Individual Project

DELGADO Fernando

March 26th 2022

Contents

Introduction

Statistical learning refers to the set of tools for modeling and understanding complex datasets. It is a mix of statistics, computer science, and machine learning. As an aspiring data scientist, I believe that statistical learning is of high importance to become competitive in the market, given its increasing popularity in fields such as marketing, finance, and other business disciplines.

The main purpose of this project is to assess two important subjects of statistical and machine learning: the understanding of machine learning mechanisms and the ability to setup a machine learning pipeline project.

Statistical Learning

Statistical learning refers to a set of approaches to make predictions or infer a result. It helps us answer the following questions such as, which predictors are associated with which response? What is the relationship between the response and each predictor?

Supervised vs. Unsupervised

Most statistical learning problems belong to two different categories, supervised or unsupervised. The first one explains that for each observation of the prediction measurements $x_i, i = 1, \dots, n$, there is an associated response of measurement y_i. For the unsupervised, on the other hand, it describes a more challenging situation in which for every observation $i = 1, \dots, n$, we observe a vector of measurements x_i, but no associated response y_i.

Classification vs. Regression

Furthermore, variables can be characterized as either quantitative or qualitative(categorical). Quantitative variables take on numerical values, while qualitative variables take on one of K classes or categories (such as, will the customer churn? Yes/No). We tend to refer to problems with a quantitative variable as regression problems and those involving qualitative results as classification problems.

Additionally, when it comes to modeling: no model is the best above all as some models are better for some specific situations but may not be so good for others. Thus, it is important to know and understand a wide variety of models.

Part I: Understanding Machine Learning Algorithms

1. Logistic Regression

Despite its name, logistic regression is a classification model (not a regression model). Logistic regression is a simple and efficient method for binary classification problems that is relatively easy to perform.

1.1 How is Logistic Regression Different from Linear Regression?

In order to understand logistic regression, we must also comprehend the basics of linear regression and why we shouldn't use it in for classification problems. Linear regression is an approach for predicting a target variable *Y* with predictor variables *X*. It assumes that there is approximately a linear relation between *X* and *Y*.

Mathematically we write the linear relationship as:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_p X_p + \epsilon$$

The objective is to compute β_0 (the intercept) and all of the other β_p that represent the slope for each predictor. However, if we use this approach to predict a categorical variable (such as Yes or No), then we may obtain the following result:

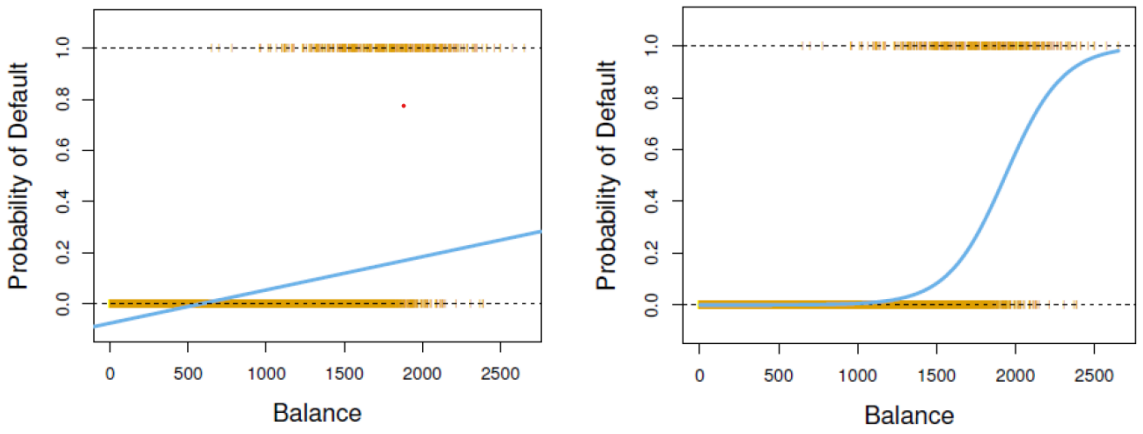

Figure 1. Linear Regression Vs Logistic Regression. Source: James, G., Witten, D., Hastie , T., & Tibshirani , R. (2014). An Introduction to Statistical Learning: With Applications in R . Springer Publishing Company , Incorporated

As we can observe in the left panel of Figure 1, the problem with linear regression for classification problems predictions is that as we approach values close to zero, we predict negative probabilities of default payment next month, as well we could predict values bigger than 1. Hence, these predictions are not sensible since the true probability of default payment lies between Yes and No (1 and 0), regardless of the values of our predictor variables. This would make sense for a quantitative type of problem, but not for this example.

To avoid the problem shown with Figure 1, logistic regression models $p(X)$ using a function that gives outputs between 0 and 1 for all values of X. We can observe the logistic function (also known as sigmoid function) as follows:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

As shown on the right panel of figure 1, the predictions may approach 1 and 0 but never go beyond it. The logistic function always produces an *S-shaped* curve (sigmoid curve). The values between 0 and 1 are simply the probability of our prediction.

