Universidad Nacional Autónoma De México

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN
INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS
Y EN SISTEMAS

Investigación, implementación y medición de distintos algoritmos para la detección de anomalías

T E S I S

QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
SERGIO IVÁN MOTA PANTOJA

DIRECTOR DE TESIS:
DR. JOSÉ ANTONIO NEME CASTILLO
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

Unidad Académica del IIMAS, Yucatán. **Abril, 2021**

Resumen

Barnett y Lewis (1994) definen las anomalías como observaciones (o subconjunto de observaciones) que parecen ser inconsistentes con el resto de datos en un cierto conjunto.

La detección de anomalías ha sido utilizada para la localización y posterior eliminación de observaciones peculiares en un conjunto de datos. En las últimas décadas ha aumentado la investigación en detección de anomalías, ya que éstas son una fuente de información invaluable respecto al tema de interés en una investigación. Una anomalía puede indicar un comportamiento dentro del sistema al que hay que prestarle atención.

La detección de anomalías en el pasado pertenecía exclusivamente al campo de la estadística, pero ha logrado consagrarse junto a las ciencias de la computación, esto gracias a los múltiples beneficios que ha brindado a diferentes problemáticas. Algunos ejemplos de aplicaciones que recurren a la detección de anomalías son las siguientes: detección de fraudes, investigaciones en el área farmacéutica, procesamiento de préstamos, monitoreo de condiciones médicas, rendimiento en redes de computadoras, detección de anormalidades en imágenes, análisis de imágenes satelitales, monitoreo en series de tiempo, entre otros (Hodge & Austin, 2004).

La presente tesis explora un conjunto de algoritmos para la detección de anomalías, con el objetivo de ayudar a los investigadores a identificar cuál de las propuestas existentes es la más aconsejable para cada muestra de datos.

Agradecimientos

A mi padre José Mota Soto que me dio todo sin esperar nada, por ser la ayuda que siempre requerí, por enseñarme todo lo que sé ahora. Siempre vivirás conmigo en mi corazón.

A mi madre María Rosario Pantoja Torres, por darme más de lo que he merecido, por sacrificar todo en pro de la familia. Te quiero mamá.

A mi hermana Maricela Mota Pantoja, por ser una segunda madre para mí, por apoyarme en momentos complicados. Te quiero hermana.

A mi pareja Ana Kárem Allende Calderón, por ser mi apoyo sentimental y siempre estar a mi lado sin importar la circunstancia. Te amo.

A mi tutor José Antonio Neme Castillo, por apoyarme en momentos difíciles, ser un excelente maestro, ser un gran amigo, pero más importante, ser mejor persona. Espero que sigamos trabajando juntos.

A los Doctores del IIMAS, Mérida, por arroparme en una ciudad nueva para mí, por tenerme paciencia y comprensión. Espero verlos pronto.

A mi cuñado Fernando Ortiz por apoyar a la familia en todo. Sabes que cuentas conmigo.

A mi hermana Beatriz Angélica Mota Pantoja, por ser parte importante de la familia. Te quiero hermana.

A mis sobrinas Ivette Anyari Tepepa Mota y Nataly Yamilé Tepepa Mota, por permitirme actuar como su tutor, eso me ha enseñado a tomar responsabilidad. Sigan adelante con sus estudios.

Al CONACYT, por permitirme seguir avanzando como profesional y como persona.

Lista de Acrónimos

BST *Binary Search Tree* (Árboles Binarios de Búsqueda).

DA Detección de Anomalías.

ESD Desviación Extrema Estudiantil

IA Inteligencia Artificial.

KDE *Kernel Density Estimation* (Estimación de Densidad por Kérnel)

k-nn *k-nearest neighbor* (k vecinos más cercanos).

LOF *Local Outlier Factor*.

LRD *Local Reachability Density* (Distancia Local de Densidad).

MAD *Median Absolute Deviation* (Desviación Absoluta Mediana)

ML *Machine Learning* (Aprendizaje Automático).

NP No Polinomiales.

OE Operaciones Elementales.

P Polinomiales.

PCA *Principal Component Analysis* (Análisis de Componentes Principales).

RD *Reachability Distance* (Distancia de Accesibilidad).

Índice general

Índice de figuras

ÍNDICE DE FIGURAS

Índice de tablas

CAPÍTULO 1

Introducción

En las últimas décadas el análisis de datos se ha convertido en una herramienta vital para mejorar la comprensión de nuestro mundo. El análisis de datos ha brindado un marco de referencia en el cual se pueden basar distintas investigaciones, esto ayuda a crear una metodología propia para cada tipo de investigación. Gracias a este marco de referencia, así como a nuevas investigaciones propias del análisis de datos, se ha podido avanzar a tal punto que es posible, hoy en día, el tratamiento de grandes cantidades de datos. El tratamiento de esta enorme cantidad de información solo es posible con la metodología correcta y algoritmos apropiados.

Existen diferentes ramas de la inteligencia artificial, como lo son la visión computacional, el aprendizaje automático y el procesamiento de lenguaje natural, que se sirven del análisis de datos para obtener una mejor comprensión del fenómeno a investigar, esto ocurre en fases anteriores a la etapa de modelado. La inteligencia artificial también aporta mucho al análisis de datos, ya que permite que existan análisis con características de predicción, lo último ayuda a que los análisis sean más robustos.

El análisis de datos está constituido por una serie de procesos que se tienen que llevar a cabo para poder crear descripciones de una investigación, o generar predicciones en el caso de hacer uso de modelos de inteligencia artificial. Uno de estos procesos es el llamado análisis exploratorio de datos, en este lo que se busca es conocer de forma somera al menos, el comportamiento de los datos, así como

deducir qué algoritmos pueden dar mejores resultados y lo más importante de este proceso, detectar posibles contratiempos que puedan surgir en etapas posteriores. Una manera de evitar posibles contratiempos es encontrar anormalidades en los datos, por esa razón en el análisis exploratorio de datos se contempla una actividad llamada detección de anomalías, a la cual está enfocada la presente tesis.

La detección de una anomalía no es un proceso trivial, ya que esta por su carácter distintivo de los demás datos no es fácil de encontrar a simple vista. Para hallar las anomalías en distintos tipos de datos es preferible conocer diferentes algoritmos para su detección. La presente tesis explora un conjunto de dichos algoritmos para la detección de anomalías, con el objetivo de ayudar a los investigadores a identificar cuál proceso es el mejor en determinado tipo de datos en los que estén trabajando.

1.1 Problemática

Actualmente la ciencia de datos ha tomado mucha relevancia, esto gracias a la enorme cantidad de información que se genera hoy en día, y las dificultades que conllevan su adecuado procesamiento. Lo último debido, principalmente, a las redes sociales que generan información en cantidades colosales.

Uno de los problemas de trabajar con datos masivos es identificar cuando la información es relevante a la investigación, por esa razón es necesario conocer temas referentes al análisis de datos, de esa manera se puede sacar el mayor provecho a la enorme cantidad de información generada hoy en día.

Como se ha dicho anteriormente, la detección de anomalías no es un asunto trivial, más aún cuando se tratan de identificar dentro de un conjunto de datos de gran escala. Detectar las anomalías siempre va a ser redituable al avanzar en la investigación, ya que se puede prevenir algún improvisto en nuestra etapa de modelado.

Las anomalías pueden ser una fuente invaluable de conocimiento, pues pueden brindar información rara vez vista en un fenómeno estudiado.

La detección de anomalías representa un desafío, ya que no hay un método universal para realizar dicha identificación. Lo anterior ocurre no nada más por la heterogeneidad de datos, sino también como consecuencia del origen diverso de dichos datos. Los conjuntos de datos pueden tener distintas cantidades de atributos, aunado a que cada atributo cuenta con una distribución que varía en cada caso.

Hay una amplia variedad de algoritmos para realizar detección de anomalías, varios de ellos encuentran su origen en distintas disciplinas, como pueden ser la estadística y el aprendizaje automático. Dentro del aprendizaje automático se pueden encontrar métodos de detección basados en redes neuronales artificiales, de igual manera se pueden hallar soluciones mediante aprendizaje profundo. Esta tesis busca dar una guía para que los investigadores encuentren el procedimiento acorde a su información, dependiendo de: tipos de datos, cantidad de datos, cantidad de atributos, tipos de conjuntos de datos, etc.

1.2 Objetivo General

Lograr un compendio de distintos algoritmos para la detección de anomalías, esto con sus respectivos análisis en la eficacia, en circunstancias determinadas, y la eficiencia del algoritmo. De igual manera se bridarán ejemplos de la ejecución de los algoritmos, esto en el lenguaje de programación *Python*.

Se busca crear una tesis la cual sirva como referencia para futuros análisis de datos, esto será de ayuda en distintas áreas de investigación, como pueden ser: la Minería de Datos, la Inteligencia Artificial, Ciencia de Datos, Bioinformática, etc.

1.3 Objetivos Específicos

1. Descripción de distintos tipos de algoritmos para la detección de anomalías.
2. Definir en qué condiciones es mejor usar un algoritmo en lugar de otro.
3. Revisión de los aspectos matemáticos en general y geométricos en particular que subyacen en los supuestos detrás de cada algoritmo.

4. Mostrar la eficiencia y eficacia de cada algoritmo descrito.

1.4 Hipótesis

Si se logra crear un material de consulta para la detección de anomalías, ayudará a las distintas áreas de investigación a encontrar de manera rápida el algoritmo que más les satisfaga. Identificar al algoritmo de detección de anomalías más eficiente restará el tiempo de indagación, y al mismo tiempo acelerará la obtención de los resultados finales del análisis exploratorio.

La detección de anomalías es una etapa del análisis de datos que suele soslayarse. Aquí proponemos, que el contar con un estudio sistemático dotará a los especialistas de una idea más clara sobre el tipo de algoritmo que más conviene a sus datos, por ende, a su investigación.

1.5 Estructura de la Tesis

La actual tesis se secciona por capítulos, se da un bosquejo a continuación:

- En el capítulo 1, se da una introducción al tema de detección de anoamalías, los objetivos de la presente tesis y la hipótesis de la investigación.
- En el capítulo 2, se dan los antecedentes, también una descripción de los diferentes tipos de anomalías que existen.
- En el capítulo 3, se describe la detección de anomalías por enfoque estadístico, como lo son la prueba de Grubbs, gráficas de cajas y bigotes y MAD.
- En el capítulo 4, se describe la detección de anomalías con enfoque por densidad, se explican los algoritmos de LOF y bosque de aislamiento.
- En el capítulo 5, se describe la detección de anomalías con enfoque por medio de redes neuronales artificiales, también se explican las redes neuronales autocodificadoras.

- En el capítulo 6 se dan las conclusiones generales de la investigación.
- También se cuenta con un apéndice, en este se explica la complejidad computacional de un algoritmo.

CAPÍTULO 2

Antecedentes

El término análisis de datos fue descrito por primera vez por el matemático John Tukey, Tukey nos dice:

“procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data." ((Son los) procedimientos para analizar datos, (las) técnicas para interpretar los resultados de dichos procedimientos, (las) formas de planificar la recopilación de datos para hacer su análisis más fácil, más preciso o más exacto, y toda la maquinaria y resultados de las estadísticas (matemáticas) que se aplican al análisis de datos) (Tukey, 1962).

John Tukey, al desarrollar de forma sistemática las metodologías para el análisis de datos, definió el proceso de análisis exploratorio de datos bajo una simple idea, Tukey nos comenta “It is important to understand what you CAN DO before you learn to measure how WELL you seem to have DONE it.” (Es importante comprender lo que se puede hacer antes de aprender a medir qué tan bien parece haberlo hecho) (Tukey, 1977).

El análisis exploratorio de datos se basa en identificar de una manera somera el alcance del análisis, esto por medio de diferentes gráficas y métodos estadísticos. El análisis exploratorio de datos también comprende la detección de anomalías, la

cual ayuda, como lo describió John Tukey, a hacer el análisis más fácil, preciso o exacto.

Todos los siguientes son términos semejantes: anomalía, valor atípico, medición atípica, dato anómalo, vector anormal, o una variante de dichos términos.

2.1 Diferentes tipos de anomalías y su tratamiento

Existen diferentes tipos de anomalías, estás se pueden clasificar por sus distintos orígenes. Ya hemos hablado de algunas, pero en este capítulo se busca dar una catalogación formal que servirá como referencia para más adelante.

Debido a los distintos orígenes con los que cuentan las anomalías, estas tienen un diferente tratamiento al momento de ser identificadas. Los procesos aquí descritos son marcos de referencias.

Básicamente, existen tres tipos de anomalías:

1. Por error humano, estas anomalías surgen al ser un error de captura, medición, valores faltantes, etc.

2. Por datos de origen propio de la distribución, pero pueden llegar a ser catalogados como anomalía, esto dado a su poca probabilidad de ocurrencia dentro del fenómeno.

3. Por comportamiento sospechoso, dependiendo del enfoque que maneje nuestro algoritmo para la DA, esto se explicará en capítulos más adelante.

En el procedimiento de DA puede suceder que a simple vista se distinga una anomalía, esta al ser fácilmente identificable es probable que pertenezca al tipo de error humano, lo anterior se puede explicar dada la falta de coherencia en el dato. Por ejemplo, una edad con valor menor a cero, una distancia negativa, una altura de cien metros para un humano, etc.

Las anomalías propias de la distribución y por comportamiento sospechoso pueden ser más complejas de identificar. La presente tesis presenta un conglomerado de algoritmos que nos ayudan a detectar dichas anomalías.

Teniendo en cuenta los diferentes tipos de orígenes que pueden tener las anomalías, se dan las siguientes recomendaciones para su debido tratamiento.

2.1.1 Anomalías con valor informativo

Una observación con valor informativo puede surgir cuando la anomalía es proveniente del comportamiento natural de los datos, solo que esta difiere de los datos restantes. Esta observación, dada sus características poco probables a ocurrir, puede llegar a ser una fuente de información sumamente importante, por lo que es recomendable aislarlas para una investigación independiente.

También es posible anexarlas al conjunto de datos de la muestra, esto dependiendo de si la metodología es lo suficientemente robusta para que no afecte los resultados del experimento.

2.1.2 Anomalías procedentes de comportamiento sospechoso

Un comportamiento sospechoso es un evento poco probable generado con cierta malicia, lo anterior se hace para sacar algún beneficio de la poca resiliencia del sistema ante esa eventualidad. Este tipo de anomalías cuentan con un valor informativo invaluable.

Existen algoritmos para la DA que tienen como objetivo generar una clasificación con una sola clase, esta clase representa un comportamiento no anómalo[1], con lo cual, al entrar nuevos datos se verifica si ese dato existe dentro de dicha clase o es categorizado como anomalía. Este tipo de método es llamado Detección de Novedades (Novelty Detection) (Yahaya, Lotfi, & Mahmud, 2019), ya que puede

[1] Con no anómalo nos referimos a una clase que no contiene ninguna anomalía.

detectar anormalidades de una manera inmediata, por eso mismo, es ampliamente utilizado para detectar fraudes financieros, fallas estructurales, correo spam, etc.

A manera de ejemplo, un individuo que lleva una tarjeta, que no es de su propiedad, al hacer un uso de dicha tarjeta el banco puede activar una alarma y proceder con su protocolo para impedir un uso indebido de los fondos de la tarjeta. Esta detección se alcanza mediante un algoritmo de DA.

Ahora, ya conociendo un poco qué es un comportamiento sospechoso, procedemos a sugerir su tratamiento.

Este tipo de anomalías, una vez detectadas, son inmediatamente aisladas, se levanta una alerta y son reportadas a las instancias pertinentes. De igual manera son investigadas para validar su anormalidad. Una vez que se comprobó que pertenecen a casos fuera de la uniclase, son tomados como futuras referencias para detectar próximos casos sospechosos.

2.1.3 Anomalías provenientes de un error humano

En una investigación es probable que surjan errores al momento de tomar los datos y registrarlos, estos datos al no ser propios de la muestra investigada, se les conoce como anomalías provenientes de error humano[2].

Una vez identificadas estas anomalías, es recomendable, dentro de lo posible, volver a tomar el dato y registrarlo en sustitución del anterior. Cuando no es viable volver a tomar la medición, se puede sugerir su eliminación, esto si no crea problemas con datos faltantes posteriormente en la investigación.

Otra posibilidad de su tratamiento puede ser la eliminación del dato anómalo, posteriormente generar una estimación estadística utilizando Estimación de la Densidad por Kérnel (KDE) (Hastie, Tibshirani, & Friedman, 2001). Se comenta lo anterior en modo de sugerencia para el lector, ya que KDE no es abarcado en la presente tesis.

[2] Son de error humano a pesar de que pueden tener su origen en un aparato de medición defectuoso.

2.2 Código ocupado para la presente tesis

Como ya se había comentado, el lenguaje que se va a ocupar para las distintas demostraciones es *Python*, el cual es un lenguaje robusto, también es uno de los lenguajes más usados dentro de la ciencia de datos y cuenta con las herramientas necesarias para cubrir los objetivos de la tesis.

Para contar con una pedagogía que permita lograr una de las metas de la tesis, que es “servir como futura referencia”, se publicará la totalidad del código generado en los ejemplos. Esto dará una mejor comprensión de la tesis y servirá como tutorial para los futuros investigadores.

El código se encuentra en la página de GitHub siguiente:

https://github.com/amigo20th/Tesis_Deteccion_Anomalias

CAPÍTULO

3

Detección de Anomalías con Enfoque Estadístico

3.1 Prueba de Grubbs

Es un método encontrado en la estadística univariada, es decir, solamente se centra en el análisis de una sola variable; también es paramétrica, o sea que es necesario conocer la distribución de la muestra; este método también es conocido como Desviación Extrema Estudiantil (ESD). Esta prueba solo es funcional bajo la hipótesis que dicha variable tenga una distribución normal o distribución gaussiana. Es una prueba sencilla de comprender, pero poco robusta en su funcionamiento, más adelante se comentarán sus puntos negativos.

Este método, al estar basado en estadística univariada, tiene poca exigencia computacional, en otras palabras, no se requiere mucho tiempo o poder de procesamiento para que realice la respectiva detección de anomalías. La prueba de Grubbs, al trabajar bajo el supuesto de tener una variable con una distribución normal, es necesario realizar una prueba estadística para verificar que efectivamente esta variable tiene dicha distribución, esto se puede realizar con una prueba de Shapiro-Wilk (Shapiro & Wilk, 1965) o una prueba de Kolmogorov–Smirnov (Shiryayev, 1992), ambas pruebas validan que determinada variable tenga una distribución normal.

La prueba de Grubbs consta principalmente de encontrar las posibles anomalías en los extremos de la distribución normal, estos extremos pueden ser los valores más bajos o altos que toma la variable. Al tener su base en una distribución gaussiana, se basa en la media y la distribución estándar para realizar la detección. A partir de la información estadística de la variable se calcula $z - score$, es un valor de

proporción que nos indica el nivel de lejanía que tiene determinada observación respecto a la media, esto en proporción a la desviación estándar, es decir:

Sea x una variable numérica, donde $x = \{x_0, x_1, x_2, \dots, x_{n-1}\}$ entonces se tiene:

$$z - score = \frac{|\overline{x} - x_i|}{\sigma(x)} \quad \text{.....(1)}$$

Donde:

- $\overline{x}$ representa la media de la muestra.
- x_i representa el i-ésimo valor de x.
- $\sigma(x)$ representa la desviación estándar de x.

Como se puede observar, $z - score$ calcula la lejanía que tiene un elemento respecto a la media, esto a proporción de la desviación de la muestra.

Por lo tanto, $z - score$ es un valor que medirá el nivel de anormalidad de una observación respecto a toda su distribución. La idea de calcular $z - score$ es la de definir un umbral, este umbral catalogará la observación como anomalía o como valor no anómalo. Es decir, el umbral definirá una lejanía aceptable de un dato respecto a su muestra.

El problema surge al definir dicho umbral, ya que no es cosa trivial, esto dado a las peculiaridades de cada experimento, una de estas peculiaridades, y la más importante, es el número de observaciones que tenga la muestra a analizar.

Para definir un umbral que sea conveniente para el análisis de nuestra investigación, se puede hacer lo siguiente, calcular el valor de $z - score$ para la totalidad de la prueba, posteriormente graficar todos los valores de $z - score$ en un diagrama de dispersión, ver qué valores se escapan en demasía de la mayoría de ellos. Ya definidos los valores anómalos, se busca aislarlos para el posterior análisis del experto.

En dado caso que ningún valor se alejara bastante de la media, esto durante el proceso de definir un umbral, se consultará al experto para tener una referencia clara de qué valores podrían ser anómalos.

3.1.1 Ejemplo de detección de anomalías mediante prueba de Grubbs

Todo el código generado para realizar este ejemplo se puede encontrar en la liga https://github.com/amigo20th/Tesis_Deteccion_Anomalias/blob/master/DA_Prueba_de_Grubbs.ipynb[3]

A continuación, se procede a hacer una pequeña demostración de la aplicación de la prueba de Grubbs para detectar anomalías, esto en un conjunto de datos que pertenece al campo de la biometría, en dicho estudio se midieron las alas de distintos ejemplares de mosca doméstica y se fueron registrando en un conjunto de datos, este conjunto cuenta con una sola variable (Sokal & Hunter, 1955).

Se puede observar que en el conjunto de datos la distribución formada por las distintas longitudes de alas de la mosca doméstica forma una campana de Gauss (Figura 1), entonces se va a insertar un dato artificialmente el cual jugará el rol de ser una anomalía dentro de nuestro conjunto de datos, de esta manera nosotros podemos crear un ejemplo sintético para llevar a cabo nuestra demostración. Al final tenemos una muestra de 101 elementos para el ejemplo cuya distribución se observa en la Figura 2.

