6-2022

Fine-Tuning a ⍰-Nearest Neighbors Machine Learning Model for the Detection of Insurance Fraud

Alliyah Stout
The University of Southern Mississippi

Fine-Tuning a k-Nearest Neighbors Machine Learning Model for the Detection of Insurance Fraud

by

Alliyah Stout

A Thesis
Submitted to the Honors College of
The University of Southern Mississippi
in Partial Fulfillment
of Honors Requirements

May 2022

Approved by:

Jacob Chapman, Ph.D., Thesis Advisor,
School of Mathematics and Natural Sciences

Bernd Schroeder, Ph.D., Director,
School of Mathematics and Natural Sciences

Sabine Heinhorst, Ph.D., Dean
Honors College

ABSTRACT

Billions of dollars are lost within insurance companies due to fraud. Large money losses force insurance companies to increase premium costs and/or restrict policies. This negatively affects a company's loyal customers. Although this is a prevalent problem, companies are not urgently working toward bettering their machine learning algorithms. Underskilled workers paired with inefficient computer algorithms make it difficult to accurately and reliably detect fraud.

The goal of this study is to understand the idea of k-Nearest Neighbors (k-NN) and to use this classification technique to accurately detect fraudulent auto insurance claims. Using k-NN requires choosing a k value and a distance metric. The best choice of k values and distance metrics will be unique to every dataset. This study aims to break down the processes involved in determining an accurate k value and distance metric for a sample auto insurance claims dataset. Odd k values 1 through 19 and the Euclidean, Manhattan, Chebyshev, and Hassanat metrics are analyzed using Excel and R.

Results support the idea that unique k values and distance metrics are needed depending on the dataset being worked with.

Keywords: machine learning, insurance, fraud, detection, k-NN, distance

DEDICATION

This thesis is dedicated to my mother, who has been by my side throughout the duration of my college career. Thank you for your constant support, patience, and faith in me.

ACKNOWLEDGMENTS

Firstly, I would like to thank my advisor, Dr. Jacob Chapman, for the energy and patience he has given to me throughout this process. The things he has taught me through completing this research project are invaluable, and I will be carrying them with me throughout the entirety of my future career.

Next, I want to thank the Honors College for introducing me to the world of conducting research and providing funding to aid in the completion of this research project.

Lastly, I want to thank my friends and family for always being a strong support system. Their constant encouragement has kept me driven and certainly does not go unnoticed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ABBREVIATIONS

k-NN	k-Nearest Neighbors
VBA	Visual Basic for Applications

CHAPTER I INTRODUCTION

Background

Due to fraud, billions of dollars are lost within the auto insurance field alone. According to Kalwihura and Logeswaran (2020), “Insurance fraud is the most practiced fraud in the world, and for the third consecutive time in six years, SAS Coalition (2019) reports that insurers have reported an increasing amount of suspected fraud.” These types of losses not only cause problems for the insurance companies, but their loyal customers as well. When companies struggle with large losses, these can sometimes cause them to increase the prices on certain insurance premiums. This puts an unfair strain on customers that are not committing any type of fraud. Insurance companies also combat this loss by changing the policy itself. Policies then contain fewer benefits and even harsher guidelines or restrictions. The penalties loyal customers and companies face are unjust. It is evident that there is a need to improve the current system being used to detect fraud. Increasing the accuracy of fraud detection algorithms can lower insurance prices, extend policies, and save insurance companies money.

It is important to note that many sources can manipulate insurance documents and procedures; these sources are categorized by Dhara and Anjani Kumar (2013) as internal and external sources. Internal fraud is committed by employees. This type of fraud can include the misrepresenting of insurance policies, tampering with documents, or mishandling funds. The external case is, of course, customers. These customers lie on documents and/or withhold information. This research is centered around training an algorithm using patterns to detect external fraud.

Problem Statement

The current process of detecting insurance fraud is time consuming and inefficient. Companies are placing little urgency on bettering the fraud detection process. Many claim handlers holding the responsibility to detect fraudulent activity "are often inexperienced, with typical company lifetimes of less than one year" (Morley et al., 2016). Using underskilled workers to take on a very tedious and challenging task results in inefficiency, squandered work hours, and overlooked information. These results, in addition to the underutilization of computer algorithms, limits the number of fraudsters detected. A refined version of technological programs could drastically improve the current process of detecting fraudulent activity.

It is no question that there is a critical need for improvement in the technological programs being used in the workplace today (Morley et al., 2016). Finding a way to improve fraud detection will aid in improving the viability of insurance companies. A better mathematical model could pinpoint and display suspicious customers who have the potential to be involved in insurance fraud. Insurance workers can then spend time investigating these few individuals, either confirming or denying their fraudulent claim. Using a mathematical model to specify the list of candidates involved in suspected fraudulent activity lowers the chances of the crime being overlooked, increases the speed and quality of the employee's work, and increases the chances of fraudulent activity being discovered.

CHAPTER II THE *k*-NN METHOD

Introduction

Machine learning is at the forefront of thought in data science today, and many companies are starting to follow suit by implementing various machine learning models to solve certain business problems and to ultimately increase profits and efficiency. The goal of machine learning is to train a computer on enough examples so that it can begin to predict an unknown variable for a new example. This unknown variable could be categorical or numerical. A model that seeks to predict a categorical variable is called a classification model (or classifier), and one that seeks to predict a numerical variable is called a regression model. In the study of insurance fraud, one is interested to know whether a claim is fraudulent, so this variable is categorical (Yes or No). Perhaps the simplest and most well-known classifier is the k-Nearest Neighbors (k-NN) algorithm, which was created by Fix and Hodges (1951) and later expanded upon by Cover and Hart (1967).

The idea is to divide a data set into two parts, a training set and a test set. In this research, 80 percent of the data set was devoted to training and 20 percent to testing. It is beneficial to use as much data as available to train the model, but enough data should be saved to test the model to establish confidence in the model's performance. The unknown categorical variable of interest is known for each training point and each testing point, so whatever the model predicts for a testing point can be compared to the actual value. The distance between each test point and each training point is calculated. A value k is determined, which represents the number of nearest neighbors to consider for each testing point. Whichever value for the categorical variable appears the most often among these k

nearest neighbors is the value that the model assigns to the testing point. What remains is to choose the value of k and which distance metric to use. Throughout this research, four distance metrics were considered. Three are arguably the most well known: the Euclidean, Manhattan, and Chebyshev metrics. The fourth metric considered was the Hassanat distance, introduced by Ahmad Hassanat (2014). This metric has met great success in many works, particularly in regard to the k-NN algorithm (Alkasassbeh et al. 2015; Prasath et al. 2019).

The Distance Metric

Euclidean Distance Metric

The Euclidean distance is one the most common distance metrics. It measures the shortest, straight-line distance between two points. The Euclidean distance between any two points $x, y \in \mathbb{R}^n$ is defined to be

$$ED(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

To gain a visual understanding of any given distance metric, one can observe its unit circle. A unit circle for a distance metric d is defined as the set of all points $x \in \mathbb{R}^n$ such that $d(x, 0) = 1$. In $\mathbb{R}^2$, the Euclidean unit circle simplifies to the familiar circle equation $x^2 + y^2 = 1$, whose graph can be observed in Figure 1.

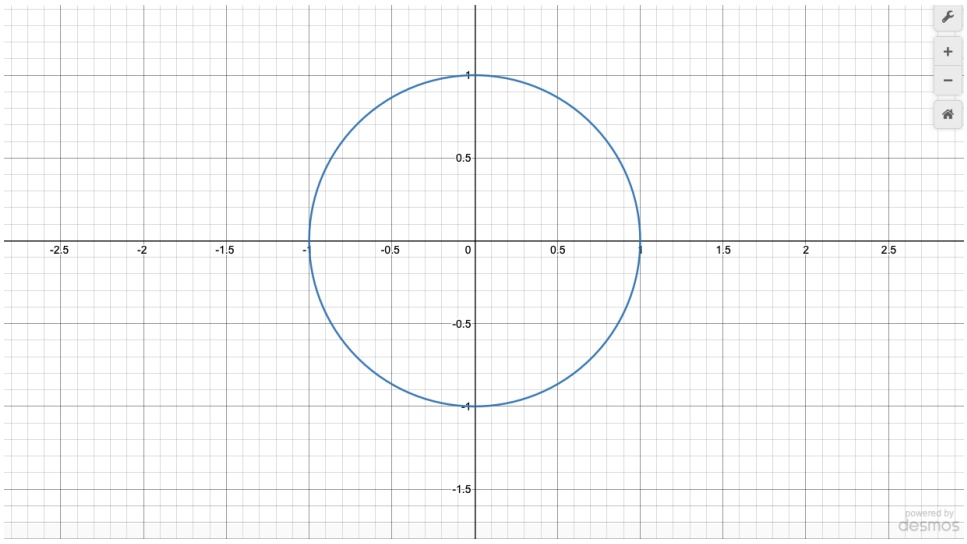


Figure 1:The Euclidean unit circle in $\mathbb{R}^2$

Manhattan Distance Metric

The Manhattan distance metric is also fairly common. This is also known as the "city block" measure, as its technique from getting from point to point is similar to the path a pedestrian would walk using city roads. The Manhattan distance between any two points $x, y \in \mathbb{R}^n$ is defined to be

$$MD(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

In $\mathbb{R}^2$, the Manhattan unit circle simplifies to the equation $|x| + |y| = 1$, whose graph can be observed in Figure 2.

