**GROUP WORK PROJECT # _2__**
**Group Number: _____7073_______**

MScFE 632: Machine Learning in Finance


| FULL LEGAL NAME | LOCATION (COUNTRY) | EMAIL ADDRESS | MARK X FOR ANY NON-CONTRIBUTING MEMBER |
|---|---|---|---|
| Pulkit Gaur | India | pulkit.gaur.iit@gmail.com | |
| Shivansh Kumar | India | Shivansh.business23@gmail.com | |
| Lawal Nimotalai Abduganiyu | Nigeria | nabduganiyu@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** | **Lawal Nimotalai Abduganiyu** |
| **Team member 2** | **Shivansh Kumar** |
| **Team member 3** | **Pulkit Gaur** |

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

**Task**

---

**Marketing Handbook** for cutting-edge machine learning methods for trading strategies.

**Team Member A:** Linear Discriminant Analysis
**Team Member B:** Support Vector Machines
**Team Member C:** Neural Networks

**Step 2**

---

**Linear Discriminant Analysis :**

**Basics:** Linear Discriminant Analysis (LDA) is a supervised learning algorithm used for classification and dimensionality reduction. Its primary goal is to find a linear combination of features that best separates two or more classes of data.
LDA works by maximizing the distance between the means of the different classes while minimizing the variance within each class. It does this by finding a projection matrix that transforms the original data into a lower-dimensional space. A linear decision boundary is created in this lower-dimensional space that can be used to classify unseen data.

**Keywords:**

- **Class separation**
- **Classification**
- **Dimensionality reduction**
- **Lower-dimensional space**
- **Linear decision boundary**
- **Projection matrix**

**Support Vector Machines :**

**Basics**: Support Vector Machine (SVM) is a supervised learning algorithm that is mostly used for classification tasks, sometimes it can also be used for regression. The mechanism of SVM is to find the optimal hyperplane that separates the data points in multiple classes in a high-dimensional space. The data points closest to the hyperplane are called "Support vectors" and help maximize the margin between classes ("An Idiot's guide to Support vector machines (SVMs)").

**Keywords:**

- **Supervised Learning**
- **Classification**
- **Hyperplane**
- **Support Vectors**
- **Margin maximization**
- **High-Dimensional Space**

**Neural Networks :**

**Basics:** NN is a computational model which is inspired by how human minds work. It has interconnected layers of neurons which adjusts their responses by training data to become better and better and then give improved results

**Keywords:**

- Neurons
- Hidden layer
- Activation functions
- Forward propagation

**Step 3**

---

**Linear Discriminant Analysis :**

- **Advantages:**
  - LDA is relatively simple and straightforward algorithm
  - It is computationally efficient making it suitable for large datasets
  - When LDA projects data onto a lower-dimensional space, the class separability is preserved thereby improving model generalization.
  - LDA outputs probability of each datapoint belonging to a particular class which can be helpful in assessing confidence of predictions.
  - It can handle high-dimensional data which makes it suitable for many real-world applications.

- **Disadvantages:**
  - LDA assumes normality of the data within each class which has a significant impact on the results when this assumption does not hold.
  - Equal covariance matrix is assumed across all classes which can pose misclassification problems when the classes have different variances.
  - Outliers can skew the means and variances used to calculate the boundaries.
  - When there is multicollinearity of the features, the results of LDA can be significantly impacted due to the covariance structure of the data.
  - Interpreting the resulting linear combination of features of LDA can be challenging in high-dimensional spaces.

- **Computation:**
  An LDA algorithm was run on data for NVDA stock returns using returns for Apple, Microsoft, IBM, Cisco and Google as the features. The goal is to predict the returns on NVDA as 1 (when exceeding 2%) and 0 otherwise.
  Please refer to the jupyter notebook for details.

- **Features:**
  - LDA projects the original data onto a lower dimensional space
  - It produces a linear combination of features that best separates the classes
  - It assumes that each class follows a Guassian (normal) distribution.
  - It assumes that all classes have the same covariance matrix
  - It can be sensitive to outliers.

