Machine Learning Assignment- 5

1. R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?

R-squared is generally considered a better measure of goodness of fit in regression compared to RSS. Here's why:

R-squared:

- Also known as the coefficient of determination

- Ranges from 0 to 1

- Represents the proportion of variance in the dependent variable explained by the independent variables

- Is scale-independent, making it easier to interpret across different datasets

R-squared = 1 - (RSS / TSS)

Where RSS is the Residual Sum of Squares and TSS is the Total Sum of Squares.

Residual Sum of Squares (RSS):

- Measures the total deviation of the response values from the fit

- Is scale-dependent, making it difficult to compare across different datasets

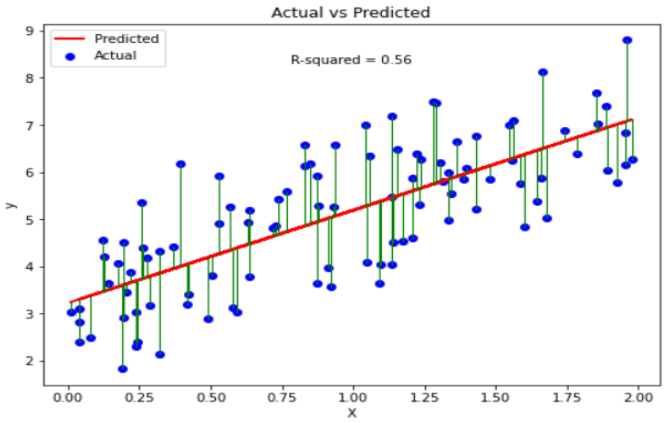
- Lower values indicate better fit, but the scale can vary widely depending on the data

RSS = Σ(y_i - ŷ_i)^2

Where y_i are the actual values and ŷ_i are the predicted values.

While RSS is useful for comparing models on the same dataset, R-squared provides a standardized measure that's easier to interpret and compare across different datasets.

Graph: I would include a scatter plot showing actual vs predicted values, with the regression line and residuals marked. The R-squared value would be displayed on the graph.



2. What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.

These three metrics are related to the variability in a regression model:

Total Sum of Squares (TSS):

- Measures the total variation in the dependent variable

- TSS = Σ(y_i - ȳ)^2, where ȳ is the mean of y

Explained Sum of Squares (ESS):

- Measures the variation explained by the model

- ESS = Σ(ŷ_i - ȳ)^2

Residual Sum of Squares (RSS):

- Measures the unexplained variation

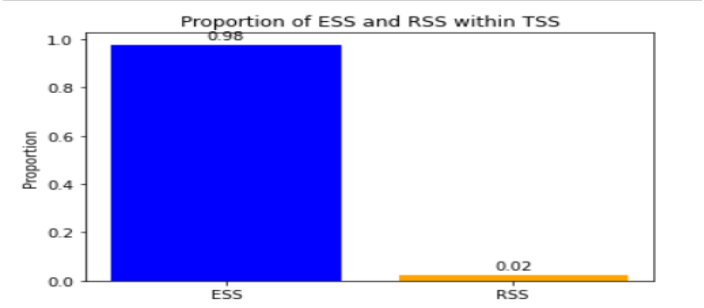
- RSS = Σ(y_i - ŷ_i)^2

The relationship between these metrics:

TSS = ESS + RSS

This equation demonstrates how the total variation in the data (TSS) is partitioned into the part explained by the model (ESS) and the unexplained part (RSS).

Graph: I would include a bar chart showing the relative proportions of ESS and RSS within TSS for a given model.



3. What is the need of regularization in machine learning?

Regularization is crucial in machine learning for preventing overfitting. It adds a penalty term to the loss function, discouraging the model from becoming too complex.

Key reasons for using regularization:

