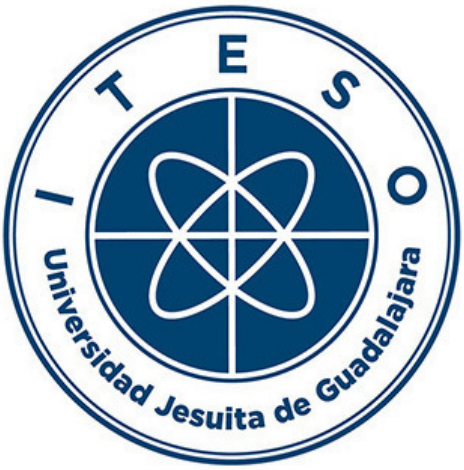
Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
Maestría en Sistemas Computacionales



AUTOENCODERS PARA MANEJO DE CLASES DESBALANCEADAS EN PROBLEMAS DE PROPENSIÓN DE COMPRA

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN SISTEMAS COMPUTACIONALES

Presenta: **EDGAR GARIBAY RUIZ**

Director **DR. ISMAEL MORENO NÚÑEZ**
Co-Director **DR. RIEMANN RUIZ CRUZ**

Tlaquepaque, Jalisco. 27 de septiembre de 2018.

AGRADECIMIENTOS

Primero que nada, quisiera agradecer a mi familia, sobre todo a mi esposa, por todo el apoyo y la paciencia durante esas largas horas de trabajo requerida para cumplir con este reto.

Así mismo, agradezco a todos mis maestros y a mi asesor por todos los conocimientos que me transmitieron.

Quiero agradecer también al CONACYT por la beca otorgada. Numero de Beca 442099.

Agradezco también al ITESO por haberme permitido ser parte de esta gran universidad y por la beca que me proporciono al ser empleado de Oracle de México. Y así mismo, agradezco a Oracle de México, la empresa donde laboro, por haber cubierto el resto de la colegiatura.

DEDICATORIA

A mi esposa, quien me inspira a ser una mejor persona y cumplir todas mis metas, y quien me ha apoyado en todo el trayecto de este reto.

RESUMEN

El problema principal que se aborda en este documento es aquel en el que un algoritmo de clasificación no cuenta con el desempeño requerido debido a que el conjunto de datos con el que fue entrenado esta desbalanceado.

Se tiene como objetivo analizar estrategias para incrementar la precisión de un algoritmo de clasificación cuando el conjunto de datos con el que es entrenado cuenta con clases desbalanceadas. Una de las estrategias que se analizan es el uso de *Autoencoders* para la generación de un conjunto de nuevas características que permitan realizar una mejor clasificación.

Al generar este conjunto de nuevas características utilizando un *Autoencoder*, se demuestra que la precisión de un algoritmo de clasificación aumenta. Esta demostración se realizó al comparar el desempeño de un clasificador al ser entrenado con y sin las nuevas características generadas por el *Autoencoder*. Se utilizaron diferentes métricas para medir el desempeño y realizar la comparación, entre ellas la matriz de confusión, precisión, *recall*, *accuracy* and F1-score.

Aunque las nuevas características generadas por el *autoencoder*, es necesario tener en cuenta la topología y configuración de este debe ser adaptada a cada problema. Así mismo, se debe tener en cuenta el tiempo requerido para el entrenamiento del *autoencoder*, y que este será proporcional la cantidad de características y número de registros del conjunto de datos.

TABLA DE CONTENIDO

AGRADECIMIENTOS **3**

DEDICATORIA **4**

RESUMEN **5**

TABLA DE CONTENIDO **6**

LISTA DE FIGURAS **8**

LISTA DE TABLAS **9**

1. INTRODUCCIÓN **10**

1.1. ANTECEDENTES 11
1.2. JUSTIFICACIÓN 11
1.3. PROBLEMA 12
1.4. OBJETIVOS 12
1.4.1. Objetivo General: 12
1.4.2. Objetivos Específicos 12
1.5. NOVEDAD CIENTÍFICA, TECNOLÓGICA O APORTACIÓN 12

2. ESTADO DEL ARTE O DE LA TÉCNICA **13**

2.1. LENGUAJES, HERRAMIENTAS Y LIBERARÍAS PARA *MACHINE LEARNING* 14
2.2. LIBRERÍAS Y FRAMEWORKS DE *MACHINE LEARNING* PARA PYTHON 14
2.3. SOLUCIONES INTEGRALES PARA MARKETING MACHINE LEARNING 16
2.4. MÉTODOS PARA CHURN PREDICTION 16
2.5. TÉCNICAS PARA PREDECIR CUSTOMER LIFETIME VALUE 17
2.6. PRE-PROCESAMIENTO Y LIMPIEZA DE DATOS 17
2.7. EVALUACIÓN DE LOS MODELOS 18
2.8. MÉTODOS PARA CLASES DESBALANCEADAS 18
2.9. EXPERIMENTACIÓN 20
2.10. *AUTOENCODERS* PARA MANEJAR CLASES DESBALANCEADAS 21

3. MARCO TEÓRICO/CONCEPTUAL **22**

3.1. PREDICCIÓN DE PROPENSIÓN DE COMPRA. 23
3.2. *AUTOENCODER* 24
3.3. *AUTOENCODER* INCOMPLETO 25
3.4. *DENOISING AUTOENCODER* 25
3.5. *STACKED AUTOENCODER* 26

4. DESARROLLO METODOLÓGICO **27**

4.1. PROCESO DE ANÁLISIS 28
4.2. *DATASETS* 28
4.2.1. Dataset 1: Bank Marketing *28*
4.2.2. Dataste 2: Historial De Compra de Clientes *31*

4.3. Pre Procesamiento y Limpieza de Datos 32
4.4. Generación de Nuevas Características 32
4.5. Validación 34
5. RESULTADOS Y DISCUSIÓN 36
5.1. Resultados 37
5.1.1. Dataset 1: Bank Marketing 37
5.1.2. Dataset 2: Historial De Compra de Clientes 38
6. CONCLUSIONES 40
6.1. Conclusiones 41
6.2. Trabajo Futuro 41
BIBLIOGRAFÍA 42

LISTA DE FIGURAS

Figura 1. Apache Spark Librerias [9] 15
Figura 2. GraphLab Framework [10] 15
Figura 3. Muestreo [24] 19
Figura 4. *Autoencoder* para generar nuevas características 21
Figura 5. Comparativa de resultados utilizando *autoencoders* 23
Figura 6. Estructura básica de una *autoencoder* 24
Figura 7. *Denoising Autoencoder* 25
Figura 8. *Stacked Autoencoder* 26
Figura 9. *Stacked Autoencoder para el dataset de Bank Marketing* 33
Figura 10. *Stacked Autoencoder para el dataset de Historial del Compras* 33
Figura 11. Flujo de datos 34
Figura 12. Matriz de confusión con *dataset* original 37
Figura 13. Matriz de confusión usando nuevas características 38
Figura 14. Matriz de confusión con *dataset* original 38
Figura 15. Matriz de confusión usando nuevas características 39

LISTA DE TABLAS

Tabla 1. Comparación de modelos [12] 17
Tabla 2. Propensión de compra de un producto bancario 23
Tabla 3 Comparativa de resultados utilizando DDAF 24
Tabla 4. Datos demográficos y de historia crediticios 29
Tabla 5. Ultimo contacto 29
Tabla 6. Otros datos 29
Tabla 7. Extracto de datos 1 30
Tabla 8. Extracto de datos 2 30
Tabla 9. Extracto de dataset de historial de compra 32
Tabla 10. Comparativa de resultados (Bank Marketing dataset) 38
Tabla 11. Comparativa de resultados (Historial de Compra) 39

1. INTRODUCCIÓN

Resumen: *En este capítulo se presentan brevemente los antecedentes del objeto de estudio, justificación del objeto de estudio, justificación y la definición del problema. Así mismo, se da una breve descripción de la aportación que generará el desarrollo de este trabajo.*

1.1. Antecedentes

En años recientes, términos como *Data Scientist, Predictive Marketing y Marketing Technologist* han comenzado a tener mucha relevancia en las grandes empresas -, ya que estas están comenzando a aprovechar nuevas tecnologías; en especial aquellas relacionadas con Big Data y ciencia de datos, para entender, atraer y retener a sus clientes.

