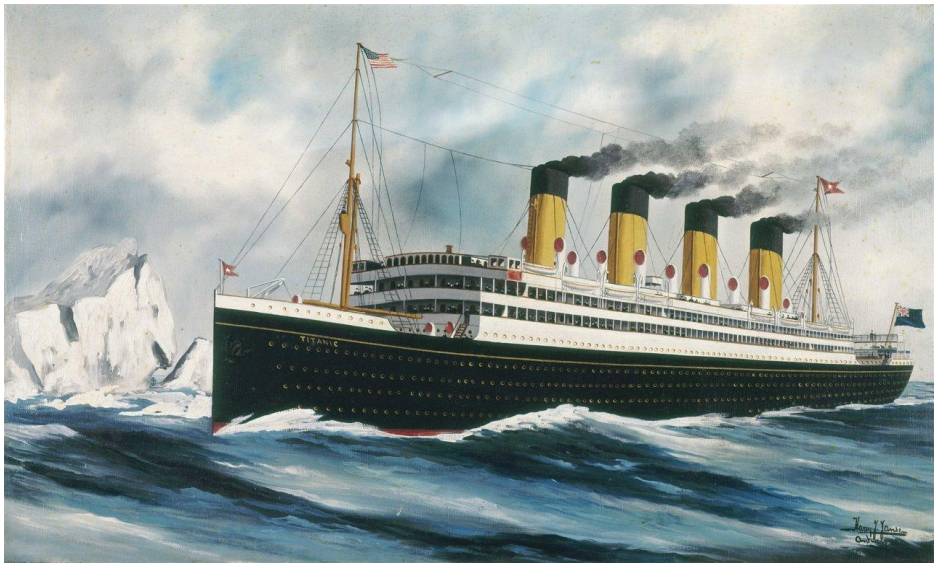
Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Inteligencia artificial avanzada para la ciencia de datos I
TC3006C.101

Reporte final



Rodolfo Sandoval Schipper A01720253
Marcelo Márquez A01720588
Arturo Garza Campuzano A00828096

11 ago 2023

Tecnológico de Monterrey

Historia de Revisión

Nombre	Fecha	Razón de Modificación	Versión
Rodolfo Sandoval Schipper	13 ago 2023	Revisión.	1
Marcelo Márquez Murillo	13 ago 2023	Revisión.	1
Arturo Garza Campuzano	22 ago 2023	Selección del modelo KNN.	1

Tecnológico de Monterrey

Índice general

Tecnológico de Monterrey

1. Introducción

El 15 de abril de 1912 el Titanic RMS se hundió después de haber chocado con un iceberg. Desafortunadamente, no había suficientes botes salvavidas para todos los que estaban a bordo, lo cual resultó en el fallecimiento de 1502 de los 2224 pasajeros y tripulantes. Si bien hubo algún elemento de suerte involucrado en la supervivencia, parece que algunos grupos de personas tenían más probabilidades de sobrevivir que otros.

Nuestro trabajo consiste en construir un modelo de predicción que responda a la siguiente pregunta: ¿qué tipo de personas tenían más probabilidades de sobrevivir? Utilizando los datos de los pasajeros que están en los archivos train.csv y test.csv, disponibles en la plataforma Kaggle: Titanic - Machine Learning from Disaster

El archivo train.csv contiene detalles de un subconjunto de 891 pasajeros, en el cual se revela cuáles de estos pasajeros sobrevivieron o no al accidente (*ground truth*). Por otro lado, el archivo test.csv contiene información similar pero no revela el resultado de supervivencia de cada pasajero. Este último archivo se utilizará para probar la efectividad de nuestro modelo de predicción. Utilizando los patrones encontrados en los datos de train.csv, nosotros necesitamos predecir si los otros 418 pasajeros a bordo (test.csv) sobrevivieron.

Por lo tanto, en resumidas cuentas, nuestro objetivo es crear un modelo que predice cuáles pasajeros sobrevivieron al naufragio del Titanic. La métrica de éxito para este reto corresponde al porcentaje de similitud que tienen los resultados de nuestro modelo comparado con los datos reales.