1. Prevents overfitting: Helps the model generalize better to unseen data

2. Feature selection: Can automatically reduce the impact of less important features

3. Improves model stability: Makes the model less sensitive to small changes in the input data

4. Handles multicollinearity: Useful when dealing with correlated features

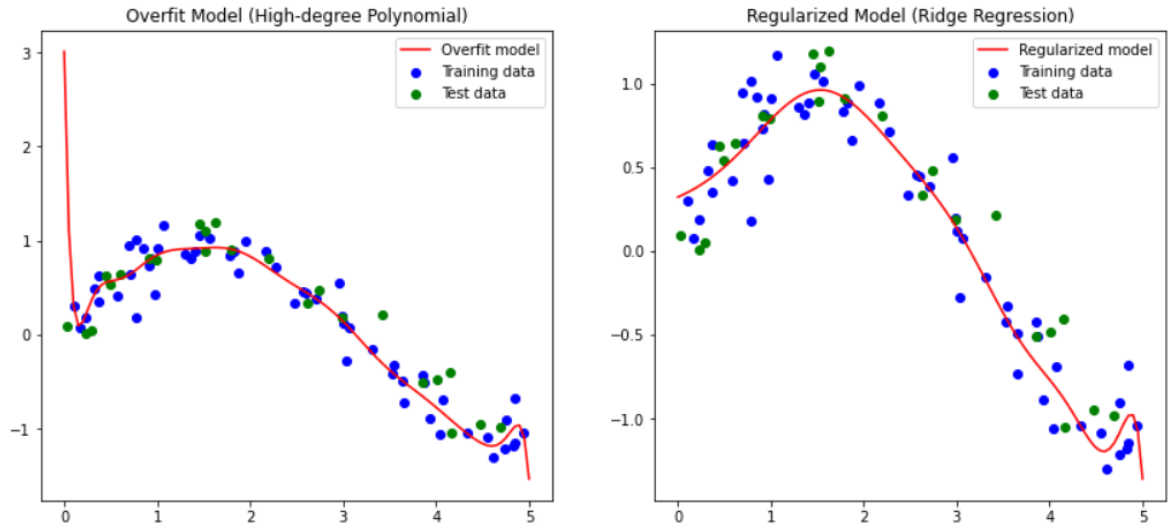
Common regularization techniques:

- L1 (Lasso): Adds absolute value of magnitude of coefficients to the loss function

- L2 (Ridge): Adds squared magnitude of coefficients to the loss function

- Elastic Net: Combines L1 and L2 regularization

Graph: I would show two plots side by side - one showing an overfit model on training data, and another showing a regularized model that generalizes better to test data.



4. What is Gini–impurity index?

The Gini impurity index is a measure used in decision trees to determine the quality of a split. It quantifies the probability of incorrectly classifying a randomly chosen element if it were randomly labeled according to the distribution of labels in the subset.

Formula:

Gini Impurity = 1 - Σ(p_i)^2

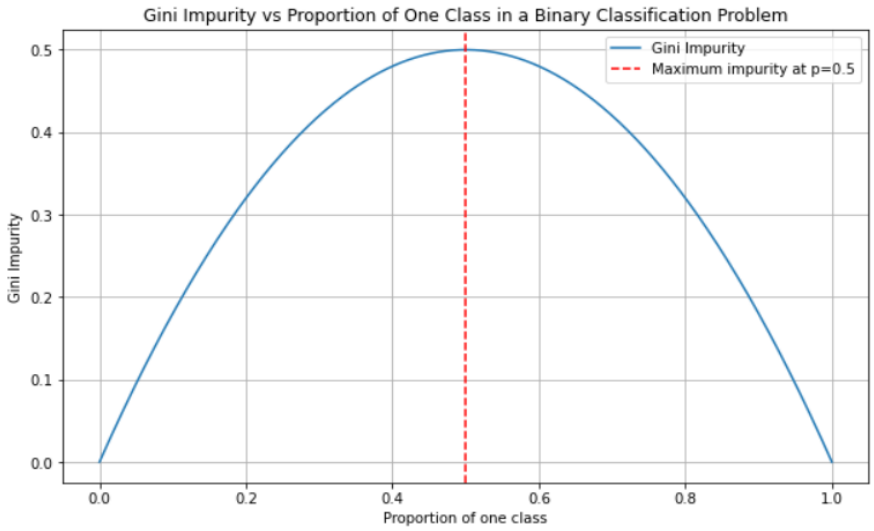
Where p_i is the probability of an item being classified for a particular class.

Properties:

- Ranges from 0 (pure node) to 0.5 (maximum impurity for binary classification)

- Lower values indicate better splits

Graph: I would include a plot showing Gini impurity vs proportion of one class in a binary classification problem. The curve would be parabolic, with minimum values at 0 and 1, and the maximum at 0.5.



5. Are unregularized decision-trees prone to overfitting? If yes, why?

Yes, unregularized decision trees are prone to overfitting. Here's why:

1. Unlimited depth: Without constraints, trees can grow very deep, potentially creating a leaf for each training example.

2. Memorization: Deep trees can essentially memorize the training data, including noise.

3. Low bias, high variance: They can capture complex patterns but are sensitive to small changes in the data.

4. Poor generalization: Overfit trees perform well on training data but poorly on unseen data.

Regularization techniques for decision trees:

- Maximum depth

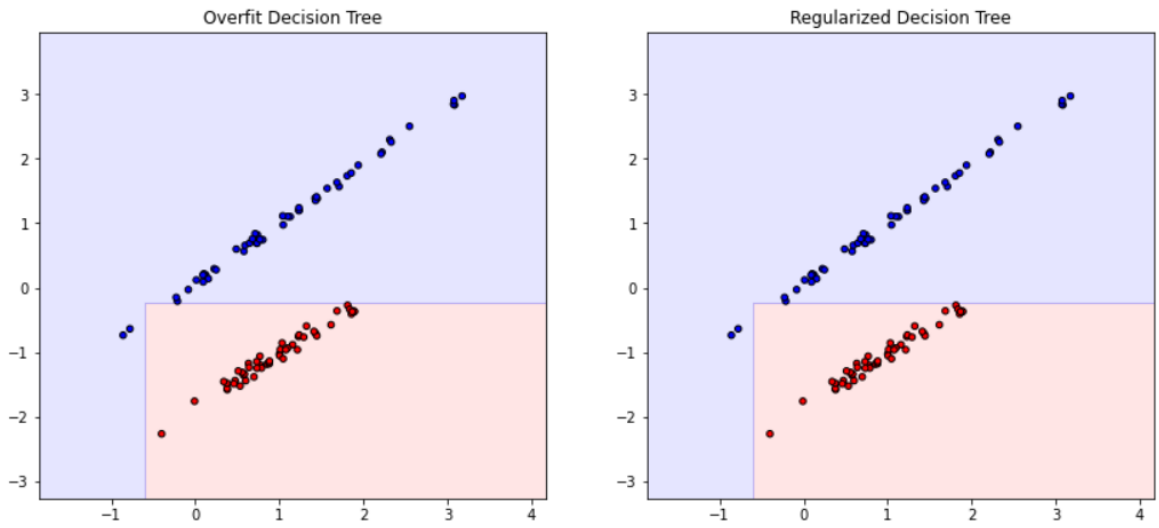
- Minimum samples per leaf

- Minimum samples for split

- Maximum number of leaf nodes

- Pruning

Graph: I would show two decision boundaries - one from an overfit tree (very complex) and another from a regularized tree (smoother, more generalized).



6. What is an ensemble technique in machine learning?

Ensemble techniques combine multiple machine learning models to create a more powerful predictive model.

Key concepts:

1. Diversity: Using different models or training on different subsets of data

2. Aggregation: Combining predictions through voting, averaging, or more complex methods

3. Bias-variance trade-off: Ensembles often reduce variance while maintaining low bias

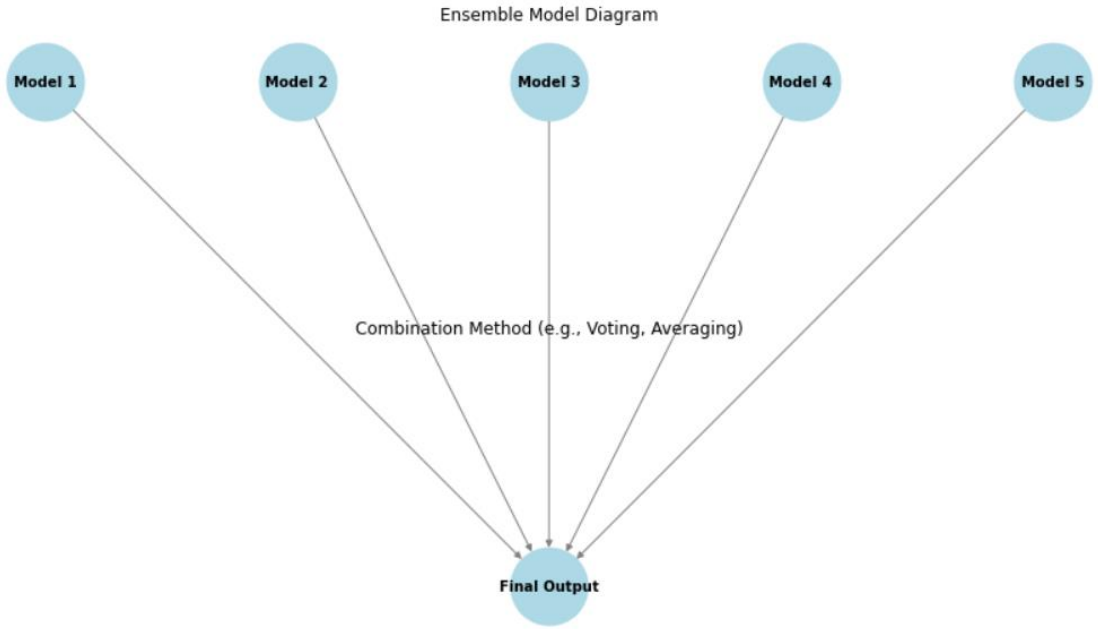
Common ensemble methods:

- Random Forests
- Gradient Boosting Machines (GBM)
- AdaBoost
- Stacking

Benefits:

- Improved accuracy
- Better generalization
- Reduced overfitting
- Increased stability

Graph: I would show a diagram illustrating how multiple models feed into an ensemble, with their predictions being combined for a final output.