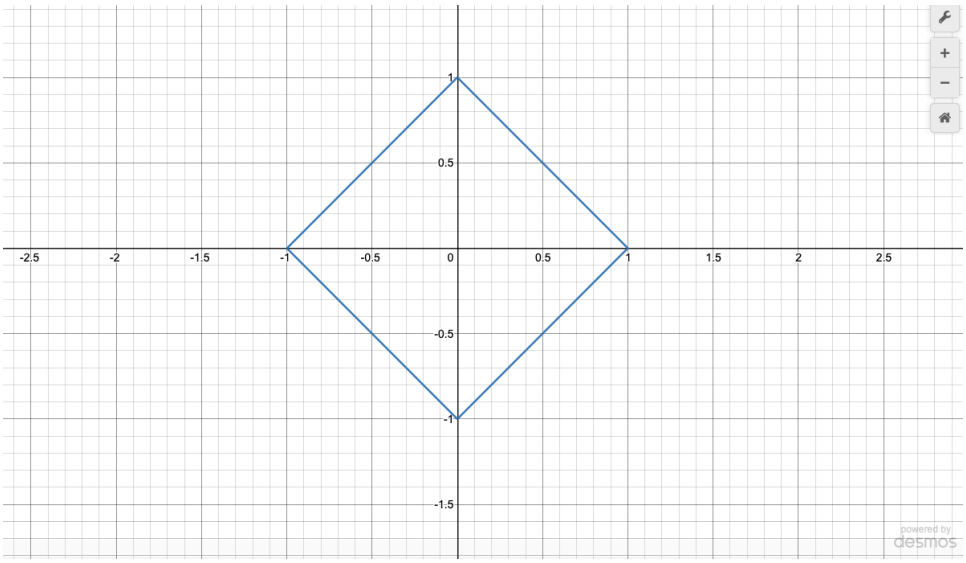


Figure 2: The Manhattan unit circle in $\mathbb{R}^2$

Chebyshev Distance Metric

The Chebyshev distance metric measures the distance between two points as the maximum absolute difference between their corresponding coordinates. Its path can be related to the minimum number of moves it would take a king in chess to move from one square to another. The Chebyshev distance between any two points $x, y \in \mathbb{R}^n$ is defined to be

$$CD(x, y) = \max_i |x_i - y_i|$$

In $\mathbb{R}^2$, the Chebyshev unit circle simplifies to the equation $\max(|x|, |y|) = 1$, whose graph can be observed in Figure 3.

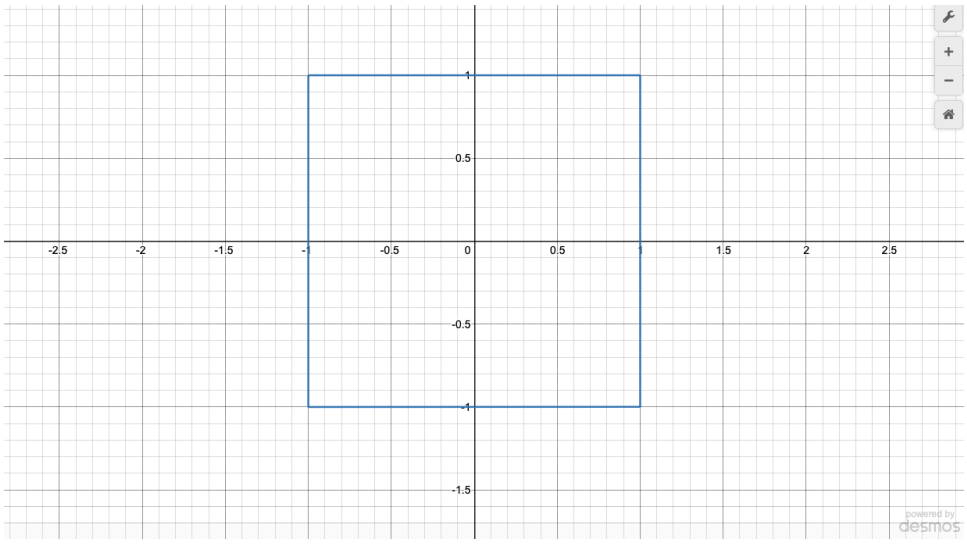


Figure 3: The Chebyshev unit circle in $\mathbb{R}^2$

Hassanat Distance Metric

The Hassanat distance metric is somewhat complex. Due to research done by Prasath et al. (2019) and Alkasassbeh et al. (2015) emphasizing its accuracy, this measure was tested. It ensures the distance measure is not altered by variations like "different scale, noise and outliers" (Alkasassbeh et al., 2015). The Hassanat distance between any two points$x, y \in \mathbb{R}^n$ is defined to be

$$HD(x,y) = \sum_{i=1}^{n} D(x_i, y_i)$$

where

$$D(x_i, y_i) = \begin{cases} 1 - \dfrac{1 + \min(x_i, y_i)}{1 + \max(x_i, y_i)}, & \text{if } \min(x_i, y_i) \geq 0 \\ 1 - \dfrac{1 + \min(x_i, y_i) + |\min(x_i, y_i)|}{1 + \max(x_i, y_i) + |\min(x_i, y_i)|}, & \text{if } \min(x_i, y_i) < 0 \end{cases}$$

Due to the complexity of the Hassanat distance metric, the unit circle takes the form of a hyperbola (or higher-dimensional analog), as seen in Figure 4. When one looks

at the set of all points a distance d from the origin, where $d \neq 1$, the Euclidean, Manhattan, and Chebyshev unit circles simply grow or shrink everywhere proportionally. However, for any $0 < d < 1$, the set of points having Hassanat distance d from the origin take a fundamentally different shape, that of a distorted diamond that is no longer unbounded as shown in Figure 5.

One can see that the Hassanat distance in $\mathbb{R}^n$ takes values in $[0, n]$. The Hassanat distance packs the infinite range $[0, \infty)$ of the Euclidean distance into the finite range $[0, n]$ in such a way that two vectors can be considered close even if one coordinate is vastly different. For instance, in the 2-D case, both coordinates must be somewhat close to each other for the Hassanat distance to be in $[0,1)$, whereas as soon as the Hassanat distance is 1, as we can see in the unit circle, one coordinate in the vectors can be widely different as long as the values for the other coordinates are close. This seems like a useful property to have when detecting fraud. Two individuals exhibiting similar characteristics but living in two very distant cities should not be thought to be too different from each other.

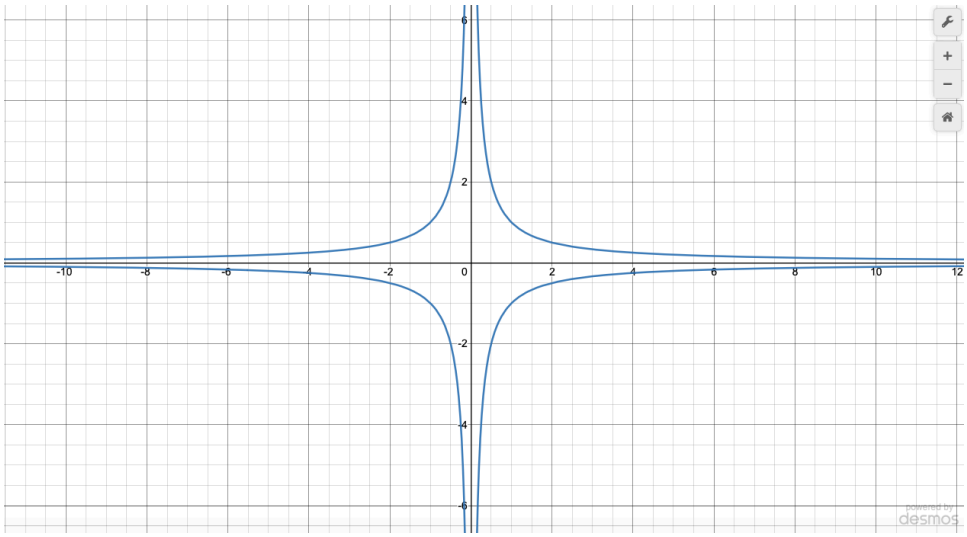


Figure 4: The Hassanat unit circle in $\mathbb{R}^2$

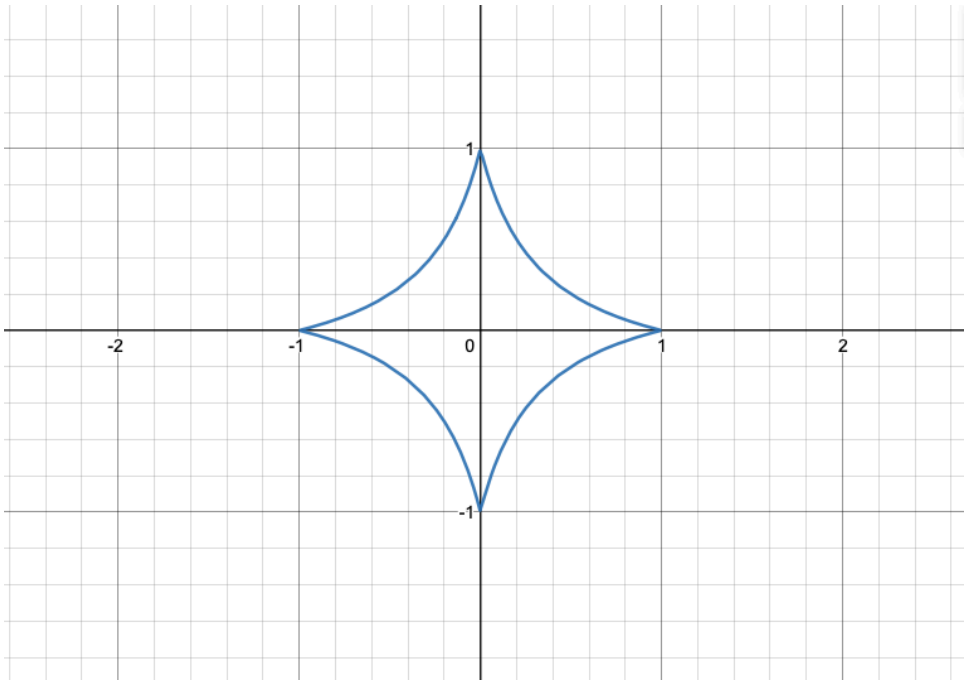


Figure 5: The set of all points in $\mathbb{R}^2$ *having Hassanat distance ½ from the origin*

