

1. **MAE (Mean Absolute Error)** - Average of all absolute errors. **Simple** to understand.
2. **MSE (Mean Squared Error)** - Average of **squared errors**. **Penalizes larger errors** more.
3. **RMSE (Root Mean Squared Error)** - Square root of MSE. Same **units** as target variable.
4. **R^2 Score (R-squared)** - Shows **how much variance** is explained by the model. **Range 0 to 1**, closer to 1 is better."

Keywords: MAE, MSE, RMSE, R-squared, variance, error metrics

Q3: What is Multiple Linear Regression?

Answer: "When we have **multiple independent variables** (features) predicting one dependent variable, it's **Multiple Linear Regression**. For example, predicting **house price** based on **area, bedrooms, location**. Formula becomes `y = m1*x1 + m2*x2 + ... + c`. We're fitting a **plane** or **hyperplane** in higher dimensions instead of a line."

Keywords: Multiple features, hyperplane, multiple independent variables

Q4: What's the difference between Linear and Logistic Regression?

Answer: "**Linear Regression** predicts **continuous values** (like price, temperature), while **Logistic Regression** is for **classification** (yes/no, spam/not spam). Logistic uses a **sigmoid function** to convert output to **probability between 0 and 1**, then we apply a **threshold (usually 0.5)** to classify."

Keywords: Continuous vs categorical, sigmoid function, classification, probability

Q5: What is Lasso Regression?

Answer: "**Lasso (L1 Regularization)** adds a **penalty term** to prevent overfitting. It can **shrink some coefficients to exactly zero**, effectively doing **feature selection** by removing unimportant features. It's useful when we have **many features** and want to identify the **most important ones**."

Keywords: L1 regularization, feature selection, penalty term, overfitting prevention

Main Functions Used:

- `LinearRegression()` - Creates the model
- `fit(X, y)` - Trains the model on data
- `predict(X)` - Makes predictions
- `score(X, y)` - Returns R^2 score
- `mean_absolute_error()`, `mean_squared_error()` - Calculate errors
- `train_test_split()` - Splits data into training/testing sets

Why: These functions simplify the entire ML pipeline from training to evaluation.

□ ASSIGNMENT 2: k-NN AND NAIVE BAYES

Q1: Explain k-Nearest Neighbors (k-NN)?

Answer: "k-NN is like asking **'Who are my k closest neighbors?'** and taking a **majority vote**. Imagine you're at a party and want to know if you'll like a movie. You ask the **k people closest to you** (similar tastes), and if most say 'yes', you predict 'yes'. It's a **lazy learner** because it doesn't learn during training - it memorizes all data and works during prediction. We use **distance metrics** like **Euclidean distance** to find nearest neighbors."

Keywords: Lazy learning, distance metrics, Euclidean distance, majority voting, k neighbors

Q2: How do you choose the value of k?

Answer: "Choosing k is critical. **Too small k (like k=1)** makes the model **sensitive to noise** (overfitting). **Too large k** makes it **oversimplified** (underfitting). We typically use **odd numbers** to avoid ties in binary classification. Best practice is to try **multiple values** and use **cross-validation** to find optimal k. Usually **k = √n** (square root of training samples) is a good starting point."

Keywords: Overfitting, underfitting, cross-validation, hyperparameter tuning

Q3: What is Naive Bayes?

Answer: "Naive Bayes is based on **Bayes' Theorem** from probability. It's called 'naive' because it assumes **features are independent** (which is rarely true, but works well anyway!). Think of it like this: To predict if an email is spam, we calculate **P(Spam|Words)** - the probability of spam given the words. It **multiplies probabilities** of each word appearing in spam emails. It's **very fast** and works great for **text classification**."

Keywords: Bayes' Theorem, conditional probability, feature independence, text classification, fast algorithm

Q4: Types of Naive Bayes?

Answer: "Three main types:

1. **Gaussian NB** - For **continuous data** (features follow normal distribution)
2. **Multinomial NB** - For **discrete counts** (word frequencies in text)
3. **Bernoulli NB** - For **binary features** (word present/absent)"

Keywords: Gaussian, Multinomial, Bernoulli, feature types

Q5: Compare k-NN and Naive Bayes?

Answer: "**k-NN** is **instance-based** (stores all training data), **slow at prediction** but needs **no training time**. **Naive Bayes** is **model-based** (learns probability distributions), **very fast at prediction**, and handles **high-dimensional data** better. k-NN needs **feature scaling**, Naive Bayes doesn't. k-NN works for **any type of data**, Naive Bayes assumes **feature independence**."

