

Universidad Andrés Bello

Facultad de Ingeniería

Ingeniería Civil Informática

Matías Poblete, Claudio Almarza, José Martínez, Gustavo Galvez

PROYECTO DE CIENCIA DE DATOS



**Universidad
Andrés Bello®**
Conectar • Innovar • Liderar

Trabajo dirigido por
Profesor Billy Peralta

Curso Ciencia de Datos 2020

Índice

Índice	1
1. Definición de Proyecto	2
2. Descripción de los datos	3
2.1. Preprocesamiento de Datos	3
2.2. Análisis Exploratorio	6
2.3. Diseño de Experimentos	9
3. Implementación de algoritmos básicos	9
3.1. Trabajos relacionados con el Proyecto que utilicen algoritmos básicos	9
3.2. Descripción de los algoritmos básicos	12
3.2.1. Contextualización para abarcar Hito 2	12
3.2.2. Definición de algoritmos básicos utilizados en el Hito 2:	12
3.3. Implementación del algoritmo básico	15
3.4. Resultados Obtenidos	17
4. Implementación de algoritmos innovadores y experimentos	18
4.1. Trabajos relacionados con el Proyecto que utilicen algoritmos innovadores y experimentos con los datos	18
4.2. Descripción Algoritmos innovadores y nuevos métodos	19
4.3. Implementación en código del Algoritmo Innovador y del Experimento	22
4.3.1. Contexto de Implementación de Algoritmo Innovador y Experimento	22
4.4. Implementación Algoritmo Innovador	23
5. Resultado algoritmo innovador	27
6. Conclusión Final	29
Bibliografía	31

1. Definición de Proyecto

El proyecto consiste en la aplicación de distintos conceptos de la Ciencia de Datos en los cuales involucraremos diferentes tipos de algoritmos y análisis para predecir si una persona comprará o no en un comercio electrónico. Los tópicos esperados en este proyecto involucran el marketing y las ventas online, y el objetivo principal es predecir si una persona comprará en una tienda online basándose en su navegación.

Esto se realizará en tres fases secuenciales, Análisis y preprocesamiento de datos, Implementación de algoritmos básicos y Resultados de innovación de algoritmos. Cada punto servirá para ir escalando en el estudio estadístico de la hipótesis establecida.

En la fase de Análisis y preprocesamiento de datos se realizará un análisis exploratorio 1D y 2D de los datos para resolver preguntas acerca de los datos y así buscar respuestas visualizando y modelando estos datos recolectados. Luego se debe realizar un preprocesamiento de datos para eliminar datos impuros y así generar un nuevo conjunto de datos para mejorar el proceso de minería de datos. Al concluir estos puntos se definirán los diseños de los experimentos que van a ser planteados para realizar durante el proyecto.

En la fase de Implementación de algoritmos básicos se hará una revisión de al menos 4 trabajos que tengan una relación con el proyecto, los trabajos deben ser desde el 2015 en adelante. Se realizará una descripción de los algoritmos básicos escogidos para la tarea en acuerdo con el profesor guía del curso. Teniendo el algoritmo básico definido se implementará en la tarea y luego se realizarán los reportes pertinentes de los resultados obtenidos con los datos.

En la tercera y ultima fase, Resultados de innovación de algoritmos también se realizará una revisión de al menos 3 trabajos relacionados al proyecto, pero en este punto estos trabajos tendrán más relación a algoritmos innovadores que resuelvan la problemática de manera poco convencional, observando los resultados obtenidos por estas técnicas y así poder describir e implementar un algoritmo innovador, este algoritmo estará previamente estudiado por nosotros y nuestro profesor guía. Luego de implementarlo en el proyecto se realizarán reportes de los resultados obtenidos con sus conclusiones pertinentes.

Por último, con toda la información recopilada a lo largo de todo el proyecto se realizará una presentación al curso, mostrando los procesos realizados, los datos obtenidos durante todo el ciclo del proceso y los resultados finales.

2. Descripción de los datos

El dataset se formó para que cada sesión de navegación pertenezca a un usuario diferente en un período de 1 año para evitar cualquier tendencia a una campaña específica, día especial (Cyberday), perfil de usuario o período. El dataset consta de 10 atributos numéricos y 8 categóricos. El atributo “Ingresos” (Revenue) se puede usar como etiqueta de clase.

EL dataset está compuesto por los siguientes atributos: “Administrativo”, “Duración administrativa”, “Informativo”, “Duración informativa”, “Relacionado con el producto” y “Duración relacionada con el producto” son de tipo de datos numérico, representan el número de diferentes tipos de páginas visitadas por el visitante en esa sesión y el tiempo total dedicado a cada una de ellas en esas categorías de página. Los valores de estas funciones se derivan de la información de URL de las páginas visitadas por el usuario y se actualizan en tiempo real cuando un usuario realiza una acción, moviéndose de una página a otra. Los atributos “Tasa de rebote”, “Tasa de salida” y “Valor de página” representan las métricas utilizadas por Google Analytics en cada página de los sitios de comercio electrónico. El valor del atributo “Porcentaje de rebote” en una página web tiene referencia al porcentaje de visitantes que ingresan a la página y luego se van sin activar ninguna otra solicitud del servidor durante esa sesión. El valor del atributo “Tasa de salida” en una página web en concreto, calcula para todas las visitas de la página, el porcentaje de la última sesión.

El atributo “Valor de la página” representa el valor promedio de la página web que el usuario visitó antes de completar una transacción en un comercio electrónico. El atributo “Día especial” indica la proximidad del tiempo en que se visita un sitio en un día especial específico, por ejemplo San Valentín, donde es más probable que las sesiones finalicen con una transacción. Este valor se determina considerando la dinámica del comercio electrónico, o sea, la duración entre la fecha del pedido y la fecha de entrega. Por ejemplo, para el día de Valentín, el valor tomará un valor distinto de cero entre el 2 de febrero y el 12 de febrero, y su valor máximo de 1 el 8 de febrero.

El dataset también incluye los siguientes atributos: funcionamiento del sistema, navegador, región, tipo de tráfico, tipo de visitante (visitante recurrente, nuevo visitante), fechas si la visita es en fin de semana y el mes del año (valor booleano).

De las 12,330 registros en el dataset, 84.5 % (10,422) fueron muestras de clase negativas que no terminaron en compras, y el resto (1908) fueron muestras de clases positivas que si terminaron en compras.

2.1. Preprocesamiento de Datos

Para entender el dataset “Online Shoppers Purchasing Intention” se debe comprender qué variables contiene, los tipos de datos de estas, cuantos datos contiene, la dimensión del dataset o la cantidad de datos faltantes por ejemplo. En el preprocesamiento de datos lo que se hace es buscar esta información usando código python

en este caso, y luego, esta información es analizada y modificada para adaptarla al formato necesario para que ciertas librerías logren procesar esta información correctamente.

df.dtypes		df.dtypes	
Administrative	int64	Administrative	float64
Administrative_Duration	float64	Administrative_Duration	float64
Informational	int64	Informational	float64
Informational_Duration	float64	Informational_Duration	float64
ProductRelated	int64	ProductRelated	float64
ProductRelated_Duration	float64	ProductRelated_Duration	float64
BounceRates	float64	BounceRates	float64
ExitRates	float64	ExitRates	float64
PageValues	float64	PageValues	float64
SpecialDay	float64	SpecialDay	float64
Month	int64	Month	float64
OperatingSystems	int64	OperatingSystems	float64
Browser	int64	Browser	float64
Region	int64	Region	float64
TrafficType	int64	TrafficType	float64
VisitorType	int64	VisitorType	float64
Weekend	bool	Weekend	float64
Revenue	bool	Revenue	float64
dtype: object		dtype: object	

Tipos de Datos de las variables que contiene el Dataset y la conversión de los tipos de datos a float64

Se consulta la cantidad de NULL para luego ser reemplazados con otros valores

```
[ ] # Comprobando los datos NULL del dataset
df.isnull().sum()
```

```
Administrative      0
Administrative_Duration  0
Informational      0
Informational_Duration  0
ProductRelated     0
ProductRelated_Duration  0
BounceRates        0
ExitRates          0
PageValues         0
SpecialDay         0
Month              0
OperatingSystems   0
Browser            0
Region             0
TrafficType        0
VisitorType        0
Weekend            0
Revenue            0
dtype: int64
```