Detailing *k*-NN Through an Example

Brett Lantz (2019) further explains the details of *k* -nearest neighbors using a food sorting example. For instance, an ingredient must be classified as either a fruit, vegetable, or protein. Each ingredient will be rated on a scale from 1 to 10 of how crunchy it is and

1 to 10 of how sweet it is. The k -NN algorithm then treats the feature ratings as coordinates. The algorithm can plot on a multidimensional space, but for simplicity, this example is plotted in two dimensions (using two features). How crunchy the food is is plotted on the y-axis and how sweet the food is is plotted on the x-axis. Similar types of food are grouped together. This allows for categories of fruits, vegetables, and proteins to be established. These groups can be used as a pattern to determine what other ingredients are.

The nearest neighbor approach, now programmed with preset patterns, will be used to determine whether a tomato is a fruit, vegetable, or protein. This step requires a distance metric. The Euclidean distance measure is most commonly used, but there are several other techniques to determine the distance between two points, all of which yield various levels of accurate results. A value of k must also be determined. Choosing too large a k value works to reduce the impact of noisy data and could potentially create an algorithm that ignores smaller, important pattern qualities. Too large of a k value will result in underfitting the data. On the other hand, choosing too small a k value allows outliers to have a major impact on the classification process, causing the machine learning program to give too much weight to individual data values, resulting in an overfitting of the data. Depending on the type of data being analyzed, the optimal k value will fluctuate.

Using $k = 3$, suppose that a tomato's three closest neighbors are an orange, grapes, and nuts. Two of three of the neighbors are a fruit, therefore the tomato is categorized as a fruit.

Confusion Matrix

"A confusion matrix is a table that categorizes predictions according to whether they match the actual value" (Lantz, 2019). The four possibilities include true positives, false positives, true negatives, and false negatives.

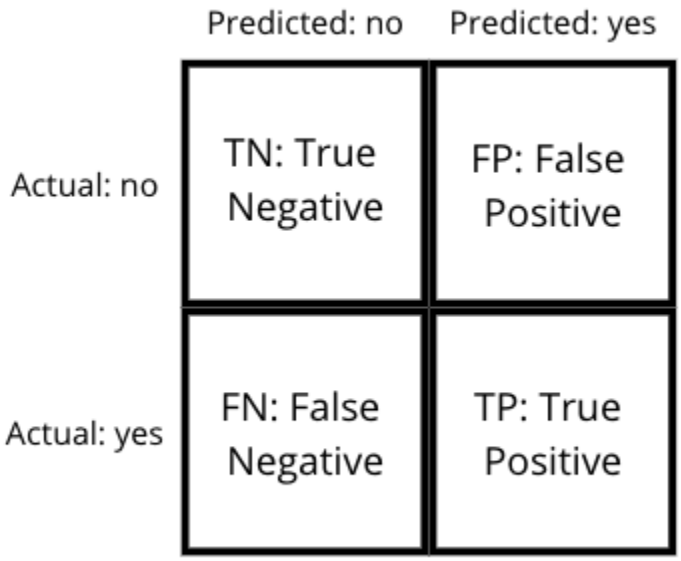


Figure 6: Sample 2x2 confusion matrix

It is important to understand that the titles of "positive" and "negative" do not always refer to "good" and "bad." They are simply titles that refer to two different outcomes. True positives and true negatives can be described as predicted outcomes that correctly matched the results. A false positive is produced when the predicted value is positive, but the actual value is negative. Similarly, a false negative is produced when the predicted value is negative, but the actual value is positive.

A confusion matrix is used to determine how well a model performed. Components such as accuracy, error rate, sensitivity, specificity, precision, and recall can

all be calculated using the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

Accuracy measures the "success rate". It takes the total number of true positives and true negatives and divides them by the total number of predictions.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Error rate is essentially the opposite of accuracy. It measures the number of incorrect predictions divided by the total number of predictions.

$$\text{error rate} = \frac{FP + FN}{TP + TN + FP + FN} = 1 - \text{accuracy}$$

Sensitivity displays a "true positive rate." This measure takes the number of true positives and divides them by the sum of true positives and false negatives.

$$\text{sensitivity} = \frac{TP}{TP + FN}$$

Specificity displays a "true negative rate." Like sensitivity, specificity measures the number of true negatives and divides them by the sum of true negatives and false positives.

$$\text{specificity} = \frac{TN}{TN + FP}$$

Precision displays the number of positive examples that are truly positive. This is a very good measure of how trustworthy a model is.

$$\text{precision} = \frac{TP}{TP + FP}$$

Recall has the same formula as sensitivity but can be used to draw different conclusions. Higher recall means that the model is capturing a large portion of positive examples.

$$\text{recall} = \frac{TP}{TP + FN}$$

Overview of Analysis

The k-NN algorithm is simple, yet it is a versatile and widely-used classifier (Lantz, 2019). This research is centered around using Excel and R to find the optimal k value and distance metric that yields the best performance for the insurance claims data set.

Data being used has been obtained from https://www.kaggle.com/code/buntyshah/insurance-fraud-claims-detection/notebook. It can be noted that "insurance data are generally proprietary information of the insurance companies and are not publicly available" (Kalwihura & Logeswaran, 2020). Due to this difficulty in obtaining such private information, it cannot be confirmed that the dataset used within this research represents real data. However, if the data in this dataset is synthetic it is still suitable for use in developing methods and tools which can then be applied to real data sets. Furthermore, the question can be posed: "If given a set of insurance fraud data, how do I select the optimal k value and distance metric to train a k -NN model for accurately predicting fraud?" Processes outlined aim to demonstrate the idea behind k -NN, investigate the structure of machine learning algorithms, and analyze how to determine appropriate k values and distance metrics.

One of the weaknesses that accompanies the k -NN algorithm is the uncertainty when choosing a value for k . Within this research, odd k values 1 through 19 are implemented and observed. Even values of k are avoided due to ties in the majority vote of the nearest neighbors. Several values are tested due to the fact that every dataset

requires a certain value of k to yield the most accurate results. Qualities such as the type and number of variables all contribute to the effectiveness of chosen k values. Creating an accurate model greatly depends on testing data using several k values and analyzing each of their performances.

The distance metrics being used in the analysis are Euclidean, Manhattan, Chebyshev, and Hassanat. Each will be observed on its performance by taking its accuracy, error rate, sensitivity, specificity, precision, and recall. These items are calculated by using the results of a confusion matrix.

Although all values calculated from the confusion matrix are important, some need to be analyzed with this particular dataset in mind. Precision and recall are both very important to look at in this case. Higher precision is desirable when the costs of false positives are high. A higher recall is desirable when the costs of false negatives are high.

Within this dataset, false positives indicate that the algorithm has categorized a claim as fraudulent, but the claim is honest. False negatives indicate that the algorithm has categorized a claim as honest, but the claim is fraudulent. When thinking about false positives and false negatives in this instance, one can conclude that the cost of false positives would be higher than false negatives. Although false negatives result in customers getting away with fraudulent activity, if a model determined that a certain customer's insurance claim was positive for fraudulent activity, when in reality it is not, this can cause friction between an insurance firm and its customers. An insurance company confronting a loyal customer could result in the customer having to go through strenuous processes to prove their authenticity. An insurance company does not want the

reputation of falsely accusing its valued and honest customers. Therefore, the level of precision should be held at a higher consideration than recall.

CHAPTER III PROCESS

Process for Excel

The raw dataset detailing the information regarding insurance claims is not immediately usable. It must be determined which variables can be disregarded, which need to be one-hot encoded (see definition below), and which need to be normalized.