De acuerdo con [3] no hay suficientes profesionales calificados para desempeñar un puesto de data science y que para 2018 puede haber entre 140,000 y 190,000 vacantes pendientes. Peor aún, se pronostica que 1.5 millones de directivos necesitarán optimizar la información disponible [1], [3].

Un alto porcentaje de grandes firmas tienen dentro de su nómina empleados con el puesto de Marketing Technologists, capaces de integrar la tecnología con la toma de decisiones relacionadas con el marketing.

Por otro lado, la ciencia de datos ha logrado posicionarse como la nueva estadística, esto es, una combinación de técnicas de modelado, tecnologías de la información y conocimientos del área de aplicación. De esta forma, la ciencia de datos está en el nuevo panorama de investigación de mercados [1].

Todas estas nuevas tendencias no tienen ningún sentido si no se aplican con un propósito, este propositito por lo general es el de conocer a los clientes y encontrar patrones en ellos. Siendo un poco más específicos, a las empresas les interesa conocer cuál de sus clientes actuales continuara adquiriendo un producto o servicio, o de los clientes que ya dejaron de adquirirlo, cual es más propenso a que vuelva a adquirirlo. Si esta estimación se logra hacer de la manera más precisa posible, esta información pudiera ser muy útil para diseñar y generar estrategias de marketing más efectivas.

1.2. Justificación

En los últimos años, las empresas han comenzado a recurrir a la ciencia de datos y técnicas de *machine learning* para generar mejores estrategias de marketing, sin embargo, esto no es una tarea fácil ya que no solo implica aplicar un algoritmo a un conjunto de datos. Existen diversos factores que deben ser analizados antes de utilizar cualquier algoritmo ya que pueden afectar los resultados.

Entre estos factores se encuentran la correcta selección de las variables que alimentaran al algoritmo, la distribución y balanceo del conjunto de datos. Estos puntos son muy importantes ya que si no se toman en cuenta se corre el riesgo de que nuestro algoritmo no pueda encontrar patrones en los datos y por consiguiente el modelo generado no se desempeñe de la mejor forma, causando falsos positivos o falsos negativos.

Contar con herramientas que ayuden a analizar y resolver estos problemas sería de gran utilidad ya que ayudarían a simplificar el proceso y a minimizar el margen de error, dando como resultado modelos que puedan predecir o categorizar de una forma más precisa.

1.3. Problema

Se ha demostrado que utilizar ofertas a la medida para hacer que un cliente regrese es una de las mejores soluciones, para esto se requiere analizar el comportamiento de los clientes. Sin embargo, esto se vuelve más complejo para compañías grandes que ofrecen una amplia gama de servicios [4].

Por otro lado, las empresas tienen que optimizar sus costos de tal forma que las estrategias de marketing deben ser lo más efectivas o llegar a los clientes potenciales con las más altas probabilidades de integrarse como futuros clientes.

Aplicar un algoritmo de *machine learning* a un conjunto de datos no garantiza lograr esta efectividad. En muchas ocasiones el desempeño de los algoritmos se ve afectado por la distribución de los datos. Por ejemplo, si un conjunto de datos no está balanceado, es decir que cuenta significativamente con más elementos pertenecientes a una misma clase (clase dominante), el algoritmo de clasificación será muy bueno clasificando elementos que perteneces a la clase dominante, pero tendrá problemas para clasificar la clase no dominante.

Otro elemento que puede afectar el desempeño de un algoritmo es la selección de las variables independientes. Puede haber variables que solo introducen ruido y reducen la exactitud y precisión de un algoritmo.

1.4. Objetivos

1.4.1. Objetivo General:

Proponer e implementar estrategias que permitan incrementar la precisión y el performance de un algoritmo de clasificación en *machine learning*, aplicado a un problema de marketing.

1.4.2. Objetivos Específicos

- Implementar estrategias que permitan obtener un mejor resultado cuando se trabaja con conjuntos de datos desbalanceados.
- Implementar estrategias para determinar que variables deben ser seleccionada para obtener un mejor resultado.

1.5. Novedad científica, tecnológica o aportación

Se diseñarán e implementarán arquitecturas específicas de *autoencoders* para mejorar el desempeño de un clasificador cuando las condiciones del conjunto de datos no son las más óptimas, esto es cuando se cuenta con clases desbalanceadas. Los *autoencoders* se utilizarán para generar nuevas características que permitan al clasificador a tener una mejor precisión.

2. ESTADO DEL ARTE O DE LA TÉCNICA

2.1. Lenguajes, herramientas y liberarías para *Machine learning*

El panorama de librerías y herramientas para *Machine learning* ha evolucionado mucho en los últimos años.

De acuerdo a [6] el lenguaje de programación R es el más usado para realizar análisis y ciencia de datos, sin embargo en los últimos años Python ha tenido una gran adaptación para este tipo de tareas.

Este incremento en el uso de Python no solo se debe a que es muy sencillo de aprender, implementar y mantener, sino que además es bien sabido que se adapta muy bien para la realización de análisis de datos y a que es muy versátil en el sentido de que se puede usar para desarrollar desde prototipos básicos para pruebas hasta aplicaciones más complejas [5].

2.2. Librerías y Frameworks de *Machine learning* para python

2.2.1. SciKit-Learn

Una de las librerías que está teniendo mucha popularidad den los últimos años es "scikit-learn" [7], esta es una librería que está desarrollada sobre SciPy, un stack de librerías open source destinadas específicamente para computación científica. Algunas librerías que se encuentran en el core de SciPy son NumPy, Matplotlib and pandas.

La popularidad de esta librería se debe principalmente a su madures y a la gran cantidad de algoritmos implementados destinados a realizar diversos tipos de análisis y para diferente cantidad de datos con los que se está trabajando.

Sin embargo, una de las principales desventajas frente a otras librerías es el hecho de no estar diseñada para correr de manera distribuida, sino que está diseñada para procesar los en memoria. Esto representa una limitante cuando se requiere analizar una cantidad masiva de datos.

Para realizar el preprocesamiento de los datos de utiliza la librería pandas, que viene incluida en el core de SciPy.

2.2.2. Apache Spark

Apache Spark es un motor de procesamiento de datos distribuidos basado en Java Virtual Machine [8]. Utiliza un método de procesamiento distribuido en memoria, lo que hace que sea uno de los más rápidos. [9]. Al ser un sistema distribuido es apropiado para el análisis de datos a gran escala.

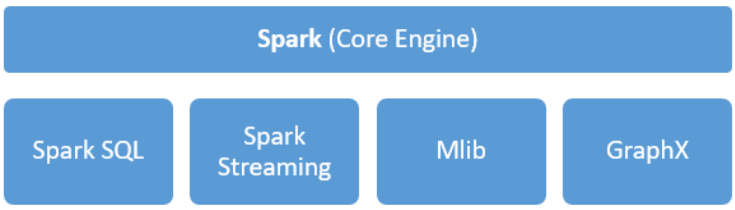


Figura 1. Apache Spark Librerias [9]

Como se puede observar en la Figura 1, Apache Spark cuenta con una plataforma unificada que consiste del motor principal y cuatro librerías, una de ellas para *Machine learning*.

La librería Spark SQL nos permite realizar tareas de preparación de los datos antes de aplicar los algoritmos de *machine learning*. Considerando que estas taras toman entre el 70% y 80% del tiempo, contar con esta librería que aproveche el poder de cómputo distribuido es muy útil.

2.2.3. GraphLab/Turi Distributed

Es un producto comercial, que similar a Apache Spark, es posible aprovechar el poder del cómputo distribuido para la ejecución de tareas de *Machine learning*.

Al igual que Apache Spark, GraphLab cuenta con herramientas para la manipulación de los datos antes de ser analizados.