1.1 Objetivo

1.2 Alcance

Las etapas de nuestro proyecto son las siguientes:

1. Definición del problema
2. Adquisición de datos para entrenamiento (training) y prueba (testing)
3. Limpieza del conjunto de datos
4. Analizar, identificar patrones y explorar datos
5. Modelo, predicción y resolver el problema

Los objetivos de este proyecto son:

- Clasificar
- Correlacionar
- Convertir
- Completar

- Corregir
- Crear
- Graficar

2. Limpieza del Conjunto de Datos

Extracción, transformación y carga (Extract Transform Load, ETL) es un proceso de tres fases que toma los datos de un origen y los convierte a un formato válido. Organizaciones utilizan esta herramienta para recopilar datos de distintas fuentes para luego reunirlos a fin de facilitar el descubrimiento, la generación de informes, el análisis y toma de decisiones. En este caso, los datos que se desean **extraer**, **transformar** y **cargar** son aquellos que están contenidos en los archivos train.csv y test.csv.

2.1 Extraer

Se lleva a cabo la identificación de datos y se copian desde sus orígenes, con el propósito de transportarlos hacia el almacén de datos de destino; el cual es un archivo .csv con los datos ya transformados para su uso adecuado en los modelos. Los datos que se trabajarán más adelante proceden, afortunadamente, de orígenes estructurados; lo cual facilita la extracción de los mismos.

Este proceso de extracción comprende de tres etapas:

1. Análisis previo de las necesidades: se evalúan las necesidades concretas de la organización en cuanto a movimiento y transformación de datos.
2. Identificación de archivos: se identifica de qué tipo son los archivos y en qué formato se encuentran
3. Extracción de los datos: en función de las necesidades detectadas, se procede a la extracción en sí de dichos datos.

Métodos de extracción

1. Extracción total: consiste en extraer la totalidad de los datos.
2. Extracción incremental: se va procesando por lotes únicamente lo que fue modificado o agregado.
3. Notificación de actualizaciones: se van extrayendo los datos a medida que se produce una actualización.

2.2 Transformar

Dado que los datos extraídos no están procesados en su formato ideal, se deben asignar y transformar a fin de prepararlos para el almacén de datos final. Consiste en aplicar reglas de los datos extraídos para convertirlos en datos que se van a guardar en destino.

Algunas consideraciones para la transformación son las siguientes:

- Datos nulos
- Unidades de medida
- Traducir códigos
- Obtener valores calculados
- Relacionar diferentes fuentes de datos
- Agrupar datos
- Generar campos ID
- Dividir columnas
- Definir tratamiento de excepciones para datos erróneos
- Datos no estructurados

Dentro de este proceso de transformación está la limpieza de datos, cuyo objetivo es pasar solo datos adecuados al destino. Este proceso consiste en asegurar la calidad de los datos que serán cargados posteriormente. Los datos deben cumplir los siguientes requisitos:

- Correctos: que los valores de los datos describen los objetivos asociados con veracidad y fidelidad.
- Inequívocos: los valores y las descripciones de los datos pueden tomar sólo un sentido.
- Consistentes: sea un solo valor.
- Completos: asegurar que los valores individuales y descripciones de los datos se definen para cada instancia y asegurar que el número total de registros se ha completado.

La limpieza de datos también es conocida como depuración de datos y se refiere a la detección y eliminación o corrección de registros incorrectos o inconsistencias de datos para mejorar la calidad de los mismos. Su proceso es el siguiente:

- Eliminar variables irrelevantes.
- Verificar el tipo de valores perdidos
- Identificar variables categóricas.
- Observar los límites superior e inferior para datos numéricos.

La imputación de los datos no es más que un conjunto de métodos para manejar los datos faltantes en una base de datos, sus objetivos son:

1. Minimizar el sesgo entre los datos
2. Maximizar el uso de la información disponible
3. Obtener estimaciones apropiadas de incertidumbre

Técnicas de imputación

1. Análisis completo de casos
2. Imputación media
3. Imputación única

Bajo el supuesto de que cada columna obtenida del conjunto de datos es una variable se tiene la siguiente tabla categorizando cada una de ellas de acuerdo a todos sus valores posibles…

2.2.1 Identificación de duplicados

Eliminar los valores duplicados es un proceso esencial para la transformación de datos. Ya que puede disminuir la calidad del resultado debido a la alimentación repetida en caso de que se analice una interpretación incorrecta. Los valores duplicados pueden llegar a captar relaciones que no son significativas para afectar el rendimiento de un modelo. ya que para un modelo de aprendizaje solo se necesitan registros diferentes para calcular la mayor cantidad de posibilidades en caso de que se requiera.

El propósito de la eliminación de valores duplicados involucra:

1. Aumentar la eficiencia del modelo.
2. Eliminar el sesgo de predicciones. (El modelo puede aprender a dar más peso a ciertos patrones).
3. Evitar la distorsión de datos.

