Proyecto 1

Mario Andrade Vargas (201711906)
María José Ruíz García (202010184)
Juan Camilo Villamarin (201816898)

Universidad de los Andes

Inteligencia de negocios

2021

1. Comprensión del negocio y enfoque analítico

Oportunidad/problema Negocio	Se quiere saber cómo es el movimiento del bitcoin en el mercado según la opinión de los tweets en cada día de la semana, teniendo en cuenta palabras clave usadas en redes sociales sobre el bitcoin.
Descripción del requerimiento desde el punto de vista de aprendizaje de máquina	Se requiere clasificar tweets sobre bitcoin, según unas palabras claves encontradas en un análisis de sentimientos previamente realizado, para ver cuales son los tweets que afectan positiva y negativamente el precio del bitcoin.

Detalles de la actividad de minería de datos		
Tarea	**Técnica**	**Algoritmo e hiper-parámetros utilizados (con la justificación respectiva)**
Clasificar los tweets de acuerdo a las palabras claves encontradas y el día de la semana de la publicación de estos.	Clasificación	Support Vector Machines (Mario Andrade) Hiperparametros: - Kernel : Lineal - C = 0.1 - gamma = 0.1 La justificación de la selección de estos hiper-parámetros se hace en el apartado de modelado.
		Arbol de decision (Juan Villamarín) Hiperparametros: -Criterion: entropy

		-Max_depth: 4 -Min_samples_split: 2
		KNN (María José Ruiz): -Best p:1 - Metric ‘Manhattan’ -Best n_neighbours:9

2. Comprensión de los datos y preparación de los datos

Para dar inicio al análisis, se realizó la importación de los datos y se determinó la cantidad de filas con un total de 19344048 y las columnas con un total de 3

```
# Se cargan los datos.
df=pd.read_csv('mbsa.csv',  delimiter=",", header=0)
# Cantidad de datos y número de variables
df.shape
```

✓ 1m 25.1s

```
(19344048, 3)
```

Al visualizar la tabla se obtienen los atributos Date, text y Sentiment.

```
# Mostrar los datos
df.head()
```

✓ 0.9s

	Date	text	Sentiment
0	2019-05-27	È appena uscito un nuovo video! LES CRYPTOMONN...	Positive
1	2019-05-27	Cardano: Digitize Currencies; EOS https://t.co...	Positive
2	2019-05-27	Another Test tweet that wasn't caught in the s...	Positive
3	2019-05-27	Current Crypto Prices! \n\nBTC: $8721.99 USD\n...	Positive
4	2019-05-27	Spiv (Nosar Baz): BITCOIN Is An Asset & NO...	Positive

Se toma la decisión de usar una muestra de los datos ya que son demasiados para procesarlos

```
# Toma de muestra de los datos
dft = df.sample(50000)
```

Se filtran los tweets por idioma para tener solo los tweets en inglés. Para esto se usó la librería “langdetect” que identifica el idioma de un texto.

```
# funcion para asignar idioma
def getLanguage (row):
    try:
        return detect(row['text'])
    except:
        return ''

# filtrar por idoma idioma
dft['language']=dft.apply (lambda row: getLanguage(row), axis=1)
dft=dft[dft['language']=='en']
```

Se transforman los datos de la columna Sentiment para darles una representación numérica donde Positive es 1 y Negative es 0

```
# Transformar texto a valor numerico
dft['Sentiment'].replace({'Positive':1, 'Negative':0}, inplace = True)
dft=dft[dft['Sentiment'].isin([1,0])]
```

Para la columna Date se filtran las fechas inválidas. De esta manera se obtienen los registros únicamente con fechas válidas.

```
def is_date(dates):
    result = []
    for date in dates:
        try:
            datetime.datetime.strptime(date, '%Y-%m-%d')
            result.append(date)
        except ValueError:
            a=''
    return result

validDates = is_date(dft['Date'])

dft=dft[dft['Date'].isin(validDates)]
```