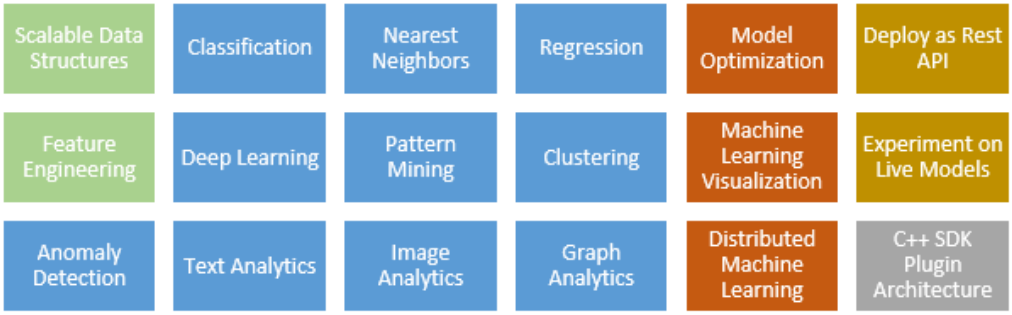


Figura 2. GraphLab Framework [10]

En la Figura 2 se puede observar a grandes rasgos los componentes de GraphLab framework, se pueden observar los diferentes algoritmos que maneja el framework (rojo), así como las herramientas para la manipulación y transformación de datos (verde).

Este framework, al ser comercial, cuenta con una licencia de un año para uso académico [10].

2.3. Soluciones integrales para Marketing Machine Learning

2.3.1. Marketing Analytics Software

La compañía G-Stat Analytics [11] cuenta con una herramienta, G-STAT BRAINs™, el cual promete automatizar el proceso de *machine learning* para proveer información referente a tres aspectos:

- Recomendaciones a la medida para maximizar posibles ganancias.
- Retención de Clientes
- Valor de la vida del cliente

Dentro de la página web de la empresa, en la sección "Technology", mencionan brevemente algunas de las tecnologías que emplearon para el desarrollo de del producto, así como a los retos a los que se enfrentaron:

Para el procesado y modelado de los datos, y el análisis con los algoritmos de *Machine learning* se usó Apache Spark, incluyendo sus librerías Spark SQL, Spark Streaming y Mlib. Además, se utilizó la funcionalidad de in-memory para el desarrollo y procesado de los modelos.

Como ya fue mencionado con anterioridad, dado que Apache Spark tiene la capacidad de hacer el procesamiento de manera distribuida, es una muy buena opción para este tipo de escenario donde procesar cantidades masivas de datos.

Como parte de los retos que se tuvieron, se menciona que combinar Spark MLib y R para hacer el desarrollo no fue tan intuitivo y fácil como se esperaba, ya que se requiero que los miembros del equipo tuvieran un nivel de experiencia mayor lo que significo más tiempo de desarrollo.

2.4. Métodos para Churn Prediction

Churn Prediction es la tarea de determinar la probabilidad de que un cliente deje de consumir un producto o servicio. Conocer esta probabilidad es de gran utilidad para las empresas ya que ayudan a crear campañas de marketing mejor dirigidas para la retención de los clientes.

Existen diferentes modelos adecuados para modelar y predecir la probabilidad de que un cliente deje de consumir un producto o servicio (churn prediction), de los cuales los más usados son los modelos regresión y árboles de decisión. Liu en [9] hace una comparación entre los siguientes modelos: regresión lineal, regresión logística, arboles de decisión y bosques aleatorios (random forest). De los anteriores el que mejor resultados dio fue el modelo de bosques aleatorios.

Otro modelo ideal para churn prediction es el de Support Vector Machine. Xia y Jin en [12] compararon este método contra redes neuronales, arboles de decisión, regresiones logísticas y clasificador bayesiano ingenuo. En su investigación encontraron que este método cuenta con una mejor precisión. En la tabla 1 podemos observar los resultados de esta comparación.

Model type	*Accuracy rate*	*Hit rate*	*Coverage rate*	*Lift Coefficient*
SVM	0.9088	0.8333	0.4018	6.2186
ANN	0.8983	0.7538	0.3625	5.6256
Decision tree C4.5	0.8386	0.3869	0.3437	2.8876
Logistic regression	0.8716	0.6190	0.1160	4.6198
Naive bayesian classifier	0.8782	0.7142	0.1562	5.3305

Tabla 1. Comparación de modelos [12]

2.5. Técnicas para predecir Customer Lifetime Value

El Customer Lifetime Value se define como el valor de las futuras ganancias obtenidas de un cliente durante su relación con la empresa.

En [13] Tarun y Vadlamani analizaron diversas técnicas de *machine learning* para calcular este valor.

Las técnicas que evaluaron son:

- Support Vector Machines
- Additive Regression and K-Star
- Multilayer Perceptron
- Classification and Regression Trees (CART)

En los resultados obtenidos, podemos observar que el Multilayer Perceptro dio los mejores resultados al tener el menor porcentaje de error. Sin embargo, mencionan que CART pudiera ser mal útil ya que al clasificar a los clientes en diferentes segmentos se pudieran analizar las reglas para dicha clasificación y así entender mejor a cada segmento de clientes.

2.6. Pre-procesamiento y limpieza de datos

Apache Spark cuenta con las utilidades necesarias para realizar la limpieza y preparación de los datos antes de iniciar el análisis de estos, dentro de las tareas de preparación de encuentran [9]:[9]:

- Accessing and loading *dataset*s
- Data cleaning
- Identity matching
- Data reorganizing
- Joining data
- Feature extraction

La librería Pandas, también proporciona los elementos necesarios para realizar manipulaciones y limpieza de datos, sin embargo, al ser una librería que trabaja con datos en memoria, está limitada por la

cantidad de memoria disponible en la computadora. A diferencia de Apache Spark, que tiene la capacidad de utilizar un cluster para realizar el procesado de los datos de manera distribuida.

2.7. Evaluación de los Modelos

Existen diferentes métricas que nos pueden ayudar a evaluar que tan bien se está comportando el modelo creado. Algunas de estas métricas son:

- Matriz de confusión: Es una matriz en la cual cada columna representa el número de predicciones hechas para cada clase, y las filas las indican la clase real. Este método nos permite observar rápidamente si nuestro modelo está produciendo falsos positivos o falsos negativos.
- Precisión: Mide el porcentaje de las predicciones positivas que fueron acertadas. Esta dada por la relación de los verdaderos positivos entre en total de predicciones positivas.
- Recall: Es el porcentaje de verdaderos que fueron correctamente predichos. Es la relación de los verdaderos positivos entre el total de verdaderos (falsos positivos y verdaderos positivos).
- F1 score: Es el promedio harmónico de precisión y recall. Un valor cercano a 1 representa un mejor desempeño del modelo.
- Curva ROC: Ayuda a evaluar a exactitud de un modelo independientemente de cualquier limite. Es una representación gráfica de la razón de verdaderos positivos contra la razón de falso positivos. Mientras la curva esté más cerca de la esquina superior izquierda, mejores son las predicciones del modelo.

2.8. Métodos para Clases Desbalanceadas

De acuerdo [14] existen diferentes métodos para manjar el problema del desbalanceo de clases en un conjunto de datos. Dentro de estos métodos, hay algunos que están relacionados con los datos y otros con el algoritmo usado para hacer el entrenamiento.

2.8.1. Métricas de evaluación apropiadas.

Como se menciona [18], la exactitud (accuracy) pone más peso en la clase más común, por lo que hace más difícil evaluar el desempeño al clasificar la clase minoritaria. La curva ROC y el área bajo la curva, AUC por sus siglas en inglés, resultan ser algunas de las métricas más apropiada para la evaluación de un clasificador cuando se trata de un conjunto de datos desbalanceado, ya que estas no ponen más énfasis en una clase sobre la otra, evaluando el desempeño en general de la clasificación [14][18].

2.8.2. Muestreo.

Es una de las técnicas más usadas para eliminar la rareza en los datos. Se basa en eliminar elementes de la clase mayoritaria o generar nuevos elementos que pertenezcan a la clase minoritaria, su forma más simple, duplicando los elementos de la clase minoritaria.

Esta técnica puede tener sus desventajas, al eliminar elementos de la clase mayoritaria se corre el riesgo de eliminar elementos clave. Generar nuevos elementos de la clase minoritaria duplicándolos puede llevar a sobrealimentar el algoritmo.

Existen otros métodos más avanzados de muestreo que intentan eliminar estos problemas, por ejemplo SMOTE [15], que se basa en aplicar operación a los datos existentes para generar nuevos elementos.[15], que se basa en aplicar operación a los datos existentes para generar nuevos elementos.