1.2 Regression Coefficients and Maximum Likelihood

In order to estimate the β coefficients for logistic regression, we use a maximum likelihood approach. Basically, with our previous example, we are looking for estimates of β so that putting these estimates into the model for $p(X)$ returns a number close to one for all individuals with a default payment next month, or zero for those who don't. In other words, the maximum likelihood attempts to find for the values of β that minimize the error in the probabilities predicted by the model. This intuition can be observed with the following mathematical function:

$$l(\beta_0, \beta_1) \prod_{i:yi=1} p(x_i) \prod_{i':yi'=0} (1 - p(x_{i'}))$$

Thankfully for us, we can fit logistic regression and other models easily with Python statistical packages without going into the depths of the mathematics of maximum likelihood. Specifically for this project we will describe how to fit our models in Python with Scikit Learn, which in-depth on the exploration part of the report.

2. Linear Discriminant Analysis (LDA)

Moving forward, Linear Discriminant Analysis (or LDA for short), is another more complex type of classification algorithm for machine learning. While logistic regression attempts to model $\Pr(Y = k \mid X = x)$ directly, LDA considers a less direct approach to estimate probabilities. It models distribution of the predictors X separately in each response class and use the Bayes' theorem to flip them into estimates for probabilities (instead of the maximum likelihood). Additionally, LDA help reduces the dimensionality of our data by focusing on maximizing the separability among known categories.

2.1 Bayes' Theorem

This approach describes the probability of an event, based on prior knowledge of conditions that might be related to the event. When classifying an observation into one of K classes, π_k represents the prior probability that an observation belongs to a kth class. Next, we can understand a density function of X for an observation that comes from the kth class as: $\Pr(Y = k|X = x)$. Mathematically, we observe the Bayes as:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

Basically, we can estimate approaches of π_k instead of computing the probability of $p_k(X)$ like in the logistic regression. Since a bayes classifier has the lowest possible error rate when it classifies an observation to the class for which $p_k(X)$ is largest, we can find a way to estimate $f_k(X)$ and develop a classifier that approximates the Bayes classifier.

2.2 LDA and Multivariate Gaussian Distribution

For this report, we will focus only on LDA classifier for cases where we have more than one predictor. We will assume that $X = (X_1, X_2, \dots, X_p)$ is drawn from a multivariate Gaussian (or multivariate normal) distribution, which is a class-specific mean vector and a common covariance matrix.

Such distribution assumes that each individual predictor follows a one-dimensional normal distribution as in the following formula, with some correlation between each pair of predictors, where μ_k and σ_k^2 are the mean and the variance parameters for the kth class:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\,\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

The following figure displays two multivariate Gaussian density functions are shown with two predictors $p = 2$. The left graph shows two uncorrelated predictors while the right one show two variables with a correlation of 0.7. The bell shape shown on the left illustrates the example of the variables having a correlation of 0. Moreover, the bell shape is distorted if the predictors are correlated or have unequal variances as shown on the right graph.

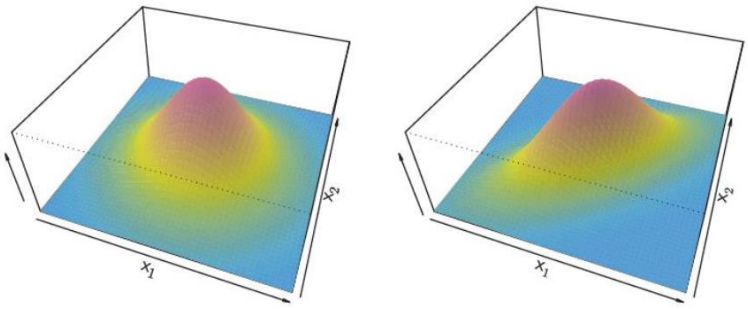

Figure 2. Multivariate Gaussian Distribution. Source: James, G., Witten, D., Hastie , T., & Tibshirani , R. (2014). An Introduction to Statistical Learning: With Applications in R . Springer Publishing Company , Incorporated

The height of the surface at any particular point of the bell represents the probability that both X_1 and X_2 fall in the same region or classification. If the surface is cut along the X_1 axis or along the X_2 axis, the resulting cross-section will have the shape of a one-dimensional normal distribution.

Mathematically, the multivariate Gaussian density function is defined as:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\sum|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \sum^{-1} (x - \mu)\right)$$

The LDA classifier assumes that the observations for the kth class are taken from the multivariate Gaussian distribution $N(\mu_k, \sum)$, where μ_k is a class-specific mean vector, and $\sum$ is a covariance matrix that is common to all K classes. Then, we can observe how the Bayes classifier assigns an observation $X = x$ to the class for which the discriminant score is largest:

$$\delta_k(x) = x^T \sum^{-1} \mu_k - \frac{1}{2} \mu_k^T \sum^{-1} \mu_k + log\, \pi_k$$

2.3 How is LDA different than Logistic Regression?

LDA is a good approach over logistic regression for the following reasons because when the classes are well separated, logistic regression can be unstable. Additionally, if there are a small number of observations and the distribution of the predictors is approximately normal, LDA is more stable. What's more, LDA is a popular approach when the independent variable has more than 2 classes, but it also works for two classes such as Yes or No.

Since logistic regression and LDA differ only in their fitting procedures, one might expect the two approaches to give similar results. Nevertheless, logistic regression can outperform LDA if the Gaussian assumptions are not met.

