GROUP WORK PROJECT #2
GROUP NUMBER: 6127

MScFE 632: Machine Learning in Finance

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Yang Xue	China	KierenXue@gmail.com	
Carlos García Guillamón	Spain	carlosgguil@outlook.es	
Ebenezer Yeboah	Ghana	ebenezeryeboah46@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	**Yang Xue**
Team member 2	**Carlos García Guillamón**
Team member 3	**Ebenezer Yeboah**

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.

Step 2 - Basics and keywords

- Category 5: Linear Discriminant Analysis

BASICS

Similarly to PCA, Linear Discriminant Analysis (LDA) is a tool for dimensionality reduction. As in the first one, LDA aims at finding a linear relation between a reduced set of manifolds with respect to the original set of features representing the data. The main difference is that while PCA tries to find the coordinates based on maximizing the variance and can be applied to unlabelled data, LDA aims at finding the best coordinate system that can find the difference between the classes of data and, therefore, group them for classification. Therefore, LDA is a supervised learning algorithm.

In LDA, data must be standardized before its treatment with LDA. Also, a distinction must be made between inputs and outputs, and then Bayes classifiers are applied. Then, to find a suitable set of coordinates, LDA makes use of eigenvalues and eigenvectors (eigenproblem) to find orthonormal coordinates (similarly to PCA).

KEYWORDS

- Dimensionality reduction
- Supervised learning
- Classification
- Bayes classifiers
- Eigenvalues
- Eigenvectors

- Category 6: Support Vector Machines

BASICS

Support Vector Machines (SVMs) are supervised learning models used for classification and regression analysis. They are particularly effective in high-dimensional spaces and situations where the number of dimensions exceeds the number of samples. The core idea behind SVMs is to find the optimal hyperplane that best separates the data into different classes.

Under Linear SVM, the best linear separating hyperplane that divides the dataset into two classes is aimed to be found. The best hyperplane is the one that maximizes the margin, which is the distance between the hyperplane and the nearest data points from either class, called support vectors.

The Non-Linear SVM extends the linear model by applying a kernel trick. Kernels are functions that transform the data into a higher-dimensional space where a linear separator can be found.

Support Vector Regression (SVR) finds a function that approximates the relationship between input features and continuous target values. The objective is to ensure that most data points lie within a certain margin of the predicted value.

KEYWORDS

- Supervised Learning
- Classification
- Regression
- Hyperplane
- Support Vectors
- Kernel
- Regularization Parameter ©

- Category 7: Neural Networks

BASICS

Neural Networks are a type of machine-learning model inspired by the human brain. Neural Networks consist of layers of interconnected nodes which are called neurons, and they work together to generate solid models to make reasonable predictions.

There are three layers to make neural networks work, which are the input, hidden, and output layers. The input layer is the first layer to handle the raw data; the hidden layer is the layer after the input layer, which performs computation. In addition, there can be various hidden layers between the input and the output layer. The output layer, as its name, produces the final result. In each layer, there are many nodes which are called neurons.

Now, neural networks are widely used in many applications such as game playing, image recognition, and natural language processes. Also, some applications in finance such as stock price regression and currency index regression.

KEYWORDS

- Neural Networks
- Deep learning
- Layer and Node
- Keras, Tensorflow, PyTorch
- Hidden layers
- Neurons

Step 3

Category 5: Linear Discriminant Analysis

- Advantages: LDA is simple to implement and computationally cheap. This is an advantage when dealing with high-dimensional data, since other classification and/or regression methods might greatly increase their computational cost. Also, LDA can deal with multicollinearity, as through dimensionality reduction it can retrieve most of the original information.
- Computation: code can be found in the 'Step 3' section of the associated jupyter notebook. The application case is German credit data to state whether a credit risk is good or bad.
- Disadvantages: LDA is not suitable for unlabelled data, since it is a supervised learning algorithm. It assumes that data is normally distributed, so it is not suitable for cases where data might not follow this pattern. Also, linearity is assumed between the class labers and features: if this does not hold, the algorithm might not perform well.
- Equations: Consider a matrix with features **X** = {X_1, X_2, … X_p}, which have previously being standardized (i.e. centered to 0). The objective is to find a linear combination between these features and some variables, named the linear discriminants, which can be used for classification of the original data. Let's consider a first linear discriminant Y_1, which would be related to the original features as follows:

$$Y_1 = w_{11}X_1 + w_{21}X_2 + \ldots + w_{p1}X_{1p}$$

where w_{ij} represents the different weights. In matrix form, the linear relation could be represented as:

$$\mathbf{Y = X\,W}$$

with **W** being the matrix containing the weights. Let's consider now a 2-class problem. In order to find the weight matrix, a measured S is used, which is defined as the ratio between the between-classes variance and the in-class variance:

$$S = \frac{\sigma^2_{between}}{\sigma^2_{within}} = \frac{(w\,(\mu_1 - \mu_2))^2}{w^T(\Sigma_1 - \Sigma_2)w}$$