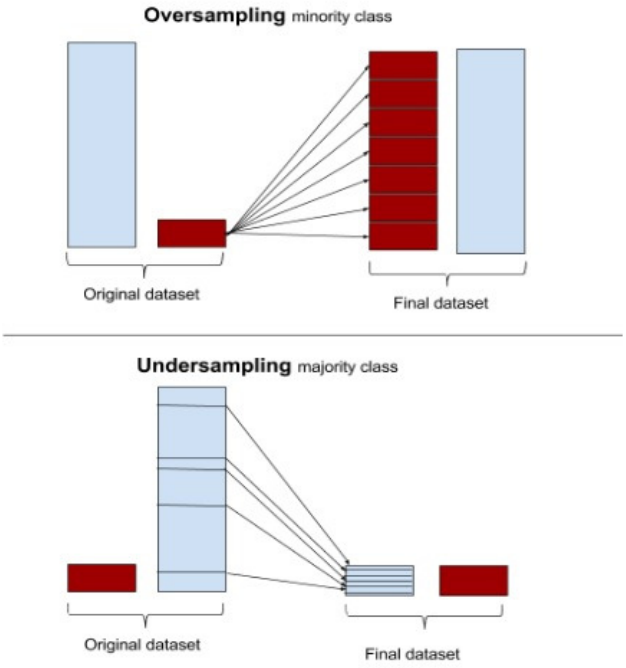


Figura 3. Muestreo [24]

2.8.3. Cost-sensitive learning.

Esto se basa en asignar un costo mayor a los falsos negativos, esto incrementa el rendimiento con respecto a la clase positiva. [18]

2.8.4. Boosting.

Es una técnica para mejorar la precisión de una función predictiva aplicando la función repetidamente en una serie y combinando la salida de cada función con la ponderación para que el error total de la predicción se minimice.
En [16] se analizó el uso de ensemble methods, incluyendo Boosting, para la mejorar precisión de customer choice, obteniendo mejores resultados.

2.9. Experimentación

Se realizó una comparación de diferentes algoritmos de *machine learning* para predecir la probabilidad de que un cliente contrate un producto bancario. El *dataset* fue obtenido de [17]. Se realizó un pre-procesamiento básico a los datos antes de aplicar los algoritmos, se removieron columnas que se consideraron innecesarias, se removieron elementos que contenían valores nulos y se transformaron las variables categorías a numéricas. Cabe mencionar que después de este pre-procesamiento, el *dataset* quedo desbalanceado, contando con 6056 elementos pertenecientes a la clase 0 y 1786 pertenecientes a la clase 1.

Estos son los resultados:

Algoritmo	**Crossvalidation (3 splits)**	**Validation**				
	Accuracy	Accuracy	Precision	Recall	f1-score	auc
Logistic Regression	0.821	0.806	0.633	0.376	0.471	0.793
Gradient Boosting Classifier	0.829	0.818	0.653	0.448	0.531	0.833
Support vector machines	0.827	0.82	0.647	0.486	0.555	0.771
Random forest	0.815	0.806	0.634	0.406	0.495	0.783
K-nearest-neighbors	0.803	0.802	0.595	0.45	0.513	0.756
Bagging Classifier	0.807	0.812	0.636	0.414	0.502	0.784

Se aplicó la técnica SMOTE [15] para balancear el *dataset* y tratar de obtener mejores resultados. Después de aplicar esta técnica, la estructura del *dataset* quedo de la siguiente manera, 6056 elementos pertenecientes a la clase 0 y 5925 pertenecientes a la clase 1.

Algoritmo	**Crossvalidation (3 splits)**	**Validation**				
	Accuracy	Accuracy	Precision	Recall	f1-score	auc
Logistic Regression	0.746	0.758	0.782	0.712	0.745	0.818
Gradient Boosting Classifier	0.867	0.862	0.891	0.824	0.856	0.937
Support vector machines	0.78	0.8	0.819	0.767	0.792	0.864
Random forest	0.861	0.878	0.902	0.841	0.871	0.942
K-nearest-neighbors	0.81	0.838	0.796	0.905	0.847	0.907
Bagging Classifier	0.861	0.863	0.889	0.836	0.862	0.935

Como se puede observar, el algoritmo *Gradient Boosting Classifier* fue el que mejores resultados dio en ambos escenarios.

2.10. *Autoencoders* para manejar clases desbalanceadas

Como se explica en el siguiente capítulo, un *autoencoder* es capaz de encontrar un set de nuevas características que representen el set original.

Situando un *autoencoder* en el contexto del problema de clases desbalanceada, un *autoencoder* puede ser usado para generar características que nos proporcionen una mejor capacidad de clasificación hacia la clase minoritaria [20]. Así mismo, dada su habilidad de aprender nuevas características, un *autoencoder* puede ser utilizado para reducir la dimencionalidad de un conjunto de datos. Esto se lograría usando la salida de la capa de codificación como nuestro nuevo conjunto de datos. [22][20]. Así mismo, dada su habilidad de aprender nuevas características, un *autoencoder* puede ser utilizado para reducir la dimensionalidad de un conjunto de datos. Esto se lograría usando la salida de la capa de codificación como nuestro nuevo conjunto de datos.

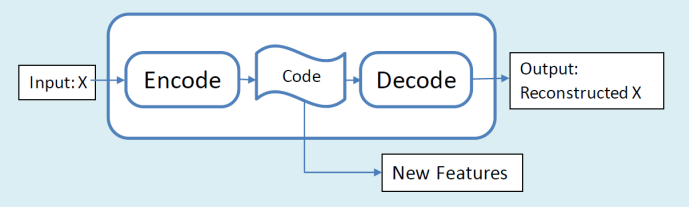


Figura 4. *Autoencoder* para generar nuevas características

En [20] y [21] se utilizaron *autoencoders* para resolver el problema de clases desbalanceadas, en ambos casos de utilizaron dual *autoencoders*, con la diferencia de que en la segunda se utilizaron *denoising autoencoders.*

3. MARCO TEÓRICO/CONCEPTUAL

Resumen: *En este capítulo se presentan las bases teóricas y conceptuales sobre autoencoders y selección de características.*

3.1. Predicción de Propensión de Compra.

A menudo, trabajar en este tipo de problemas implica lidiar con un conjunto de datos con clases no balanceadas. Un claro ejemplo de esta situación se puede observar en el conjunto de datos obtenido de [17]. Este conjunto de datos describe la posibilidad de que un cliente contrate un producto bancario. En la tabla 2, se muestra que de los 45,211 registros con los que se cuentas cual es la proporción de clientes que adquirieron el producto ofrecido.

Tabla 2. Propensión de compra de un producto bancario

Adquirieron el Producto (Clase Positiva)	**Declinaron la Compra (Clase Negativa)**
5,289 **(11.69%)**	39,922 (88.31 %)

Claramente se puede observar el desbalanceo de las clases, ya que el porcentaje de clientes que en realidad estarían interesados en adquirir un nuevo producto o servicio es muy bajo en comparación al al porcentaje de clientes que rechazarían el producto.

Más adelante, en el siguiente capítulo, se describirá más a fondo este *dataset*.

Como se ha demostrado en [20] y [21] los *autoencoders* pueden ser utilizados para generar nuevas características que cuenten con una mejor capacidad de clasificación aun en casos cuando se tienen clases desbalanceadas.

En [20] se utilizó un dos *stacked autoecoders* para generar nuevas características con mejores capacidades de clasificación. Uno de estos *autoencoders* utilizo una función de activación *sigmoid*, mientras que en el otro de utilizo *tanh*. Al combinar las nuevas características generadas por cada uno de estos *autoencoders*, se obtuvieron mejore resultados al clasificar dataset con clases desbalanceadas. En la Figura 5 se puede observar que con las características generadas por los *autoencoders* (DAF), se obtuvieron mejores resultados que utilizando otros métodos.