Muestra visual de las variables y si contienen atributos con valor NULL

Luego para conocer las estadísticas clásicas de los valores de la data, se procede a describir que muestra el análisis 1D de los datos

```
[ ] df.describe()
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay	OperatingSystems	Browser	Region	TrafficType
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000
mean	2.315166	80.810611	0.503569	34.472396	31.731468	1194.746220	0.022191	0.043073	5.889258	0.061427	2.124006	2.357097	3.147364	4.069586
std	3.321784	176.779107	1.270156	140.749294	44.475503	1913.669288	0.040480	0.040597	10.568437	0.198917	0.911325	1.717277	2.401591	4.025169
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000
25%	0.000000	0.000000	0.000000	0.000000	7.000000	104.137500	0.000000	0.014286	0.000000	0.000000	2.000000	2.000000	1.000000	2.000000
50%	1.000000	7.500000	0.000000	0.000000	10.000000	598.936905	0.003112	0.025156	0.000000	0.000000	2.000000	2.000000	3.000000	2.000000
75%	4.000000	93.256250	0.000000	0.000000	30.000000	1464.157213	0.016813	0.050000	0.000000	0.000000	3.000000	2.000000	4.000000	4.000000
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	63973.522230	0.200000	0.200000	361.763742	1.000000	8.000000	13.000000	9.000000	20.000000

Análisis 1D de los Datos.

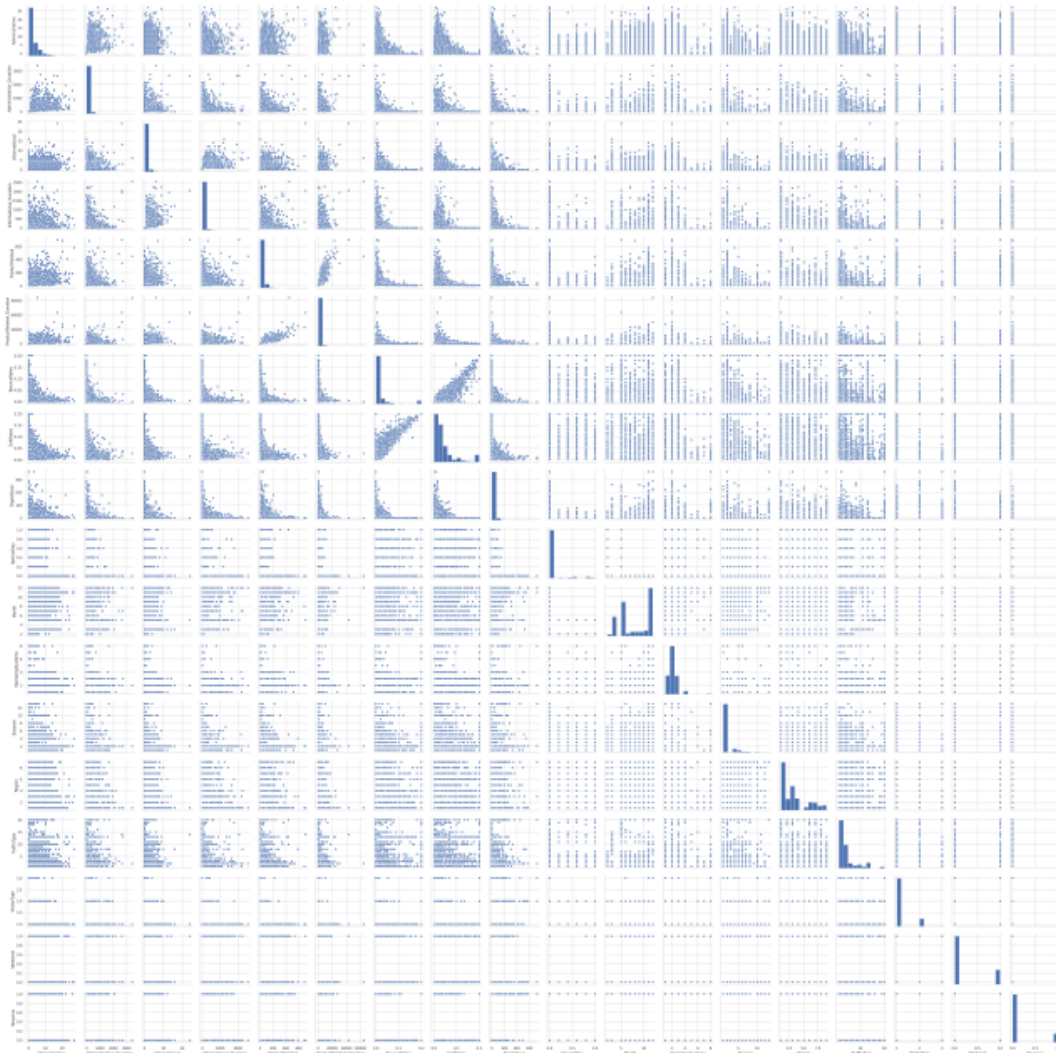
2.2. Análisis Exploratorio

Ya habiendo realizado el análisis básico y la preparación de los datos, se puede proceder a este siguiente paso, que consiste en un análisis más avanzado de estos utilizando herramientas de visualización de datos que facilitan enormemente en el rendimiento de la relación entre las distintas variables en este caso. Con el Pairplot podemos ver de forma detallada la correlación entre las distintas variables, donde en un Scatter que tenga una tendencia $x = y$ significará que existe una correlación positiva.

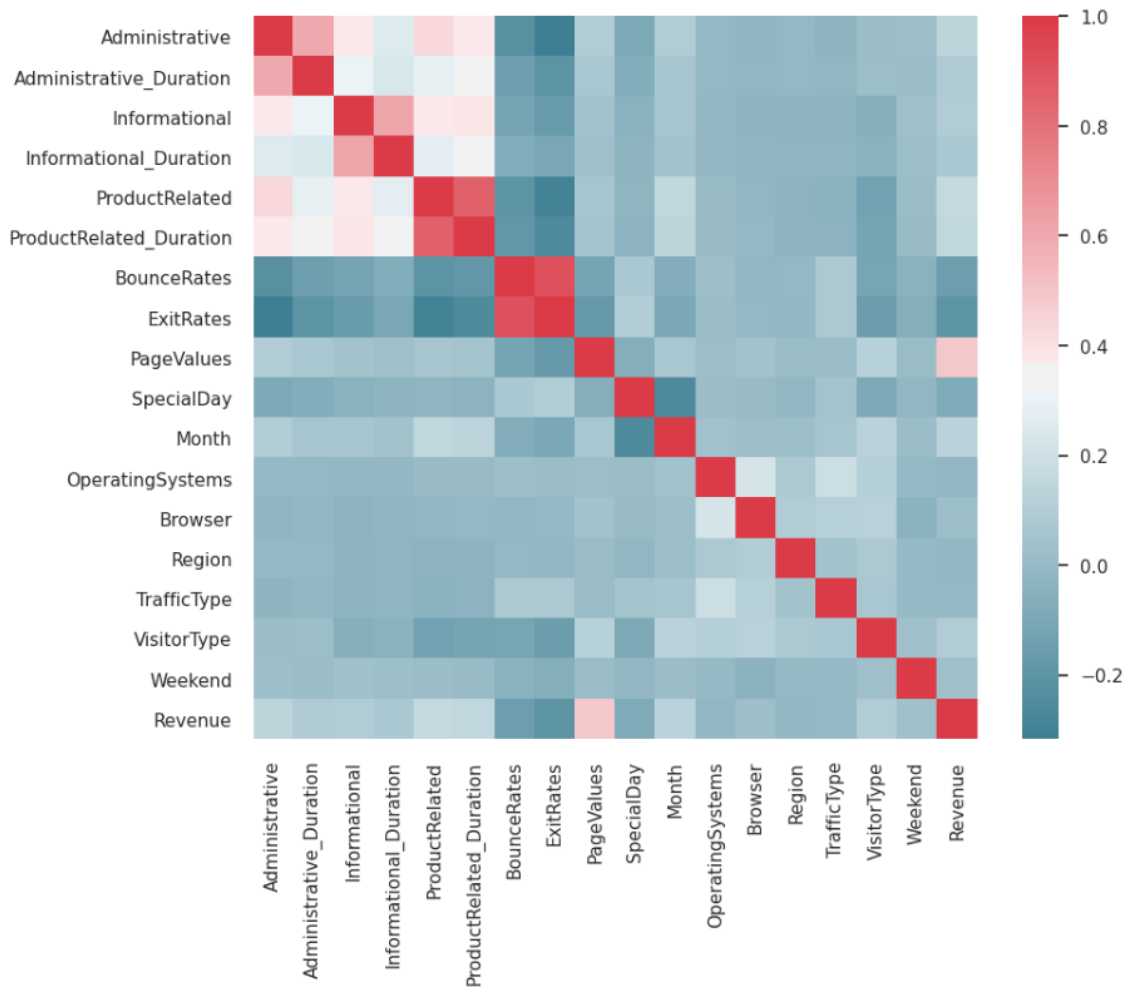
Acá se observa que entre “Administrativo”, “Duración administrativa”, “Informativo”, “Duración informativa”, “Relacionado con el producto” y “Duración relacionada con el producto” es donde existe mayor correlación entre las variables

También se observa que “BounceRates”, “ExitRates” y “PageValues” presentan una correlación negativa, ya que, se aprecia una forma de curva en lugar de una recta cercana al $x = y$, aunque existe una excepción en la que “BounceRates” y “ExitRates” tienen una correlación positiva mas fuerte y probablemente la de mayor correlación entre todas las variables.

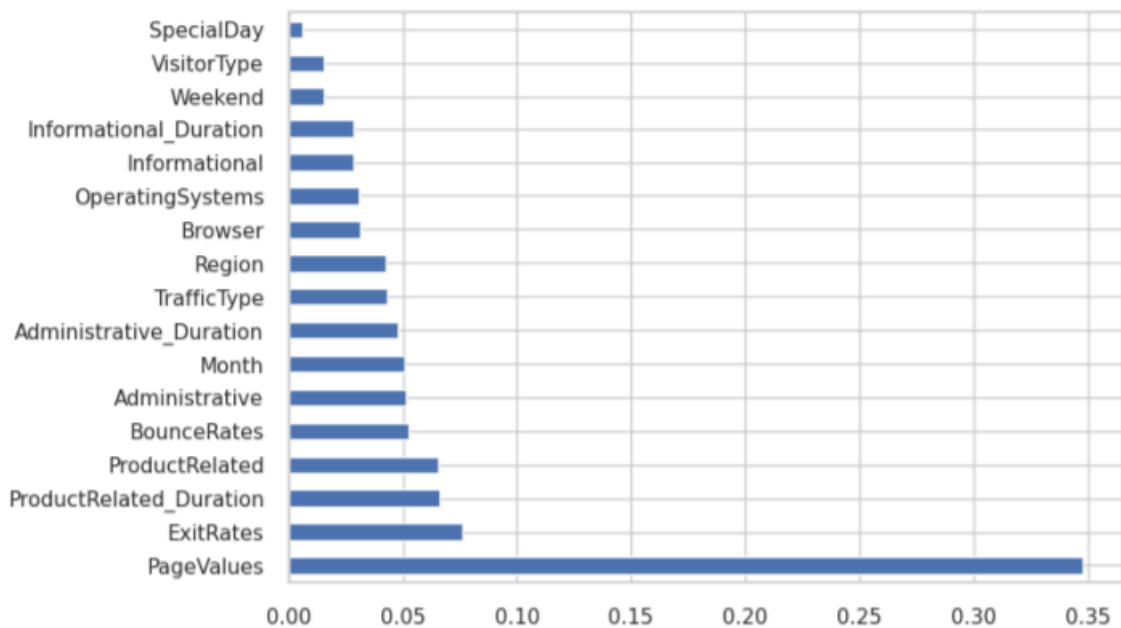
Por otro lado tambien se puede extraer del gráfico que no hay ninguna variable que influya directamente en el resultado de la variable 'Revenue' (la clase del dataset).



Para buscar mas información que se pueda extraer del dataset y la correlación de sus variables se plotó un heatmap en el cual se puede confirmar que las afirmaciones concluidas con el pairplot eran correctas y además se puede ver información extra. En la variable Revenue, en la que no se podía ver a simple vista que existía correlación con el Pairplot en ninguna de las variables, con esto se logra apreciar que en realidad si existe correlacion con una de ellas, PageValues, lo que significa que está variable tiene una mayor influencia que otras variables en el resultado de clase Revenue.



En el grafico de barras horizontales Por último para obtener mas detalles de este hallazgo, utilizamos la función ExtraTreesClassifier que nos ofrece Sklearn. Con esta función confirmamos lo que nos decia el Heatmap visualmente. PageValues es la variable que más se relaciona con la clase y por una gran diferencia con respecto a las demas variables.



2.3. Diseño de Experimentos

Un experimento seria comparar con 3 tipos de clasificadores y medir su accuracy y de esa forma determinar cual técnica es más apropiada para el modelo en cuanto al rendimiento de precision en los resultados obtenidos. Este punto es realizado en el punto 2 del Proyecto

El otro experimento que se podría realizar en este dataset es el de clustering, es el experimento perfecto para visualizar el comportamiento de agrupamiento de las variables con respecto al resultado final de la clase. Este experimento se abordará en el punto 3 de este Proyecto.

El experimento final podría ser cualquiera de los dichos recientemente o alguno sugerido por el profesor examinador.

3. Implementación de algoritmos básicos

3.1. Trabajos relacionados con el Proyecto que utilicen algoritmos básicos

1) Combined artificial bee colony algorithm and machine learning techniques for prediction of online consumer repurchase intention.

Este trabajo propone un método para poder predecir la intención de volver a comprar de un consumidor en línea, esto ya que según la investigación de Kumar, Kabra,

Mussada, Dash & Rana (2019) las plataformas web aunque pueden ser personalizadas a la hora de ofrecer múltiples servicios novedosos, tienen dudosa calidad, y esto puede afectar negativamente en la intención de compra en un consumidor, y considera que debería ser un foco de trabajo primordial el poder predecir estos escenarios de intención de compra. En este estudio, se combinaron algoritmos de Machine Learning con algoritmos de Colonia de Abeja (ABC). El estudio comienza con la identificación de las características del consumidor para la intención de volver a comprar, seguido de determinar la selección de características del consumidor, luego se usó el método de validación k-fold cross para medir el cual es el mejor modelo de clasificación, los modelos de clasificación utilizados fueron: Decision Tree árboles de decisión (C5.0), AdaBoost, Random Forest (bosque aleatorio), SVM (máquinas de vectores soporte) y Neural Networks (red neuronal), la partición de datos en entrenamiento y testeo fueron de (70-30 %), como resultado, el método de Adaboost supera a los otros modelos de clasificación, con precisión de 97.58 %.

Los datos de este estudio contienen los atributos relacionados a la variedad, calidad, atractivo, marca, valor y reputación de productos.

En este estudio, todos los modelos de clasificación propuestos obtuvieron sobre un 90 % de precisión, de los cuales, el algoritmo básico de Decision Tree obtuvo un 93,77 %.

2) Analysis of Different Predicting Model for Online Shoppers' Purchase Intention from Empirical Data.

Este paper habla sobre el crecimiento de la población en internet que compra en línea y su aumento en el pasar de los años esto por Algunos factores como la conveniencia, la variedad de productos, la política de devolución amigable, las revisiones de los clientes, etc. Explica la importancia que tiene la comprensión del comportamiento y la intención de los clientes en línea para el marketing, ya que, mejora la experiencia del cliente y por ende aumentan las ventas, por lo que el análisis de la intención de compra del comprador en línea se ha convertido en un campo emergente de investigación en el área de la computación y la minería de datos. También el paper explica que el análisis del comportamiento del comprador en línea se conoce como análisis de flujo de clics, ya que, el usuario solicita una serie de páginas web dentro de una sesión, el análisis de estos datos de flujo de clics mina los comportamientos de los compradores en línea a través de las solicitudes que realiza en la página web. El paper experimenta diferentes técnicas de minería de datos, incluyendo algoritmos de clasificación supervisados.

Este paper se enfoca en utilizar diferentes enfoques de aprendizaje automático para determinar a través del registro de los datos de un usuario si compro un artículo o no, esto pre-procesando el dataset, sustituyendo cualquier dato categórico por números y luego aplicando algoritmos de clasificación y métodos de conjunto.

Los algoritmos de clasificación utilizados en el paper son, los árboles de decisión, Random Forest, máquina de vectores de soporte (SVM) y Naive Bayes, y los métodos

de conjunto utilizados son Stacking (apilamiento), Bagging (Embolsado) y Boosting (Impulso).

El dataset que se utiliza en el paper es el mismo que utilizamos nosotros en nuestro proyecto.

Los resultados para los algoritmos de clasificación utilizados indicaron que Random Forest funciona mejor entre los cuatro algoritmos de clasificación (89.55 %) en predecir la intención de compra de los clientes. Se buscó aumentar la precisión de los algoritmos implementando diferentes métodos de conjunto. Luego, hemos tratado de aumentar la precisión de nuestro algoritmo incorporando diferentes métodos de conjunto de algoritmos de aprendizaje automático. Se hizo Stacking al crear un conjunto de datos secundario a partir de la predicción de los cuatro algoritmos de predicción y aplicando Random Forest aumentó la precisión en un 89.65 %. También se experimentó con dos métodos de Bagging. El primer método usa el modelo Random Forest aumentando la precisión al 90.25 %, y el segundo ocupa un modelo Extra Tree con una precisión del 89.88 %. Por último se utilizaron métodos de refuerzo, Adaboosting brindando una precisión del 89.2 % y Gradient Boosting brindando una precisión del 90.34 %.