- **Equations:**
  Linear Discriminant Analysis aims at finding a set of linear discriminants that maximize the ratio of between-class variance to within-class variance
  For a two-class Linear Discriminant Analysis: assume $\mu 1$ and $\mu 2$ are the means of the sample classes c1 and c2 before projection, then,the means of c1 and c2 after projection are $\boldsymbol{\mu}'1$ and $\boldsymbol{\mu}'2$ respectively.

$$\mu'_1 = \frac{1}{n}\sum_{x \in c_{1_i}}^{n_1} \nu^T x_i = \nu^T \mu_1$$

  In the same sense,

$$\mu'_2 = \nu^T \mu_2$$

  where $x_i,\ i = 1, 2,..., n$ represent the data points
  $\nu^T$ is the set of unit vectors that discriminate between the two classes.
  To determine the unit vector v that best discriminates between the classes, we solve the problem:

$$max\ (v: ||v|| = 1)\left|\mu'_1 - \mu'_2\right|$$

  The variances for the samples of classes c1 and c2 are given by:

$$s'^2_1 = \sum_{y_i \in c_1} (y_i - \mu_1)^2$$

$$s'^2_2 = \sum_{y_i \in c_2} (y_i - \mu_2)^2$$

  where $y_i = \nu^T x_i$
  The optimal choice of v should be such that:
  - $(\mu'_1 - \mu'_2)^2$ is large and
  - $s'^2_1\ and\ s'^2_2$ are both small

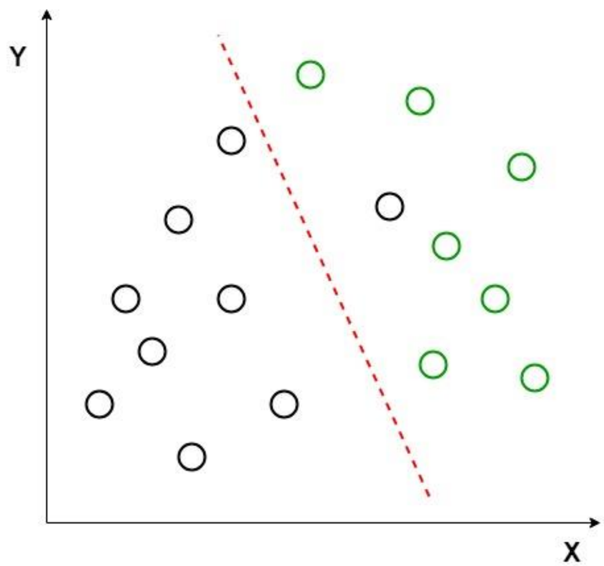
  To achieve this, we solve the problem:

$$max\ (v: ||v|| = 1)\ \frac{(\mu'_1 - \mu'_2)^2}{s'^2_1 + s'^2_2}$$

- **Guides:**
  - **Inputs**
    - Training Data: Features
    - Labels: for each class
  - **Outputs**
    - Predicted class labels for new data points
    - Discriminant score; showing how well each data point fits the predicted class
    - Coefficients of linear combination of features that maximize class separation

- **Hyperparameters:**
  LDA is a simple algorithm that usually needs no hyperparameters, however, we can regularize the model by:
  - Including prior probabilities of the classes if we have some knowledge about the probability of a datapoint belonging to a particular class.
  - Standardizing the features where they have different scales

