Deloitte.

Citizen Data Scientist – Unit 2 | Part 3 Supervised Learning

Julio César Álvarez Iglesias

April 2024

MAKING AN IMPACT THAT MATTERS since 1845

Curse of Dimensionality

This phenomenon occurs when the feature space increases significantly, which leads to a vast increase in the volume of that space. This growth in dimensionality makes the available data sparse, meaning that the density of the data decreases rapidly, making the data less and less representative of the space it occupies.

Curse of Dimensionality

This phenomenon occurs when the feature space increases significantly, which leads to a vast increase in the volume of that space. This growth in dimensionality makes the available data sparse, meaning that the density of the data decreases rapidly, making the data less and less representative of the space it occupies.

Why Does It Happen?

This phenomenon happens due to the volume of the space increasing so much that the available data becomes sparse. This sparsity is problematic because it makes any conclusion drawn from the data less reliable. It becomes difficult to predict when new data points might appear, and it’s challenging to determine what the outcomes or targets associated with these new points will be.

Curse of Dimensionality

This phenomenon occurs when the feature space increases significantly, which leads to a vast increase in the volume of that space. This growth in dimensionality makes the available data sparse, meaning that the density of the data decreases rapidly, making the data less and less representative of the space it occupies.

Why Does It Happen?

This phenomenon happens due to the volume of the space increasing so much that the available data becomes sparse. This sparsity is problematic because it makes any conclusion drawn from the data less reliable. It becomes difficult to predict when new data points might appear, and it's challenging to determine what the outcomes or targets associated with these new points will be.

Strategies to Mitigate the Curse of Dimensionality

1. Feature Selection: Identifying and using only the most relevant features can reduce dimensionality without much loss of information.
2. Dimensionality Reduction: Techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) help by transforming features into a lower-dimensional space while retaining the essence of the original data.
    I. PCA works by finding the directions of maximum variance in high-dimensional data and projecting it onto a new subspace with fewer dimensions.
    II. LDA tries to find a feature subspace that enhances class separability.

PCA vs LDA

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This technique is used primarily to reduce the dimensionality of the data while retaining as much variability as possible.

- Mechanism: PCA works by identifying the directions (principal components) along which the variance of the data is maximized. This is often visualized as fitting an n-dimensional ellipse to the data, where each axis represents a principal component.
- Usage: PCA is unsupervised, meaning it does not consider any labels in the data. It's typically used for exploratory data analysis, noise reduction, and feature extraction and construction.

Linear Discriminant Analysis (LDA), on the other hand, is a supervised dimensionality reduction technique used primarily to maximize the separability among known categories. It aims to project the data onto a lower-dimensional space with good class-separability in order avoid performance loss.

- Mechanism: LDA works by determining axes that maximize the separation between multiple classes. It projects data points onto a line so that the distances between the means of the different categories are as large as possible while also minimizing the variation (scatter) within each category itself.
- Usage: LDA is used when the classes of the dataset are known and labeled, making it suitable for preparing data for classification tasks.

Linear Discriminant Analysis

LDA is based on concepts from statistics, specifically discriminant functions that attempt to express the separation between classes using a combination of features. The primary objectives are: Maximize the distance between the means of different classes and Minimize the scatter (variance) within each class. These objectives ensure that the classes are as distinct as possible when projected onto a lower-dimensional space.

Mathematical Formulation

Given classes $C_1, C_2 \dots C_k$, the formula for LDA involves several steps:

1. Compute the within-class scatter matrix S_W:

$$S_W = \sum_{i=1}^{k} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$

   where μ_i is the mean of the samples in class C_i.