El paper llega a la conclusión de que Random Forest es más adecuado que otros en términos de diferentes métricas de evaluación, y que Gradient Boosting aumenta la precisión que obtuvimos usando sólo Random Forest.

3) User behaviour modeling, recommendations, and purchase prediction during shopping festivals

Este paper se encarga de investigar los comportamientos de los usuarios en compras online, con lo que predice la acción que este tomará. Con los resultados obtenidos también se usan filtros para lograr entregar recomendaciones lo más acertadas posibles.

El dataset que usaron contiene un registro anónimo de compras en línea de 47,906 usuarios con 236,809 artículos. 581,430 entradas registran los comportamientos de compra en línea de los usuarios a través del sitio web JD.com durante 3 días de alta afluencia de noviembre de 2016. Cada entrada consta de identificación de usuario anónima, marca de tiempo, tipo de acción e identificación de elemento.

La tasa de aciertos fue calculada usando 5-fold cross validation, técnica usada para entrenar ciertas secciones del dataset con las otras, de modo que, en este caso, 4 partes del dataset (de 5) sean entrenadas mientras que la quinta parte de este es usada como datos de validación. Este proceso se repite 5 veces, de modo que todas las selecciones (1/5 del dataset) hayan pasado por el proceso de validar las otras 5 partes que se estaban entrenando.

4) Linear and Non-Linear Models for Purchase Prediction

El enfoque, se trata en la tarea como un problema de clasificación y se usa modelos lineales y no lineales para

hacer las predicciones, y luego construir un sistema de conjunto basado en la salida de los sistemas individuales. Los resultados muestran que MaxEnt y FM superan al NaiveSystem mientras que MLP es un poco peor que el NaiveSystem. Se construyó un sistema de conjunto como sistema final. Los resultados de la evaluación muestran que el sistema final es efectivo en los datos de la prueba. Sin embargo, el modelo MLP no ha mostrado la efectividad en los datos de la prueba. Se planea investigar cómo hacer que MLP funcione en esta tarea en el trabajo futuro.

3.2. Descripción de los algoritmos básicos

3.2.1. Contextualización para abarcar Hito 2

El algoritmo de Árbol de decisión (Decision Tree) fue elegido por la gran variedad de uso en trabajos de investigación relacionados y por la alta precisión que ha dado en los resultados al usarlo como metodo de clasificacion, también quisimos comparar con otros 2 clasificadores populares vistos en clase como lo son los algoritmos Naive Bayes y KNN.

3.2.2. Definicion de algoritmos básicos utilizados en el Hito 2:

Algoritmo de Árbol de Decisión (Decision Tree): Un árbol de decisión es una estructura de árbol donde un nodo interno representa una característica (atributo), la rama representa una regla de decisión y cada nodo hoja representa el resultado. El nodo raíz particiona el árbol en función del valor del atributo, dividiendolo de manera recursiva. Cada nodo en el árbol es un caso de prueba para un atributo en específico, y cada borde que desciende de ese nodo corresponde a una de las posibles respuestas al caso de prueba. Por lo que, el algoritmo de árbol de decisión selecciona el mejor atributo utilizando Medidas de selección de atributos (ASM) y toma un atributo el cuál será su nodo de decisión para dividir el conjunto de datos en subconjuntos más pequeños hasta que todas tuplas pertenezcan al mismo valor de atributo, no queden más atributos o no hayan más instancias.

Puntos a considerar en los arboles de decisiones:

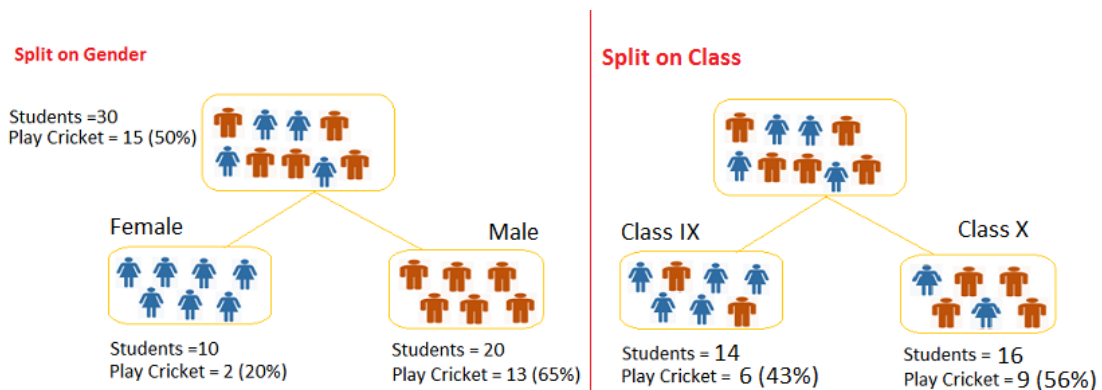
La medida de selección de atributos (ASM) es una heurística para seleccionar el criterio de división de los datos, esto nos ayuda a determinar puntos de interrupción para tuplas en un nodo dado, además nos proporciona un rango para cada atributo y el atributo de mejor puntuación se seleccionará como un atributo de división.

La ganancia de información es una propiedad estadística que mide qué tan bien

un atributo dado separa los ejemplos de entrenamiento de acuerdo con sus clasificación objetivo.

La entropía mide la impureza de un conjunto de entrada. Matemáticamente la entropía se define como la aleatoriedad o impureza en un sistema. En teoría de la información, se refiere a la impureza en un grupo de ejemplos. Dicho esto podemos inferir que la ganancia de información es una disminución de la entropía, por lo tanto, podemos decir que cuanto más “impuro” sea un conjunto de datos, mayor será la entropía y cuando menos “impuro” sea un conjunto de datos, menor será su entropía

Gini: medida de cuán a menudo un elemento elegido aleatoriamente del conjunto sería etiquetado incorrectamente si fue etiquetado de manera aleatoria de acuerdo a la distribución de las etiquetas en el subconjunto. La impureza de Gini se puede calcular sumando la probabilidad de cada elemento multiplicado por la probabilidad de un error en la categorización de ese elemento. Si alcanza el mínimo, que en este caso es cero significa que todos los casos del nodo corresponden a una sola categoría de destino.



Representación gráfica de un árbol de decisión.

Algoritmo de Naive Bayes (Clasificador): Un clasificador Naive Bayes es un modelo probabilístico de aprendizaje automático que se utiliza para la tarea de clasificación. El funcionamiento del clasificador se basa en el teorema de Bayes. El algoritmo clasificador de Naive Bayes asume de que la presencia o ausencia de un atributo particular no está relacionada con la presencia o ausencia de cualquier otro atributo, considera que cada uno de estos atributos contribuyen de manera independiente a la probabilidad de que un atributo sea de una manera en particular, independientemente de la presencia o ausencia de otros atributos. Se utiliza generalmente el método de máxima verosimilitud para estimar los parámetros en los modelos de Bayes.

El Teorema de Bayes se expresa con la siguiente ecuación:

$$P(A/B) = \frac{P(B/A)P(A)}{P(B)}$$

$P(A)$: probabilidad a priori, introduce conocimiento sobre los valores que puede tomar la hipótesis.

$P(B/A)$: es la función de verosimilitud de una hipótesis A dados los datos de B, es decir, la probabilidad de obtener B dado que A es verdadera.

$P(B)$: es la probabilidad marginal la cual observa los datos de B promediado sobre todas las posibles hipótesis de A.

$P(A/B)$: es la probabilidad a posteriori, que es la distribución de probabilidad final para la hipótesis

Usando el teorema de Bayes, podemos encontrar la probabilidad de que ocurra A, dado que B ha ocurrido. Aquí, B es la evidencia y A es la hipótesis. La suposición hecha aquí es que las características (atributos) son independientes. Es decir, la presencia de una característica particular no afecta a la otra. Por eso se llama ingenuo. Cuando este teorema se encuentra en el contexto del clasificador, la ecuación pasa a ser de esta forma.

$$P(y/X) = \frac{P(X/y)P(y)}{P(X)}$$

La variable y es la variable de clase (por ejemplo: jugar al golf), que representa si es adecuado para jugar al golf o no, dadas las condiciones. La variable X representa los parámetros / características, las cuales pueden ser múltiples atributos dentro de X.

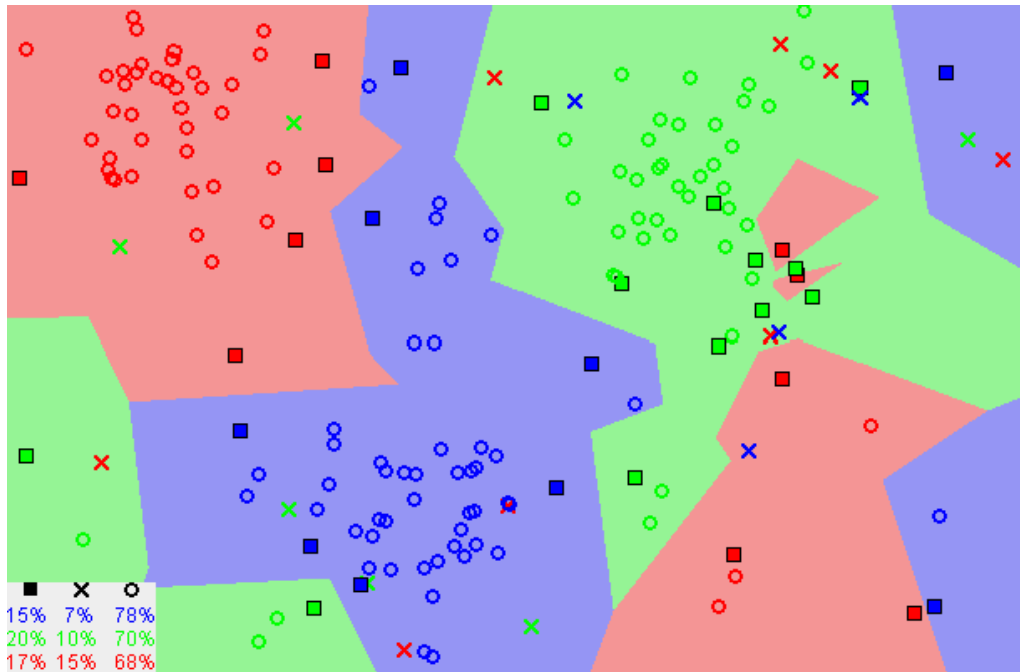
El método calcula 2 posibles probabilidades de las cuales usa una función para determinar la mayor probabilidad, que luego es la clasificación del registro.

$$y = \arg \max_i P(y) \prod_{i=1}^n P(X_i/y)$$

En este proyecto, se requiere usar por paquete de librería del lenguaje Python, el tipo de clasificador Gaussian Naive Bayes.

Algoritmo de K-Nearest Neighbors (Vecinos Cercanos): Es un método que busca en las observaciones más cercanas y clasifica el punto de interés basado en la mayoría de datos que le rodean. Es un algoritmo supervisado, basado en instancias, esto significa que el algoritmo predice o clasifica los nuevos datos en base a los casos similares que ya conocemos y es un algoritmo perezoso, ya que, no gasta tiempo en entrenamiento previo de la data. El algoritmo KNN Calcula la distancia entre un ítem a clasificar y el resto de ítems del dataset de entrenamiento, seleccionando los K elementos más cercanos, para decidir la clase de un punto KNN toma valor de k, pues este define a qué grupo pertenece cada punto. Las formas más típicas

para medir la cercanía entre puntos son la distancia Euclidiana y la Cosine Similarity



Representación gráfica de un clasificador KNN.

3.3. Implementación del algoritmo básico

Dado el trabajo realizado en el Hito 1 ya se tiene una mejor comprensión de que es lo que contiene el DataSet y su comportamiento a nivel de variables, por lo que se decidió pasar a la siguiente etapa la cuál es utilizar distintos algoritmos básicos para comprender como se comportan los atributos en relación a la clase, o sea, como los atributos influyen en el resultado final de la clase. Los algoritmos utilizados fueron los, Arboles de decisión, Naive Bayes y KNN.

Primero que todo se realizó una separación de las variables de la clase, paso necesario para los puntos posteriores