El proceso involucra:

1. Eliminación
2. Agregación
3. Análisis

2.2.2 Tratamiento de valores nulos

El tratamiento de valores nulos se enfoca en eliminar valores atípicos en el conjunto de datos. Este proceso es esencial para mantener la integridad de los datos, ya que los valores nulos representan la ausencia de información. Esto nos puede llevar a obtener resultados incorrectos o incoherentes en el modelo. Pero, los modelos pueden tratar con valores nulos de maneras diferentes que podrían distorsionar las relaciones de esos datos con de acuerdo a otros atributos en el conjunto.

El objetivo de eliminar estos valores es necesario para:

1. Mantener la integridad.
2. Dividir los datos de manera eficiente.
3. Evitar inconsistencias en los datos.
4. Mejorar la precisión del modelo.

2.2.3 Transformación y estandarización

La importancia de la transformación de datos se debe de mantener cómo un reglamento para la limpieza y la alimentación hacia un modelo. La transformación de

datos es el modificar o cambiar la estructura para adaptar la información de acuerdo a los requisitos del modelo para finalmente mejorar su calidad.El proceso se enfoca en normalizar la distribución para mejorar el rendimiento. Las resoluciones del sesgo para evitar colas largas en una dirección. El manejo de valores atípicos cómo valores lejos de la curva o ceros, y el escalamiento que ayuda con los cálculos basados en distancias. La transformación y la estandarización son procesos importantes y son esenciales para preparar los datos antes de alimentarlos a un modelo de aprendizaje automático. Estos procesos pueden mejorar la calidad de los datos, facilitar el proceso de modelado y llevar a mejores resultados de predicción y generalización.

2.2.4 Eliminación de outliers

El proceso de transformación requiere estandarizar la eliminación de valores atípicos (valores extremos en un conjunto de datos). Estos datos se alejan de la mayoría y pueden cambiar el resultado de un modelo. Los outlier afectan negativamente el impacto de la precisión ya que los modelos son más sensibles al ruido de datos y pueden afectar la media y la desviación estándar.

2.3 Cargar

Después de transformar los datos estos se cargan al destino seleccionado. Los datos transformados se mueven desde el área de ensayo a un almacén de datos destino.

3. Selección, Configuración y Entrenamiento del Modelo

En esta sección se presenta la selección, configuración y entrenamiento de los modelos candidatos que resolverán la problemática planteada en la introducción de este documento. A continuación se muestran los procesos de selección, configuración y entrenamiento de cada uno de ellos.

3.1 Selección del Modelo

3.1.1 Random Forest

3.1.1.1 Descripción (breve)

El **random forest** classifier es un modelo utilizado para problemas de clasificación y de regresión. Es un modelo que involucra el conjunto de varios árboles de decisión para aumentar la precisión y los resultados del modelo. El modelo de random forest se enfoca en la división de datos de entrenamiento y de prueba. En este caso, la

división de datos es distribuida con el ETL, tal que podemos analizar los casos prueba y los resultados dentro de un mismo archivo.

El modelo de random forest calcula la impureza gini para medir que tan mezcladas están las clases dentro de un conjunto de datos. Luego se hace la división en dos subconjuntos según el valor umbral para obtener valores menores o iguales y otro subconjunto para los valores mayores al umbral.

Finalmente se hace la construcción del árbol, que construye un árbol de acuerdo a las divisiones utilizadas. Si se alcanza una profundidad máxima o si todas las etiquetas son iguales, se crea un nodo con la clase más frecuente. Si no, sigue dividiendo el conjunto de datos en subconjuntos más pequeños y construyendo el árbol. Luego, hacemos la construcción del random forest para construir múltiples árboles de decisión usando subconjuntos aleatorios del entrenamiento. Cada árbol se construye con una profundidad máxima y se agrega al bosque.

Luego se hacen las predicciones con el random forestal y se analiza el resultado de cada árbol y devuelve la clase con mayor cantidad de votos.

3.1.1.2 ¿Porqué se seleccionó como modelo?

Se seleccionó debido a la precisión y generalización de los resultados. Esto es debido a que al combinar múltiples árboles de decisión, el modelo puede ser menos propenso al overfitting. Esto nos permite manejar características irrelevantes en el conjunto de datos y es seleccionado debido a que se proporciona información de importancia de cada característica en la toma de decisiones. Debido a esto, tuvimos que incorporar el proceso de ETL para mejorar el rendimiento y la eficiencia del modelo. Cuyo, afecta la toma de decisión y aumenta la significancia del valor dado y su precisión.