Keywords: Instance-based vs model-based, prediction speed, training speed, scalability

Main Functions Used:

- `KNeighborsClassifier(n_neighbors=k)` - Creates k-NN model
- `GaussianNB()`, `MultinomialNB()` - Naive Bayes variants
- `fit(X, y)` - Trains the model
- `predict(X)` - Makes predictions
- `accuracy_score()` - Calculates accuracy
- `confusion_matrix()` - Shows classification results
- `StandardScaler()` - Scales features (important for k-NN)

Why: k-NN needs scaling because it uses distances. Naive Bayes calculates probabilities so scaling isn't needed.

ASSIGNMENT 3: DECISION TREES

Q1: What is a Decision Tree?

Answer: "A Decision Tree is like a **flowchart of questions**. Imagine deciding whether to play tennis: First question - 'Is weather sunny?' If yes, next question - 'Is humidity high?' and so on. Each **internal node** is a question/test on a feature, each **branch** is the answer, and each **leaf node** is the final prediction. It uses **if-else rules** that are **easy to understand and visualize**. The tree learns by **recursively splitting** the data to create the **most pure subsets** (where most samples belong to one class)."

Keywords: Flowchart, if-else rules, internal nodes, leaf nodes, recursive splitting, purity

Q2: How does a Decision Tree decide where to split?

Answer: "The tree tries **all possible splits** and picks the one that creates the **purest subgroups**. For **classification**, it uses:

1. **Information Gain** - Based on **entropy** (measure of impurity). Split that gives **highest information gain** wins.
2. **Gini Impurity** - Measures how often a random sample would be **incorrectly classified**.

For **regression**, it uses **variance reduction** - split that **reduces variance** the most."

Keywords: Information gain, entropy, Gini impurity, variance reduction, splitting criteria

Q3: What is overfitting in Decision Trees? How to prevent it?

Answer: "Decision Trees can become **too complex** and memorize the training data instead of learning patterns - this is **overfitting**. The tree might grow **very deep** with many specific rules. To prevent:

1. **Pruning** - Cut off branches that don't help much
2. **Max depth** - Limit how deep the tree can grow
3. **Min samples split** - Need minimum samples to create a split
4. **Min samples leaf** - Need minimum samples in leaf nodes"

Keywords: Overfitting, memorization, pruning, max_depth, hyperparameters, stopping criteria

Q4: Classification vs Regression Trees?

Answer: "**Classification Trees** predict **categories** (yes/no, species type). Leaf nodes contain **class labels**. Use **Gini or Entropy** for splitting.

Regression Trees predict **continuous numbers** (price, temperature). Leaf nodes contain **mean/median values**. Use **variance reduction** for splitting.

Same tree structure, different outputs and splitting criteria!"

Keywords: Categorical vs continuous, class labels, mean values, splitting criteria

Q5: What is k-Fold Cross Validation?

Answer: "Instead of just one train-test split, we **divide data into k parts (folds)**. We train **k times**, each time using **k-1 folds for training** and **1 fold for testing**, rotating which fold is the test set. Finally, we **average the k accuracy scores**. This gives a **more reliable estimate** of model performance and **reduces variance** in evaluation. Common value is **k=5 or k=10**."

Keywords: k folds, rotation, average performance, reliable estimate, reduced variance

Q6: Advantages and Disadvantages of Decision Trees?

Answer:
"**Advantages:**

- **Easy to understand and visualize**
- **No need** for feature scaling/normalization
- **Handles missing values** well
- Works with both **numerical and categorical** data

Disadvantages:

- **Prone to overfitting**
- **Small changes** in data can cause large changes in tree structure (**unstable**)
- Can create **biased trees** if some classes dominate
- **Greedy algorithm** - doesn't guarantee global optimal tree"

Keywords: Interpretability, no preprocessing, instability, greedy approach, overfitting

Main Functions Used:

- `DecisionTreeClassifier()`, `DecisionTreeRegressor()` - Creates model
- `fit(X, y)` - Builds the tree
- `predict(X)` - Makes predictions
- `max_depth`, `min_samples_split`, `min_samples_leaf` - Hyperparameters to control tree growth
- `feature_importances_` - Shows which features are most important
- `plot_tree()` - Visualizes the tree
- `cross_val_score(cv=k)` - Performs k-fold cross validation

Why: These parameters control overfitting. `feature_importances_` helps understand what drives predictions. Cross-validation gives robust performance estimates.