```
Primero separamos los atributos de la clase

[36] import sklearn.datasets as datasets
import pandas as pd
X = df.iloc[:, :17]
y = df.iloc[:, 17:18]
print("Valores de X")
print(X.head())

print("\n\n-----\n\n")
print("Valores de y")
print(y)
```

Separación de atributos.

También separamos nuestras variables X e Y en entrenamiento y test usando un 20 % de las filas totales para test y el otro 80 % para el entrenamiento. Lo que nos deja con 4 variables: Xtrain, xtest, ytrain y ytest. Luego revisamos imprimiendo las variables.

```
[ ] from sklearn.tree import DecisionTreeClassifier
    dtree = DecisionTreeClassifier(criterion='entropy',max_depth=5,min_samples_leaf=5)
    dtree.fit(X,y)

[ ] DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
    max_depth=5, max_features=None, max_leaf_nodes=None,
    min_impurity_decrease=0.0, min_impurity_split=None,
    min_samples_leaf=5, min_samples_split=2,
    min_weight_fraction_leaf=0.0, presort='deprecated',
    random_state=None, splitter='best')
```

Primero se partió con el algoritmo de Árbol de Decisión y se almacenó en la variable dtree, misma variable que posteriormente se entrena usando las variables X e Y.

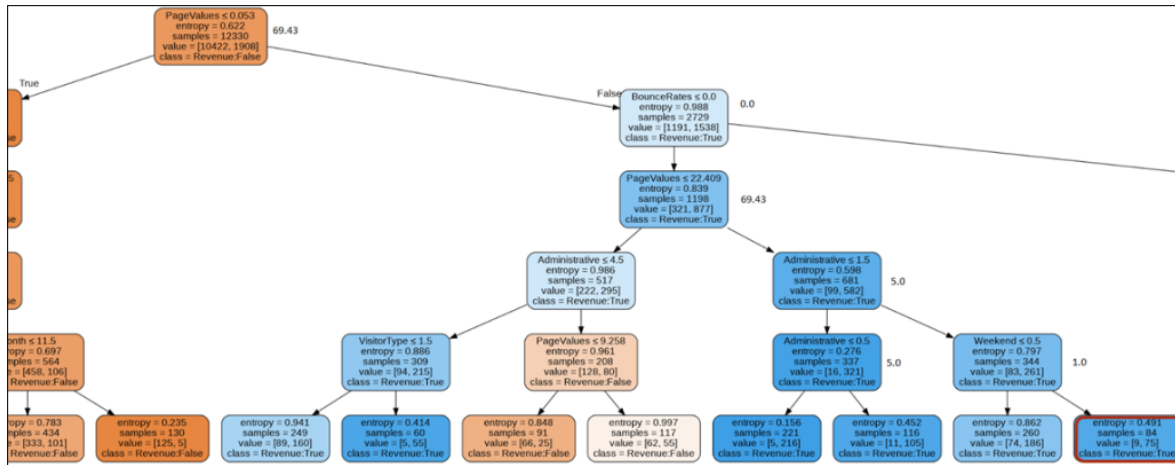
En la construcción del arbol se usaron como clases Revenue:True(Si Compra) y Revenue:False(No Compra). Se puede observar que el arbol considera como nodo principal el correspondiente a la variable PageValues, la misma variable que fue considerada como la mas relevante en el preprocesamiento, lo que ya indica que tiene sentido y se a ido por buen camino.

Se analizaron varias mediciones sobre el resultado de los entrenamientos, por lo que se creó la función accuracySuite.

Luego, se probó el siguiente registro en el modelo.

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BrandDates	ExitDates	PageValues	SpecialBuy	Month	OperatingSystem	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
895	5.0	100.916667	0.0	0.00	56.0	1432.223333	0.000000	0.002106	69.430432	0.0	3.0	2.0	2.0	3.0	3.0	1.0	1.0	1.0

En la imagen de abajo, se aprecia el registro de prueba para testear el modelo, por lo tanto la rama de decision muestra que clase Revenue (Compra) del atributo es clasificada como verdadera,



3.4. Resultados Obtenidos

Los resultados obtenidos de la comparacion entre los modelos se detalla a continuacion.

	True Negative	True Positive	False Positive	False Negative	Accuracy
Arbol de decisiones	2000	330	48	79	0.94831095
Naïve Bayes	1900	220	190	200	0.84462151
KNN	2000	81	23	330	0.85497124

A partir de lo visto en los experimentos realizados y con los modelos presentado, se puede asegurar sin ninguna duda que el mejor de estos algoritmos para este dataset en específico es el de Árboles de A partir de lo visto en los experimentos realizados y con los modelos presentado, se puede asegurar sin ninguna duda que el mejor de estos algoritmos para este dataset en específico es el de Árboles de Decisiones ya que 95 % de los datos fueron clasificados correctamente según la métrica de Precisión (Accuracy) según la matriz de confusión, mientras que Naive Bayes solo un 84 % y KNN un 85 %.

Estos resultados obtenidos en Árboles de decisiones nos dicen que los buenos resultados obtenidos en las distintas métricas no se deben a un sobreajuste (Overfitting), y por ende, son resultados fiables y aceptables si se quiere llegar a precedir sobre nuevos registros.ca de un clasificador KNN. La figura muestra el resultado de los experimentos realizados, mostrando los 3 modelos presentados en este punto del proyecto junto con precisión al momento de tomar las métricas en cuanto a los datos de test.

4. Implementación de algoritmos innovadores y experimentos

4.1. Trabajos relacionados con el Proyecto que utilicen algoritmos innovadores y experimentos con los datos

1) Online-Shoppers-Purchasing-Intention: Project by sharmaroshan

En este proyecto e investigacion se analiza el dataset visto en el informe con fines de analizar el comportamiento individual, entre pares y grupal de cada parametros y conocer el reporte de cada una de ellas. En primera instancia, se analizan las univariadas con graficos para conocer su representacion en el dataset, luego se procede a realizar lo mismo pero con pares de variables. En segunda instancia se analizan las variables de forma bi-variada conjunto a la etiqueta de clase del dataset (Revenue), todo esto mostrandose de forma grafica para visualizar. En tercera instancia, se usa el analisis multivariado en los atributos para conocer el rendimiento de la relacion entorno a la etiqueta. En ultima instancia, usando el metodo de clustering sobre los datos, se intenta descubrir el comportamiento del usuario entorno al tiempo de navegacion entre los sitios y paginas web determinadas en el dataset, mostrando algunos analisis a estos. En este punto, se utilizó el metodo de Elbow sobre el algoritmo de K Means para poder encontrar el mayor numero de clusters optimos, esto muestra resultados bastante conclusivos y concretos sobre las caracteisticas de prediccion del usuario. Esta parte es sumamente importante ya que nos da la idea y el concepto de como aplicar este algoirtmo de forma eficiente para nuestro proyecto. Luego de este paso, el analisis de la investifacion apunta a preprocesar y analizar con tecnicas de aprenidzaje supervisado el conjunto de datos, y de esta forma entregar distintos clasificadores con sus respectivas precisiones.

2) Real-Time Prediction of Online Shoppers' Purchasing Intention Using Random Forest

En esta investigación, se sugiere una respuesta en tiempo real para la predicción del comportamiento de comprador en línea , sostiene un sistema de predicción en la intención de compra del visitante tan pronto como se visita el sitio web. Para hacer eso, se utiliza la sesión y la información del visitante junto con los modelos de clasificación como Naive Bayes, Decision Tree C4.5 (Variación del algoritmo de Árbol de Decisión) y Random Forest. Además, se utiliza un muestreo especial para mejorar el rendimiento y la escalabilidad de cada clasificador. Los resultados muestran que Random Forest produce una precisión significativamente mayor y un puntaje F1 que Las técnicas comparadas.

En términos generales, el documento plantea en primera instancia la revisión de diferentes métodos como Support vector machines (SVMs) y k-Nearest Neighbor (k-NN) para un estudio en tiempo de fuera de línea, mientras que para el uso del tiempo real se investigaron técnicas como Multilayer Perceptron Network (MLP) y

Long Short-Term Memory-based Recurrent Neural Network (LSTM-RNN), los cuales obtuvieron un alto rendimiento en su medición el modelo respectivo. En segunda instancia el documento comienza con la descripción del dataset atributo por atributo para finalizar con la medición de los 3 modelos presentados, mostrando que el método de Naive Bayes obtienen el mejor y más alto porcentaje de precisión con los otros 2 métodos, realizando una comparación de resultados con distintos muestreos de la data.

3) Using k-nearest-neighbor classification in the leaves of a tree

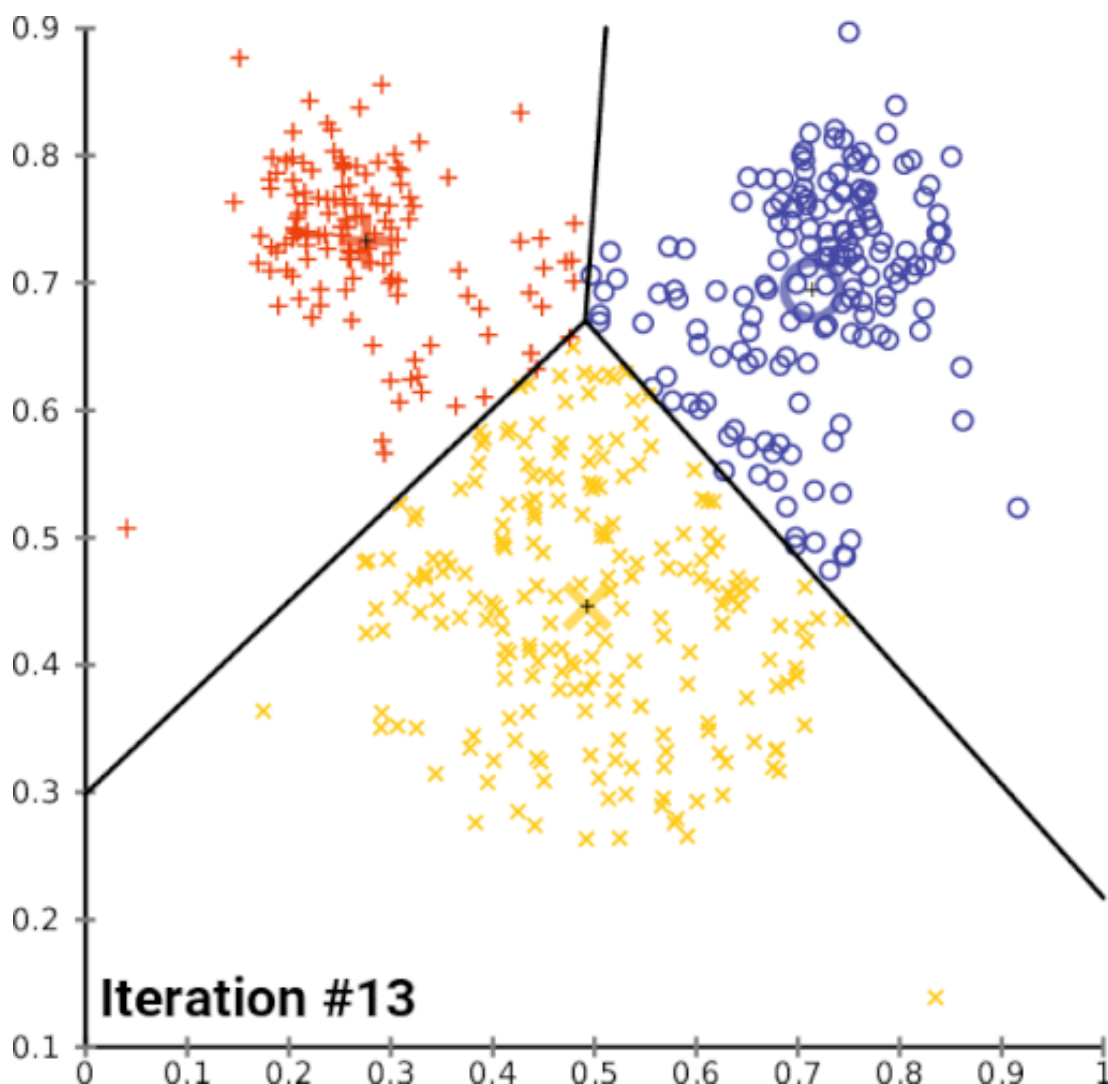