3.1.1.3 Metodología utilizada para construir el modelo

La metodología se explica en el punto 3.1.1.1. Sin embargo, es importante notar que también se calcula la precisión del cálculo comparando las predicciones del random forest con las etiquetas reales en el conjunto de prueba. La precisión es dividida por el número de predicciones correctas entre el número total de muestras. Estos resultados los guardamos dentro de un archivo csv para comparar si el pasajero sobrevivió o no.

3.1.2 Regresión Logística Multivariable

3.1.2.1 Descripción (breve)

La regresión logística multivariable (o múltiple) es un modelo de aprendizaje el cual se utiliza para problemas de clasificación donde hay varios valores a considerar. En sí, este modelo es una extensión de la regresión logística binaria pero que permite

más de una variable independiente (x) mientras que el resultado siga siendo una variable dependiente binaria (y).

En si la regresión logística múltiple tiene dos objetivos:

1. Entender cómo ciertos factores pueden influir en la probabilidad de que algo específico suceda. Por ejemplo, cómo afecta la edad y el sexo en la probabilidad de comprar algún producto o en el caso del reto, la probabilidad de sobrevivir un suceso como el Titanic.
2. Encontrar la forma más simple y efectiva de usar varias variables con el fin de predecir algún resultado.

3.1.2.2 ¿Porqué se seleccionó como modelo?

Dado a la naturaleza de los datos de este reto, disponemos de un conjunto de datos los cuales detallan diversas características de los pasajeros del Titanic, como su clases social, edad y sexo, al igual de si sobrevivieron o no el desastre expresado de manera binaria (1 para sobrevivientes y 0 para no sobrevivientes). Este modelo fue seleccionado por las siguientes razones:

1. Facilidad de interpretación: A diferencia de modelos más complejos como el Random Forest o KNN, la regresión logística múltiple ofrece una interpretación más directa y comprensible de cómo cada variable influye en la probabilidad de que un pasajero sobreviva.
2. Eficiencia: Este modelo es relativamente rápido de entrenar al igual que validar, el cual es útil cuando se necesita iterar y/o ajustar el modelo varias veces.
3. La variable dependiente es binaria.

3.1.2.3 Metodología utilizada para construir el modelo

1. Selección de variables: Se deben de escoger qué variables independientes serán utilizadas para entrenar al modelo. La inclusión de variables irrelevantes pueden llegar a afectar la eficiencia del modelo.
2. Usos de theta: Considerando la cantidad de variables independientes que se utilizaran, se tendrá la misma cantidad de thetas mas uno (tomando en cuenta la base o theta0) todos con valor de 1.
3. Número de iteraciones y el *learning rate*: A prueba y error, se cambiaban los estos valores con fin de conseguir un modelo que esté mejor entrenado y logre acertar correctamente a las predicciones.
4. Validación del Modelo: A través del empleo de herramientas analíticas como la matriz de confusión, y métricas especializadas como la exactitud, la precisión, la sensibilidad y la puntuación F1, somos capaces de evaluar de manera exhaustiva el rendimiento del modelo entrenado. Este enfoque nos permite iterar y refinar el modelo con el objetivo de optimizar los resultados obtenidos.

3.1.3 Clasificador KNN

3.1.3.1 Descripción

El algoritmo de **k vecinos más cercanos** (KNN) es un algoritmo clasificador de aprendizaje de máquina supervisado no paramétrico que utiliza la proximidad para hacer clasificaciones o predicciones sobre la agrupación de un punto de datos individual (IBM, s.f.).

El **proceso** del algoritmo se resumen en los siguientes pasos:

1. Elección del número de k y una métrica de distancia.
2. Encontrar los k vecinos más cercanos de la muestra que se quiere clasificar.
3. Asignar la etiqueta de clase por mayoría de votos.

KNN es un modelo de *aprendizaje perezoso*, lo cual indica que no aprende una función discriminativa en el conjunto de datos de entrenamiento, sino que **memoriza** este mismo conjunto.

La principal **ventaja** de este enfoque basado en la memoria es que el clasificador se adapta inmediatamente a medida que se recopilan nuevos datos de entrenamiento. Sin embargo, la **desventaja** es que la complejidad computacional para clasificar nuevas muestras crece linealmente con el número de muestras del conjunto de datos de entrenamiento en el peor de los casos, a menos que el conjunto de datos tenga muy pequeñas dimensiones (características) y el algoritmo se haya implementado utilizando estructuras de datos eficientes como árboles KD (Sebastian Raschka, 2015).

¿Cómo se define el valor de k en nuestro modelo?

En realidad, no se tiene una regla definida para encontrar el valor más óptimo para k vecinos. No obstante, se sabe que su adecuada elección es crucial para encontrar un buen balance entre subajuste y sobreajuste (Sebastian Raschka, 2015). También es importante encontrar la métrica de distancia apropiada para los atributos del conjunto de datos objetivo.

