Consigna

Para este trabajo practico primeramente se debía extraer información para lo cual usamos 3 métodos

- API
- SCRAPING
- MANUALMENTE

Recolección de datos

SCRAPING

Usamos beautifulsup ya que posee algunas funcionalidades que nos sirve para extraer información de la web

```
texto_com = ""
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
paragraphs = soup.find_all('p')
titulo = soup.title.text.strip()
texto_com = " ".join(paragraph.text for paragraph in paragraphs)
```

Se sacará los datos de la página de Bing

```
target_url=f"https://www.bing.com/news/search?q={cripto}&mkt={market}"
headers={"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWe
resp=requests.get(target_url,headers=headers)
so
fo                                                    '):
  bingUrl = result.select_one('.title')['href']
  l.append(scrape_url(bingUrl))
```

(variable) bingUrl: str | list[str] | Any

API

Se realiza el pedido de información de la siguiente manera.

```
url="https://newsapi.org/v2/everything"
api_key="12053cc7ad9a40ea900480710c3b6b01"
```

Con los parámetros

```
query="btc"
language='en'
from_date='2023-10-00'
to_date='2023-11-11'
```

Como se observa los datos pedidos son de bitcoin y las noticias en inglés (que serían más completas)

```
response=requests.get(url,params=params)
data=response.json()
```

Luego se realiza una serie de pasos como:

a) Cargar los datos: que se obtuvieron en formato json a un dataframe para podernos manejar mejor

b) Normalizamos:

```
# Traemos nuevamente las stopwords
stopwords = nltk.corpus.stopwords.words('english')

titular_list=[]
for titular in df.titulo:
    # Vamos a reemplzar los caracteres que no sean leras por espacios
    titular=re.sub("[^a-zA-Z]"," ",str(titular))
    # Pasamos todo a minúsculas
    titular=titular.lower()
    # Tokenizamos para separar las palabras del titular
    titular=nltk.word_tokenize(titular)
    # Eliminamos las palabras de menos de 3 letras
    titular = [palabra for palabra in titular if len(palabra)>3]
    # Sacamos las Stopwords
    titular = [palabra for palabra in titular if not palabra in stopwords]

    ## Hasta acá Normalizamos, ahora a stemmizar
    # Aplicamos la funcion para buscar la raiz de las palabras
    titular=[stemmer.stem(palabra) for palabra in titular]
    # Por ultimo volvemos a unir el titular
    titular=" ".join(titular)

    # Vamos armando una lista con todos los titulares
    titular_list.append(titular)
```

Con **`corpus.stopwords.words('english')`** se quitan palabras no relevantes, dividimos la cadena de texto en una lista de palabras.

```
stemmer.stem(palabra)
```

Lo que hace stemmer es reducir la palabra en su forma base eliminando prefijos o sufijos, lo cual nos servirá para agrupar palabras relacionadas para simplificar el análisis.

```
model_name = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name)
```

Usamos este modelo preentrenado bert-base-uncased, lo cual me dará una clasificación numerica según el model_name.

La columna sentiment puede tener 3 tipos de valores

```
# LABEL_0 negativa

# LABEL_1 positiva

# LABEL_2 neutra
```

c) Vectorizar

```
is_recomend = df['sentiment'].apply(lambda x: int(x[-1])).values
```

CountVectorizer convierte la colección de texto que teníamos anteriormente sobre las noticias en la columna news del dataframe a en una matriz de frecuencia de tokens

d) Modelamos: usaremos 2 modelos

```
print('acc : ', nb.score(xtest,ytest))
```

```
print('acc : ', svc.score(xtest,ytest))
```

Creamos un clasificador d maquinas vectoriales evaluamos el error con el score lo que me da el error entre los datos de testeo

Guardaremos el modelo y el vectorizado para luego usarlo

```
joblib.dump(cou_vec, 'count_vectorizer.pkl')
```

Guardar datos

Guardaremos os datos en una base de datos mysql

Información de la criptomoneda

SENTIMIENTO: ya se obtuvo anteriormente y se guardó.

NER:

Cargo todas las noticias como un solo string

```
noticias = " ".join(noticia.lower() for noticia in df_read['news'] )
```

Luego usando el método spice el cual me permitirá el procesamiento para clasificar entidades

```
doc = nlp(sentence)

# Crear listas para almacenar los resultados
entities = []
categories = []

# Iterar sobre las entidades identificadas
for entity in doc.ents:
    entities.append(entity.text)
    categories.append(entity.label_)
```

FRECUENCIA

Realizamos una nube de palabras para identificar las frecuencias de las palabras que mas se repitieron que denotaran las de mayor frecuencia por su tamaño

Lo cual se verificará con la función, lo cual básicamente tokenizara(separar las palabras) saca la frecuencia y ordena de mayor a menor

```
return df_frec
```