One-Hot Encoding

Because a distance function cannot operate with nominal values, variables must be converted to a numerical format. A process for doing this is called one-hot encoding. The variable is broken into categories. A 1 indicates that the categorical variable takes some specific value and a 0 indicates otherwise. This creates a format that consists of all 1s and 0s, which conveniently scales in coherence with normalized data. For data with only two categories, items will be constructed similarly to the following example (male and female):

$$\text{male} = \begin{cases} 1 & \text{if } x = \text{male} \\ 0 & \text{otherwise} \end{cases}$$

For cases in which there are more than two categories, items will be constructed similarly to the following example (cold, medium, hot):

$$\text{hot} = \begin{cases} 1 & \text{if } x = \text{hot} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{medium} = \begin{cases} 1 & \text{if } x = \text{medium} \\ 0 & \text{otherwise} \end{cases}$$

Normalizing

Normalizing is a technique used to ensure all variables are on the same scale. The formula takes the difference between the variable, X, and the minimum of X and divides this value by the range of X.

$$X_{\text{new}} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

This produces values that range from 0 to 1. Normalizing is necessary because, for example, if one variable ranges from 0 to 1 while another variable ranges from 0 to 1000, the distance between the two vectors will be completely dominated by the variable with the larger range. It is necessary that all variables contribute equally to the computation. Once one-hot encoding and normalizing are complete, the data is ready to be used for calculations.

Formula Descriptions and Explanations

The Excel workbook consists of four sheets: "Data," "Sheet 1," "Sheet 2," and "Sheet 3." "Sheet 1" of the workbook contains the original dataset along with two additional columns, AO and AP, where customers' vehicle make and model, and city and state were concatenated, respectively. This was done using the & command. It was decided to combine these variables because, for example, two city names could be the same but could be in different states.

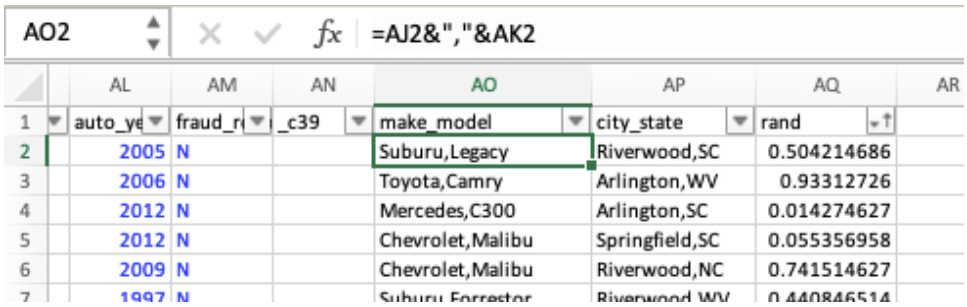
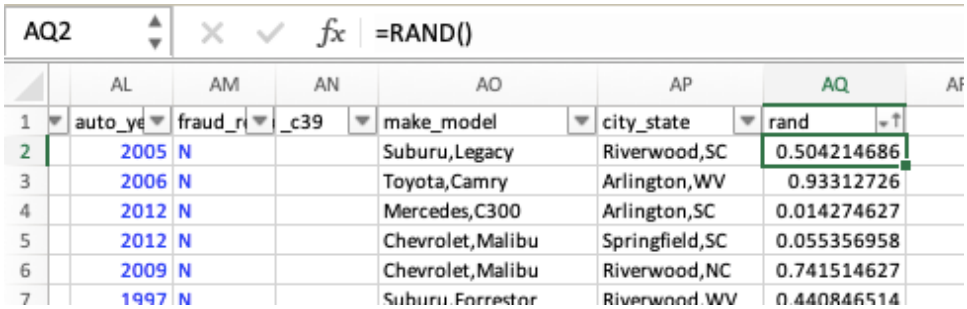


Figure 7: Excel formula created to combine "make" with "model" and "city" with "state"

"Data" also contains a third added column, AQ. Since the dataset came from a source with little background information, it cannot be concluded that the data was not listed in a specific order. Having the data in an ordered fashion would cause inaccurate results due to the fashion in which test and training rows are assigned. This column, AQ, assigns each row a random number between 0 and 1 using the RAND() command. The dataset was sorted in ascending order according to the values in AQ, ensuring rows were randomized.

"Sheet 2" of the workbook contains all variables after deletion, normalizing, one hot-encoding, and labeling.



	AL	AM	AN	AO	AP	AQ
1	auto_ye	fraud_r	_c39	make_model	city_state	rand
2	2005	N		Suburu,Legacy	Riverwood,SC	0.504214686
3	2006	N		Toyota,Camry	Arlington,WV	0.93312726
4	2012	N		Mercedes,C300	Arlington,SC	0.014274627
5	2012	N		Chevrolet,Malibu	Springfield,SC	0.055356958
6	2009	N		Chevrolet,Malibu	Riverwood,NC	0.741514627
7	1997	N		Suburu,Forrestor	Riverwood,WV	0.440846514

Figure 8: Excel formula used to randomize rows

Variables that have been categorized as carrying irrelevant information are policy number, insured's ZIP code, and incident location. Policy number was removed due to the irrelevant nature of the numbers. Policy numbers are a string of random numbers that indicate specific cases. Ordering these through normalizing would not yield any relevant information. Insured's ZIP code was removed because the customer's city and state are already recorded. Similarly, incident location (street address) was removed. If street address and ZIP code were used in conjunction with the city and state, each customer

would have their own category when the one-hot encoding was carried out. This again would not yield relevant information.

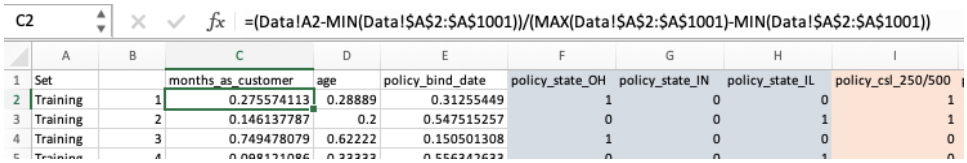
Variables that have been normalized (no color, “N”) and one-hot encoded (colored, “OHE”) can be found in Figure 9. It is important to note varying colors were used for organizational purposes, and do not indicate different alterations to variables.

months_as_customer	N
age	N
policy_bind_date	N
policy_state	OHE
policy_csl	OHE
policy_deductable	N
policy_annual_premium	N
umbrella_limit	N
insured_sex	OHE
insured_education_level	OHE
insured_occupation	OHE
insured_hobbies	OHE
insured_relationship	OHE
capital_gains	N
capital_loss	N
incident_date	N
incident_type	OHE
collision_type	OHE
incident_severity	OHE
authorities_contacted	OHE
incident_city_state	OHE
incident_hour_of_the_day	N
number_of_vehicles_involved	N
property_damage	OHE
bodily_injuries	N
witnesses	N
police_report_avaliable	OHE
total_claim	N
injury_claim	N
property_claim	N
vehicle_claim	N
auto_make_model	OHE
auto_year	N
fraud_report	

Figure 9: Organized list depicting which variables were normalized (no color,”N”) and one-hot encoded (colored, “OHE”)

Normalizing uses MAX() and MIN() functions provided by Excel. Cell C2 references information (A2) from “Data” and subtracts it from the MIN of the entire A column, again from "Data”. This numerical value is then divided by the difference

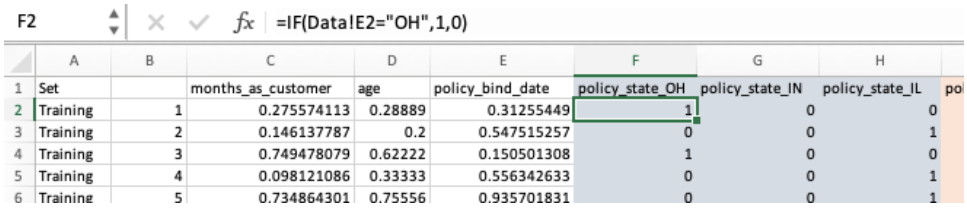
between the MAX of column A and the MIN of column A from “Data”. A similar process is done for each of the variables that need to be normalized.



	A	B	C	D	E	F	G	H	I
1	Set		months_as_customer	age	policy_bind_date	policy_state_OH	policy_state_IN	policy_state_IL	policy_csl_250/500
2	Training	1	0.275574113	0.28889	0.31255449	1	0	0	1
3	Training	2	0.146137787	0.2	0.547515257	0	0	1	1
4	Training	3	0.749478079	0.62222	0.150501308	1	0	0	0

Figure 10: Excel formula created to mimic the normalizing formula

One-hot encoding uses the IF() function provided by Excel. This formula takes the cell’s value (E2) from “Data” and assigns it a 1 if it matches the indicated value in parenthesis and a 0 if it does not.



	A	B	C	D	E	F	G	H
1	Set		months_as_customer	age	policy_bind_date	policy_state_OH	policy_state_IN	policy_state_IL
2	Training	1	0.275574113	0.28889	0.31255449	1	0	0
3	Training	2	0.146137787	0.2	0.547515257	0	0	1
4	Training	3	0.749478079	0.62222	0.150501308	1	0	0
5	Training	4	0.098121086	0.33333	0.556342633	0	0	1
6	Training	5	0.734864301	0.75556	0.935701831	0	0	1

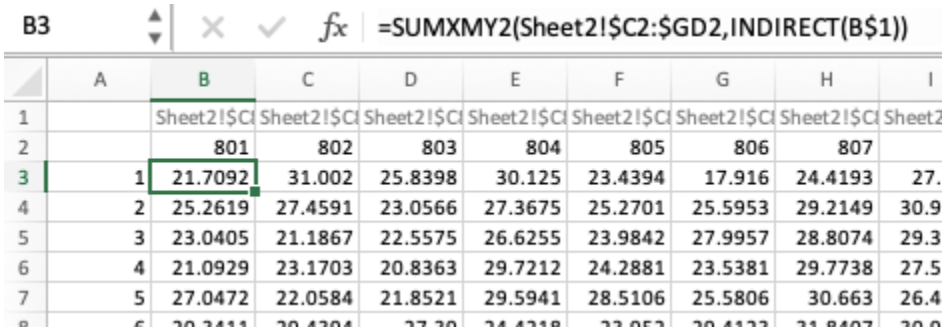
Figure 11: Excel formula used to one-hot encode certain variables

The next step in the process is to label rows as either “test” or “training” rows. It is recommended that 80% of data be used in the training set, while 20% function as the test set (Lantz, 2019). There are 1000 rows in this dataset. After re-sorting the data in ascending order based on the random number assigned each row, the first 800 rows are labeled as “training rows," and the last 200 are labeled as “test rows.” This is simply done by adding two blank columns before the data. Column B is used to number the rows and assign column A with “Training” for the first 800, and “Test” for the last 200. Rows being randomized is important for this step. If rows were in order, for example, by policy bind date (oldest to newest), the normalized information would range from 0 to 1, in that

order. The highest values would be located within the test rows and would reduce the predictive power of the model because of bias.