Cabe mencionar que en algunos recursos audiovisuales se encontraron los siguientes criterios para aproximarnos al valor ideal de k:

1. El valor de k puede ser estimado por medio de la siguiente fórmula: $k = \sqrt{n}$. Donde n es el número total de datos puntuales.
2. Se recomienda utilizar un valor impar para k para efectos de facilitar el proceso de votación.
3. Procurar que el valor de k no sea muy pequeño ni muy grande.

3.1.3.2 ¿Porqué se seleccionó como modelo?

Su selección se basa en las siguientes observaciones del conjunto de datos:

1. El tamaño del conjunto no es tan grande.
2. Cuenta con muy pocas características a comparación de otros conjuntos de datos populares.

3.1.3.3 Metodología utilizada para construir el modelo

La metodología utilizada para la construcción del modelo fue la siguiente:

1. Investigación del algoritmo KNN
2. Implementación del algoritmo sobre otros conjuntos de datos
3. Implementación del algoritmo sobre la problemática del Titanic

3.2 Configuración del Modelo

3.2.1 Random Forest

3.2.1.1 Configuración 1

Para el modelo de random forest se utilizaron varios parámetros para configurar la precisión. Primero configuramos el número de árboles **num_arboles** = 100. Esto indica el número de árboles que se van a construir en el random forestal. Luego ajustamos la profundidad máxima. **max_depth** = 3, el cual establece la profundidad de cada árbol de decisión. Un valor más alto permite obtener árboles más profundos. Lo dejamos en 3 debido a que un valor alto puede llegar a ser overfitting.

Utilización en el código.-
En la función **build_random_forest** se construye el bosque aleatorio creando múltiples árboles de decisión en base a los subconjuntos de datos aleatorios. Cada árbol se construye con la configuración de **max_depth**.

```
num_arboles = 100
max_depth = 3
forest = build_random_forest(train_data, num_arboles, max_depth)
```

La configuración óptima para el modelo Random Forest, con num_arboles = 100 y max_depth = 3, se basa en una serie de consideraciones clave que equilibran la complejidad del modelo y su capacidad de generalización.

Primero, al establecer un límite de profundidad de max_depth = 3, estamos tomando una decisión deliberada para evitar que los árboles de decisión se vuelvan demasiado profundos. Esta restricción tiene una razón de ser: evitar el sobreajuste. Limitar la profundidad de los árboles ayuda a prevenir que el modelo se adapte excesivamente a los datos de entrenamiento y, en cambio, capture patrones

generales y relevantes. En un conjunto de datos relativamente pequeño como el del Titanic, existe el riesgo de que los árboles complejos se ajusten demasiado a las particularidades de los datos, lo que podría resultar en un rendimiento deficiente en nuevos datos.

La elección de num_arboles = 100 refleja el deseo de aprovechar los beneficios del enfoque de bosque aleatorio. Un mayor número de árboles generalmente mejora la precisión y la capacidad de generalización del modelo. Sin embargo, también hay un equilibrio que considerar. A medida que aumentamos el número de árboles, los beneficios pueden comenzar a disminuir, lo que lleva a un rendimiento marginalmente mejor a expensas de un tiempo de entrenamiento más prolongado. En este caso, 100 árboles se considera una cantidad suficiente para lograr una buena generalización sin caer en un exceso de complejidad computacional.