ASSIGNMENT 4: SVM (SUPPORT VECTOR MACHINES)

Q1: What is SVM?

Answer: "SVM finds the **best boundary** (called **hyperplane**) to separate different classes. Imagine two groups of points - SVM finds the line/plane that **maximizes the margin** (distance) between the groups. The points closest to this boundary are called **support vectors** - they 'support' the decision boundary. Only these support vectors matter, not all training points! SVM works by **maximizing the minimum distance** from the boundary to any point."

Keywords: Hyperplane, maximum margin, support vectors, optimal boundary, decision boundary

Q2: What is the kernel trick in SVM?

Answer: "When data isn't **linearly separable** (can't draw a straight line to separate), we use the **kernel trick**. It's like **lifting the data into higher dimensions** where it becomes separable, without actually computing those high-dimensional coordinates (which would be expensive!).

Common kernels:

1. **Linear** - Just a straight line (fast, for linearly separable data)
2. **RBF (Radial Basis Function)** - Can create **circular** boundaries (most common, works for complex data)
3. **Polynomial** - Creates **curved** boundaries
4. **Sigmoid** - Similar to neural networks"

Keywords: Kernel trick, non-linear separation, RBF kernel, higher dimensions, polynomial, sigmoid

Q3: Compare SVM with Decision Tree?

Answer: "**SVM:**

- Finds **optimal boundary** with maximum margin
- Better for **high-dimensional data**
- Uses **kernel trick** for non-linear problems
- **Less interpretable** (black box)
- **Slower to train** on large datasets
- **Less prone to overfitting**

Decision Tree:

- Creates **if-else rules**

- **Easy to visualize and explain**
- Can **overfit** easily
- **Faster** to train
- Works naturally with **categorical data**
- **Doesn't need feature scaling**

Choose SVM when you want **accuracy** and have **complex data**. Choose Decision Tree when you need **interpretability**."

Keywords: Optimal boundary, black box, interpretability, overfitting, feature scaling, high-dimensional

Q4: What are hyperparameters in SVM?

Answer: "Key hyperparameters:

1. **C (Regularization)** - Controls **trade-off** between **smooth boundary** and **classifying training points correctly**. Large C = **fewer misclassifications** but might **overfit**. Small C = **smoother boundary** but more mistakes.
2. **Gamma (for RBF kernel)** - Defines how far the **influence of a single training example** reaches. High gamma = **close points matter more** (can overfit). Low gamma = **far points also matter** (smooth decision boundary).
3. **Kernel** - Type of kernel function to use."

Keywords: C parameter, regularization, gamma, kernel choice, overfitting control

Q5: Applications of SVM?

Answer: "SVM works great for:

- **Image classification** (face detection, handwriting recognition)
- **Text classification** (spam detection, sentiment analysis)
- **Bioinformatics** (protein classification, cancer detection)
- **Anomaly detection**

It's especially good when you have **more features than samples** and when you need **accurate boundaries**."

Keywords: Image classification, text mining, bioinformatics, high-dimensional data

Main Functions Used:

- `SVC(kernel='rbf', C=1.0, gamma='scale')` - Creates SVM classifier
- `fit(X, y)` - Trains the model (finds support vectors)
- `predict(X)` - Classifies new points
- `support_vectors_` - Shows the support vectors
- `n_support_` - Number of support vectors per class
- `GridSearchCV()` - Finds best hyperparameters

Why: The kernel handles non-linearity. C and gamma control the decision boundary complexity. GridSearch automates finding the best parameters.

ASSIGNMENT 5: ENSEMBLE METHODS & RANDOM FOREST

Q1: What is Ensemble Learning?

Answer: "Ensemble is like **asking multiple experts** instead of one. Think of it as a **committee decision** - multiple models vote and the **majority decides**. The principle is: **'Many weak learners together make a strong learner'**. Instead of building one complex model, we build **many simple models** and **combine their predictions**. It typically gives **better accuracy** and **reduces overfitting**."

Keywords: Committee decision, multiple models, weak learners, strong learner, voting, combined predictions

Q2: What are Bagging, Boosting, and Stacking?

