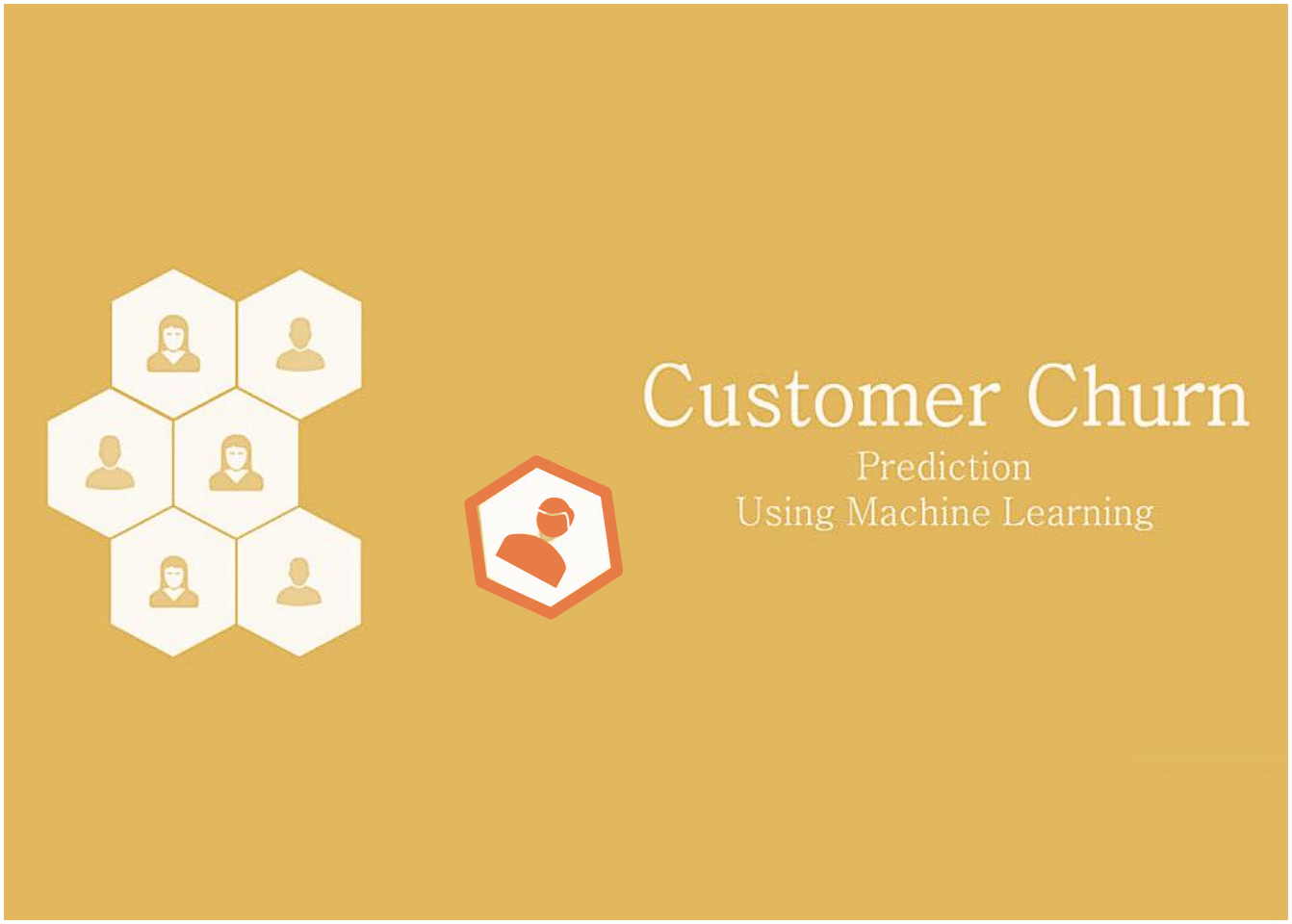

Predicting Churn Rate for Telecom Company Orange

SHRUTI BAJPAI | APRIL 18 2021 | MACHINE LEARNING

Table of Contents

INTRODUCTION

Existing studies show that retaining a customer costs less for an organization than acquiring new clients. Additionally, a customer retention of 5% can increase profit by 25% - 95%. This project utilizes information provided by the French Telecom enterprise Orange; the dataset specifically was created to be anonymized by randomizing numerical and categorical variables to create heterogenous noise and unbalanced class distributions. The goal of this project was to utilize machine learning techniques and provide predictions of each potential customer that might churn; based on the given dataset, 7.36% of customers were deemed to have churned prior to any data processing. The intention of this paper is to explain the data science methodologies used. A brief introduction of the data processing will be provided. This will be followed by a detailed description of the five machine learning algorithms utilized for this dataset.

Data Processing

The dataset provided was scrambled to prevent an understanding of the values within the data. The intent behind this was to focus more on the mechanisms and the success of the machine learning algorithms. Therefore, the first step in the data processing was to create two major categories – categorical and numerical. For both categorical and numerical data, constant variables were also removed given that they do not aid in training the dataset for better prediction rates. 38.6% of the numerical variables and 3.6% of the categorical variables of the train data had missing values; to ensure the best possible prediction, missing data were imputed using mean for both train and test data on numerical variables. All categorical variables with missing data were imputed with the value "missing" to prevent any data aspect of the data-frame from having empty values. Once all the data was imputed, outliers outside of three standard deviations were removed. After this, all categorical variables were encoded using the ordinal encoder function to have numerical values.

The next step was to then feature engineer by using Fisher's score, which helped select the most important feature by taking the mean of the non-churners and subtracting it from the mean of churning and then taking its absolute value to subtract from the square root of the variance of the difference between the non-churners against the churners. Fischer's score individually analyzed the importance of each feature to maximize the total score; a feature is deemed "good" if the variables that were dissimilar were calculated with greater variance and the variables that were similar had as little variance as possible. This helped provide a good base for the features, however the original amount of variables in the dataset were 219, including the user's identification and the churn (target) variable.

To help with dimension reduction, the principal component analysis (PCA) was utilized to analyze multicollinearity by observing outliers, trends, and clusters. The fit function of the PCA helped to calculate the eigenvectors/values to identify the principal components of the data. PCA tries to put maximum possible information in the first component, then the maximum remaining information in the second and so on. Once this is done, we then will plot the data to show the maximization of the variance → the average of the squared distances from the projected points are known as eigen vectors. Once the principal components are calculated, you then perform dimension reduction by selecting the number of principal components ($\boldsymbol{m}$). Principal components regression discards the smallest eigenvalue components and then it is easier to select the required features. This helps provide a good base-table for evaluation of churn rates by using machine learning algorithms.

Machine Learning Algorithms

While the dataset was tested and trained through eight different machine learning algorithms, this paper intends to explain only five of them to display a complete understanding of their functionality. The reason for choosing these specific methods will be described based on how successful or unsuccessful the predictions were.