3.2.2 Regresión Logística Multivariable

3.2.2.1 Configuración 1

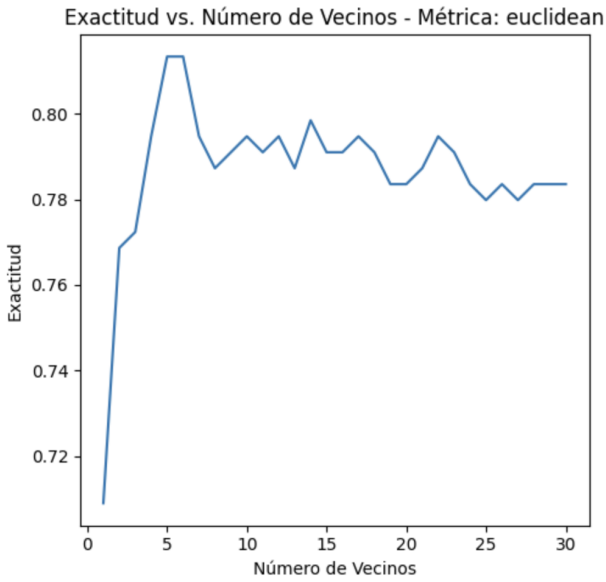
Para la primera configuración, lo primero que se consideró fue la cantidad de datos. Originalmente, el dataset del Titanic ya modificado (por el proceso de *ETL*) cuenta con 891 renglones (cada renglón siendo un pasajero) pero el número de personas que sobrevivieron es mucho menor que esos que no sobrevivieron (342 y 549) así que se creó un archivo que mantenga la misma cantidad de sobrevivientes y no sobrevivientes (684 en total). Los otros dos datos a considerar son esos del número de iteraciones al igual que su *learning rate,* por medio de pruebas con matrices de confusión al igual que métricas especializadas como la exactitud, la precisión, la sensibilidad y la puntuación F1, se corrió el entrenamiento del modelo retocando estos valores tratando de conseguir la mejor puntuación de las métricas. Después de prueba y error se decidió utilizar el valor de 300,000 para el número de iteraciones al igual que 0.1 para el *learning rate*. Por último, se hicieron pruebas con el modelo ya entrenado utilizando los datos completos.

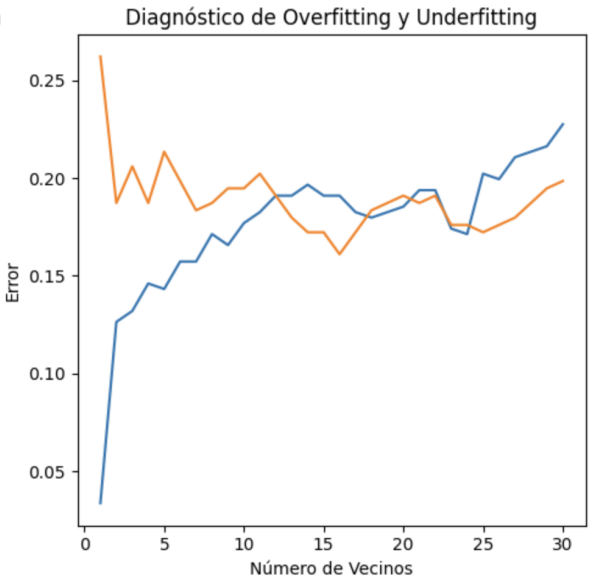
3.2.2.2 Configuración 2

Para la segunda configuración, se hizo el mismo proceso a la modificación de los datos utilizando *ETL* como la primera configuración mientras que no se consideró la misma cantidad de sobrevivientes y no sobrevivientes, se decidió dividir los datos de la siguiente manera: 70% para entrenar el modelo y 30% para las pruebas y predicciones. Similar a la configuración 1, también se tomó mucho en consideración ambos valores de número de iteraciones (300,000) al igual que su *learning rate* (0.1). Tras prueba y error se llegó al valor de

