Detección Automática y Análisis de Competencia en Plataformas de Comercio Electrónico.

Ing. Gerardo Alberto López Vega
Instituto Tecnológico de Acapulco
Acapulco, México
mcgalv@gmail.com

MTI. Juan Miguel Hernández Bravo
Instituto Tecnológico de Acapulco
Acapulco, México
jmhernan@yahoo.com

Dr. José Antonio Montero Valverde
Instituto Tecnológico de Acapulco
Acapulco, México
jamontero1@infinitummail.com

Dr. Eduardo de la Cruz Gámez
Instituto Tecnológico de Acapulco
Acapulco, México
gamezeduardo@yahoo.com

Resumen *—— El presente trabajo expone el desarrollo de una aplicación de aprendizaje automático mediante la implementación de un clasificador bayesiano desarrollado en Python con ayuda de la librería Scikit-Learn[1] para la detección automática y análisis de competencia en plataformas de comercio electrónico. Se plantea el uso del algoritmo de Naive Bayes para clasificar las instancias presentes en el conjunto de datos de entrenamiento identificándolas con la clase "Es Competidor" tras evaluar las diferentes características de vendedores y productos publicados en la plataforma de ventas tales como el precio del producto, calificación del vendedor, tipo de publicación, tipo de envío y forma de pago. El modelo ha sido entrenado con un conjunto de datos obtenidos de la plataforma de comercio electrónico, los cuales se encontraban en formato JSON[2] por lo que antes de utilizarlo en el modelo pasaron por una etapa de preprocesamiento y fueron transformados a un formato CSV válido. Los resultados obtenidos después de haber entrenado el modelo fueron satisfactorios ya que se alcanzó una precisión del 96% para el algoritmo, prediciendo correctamente casi en su totalidad las instancias de prueba proporcionadas. Una vez que se validó el modelo, se desarrolló una aplicación integrada con la plataforma de comercio electrónica aplicando el modelo desarrollado, la cual ha sido capaz de identificar automáticamente a los competidores con lo que el usuario puede analizar la información obtenida de dichos vendedores.*

Palabras clave *—— clasificación, detección, aprendizaje automático, análisis competencia, preprocesamiento.*

I. INTRODUCCIÓN

De acuerdo con la Encuesta Nacional sobre Disponibilidad y Uso de Tecnologías de la Información en los Hogares, en los últimos años ha ido al alza el interés de los mexicanos por el comercio electrónico, y actualmente en el país se calcula que cerca de 20 por ciento de las personas que tienen acceso a Internet lo usa para ordenar o comparar productos (ENDUTIH, 2018).

El mundo moderno es sumamente competitivo y exigente, la industria actual requiere de soluciones integrales de automatización y control cada vez más avanzadas que les permitan a las empresas optimizar su productividad y los recursos de modo que puedan competir en un entorno globalizado.

II. DETECCIÓN Y ANALISIS COMPETENCIA

A. Competencia

De acuerdo con la comisión Federal de Competencia (COFESE), la definición de competencia significa rivalidad entre empresas que participan en un mercado aplicando sus mejores estrategias de manera que pueden minimizar sus costos, maximizar sus ganancias y así mantenerse activas e innovadoras frente a otras empresas rivales (COFECE, 2018).

La competencia puede ser directa o indirecta, la competencia directa son aquellas empresas que producen o venden un producto igual o similar al nuestro y que además los venden en el mismo mercado que nosotros. Por otro lado, la competencia indirecta la conforman todas aquellas empresas que intervienen de forma lateral al mercado y clientes de los que se encuentra la empresa, y que buscan satisfacer las mismas necesidades, pero de otra forma o con otros productos (Endeavor, 2010).

B. Análisis de Competencia

El análisis de la competencia es el análisis de los recursos, capacidades, estrategias, ventajas competitivas, fortalezas, debilidades y demás características de los actuales y potenciales competidores de una empresa, que se realiza con el fin de poder, en base a dicho análisis, tomar decisiones o formular estrategias que permitan competir con ellos de la mejor manera posible.

Realizar el análisis de la competencia permite a las empresas estar prevenidas ante las nuevas acciones o estrategias de sus competidores, además de aprovechar sus debilidades, con el fin de bloquear o hacer frente a sus fortalezas, y tomar como referencia sus productos o las estrategias que mejores resultados les estén dando.

[1] Biblioteca de aprendizaje automático de software libre para el lenguaje de programación Python

[2] (Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos.