2. Compute the between-class scatter matrix S_B:

$$S_B = \sum_{i=1}^{k} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

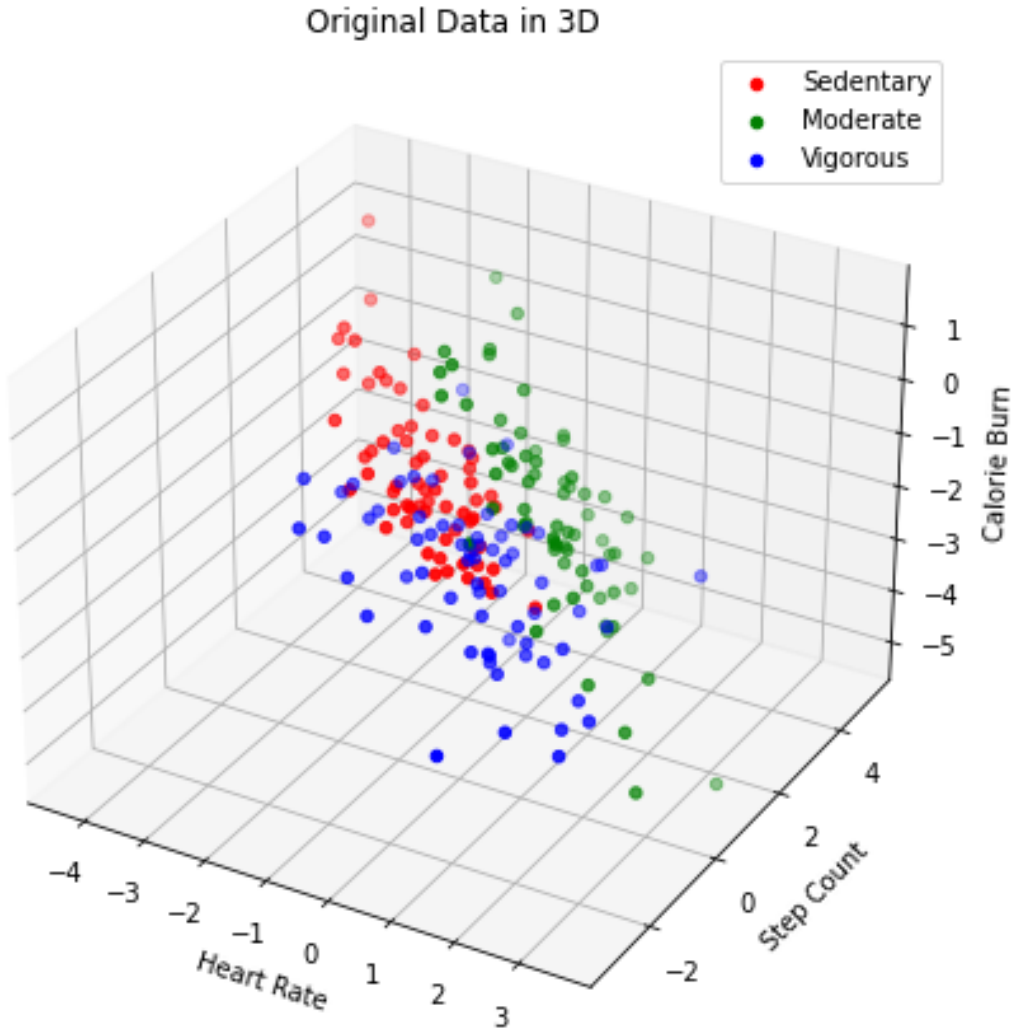
   where μ is the overall mean of the data, and N_i is the number of samples in class C_i.

