**GROUP WORK PROJECT #** 2
**GROUP NUMBER:** 6200

MScFE 632: Machine Learning in Finance

| FULL LEGAL NAME | LOCATION (COUNTRY) | EMAIL ADDRESS | MARK X FOR ANY NON-CONTRIBUTING MEMBER |
|---|---|---|---|
| Boyan Davidov | Bulgaria | davidovg@abv.bg | |
| Ivan Shigolakov | Russia | Shigolakov@yandex.ru | |
| Bharat Swami | India | bharatswami1299@gmail.com | |

| **Statement of integrity:** By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above). | |
|---|---|
| **Team member 1** | |
| **Team member 2** | |
| **Team member 3** | |

Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed.
**Note:** You may be required to provide proof of your outreach to non-contributing members upon request.

# Linear Discriminant Analysis (LDA)

## Basics:

LDA is used to solve multi-class problems with multiple features in supervised learning classification problems. It uses the Bayes Theorem to calculate the probability of a sample input belonging to a class. It follows a generative model framework that uses dimension reduction techniques like PCA (principal component analysis) to separate the multi-class. LDA is also known as NDA (Normal Discriminant Analysis) and DFA (Discriminant Function Analysis).

## Keywords:

Linear Discriminant Analysis, LDA, NDA, DFA, PCA, Bayes Theorem, Probability Distribution, Classification, Dimension Reduction

## Advantages:

- Dimension Reduction: LDA reduces the dimension of the dataset while conserving the variance as much as possible.
- Advance Classification: LDA improves the classification and extends the logistic regression to multi-class with multiple feature problems.
- Feature Selection: LDA selects the features which are most discriminative and improves the model and decrease the overfitting problem.
- Simplicity: LDA is a simple extension of logistic regression which is very simple to understand while applying to real-world problems.
- Probabilistic Approach: LDA uses the baye's theorem to classify an input to a class while using the probability distribution.

## Computation:

Computation is in the attached colab-notebook.

- Data Loading: The sklearn.datasets directory contains the Iris dataset.
- Data Splitting: Training and testing sets are created from the data.
- Standardization: A zero mean and unit variance are applied to the characteristics.
- LDA Model: The training data is used to develop and fit the LDA model.
- Prediction: The test set's classes are predicted using the model.
- Evaluation: The predictions' accuracy and confusion matrix are computed and printed.

## Disadvantages:

- Linearity Assumption: LDA assumes that the multi-class problem can be simplified using a linear combination of multiple features, which makes LDA restricted to the number of problems.
- Normalization: LDA needs normalized data for each feature, which may not alway the case in the real world. This adds one more step for modeling LDA.
- Shared Distribution: LDA mainly works around the mean and variance of a class in a feature, if the mean of two classes overlap make LDA to give misclassification. For example if penguins have features like peak length and width and mean for both are same for two different classes of penguin, then LDA may give misclassification here.
- Multivariable-Normality : LDA assumes that the independent variables are normal for each level of class.

## Equations:

Below are the steps to performing LDA:

1. If each class C has a mean $\mu_i$ and the same covariance $\Sigma$. The scatter of classes can be defined as-

$$\Sigma_b = \frac{1}{C}\sum_{i=1}^{C}(\mu_i - \mu)(\mu_i - \mu)^T$$

where μ is class means

2. Now the separation of class in some direction to vector $w$ is given by

$$S = \frac{w^T\Sigma_b w}{w^T\Sigma\ w}$$ which means w is Eigenvector for $\Sigma^{-1}\Sigma_b$

## Features:

- LDA is used for Supervised learning for linear feature correlations.
- LDA reduces the dimension of data.
- LDA can be used for a dataset with a large number of features with small sample size without overfitting.

## Guide:

- The LDA model takes:

- An array X (training set or predictors) with the size of (N samples, M features)
- Array Y (labels for training set) with the size of N samples

- The output of the model:
    - Returns the coefficient for linear combination of features to predict the output class for sample.

## Hyperparameters:

- solver: The algorithm to use for the computation. 'svd' (default, singular value decomposition), 'lsqr' (Least squares solution), 'eigen' (Eigenvalue decomposition)
- shrinkage: To regularize the covariance estimation..
- priors: prior probabilities for class.
- n_components: Number of components (< n_classes - 1) for dimensionality reduction.
- tol: Threshold used for rank estimation in the svd solver.