Viendo el histograma de nuestro conjunto de datos se puede observar que el dato ingresado artificialmente, es decir, nuestra anomalía, tiene un valor un poco mayor a 60.

A primera vista se puede identificar la normalidad de la distribución, y es más notorio cuando la muestra es grande. De cualquier manera, se debe aplicar una prueba para comprobar formalmente la hipótesis.

[3] Si se requiere es posible copiar o descargar el código para modificarlo al gusto.

En este caso vamos a aplicar la prueba de Shapiro-Wilk para corroborar la distribución normal de la muestra (Shapiro & Wilk, 1965). Nuestro tamaño de muestra es de 101 elementos.

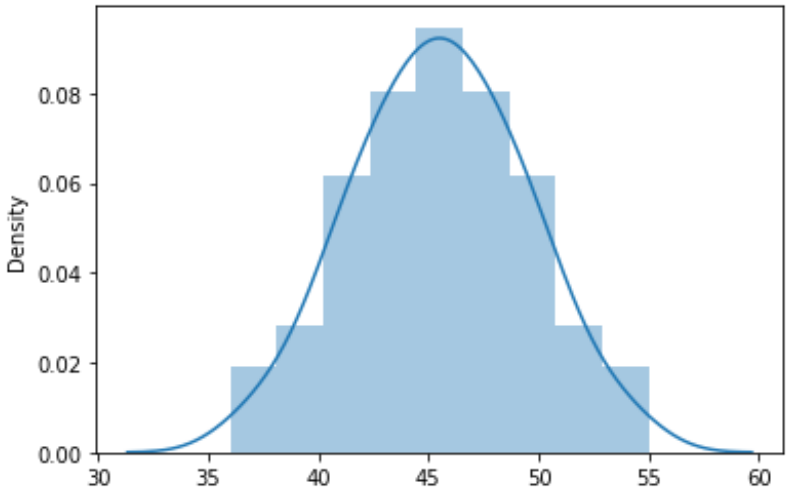

Figura 1. Distribución del tamaño de las alas de la mosca doméstica para la Prueba de Grubbs.

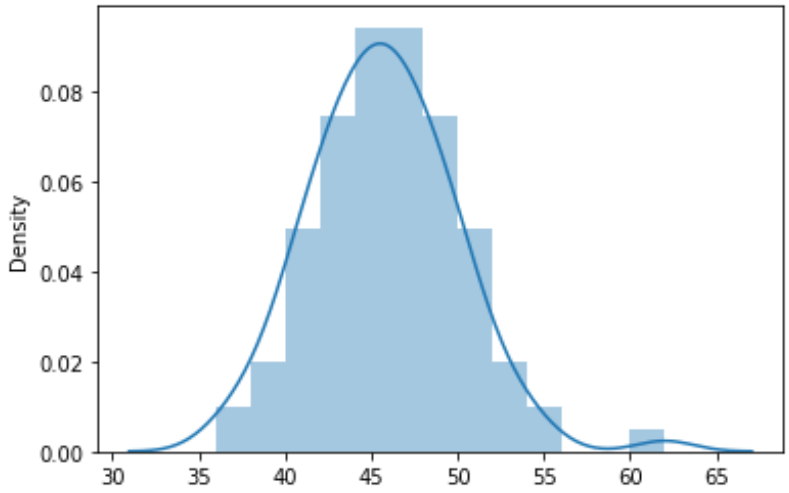

Figura 2. Histograma de alas de la mosca doméstica con una anomalía insertada artificialmente.

La significancia a será del 0.05, es decir del 5 %, si nuestro $p - value$ de la muestra es mayor a la significancia a tendremos evidencia sólida de que nos encontramos ante una distribución normal.

Una vez asegurada la normalidad de la distribución podemos calcular el $z - score$ para cada uno de los datos de la distribución, se grafica el $z - score$ de toda la muestra en una gráfica de dispersión para observar mejor los valores de $z - score$ (Figura 3).

Como se puede observar, el dato que fue insertado de manera artificial se encuentra lejos de las demás observaciones, la media del $z - score$ está representada por la línea verde. Hay que recordar que se ingresó el dato de manera manual para ser una anomalía dentro de nuestra distribución.

Para definir el umbral deseado aquí es el momento perfecto para estipularlo, ya que observando la gráfica es posible determinar un divisor para distinguir anomalías de las que no lo son.

Para definir el umbral que nos ayudará a distinguir las anomalías nos basamos en el gráfico de la Figura 3. En este caso en particular un valor de 3.5 es adecuado, con un valor más pequeño podemos generar un sobreajuste[4], por el contrario, si elegimos un umbral muy grande podemos caer en un bajo ajuste[5].

Por lo tanto, todo dato que sobrepase el $z - score$ de 3.5 será catalogado como anomalía, esto está representado en la Figura 4, el umbral está mostrado con una recta de color naranja.

Ya para terminar el proceso de detección de anomalías, solo quedaría apartar el valor anómalo y presentarlo ante un experto, proceso que se explicó en **Diferentes tipos de anomalías y su tratamiento (Sección 2.1)**, para que él nos pueda determinar su procedencia (de ser posible), su debido tratamiento y develar información valiosa que pueda aportar a la investigación.

[4] A sobreajuste nos referimos al término *overfitting* utilizado en ML.

[5] A bajo ajuste nos referimos al término *underfitting* utilizado en ML.

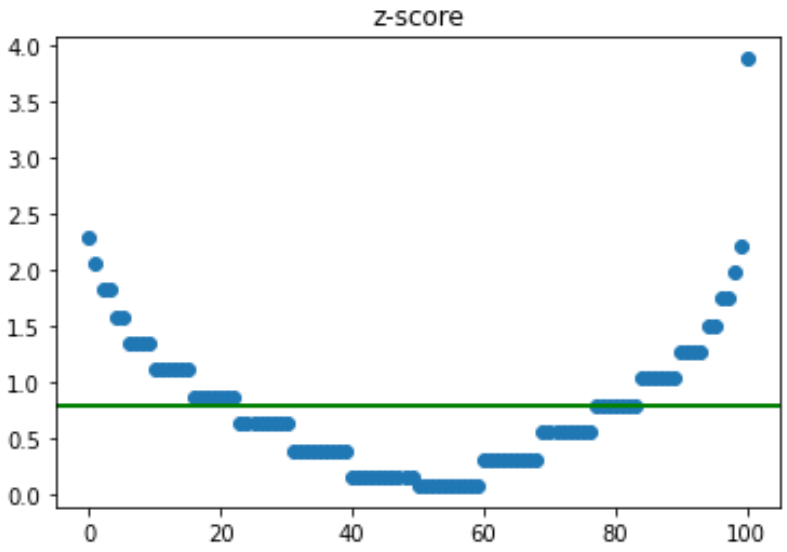

Figura 3. Gráfica de dispersión del $z - score$ de la muestra.

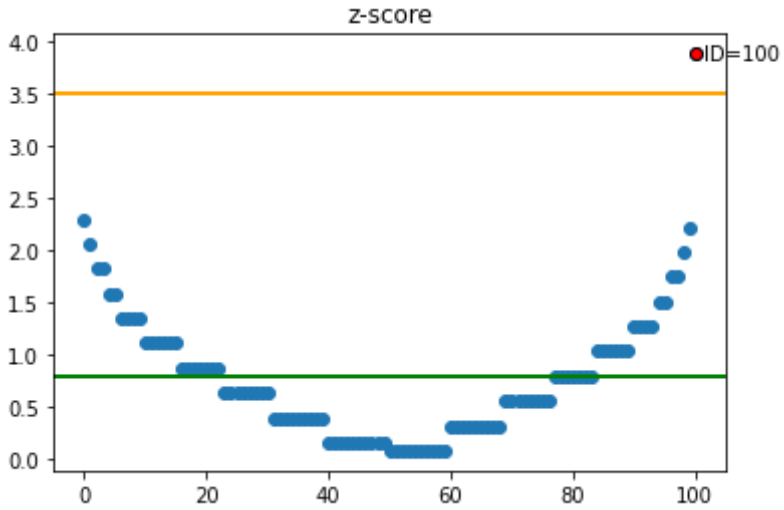

Figura 4. Umbral para catalogar anomalías, recta naranja.

Para un ejercicio de detección de novedades, es decir, cuando se agregan datos nuevos a nuestra variable, que en este caso es la medición de las alas de distintas moscas domésticas, se calcularía el $z - score$ del nuevo dato y si es menor al umbral estipulado, se agrega a la variable, si no, es rechazado. Cabe señalar que en la prueba de Grubbs, en un ejercicio de detección de novedades, es bajo el costo

computacional requerido, ya que se conoce el valor de la media y de la desviación estándar con anterioridad.

Por lo tanto, la prueba de Grubbs es una buena opción para realizar la detección de anomalías en sistemas que funcionan en tiempo real.

3.1.2 Complejidad computacional de la prueba de Grubbs

Para el cálculo de la complejidad computacional de la prueba de Grubbs, es necesario tomar en cuenta los siguientes pasos:

1. Es necesario el ordenamiento de los datos de la muestra, eso se hace mediante *Merge Sort,* el cual tiene un orden de $O(n\ log\ n)$ (Davidson, Tarjan, Garland, & Owens, 2012).
2. Posteriormente, se calcula el $z - score$ para cada uno de los datos, para dicho cálculo es necesario recorrer tres veces la muestra puesto que es necesario conocer la desviación estándar, entonces el orden de calcular la Ecuación 1 es $O(n)$.
3. Para la detección de anomalías, solo es necesario validar qué datos sobrepasan el umbral estipulado, para hacer lo anterior es necesario recorrer una sola vez la muestra, entonces este punto es de orden $O(n)$.

Por lo tanto, la función asintótica que crece mayormente a la complejidad de la prueba de Grubbs es $O(n\ log\ n)$.

3.1.3 Conclusión

La prueba de Grubbs tiene como principal ventaja tener una complejidad computacional baja, por lo tanto, es excelente para la detección de anomalías en un conjunto de datos grande. Al igual es recomendable en casos de detección de novedades y detección de anomalías en tiempo real.

La parte negativa de la prueba de Grubbs radica en ser paramétrica, es decir, al ser necesario que la muestra tenga una distribución normal, limita a la cantidad de experimentos donde es posible aplicar esta prueba.

Otra desventaja que tiene la prueba de Grubbs es su dependencia con la media, dicho en otras palabras, usar la media como punto de referencia para calcular la anormalidad de un dato no es recomendado en todos los casos, esto es dado que la media cuenta con una alta sensibilidad a las anomalías (Leys, Klein, Bernard, & Licata, 2013).

3.2 Gráficas de Cajas y Bigotes (BoxPlots)

Fueron introducidas como una herramienta dentro del análisis exploratorio de datos por John Tukey en su libro *Exploratory Data Analysis* (Tukey, 1977).

Este tipo de gráficas pertenece al campo de la estadística descriptiva, tienen la característica de ser no paramétricas, es decir, no es necesario comprobar algún supuesto respecto a la distribución que presente la muestra. También las gráficas de cajas y bigotes se encuentran en la estadística univariada como Grubbs, la diferencia radica en que las gráficas de cajas pueden contener todas las variables de una población en un solo gráfico. Las variables tienen que ser del tipo cuantitativa

Estos diagramas ayudan a visualizar la dispersión y simetría con la que cuenta nuestra muestra dentro de una recta numérica, esto quiere decir que nos ayuda a ver entre cuáles valores se encuentra el 50 % más cercano a la mediana de nuestra muestra, lo último es representado por un par de cajas cerradas.

Nos auxilia a observar los valores máximos y mínimos del conjunto de datos, estos vienen escenificados por segmentos de recta, conocidos como bigotes.

Por último, nos sirven para observar gráficamente los datos atípicos de una muestra, estos son todos los valores que son más pequeños que el valor mínimo, o los que son mayores al valor máximo.

Las gráficas de cajas y bigotes ayudan a mostrar de una manera resumida el comportamiento que tiene la población, al ser un método visual son muy intuitivas de interpretar.

Las gráficas, en general, numerosas veces ayudan a visualizar las anomalías, pero no son un método de identificación por sí mismas. Es necesario contar con un algoritmo para dicha detección, de esta manera se puede contar con un método computable y consistente para la detección de atípicos, más adelante se explicará el método usado para la detección de anomalías que se utiliza en los diagramas de cajas y bigotes.

Existen distintas modificaciones al diagrama de cajas y bigotes realizadas por los investigadores a través del tiempo, en esta tesis nos apegaremos a la versión original presentada por Tukey.

3.2.1 Cuartiles de una muestra de datos

Los percentiles fueron planteados por Francis Galton en 1885 en la publicación *Anthropometric Percentiles* (Galton, 1885), Galton planteó en dividir una muestra de datos en cien partes iguales y llamar a los divisores que crean estas partes como percentiles. Los cuartiles son similares a los percentiles, solo que la muestra se divide en cuatro partes, no cien.

Antes que todo, cabe recordar que estamos sobre estadística univariada, así que para la explicación solo se manejará una variable.

Primero, es necesario que nuestra variable la ordenemos de menor a mayor para poder realizar la división, la cantidad de cuartiles a calcular son tres, ya que eso creará una división de nuestra muestra en cuatro partes iguales.

Posteriormente se calculan los cuartiles de la siguiente manera:

$$Q_k = \frac{k(n+1)}{4} \quad \text{.....(2)}$$

Teniendo que:

k es el número del cuartil, es decir 1, 2 o 3.

n es el número de elementos en la muestra.

Por ejemplo, para un tamaño de muestra de 9 elementos los cuartiles quedarían de la siguiente manera:

$$Q_1 = \frac{1\ (9+1)}{4} = \frac{10}{4} = \ 2.5$$

$$Q_2 = \frac{2\ (9+1)}{4} \ = \ \frac{10}{2} = \ 5.5$$

$$Q_3 = \frac{3\ (9+1)}{4} = \frac{30}{4} = \ 7.5$$

Entonces, en este caso en particular se encontrarían los cuartiles en la posición 2.5, 5.5 y 7.5.

Los cuartiles son las posiciones donde es necesario crear los cortes en la muestra para dividirla en 4 partes iguales, cada posición cuenta con un valor y este valor depende totalmente de la muestra tomada, por lo tanto, si se quiere conocer los valores que tiene cada cuartil en la muestra es necesario calcular a qué valor corresponde cada cuartil.

Cuando los cuartiles existen en los enteros solo se toma el valor que tiene la posición del cuartil. En el ejemplo anterior vemos que cuenta con decimales, para estos casos el valor que tome la posición del cuartil va a ser el promedio entre el valor que tome la posición menor más próxima y el valor de la posición mayor más próxima del cuartil.

Otro concepto clave que es necesario introducir en el tema de los cuartiles es el rango intercuartil, el cuál es la diferencia entre el cuartil tres y el cuartil uno, este

rango nos servirá para poder representar el 50% de nuestra muestra. Quedando de la siguiente manera:

$$IQR = Q_3 - Q_1 \quad \text{.....(3)}$$

Cabe destacar que el cuartil dos siempre coincidirá con la mediana de la muestra.

Ya conociendo que son los cuartiles y el rango intercuartil, podemos pasar a explicar cómo se realiza una gráfica de cajas y bigotes.

3.2.2 Diagrama de cajas y bigotes

Como lo mencionamos anteriormente el diagrama de cajas y bigotes fue introducido por John Tukey. Larsen (1985) nos comenta, "These displays have been found to be useful in numerous areas. The author has found that such displays can be effectively used for the presentation of the diverse collections of data..." (Se ha descubierto que estas exhibiciones son útiles en numerosas áreas. El autor ha descubierto que estas exhibiciones pueden usarse eficazmente para la presentación de diversas colecciones de datos ...).

Los diagramas de cajas y bigotes son propios del análisis exploratorio de datos, cabe recordar que, en este segmento del análisis, uno de los pasos de principal interés es dar una visualización somera para poder estimar cuáles son los mejores métodos o modelos para aplicar posteriormente. Estos diagramas, son muy efectivos a la hora de mostrar un resumen visual de nuestra muestra de datos.

Los diagramas de cajas y bigotes son robustos, la robustez radica en que no es necesario estimar ningún tipo de distribución en nuestra hipótesis, el diagrama puede graficarse sin importar la distribución que tenga nuestra muestra.

Otra característica muy importante, y que es la principal para la presente tesis, es que logran dar una muestra visual de cómo se comportan las anomalías dentro de nuestra muestra. Estas anomalías son mostradas fuera de un rango, este rango es conocido como rango de aceptación (Barbato, Barini, Genta, & Levi, 2011) y está delimitado por los bigotes del diagrama.

El diagrama de cajas y bigotes toma como anomalías los datos que se distancian mucho de la mediana dentro de la muestra, ya sea que este alejamiento sea por ser valores muy pequeños o valores muy grandes (Figura 5).

La función principal de los bigotes en un diagrama de cajas es la de establecer un rango de aceptación de nuestra variable, este rango de aceptación va a contener la mayoría de nuestros datos. Todo dato que se encuentre fuera de los bigotes, es decir, fuera de nuestro rango de aceptación, va a ser considerado una anomalía. En la Figura 5 es posible reconocer las anomalías porque tienen una forma de diamante.

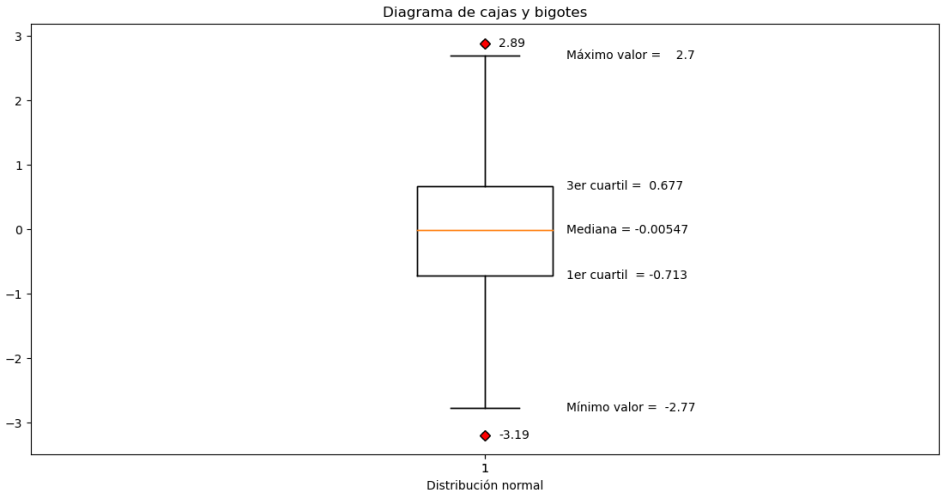

Figura 5. Diagrama de cajas y bigotes de una distribución normal generada aleatoriamente.

3.2.3 Creación del diagrama de cajas y bigotes

Para la creación del diagrama de cajas y bigotes es necesario ordenar nuestra variable de manera ascendente, es decir de menor a mayor. Una vez teniendo nuestra variable ordenada de esa manera, se pasa a calcular la mediana de la muestra, esta mediana va a ser nuestro segundo cuartil.

Posteriormente se calculan los cuartiles primero y tercero, estos le darán forma a la caja, este procedimiento se hace de la misma manera que se explicó anteriormente.

Ahora solo falta determinar el rango de aceptación establecido por los bigotes del diagrama. Esto se calcula de la siguiente manera.

$$Bigote\ Inferior = Q_1 - 1.5 * IQR \text{.....(4)}$$

$$Bigote\ Superior = Q_3 + 1.5 * IQR \text{.....(5)}$$

La constante 1.5 por la que está multiplicado el rango intercuantil de los bigotes viene establecido por bibliografía (Barbato, Barini, Genta, & Levi, 2011).

Ya teniendo los valores en los cuales radican el bigote superior y el bigote inferior, se trazan segmentos de recta hacia la caja, anteriormente creada, desde el valor de cada bigote.

3.2.4 Ejemplo de detección de anomalías mediante el diagrama de cajas

A continuación, se procede a realizar un ejemplo para la creación de un diagrama de cajas y bigotes aplicado en *Python* con la librería *matplotlib*.

Todo el código generado para realizar este ejemplo se puede encontrar en la liga https://github.com/amigo20th/Tesis_Deteccion_Anomalias/blob/master/DA_Diagrama_de_cajas.ipynb[6]

El conjunto de datos a utilizar en este ejemplo es referente a automóviles del año de 1985, el cual está diseñado para determinar el costo de su seguro dependiendo de sus características, este conjunto de datos fue tomado de la página *UCI Machine Learning Repository*.

Para el ejemplo solamente se tendrá contemplada la variable de anchura que tienen los automóviles, ya que esta variable es cuantitativa y presenta unas cuantas anomalías que resulta interesante analizar.

La liga al conjunto de datos utilizado es la siguiente:

https://archive.ics.uci.edu/ml/datasets/automobile

Para aplicar un diagrama de cajas y bigotes en *matplotlib* solamente es necesario importar la librería y crear el diagrama con la siguiente instrucción.

[6] Si se requiere es posible copiar o descargar el código para modificarlo al gusto.

$$matplotlib.pyplot.boxplot(anchura)$$

Aquí $anchura$ hace referencia a nuestra variable dentro del conjunto de datos. La ejecución de esa instrucción da un resultado aproximado al mostrado en la Figura 6.