7. What is the difference between Bagging and Boosting techniques?

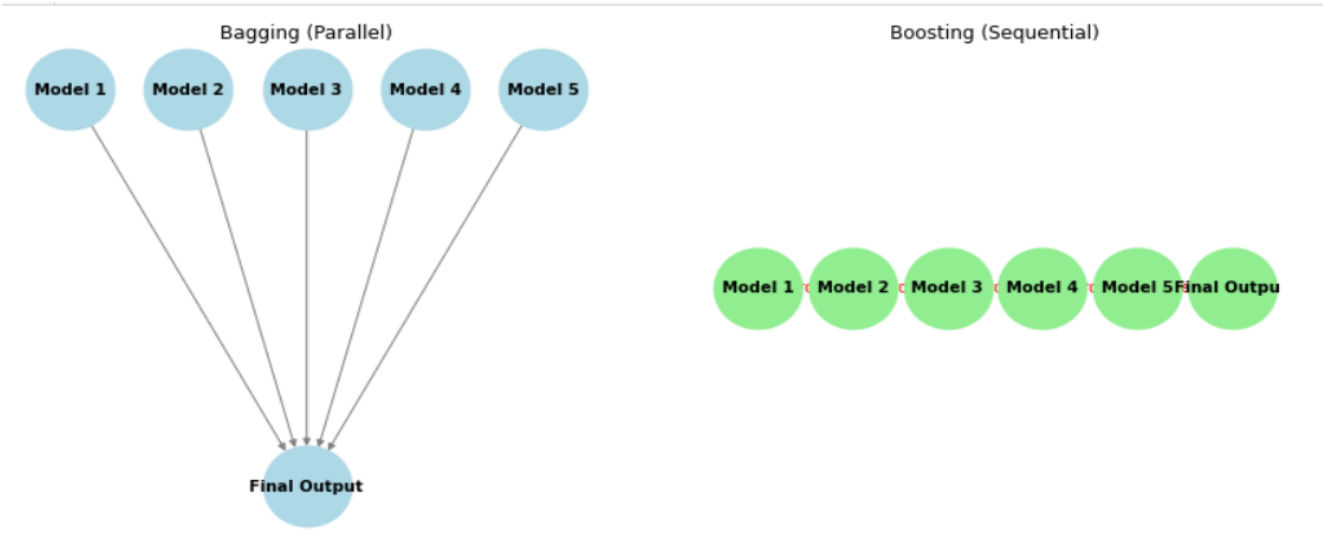
Bagging (Bootstrap Aggregating):

- Builds multiple independent models in parallel
- Each model is trained on a random subset of the data (with replacement)
- Final prediction is average (regression) or majority vote (classification)
- Reduces variance, good for high-variance low-bias models
- Example: Random Forests

Boosting:

- Builds models sequentially
- Each new model focuses on the errors of the previous ones
- Assigns higher weight to misclassified instances
- Reduces bias, good for high-bias low-variance models
- Examples: AdaBoost, Gradient Boosting

Graph: I would create a side-by-side comparison showing the parallel nature of bagging vs the sequential nature of boosting, with arrows indicating how data and errors flow through each process.



8. What is out-of-bag error in random forests?

Out-of-bag (OOB) error is a method of measuring prediction error in random forests using only the training data.

Key points:

- In random forests, each tree is trained on a bootstrap sample (about 2/3) of the data

- The remaining 1/3 of the data not used for a particular tree is called the out-of-bag sample for that tree

- OOB error is calculated using predictions on these OOB samples

Process:

1. For each observation, find all trees where it was OOB

2. Use these trees to predict for this observation

3. Calculate the error rate across all observations

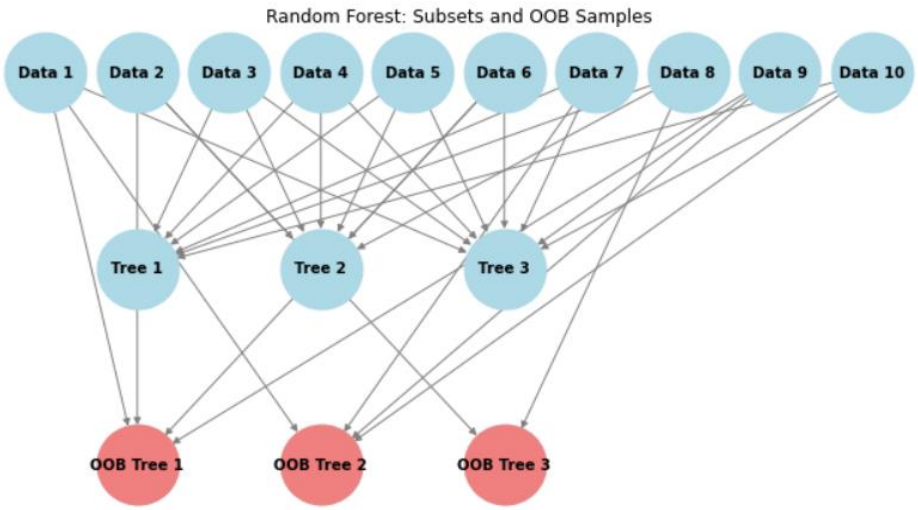
Benefits:

- Provides an unbiased estimate of the test error

- Eliminates the need for a separate validation set

- Computationally efficient

Graph: I would show a diagram illustrating how different trees in a random forest use different subsets of data, with OOB samples highlighted for each tree.