I) ARTIFICIAL NEURAL NETWORK (ANN)

BRIEF DEFINITION & HISTORY

Artificial neural network or deep neural network was originally inspired by the notion of a brain's mechanisms; the neurons had dendrites, which were like layers in the algorithm that connected each node to provide an output or an action. The notion of this method was highlighted initially through Warren McCulloh's and Walter Pitts's 1943 published paper. ANN was further advanced by the Turing Test, Adeline, Madeline, and John Hopefield's suggestion of bidirectional lines; all these contributions created major advancements for neural networks. The next major advancement took place with back propagation in 1986 by three Stanford researchers looking to improve an algorithm by Widrow and Hoff. Today, advancements in neural network enable the algorithm to be used for image and object recognition, recommendation systems, and other artificial intelligence tools.

GENERAL OVERVIEW OF HOW THE ALGORITHM WORKS

Mathematically speaking, neural networks are comprised of several other functions that are expressed as a network structure. The functions are enabled through the following variables/vectors:

- Input Vector: a vector with $\boldsymbol{x}$ values or nodes from a previous layer
- Output Vector: a vector with $\boldsymbol{y}$ values or nodes of a final value
- Weight Vector: a vector with $\boldsymbol{w}$ values that represents the weights of the connection between the input node and the final node; is used to not only activate the various functions within neural network, but also provide impact of the weight
- Intermediary Vector: a vector with $\boldsymbol{H}$ values that
- Bias Vector: a vector with $\boldsymbol{B}$ values not connected to any previous layer or input; prevents the training data from overfitting; In thinking of the linear algebraic equation $\mathbf{y = mx + b}$, $\boldsymbol{m}$ is the constant and $\boldsymbol{b}$ is the linear intercept that helps the formation of the line; similarly, for neural network, $\boldsymbol{m}$ serves as the weight while $\boldsymbol{b}$ serves as the bias

The neural network algorithm works by having each input ($\boldsymbol{x}$) connects to an intermediary ($\boldsymbol{H}$) by summing the input vector with $\boldsymbol{B}$ values and then multiplying each sum with the total weights ($\boldsymbol{w}$). The bias vector act as a constant to help fit the weighted sums of the input for each neuron or node; this showcases the individual neural functionality (Figure 1). The intermediary vector continues to utilize functions through the weights which help connect to another layer and each intermediary layer is curbed to prevent overfitting with $\boldsymbol{B}$ values that are added to each intermediary layer ultimately until there are no more intermediary layers, which provide an output layer (Figure 2).

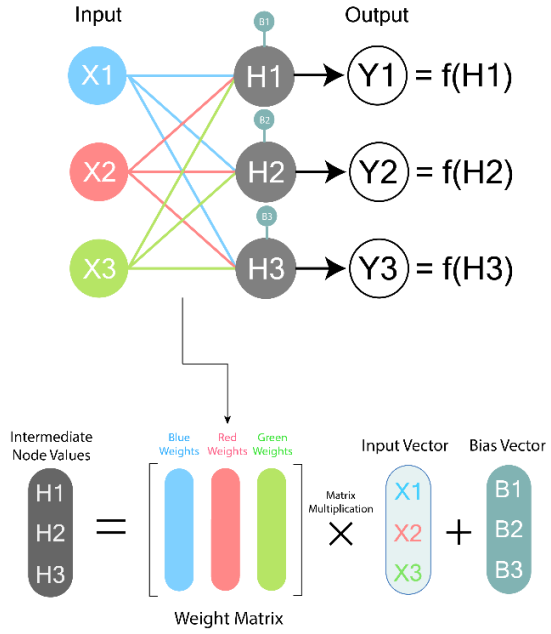

Figure 1: Individual Neural Functionality

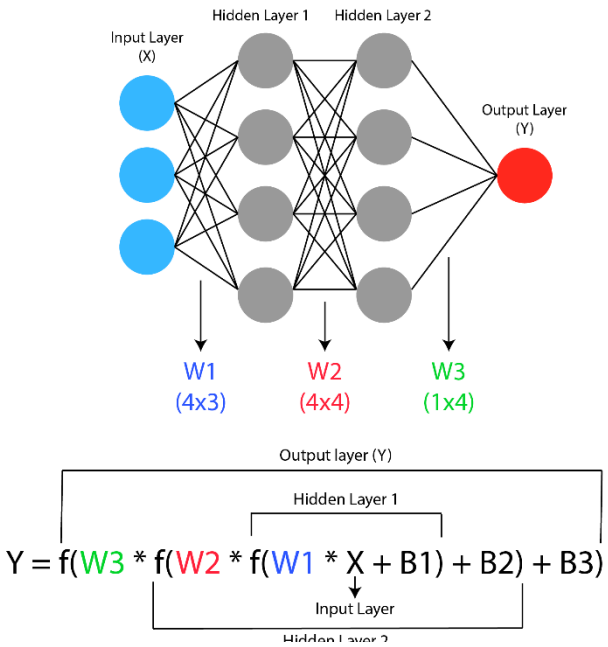

Figure 2: Complete Neural Network

FITTING NEURAL NETWORK WITH THE ORANGE DATASET

Backpropagation is a common method used for a classification problem such as this one. The first step required fitting the basetable that is split into x input values and y output values; as the data was used to compare against the final output in Kaggle, the predictions were completed on the final dataset known as test_pca. Once the data was fitted, the predict and predict_proba functions were used on the test_pca table, which is the table to test if a customer would churn or not. In analyzing how the functions of the neural network worked, it is important to compare the y_pred (predicted value) vs. the actual value y and analyze how often the result was correct. The AUC of the test split of the trained data showed a 0.9117, while the accuracy score was 100%, which positively indicated that the algorithm would accurately churners vs non-churners. However, when implementing this algorithm on the test_pca table against the actual values (through submission of Kaggle), the AUC was 0.63523; this signified that the predicted values either overestimated or highly underestimated the customer that might churn and indicates a larger error that is associated with methods of backpropagation in terms of the partial derivates and weights allotted to each y value.

ADVANTAGES & DISADVANTAGES

In using such techniques and noting that it was not successful in predicting, it is important to understand the advantages and disadvantages of the neural network algorithm.

Advantages:

1. Robust – this algorithm was very useful for the orange dataset as it worked regardless of missing data, incomplete data or even training data with significant errors.
2. Computational Power – this algorithm was able to handle the Orange dataset that consisted of so many variables and so many values.