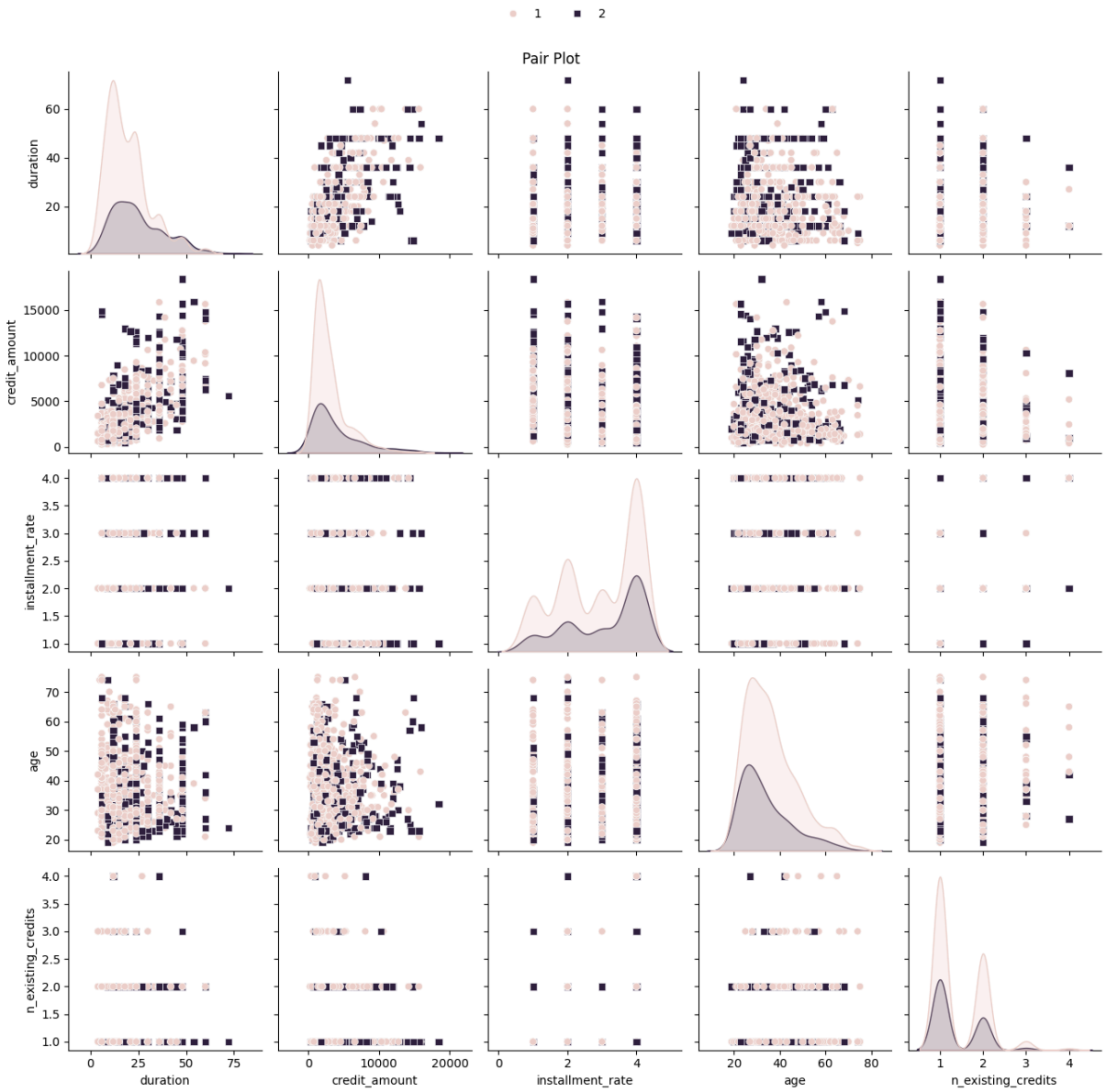
Where subscripts 1 and 2 refers to the two classes, μ_i is the mean for the observations of class i, and Σ_i are the covariances. Therefore, the objective is to find the matrix w (the weights) such that the distance S is maximized. This problem can also be extended to a multi-class problem.