3. Decision Tree

Tree based methods involve segmenting the predictor space into a number of regions or splits. Since the set of splitting rules used to segment the predictor space can be visualized as a tree, these types of approaches are known as decision tree methods. Decision trees can be applied to both regression and classification problems, however we will focus on classification trees for this report, to make it comparable with our previous two models.

For decision trees, each non-leaf node (each node that is not at the bottom of the tree) represents a test on a feature attribute. Each branch represents the output of a feature attribute in a certain range, and each leaf node stores a category. Decision trees work by starting from the root node (the top), testing the feature attributes of the items to be classified, selecting the output branches, and finally using the category stored on the leaf node as the final result (the bottom). Take the following figure as an example:

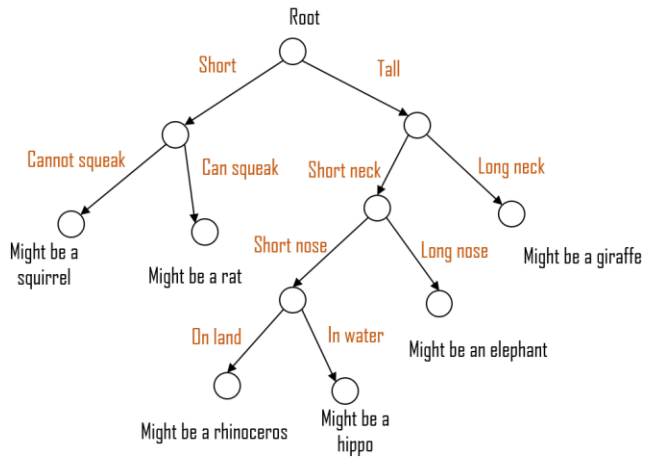

Figure 4. Decision Tree Example. Source : https://forum.huawei.com/enterprise/en/machine-learning-algorithms-decision-trees/thread/710283-895

3.1 Classification Trees

As one would expect, regression trees are used to predict a quantitative response while classification trees are used for qualitative predictions. On one hand, regression trees use the predicted response for an observation that belongs to the same terminal node. On the other hand, classification trees predict that each observation belongs to the most commonly occurring class of observations in the region which it belongs (“might be a rhinoceros”, for example). Moreover, classification trees help us not only to predict a class corresponding to a particular terminal node, but also the class proportions among the training observations that fall into the region.

Besides , regression trees use RSS as a criterion in order to make the splits between nodes, but for classification trees we attempt to assign an observation in a given region to the most commonly occurring class of observations in the region. The classification error rate is the fraction of the observations in the region that does not belong to the mentioned class.

3.2 Evaluation

Mathematically, decision trees use two different methods to evaluate the quality of a particular split: Gini Index or Entropy.

The Gini Index is defined by a measure of total variance across K classes:

$$G = \sum_{k=1}^{k} pmk(1 - pmk)$$

In essence, Gini index takes on a small value if all of the pmk are close to zero or one. In other words, a small value indicates that a node contains predominantly observations from a single class.

Entropy is given by:

$$D = \sum_{k=1}^{k} pmk \log pmk$$

Entropy will take a value near zero if the pmk are all near zero or one. Thus, entropy will take on a small value if the mth node is pure.

3.3 Tree pruning

With decision trees, it is very common to overfit the data because the resulting tree might be too complex or too large. We understand overfitting the data as creating an analysis that corresponds exactly (or way too close) to a particular set of data, and therefore could fail to fit to preform predictions reliably. Hence, a smaller tree with fewer splits might lead to a lower variance and better interpretation at the cost of a little bias.

Therefore, by growing large trees and then pruning them back in order to obtain a subtree we can obtain better results. The objective is to select the subtree with the lowest error rate, which can be estimated using a cross-validation approach. However, doing this could take forever given that there is a very large number of possible trees. Thus, cost complexity pruning helps us by considering a sequence of trees indexed by a non-negative tuning parameter α. We want to tune α to find the best possible tree with the lowest error rate.

3.4 Building a Decision Tree

To build a decision tree we follow the steps of the next algorithm:

1. Grow a large tree with the data, stopping only when each terminal node has fewer than some terminal observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross validation to choose α.
    a. Repeat steps 1 and 2 on all but the kth fold of the data.
    b. Evaluate the mean squared prediction error on the data in the left-out kth fold as a function of α.

Average the results for each value of α and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

3.5 How are Trees Different from Linear Regression Models?

In essence, if the relationship between the features and the response is well approximated by a linear model , then an approach such as linear regression would be fit and could outperform the accuracy of a tree. However, if there is a highly non-linear relationship between the features and the response, decision trees may outperform classic linear approaches.

3.6 Advantages and Disadvantages

Advantages	Disadvantages
Trees are very easy to comprehend.	Trees do not have the same level of predictive accuracy as some other classification approaches.
Decision trees mirror human decision-making more closely than other classification approaches.	A very small change in the data can cause a large change in the final estimated tree.
Trees can be displayed graphically and are easily interpreted.	
Trees can easily handle qualitative predictors without dummy variables.	