Como parte del análisis vamos a tener en cuenta la importancia del día de la semana en el que se realiza el tweet. El día de la semana se calcula a partir de la columna Date, donde se asigna un valor de 0 a 6, siendo 0 lunes,1 martes, 2 miércoles, 3 jueves, 4 viernes, 5 sábado y 6 domingo

```
# Ahora definimos la función que nos va a permitir construir nuestra clase.
def labelWeekDay (row):
    return datetime.datetime.strptime(row['Date'], '%Y-%m-%d').weekday()

dft['weekday_label']=dft.apply (lambda row: labelWeekDay(row), axis=1)
```

En el análisis se decide tener en cuenta la cantidad de veces que una palabra clave aparece en un único tweet. Para esto se agregan columnas en las que se guardará el valor numérico de esta cuenta (Figura 8). Las palabras claves fueron seleccionadas con respecto a un artículo publicado por la revista científica *Royal Society Open Science* donde se determina qué palabras importan a medida que cambia la dinámica de precios del Bitcoin (Burnie & Yilmaz, 2019).

```
def labelKeyWord(row,word):
    return row['text'].count(word)

#Seleccion de palabras clave
keyWords=['bitcoin','$','buy','get','use','blockchain','market','crypto','time','price','wallet','transact','exchange', 'cryptocurr']

for word in keyWords:
    dft[word]=dft.apply (lambda row: labelKeyWord(row, word), axis=1)
```

Para terminar la preparación de los datos se eliminan las columnas donde se tenía el idioma del tweet, la fecha y el texto del tweet obteniendo la siguiente tabla

```
dft = dft.drop(['text', 'Date', 'language'], axis=1)
dft.head()
```

✓ 0.4s

	Sentiment	weekday_label	bitcoin	$	buy	get	use	blockchain	market	crypto	time	price	wallet	transact	exchange	cryptocurr
16020949	0.0	5	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1282301	0.0	6	2	0	0	0	0	0	0	0	0	0	0	0	0	0
5807147	0.0	3	0	0	0	0	0	0	0	1	0	0	0	0	0	0
12787632	0.0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	1
13680025	0.0	6	0	0	0	0	0	0	0	0	0	0	1	0	0	0

3. Modelado y evaluación

Árbol de decisión (Juan Camilo Villamarín): Se hizo uso de la seed:3301

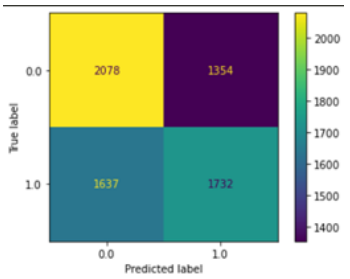
Se selecciona la variable objetivo (Sentiment)

```
# Se selecciona la variable objetivo, en este caso "Sentiment".
Y = dft['Sentiment']
# Del conjunto de datos se elimina la variable "Sentiment"
X = dft.drop(['Sentiment'], axis=1)
```

Se dividen los datos en los de pruebas y los de entrenamiento

```
# Dividir los datos en entrenamiento y test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

Se crea el árbol de decisión y la matriz de confusión obteniendo los siguientes resultados



	precision	recall	f1-score	support
0.0	0.56	0.61	0.58	3432
1.0	0.56	0.51	0.54	3369
accuracy			0.56	6801
macro avg	0.56	0.56	0.56	6801
weighted avg	0.56	0.56	0.56	6801

A partir de los datos de entrenamiento se procede a dividirse en dos que serán los nuevos datos de entrenamiento y los de prueba para la búsqueda de la mejor combinación de hiperparametros

```
# Dividimos el conjunto de entrenamiento en dos: una para la construcción del modelo (sería el nuevo conjunto de
# entrenamiento) y otro para la validación, el cuál será utilizado para determinar el rendimiento del modelo con una
# combinación específica de hiperparámetros.
X_trainval, X_val, Y_trainval, Y_val = train_test_split(X_train, Y_train, test_size = 0.15, random_state = 0)
```

Se establece el espacio de búsqueda de los hiperparametros se escogen valores de profundidad no muy grandes ya que basado en lo visto en clase los valores de profundidad muy grandes pueden dar resultados no concluyentes.