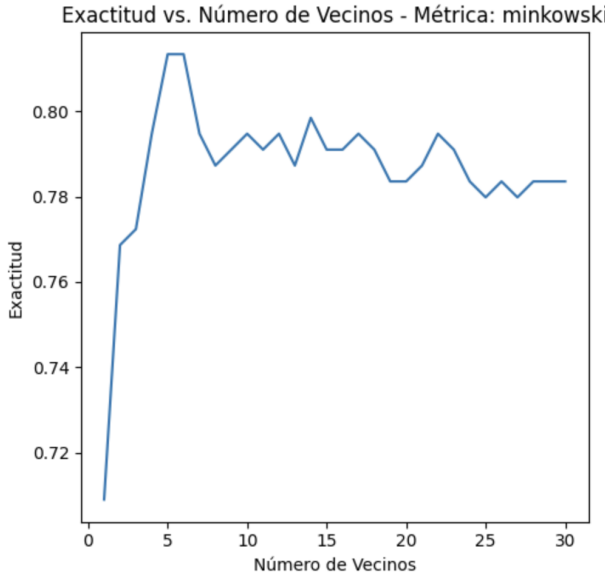
3.2.3 Clasificador KNN

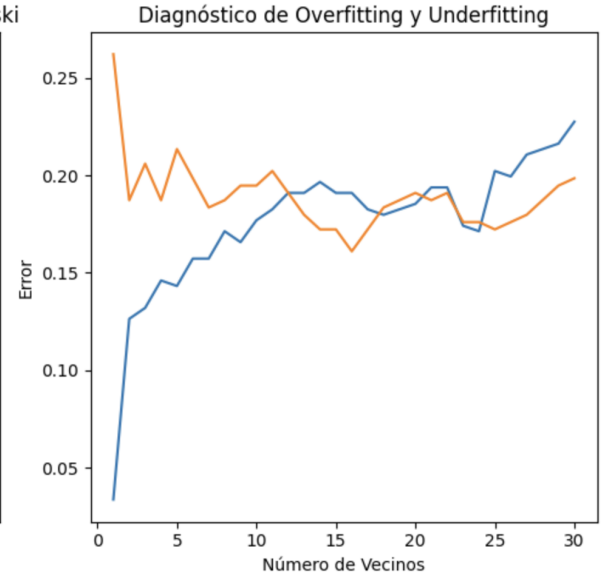
3.2.3.1 Configuración 1: Métrica Euclidiana



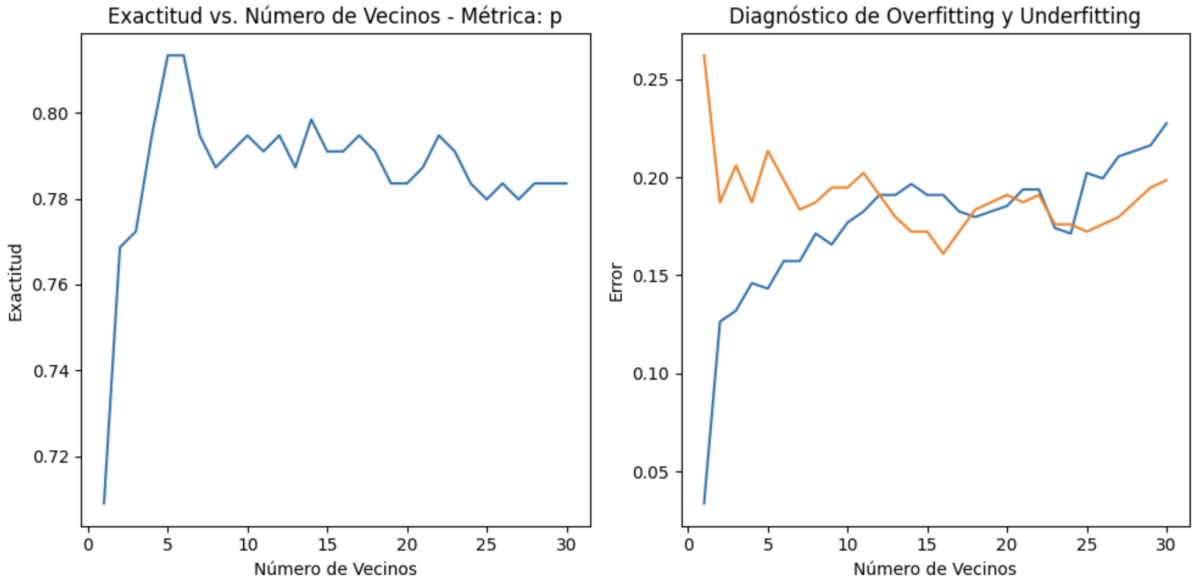


3.2.3.2 Configuración 2: Métrica Minkowski





3.2.3.3 Configuración 3: Métrica P



3.3 Entrenamiento del Modelo

3.3.1 Random Forest

3.3.1.1 Entrenamiento

Con de acuerdo a la configuración anterior establecida en el random forest. El entrenamiento funciona de la siguiente manera. El modelo elige un set random de datos, esto siendo una cantidad predefinida en la configuración dentro del código. Sin embargo, estos datos son los que se utilizan para entrenar el modelo y así predecir si el pasajero sobrevive o no. Es necesario mantener estas configuraciones estables para ignorar la implementación de un modelo que realiza un underfitting o overfitting de datos. Tal que, es recomendable dejar la configuración cómo ya está establecida en el código.

3.3.1.2 Resultados

Los resultados del modelo se aproximan al .87 de precisión cómo la media de los resultados. Esto es debido a que se analizan cachos de datos para realizar el entrenamiento del modelo. Este rango de precisión oscila entre 76 a 90 por ciento de precisión. Sin embargo, recomendamos dejar estas configuraciones ya que ayuda a entrenar el modelo en diferentes casos.

3.3.2 Regresión Logística Multivariable

3.3.2.1 Entrenamiento

Como se mencionó anteriormente (3.2.2), la principal diferencia entre las dos configuraciones para el entrenamiento del modelo de regresión logística múltiple

radica en la forma en que se introdujeron los datos. Dicho esto, ambos escenarios se sometieron a pruebas rigurosas para optimizar los parámetros clave, específicamente el número de iteraciones y la tasa de aprendizaje. Se realizaron múltiples corridas con diferentes combinaciones de estos parámetros, ajustándose iterativamente para alcanzar un rendimiento óptimo, conforme a métricas de evaluación establecidas para este tipo de modelos.