- **Illustration:**

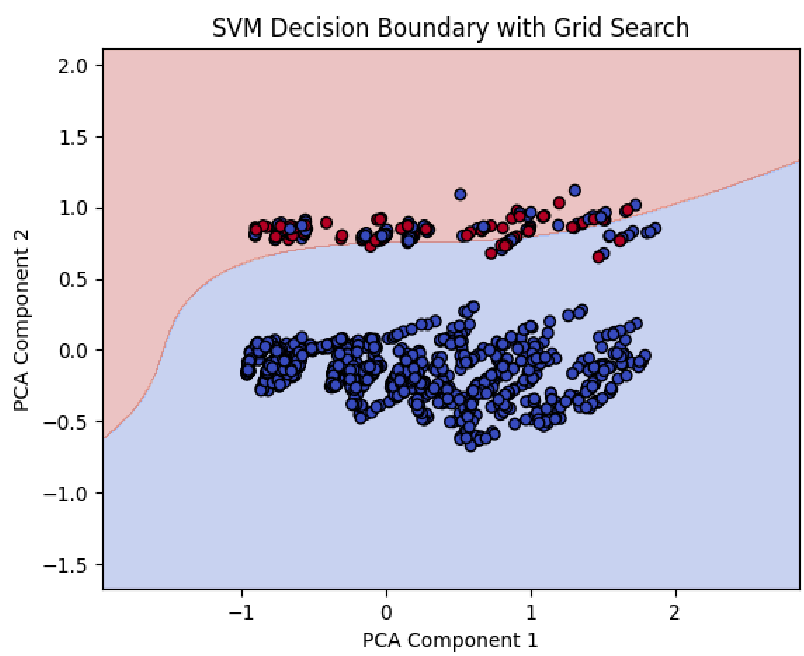



The above illustration displays how LDA uses both X and Y axes to create a new axis that maximizes the separation of the data into two classes and hence, reduces the 2D graph into a 1D graph.

- **Journal:**
  - Tharwat, Alaa et al, May 2017 “Linear discriminant analysis: A detailed tutorial”, ResearchGate.
    **DOI:** [10.3233/AIC-170729]

**Support Vector Machines :**

- **Advantages:**
  - SVM is highly effective in high-dimensional spaces. It why it is very suitable for classification tasks as it can deal with large datasets with n number of features.
  - It is also memory efficient as it uses support vectors in the decision function.
  - It is also very robust to overfitting as it uses the margin maximization principle to achieve a balance between model complexity and generalization.
  - It can also handle imbalance datasets as it focuses on the most important data points i.e. support vectors.
  - It is also very versatile as it can be adapted for both linear and non-linear classification problems using kernel tricks.
- **Computations:**
  - For our computation, we have implemented a simple BTC Price Movement Classification using SVM to predict whether a cryptocurrency's price (e.g., Bitcoin) will move up(1) or down(0) based on historical price data and technical indicators.
  - You can find a detailed report in the colab file submitted with this report, please refer to that.
  - Results: Here in the below image we can see SVM decision boundary results and how it has separated data points.



- **Disadvantages:**
  - Training SVM is very computationally expensive.
  - SVM is very sensitive to the scale of features, it requires feature scaling.
  - It is also hyperparameter sensitive which needs to be tuned properly.
  - Sometimes due to its difficulty to interpret SVM models are often considered as black boxes.
  - SVM is also very sensitive to noise i.e. the dataset with overlapping classes and misclassified points.

