ML Handbook

Introduction

Machine learning is a subtype of artificial intelligence that enables computers to learn from data without being consciously programmed. Below discusses Linear Discriminant Analysis (LDA) , Support Vector Machines (SVM), and Neural Networks (NN), which are popular machine learning techniques that have lots of real-world applications.

SVM is commonly used for bioinformatics, text classification and image classification. LDA is frequently used for text classification, computer vision and face recognition. NN is widely used for natural language processing, speech recognition, image and video recognition.

This handbook provides basic and technical guides for these three methods.

1. Category 5: Linear Discriminant Analysis

Keywords: Discriminant function, multivariate analysis, classification, feature extraction, dimensionality reduction, covariance matrix, supervised learning, Bayes decision theory, normal distribution

1.1 Basics/Definitions and Illustration of LDA

As the name implies, a linear model for classification and dimensionality reduction is used in linear discriminant analysis. It utilized the most frequently to extract features from situations involving pattern categorization. This has been present for a while. Fisher first developed the linear discriminant in 1936 for two classes, and C.R. Rao later extended it for many classes in 1948. Data from a D dimensional feature space is projected using LDA into a D' $(D > D')$ dimensional space in order to increase variability across classes while minimizing variability within classes. (Kumar)

The below illustration shows three mixed classes are successfully separated by boundaries (blue lines) learnt via mixture discriminant analysis (MDA) which is also one of the strong extensions of LDA. The following are two well-known applications of LDA (and its variants):

1) Bankruptcy prediction: Edward Altman's 1968 model forecasts the likelihood of corporate insolvency. Using data from 31 years, the accuracy was estimated to be between 80% and 90%.

2) Face recognition: While the features learnt from Principal Component Analysis (PCA) are known as Eigenfaces, those learnt via LDA are known as Fisherfaces, named after Sir Ronald Fisher. (Xiaozhou)

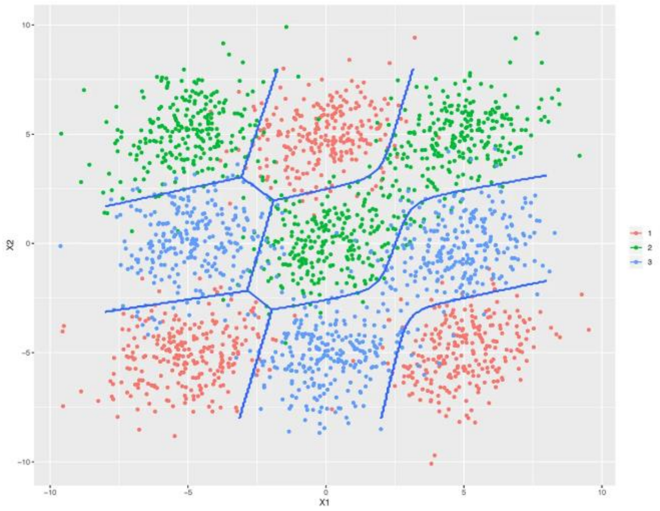


Fig 1.1: Mixture Discriminant Analysis (MDA), extensions of LDA (Xiaozhou)

1.2 Advantages and Disadvantages of using LDA

As mentioned earlier, the advantages and disadvantages of using LDA are as follows.

1.2.1 Advantages

Advantages of LDA include:

1) Simple classifier prototype: It is employed and is easy to comprehend as distance to the class means.
2) Linearity and easy application: The classification is reliable and the decision boundary is linear, making it easy to apply.
3) Dimension reduction: It gives the data an insightful, low-dimensional representation that is helpful for both feature structuring and visualization.

1.2.2 Disadvantages

Disadvantages of LDA include following:

1) The classes may not be sufficiently separated by linear decision boundaries. Support for broader restrictions is desired.
2) LDA employs an excessive number of parameters in high-dimensional settings. It would be ideal if LDA were regularized.
3) Support for a more sophisticated classification of prototypes is desired.

1.3 Equation and Features of LDA

Suppose the feature vectors be X and class label be Y. The Bayes rule states that the best course of action for Y is to select a class that has the highest posterior probability given X if the joint distribution of X and Y exists and if X is known.

The classification method known as generative modeling includes discriminant analysis, where we attempt to calculate the within-class density of X given the class label. The posterior probability of Y can be calculated using the Bayes formula in combination with the prior probability (unconditioned probability) of classes.

Assuming the marginal PMF - probability mass function of the class k is expressed as π_k, with $\sum_{k=1}^{K} \pi_k = 1$, where usually $\pi_k = \frac{\#\ of\ samples\ in\ class\ k}{Total\ \#\ of\ samples}$. As per the Bayes' Rule, the posterior probability can be calculated with

$Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$, a conditional probability of class G given X.

For the Bayes rule for 0-1:

$$\hat{G}\ (X) = arg\ max_k\ f_k(x)\ \pi_k$$

With LDA, assuming the density for X, with every class K following a Gaussian distribution, the density formula for multivariate Gaussian distribution is

$$f_k(x) = \frac{1}{(2\pi)^{p/2}\ |\ \sum_k\ |^{1/2}}\ .exp\ (-\frac{1}{2}x - \mu_k\ )^T \sum_k^{-1}\ (\ x - \mu_k)$$

where p is the dimension and $\sum_k$ is the covariance matrix. The vector X and the mean vector μ_k are both column vectors. For LDA : $\sum_k = \sum, \quad \forall k$

Summarizing the important features of LDA include:

1) Classification: LDA can be used to predict the class of a target based on a set of feature variables.
2) Reduction in Dimensions: LDA reduces the dimensionality of the feature space by projecting it onto a lower-dimensional space while preserving the class separation.
3) Supervised learning: LDA requires labeled training data to learn the discriminant function.
4) Maximization of class separation: LDA maximizes the ratio of between-class variance to within-class variance to find the best linear combination of features that separates the classes.
5) Extraction of feature: LDA can be used to extract relevant features from high-dimensional data, thus, reducing noise and redundancy in the data.
6) Normal distribution assumption: LDA assumes that the distribution of each class is normal and that the covariance matrices are equal.

7) Bayesian decision theory: LDA uses Bayes' theorem to calculate the posterior probabilities of each class.

1.4 Guide

LDA is used to identify a linear combination of features that may most effectively distinguish between two or more classes in a dataset.