4. Bagging, Random Forests and Boosting

Bagging, random forests and boosting use decision trees as a base to construct improved and more powerful prediction models.

4.1 Bagging

The main problem of decision trees is that they suffer from high variance. For example, if we would split the data into two random splits and fit the same model to both, we might end up with two very different trees as a result.

Bagging is a process with the purpose of reducing the variance of any statistical method. The way it works is that it reduces the variance and increases the prediction accuracy by taking many different data sets and averaging the resulting predictions. However, this is not very practical since it's not normal to have access to many different multiple data training sets. Thus, we can bootstrap the same dataset by taking repeated samples of the same data. By training the method on the bth bootstrapped training set in order to get $f^{*b}(x)$ and finally averaging all the predictions to obtain:

$$fbag(x) = \frac{1}{B}\sum_{b=1}^{B} f^{*b}(x)$$

When we build bagging trees, they are grown deep and not pruned, which means each individual tree has a high variance and a low bias. Particularly for classification trees, we can record the class predicted by each of the B trees and take a majority vote. This means that the overall prediction is the most common occurring class among the B predictions.

Even though bagging improves the resulting accuracy of the predictions, it can be difficult to interpret the results. In other words, bagging improves the accuracy of our results at the cost of interpretability.

4.2 Random Forests

Random Forests provide an improvement over bagging methods by decorrelating trees. Just like with bagging, we bootstrap a number of dataset training samples, but when we build the trees, a random sample of m predictors is chosen as a split candidate from the full set of p predictors.

That's to say, when we build a random forest, at each split of the tree, the algorithm is not allowed to consider a majority of the available predictors. The reason behind this is that if for some reason we had one very strong predictor in the data set (alongside other not so strong predictors), then at the collection of bagged trees, most trees would have the same strong predictor on the top split and the trees would be highly correlated between each other.

Random forests overcome this problem by forcing each split to consider only a subset of the predictors. Thus, on average $(p - m)/p$ of the splits will not consider the strong predictor, and it would give other predictors a chance.

4.3 Boosting

Finally, boosting is another is another approach for improving the predictions resulting from a decision tree. Boosting works in a similar way as bagging except that the trees are grown sequentially: each tree is grown using information from previous tree. Each tree is fit on a modified version of the original dataset.

Unlike fitting a single dataset to a single large decision tree (which usually leads to overfitting, the boosting approach instead learns slowly. We fit a decision tree to the residuals from the model. In other words, we fit a tree using the current residuals, rather than the outcome Y. We then add this new decision tree into the fitted function in order to update the residuals.

By fitting small trees to the residuals, we slowly improve the predictions in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

4.3.1 Parameters

Boosting has three tuning parameters:

1. The number of trees B. To avoid overfitting, we use cross-validation to select B.
2. The shrinkage parameter λ, which is a small positive number. This controls the rating at which the boosting tree learns.
3. The number d of splits in each tree, which controls the complexity of the boost. Often $d = 1$ works well, in which case each tree consists of a single split.

4.3.2 Boosting for Regression Trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat.
    a. Fit a tree $\hat{f}^b$ with d splits to the training data.
    b. Update $\hat{f}$ by adding in a shrunken version of the new tree

c. Update the residuals

3. Output the boosted model:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

5. Neural Networks

As our final model for the report, neural networks are another subset of machine learning algorithms and are considered to be the heart of deep learning. In essence, neural networks reflect the behavior of the human brain since they recognize patterns by mimicking the way that human neurons signal to one another. Biological neurons act as a computational unit, accepting input from dendrites and outputting different signals through the axon terminals. Finally, the brain triggers actions when a specific combination of neurons are activated.

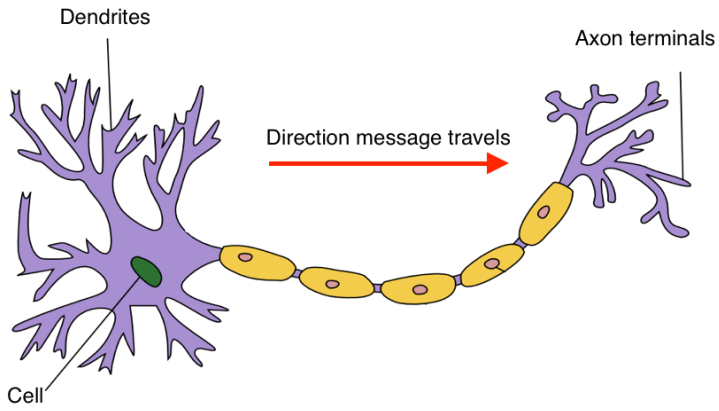

Figure 5. Biological Neuron Source: https://www.jeremyjordan.me/intro-to-neural-networks/

Neural networks are built of node layers (neuron layers), containing an input layer (dendrites), one or more hidden layers, and an output layer (axon terminals). Each node connects to another and has an associated weight and threshold. If any neuron is above the specified threshold value, it will send information to the next layer. Otherwise , no information will be sent along the next layer.