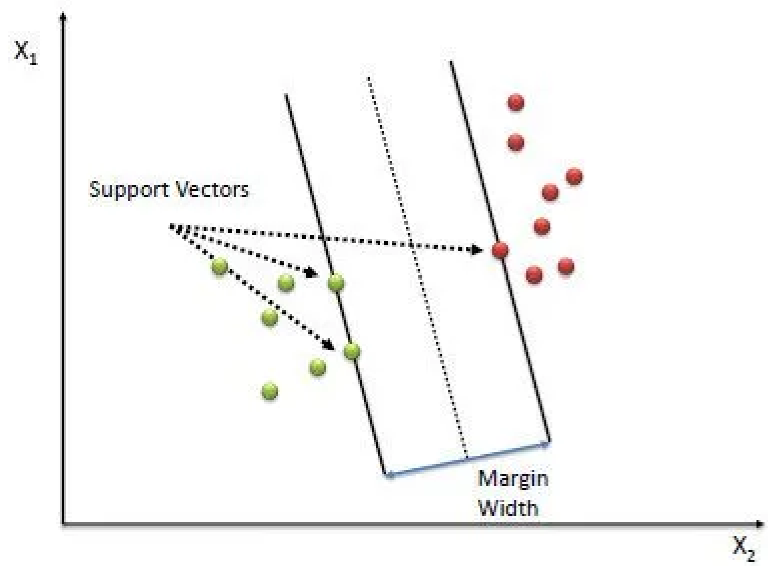
- **Equations:**
  - The main objective of SVM is to find a hyperplane that maximizes the margin between the classes. We can formulate it as follows (“Understanding the mathematics behind Support Vector Machines”):
    - $min_{w,b,\zeta}\ 1/2\ ||w||^2\ +\ C\ \sum_{i-1}^{n} \zeta_i$
    - Subject to :
      - $y_i(w\ .\ x_i\ +\ b)\ \geq\ 1\ -\ \zeta_i\ for\ all\ i$
      - $\zeta_i\ \geq\ 0\ for\ all\ i$
      - Where:
        - W is the weight vector
        - B is the bias term
        - ζ is the slack variable to handle nonseparable cases
        - C is the regularization parameter
        - $y_i$ is the class label of the i-th data point
        - $x_i$ is the feature vector of the i-th data point.
    - Now we will talk about the **Kernel trick,** these are used to handle non-linear classification by mapping the input data into a higher-dimensional space.
    - The kernel function $K(x_i\ ,\ x_j)$ is used and in the original space, it is replaced by the dot product $x_i\ .\ x_j$ .
    - There are many kernel functions but we will here discuss only a few (“Understanding the mathematics behind Support Vector Machines”):
      - **Linear Kernal:** $x_i\ .\ x_j$
      - **Polynomial Kernel:** $(x_i\ .\ x_j\ +\ C)^d$
      - **RBF Kernel:** $exp(-\ \gamma||x_i\ -\ x_j||^2)$
      - **Sigmoid Kernel:** $tanh(kx_i\ .\ x_j\ +\ C)$
    - At last, we will discuss the **Decision function,** which is used for new data to make decisions, so on a new data point x it is given by:

- $f(x) = sign(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b)$
- Where:
  - $\alpha_i$ are the Lagrange multiplier which finds optimal solutions for the SVM's decision boundary.
  - $K(x_i, x)$ is the Kernel function.
  - b is the bias term.

- **Features:**
  - Works well with high-dimensional data.
  - It is very memory efficient as uses only a subset of data "support vectors".
  - Very Versatile as it can handle both linear and non-linear data using kernel functions.
  - Due to margin maximization, SVM is also less prone to overfitting.
  - Requires feature scaling.

- **Guide:**
  - **inputs :**
    - Training Data:
      - Feature Vectors $x_i$ and corresponding class labels $y_i$.
    - Hyperparameters:
      - $C$: Regularization parameter.
      - γ: Kernel coefficient for polynomial, and sigmoid Kernels.
      - Kernel Type: Linear, polynomial, Sigmoid, RBF
      - Degree d: Degree of polynomial kernel
      - Coefficient $c$: Coefficient for polynomial and sigmoid kernels.
  - **Outputs :**
    - Model Parameters:
      - Weight vector $w$
      - Bias term $b$
    - Decision Function:
      - For a new data point $x$, the decision function $f(x)$ outputs the predicted class labels.
- **Hyperparameters:**
  - $C$:
    - Regularization parameter: It controls the trade-off between achieving a low training error and a low testing error.
  - γ:
    - Kernel coefficient for RBF, polynomial, and sigmoid kernels. Controls the influence of individual training examples.
  - Kernel Type:
    - Linear, Polynomial, RBF, Sigmoid. Determines the type of decision boundary.

- Degree $d$:
    - Degree of the polynomial kernel is applicable only to polynomial kernels.
- Coefficient $c$:
    - Coefficient for polynomial and sigmoid kernels

- **Illustration:**
    - 

    - Here in the above Image, we see how an SVM algorithm works, you can see how the hyperplane separates the points and you can also see the points closest to a hyperplane called the support vector and the margin which is the distance between vectors and the hyperplane. Basically, the farther the SV points from the hyperplane, the more likely it is to classify the point in a respective class correctly (Yadav).

- **Journal:**
    - Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting stock market movement direction with support vector machine." Computers & operations research 32.10 (2005): 2513-2522.

