# Ethereum Close Prediction Model


## Author: Eric Romano


# Overview

During the week of May 15th , the market experienced a volatile environment for cryptocurrencies with Bitcoins plugging to half of it all time high of 60K. A total of 830 billion lost by investors causing many investors to go bankrupt.

## Business Problem

Many retail investors would like to have tools to have at there disposal to make predictions that can lead to better returns or better exit strategies. The crypto market is strongly connected to retail investors and their emotions. Using Sentiment Analysis and other metrics such as Relative Strength Index we believe these will be useful in creating tools that will classify if the market closes higher or lower than the day before. The first crypto currency that will be explored in is Ethereum and will use these models to use for other top trending crypto currencies.

1. Can we successfully use sentiment analysis features to classify if the market will close higher or lower than the previous day?
2. Can we extract features using financial technical analysis tools such as Relative Strength Index that is used when evaluating momentum? To visualize and isolate conditions of overbought and oversold
3. Can previous historical data, such as the day before, be a strong feature used in modeling?
4. What are the important features of the models and can we extract promising insight for these features that can be used in creating investment strategies?

In [1]:
```
#Import  the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from textblob import TextBlob
import re
import time
from datetime import datetime
import matplotlib.dates as mdates
import matplotlib.ticker as ticker
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, precis
ion_score, recall_score
from sklearn.metrics import roc_curve, auc, roc_curve, confusion_matrix
from sklearn import metrics

import seaborn as sns
sns.set_style('whitegrid')
```

In [2]:
```
#Store headlines data into df_ethereum_headlines
df_ethereum_headlines = pd.read_csv('df_ethereum_headlines.csv')
#Remove the first column
df_ethereum_headlines= df_ethereum_headlines.drop('Unnamed: 0', axis=1)
```

In [3]:
```
#Show the dataset
df_ethereum_headlines.head(15)
```

Out[3]:

| | headlines | news_center | time | website |
|---|---|---|---|---|
| 0 | Here’s What Could Be Next for Bitcoin, Ethereu... | The Daily Hodl | 5-23-2021 | https://dailyhodl.com/2021/05/23/heres-what-co... |
| 1 | How Leveraged Positions Could Have Accelerated... | Bitcoinist.com | 5-23-2021 | https://coinmarketcap.com/headlines/news/how-l... |
| 2 | Confirmed: Ethereum’s Berlin hard fork solved ... | Finbold | 5-23-2021 | https://finbold.com/confirmed-ethereums-berlin... |
| 3 | These 3 factors will determine if Ethereum’s r... | AMBCrypto | 5-23-2021 | https://ambcrypto.com/these-3-factors-will-det... |
| 4 | Bitcoin & Ethereum: Here’s the reality check o... | AMBCrypto | 5-23-2021 | https://ambcrypto.com/bitcoin-ethereum-heres-t... |
| 5 | Crypto Market Tanks 14% to 3-Month Low Under $... | Coingape | 5-23-2021 | https://coinmarketcap.com/headlines/news/crypt... |
| 6 | Goldman Sachs: Ethereum (ETH) Might Overtake B... | Coingape | 5-23-2021 | https://coinmarketcap.com/headlines/news/goldm... |
| 7 | Why did Bitcoin and Ethereum’s price drop so q... | AMBCrypto | 5-22-2021 | https://ambcrypto.com/why-did-bitcoin-and-ethe... |
| 8 | Why Ethereum’s Vitalik Buterin doesn’t ‘really... | AMBCrypto | 5-22-2021 | https://ambcrypto.com/why-ethereums-vitalik-bu... |
| 9 | British MP says Ethereum ‘flippening’ is takin... | CryptoSlate | 5-21-2021 | https://coinmarketcap.com/headlines/news/briti... |
| 10 | British Member of Parliament: Ethereum Will Fl... | Decrypt | 5-21-2021 | https://decrypt.co/71633/british-member-of-par... |
| 11 | Could Dreams Of Ethereum Flippening Bitcoin Be... | Bitcoinist.com | 5-21-2021 | https://coinmarketcap.com/headlines/news/could... |
| 12 | Crypto Crash Cost Ethereum Boss His Billionair... | NewsBTC | 5-21-2021 | https://coinmarketcap.com/headlines/news/crypt... |
| 13 | Bitcoin Cash, Ethereum Classic, Filecoin Price... | AMBCrypto | 5-21-2021 | https://eng.ambcrypto.com/bitcoin-cash-ethereu... |
| 14 | How Ethereum plans to get around a major drag ... | Quartz | 5-21-2021 | https://qz.com/2011329/ethereums-recovery-is-t... |

In [4]:
```
# Make NLTK think like a financial journalist
import nltk
nltk.download('vader_lexicon')

# NLTK VADER for sentiment analysis
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# New words and values
new_words = {
    'crushes': 10,
    'beats': 5,
    'misses': -5,
    'trouble': -10,
    'falls': -100,
}

# Instantiate the sentiment intensity analyzer with the existing lexicon
vader = SentimentIntensityAnalyzer()
# Update the lexicon
vader.lexicon.update(new_words)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\egust\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [5]:
```
#Iterate through the headlines and get the polarity scores
scores =[vader.polarity_scores(headline) for headline in df_ethereum_headlines
.headlines]
#Convert the list of dicts into a df
scores_df = pd.DataFrame(scores)
scores_df
```

Out[5]:

| | neg | neu | pos | compound |
|---|---|---|---|---|
| **0** | 0.000 | 0.893 | 0.107 | 0.2023 |
| **1** | 0.000 | 1.000 | 0.000 | 0.0000 |
| **2** | 0.335 | 0.539 | 0.126 | -0.3612 |
| **3** | 0.000 | 1.000 | 0.000 | 0.0000 |
| **4** | 0.000 | 1.000 | 0.000 | 0.0000 |
| **...** | ... | ... | ... | ... |
| **280** | 0.000 | 0.753 | 0.247 | 0.3818 |
| **281** | 0.000 | 0.857 | 0.143 | 0.1280 |
| **282** | 0.000 | 0.850 | 0.150 | 0.3182 |
| **283** | 0.000 | 0.760 | 0.240 | 0.3680 |
| **284** | 0.000 | 1.000 | 0.000 | 0.0000 |

285 rows × 4 columns

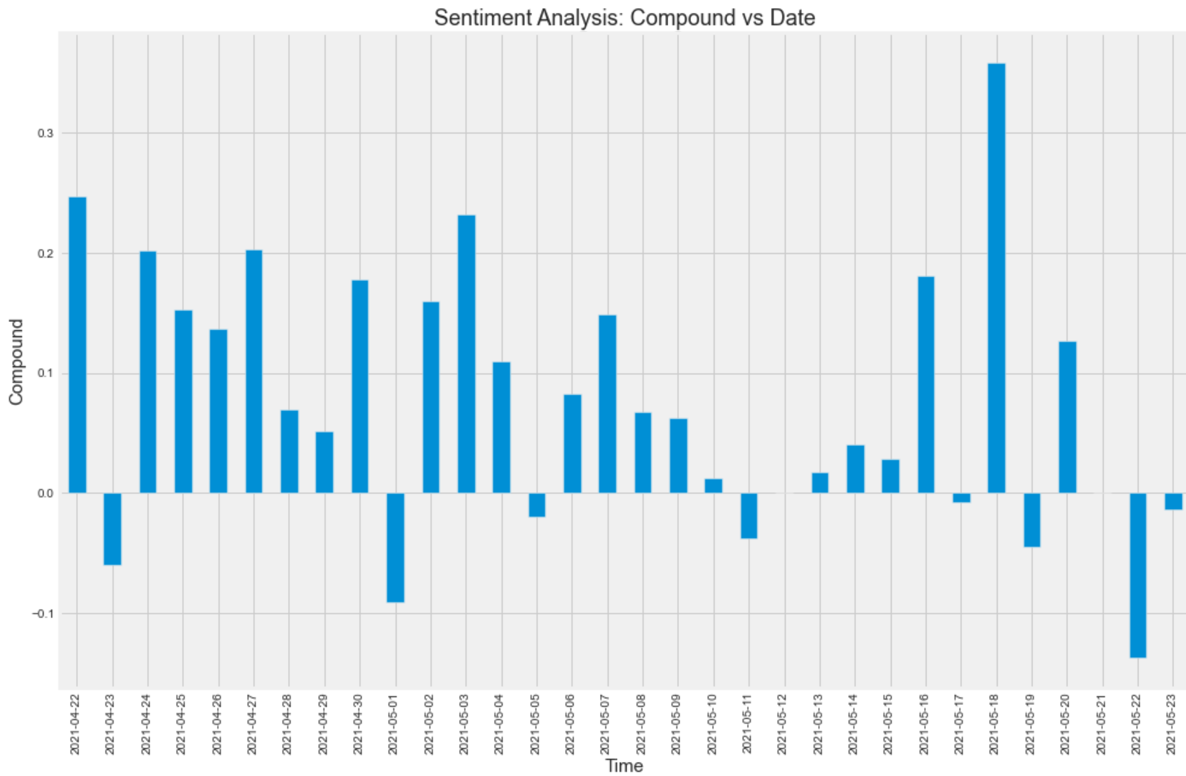
In [6]:
```
# Join the dataframes
df_ethereum_headlines = df_ethereum_headlines.join(scores_df)
```

In [7]:
```
# Convert the date column from string to datetime
df_ethereum_headlines['time'] = pd.to_datetime(df_ethereum_headlines.time).dt.
date
df_ethereum_headlines.head()
```

Out[7]:

| | headlines | news_center | time | website | neg | neu | |
|---|---|---|---|---|---|---|---|
| **0** | Here's What Could Be Next for Bitcoin, Ethereu... | The Daily Hodl | 2021-05-23 | https://dailyhodl.com/2021/05/23/heres-what-co... | 0.000 | 0.893 | 0. |
| **1** | How Leveraged Positions Could Have Accelerated... | Bitcoinist.com | 2021-05-23 | https://coinmarketcap.com/headlines/news/how-l... | 0.000 | 1.000 | 0. |
| **2** | Confirmed: Ethereum's Berlin hard fork solved ... | Finbold | 2021-05-23 | https://finbold.com/confirmed-ethereums-berlin... | 0.335 | 0.539 | 0. |
| **3** | These 3 factors will determine if Ethereum's r... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/these-3-factors-will-det... | 0.000 | 1.000 | 0. |
| **4** | Bitcoin & Ethereum: Here's the reality check o... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/bitcoin-ethereum-heres-t... | 0.000 | 1.000 | 0. |

In [8]:
```
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
%matplotlib inline