9. What is K-fold cross-validation?

K-fold cross-validation is a resampling method used to evaluate machine learning models.

Process:

1. Divide the dataset into K equally sized subsets (folds)

2. For each fold i from 1 to K:

   - Use fold i as the validation set

   - Use the remaining K-1 folds as the training set

   - Train the model and evaluate on the validation set

3. Average the K validation results

Common choices for K: 5, 10

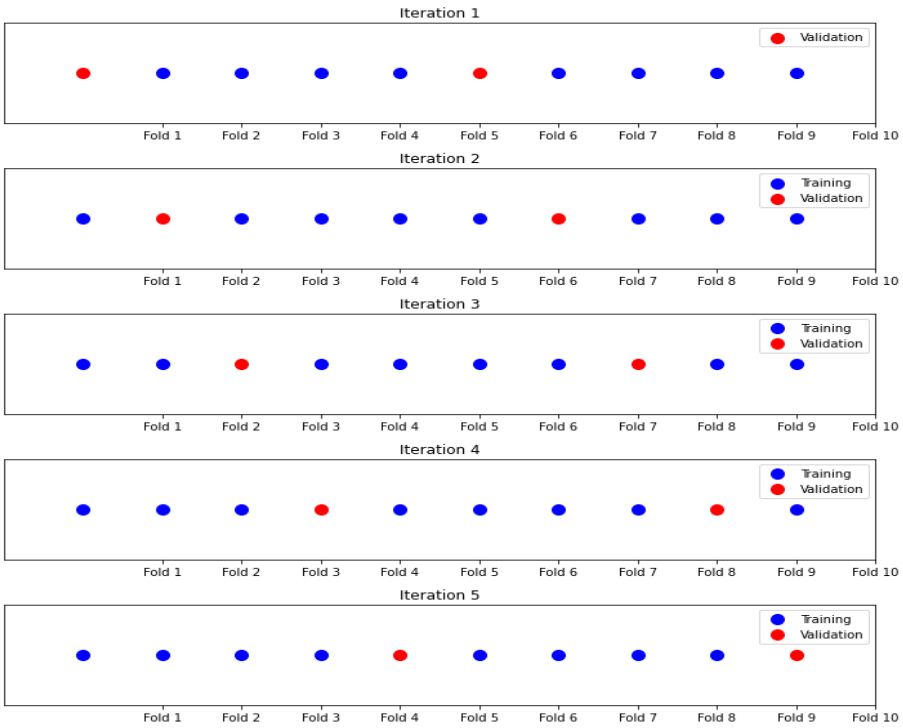
Benefits:

- More reliable estimate of model performance

- Uses all data for both training and validation

- Reduces bias compared to a single train-test split

Graph: I would create a diagram showing a dataset divided into K folds, with one fold highlighted as the validation set and the rest as training data, then showing how this rotates for each iteration.



10. What is hyper parameter tuning in machine learning and why it is done?

Hyperparameter tuning is the process of finding the optimal values for a model's hyperparameters.

Why it's done:

- Improve model performance

- Adapt the model to specific datasets

- Balance bias-variance trade-off

Common methods:

1. Grid Search: Exhaustive search through a specified parameter grid

2. Random Search: Randomly samples from the parameter space

3. Bayesian Optimization: Uses probabilistic model to guide the search

Steps:

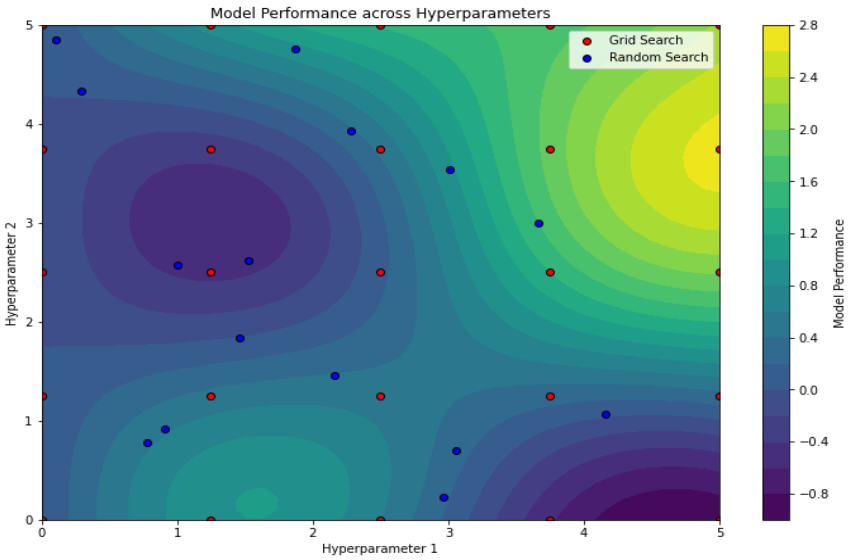
1. Define the hyperparameter search space

2. Choose a search strategy

3. Use cross-validation to evaluate each configuration

4. Select the best performing hyperparameters

Graph: I would show a contour plot of model performance across two hyperparameters, with points indicating sampled configurations for different search strategies.