- Features: the model is computationally fast, easy to implement, and works for two-class and multi-class types of problems.
- Guide:
  - Inputs: the feature data X, and target variable y. For example, y data in a two-class problem could be whether a loan debtor has incurred default or not, and the feature data would be the characteristics of the loan (age, income, etc.)
  - Outputs: variances, linear discriminant components, and loading matrix
- Hyperparameters:
  - Obtention method for weighting function (can be an optimizer, or singular value decomposition to resolve an eigenproblem).
  - In multi-class problems: number of linear components to retrieve.
- Illustration: We'll illustrate this with the German credit data problem dealt in the jupyter notebook (the dataset can be found at this link). 5 features are retrieved from the debtors: credit duration, credit amount, installment rate, age and number of existing credits. The output class is whether it is a good credit (y = 1) or not (y = 2). An example of the dataset is:

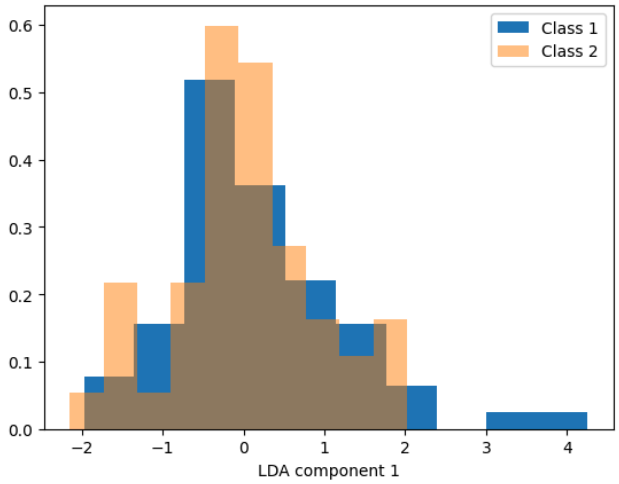
	duration	credit_amount	installment_rate	age	n_existing_credits	class
0	6	1169	4	67	2	1
1	48	5951	2	22	1	2
2	12	2096	2	49	1	1
3	42	7882	2	45	1	1
4	24	4870	3	53	2	2
..	...	...	...	...	...	...
995	12	1736	3	31	1	1
996	30	3857	4	40	1	1
997	12	804	4	38	1	1
998	45	1845	4	23	1	2
999	45	4576	3	27	1	1

[1000 rows x 6 columns]

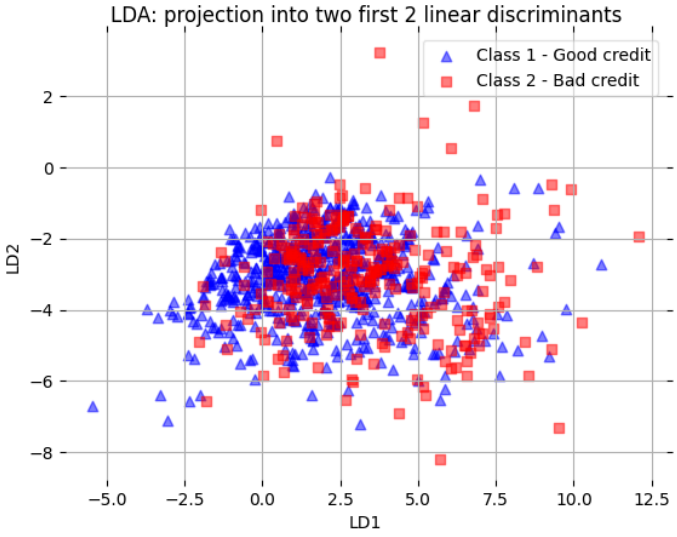
Relations among data can be shown through a pair plot as follows:



where the black dots are class 1, and the black ones are class 2. The PDFs show that in many cases, the data distribution is not normal, which could indicate that LDA is not suitable for application in this dataset. Let's test this hypothesis by applying LDA to this case, with two output classes and 5 features. Since there are two classes, the objective is to find a LD component (LD1) that would separate both data. By applying the algorithm (see associated jupyter notebook), the resulting distribution of LD1 conditioned on the classes is found:



Which, as observed due to the overlapping, is not useful for differentiating between classes (unless the LD component is larger than a value of 2, in which case classification Class 1 is predicted). Another test can be performed by applying multi-class LDA instead. In this case, two LD components (LD1, LD2) are calculated, and the projected dataset in the LD1-LD2 space is as follows:



From which, again, no clear classification can be deduced. It can be concluded that this datased might not be a good candidate problem to apply LDA, maybe due to non-normality of data or non-linearity between variables (further tests should be done to assess this). Other classification algorithms should be tested instead.

- Journal: Mishraz, N. Ashok, S. Tandon, D., 2021. 'Predicting Financial Distress in the Indian Banking Sector: A Comparative Study Between the Logistic Regression, LDA and ANN models'. Global Business Review, 1-19. Available online.

Category 6: Support Vector Machines

- **Advantages**
  - ❖ SVMs are particularly powerful in cases where the number of dimensions is greater than the number of samples. They handle high-dimensional data very well.
  - ❖ SVMs are robust to overfitting. SVMs have a regularization parameter (C) that helps in avoiding overfitting, especially when there is a clear margin of separation between classes.
  - ❖ SVMs can be effective even when the number of samples is small compared to the number of dimensions, making them suitable for applications like text classification and gene expression data analysis.
  - ❖ Its versatility with kernel functions allows SVMs to handle non-linear classification by transforming the data into a higher-dimensional space where a linear separator can be found

- **Computation** - Code is available in the attached Jupyter Notebook under step 3

- **Disadvantages**
  - ❖ SVMs can be computationally intensive, especially with large datasets. This makes SVMs less suitable for extremely large datasets.
  - ❖ Choosing the appropriate kernel and its parameters can be difficult and requires domain knowledge or extensive experimentation.
  - ❖ SVMs have several hyperparameters that need to be tuned, often requiring cross-validation which can be time-consuming.
  - ❖ SVMs do not scale well to large-scale problems with millions of samples and features
  - ❖ SVMs are sensitive to noise and outliers because these can significantly affect the position of the hyperplane.

- **Equations**
  - ❖ Decision function:

$$f(x) = w^{t} + b$$

  - ❖ Kernel trick

$$f(x) = \sum_{i=1}^{n} aiyiK(xi, x) + b$$

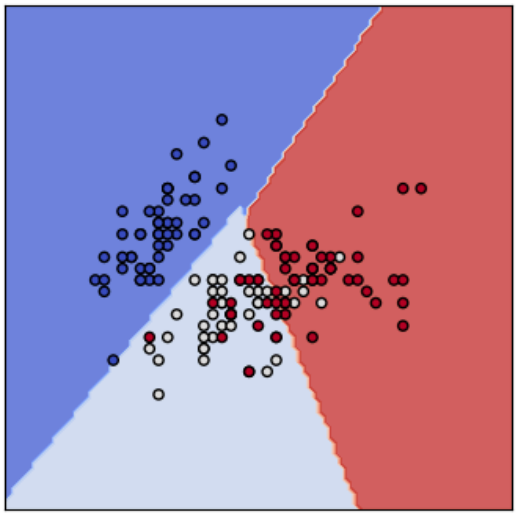
- **Features**
  - ❖ SVM is suitable for high-dimensional data as it performs well when the number of features is large.
  - ❖ SVM aims to maximize the margin between different classes, which generally leads to better generalization performance on unseen data.
  - ❖ SVM can be adapted to handle various types of data including binary classification.
  - ❖ The margin maximization and regularization framework of SVM makes it somewhat robust to outliers.
  - ❖ By using non-linear kernels, SVM can effectively model complex decision boundaries that are not possible with linear models.