# Group by date and ticker columns from scored_news and calculate the mean
mean_c = df_ethereum_headlines.groupby(['time']).mean()
# Get the cross-section of compound in the 'columns' axis
mean_c = mean_c.xs('compound', axis='columns')
# Plot a bar chart with pandas
plt.figure(figsize = (15,10))
mean_c.plot.bar(figsize = (15,10))
plt.ylabel('Compound', size = 15)
plt.xlabel('Time', size = 15)
plt.title('Sentiment Analysis: Compound vs Date', size = 18)
plt.savefig('Sentiment_Analysis_Compound_vs_Date')
plt.show()
```




In [9]:
```
df_mean_c = pd.DataFrame(mean_c)
df_mean_c.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 32 entries, 2021-04-22 to 2021-05-23
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   compound  32 non-null     float64
dtypes: float64(1)
memory usage: 512.0+ bytes
```

In [10]:
```
df_mean_c['date'] = df_mean_c.index
df_mean_c.reset_index(drop=True, inplace=True)
df_mean_c.head()
```

Out[10]:

| | compound | date |
|---|---|---|
| **0** | 0.246581 | 2021-04-22 |
| **1** | -0.059558 | 2021-04-23 |
| **2** | 0.202300 | 2021-04-24 |
| **3** | 0.152933 | 2021-04-25 |
| **4** | 0.136886 | 2021-04-26 |

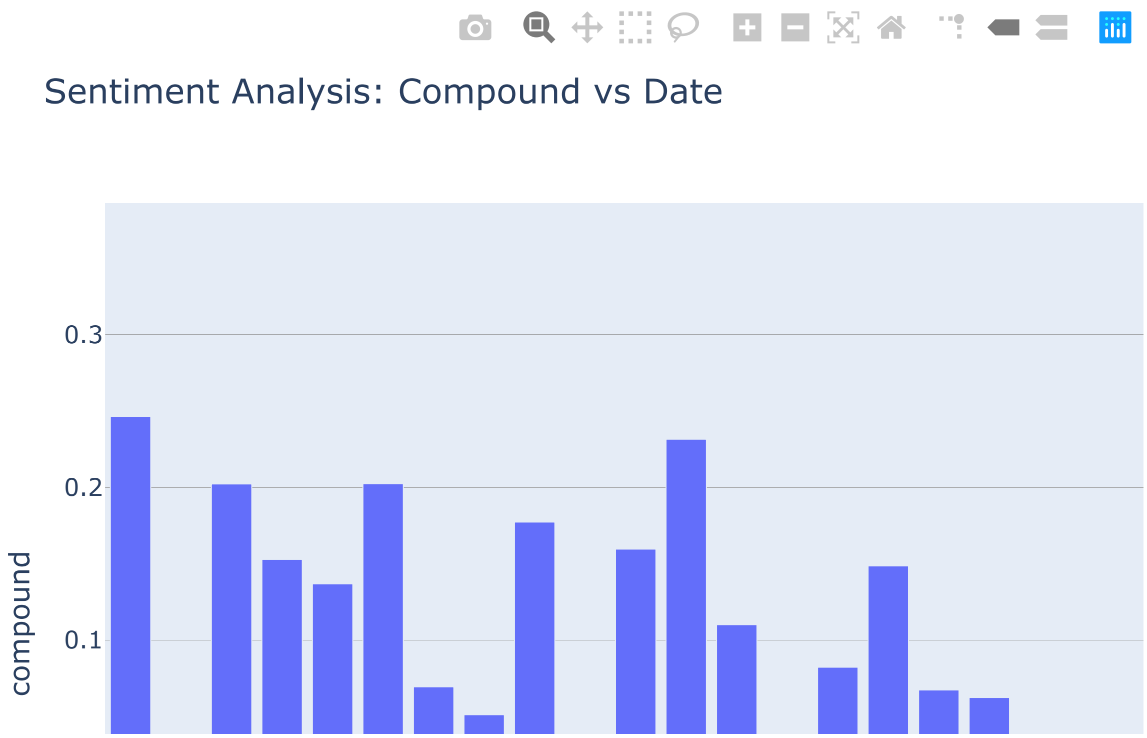
In [11]:
```
pip install psutil
```

```
Requirement already satisfied: psutil in c:\users\egust\anaconda3\envs\learn-
env\lib\site-packages (5.8.0)
Note: you may need to restart the kernel to use updated packages.
```

In [12]:
```
conda install -c plotly plotly-orca
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [13]:
```
import plotly.express as px
fig = px.bar(df_mean_c, x= 'date', y= 'compound', title='Sentiment Analysis: C
ompound vs Date')
fig.show()
fig.write_image('images/fig.png')
```




In [14]:
```
#Check if there are any headline duplicates
num_news_before = df_ethereum_headlines.headlines.count()
print(num_news_before)
#Drop any duplicates
df_ethereum_headlines_clean = df_ethereum_headlines.drop_duplicates(subset=['h
eadlines'])
#Check if there are any remaining duplicates
num_news_after = df_ethereum_headlines_clean.headlines.count()
print(num_news_after)
```

```
285
285
```

In [15]:
```
#list of all the dates that is in df_ethereum_headlines
headlines_list = df_ethereum_headlines.time.unique().tolist()

#All headlines for specified date
grouped_headlines = []
for i in range(len(headlines_list)):
    grouped_headlines.append(' '.join( str(x) for x in df_ethereum_headlines.l
oc[df_ethereum_headlines.time == headlines_list[i]].headlines))
```

In [16]:
```
#Show aggregated news text
grouped_headlines[0]
```

Out[16]:
```
'Here’s What Could Be Next for Bitcoin, Ethereum, and Two Low-Cap Altcoins, A
ccording to Top Trader How Leveraged Positions Could Have Accelerated Ethereu
m’s Slump Confirmed: Ethereum’s Berlin hard fork solved a flaw that exposed t
he network to attacks These 3 factors will determine if Ethereum’s recovery w
ill be sustainable Bitcoin & Ethereum: Here’s the reality check on their pric
e trajectories Crypto Market Tanks 14% to 3-Month Low Under $1.35 Trillion, E
thereum (ETH) Under $2000 Goldman Sachs: Ethereum (ETH) Might Overtake Bitcoi
n (BTC) As A Store of Value'
```

In [17]:
```
#Create dataframe for the group headlines
grouped_headlines_df = pd.DataFrame(data={'date':headlines_list, 'headlines':g
rouped_headlines})
grouped_headlines_df.head()
```

Out[17]:

| | date | headlines |
|---|---|---|
| **0** | 2021-05-23 | Here’s What Could Be Next for Bitcoin, Ethereu... |
| **1** | 2021-05-22 | Why did Bitcoin and Ethereum’s price drop so q... |
| **2** | 2021-05-21 | British MP says Ethereum ‘flippening’ is takin... |
| **3** | 2021-05-20 | Bank of Canada: Intrinsic Value of Bitcoin, Et... |
| **4** | 2021-05-19 | How Justin Sun Almost Caused Ethereum To Drop ... |

In [18]:
```
# Load data
historical_data = pd.read_csv('historical_data.csv')
```

In [19]:
```
# Convert the date column from string to datetime
historical_data.date = pd.to_datetime(historical_data.date).dt.date

# Drop first column
historical_data = historical_data.drop('Unnamed: 0', axis=1)

historical_data
```

Out[19]:

| | date | open | high | low | close | volume | market_cap |
|---|---|---|---|---|---|---|---|
| **0** | 2021-06-02 | $2,634.46 | $2,801.39 | $2,555.40 | $2,706.12 | $27,723,267,359 | $314,266,256,163 |
| **1** | 2021-06-01 | $2,707.56 | $2,739.74 | $2,531.16 | $2,633.52 | $27,363,223,090 | $305,798,597,367 |
| **2** | 2021-05-31 | $2,387.20 | $2,715.86 | $2,279.51 | $2,714.95 | $31,007,383,150 | $315,217,277,483 |
| **3** | 2021-05-30 | $2,278.29 | $2,472.19 | $2,188.83 | $2,390.31 | $25,876,619,428 | $277,492,990,927 |
| **4** | 2021-05-29 | $2,414.07 | $2,566.94 | $2,208.49 | $2,279.51 | $33,773,720,220 | $264,600,384,052 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **358** | 2020-06-09 | $246.18 | $248.34 | $242.34 | $244.91 | $8,446,545,788 | $27,254,678,255 |
| **359** | 2020-06-08 | $245.18 | $246.64 | $241.54 | $246.31 | $8,076,783,299 | $27,406,923,226 |
| **360** | 2020-06-07 | $241.91 | $245.44 | $236.33 | $245.17 | $9,544,883,157 | $27,276,459,502 |
| **361** | 2020-06-06 | $241.20 | $245.98 | $239.72 | $241.93 | $8,114,873,845 | $26,913,140,925 |
| **362** | 2020-06-05 | $244.35 | $247.33 | $240.68 | $241.22 | $9,293,963,914 | $26,830,954,614 |

363 rows × 7 columns

In [20]:
```
#Create a label column
historical_data['label'] = 'label'
historical_data
```

Out[20]:

| | date | open | high | low | close | volume | market_cap | label |
|---|---|---|---|---|---|---|---|---|
| **0** | 2021-06-02 | $2,634.46 | $2,801.39 | $2,555.40 | $2,706.12 | $27,723,267,359 | $314,266,256,163 | label |
| **1** | 2021-06-01 | $2,707.56 | $2,739.74 | $2,531.16 | $2,633.52 | $27,363,223,090 | $305,798,597,367 | label |
| **2** | 2021-05-31 | $2,387.20 | $2,715.86 | $2,279.51 | $2,714.95 | $31,007,383,150 | $315,217,277,483 | label |
| **3** | 2021-05-30 | $2,278.29 | $2,472.19 | $2,188.83 | $2,390.31 | $25,876,619,428 | $277,492,990,927 | label |
| **4** | 2021-05-29 | $2,414.07 | $2,566.94 | $2,208.49 | $2,279.51 | $33,773,720,220 | $264,600,384,052 | label |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **358** | 2020-06-09 | $246.18 | $248.34 | $242.34 | $244.91 | $8,446,545,788 | $27,254,678,255 | label |
| **359** | 2020-06-08 | $245.18 | $246.64 | $241.54 | $246.31 | $8,076,783,299 | $27,406,923,226 | label |
| **360** | 2020-06-07 | $241.91 | $245.44 | $236.33 | $245.17 | $9,544,883,157 | $27,276,459,502 | label |
| **361** | 2020-06-06 | $241.20 | $245.98 | $239.72 | $241.93 | $8,114,873,845 | $26,913,140,925 | label |
| **362** | 2020-06-05 | $244.35 | $247.33 | $240.68 | $241.22 | $9,293,963,914 | $26,830,954,614 | label |