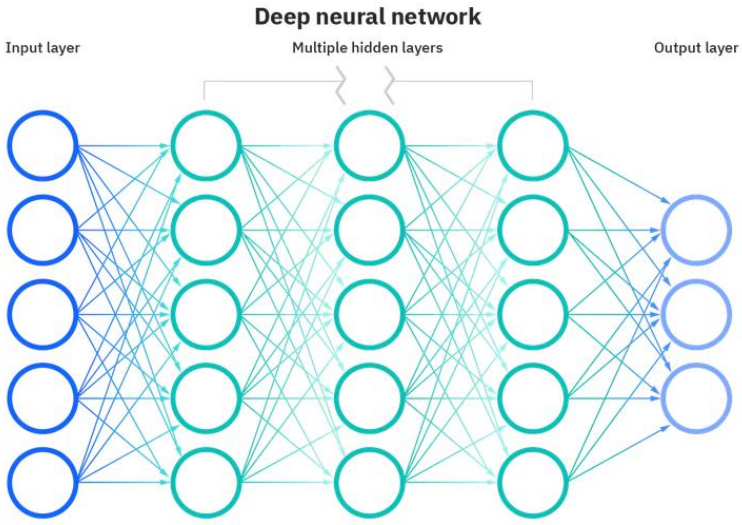

Figure 6. Deep Neural Network. Source: https://www.ibm.com/cloud/learn/neural-networks

Furthermore, neural networks work by learning from the data and improving the accuracy over time. Once these algorithms are tuned, and thanks to their brain-like ability to learn and adapt, neural networks can become very powerful tools for artificial intelligence (AI), allowing to classify and cluster data at a very high speed. We understand AI as intelligence originating from machines. Thanks to machine learning, we can give machines a way to think, learn and adapt.

Basically, neural networks are important and useful because they can work with high complex pattern recognition, and they can learn and do things that humans don't know exactly how to do. Additionally, they are the building blocks for advanced machine learning techniques and are considered to be the evolution of computers.

5.1 How do Neural Networks work?

We may think of each individual neuron of the model as an individual logistic or linear regression, composed of an input, data, weights, a bias(threshold) and an output:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_2 + w_2 x_2 + w_3 x_3 + bias$$

$$f(x) = \{1 \; if \; \sum w_1 x_1 \geq threshold | 0 \; if \sum w_1 x_1 < threshold \}$$

When an input layer is determined, weights (w) are assigned. Weights are a way to determine the importance of each feature, with the larger ones contributing more to the output. Then, all inputs are multiplied by their respective weights and summed. Following, the output is passed through an activation function in order to determine the output. If such output surpasses the threshold, it activates the neuron, passing data to the next layer in the network. This results in the output of one neuron becoming the input of the next.

If we try to take a more tangible example, we may think of binary values. For instance: Are we going to order pizza? Yes: 1, No: 0. The decision to order pizza or not will be our predicted outcome. We can assume the following factors that can influence the decision making:

1. Are we hungry? Yes: 1, No: 0
2. Do we have enough money? Yes: 1, No: 0.
3. Do we have friends to share it with? Yes: 1, No: 0.

Let's assume the following inputs:

- $X1$ = 1, since we are hungry.
- $X2$ = 0, since we are students without money.
- $X3$ = 1, since we are going to share the pizza together.

Then, we need to assign weights of importance to each variable. A larger weight would mean that the variable is of greater importance for us to make the decision.

- $w1$ = 3, since being hungry is just our normal biological need.
- $w2$ = 2, for one day we will be rich data scientists to pay our debts.
- $w3$ = 5, since good company should never be taken for granted.

Finally, we can assume a threshold value of 3, which would also mean a bias value of -3. With this in mind we can plug our formula to:

$$f(x) = (1 * 3) + (0 * 2) + (1 * 5) - 3 = 5$$

By following the activation function, the output of this node would be 1, since 5 is greater than our threshold of 3. With this example, we would be receiving a very warm and tasty pizza soon since we have decided to order. However, if we adjust the threshold to different values, we can receive different outcomes.

This is a simple example above in which we use perceptrons to illustrate the mathematics behind the neural network, however, neural networks work by passing the linear combination of inputs through the logistic (or sigmoid) function, which is also known as the activation. The difference between sigmoid and perceptrons is that sigmoid takes values between 0 and 1, which will reduce the impact of any given change of a single variable on the output of any given neuron, and thus, the output of the neural network.

5.2 Cost Function and Point of Convergence

As with the other models, we want to measure the accuracy of our neural network. We can evaluate neural networks accuracy by using the following cost function (which is also known as the MSE function):

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} (yhat - y)^2$$

The idea is to minimize our cost function to make sure our model is fit correctly into any given observation. As the model adjusts its weight and bias, it uses the cost function and reinforcement

learning to reach a point of convergence, which is the minimum point of error. This process is called gradient descent, which shows the model which direction to take in order to reduce errors.