1.4.1 Inputs

Inputs of LDA include:

1) A set of predictor variables or features for each observation in the dataset, such as $(x_1, x_2, x_3, \dots, x_n)$.
2) a target variable (y), often known as a categorical variable, identifying the class or group to which each observation belongs.

1.4.2 Outputs

Outputs of LDA include:

1) The predictor variables are combined in a linear fashion using the discriminant function coefficients or weights $(W_1, W_2, W_3, \dots, W_n)$, which optimize the separation between the groups.
2) The dot product of the feature vector (x) and the weights of the discriminant function (W) yields the discriminant score or value (Z) for each observation.
3) Every observation is categorized according to the group that has the highest discriminant score.
4) the classification model's overall accuracy, as determined by measures like the confusion matrix, accuracy score, precision, recall, and F1 score.

1.5 Hyperparameters of LDA

Here is the list of hyperparameters of LDA that need tuning by the user or chosen through a search process.

1) Prior probabilities: They can be used to balance the model's class contributions. The prior probability can, however, be assumed from the data if they are not known.
2) Regularization parameter: Being usually denoted by λ It controls the amount of regularization applied to the model. A high value of λ will result in a more regularized model, while a low value of λ will result in a less regularized model.

3) Scaling method:It is used to standardize the input data to ensure that all the features have equal importance in the model. Standard scaling and min-max scaling are the two most widely used scaling techniques in LDA.
4) Dimensionality reduction method: The number of dimensions can be specified by the user.
5) Solver method: The choice of solver method can affect the computation time and accuracy of the model.

1.6 Computation of LDA

In Python language, in order to compute LDA, the scikit-learn module is required to be imported into our notebook. Below are the codes for necessary modules:

```
`from sklearn.datasets import make_classification`
`from sklearn.discriminant_analysis import LinearDiscriminantAnalysis`
`    `from sklearn.model_selection import GridSearchCV`
```

We use `LinearDiscriminantAnalysis` to train the LDA model on the random dataset generated from the Scikit-learn datasets by using `make_classification` in order to classify the class. We have randomly generated 1,000 observations each with 10 features to compute the LDA in the Python Notebook. After that, we use mean_accuracy to determine the performance of the model. Details about computation are included in the Python notebook.

2. Category 6: Support Vector Machines

Keywords: Hyperplane, classification, regression, supervised learning, margin, support vectors, kernel function, optimization, overfitting, decision boundary

2.1 Basics/Definition and Illustrations

Support Vector Machines (SVMs) are a class of supervised learning algorithms used in regression and classification. The basic idea behind SVMs is to find a hyperplane (decision boundary) that best divides the data points into several groups/classes. Moreover, SVMs use a kernel function to convert the input space into a higher-dimensional feature space in the case of non-linearly separable data so that a linear hyperplane can be applied for classification. Below is the illustration of SVM.

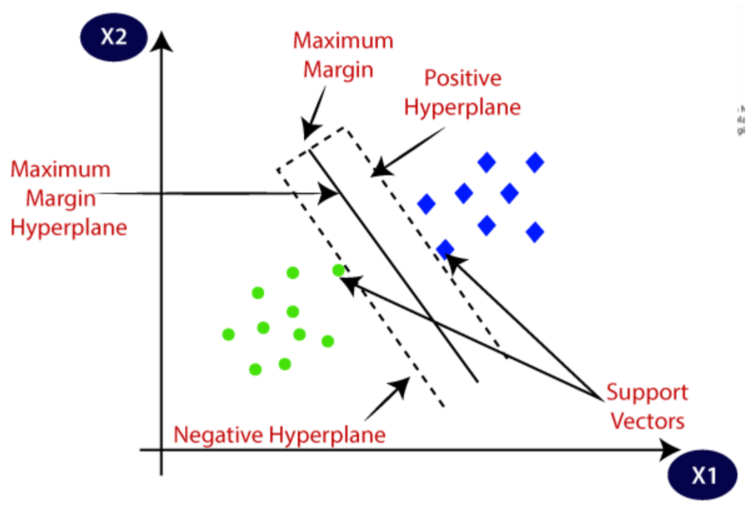


Fig 2.1: Support Vector Machines (TutorialsPoint (JavaTPoint)

The goal of SVM is to maximize the margin distance between the hyperplane and the nearest data points (support vectors). This enables the model to have more robustness to noise/outliers and can enhance its generalization performance. SVMs can effectively handle small to medium-sized datasets and can handle high-dimensional data. (Parna)

Besides classification, SVMs can be utilized for regression analysis by locating a hyperplane that best fits the data points. SVMs are a powerful and popular ML technique, and they are well known for their adaptability, accuracy, and capacity to handle various data types and sizes.

2.2 Advantages and Disadvantages of SVM

SVM methods also possess both advantages and disadvantages when utilizing specific types of data and different situations.

2.2.1 Advantages

Advantages of SVM include -

1) Effectiveness in high-dimensionality: SVMs work well in high-dimensional spaces, which makes them an effective tool for dealing with issues involving several features.
2) Robustness to outliers: SVMs place a strong focus on maximizing the margin between the decision boundary and the closest data points, which makes them less susceptible to outliers than many other machine learning algorithms.
3) Good generalization: SVMs aim to find a decision boundary that best separates the data into classes and maximizes the margin, which can lead to better generalization performance and reduced overfitting.

4) SVM is relatively memory efficient. (K)
5) Flexibility with kernels: SVMs can change the input space and handle non-linearly separable data using a variety of kernel functions, including linear, polynomial, and radial basis functions (RBF). Due to their adaptability, SVMs may be used to solve a variety of issues, making them a popular option in many industries.

2.2.2 Disadvantages

Disadvantages of SVM include -

1) Computationally intensive: SVM algorithm is not very suitable for large data sets, so that is impractical for some real-world applications.
2) Sensitive to kernel choice: The choice of kernel function can hugely affect the performance of SVMs, and selecting the right kernel can sometimes be challenging.
3) Not suitable for non-stationary data: When the target classes are overlapping and the data set includes more noise, SVM does not perform very well. (K)
4) Difficult in interpretability of algorithm: SVMs can be challenging to interpret because they are frequently referred to as "black box" models, which makes it challenging to comprehend how the algorithm came to its conclusion. This could be a concern in situations where clarity and understandability are crucial. (mzc)

2.3 Equations and Features

The hyperplane generated by SVMs can be mathematically expressed by the equation of the straight line. In general for p dimensions is

$$\sum_{j=1}^{p} \beta_j X_j + \beta_0 = 0$$

where X is the set of predictor variables.