- **Guide**
  - ❖ Inputs - Training Data, Labels, Kernel, Regularization Parameter C
  - ❖ Outputs - Support vectors, model, intercept, decision function

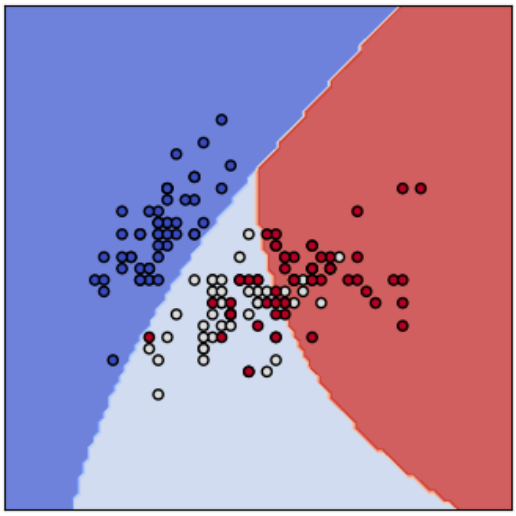
- **Hyperparameters**
  - ❖ Regularization Parameter ©
  - ❖ Kernel

- **Illustration**

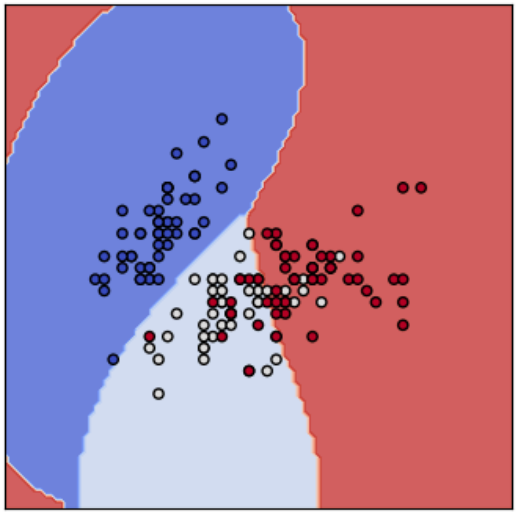
SVC with linear kernel



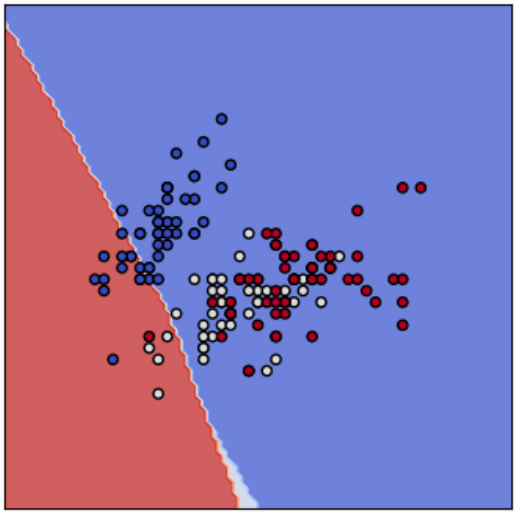
SVC with polynomial (degree 3) kernel

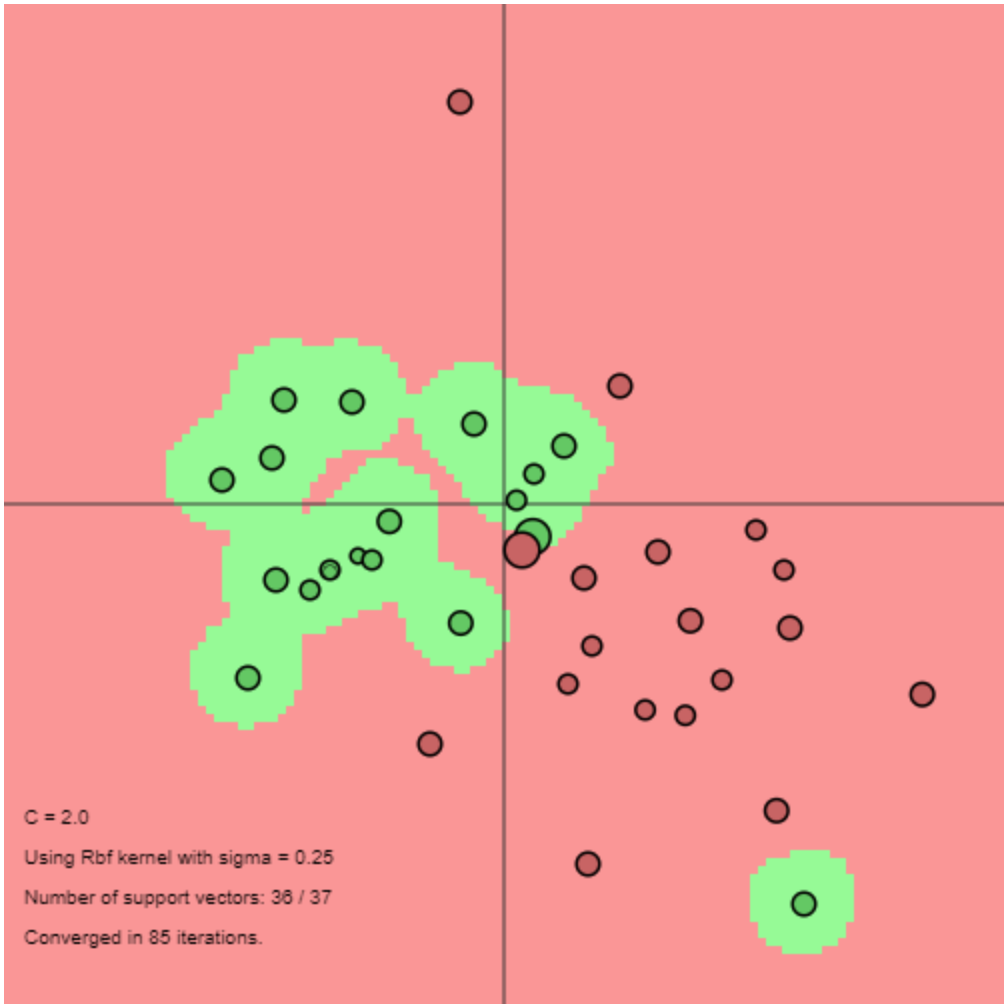


SVC with RBF kernel



SVC with sigmoid kernel





Source: https://cs.stanford.edu/people/karpathy/svmjs/demo/

- **Journal:** Evgeniou, Theodoros, and Pontil, Massimiliano. "Support Vector Machines: Theory and Applications." *Lecture Notes in Computer Science*, vol. 2049, 2001, pp. 249-257. doi:10.1007/3-540-44673-7_12. Available online at: Support Vector Machines: Theory and Applications.

Category 7: Neural Networks

- Advantages: The complexity of neural networks usually leads to pretty accurate results among other models. Neural networks can learn non-linear relationships, also, they are more capable of dealing with complicated questions. Moreover, it is more flexible and scalable. Unlike traditional machine learning models such as regressions and PCA, neural networks don't require any feature extraction, since feature extraction is automatic.
- Computation: For demonstration purposes, the S&P 500 from 2010 to 2022 is modeled, and the year 2023 data is used for testing. Please refer to the Jupyter Notebook.