11. What issues can occur if we have a large learning rate in Gradient Descent?

A large learning rate in Gradient Descent can cause several problems:

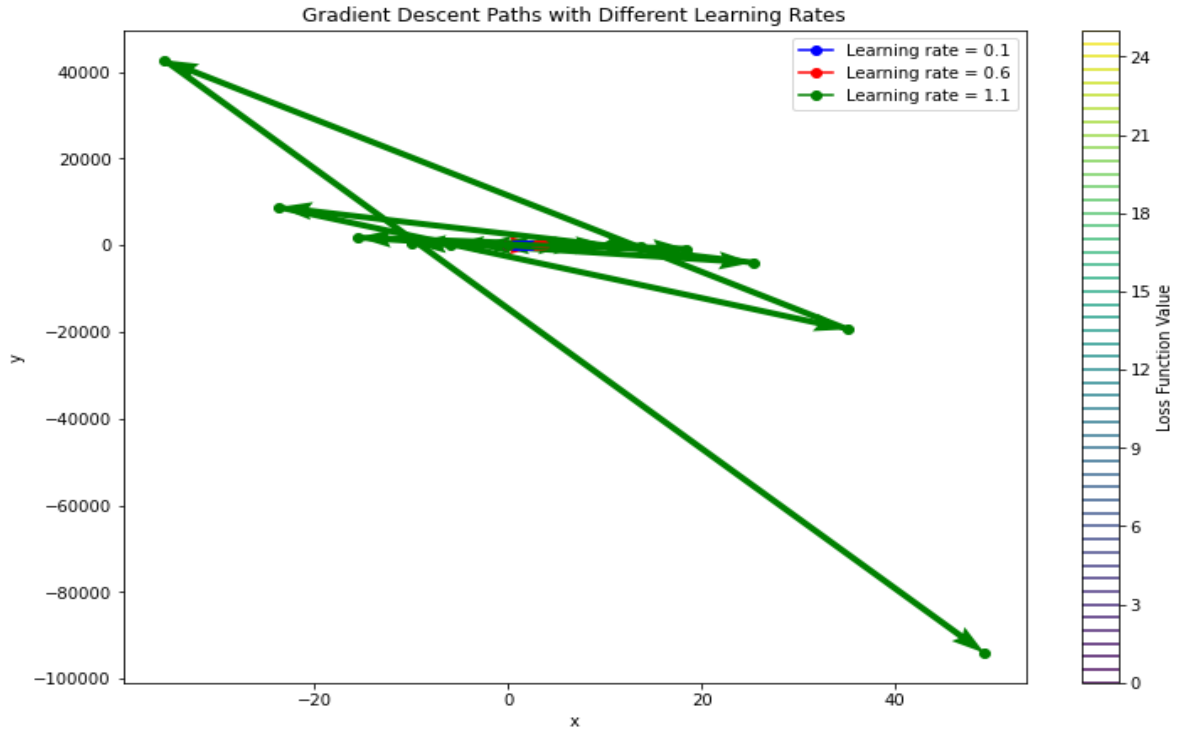
1. Overshooting: The algorithm may jump over the minimum of the loss function

2. Divergence: The loss may increase instead of decreasing

3. Oscillation: The algorithm may bounce back and forth around the optimal point

4. Instability: Small changes in the data can lead to large changes in the model

Graph: I would show a 2D plot of a loss function with arrows indicating the path of gradient descent for different learning rates - one converging nicely, one overshooting, and one diverging.



12. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

Logistic Regression is inherently a linear classifier and cannot directly classify non-linear data without modifications.