Usually for classification problem, instead of class 0 and 1 for the binary problem, supposing the class labels y_i as -1 and 1, a correct classification for the above equation shall become

$$G(X) = y_i \left(\sum_{j=1}^{p} \beta_j X_{i.j} + \beta_0 \right) > 0,$$

vice versa $G(X) \leq 0$ for misclassification, where β is the weights.

Also supposing the perpendicular distance of the hyperplane to the data points (support vectors) as D, the above equation shall become as

$$y_i \left(\sum_{j=1}^{p} \quad \beta_j X_{i.j} \ + \beta_0 \right) > D$$

This method of determining the weights β in an iterative way is known as the Perceptron algorithm. (WQU 1). This Perceptron method is not suitable for non-linearly separable data. However, other SVM algorithms use different kernel functions such as non-linear, polynomial, radial basis function (RBF), sigmoid etc., to process on various dataset. The general Kernel or window function is as follows:

$$K(\underline{X}) = 1, \text{ if } ||\underline{X}|| \leq 1 \text{ or otherwise } K(\underline{X}) = 0$$

where K denotes the kernel.

The polynomial kernel equation, popular with image processing, is

$$K(X_i, X_{i'}) = (X_i . X_{i'} + 1)^d$$

where d is the degree of polynomial.

The other commonly used equation is the Gaussian Radial Basis Function (RBF) and normal Gaussian kernel equation. Normal Gaussian Equation is

$$K(X_i, X_{i'}) = exp\left(-\frac{||X_i - X_{i'}||^2}{2\sigma^2}\right),$$

The Gaussian RBF is

$$K(X_i, X_{i'}) = exp(-\gamma \, ||X_i - X_{i'}||^2), \text{ for}$$

$$\gamma = \frac{1}{2}\sigma^2 > 0$$

where $||X_i - X_{i'}||$ is the euclidean distance between different observation X_i and $X_{i'}$, and γ is the width of the Gaussian distribution or standard deviation.

With the main idea to find the best hyperplane, some important features of SVMs include:

- Hyperplane: It is typically a linear boundary that best separates the data points into different classes, though SVMs can also use non-linear boundaries through the use of kernel functions.
- Types of margins: **Hard Margin** seeks the optimum hyperplane while disallowing all classification errors. With **Soft Margin**, it considers the SVM's tolerance level. In this way, the model is allowed to purposefully misclassify a small number of data points, which helps finding a hyperplane that can generalize results more effectively to new data.
- Support vectors: They are the closest-to-the-hyperplane data points that affect the hyperplane's position. For establishing the decision border and increasing the margin, these considerations are essential.
- Kernel Trick: As mentioned earlier in equations, SVMs allow non-linearly separable data into higher dimensional space so that linear separation method can be applied. (Paolo)

- Decision Boundary enhancing: SVMs algorithm maximizes the margin, making them less sensitive to outliers in the data, which in turn, can lead to better generalization performance and reduced overfitting..

2.4 Guide

Here are the inputs and outputs of the Support Vector Machines.

2.4.1 Inputs

The inputs of a SVM usually depend on the certain situation and the type of SVM being used. However, generally, can be categorized as follows:

1) Training data: This is the supervised learning (labeled) dataset, used to train SVM models. It consists of a number of feature vectors and the labels that go with them. While the labels identify the class that each data point belongs to, the feature vectors represent the qualities of the data points.
2) Kernel function: With the help of this function, the input data can be transformed into a higher-dimensional space where it can be more readily divided into several classes. SVMs can be employed with a variety of kernel functions, including linear, polynomial, and radial basis functions (RBF).
3) Regularization parameter: The trade-off between maximizing the margin (the space between the decision boundary and the nearest data points) and minimizing the classification error is controlled by this tuning parameter. While a low value of the regularization parameter will provide a larger margin and a higher training error, a high value will produce a narrower margin and a lower training error.
4) Other controlled parameters: Other parameters, such as the tolerance for stopping the optimization method and the type of SVM, might also need to be specified.

2.4.2 Outputs

The main output of a Support Vector Machine (SVM) is a decision boundary or a hyperplane. In addition to the hyperplane, some outputs can be produced as follows.

1) Class labels: An SVM can be used to divide fresh data points into one of the two classes once the decision boundary has been established. A new data point's label is determined by the SVM's assessment of which side of the decision boundary it lies.

2) Support vectors: These data points are crucial for identifying the decision boundary because they are the ones that are closest to it. To choose the best hyperplane, these points are chosen during the training phase.
3) Confidence value or classification value: For some SVM types, such probabilistic SVMs, the algorithm can output a probability score or a confidence value for each classification decision, letting the user know how confident the machine is in the classification.

2.5 Hyperparameters of SVMs

Hyperparameters are options that are given by the user or found through a search procedure rather than being learned from the data. A number of hyperparameters in SVM can be changed to modify the model's performance and behavior. The following are some typical hyperparameters for classification trees:

1) Kernel type: The input data can be transformed into a higher-dimensional space using a variety of kernel functions that SVM can use. Radial basis function (RBF) kernel, polynomial kernel, and linear kernel are the three most popular types of kernel functions utilized in SVM.
2) C parameter: Aka, the regularization parameter, chooses the trade-off between minimizing the number of incorrectly categorized training samples and locating the ideal boundary that divides the classes. While a bigger value of C results in a narrower margin and a more complicated decision boundary, a smaller value of C results in a broader margin and a smoother decision boundary.
3) Gamma γ parameter: When using RBF, γ determines the width of the Gaussian kernel and has a significant impact on the decision boundary. A smoother choice border results from a lower value of gamma, whereas a more complex decision boundary results from a greater value.
4) Degree d parameter: When using the polynomial kernel function, the degree of the polynomial function used to convert the input data into a higher-dimensional space is determined by degree d. A decision boundary becomes more complicated as the degree increases.
5) Class weights β: SVM can manage unbalanced datasets by giving various classes different weights. This can be achieved by allocating different classes different weights, with the underrepresented class receiving a higher weight. (scikit-learn)

2.6 Computation of SVMs in Python

In Python language, in order to compute the SVM, the scikit-learn module is also required to be imported into our notebook. Below are the codes for necessary modules:

```
`from sklearn import svm`
`from sklearn.model_selection import train_test_split, GridSearchCV`
`from sklearn.metrics import roc_auc_score, roc_curve`
```

This time, we use `svm` to train SVM models for the provided dataset in order to predict the class of wine from the dataset. This time, we use SVM to classify multi-classification examples. We use wine data from scikit-learn datasets. The dataset is not very big, but its multi-classification to classify the wine class takes a bit different from binary problem as well as the quality metrics consideration.