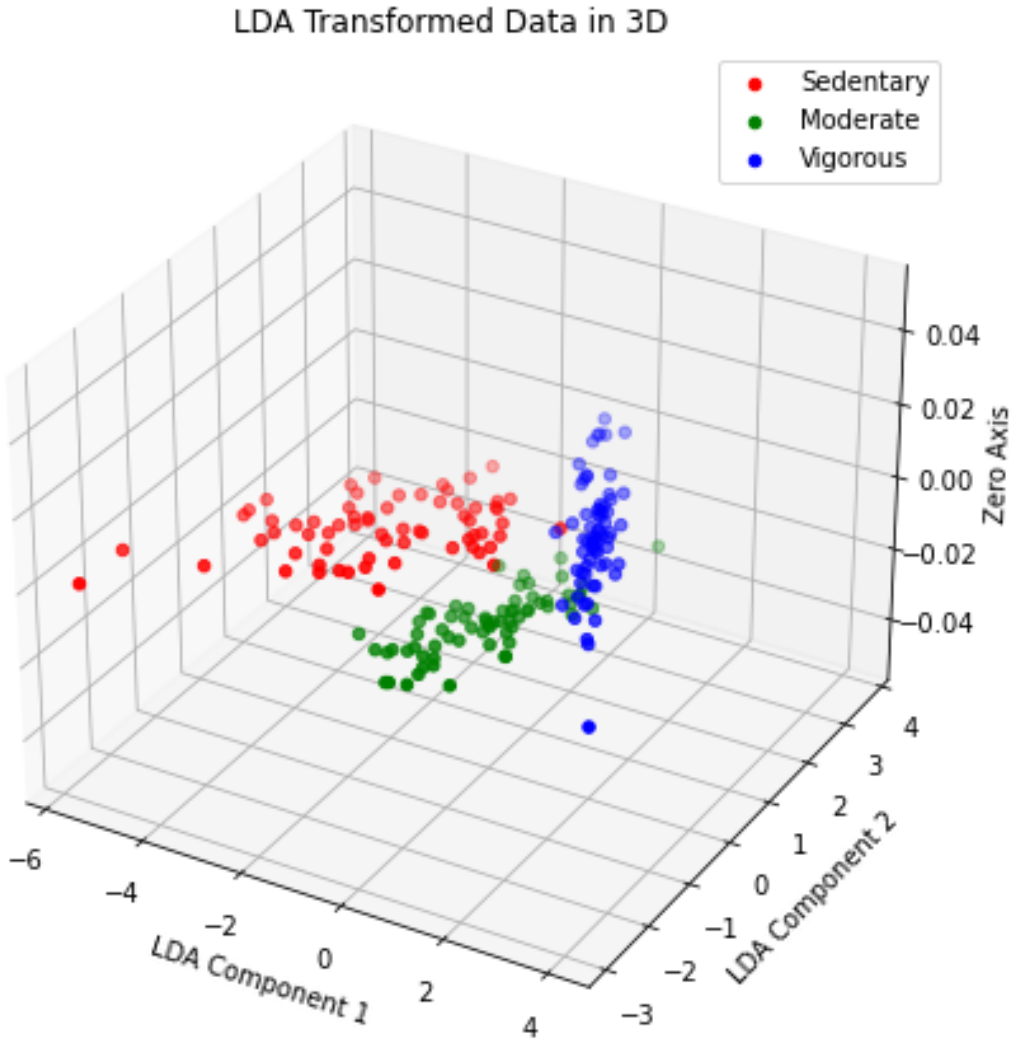
3. Solve the generalized eigenvalue problem for the matrix ${S_W}^{-1} S_B$:

$${S_W}^{-1} S_B v = \lambda v$$

   where the eigenvectors v that correspond to the largest eigenvalues λ are the directions that best separate the classes. These directions are used to project the data into a lower-dimensional space.

Linear Discriminant Analysis





Hyperparameter Tuning

Hyperparameter tuning is a crucial step in the process of designing and training machine learning models. It involves adjusting the parameters of a model that are not learned from data but are set prior to the training process. These parameters, known as hyperparameters, can significantly influence the performance of a model on a given task.

Hyperparameters are the settings or configurations that govern the overall behavior of machine learning algorithms. Unlike model parameters, which are learned during training, hyperparameters are set before training begins.

Importance of Hyperparameter Tuning:

1. Optimizing Model Performance: Proper tuning can lead to an optimal model that has the best possible performance metrics, such as accuracy, precision, recall, or F1-score, on unseen data.
2. Avoiding Overfitting and Underfitting: By adjusting hyperparameters, you can control model complexity and make a balance between bias and variance—key to avoiding overfitting and underfitting.
3. Enhancing Generalization: Well-tuned hyperparameters help the model to generalize better from the training data to unseen data, which is the ultimate goal of a predictive model.