## Illustration:

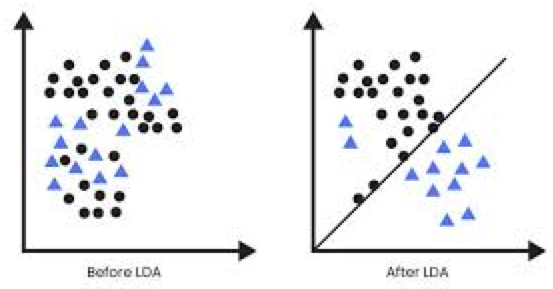


Figure 1

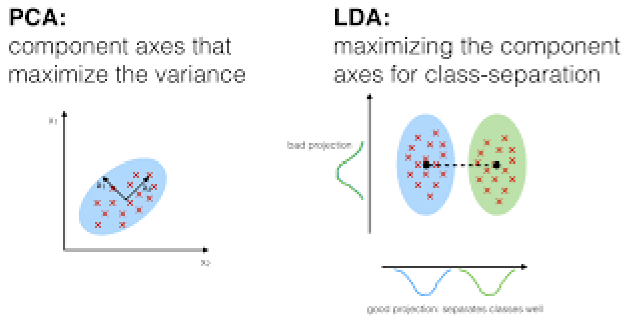

PCA:
component axes that maximize the variance
LDA:
maximizing the component axes for class-separation
bad projection
good projection: separates classes well

Figure 2

## Journal:

May 2017 AI Communications 30(2):169-190, May 201730(2):169-190, DOI:10.3233/AIC-170729 Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim and Aboul Ella Hassanien , Linear discriminant analysis: A detailed tutorial

Above paper provides a brief knowledge and introduction for LDA. LDA is used in a variety of ML model step.Before modeling LDA is used in pre-processing steps to extract information from data about the features and reduce the dimension of data.

## Neural Networks

### Basics

Neural networks (NN) is a machine learning technique that mimics cognitive processes in human's brain. Neurons receive signals, transform them mathematically before sending them to another set of neurons. A signal can be passed through several layers of neurons before the output is created. NN can be used for both supervised and unsupervised models. Classic problems to which they are applied are recognizing images, handwriting, translating from one language to another, anomaly detection etc.

### Keywords :

ArtificialNeuralNetworks (ANN), Short Term Predictions, Long Short Term Memory (LSTM), Dense Layers, Non-linear Dependencies, Prediction Accuracy

### Advantages:

- Complexity and non-linearity: NNs can handle non-linear relationship and by allowing for several layers of neurons they can model complex patterns
- Flexible: Work well with different datasets regardless of size or type of data (financial, climate, etc.)
- Enhanced Prediction Accuracy

### Computation:

Computation part is in the attached colab notebook

### Disadvantages:

- Data might not be stationary (especially financial data that exhibits regime changes). This would lead to non-converging loss functions (it fails to reach minimum value but instead fluctuates, increasing or decreasing with the time). We might end up not having a stable solution or such that performs poorly on unseen data. Such an example is shown below where after 30 epochs the mse (loss function) was still nan:

```
Epoch 30/30
304/304 ━━━━━━━━━━━━━━━━━━━━ 0s 569us/step - loss: nan - mse: nan
```

- There are several parameters that require pre-setup - how many hidden layers, what learning rate, activation function etc.. All of these decisions can influence the final output and require some discretion.
- Black box - it is hard to interpret the output or to trace the steps to it especially when there is complex layer architecture.

## Equations:

Assuming following architecture:

1. Input layer: $X_0$
2. Hidden layer: $X_1$
3. Output layer: Y

Here we describe only the forward propagation technique. In the forward propagation step moving from $X_0$ to $X_1$, we have the following calculations taking place:

$$Y = f(X_0 w_0 + b_0) \qquad (1)$$

where $w_0$ is a matrix of p * n weights for a hidden layer with n nodes, and $f$ is a user specified activation function. The term $b_0$ is a bias term that enables the activation function to shift, which allows more flexibility of the algorithm to fit the data.