"Sheet 3" of the workbook contains the final steps in preparing the data so that it can be implemented into a cross table and conclusions can be drawn.

The distance between each "training" and "test" row is calculated using the Euclidean, Manhattan, Chebyshev, and Hassanat distance metrics. Visual Basic for Applications (VBA) is a developer tool used within Excel to create new functions. Euclidean, Manhattan, and Hassanat commands were created and can be found within Appendix A. An example of the distance metric process can be seen in Figure 12. SUMXMY2 is used in this step because is it already stored in Excel and is identical to the Euclidean formula; the VBA code "EUCLID" can be found in Appendix A and used interchangeably. The SUMXMY2 command is used to find the squared distance between rows 1-800 (column A) of training rows and rows 801-1000 (row 2) of test rows. A formula was derived to reference from "Sheet2", using the first training and first test row and calculating the distance between the two. Each cell works in this manner to create an 800x200 grid calculating the distance between all training and test rows.



	A	B	C	D	E	F	G	H	I
1		Sheet2!$C	Sheet2!$C	Sheet2!$C	Sheet2!$C	Sheet2!$C	Sheet2!$C	Sheet2!$C	Sheet2
2		801	802	803	804	805	806	807	
3	1	21.7092	31.002	25.8398	30.125	23.4394	17.916	24.4193	27.
4	2	25.2619	27.4591	23.0566	27.3675	25.2701	25.5953	29.2149	30.9
5	3	23.0405	21.1867	22.5575	26.6255	23.9842	27.9957	28.8074	29.3
6	4	21.0929	23.1703	20.8363	29.7212	24.2881	23.5381	29.7738	27.5
7	5	27.0472	22.0584	21.8521	29.5941	28.5106	25.5806	30.663	26.4

Figure 12: Excel formula used to obtain the Euclidean distance between "training row" 1 and "test row" 1

Excel's SMALL() function is then used to locate the 1-19 smallest distances from the Test column. For this cell, the SMALL function highlights all of column B in blue and displays the 1st (A805 in red) smallest distance. All cells to the right of B805 will do the same, but for columns B, C, D, E, etc. All cells below B805 will calculate the 2nd, 3rd, 4th, etc. smallest distance. This forms a 19x200 grid displaying each column's 1-19th smallest distance.

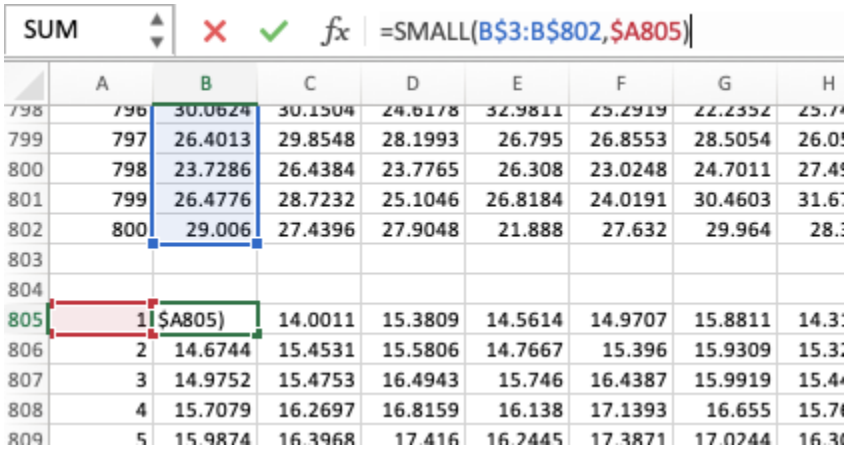


Figure 13: Excel formula used to located the k smallest distance

A combination of INDEX() and MATCH() functions determines which row is associated with each distance within the grid. Figure 14 shows the formula created, while Figure 15 gives a more detailed visual of what is being done. The INDEX() function locates the row number in the array (column A in blue) where cell B805 (red) exactly matches a value in the second array (column B in purple).

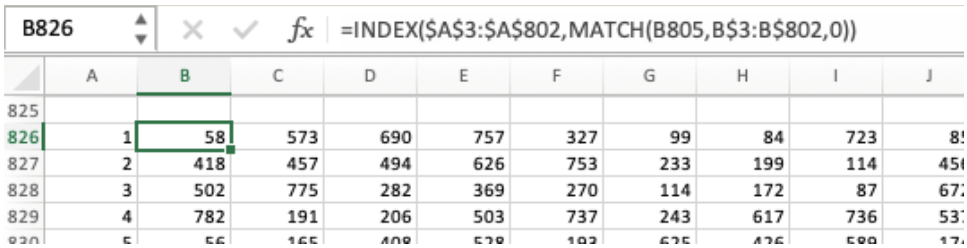


Figure 14: Excel formula created to find the row associated with corresponding distance

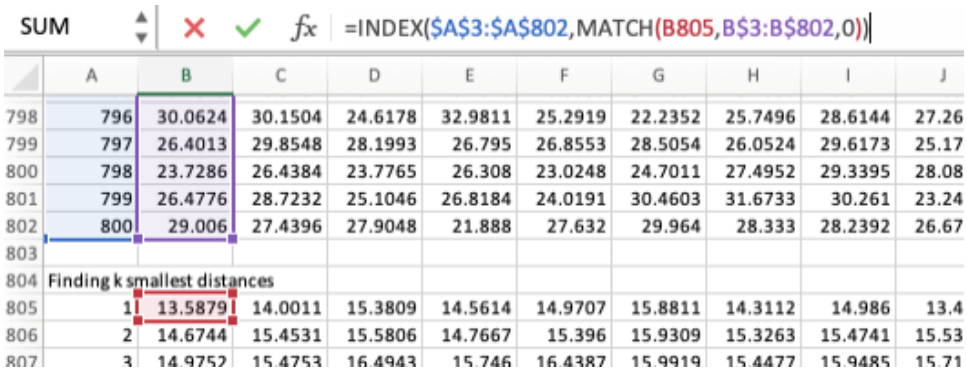


Figure 15: Color-coded Excel formula associated with Figure 14

Next, it is then determined whether these rows are associated with fraudulent activity or not. Another combination of INDEX() and MATCH() functions is used similarly to the previous step. The INDEX() function locates the value (Y/N) in the array from "Sheet2" (column GE) where cell B826 exactly matches the value from "Sheet2" (column B).

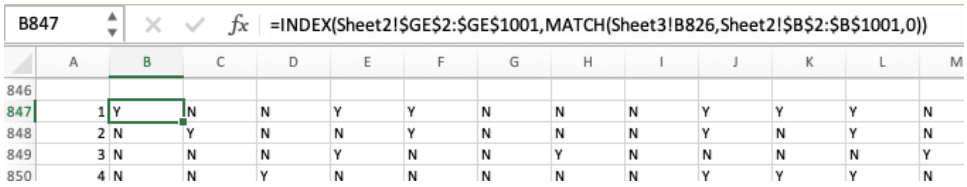


Figure 16: Excel formula created to find Y/N value associated with corresponding row

The majority winner (for varying k values) within each row can then be determined. Only an odd number of rows is considered to eliminate the possibility of a tie. IF() and COUNTIFS() functions are used to create this function. The formula places a "N" in cell B868 if more than half of the array (blue) is N's and a Y if not, essentially displaying a majority winner. This produces the predicted values.

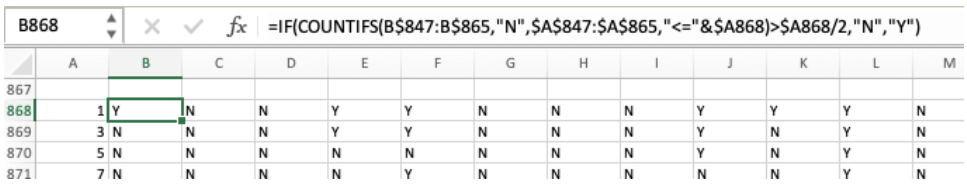


Figure 17: Excel formula created to determine a majority winner

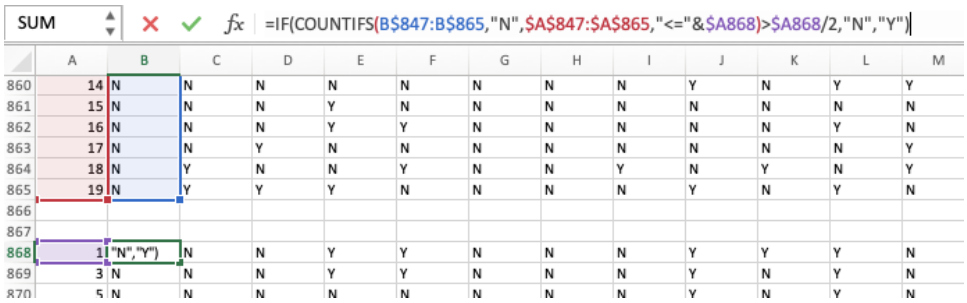


Figure 18: Color-coded Excel formula associated with Figure 17

A row with the actual values of the test rows are transposed into row 880. These values will be used to compare to the predicted values.