- Disadvantage:
    - The most significant disadvantage of neural networks is their cost. They cost lots of GPU or NPU to generate accurate models based on provided data.
    - Neural network models typically require a large amount of data to perform well. Small amounts of datasets may cause overfits or poor performance.
    - It is hard to interpret. Since we can't see the computation details of neural networks, it can be hard to interpret models and predictions directly.
- Equations:

  There is a methodology and an architecture that is widely used in neural networks. Neural networks involve a simple process:

  Input -> analyze -> decision

  The inputs are dealt with by the input layer, and hidden layers process the analysis of the data. Finally, the decision is made by the output layer. The specific flow can be referred to as follows:

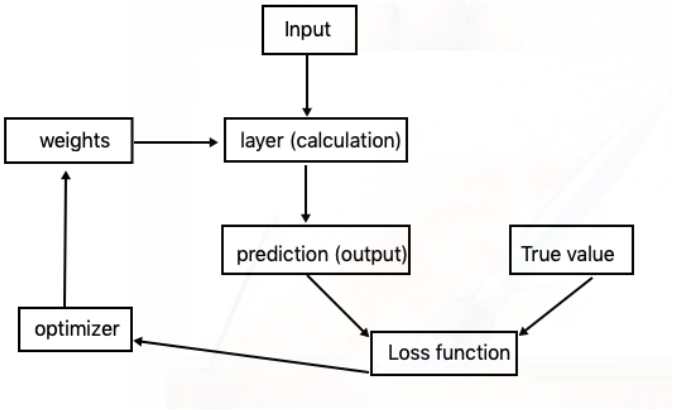


Figure: neural networks flow (Quoted from M5L3 Lesson note)

To model data concerning neural networks, forward and backward propagation are introduced. In this case, we represent the input layer, hidden layers, and the out layer

X_0, X_1, Y respectively. From the input layer to the output layer, the question is the following:

$$X_1 = f(X_o\omega_0 + b_0)$$

Where ω is a matrix of weight, f is a user-specified activation function, and b is a bias term.