**Neural Network:**

- **Advantages:**
    - NN can model complex and nonlinear relationships in data, from image to even speech recognition
    - It is highly scalable
    - It automatically extracts features
    - It can do parallel processing
    - It is very robust and easily combines with other AI techniques
- **Computations:**
    - For our computation, we have implemented a simple BTC Price Movement Classification using Keras NN to predict whether a cryptocurrency's price (e.g., Bitcoin) will move up(1) or down(0) based on historical price data and technical indicators.
    - You can find a detailed report in the colab file submitted with this report, please refer to that.

- **Disadvantages:**
    - Training NN is very computationally expensive.
    - NN requires large amounts of data to work accurately
    - It is also hyperparameter sensitive which needs to be tuned properly.
    - Sometimes it can be overfitted and gives a false sense of security
    - It can depend on the initialization

- **Equations:**
    - **Weighted sum:**
      **For a single neuron:**

$$z = \sum_{i=1}^{n} (w_i \cdot x_i) + b$$

    - **Activation function:**
        - **Sigmoid:**

$$a = \sigma(z) = \frac{1}{1 + e^{-z}}$$

        -
        - **ReLu**

$$a = \mathrm{ReLU}(z) = \max(0, z)$$

- **Loss Function:**

$$L(y,\hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- **Back Propagation:**

$$w^{(l)} = w^{(l)} - \eta \cdot \frac{\partial L}{\partial w^{(l)}}$$

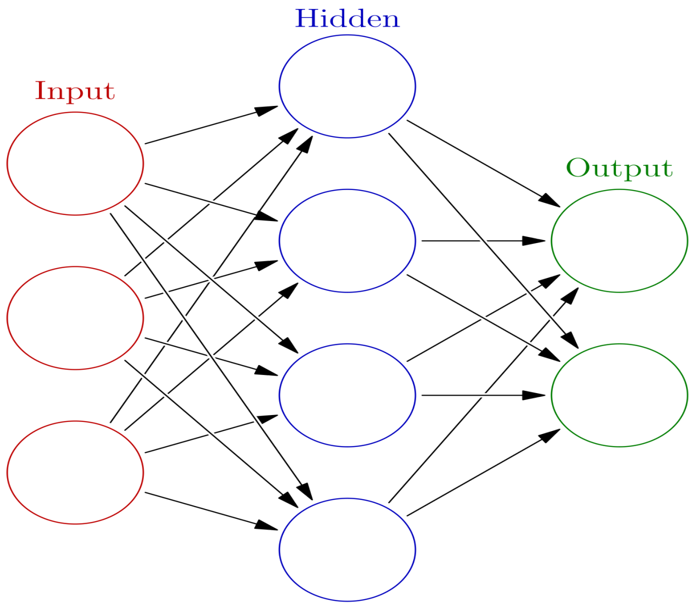
- **Gradient Descent Update:**

$$w^{(l)} = w^{(l)} - \eta \cdot \frac{\partial L}{\partial w^{(l)}}$$

$$b^{(l)} = b^{(l)} - \eta \cdot \frac{\partial L}{\partial b^{(l)}}$$

- **Features:**
  - It has layered architecture
  - It can easily handle non linearity
  - It tries to learn like how human brains learn
  - It gives us a sort of universal approximation making them highly versatile
  - It automatically extracts features
- **Guide:**
  - **inputs :**
    - Training Data:
      - Feature Vectors $x_i$ and corresponding class labels $y_i$.
    - Hyperparameters:
      - Batch Size: the no. of training samples used in 1 iteration
      - No. of epochs: The total no. of complete passes through training data
      - No. of neurons
      - No. of layers
      - Activation functions
      - Initial weights
  - **Outputs :**

    Output is usually an NN model that can perform regression, classifications, complex image generation, etc.

- **Illustration:**
    - 

    - 
    - Here in the above Image, we see how an NN works.

- **Journal:**
    - González-Cortés, D., et al. "The application of artificial neural networks to forecast financial time series." Logic Journal of the IGPL (2024): jzae050.