Disadvantages:

1. Difficulty of understanding the algorithm's function - given the various functions and complexity of the algorithm, if the output values overpredict or underpredict, it is hard to pinpoint which it is and where exactly the algorithm went wrong. For this reason, it was not easy to understand exactly if there was an error that required more imputation in the preprocessing or processing of the data or if the weights assigned to the input and intermediary values led to high errors on the output vs. the predicted output of the data.
2. Quantity of Data - ANN is infamously known to require a very high volume of data to provide correct prediction; in the context of this dataset, there most likely was not enough data, which led to overfitting.

II) NEAREST NEIGHBORS (KNN)

BRIEF DEFINITION & HISTORY

According to Towards Data Science, the algorithm KNN can be best described as “Birds of a feather flock together.” In essence, the closeness a data point to another data point plays a significant role in gauging churn. Dr. Evelyn Fix and Dr. Joseph Lawson Hodges Jr. were two statisticians that collaborated in 1951 to introduce the idea of non-parametric classification method (discriminant analysis), which later became known as the k-NN method. This method was further advanced in 1967 by Cover and Hart’s paper on bound error rates with a multiclass k-NN classification. This paved the way for many other researchers such as Dudani, Jozwik, and Keller to enhance the algorithm.

GENERAL OVERVIEW OF HOW THE ALGORITHM WORKS

KNN as an algorithm calculates a particular point ($\boldsymbol{x}$) in the dataset based on Euclidean distance (or other similar methodologies that calculate distance) and determines the distance of each point from point $\boldsymbol{x}$. A step by step understanding of this can be evaluated through the Euclidean distance’s mathematical formula that is shown below:

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=0}^{n}(q_1 - p_1)^2 + (q_2 - p_2)^2 + ..} = \sqrt{\sum_{i=0}^{n}(q_i - p_i)^2}$$

Figure 3: Euclidean Formula

While seeming computationally difficult, the Euclidean distance is calculated by taking a specific value/point/observation from the dataset and subtracting it from each row below (in a dataset). Once subtracted, the value is squared to obtain the actual distance from each point without carrying the meaningless negative sign as the negative sign holds no specifically relevant information on the distance. This process continues as each point within the dataset is subtracted from each other and squared; once subtracted and squared, each point is them summed together and then square rooted to help formulate the distance between each point. The Euclidean distance is then sorted to understand the smallest distance to obtain the $\boldsymbol{k}$ smallest distances of each point from each other to help classify the groups.

In using the image below, it can be seen how KNN is visually executed. Inputting the value for $\boldsymbol{k}$ = 3, the algorithm looks to find the three datapoints nearest to the center point (the blue star). The Euclidean distance is then taken from each point (red and green) to the star. This process is repeated to help the algorithm calculate the points that have been considered and the datapoints that have been excluded.

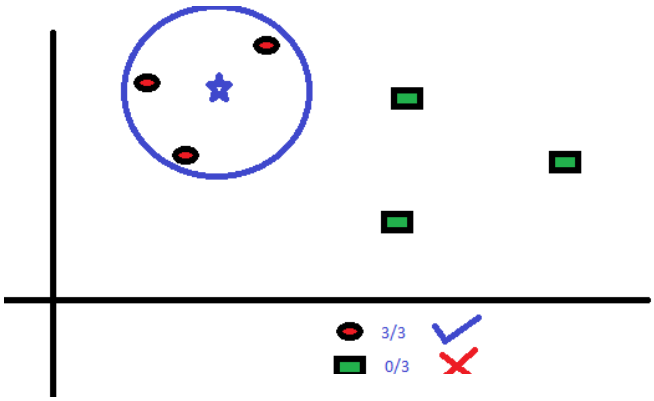

Figure 4: KNN Algorithm (K=3)

FITTING NEAREST NEIGHBORS (KNN) WITH THE ORANGE DATASET

The initial step of fitting the KNN algorithm begins with utilizing the Standard Scalar functionality in python. Given KNN as an algorithm calculates the distance, it is quite important to ensure that the data is standardized as to not have skewed data be used for prediction. Therefore, the scalar function was employed to normalize the data to have a mean of 0 and a standard deviation of 1. The KNN function was then employed to help calculate the Euclidean distances; in this situation, given the large dataset, the default settings were chosen to analyze how well the algorithm would run without specifying parameters. In other words, Euclidean distance was used rather than Manhattan or Cosine and the default value of K was used (K=5). Similar to the Neural Network algorithm, predict and predict_proba were functionalities also built in to be used for the KNN algorithm and predict churn on the test_pca table, which is the table to test if a customer would churn or not. When training the data, the AUC of the test split of the trained data showed a 0.9117, while the accuracy score was 0.9293, which positively indicated that the algorithm would accurately churners vs non-churners. However, when implementing this algorithm on the test_pca table against the actual values (through submission of Kaggle), the AUC was 0.53507; this signifies that the method to classify k might have excluded more or included too many extraneous datapoints. Manual computation of the K groups alongside a grid search might have helped improve this AUC score.

ADVANTAGES & DISADVANTAGES

Similar to Neural Network, KNN was not successful at predicting churn rates; however, unlike Neural Network, KNN was even more unsuccessful. Evaluating the advantages and disadvantages of this method might provide insight as to why the prediction was only partly successful at predicting churners.

Advantages:

1. Robust and Easy Implementation – similar to neural network, the KNN algorithm is easy to implement in that it can continue to calculate predictions even if new data is added or existing data is modified.
2. Easy to Understand & Simple Hyperparameters – this algorithm is very different from neural network in that if one had the time, the values could be calculated to create predictions manually, making it easy to understand. The additional feature of simply classifying the K values is also an added feature helpful for quick computation with such a large dataset. Rather than having to calculate k manually, it is easy to adjust K parameters and see how well the prediction fares.

Disadvantages:

1. Difficulty with Large Datasets – while having the ability to calculate KNN can be quick and easy, this dataset was quite large, making the total time to calculate each prediction longer. If the accuracy was low, the hyperparameters are easy to use and modify, however time constraints make it limited to really change the hyperparameters. For this reason, this algorithm might not have fared well with such a large dataset.
2. Sensitivity to Outliers – the preprocessing is a crucial aspect as KNN does not fare well with outliers. It gives equal importance to each datapoint regardless of the data having outliers. This makes it difficult because if there are a lot of outliers, with the data being intentionally scrambled to make it hard to understand the data, which makes it even more difficult to decipher the outliers.

III) RANDOM FOREST

BRIEF DEFINITION & HISTORY

Imagine a tree with a large trunk; as the tree grows it has branches and twigs that give it additional dimension. The Random Forest algorithm takes a similar ensemble approach by building “trees” of uncorrelated datapoints/decisions to help provide the best prediction. In 1995, Ho proposed a method that focused on enhancing the training data through oblique hyperplanes that would reduce the chances of overfitting. In other words, he provided a method for decision trees to delve into deeper analysis without training the data too well, which was also addressed in Kleinberg’s Theory of Stochastic Discrimination. The algorithm was then furthered by Amit and Geman’s 1997 proposal, which attempted to approach a better prediction rate through the shape of the dataset and sub-setting of the features. Ho then again made another proposal in 1998 to help increase accuracy, while also reducing the chances of overfitting. While there have been more advancements for the algorithm, it can be agreed upon that the most common use cases for this

algorithm are seen in sectors/industries such as: telecom, banking, medicine, e-commerce, and the stock market.