To sequentially move to the output layer, we then have the following calculation,

$$Y = f(X_1 w_1 + b_1) = f(f(X_0 w_0 + b_0)w_1 + b_1) \qquad (2)$$

by substituting (1) for $X_1$. The dimension of $w_1$ will be n*m since we have n nodes in the hidden layer and m nodes in the output layer. This sequential process of matrix multiplication results in the following dimension changes:

- $X_0 * w_0$: $(N * p) * (p * n) = N * n$
- $X_1 * w_1$: $(N * n) * (n * m) = N * m$

Training the neural networks essentially mean finding the network weights that achieve the lowest loss. $W^* = argmin_w \frac{1}{n}\sum L(f(X, W), Y)$

## Features:

- It can handle high-dimensional data (eg. many features) and can perform relatively good in problem s where other methods suffer from the curse of dimensionality
- It might not be suitable for time-series data because the timestamp will not play a role in the model

## Guide:

- Input:

  Input layer: $X_0$

  Hidden layer: $X_1$ (or more than one hidden layers eg. $X_{2,3...}$)

  In addition several parameters to be set before training the model

- Output:

  Output layer: Y

  Performance metrics (mae, mse)

## Hyperparameters:

- Learning Rate: Controls the step size during gradient descent.
- Number of Layers: Number of hidden layers in the network.
- Number of Neurons per hidden layer
- Activation Functions
- Epochs: Number of times the entire training dataset is passed through the network.
- Optimizer: How to optimize the weights (e.g., Adam, SGD algorithms).

## Illustration: Visuals (figures, flowcharts, graphs) that show HOW the model works;

A neural network's architecture is shown in the figure below. Input layer transmits the signal (from different features x1, x2, etc.) to the two hidden layers (where the processing takes place), and after the mathematical manipulation an output is created (in our case one single node - eg. price or label)

Figure 1

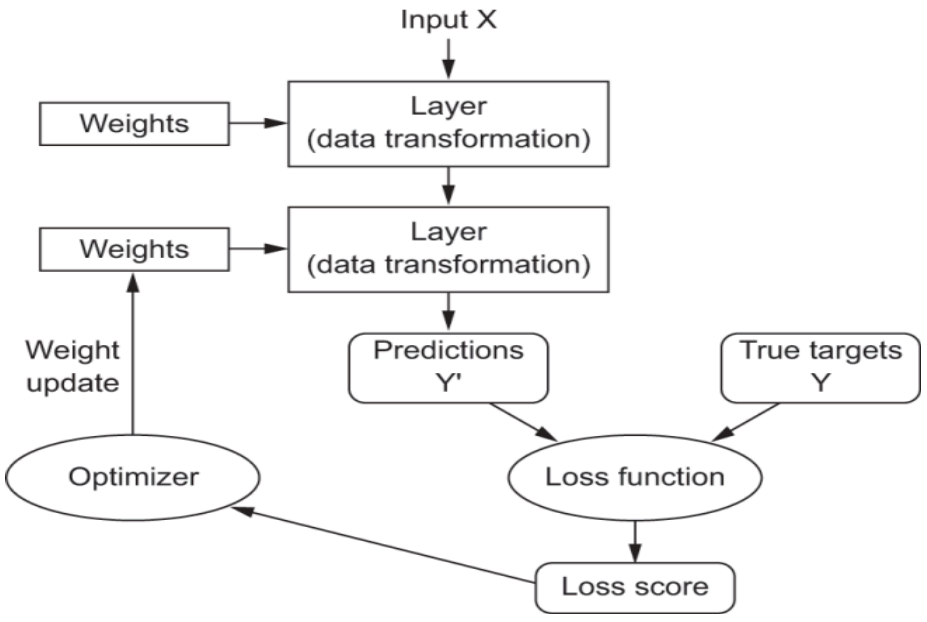

Input X
Weights
Layer (data transformation)
Weights
Layer (data transformation)
Weight update
Predictions Y'
True targets Y
Optimizer
Loss function
Loss score