```
# Establecemos el espacio de búsqueda para los hiperparámetros que deseamos ajustar.
param_grid = {'criterion':['gini', 'entropy'],'max_depth':[4,6,8,10,20],'min_samples_split':[2, 3, 4, 5]}
```

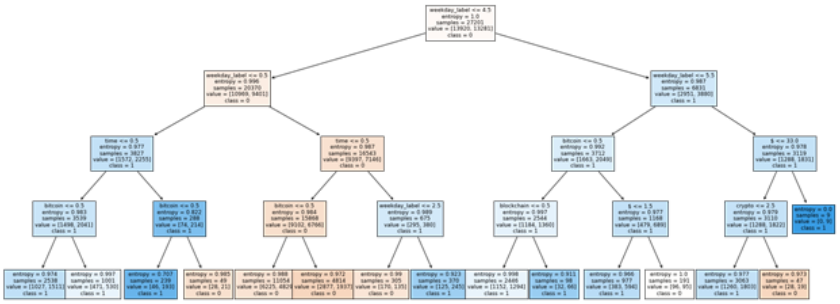
Se realiza la búsqueda del mejor modelo con GridSearch sobre el espacio definido y se crea el nuevo modelo

```
# Ahora utilizamos GridSearch sobre el grid definido y con 10 particiones en la validación cruzada.
mejor_modelo = GridSearchCV(arbol, param_grid, cv=particiones)
# Ajuste del modelo
mejor_modelo.fit(X_train, Y_train)
```

Se obtienen los siguientes valores con el nuevo modelo

	precision	recall	f1-score	support
0.0	0.57	0.69	0.62	3432
1.0	0.59	0.47	0.52	3369
accuracy			0.58	6801
macro avg	0.58	0.58	0.57	6801
weighted avg	0.58	0.58	0.57	6801

Árbol resultante



Support Vector Machines (Mario Andrade)

Se seleccionó el algoritmo de Support Vector Machines debido a que es recomendado utilizarlo en problemas donde la variable objetivo requiere una clasificación binaria. En nuestro caso, la variable objetivo Sentiment, fue ajustada para que tomara dos valores, 1 o 0, por lo que se beneficia de esto.

Se seleccionaron los datos de entrenamiento, que son el 80% del dataset,y los datos de prueba, que son el 20% restante. Esos datos se tomaron con una random seed de valor 3301, que es la misma que los dos otros modelos, por consistencia.

Inicialmente se utilizó el hiper-parámetro kernel, con valor lineal, ya que los valores que tomaban las columnas de cada palabra clave, eran números pequeños, porque los tweets normalmente son párrafos muy cortos. Los resultados son los siguientes:

	precision	recall	f1-score	support
0.0	0.50	0.94	0.65	3414
1.0	0.50	0.06	0.11	3397
accuracy			0.50	6811
macro avg	0.50	0.50	0.38	6811
weighted avg	0.50	0.50	0.38	6811

Los resultados son decepcionantes, especialmente en el grupo de positivos (1), donde el recall es de 0.06, por lo cual se va a buscar hiper-parámetros más optimizados para este modelo. La optimización se hace con una grid con unos valores posibles por parámetro:

```
#Hiperparametros para probar
param_grid = {'C': [0.1,1, 10],
              'gamma': [0.1,0.01,0.001],
              'kernel': ['linear']}
```

Los valores de C son los castigos que se dan por no clasificar bien los datos, y entre más grande su valor, el algoritmo va a tratar de minimizar el número de puntos mal clasificados. Los valores de gamma dicen que tan cerca deben estar los datos para estar clasificados en el mismo grupo; en caso de que sean más grande el valor, los puntos deben estar más cerca.

El valor del kernel si fue únicamente lineal, descartando las opciones polinómicas por la naturaleza de los datos; como lo mencione anteriormente, son valores muy pequeños que no varían mucho.

El resultado de utilizar los hiper-parámetros mejor optimizados fue el siguiente:

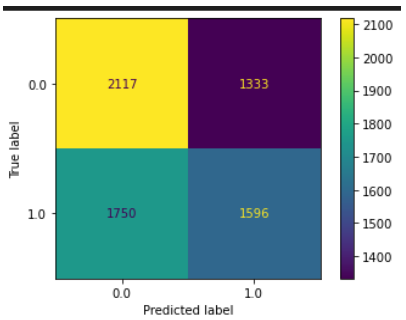
	precision	recall	f1-score	support
0.0	0.50	0.94	0.65	3414
1.0	0.51	0.06	0.11	3397
accuracy			0.50	6811
macro avg	0.51	0.50	0.38	6811
weighted avg	0.51	0.50	0.38	6811

Básicamente el mismo resultado. En el apartado de Resultados se hablará de las posibles razones por las cuales esto sucedió.

KNN (María José Ruíz)

Para empezar,el algoritmo KNN se caracteriza por su fácil implementación ,es un algoritmo de aprendizaje lento y la fase de entrenamiento no es explícita,sin embargo es una fase rápida. En nuestro caso, la variable objetivo Sentiment, fue ajustada para que tomara dos valores, 1 o 0, por lo que se beneficia de esto.

Para hacer la selección de los datos de entrenamiento se escogieron el 80% de ellos y el restante eran los de prueba.Para la seed se escogió la 3301 que corresponde a la utilizada en los otros modelos.



Se hizo una primera prueba con vecinos y p aleatoria y se obtuvo una precisión del 55% y un recall del 61%

	precision	recall	f1-score	support
0.0	0.55	0.61	0.58	3450
1.0	0.54	0.48	0.51	3346
accuracy			0.55	6796
macro avg	0.55	0.55	0.54	6796
weighted avg	0.55	0.55	0.54	6796

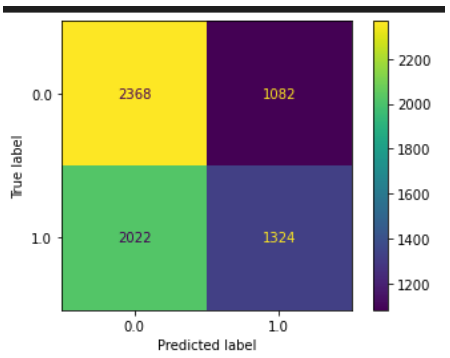
Luego se determinaron los mejores hiper parámetros son p de 1,métrica manhattan y 9 vecinos

```
Best p: 1
Best n_neighbors: 9
```

Cuando se probó con los nuevos hiper parámetros la precisión bajó un punto y el recall subió al 69%

	precision	recall	f1-score	support
0.0	0.54	0.69	0.60	3450
1.0	0.55	0.40	0.46	3346
accuracy			0.54	6796
macro avg	0.54	0.54	0.53	6796
weighted avg	0.54	0.54	0.53	6796

Se puede observar que en la matriz de confusión existe cierto balance entre los valores



4. Resultados

Con el modelo que se utilizó el algoritmo de árboles de decisión, encontramos que este modelo tiene una precisión y recall de aproximadamente el 50% para la clasificación positiva, por lo que podemos esperar que la mitad de los tweets sean correctamente clasificados, mientras que para la clasificación negativa es las métricas son mejores, por lo que se espera poder clasificar estos tweets mejor.

En el modelo en el que se hizo uso del algoritmo de Support Vector Machines, el resultado no fue tan bueno como en el modelo anterior; la precisión y especialmente el recall fueron muy bajos para la clasificación de tweets positivos; mientras que la clasificación de negativos es aceptable. Sin embargo, al intentar optimizar los hiper-parámetros, los resultados se mantienen iguales y el modelo no es lo suficientemente bueno para recomendarlo para resolver el problema de negocio.

El modelo con el algoritmo de KNN, las métricas presentan resultados similares a el modelo de árboles de decisión, tanto como para la clasificación de tweets positivos, como negativos. Al buscar y aplicar los hiper-parámetros optimizados, se observan resultados levemente mejores, pero no es una mejora suficiente para decir que supera a árboles de decisión.

Teniendo en cuenta todo esto, el modelo que recomendamos para clasificar la influencia del tweet sobre el precio del bitcoin, es el que aplica el algoritmo de árboles de decisión. Recomendamos el uso de este, no solo por tener mejores puntajes en las métricas, sino también porque ayuda a entender cómo se relacionan las palabras claves de los tweets. Además, nos ayudó a encontrar que el día de la semana es la variable más influyente para clasificar los tweets, que es parte de lo que busca responder este proyecto.

5. Referencias

Burnie, A., & Yilmaz, E. (2019). Social media and bitcoin metrics: which words matter. *Royal Society Open Science*, *6*(10), 191068. https://doi.org/10.1098/rsos.191068