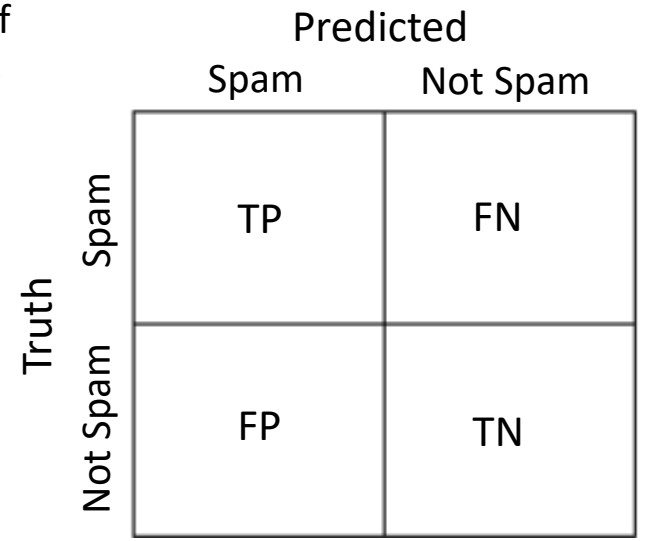
Methods of Hyperparameter Tuning:

1. Grid Search: Searches specified hyperparameter space exhaustively, guided by performance metrics.
2. Random Search: Samples random hyperparameter combinations, more efficient when few hyperparameters impact performance.
3. Bayesian Optimization: Uses probability models and past results to predict and improve model performance.
4. Gradient-based Optimization: Tunes hyperparameters by optimizing the objective function gradient.
5. Evolutionary Algorithms: Applies biological evolution principles like mutation and selection for complex optimizations.

Understanding Metrics in Machine Learning

A **Confusion Matrix** is a table used to describe the performance of a classification model on a set of test data for which the true values are known. It is especially useful for measuring Recall, Precision, Accuracy, and F1-Score.

- True Positives (TP): Correctly predicted positive observations.
- False Positives (FP): Incorrectly predicted as positive, but are truly negative.
- True Negatives (TN): Correctly predicted negative observations.
- False Negatives (FN): Incorrectly predicted as negative, but are truly positive.



Metrics Derived from the Confusion Matrix:

1. Accuracy: The ratio of correctly predicted observations to the total observations.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. Precision: The ratio of correctly predicted positive observations to the total predicted positives.

$$Precision = \frac{TP}{TP + FP}$$

3. Recall (or Sensitivity): The ratio of correctly predicted positive observations to all actual positives.

$$Recall = \frac{TP}{TP + FN}$$

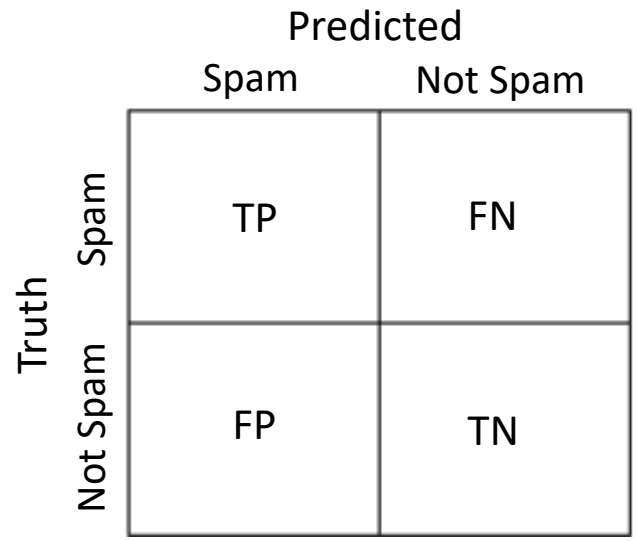
4. F1-Score: The weighted average of Precision and Recall, taking both false positives and false negatives into account.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Practical Example with Visual Aid