363 rows × 8 columns

In [21]:
```
for i in range(0,(len(historical_data)-1)):
    if historical_data['close'][i] > historical_data['close'][i+1]:
        historical_data['label'][i] = 1 # The market closed higher than the da
y before
    elif historical_data['close'][i] < historical_data['close'][i+1]:
        historical_data['label'][i] = 0 # The market closed lower than the day
before
    else:
        historical_data['label'][i] = 2 # market stay the same
```

In [22]:
```
# Show data
historical_data
```

Out[22]:

| | date | open | high | low | close | volume | market_cap | label |
|---|---|---|---|---|---|---|---|---|
| **0** | 2021-06-02 | $2,634.46 | $2,801.39 | $2,555.40 | $2,706.12 | $27,723,267,359 | $314,266,256,163 | 1 |
| **1** | 2021-06-01 | $2,707.56 | $2,739.74 | $2,531.16 | $2,633.52 | $27,363,223,090 | $305,798,597,367 | 0 |
| **2** | 2021-05-31 | $2,387.20 | $2,715.86 | $2,279.51 | $2,714.95 | $31,007,383,150 | $315,217,277,483 | 1 |
| **3** | 2021-05-30 | $2,278.29 | $2,472.19 | $2,188.83 | $2,390.31 | $25,876,619,428 | $277,492,990,927 | 1 |
| **4** | 2021-05-29 | $2,414.07 | $2,566.94 | $2,208.49 | $2,279.51 | $33,773,720,220 | $264,600,384,052 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **358** | 2020-06-09 | $246.18 | $248.34 | $242.34 | $244.91 | $8,446,545,788 | $27,254,678,255 | 0 |
| **359** | 2020-06-08 | $245.18 | $246.64 | $241.54 | $246.31 | $8,076,783,299 | $27,406,923,226 | 1 |
| **360** | 2020-06-07 | $241.91 | $245.44 | $236.33 | $245.17 | $9,544,883,157 | $27,276,459,502 | 1 |
| **361** | 2020-06-06 | $241.20 | $245.98 | $239.72 | $241.93 | $8,114,873,845 | $26,913,140,925 | 1 |
| **362** | 2020-06-05 | $244.35 | $247.33 | $240.68 | $241.22 | $9,293,963,914 | $26,830,954,614 | label |

363 rows × 8 columns

In [23]:
```
df_merge = pd.merge(historical_data, grouped_headlines_df)
df_merge
```

Out[23]:

| | date | open | high | low | close | volume | market_cap | label | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2021-05-23 | $2,298.37 | $2,384.41 | $1,737.47 | $2,109.58 | $56,005,721,977 | $244,704,904,961 | 0 | C |
| **1** | 2021-05-22 | $2,436.01 | $2,483.98 | $2,168.12 | $2,295.71 | $42,089,937,660 | $266,263,966,984 | 0 | |
| **2** | 2021-05-21 | $2,772.34 | $2,938.21 | $2,113.35 | $2,430.62 | $53,774,070,802 | $281,879,243,639 | 0 | sa |
| **3** | 2021-05-20 | $2,439.64 | $2,993.15 | $2,170.23 | $2,784.29 | $67,610,826,680 | $322,857,390,499 | 1 | Ir of |
| **4** | 2021-05-19 | $3,382.66 | $3,437.94 | $1,952.46 | $2,460.68 | $84,482,912,776 | $285,298,709,245 | 0 | |
| **5** | 2021-05-18 | $3,276.87 | $3,562.47 | $3,246.40 | $3,380.07 | $40,416,525,218 | $391,850,295,263 | 1 | S Se |
| **6** | 2021-05-17 | $3,581.34 | $3,587.77 | $3,129.01 | $3,282.40 | $54,061,732,774 | $380,482,843,865 | 0 | |
| **7** | 2021-05-16 | $3,641.83 | $3,878.90 | $3,350.95 | $3,587.51 | $47,359,478,734 | $415,801,534,962 | 0 | R |
| **8** | 2021-05-15 | $4,075.95 | $4,129.19 | $3,638.12 | $3,638.12 | $42,422,321,751 | $421,619,090,683 | 0 | t |
| **9** | 2021-05-14 | $3,720.12 | $4,171.02 | $3,703.40 | $4,079.06 | $48,174,271,215 | $472,663,570,788 | 1 | Ar As |
| **10** | 2021-05-13 | $3,828.92 | $4,032.56 | $3,549.41 | $3,715.15 | $78,398,214,539 | $430,445,282,301 | 0 | T |
| **11** | 2021-05-12 | $4,174.64 | $4,362.35 | $3,785.85 | $3,785.85 | $69,023,382,175 | $438,585,075,674 | 0 | In etl |

| | date | open | high | low | close | volume | market_cap | label | |
|---|---|---|---|---|---|---|---|---|---|
| **12** | 2021-05-11 | $3,948.27 | $4,178.21 | $3,783.89 | $4,168.70 | $52,679,737,865 | $482,881,900,491 | 1 | Etl |
| **13** | 2021-05-10 | $3,924.41 | $4,197.47 | $3,684.45 | $3,952.29 | $62,691,789,007 | $457,761,219,807 | 1 | |
| **14** | 2021-05-09 | $3,911.46 | $3,981.26 | $3,743.99 | $3,928.84 | $50,568,290,278 | $454,991,994,900 | 1 | C w |
| **15** | 2021-05-08 | $3,481.99 | $3,950.16 | $3,453.77 | $3,902.65 | $50,208,491,286 | $451,905,650,094 | 1 | E |
| **16** | 2021-05-07 | $3,490.11 | $3,573.29 | $3,370.26 | $3,484.73 | $39,607,240,515 | $403,465,702,897 | 0 | S& C |
| **17** | 2021-05-06 | $3,524.93 | $3,598.90 | $3,386.24 | $3,490.88 | $44,300,394,788 | $404,131,394,792 | 0 | Cı |
| **18** | 2021-05-05 | $3,240.55 | $3,541.46 | $3,213.10 | $3,522.78 | $48,334,198,383 | $407,777,080,466 | 1 | T |
| **19** | 2021-05-04 | $3,431.13 | $3,523.59 | $3,180.74 | $3,253.63 | $62,402,045,158 | $376,577,399,574 | 0 | Et Cr |
| **20** | 2021-05-03 | $2,951.18 | $3,450.04 | $2,951.18 | $3,431.09 | $49,174,290,212 | $397,069,786,479 | 1 | inv |
| **21** | 2021-05-02 | $2,945.56 | $2,984.89 | $2,860.53 | $2,952.06 | $28,032,013,047 | $341,593,133,886 | 1 | E |
| **22** | 2021-05-01 | $2,772.84 | $2,951.44 | $2,755.91 | $2,945.89 | $28,726,205,272 | $340,840,444,354 | 1 | Th |
| **23** | 2021-04-30 | $2,757.73 | $2,796.05 | $2,728.17 | $2,773.21 | $29,777,179,889 | $320,822,874,721 | 1 | St D |

| | date | open | high | low | close | volume | market_cap | label | |
|---|---|---|---|---|---|---|---|---|---|
| **24** | 2021-04-29 | $2,748.65 | $2,797.97 | $2,672.11 | $2,756.88 | $32,578,127,990 | $318,896,956,051 | 1 | ( |
| **25** | 2021-04-28 | $2,664.69 | $2,757.48 | $2,564.08 | $2,746.38 | $34,269,031,076 | $317,645,696,234 | 1 | El Bil |
| **26** | 2021-04-27 | $2,534.03 | $2,676.39 | $2,485.38 | $2,662.87 | $32,275,969,215 | $307,950,352,777 | 1 | T F $ |
| **27** | 2021-04-26 | $2,319.48 | $2,536.34 | $2,308.32 | $2,534.48 | $35,208,325,408 | $293,069,092,286 | 1 | Et |
| **28** | 2021-04-25 | $2,214.41 | $2,354.09 | $2,172.52 | $2,316.06 | $31,814,355,546 | $267,780,847,561 | 1 | 4 W |
| **29** | 2021-04-24 | $2,367.20 | $2,367.74 | $2,163.69 | $2,211.63 | $31,854,226,936 | $255,676,477,656 | 0 | W tc |
| **30** | 2021-04-23 | $2,401.26 | $2,439.54 | $2,117.04 | $2,363.59 | $55,413,933,925 | $273,212,191,169 | 0 | [ H |
| **31** | 2021-04-22 | $2,357.87 | $2,641.09 | $2,315.96 | $2,403.54 | $53,575,904,724 | $277,797,467,179 | 1 | T S |

In [24]:
```
# create a function to get the subjectivity
def getSubjectivity(text):
    return TextBlob(text).sentiment.subjectivity

#Create a fuction to get the polarity
def getPolarity(text):
    return TextBlob(text).sentiment.polarity
```

In [25]:
```
# Create two columns
# Subjectivity
df_merge['Subjectivity'] = df_merge['headlines'].apply(getSubjectivity)
# Polarity
df_merge['Polarity'] = df_merge['headlines'].apply(getPolarity)
```

In [26]:
```
# Show Columns
df_merge.head()
```

Out[26]:

| | date | open | high | low | close | volume | market_cap | label | he |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2021-05-23 | $2,298.37 | $2,384.41 | $1,737.47 | $2,109.58 | $56,005,721,977 | $244,704,904,961 | 0 | C I Et |
| **1** | 2021-05-22 | $2,436.01 | $2,483.98 | $2,168.12 | $2,295.71 | $42,089,937,660 | $266,263,966,984 | 0 | \ Bitc Ethe pri |
| **2** | 2021-05-21 | $2,772.34 | $2,938.21 | $2,113.35 | $2,430.62 | $53,774,070,802 | $281,879,243,639 | 0 | Bri Et ‘flip is |
| **3** | 2021-05-20 | $2,439.64 | $2,993.15 | $2,170.23 | $2,784.29 | $67,610,826,680 | $322,857,390,499 | 1 | C \ |
| **4** | 2021-05-19 | $3,382.66 | $3,437.94 | $1,952.46 | $2,460.68 | $84,482,912,776 | $285,298,709,245 | 0 | Hov , Et To |