Methods	AUC	F_1-score	G-Mean
RF	0.979 ± 0.022	0.891 ± 0.095&	0.915 ± 0.075*
ROS	0.978 ± 0.023	0.896 ± 0.087@	0.925 ± 0.064
RUS	0.976 ± 0.0236&	0.857 ± 0.116*	0.931 ± 0.044
IRUS	0.978 ± 0.023	0.777 ± 0.171*	0.907 ± 0.046*
SMOTE	0.978 ± 0.022	0.895 ± 0.088#	0.925 ± 0.061
RUSBoost	0.963 ± 0.033*	0.836 ± 0.113*	0.916 ± 0.050*
PCA	0.803 ± 0.096*	0.707 ± 0.163*	0.790 ± 0.111*
DPCA	0.830 ± 0.088*	0.753 ± 0.131*	0.822 ± 0.096*
DAF	0.980 ± 0.024	0.905 ± 0.090	0.928 ± 0.070

Figura 5. Comparativa de resultados utilizando *autoencoders*

A su vez en [21], se utilizó un acercamiento muy similar al utilizado en [20], con la diferencia de que utilizaron *denoising autoencocers*. De igual manera, se combinaron las características generadas por dos *stacked autoencoders*, uno utilizando la función de activación *sigmoid* y el otro *tanh*. En la tabla 3, se puede observar que utilizar las características generadas por el *denoising autoencoder* fue posible obtener mejores resultados.

Tabla 3 Comparativa de resultados utilizando DDAF

Metrics	Methods		
	RUSBoost	***DAF***	***DDAF***
ACC	0.8110±0.1111	0.9596±0.0351	**0.9634±0.0312**
AUC	0.8443±0.0964	0.9765±0.0242	**0.9883±0.0229**
F_1-score	0.8480±0.0932	0.9691±0.0325	**0.9844±0.0191**
G-Mean	0.8351±0.1051	0.9659±0.0250	**0.9670±0.0230**

3.2. *Autoencoder*

Un *autoencoder* es una red neuronal que es entrenada de tal forma que su salida sea lo más aproximada a la entrada [19]. El *autoencoder* consiste en dos partes, una función de codificación, y una función de decodificación que produce una reconstrucción de la entrada a partir del código generado por la función de codificación.

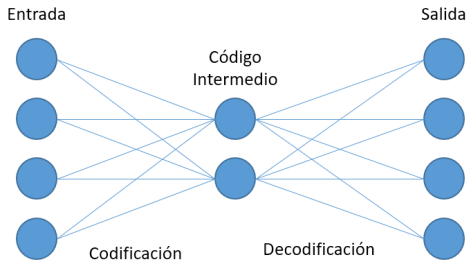


Figura 6. Estructura básica de una *autoencoder*

Las funciones de codificación y decodificación pueden ser definidas como:

$$f(x) = s_e(W_e x + b_e) \tag{1}$$

$$g(x) = s_d(W_d x + b_d) \tag{2}$$

La salida del *autoencoder* sería entonces:

$$y = g(f(x)) \tag{3}$$

El objetivo del *autoencoder* es aprender una representación de características en la capa de codificación de modo que las salidas del *autoencoder* reconstruyan las entradas. Por lo tanto, el problema de aprendizaje del *autoencoder* es encontrar un conjunto de parámetros para minimizar el error de reconstrucción entre las entradas y las salidas del *autoencoder* [20].

3.3. *Autoencoder* Incompleto

Un *autoencoder* incompleto es aquel en el cual el código intermedio tiene una dimensión menor a la dimensión de la entrada [19]. Generando una representación de menor dimensión ayuda a que esta representación capture las características más importantes del conjunto de datos.

Cuando un autencoder se utiliza con funciones lineares y la función de costo es el promedio del cuadrado de los errores, este tipo de *autoencoder* puede generar nuevas características con propiedades similares a las que se pudieras generar con un análisis de componentes principales o PCA, por sus siglas en ingles. [19]

Cuando una función no linear es usada, el *autoencoder* puede generar una representación más útil que la generada utilizando PCA.

3.4. *Denoising Autoencoder*

Este tipo de *autoencoder* recibe como entrada una versión de los datos corrompidos, y el *autoencoder* es entrenado para generar los datos originales. [19]

En la figura 7 se pude observar el diagrama general de este tipo de *autoencoder*, donde $C(\tilde{x}|x)$ es el proceso de corrupción de datos donde se introduce el ruido. "f" y "y" son las funciones de codificación y decodificación, respectivamente, h es el código intermedio y L es la función de costo, la cual es minimizada.

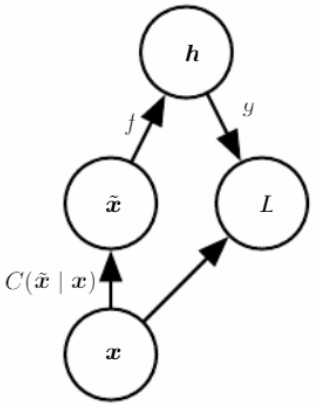


Figura 7. ***Denoising Autoencoder***

El proceso de corrupción de los datos se da al descartar aleatoriamente algunos valores de entrada de muestras seleccionadas aleatoriamente de conjunto de datos de entrenamiento. Estas pequeñas modificación o ruido introducido en el conjunto de datos de entrenamiento hacen que el conjunto se datos sea más parecido a un conjunto de datos de pruebas más real, incrementando las robustez del *autoencoder* y haciendo que tenga un mejor desempeño con nuevos datos futuros [21].

Similar a un *autoencoder* tradicional, las funciones de codificación y decodificación se definen como:

$$y = f(\tilde{x}) = S_e(W_e\tilde{x} + b_e) \tag{4}$$

$$z = g(y) = S_d(W_d y + b_d) \tag{5}$$

Donde $\tilde{x}$ es el conjunto de datos corrompido.

3.5. *Stacked Autoencoder*

Anteriormente se describió la arquitectura básica de un *autoencoder*, la cual consiste de una capa de codificación una de decodificación. Sin embargo, es posible crear una arquitectura más compleja al apilar dos *autoendoders*, esto con el propósito de mejorar la capacidad del *autoencoder* de generar nuevas características que representen de una mejor manera los datos de entrada [20].

En este tipo de *autoencoder*, la salida de la capa de codificación, es decir el código intermedio, es la entrada del segundo *autoencoder*. Y el código intermedio de este segundo *autoencoder* es la representación final de los datos originales.

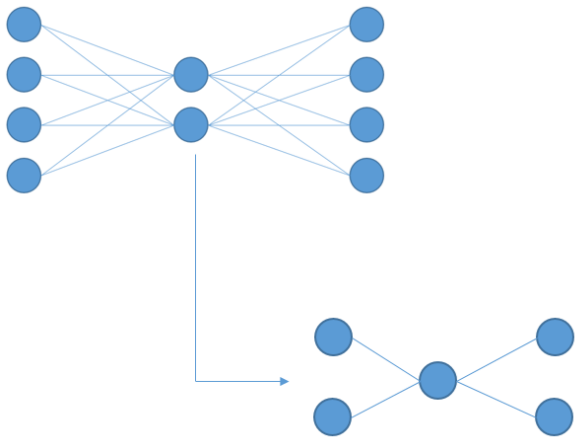


Figura 8. ***Stacked Autoencoder***

4. DESARROLLO METODOLÓGICO

Resumen: *En este capítulo se presenta en detalle el desarrollo metodológico que incluye el proceso de análisis de los datos y de los autoencoders utilizados.*

4.1. Proceso de Análisis

Se analizará información sobre los clientes y su historial de compras para predecir si continuaran adquiriendo un producto o si estarían dispuestos a adquirir un nuevo producto.

El análisis de los datos se realiza de acuerdo con el proceso descrito a continuación:

El punto de interés se encuentra en el pre-procesamiento de los datos, que es donde se utilizaran los *autoencoders* para mitigar los problemas causados por el desbalanceo de las clases.

4.2. *Datasets*

4.2.1. Dataset 1: Bank Marketing

El primero *dataset* utilizado, obtenido de [17], describe los resultados obtenidos de una campaña de marketing directo para un banco portugués. Esta campaña ofrecía a los clientes la adquisición de un fondo de inversión. El *dataset* contiene información sobre los clientes a los cuales se contactó y si estos adquirieron o no el producto.