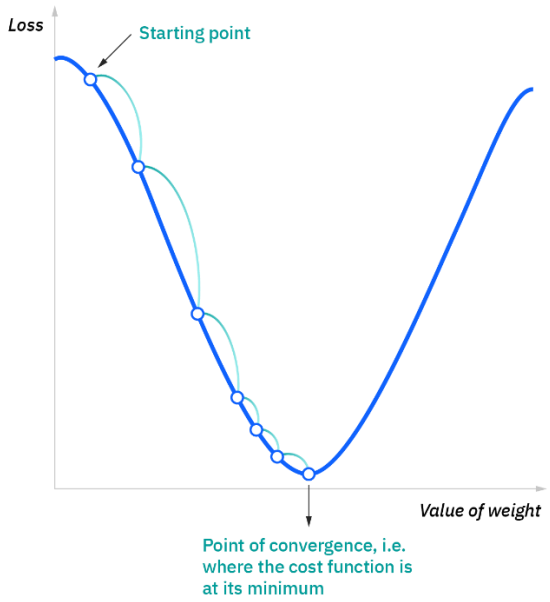

Figure 7. Cost Function and Point of Convergence. Source: https://www.ibm.com/cloud/learn/neural-networks

Wrapping up, deep learning and neural networks tend to be used together in conversation and sometimes mixing them can be confusing. Its only important to note that the "deep" in deep learning refers to the depths of the layers in the neural network. We consider a deep neural network as anything beyond 3 layers or more, while anything with less is a simple neural network.

Part II: Benchmark Experiment

Moving on to the second section of this report, we will now showcase an application of the previously explained models with a benchmark experiment that comprehends feature selection and cross-validation with a credit card default payment dataset.

6. Data-Processing

6.1 Data Description

We are working with a credit card payment dataset. This dataset contains 20,000 observations for 25 variables, where we have 23 predictors variables, 1 ID variable, and 1 target variable.

In essence, we want to estimate our target variable of default payment to predict whether our customers will perform a default or not by analyzing the behavior of our predictor variables. Moreover, given that for each observation of the predictor measurements $X_i = , \ldots \ , n$, there is an associated response measurement y_i, we are dealing with a supervised learning domain. Additionally, due to the fact that our prediction variable takes on values in one of 2 classes, default payment or not (a qualitative response), we are dealing with a classification problem.

6.2 Train-Test Split

We split our data into Train and Test with an 80/20 partition ratio. A Train-Test split is a technique for evaluating the performance of a machine learning algorithm by splitting a dataset into 2 parts. The reason we perform this split before applying any pre-processing steps is to make sure there is no data leakage within our test set. In Machine Learning, data leakage refers to accidentally sharing data from the train set into the test set.

6.3 Data Processing Pipeline

Since data processing is not the objective of this report, we won't detail the explanation of our pre-processing steps. However, it is important to know that we've done error correction, feature engineering, value transformation, and value representation.

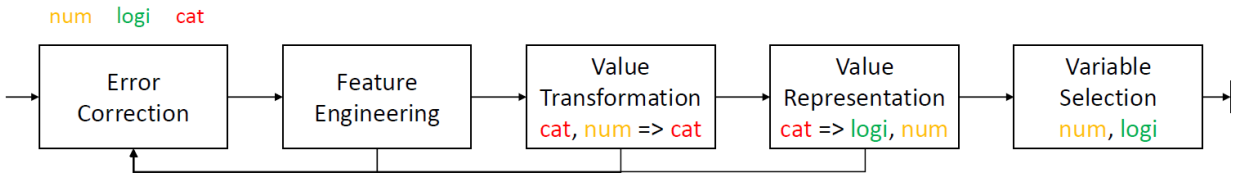

Figure 8. Data Processing Pipeline **Source:** PHAN, M. (2022). Statistical & Machine Learning. [Course]. Lille: IESEG Management School. MSc in Big Data Analytics.

After our preprocessing, the dimensions of our data end up as:

```
#Check dimensions
print('Train split size: ', train.shape)
print('Test split size: ', test.shape)
```

```
Train split size:  (16000, 67)
Test split size:  (4000, 67)
```

6.4 Variable Selection

Variable selection is the process of isolating the most relevant features. The main goal of the selection is to improve the performance of our models as well as reducing the computational cost of modeling.

The are many ways of selecting features, but for this report, we will work with Fishers' score. Fishers' score is a filter-based supervised feature selection method with feature weights. In essence, Fisher score helps us find a subset of features, such that the distances between data points in different classes are as large as possible, while distances between data points in the same class are as small as possible.

After calculating our Fishers scores for each variable, we can order them by descending order from the most relevant to the least, and visualize how the AUC changes as we add more variables:

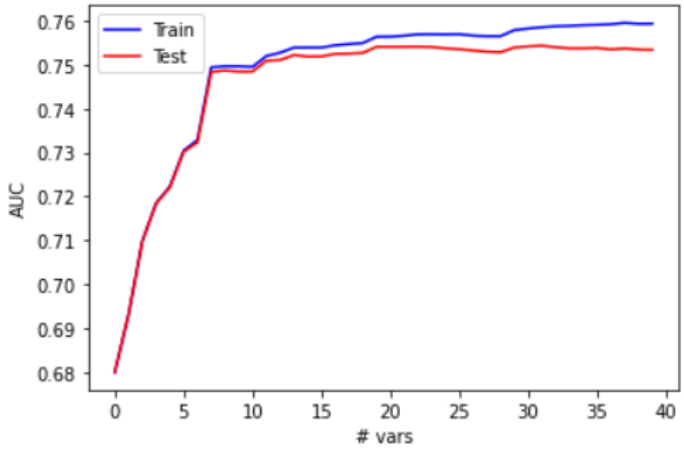

Figure 10. AUC curve with top variables. Source: PHAN, M. (2022). Statistical & Machine Learning. [Course]. Lille: IESEG Management School. MSc in Big Data Analytics.