We will use separating plane illustrations and confusion matrices among classes to determine the quality of the model trained. Although the dataset has the total 13 features on the target class column. We will focus on the first two feature variables to train and compare the models. Details about computation are included in the Python notebook separately attached from this handbook.

3. Category 7: Neural Networks

Keywords: Artificial intelligence (AI), deep learning, convolutional neural networks (CNN), recurrent Neural networks (RNN), backpropagation, autoencoder, neurons

3.1 Basics/Definitions and Illustration of NN

A neural network is a computational system inspired by the structure and function of biological neural networks, such as the human brain. It consists of a large number of interconnected processing elements, known as neurons, which are organized into layers. The neurons in each layer receive input signals from the previous layer, process the input, and transmit the output signals to the next layer. This process continues until the output of the last layer is produced, which represents the final result of the computation.

Neural networks are typically used for tasks that involve pattern recognition, classification, and prediction, such as image and speech recognition, natural language processing, and time series forecasting. They are trained using large amounts of labeled data, through a process called backpropagation, which adjusts the weights and biases of the neurons to minimize the error between the predicted output and the actual output. Neural networks can have different architectures, such as feedforward networks, recurrent networks, and convolutional networks, each suited to different types of tasks.

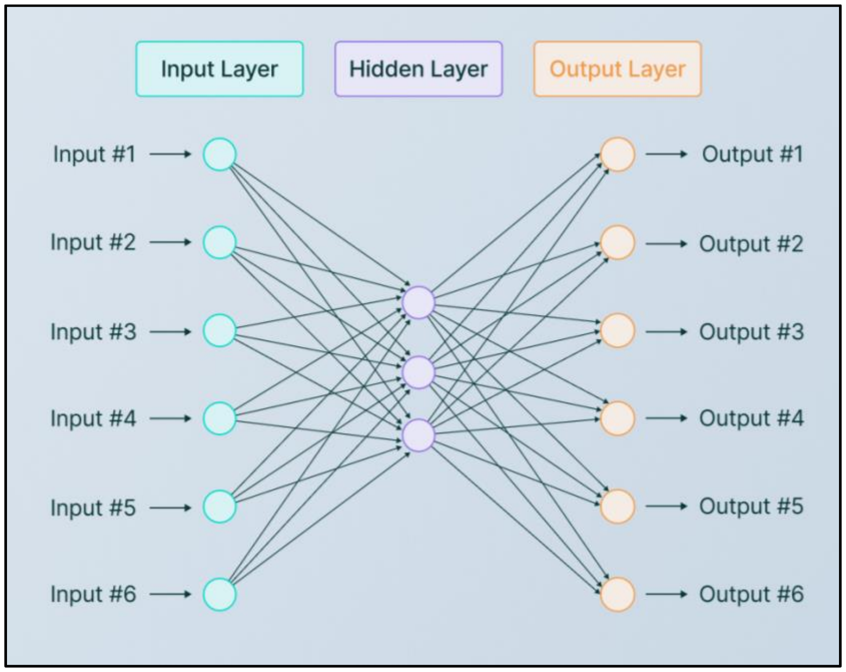


Fig 3.1: Neural Network Working (Baheti)

In this diagram, the neural network has three layers: an input layer, a hidden layer, and an output layer. Each layer is composed of multiple interconnected neurons, represented by the circles. The input layer receives the input data, which is then processed by the hidden layer and eventually produces an output in the output layer.

Each connection between neurons is associated with a weight, represented by the lines connecting them. These weights are learned during the training process to optimize the performance of the network on a specific task.

In a feedforward neural network like this one, the signal flows in one direction only, from the input layer to the output layer, with no feedback connections. The activation of each neuron is determined by applying an activation function to the weighted sum of the inputs. The choice of activation function depends on the specific task and the architecture of the network.

3.2 Advantages and Disadvantages of Neural Networks

Below are the advantages and disadvantages of Neural Networks

3.2.1 Advantages

Advantages of neural networks include:

1) Adaptability: Neural networks can learn from experience and adjust to new inputs. They can generalize from examples and detect patterns that may not be immediately obvious to humans.
2) Non-linearity: Neural networks can model complex, non-linear relationships between inputs and outputs. They are able to capture patterns and dependencies that may be missed by linear models.
3) Parallel processing: Neural networks can perform many computations simultaneously, which makes them well-suited for tasks such as image and speech recognition, where large amounts of data need to be processed quickly.

3.2.2 Disadvantages

Disadvantages of neural networks include

1) Complexity: Neural networks can be complex and difficult to design, implement, and interpret. The number of layers, neurons, and connections can have a significant impact on the model's performance, making it difficult to optimize.
2) Overfitting: Neural networks are prone to overfitting, which occurs when the model is too complex and captures noise in the training data instead of the underlying patterns. Regularization techniques such as dropout and weight decay can help mitigate this issue.
3) High cost: Training a neural network can be computationally expensive, especially for large datasets and complex models.
4) Inability to handle complex relationships and data overlapping: K-means clustering is a linear technique that cannot capture complex nonlinear interactions between variables, usually found in real-world datasets.

3.3 Equations and Features of NN

Some important features of neural networks include:

- Input layer: The first layer of the neural network that takes in the input data.
- Hidden layers: One or more layers between the input and output layers, which perform mathematical operations on the input data.
- Output layer: The final layer of the neural network that produces the output based on the input and the mathematical operations performed on it.
- Activation functions: Non-linear functions that are applied to the output of each neuron in the hidden and output layers to introduce non-linearity into the network.

- Weights: Values assigned to the connections between the neurons that determine the strength of the connection.
- Bias: An additional term added to the output of each neuron to adjust the output to a desired range.
- Backpropagation: The process of adjusting the weights and biases of the neural network during training to minimize the error between the predicted and actual output.
- Gradient descent: An optimization algorithm used to update the weights and biases of the neural network during training.
- Loss function: A function used to measure the difference between the predicted and actual output of the neural network.