El dataset contiene columnas las cuales describen información demográfica de los clientes contactados. Así mismo, también se cuenta con algunas columnas las cuales indican si el cliente cuenta con algún otro producto bancario (préstamo personal, crédito hipotecario).

En la tabla 4 se hace un desglose de las columnas que contienen los datos demográficos y de historial crediticio que contiene el *dataset* así como el tipo de dato de cada una.

En la tabla 5 se describen los datos relacionados a la última vez que se contactó a cliente para ofrecerle le product.

La tabla 6 describe los datos referentes a compañas de *marketing* anteriores.

Tabla 4. Datos demográficos y de historia crediticios

Columna	Tipo de Dato
Edad	Numérico
Tipo de trabajo	Categórico (admin, blue-collar, entrepreneur, housemaid, management, retired, self-employed, services, student , technician , unemployed, unknown)
Estado Marital	Categórico (divorced , married , single , unknown)
Nivel de Estudios	Categórico (basic.4y, basic.6y, basic.9y, high.school, illiterate, professional.course, university.degree, unknown)
Default (Falta de pagos)	Categórico (no , yes, unknown)
Balance	Numérico (Promedio anual en la cuenta)
Crédito Hipotecario	Categórico (no, yes, unknown)
Crédito Personal	Categórico (no, yes, unknown)

Tabla 5. Ultimo contacto

Columna	Tipo de Dato
Vía de contacto	Categórico (cellular, telephone)
Mes	Categórico (jan, feb, mar, ..., nov, dec)
Día	Numérico
Duración (en segundos)	Numérico

Tabla 6. Otros datos

Columna	Tipo de Dato
Numero de intentos de contacto.	**Numérico**
Días desde último contacto para una campaña anterior.	**Numérico**

Numero de intentos de contacto para campañas anteriores.	**Numérico**
Resultado de campañas anteriores.	**Categórico (failure, nonexistent, success)**

Este *dataset* es perfecto para analizar el problema de las clases desbalanceadas, ya que de los 41188 registros con los que cuenta, solo 4640 adquirieron el producto, esto deja un desbalanceo muy marcado. Si el propósito es predecir si un futuro cliente adquirirá el producto, se complica realizar la predicción ya que los elementos pertenecientes a esta clase son muy escasos comparados con los que no adquieran el producto.

En tabla 7 y 8 se muestra un extracto de los datos utilizados.

Tabla 7. Extracto de datos 1

age	job	marital	education	default	housing	loan
30	unemployed	married	primary	no	1787	no
33	services	married	secondary	no	4789	yes
35	management	single	tertiary	no	1350	yes
30	management	married	tertiary	no	1476	yes
59	blue-collar	married	secondary	no	0	yes
35	management	single	tertiary	no	747	No
30	unemployed	married	primary	no	1787	no

Tabla 8. Extracto de datos 2

contact	day	month	duration	campaign	pdays	previous	poutcome
cellular	19	oct	79	1	-1	0	unknown
cellular	11	may	220	1	339	4	failure
cellular	16	apr	185	1	330	1	failure
unknown	3	jun	199	4	-1	0	unknown
unknown	5	may	226	1	-1	0	unknown
cellular	23	feb	141	2	176	3	failure
cellular	19	oct	79	1	-1	0	unknown

4.2.2. *Dataste* 2: Historial De Compra de Clientes

El segundo *dataset* fue proporcionado por una empresa comerciante. Debido a cuestiones de confidencialidad con la empresa que proporciono el *dataset,* se omiten detalles específicos sobe la información que este contiene. A grandes rasgos, este *dataset* consiste del historial de 5 años de las compras realizadas por sus clientes.

Lo que nos interesa conocer sobre en este escenario, es si un cliente determinado realizara una compra de algún producto. Dado que se cuenta con el historial de compra se sabe en qué meses realizo alguna compra y en cuáles no.

Para predecir si un cliente adquirirá un producto el mes entrante, se necesita realizar una transformación a los datos original, la cual se describe a continuación:

$$y\{0,1\}_t = F(X_{t-1:t-N}) \tag{6}$$

$$t = month \tag{7}$$

$$x^0 = customer\ info,\ x^1 = Number\ of\ product\ bought,\ x^2 = amount\ spent,\ x^3 = Number\ of\ different\ products,\ x^4 = month, \ldots \tag{8}$$

$$x_{t=-1} = [x^0, x^1, x^2, x^3, x^4, \ldots] \tag{9}$$

$$y\{0,1\}_0 = F(X_{t-1}, X_{t-2}, \ldots, X_{t-12}) \tag{10}$$

Así, para cada cliente se tendrá:

$$\begin{bmatrix} y_1|X_1 \\ y_2|X_2 \\ y_3|X_3 \\ \ldots \\ y_{12}|X_{12} \end{bmatrix} \tag{11}$$

Después de esta transformación se cuenta con un *dataset* el cual tiene la estructura presentada en la tabla 9. Donde la columna 36 representa un mes dato y la 37 si se realizó alguna compra. 1indica que si se realizó una compra y 0 que no se realizó compra. Las columnas de la 0 a la 35 se agrupan de tres en tres y representan lo siguiente, la cantidad de producto diferentes comprados, la cantidad total de productos y el mes de la compra, teniendo en total el historial de compra de doce meses atrás.

Tabla 9. Extracto de dataset de historial de compra

	0	1	2	3	4	5	6	7	8	9	...	31	32	33	34	35	36	37
0	1	1	5	1	1	4	1	1	3	0	...	0	7	0	0	6	6	1
1	1	1	4	1	1	3	0	0	2	0	...	0	6	0	0	5	5	1
2	1	1	3	0	0	2	0	0	1	0	...	0	5	0	0	4	4	1
3	0	0	2	0	0	1	0	0	12	0	...	0	4	0	0	3	3	1
4	0	0	1	0	0	12	0	0	11	0	...	0	3	0	0	2	2	0
5	0	0	12	0	0	11	0	0	10	0	...	0	2	0	0	1	1	0

4.3. Pre Procesamiento y Limpieza de Datos

El conjunto de datos que requirió de pre procesamiento y limpieza de los datos fue el relacionado al banco portugués, ya que muchas de las columnas son categóricas.

A cada valor de cada columna categoría se le asignó un valor numérico. Por ejemplo, la columna "marital", puede contener los siguientes valores '*divorced'*, '*married'*, '*single' y 'unknown'*. Estos valores se sustituyeron por números del 0 al 3, respectivamente. Esto es un paso importante ya que los algoritmos trabajan con valores numéricos. Para esta tarea se utilizó la librería LabelEncoder perteneciente a *SciKit Learn*.

Así mismo, todos los datos fueron pasados por un proceso de normalización, esto es que todos los valores tengan una escala en común.

Una de las columnas que se eliminó del *dataset,* es la de duración ya que esta no se conoce antes de realizar una llamada para ofrecer el producto a un cliente.

El segundo dataset, relacionado con el historial de compra, se creó un script en Python el cual recibe como entrada el dataset original y a la salida se obtiene el *dataset* descrito en la tabla 9.

4.4. Generación de Nuevas Características

Como se mencionó en el capítulo pasado, se utilizaron *autoencoders* para la generación de nuevas características y así ayudar con el problema del desbalanceo de clases.

Se utilizaron dos *autoencoders* con la estructura mostrada en la figura 9 para para generación de nuevas características. Uno de ellos con la función de activación sigmoid y el otro con la función de activación tanh, esto resulto en 8 nuevas características.

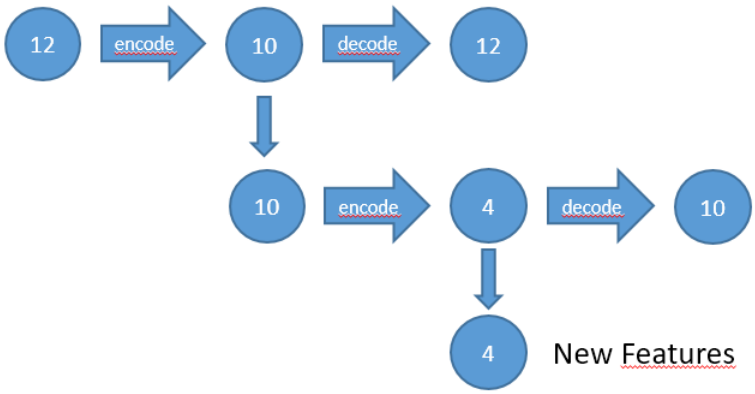


Figura 9. ***Stacked Autoencoder para el dataset de Bank Marketing***

Como se mencionó anteriormente en 3.3, donde se describe un *autoencoder* incompleto, cuando se tiene un código intermedio con una dimensión menor a los datos de entrada, el *autocoder* puede aprender a generar nuevas características que representen de una mejor manera a los datos de entrada. Esto por eso que el primer *autoencoder* reduce el número de entradas a 10 y a su vez el segundo *autoencoder* a 4, siendo estas últimas las que potencialmente conjugan las características más importantes de los datos de entrada.