Answer: "Three main ensemble techniques:

1. Bagging (Bootstrap Aggregating):

- Create **multiple subsets** of data by **random sampling with replacement**
- Train **separate models** on each subset
- **Average predictions** (regression) or **majority vote** (classification)
- Example: **Random Forest**

2. Boosting:

- Train models **sequentially**
- Each new model focuses on **mistakes of previous models**
- Give **higher weight** to misclassified samples
- **Weighted combination** of all models
- Example: **AdaBoost, Gradient Boosting**

3. Stacking:

- Train **multiple different algorithms**
- Use their predictions as **input to a final meta-model**
- Meta-model learns to **combine base models optimally**"

Keywords: Bootstrap sampling, sequential training, weighted combination, parallel vs sequential, meta-model

Q3: What is Random Forest?

Answer: "Random Forest is **Bagging applied to Decision Trees** with an extra twist. We create **many decision trees** using:

1. **Different bootstrap samples** of data (random rows)
2. **Random subset of features** at each split (random columns)

Then we **average their predictions** (regression) or **majority vote** (classification).

The 'random' comes from **randomizing both data and features**. This **reduces correlation between trees** and **prevents overfitting** better than a single decision tree. It's one of the **most accurate** and **easy-to-use** algorithms!"

Keywords: Multiple decision trees, bootstrap samples, random features, majority vote, reduced correlation, averaging

Q4: Why is Random Forest better than a single Decision Tree?

Answer: "Single Decision Tree problems:

- **Overfits** easily
- **Unstable** - small changes in data cause big changes
- **High variance**

Random Forest solutions:

- **Multiple trees** with **different perspectives** reduce overfitting
- **Averaging** reduces variance
- **More stable** - one tree's error is compensated by others
- **Better generalization** to new data
- **No need to prune** trees

Trade-off: Less interpretable than single tree, but much more accurate!"

Keywords: Reduced variance, stability, generalization, averaging effect, overfitting prevention

Q5: What is feature importance in Random Forest?

Answer: "Random Forest can tell us **which features are most important** for predictions. It calculates this by seeing **how much each feature decreases impurity** across all trees. Features that frequently help create **pure splits** get higher importance scores. This helps in:

- **Feature selection** - removing unimportant features
- **Understanding** what drives predictions
- **Dimensionality reduction**"

Keywords: Feature importance, impurity decrease, feature selection, interpretability

Q6: What evaluation metrics do we use?

Answer: "**Confusion Matrix** - Shows 4 outcomes:

- **True Positive (TP)** - Correctly predicted positive
- **True Negative (TN)** - Correctly predicted negative
- **False Positive (FP)** - Wrongly predicted positive (Type I error)
- **False Negative (FN)** - Wrongly predicted negative (Type II error)

From this we calculate:

- **Accuracy** = (TP + TN) / Total - Overall correctness
- **Precision** = TP / (TP + FP) - Of all predicted positive, how many correct?
- **Recall** = TP / (TP + FN) - Of all actual positive, how many found?
- **F1-Score** = Harmonic mean of Precision and Recall"

Keywords: Confusion matrix, TP/TN/FP/FN, accuracy, precision, recall, F1-score

Main Functions Used:

- `RandomForestClassifier(n_estimators=100)` - Creates RF with 100 trees
- `BaggingClassifier(base_estimator, n_estimators)` - General bagging
- `AdaBoostClassifier(n_estimators)` - Boosting algorithm
- `fit(X, y)` - Trains all trees
- `predict(X)` - Gets majority vote
- `feature_importances_` - Shows important features
- `confusion_matrix()` - Creates confusion matrix
- `classification_report()` - Shows precision, recall, F1

Why: n_estimators controls number of trees (more = better but slower). feature_importances helps understand the model. confusion_matrix gives detailed performance breakdown.

ASSIGNMENT 6: CLUSTERING

Q1: What is Clustering?

Answer: "Clustering is **unsupervised learning** - we have **no labels**, just data. The goal is to **group similar items together**. Think of organizing a messy closet - you group **similar clothes** without anyone telling you how. We want **high similarity within clusters** and **low similarity between clusters**. It's used for **customer segmentation, image compression, anomaly detection**."

Keywords: Unsupervised learning, grouping, similarity, no labels, pattern discovery

Q2: Explain K-Means Clustering?

Answer: "K-Means is like **placing K flags** (centroids) and asking everyone to **go to their nearest flag**. Algorithm steps:

1. **Initialize**: Randomly place K centroids
2. **Assignment**: Assign each point to nearest centroid
3. **Update**: Move centroids to **center of their group**
4. **Repeat** steps 2-3 until centroids **stop moving**

It's **fast** and **simple**, but you must **choose K** beforehand. It works well when clusters are **spherical and similar-sized**."

Keywords: Centroids, iterative algorithm, K clusters, distance-based, convergence

Q3: How to choose the number of clusters (K)?