Limitations:

- Assumes a linear decision boundary

- Cannot capture complex, non-linear relationships between features

Possible solutions for non-linear data:

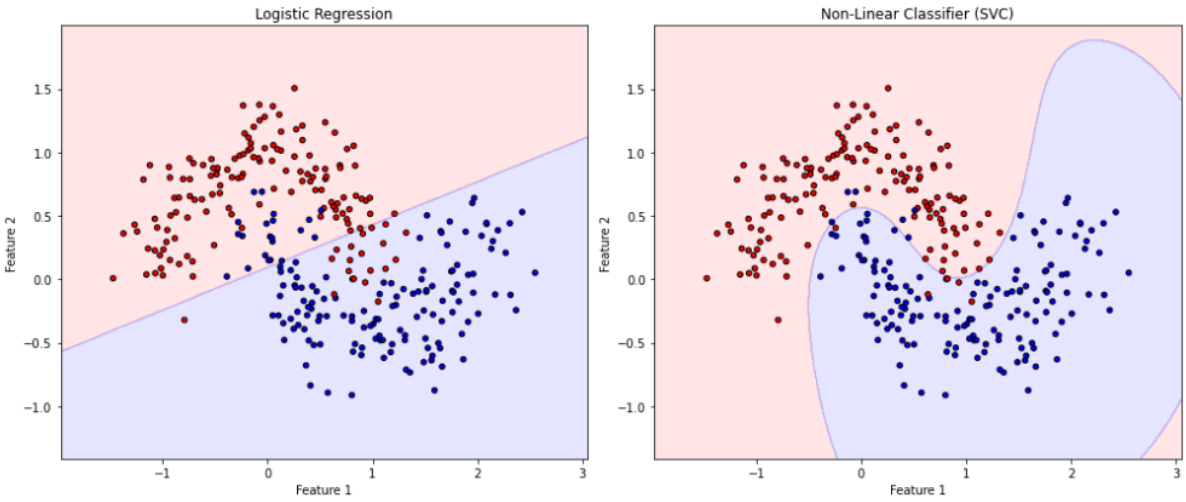
1. Feature engineering: Create non-linear features manually

2. Polynomial features: Add polynomial terms of existing features

3. Kernel trick: Implicitly map to a higher-dimensional space (e.g., Kernel Logistic Regression)

4. Use naturally non-linear models: Decision Trees, SVMs with non-linear kernels, Neural Networks

Graph: I would show a 2D plot with non-linearly separable data points and two decision boundaries - a straight line for logistic regression (performing poorly) and a curved line for a non-linear classifier (performing well).



13. Differentiate between Adaboost and Gradient Boosting.

AdaBoost (Adaptive Boosting):

- Focuses on hard-to-classify instances

- Adjusts the sample distribution by increasing weights of misclassified samples

- Typically uses shallow decision trees (stumps) as weak learners

- Combines weak learners using a weighted sum

Gradient Boosting:

- Focuses on the residuals of the previous models

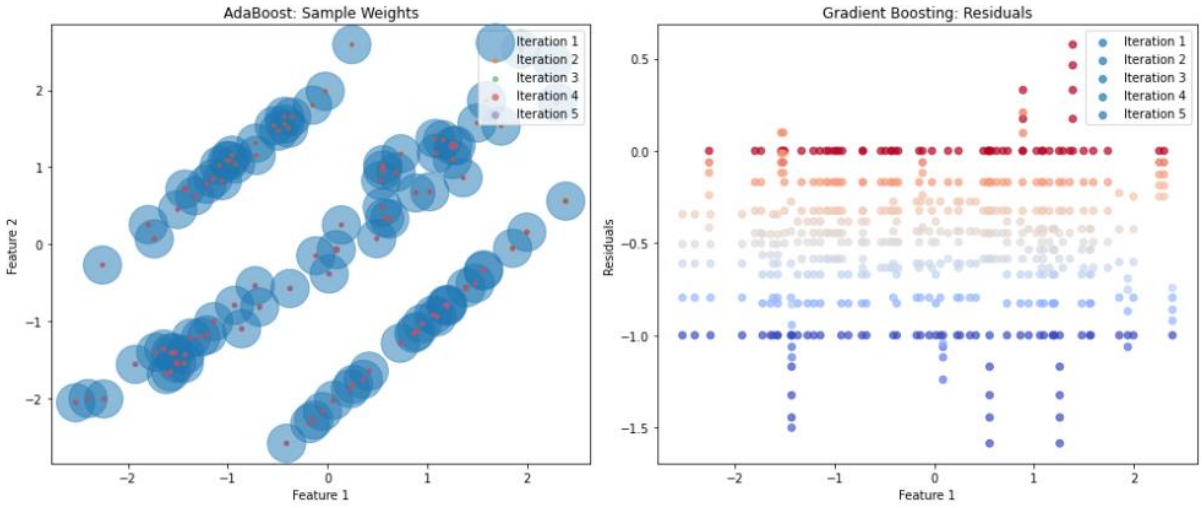
- Fits new models to the negative gradient of the loss function

- Can use various differentiable loss functions

- Typically uses deeper trees as base learners

- Combines models through addition

Graph: I would create a side-by-side comparison showing how AdaBoost adjusts sample weights vs how Gradient Boosting fits to residuals, with multiple iterations for each method.



14. What is bias-variance trade off in machine learning?

The bias-variance trade-off is a fundamental concept in machine learning that deals with the balance between a model's ability to fit the training data (low bias) and its ability to generalize to new data (low variance).