GENERAL OVERVIEW OF HOW THE ALGORITHM WORKS

Random Forest is built upon the notion of decision trees. There lies a decision node that is separated into two sub-decision trees with two leaf nodes; decision nodes provide the decision made, while the leaf nodes provide the output or the consequences of those decisions. The first image on the left (Figure 5) shows a decision tree broken down to help visualize this, while the second image below on the right (Figure 6) shows an execution of a random forest algorithm that consists of several decision nodes.

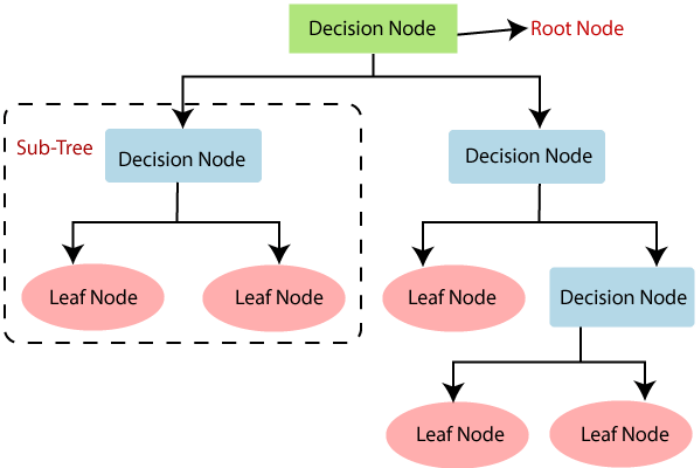

Figure 5: Decision Tree Nodes

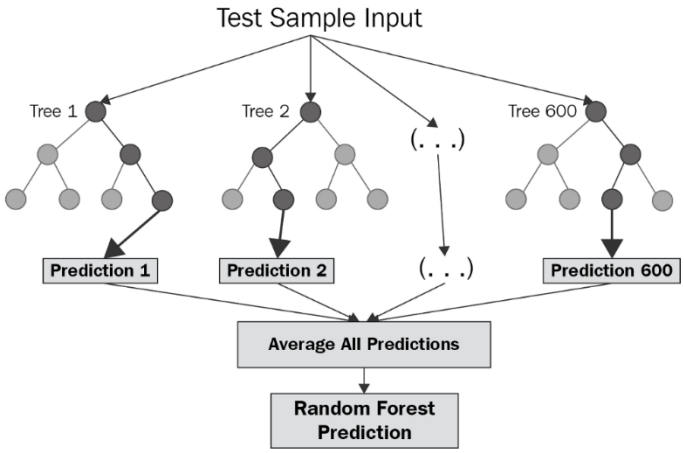

Figure 6: Random Forest Algorithm Visualized

The difference between a decision tree and random forest is most simply the fact that decision trees maintain a singular root node while the random forest algorithm outputs several root decision nodes; it considers a root node that is typically created by a decision tree and splits the root nodes into decision nodes and sub-nodes ito add additional trees.

Initially an empty vector is created where there is a randomized sample of the training dataset. The root node is split through the process of Gini Importance, also known as Mean Decrease in Impurity (MDI). Towards Data Science best explains MDI as the "*calculate[ion of] each feature importance [based on]* ***the sum over the number of splits*** *(across all tress) that include the feature, proportionally to the number of samples it splits.*" In other words, the total number of features are summed and then square rooted based on the sample size taken and the features taken into consideration; based on this value, the root node of the decision tree is then further split based on the randomized sample and their respective nodes and sub-nodes.

The split essentially creates vectors of the sample data taken and finds the best split, which is calculated based on feature importance. Feature importance is calculated by analyzing the decrease in the sameness/homogeneity (which is addressed by MDI) of the node against the overall probability of attaining the node (known as node probability). The node probability sums the total number of samples that reach a specific node and divides it by the total number of samples. This feature importance helps create the split which is based off weight provided or calculated based on the average of all trees calculated through the equation below:

$$RFfi_j = \frac{\sum j\ (all\ trees\ norm\ fi_{ij})}{T}$$

Figure 7: Importance of Features Calculated through Random Forest

The equation shows the importance of all the features calculated $RFfi_j$ by summing the all the trees and their tree (***j***) and multiplying it with the sum of the normalization of each feature (***i***) and dividing that total by the total number of trees (***T***).

FITTING RANDOM FOREST WITH THE ORANGE DATASET

Unlike the KNN algorithm where the Standard Scalar functionality was used, the Random Forest method did not use the standard scalar method as this algorithm was not as easily affected by outliers as the KNN algorithm. The Random Forest was similar in that this algorithm was fitted based on present values just as Neural Network was. The present functions within sklearn utilized 100 trees as the default estimators for the total trees in the algorithm. Furthermore, the default criterion used was Gini Index (MDI) as discussed previously to help with minimizing the split in order to reduce the chances of inaccurately classifying the node's distribution from the random nodes selected. The splits were also defaulted from sklearn and maintained as random splits, with no maximum depth of any and all trees. In essence, the algorithm was simply fitted using the train_x and train_y datasets to help the algorithm learn. The predict and predict_proba were functionalities also built in to be used for the Random Forest algorithm and predict churn on the test_pca table, which is the table to test if a customer would churn or not. When implementing this algorithm on the test_pca table against the actual values (through submission of Kaggle), the AUC was 0.61842. Given that the train was higher than the validation and test set, there is indication of the random forest overfitting, which might be a result of using the default parameters rather than indicating what this value might be.

ADVANTAGES & DISADVANTAGES

Random Forest did not predict as successfully, which is similar to the Neural Network and KNN. Evaluating the reasoning behind this through the advantages and disadvantages of the algorithm might help give more insight on the algorithm itself and why it may not have been successful.

Advantages:

1. Efficient on Large Datasets – similar to neural network, Random Forest is quite useful for large datasets; with this dataset originally having 200 possible features, this algorithm was a great choice to see how it would fare with such a large dataset.
2. Robust to Outliers – unlike KNN, the functionality of the algorithm showed the ability to normalize the data. Despite the data processing, there could easily be several outliers that make it difficult to account for given the variables in the dataset were scrambled. This algorithm was intended to address this potential issue of having outliers in the data.