Esta investigación explica un clasificador híbrido, el cual es una combinación de Decision Tree (Árboles de decisión) y k-Nearest Neighbor (K vecinos más cercanos). Lo que se explica es que con un árbol de decisión se dividen las características y con KNN se clasifican los elementos del conjunto de prueba entre los elementos de entrenamiento en la misma hoja con el fin de reducir la carga computacional. En los problemas de clasificación se entrega un conjunto de entrenamiento donde se tienen mediciones por cada ítem y sus respectivas clases, el objetivo de estos problemas es predecir los nuevos miembros de la clase de un nuevo elemento o conjunto de elementos, en este artículo se describe una nueva técnica el cual aplicará en cada hoja KNN, es decir un elemento del conjunto de prueba que cae en una hoja del árbol esta clasificado con KNN y solo los elementos del conjunto de entrenamiento que caen en esa hoja se considerarán posibles vecinos. Se describe que la implementación de Árboles y KNN se hace por separado, cuando se realiza la clasificación con KNN se usan solo los datos de entrenamiento que caen en una hoja específica, ya que, al restringir los elementos del conjunto de entrenamiento se reduce la carga computacional. El procedimiento de árbol habitual divide el nodo raíz en dos nodos secundarios, la izquierda se divide en dos hojas más donde la clasificación KNN se realiza por separado dentro de cada una de esas hojas, de manera similar el lado derecho está dividido en dos nodos. En el modelo de árbol ordinario, el mejor tamaño del árbol se elige mediante Cross Validation y en este caso eligen el mejor tamaño para el árbol mediante su propia validación cruzada, siendo este denotado por KNN.

4.2. Descripción Algoritmos innovadores y nuevos métodos

Algoritmo K-Means: K-Means (traducido como K-Medias en español), es un método de agrupamiento o clustering. El clustering es una técnica para encontrar y clasificar K grupos de datos (clusters). Así, los elementos que comparten características semejantes estarán juntos en un mismo grupo, separados de los otros grupos con los que no comparten características. Para saber si los datos son parecidos o diferentes el algoritmo K-medias utiliza la distancia entre los datos. Las observaciones que se parecen tendrán una menor distancia entre ellas. En general, como medida se utiliza la distancia euclideana aunque también se pueden utilizar otras funciones. Funcionamiento: Los algoritmos de clustering son considerados de aprendizaje no

supervisado. Este tipo de algoritmos de aprendizaje no supervisado busca patrones en los datos sin tener una predicción específica como objetivo (no hay variable dependiente). En lugar de tener una salida, los datos solo tienen una entrada que serían las múltiples variables que describen los datos.

K-means necesita como dato de entrada el número de grupos en los que vamos a segmentar la población. A partir de este número k de clusters, el algoritmo coloca primero k puntos aleatorios (centroides). Luego asigna a cualquiera de esos puntos todas las muestras con las distancias más pequeñas. A continuación, el punto se desplaza a la media de las muestras más cercanas. Esto generará una nueva asignación de muestras, ya que algunas muestras están ahora más cerca de otro centroide. Este proceso se repite de forma iterativa y los grupos se van ajustando hasta que la asignación no cambia más moviendo los puntos. Este resultado final representa el ajuste que maximiza la distancia entre los distintos grupos y minimiza la distancia intragrupo. En esta imagen de ejemplo se ve cómo se mueven los centroides y cambian los grupos con las distintas iteraciones.



Representación de una iteración del algoritmo de K-Means.

Método Cross-Validation: Cross-validation (validación cruzada) es un procedimiento de remuestreo utilizado para evaluar modelos de Machine Learning (aprendizaje automático) en una muestra de datos limitada. El procedimiento tiene un único parámetro llamado k que se refiere al número de grupos en los que se dividirá una muestra de datos determinada. Como tal, el procedimiento a menudo se llama k -fold Cross-validation. Cuando se elige un valor específico para k , se puede usar en lugar de k en la referencia al modelo, como $k = 10$ convirtiéndose en una validación cruzada de 10 veces. La validación cruzada se utiliza principalmente en el aprendizaje automático aplicado para estimar la habilidad de un modelo de aprendizaje automático en datos no vistos. Es decir, usar una muestra limitada para estimar cómo se espera que el modelo funcione en general cuando se usa para hacer predicciones sobre datos que no se usaron durante el entrenamiento del modelo. Es un método popular porque es simple de entender y porque generalmente resulta en una estimación menos sesgada de la habilidad del modelo que otros métodos, como una división simple de entrenamiento / prueba.

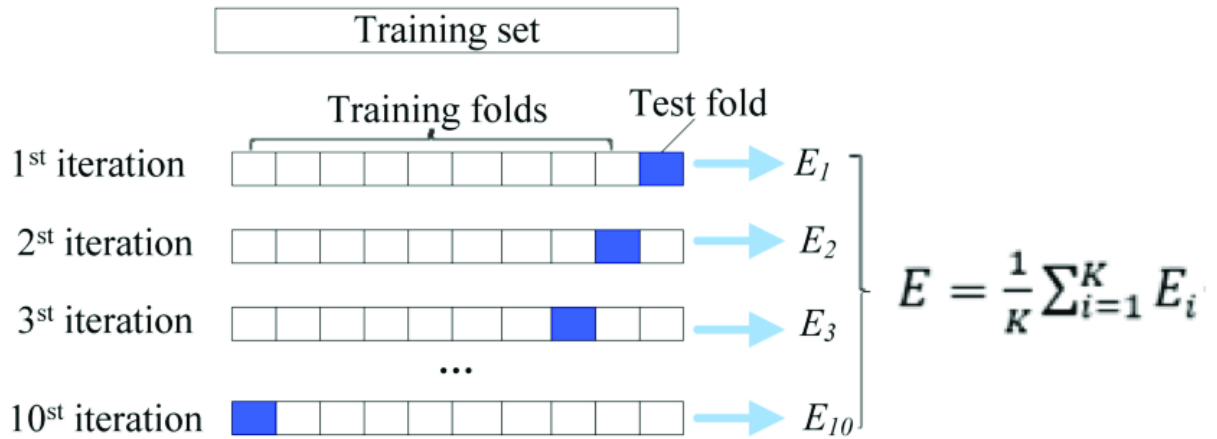
El procedimiento general es el siguiente: Mezclar aleatoriamente el conjunto de datos, Dividir el conjunto de datos en k grupos, Para cada grupo único: Tomar el grupo como un conjunto de datos de prueba o espera, Tomar los grupos restantes como un conjunto de datos de entrenamiento, Ajustar un modelo en el conjunto de entrenamiento y evaluar en el conjunto de prueba, Retener el puntaje de evaluación y descartar el modelo, Resuma la habilidad del modelo utilizando la muestra de puntajes de evaluación del modelo.

Es importante destacar que cada observación en la muestra de datos se asigna a un grupo individual y permanece en ese grupo durante la duración del procedimiento. Esto significa que cada muestra tiene la oportunidad de ser utilizada en el grupo de espera 1 vez y se usa para entrenar el modelo $k-1$ veces.

Los resultados de una ejecución de k -fold Cross-validation a menudo se resumen con la media de los puntajes de habilidad del modelo. También es una buena práctica incluir una medida de la varianza de los puntajes de habilidad, como la desviación estándar o el error estándar.

Configuración de k : El valor k debe elegirse cuidadosamente para su muestra de datos. Un valor mal elegido para k puede dar lugar a una idea mal representativa de la habilidad del modelo, como un puntaje con una alta varianza (que puede cambiar mucho en función de los datos utilizados para ajustar el modelo), o un alto sesgo, (como una sobreestimación de la habilidad del modelo). Tres tácticas comunes para elegir un valor para k son las siguientes: Representante: El valor de k se elige de manera que cada tren / grupo de prueba de muestras de datos sea lo suficientemente grande como para ser estadísticamente representativo del conjunto de datos más amplio, $k = 10$: el valor de k se fija en 10, un valor que se ha encontrado mediante la experimentación que generalmente da como resultado una estimación de habilidad modelo con un bajo sesgo y una variación modesta, $k = n$: el valor de k se fija en n , donde n es el tamaño del conjunto de datos para dar a cada muestra de prueba

la oportunidad de ser utilizada en el conjunto de datos de espera. Este enfoque se denomina validación cruzada de dejar uno fuera.



Representación de las iteraciones del procedimiento de Cross-validation.

4.3. Implementacion en codigo del Algoritmo Innovador y del Experimento

4.3.1. Contexto de Implementación de Algoritmo Innovador y Experimento

En esta parte final del proyecto de Ciencia de Datos se llevarán a cabo los siguientes pasos especificados por el profesor.

El procedimiento general por desarrollar es el siguiente:

- 1) Generar CrossValidation con un fold de 10 particiones en training y test.
- 2) Para cada Partición P:
 - Elegir un número de clústeres desde $K=2$ hasta $K=20$, o sea, con un NK de 10.
- 3) Para cada K:
 - Aplicar K-means en Set de Training por cada partición.
- 4) Para cada Clustering:
 - Hallar KNN en cada Clúster usando Set de Training desde $K=1$ hasta $K=10$.
 - Elegir el K con mejor rendimiento de KNN probado en el Set de Training (ignorar 1NN, dado que da clasificación perfecta).
- 5) Guardar Accuracy del Set de Test considerando los índices NK y P.
- 6) Promediar los Accuracy sumando en P, además calcular la Desviación Estándar, mostrar los datos en una tabla de tamaño NK que indique media de error y Desvia-

ción Estándar.

7) Probar aparte con la misma data entregada por CrossValidation en KNN con K=1 hasta K=10.

8) Analizar la Tabla y sacar conclusiones.

4.4. Implementación Algoritmo Innovador

En la siguiente imagen se muestra el algoritmo innovador implementado:

```
from sklearn.cluster import KMeans
from sklearn.model_selection import KFold

scores_knn = []
cv = KFold(n_splits=10, random_state=42, shuffle=True)
nfold, index = 1, 0
train_test_sets, score_per_k, best_score_per_k = [], [], []
scores_knn_dict = {}

for train_index, test_index in cv.split(X):
    ...
    print("Train Index: ", train_index)
    print("Test Index: ", test_index)
    print("\n")
    ...
    print(f'\nParticion de CV con Fold: {nfold}')
    X_train_fold, X_test_fold, y_train_fold, y_test_fold = X.iloc[train_index], X.iloc[test_index], y.iloc[train_index], y.iloc[test_index]
    train_test_sets.append([[nfold], [X_train_fold], [X_test_fold]])
    nfold += 1

for cluster in range(2, 21, 2):
    km_model = KMeans(n_clusters=cluster, random_state=42)
    km_model.fit(X_train_fold)
    km_labels = km_model.labels_
    print(f'\nLabel {cluster}, {km_labels}')
    print(f'Fold(CV), Clusters (K-means), K (KNN), Score')

    for k_neighbours in range(2, 12):
        (Ø_Ø) = k_neighbours
        knn_model = KNeighborsClassifier(n_neighbors=(Ø_Ø))
        knn_model.fit(X_train_fold, km_labels)
        knn_score = knn_model.score(X_train_fold, km_labels)
        scores_knn.append([nfold - 1, cluster, k_neighbours-1, knn_score])
        scores_knn_dict[k_neighbours-1] = knn_score
        score_per_k.append(knn_score)
        print(scores_knn[index])
        index += 1

best_k_neighbours = get_key(max(score_per_k), scores_knn_dict)
best_score_per_k.append([[nfold - 1], [cluster], [best_k_neighbours], [max(score_per_k)]])
score_per_k.clear()
scores_knn_dict = {}
```

Algoritmo híbrido K-Means y KNN.

Primeramente, se importa K-Means y K-Fold, K-Fold es el que realizará las particiones de Training y Test lo que nos servirá para poder saber cual partición es la mejor, dado los clústeres entregados por K-Means.

scores_knn es una lista vacía que guarda los Accuracy, los Scores y el Error de cada iteración realizada por KNN. La variable cv guarda las particiones que creamos con K-Fold, 10 particiones en este caso.

`nfold` es un índice que indica el número de fold e `index` que es el índice utilizado en el for que implementa a KNN, `nfold` parte en 1 e `index` en 0.

`train_test_sets` es una lista que guarda todas las particiones con Training y Test, `score_per_k` es una lista que hace lo mismo que `scores_knn` pero en este caso lo hace por cada K y el `best_score_per_k` es una lista que guarda el mejor score por cada K vecino.

`scores_knn_dict` es un diccionario que sirve para identificar los scores dado su KNN específico.

Luego de estas declaraciones vienen los For donde se implementa el algoritmo.

El primer For es para cada partición, donde `train_index` es el índice de los Training, `test_index` es el índice de los Test y `cv.split` realiza el split por cada fold.

Luego comienza a crear las particiones, donde primero guardará el Training en `X_train_fold` y el Test en `X_test_fold` y el Training en `y_train_fold` y el Test en `y_test_fold`, para luego agregar las particiones en la lista `train_test_sets`, donde se guarda la partición y los valores de esta partición, finalizando con el aumento de `nfold`. Se debe tener en consideración que cada vuelta al for significa que irán cambiando los datos de `X_train_fold`, `X_test_fold`, `y_train_fold` e `y_test_fold` hasta realizar las 10 particiones en el DataSet.

El segundo for se encarga de K-Means donde se generarán 10 clústeres desde $K=2$ hasta $k=20$ de 2 en 2.

`km_model` es la variable donde realizamos K-Means con el clúster dado por el for, se entrena `km_model` con una partición en específico y en la variable `km_labels` se guarda la etiqueta relacionada a esa partición.

El tercer for se encarga de KNN donde se parte desde el 2 porque no se considera el 1KNN dado que nos entrega un resultado perfecto.

La variable `carita` guarda el K utilizado en KNN, `knn_model` realiza el KNN con el valor entregado por `carita` y luego se entrena `knn_model` con una partición en específico y su label correspondiente para luego sacar el score de `knn_model` y pasárselo a `knn_score`.

`scores_knn` guarda los scores de KNN, el fold, el clúster y el número de vecinos usado, `scores_knn_dict` es el diccionario que guarda cada vecino con su respectivo score y `score_per_k` guarda el score por cada K de KNN.

Fuera de todos los for se encuentra la variable `best_k_neighbours` la que guarda

el mejor score dado por un K específico, esto los saca gracias al diccionario que entrega un K dado su score. En el caso de que diferentes vecinos dieran el mismo score, el algoritmo deja el primer mejor K que encuentre.

Por último, `best_score_per_k` guarda en que fold y en que clúster se encuentra el mejor K de KNN y su score, luego se reiniciar `score_per_k` y `scores_knn_dict`.

En la siguiente imagen se muestra el resultados obtenido con todas las combinaciones posibles:

```

Particion de CV con Fold: 1

Label 2, [0 0 0 ... 0 0 0]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 2, 1, 0.9992790844372353]
[1, 2, 2, 0.9998197711093089]
[1, 2, 3, 0.9994593133279265]
[1, 2, 4, 0.9997296566639632]
[1, 2, 5, 0.9990988555465441]
[1, 2, 6, 0.9992790844372353]
[1, 2, 7, 0.998918626655853]
[1, 2, 8, 0.9992790844372353]
[1, 2, 9, 0.9992790844372353]
[1, 2, 10, 0.9991889699918897]