Figure 3

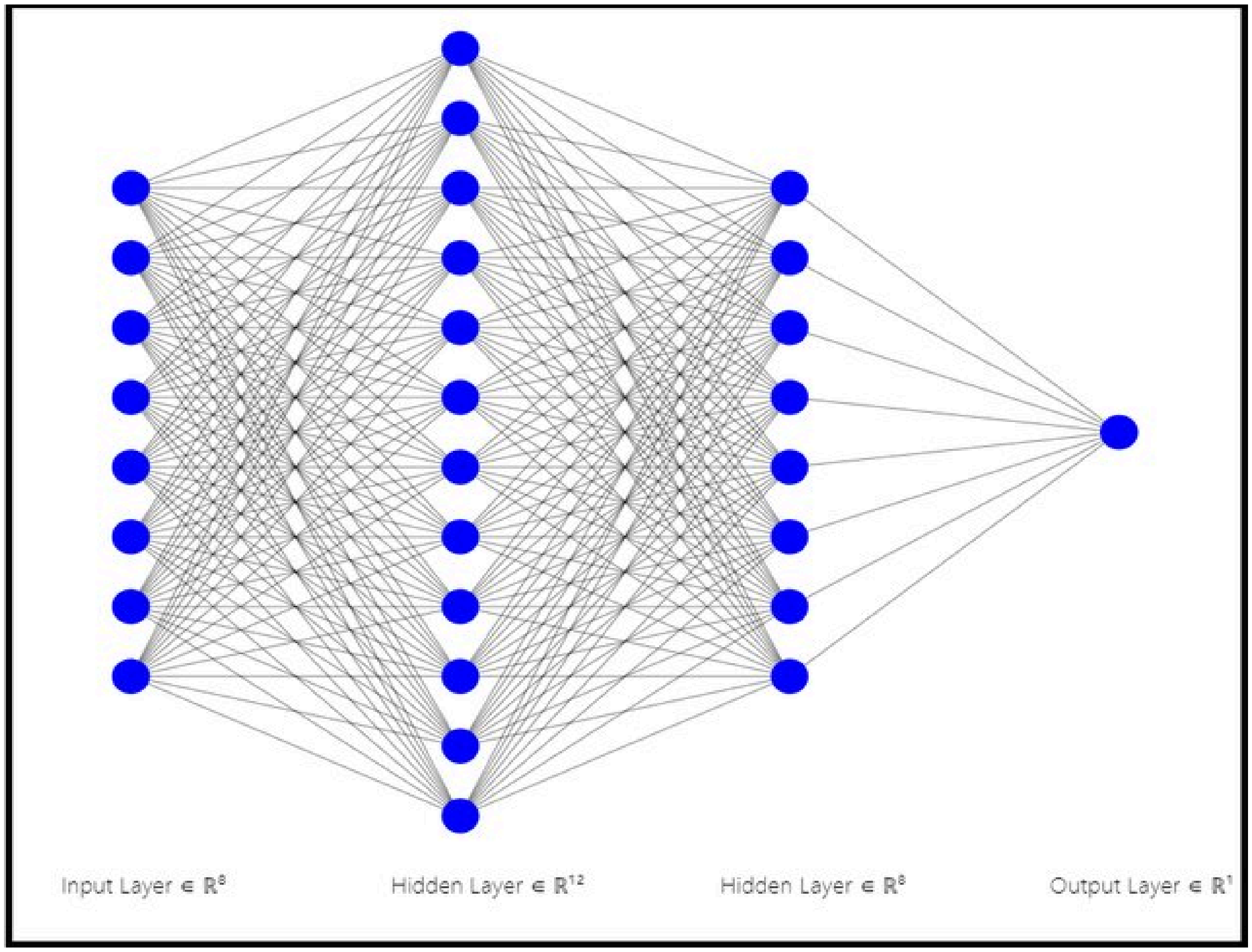
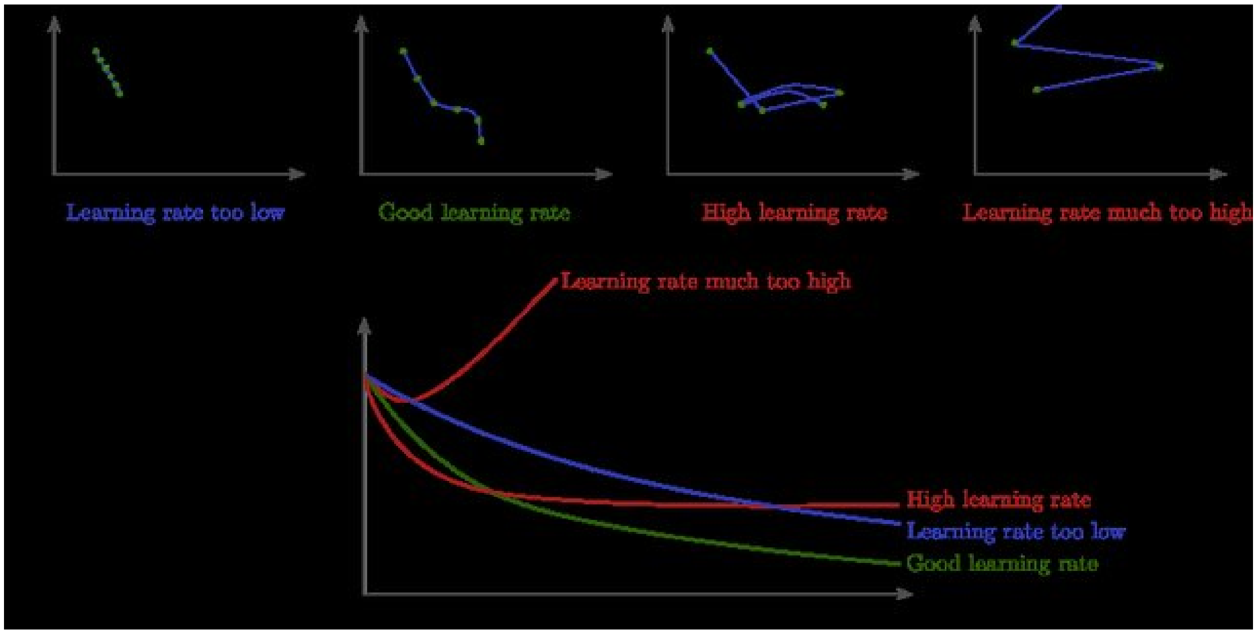