C. *Aprendizaje automático*

Desde que nacen hasta que mueren los seres humanos llevan a cabo diferentes procesos, entre ellos se encuentra el del aprendizaje, por medio del cual adquieren conocimientos y desarrollan habilidades para poder analizar y entender el mundo que los rodea.

A diferencia de los seres humanos, a las máquinas se les debe indicar lo que tienen que hacer y como aprender de los datos que se les proporcionan para que en base a la experiencia puedan obtener mejores resultados.

El aprendizaje automático o aprendizaje de máquinas (machine learning en inglés) es una rama de la inteligencia artificial que busca desarrollar técnicas para propiciar el aprendizaje de las computadoras. En otras palabras, sería encontrar la forma de transformar datos en información que entienda y pueda utilizar la computadora para realizar alguna acción para la cual no haya sido previamente programada explícitamente.

En su forma más básica, el aprendizaje automático utiliza algoritmos programados que reciben y analizan datos de entrada para predecir los valores de salida dentro de un rango aceptable. A medida que se introducen nuevos datos en estos algoritmos, aprenden y optimizan sus operaciones para mejorar el rendimiento, desarrollando "inteligencia" con el tiempo (APD, 2019).

D. *Tipos de aprendizaje automático*

Los algoritmos de aprendizaje automático se agrupan de acuerdo al resultado de los mismos. La clasificación más general es:

- Aprendizaje supervisado, este tipo de algoritmo es entrenado previamente con datos que ya han sido identificados y etiquetados.
- Aprendizaje no supervisado, este tipo de algoritmo no se alimenta con datos de entrenamiento, sino que analiza los datos obtenidos y busca similitudes entre los datos proporcionados para crear grupos de registros con características similares.
- Aprendizaje por refuerzo, este tipo de aprendizaje va mejorando conforme se utiliza ya que recibe una retroalimentación por cada resultado obtenido.

E. *Clasificación*

En aprendizaje automático la clasificación es el problema de identificar a algo dentro de un conjunto de categorías (subpoblaciones) que pertenece a una nueva observación, sobre la base de un conjunto de datos que contiene observaciones (o instancias) cuya categoría de miembros es conocida, en la Fig. 1 puede observarse un ejemplo de clasificación. La clasificación está considerada como un caso de aprendizaje supervisado, es decir, un aprendizaje en el que se dispone de un conjunto de observaciones correctamente identificadas. Un algoritmo que implementa la clasificación, se conoce como un clasificador, estos algoritmos predicen una o más variables discretas, basándose en los demás atributos del conjunto de datos.

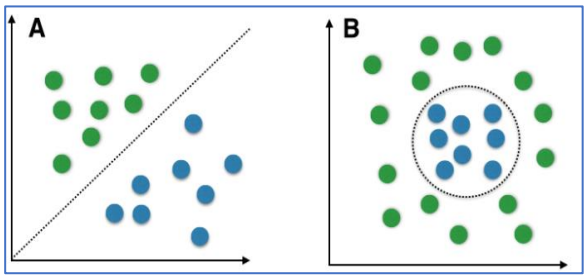

Fig. 1. Ejemplo de clasificación, a) empleando un modelo lineal, b) empleando un modelo no lineal.

F. *Clasificador Naive Bayes*

Los problemas de aprendizaje automático pueden resolverse aplicando el algoritmo adecuado, dependiendo el problema que se presente (Microsoft, 2018). Los algoritmos bayesianos, están fundamentados en el teorema de Bayes y trabajan con el supuesto de que cada propiedad es independiente de las demás con lo que puede predecir una clase en base al conjunto de datos utilizando la probabilidad. Esta suposición se denomina independencia condicional de clase. Naive Bayes o el Ingenuo Bayes es uno de los algoritmos más simples y poderosos para la clasificación basado en el Teorema de Bayes, la fórmula del teorema de Bayes puede observarse en (1). Donde:

- P(D): probabilidad de los datos independientemente de la hipótesis.
- P(h): es la probabilidad de que la hipótesis h sea cierta independientemente de los datos.
- P(h|D): es la probabilidad de la hipótesis h dada los datos D.
- P(D|h): es la probabilidad de los datos d dado que la hipótesis h era cierta.

$$P(h \mid D) = \frac{P(D|h)P(h)}{P(D)} \quad (1)$$

G. *SciKit-Learn (Anaconda, s.f.)*

Scikit-learn es una librería que proporciona un modelo bastante completo para implementar los diferentes algoritmos de Machine Learning en Python, es de código abierto. Proporciona una amplia gama de algoritmos de aprendizaje supervisados y no supervisados (scikit-learn, s.f.).