Disadvantages:

1. Biased with Categorical Variable – while the data provided has several categorical variables with observations in numerical value, understanding what the categorical data signified was difficult. This subjected to problems if one-hot encoding was not used. In this particular situation and the methodology of the data analysis, one-hot encoding was not used and therefore served as a potential reason for why the algorithm did not predict as well as it could have.
2. Prone to Overfitting – with higher amounts of trees being added, there is a large chance of overfitting due to an inverse in the relationship between increase in value of trees, while decreasing in value of the generalization error as it approaches 0. For this reason, tuning the hyper-parameter is extremely important; as no hyper-parameter was tuned and all default settings were used, this may explain the reason for the high generalization error and low accuracy in prediction.

IV) LOGISTIC REGRESSION

BRIEF DEFINITION & HISTORY

Logistic Regression is quite similar in its approach to linear regression, but more so uses the logistic function for most binary classification problems. The history of logistic regression began back in the 19th century during the invention of the logistic function which was specifically to calculate the growth of the population and analyze chemical reactions. However, the invention was not very highly known. A second wave renewed the logistic function during the United States' study of their population by Pearl and Reed. These two researchers went on to continue the application of the function in various populations, human or otherwise. Other major contributors to enhancing the logistic function were Wilson and Worcester and Joseph Berkson.

The next major contributor came during 1966 with Cox inventing multinomial logit models that were similarly initially proposed by Joseph Berkson. With Cox's contributions alongside McFadden's contributions in 1973 to connecting the logit model with discrete theory, the algorithm greatly advanced and is used in various fields. Today, Logistic Regression is well known to be used in credit card fraud, tumor prediction, marketing, telecom, and many more similar areas.

GENERAL OVERVIEW OF HOW THE ALGORITHM WORKS

In understanding the algorithm, it is important to understand that this is best served where binary outputs are required; in other words, the output value of Y will result in either a value of 0 or a value of 1. This begins with the calculation of the maximum likelihood function, which analyzes the maximum likelihood (estimation) by multiplying the individual likelihood of each observation in the dataset.

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i))$$

Figure 8: Maximum Likelihood Function

Once calculated, the function then converts the probability into a binary value through the logistic function or the sigmoid function (left), which can be more easily related to based on the visual on the right.

$$S(x) = \frac{e^x}{e^x + 1}$$

Figure 9: Logistic/Sigmoid Function

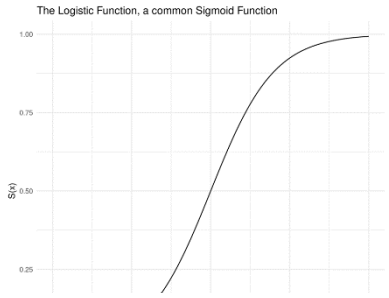

Figure 10: Graphing of the Sigmoid Function

The sigmoid function calculates the dataset to be between 0 and 1 or between -1 and 1 in order for it to be interpreted as the probability. As x becomes larger, the total value of $S(x)$ comes closer and closer to 1; this function provides flexibility in this manner as the value of $S(x)$ can near 0, 1, or -1, but never be larger providing limitation of the value, while also ensuring that the slope does not equal 0. The image below showcases the steps to provide a proper visual of how logistic regression is implemented.

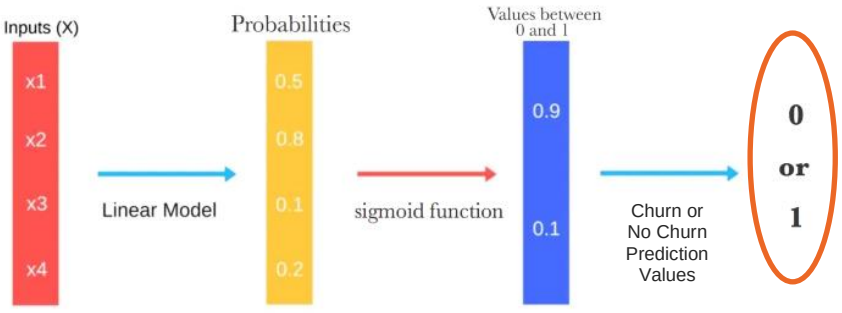

Figure 11: Showcasing the Process of Logistic Regression Calculation

FITTING LOGISTIC REGRESSION WITH THE ORANGE DATASET

In line with neural network and random forest, standardization was also not utilized. Similarly, pre-existing parameters of the Logistic Regression function in the sklearn package were used. Total weights are an important factor to calculating the probability of each value in the maximum likelihood function; as such, class_weight is an important hyperparameter that was not defined when training the data, therefore the weight value was listed as one. In addition, other hyperparameters such as penalty, dual, fit_intercept, and intercept_scaling were also not defined. In other words, similar to the previous algorithms, the hyperparameters were not further defined or tuned in any way to see how the function would work without fine-tuning. Just as the other algorithms, Logistic Regression was fitted fitted using the train_x and train_y datasets to help the algorithm learn. The predict and predict_proba were functionalities also built to predict churn on the test_pca table, which is the table to test if a customer would churn or not. The training of this data showed a 0.996891 for AUC score and 0.958833 for Accuracy. This indicated that if the data was tested against the Kaggle, it would perform quite well given the accuracy of a dataset indicated the amount of times a prediction was right over the total prediction. When implementing this algorithm on the test_pca table against the actual values (through submission of Kaggle), the AUC was 0.68382. The training indicated correctly as the AUC decently performed, particularly since this was the highest AUC score in all the Kaggle submissions.

ADVANTAGES & DISADVANTAGES

Unlike the other previous algorithms, Logistic Regression performed fairly well without any modification of the hyperparameters. With that being said, the AUC score could have potentially been higher and better; therefore, it is important to understand the advantages and disadvantages of using this algorithm in general as well as with the given dataset.

Advantages:

1. Simpler & More Efficient Algorithm to Implement – unlike neural network, Logistic Regression is similar to Random Forest in that it does not require large computation power and is simple to implement. As a result, it is highly noted as one of the more successful algorithms in machine learning. Understanding the algorithm in addition to having this knowledge helps provide insight as to why this machine learning algorithm has been the most successful with the given dataset.
2. Does Not Require Fine Tuning of The Hyperparameters – unlike KNN, the functionality of the algorithm in its implementation using the sklearn package does not require fine tuning of the hyperparameters. This aids to the efficiency in calculation and also does not require too much time focusing on the hyperparameters as the reason for the model to be unsuccessful.

Disadvantages:

1. Reliance on Data Processing and Feature Engineering – data processing plays a key role in the accuracy of this algorithm being successful; the features used will provide a better insight as to how successful the prediction will be. This data originally went through a series of data processing and feature engineering; pinpointing if imputation was the reason for the AUC not being higher or if there was some other reason might be difficult for such a large dataset.
2. Functions Best with Discrete Data – with the inability to understand the data, it would be hard to denote a specific variable or series of variable as discrete or continuous. There may be an instance where categorical data can be modified to be addressed as discrete values, but that is not as easy given the data is scrambled and it is hard to interpret. Removing all categorical data also would not guarantee that the other variables would be discrete; the inability to do this might explain why the AUC did not perform as highly as hoped for.