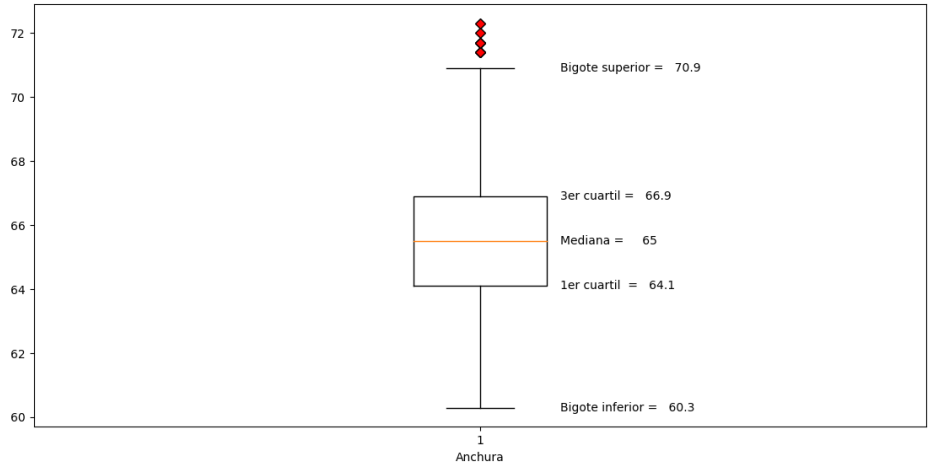

Figura 6. Creación de un diagrama de cajas y bigotes en *Python*.

Aquí se puede ver un resumen de nuestra variable anchura, teniendo la mediana con un valor de 65 pulgadas; con nuestro bigote inferior, es decir nuestro valor mínimo antes de que sea catalogado como un dato anómalo es de 60.3 pulgadas; el 50 % de las anchuras de los automóviles se encuentra entre los valores de 64.1 y 66.9 pulgadas; por último, el valor máximo del conjunto de datos es de 70.9 pulgadas, es decir, toda anchura mayor al valor máximo, será etiquetado como anomalía.

Los automóviles que cuentan con una anchura anómala respecto al conjunto de datos con el que contamos son los siguientes:

- Los autos de la marca *Audi* que tuvieron una anchura anómala son:
  - *Sedan* con cuatro puertas.
  - *Wagon* con cuatro puertas.
- Los autos de la marca *Mercedes-Benz* que tuvieron una anchura anómala son:

    - *Sedan* con cuatro puertas.
    - *Hardtop* con dos puertas.
- El auto de la marca *Porsche* es uno *Hatchback* de dos puertas.

Entonces, se puede observar que los autos que tienen una anchura anómala, que resultan mayores a nuestro valor máximo, son pertenecientes a marcas que cuentan con un coste elevado en su precio, lo cual es consistente, ya que teniendo en cuenta los precios de los materiales y la comodidad que ofrecen dichas marcas, deben de contar con una anchura mayor a la de la mayoría de los automóviles en el mercado.

3.2.5 Complejidad computacional del diagrama de cajas y bigotes

Este tipo de análisis, al pertenecer dentro de la disciplina de la estadística, no tiene una complejidad computacional grande, es decir, no requiere ni de tanto poder de cómputo ni de tanto tiempo para realizar la respectiva detección de anomalías. Lo anterior se puede observar tomando en cuenta los siguientes pasos:

1. Primero, se tiene que ordenar la variable, esto se puede hacer mediante *Merge Sort*, el cual cuenta con una complejidad $O(n\ log\ n)$ (Davidson, Tarjan, Garland, & Owens, 2012).
2. Lo siguiente es calcular sus cuartiles y encontrar su respectivo valor dentro de la muestra. El cálculo de los cuartiles es de $O(1)$ ya que solo es realizar una operación; para encontrar su respectivo valor en la muestra es necesario recorrer toda la variable solo una vez, es decir, es del orden $O(n)$. Por lo tanto, encontrar los cuartiles no crece más rápido que la asíntota $O(n)$.
3. Posteriormente, es necesario calcular los bigotes de la caja, para eso es necesario calcular el rango intercuantil, el cual toma un orden $O(1)$ y luego calcular el valor de los bigotes, de igual manera toma un orden $O(1)$.
4. Para realizar la detección de anomalías es necesario recorrer todos los datos de la muestra, entonces cuenta con un orden $O(n)$.

Por lo tanto, tomando los diferentes órdenes que puede tener cada parte de nuestro análisis, podemos concluir que, para crear una detección de anomalías mediante

un diagrama de cajas y bigotes, su complejidad computacional no crece más rápido que la función asintótica $O(n\ log\ n)$.

3.2.6 Conclusiones

Los diagramas de cajas y bigotes son resúmenes de la información que nos brinda nuestra muestra, esto en una sola gráfica, de igual manera son fáciles de calcular, realizar y de interpretar.

Dentro del análisis exploratorio de datos, particularmente en la detección de anomalías, son robustos ya que no tienen ninguna restricción respecto a la distribución que mantienen los datos.

Los diagramas de cajas y bigotes, además de ayudarnos a mostrar de una manera comprimida nuestra información, en general, nos auxilian a darnos una idea más detallada de la distribución de nuestros datos. De igual manera, fomentan la visualización de los valores que tiene la muestra respecto al porcentaje de la cantidad de los datos, es decir, los cuartiles que tenemos como referencia, como lo son el primer y tercer cuartil que representan el 50% de la cantidad de los datos, este porcentaje se encuentra visualmente dentro de la caja.

Para una aplicación de detección de novedades por medio de diagramas de cajas y bigotes, solamente es necesario tener como referencia el valor del bigote superior y el valor del bigote inferior, de esa manera al recibir un dato nuevo a nuestro conjunto de datos se validaría que exista entre esos dos valores para ser catalogado como un dato propio de la distribución, de manera contraria sería catalogado como anomalía y sesgado para su posterior análisis, el cual llevaría a cabo el experto.

Como punto negativo, este tipo de análisis al ser de carácter univariado, toma las variables de una manera sesgada, es decir que no toma en cuenta la correlación que puedan tener las variables entre sí, esto último a pesar de que pueden tener distintas gráficas de cajas y bigotes en un solo diagrama, cada gráfica pertenecería a una variable del conjunto de datos.

Más adelante veremos algoritmos que tratan a los vectores como un todo, independientemente de la cantidad de variables que tengan, de esa manera encuentran anomalías dentro de su propio conjunto.

3.3 Desviación Absoluta Mediana (MAD)

La Desviación Absoluta Mediana o MAD por su acrónimo en inglés, es una medida robusta de la variabilidad de la muestra. Nos encontramos aún en el campo de la estadística univariada, es decir, MAD solo es aplicable a una dimensión de la muestra a la vez; de manera similar a los dos métodos de detección de anomalías vistos anteriormente, la variable tiene que ser del tipo cuantitativa.

Si bien, para MAD no es necesario el tipo de distribución que tenga la variable, es verdad que, al utilizar una medida de anormalidad basada en el $z - score$, es necesario conocer y cumplir una hipótesis, esta hipótesis es la de normalidad, esto hace que MAD, como método de detección de anomalías, exista dentro de la estadística paramétrica.

MAD fue introducido por Frederick Mosteller y John W. Tukey en su libro *Data Analisys and Regression*. Los autores definen MAD como:

$$MAD = Median\ Absolute\ Deviation = median|x_i - x^*| \quad \text{.....(6)}$$

(Mosteller & Tukey, 1977).

Siendo:

x_i El i-ésimo valor de la muestra.

x^* Es una estimación resistente a la ubicación.

Mosteller y Tukey en su libro toman al primer cuartil de la muestra como x^*, en la presente tesis se tomará la mediana o el segundo cuartil.

Entonces, se cuenta con MAD que, como se dijo anteriormente, es una medida robusta de la variabilidad de la muestra. Recordando, la mediana es más resistente que la media ante la presencia de anomalías en el conjunto de datos.

Ahora bien, se tiene el $z - score$, este representa la estimación de anormalidad, con él trabajamos en la **Prueba de Grubbs**, solamente que en este método se va a ocupar una variación, aunque parte de la misma lógica. Está vez el $z - score\ modificado$ se define como:

$$modified\ z - score = M_i = \frac{0.6745(x_i - x^*)}{MAD} \text{.....(7)}$$

(Iglewicz & Hoaglin, 1993).

Donde x_i y x^* son los mismos que en la ecuación MAD.

Se puede observar una constante dentro de la ecuación anterior, esto hace referencia a la normalidad de la muestra, Howell nos dice

> *"For a normally distributed variable, the M.A.D. is equal to 0.6745 times the standard deviation (SD) and 0.8453 times the average deviation (AD)."* (Para una variable con distribución normal, MAD es igual a 0.6745 veces la desviación estándar (SD) y 0.8453 veces la desviación media (AD)) (2005).

La anterior es la razón principal por la cual es necesario efectuar una prueba de normalidad, como lo pueden ser la prueba de Shapiro-Wilk (Shapiro & Wilk, 1965) o la prueba de Kolmogorov-Smirnov (Shiryayev, 1992).

3.3.1 Detección de anomalías mediante MAD

Una vez que se cumple la prueba de la hipótesis, es decir, se tiene la certeza de que se está ante una variable con distribución gaussiana, se procede a calcular la mediana de nuestra muestra, ya contando con esa información, se puede calcular MAD, esta es la mediana de las diferencias de los datos respecto a la mediana de la muestra.

Ya teniendo MAD, se calcula el $z - score\ modificado$ para cada uno de los datos. Usualmente, si el valor absoluto de algún $z - score\ modificado$ sobrepasa a 3.5, el respectivo dato es catalogado como anomalía (Kannan, Manoj, & Arumugam, 2015).

3.3.2 Ejemplo de detección de anomalías mediante MAD

El presente ejemplo es demostrativo, todo el código utilizado se encuentra en la siguiente liga:

https://github.com/amigo20th/Tesis_Deteccion_Anomalias/blob/master/DA_MAD_z_modificado.ipynb[7]

Se ocupará el mismo conjunto de datos que se utilizó en el apartado **Prueba de Grubbs**. Recordando, estos datos hacen referencia a una muestra de moscas domésticas a las cuales se les midió sus alas. Dicho conjunto de datos pertenece al área de la biometría.

Haciendo un análisis exploratorio, generando el histograma, se puede observar que existe una distribución normal casi perfecta (Figura 1), entonces se deduce que no hay anomalías en el conjunto de datos original, por lo que se insertará un dato que jugará el rol de anomalía en nuestro conjunto de datos, de esa manera crearemos un ejemplo de sintético para lograr aplicar MAD satisfactoriamente.

Una vez insertado el dato artificialmente, se puede ver en el histograma una pequeña perturbación alrededor del eje x cuando $x = 62$ (Figura 2). Entonces, ya tenemos los elementos necesarios para poder aplicar MAD y detectar las anomalías.

Ya teniendo el conjunto de datos, es necesario aplicar la prueba de Shapiro-Wilk para determinar si el conjunto de datos cuenta con una distribución normal. Aunque

[7] Para consultar los detalles sobre los resultados aquí mostrados, favor de ingresar y descargar la libreta de Jupyter, siéntase libre de tomar el código y usarlo libremente en su proyecto.

se puede observar a simple vista, mediante el uso del histograma, que estamos ante una distribución normal, la rigurosidad de un análisis de datos que esté correctamente desarrollado, nos pide comprobarlo por medio de la prueba anterior mencionada.

Una vez que hemos comprobado la hipótesis de normalidad, se procede a calcular MAD mediante la Ecuación 6, quedando de la siguiente manera

$$MAD = np.median(np.abs(x\_anom - np.median(x\_anom)))$$

Donde x_anom es nuestra variable con una anomalía artificialmente anexada.

La sentencia en *Python* anterior nos arroja el resultado de que $MAD = 3$, entonces, ya teniendo el valor de MAD, se procede a calcular el $z - score\ modificado$ a todos nuestros elementos de nuestra variable. Lo anterior se logra aplicando la Ecuación 7 en *Python* de la siguiente manera.

$$modified\_z\_score = (0.6745 * (x\_anom - np.median(x\_anom))) / MAD$$

Ahora tenemos una cadena de $z - score\ modificado$ que hace referencia a cada respectivo dato en nuestro conjunto de datos. Se grafica un diagrama de dispersión con $z - score\ modificado$, utilizamos como abscisa un número consecutivo que cuenta desde uno hasta la cantidad de datos que tiene nuestra variable. La gráfica de dispersión queda de la siguiente manera (Figura 7).

Referente a la Figura 7, la línea verde es el promedio de $z - score\ modificado$, la línea naranja es nuestro umbral que determinará si un dato es anómalo o, de lo contrario, es un dato inherente a la distribución. Se puede observar un dato destacado en color rojo, esa es nuestra anomalía.

El dato rojo cuenta con un ID = 100, lo último comprueba que es nuestro dato insertado artificialmente para que jugara el papel de anomalía.

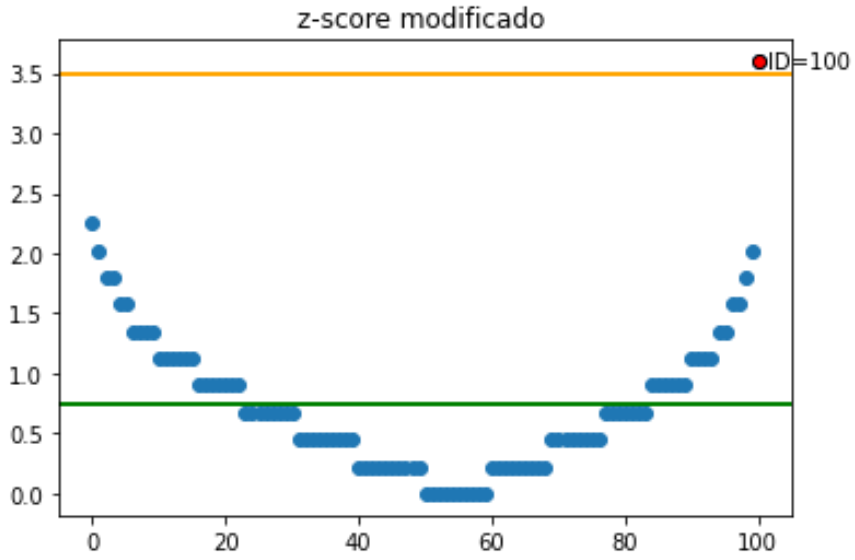

Figura 7. Diagrama de dispersión de $z - score\ modificado$.

3.3.3 Complejidad computacional de MAD

MAD con $z - score\ modificado$ no son muy exigentes computacionalmente hablando, repasando los pasos a seguir:

1) Se ordena la variable, un ordenamiento por *Merge Sort* tiene $O(n\ log\ n)$ (Davidson, Tarjan, Garland, & Owens, 2012).
2) Se calcula MAD, gracias a Ecuación 6 se observa que tiene $O(n)$, ya que son necesarios 2 recorridos a nuestra variable para el cálculo de MAD.
3) Se calcula el $z - score\ modificado$, semejante al caso de MAD es de carácter $O(n)$, ya que si se guarda el resultado de x^* desde el cálculo anterior de MAD, se recorre la variable solo una vez.

Por lo tanto, al ver el método en su totalidad no crece más rápido que una cota asintótica de $O(n\ log\ n)$, podemos concluir que la complejidad computacional de MAD es de $O(n\ log\ n)$.

3.3.4 Conclusiones

Como punto a favor, MAD es un método ligeramente más robusto que la prueba de Grubbs, esto porque toma como punto de referencia la mediana en lugar de la media, que como recordamos, es más estable a la hora de tratar con anomalías.

Otro punto positivo, cuenta con una rapidez al momento de calcular si un conjunto de datos tiene anomalías o no, puesto que cuenta con un orden de complejidad relativamente bajo, esto hace que en los ejercicios de detección de anomalías en tiempo real o detección de novedades sea muy eficaz respecto al tiempo.

Para un ejercicio de detección de novedades, solo se calcularía el $z-score\ modificado$ del nuevo dato y se comprobaría que este no rebase el umbral anteriormente estipulado.

Como punto en contra, es su propia dependencia a la prueba de hipótesis de normalidad, esto limita su aplicación en distintos casos en los que la muestra no cuenta con una distribución gaussiana.

Otro punto negativo, es ser perteneciente a la estadística univariada, como sabemos, no tiene en cuenta todos los atributos de un vector que exista en más de una dimensión de una manera integral.

En el siguiente capítulo conoceremos algoritmos que no se ven limitados por la distribución de los datos y que, además, son aplicables a conjuntos de datos que existen en varias dimensiones.

CAPÍTULO 4

Detección de Anomalías con Enfoque por Densidad

Este enfoque de DA se basa en encontrar vectores en el conjunto de datos que se encuentren a una distancia atípica de los demás. La distancia para determinar si un dato es anómalo será relativa a la densidad que se observe en el conjunto de datos.

Al calcular la distancia entre los datos, se tienen en cuenta todos los atributos ya que los vectores son tomados de manera atómica[8], situación que no ocurre en un enfoque estadístico.

Teniendo en cuenta la totalidad de dimensiones de un vector para determinar si es anómalo podemos evitar la problemática de no tomar la correlación entre variables que se tenía en el enfoque estadístico.

Por ejemplo, sea un vector de dimensión n diferente en una dimensión i, y que este difiera en una cantidad pequeña al resto de datos en el atributo, es probable que realizando una DA por enfoque estadístico este vector sea catalogado como anomalía. Al contrario, realizar un algoritmo de DA que toma al vector de manera atómica, el vector probablemente no sea etiquetado como anomalía, pues difiere por un solo atributo y de una manera no significante.

A continuación, se mostrarán dos algoritmos con enfoque por densidad. El primero llamado *Local Outlier Factor*, el cual toma como idea principal el algoritmo de k vecinos más cercanos (k-NN) (Guo, Wang, Bell, Bi, & Greer, 2003). El segundo es un algoritmo llamado Bosques de Aislamiento y la idea básica consiste en ir

[8] El término atómico es usado para determinar un objeto indivisible respecto a sus dimensiones.

partiendo el espacio de los datos hasta, como su nombre lo indica, aislar un solo dato, por último, dependiendo del número de cortes que sean necesarios para dicho aislamiento, se determinará si el dato aislado es anómalo o no lo es.

4.1 *Local Outlier Factor* (LOF)

El algoritmo *Local Outlier Factor* fue presentado por primera vez por Breunig, Kriegel, Ng y Sander en el año 2000, esto en la publicación *LOF: Identifying Density-Based Local Outliers* (Breunig, Kriegel, Ng, & Sander, 2000).

Es un algoritmo relativamente nuevo, pero que ha tenido grandes usos en el campo de la seguridad, en el comercio electrónico, identificación de comportamiento cibercriminal, entre otros campos.

LOF tiene la característica de dar como resultado un factor de anormalidad, es decir que en LOF no existe la dicotomía de ser o no anomalía. LOF es un número continuo, de este número se puede apoyar el experto para determinar un umbral de anomalía, y así determinar qué datos son valores atípicos.

Como se comentó, LOF toma el algoritmo de k vecinos más cercanos como base para su construcción. K-nn es un algoritmo de clasificación supervisada que pertenece al campo del ML, en este algoritmo los datos de prueba serán catalogados con la misma clase que la mayoría de sus k vecinos más cercanos del conjunto de entrenamiento.

El concepto clave de LOF, un dato cuenta con sus k vecinos más cercanos, este a su vez no es vecino más cercano de sus k vecinos, entonces dicho dato será candidato para ser dato anómalo. Una representación de esta idea se puede observar en la Figura 8, en esta imagen se toma $k = 2$.

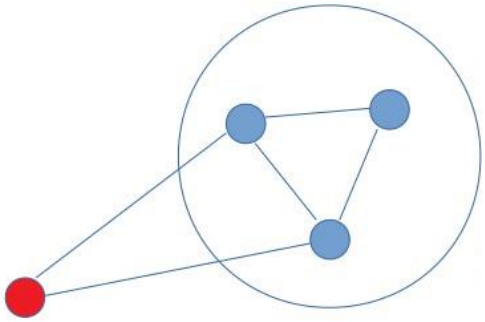

Figura 8. El punto rojo no es vecino más cercano de sus 2 vecinos.

Resulta interesante el término *Local* en el nombre de LOF, esto especifica la existencia de anomalías globales y locales. La diferencia de estos tipos de anomalías nos las da Filzmoser (2014) “While global outliers are data points that are located away from the bulk of the data in the multivariate space, local outliers differ in their non-spatial attributes from observations within a locally restricted neighbourhood.” (Mientras que los valores atípicos globales son puntos de datos que están ubicados lejos de la mayor parte de los datos en el espacio multivariando, los valores atípicos locales difieren en sus atributos no espaciales de las observaciones dentro de un vecindario restringido localmente).

Los algoritmos y métodos mostrados en la presente tesis se enfocan en la detección de anomalías globales. El único algoritmo que veremos con la capacidad de señalar valores atípicos locales es LOF, lo anterior no significa que LOF no logre detectar anomalías globales, al contrario, LOF es un algoritmo robusto capaz de realizar una DA de manera global y local.

En la Figura 9 (Breunig, Kriegel, Ng, & Sander, 2000), podemos observar un ejemplo de anomalía global y otro de anomalía local, ambas se encuentran representadas como o_1 y o_2 respectivamente.