**Step 4**

**Technical Section:-**

---

- **Linear Discriminant Analysis:**
  The LDA algorithm did not require hyperparameters as it is relatively simple and linear in nature.
- **SVM :**
  - As we have already discussed the multiple hyperparameter that requires tuning for SVM, here we will discuss how we have done the tuning for our model. We have C, kernel, gamma degree, etc as our hyperparameters. All these hyperparameters interact with each other means optimal values depend on other hyperparameters. These are also non-linear and difficult to analyze and we also do not have any closed-form formula to find the optimal hyperparameter, so to counter all of this we need a brute force approach which is done by a search method called GridSearchCV, it will help us explore hyperparameters space for the optimal values.
  - The major advantages of using this method are that it can do an exhaustive search of all combinations of hyperparameters after that it identifies the best combination of hyperparameters do a cross-validation on unseen data and avoid overfitting.
  - Now let's discuss how it works, we first define a hyperparameter grid which is the range for each hyperparameter.
  - e.g: C":[1, 100,1000], "kernel":["linear", "poly", "sigmoid"], 'gamma': ['scale', 'auto'],' degree': [2, 3]
  - Then it splits the data into training and validation sets after that it trains the SVM model on the training set for each hyperparameter combination and evaluates the performance on the validation set in our case accuracy i.e. accuracy: 0.9617117117117117 in our case.
  - At last, it selects the best hyperparameter combination based on the metric i.e. best hyperparameters: {'C': 1000, 'degree': 2, 'gamma': 'scale', 'kernel': 'poly'} and we use that hyperparameter to train our final complete model.

**NN :**

- As we have already discussed the multiple hyperparameter requires tuning for NN. We have designed a custom function that does a grid search for this section just to give the reader an idea of how tuning actually works
- The major advantages of using this method are that it can do an exhaustive search of all combinations of hyperparameters after that it identifies the best combination of hyperparameters do a cross-validation on unseen data and avoid overfitting.
- Now let's discuss how it works, we first define a hyperparameter grid which is the range for each hyperparameter.

- Then it splits the data into training and validation sets after that it trains the SVM model on the training set for each hyperparameter combination and evaluates the performance on the validation set in our case accuracy i.e. accuracy: 0.94 in our case.
- At last, it selects the best hyperparameter combination based on the metric i.e. best hyperparameters: {'neurons': 32, 'optimizer': 'adam', 'batch_size':16} and we use that hyperparameter to train our final complete model.

**Step 5**

**Marketing Alpha:-**

---

- **LDA:**
  - LDA provides an alternative approach to PCA when classification (and not regression) is required.
  - It reduces the dimension of the data without the risk of data loss by projecting the data on a lower-dimensional space.
  - The optimal linear combination of features is the one that maximizes the means of the classes after projection and at the same time minimizes the variances of the classes after projection.
  - As financial time series data is usually non-normal in nature, the accuracy of the LDA model when used to project the NVDA stock returns was about 58%. This is impacted by the normality assumption of the LDA model and the usually non-linear nature of stock returns.

- **SVM :**
  - SVM provides us with several advantages which is why it is a very popular choice for ML tasks, it is highly effective in high-dimensional spaces and robust against noise and outliers. It can also handle non-linear relationships through flexible kernel functions and control overfitting using the regularization parameter (C). Additionally, SVMs have fast training and prediction times, and their interpretability is enhanced by the importance of features. At last, SVMs can handle non-linearly separable data, making them a versatile algorithm.
  - SVM also possesses several key features that contribute to its effectiveness. These include kernel functions (linear, polynomial, RBF, sigmoid), regularization parameter (C), soft margin (slack variables), multi-class classification (one-vs-one, one-vs-all), and regression (SVR) and classification (SVC) variants.
  - They are used in image, text, bioinformatics (protein classification), financial forecasting, medical diagnosis, and natural language processing.

.

- **NN :**
  - NN provides us with flexibility and scaling. It is very popular because of how it automatically extracts the features and is able to do complex human related tasks like speech recognition, image recognition and generation. It is also very effective in real world applications and can be helpful in financial market evaluations.
  - NN has layered architecture which uses various layers to enable complex mappings. It has various kinds of activation functions which are used to allow it to learn complex patterns and it has varieties of architectures like CNN, RNNs and GANs.