To move further, it requires more activation functions, it depends on how many hidden layers are involved in this case.

- Features:
  - **Large amounts of data capability and perform poorly with small amounts of data:** neural network models normally perform well for large amounts of data, however, it doesn't perform very well when the dataset is limited which happens in the computation example in the note;
  - **High computation power consumption:** compared with traditional machine-learning models, it requires a longer time and quite a lot of computations to complete.
  - **The number of hidden layers dependence:** the model performance may depend on the number of hidden layers.
- Guide:
  - Input: the training data, neural network framework (Pytorch, Tensorflow, or others), epochs, and batch size.
  - Output: the prediction
- Hyperparameters:

  From the equation part, we can find that ω , f, and b are key parameters for neural network models. And they need to be pre-defined.

- **Illustration:**

  In this testing case, the average hourly earnings of all employees, all employees manufacturing, monthly producer price index by Industry, durable goods new orders, and fed fund rate are used as the predictor variables and S&P 500 as the dependent variable. Monthly data are collected from 2010 to 2023. 70% data are used for training, while 30% of them are testing. A total of 168 data points are collected for each column. As stated, small amounts of data points may lead to poor performance in modeling. The mean absolute error and mean squared for this model are 0.25 and 0.07 respectively. In 2023, there was a recession in the US market. However, neural network models interpret this dramatically.

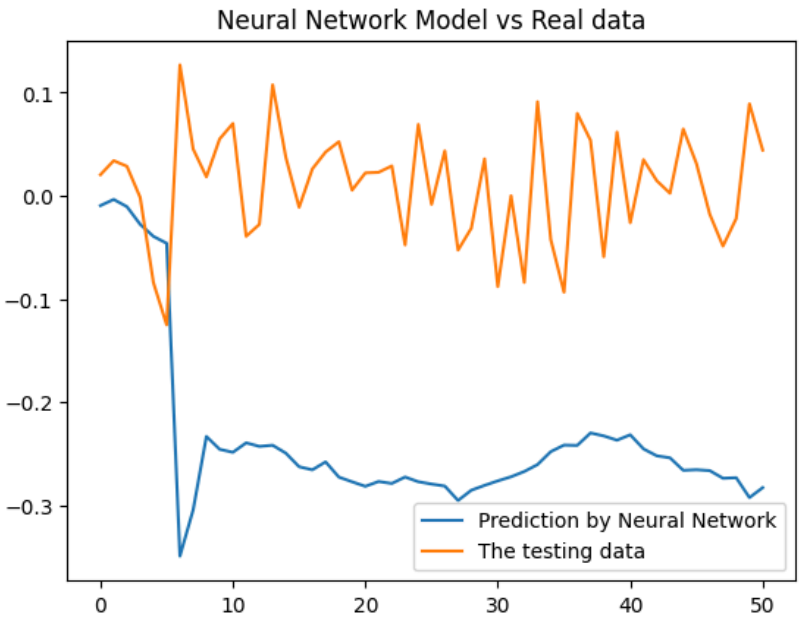


Figure: predicted data by the neural network vs real data

In the first couple of data points, the neural network model performs fine. However, in the following data points, the neural network performs extremely poorly.

- **Journal:** Wu, Y. C., & Feng, J. W. (2018). Development and application of artificial neural network. *Wireless Personal Communications*, *102*, 1645-1656.

Step 4

- **Category 5: Linear Discriminant Analysis**

  Linear discriminant analysis (LDA) is a classification technique that aims at finding the linear features that best distinguish different classes. For GWP, LDA hyperparameters were tuned using the GridSearchCV method, and allows testing different combinations of hyperparameters to find the optimal combination. This tuning process ensured the selection of the best model parameters to maximize classification accuracy on the Iris dataset.

- **Category 6: Support Vector Machines**

  Support Vector Machines (SVM) are good classifiers that work well with both linear and non-linear data. The SVM hyperparameters were manually tuned by iterating over the values of C (regular parameter), gamma (kernel coefficient) and kernel type. The best combination of hyperparameters was selected based on the highest accuracy achieved on the test set, indicating the importance of hyperparameter tuning in improving model performance.