In [27]:
```
# Create a function to get the sentiment scores
def getSent(text):
    sent = SentimentIntensityAnalyzer()
    sentiment = sent.polarity_scores(text)
    return sentiment
```

In [28]:
```
# Get the sentiment scores for each day
compound = []
pos = []
neg = []
neu = []
SENT =0

for i in range(0, len(df_merge)):
    SENT = getSent(df_merge['headlines'][i])
    compound.append(SENT['compound'])
    pos.append(SENT['pos'])
    neg.append(SENT['neg'])
    neu.append(SENT['neu'])
```

In [29]:
```
#Store the sentiment data onto dataframe
df_merge['Compund']= compound
df_merge['pos'] = pos
df_merge['neg'] = neg
df_merge['neu'] = neu
# Show dataframe
df_merge.head()
```

Out[29]:

| | date | open | high | low | close | volume | market_cap | label | he |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2021-05-23 | $2,298.37 | $2,384.41 | $1,737.47 | $2,109.58 | $56,005,721,977 | $244,704,904,961 | 0 | C<br>I<br>Et |
| **1** | 2021-05-22 | $2,436.01 | $2,483.98 | $2,168.12 | $2,295.71 | $42,089,937,660 | $266,263,966,984 | 0 | \<br>Bitc<br>Ethe<br>pri |
| **2** | 2021-05-21 | $2,772.34 | $2,938.21 | $2,113.35 | $2,430.62 | $53,774,070,802 | $281,879,243,639 | 0 | Bri<br>Et<br>‘flip<br>is |
| **3** | 2021-05-20 | $2,439.64 | $2,993.15 | $2,170.23 | $2,784.29 | $67,610,826,680 | $322,857,390,499 | 1 | C<br>\ |
| **4** | 2021-05-19 | $3,382.66 | $3,437.94 | $1,952.46 | $2,460.68 | $84,482,912,776 | $285,298,709,245 | 0 | Hov<br>‘<br>Et<br>To |

In [30]:
```
# Create a list of columns to keep
columns = ['Subjectivity', 'Polarity', 'Compund', 'pos', 'neg', 'neu', 'label'
]
df = df_merge[columns]
df
```

Out[30]:

| | Subjectivity | Polarity | Compund | pos | neg | neu | label |
|---|---|---|---|---|---|---|---|
| **0** | 0.468333 | 0.121667 | -0.1027 | 0.070 | 0.086 | 0.844 | 0 |
| **1** | 0.350000 | 0.266667 | -0.2732 | 0.000 | 0.100 | 0.900 | 0 |
| **2** | 0.356548 | 0.119940 | 0.0000 | 0.042 | 0.042 | 0.916 | 0 |
| **3** | 0.294111 | -0.057444 | 0.6921 | 0.120 | 0.050 | 0.830 | 1 |
| **4** | 0.250000 | -0.316667 | 0.5423 | 0.091 | 0.058 | 0.851 | 0 |
| **5** | 0.463573 | 0.263447 | 0.9808 | 0.245 | 0.000 | 0.755 | 1 |
| **6** | 0.600000 | 0.112500 | -0.0258 | 0.031 | 0.043 | 0.926 | 0 |
| **7** | 0.000000 | 0.000000 | 0.3612 | 0.102 | 0.000 | 0.898 | 0 |
| **8** | 0.666667 | 0.333333 | 0.1372 | 0.060 | 0.052 | 0.888 | 0 |
| **9** | 0.477083 | 0.106629 | 0.8808 | 0.131 | 0.052 | 0.817 | 1 |
| **10** | 0.441667 | 0.300000 | 0.1280 | 0.042 | 0.026 | 0.932 | 0 |
| **11** | 0.341136 | 0.131591 | 0.0926 | 0.044 | 0.044 | 0.912 | 0 |
| **12** | 0.363704 | 0.123333 | -0.3947 | 0.021 | 0.044 | 0.935 | 1 |
| **13** | 0.393056 | -0.086458 | -0.1154 | 0.051 | 0.049 | 0.900 | 1 |
| **14** | 0.389646 | 0.239394 | 0.6249 | 0.034 | 0.000 | 0.966 | 1 |
| **15** | 0.450000 | 0.250000 | 0.2023 | 0.039 | 0.000 | 0.961 | 1 |
| **16** | 0.483074 | -0.001602 | 0.8299 | 0.086 | 0.019 | 0.895 | 0 |
| **17** | 0.563119 | 0.137008 | 0.7476 | 0.115 | 0.061 | 0.824 | 0 |
| **18** | 0.382686 | 0.106157 | -0.1531 | 0.036 | 0.041 | 0.923 | 1 |
| **19** | 0.363140 | 0.226942 | 0.9201 | 0.119 | 0.029 | 0.852 | 0 |
| **20** | 0.572917 | 0.141667 | 0.8979 | 0.142 | 0.000 | 0.858 | 1 |
| **21** | 0.650000 | 0.350000 | 0.5859 | 0.107 | 0.034 | 0.859 | 1 |
| **22** | 0.612727 | 0.001818 | -0.3612 | 0.000 | 0.056 | 0.944 | 1 |
| **23** | 0.512500 | 0.154167 | 0.7579 | 0.151 | 0.035 | 0.814 | 1 |
| **24** | 0.680272 | 0.123469 | 0.6124 | 0.059 | 0.025 | 0.916 | 1 |
| **25** | 0.242424 | 0.147727 | 0.6786 | 0.043 | 0.009 | 0.948 | 1 |
| **26** | 0.199318 | 0.099545 | 0.8910 | 0.135 | 0.000 | 0.865 | 1 |
| **27** | 0.401936 | 0.071970 | 0.9042 | 0.114 | 0.035 | 0.850 | 1 |
| **28** | 0.633333 | 0.000000 | 0.4588 | 0.100 | 0.000 | 0.900 | 1 |
| **29** | 0.000000 | 0.000000 | 0.2023 | 0.141 | 0.000 | 0.859 | 0 |
| **30** | 0.368254 | 0.121230 | -0.6124 | 0.062 | 0.080 | 0.858 | 0 |
| **31** | 0.497637 | 0.261963 | 0.9757 | 0.168 | 0.000 | 0.832 | 1 |

In [31]:
```
df['label'] = pd.to_numeric(df['label'])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32 entries, 0 to 31
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Subjectivity  32 non-null     float64
 1   Polarity      32 non-null     float64
 2   Compund       32 non-null     float64
 3   pos           32 non-null     float64
 4   neg           32 non-null     float64
 5   neu           32 non-null     float64
 6   label         32 non-null     int64  
dtypes: float64(6), int64(1)
memory usage: 3.2 KB
```

```
C:\Users\egust\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.
py:1: SettingWithCopyWarning: 


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [32]:
```
#Create the feature data set
X = df
X = np.array(X.drop(['label'], 1))
# Target dataset
y = np.array(df['label'])
```

In [33]:
```
# Split the data into 80% training and 20% test
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, ran
dom_state =0)
```

In [34]:
```
#Create and train the model
#Instatiate
LDA = LinearDiscriminantAnalysis()
model = LDA.fit(x_train, y_train)
```

In [35]:
```
# Show the models prediction
pred = model.predict(x_test)

# show the model metrics
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         4
           1       0.60      1.00      0.75         3

    accuracy                           0.71         7
   macro avg       0.80      0.75      0.71         7
weighted avg       0.83      0.71      0.70         7
```

## Create a pipeline

1. Remove punctuation
2. Tokenization
3. Remove stopwords
4. Lemmatize/Stem

In [36]:
```
import nltk
import string
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\egust\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\egust\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\egust\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\egust\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[36]: True

In [37]:
```
stopwords = nltk.corpus.stopwords.words('english')
ps = nltk.PorterStemmer()
wn = nltk.WordNetLemmatizer()
string.punctuation
```

Out[37]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [38]:
```
def clean_headlines(headline):
    headline = "".join([word for word in headline if word not in string.punctuation])
    tokens = re.split('\W+', headline)
    headline = [wn.lemmatize(word) for word in tokens if word not in stopwords]
    return headline

df_merge['headline_clean'] = df_merge['headlines'].apply(lambda x: clean_headlines(x.lower()))
```

In [39]:
```
df_merge.head()
```

Out[39]:

| | date | open | high | low | close | volume | market_cap | label | he |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2021-05-23 | $2,298.37 | $2,384.41 | $1,737.47 | $2,109.58 | $56,005,721,977 | $244,704,904,961 | 0 | C I Et |
| **1** | 2021-05-22 | $2,436.01 | $2,483.98 | $2,168.12 | $2,295.71 | $42,089,937,660 | $266,263,966,984 | 0 | \ Bitc Eth pri |
| **2** | 2021-05-21 | $2,772.34 | $2,938.21 | $2,113.35 | $2,430.62 | $53,774,070,802 | $281,879,243,639 | 0 | Bri Et 'flip is |
| **3** | 2021-05-20 | $2,439.64 | $2,993.15 | $2,170.23 | $2,784.29 | $67,610,826,680 | $322,857,390,499 | 1 | ( \ |
| **4** | 2021-05-19 | $3,382.66 | $3,437.94 | $1,952.46 | $2,460.68 | $84,482,912,776 | $285,298,709,245 | 0 | Hov Et To |

In [40]:
```
df_increase = df_merge.loc[df_merge['label'] == 1 ]
df_increase.head()
```

Out[40]:

| | date | open | high | low | close | volume | market_cap | label | he |
|---|---|---|---|---|---|---|---|---|---|
| **3** | 2021-05-20 | $2,439.64 | $2,993.15 | $2,170.23 | $2,784.29 | $67,610,826,680 | $322,857,390,499 | 1 | ( |
| **5** | 2021-05-18 | $3,276.87 | $3,562.47 | $3,246.40 | $3,380.07 | $40,416,525,218 | $391,850,295,263 | 1 | Co<br>C |
| **9** | 2021-05-14 | $3,720.12 | $4,171.02 | $3,703.40 | $4,079.06 | $48,174,271,215 | $472,663,570,788 | 1 | E |
| **12** | 2021-05-11 | $3,948.27 | $4,178.21 | $3,783.89 | $4,168.70 | $52,679,737,865 | $482,881,900,491 | 1 | E<br>I |
| **13** | 2021-05-10 | $3,924.41 | $4,197.47 | $3,684.45 | $3,952.29 | $62,691,789,007 | $457,761,219,807 | 1 | E<br>Pr<br>Up |