Neural networks can be represented by the equation below:

$$output = a(z) = a\left(\left(\sum_{i=1}^{n} X_i \times w_i\right) + b\right)$$

where a is the non-linear activation function, z is the linear output, X is the input, w is the weight, and b is the bias.

3.4 Guide (Inputs/Outputs) of NN

A neural network's inputs and outputs are based on the particular problem being utilized to address.In general, the data points that are being processed or analyzed are the inputs to a neural network, and the predictions or classifications that the network produces as a result of those inputs are the outputs.

For instance, in an image classification task, a digital image represented as an array of pixels could be the input to the neural network, and the output could be a prediction of the object or objects contained in the image (such as "dog," "cat," "tree," etc.). In a task involving natural language processing, the input could consist of a list of words or sentences, and the output could be a sentiment analysis result or a synthetic answer.

Neural networks occasionally may contain many inputs or outputs. In a voice recognition job, for instance, the input may be a sound wave, and the output could be a list of words that were identified or a transcription of the spoken language.

3.5 Hyperparameters of NN

There are several hyperparameters that can affect the performance of neural networks:

1) Number of hidden layers: This hyperparameter determines the depth of the neural network and can impact the model's ability to learn complex patterns.
2) Number of neurons in each layer: This hyperparameter determines the width of the neural network and can impact the model's capacity to learn complex patterns.
3) Learning rate: This hyperparameter determines how much the weights are updated during each iteration of the training process and can impact the model's speed of convergence and ability to avoid local minima.
4) Activation functions: This hyperparameter determines the function used to introduce non-linearity into the model and can impact the model's ability to model complex relationships.
5) Regularization: This hyperparameter determines the penalty term used to reduce overfitting and can impact the model's generalization performance.
6) Dropout rate: This hyperparameter determines the probability of randomly dropping out neurons during training and can impact the model's ability to generalize and reduce overfitting.
7) Batch size: This hyperparameter determines the number of samples used to update the weights during each iteration of the training process and can impact the model's speed of convergence and ability to generalize.
8) Number of epochs: This hyperparameter determines the number of times the model iterates over the entire training dataset and can impact the model's ability to generalize and avoid overfitting.

3.6 Computation of NN in Python

In Python language, the tensorflow module is used to compute NN. Below are the codes for some necessary modules:

```
`import tensorflow as tf`
`from tensorflow import keras`
`from tensorflow.keras.layers import Dense, Dropout`
`from tensorflow.wrappers.scikit_learn import KerasClassifier`
```

Apart from these modules, we import `from sklearn.metrics` a few scoring metrics are imported to determine the quality of the data model.

The dataset used is US Census data which is an extraction of the 1994 census data which was donated to the UC Irvine’s Machine Learning Repository. The data contains approximately 32,000 observations with over 15 variables. The dependent variable in our analysis will be income level and who earns above $50,000 a year. We have selected 4 features for computation including age, education number, gender and working hours per week.

Besides, we have set several parameters including the reLU activation function for the input and hidden layers in the neural network. Since this is a binary classification problem, the sigmoid function is used to classify the target and set cross entropy to be the loss function. Details about computation are included in the Python notebook separately from this handbook.

4. Technical Section

Here, we discuss the process and functions used to code in the Jupyter Notebook.

4.1 Linear Discriminant Analysis (LDA)

Before starting the models, we have generated the random sample datasets from the Scikit-learn by using the following code:

`X, y = make_classification(n_samples=1000, n_features=10, n_informative=10, n_redundant=0, random_state=1)`. The sample dataset is generated with the n_samples : 1,000 (observations) and n_features : 10 (for each observation). After training the model, the evaluation method used is **`cross_val_score`**, for accuracy score. After that, the model is fitted with generated X and y. A new data (row) is introduced as an example of classification to know whether the model actually works or not.

Then, we use **`GridSearchCV`** and **`shrinkage`** to tune the hyperparameters of LDA.

4.2 Support Vector Machines (SVM)

Before we start training the models, most parameters are regarded as default values. We have run the SVC types as linear, polynomial and Gaussian radial basis functions. First of all, we have loaded a dataset about wine from Scikit-learn with the following code.

```
`from sklearn import datasets`
`df = datasets.load_wine()`
```

After doing some basic EDA, the model is divided into two sets, test and train datasets, with only 25% of the test dataset. Then, we import SVC in Python to train the 3 different models with following code

```
`from sklearn.svm import SVC`
```

After training the models, we use the following code to determine the accuracy score of each model.

```
`from sklearn.metrics import accuracy_score`
```

In order to visualize more clearly, we use the first two features to illustrate the separable plane. Moreover, we also use the confusion matrices to determine the quality of each model trained and the best parameter to train on. Lastly, we get to know the classification reports by using `from sklearn.metrics import classification_report` code.

4.3 Neural Networks (NN)

Before starting training the models, we import the necessary dataset "Adult Data.csv" into our Python notebook with the following code.

```
`from google.colab import files`
`uploaded = files.upload()`
`import io`
`data = pd.read_csv(io.BytesIO(uploaded['Adult Data.csv']))`
```

After that, we perform basic EDA to get familiar with the dataset, and select four features for demonstration purposes. Apart from the 'sex' column, other feature columns are of numeric value. Thus, in order to be computationally efficient, we shall convert the 'sex' column value to numeric by regarding 'Male' as 1 and 'Female' as 0. The code is as follows:

```
`df['sex'] = np.where(df['sex'] == ' Male', 1, 0)`
```

Moreover, since our purpose is to find the person with an income level more than $50,000 a year, we shall also convert the target column into numeric value by regarding the income level who earns less than $50,000 a year as 0. The code is as follows:

```
`data['target'] = np.where(data.target == ' <=50K', 0, 1)`
```

Now, we shall split the dataset into two, train and test datasets with test data size as 20%. After that, we select a `keras` model, a sequential function that allows us to specify our neural network architecture. First Layer is dense implying that the features are connected to every single node in the first hidden layer. The code is as follows:

```
`model.add(Dense(128, kernel_initializer="normal", activation="relu", input_dim=4))`
`model.add(Dense(64, kernel_initializer="normal", activation="relu"))`
`model.add(Dense(8, kernel_initializer="normal", activation="relu"))`
```

And, drop out is added to ignore irrelevant neurons The code is as follows:

```
`model.add(Dropout(dropout_rate))`
```

Sigmoid function is also added to classify the target, and the loss function is binary cross entropy because the problem is a binary classification. The codes are as follows:

```
`model.add(Dense(1, activation="sigmoid"))`
`model.compile(loss="binary_crossentropy", optimizer="rmsprop")`
```

After that, we use `KerasClassifier` to classify the target variable by applying previously define model function definition.