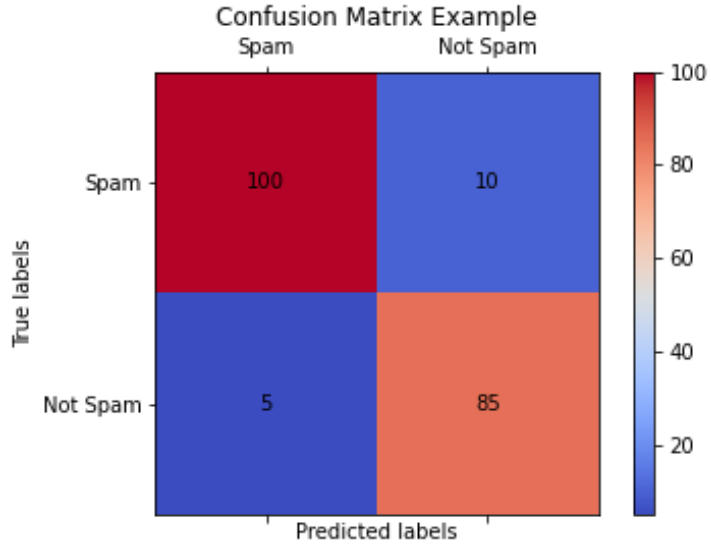
Let's consider a scenario where you are classifying emails (200 emails in total) as "spam" or "not spam“:

- True Positives (TP): An email correctly classified as spam.
- False Positives (FP): An email incorrectly classified as spam (actually not spam).
- True Negatives (TN): An email correctly classified as not spam.
- False Negatives (FN): An email incorrectly classified as not spam (actually spam).



Metrics Derived from the Confusion Matrix:

1. $Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{100+85}{100+85+5+10} = \frac{185}{200} = 0.925 = 92.5\%$
2. $Precision = \frac{TP}{TP+FP} = \frac{100}{100+5} = \frac{100}{105} = 0.952 = 95.2\%$
3. $Recall = \frac{TP}{TP+FN} = \frac{100}{100+10} = \frac{100}{110} = 0.909 = 90.9\%$
4. $F1 - Score = 2 * \frac{Precision*Recall}{Precision+Recall} = 2 * \frac{0.952*0.909}{0.952+0.909} = \frac{1,730736}{1,861} = 0.930 = 93.0\%$