In [41]:
```
df_decrease = df_merge.loc[df_merge['label'] == 0 ]
df_decrease.head()
```

Out[41]:

| | date | open | high | low | close | volume | market_cap | label | he |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2021-05-23 | $2,298.37 | $2,384.41 | $1,737.47 | $2,109.58 | $56,005,721,977 | $244,704,904,961 | 0 | C l Et \ |
| **1** | 2021-05-22 | $2,436.01 | $2,483.98 | $2,168.12 | $2,295.71 | $42,089,937,660 | $266,263,966,984 | 0 | Bitc Ethe pri |
| **2** | 2021-05-21 | $2,772.34 | $2,938.21 | $2,113.35 | $2,430.62 | $53,774,070,802 | $281,879,243,639 | 0 | Bri Et 'flip is |
| **4** | 2021-05-19 | $3,382.66 | $3,437.94 | $1,952.46 | $2,460.68 | $84,482,912,776 | $285,298,709,245 | 0 | Hov ' Et To |
| **6** | 2021-05-17 | $3,581.34 | $3,587.77 | $3,129.01 | $3,282.40 | $54,061,732,774 | $380,482,843,865 | 0 | F Co Tra E |

In [42]:
```
# Most/Least frequent words
increase_list = [] # list containing all words that are involved with market t
o close higher then previous day
for x in df_increase['headline_clean']: # loop over list in df
    increase_list += x # append elements of lists to full list

increase_val_counts = pd.Series(increase_list).value_counts()
increase_val_counts
```

Out[42]:
```
ethereum    167
bitcoin      45
eth          17
crypto       16
could        15
           ...
future        1
much          1
fallout       1
capacity      1
charity       1
Length: 720, dtype: int64
```

In [43]:
```
increase_val_counts[-20:]
```

Out[43]:
```
121            1
american       1
bridge         1
influencer     1
turning        1
dogg           1
altcoin        1
recordhigh     1
hong           1
state          1
peter          1
interested     1
rubicon        1
breaking       1
postsnl        1
future         1
much           1
fallout        1
capacity       1
charity        1
dtype: int64
```

In [44]:
```
# Most/Least frequent words
decrease_list = [] # list containing all words that are involved with market t
o close low then previous day
for x in df_decrease['headline_clean']: # loop over list in df
    decrease_list += x # append elements of lists to full list

decrease_val_counts = pd.Series(decrease_list).value_counts()
decrease_val_counts
```

Out[44]:
```
ethereum       114
bitcoin         39
crypto          27
price           12
eth             12
              ...
ethan            1
condominium      1
momentum         1
determine        1
warns            1
Length: 532, dtype: int64
```

In [45]:
```
from textblob import TextBlob, Word
from wordcloud import WordCloud
```

In [46]:
```
wordcloud = WordCloud(max_words=100, width=400, height=200).generate(str(increase_val_counts))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.figure(figsize=(20,10))
plt.show()
```

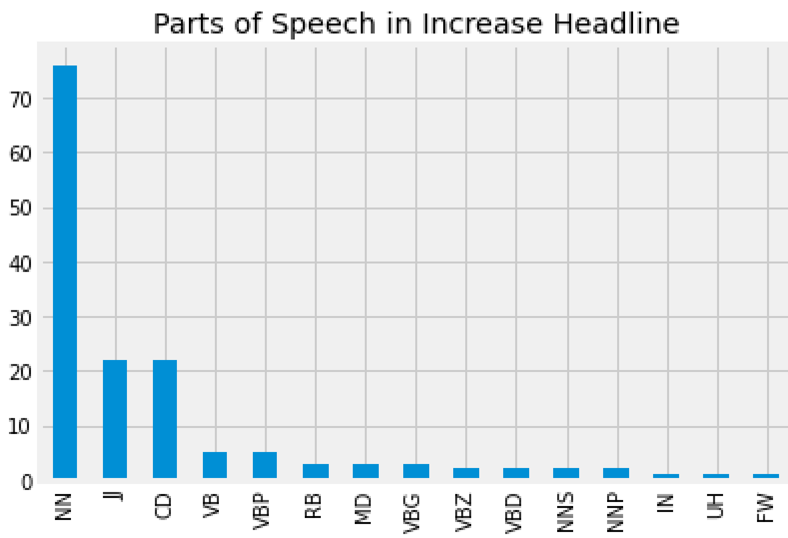

```
<Figure size 1440x720 with 0 Axes>
```

In [47]:
```
wordcloud = WordCloud(max_words=100, width=400, height=200).generate(str(decrease_val_counts))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.figure(figsize=(20,10))
plt.show()
```


```
<Figure size 1440x720 with 0 Axes>
```

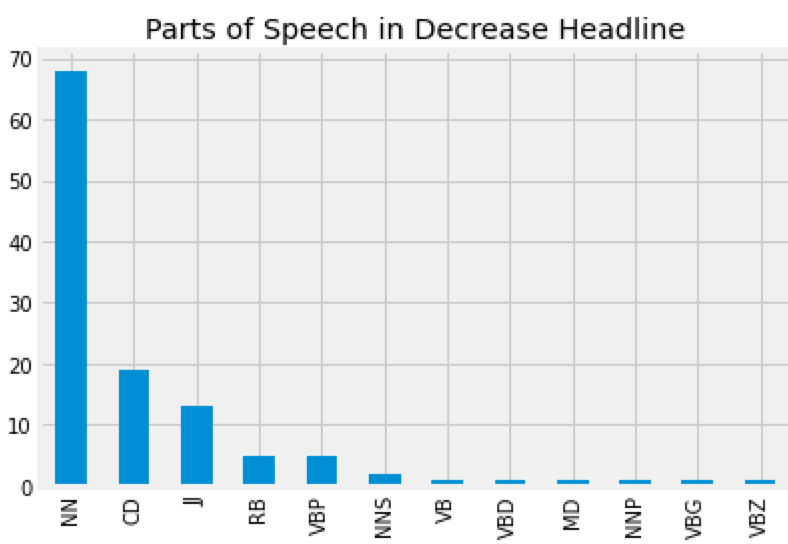
In [48]:
```
blob = TextBlob(str(df_increase['headline_clean']))
pos_df = pd.DataFrame(blob.tags, columns = ['word' , 'pos'])
pos_df = pos_df.pos.value_counts()[:20]
pos_df.plot(kind='bar', title="Parts of Speech in Increase Headline")
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x10fbf23ea20>




In [49]:
```
blob = TextBlob(str(df_decrease['headline_clean']))
pos_df = pd.DataFrame(blob.tags, columns = ['word' , 'pos'])
pos_df = pos_df.pos.value_counts()[:20]
pos_df.plot(kind='bar', title="Parts of Speech in Decrease Headline")
```

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x10fbe6a1048>



In [50]:
```
(pd.Series(nltk.ngrams(increase_list, 2)).value_counts())[:20]
```

Out[50]:
```
(bitcoin, ethereum)      20
(ta, ethereum)           11
(ethereum, blockchain)    5
(alltime, high)           5
(buy, ethereum)           4
(eth, price)              4
(ethereum, price)         4
(ethereum, 20)            4
(european, investment)    4
(bond, ethereum)          4
(eth, could)              4
(investment, bank)        4
(ethereum, rally)         3
(reason, ethereum)        3
(could, trigger)          3
(mark, cuban)             3
(elon, musk)              3
(crypto, analyst)         3
(ethereum, new)           3
(new, ath)                3
dtype: int64
```

In [51]:
```
(pd.Series(nltk.ngrams(decrease_list, 2)).value_counts())[:20]
```

Out[51]:
```
(bitcoin, ethereum)    23
(ta, ethereum)          6
(ethereum, price)       5
(ethereum, etf)         5
(mark, cuban)           4
(ethereum, eth)         4
(sp, dow)               4
(file, ethereum)        4
(vitalik, buterin)      4
(dow, jones)            4
(vaneck, file)          4
(day, ethereum)         3
(crypto, market)        3
(ethereum, index)       3
(ethereum, crypto)      3
(ethereum, classic)     3
(index, debut)          3
(price, drop)           2
(price, rally)          2
(ethereum, defi)        2
dtype: int64
```

In [ ]:

# Model Baseline:

- Model 2: Features engineered via the Sentiment Analysis(SA)
- Model 3: Features engineered via the Relative Strength Index(RSI)
- Model 4: Features engineered via the Historical Data(HD)
- Model 5: Features engineered via the SA and RSI
- Model 6: Features engineered via all features engineered, SA + RSI + HD

In [36]:
```
df_ethereum_headlines['date'] = df_ethereum_headlines['time']
```

In [37]:
```
df_ethereum_headlines.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   headlines    285 non-null    object 
 1   news_center  285 non-null    object 
 2   time         285 non-null    object 
 3   website      285 non-null    object 
 4   neg          285 non-null    float64
 5   neu          285 non-null    float64
 6   pos          285 non-null    float64
 7   compound     285 non-null    float64
 8   date         285 non-null    object 
dtypes: float64(4), object(5)
memory usage: 20.2+ KB
```

In [38]:
```
# Get the sentiment scores for each day
compound = []
pos = []
neg = []
neu = []
SENT =0