Input Layer ∈ $\mathbb{R}^8$
Hidden Layer ∈ $\mathbb{R}^{12}$
Hidden Layer ∈ $\mathbb{R}^8$
Output Layer ∈ $\mathbb{R}^1$

Figure 4

Important feature of NNs is the setting of learning rate. The image below shows examples of learning rates - in case it is too high (the two charts in the upper right side) the NNs are not converging (i.e. loss function fluctuates)

Figure 5



## Journal:

https://www.researchgate.net/publication/344777084_The_use_and_implementation_of_Artificial_Neural_Networks_in_predicting_directional_movements_of_short-term_cryptocurrency_transactions

Author: Cezary Klimczuk

This is a paper that discusses the implementation of Neural Networks in predicting directional movements of short-term cryptocurrency transactions (BTC/USD and ETH/USD). It is a good read as it describes the architecture and mathematical manipulation in understandable language, at the same time it shows the pitfalls in implementing such model (eg. discussion on learning rates, accuracy of predictions etc.)

## SVM

### Basics:

Support Vector Machine is a supervised machine learning algorithm that can be used for classification and regression tasks. It can be text and image classification, spam detection etc. The main goal of SVM algorithm is to find optimal hyperplane in a N-dim space.

### Keywords:

Support Vector Machines, SVM

### Advantages:

Below are the advantages of using SVM-

- Can be effectively used in high-dimensional space
- Memory-efficient because the algorithm uses only a subset of training data called support vectors
- Different kernel functions can be specified for decision function

### Computation

For the computation purposes we will use the dataset which provides insights into factors influencing hiring decisions (Kaggle). The goal is to classify/predict whether a record (candidate) will be hired or not..

### Disadvantages:

- Not useful for large datasets
- Gets more complex with more features
- Bad performance on high noise
- Does not determine local optima

### Equations:

The first task in constructing the SVM is to determine maximal margin hyperplane.

It's equation: $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p$. The classification rule is determined by the sign of the equation.

To find the maximal margin hyperplane we need to solve an optimization problem: maximize M subject to

$$\sum_{j=1}^{p} \beta^2 = 1 \,,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \geqslant M \quad \forall i = 1,..., n$$

In some cases we can allow the model to classify the data not very strictly. For that purpose a hyperparameter "C" is used. It lets the algorithm misclassify some data points.

In this case we need to solve the following problem:

maximize M subject to

$$\sum_{j=1}^{p} \beta^2 = 1 \,,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \geq M(1 - \epsilon i) \quad \epsilon i \geqslant 0, \quad \sum_{j=1}^{n} \epsilon_i \leqslant C$$

## Features:

Below are the features of the SVM:

- Can be used with complex data
- Robustness to outliers and noise
- Can handle unbalanced datasets