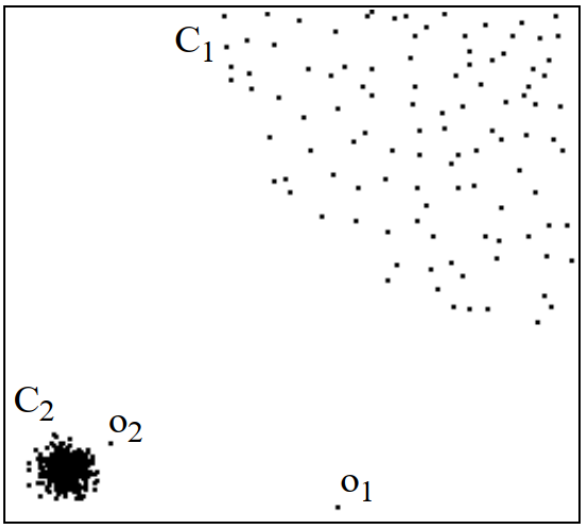

Figura 9. Ejemplo de anomalía global (o_1) y local (o_2) (Breunig, Kriegel, Ng, & Sander, 2000).

Realizando un ejercicio de DA que tome a todo el conjunto de datos como referencia, el algoritmo devolverá como dato atípico a o_1, mientras que o_2 es probable que no sea catalogado de la misma forma, esto es dado a que o_2 no difiere tanto al subgrupo de datos (C_2). Por otro lado, al ejecutar LOF en este conjunto de datos, es probable que o_1 y o_2 sean etiquetados como anomalías, o_1 por ser diferente al conjunto total de los datos y o_2 por ser diferente a sus vecinos más próximos, dichos vecinos se encuentran en C_2.

4.1.1 Ejemplo de detección de anomalías mediante LOF

Se realizará un ejemplo sintético para ser más didáctica la comprensión del algoritmo de LOF, posteriormente se continuará con una ejecución de DA mediante LOF en un conjunto de datos real aplicado en la composición química de cerámica china.

4.1.1.1 Ejemplo sintético realizado con $k = 3$

Se tiene un conjunto de datos C, este cuenta con dos atributos, entonces:

$$C = \{(1,4),\ (1,5),\ (2,4),(2,5),(3,0),(3,1),(7,6)\}$$

La grafica de dispersión resultante es la que muestra la Figura 10.

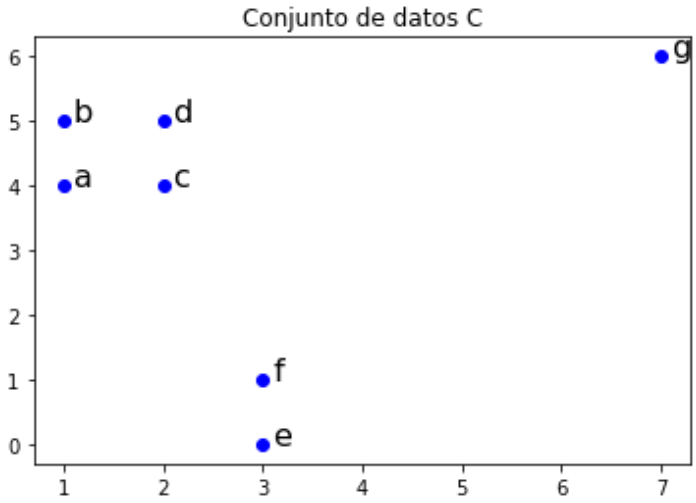

Figura 10. Diagrama de dispersión de C.

Se etiquetaron los puntos para mayor visibilidad y comprensión, el vector a ser el principal candidato para etiquetarse como valor atípico es el punto g.

Como primer paso del algoritmo es necesario crear el conjunto de todas las distancias entre dos puntos de C, es decir:

$$dist\_C = \{\, d(p_1, p_2) \mid p_1, p_2 \in C \ \wedge p_1 \neq p_2 \}$$

El conjunto de distancias (Tabla 1) nos guiará a identificar los primeros 3 vecinos más cercanos respecto a todos los puntos de C.

Ya contando con todas las distancias entre los puntos de C, ahora se procede a elegir las 3 distancias más pequeñas que incluyan al punto g (Tabla 2), solo se calculará el factor de anormalidad del vector g, ya que se sabe de antemano que es una anomalía, esto al ser un ejemplo demostrativo. En un ejercicio real se tendrían que buscar los 3 puntos más cercanos para cada punto de C, es necesario tener esa información para realizar los cálculos siguientes.

ab	1
ac	1
ad	1.414
ae	4.472
af	3.606
ag	6.325
bc	1.414
bd	1
be	5.385
bf	4.472
bg	6.083
cd	1
ce	4.123
cf	3.162
cg	5.385
de	5.099
df	4.123
dg	5.099
ef	1
eg	7.211
fg	6.403

Tabla 1. Distancias entre los puntos de C.

Rectas	Distancias	Vecinos de g	Vecinos de d	Vecinos de c	Vecinos de b	Vecinos de a
ab	1				1er	1er
ac	1			1er		2do
ad	1.414		3er			3er
ae	4.472					
af	3.606					
ag	6.325					
bc	1.414			3er	3er	
bd	1		1er		2do	
be	5.385					
bf	4.472					
bg	6.083	3er				
cd	1		2do	2do		
ce	4.123					
cf	3.162					
cg	5.385	2do				
de	5.099					
df	4.123					
dg	5.099	1er				
ef	1					
eg	7.211					
fg	6.403					

Tabla 2. 3-Vecinos significativos.

Entonces, los vecinos más cercanos de g son d, c y b respectivamente. Estos vectores son los relevantes para este ejemplo ya que desde un principio se definió $k = 3$.

Ahora definimos la Distancia de Accesibilidad (RD):

Definición Sea k un número natural. La *Distancia de Accesibilidad* (RD)

de un objeto p con respecto al objeto o está definido como

$$reach - dist_k(p, o) = max\{ k - distancia(o), d(p, o) \} \ldots\ldots\ldots \quad (8)$$

(Breunig, Kriegel, Ng, & Sander, 2000).

RD es una métrica relativa a la distancia, ya que el enfoque de LOF es relativa a la densidad de los datos en una región local del espacio multivariante, es necesario realizar una conversión de RD, por lo tanto, Breunig, Kriegel, Ng, y Sander (2000) nos proponen lo siguiente:

Definición La Densidad de Accesibilidad Local (LRD) de p está definida

Como:

$$LRD_k(p) = 1/\left[\frac{\sum_{o \in N_k(p)} reach-dist_k(p,o)}{|N_k(p)|}\right] \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

Teniendo que

$N_k(p)$ es el conjunto de vectores que son k-vecinos más cercanos de p.

$|N_k(p)|$ es el número de elementos que contiene ese conjunto.

Se puede observar, el divisor de la Ecuación 9 es el promedio de las RD de p respecto a sus k vecinos más cercanos. Por lo tanto, LRD, que ya es una métrica de distribución, es el inverso de dicho promedio de las RD de p.

Continuando con nuestro ejemplo, procedemos a calcular las RD del vector g para posteriormente deducir LRD de g.

Tomando la Ecuación 8, tenemos las distintas RD:

$$RD_k(g,d) = \max\{\ 3 - distancia(d), d(g,d)\} = \max\{1.414, 5.099\} = 5.099$$

$$RD_k(g,c) = \max\{3 - distancia(c), d(g,c)\} = \max\{1.414, 5.385\ \} = 5.385$$

$$RD_k(g,b) = \max\{\ 3 - distancia(b), d(g,b)\} = \max\{1.414, 6.083\} = 6.083$$

Calculamos LRD, con la Ecuación 9, para el punto g con 3 vecinos más cercanos.

$$LRD_3(g) = 1/\left[\frac{5.099 + 5.385 + 6.083}{3}\right] = 1/\left[\frac{16.567}{3}\right] = \frac{1}{5.522} = 0.181$$

Análogamente calcularemos los LRD de los 3 vecinos más cercanos de g ya que es necesario para realizar LOF de g, para lo anterior es primordial calcular RD de los 3 vecinos más cercanos de g, que son d, c y b.Nos apoyaremos de la Tabla 2.

$$RD_k(d,b) = \max\{\ 3 - distancia(b), d(d,b)\} = \max\{1.414, 1\} = 1.414$$

$$RD_k(d,c\ ) = \max\{3 - distancia(c), d(d,c)\} = \max\{1.414, 1\ \} = 1.414$$

$$RD_k(d,a) = \max\{\ 3 - distancia(a), d(d,a)\} = \max\{1.414, 1.414\} = 1.414$$

$$LRD_3(d) = 1/\left[\frac{1.414 + 1.414 + 1.414}{3}\right] = \frac{1}{1.414} = 0.707$$

$$RD_k(c,a) = \max\{\ 3 - distancia(a), d(c,a)\} = \max\{1.414, 1\} = 1.414$$

$$RD_k(c,d\ ) = \max\{3 - distancia(d), d(c,d)\} = \max\{1.414, 1\ \} = 1.414$$

$$RD_k(c,b) = \max\{\ 3 - distancia(b), d(c,b)\} = \max\{1.414, 1\} = 1.414$$

$$LRD_3(c) = 1/\left[\frac{1.414 + 1.414 + 1.414}{3}\right] = \frac{1}{1.414} = 0.707$$

$$RD_k(b,a) = \max\{\ 3 - distancia(a), d(b,a)\} = \max\{1.414, 1\} = 1.414$$

$$RD_k(b,d\ ) = \max\{3 - distancia(d), d(b,d)\} = \max\{1.414, 1\ \} = 1.414$$

$$RD_k(b,c) = \max\{\ 3 - distancia(c), d(b,c)\} = \max\{1.414, 1.414\} = 1.414$$

$$LRD_3(b) = 1/\left[\frac{1.414 + 1.414 + 1.414}{3}\right] = \frac{1}{1.414} = 0.707$$

Ahora, tenemos los LRD de todos los puntos necesarios para el cálculo de LOF, Breunig, Kriegel, Ng, y Sander (2000) lo definen como:

Definición El *Local Outlier Factor* de p está definido como

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{LRD_k(o)}{LRD_k(p)}}{|N_k(p)|} \text{.....(10)}$$

Haciendo un poco de álgebra sobre la Ecuación 10, podemos llegar a la conclusión de que LOF es el promedio de LRD de sus k vecinos más cercanos entre LRD de p, para este ejemplo, es decir:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{LRD_k(o)}{|N_k(p)|}}{LRD_k(p)}$$

Continuando con nuestro ejemplo, calculamos LOF de g, quedando de la siguiente manera:

$$LOF_3(g) = \frac{\frac{LRD_3(d) + LRD_3(c) + LRD_3(b)}{3}}{LRD_3(g)} = \frac{0.707}{0.181} = 3.91$$

Por lo tanto, el factor de anormalidad o LOF para el punto g es 3.91, lo que faltaría realizar es calcular LOF para todos los vectores del conjunto de datos, eso nos dará

la idea para determinar un umbral, el cual servirá para decidir qué datos son atípicos y cuáles no.

4.1.1.2 Ejecución de DA mediante LOF con $k = 3$

Ahora se continuará con realizar una ejecución del algoritmo de LOF aprovechando la biblioteca *sklearn* de *Python*. Para este ejemplo se utilizará un conjunto de datos que cuenta con información de la composición química de diferentes tipos de cerámica.

El conjunto de datos lo podemos encontrar en la siguiente liga https://archive.ics.uci.edu/ml/datasets/Chemical+Composition+of+Ceramic+Samples en este conjunto podemos observar que sus atributos son compuestos químicos, los cuales fueron resultado de una medición mediante Fluorescencia de Rayos X de Dispersión de Energía (EDXRF por sus siglas en inglés), este análisis fue hecho tanto para el cuerpo de cada cerámica como para su cubierta o glaseado. El conjunto de datos cuenta con 17 atributos referentes a la composición química, una dimensión para el nombre de la cerámica y otra columna para diferenciar si la composición pertenece al cuerpo o al glaseado de la cerámica. Tiene 88 instancias este conjunto de datos, siendo 44 vectores para la química del cuerpo de la cerámica y 44 para el glaseado de esas mismas muestras.

La biblioteca *sklearn* es ampliamente conocida en el campo del ML, eso es por su enorme variabilidad de algoritmos y eficientes ejecuciones de los mismos, en este caso se utilizará la biblioteca *sklearn.neighbors* la cual contiene el algoritmo de LOF.

Todo el código generado para realizar este ejemplo se puede encontrar en la liga

https://github.com/amigo20th/Tesis_Deteccion_Anomalias/blob/master/DA_LOF_Celadon.ipynb[9]

Un paso importante antes de la aplicación de LOF, exclusivo para este conjunto de datos en particular, es separar las composiciones químicas referentes al cuerpo de las del tipo glaseado, pese a que pertenecen a las mismas muestras, es necesario

[9] Si se requiere es posible copiar o descargar el código para modificarlo al gusto.

diferenciarlas ya que pueden generar problemas al ser dos tipos de composiciones diferentes.

Las cerámicas son provenientes de dos regiones de China, siendo de *Longquan* o de *Jindgezhen*, esta diferenciación la hace el conjunto de datos en la nomenclatura que tienen en el nombre cada una de las cerámicas.

Otro aspecto que se tomó en cuenta, para facilitar la visualización de este ejemplo, ya que para ver un objeto en 17 dimensiones es realmente complejo, nos podríamos apoyar en algoritmos para reducir la dimensionalidad del conjunto de muestra, estos algoritmos podrían ser PCA y SOM. Los investigadores He, Zhang y Zhang en su publicación "*Data-driven research on chemical features of Jingdezhen and Longquan celadon by energy dispersive X-ray fluorescence*" realizaron la labor de ordenar por importancia los químicos de cada tipo de cerámica (He, Zhang, & Zhang, 2015), por lo tanto, tomaremos la recomendación de los expertos y tendremos en cuenta solo los cuatro compuestos químicos más importantes, tanto para el cuerpo como para el glaseado de las cerámicas.

Para el cuerpo de la cerámica, se tomaron los cuatro compuestos ZrO_2, Fe_2O_3, CaO y Y_2O_3, su diagrama de dispersión se puede observar en la Figura 11. Para el glaseado de cada muestra se tomaron SrO, ZrO_2, Rb_2O y Na_2O, se puede observar la distribución de los datos en la Figura 12.

Los diagramas de dispersión que representan al cuerpo y el glaseado de las cerámicas (Figura 11 y Figura 12) tienen en su diagonal los histogramas pertenecientes a cada uno de los atributos del muestreo, ningún histograma muestra un comportamiento gaussiano, por ende, no sería posible la aplicación de métodos de DA que existan en la estadística paramétrica.

Los diagramas de dispersión de la Figura 11 y la Figura 12, pertenecientes al conjunto muestral del cuerpo y glaceado de las cerámicas, cuentan con diferentes rangos en sus valores, por esa razón es necesario normalizar los atributos de ambas muestras.

Entonces, ya es posible aplicar LOF en los dos conjuntos de datos, pero surge una pregunta, ¿para qué valor de k es correcto ejecutar LOF?, dicha pregunta no tiene una respuesta específica, lo que sugieren los autores Breunig, Kriegel, Ng y Sander es realizar varias ejecuciones de LOF para distintos valores de k, dado que cada conjunto de datos tiene sus propias distribuciones y características.

Realizar distintas ejecuciones de LOF para distintos valores de k resulta ser una labor costosa, computacionalmente hablando, debido a que para valores diferentes de k, es necesario calcular distintos promedios para cada uno de los vecinos de cada punto. Aunado a que es posible que el conjunto muestral tenga miles, cientos de miles o millones de datos.

En esta ejecución aplicaremos LOF con $k = 3$, $k = 5$ y $k = 10$, en ambos conjuntos de datos, eso nos dará una idea del comportamiento que puede tener este algoritmo conforme va aumentando el valor de k.

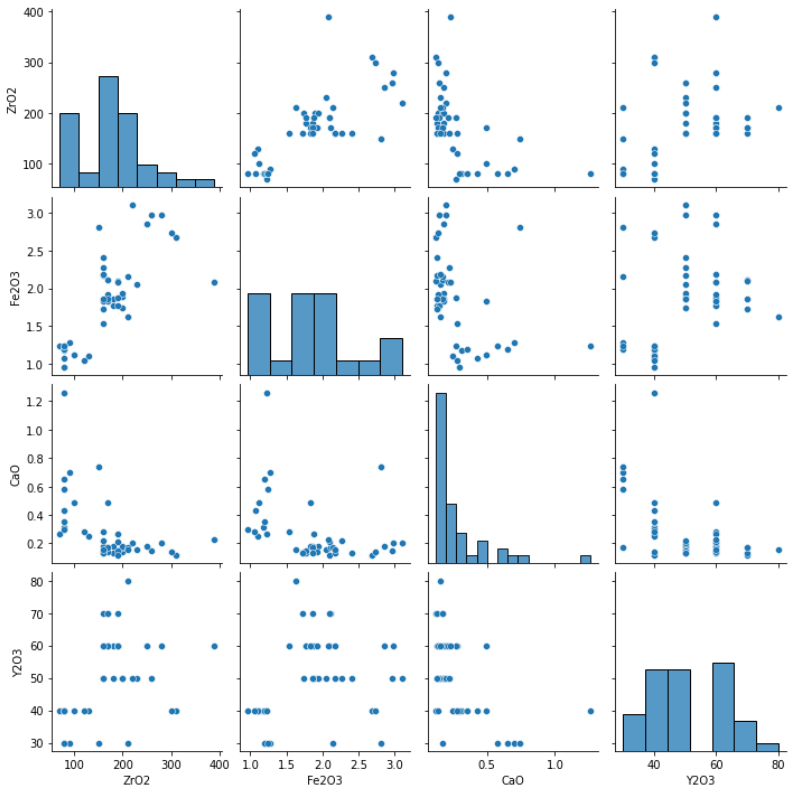

Figura 11. Diagrama de dispersión para el cuerpo de las cerámicas

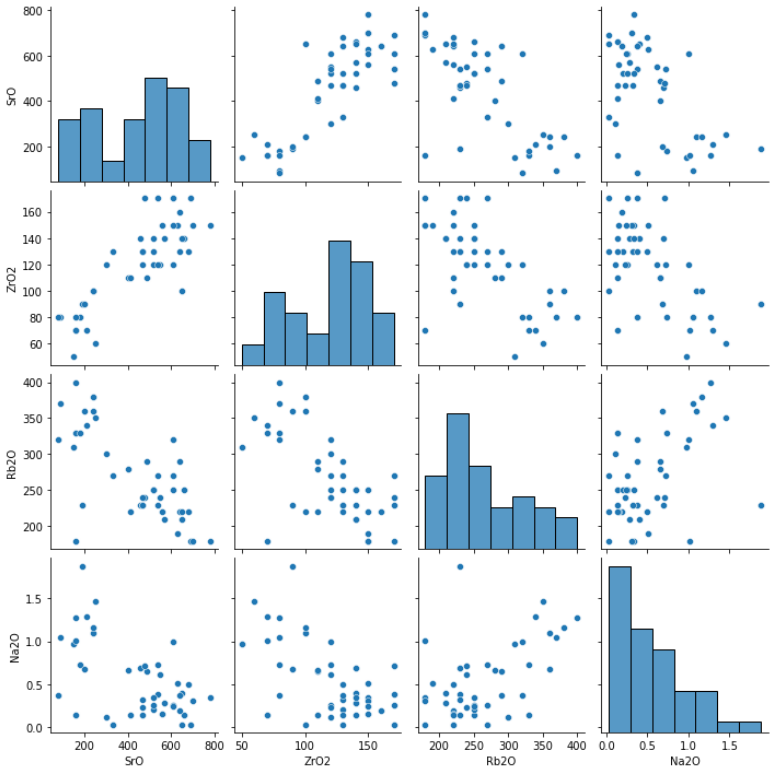

Figura 12. Diagrama de dispersión para el glaseado de las cerámicas

Gráficamente es posible observar las anomalías detectadas en el cuerpo de las 44 diferentes cerámicas, esto es en las gráficas Figura 13, Figura 14 y Figura 15, como se comentó se aplicó LOF con $k = 3, k = 5$ y $k = 10$, respectivamente. Los valores atípicos registraron de la siguiente manera:

k=3	k=5	k=10
FLQ-9-b	FLQ-5-b	FLQ-9-b
FLQ-12-b	FLQ-7-b	FLQ-12-b
DY-BS-7-b	FLQ-9-b	DY-NS-3-b
DY-NS-3-b	FLQ-12-b	DY-Y-3-b
DY-NS-6-b	DY-NS-3-b	DY-Y-4-b
DY-NS-7-b	DY-NS-7-b	DY-Y-6-b
DY-Y-6-b	DY-Y-6-b	DY-QC-3-b
DY-QC-3-b	DY-QC-3-b	

Tabla 3. Anomalías detectadas por LOF en el cuerpo de las cerámicas.

Existen cinco cuerpos de cerámicas detectados como anomalías en los tres ejemplos, estas están resaltadas por color rojo. Otra observación es que al aumentar k va aumentando la localidad para realizar la DA, esto hace que aparezcan menos anomalías. Es posible tomar en cuenta las distintas ejecuciones para proponer como anomalías a las cinco cerámicas que aparecen en los tres diferentes valores de k, pero la última palabra la tendrán los expertos en el tema de investigación.

Ahora se continúa con el conjunto de datos alusivo a el glaseado de las cerámicas, las anomalías detectadas se pueden observar gráficamente en las Figuras 16, 17 y 18, las cuales pertenecen a LOF cuando $k = 3, k = 5$ y $k = 10$, respectivamente.

k=3	*k=5*	*k=10*
FLQ-8-g	FLQ-8-g	FLQ-8-g
FLQ-9-g	FLQ-9-g	DY-Y-5-g
DY-NS-5-g	DY-NS-5-g	
DY-Y-2-g	DY-Y-2-g	
DY-M-1-g	DY-M-1-g	
DY-QC-4-g		

Tabla 4. Anomalías detectadas por LOF en el glaseado de las cerámicas.