Answer: "Two main methods:

1. Elbow Method:

- Plot **total distance** from points to their centroids (called **inertia**) vs K
- Look for the **'elbow'** - where the improvement slows down
- That K is usually optimal

2. Silhouette Score:

- Measures how **similar** a point is to its own cluster vs nearest cluster
- Score ranges **-1 to 1**
- Higher = **better clusters**
- Choose K with **highest average silhouette score**"

Keywords: Elbow method, inertia, silhouette score, optimal K, validation

Q4: What is Spectral Clustering?

Answer: "Spectral Clustering uses **graph theory**. Steps:

1. Create a **similarity graph** - points are nodes, edges show similarity
2. Calculate **graph Laplacian** matrix
3. Find **eigenvalues and eigenvectors**
4. Use eigenvectors as **new features**
5. Apply **K-Means** on these new features

It can find **non-convex clusters** (complex shapes like crescents, circles) that K-Means can't. Uses **connectivity** instead of just distance."

Keywords: Graph-based, eigenvalues, Laplacian matrix, non-convex shapes, connectivity

Q5: What is DBSCAN?

Answer: "**DBSCAN (Density-Based Spatial Clustering)** groups points in **dense regions**. It's like finding **crowds in a park**. Key concepts:

- **Core point**: Has **minimum neighbors (min_samples)** within **radius (eps)**
- **Border point**: Within eps of a core point but not core itself
- **Noise point**: Neither core nor border (**outliers**)

Advantages:

- **Automatically finds** number of clusters
- Can find **arbitrary shaped** clusters
- Identifies **outliers**

Disadvantages:

- Struggles with **varying densities**
- Needs to tune **eps and min_samples**"

Keywords: Density-based, core points, eps, min_samples, arbitrary shapes, outlier detection

Q6: Compare K-Means, Spectral, and DBSCAN?

Answer:
"**K-Means:**

- **Need to specify K**
- Fast, simple
- Only finds **spherical** clusters
- Sensitive to **outliers**

Spectral:

- **Need to specify K**
- Finds **complex shapes**
- **Slower** (eigenvalue computation)
- Good for **small-medium datasets**

DBSCAN:

- **Finds K automatically**
- Handles **noise** and **arbitrary shapes**
- Struggles with **varying densities**
- Sensitive to **eps and min_samples** parameters"

Keywords: Shape detection, outlier handling, parameter sensitivity, speed comparison

Main Functions Used:

- `KMeans(n_clusters=k)` - Creates K-Means model
- `SpectralClustering(n_clusters=k, affinity='nearest_neighbors')` - Spectral clustering
- `DBSCAN(eps=0.5, min_samples=5)` - DBSCAN
- `fit(X)` or `fit_predict(X)` - Performs clustering (no y needed - unsupervised!)
- `labels_` - Shows cluster assignments
- `silhouette_score(X, labels)` - Evaluates clustering quality
- `inertia_` - Sum of squared distances (K-Means)

Why: No y in fit because it's unsupervised. silhouette_score helps compare different clustering approaches. Different algorithms need different parameters.

GENERAL QUESTIONS ACROSS ALL ASSIGNMENTS

Q1: What is the difference between supervised and unsupervised learning?

Answer: "Supervised Learning:

- Has **labeled data** (X and y)
- Model learns from **examples** with answers
- Used for **prediction**
- Examples: **Classification, Regression**

Unsupervised Learning:

- **No labels** (only X)
- Model finds **patterns** on its own

- Used for **exploration**
- Examples: **Clustering, Dimensionality Reduction**"

Keywords: Labels, prediction vs exploration, classification/regression vs clustering

Q2: What is overfitting and underfitting?

Answer: "**Overfitting:**

- Model **memorizes** training data
- **Great on training, poor on test** data
- Too **complex** model
- Solution: **Regularization, more data, simpler model**

Underfitting:

- Model is **too simple**
- **Poor on both training and test** data
- Hasn't learned patterns
- Solution: **More complex model, more features, train longer**

Sweet spot is **good generalization** - performs well on unseen data."

Keywords: Memorization, generalization, bias-variance trade-off, model complexity

Q3: What is train-test split and why do we use it?

Answer: "We **divide dataset** into:

- **Training set (70-80%)** - Model **learns** from this
- **Testing set (20-30%)** - Model **evaluated** on this (unseen during training)

Why? If we test on same data we trained on, model might have **memorized** answers. Testing on separate data checks if model **generalizes** to new situations. It's like **practicing for exam vs taking actual exam**."