Para la creación de este *stacked autoencoder* se utilizó la librería obtenida de [25], la cual contiene clases para la implementación de *autoencoders*.

Siguiendo la misma lógica, para el dataset del historial de compra en el cual predeciremos si un cliente comprara algún producto el mes entrante, el *autoencoder* se diseñó de la manera mostrada en la figura 10. De igual manera, las capas intermedias, de las cuales se obtiene el código intermedio, se mantuvieron de una dimensión menor.

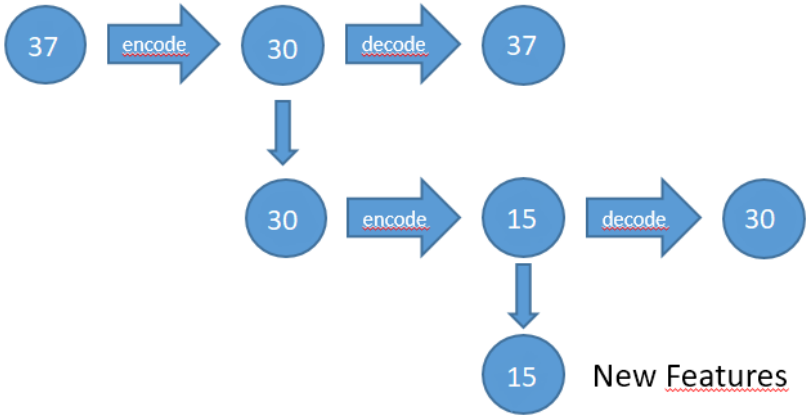


Figura 10. ***Stacked Autoencoder para el dataset de Historial del Compras***

4.5. Validación

La validación se realizará utilizando el clasificador *Balanced Bagging Classifie*, este es un clasificador que toma en cuenta el desbalanceo de clases en un dataset. Se harán dos pruebas, en la primera se entrena el clasificador con el *dataset* original y en la segunda se entrena utilizando las nuevas características generadas por el *autoencoder*. El mismo proceso se realizará para ambos *datasets*, con sus respectivos *autoencoders*.

El flujo de los datos será como se muestra en la figura 11.

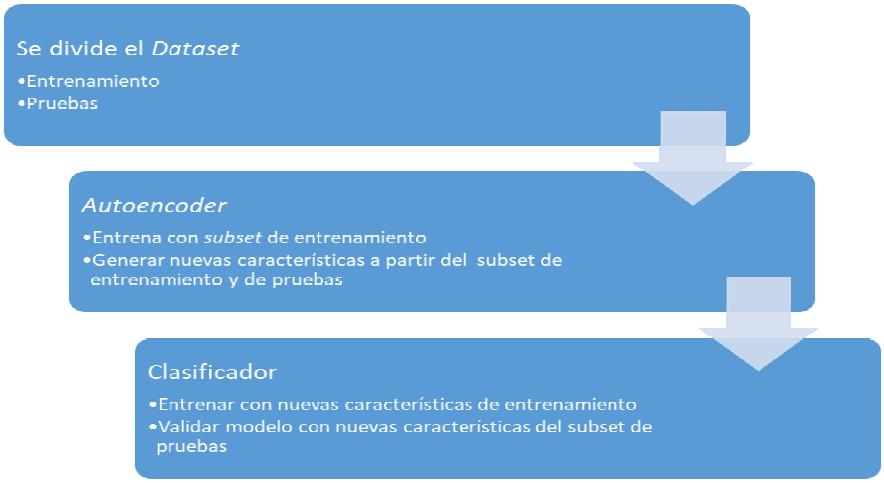


Figura 11. Flujo de datos

El principal método usado para realizar la validación de los modelos fueron las matrices de confusión. Este método ofrece de manera visual un resumen rápido del desempeño de los resultados. Además de presentar de manera visual los resultados, a partir de esta es posible calcular otras métricas tales como la precisión, el F1-score, entre otros.

En la matriz de confusión se pueden observar los siguientes valores: la cantidad de falsos verdaderos, falsos positivos, ciertos negativos y ciertos verdaderos.

	Predicho verdadero	Predicho negativo
verdadero	Verdadero positivo (TP)	Falso positivo (FP)
negativo	Falso verdadero (FN)	Verdadero negativo (TN)

A partir de estos valores es posible calcular las siguientes métricas:

Accuracy: $AAC = \frac{TP+TN}{P+N}$

Precision: $PPV = \frac{TP}{TP+TF}$

Recall: $TPR = \frac{TP}{p}$

F1 Score: $F1 = 2(\frac{PPV \times TPR}{PPV+TPR})$

5. RESULTADOS Y DISCUSIÓN

Resumen: *En este capítulo se presentan los resultados obtenidos del desarrollo de este trabajo.*

5.1. Resultados

En la figura 11, presentada en el capítulo anterior, se describe a grandes rasgos el flujo que se siguió para este experimento. En primera instancia, el dataset se dividió en dos subconjuntos, uno de entrenamiento y otro para pruebas. El entrenamiento del *autoencoder* se alimentó con el conjunto de entrenamiento. Una vez terminado el entrenamiento del *autoencoder*, se generaron las nuevas características, tanto para el conjunto de entrenamiento como para el de pruebas.

El clasificador fue entrenado con el conjunto de pruebas, y después se hizo la predicción con el conjunto de pruebas. En paralelo, un segundo clasificador fue entrenado usando las nuevas características generadas por el *autoencoder*.

Este proceso se siguió para ambos *datasets*.

5.1.1. Dataset 1: Bank Marketing

En las figuras 12 y 13 se puede observar los resultados obtenidos al entrenar el clasificador con el *dataset* original, así como con las nuevas características generadas por el *autoencoder*, respectivamente. Se puede observar como el número de falsos positivos y falso negativos disminuyeron, dando pie a que el número de aciertos en ambas clases, positivas y negativas, incrementara. Así mismo en la tabla 10 se desglosan las métricas para cada caso, ahí se puede observar que todas ellas tuvieron una mejora derivada del incremento de aciertos.

Al usar las características generadas por el autoencoder, se incrementó en 1.5% el número de verdaderos positivos. La métrica F1-score incremento un 1.9%.

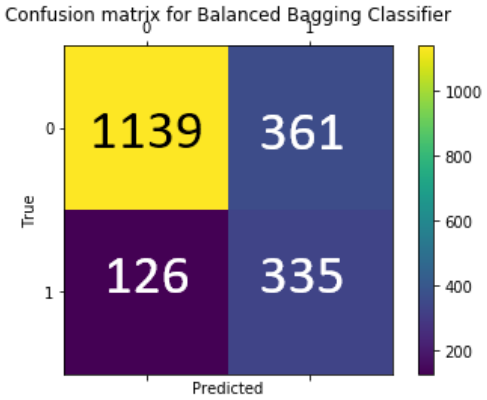


Figura 12. Matriz de confusión con *dataset* original

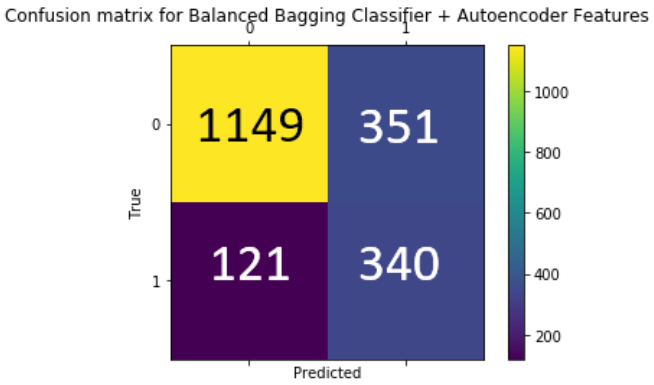


Figura 13. Matriz de confusión usando nuevas características

Tabla 10. Comparativa de resultados (Bank Marketing dataset)

	BALANCED BAGGING CLASSIFIER	***+ AUTOENCODER FEATURES***
ACCURACY	0.7517	**0.7593**
PRECISION	0.4813	**0.4920**
RECALL	0.7267	**0.7375**
F1-SCORE	0.5791	**0.5903**

5.1.2. *Dataset* 2: Historial De Compra de Clientes

En las figuras 14 y 15 se puede observar que incremento de verdaderos positivos fue mucho menos en este caso, siendo solo de 0.3%.