La Tabla 4 es una mejor muestra de cómo van disminuyendo las anomalías conforme va aumentando k. Un caso muy particular es el caso de la cerámica DY-Y-5-g, ya que esta aparece como atípica en $k = 10$ pero no aparece en los valores de k anteriores, en la Figura 18 se puede observar que DY-Y-5-g efectivamente se encuentra un tanto alejado de los demás puntos, pero no tan alejado para ser catalogado como una anomalía respecto a la vecindad con $k = 3$ y $k = 5$.

En lo que respecta al glaseado de las 44 cerámicas, solo una se repite para los tres distintos valores de k, siendo FLQ-8-g. Aquí tenemos un caso en el cual es necesaria una búsqueda exhaustiva para k, visto que hay anomalías que se presentan en $k = 3$ y $k = 5$, pero podemos estar perdiendo atípicos al saltar directamente a $k = 10$.

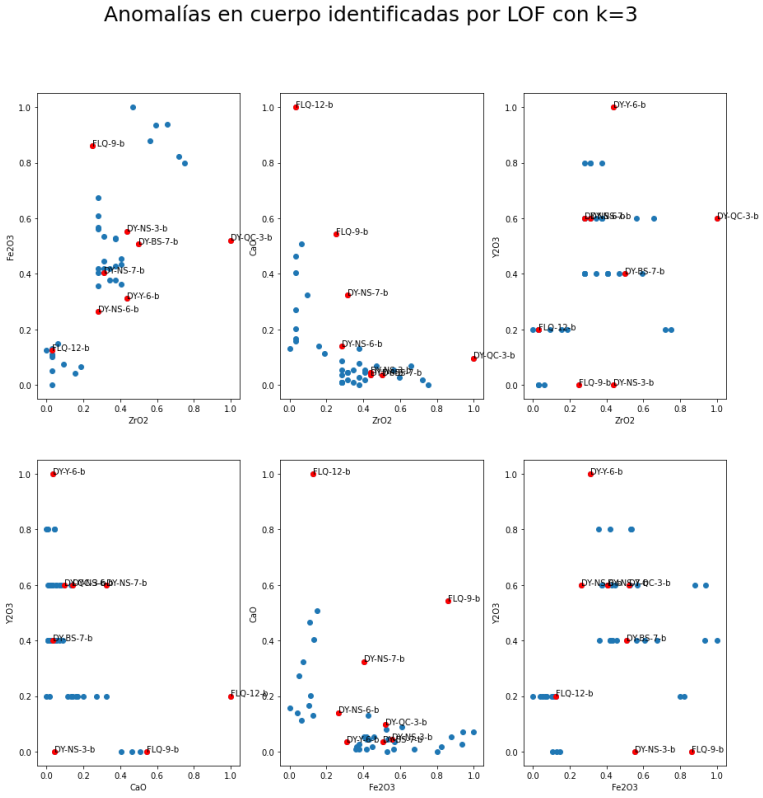

Figura 13. LOF para el cuerpo de la cerámica con $k = 3$.

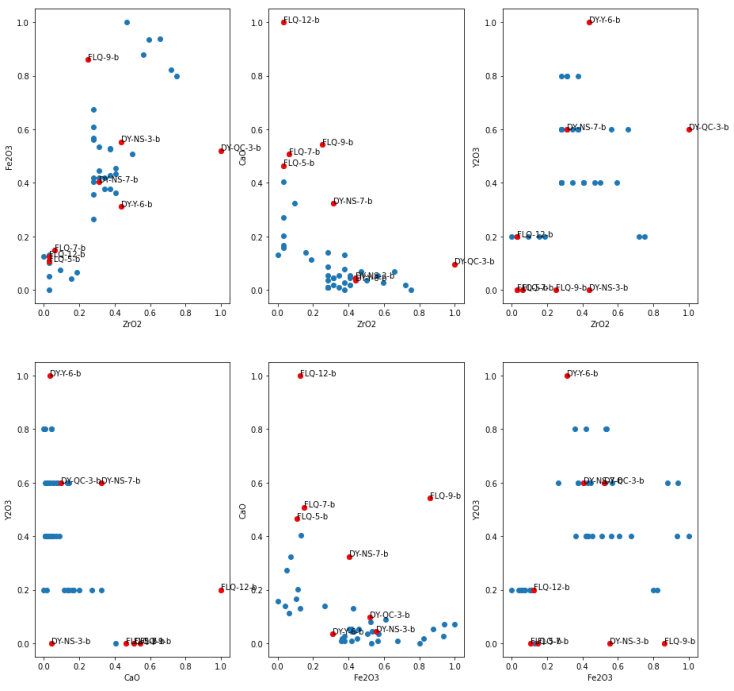

Figura 14. LOF para el cuerpo de la cerámica con $k = 5$.

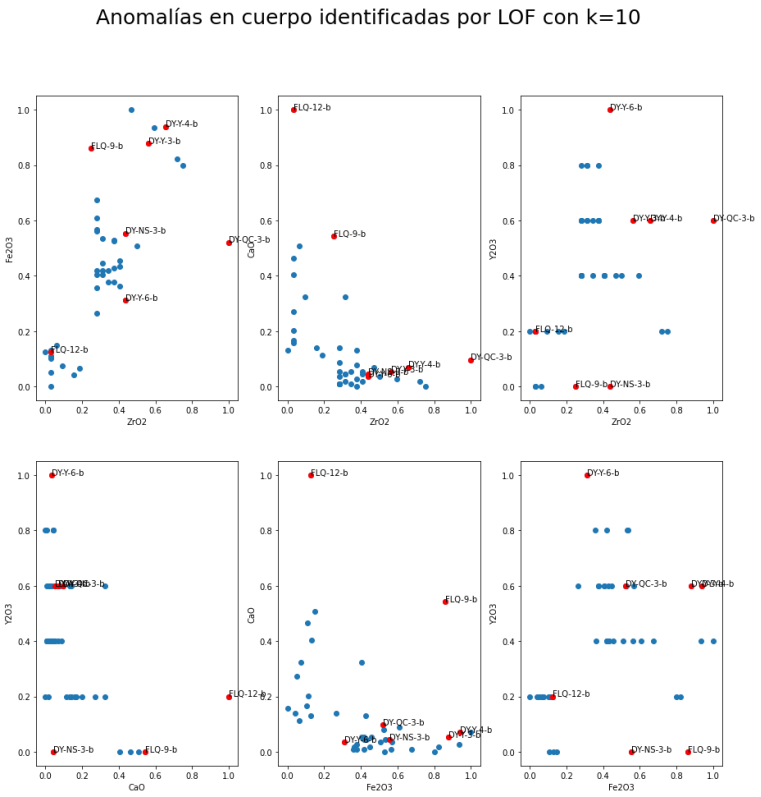

Figura 15. LOF para el cuerpo de la cerámica con $k = 10$.

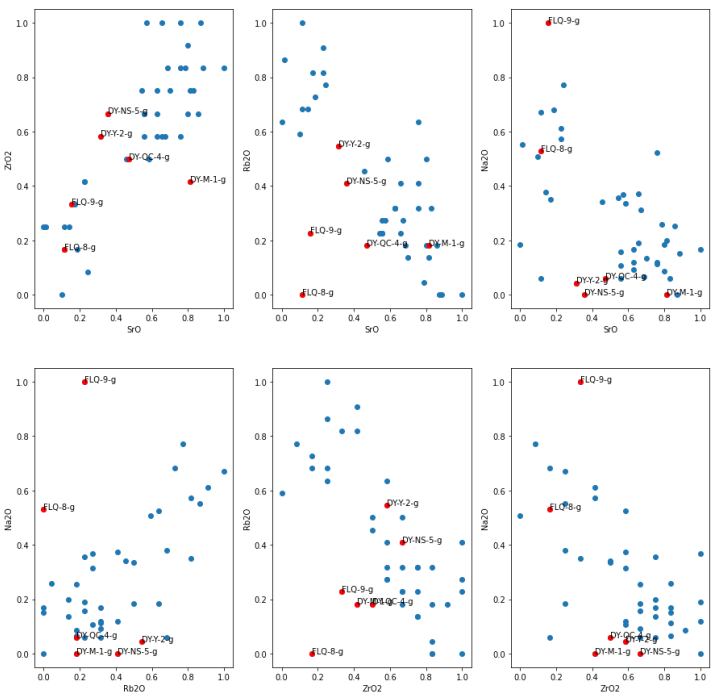

Figura 16. LOF para el glaseado de la cerámica con $k = 3$.

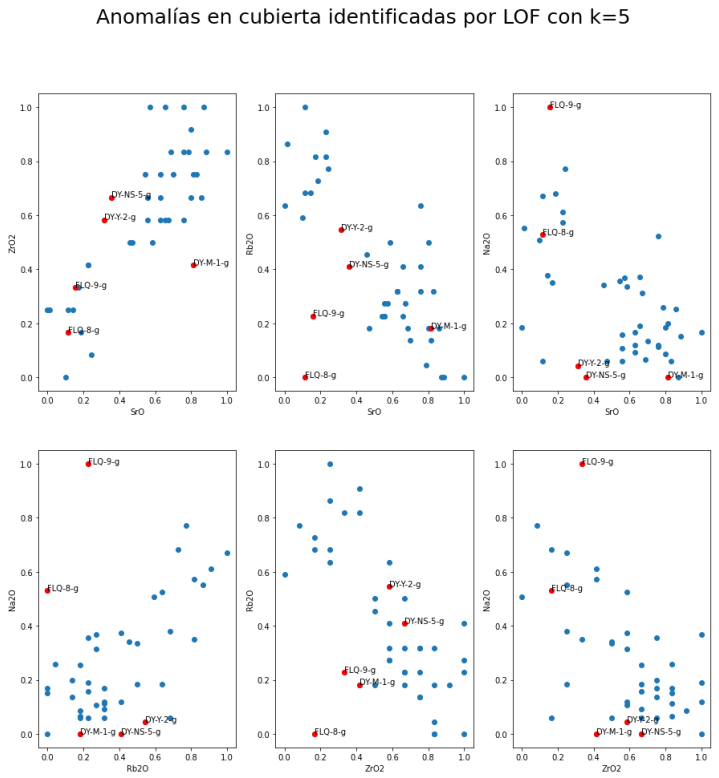

Figura 17. LOF para el glaseado de la cerámica con $k = 5$.

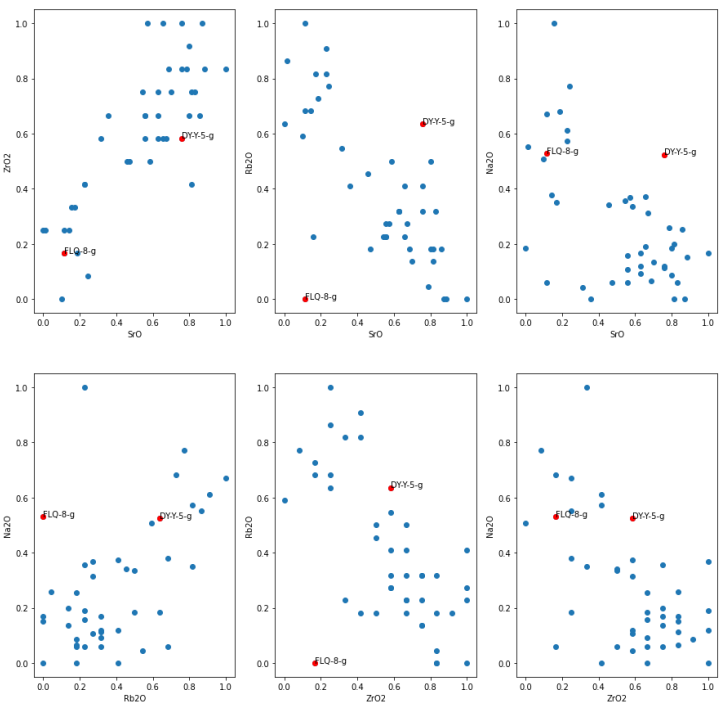

Figura 18. LOF para el glaseado de la cerámica con $k = 10$.

4.1.2 Complejidad computacional de LOF

Breunig, Kriegel, Ng y Sander ya realizaron un completo análisis de la complejidad en su algoritmo (2000). Para ello, separan en dos pasos el algoritmo de LOF para poder calcular efectivamente la complejidad computacional. El primer paso es básicamente la creación de una tabla como la mostrada en la Tabla 2, este paso es el realmente costoso ya que el segundo es solo el cálculo de LOF.

La complejidad computacional normalmente es tomada en el peor de los casos, por esa razón, el primer paso del algoritmo no crece más que la función asintótica de $O(n^2 \log n)$, lo anterior nos indica que estamos ante un algoritmo de orden cuadrático.

El segundo paso es realizar los cálculos de LOF para todos los puntos, pero al ya tener definida la tabla con las distintas distancias y los respectivos vecinos más cercanos, resulta ser una tarea sencilla, el segundo paso es de $O(n)$.

4.1.3 Conclusiones

LOF es un algoritmo robusto que logra efectuar una DA efectiva, ya que tiene la finalidad de detectar los atípicos de manera local y no solo global, por este hecho es uno de los métodos de DA más utilizado hoy en día.

Como punto positivo ya se estipuló su habilidad para encontrar anomalías de manera local, también se añadiría su fácil comprensión, esto dada la idea base de su funcionamiento, es sencilla de entender y de calcular. Otro punto a favor es su capacidad de actuar en el espacio multivariado, esto es de ayuda en épocas actuales, hoy en día los datos nuevos se crean por teras diariamente, esta información debe de tener la capacidad de cruzarse entre sí para realizar análisis más completos.

Como punto negativo es su alta complejidad computacional, definitivamente no es viable para conjuntos de datos grandes, es decir, que tengan bastantes instancias y que existan en un espacio con muchas dimensiones. Existen distintos algoritmos que tratan de utilizar las bondades de LOF y al mismo tiempo aplicar diferentes

estrategias para disminuir su complejidad, analizar esos algoritmos es para un trabajo futuro.

Otro punto negativo es el hiperparámetro k, dado que no existe un método claro para estipularlo correctamente es necesario realizar varias ejecuciones de LOF, lo cual aunado a su alta complejidad no es para nada recomendable, se puede facilitar esto al crear una tabla intermedia (similar a la Tabla 2) con un k muy alto, después utilizar esa tabla para calcular LOF con $k's$ cada vez más pequeñas, lamentablemente eso requeriría programar todo el algoritmo desde cero y no aprovechar las bondades de las librerías aquí utilizadas.

LOF al lograr ejecutarse en el espacio multivariado es suceptible a la maldición de la dimensionalidad[10], es un problema que aqueja al ML en general se puede leer más del asunto en (Hughes, 1968).

4.2 Bosque de Aislamiento (Isolation Forest)

El presente algoritmo fue introducido por primera vez por los investigadores Liu, Ting y Zhou (2008). Bosque de aislamiento es un algoritmo ensamblado[11] basado en árboles, de ahí su nombre de bosque. Se le conoce como algoritmo de ensamblado debido a que genera varias instancias de un mismo método, en este caso, un árbol de aislamiento (*Isolation Tree*). Otro término para tener en cuenta es aislamiento (*isolation*). En el artículo original, los autores nos comentan "*the term isolation means 'separating an instance from the rest of the instances'*" (el término aislamiento significa 'separando una instancia del resto de las instancias'). (Liu, Ting, & Zhou, 2008)

Un árbol de aislamiento surge de una idea sencilla y a la vez efectiva. El objetivo del aislamiento es separar una instancia del resto de los vectores. En este enfoque, las anomalías, a su vez, son instancias que son distintas de las demás instancias

[10] Efecto Hughes o *curse of dimensionality* en inglés.
[11] En ML un algoritmo ensamblado es la mezcla de varios algoritmos o varias instancias de un mismo algoritmo con la finalidad de dar mejores resultados (Dietterich, 2000)

(Barnett & Lewis, 1994), bajo el supuesto de que es más sencillo aislar anomalías que a datos usuales.

El algoritmo de árbol de aislamiento también se basa en la idea de los árboles, estos son estructuras de datos ampliamente estudiadas en el campo de las ciencias de la computación. Los árboles se encuentran presentes en una gran variedad de programas y aplicaciones hoy en día, pues entre otras características, son capaces de realizar búsquedas que dan como resultado una complejidad computacional baja, normalmente de $O(\log n)$.

Otro elemento de las estructuras de árboles ampliamente estudiado, aparte de su eficacia para búsquedas sistemáticas, es el cálculo de la longitud de un árbol. Esta es la característica con mayor relevancia en el algoritmo de árbol de aislamiento para detección de anomalías.

El árbol de aislamiento es un algoritmo recursivo[12], es decir, crea instancias del mismo algoritmo aplicado a un subespacio de datos provenientes del espacio original. La recursividad es la base para la programación dinámica, esta fue introducida por Richard Bellman en su publicación *On the Theory of Dynamic Programming* (1952), en la programación dinámica se ocupan estructuras de datos, en estas se van almacenando los resultados de subinstancias de un algoritmo recursivo, esto hace que la complejidad computacional de dicho algoritmo baje drásticamente, pues no tiene que ejecutar redundancia entre instancias.

En la Figura 19, teniendo como dato anómalo a x_0 se observa que fueron necesarios cuatro cortes en la dimensión mostrada para sesgarlo del resto de los datos; en cambio, para un dato usual del muestreo x_i fueron necesarios once cortes. Sin embargo, los cortes son seleccionados de manera aleatoria, entonces este número de cortes para aislar un dato no es un número predeterminado, ya que dependiendo de la manera en que se vayan conformando los cortes, estos variarán en cantidad.

[12] Son algoritmos que tienen enfoque divide y vencerás (Cormen, Leiserson, Rivest, & Stein, 2009)

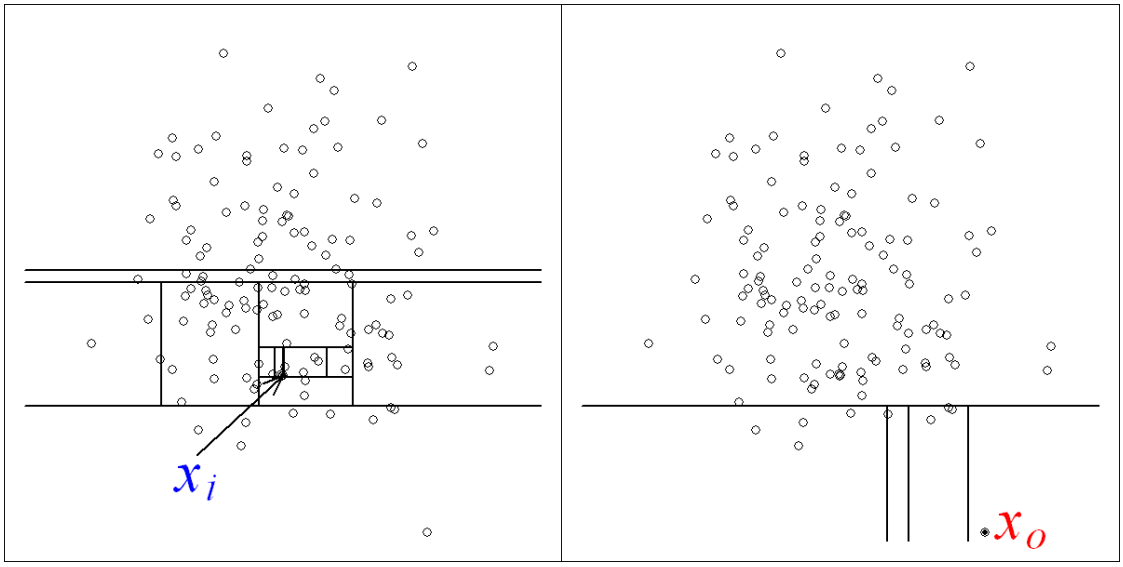

Figura 19. Cortes necesarios para el aislamiento de un dato anómalo (x_0) y un dato no atípico (x_i) (Liu, Ting, & Zhou, 2008).

Debido a la aleatoriedad con la que se crean los cortes, el número de cortes para aislar un vector, en diferentes ejecuciones del algoritmo, no siempre es el mismo.

Para encontrar un número de cortes consistente, se crean varios árboles de aislamiento, estos se encargan de aislar un dato, entonces se toma el promedio de la cantidad de cortes que se obtuvieron en los árboles de aislamiento.

Tomando como ejemplo los puntos x_0 y x_i de la Figura 19, se crean varios árboles de aislamiento para aislar a dichos datos, se van registrando la cantidad de árboles creados, así como los cortes necesarios para separar a x_0 y x_i.

Graficando el número de árboles creados, así como el promedio de cortes, obtenemos la Figura 20, en esta se observa que el promedio converge a un valor dado.

En la Figura 20 el punto x_0 se estabiliza a un promedio de 4.02 cortes aproximadamente y x_i necesita un promedio de cortes de 12.82 aproximadamente.