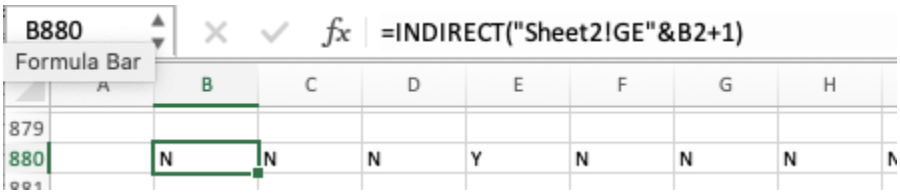
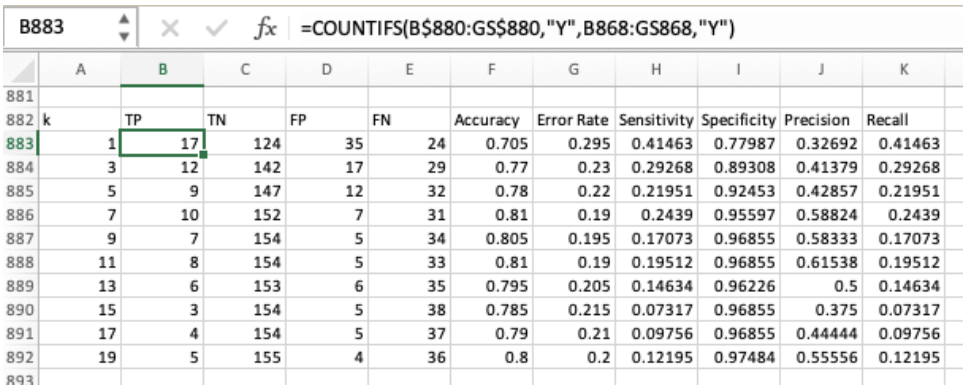


Figure 19: Excel formula used to transpose actual values

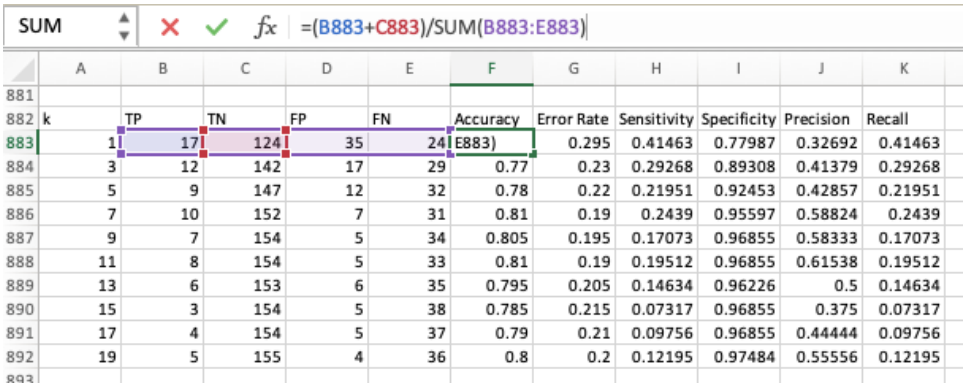
Row 880 is compared to each row (868-877) using the COUNTIFS() function. Figure 20 shows the formula for cell B883. Row 880 values are compared to those in row 883. If both values are “Y”, it is counted. There are 17 cells between row 880 and 883 in which values match (both “Y”). The formula is manipulated dependent on if TN, TP, FP, or FN needs to be calculated. For instance, FP would require row 880 (actual value) to be “N” and 883 (predicted value) to be “Y”. These values produce a confusion matrix for each value of k.



	A	B	C	D	E	F	G	H	I	J	K
881											
882	k	TP	TN	FP	FN	Accuracy	Error Rate	Sensitivity	Specificity	Precision	Recall
883	1	17	124	35	24	0.705	0.295	0.41463	0.77987	0.32692	0.41463
884	3	12	142	17	29	0.77	0.23	0.29268	0.89308	0.41379	0.29268
885	5	9	147	12	32	0.78	0.22	0.21951	0.92453	0.42857	0.21951
886	7	10	152	7	31	0.81	0.19	0.2439	0.95597	0.58824	0.2439
887	9	7	154	5	34	0.805	0.195	0.17073	0.96855	0.58333	0.17073
888	11	8	154	5	33	0.81	0.19	0.19512	0.96855	0.61538	0.19512
889	13	6	153	6	35	0.795	0.205	0.14634	0.96226	0.5	0.14634
890	15	3	154	5	38	0.785	0.215	0.07317	0.96855	0.375	0.07317
891	17	4	154	5	37	0.79	0.21	0.09756	0.96855	0.44444	0.09756
892	19	5	155	4	36	0.8	0.2	0.12195	0.97484	0.55556	0.12195
893											

Figure 20: Excel formula used to determine the amount of true postives, false positives, true negatives, and false negatives

With a confusion matrix now produced, it can be determined how well the model performed. Accuracy, error rate, sensitivity, specificity, precision, and recall are all calculated using basic formulas. Basic formulas are translated into Excel by referencing appropriate cells. This information will allow conclusions to be drawn and enables one to choose an optimal value of k.



	A	B	C	D	E	F	G	H	I	J	K
881											
882	k	TP	TN	FP	FN	Accuracy	Error Rate	Sensitivity	Specificity	Precision	Recall
883	1	17	124	35	24	E883)	0.295	0.41463	0.77987	0.32692	0.41463
884	3	12	142	17	29	0.77	0.23	0.29268	0.89308	0.41379	0.29268
885	5	9	147	12	32	0.78	0.22	0.21951	0.92453	0.42857	0.21951
886	7	10	152	7	31	0.81	0.19	0.2439	0.95597	0.58824	0.2439
887	9	7	154	5	34	0.805	0.195	0.17073	0.96855	0.58333	0.17073
888	11	8	154	5	33	0.81	0.19	0.19512	0.96855	0.61538	0.19512
889	13	6	153	6	35	0.795	0.205	0.14634	0.96226	0.5	0.14634
890	15	3	154	5	38	0.785	0.215	0.07317	0.96855	0.375	0.07317
891	17	4	154	5	37	0.79	0.21	0.09756	0.96855	0.44444	0.09756
892	19	5	155	4	36	0.8	0.2	0.12195	0.97484	0.55556	0.12195
893											

Figure 21: Excel formula used to mimic formulas for accuracy, error rate, sensitivity, specificity, precision, and recall

Process for R

The process for producing a confusion matrix for the dataset in R mimics the process through Excel. The same steps are simply modified according to the functions allowed in R. The data is saved as a .csv file for easy upload. The ratio of Y/N variables

in the fraud reported column is rewritten as a decimal. Incident city/state and auto make/model are combined using the paste( ) command.

The same irrelevant columns are then removed using the –c( ) command. In R, the columns that display dates must be reformatted. This is done through the julian( ) and as.Date( ) command.

Next, the appropriate columns must be normalized and one-hot encoded. Normalizing is done through creating a 'normalize' function using the formula and using it on the appropriate columns. One-hot encoding is done using the dummyVars( ) command.

Lastly, training rows are assigned to be rows 1:800 and test rows are assigned to be rows 801:1000. A knn( ) command allows a certain k to be used in conjunction to the Euclidean metric. After running this command, a confusion matrix is produced using the CrossTable( ) command. The cross-table values then allow for computation of accuracy, error rate, sensitivity, specificity, precision, and recall. The script created following this process can be found within Appendix B.

CHAPTER IV RESULTS

Varying *k* value

Results for confusion matrix using varying *k* values in Excel can be found in Figure 22. Results calculated in R can be found in Tables 1-10. Each confusion matrix was composed of *k* values 1-19 (only odd).