In order to check the quality of the model train, we use ROC-AUC-metric, classification report and brier score. Detailed computation is mentioned in the Python notebook.

5. Marketing Alpha

In this section, the computed results are mentioned in accordance with each individual report.

5.1 LDA

After training the model, the very first mean accuracy score obtained by the model using `` `cross_val_score` `` is 0.893 and standard deviation score is only 0.033. Even after testing the model with new data, the model returns predicted class as 1, meaning the model works smoothly. As we try to tune the hyperparameter of the model with `` `GridSearchCV` ``, we find that the default SVD - Singular Value Decomposition works the best compared to other solvers.

Besides, in order to further improve the model performance, we decided to add shrinkage to the model trained, this shall reduce the complexity of the model as well. This time we cross-validate the model with the shrinkage value of 0 to 1 with increment of 0.01, and the solver as `` `lsql` ``. After that, we can see that with a value of 0.02 utilizing shrinkage, performance is slightly improved from 89.3 to 89.4 percent.

5.2 Support Vector Machines (SVM)

After training the models, we compute the accuracy score to determine the quality of the trained models. The generated results of accuracy Scores are RBF as 0.8222, Polynomial as 0.8444 and Linear as 0.9778. Based on the results, we find that the linear model has the best accuracy score of 97.78%. Besides, the first two features were used to illustrate the separable planes for each SVM kernel.

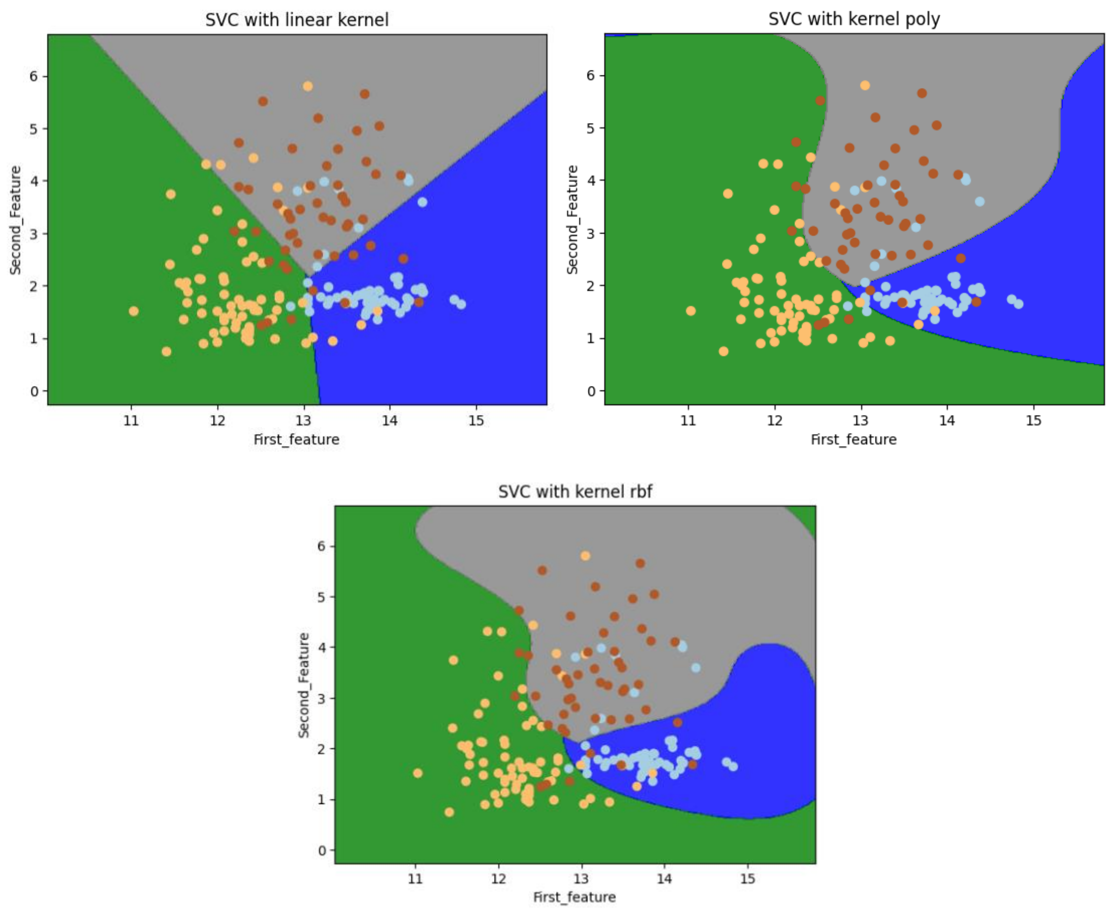
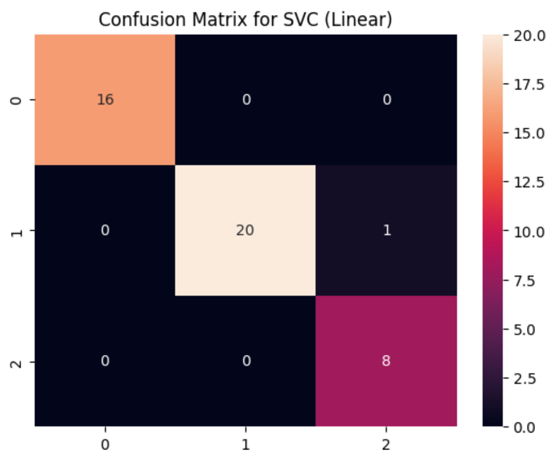
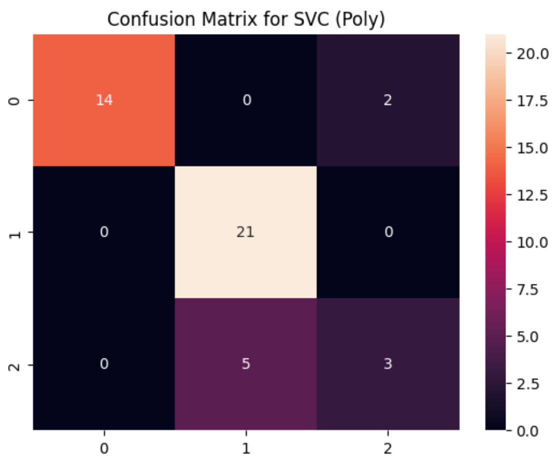