## Guide

- Input parameters:
    - C: (regularization parameter)
    - kernel: 'poly', 'rbf','sigmoid', 'precomputed', 'linear' indicates the kind of kernel that will be applied to the algorithm.
    - degree: polynomial kernel function ('poly') degree: int = 3
    - gamma: {'auto','scale'} Kernel coefficient for poly, sigmoid, and rbf
    - coef0: float = 0
    - shrinking: boolean
    - probability: boolean
    - tol: float = 0.001
    - cache_size: int = 200

  - class_weight: Any
  - verbose: bool
  - max_iter: int = -1
  - decision_function_shape: str = "ovr"
  - break_ties: bool
  - random_state: Any
- Output:
  - classification (prediction) of target variables

## Hyperparameters

Below are the hyperparameters that need tuning in SVM:

- C
- gamma
- kernel

## Illustration

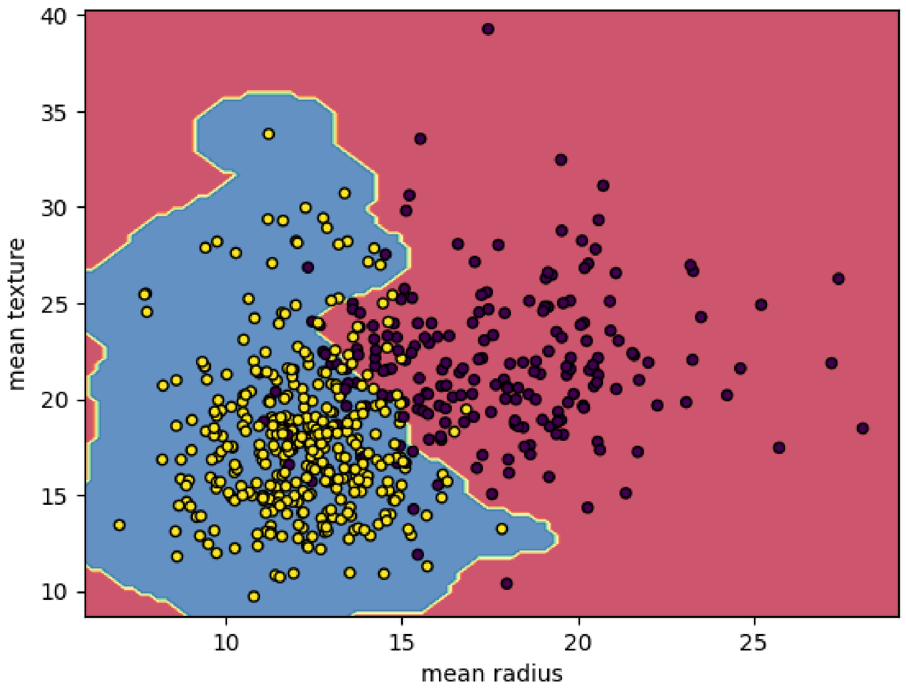


Figure 6

## Journal:

VAN GESTEL, Tony, et al., 2001. https://svms.org/finance/VanGestel-etal2001.pdf

## Technical Section

Support Vector Machine has three hyperparameters, NN has six hyperparameters and LDA has five hyperparameters :

- C (always positive): this is the l2 regularization parameter. The value of C is inversely proportional to the strength of the regularization
- gamma: the kernel coefficient for rbf, poly, and sigmoid.
- kernel: makes the training dataset linearly separable by transforming it into higher dimensions. The Radial Basis Function, or rbf for short, is the default kernel function used in the Python version of the support vector classifier. Additional choices for the kernel function include linear, poly, rbf, sigmoid, and precomputed
- Learning Rate: Controls the step size during gradient descent.
- Number of Layers: Number of hidden layers in the network.
- Number of Neurons per hidden layer
- Activation Functions
- Epochs: Number of times the entire training dataset is passed through the network.
- Optimizer: How to optimize the weights (e.g., Adam, SGD algorithms).
- solver: The algorithm to use for the computation. 'svd' (default, singular value decomposition), 'lsqr' (Least squares solution), 'eigen' (Eigenvalue decomposition)
- shrinkage: To regularize the covariance estimation..
- priors: prior probabilities for class.
- n_components: Number of components (< n_classes - 1) for dimensionality reduction.
- tol: Threshold used for rank estimation in the svd solver.

By changing the hyperparameters (finding the best set) we can obtain better results in searching for a good model. In searching the optimal hyperparameters we will use grid search algorithm.

Computation described in attached notebook.