V) SUPPORT MACHINE VECTOR (SVM)

BRIEF DEFINITION & HISTORY

Also known as support vector network, Support Machine Vector is a non-binary linear classifier that analyzes trained data to calculate and classify new data into one of the two categories. Aronszajn's 1950 publish of "The Theory of Reproducing Kernals" alongside Frank Rosenblatt's 1957 invention of perceptron (which is a simple linear classifier method) became the chain of events that eventually lead to Vapnik's work with this algorithm. SVM was initially proposed by Vapnik and Chervonekis in 1992, which continued to be enhanced throughout their work including Vapnik's Statistical Learning Theory in 1998; this theory is said to have been based off the initial notions of the SVM algorithm. These inventions and knowledge have been pooled together to create the flexible algorithm that is SVM today, which is widely used in face detection, handwriting recognition, and even the image classifications.

GENERAL OVERVIEW OF HOW THE ALGORITHM WORKS

The process of calculating a prediction in SVM, which begins with creating a N-dimensional hyperplane, followed by deciphering which datapoints belong into which of the two categories. Support vectors are reliant on the notion of hyperplanes; hyperplanes are essentially lines that assist in segregating and classifying the data. These hyperplanes are important in SVM modeling because they help graph the support vectors datapoints nearest to the hyperplanes are mapped out; if removed, the hyperplanes would need to be altered. A hyperplane that requires a linear classifier is known as a decision hyperplane that are perpendicular to decision hyperplane vectors (also known as weight vectors and denoted as w^T). Margins are defined as the distance between the two nearest datapoints on either side of a hyperplane (see image below, Figure 12).

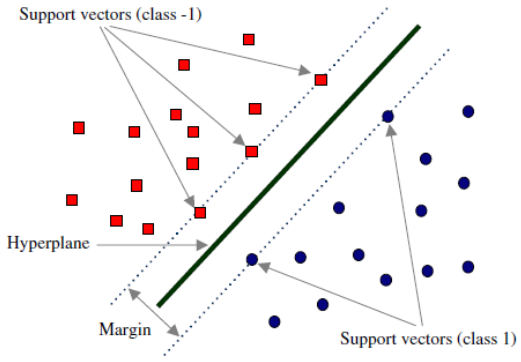

Figure 12: SVM & Key Terminologies

The goal of accurately predicting with new data (test data) becomes more precise as the primary intent is to decide the hyperplane best suited for a datapoint based on the greatest margin between the datapoints and the hyperplane. This can be mathematically expressed as seen in Figure 13.

$$d_H(\phi(x_0)) = \frac{|w^T(\phi(x_0)) + b|}{||w||2} \qquad given\ by \qquad w^* = arg_w \max[min_n d_H(\phi(x_n))]$$

Figure 13: Hyperplane Equation & the Objective Function of the Maximization of the Minimum Margin

When these datapoints on the hyperplane are plotted, they are typically plotted on a hyperplane with positive and negative values; it can be easily visualized in the 2D thought of a x and y axis with positive and negative values. The equation on the left shows a stochastic differential equation for a hyperplane equation that states the distance on hyperplane of phi on point x is based on the notion that if the point on the hyperplane is positive, then $\boldsymbol{w^T(\Phi(x)) + b > 0}$ must also be greater than zero. However, if the point on the hyperplane is negative, the training data must adhere to a similarly inversed rule that the value will be less than zero given $\boldsymbol{wT(\Phi(x)) + b < 0}$. In other words, the points above a hyperplane might be classified as +1, while points below a hyperplane will be classified as -1. Given the datapoints used will be calculated on the training data, the testing data must also follow a +1 and -1 rule, which is important when multiplying the overall predicted data values and them summing them. This will aid in understanding the comparison of how many predictions were accurately classified in the correct hyperplane.

This description of maximizing the minimum margins of the support vectors for training datapoints on hyperplanes is based on the notion that the hyperplane is a straight line; in other words, the datapoints classifications are quite critical into predicting if new datapoints will be classified the same way. The primary method stated above was based on a general idea of the hyperplane being a straight line and datapoints being classified on one side or the other of a hyperplane. However, the SVM algorithm may not always easily classify datapoints correctly (as churn or not churn); misclassification can occur with an optimal hyperplane classifying a data incorrectly. As such the notion of a non-perfect separation can be modified as seen in Figure 14. The

equation's restructuring adapts the way in which the prediction and actual values are compared; the non-perfect separation of the datapoints on a hyperplane allows the datapoints to be represented in the form of beta such that all data correctly classified are given a value of 0, while all incorrectly classified data are given a value of 1.

$$Instead\ of\ y_n \left[w^T \phi(x_n) + b \right] > 0 \ ; \ \forall n$$
$$Equation\ becomes\ y_n \left[w^T \phi(x_n) + b \right] \leq 0 \ , \ \exists n$$

Figure 14: Non-Perfect Separation of Datapoints on an Optimal Hyperplane

While Figure 12 shows a 2D image with only one hyperplane, the typical dataset is quite more complex and requires analyzing the dimensions of the datasets as they are divided using the process of Kernelling. Kernelling essentially helps with pattern analysis by transforming the training data (or raw data) into a vector (known as a feature space) without having to calculate the coordinates of data in a higher dimensional space. This helps save computational power as the data becomes more complex. Taking the example below, x and y are two datapoints that must be mapped from a three dimensional space to a nine dimensional space. A traditional method (Figure 15 on the left) showcases the computational complexity, while the kernel trick (Figure 16 on the right) showcases the dot-product computation to transpose x and y in a three dimensional space for the nine dimensional space. This in essence shortens the computational power required by calculating in the original dimensional space, rather than having to calculate for a conversion into another higher dimensional space.

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$
$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

Figure 15: Non-Kernal Computation

$$\begin{aligned} k(\mathbf{x}, \mathbf{y}) &= (\mathbf{x}^T\mathbf{y})^2 \\ &= (x_1y_1 + x_2y_2 + x_3y_3)^2 \\ &= \sum_{i,j=1}^{3} x_ix_jy_iy_j \end{aligned}$$

Figure 16: Kernel-Trick

The overall process of SVM can as such be summarized similarly based on the steps below. For each project, the vectorization will be critical as that is where the training datapoints are. Taking the description of each datapoint is essentially noted as the data processing and provision of a training dataset. The training data will then be calculated into a normalized vector to help reduce any outliers. The data will then learn by taking the input value (x) and output value (y), which will be inputted in the SVM algorithm where the maximization of the minimum vector will be calculated, the hyperplane will be created, and the data will be classified based on non-perfect and perfect separations of the calculated optimal hyperplanes. Once completed, the kernel trick will be used to ultimately help calculate vectors and scalars for which predictions can be made based on the decision rule.