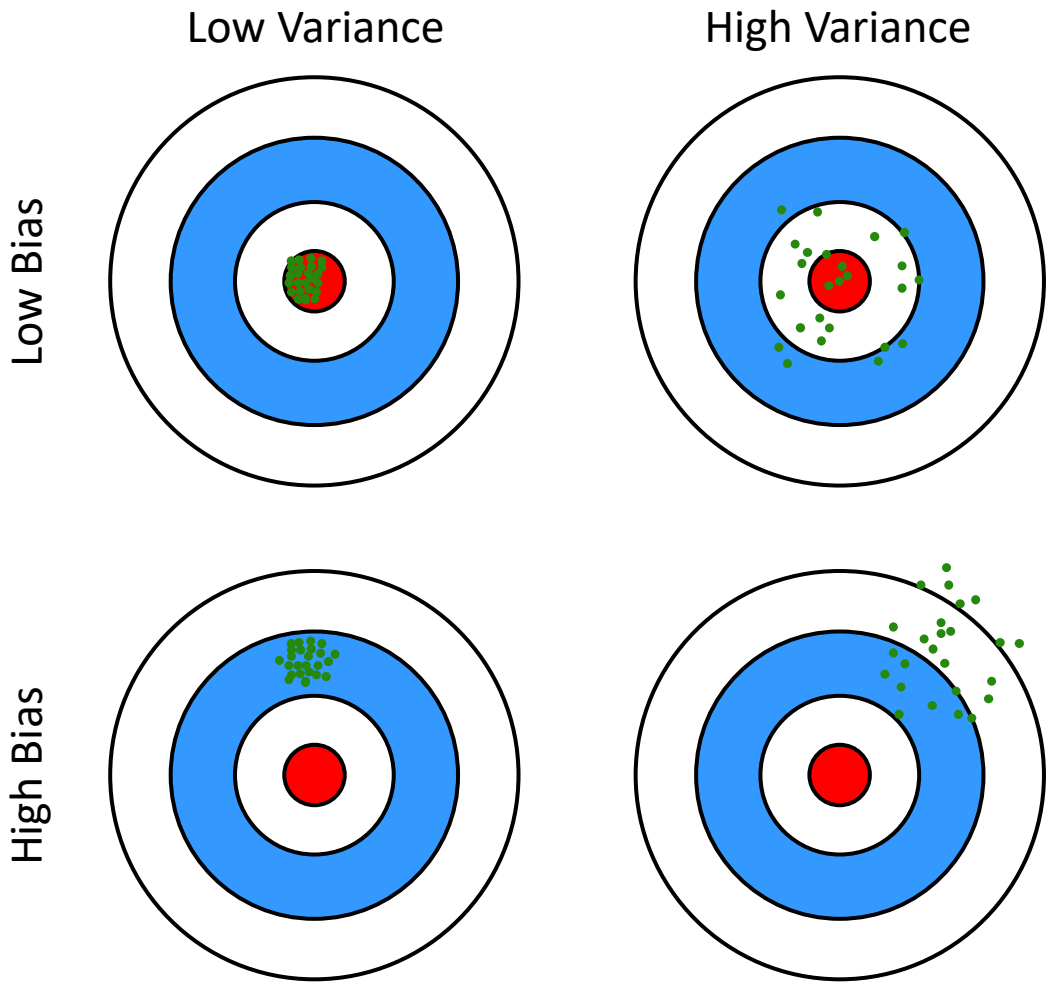


Bias-Variance Tradeoff

The bias-variance tradeoff is a fundamental concept in machine learning that describes the tension between the error introduced by approximating a real-world problem (bias) and the error introduced by random fluctuations in the training data (variance). Understanding and managing this tradeoff is crucial for developing models that generalize well to new, unseen data.

Definition of Bias and Variance:

- Bias: Bias refers to the error that arises when a model makes overly simplistic assumptions in the learning algorithm. High bias can cause an algorithm to miss relevant relations between features and target outputs (underfitting). A model that exhibits small variance and high bias will underfit the target.
- Variance: Variance refers to the error that arises when a model is excessively sensitive to small fluctuations in the training set. High variance can cause modeling noise present in the training data, leading the model to perform well on its training data but poorly on unseen data (overfitting). A model with high variance and little bias will overfit the target.



k-Fold Cross-Validation

k-Fold Cross-Validation is a statistical technique used to evaluate the performance of machine learning models. It is particularly useful for assessing how the results of a statistical analysis will generalize to an independent data set. Essentially, it is used to validate the reliability of a model by partitioning the original data into a set of k equally (or nearly equally) sized subsets or "folds.“

In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Out of these k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation.

Importance of k-Fold Cross-Validation:

1. Model Validation: It provides a robust estimate of the performance of a model on unseen data, unlike a simple train/test split.
2. Mitigating Overfitting: By using multiple iterations and subsets, it ensures that the model is tested across all the data, reducing the risk of overfitting to a particular subset.
3. Bias-Variance Tradeoff: Helps in achieving a good balance by allowing tuning of the model using a more comprehensive range of data points, thus improving the model's generalizability.

How to Use k-Fold Cross-Validation

1. Split the Data: Divide the entire dataset into k equally sized segments or folds.
2. Iterate Through Folds: For each iteration, hold out one fold as the test set and use the remaining folds as the training set.
3. Train and Validate: Train the model on the training set and validate it on the test set. Retain the model’s evaluation score and discard the model.
4. Average the Results: After cycling through all the folds, take the average of the evaluation scores as the overall result.