for i in range(0, len(df_ethereum_headlines)):
    SENT = getSent(df_ethereum_headlines['headlines'][i])
    compound.append(SENT['compound'])
    pos.append(SENT['pos'])
    neg.append(SENT['neg'])
    neu.append(SENT['neu'])
```

In [39]:
```
#Store the sentiment data onto dataframe
df_ethereum_headlines['Comp']= compound
df_ethereum_headlines['positive'] = pos
df_ethereum_headlines['negative'] = neg
df_ethereum_headlines['neutural'] = neu
# Show dataframe
df_ethereum_headlines.head()
```

Out[39]:

| | headlines | news_center | time | website | neg | neu | |
|---|---|---|---|---|---|---|---|
| **0** | Here's What Could Be Next for Bitcoin, Ethereu... | The Daily Hodl | 2021-05-23 | https://dailyhodl.com/2021/05/23/heres-what-co... | 0.000 | 0.893 | 0. |
| **1** | How Leveraged Positions Could Have Accelerated... | Bitcoinist.com | 2021-05-23 | https://coinmarketcap.com/headlines/news/how-l... | 0.000 | 1.000 | 0. |
| **2** | Confirmed: Ethereum's Berlin hard fork solved ... | Finbold | 2021-05-23 | https://finbold.com/confirmed-ethereums-berlin... | 0.335 | 0.539 | 0. |
| **3** | These 3 factors will determine if Ethereum's r... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/these-3-factors-will-det... | 0.000 | 1.000 | 0. |
| **4** | Bitcoin & Ethereum: Here's the reality check o... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/bitcoin-ethereum-heres-t... | 0.000 | 1.000 | 0. |

In [40]:
```
# Load data
df_historical_data = pd.read_csv('df_historical_data.csv')
# Show data
df_historical_data.head()
```

Out[40]:

| | Unnamed: 0 | date | open | high | low | close | volume | market_cap | RSI |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2021-06-02 | 2634.46 | 2801.39 | 2555.40 | 2706.12 | 27723267359 | 314266256163 | 54.123517 |
| 1 | 1 | 2021-06-01 | 2707.56 | 2739.74 | 2531.16 | 2633.52 | 27363223090 | 305798597367 | 40.235790 |
| 2 | 2 | 2021-05-31 | 2387.20 | 2715.86 | 2279.51 | 2714.95 | 31007383150 | 315217277483 | 42.609654 |
| 3 | 3 | 2021-05-30 | 2278.29 | 2472.19 | 2188.83 | 2390.31 | 25876619428 | 277492990927 | 34.328202 |
| 4 | 4 | 2021-05-29 | 2414.07 | 2566.94 | 2208.49 | 2279.51 | 33773720220 | 264600384052 | 31.930542 |

5 rows × 25 columns

In [41]:
```
# Convert the date column from string to datetime
df_historical_data.date = pd.to_datetime(df_historical_data.date).dt.date

# Drop first column
df_historical_data = df_historical_data.drop('Unnamed: 0', axis=1)

df_historical_data.head()
```

Out[41]:

| | date | open | high | low | close | volume | market_cap | RSI | delta | se |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2021-06-02 | 2634.46 | 2801.39 | 2555.40 | 2706.12 | 27723267359 | 314266256163 | 54.123517 | 72.60 | |
| 1 | 2021-06-01 | 2707.56 | 2739.74 | 2531.16 | 2633.52 | 27363223090 | 305798597367 | 40.235790 | -81.43 | |
| 2 | 2021-05-31 | 2387.20 | 2715.86 | 2279.51 | 2714.95 | 31007383150 | 315217277483 | 42.609654 | 324.64 | |
| 3 | 2021-05-30 | 2278.29 | 2472.19 | 2188.83 | 2390.31 | 25876619428 | 277492990927 | 34.328202 | 110.80 | |
| 4 | 2021-05-29 | 2414.07 | 2566.94 | 2208.49 | 2279.51 | 33773720220 | 264600384052 | 31.930542 | -140.40 | |

5 rows × 24 columns

In [42]:

```
df_historical_data['label'] = 'label'

for i in range(0,(len(df_historical_data)-1)):
    if df_historical_data['close'][i] > df_historical_data['close'][i+1]:
        df_historical_data['label'][i] = 1 # The market closed higher than the
day before
    elif df_historical_data['close'][i] < df_historical_data['close'][i+1]:
        df_historical_data['label'][i] = 0 # The market closed lower than the
 day before
    else:
        df_historical_data['label'][i] = 2 # market stay the same
```

```
C:\Users\egust\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.
py:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\egust\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.
py:7: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [43]:
```
hd = df_historical_data.set_index(pd.to_datetime(df_historical_data['date']))
hd
```

Out[43]:

| date | date | open | high | low | close | volume | market_cap | RSI | delta |
|---|---|---|---|---|---|---|---|---|---|
| **2021-06-02** | 2021-06-02 | 2634.46 | 2801.39 | 2555.40 | 2706.12 | 27723267359 | 314266256163 | 54.123517 | 72.60 |
| **2021-06-01** | 2021-06-01 | 2707.56 | 2739.74 | 2531.16 | 2633.52 | 27363223090 | 305798597367 | 40.235790 | -81.43 |
| **2021-05-31** | 2021-05-31 | 2387.20 | 2715.86 | 2279.51 | 2714.95 | 31007383150 | 315217277483 | 42.609654 | 324.64 |
| **2021-05-30** | 2021-05-30 | 2278.29 | 2472.19 | 2188.83 | 2390.31 | 25876619428 | 277492990927 | 34.328202 | 110.80 |
| **2021-05-29** | 2021-05-29 | 2414.07 | 2566.94 | 2208.49 | 2279.51 | 33773720220 | 264600384052 | 31.930542 | -140.40 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **2020-06-24** | 2020-06-24 | 244.19 | 248.51 | 232.81 | 235.77 | 8815030025 | 26285822239 | 41.619991 | -8.37 |
| **2020-06-23** | 2020-06-23 | 242.54 | 244.86 | 239.76 | 244.14 | 6624530348 | 27215606543 | 49.396457 | 1.61 |
| **2020-06-22** | 2020-06-22 | 229.00 | 243.78 | 228.93 | 242.53 | 9079586552 | 27032930743 | 47.027367 | 13.54 |
| **2020-06-21** | 2020-06-21 | 229.22 | 232.36 | 228.49 | 228.99 | 5600408178 | 25520273837 | 34.193044 | -0.28 |
| **2020-06-20** | 2020-06-20 | 226.98 | 231.45 | 226.64 | 229.27 | 6252830566 | 25548837121 | 38.308090 | 2.13 |

348 rows × 25 columns

In [44]:
```
df_ethereum_headlines['date'] = pd.to_datetime(df_ethereum_headlines['date'])
```

In [45]:
```
df_headlines_merge = df_ethereum_headlines.join(hd, on="date", how='left', lsu
ffix='_left', rsuffix='_right')
df_headlines_merge
```

Out[45]:

| | headlines | news_center | time | website | neg | n |
|---|---|---|---|---|---|---|
| **0** | Here's What Could Be Next for Bitcoin, Ethereu... | The Daily Hodl | 2021-05-23 | https://dailyhodl.com/2021/05/23/heres-what-co... | 0.000 | 0.8 |
| **1** | How Leveraged Positions Could Have Accelerated... | Bitcoinist.com | 2021-05-23 | https://coinmarketcap.com/headlines/news/how-l... | 0.000 | 1.0 |
| **2** | Confirmed: Ethereum's Berlin hard fork solved ... | Finbold | 2021-05-23 | https://finbold.com/confirmed-ethereums-berlin... | 0.335 | 0.5 |
| **3** | These 3 factors will determine if Ethereum's r... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/these-3-factors-will-det... | 0.000 | 1.0 |
| **4** | Bitcoin & Ethereum: Here's the reality check o... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/bitcoin-ethereum-heres-t... | 0.000 | 1.0 |
| **...** | ... | ... | ... | ... | ... | |
| **280** | Value of staked ethereum reaches $10 billion m... | Finbold | 2021-04-22 | https://finbold.com/value-of-staked-ethereum-r... | 0.000 | 0.7 |
| **281** | Regulated Bitcoin and Ethereum funds have laun... | CryptoSlate | 2021-04-22 | https://cryptoslate.com/regulated-bitcoin-and-... | 0.000 | 0.8 |
| **282** | Key Reasons Why Ethereum Just Hit Fresh Record... | U.Today | 2021-04-22 | https://u.today/key-reasons-why-ethereum-just-... | 0.000 | 0.8 |
| **283** | JP Morgan is Now Hiring Ethereum and Blockchai... | Feed - Cryptopotato.Com | 2021-04-22 | https://cryptopotato.com/jp-morgan-is-now-hiri... | 0.000 | 0.7 |
| **284** | Charles Hoskinson hits back after Ethereum inf... | CryptoSlate | 2021-04-22 | https://cryptoslate.com/charles-hoskinson-hits... | 0.000 | 1.0 |

285 rows × 38 columns

In [46]:
```
df_headlines_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 38 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   headlines       285 non-null    object        
 1   news_center     285 non-null    object        
 2   time            285 non-null    object        
 3   website         285 non-null    object        
 4   neg             285 non-null    float64       
 5   neu             285 non-null    float64       
 6   pos             285 non-null    float64       
 7   compound        285 non-null    float64       
 8   date_left       285 non-null    datetime64[ns]
 9   Comp            285 non-null    float64       
 10  positive        285 non-null    float64       
 11  negative        285 non-null    float64       
 12  neutural        285 non-null    float64       
 13  date_right      285 non-null    object        
 14  open            285 non-null    float64       
 15  high            285 non-null    float64       
 16  low             285 non-null    float64       
 17  close           285 non-null    float64       
 18  volume          285 non-null    int64         
 19  market_cap      285 non-null    int64         
 20  RSI             285 non-null    float64       
 21  delta           285 non-null    float64       
 22  sell_points     285 non-null    int64         
 23  buy_points      285 non-null    int64         
 24  pct_change      285 non-null    float64       
 25  RSI_Delta       285 non-null    float64       
 26  RSI_pct_change  285 non-null    float64       
 27  bought_sold     285 non-null    int64         
 28  open_class      285 non-null    int64         
 29  high_class      285 non-null    int64         
 30  low_class       285 non-null    int64         
 31  close_class     285 non-null    int64         
 32  volume_class    285 non-null    int64         
 33  open_prev       285 non-null    float64       
 34  high_prev       285 non-null    float64       
 35  low_prev        285 non-null    float64       
 36  volume_prev     285 non-null    int64         
 37  label           285 non-null    object        
dtypes: datetime64[ns](1), float64(20), int64(11), object(6)
memory usage: 84.7+ KB
```

In [47]:
```
# Create two columns
# Subjectivity
df_headlines_merge['Subjectivity'] = df_headlines_merge['headlines'].apply(get
Subjectivity)
# Polarity
df_headlines_merge['Polarity'] =df_headlines_merge['headlines'].apply(getPolar
ity)
# Show new df
df_headlines_merge.head()
```