- **Category 7: Neural Networks**

  Neural Networks, particularly deep learning models, have numerous hyperparameters that can significantly affect their performance. This tuning process aimed to achieve the highest accuracy on the validation set, demonstrating the critical role of hyperparameter tuning in neural networks for optimizing model performance and generalization.

Step 5

This report has analyzed three different machine learning strategies: Linear Discriminant Analysis (LDA), Support Vector Machines (SVM) and Neural Networks (NN). The basics of the methods and their main advantages and drawbacks have been discussed. Some numerical examples have also been included, including financial data and known databases, to show their possible application.

The first method analyzed, LDA, is a classification algorithm for multi-class problems. It is based on dimensionality reduction into a lower set of variables (such as PCA), and resolved also an eigenproblem but with the objective of finding the coordinates (the local discriminant components) that better separate the data into different classes. As it was shown in the jupyter notebook, its numerical implementation is easy and is computationally inexpensive while being applied to a real case from German credit data. Even though in this case it did not produce a good classification of this case, this was attributed to the fact that the database used does not follow the assumptions upon which LDA is based (data linearity, normally distributed), which could be guessed a-priori from an analysis of the database in a pair plot. It would be interesting to find real data which respects these hypotheses, to show how well LDA can perform (even though other dummy databases are available, which have shown the good performance of LDA, it has been prioritized the application of LDA to real cases).

Secondly, SVMs are a family of supervised learning algorithms that can be used for classification and regression. Its easiness to deal with high-dimensional datasets makes it a suitable tool for handling cases where the difference among the number of dimensions is considerable with than the number of samples, either for cases where there are more data samples than dimensions (e.g. text classification) or vice-versa. Furthermore, the introduction of the margins controlled by the *c* parameter make it a robust algorithm for avoiding overfitting, and its ability to introduce kernel functions for processing make SVM able to deal with non-linear data. Even though the code was tested on a dummy database, the good performance of SVMs make it become an ideal candidate to deal with similar real classification problems.

Finally, NNs were detailed and tested. Even though their implementation might seem like a ‘black box’ sometimes, their complexity allows them to obtain accurate results and deal with both linear and non-linear data without needing to change any special features to the algorithm (for example, remember that in SVMs non-linearity is addressed through kernel functions). On the other hand, they can be costly, yet current advancements in computing (HPC through parallelization of CPUs and GPUs) make them more affordable nowadays.

Step 6

Linear Discriminant Analysis (LDA) stands out for its simplicity and ease of interpretation, making it an excellent choice for small datasets (Mishraz, Ashok, and Tandon, 2021). Additionally, LDA is generally computationally efficient. However, when it comes to neural networks, they tend to underperform with small datasets but can achieve remarkable results with larger ones. On the flip side, LDA's assumption of linearity in models and data can be a limitation.

Support Vector Machines (SVM) and neural networks shine in handling non-linear data (Evgeniou and Pontil 2001). Thanks to their complexity, these methods are adept at managing high-dimensional data and more intricate cases. SVM can also handle various types of data including binary classification. Compared with traditional machine-learning techniques, SVM is somehow robust to outliers.

Neural networks are outstanding in their scalability and flexibility (Wu and Fang 2018), so they can be fitted into many big data scenarios. In the meanwhile, as people see the great potential of neural networks, many neural network frameworks such as Tensorflow and PyTorch make neural networks implementation easy.

Step 7 Comparing Models

	Linear Discriminant Analysis	Support Vector Machine	Neural Networks
Dataset	Small	Medium or Large	Large
Simplicity and Interpretability	High. Simple and easy to interpret.	Medium. More complex than LDA but interpretable with kernel choice.	Low. Complex and harder to interpret.
Computation	Small consumption	Medium to large consumption which depends on the amount of dataset	Large consumption
Linearity	Assume linear	Work for both linear and non-linear data	Work for both linear and non-linear data
Scalability	Medium. Limited scalability with very large datasets.	High. Scales well with high-dimensional data.	High. Can handle vast amounts of data and complex structures.
Robustness to Overfitting	Medium. Can overfit if the assumptions are not met.	High. Effective margin reduces overfitting.	Medium. Prone to overfitting, but can be mitigated with techniques like dropout.