Esta librería está construida sobre SciPy (Scientific Python) e incluye un conjunto bastante amplio de librerías y paquetes como los siguientes:

- NumPy es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices.

- Pandas es una biblioteca de software escrita como extensión de NumPy para manipulación y análisis de datos para el lenguaje de programación Python.
- SciPy es una biblioteca libre y de código abierto para Python. Se compone de herramientas y algoritmos matemáticos.
- Matplotlib es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arreglos en el lenguaje de programación Python y su extensión matemática NumPy.

III. METODOLOGÍA

La metodología utilizada para este estudio está basada en el proceso KDD[3] para la obtención de conocimiento el cual ha sido adaptado para ajustarse a los objetivos y características del proyecto. El desarrollo del proyecto se encuentra divido principalmente en 4 etapas como puede observarse en la Fig. 2, las cuales serán descritas a continuación.

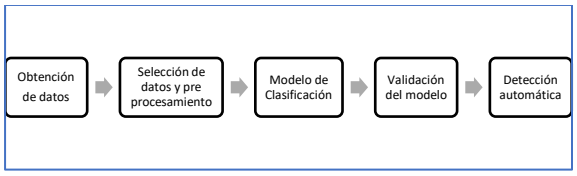

Fig. 2 Metodología del proyecto

A. *Obtención de los datos*

Para este proyecto, los datos fueron obtenidos de un sistema que se encuentra actualmente integrado con una plataforma de comercio electrónico, la cual mediante la conexión con su API[4] REST permite consultar los datos de vendedores y productos publicados en la plataforma de ventas. Los datos obtenidos de dicha plataforma se encuentran codificados en formato JSON como puede observarse en la Fig. 3.

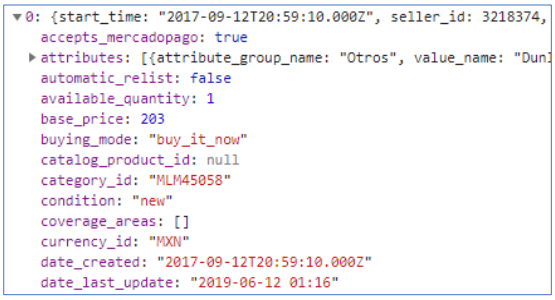

Fig. 3. Datos obtenidos en formato JSON

La estructura obtenida incluye los datos que van a utilizarse para hacer el análisis y aplicar el algoritmo de clasificación correspondiente, algunos de los datos obtenidos son precio, cantidad disponible, categoría, entre muchos otros, sin embargo, estos datos necesitan pasar por una etapa de preprocesamiento para poder utilizarla con Scikit-Learn.

B. *Selección y Preprocesamiento de datos*

En esta etapa se seleccionaron el precio del producto, la calificación del vendedor, tipo de publicación, tipo de envío y forma de pago en base a las especificaciones del proyecto. El atributo de clase seleccionado es Competidor.

El preprocesamiento consiste dar un tratamiento inicial a los datos obtenidos en el cual se puedan limpiar y transformar para encontrar el conjunto de valores para las características que permitirán realizar la clasificación óptima de los competidores. La Tabla 1 muestra un ejemplo del conjunto de datos de entrenamiento obtenido después del preprocesamiento. Las transformaciones realizadas a los valores del conjunto de atributos seleccionados son las siguientes:

- Para la clase objetivo, tomara el valor de 1 si es competencia y 2 en caso contrario.
- Para la calificación del vendedor se toma el valor de "3" para vendedores con reputación negativa, "2" para vendedores con reputación regular y "1" para vendedores con excelente reputación.
- En el caso del precio del producto es necesario compararlo con el precio que maneja el usuario y determinar si el precio es mayor o menor, por lo que se procede a realizar dicha transformación, tomando los valores de "2" si el producto del vendedor es más alto que el del usuario y "1" en caso contrario.
- El tipo de publicación toma el valor de "1" para publicaciones con mayor exposición y "2" para publicaciones normales.
- La forma de pago toma valores de "1" en caso de que el vendedor acepte pagos en mensualidades o con tarjeta y "2" en caso contrario.
- El tipo de envío toma el valor de "1" en caso de ofrecer envío gratis del producto y "2" en caso contrario.