Label 4, [0 0 0 ... 2 0 0]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 4, 1, 0.9982878255384338]
[1, 4, 2, 0.9987383977651617]
[1, 4, 3, 0.9982878255384338]
[1, 4, 4, 0.9986482833198161]
[1, 4, 5, 0.9977471388663602]
[1, 4, 6, 0.9981977110930882]
[1, 4, 7, 0.9979273677570515]
[1, 4, 8, 0.9982878255384338]
[1, 4, 9, 0.9977471388663602]
[1, 4, 10, 0.9981075966477426]

Label 6, [5 5 5 ... 0 5 5]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 6, 1, 0.9974767955303235]
[1, 6, 2, 0.998468054429125]
[1, 6, 3, 0.997386681084978]
[1, 6, 4, 0.997386681084978]
[1, 6, 5, 0.9972965666396323]
[1, 6, 6, 0.9975669099756691]
[1, 6, 7, 0.9977471388663602]
[1, 6, 8, 0.9981075966477426]
[1, 6, 9, 0.9975669099756691]
[1, 6, 10, 0.9976570244210147]

Label 8, [0 0 0 ... 6 0 0]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 8, 1, 0.9963954221861765]
[1, 8, 2, 0.997386681084978]
[1, 8, 3, 0.9967558799675588]
[1, 8, 4, 0.9970262233035956]
[1, 8, 5, 0.9963954221861765]
[1, 8, 6, 0.9967558799675588]
[1, 8, 7, 0.9961250788501397]
[1, 8, 8, 0.9962151932954852]
[1, 8, 9, 0.9952239343966838]
[1, 8, 10, 0.9953140488420293]

Label 10, [9 9 9 ... 6 9 9]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 10, 1, 0.995404163287375]
[1, 10, 2, 0.99693610885825]
[1, 10, 3, 0.9950437055059926]
[1, 10, 4, 0.995404163287375]
[1, 10, 5, 0.995404163287375]
[1, 10, 6, 0.9954942777327206]
[1, 10, 7, 0.9953140488420293]
[1, 10, 8, 0.995404163287375]
[1, 10, 9, 0.995404849959485]
[1, 10, 10, 0.995404163287375]

Label 12, [11 11 11 ... 4 0 11]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 12, 1, 0.992971073263044]
[1, 12, 2, 0.9947733621699558]
[1, 12, 3, 0.9921600432549338]
[1, 12, 4, 0.9930611877083897]
[1, 12, 5, 0.9930611877083897]
[1, 12, 6, 0.9930611877083897]
[1, 12, 7, 0.9925205010363162]
[1, 12, 8, 0.991439127692169]
[1, 12, 9, 0.991439127692169]
[1, 12, 10, 0.9922501577002794]

Label 14, [11 11 11 ... 4 0 11]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 14, 1, 0.9937821032711543]
[1, 14, 2, 0.9958547355141029]
[1, 14, 3, 0.9932414165990808]
[1, 14, 4, 0.9936018743804632]
[1, 14, 5, 0.9933315310444264]
[1, 14, 6, 0.9944129043885734]
[1, 14, 7, 0.9926106154816617]
[1, 14, 8, 0.9935117599351176]
[1, 14, 9, 0.9931513021537353]
[1, 14, 10, 0.9941425610525367]

Label 16, [0 0 0 ... 14 9 0]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 16, 1, 0.9917995854735514]
[1, 16, 2, 0.9937821032711543]
[1, 16, 3, 0.9906280976840588]
[1, 16, 4, 0.9927007299270073]
[1, 16, 5, 0.9904478687933675]
[1, 16, 6, 0.991439127692169]
[1, 16, 7, 0.9900874110119853]
[1, 16, 8, 0.9913490132468235]
[1, 16, 9, 0.9907182121294044]
[1, 16, 10, 0.9912588988014779]

Label 18, [7 7 7 ... 12 16 7]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 18, 1, 0.9901775254573308]
[1, 18, 2, 0.9936919888258008]
[1, 18, 3, 0.9913490132468235]
[1, 18, 4, 0.9927908443723529]
[1, 18, 5, 0.9902676399026764]
[1, 18, 6, 0.9911687843561323]
[1, 18, 7, 0.9901775254573308]
[1, 18, 8, 0.9913490132468235]
[1, 18, 9, 0.9908984410200955]
[1, 18, 10, 0.9906280976840588]

Label 20, [11 11 11 ... 4 0 11]
[Fold(CV), Clusters (K-means), K (KNN), Score]
[1, 20, 1, 0.9897269532306029]
[1, 20, 2, 0.9927908443723529]
[1, 20, 3, 0.9900874110119853]
[1, 20, 4, 0.991889699918897]
[1, 20, 5, 0.9919798143642425]
[1, 20, 6, 0.9917094710282058]
[1, 20, 7, 0.9908083265747499]
[1, 20, 8, 0.9910786699107867]
[1, 20, 9, 0.9899972965666396]
[1, 20, 10, 0.9908984410200955]

```

Resultados con todas las combinaciones.

Y en la siguiente imagen se muestra el resultado obtenido de cada fold pero con su mejor KNN.

Proyecto de Ciencia de Datos

Ciencia de Datos - Curso 2020

Matias Poblete, Claudio Almarza, José Martínez, Gustavo Galvez


```
[[[1], [2], [2], [0.9998197711093089]], [[5], [18], [2], [0.9943227899432279]],  
[[1], [4], [2], [0.9987383977651617]], [[5], [20], [2], [0.9923402721456249]],  
[[1], [6], [2], [0.998468054429125]], [[6], [2], [6], [0.9997296566639632]],  
[[1], [8], [2], [0.997386681084978]], [[6], [4], [4], [0.9991889699918897]],  
[[1], [10], [2], [0.99693610885825]], [[6], [6], [2], [0.998468054429125]],  
[[1], [12], [2], [0.9947733621699558]], [[6], [8], [2], [0.9972965666396323]],  
[[1], [14], [2], [0.9958547355141029]], [[6], [10], [2], [0.9963053077408308]],  
[[1], [16], [2], [0.9937821032711543]], [[6], [12], [2], [0.9961250788501397]],  
[[1], [18], [2], [0.9936919888258088]], [[6], [14], [2], [0.9953140488420293]],  
[[1], [20], [2], [0.9927908443723529]], [[6], [16], [2], [0.9943227899432279]],  
[[2], [2], [2], [0.999549427773272]], [[6], [18], [2], [0.9941425610525367]],  
[[2], [4], [4], [0.9991889699918897]], [[6], [20], [2], [0.9935117599351176]],  
[[2], [6], [2], [0.998468054429125]], [[7], [2], [2], [0.9994593133279265]],  
[[2], [8], [2], [0.9981075966477426]], [[7], [4], [2], [0.9994593133279265]],  
[[2], [10], [2], [0.9966657655222132]], [[7], [6], [2], [0.9982878255384338]],  
[[2], [12], [2], [0.995404163287375]], [[7], [8], [2], [0.9975669099756691]],  
[[2], [14], [2], [0.9955843921780662]], [[7], [10], [2], [0.9970262233035956]],  
[[2], [16], [2], [0.9939623321618456]], [[7], [12], [2], [0.9956745066234117]],  
[[2], [18], [2], [0.9937821032711543]], [[7], [14], [2], [0.9950437055059926]],  
[[2], [20], [2], [0.9931513021537353]], [[7], [16], [2], [0.9944129043885734]],  
[[3], [2], [2], [0.9997296566639632]], [[7], [18], [2], [0.9931513021537353]],  
[[3], [4], [2], [0.9985581688744706]], [[7], [20], [2], [0.9923402721456249]],  
[[3], [6], [8], [0.998468054429125]], [[8], [2], [2], [0.9996395422186176]],  
[[3], [8], [2], [0.9962151932954852]], [[8], [4], [2], [0.998918626655853]],  
[[3], [10], [4], [0.9965756510768676]], [[8], [6], [8], [0.9983779399837794]],  
[[3], [12], [2], [0.995404163287375]], [[8], [8], [2], [0.9974767955303235]],  
[[3], [14], [2], [0.995404163287375]], [[8], [10], [8], [0.9963053077408308]],  
[[3], [16], [2], [0.9919798143642425]], [[8], [12], [2], [0.9962151932954852]],  
[[3], [18], [4], [0.9930611877083897]], [[8], [14], [2], [0.9956745066234117]],  
[[3], [20], [2], [0.9935117599351176]], [[8], [16], [2], [0.9942326754978823]],  
[[4], [2], [4], [0.9997296566639632]], [[8], [18], [2], [0.9921600432549338]],  
[[4], [4], [2], [0.9993691988825809]], [[8], [20], [2], [0.9931513021537353]],  
[[4], [6], [2], [0.9981977110930882]], [[9], [2], [2], [0.9994593133279265]],  
[[4], [8], [2], [0.997386681084978]], [[9], [4], [2], [0.9986482833198161]],  
[[4], [10], [2], [0.9961250788501397]], [[9], [6], [2], [0.998918626655853]],  
[[4], [12], [2], [0.9958543921780662]], [[9], [8], [2], [0.9976570244210147]],  
[[4], [14], [2], [0.9946832477246103]], [[9], [10], [2], [0.9967558799675588]],  
[[4], [16], [2], [0.993421645489772]], [[9], [12], [2], [0.9957646210687573]],  
[[4], [18], [2], [0.9936018743804632]], [[9], [14], [2], [0.9958547355141029]],  
[[4], [20], [2], [0.9928809588176984]], [[9], [16], [2], [0.9940524466071912]],  
[[5], [2], [2], [0.9996395422186176]], [[9], [18], [2], [0.9936919888258088]],  
[[5], [4], [2], [0.9987383977651617]], [[9], [20], [2], [0.9938722177164999]],  
[[5], [6], [2], [0.998468054429125]], [[10], [2], [2], [0.9994593133279265]],  
[[5], [8], [2], [0.9975669099756691]], [[10], [4], [2], [0.9990087411011985]],  
[[5], [10], [4], [0.9960349644047941]], [[10], [6], [2], [0.9979273677570515]],  
[[5], [12], [2], [0.9963954221861765]], [[10], [8], [2], [0.9972965666396323]],  
[[5], [14], [2], [0.9942326754978823]], [[10], [10], [2], [0.9964855366315221]],  
[[5], [16], [2], [0.993421645489772]], [[10], [12], [2], [0.9956745066234117]],  
[[5], [18], [2], [0.9943227899432279]], [[10], [14], [2], [0.9954942777327206]],  
[[5], [20], [2], [0.9935117599351176]], [[10], [16], [2], [0.9930611877083897]],  
[[5], [2], [2], [0.9996395422186176]], [[10], [18], [2], [0.9936919888258088]],  
[[5], [4], [2], [0.9987383977651617]], [[10], [20], [2], [0.9935117599351176]]]
```

Resultados con el mejor KNN por fold

La siguiente imagen muestra la parte de código que encuentra el mejor KNN con el mejor Score en cada una de las particiones.