Results for Excel

k	TP	TN	FP	FN	Accuracy	Error Rate	Sensitivity	Specificity	Precision	Recall
1	17	124	35	24	0.705	0.295	0.41463	0.77987	0.32692	0.41463
3	12	142	17	29	0.77	0.23	0.29268	0.89308	0.41379	0.29268
5	9	147	12	32	0.78	0.22	0.21951	0.92453	0.42857	0.21951
7	10	152	7	31	0.81	0.19	0.2439	0.95597	0.58824	0.2439
9	7	154	5	34	0.805	0.195	0.17073	0.96855	0.58333	0.17073
11	8	154	5	33	0.81	0.19	0.19512	0.96855	0.61538	0.19512
13	6	153	6	35	0.795	0.205	0.14634	0.96226	0.5	0.14634
15	3	154	5	38	0.785	0.215	0.07317	0.96855	0.375	0.07317
17	4	154	5	37	0.79	0.21	0.09756	0.96855	0.44444	0.09756
19	5	155	4	36	0.8	0.2	0.12195	0.97484	0.55556	0.12195

Figure 22: All results calculated in Excel for k values 1-19 (odd)

Results for R

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	124	35	159
	0.780	0.220	0.795
	0.838	0.673	
	0.620	0.175	
Y	24	17	41
	0.585	0.415	0.205
	0.162	0.327	
	0.120	0.085	
Column Total	148	52	200
	0.740	0.260	

Table 1: Confusion matrix calculated for $k = 1$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	142	17	159
	0.893	0.107	0.795
	0.830	0.586	
	0.710	0.085	
Y	29	12	41
	0.707	0.293	0.205
	0.170	0.414	
	0.145	0.060	
Column Total	171	29	200
	0.855	0.145	

Table 2: Confusion matrix calculated for $k = 3$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	147	12	159
	0.925	0.075	0.795
	0.821	0.571	
	0.735	0.060	
Y	32	9	41
	0.780	0.220	0.205
	0.179	0.429	
	0.160	0.045	
Column Total	179	21	200
	0.895	0.105	

Table 3: Confusion matrix calculated for $k = 5$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	152	7	159
	0.956	0.044	0.795
	0.831	0.412	
	0.760	0.035	
Y	31	10	41
	0.756	0.244	0.205
	0.169	0.588	
	0.155	0.050	
Column Total	183	17	200
	0.915	0.085	

Table 4: Confusion matrix calculated for $k = 7$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	154	5	159
	0.969	0.031	0.795
	0.819	0.417	
	0.770	0.025	
Y	34	7	41
	0.829	0.171	0.205
	0.181	0.583	
	0.170	0.035	
Column Total	188	12	200
	0.940	0.060	

Table 5: Confusion matrix calculated for $k = 9$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	154	5	159
	0.969	0.031	0.795
	0.824	0.385	
	0.770	0.025	
Y	33	8	41
	0.805	0.195	0.205
	0.176	0.615	
	0.165	0.040	
Column Total	187	13	200
	0.935	0.065	

Table 6: Confusion matrix calculated for $k = 11$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	153	6	159
	0.962	0.038	0.795
	0.814	0.500	
	0.765	0.030	
Y	35	6	41
	0.854	0.146	0.205
	0.186	0.500	
	0.175	0.030	
Column Total	188	12	200
	0.940	0.060	

Table 7: Confusion matrix calculated for $k = 13$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	154	5	159
	0.969	0.031	0.795
	0.802	0.625	
	0.770	0.025	
Y	38	3	41
	0.927	0.073	0.205
	0.198	0.375	
	0.190	0.015	
Column Total	192	8	200
	0.960	0.040	

Table 8: Confusion matrix calculated for $k = 15$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	154	5	159
	0.969	0.031	0.795
	0.806	0.556	
	0.770	0.025	
Y	37	4	41
	0.902	0.098	0.205
	0.194	0.444	
	0.185	0.020	
Column Total	191	9	200
	0.955	0.045	

Table 9: Confusion matrix calculated for $k = 17$

	claims_test_pred		
claims_test_labels	N	Y	Row Total
N	155	4	159
	0.975	0.025	0.795
	0.812	0.444	
	0.775	0.020	
Y	36	5	41
	0.878	0.122	0.205
	0.188	0.556	
	0.180	0.025	
Column Total	191	9	200
	0.955	0.045	

Table 10: Confusion matrix calculated for $k = 19$

All results for accuracy error rate, sensitivity, specificity, precision, and recall from R exactly match Excel results (Figure 22).

Varying Metric

Results for confusion matrix using varying distance metric in Excel can be found in Figures 23-25. Excel results contain all odd k values 1-19; however, results for only $k = 11$ are used in this portion of the analysis. Results calculated in R can be found in Figure 26. Each confusion matrix was composed using Euclidean, Manhattan, and Hassanat distance metrics using $k = 11$.

Results for Excel

k	TP	TN	FP	FN	Accuracy	Error Rate	Sensitivity	Specificity	Precision	Recall
1	17	124	35	24	0.705	0.295	0.41463	0.77987	0.32692	0.41463
3	12	142	17	29	0.77	0.23	0.29268	0.89308	0.41379	0.29268
5	9	147	12	32	0.78	0.22	0.21951	0.92453	0.42857	0.21951
7	10	152	7	31	0.81	0.19	0.2439	0.95597	0.58824	0.2439
9	7	154	5	34	0.805	0.195	0.17073	0.96855	0.58333	0.17073
11	8	154	5	33	0.81	0.19	0.19512	0.96855	0.61538	0.19512
13	6	153	6	35	0.795	0.205	0.14634	0.96226	0.5	0.14634
15	3	154	5	38	0.785	0.215	0.07317	0.96855	0.375	0.07317
17	4	154	5	37	0.79	0.21	0.09756	0.96855	0.44444	0.09756
19	5	155	4	36	0.8	0.2	0.12195	0.97484	0.55556	0.12195

Figure 23: All results calculated in Excel using the Euclidean metric

k	TP	TN	FP	FN	Accuracy	Error Rate	Sensitivity	Specificity	Precision	Recall
1	14	123	36	27	0.685	0.315	0.34146	0.77358	0.28	0.34146
3	12	141	18	29	0.765	0.235	0.29268	0.88679	0.4	0.29268
5	8	146	13	33	0.77	0.23	0.19512	0.91824	0.38095	0.19512
7	7	152	7	34	0.795	0.205	0.17073	0.95597	0.5	0.17073
9	9	153	6	32	0.81	0.19	0.21951	0.96226	0.6	0.21951
11	7	153	6	34	0.8	0.2	0.17073	0.96226	0.53846	0.17073
13	7	153	6	34	0.8	0.2	0.17073	0.96226	0.53846	0.17073
15	4	154	5	37	0.79	0.21	0.09756	0.96855	0.44444	0.09756
17	5	153	6	36	0.79	0.21	0.12195	0.96226	0.45455	0.12195
19	5	155	4	36	0.8	0.2	0.12195	0.97484	0.55556	0.12195

Figure 24: All results calculated in Excel using the Manhattan metric

k	TP	TN	FP	FN	Accuracy	Error Rate	Sensitivity	Specificity	Precision	Recall
1	13	123	36	28	0.68	0.32	0.31707	0.77358	0.26531	0.31707
3	12	143	16	29	0.775	0.225	0.29268	0.89937	0.42857	0.29268
5	9	145	14	32	0.77	0.23	0.21951	0.91195	0.3913	0.21951
7	9	150	9	32	0.795	0.205	0.21951	0.9434	0.5	0.21951
9	8	155	4	33	0.815	0.185	0.19512	0.97484	0.66667	0.19512
11	6	154	5	35	0.8	0.2	0.14634	0.96855	0.54545	0.14634
13	6	153	6	35	0.795	0.205	0.14634	0.96226	0.5	0.14634
15	5	153	6	36	0.79	0.21	0.12195	0.96226	0.45455	0.12195
17	4	153	6	37	0.785	0.215	0.09756	0.96226	0.4	0.09756
19	5	154	5	36	0.795	0.205	0.12195	0.96855	0.5	0.12195

Figure 25: All results calculated in Excel using the Hassanat metric

Results for R

	Accuracy	Error Rate	Sensitivity	Specificity	Precision	Recall
EUCLID	0.81	0.19	0.19512	0.96855	0.61538	0.19512
MANHAT	0.8	0.2	0.17073	0.96226	0.53846	0.17073
HASS	0.8	0.2	0.14634	0.96855	0.54545	0.14634

Figure 26: All results calculated in R using the Euclidean, Manhattan, and Hassanat distance metric and $k = 11$

Due to one-hot encoding certain variables, all results using the Chebyshev distance metric concluded to one. This is because the Chebyshev metric displays the max difference between any variables within the training and test rows. Since several of the variables' values were 0 or 1 (after being one-hot encoded), the max difference was always one. Since this is the case, the Chebyshev distance metric does not yield any relevant results.

CHAPTER V CONCLUSIONS

Varying *k* Value

Accuracy, precision, and recall are most closely analyzed. Error rate is simply the residual value from accuracy, and therefore redundant to analyze. Sensitivity produces the same values as recall. Sensitivity and recall differ only slightly through interpretation. Recall represents the "breadth" of results, while sensitivity is strictly a proportional value. Therefore, of these two measures, only recall will be considered. Specificity calculates the proportion of negative examples correctly classified. Since customers not committing insurance fraud are not of particular interest, this value is also of less relevance.

When studying the accuracy results in Figure 22, one can see that the accuracy peaks between $k = 7$ and $k = 11$. One observes very high accuracy toward the higher k values as well. This finding can be disregarded, as it is now overfitting the data. Precision also peaks around the same k values. Recall, however, peaks with lower k values. Recall is peaking between $k = 3$ and $k = 7$. $k = 1$ can be disregarded as it is likely overfitting the data.

As the cost of false positives is high, precision has higher importance. Higher precision and accuracy are both displayed between $k = 7$ and $k = 11$, the highest values of each being $k = 11$. Although recall is not displaying the best results for $k = 11$, the recall is still performing fairly well at $k = 11$. This tradeoff in this particular dataset is necessary due to the complications that are the consequence of a lower precision rate.