Los datos una vez procesados se almacenan en un archivo con extensión CSV[5] para poder ser utilizados por la librería de SciKit-Learn y Pandas.

[3] Descubrimiento de Conocimiento en Bases de Datos o KDD se refiere al proceso de identificar patrones válidos, novedosos, potencialmente útiles y principalmente entendibles.

[4] API (Application Programming Interface), es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

[5] CSV, Valores Separados por Comas

TABLA I. CONJUNTO DE DATOS DE ENTRENAMIENTO PARA EL MODELO DE CLASIFICACIÓN

Calificación	Pago	Precio	Envío	Tipo	Competencia
1	1	1	1	1	1
2	2	2	1	1	2
3	2	2	2	1	2
3	2	2	1	1	2
2	1	1	1	2	1
1	2	1	1	2	1
1	1	2	1	2	1
2	1	1	2	2	1
2	2	2	2	1	2
1	2	2	2	1	2
1	2	1	1	1	2
3	1	2	2	2	1
2	1	1	1	2	1
1	2	1	1	2	1

C. *Modelo de Clasificación*

Para la siguiente etapa se va a definir y entrenar el modelo de clasificación que va a utilizarse para poder analizar y clasificar las instancias del conjunto de datos de entrenamiento. Para ello se han seleccionado las siguientes herramientas:

- Entorno de desarrollo Anaconda
- Python Versión 3.7
- Scikit-Learn Versión 0.22.1
- Spyder Versión 3.3.6

Como ya se había mencionado anteriormente la librería SciKit-Leam permite trabajar con diferentes algoritmos de aprendizaje automático y con ayuda de la librería de pandas se puede importar el conjunto de datos de entrenamiento para entrenar el modelo en el lenguaje Python.

Anaconda es un entorno de desarrollo de código abierto para los lenguajes Python y R (Anaconda, s.f.), utilizado en ciencia de datos, y aprendizaje automático. Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Anaconda viene precargado con las librerías necesarias y editores de código para trabajar con el modelo de clasificación, por lo que no es necesario descargar la librería de Scikit-Learn ni Pandas como puede observarse en la Fig. 4

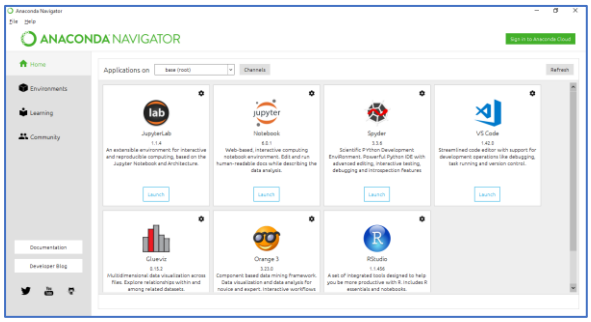

Fig. 4 Entorno de desarrollo anaconda.

Una vez preparado el entorno de desarrollo, el primer paso es elegir un editor de código, en este trabajo se utilizó el editor de código Spyder que viene incluido en Anaconda, el cual permite realizar varias funciones como la exploración de variables y depuración de código las cuales son bastante útiles cuando se trabaja con Python, en la Fig. 5 pueden observarse los diferentes componentes que pueden utilizarse.

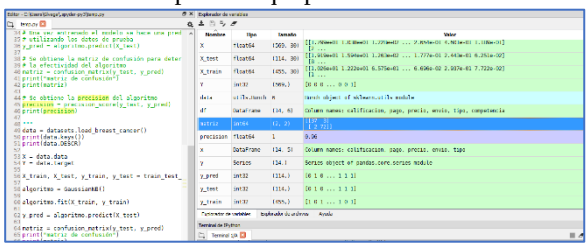

Fig. 5. Editor de código Spyder

El siguiente paso es cargar los datos al proyecto, para ello se utiliza la librería de Pandas las cual como se mencionó anteriormente sirve para la manipulación y análisis de datos (Pandas, s.f.). En la Fig. 6 se muestra cómo se importa la librería y se obtienen los datos para su utilización. El archivo CSV que se está cargando debe estar en la misma ubicación que el archivo Python que se está ejecutando. En este caso se

```
# Importar la librería de Pandas
import pandas as pd

# Cargar el conjunto de datos de
# entrenamiento para el modelo
data = pd.read_csv("competencia.csv")

print(data)
```

Fig. 6. Cargar los datos de entrenamiento con pandas

Una vez cargados los datos podemos visualizarlos mediante el explorador de variables disponible en el editor de código como se muestra en la Fig. 7.