Keywords: Generalization, unseen data, evaluation, train-test split ratio

Q4: What is feature scaling and why is it important?

Answer: "**Feature scaling** brings all features to **similar range**. Methods:

1. Normalization (Min-Max): Scale to[1]
2. Standardization (Z-score): Mean=0, SD=1

Why needed?

- Algorithms using **distance** (K-NN, SVM) are **affected by scale**
- Feature with **larger values dominates**
- Example: Age (0-100) vs Salary (0-100000) - salary would dominate without scaling

Not needed for: Decision Trees, Random Forest (they use splits, not distances)"

Keywords: Normalization, standardization, distance-based algorithms, scale invariance

Q5: What is cross-validation?

Answer: "Instead of single train-test split, use **k-fold cross-validation**:

1. **Split data into K folds**
2. Train on K-1 folds, test on 1 fold
3. **Repeat K times**, each time using different fold as test
4. **Average** the K scores

Advantages:

- **More reliable** estimate
- Uses **all data** for both training and testing
- **Reduces variance** in evaluation
- Especially good for **small datasets**"

Keywords: K-fold, reliable estimate, variance reduction, complete data usage

Q6: What is a confusion matrix?

Answer: "A **table showing 4 outcomes** for binary classification:

```
                Predicted
              Positive  Negative
Actual Pos    TP        FN
       Neg    FP        TN
```

TP (True Positive): Correctly predicted positive
TN (True Negative): Correctly predicted negative
FP (False Positive): Wrongly predicted positive (**Type I error**)
FN (False Negative): Wrongly predicted negative (**Type II error**)

From this we calculate: **Accuracy, Precision, Recall, F1-Score**"

Keywords: TP/TN/FP/FN, Type I/II errors, classification metrics

Q7: Explain bias-variance trade-off?

Answer: "**Bias**: Error from **wrong assumptions** (underfitting)

- **High bias** = model too **simple**, misses patterns
- Example: Using linear model for non-linear data

Variance: Error from **sensitivity to training data** (overfitting)

- **High variance** = model too **complex**, captures noise
- Example: Very deep decision tree memorizing training data

Trade-off: Reducing bias increases variance, and vice versa. Goal is to find **optimal balance** where **total error is minimum**."

Keywords: Underfitting vs overfitting, model complexity, total error, optimal balance

TIPS FOR VIVA:

1. **Start with the concept story** (explain like teaching)
2. **Then use keywords** (shows technical knowledge)
3. **Give an example** if time permits
4. **Connect to real-world** application
5. **If you don't know**: Say "I know the basic concept..." and explain what you DO know
6. **Confidence**: Even if you forgot some details, explain what you remember clearly

⚡ QUICK REVISION - JUST BEFORE VIVA (5 minutes):

Remember these core concepts:

1. **Linear Regression**: Best-fit line, minimize error
2. **k-NN**: Find k nearest neighbors, majority vote
3. **Naive Bayes**: Probability based, assumes independence
4. **Decision Tree**: If-else rules, splits on features
5. **SVM**: Maximum margin, support vectors, kernel trick
6. **Random Forest**: Many decision trees, voting
7. **K-Means**: K centroids, assign points to nearest
8. **DBSCAN**: Density-based, finds arbitrary shapes

Universal answers that work for many questions:

- Overfitting solution: "Regularization, more data, cross-validation"
- Choosing parameters: "Grid search with cross-validation"
- Evaluation: "Use appropriate metrics and confusion matrix"

- Data preprocessing: "Handle missing values, scale features, encode categoricals"

YOU'VE GOT THIS!

This guide covers **all major concepts** with explanations, keywords, and the **storytelling approach** you requested. Read through **2-3 times**, and you'll be able to answer **any viva question** confidently. Even with 10-20% retention, you'll have enough to give **solid answers**.

Good luck with your viva!

⁂

1. Assignmnet-No-07-Regression_write-up.docx
2. Linear_Regression.ipynb
3. ML_Assignment-6_Clustering_Write-Up.docx
4. ML1_LabAssignment1-Write-Up.docx
5. ML-Assignement-2-kNN-and-NB_write_up.docx
6. Assignment-3_DecisionTree_WriteUp.docx
7. ML-Assignment-4_SVM_DT_comparison_Write-Up.docx
8. ML_Assignment-5_Ensemble_Random-Forest_Write-Up.docx
9. Untitled1.ipynb
10. ensemble_random_forest_ML_Lab_5.ipynb