Out[47]:

| | headlines | news_center | time | website | neg | neu | |
|---|---|---|---|---|---|---|---|
| **0** | Here's What Could Be Next for Bitcoin, Ethereu... | The Daily Hodl | 2021-05-23 | https://dailyhodl.com/2021/05/23/heres-what-co... | 0.000 | 0.893 | 0. |
| **1** | How Leveraged Positions Could Have Accelerated... | Bitcoinist.com | 2021-05-23 | https://coinmarketcap.com/headlines/news/how-l... | 0.000 | 1.000 | 0. |
| **2** | Confirmed: Ethereum's Berlin hard fork solved ... | Finbold | 2021-05-23 | https://finbold.com/confirmed-ethereums-berlin... | 0.335 | 0.539 | 0. |
| **3** | These 3 factors will determine if Ethereum's r... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/these-3-factors-will-det... | 0.000 | 1.000 | 0. |
| **4** | Bitcoin & Ethereum: Here's the reality check o... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/bitcoin-ethereum-heres-t... | 0.000 | 1.000 | 0. |

5 rows × 40 columns

In [48]:
```
#Check the value count for news_center
df_headlines_merge.news_center.value_counts()
```

Out[48]:
```
Decrypt                     37
NewsBTC                     34
The Daily Hodl              23
Cointelegraph.com News      23
CryptoSlate                 21
U.Today                     21
AMBCrypto                   15
Seeking Alpha               12
Bitcoinist.com              12
Blockchain News             11
Coingape                    10
Crypto Briefing             10
Feed - Cryptopotato.Com      9
The Block                    7
CryptoGlobe                  6
Finbold                      6
BeInCrypto                   4
Crypto Daily™                4
Forbes                       4
Bitcoinist                   3
BTCMANAGER                   2
ZyCrypto                     2
CryptoBriefing               2
Ethereum World News          1
Trustnodes                   1
Forkast.News                 1
Quartz                       1
CryptoNinjas                 1
Blockworks                   1
Avalanche Reddit             1
Name: news_center, dtype: int64
```

In [49]:
```
names = pd.unique(df_headlines_merge.news_center.ravel())
names_data = {'news_center':names, 'id':np.arange(len(names))}
names_df = pd.DataFrame(data=names_data)
```

In [50]:
```
names_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   news_center  30 non-null     object
 1   id           30 non-null     int32 
dtypes: int32(1), object(1)
memory usage: 488.0+ bytes
```

In [51]:
```
df_headlines_merge = df_headlines_merge.join(names_df.set_index('news_center'
), on="news_center", how='left', lsuffix='_left', rsuffix='_right')
df_headlines_merge
```

Out[51]:

| | headlines | news_center | time | website | neg | n |
|---|---|---|---|---|---|---|
| **0** | Here's What Could Be Next for Bitcoin, Ethereu... | The Daily Hodl | 2021-05-23 | https://dailyhodl.com/2021/05/23/heres-what-co... | 0.000 | 0.8 |
| **1** | How Leveraged Positions Could Have Accelerated... | Bitcoinist.com | 2021-05-23 | https://coinmarketcap.com/headlines/news/how-l... | 0.000 | 1.0 |
| **2** | Confirmed: Ethereum's Berlin hard fork solved ... | Finbold | 2021-05-23 | https://finbold.com/confirmed-ethereums-berlin... | 0.335 | 0.5 |
| **3** | These 3 factors will determine if Ethereum's r... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/these-3-factors-will-det... | 0.000 | 1.0 |
| **4** | Bitcoin & Ethereum: Here's the reality check o... | AMBCrypto | 2021-05-23 | https://ambcrypto.com/bitcoin-ethereum-heres-t... | 0.000 | 1.0 |
| **...** | ... | ... | ... | ... | ... | |
| **280** | Value of staked ethereum reaches $10 billion m... | Finbold | 2021-04-22 | https://finbold.com/value-of-staked-ethereum-r... | 0.000 | 0.7 |
| **281** | Regulated Bitcoin and Ethereum funds have laun... | CryptoSlate | 2021-04-22 | https://cryptoslate.com/regulated-bitcoin-and-... | 0.000 | 0.8 |
| **282** | Key Reasons Why Ethereum Just Hit Fresh Record... | U.Today | 2021-04-22 | https://u.today/key-reasons-why-ethereum-just-... | 0.000 | 0.8 |
| **283** | JP Morgan is Now Hiring Ethereum and Blockchai... | Feed - Cryptopotato.Com | 2021-04-22 | https://cryptopotato.com/jp-morgan-is-now-hiri... | 0.000 | 0.7 |
| **284** | Charles Hoskinson hits back after Ethereum inf... | CryptoSlate | 2021-04-22 | https://cryptoslate.com/charles-hoskinson-hits... | 0.000 | 1.0 |

285 rows × 41 columns

In [52]:
```
df_headlines_merge.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 41 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   headlines       285 non-null    object
 1   news_center     285 non-null    object
 2   time            285 non-null    object
 3   website         285 non-null    object
 4   neg             285 non-null    float64
 5   neu             285 non-null    float64
 6   pos             285 non-null    float64
 7   compound        285 non-null    float64
 8   date_left       285 non-null    datetime64[ns]
 9   Comp            285 non-null    float64
 10  positive        285 non-null    float64
 11  negative        285 non-null    float64
 12  neutural        285 non-null    float64
 13  date_right      285 non-null    object
 14  open            285 non-null    float64
 15  high            285 non-null    float64
 16  low             285 non-null    float64
 17  close           285 non-null    float64
 18  volume          285 non-null    int64
 19  market_cap      285 non-null    int64
 20  RSI             285 non-null    float64
 21  delta           285 non-null    float64
 22  sell_points     285 non-null    int64
 23  buy_points      285 non-null    int64
 24  pct_change      285 non-null    float64
 25  RSI_Delta       285 non-null    float64
 26  RSI_pct_change  285 non-null    float64
 27  bought_sold     285 non-null    int64
 28  open_class      285 non-null    int64
 29  high_class      285 non-null    int64
 30  low_class       285 non-null    int64
 31  close_class     285 non-null    int64
 32  volume_class    285 non-null    int64
 33  open_prev       285 non-null    float64
 34  high_prev       285 non-null    float64
 35  low_prev        285 non-null    float64
 36  volume_prev     285 non-null    int64
 37  label           285 non-null    object
 38  Subjectivity    285 non-null    float64
 39  Polarity        285 non-null    float64
 40  id              285 non-null    int32
dtypes: datetime64[ns](1), float64(22), int32(1), int64(11), object(6)
memory usage: 90.3+ KB
```

In [53]:
```
# Create a list of columns to keep
columns_all = ['id', 'Subjectivity', 'Polarity', 'Comp', 'positive', 'negativ
e', 'neutural',
           'sell_points', 'buy_points', 'bought_sold', 'open_class', 'high_cla
ss', 'low_class', 'volume_class',
           'open_prev', 'high_prev', 'low_prev', 'volume_prev', 'label']

columns_RSI_hd = ['sell_points', 'buy_points', 'bought_sold', 'open_class', 'h
igh_class', 'low_class', 'volume_class',
               'open_prev', 'high_prev', 'low_prev', 'volume_prev', 'label']

columns_sent_hd = ['id', 'Subjectivity', 'Polarity', 'Comp', 'positive', 'nega
tive', 'neutural',
               'open_prev', 'high_prev', 'low_prev', 'volume_prev', 'label']

columns_sent_RSI = ['id', 'Subjectivity', 'Polarity', 'Comp', 'positive', 'neg
ative', 'neutural',
               'sell_points', 'buy_points', 'bought_sold', 'open_class', 'high
_class', 'low_class', 'volume_class', 'label']

columns_hd = ['open_prev', 'high_prev', 'low_prev', 'volume_prev', 'label']


columns_RSI = ['sell_points', 'buy_points', 'bought_sold', 'open_class', 'high
_class', 'low_class', 'volume_class', 'label']

columns_sentiment = ['id', 'Subjectivity', 'Polarity', 'Comp', 'positive', 'ne
gative', 'neutural', 'label']
```

In [54]:
```
def print_metrics(labels, preds):
    print("Precision Score: {}".format(precision_score(labels, preds)))
    print("Recall Score: {}".format(recall_score(labels, preds)))
    print("Accuracy Score: {}".format(accuracy_score(labels, preds)))
    print("F1 Scorce: {}".format(f1_score(labels, preds)))
```

# Model 2: Features engineered via the Sentiment Analysis(SA)

In [55]:
```
# Model 2: Features engineered via the Sentiment Analysis(SA)
df_model_2 = df_headlines_merge[columns_sentiment]
# Show Data
df_model_2.head()
```

Out[55]:

| | id | Subjectivity | Polarity | Comp | positive | negative | neutural | label |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0.250000 | 0.250000 | 0.2023 | 0.107 | 0.000 | 0.893 | 0 |
| **1** | 1 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 |
| **2** | 2 | 0.770833 | 0.054167 | -0.3612 | 0.126 | 0.335 | 0.539 | 0 |
| **3** | 3 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 |
| **4** | 3 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 |

In [56]:
```
df_model_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            285 non-null    int32  
 1   Subjectivity  285 non-null    float64
 2   Polarity      285 non-null    float64
 3   Comp          285 non-null    float64
 4   positive      285 non-null    float64
 5   negative      285 non-null    float64
 6   neutural      285 non-null    float64
 7   label         285 non-null    object 
dtypes: float64(6), int32(1), object(1)
memory usage: 16.8+ KB
```