Fig 5.1: SVC plot with different Kernel SVM functions (Python Notebook)

We also illustrate the confusion matrices for all kernel functions to determine the quality of functions. Generated matrices are as follows:





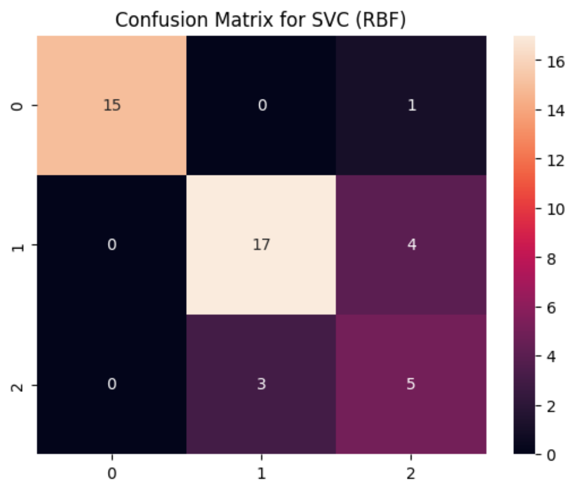
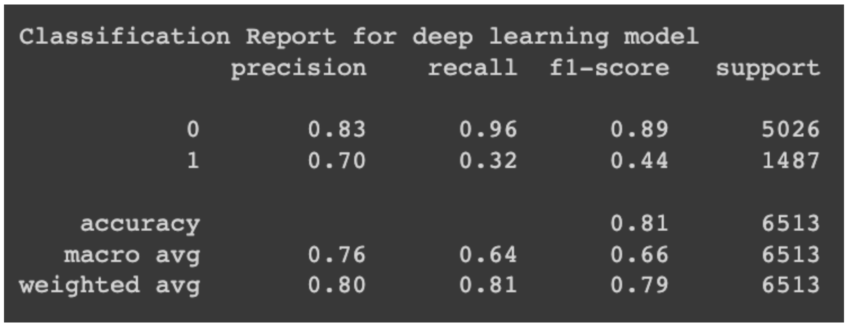


Fig 5.2: Confusion Matrices for difference SVM kernel functions (Python Notebook)

By referring to these figures, we can easily notice that linear kernel function has the best quality results for the dataset. Only 1 out of 45 was misclassified in the linear kernel function while 7 to 8 out of 45 were misclassified in other functions.

5.3 Neural Networks (NN)

After training the model, we use **`roc_auc_score`** to determine the model score. The generated ROC_AUC_Metric is 0.6394, meaning the model's performance is at a better level distinguishing between the positive and negative classes. The generated classification report is as follows:



```
Classification Report for deep learning model
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      5026
           1       0.70      0.32      0.44      1487

    accuracy                           0.81      6513
   macro avg       0.76      0.64      0.66      6513
weighted avg       0.80      0.81      0.79      6513
```

The Brier score for the model is only ~ 0.1867, suggesting that the model has pretty low accuracy of probabilistic predictions. By looking at the confusion matrix, we can see that there exists a lot of misclassification.

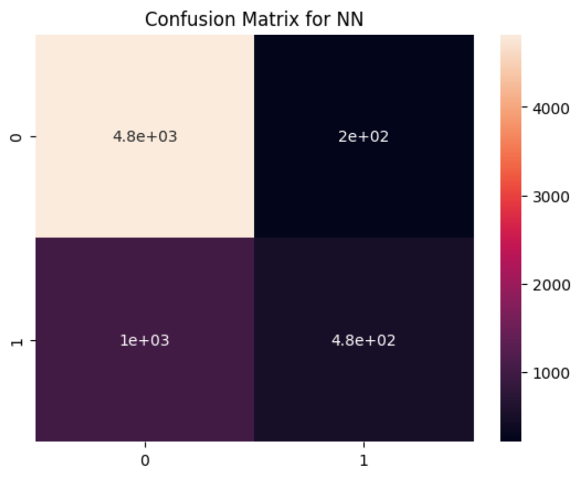


Fig 5.3: Confusion matrix for NN (Python Notebook)

When we try to plot the ROC_AUC curve, the generated illustration is as follows:

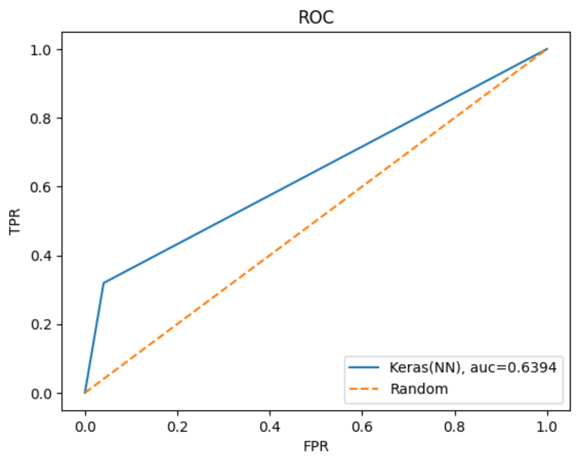


Fig 5.4: ROC Curve and AUC value for Neural Network Example

6. Models Comparison

Here is a comparison table for different machine learning methods, i.e., Linear Regression (LR), Decision Tree (DT), Support Vector Machines (SVM), Neural Networks (NN), K-means clustering (K-Means), Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

Rows are the factors / characteristics of the models and columns are the different types of ML methods.

Factors	LR	DT	SVM	NN	K-Means	PCA	LDA
Accuracy Of Model	Handles Good	Handles Poor	Handles Good	Handles Good	Handles Poor	Not Applicable	Handles Good

Efficient Training Time	Handles Good	Handles Good	Handles Poor	Handles Poor	Handles Good	Handles Good	Handles Good
Handles Outliers	Handles Poor	Handles Poor	Handles Poor	Handles Poor	Handles Good	Handles Good	Handles Poor
Handles Missing Data	Handles Poor	Handles Good	Handles Poor	Handles Poor	Handles Poor	Handles Poor	Handles Poor
Easy to Understand	Handles Good	Handles Good	Handles Poor	Handles Poor	Handles Poor	Handles Good	Handles Good
Large Datasets	Handles Good	Handles Good	Handles Good	Handles Good	Handles Good	Handles Good	Handles Good
Generalize to new data	Handles Poor	Handles Poor	Handles Good	Handles Good	Handles Poor	Handles Good	Handles Good

Table 6.1: Comparing different ML models briefly

"Handles Good" indicates how well the model performs in the relevant category.