3.3.2.2 Resultados

Para determinar cuál de las dos configuraciones rinde mejor, recurrimos a diversas métricas de evaluación. Utilizamos los cuatro cuadrantes de la matriz de confusión —Verdaderos Positivos, Verdaderos Negativos, Falsos Positivos y Falsos Negativos—, así como indicadores adicionales como la exactitud, la precisión, la sensibilidad y la puntuación F1. El objetivo era realizar una comparativa exhaustiva que nos permitiera seleccionar el modelo más eficiente.

```
Configuración 1
Model: vp - 254, vn - 462, fp - 88, fn - 87
Accuracy: 0.8, Precision: 0.74, Recall: 0.74, F1 Score: 0.74
```

```
Configuración 2
Model: vp - 73, vn - 143, fp - 27, fn - 25
Accuracy: 0.81, Precision: 0.73, Recall: 0.74, F1 Score: 0.73
```

Aunque las diferencias en el rendimiento de ambos modelos resultaron ser marginales, con una variación de apenas ±1 punto porcentual, optamos por emplear la Configuración 1 para el entrenamiento definitivo del modelo, dado que demostró tener un rendimiento ligeramente superior.

3.3.3 Clasificador KNN

3.3.2.1 Entrenamiento

Para implementación del modelo KNN sobre la presente problemática se tomaron en cuenta las siguientes cosas:

1. División del conjunto de datos dedicado al entrenamiento del modelo en los subconjuntos de entrenamiento (60%), validación (20%) y prueba (20%).
2. La configuración del modelo consistirá en el uso de diferentes métricas para encontrar la distancia de los vecinos más cercanos de los datos puntuales (pasajeros).

3. Las mejores configuraciones serán aquellas que proporcionen la mejor exactitud con el valor más pequeño de k. Entre más grande sea el valor de k más procesamiento requiere el modelo.
4. Para cada configuración (métrica) se probó con un rango de valores de k debido a que no hay una regla general para encontrar su valor óptimo.

3.3.2.2 Resultados

En este caso, el modelo se probó con trece métricas de distancia y se obtuvieron los siguientes resultados:

	metric	max_acuraccy	k_value
0	minkowski	0.813433	5
3	sqeuclidean	0.813433	5
9	euclidean	0.813433	5
10	p	0.813433	5
12	braycurtis	0.813433	8
2	infinity	0.809701	5
4	chebyshev	0.809701	5
6	cosine	0.809701	10
7	canberra	0.809701	21
5	manhattan	0.805970	8
11	cityblock	0.805970	8
8	correlation	0.787313	5
1	hamming	0.783582	6

Imagen 1. Resultados de las configuraciones del modelo KNN

En la sección de configuración se puede encontrar más detalle sobre los resultados de las mejores configuraciones.

3.4 Elección del mejor modelo

Tomando en cuenta las mejores configuraciones de cada modelo se puede concluir que el modelo de **random forest classifier** tiene las mejores características y opciones de configuración para obtener los resultados más precisos para el conjunto de datos en la situación problema “titanic”. El random forest es ideal para el manejo de clasificación de un dataset. En este caso hemos configurado las mejores propiedades para eficientizar el proceso y obtener un resultado con más precisión. Hemos elegido este modelo debido a los resultados, su eficiencia, precisión, margen de error, y la comparación con todos los modelos anteriores. Logramos obtener una

media más alta de resultados con random forest classifier, esto indica que el modelo es ideal para problemáticas cómo la del titanic. La ventaja de este modelo también involucra el adaptar nuevos atributos dentro del conjunto de datos sin afectar el rendimiento. Tal que, solo sería necesario configurar los parámetros para seguir un estándar donde los resultados y el entrenamiento sea consistente y preciso.

4. Referencias bibliográficas

1. Sebastian Raschka. (2015). *Python Machine Learning*. Packt Publishing.
2. Oracles. “¿Qué Es ETL?” *Oracle.com*, 2021, www.oracle.com/mx/integration/what-is-etl/#:~:text=Extracci%C3%B3n%2C%20transformaci%C3%B3n%20y%20carga%20(Extract. Accessed 22 Aug. 2023.
3. IBM. “¿Qué Es El Algoritmo de K Vecinos Más Cercanos? | IBM.” Www.ibm.com, www.ibm.com/mx-es/topics/knn.