As we can observe, after 35 variables, despite our train AUC increasing, the test is starting to stray further away from the train curve, which could mean model overfitting. Thus, we decide to keep the top 35 variables for our modeling.

7. Modeling

7.1 Hyper-Parameters and Grid-Search

Machine learning models have multiple parameters that can't be trained with our data. These parameters can control the accuracy of our predictions, and therefore, they are very important for any data-science project. For example, the learning rate of a neural network is a hyper-parameter because it is set by the model before the data is fed into the model. Nevertheless, the weights of the neural network are not hyperparameters because they are trained with the data.

Grid Search is a tuning technique that attempts to find the optimum values for our hyper-parameters. This is also known as estimator.

7.2 K-Fold Cross-Validation

To avoid overfitting, it is a normal practice to perform cross validation by partitioning the training set into *k* smaller sets. The idea is to split the data several times and fit the model to all of the train sets and compare the accuracies to make sure they are all similar. Then we obtain a mean accuracy and can tell whether it's good or not.

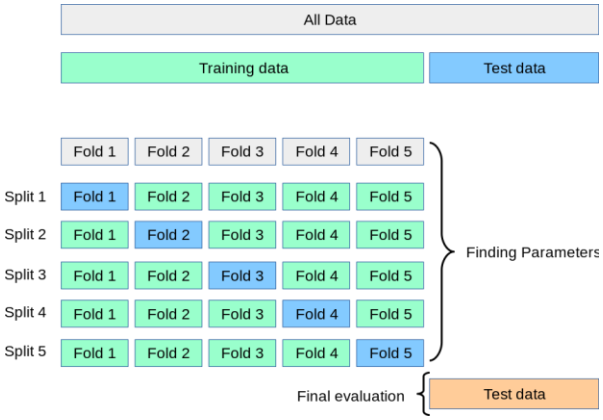

Figure 11. K-Fold Cross Validation Source: https://scikit-learn.org/stable/modules/cross_validation.html

7.2 Logistic Regression

We fit the following parameters for logistic regression and subsequently apply a cross-validation:

```
#Parameters
parameters={"C":np.logspace(-4, 4, 50), "penalty":["l1", "l2"]}# l1 lasso l2
ridge
#Setup model
lr=LogisticRegression()
#Setup Cross-Validation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
#Grid Search
lr_cv=RandomizedSearchCV(lr, parameters, cv=3)
#Cross-validate model
scores = cross_val_score(lr_cv, train[top_fs_vars], train['target'],
scoring='accuracy', cv=cv, n_jobs=-1)
```

After running the grid search, we obtain our best parameters with a mean accuracy of 0.808 which is a good result. Then, we fit the best parameters obtained from our grid search into the model and make predictions over the test set. We obtain the following performance:

	Train_AUC	Train_Accuracy	Test_AUC	Test_Accuracy
LogisticRegression	0.761012	0.809562	0.751847	0.81125

It is important to observe that the test accuracy our test accuracy is performing better than the train accuracy which means our model is very well fitted. Also, this accuracy score is not far from our cross-validated score, which is a good indicator.

7.3 Linear Discriminant Analysis

Following the same logic, we perform a parameter tuning and cross-validation for LDA:

```
lda = LinearDiscriminantAnalysis(solver='lsqr')
#Parameters
grid = dict()
grid['shrinkage'] = arange(0, 1, 0.01)
#Set up Crossvalidation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
#Grid Search
lda_cv = RandomizedSearchCV(lda, grid, scoring='accuracy', cv=3, n_jobs=-1)
#Cross-validate model
scores = cross_val_score(lda_cv, train[top_fs_vars], train['target'],
scoring='accuracy', cv=cv, n_jobs=-1)
```

We obtain a mean accuracy of 0.808, and after fitting our data we obtain the following performance:

	Train_AUC	Train_Accuracy	Test_AUC	Test_Accuracy
LDA	0.754352	0.810187	0.746116	0.81125

The results are very similar to logistic regression, but with a slightly lower AUC. Again, the mean cross-validated accuracy is similar to the test which is a good indicator.

7.4 Decision Tree

For decision tree, we apply the following grid search with stratified k-fold cross validation by taking both Gini and Entropy criterion:

```
#setup parameters
param = {'criterion':['gini','entropy'],
    'max_depth':[5,10,15,20,30,50]}
#setup cross validation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
#Grid Search
clf = RandomizedSearchCV(DecisionTreeClassifier(), param, scoring='accuracy',
cv=3, n_jobs=-1)
#Cross-validate model
scores = cross_val_score(clf, train[top_fs_vars], train['target'],
scoring='accuracy', cv=cv, n_jobs=-1)
```

We obtain the mean accuracy of 0.806 and proceed to obtain the following performance:

	Train_AUC	Train_Accuracy	Test_AUC	Test_Accuracy
decisiontree	0.76766	0.809125	0.74654	0.8065

The performance is slightly worse than our previous two models, but it makes sense since the decision tree is not known for having the best performance but is highly interpretable.