Bias:

- Error due to overly simplistic assumptions

- High bias leads to underfitting

- Example: Linear regression on non-linear data

Variance:

- Error due to excessive sensitivity to small fluctuations in the training set

- High variance leads to overfitting

- Example: High-degree polynomial regression

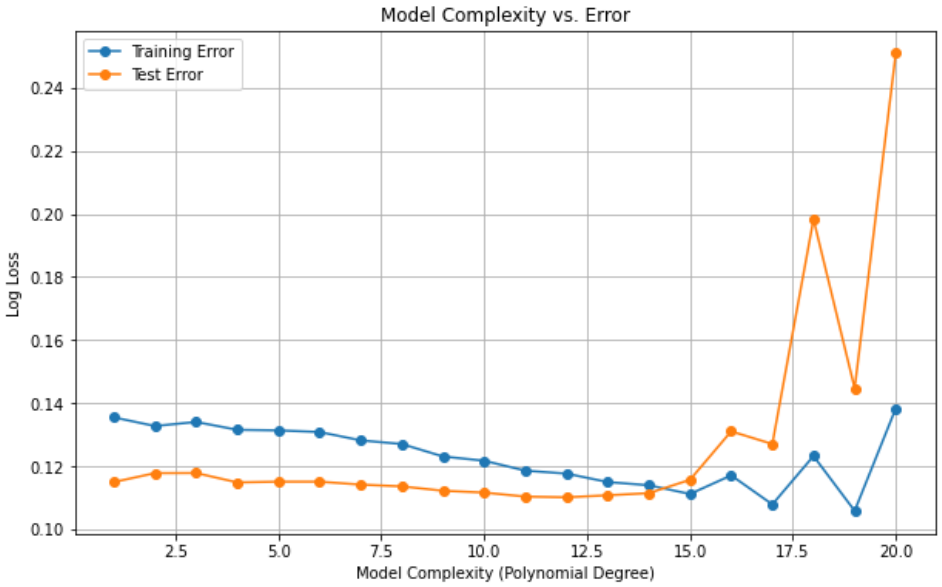
Trade-off:

- Decreasing bias often increases variance and vice versa

- The goal is to find the sweet spot that minimizes total error

Total Error = (Bias)^2 + Variance + Irreducible Error

Graph: I would show a classic U-shaped curve plotting model complexity against error, with separate lines for training error (decreasing) and test error (U-shaped). The point of minimum test error represents the optimal trade-off.



15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Support Vector Machines (SVMs) use kernels to transform the input space, allowing for non-linear decision boundaries.

Linear Kernel:

- Simplest kernel

- $K(x, y) = x^T y$

- Suitable for linearly separable data

- Fast to compute, but limited in expressiveness

RBF (Radial Basis Function) Kernel:

- Also known as Gaussian kernel

- $K(x, y) = \exp(-\gamma ||x - y||^2)$

- Maps to an infinite-dimensional space

- Versatile, works well for many types of data

- γ parameter controls the decision boundary's flexibility

Polynomial Kernel:

- $K(x, y) = (\alpha x^T y + c)^d$

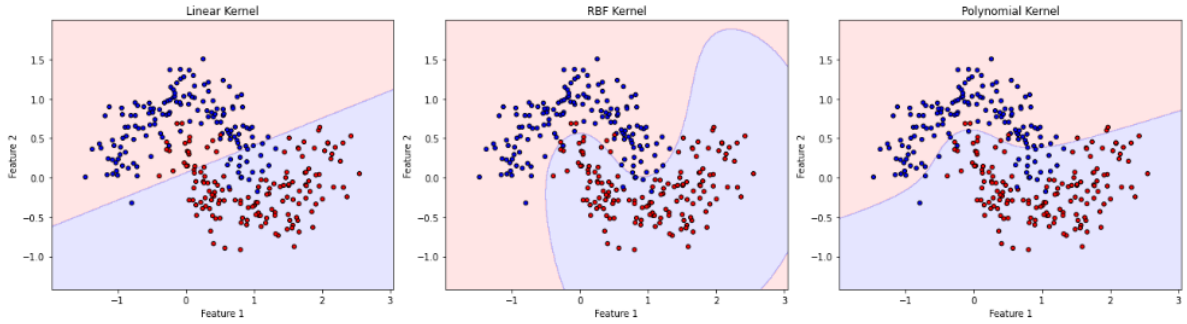
- Allows for curved decision boundaries

- Parameters: degree d, α (scale), and c (offset)

- Can capture more complex relationships than linear kernel

- Risk of overfitting with high degrees

Graph: I would create a 2D plot showing decision boundaries for each kernel type on the same non-linearly separable dataset. This would illustrate how the linear kernel fails, while RBF and polynomial kernels can create suitable non-linear boundaries.



These kernels allow SVMs to handle a wide range of data types and complexities, making them versatile for various classification tasks.