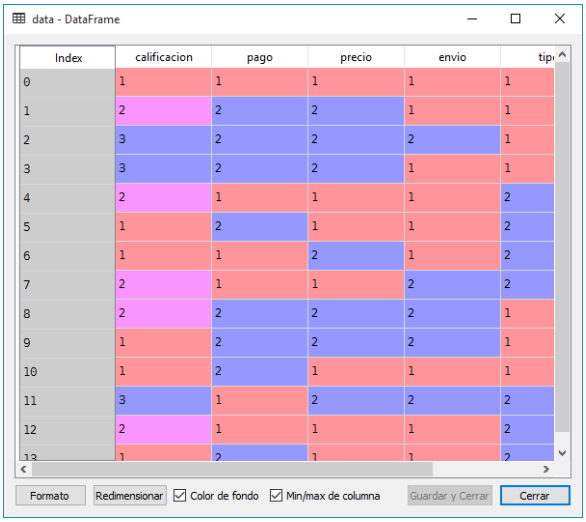

Fig. 7. Datos leídos y visualizados dentro de la aplicación.

El entrenamiento del modelo es una de las partes más importantes al trabajar con aprendizaje automático ya que permite mejorar la precisión de los modelos desarrollados con el fin de reducir el valor del error obtenido. Antes de comenzar con el entrenamiento, se deben de separar los datos que se usaran para probar el modelo como puede verse en la Fig. 8.

```
df = pd.DataFrame(data)

x = df[['calificacion', 'pago', 'precio', 'envio', 'tipo']]
y = df['competencia']

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

Fig. 8. Separación de los datos de entrenamiento y pruebas.

Una vez separados los datos, procedemos a definir el tipo de modelo que va a utilizarse, que para este caso es Naive Bayes el cual se define llamando a la librería GaussianNB de Scikit-Learn. El cual una vez haya sido definido se procede a entrenar con los datos de entrenamiento como puede observarse en la Fig. 9. Una vez el modelo ha sido entrenado se procede a validarlo haciendo una predicción proporcionando los datos de prueba que se separaron al inicio.

```
# Se define el modelo a utilizar
algoritmo = GaussianNB()

# Se entrena el modelo
algoritmo.fit(X_train, y_train)

# Una vez entrenado el modelo se hace una predicción
# utilizando los datos de prueba
y_pred = algoritmo.predict(X_test)
```

Fig. 9. Definición y Entrenamiento del modelo.

D. Validación del modelo de clasificación

Una vez que se han realizado las predicciones a los datos de prueba con el modelo entrenado, se procede a determinar el desempeño obtenido por el mismo, para ello se genera una matriz de confusión a la cual se le proporciona mediante parámetros, los datos de entrenamiento y los datos que se predijeron, de modo de poder visualizar el número de registros que fueron correctamente clasificados y cuáles no, como se muestra en la Fig. 10.

```
# Se obtiene la matriz de confusión para determinar
# la efectividad del algoritmo
matriz = confusion_matrix(y_test, y_pred)
print("matriz de confusión")
print(matriz)
```

Fig. 10. Obtención de la matriz de confusión

En la Fig. 12 se muestra la matriz de confusión obtenida en donde se pudieron clasificar correctamente 109 registros de prueba de un total de 114 con lo que se puede concluir que el modelo desarrollado tiene un desempeño aceptable y que puede utilizarse para predecir nuevos registros.

matriz	int64	(2, 2)	[[37 3] [2 72]]

Fig. 11. Matriz de confusión obtenida del modelo generado.

Del mismo modo que la matriz de confusión, se busca determinar el nivel de precisión del modelo generado, para ello, se utiliza la función de *precision_score* de SciKit-Learn mostrada en la Fig.12, a la cual se le proporcionan los datos de entrenamiento, así como también los datos que se predijeron.