"Handles Poor" indicates that the model might underperform in the relevant category.

"Not Applicable" indicates that the model does not apply to the relevant category.

In words,

- Linear Regression performs well in accuracy and efficient training, but may not handle outliers and missing data well. It is easy to understand by humans and can handle large-scale datasets but may not generalize well to new data.
- Decision Tree performs well in efficient training and handling missing data, but may not perform well in accuracy and handling outliers. It is easy to understand by humans and can handle large-scale datasets but may not generalize well to new data.
- SVM performs well in accuracy and handling large-scale datasets, but may not perform well in efficient training and handling outliers and missing data. It may not be easy to understand by humans but can generalize well to new data.
- Neural Networks perform well in accuracy and handling large-scale datasets, but may not perform well in efficient training and handling outliers and missing data. They may not be easy to understand by humans but can generalize well to new data.

- K-means Clustering performs well in efficient training and handling outliers, but may not perform well in accuracy and handling missing data. It may not be easy to understand by humans and may not generalize well to new data.
- PCA is a dimensionality reduction technique and does not apply to categories such as accuracy and handling outliers. It performs well in efficient training, handling outliers, and handling large-scale datasets. It may not handle missing data well but is easy to understand by humans and can generalize well to new data.
- LDA is a classification technique and performs well in accuracy and efficient training. It may not handle outliers and missing data well. It is easy to understand by humans and can generalize well to new data. It can handle large-scale datasets with appropriate implementation.

7. Conclusion

To summarize, support vector machines (SVM), neural networks (NN), and linear discriminant analysis (LDA) are all effective methods for classification jobs. SVM seeks to discover the optimum hyperplane that divides classes, whereas LDA concentrates on finding the best linear combination of characteristics that do so. NNs, on the other hand, mimic the structure and functioning of the human brain to learn from data. Choosing what method to be used usually depends on the type of situations and data characteristics. While neural networks are capable of handling bigger datasets and more challenging issues, LDA and SVM are frequently utilized for small to medium-sized datasets. Each approach has advantages and disadvantages, thus it is important to carefully analyze the issue at hand before choosing the best algorithm.

Cited Journals and Learn More

Below are the cited references and journal articles that are referenced in this handbook in MLA format.

1) Alam, Bashir. “SVM Python - Easy Implementation Of SVM Algorithm 2023.” *Hands-On.Cloud*, 15 January 2022, https://hands-on.cloud/svm-python-tutorial/. Accessed 18 April 2023.
2) Baheti, Pragati. “The Essential Guide to Neural Network Architectures.” *V7 Labs*, 2 March 2023, https://www.v7labs.com/blog/neural-network-architectures-guide. Accessed 18 April 2023.
3) Brownlee, Jason. “Linear Discriminant Analysis With Python - MachineLearningMastery.com.” *Machine Learning Mastery*, 28 September 2020, https://machinelearningmastery.com/linear-discriminant-analysis-with-python/. Accessed 18 April 2023.
4) C. M. Bishop, Neural Networks for Statistical Pattern Recognition (Oxford University Press, Oxford, 1994).
5) DataFlair. “Kernel Functions-Introduction to SVM Kernel & Examples.” *DataFlair*, https://data-flair.training/blogs/svm-kernel-functions/. Accessed 18 April 2023.
6) JavaTPoint. “Support Vector Machine (SVM) Algorithm.” *Javatpoint*, https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm. Accessed 18 April 2023.
7) K, Dhiraj. “Top 4 advantages and disadvantages of Support Vector Machine or SVM.” *Dhiraj K*, 14 June 2019, https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107. Accessed 18 April 2023.
8) Kumar, Sunil. “Linear Discriminant Analysis | What is Linear Discriminant Analysis.” *Analytics Vidhya*, 18 August 2021, https://www.analyticsvidhya.com/blog/2021/08/a-brief-introduction-to-linear-discriminant-analysis/. Accessed 18 April 2023.
9) Liu, Y., Zhang, J., Xu, Y., & Guan, L. (2020). Application of neural networks in predicting the properties of composite materials. Journal of Materials Science & Technology, 47, 8-15.
10) mzc. “Black Box Methods 2 - Support Vector Machines.” *Amazon AWS*, 26 January 2016, https://rstudio-pubs-static.s3.amazonaws.com/147230_ba8b9faaa31347a691d71a84ac61c1db.html. Accessed 18 April 2023.
11) Paolo, Pier. “SVM: Feature Selection and Kernels | by Pier Paolo Ippolito.” *Towards Data Science*, 3 June 2019, https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c. Accessed 18 April 2023.

12) Parna, Goela. “Support vector machine in Machine Learning.” *GeeksforGeeks*, 19 January 2023, https://www.geeksforgeeks.org/support-vector-machine-in-machine-learning/. Accessed 18 April 2023.
13) RoobiyaKhan. “Bankruptcy-Prediction-using-Machine-Learning-Algorithms-in-Python/README.md at master · RoobiyaKhan/Bankruptcy-Prediction-using-Machine-Learning-Algorithms-in-Python.” *GitHub*, https://github.com/RoobiyaKhan/Bankruptcy-Prediction-using-Machine-Learning-Algorithms-in-Python/blob/master/README.md. Accessed 18 April 2023.
14) scikit-learn. “1.4. Support Vector Machines — scikit-learn 1.2.2 documentation.” *Scikit-learn*, https://scikit-learn.org/stable/modules/svm.html#tips-on-practical-use. Accessed 18 April 2023.
15) WQU. *MODULE 5 | LESSON 1 | MACHINE LEARNING IN FINANCE | SUPPORT VECTOR MACHINES*. 2023. *learn.wqu.edu*, WorldQuant University, https://vm.wqu.edu/lab/tree/work/mscfe-machine-learning/module-5/lesson-1/ML_Module_5_Lesson_1.ipynb. Accessed 5 April 2023.
16) Xiaozhou, Yang. “Linear Discriminant Analysis, Explained | by YANG Xiaozhou.” *Towards Data Science*, 9 May 2020, https://towardsdatascience.com/linear-discriminant-analysis-explained-f88be6c1e00b. Accessed 18 April 2023.