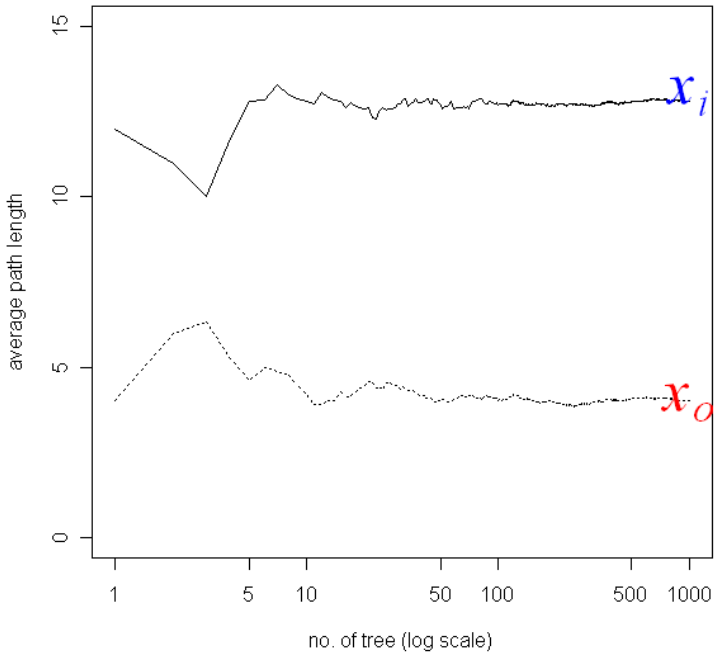

Figura 20. Convergencia en el número de cortes para aislar a x_0 y x_i (Liu, Ting, & Zhou, 2008).

Es necesario crear un mapeo de un árbol de aislamiento a una estructura de árbol computacional, ya que para calcular el número de cortes necesarios para sesgar un dato es más sencillo calcular la longitud de un árbol computacional, que como veremos, dan como resultado el mismo valor.

La construcción del árbol computacional es de la siguiente manera. Cada uno de los cortes en el atributo está representando un nivel en un árbol computacional, como podemos ver en la Figura 21.

El código de colores representa un corte en el atributo y un nuevo nivel creado en el árbol, es decir, primero se eligió el corte hecho por la recta negra, eso parte el espacio en 2 secciones, entonces se crean dos subárboles con raíz negra. Posteriormente, se crea el corte de color amarillo dentro de uno de los dos subespacios creados anteriormente, eso concibe dos subárboles con raíz amarilla.

Se continúan generando cortes, y sus respectivos niveles en el árbol computacional, hasta que uno de los datos es aislado, en la Figura 21 son los puntos que están apartados por la línea de color verde.

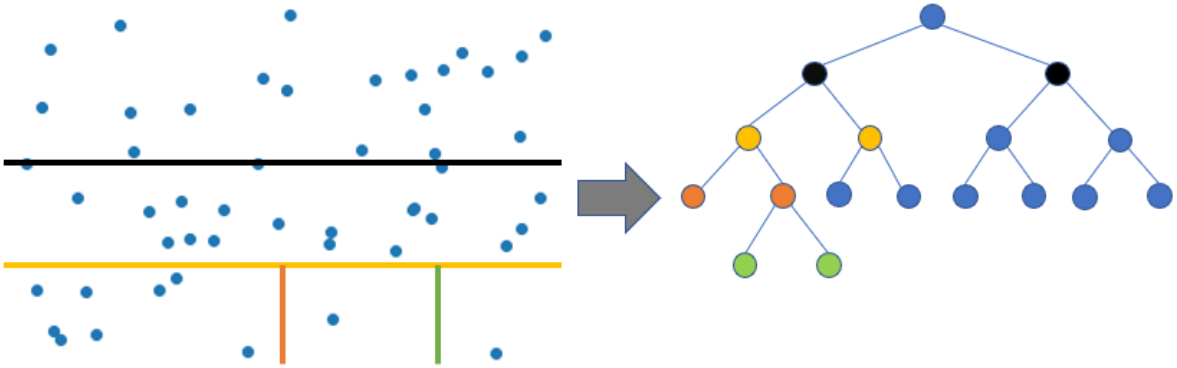

Figura 21. Construcción de un árbol computacional a partir de los cortes aleatorios.

Para mayor comprensión se procede a mostrar el algoritmo árbol de aislamiento (Figura 22), consecuentemente se explicarán sus características y funciones.

Algoritmo 1: $iTree(X, e, l)$

Entrada: X − Datos de entrada, e − longitud del árbol,
l − longitud límite

Salida: Un árbol de aislamiento

Si $e \geq l$ or $|X| \leq 1$ **entonces**
  regresa $exNodo\{longitud \leftarrow |X|\}$
Otro
  Sea Q la lista de atributos de X
  aleatoriamente seleccionar un atributo $q \in Q$ |
  aleatoriamente seleccionar un punto de corte p
  entre los valores *máx* y *mín* del atributo q en X
  $X_i \leftarrow filtrar(X, q < p)$
  $X_d \leftarrow filtrar(X, q \geq p)$
  Regresar $inNodo\{Izquierda \leftarrow iTree(X_i, e+1, l),$
    $Derecha \leftarrow iTree(X_d, e+1, l),$
    $AtribCorte \leftarrow q,$
    $ValorCorte \leftarrow p\}$
Fin Si

Figura 22. Algoritmo árbol de aislamiento (Liu, Ting, & Zhou, 2008).

Como entrada son necesarios los siguientes parámetros:

- X es el conjunto muestral.
- e la longitud del árbol en una instancia dada.
- l la longitud máxima para los árboles de aislamiento.

Al ser un algoritmo recursivo es necesaria una condición de paro. Esta condición está dada por la longitud del árbol, si el árbol ya sobrepasó el límite máximo o si se logró aislar un dato, la última instancia regresa el tamaño final de ese árbol.

Por el contrario, si no se cumple la condición de parada, se tiene el conjunto Q que contiene todos los atributos de la agrupación muestral, se selecciona de manera aleatoria un atributo $q \in Q$, posteriormente se realiza un corte, de manera aleatoria, en dicha dimensión q. Ya teniendo los dos subconjuntos del atributo q, se crean dos instancias del mismo algoritmo árbol de aislamiento, solo que esta vez se aplica iTree a esas dos secciones recien creadas, de esa manera se cumple el enfoque divide y vencerás. Las nuevas instancias creadas también van a aumentar en una unidad la longitud del árbol, ya que ellas suman un nivel más. Los atributos de corte llamado $SplitAtt$ y el atributo de dimensión seleccionada $SplitValue$, son valores que se van heredando en todo el árbol de aislamiento.

Para lograr calcular el promedio de cortes necesarios para sesgar a cada dato, es menester crear múltiples instancias del algoritmo árbol de aislamiento, a ese conjunto de instancias del algoritmo anterior es lo que conocemos como algoritmo bosque de aislamiento. Dicho algoritmo de bosque es de la siguiente forma (Figura 23).

Algoritmo 2: $iForest(X, t, \psi)$

Entrada: X − Datos de entrada, t − número de árboles,
l − tamaño del submuestreo

Salida: Un conjunto de $iTrees$ de tamaño t

1: **Inicializar** $Bosque$
2: establecer la longitud límite $l = funciónTecho(\log_2 \psi)$
3: **Para** $i = 1$ **Hasta** t **Hacer**
4: $X' \leftarrow submuestrear(X, \psi)$
5: $Bosque \leftarrow Bosque \cup iTree(X', 0, l)$
6: **Fin Para**
7: **Regresar** $Bosque$

Figura 23. Algoritmo bosque de aislamiento o *iForest* (Liu, Ting, & Zhou, 2008).

Bosque de aislamiento requiere tres parámetros:

- X es el conjunto muestral.
- t el número de árbol de aislamiento a crear.
- ψ tamaño de las submuestras.

Los autores Liu, Ting y Zhou sugieren que se tome $t = 100$ puesto que para diversos conjuntos de datos es un buen número donde alcanzan a converger los promedios de cortes para aislar, otra recomendación es tomar $\psi = 256$, es decir que un submuestreo de 2^8 es indicado para tener un buen muestreo que represente la totalidad del conjunto de datos, al mismo tiempo manteniendo un rendimiento eficiente.

Bosque de aislamiento es un algoritmo de DA, por lo tanto, los autores brindan un método para calcular el valor de anormalidad que tiene cada dato dentro de una muestra. Este se encuentra dado por:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \quad \text{.....(11)}$$

(Liu, Ting, & Zhou, 2008).

Explicando los elementos de la Ecuación 11, se tiene $c(n)$ la cual es una función proveniente del estudio sistemático, como se había comentado anteriormente, que han hecho los científicos de la computación a los árboles binarios, precisamente nos referimos a los Árboles Binarios de Búsqueda (BST). Dentro de los BST se determinó qué, dentro de un conjunto de datos con n elementos, el promedio de longitud de un BST que realizó una búsqueda insatisfactoria está dado por la siguiente ecuación:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \quad \text{.....(12)}$$

(Liu, Ting, & Zhou, 2008).

Dentro de la Ecuación 12 se observa una función $H(i), i \in n$ esta función es conocida como número armónico y puede ser calculado por:

$$H(i) = \ln(i) + 0.577 \quad \text{.....(13)}$$

(Liu, Ting, & Zhou, 2008).

Retomando la explicación de la Ecuación 11, se observa la función $E(h(x))$, primero, es importante definir $h(x)$, esta es la longitud real de un árbol de aislamiento, y la función E es la esperanza de todas las longitudes de los distintos árboles de aislamiento que contiene el bosque.

Los autores Liu, Ting y Zhou al normalizar las longitudes reales de todo el bosque con las longitudes promedio esperadas en BST, logran dar una idea aproximada de cuánto valor de anormalidad es considerado correcto para catalogar un dato como atípico.

Si un dato tiene un valor de anormalidad $s(x_0, n) < 0.5$ es probable que ese dato no resulte anómalo; si otro dato tiene un valor de $s(x_1, n) > 0.5$ es probable que sea

una anomalía, pero si un dato tiene un valor de $s(x_2, n) \approx 1$ el dato x_2 en definitiva es un atípico.

4.2.1 Detección de anomalías mediante el algoritmo bosque de aislamiento

Para el presente ejercicio se utilizará nuevamente un conjunto de datos tomado de la *UCI Machine Learning Repository*, en particular nos referimos al llamado *seeds* o semillas en su nombre en español. Este conjunto muestral se puede adquirir en la siguiente liga https://archive.ics.uci.edu/ml/datasets/seeds.

El conjunto de datos es creado a partir de tres tipos de semillas de trigo, siendo las especies Kama, Rosa y la Canadiense, se tomaron 70 muestras de cada uno de estos tres tipos de semillas elegidas aleatoriamente. Posteriormente, se tomaron distintas mediciones de las semillas mediante una técnica de rayos x suaves, para saber más sobre el conjunto de datos se recomienda la lectura de la publicación *Complete Gradient Clustering Algorithm for Features Analysis of X-Ray Images* (Charytanowicz, y otros, 2010).

El conjunto de datos original está conformado por 210 instancias y 8 atributos, siendo los siguientes:

- El área de la semilla.
- El perímetro de la semilla.
- La compacidad de la semilla (calculada mediante el área y el perímetro).
- La longitud del grano.
- La anchura del grano.
- El coeficiente de asimetría.
- La longitud de la ranura de la semilla.
- Especie de la semilla

Para el ejemplo se decidió no tener en cuenta el área y el perímetro, ya que ambos atributos presentan una correlación lineal y, además, ya se encuentran

representados por la dimensión de compacidad, lo último ya que la fórmula que tomaron los autores para calcularla fue $C = \frac{4\pi(área)}{(perímetro)^2}$.

Todo el código generado para realizar este ejemplo se puede encontrar en la liga

https://github.com/amigo20th/Tesis_Deteccion_Anomalias/blob/master/DA_iForest.ipynb[13].

Para comenzar con el análisis, es posible observar que dentro del conjunto de datos hay campos faltantes, es posible estimar dichos datos por medio de algoritmos como lo es *Kernel Density Estimation* (*KDE*) (Hastie, Tibshirani, & Friedman, 2001), lamentablemente esos campos no aportarían información de relevancia hacia un análisis de DA. Por ello, el autor de la presente tesis llegó a la conclusión de eliminar todos los vectores con datos faltantes. Las instancias eliminadas son únicamente 10, que muestran algún atributo faltante. Adicionalmente, para el análisis de DA, no importa el desbalanceo de la muestra porque estamos tratando cada semilla de forma individual

Ya teniendo nuestro muestreo de 200 filas, se procede a normalizar los datos para que no haya desproporción en las distintas dimensiones.

El diagrama de dispersión del conjunto de datos se observa en la Figura 23.

Estadísticamente se pueden observar aspectos interesantes en los histogramas que se encuentran en la diagonal de la Figura 24. Primero, es posible identificar una posible distribución normal en el coeficiente de asimetría (faltaría validarlo por medio de una prueba estadística) y una distribución bimodal en la anchura de la ranura de la semilla.

[13] Si se requiere es posible copiar o descargar el código para modificarlo a conveniencia.

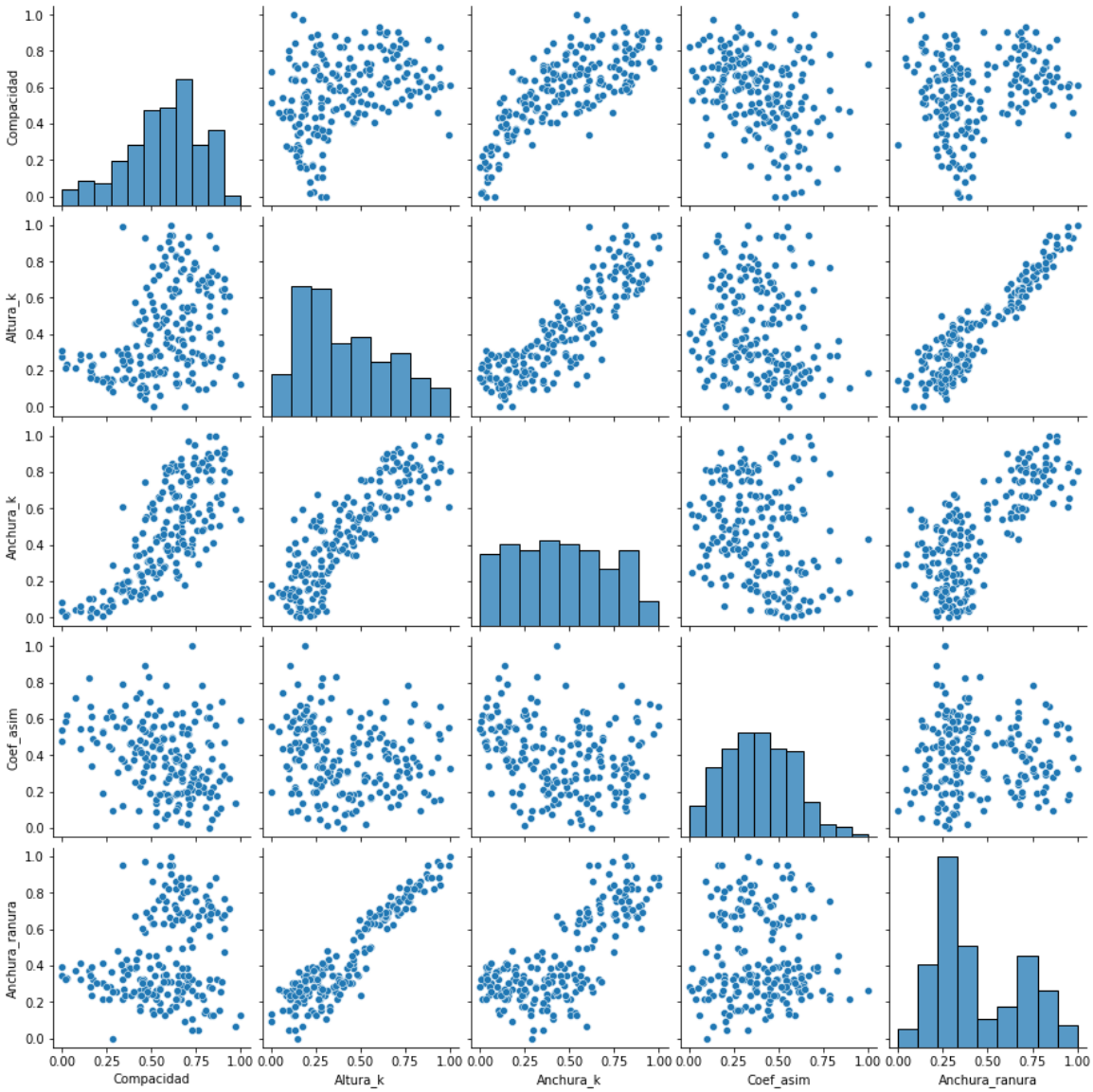

Figura 24. Diagramas de dispersión de las semillas de trigo.

Para la ejecución introduciremos los parámetros recomendados por los autores del algoritmo, es decir, un número de árboles de aislamiento igual a $t = 100$ y el submuestreo igual a $\psi = 256$.

Para la selección de anomalías por medio del bosque de aislamiento *sklearn* da dos diferentes opciones para obtenerlas, una de manera automática, es decir que el propio algoritmo te da las sugerencias de qué vectores son etiquetados como

anomalías y cuáles no. La segunda opción es que el algoritmo calcule el valor de anormalidad $s(x,n)$ para cada una de las instancias del muestreo.

La segunda opción es la que se ocupará para este ejemplo, la razón es que *sklearn* directamente etiqueta como valor atípico a todo dato que tenga $s(x,n) > 0.5$, para este ejemplo no es preciso, por consiguiente, el umbral propuesto para catalogar las anomalías se fijó a $s(x,n) > 0.56$.

Dando por resultado como datos anómalos a las semillas con el identificador 16, 18, 59, 87, 88, 89, 94, 108 y 114, su posición en un diagrama de dispersión es visible en la Figura 24. Los datos atípicos que se encuentran etiquetado por su identificador, se ven en zonas poco densas de la mayoría de las gráficas.

4.2.2 Complejidad computacional de bosque de aislamiento

Los autores Liu Ting y Zhou (2008) dividen el algoritmo de bosque de aislamiento en dos partes, esto para realizar el cálculo de la complejidad computacional de su algoritmo.

La primera parte es la encargada de la creación de todos los árboles de aislamiento que conformarán la totalidad del bosque, la complejidad de este paso no crece mayor a $O(t\psi \log \psi)$, puesto que para la mayoría de conjuntos de datos se toma $t = 100$ y $\psi = 2^{8}$, o por lo menos se deben tomar cantidades parecidas, estamos ante un algoritmo de poca complejidad computacional en esta parte del algoritmo.

La segunda parte del algoritmo es la encargada de evaluar la longitud de todos los árboles de aislamiento, es decir, la encargada de calcular el valor de anormalidad $s(x,n)$ de cada uno de los datos del conjunto muestral, la complejidad de esta parte del algoritmo es $O(nt \log \psi)$. Esta sección del algoritmo también toma poca exigencia computacional.

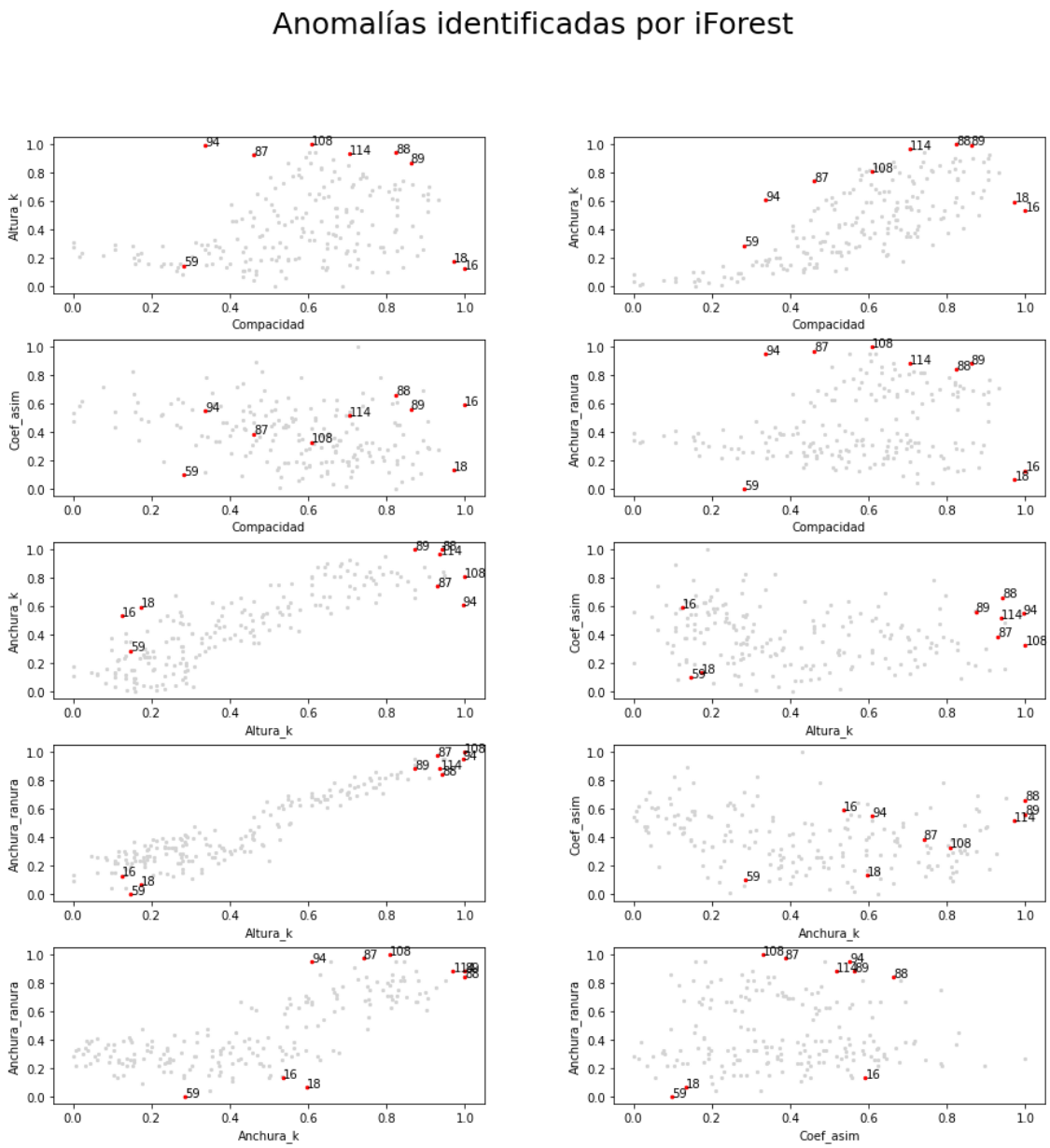

Figura 25. Diagrama de dispersión de los datos propuestos como anomalías.