7.5 Random Forest

Following, we perform a grid search and cross-validation for our random forest:

```
param = {'bootstrap': [True],
 'max_depth': [5, 10],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [400]}
#setup cross validation
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3)
#Grid Search
rf_random = RandomizedSearchCV(RandomForestClassifier(), param, cv=3)
#Cross-validate model
scores = cross_val_score(rf_random, train[top_fs_vars], train['target'],
scoring='accuracy', cv=cv, n_jobs=-1)
```

We obtain a mean accuracy of 0.810 and proceed to obtain our performance on the test set:

	Train_AUC	Train_Accuracy	Test_AUC	Test_Accuracy
randomforest	0.861502	0.844562	0.771493	0.8145

This is our best result so far but as we can observe, the test accuracy is a bit far from our train accuracy, which could mean our model is slightly overfitted and could be further improved with a more extensive grid search.

7.6 Neural Network

Finally, we perform a grid search and cross-validation for a MLP Neural Network:

```
grid = {
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.001, 0.01]
}

cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3)

nn_random = RandomizedSearchCV(MLPClassifier(), grid, n_jobs=-1, cv=3)
# evaluate model
scores = cross_val_score(nn_random, train[top_fs_vars], train['target'],
scoring='accuracy', cv=cv, n_jobs=-1)
```

We obtain a mean accuracy of 0.808. Then, we obtain the following performance results:

	Train_AUC	Train_Accuracy	Test_AUC	Test_Accuracy
NeuralNetwork	0.861502	0.809813	0.771493	0.808

8. Conclusion and Further Steps

As an overview we can observe the following table:

	Train_AUC	Train_Accuracy	Test_AUC	Test_Accuracy
randomforest	0.861502	0.844562	0.771493	0.81450
LDA	0.754352	0.810187	0.746116	0.81125
LogisticRegression	0.761012	0.809562	0.751847	0.81125
NeuralNetwork	0.861502	0.809813	0.771493	0.80800
decisiontree	0.767660	0.809125	0.746540	0.80650

Our best model in terms of accuracy is the random forest, but since it may be slightly overfitted, perhaps its better to go with LDA or logistic regression as a more reliable model. Additionally, our neural network is not performing very well but it may because it is a very simple network with a non-extensive grid search. Perhaps by implementing more extensive hyper-parameter tuning grid search in both Neural Network and Random Forest we may obtain better results without over-fitting.

References

Burns, E. What is artificial intelligence (AI)? - AI definition and how it works. (2021, August 9). SearchEnterpriseAI. https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence

Browlee, J. Linear discriminant analysis with Python. (2020, August 2). Machine Learning Mastery. https://machinelearningmastery.com/linear-discriminant-analysis-with-python/

Browlee, J. Logistic regression for machine learning. (2020, August 14). Machine Learning Mastery. https://machinelearningmastery.com/logistic-regression-for-machine-learning/

Chandra, R. Neural networks: Applications in the real world. (2022, February 22). upGrad blog. https://www.upgrad.com/blog/neural-networks-applications-in-the-real-world/

Coussement, K., Lessmann, S., & Verstraeten, G. (2017). A comparative analysis of data preparation algorithms for customer churn prediction: A case study in the telecommunication industry. Decision Support Systems*, 95, 27-36.*

Enespolat. (2018, September 8). *Grid search with logistic regression*. Kaggle: Your Machine Learning and Data Science Community. https://www.kaggle.com/code/enespolat/grid-search-with-logistic-regression/notebook

IBM Cloud Education. What are neural networks? (2020, August 17). IBM - United States. https://www.ibm.com/cloud/learn/neural-networks

James, G., Witten, D., Hastie , T., & Tibshirani , R. (2014). An Introduction to Statistical Learning: With Applications in R . Springer Publishing Company , Incorporated

Jordan, J. Neural networks: Representation. (2018, January 26). Jeremy Jordan. https://www.jeremyjordan.me/intro-to-neural-networks/

Kumarappan, S. (2020, August 16). *Feature selection by lasso and ridge Regression-Python code examples*. Medium. https://medium.com/@sabarirajan.kumarappan/feature-selection-by-lasso-and-ridge-regression-python-code-examples-1e8ab451b94b

Malik, F. (2022, March 7). *What is grid search?* Medium. https://medium.com/fintechexplained/what-is-grid-search-c01fe886ef0a

Phan, M. (2022). Statistical & Machine Learning. [Course]. Lille: IESEG Management School. MSc in Big Data Analytics.

Prabakharan, S. LDA - How to grid search best topic models? (with examples in Python). (2018, April 4). Machine Learning Plus. https://www.machinelearningplus.com/nlp/topic-modeling-python-sklearn-examples/

Qiao, F. (2019, January 8). *Logistic regression model tuning with scikit-learn — Part 1*. Medium. https://towardsdatascience.com/logistic-regression-model-tuning-with-scikit-learn-part-1-425142e01af5

Verbeke, W., Dejaeger, K., Martens, D., Hur, J., & Baesens, B. (2012). New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. European Journal of Operational Research*, 218(1), 211-229.*