Varying Distance Metric

Since $k = 11$ yielded the most accurate results for the Euclidean distance metric, this k value is used to compare the output from other tested metrics. Results are easily

comparable in Figure 26. By visual inspection, it is observed that the Euclidean distance outperforms all metrics in accuracy, precision, and recall.

These results are interesting due to the claims made by Alkasassbeh et al. (2015). These researchers state that not only is the Hassanat distance metric more successful for their dataset, but that it is the most successful for nearly all datasets tested. The researchers claim that the results of the Hassanat distance metric "demonstrate the superiority of this distance metric over the traditional and most-used distances, such as Manhattan distance and Euclidean distance." However, Prasath et al. (2019) conclude "that Hassanat distance performed the best when applied on *most* datasets." Important statements are made stating that there is never an overall optimal distance metric. This conclusion is most widely shared amongst previous researchers.

Nevertheless, considering the support for the Hassanat distance metric, it is interesting that the Euclidean distance metric yielded the best results in this study. Although this finding does not support conclusions made by some authors, the findings further support the idea that no singular distance metric will be suitable for all datasets.

Summary

The process performed gives insight into the processes of machine learning and how machine learning algorithms are built. Using Excel to replicate R commands/procedures shows how data is being manipulated throughout the algorithm. Having these steps broken down and visualized aids in understanding the data, allowing for a deeper understanding of ways to improve models.

These processes are also applicable to any dataset. Conclusions drawn confirmed the idea that no distance metric will yield the best results for all datasets. Knowing this

shortcoming, it is important to understand and have the ability to determine the best route for choosing an appropriate k value and distance metric.

After values are decided, a unique algorithm can be implemented within the workplace to more accurately identify the customers committing fraudulent activity. Having a custom algorithm that takes into account the structure and different types of variables present in a particular data set is the first step in devising a better system that detects fraud.

Important Notes

Although many conclusions were drawn from the research, there are many notes to keep in mind. This research was done with a potentially fabricated dataset. This means that k values and distance metrics that performed well may not perform the best on other auto insurance data. However, the process for acquiring the most suitable k value and distance metric will follow the same pattern.

It should also be noted that an abundant amount of valid research was conducted for the Hassanat distance metric. The Hassanat metric performed very well when used on several different datasets. This research does not aim to discredit the accuracy of the Hassanat distance metric. It instead further proves that regardless of the previous success rate of a distance metric (or k value), a singular algorithm will not always provide the most accurate results.

APPENDIX A EXCEL/VBA CODE

VBA CODE “EUCLID”

```
Public Function EUCLID(r1 As Range, r2 As Range) As Double
Dim i As Long, n As Integer, psum As Double
n = r1.Cells.Count
ReDim newvec(n) As Double
For i = 1 To n
newvec(i) = (r1(i).Value - r2(i).Value) ^ 2
Next i
psum = 0
For i = 1 To n
psum = psum + newvec(i)
Next i
EUCLID = Sqr(psum)
End Function
```

VBA CODE “MANHATTAN”

```
Public Function MANHAT(r1 As Range, r2 As Range) As Double
Dim i As Long, n As Integer, psum As Double
n = r1.Cells.Count
ReDim newvec(n) As Double
For i = 1 To n
newvec(i) = (r1(i).Value - r2(i).Value)
Next i
```

```
psum = 0
For i = 1 To n
psum = psum + Abs(newvec(i))
Next i
MANHAT = Abs(psum)
End Function
```

VBA CODE “CHEBYSHEV”

```
Public Function CHEB(r1 As Range, r2 As Range) As Double
Dim i As Long, n As Integer, Result As Double
n = r1.Cells.Count
ReDim newvec(n) As Double
For i = 1 To n
newvec(i) = Abs(r1(i).Value - r2(i).Value)
Next i
CHEB = Application.WorksheetFunction.Max(newvec)
End Function
```

VBA CODE “HASSANAT”

```
Public Function HASS(r1 As Range, r2 As Range) As Double
Dim i As Long, n As Integer, psum As Double
n = r1.Cells.Count
ReDim newvecmin(n) As Double
```

```
ReDim newvecmax(n) As Double

ReDim D(n) As Double

psum = 0

For i = 1 To n

newvecmin(i) = Application.WorksheetFunction.Min(r1(i).Value, r2(i).Value)

newvecmax(i) = Application.WorksheetFunction.Max(r1(i).Value, r2(i).Value)

D(i) = 1 - (1 + newvecmin(i)) / (1 + newvecmax(i))

psum = psum + D(i)

Next i

HASS = psum

End Function
```

APPENDIX B R CODE

R CODE USED TO CREATE CROSS TABLES

```
install.packages("gmodels")

library(gmodels)

install.packages("class")

library(class)

install.packages("caret")

library(caret)

claims <- read.csv("Auto Insurance Claims Data Randomized.csv", stringsAsFactors=

FALSE)

str(claims)

table(claims$fraud_reported)

round(prop.table(table(claims$fraud_reported))*100,digits=1)

claims$CityState <- paste(claims$incident_city,claims$incident_state)

claims$MakeModel <- paste(claims$auto_make, claims$auto_model)

str(claims)

claims <- claims[-c(10,25,3,23,24,36,37,40)]

str(claims)

claims$policy_bind_date <- as.Date(claims$policy_bind_date)

str(claims$policy_bind_date)

claims$policy_bind_date <- julian(claims$policy_bind_date)

claims$incident_date <- as.Date(claims$incident_date)

claims$incident_date <- julian(claims$incident_date)
```

```
str(claims)
normalize <- function(x) {return((x-min(x))/(max(x)-min(x)))}
claims_n <-
as.data.frame(lapply(claims[c(1,2,3,6,7,8,14,15,16,21,22,24,25,27,28,29,30,31)],
normalize))
str(claims_n)
claims_o <- claims[-c(1,2,3,6,7,8,14,15,16,21,22,24,25,27,28,29,30,31,32)]
str(claims_o)
dmy <- dummyVars("~ .", data=claims_o)
claims_o <- data.frame(predict(dmy, newdata=claims_o))
str(claims_o)
claims_final <- cbind(claims_n, claims_o,claims$fraud_reported)
str(claims_final)
claims_train <- claims_final[1:800, 1:184 ]
claims_test <- claims_final[801:1000, 1:184 ]
claims_train_labels <- claims[1:800, 32]
claims_test_labels <- claims[801:1000, 32]
claims_test_pred <- knn(train=claims_train, test=claims_test, cl=claims_train_labels,
k=13)
CrossTable(x=claims_test_labels, y=claims_test_pred, prop.chisq=FALSE)
```

REFERENCES

Alkasassbeh, M., Altarawneh, G. A., & Hassanat, A. B. (2015). On enhancing the performance of nearest neighbor classifiers using hassanat distance metric. *Canadian Journal of Pure and Applied Sciences*, *9(1)*. Retrieved from http://www.researchgate.net/publication/270515076_On_Enhancing_The_Preformance_ Of_Nearest_Neighbor_Classifiers_Using_Hassanat_Distance_Metric

Altman, Naomi S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*. *46(3)*, 175–185. doi:10.1080/00031305.1992.10475879.

Bodaghi, A. & Teimourpour, B. (2018). The detection of professional fraud in automobile insurance using social network analysis. *arXiv*. Retrieved from http://www.researchgate.net/publication/325685541_The_Detection_of_professional_fraud_in_automobile_insurance_using_social_network_analysis

Chudgar, D. J. & Asthana, A. K. (2013). Life insurance fraud: risk management and fraud prevention. *International Journal of Marketing, Financial Services & Management Research*, *2(5).* Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2900674

Cover, Thomas M. & Hart, Peter E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory. **13** (1):* 21-27. doi:10.1109/TIT.1967.1053964.

Ehsani, R. & Drablos, F. (2020). Robust distance measures for k-NN classification of cancer data. *Cancer Informatics, 19(1-9).* doi: 10.1177/1176935120965542

Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. nonparametric discrimination: consistency properties. *International Statistical Review / Revue Internationale de Statistique*, *57*(3), 238–247. https://doi.org/10.2307/1403797

Hassanat, A. B. (2014). Dimensionality invariant similarity measure. *Journal of American Science, 10(8)*, 221-26.

Kalwihura, J. S. & Logeswaran, R. (2019). Auto-insurance fraud detection: A behavioral feature engineering approach. *Advanced Scientific research, 7(3).* doi: http://dx.doi.org/10.31838/jcr.07.03.23

Lantz, B. (2019). Machine learning with R: Expert techniques for predictive modeling. *Packt Publishing*.

Morley, N., Ormerod, T. C., & Ball, L. J. (2006). How the detection of insurance fraud succeeds and fails. *Psychology, Crime & Law, 12(2),* 163-180. doi: 10.1080/10683160512331316325

Prasath, V. B., Alfeilat, H. A., Hassanat, A. B., Lasassmeh, O., Tarawneh, A. S., Alhasanat, M. B., & Salman, H. S. (2019). Effects of distance measure choice on k-NN classifier preformance: A review. Retrieved from http://arxiv-exportlb. library.cornell.edu/pdf/1708.04321

Shung, K. P. (2020, April 10). *Accuracy, precision, recall or F1?* Medium. Retrieved from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

1.2 distance between two points; circles. (n.d.). Retrieved from https://www.whitman.edu/mathematics/calculus_online/section01.02.html