4.2.3 Conclusiones

El algoritmo de *bosque de aislamiento* parte de una idea sencilla y fácil de comprender. La idea central es medir el nivel de anomalía de un punto como el número de cortes necesarios para aislarlo del resto de los vectores. De igual manera, toma bases fuertes del estudio ya elaborado en Árboles Binarios de Búsqueda (*Binary Search Tree*).

Al basarse en árboles binarios su complejidad se ve reducida de manera logarítmica, lo cual lo hace un algoritmo viable para conjuntos de datos grande con un número de atributos considerable.

Otro punto a favor es que el algoritmo escapa de la maldición de la dimensionalidad (Hughes, 1968), al no tener que tomar en cuenta todas las dimensiones al mismo tiempo hace que no ocurran cosas inesperadas, como pueden suceder en espacios multivariados con un número considerable de dimensiones.

Ya cuenta con una excelente implementación programada, por lo tanto, es fácil de aplicar a distintos conjuntos de datos.

El algoritmo al tomar en cuenta el promedio de longitud de árboles binarios de búsqueda como proporción para calcular la anormalidad, hace que el reconocimiento de dichas anomalías sea más sencillo, pues los autores son capaces de sugerir buenos umbrales para catalogar a los datos atípicos.

Este algoritmo es capaz de calcular, por regiones en el hiperespacio, el valor de anormalidad de posibles datos que puedan existir en dicha zona, eso hace a este algoritmo capaz de funcionar en sistemas de DA en tiempo real. Su verdadera eficacia es usada en sistemas de detección de fraudes en comercio electrónico, lo cual lo hace viable para un sistema de detección de novedades.

Como punto negativo es su dependencia a la aleatoriedad en la creación de las submuestras, en la selección del atributo a afectar, así como en la generación de cortes, puede suceder que el sistema no detecte alguna anomalía por no tener los cortes adecuados, esto provoca que sea necesario ejecutar varias veces el algoritmo para validar el etiquetado de valores atípicos.

Otro punto negativo es la selección unitaria de un atributo dentro del algoritmo de *árbol de aislamiento*, esto hace que el algoritmo no tome en cuenta las correlaciones entre atributos pues cada árbol se construye en una dimensión del conjunto de datos.

De igual modo a lo anterior, el número de árboles que se seleccionen para el bosque siempre tiene que ser mayor al número de atributos en el muestreo. Por lo

tanto, el bosque debe de tener un número considerable de árboles para conjuntos de datos con dimensiones grandes.

CAPÍTULO 5

Detección de Anomalías con Enfoque con Redes Neuronales Artificiales

El cerebro del ser humano ha sido un enigma para el ser humano mismo.

Desde que el científico español Santiago Ramón y Cajal dibujó por primera vez una neurona (Ramón y Cajal & Azoulay , 1894), el ser humano ha estado un paso más cercano de entender el funcionamiento cerebral.

Históricamente, los primeros en ocupar una analogía de neurona aplicada en computación fueron McCulloch y Pitts (1943) (Figura 26), ambos influyeron en el surgimiento de las redes neuronales artificiales como las conocemos ahora.

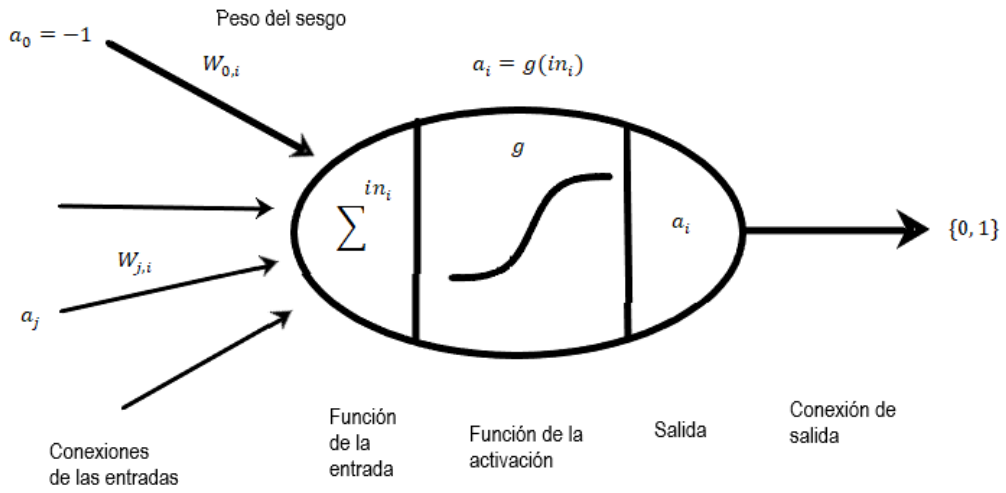

Figura 26. Primer modelo computacional de una neurona (Russell & Norvig, 2010).

Posteriormente, Frank Rosenblatt presentó lo que se conoce como perceptrón, en su publicación llamada *The Perceptron, a Perceiving and Recognizing Automaton* (1957). El perceptrón es una red que consiste en tres unidades de salida perceptrón que comparten cinco entradas (Figura 27).

Las aportaciones de Rosenblatt ayudaron a que se desarrollaran las primeras redes neuronales multicapa. Más adelante, Minsky y Papert, en su libro *Perceptrons: an introduction to computational geometry* (1969), demostraron que las redes de perceptrones de una sola capa estaban limitadas a representar conceptos linealmente separables, también apuntaban a la falta de un algoritmo de aprendizaje efectivo para las redes multicapa.

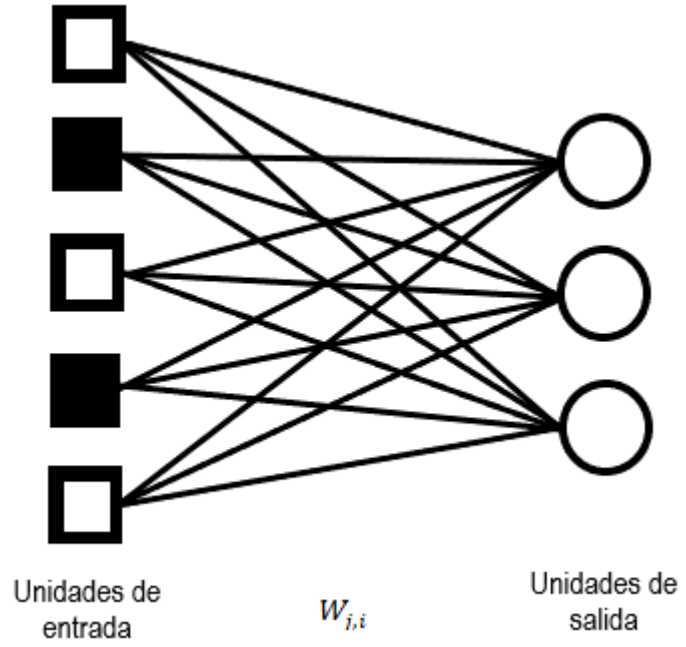

Figura 27. Perceptrón de tres unidades de salida (Russell & Norvig, 2010).

Existieron repercusiones posteriores a la publicación de Minsky y Papert, la más notoria fue la escasa indagación investigativa que tuvieron las redes neuronales, lo que hoy se conoce como el invierno de la inteligencia artificial. Otro factor a tener en cuenta, la falta de poder computacional de la época, las redes son muy

demandantes computacionalmente y la humanidad carecía de los elementos necesarios para realizar arquitecturas de redes complejas.

Fue hasta 1981 que Hinton y Anderson realizaron publicaciones basadas en la conferencia de San Diego de 1979 (Russell & Norvig, 2010), que se retomó el interés por las redes neuronales.

Aún faltaba un método de aprendizaje eficiente que permitiera el desarrollo de redes neuronales multicapa, lo anterior fue solucionado con la publicación de Rumelhart, Hinton y Williams, ellos presentaron el algoritmo de retropropagación en su publicación *Learning internal representations by error propagation* (1986). Aunado a lo anterior, el crecimiento de poder computacional que tuvieron los componentes electrónicos permitió el nacimiento de lo que hoy llamamos Aprendizaje Profundo (*Deep Learning*).

Aprendizaje profundo se le conoce a la aplicación de una red neuronal que contiene tres o más capas ocultas en su arquitectura (Krohn, Beyleveld, & Bassens, 2019).

Hoy en día las redes neuronales artificiales se han consagrado como una de las herramientas más usadas en todo tipo de aplicaciones, abarcando casi todo el espectro que ocupa el campo de la inteligencia artificial. Por ejemplo, en visión computacional, procesamiento de lenguaje natural (*NLP*), robótica y aprendizaje automático (*ML*).

5.1 Autoasociadores o Autocodificadores (*Autoassociators* o *Autoencoders*)

También conocidas como redes Diabolo (Bengio, 2009), los autocodificadores son una arquitectura en las redes neuronales artificiales, tienen como principal característica estar conformados por dos secciones, la sección de codificación y la sección de decodificación (Figura 28). Pese a estar conformado por dos secciones es catalogado como una misma arquitectura.

Originalmente este tipo de red fue diseñada para la reducción de dimensionalidad, ha resultado un mejor método para este fin que PCA (Hinton & Salakhutdinov, 2006). PCA está limitado a encontrar correlaciones lineales entre atributos, por el contrario, los autocodificadores pueden encontrar correlaciones no lineales.

En la presente tesis se limitará a ocupar una red con una capa oculta, la cual será la capa de codificación, está capa debe tener menor número de neuronas que la entrada y la salida de la red (Figura 28). Es posible ocupar aprendizaje profundo en los autoasociadores.

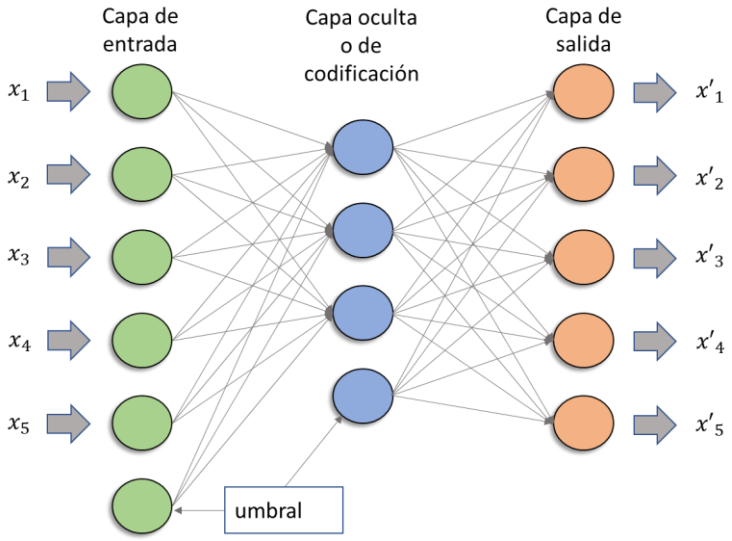

Figura 28. Arquitectura de un autoasociador con codificación de 5:3.

La idea de los autocodificadores es entrenar la red neuronal con el objetivo de reproducir como salida los datos de entrada. El algoritmo es el siguiente:

Autocodificador (entrada - datos X):

Repetir hasta $X \approx X'$:

1. Los datos X entran a la red.
2. Los datos transitan a través de la capa de codificación.
3. Obtenemos X' por la capa de salida.

4. Se calcula el error entre X y X'.
5. Se ejecuta retropropagación para calcular los nuevos pesos en proporción al error.
6. La red ajusta los nuevos pesos.

Una vez la red haya sido entrenada, se ingresan los datos X por la red, pero esta vez solo hasta la sección de codificación, es decir que la red de codificación da los nuevos datos x.

El nuevo conjunto de datos x ya cuenta con una dimensión más pequeña, este va a poseer un número de atributos igual a la cantidad de neuronas que tenga la capa de codificación.

Aprovechando las particularidades que brindan los autoasociadores, nació la detección de anomalías por medio de autocodificadores, esta fue desarrollada por Sakurada y Yairi (2014). Ellos descubrieron que los datos anómalos tienden a obtener un error grande en el algoritmo anterior.

Por lo tanto, una vez entrenada la red neuronal con nuestro conjunto de datos, esta red será capaz de reconstruir la misma entrada de la red, los datos que tengan una diferencia considerable podrán ser catalogados como anomalías.

Para etiquetar anomalías, este algoritmo recurre a dos caminos diferentes. El primero es estipulando un umbral, si el error entre un dato de entrada y su vector reconstruido supera ese umbral, el dato es etiquetado como anomalía.

En la segunda forma, se determina un porcentaje de anomalías, también conocida como contaminación y se ordena el error de forma descendente, los primeros datos que tengan los errores mayores de dicha lista, serán catalogados como anomalías.

Para determinar las anomalías por el primer método, se depende del conocimiento de un experto, ya que para determinar un umbral correcto es necesario que él lo estipule.

Para el segundo método de etiquetado de atípicos, la cantidad de datos a catalogar dependerá de que el experto determine un porcentaje de contaminación.

5.1.1 Detección de anomalías mediante Autocodificadores.

Para el siguiente ejercicio se utilizó un conjunto de datos extraído de la *UCI Machine Learning Repository*. Se trata de un muestreo de vino clasificado en tres tipos, esto a partir de ciertos atributos relacionados a la química del vino y se pueden adquirir libremente en la siguiente liga https://archive.ics.uci.edu/ml/datasets/wine.

El código generado para el siguiente ejemplo se puede encontrar en

https://github.com/amigo20th/Tesis_Deteccion_Anomalias/blob/master/DA_Autocodificador.ipynb[14]

El conjunto de datos de vino se compone de 178 instancias y 13 atributos, más el identificador de clase del vino, los atributos son los siguientes:

- Alcohol.
- Ácido málico.
- Ceniza.
- Alcalinidad de la ceniza.
- Magnesio.
- Total de fenoles.
- Flavonoides.
- Fenoles no flavonoides.
- Proantocianinas.
- Intensidad de color.
- Tono.
- OD280 / OD315 de vinos diluidos.
- Prolina.

[14] Si se requiere es posible copiar o descargar el código para modificarlo a conveniencia.

Para la DA por medio de autocodificadores en *Python*, ocuparemos una biblioteca llamada *pyod*, dicha biblioteca tiene una vasta gama de algoritmos para la detección de atípicos. Para instalarla es necesario el comando *pip*.

$$!\,pip\ install\ pyod$$

La instrucción anterior en necesaria para poder instalar la biblioteca, el signo de admiración es necesario solo en *Google Colab*, en *Colab* es donde se están desarrollando los ejemplos.

Una vez adquirimos el conjunto de muestro, para la ejecución correcta del autocodificador es necesario normalizar los datos.

Una vez ya estudiado el conjunto de datos, se decidirá la arquitectura que tendrá la red neuronal autocodificadora que desarrollaremos. En el presente ejemplo solo se ocupará una capa oculta o capa de codificación, esta capa contendrá dos neuronas, como resultado estaremos codificando los trece atributos a dos. La red para desarrollar cuenta gráficamente la estructura de la Figura 29.

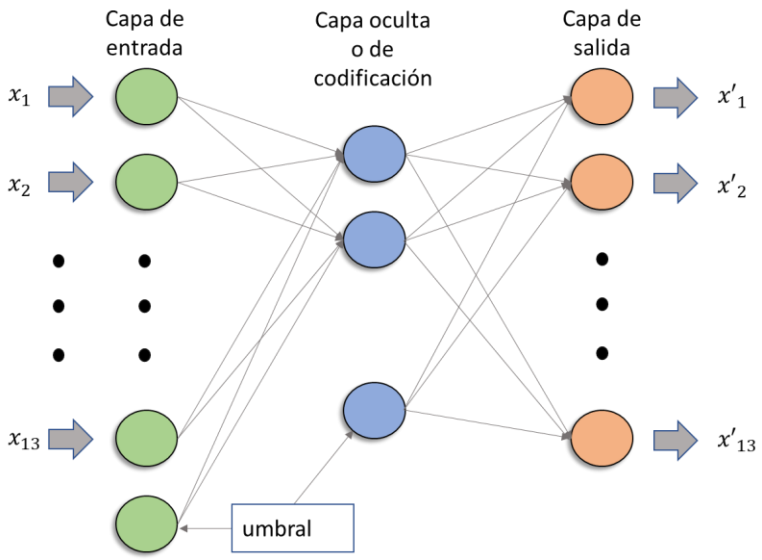

Figura 29. Arquitectura del autocodificador de ejemplo.

Entonces, ejecutamos el comando encargado de entrenar la red neuronal

$$AutoEncoder(hidden\_neurons = [2],$$
$$hidden\_activation = 'sigmoid',$$
$$epochs = 100,$$
$$contamination = 0.07)$$

En el comando pasaremos los parámetros que requiere la red neuronal. Los parámetros son:

- El número de neuronas en nuestra capa oculta.
- El tipo de activación, en este ejemplo ocuparemos la función sigmoide.
- Número de épocas
- El porcentaje de contaminación, por repeticiones del mismo ejemplo se observó que una contaminación del 7% da buenos resultados.

Predecimos el muestreo mediante el entrenamiento anterior, esto nos devolverá las anomalías 13, 14, 59, 69, 73, 95, 110, 115, 121, 137, 146, 158 y 171.

Para la visualización del resultado, es posible ocupar directamente la salida de la capa de codificación de nuestra red neuronal artificial, pero eso solo mostraría la redundancia del resultado. Si la red está mal entrenada, el etiquetado de anomalías, así como la visualización, estarían mal.

Por esa razón se creará una proyección de los datos en dos dimensiones por medio de PCA, esto es posible en el conjunto de datos del vino, este cuenta con linealidad en las correlaciones de sus atributos. De hecho, se logra observar claramente el agrupamiento de sus clases (Figura 30).

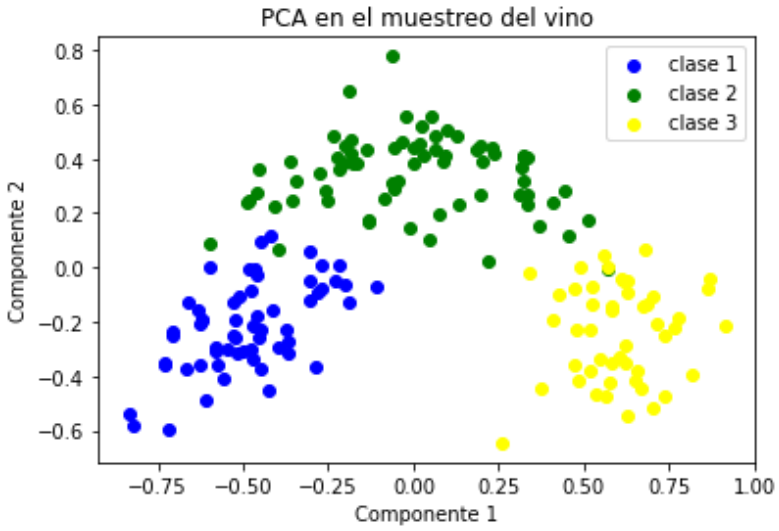

Figura 30. Proyección en dos dimensiones por medio de PCA al muestreo del vino.

Ahora, es posible graficar las anomalías sin caer en la redundancia, lo anterior nos da seguridad en los resultados obtenidos. Graficando las anomalías, obtenemos la Figura 31.

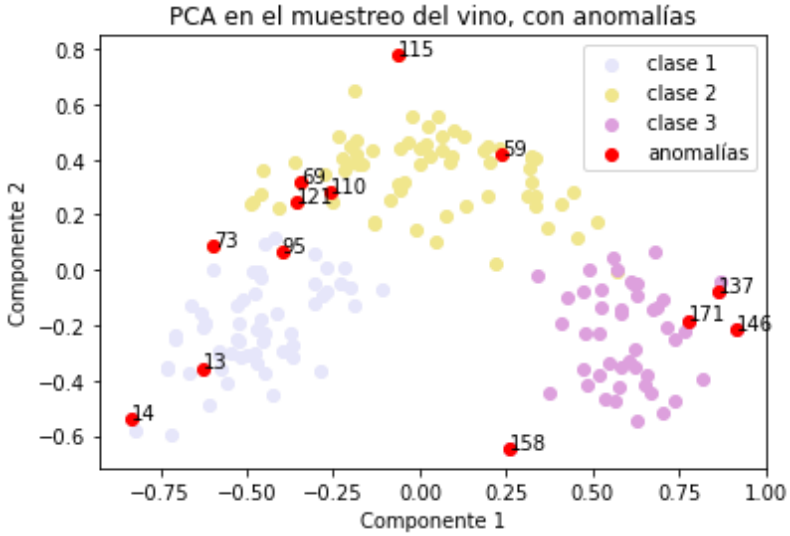

Figura 31. Anomalías detectadas por el autocodificador en el muestreo del vino

Se pueden observar puntos interesantes, logra detectar anomalías lejanas de la nube de puntos, como lo son la 14, 115, 137, 146 y 158. De igual manera detecta

atípicos en los bordes de los agrupamientos, como lo son el 69, 110 y 121. Pero los más interesantes son las muestras de vino 73 y 95, estos son los únicos puntos de vino clase 2 dentro del agrupamiento de clase 1, esto hace a ambos puntos atípicos dentro del grupo de clase 1 de vino.