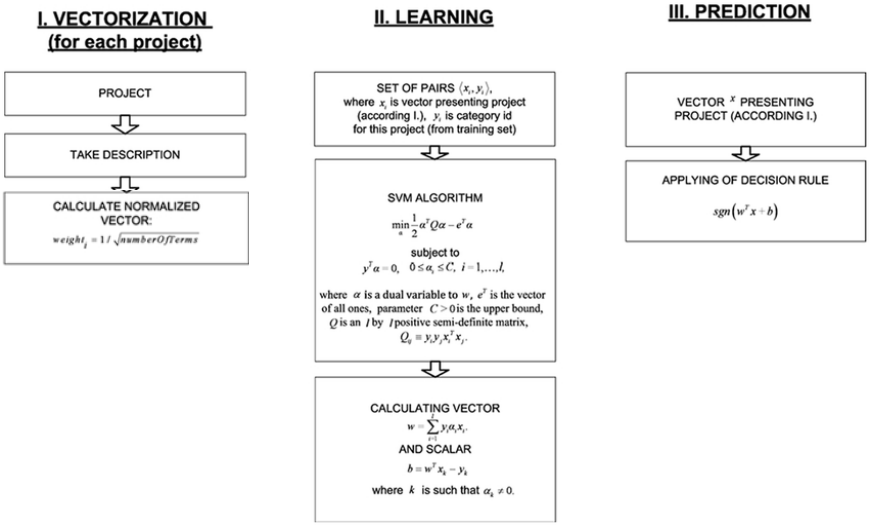

Figure 17: The Process of SVM

FITTING SVM WITH THE ORANGE DATASET

In line with neural network and random forest, standardization, while recommended based on the dataset – was not used. The SVC classifier was used from sklearn where linear was the method chosen to analyze the data given the question that needed to be addressed was churn and no-churn. To calculate the probability estimates, the SVC probability was calculated as True. The training data was then fit using the pre-existing parameters. For example, kernel was kept at the default of 'rbf' which is the most common method for classification; using gamma or other kernel methods might cause incorrect calculations that would lead to overfitting when predicting. As not that much was known about the data, it seemed like the best method to maintain the default parameters and proceed with using the predict and predict_proba functionalities to predict churn on the test_pca table. The training of this data showed a 0.9265 for AUC score and 1.0 for Accuracy. This indicated that if the data was tested against the Kaggle, it would perform quite well given the accuracy of a dataset indicated the amount of times a prediction was right over the total prediction. When implementing this algorithm on the test_pca table against the actual values (through submission of Kaggle), the AUC was 0.50327. It was the lowest performing algorithm among all the Kaggle submissions, which indicated how poorly the was able to analyze customer churn prediction.

ADVANTAGES & DISADVANTAGES

Given how poorly the dataset performed, it is quite critical to analyze what advantages SVM brings when utilizing and the disadvantages that may have caused such poor performance.

Advantages:

1. SVM Provides Memory Efficiency – as a result of the kernel trick, SVM is able to transform datapoints for calculation more effectively without removing the original state of the datapoint. For example, $\boldsymbol{x}$ can become $\boldsymbol{x_i}$. In thinking of using this algorithm, it was thought that this efficiency would help in being more accurate with prediction given that it would utilize points from the training dataset.
2. Faster Prediction – unlike the previous algorithms, SVM takes time to predict for training large datasets; however, it is quick for prediction. With a dataset originally with 200 variables, the idea of having the ability to quickly predict would be a great implementation for a telecom organization such as Orange if the predictions were accurate.

Disadvantages:

1. SVM is Mostly Suitable for Smaller Datasets – given how large this dataset was and the inability to have scalability, it was probably not the best idea to utilize this algorithm.
2. Requires Hyperparameter Tuning – unlike logistic regression, hyperparameter tuning is quite important for this dataset; while defining the SVC as linear, tuning parameters such as class_weight and max_iter were two important parameters. Additionally, while radial based function kernel (rbf) was chosen, it was perhaps a better choice to try sigmoid or poly to see how the values would have affected the prediction if at all.

Conclusion

The primary objective of this paper was to provide insight on the various machine learning methodologies employed through the telecom orange dataset to create the highest churn prediction rate possible based on AUC. While the highest AUC score was calculated using logistic regression, there are several methods that can be used to enhance the AUC score. Resampling method with bootstrapping by choosing a subset sample of the dataset to apply to this method could have helped with increasing prediction as there were many more observations that might have caused a well functioning tool such as neural network to provide such a low score. In addition, using AUC may not have been the best in gauging which machine learning algorithm would best predict the actual churn rates. Precision and recall are two alternative methods that could have been employed; precision and recall are both binary metrics. Precision precisely takes the true positive and divides it by the sum of the total predicted true positive plus the false positive. Recall is slightly different in that it takes the true positive and divides it by the sum of the total predicted true positive against the false negatives. An analysis of the data that overfitted could provide insight into if there were too many churners predicted or too little churn rates predicted; this could help indicate if recall would have been a better measure than AUC or accuracy. Additionally, all of the default parameters were used for the five algorithms; not only could this have been changed to optimize the algorithms, but also tuning the hyperparameters would have also helped. Overall, this was an enlightening process that gave insight into how the algorithms worked in their original state and how they could have been enhanced. A further analysis could be done on the same dataset to see if using the default parameters and focusing on the data processing methods could help enhance the churn prediction.

References

A. (2018, August 17). *Random forest analysis in ML and when to use it*. NewGenApps. https://www.newgenapps.com/blog/random-forest-analysis-in-ml-and-when-to-use-it/#:~:text=Random%20forest%20algorithm%20can%20be,a%20large%20proportion%20of%20data

Abraham, J. (2019, September 26). *A beginner's guide to K nearest Neighbor(KNN) algorithm with code*. Medium. https://medium.com/analytics-vidhya/a-beginners-guide-to-k-nearest-neighbor-knn-algorithm-with-code-5015ce8b227e

Advantages and disadvantages of artificial neural networks. (2020, August 22). Asquero. https://www.asquero.com/article/advantages-and-disadvantages-of-artificial-neural-networks/

Bambrick, N. (2016). *Support vector machines: A simple explanation*. KDnuggets. https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

Bommana, H. (2020, April 25). *Introduction to neural Networks — Part 2*. Medium. https://medium.com/deep-learning-demystified/introduction-to-neural-networks-part-2-c261a99f4138

Brownlee, J. (2019, October 27). *A gentle introduction to logistic regression with maximum likelihood estimation*. Machine Learning Mastery. https://machinelearningmastery.com/logistic-regression-with-maximum-likelihood-estimation/

Brownlee, J. (2020, August 15). *Logistic regression for machine learning*. Machine Learning Mastery. https://machinelearningmastery.com/logistic-regression-for-machine-learning/