- They are used in image, text, financial forecasting, medical diagnosis, and natural language processing, gaming and reinforcement learning, speech recognition, etc.

**Step 6**

**Learn More:-**

---

- **LDA:**
    - **Dr. Guangliang Chen, "Math 253: Mathematical Methods for Data Visualization Linear Discriminant Analysis (LDA)", San José State University.**
        - **https://www.sjsu.edu/faculty/guangliang.chen/Math253S20/lec11lda.pdf**
    - 
    - **Linear Discriminant Analysis in Machine Learning**
        - **https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/**
    - James, Gareth, et al. *An Introduction to Statistical Learning: With Applications in R.* 2nd ed, Springer, 2021. Chapter 4.4.

- **SVM:**
    - Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting stock market movement direction with support vector machine." Computers & operations research 32.10 (2005): 2513-2522.
    - Chen, Shiyi, Wolfgang K. Härdle, and Russ A. Moro. "Modeling default risk with support vector machines." Quantitative Finance 11.1 (2011): 135-154.
    - Ding, Yongsheng, Xinping Song, and Yueming Zen. "Forecasting financial condition of Chinese listed companies based on support vector machine." Expert Systems with Applications 34.4 (2008): 3081-3089.
    - Altan, Aytaç, and Seçkin Karasu. "The effect of kernel values in support vector machine to forecasting performance of financial time series." The Journal of Cognitive Systems 4.1 (2019): 17-21.

- **NN:**
    - A Quantitative Study of Experimental Evaluations of Neural Network Learning Algorithms: Current Research Practice
    - Guan'an Wang, Yang Yang, Tianzhu Zhang, Jian Cheng, Zengguang Hou, Prayag Tiwari, and Hari Mohan Pandey: "Cross-modality paired-images generation and augmentation for RGB-infrared person re-identification," Neural Networks, volume 128, pp. 294-304, August 2020.
    - Xiao-Lei Zhang: "Multilayer bootstrap networks," Neural Networks, volume 103, pp. 29-43, July 2018.

**Step 7**

**Comparing Models:-**

---

| Features | Linear Discriminant analysis | Support Vector machine | Neural Networks |
|---|---|---|---|
| Handles Missing data | Handles Poorly | Handles Poorly | Handles Moderately |
| Scalability | Scales well | Scale Moderately | Scales Moderately |
| Overfitting | Prone to Overfitting | Less prone to overfitting | Prone to Overfitting |
| Outlier sensitivity | Sensitive to outliers | Robust to outliers | Sensitive to outliers |
| Computational complexity | Highly interpretable | can be computationally expensive | Very complex: Black box model |
| Non-linear relationships | Very Limited, works best with linear data | Very Effective by using kernel functions | Highly effective |
| Feature importance | Provide clear feature importance | Limited feature importance | Limited feature importance |
| Multiclass Classification | Natural Support for Multiclass classification | Requires 1 v 1 or 1 v multi-approach | Natural for multiclass classification |
| Regularization | Limited uses covariance matrix instead | Built-in regularization | Very effective (L1, L2, Dropout) regularization methods |

**References**

1. “An Idiot's guide to Support vector machines (SVMs).” *MIT*, https://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf. Accessed 27 September 2024.
2. “Understanding the mathematics behind Support Vector Machines.” *Shuzhan Fan*, 7 May 2018, https://shuzhanfan.github.io/2018/05/understanding-mathematics-behind-support-vector-machines/. Accessed 27 September 2024.
3. Yadav, Ajay. “SUPPORT VECTOR MACHINES(SVM). Introduction: All you need to know… | by Ajay Yadav.” *Towards Data Science*, 20 October 2018, https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589. Accessed 27 September 2024.
4. Sckit learn: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
5. https://academic.oup.com/jigpal/advance-article-abstract/doi/10.1093/jigpal/jzae050/7665617
6. https://adriangb.com/scikeras/stable/generated/scikeras.wrappers.KerasClassifier.html
7. https://www.geeksforgeeks.org/hyperparameter-tuning-using-gridsearchcv-and-kerasclassifier/