5.1.2 Complejidad computacional de los autocodificadores

Calcular la complejidad computacional de una red neuronal artificial entrenada por retropropagación no es un problema trivial, pues la variabilidad de los parámetros, así como las condiciones de entrada afectan la complejidad.

Es intuitivo que la complejidad computacional es alta, puesto que es una restricción para el desarrollo de arquitecturas más robustas, aunque existen publicaciones que nos pueden dar un panorama cercano al cálculo de dicha complejidad.

Los investigadores Williams y Zipser en su texto *Gradient-based learning algorithms for recurrent networks and their computational complexity* (1995), nos dan una investigación sobre la historia de las complejidades que tienen diferentes arquitecturas de redes neuronales artificiales.

Vamos a realizar un analisis de complejidad para una red neuronal simple con arquitectura de autoasociador, como la ocupada en el ejemplo anterior.

Teniendo un conjunto de datos de tamaño n vectores y m dimensiones, el autocodificador va a contar de solo una capa de codificación de dimensión c, entonces la red va a contener $2m + c$ neuronas en total.

Se supondrá, todas las neuronas de la red van a tener una activación sigmoide, el método de entrenamiento va a ser por medio de retropropagación, el cálculo del costo solo será el promedio del error logarítmico de cada neurona[15], el entrenamiento será en e epocas.

[15] Más detalle para el entrenamiento de una red por medio de retropropagación en la siguiente liga https://github.com/amigo20th/Regresion_Logistica_Divorcios/blob/master/Regresion_Logistica_Divorcios_NN.ipynb

Primero, cada neurona recibe un vector y calcula la recta, posteriormente le aplica la función de activación. El cálculo de la recta prácticamente realiza un número de multiplicaciones igual a la cantidad de pesos que tiene la red.

El total de pesos de nuestra red, la capa de entrada tiene un total de $m+1$ neuronas, la neurona extra corresponde a la inclinación de la neurona (*bias*), la capa de codificación cuenta con $c+1$ neuronas y la capa de salida tiene m. Entonces la red tiene $((m+1)*(c+1))+((c+1)*m)$ pesos.

Entonces la cantidad de multiplicaciones para el cálculo de la recta es de $2m(c+1)+c+1$, a lo anterior le sumamos la función de activación que es de orden lineal respecto al número de datos. Por consiguiente, el orden del entrenamiento es de $O(cn+c+n+m)$.

Posteriormente, cada vector tiene que calcular su error respecto a cada dato de salida. Este paso es hecho en el orden de $O(n)$.

Entonces, es posible calcular la función costo, la cual es el promedio de los errores de cada vector, este paso es de orden $O(n)$.

Aquí empezamos con la retropropagación, se van a ajustar los pesos proporcionalmente a la función de costo, para una red del tipo autocodificador se tienen un total de $2m(c+1)+c+1$ pesos.

Cada uno de esos pesos va a calcular una derivada parcial, la va a multiplicar por la tasa de aprendizaje y el resultado se lo sumará al peso que tenía anteriormente. Este paso tiene una complejidad computacional de $O(cm)$.

Recordando que cada ajuste de pesos se realiza por cada vector en el conjunto de datos, entonces el orden completo del algoritmo de retropropagación es de $O(cnm)$.

Todos los pasos anteriores cuentan como una época, ya que se entrenó la red, obtuvimos el error de cada vector, el costo total del entrenamiento y se aplicó retropropagación.

Por lo tanto, la complejidad computacional estará determinada por la función mayor entre el entrenamiento y la retropropagación, siendo esta última la que crece más rápidamente, entonces una red autocodificadora simple es del orden $O(cenm)$.

5.1.3 Conclusiones

Los autocodificadores son excelentes al momento de reducir la dimensionalidad de un conjunto de datos, al mismo tiempo de detectar los posibles puntos atípicos que pueda tener el muestreo.

Con las herramientas adecuadas son fáciles de implementar, también es asequible su programación en dispositivos con poder computacional, como lo son las tarjetas de video, esto hace que el entrenamiento de una red sea más veloz.

Los autoasociadores no tiene problemas de maldición de la dimensionalidad, de hecho, la pueden disminuir, también son capaces de codificar datos con correlaciones no lineales.

Como punto negativo es su complejidad computacional, al tener una complejidad alta es recomendable que este tipo de soluciones sean las últimas en seleccionar, es mejor realizar la detección de anomalías por medio de los métodos descritos anteriormente.

Otro punto negativo es la selección de parámetros e hiperparámetros para un buen rendimiento de la red, no existe un algoritmo para determinar un número exacto de capas y tampoco la cantidad de neuronas, por lo que el entrenamiento y la puesta a punto de una red es un proceso estocástico. Aunado al hecho de que cada prueba de una red neuronal requiere un poder de procesamiento alto.

CAPÍTULO 6

Conclusiones Generales

Una anomalía es un elemento que se diferencia del resto de sus congéneres, por esa razón son una invaluable fuente de información sobre un fenómeno dado. Dada su poca probabilidad de ocurrencia, son necesarios algoritmos y métodos para su identificación.

En la presente tesis se explicaron algunos algoritmos capaces de identificar las anomalías, esto dentro de datos que surgen de una investigación.

El análisis de datos es una fase importante dentro de toda investigación, a su vez, el análisis exploratorio de datos es un proceso esencial dentro del análisis de datos. El análisis exploratorio de datos ayuda al investigador a conocer, al menos de manera superficial, la estadística y la geometría de los datos generados por un fenómeno.

Otro objetivo del análisis exploratorio es prevenir posibles contratiempos que puedan surgir en etapas posteriores del análisis de datos, una manera para evitar esas contrariedades es la detección, y posterior tratamiento, de anomalías.

Aprendimos a detectar las anomalías dependiendo de las características que tenga el conjunto muestral, ya que, como analistas o científicos de datos, es nuestro deber ocupar las opciones que mejor se adapten a la investigación.

Para conjuntos de datos univariados con distribución normal, presentamos la prueba de Grubbs y *Median Absolute Deviation*, estos cuentan con una orden de complejidad baja.

Para un conjunto muestral con poca correlación entre atributos, vimos el diagrama de cajas y bigotes, cabe señalar que este método solo detecta valores atípicos máximos o mínimos dentro de la distribución, también tiene un costo bajo computacionalmente hablando.

Para conjuntos de datos multivariados con una relativa baja dimensionalidad, se describió *Local Outlier Factor*. Dicho algoritmo es capaz de realizar detecciones de atípicos en localidades densas en el espacio de datos. *Local Outlier Factor* es susceptible a la maldición de la dimensionalidad y cuenta con un orden computacional alto.

Para muestreos con alta dimensionalidad y baja complejidad computacional, aprendimos bosque de aislamiento, el algoritmo evita la maldición de la dimensionalidad, solo que no es capaz de detectar anomalías dentro de una zona densa, no toma en cuenta la correlación entre atributos que pueda llegar a tener nuestro conjunto de datos.

Para conjuntos de datos con alta dimensionalidad que a su vez generan un mapeo a dimensiones más bajas, se presentaron los autocodificadores. Al pertenecer al campo de las redes neuronales artificiales cuentan con una complejidad computacional alta. Cuentan con la enorme ventaja de disminuir el espacio de dimensión, esto nos ayuda a la creación de los diagramas y así observar el comportamiento de los datos, esto sin necesidad de ejecuciones extras.

Las redes neuronales artificiales dan buenos resultados en detecciones de anomalías dentro de los campos de visión computacional y procesamiento de lenguaje natural. Este tipo de detecciones escapan al alcance de la presente tesis, quedan para trabajos a futuro.

Los analistas de datos o científicos de datos pueden crear un algoritmo ensamblado a partir de los que aquí fueron mostrados, cada algoritmo devolvería una cantidad de anomalías. Posteriormente, la catalogación de atípicos sería por votación, si un vector aparece en la mayoría de las listas creadas por los algoritmos, ese dato muy probablemente sería una anomalía.

El algoritmo ensamblado es de mucha ayuda cuando estamos ante un fenómeno carente de expertos, es decir, que se esté realizando una investigación totalmente nueva y que el equipo de trabajo esté adquiriendo la experiencia necesaria.

Apéndice A

A.1 Complejidad Computacional.

Para saber la eficiencia de una máquina, es necesario contar con métricas que sean compatibles entre los distintos aparatos para determinar cuál mecanismo es mejor respecto a los otros. La máquina más eficiente solo lo será respecto a la métrica utilizada.

Partiendo de la idea anterior, en ciencias de la computación se buscó la manera para determinar qué algoritmo hace mejor la tarea asignada, esto entre los diferentes algoritmos que tienen un mismo objetivo. Lo anterior dio como resultado la teoría de la complejidad computacional, la cual sirve para identificar qué algoritmos son más eficientes respecto a otros en distintas competencias, las métricas normalmente utilizadas en la complejidad computacional son referentes a la memoria que ocupa un algoritmo y el tiempo de ejecución. Esta última métrica será la de principal interés en la presente tesis.

Conocer la complejidad computacional es necesario en la detección de anomalías, puesto que existen análisis donde es necesario realizar una detección en tiempo real, un ejemplo de lo anterior puede ser una aplicación para evitar fraude bancario.

La teoría de la complejidad computacional fue introducida a principios de los 60's por J. Hartmanis y R. E. Stearns (Bečvář, 1967). Ellos idearon la manera de calcular el rendimiento de los recursos en un algoritmo[16] en función a la longitud de la entrada.

Calcular la complejidad computacional de un algoritmo respecto al tiempo no es cosa trivial, esto dada la cantidad de diferentes arquitecturas, capacidades y rendimientos que tienen las distintas computadoras personales. Entonces, ¿Cómo comparar un algoritmo en distintas computadoras personales con distintas

[16] Aquí nos referimos a algoritmo, pero los autores lo idearon principalmente para el modelo de máquina de Turing.

capacidades? No existe una respuesta fácil para esta pregunta, ya que no se podría dar una buena comparativa. Es por lo anterior que se utiliza una unidad de tiempo teórica, esta unidad de tiempo equivale una unidad por cada operación elemental.

Para conocer las Operaciones Elementales (OE), es necesario seccionar el algoritmo al cual vamos a analizar, como su nombre lo indica, estas OE son las operaciones más básicas que puede tener un algoritmo, normalmente estas OE se encuentran en distintos lenguajes de programación y tienen la misma funcionalidad, las OE se pueden representar plenamente con pseudo código. Guerequeta y Vallecillo nos comentan:

> "Consideraremos OE las operaciones aritméticas básicas, asignaciones a variables de tipo predefinido por el compilador, los saltos (llamadas a funciones y procedimientos, retorno desde ellos, etc.), las comparaciones lógicas y el acceso a estructuras indexadas básicas, como son los vectores y matrices." (1998)

Cada una de las OE que nos comentan Guerequeta y Vallecillo contabilizarán como una unidad de tiempo.

Ya contando con las unidades de tiempo de nuestra métrica, se puede tener un número para realizar nuestra comparativa, pero aún falta un tema para tener en cuenta, la longitud de la entrada, generalmente representada como *n*.

La longitud de entrada de un algoritmo hace referencia a los datos insertados al principio del algoritmo, estos son necesarios para su correcta funcionalidad. Dado que se puede ejecutar un mismo algoritmo en distintos contextos, a su vez estos contextos cuentan con una longitud de datos distinta, es difícil determinar un número exacto para poder resumir la efectividad de un algoritmo.

Para poder solventar lo anterior, la teoría de la complejidad computacional insertó las cotas de complejidad, estás cotas son tres y vienen representadas por (O, Ω, Θ). Las cotas de complejidad son funciones para determinar los límites, que en este caso es el tiempo, que tiene nuestro algoritmo. Las cotas son asintóticas, esto

respecto a la función creada por la cantidad de OE, que procesa el algoritmo, en función a diferentes longitudes de entrada.

A.1.1 Cota inferior o notación Ω

"Definición:

Sea $f: \boldsymbol{N} \to [0, \infty)$. Se define el conjunto de funciones de orden Ω de f como:

$$\Omega(f) = \{g: \boldsymbol{N} \to [0, \infty) \mid \exists c \in \boldsymbol{R}, c > 0, \exists n_0 \in \boldsymbol{N} \bullet g(n) \geq cf(n)\ \forall n \geq n_0\}.$$

Diremos que una función $t: \boldsymbol{N} \to [0, \infty)$ es de orden Ω de f si $t \in \Omega(f)$." (Guerequeta & Vallecillo, 1998).

De una manera más simple, sean g y f funciones respecto a n, se dice que g es de orden $\Omega(f)$ cuando f no crece más rápido que g. Esto acota nuestra función g, dándonos la longitud de entrada en la cual ocurre la menor cantidad de OE, dicha longitud de entrada también es conocida como mejor caso.

A.1.2 Cota superior o notación O

"Definición:

Sea $f: \boldsymbol{N} \to [0, \infty)$. Se define el conjunto de funciones de orden O de f como:

$$O(f) = \{g: \boldsymbol{N} \to [0, \infty) \mid \exists c \in \boldsymbol{R}, c > 0, \exists n_0 \in \boldsymbol{N} \bullet g(n) \leq cf(n)\ \forall n \geq n_0\}.$$

Diremos que una función $t: \boldsymbol{N} \to [0, \infty)$ es de orden O de f si $t \in O(f)$." (Guerequeta & Vallecillo, 1998).

Dicho de otra manera, sean g y f funciones respecto a n, se dice que g es de orden $O(f)$ cuando g no crece más rápido que f. De una manera análoga a la cota inferior, solo que en este caso se acota nuestra función g hacía arriba, dándonos la longitud de entrada en la cual ocurre la mayor cantidad de OE, dicha longitud de entrada es también llamada como peor caso.

A.1.3 Orden exacto o notación Θ

"Sea $f: \boldsymbol{N} \rightarrow [0, \infty)$. Se define el conjunto de funciones de orden Θ de f como:

$\Theta(f) = O(f) \cap \Omega(\mathrm{f})$" (Guerequeta & Vallecillo, 1998).

Lo anterior, se refiere al caso de las funciones que están delimitadas por la cota inferior tanto como la superior.

Tomando nuestro algoritmo a medir como función respecto a la longitud de entrada, se puede aplicar las notaciones anteriormente vistas para resumir la eficiencia del algoritmo.

En el caso computacional, se ocupa solamente la cota superior para resumir la eficiencia de un algoritmo, esto ya que estamos midiendo qué tan eficiente es el mismo en el peor de los casos.

Por lo tanto, ahora ya tenemos una métrica bien definida para hacer comparativas en los algoritmos sin importar la cantidad de información que estos procesen, esta métrica la denotaremos como $O(n)$, donde n representa la longitud de entrada.

Siendo $O(n)$ una función que sirve para describir el rendimiento de un algoritmo respecto al tiempo, existen funciones ya conocidas que son ideales para la descripción de la eficiencia, esto dado a la popularidad que tienen estás funciones en distintos campos.

Algunos ejemplos de cotas superiores que son constantes en los análisis son las siguientes:

Funciones Ordenadas por Tasa de Crecimiento
$\log n$
$log^2\, n$
$\sqrt{n}$
n
$n \log n$
n^2
n^3
2^n

Tabla 5. Una lista ordenada de simples funciones (Goodrich & Tamassia, 2001).

Las funciones mostradas por Goodrich y Tamassia nos sirven para determinar la eficiencia de un algoritmo, estas funciones catalogarán por órdenes a dichos algoritmos, es decir, para especificar el orden de un algoritmo con una función asintótica f, se escribe $O(f)$. Cabe señalar que existen más funciones asintóticas, las mostradas anteriormente solo son para ejemplificar.

Existen peculiaridades dentro de estas mismas cotas, ya que la última en aparecer en Tabla 1 es conocida por ser No Polinomial (NP), este tipo de algoritmos solo son viables con una longitud de entrada pequeña. Todas las demás cotas mostradas se les conoce como Polinomiales (P). El tema P y NP no será abarcado en el presente texto.

Bibliografía

Barbato, G., Barini, E. M., Genta, G., & Levi, R. (2011). Features and performance of some outlier detection methods. *Journal of Applied Statistics*, 2133-2149. doi:10.1080/02664763.2010.545119

Barnett, V., & Lewis, T. (1994). Outliers in Statistical Data. 3rd edition. *J. Wiley & Sons*, 582.

Bečvář, J. (1967). On the Computational Complexity of Algorithms by J. Hartmanis; R. E. Stearns. *On the Computational Complexity of Algorithms by J. Hartmanis; R. E. Stearns, 32*(1), 120-121. doi:https://doi.org/10.2307/2271275

Bellman, R. (1952). On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences*, 716-719. doi:10.1073/pnas.38.8.716

Bengio, Y. (2009). *Learning Deep Architectures for AI.* Montreal, Canadá: Foundations and Trends® in Machine Learning. doi:10.1561/2200000006

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 1-12. doi:https://doi.org/10.1145/342009.335388

Charytanowicz, M., Niewczas, J., Kulczycki, P., Kowalski, P. A., Łukasik, S., & Żak, S. (2010). Complete Gradient Clustering Algorithm for Features Analysis of X-Ray Images. *Advances in Intelligent and Soft Computing*, 15-24. doi:10.1007/978-3-642-13105-9_2

Davidson, A., Tarjan, D., Garland, M., & Owens, J. D. (2012). Efficient parallel merge sort for fixed and variable length keys. *2012 Innovative Parallel Computing (InPar)*, 1-9. doi:10.1109/inpar.2012.6339592

Filzmoser, P., Ruiz-Gazen, A., & Thomas-Agnan, C. (2014). Identification of local multivariate outliers. *Statistical Papers*, 29-47. doi:10.1007/s00362-013-0524-z

Galton, F. (1885). Anthropometric percentiles. *Nature*, 223-225.

Goodrich, M. T., & Tamassia, R. (2001). *Algorithm Design: Foundations, Analysis, and Internet Examples.* Wiley.

Guerequeta, R., & Vallecillo, A. (1998). *Técnicas de Diseño de Algoritmos.* Servicio de Publicaciones de la Universidad de Málaga.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN Model-Based Approach in Classification. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, 986-996. doi:https://doi.org/10.1007/978-3-540-39964-3_62

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Stanford, California: Springer-Verlag.

He, Z., Zhang, M., & Zhang, H. (2015). Data-driven research on chemical features of Jingdezhen and Longquan celadon by energy dispersive X-ray fluorescence. *Ceramics International*, 5123-5129. doi:10.1016/j.ceramint.2015.12.030

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 504–507. doi:10.1126/science.1127647

Hodge, V., & Austin, J. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 85-126.

Howell, D. C. (2005). Median Absolute Deviation. *Encyclopedia of Statistics in Behavioral Science*, 1193. doi:10.1002/0470013192.bsa384

Hughes, G. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 55-63. doi:10.1109/tit.1968.1054102

Iglewicz, B., & Hoaglin, D. C. (1993). *How to Detect and Handle Outliers.* Amsterdam University Press.

Kannan, K. S., Manoj, K., & Arumugam, S. (2015). Labeling Methods for Identifying Outliers. *International Journal of Statistics and Systems, 10*(2), 231-238.

Krohn, J., Beyleveld, G., & Bassens, A. (2019). *Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence.* Addison Wesley.

Larsen, R. D. (1985). Box-and-whisker plots. *Journal of Chemical Education*, 302-305. doi:10.1021/ed062p302

Leys, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 764-766. doi:10.1016/j.jesp.2013.03.013

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, 413-422. doi:10.1109/ICDM.2008.17

McCulloch, W., & Pitts, W. (1943). A Logical Calculus of The Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology*, 115-137.

Minsky, M., & Papert, S. (1969). *Perceptrons: an introduction to computational geometry.* Cambridge, Massachusetts: MIT Press.

Mosteller, F., & Tukey, J. (1977). *Data Analysis and Regression: A Second Course in Statistics.* Pearson.

Ramón y Cajal, S., & Azoulay , L. (1894). Les nouvelles idées sur la structure du système nerveux : chez l'homme et chez les vertébrés. *C. Reinwald & Cie*, 200.

Rosenblatt, F. (1957). The Perceptron, a Perceiving and Recognizing Automaton . *Cornell Aeronautical Laboratory:*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 318-362.

Russell, S., & Norvig, P. (2010). *Inteligencia Artificial, un enfoque moderno, 3rd Edition.* Madrid, España: Pearson Education.

Sakurada, M., & Yairi, T. (2014). Anomaly Detection Using Autoencoders with Nonlinear Dimensionality Reduction. *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, 4–11.

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 591-611.

Shiryayev, A. N. (1992). On The Empirical Determination of A Distribution Law. *Selected Works of A. N. Kolmogorov*, 139-146.

Sokal, R. R., & Hunter, P. E. (1955). A Morphometric Analysis of Ddt-Resistant and Non-Resistant House Fly Strains. *Annals of the Entomological Society of America*, 499-507. doi:https://doi.org/10.1093/aesa/48.6.499

Tukey, J. W. (1962). The Future of Data Analysis. *The Annals of Mathematical Statistics*, 1-67.

Tukey, J. W. (1977). *Exploratory Data Analysis.* Pearson.

Williams, R. J., & Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity.

Yahaya, S. W., Lotfi, A., & Mahmud, M. (2019). A Consensus Novelty Detection Ensemble Approach for Anomaly Detection in Activities of Daily Living. *Applied Soft Computing*, 13.