En la tabla 11, se refleja también la pequeña mejora del rendimiento del clasificador.

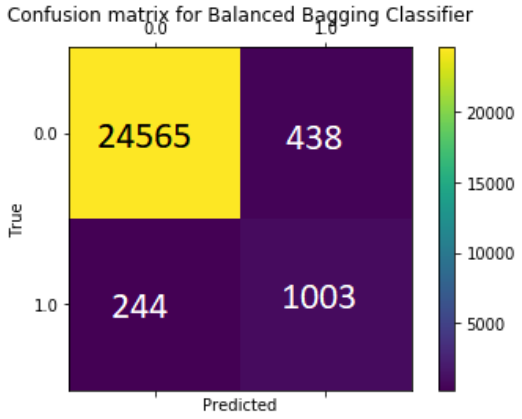


Figura 14. Matriz de confusión con *dataset* original

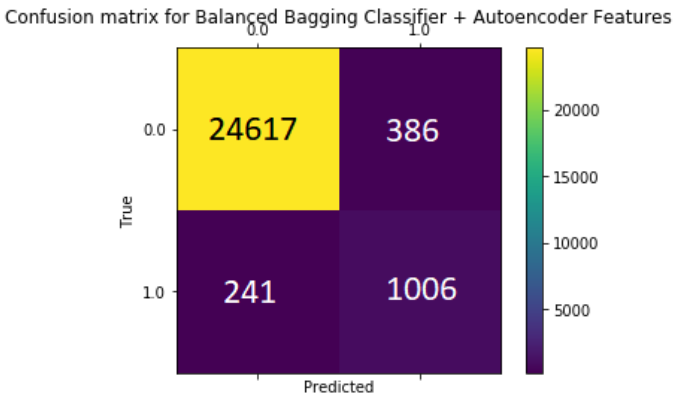


Figura 15. Matriz de confusión usando nuevas características

Tabla 11. Comparativa de resultados (Historial de Compra)

	BALANCED BAGGING CLASSIFIER	***+ AUTOENCODER FEATURES***
ACCURACY	0.974	**0.976**
PRECISION	0.696	**0.723**
RECALL	0.804	**0.807**
F1-SCORE	0.746	**0.762**

6. CONCLUSIONES

Resumen: *En este capítulo se presentan las conclusiones y trabajo futuro en relación con el uso de Autoencoders para mejorar el desempeño de un clasificador*.

6.1. Conclusiones

Como se puede observar en la sección anterior, donde se muestran los resultados, las nuevas características de los *Autoencoders* fueron útiles para incrementar la precisión del clasificador. Sin embargo, es necesario tener en cuenta que el *Autoencoder* debe ser adaptado a cada problema en específico, ya que su configuración y el número de características que será capaz de producir dependerá en gran medida de las propiedades del conjunto de datos al que se quiera adaptar. Así mismo se tiene que tomar en cuenta el tiempo requerido para entrenar el *Autoencoder*, ya que entre mayor sea la cantidad de registros y de características iniciales en nuestro conjunto de datos, mayor será el tiempo de entrenamiento.

6.2. Trabajo Futuro

Durante la definición del marco teórico de este trabajo se hizo la mención sobre algunos tipos de *autoencoders*, sin embargo, en el desarrollo de mismo solo se utilizaron *autoencoders* tradicionales, sería interesante poner a prueba si utilizando algún otro tipo de *autoencoder*, como el *denoising autoencoder*, se puede mejorar el resultado.

Así mismo, sería interesante explorar alguna metodología de *reinforcement learning*, para obtener de manera autónoma la configuración del *autoencoder* mas optima, es decir la que genera las características que mejor ayudan a incrementar la precisión del clasificador.

BIBLIOGRAFÍA

[1] Miller, Thomas W. Marketing data science: modeling techniques in predictive analytics with R and Python. FT Press, 2015.

[2] Matt Ariker, Jason Heller, Alejandro Diaz, Jesko Perrey. How Marketers can personalize at scale. Harvard Business Review, November, 2015.

[3] TRAVIS WRIGHT, CCP GLOBAL, Why data scientists and marketing technologists are the hottest jobs of 2015, VentureBeat, MARCH, 2015

[4] "Winning Back Lost Customers," Harvard Business Review, March, 2016.

[5] S. Gollapudi, Practical *Machine learning*. Packt Publishing, 2016.

[6] "R, Python Duel As Top Analytics, Data Science software – KDnuggets 2016 Software Poll Results." http://www.kdnuggets.com/2016/06/r-python-top-analytics-data-mining-data-science-software.html

[7] G. Hackeling, Mastering *Machine learning* with scikit-learn. Packt Publishing, 2014.

[8] R. Thottuvaikkatumana, *Apache Spark 2 for Beginners*. Packt Publishing, 2016.

[9] A. Liu, Apache Spark *Machine learning* Blueprints. Packt Publishing, 2016.

[10] "Fast, Scalable *Machine learning* Platform," *Turi*. [Online]. Available: https://turi.com. [Accessed: 28-Nov-2016].

[11] "G-STAT - big data analytics software." [Online]. Available: http://www.g-stat.com/. [Accessed: 28-Nov-2016].

[12] G. XIA and W. JIN, "RESEARCH PAPER: Model of Customer Churn Prediction on Support Vector Machine," *Systems Engineering - Theory & Practice Online*, vol. 28, pp. 71–77, Jan. 2008.

[13] J. Wang, "Customer Lifetime Value Measurement using *Machine learning* Techniques," in *Encyclopedia of Business Analytics and Optimization*, IGI Global, 2014.

[14] J. Burez and D. Van den Poel, "Handling class imbalance in customer churn prediction," *Expert Systems With Applications*, vol. 36, no. Part 1, pp. 4626–4636, Jan. 2009.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Jun. 2011.

[16] M. C. van Wezel and R. Potharst, "Improved customer choice predictions using ensemble methods," *Econometric Institute Research Papers*, 2005.

[17] "UCI *Machine learning* Repository: Bank Marketing Data Set." [Online]. Available: http://archive.ics.uci.edu/ml/*datasets*/Bank+Marketing. [Accessed: 18-May-2017].

[18] Weiss, Gary. (2004). Mining with rarity: A unifying framework. SIGKDD Explorations. 6. 7-19.

[19] Ian Goodfellow, Yoshua Bengio & Aaron Courville (2016). Deep Learning. MIT Press

[20] Ng, W.W., Pedrycz, W., Yeung, D.S., Zeng, G., & Zhang, J. (2016). Dual *autoencoders* features for imbalance classification problem. Pattern Recognition, 60, 875-889.

[21] Wang, T., Zeng, G., Yeung, D.S., & Li, J. (2017). Dual *Denoising Autoencoder* Features for Imbalance Classification Problems. 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 312-317.

[22] B. Chandra, Rajesh K. Sharma, Exploring *autoencoders* for unsupervised feature selection, International Joint Conference on Neural Networks (IJCNN). 2015

[23] "Choosing the right estimator — scikit-learn 0.19.1 documentation." [Online]. Available: http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html. [Accessed: 11-Jul-2018].

[24] "Learning from Imbalanced Classes," 25-Aug-2016. [Online]. Available: https://svds.com/learning-imbalanced-classes/. [Accessed: 26-Nov-2017].

[25] T. G. Smith, smrt: Handle class imbalance intelligently by using variational auto-encoders to generate synthetic observations of your minority class. 2018.