El resultado es el siguiente y su frecuencia

Extra

En este apartado se realizará una estratega que complemente la parte de como va el presio ya sea que suba o baje de tal manera que complemente con el análisis de sentimiento

Estrategia

Lo que dice es si la posición de df[low] de la 5ta pocicion > a la cuarta > a la tercera > a la segunda y así hasta la primera.

```
df["support"] = np.nan
df["resistance"] = np.nan

df.loc[(df["low"].shift(5) > df["low"].shift(4)) &
       (df["low"].shift(4) > df["low"].shift(3)) &
       (df["low"].shift(3) > df["low"].shift(2)) &
       (df["low"].shift(2) > df["low"].shift(1)) &
       (df["low"].shift(1) > df["low"].shift(0)), "support"] = df["low"]


df.loc[(df["high"].shift(5) < df["high"].shift(4)) &
(df["high"].shift(4) < df["high"].shift(3)) &
(df["high"].shift(3) < df["high"].shift(2)) &
(df["high"].shift(2) < df["high"].shift(1)) &
(df["high"].shift(1) < df["high"].shift(0)), "resistance"] = df["high"]
```

Actuará como una resistencia, así mismo la parte de df[high] actuara como soporte

Df[SMA fast]Actúan como promedios entre periodos en este caso de 30 y de 60 tienden a ser una curva mas suave

```
# Create Simple moving average 30 days
df["SMA fast"] = df["close"].rolling(30).mean()
# Create Simple moving average 60 days
df["SMA slow"] = df["close"].rolling(60).mean()
```

La condición de compra seria que el promedio rápido de 30 rompa la de 60

```
condition_2_buy = df["SMA fast"] > df["SMA slow"]
```

La de venta similar, pero en este caso de la 60 es mayor a la de 30.

```
condition_2_sell = df["SMA fast"] < df["SMA slow"]
```

Serie temporal

Traemos datos de metatrader5 1304 datos que por default lo pusimos horas

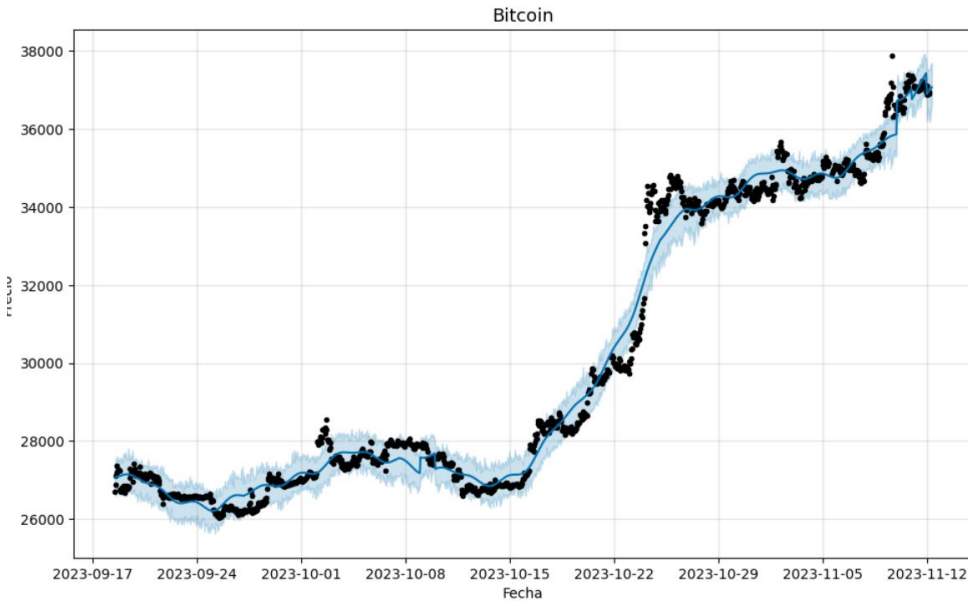
```
bit=preprocessing_mt5("BITCOIN",1304)
bit.tail(3)
```