```
# Se obtiene la precision del algoritmo
precision = precision_score(y_test, y_pred)
print(precision)
```

Fig. 12. Obtención de la precisión del modelo.

La precisión obtenida fue 0.96 como puede observarse en la Fig. 13, con esto se puede concluir que la precisión del modelo es aceptable y puede utilizarse en otras predicciones.

precision	float64	1	0.96

Fig. 13. Precisión del modelo.

E. Detección automática y análisis de competencia

Una vez entrenado el modelo de clasificación se procede a integrar el modelo dentro de una aplicación de software la cual permitirá al usuario detectar y analizar a su competencia.

Para ello se desarrolló una aplicación que permite hacer consultas a la plataforma de comercio electrónico y listar los resultados. La aplicación ha sido desarrollada utilizando Django, que es un framework[6] de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como MVT[7] (Django, s.f.).

Fig. 14. Detección automática de competencia con modelo clasificación

Una vez que se ha integrado el modelo de clasificación, la aplicación detecta aquellos productos que pueden ser considerados como competencia, permitiendo al usuario analizar su información y darles seguimiento a sus competidores, así mismo la aplicación muestra si el producto de la competencia tiene un precio más bajo o alto que el del usuario como puede observarse en la Fig. 14.

[6] Framework, entorno de trabajo o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios.

[7] MVT, Modelo Vista Template

La información mostrada de los competidores que fueron detectados por la herramienta es de utilidad para que el usuario pueda analizar a su competencia directa y así pueda tomar decisiones o desarrollar estrategias que le permitan vender mejor sus productos.

IV. RESULTADOS OBTENIDOS

Los resultados obtenidos al desarrollar el presente trabajo han sido bastante satisfactorios, ya que el modelo entrenado fue capaz de predecir casi en su totalidad los datos de pruebas con un desempeño bastante alto como se observa en la matriz de confusión de la Fig. 15. En donde solo se encontraron correctamente clasificados 109 registros de un total de 114, logrando una precisión del 96% con lo que se puede concluir que podemos aplicar el modelo desarrollado para predecir nuevos registros o utilizarlo en alguna herramienta de software como se observó anteriormente.

	Competencia	No Competencia	
Competencia	37	3	40
No Competencia	2	72	74
	39	75	

Fig. 15. Desempeño del algoritmo observado mediante una matriz de confusión.

La aplicación desarrollada para probar el modelo ha logrado predecir de manera muy precisa aquellos registros que se pueden considerar competencia, permitiendo al usuario hacer un análisis de su información Fig. 16.

Fig. 16. Análisis del precio actual del producto del usuario frente al de su competencia

V. COMENTARIOS FINALES

A. *Conclusiones*

Se ha mostrado en el presente trabajo el desarrollo de un clasificador bayesiano desarrollado en Python con ayuda de la librería SciKit-Learn utilizando un conjunto de datos de entrenamiento con registros de vendedores y productos publicados en plataformas de venta en línea con la finalidad de poder detectar automáticamente y analizar la competencia de un usuario, el cual ha demostrado tener un desempeño aceptable con el cual se ha podido realizar correctamente la predicción de registros que cumplen con las características necesarias para considerarse como competidor.

B. *Recomendaciones y trabajo a futuro.*

En el presente trabajo se ha mostrado el desarrollo de un clasificador bayesiano utilizando el algoritmo de Naive Bayes con ayuda de la librería SciKit-Learn de Python, el cual ha demostrado tener muy buenos. El algoritmo de Naive Bayes resulta bastante útil para realizar una exploración inicial de los datos, por lo que más adelante se podrían aplicar los resultados obtenidos para crear modelos de minería de datos adicionales con otros algoritmos más complejos y precisos, por lo que se recomienda continuar el presente trabajo explorando otras técnicas de clasificación como árboles de decisión o redes neuronales utilizando la librería y así poder comparar los resultados obtenidos de cada uno de ellos.

VI. BIBLIOGRAFÍA

(n.d.). From Anaconda: https://www.anaconda.com/

(n.d.). From Pandas: https://pandas.pydata.org/

(n.d.). From Django: https://www.djangoproject.com/

APD, R. (2019). *apd*. From Machine Learning: https://www.apd.es/algoritmos-del-machine-learning/

COFECE. (2018). From https://www.cofece.mx/wp-content/uploads/2018/05/1lacompetenciaeconom.pdf

Endeavor. (2010). *Emprendedor*. From La competencia directa e indirecta: http://www2.esmas.com/emprendedor/herramientas-y-articulos/marketing/184455/competencia-competencia-directa-competencia-indirecta/

ENDUTIH. (2018). From https://www.inegi.org.mx/contenidos/saladeprensa/boletines/2019/OtrTemEcon/ENDUTIH_2018.pdf

Microsoft. (2018, Abril 30). From https://docs.microsoft.com/es-es/sql/analysis-services/data-mining/data-mining-algorithms-analysis-services-data-mining?view=sql-server-2017#choosing-the-right-algorithm

scikit-learn. (n.d.). From scikit-learn: https://scikit-learn.org/