Cramer, J. S. (2002, November). *The Origins of Logistic Regression*. Tinbergen Institute Discussion Paper. https://papers.tinbergen.nl/02119.pdf

DataFlair Team. (2018, November 16). *Real-life applications of SVM (Support vector machines)*. DataFlair. https://data-flair.training/blogs/applications-of-svm/

Decision tree classification algorithm. (n.d.). www.javatpoint.com. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

Donges, N. (2018, April 23). *The logistic regression algorithm*. machinelearning-blog.com. https://machinelearning-blog.com/2018/04/23/logistic-regression-101/

Donges, N. (2019, June 16). *A complete guide to the random forest algorithm*. Built In. https://builtin.com/data-science/random-forest-algorithm

Fawagreh, K., Gaber, M., & Elyan, E. (2014, October 6). *Random forests: From early developments to recent advancements*. Taylor & Francis. https://www.tandfonline.com/doi/full/10.1080/21642583.2014.956265

Goel, A. (2018, May 21). *4 logistic regressions examples to help you understand*. Magoosh Data Science Blog. https://magoosh.com/data-science/4-logistic-regressions-examples/

Gokte, A. (n.d.). *Most popular distance metrics used in KNN and when to use them*. KDnuggets. https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html

Harrison, O. (2019, July 14). *Machine learning basics with the k-nearest neighbors algorithm*. Medium. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761#:~:text=KNN%20works%20by%20finding%20the,in%20the%20case%20of%20regression)

The Holy Python. (2020, July 29). *Logistic regression history*. HolyPython.com. https://holypython.com/log-reg/logistic-regression-history/

The Holy Python. (2020, July 27). *K nearest neighbor (kNN) history*. HolyPython.com. https://holypython.com/knn/k-nearest-neighbor-knn-

history/#:~:text=In%201983%3A%20Adam%20Jozwik%20introduced,k%2Dnearest%2Dneighbour%20classifiers

HolyPython. (2020, July 29). *Support vector machine history*. HolyPython.com. https://holypython.com/svm/support-vector-machine-history/

Holypython. (2021, March 28). *Support vector machine pros & cons*. HolyPython.com. https://holypython.com/svm/support-vector-machine-pros-cons/

Hugo Mayo. (n.d.). *History of machine learning*. Department of Computing | Faculty of Engineering | Imperial College London. https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html#:~:text=1952%20saw%20the%20first%20computer,was%20pattern%20and%20shape%20recognition

José, I. (2019, June 26). *KNN (k-nearest neighbors) #1*. Medium. https://towardsdatascience.com/knn-k-nearest-neighbors-1-a4707b24bd1d

Kumar, N. (2019, March 2). *Advantages and disadvantages of logistic regression in machine learning*. The Professionals Point. https://theprofessionalspoint.blogspot.com/2019/03/advantages-and-disadvantages-of.html

Kunchhal, R. (2020, December 28). *The mathematics behind support vector machine algorithm (SVM)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/the-mathematics-behind-svm/#:~:text=A%20Support%20Vector%20Machine%20or,to%20as%20Support%20Vector%20Classification

Lee, C. (2020, September 8). *Feature importance measures for tree models — Part I*. Medium. https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3#:~:text=Gini%20Importance%20or%20Mean%20Decrease%20in%20Impurity

%20(MDI)%20calculates%20each,number%20of%20samples%20it%20splits.&text=(weighted%20by%20the%20number%20of%20samples%20it%20splits)

ML Nerds. (2020, October 26). *How does KNN algorithm work ? What are the advantages and disadvantages of KNN ?* Ace the Data Science Interview!. https://machinelearninginterview.com/topics/machine-learning/how-does-knn-algorithm-work-what-are-the-advantages-and-disadvantages-of-knn/

MLMath.io. (2019, February 16). *Math behind support vector Machine(SVM)*. Medium. https://medium.com/@ankitnitjsr13/math-behind-support-vector-machine-svm-5e7376d0ee4d

Płoński, P. (2019, April 5). *Does random forest overfit?* MLJAR Automated Machine Learning. https://mljar.com/blog/random-forest-overfitting/#:~:text=The%20Random%20Forest%20algorithm%20does,the%20algorithm%20should%20be%20tuned

ResearchGate. (2016, February 16). *Figure 3. Workflow of the support vector machine (SVM) algorithm...* https://www.researchgate.net/figure/Workflow-of-the-Support-Vector-Machine-SVM-algorithm-implementation-A-Vectorization_fig4_51531316

Ronaghan, S. (2019, November 1). *The mathematics of decision trees, random forest and feature importance in scikit-learn and spark*. Medium. https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3

scikit learn. (n.d.). *3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.23.2 documentation*. scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation. Retrieved April 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Scikit-learn. (n.d.). *Sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.23.2 documentation*. scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation. Retrieved April 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Sekhar, S. (2019, August 30). *Math behind logistic regression algorithm*. Medium. https://medium.com/analytics-vidhya/logistic-regression-b35d2801a29c

Sharma, N. (2020, January 31). *Understanding the mathematics behind support vector machines*. Medium. https://heartbeat.fritz.ai/understanding-the-mathematics-behind-support-vector-machines-5e20243d64d5

Shrestha, P. (2020, January 23). *The advantages and disadvantages of neural networks*. GK Stuffs. https://gkstuffs.com/future-tech/advantages-and-disadvantages-of-neural-networks/

Singh, A. (2020, May 25). *A practical introduction to k-nearest neighbors algorithm for regression (with Python code)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/

skikit learn. (n.d.). *Sklearn.linear_model.LogisticRegression — scikit-learn 0.23.2 documentation*. scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation. Retrieved April 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

skikit learn. (n.d.). *Sklearn.svm.SVC — scikit-learn 0.24.1 documentation*. scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation. Retrieved April 18, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Srivastava, T. (2020, April 1). *Introduction to k-nearest neighbors: A powerful machine learning algorithm (with implementation in Python & R)*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

Towards AI Team. (2020, November 17). *Why choose random forest and not decision trees*. Towards AI — The Best of Tech, Science, and Engineering. https://towardsai.net/p/machine-learning/why-choose-random-forest-and-not-decision-trees

Wikipedia. (2005, December 15). *Kernel method*. Wikipedia, the free encyclopedia. Retrieved April 18, 2021, from https://en.wikipedia.org/wiki/Kernel_method

Wikipedia. (2019, August 18). *Mathematics of artificial neural networks*. Wikipedia, the free encyclopedia. Retrieved April 18, 2021, from https://en.wikipedia.org/wiki/Mathematics_of_artificial_neural_networks

Wood, T. (2019, May 17). *Sigmoid function*. DeepAI. https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function

Zhang, G. (2018, November 11). *What is the kernel trick? Why is it important?* Medium. https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d