Usamos el modelo **`Prophet`** porque se me hizo más fácil el uso

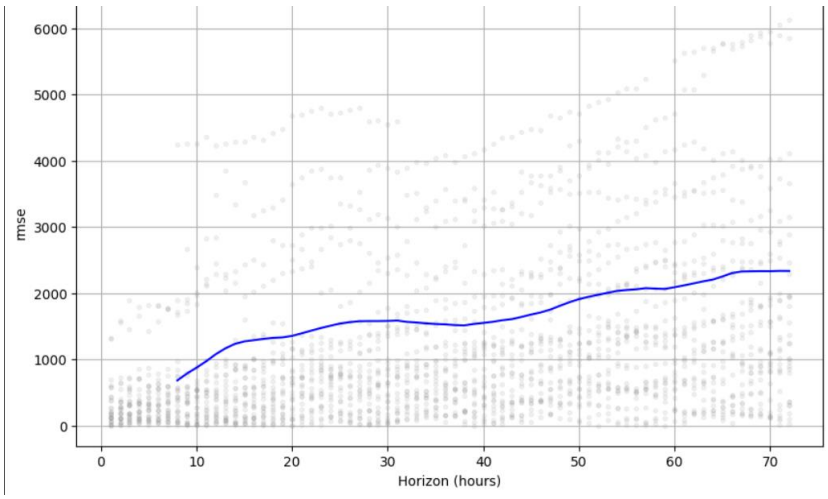
Por ejemplo para esta parte de código se vio que se predijo para las próximas 4 horas

```
future = model.make_future_dataframe(periods=4, freq='H')
forecast = model.predict(future)
```

Que dio como resultado el grafico siguiente, se ve además que la tendencia es alcista en línea azul por lo que se podría pensar que el valor a predecir será mayor que la actual



Se hallo el rmse para 72 h siguientes y como se ve tiene un error promedio de 1694



Predicción

Se uso 3 modelos una regresión lineal, regresión bayesiana y otra k-Nearest con el metamodelo staking

```
stack_model =  StackingRegressor(estimators=estimator_list, final_estimator=DecisionTreeRegressor
(max_depth=5))
```

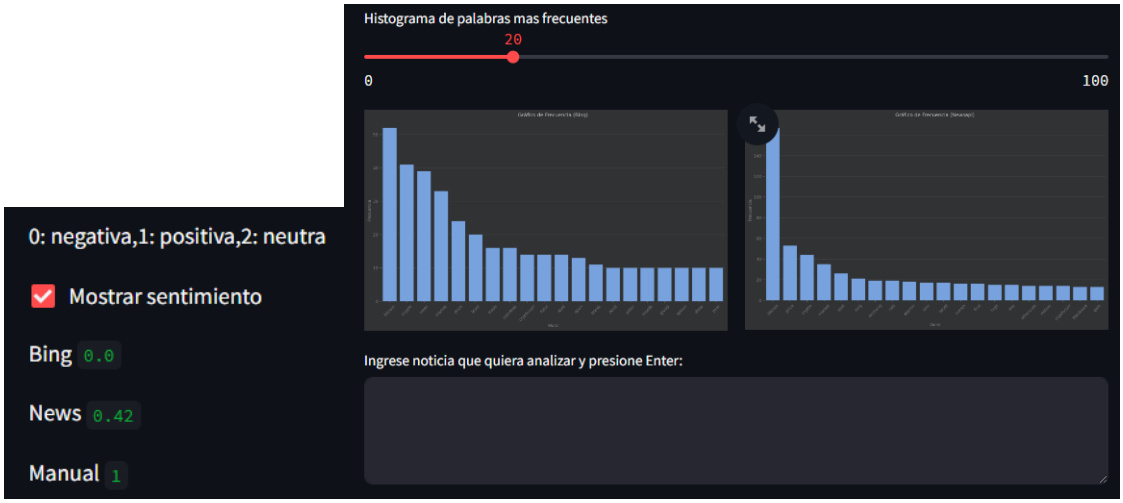
Por último, guardamos el modelo para usar a futuro, donde se observó que a periodos relativamente bajos los valores de serie temporal y de predicción son similares.

Teniendo la:

- estrategia que me dice que compre o no con las EMAS de 30 y de 60 periodos y resistencias
- Análisis de sentimiento según la información actualizada importante del día
- Las series y temporales y la predicción que predecirán el precio de la cripto

Se vaya a la suba o a la baja se podrá tener una mejor decisión a la hora de invertir

Una pequeña visualización que hice se podrá realzar usando streamlit en donde hay un cuadro donde se tendrá que poner una noticia y con el modelo guardado tendría que darte la calificación de emoción del mismo, como se ve también da calificación de bing o de newsapi que son los lugares donde se extrajo la información además de un histograma de palabras repetidas



El de series temporales te da un gráfico en donde podrás manipular el periodo de tiempo

Por último el de series temporales como se dijo los valores estaban muy parecidos

Conclusiones

Las criptomonedas son muy volátiles pueden depender de información sobre los Exchange la venta o compra de grandes inversores para lograr sacar ganancias, para ello se vio el análisis de sentimiento de las noticias, un análisis más técnico como cuáles son sus mínimos o máximos que pueda tener porque es mala inversión comprar en un valor tope cuando todos venderán que es lo que hicimos con las resistencias y los promedios. Por último, tratamos de predecir hacia donde ira el precio de nuestra criptomoneda con series temporales y la predicción del metamodelo stakin, cada persona tratara de sacar sus propias conclusiones acerca del mismo y los riesgos que deba tomar.