## Marketing Alpha

The basic features and advantages:

Features:

- Can be used with complex data
- Robustness to outliers and noise
- Can handle unbalanced datasets

Advantages:

- Can be effectively used in high-dimensional space
- Memory-efficient because the algorithm uses only a subset of training data called support vectors
- Different kernel functions can be specified for decision function

As we have already said the Support Vector Machines can be used with data having a difficult relationship. The data can be very complex. To handle the complexity issue SVM algorithm can use different types of kernel function ('linear', 'poly', 'rbf', 'sigmoid', 'precomputed').

If the data have non-linear relationships one can choose a poly-type kernel function to handle non-linearity issues changing one of the hyperparameters - degree (the degree of polynomial function).

Another very important hyperparameter is 'C'. This parameter can not be negative. If we set 'C' equal to zero we would obtain a hard margin meaning that there should not be any misclassifications. Obviously the more the hyperparameter 'C' the less strict the algorithm regarding misclassification. By changing the hyperparameter 'C' we can obtain the bias-variance trade-off.

Neural Networks:

- It is capable of managing high-dimensional data, such as data with several features, and it can function rather well in situations when other approaches are hindered by dimensionality.
- Time-series data may not be a good fit for it because the timestamp is not used in the model.

LDA:

- For linear feature correlations, supervised learning with LDA is utilized.
- LDA lowers the data's dimension.

- Without overfitting, LDA can be applied to a dataset with a large number of features and a small sample size.

## Learn More

- PANG, Bo, Lillian LEE and Shivakumar VAITHYANATHAN, 2002. http://portal.acm.org/citation.cfm%3Fid%3D1118693.1118704
- VAN GESTEL, Tony, et al., 2001. https://svms.org/finance/VanGestel-etal2001.pdf
- TAY, Francis E. H. and Lijuan CAO, 2001. https://dx.doi.org/10.1016/S0305-0483(01)00026-3
- TAY, Francis E. H. and L. J. CAO, 2002. https://svms.org/finance/TayCao2002b.pdf.
- KIM, Kyoung-jae, 2003. https://svms.org/finance/Kim2003.pdf
- Linear Discriminant Analysis (LDA) Dr. Guangliang Chen
- Tharwat, Alaa & Gaber, Tarek & Ibrahim, Abdelhameed & Hassanien, Aboul Ella. (2017). Linear discriminant analysis: A detailed tutorial. Ai Communications. 30. 169-190,. 10.3233/AIC-170729.

## Comparing Models

| Feature | SVM | LDA | Neural Network |
|---|---|---|---|
| Handles Missing Data | Poorly (requires imputation) | Poorly (requires imputation) | Poorly (requires imputation) |
| Handles Multiclass Problems | Yes (one-vs-one or one-vs-all) | Yes | Yes |
| Computational Efficiency | Moderate to High (depends on kernel) | High | Low to Moderate (depends on architecture) |
| Interpretability | Moderate | High | Low |
| Scalability | Moderate | High | Low to Moderate |
| Handle Non-linear Data | Yes (uses kernel trick) | Poorly(linear boundaries) | Yes |
| Regularization | Yes( C parameter) | Yes (shrinkage) | Yes (weight decay, dropout) |
| Hyperparameter tuning | moderate (kernel, C, gamma) | Low (few parameters) | High (many parameters) |
| Sensitivity to outliers | High | High | Moderate |
| Handles High Dimensional Data | Yes (kernel trick) | Yes (up to #classe - 1) | Yes |

## ***References:***

- Figure 1: Source -Aug 3, 2014, Sebastian Raschka, Linear Discriminant Analysis
- Figure 2: Source- Aug 23, 2023, Ambika , Linear Discriminant Analysis (LDA) in Machine Learning: Example, Concept and Applications
- Figure 3,4: Source: [https://www.researchgate.net/publication/344777084_The_use_and_implementation_of_Artificial_Neural_Networks_in_predicting_directional_movements_of_short-term_cryptocurrency_transactions]
- Paul Wilmott: Machine Learning: An Applied Mathematics Introduction
- What is linear discriminant analysis (LDA)?
- Linear Discriminant Analysis (LDA) Dr. Guangliang Chen
- Tharwat, Alaa & Gaber, Tarek & Ibrahim, Abdelhameed & Hassanien, Aboul Ella. (2017). Linear discriminant analysis: A detailed tutorial. Ai Communications. 30. 169-190,. 10.3233/AIC-170729.
- Linear discriminant analysis