```
[ ] score_knn_dict2 = {}
    knn_scores_2, results_per_fold, knn_values_2 = [], [], []
    k_val = 0
    for fold in range(10):
        for i in range(10):
            knn_scores_2.append(best_score_per_k[k_val][3][0])
            knn_values_2.append(best_score_per_k[k_val][2][0])
            k_val += 1

        max_value = max(knn_scores_2)
        index_k = knn_scores_2.index(max_value)
        results_per_fold.append([fold+1, knn_values_2[index_k], max_value])
        knn_scores_2, knn_values_2 = [], []
    print("# Fold(CV), Best K per fold (KNN), Best Score")
    results_per_fold
```



```
["# Fold(CV), Best K per fold (KNN), Best Score"]
[[1], [2], [0.9998197711093089]],
[[2], [2], [0.999549427773272]],
[[3], [2], [0.9997296566639632]],
[[4], [4], [0.9997296566639632]],
[[5], [2], [0.9996395422186176]],
[[6], [6], [0.9997296566639632]],
[[7], [2], [0.9994593133279265]],
[[8], [2], [0.9996395422186176]],
[[9], [2], [0.9994593133279265]],
[[10], [2], [0.9994593133279265]]
```

Algoritmo que encuentra el mejor KNN por fold

Dado cada Score encontrado en el algoritmo anterior que fue probando en cada una de las particiones con todas sus combinaciones posibles y después entregar los mejores KNN para cada fold probando los K-means del K=2 hasta el K=20. Ahora lo que hace esta parte de código es sacar el mejor score por cada fold sabiendo a que KNN corresponde específicamente.

5. Resultado algoritmo innovador

Realizado todos los cálculos anteriores se realiza entrenamiento con el Test arrojando los siguientes resultados.

```
1 list_scores = []
2 for fold in range(10):
3
4     X_train_subset, X_test_subset = train_test_sets[fold][1][0], train_test_sets[fold][2][0]
5     y_train_subset, y_test_subset = train_test_sets[fold][3][0], train_test_sets[fold][4][0]
6
7     knn_bestmodel = KNeighborsClassifier(n_neighbors=results_per_fold[fold][1][0])
8     knn_bestmodel.fit(X_train_subset, np.ravel(y_train_subset))
9     bestknn_score = knn_model.score(X_test_subset, y_test_subset)
10    list_scores.append(bestknn_score)
11    print(f"Fold(CV) for Test set: {fold + 1}, K(KNN) per fold: {results_per_fold[fold][1][0]}, Test Score: {bestknn_score}")

Fold(CV) for Test set: 1, K(KNN) per fold: 2, Test Score: 0.31062449310624496
Fold(CV) for Test set: 2, K(KNN) per fold: 2, Test Score: 0.28548256285482565
Fold(CV) for Test set: 3, K(KNN) per fold: 2, Test Score: 0.30170316301703165
Fold(CV) for Test set: 4, K(KNN) per fold: 4, Test Score: 0.28386050283860503
Fold(CV) for Test set: 5, K(KNN) per fold: 2, Test Score: 0.2935928629359286
Fold(CV) for Test set: 6, K(KNN) per fold: 6, Test Score: 0.29764801297648014
Fold(CV) for Test set: 7, K(KNN) per fold: 2, Test Score: 0.28629359286293593
Fold(CV) for Test set: 8, K(KNN) per fold: 2, Test Score: 0.26926196269261965
Fold(CV) for Test set: 9, K(KNN) per fold: 2, Test Score: 0.28386050283860503
Fold(CV) for Test set: 10, K(KNN) per fold: 2, Test Score: 0.2935928629359286
```

Resultados obtenidos con el Set de Prueba.

Como se puede observar el Score de cada uno de los fold resultantes disminuyó considerablemente en comparación a los resultados obtenidos con el set de entrenamiento.

Se realizó un promedio de los Accuracy de cada fold y la desviación estándar de cada fold, lo que se muestra en la siguiente imagen.

```
1 mean_acc = np.average(list_scores)
2 std_acc = np.std(list_scores)
3 print(mean_acc)
4 print(std_acc)

0.2905920519059206
0.010875365055569413
```

Promedio de todos los Accuracy y Desviación estándar de todos los Accuracy
En la siguiente imagen se compara cada KNN con los datos de CrossValidation.

```
[225] 1
2 error_rate, accuracy, precision, recal, fScore, list_acc_total = [], [], [], [], [], []
3
4 for fold in range(10):
5     X_train_subset, X_test_subset = train_test_sets[fold][1][0], train_test_sets[fold][2][0]
6     y_train_subset, y_test_subset = train_test_sets[fold][3][0], train_test_sets[fold][4][0]
7     for k_value in range(1, 11):
8         X_train_subset, X_test_subset = train_test_sets[k_values][1][0], train_test_sets[k_values][2][0]
9         y_train_subset, y_test_subset = train_test_sets[k_values][3][0], train_test_sets[k_values][4][0]
10
11     knn_model_along = KNeighborsClassifier(n_neighbors=k_values+1)
12     knn_model_along.fit(X_train_subset, np.ravel(y_train_subset))
13     y_pred = knn.predict(X_test_subset)
14     score_knn_along = accuracy_score(y_test_subset, y_pred)
15     print(f'Fold: {fold+1}, K value per fold: {k_value}, Accuracy: {score_knn_along}')
16     accuracy.append(score_knn_along)
17     list_acc_total.append(score_knn_along)
18     print(f'Promedio por Fold: {np.mean(accuracy)}')
19 accuracy = []
20
```

```

Fold: 1, K value per fold: 1, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 2, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 3, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 4, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 5, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 6, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 7, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 8, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 9, Accuracy: 0.8848337388483374
Fold: 1, K value per fold: 10, Accuracy: 0.8848337388483374
Promedio por Fold: 0.8848337388483374
```

Resultados de KNN sin considerar CrossValidation

Y esta imagen muestra el promedio final para todos los folds.

```
[227] 1 print(f'Promedio para todos los Folds: {np.mean(list_acc_total)}')
```

```

Promedio para todos los Folds: 0.8848337388483377
```

Promedio de todos los Fold

Como se puede ver, los resultados del ultimo KNN tienen un accuracy mayor al probado con el algoritmo hibrido, lo que significa que probar el algoritmo con cada cluster no muestra necesariamente una mejora en la precision de testeo del modelo clasificador, ya que el ultimo modelo que solo utiliza KNN con los datos de Cross Validation obtienen tanto para cada fold como para el promedio total un resultado mucho mayor.

6. Conclusión Final

Contextualizando, en este trabajo se realizaron muchas actividades en relación a la predicción de compra de una persona en un comercio electrónico. Primeramente se tomó una data set en particular que contenía atributos categóricos y atributos numéricos, los cuales a través de un preprocesamiento de datos que nos ayudó a para integrar los datos para que no se mezclen, calcular el análisis 1D de los datos, tratar los outliers para minimizar inconsistencias y discretizar las variables, luego de esto profundizamos en la correlación de las variables para identificar la relación

entre las variables y así encontrar cuales tienen correlación positiva con la clase del dataset, todo lo realizado en la primera fase del proyecto era para trabajar la data set, analizarla, entenderla y así poder empezar a implementar los algoritmos de predicción.

En la segunda fase del proyecto, ya con la data set más trabajada y comprendida se comenzó a implementar algoritmos básicos de predicción para identificar el comportamiento de los atributos en relación a la clase, se utilizó decisión tree, KNN y Naive Bayes para las pruebas. Se separan los atributos de la clase para que así podamos tener atributos de Prueba y de Entrenamiento, se implementaron los algoritmos de predicción de forma secuencial y los resultados entregados por cada uno de estos algoritmos fueron mostrados por una suite de precisión y matrices de confusión, la cual en nuestro caso particular entrego que Decision Tree es el mejor algoritmo para esta Data Set específica.

Y en la última fase del proyecto se buscó innovar mezclando K-Means con KNN. La idea es crear particiones con CrossValidation utilizar K-Means en cada partición y en cada clúster generado por K-Means aplicar KNN en cada partición y elegir el K con mejor rendimiento, una vez obtenido los resultados se realiza un análisis de los datos con distintas medidas para visualizar tasa de acierto, error y desviación estándar.

Este proyecto busco que pasáramos por cada uno de los puntos a considerar en las características de un proyecto de ciencia de datos, Importación de librerías, descarga del DataSet con el que se trabajará, realizar el preprocesamiento de los datos, realizar análisis de los datos, implementar algoritmos de predicción básicos para trabajar con la Data, entrenar los modelos y por último implementación de un algoritmo innovador. Cada uno de estos pasos realizados en el proyecto logro que nos desarrolláramos y aprendiéramos lo que se debe realizar en los proyectos de esta índole, aplicando cada punto de manera lineal y práctica.

Bibliografía

- [1] Sakar, C.O., Polat, S.O., Katircioglu, M. et al.(2018), Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks., **Neural Computing & Applications** **31**, 6893–6908.
- [2] Kumar, A., Kabra, G., Mussada, E. K., Dash, M. K., & Rana, P. S. (2019), Combined artificial bee colony algorithm and machine learning techniques for prediction of online consumer repurchase intention.,**Neural Computing & Applications**, **31**(2), 877-890.
- [3] Zeng, Cao, Chen, & Li(2019), User behaviour modeling, recommendations, and purchase prediction during shopping festivals., **Electronic Markets**, **29**(2), 263-274.
- [4] Data, S. B. (2019). Árbol de decisión en Machine Learning (Parte 1). ,<https://sitiobigdata.com/2019/12/14/arbol-de-decision-en-machine-learning-parte-1>
- [5] Kabir, Ashraf & Ajwad (2019). Analysis of Different Predicting Model for On-line Shoppers' Purchase Intention from Empirical Data., **2019 22nd International Conference on Computer and Information Technology (IC-CIT)**, 1.
- [6] N. (2019). Clasificar con K-Nearest-Neighbor ejemplo en Python. Recuperado 5 de junio de 2020, de <https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/>
- [7] Harrison, O. (2018), Machine Learning Basics with the K-Nearest Neighbors Algorithm. Recuperado 5 de junio de 2020, de <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- [8] Chen & Zhang, M. (2015). Linear and non-linear models for purchase prediction. In **Proceedings of the 2015 International ACM Recommender Systems Challenge** (pp. 1-4).
- [9] S. (2019) sharmaroshan/Online-Shoppers-Purchasing-Intention. Recuperado 20 de junio de 2020, de <https://github.com/sharmaroshan/Online-Shoppers-Purchasing-Intention>
- [10] Baati & Mohsil (2020). Real-Time Prediction of Online Shoppers' Purchasing Intention Using Random Forest., In **IFIP International Conference on Artificial Intelligence Applications and Innovations** (pp. 43-51). Springer, Cham.

- [11] *Brownlee (2019)*. A Gentle Introduction to k-fold Cross-Validation. Recuperado 20 de junio de 2020, de <https://machinelearningmastery.com/k-fold-cross-validation/>
- [12] *Buttrey & Karo (2002)*. Using k-nearest-neighbor classification in the leaves of a tree. **Computational Statistics & Data Analysis**, 40(1), 27-37.