In [57]:
```
df_model_2['label'] = pd.to_numeric(df_model_2['label'])
df_model_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            285 non-null    int32  
 1   Subjectivity  285 non-null    float64
 2   Polarity      285 non-null    float64
 3   Comp          285 non-null    float64
 4   positive      285 non-null    float64
 5   negative      285 non-null    float64
 6   neutural      285 non-null    float64
 7   label         285 non-null    int64  
dtypes: float64(6), int32(1), int64(1)
memory usage: 16.8 KB

C:\Users\egust\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.
py:1: SettingWithCopyWarning: 


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [58]:
```
#Checking for imbalances
df_model_2.label.value_counts()
```

Out[58]:
```
1    170
0    115
Name: label, dtype: int64
```

In [59]:
```
#Create the feature data set
X2 = df_model_2
X2 = np.array(X2.drop(['label'], 1))
# Target dataset
y2 = np.array(df_model_2['label'])
```

In [60]:
```
#Split the data into 80% training and 20% test
x_train, x_test, y_train, y_test = train_test_split(X2, y2, test_size = 0.2, r
andom_state =0)
```

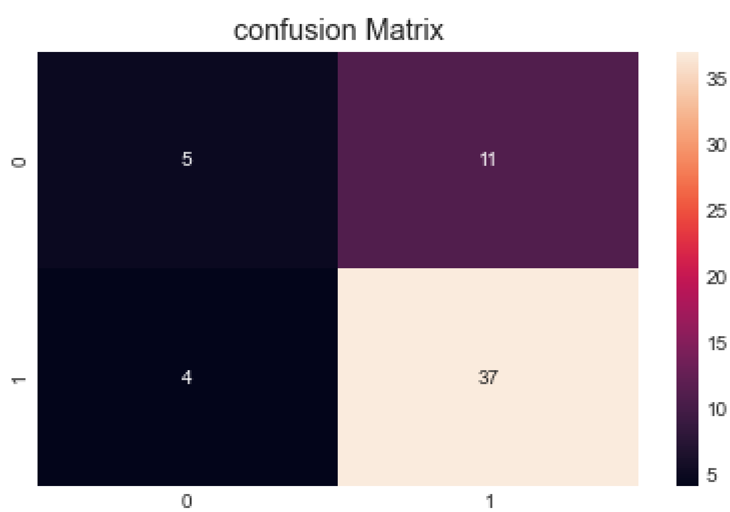
In [61]:
```
#Create and train the model
#Instatiate
LDA = LinearDiscriminantAnalysis()
model = LDA.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [62]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 0.7708333333333334
Recall Score: 0.9024390243902439
Accuracy Score: 0.7368421052631579
F1 Scorce: 0.8314606741573034
```

In [63]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_LDA_SA')
plt.show()
```

```
The accuracy is :0.74

AUC is :0.61

Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.31      0.40        16
           1       0.77      0.90      0.83        41

    accuracy                           0.74        57
   macro avg       0.66      0.61      0.62        57
weighted avg       0.71      0.74      0.71        57
```



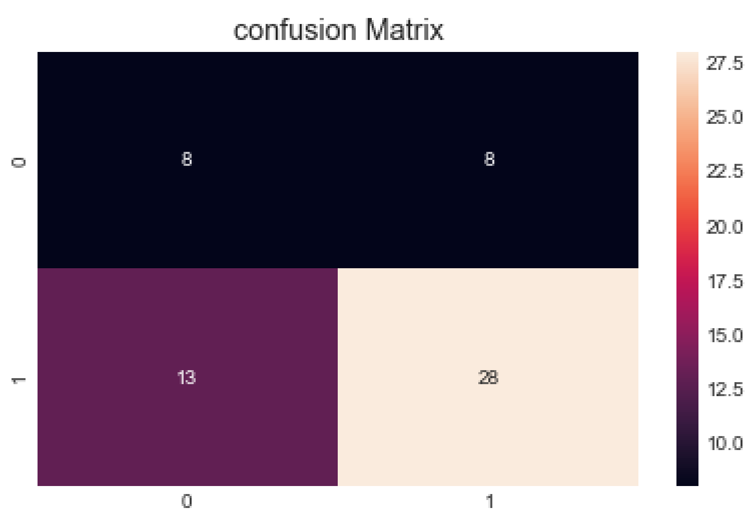

In [64]:
```
#Create and train the model
#Instatiate
rf_classifier = RandomForestClassifier(n_estimators=100)
model = rf_classifier.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [65]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 0.7777777777777778
Recall Score: 0.6829268292682927
Accuracy Score: 0.631578947368421
F1 Scorce: 0.7272727272727273
```

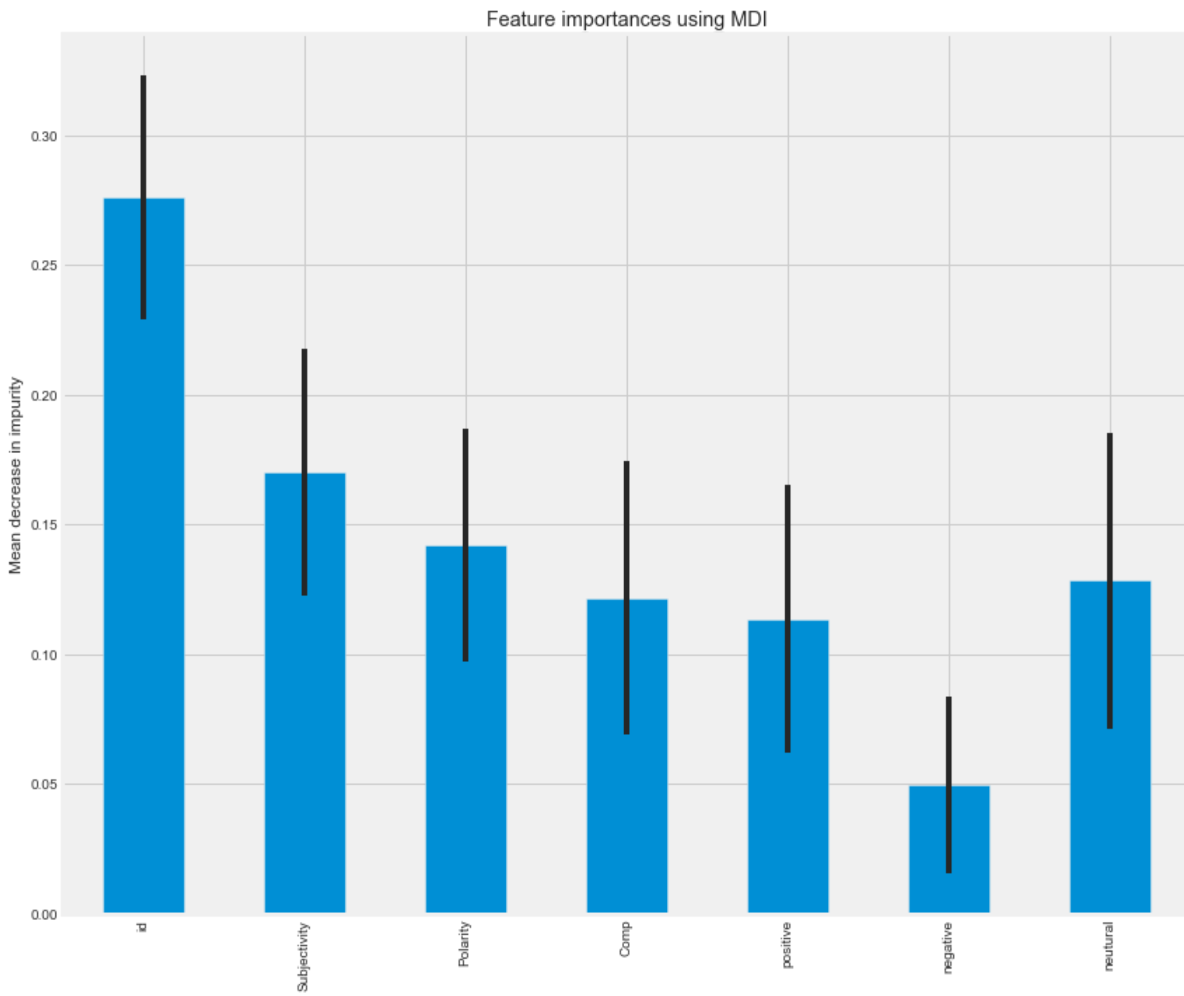
In [66]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_RF_SA')
plt.show()
```

```
The accuracy is :0.63

AUC is :0.59

Classification Report:
              precision    recall  f1-score   support

           0       0.38      0.50      0.43        16
           1       0.78      0.68      0.73        41

    accuracy                           0.63        57
   macro avg       0.58      0.59      0.58        57
weighted avg       0.67      0.63      0.64        57
```

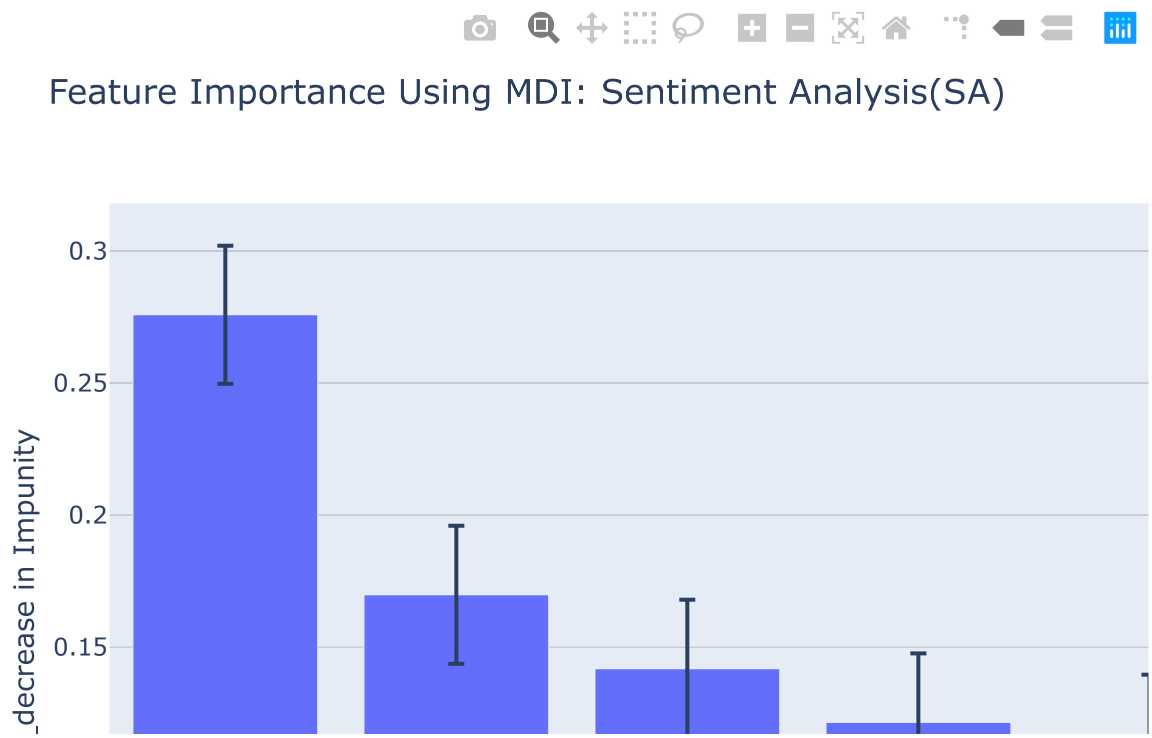



In [67]:
```
start_time = time.time()
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0
)
elapsed_time = time.time() - start_time
```

In [68]:
```
X2_columns = df_model_2.columns[0:7].to_list()
forest_importances = pd.Series(importances, index= X2_columns)

fig, ax = plt.subplots(figsize=(12,10))
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title('Feature importances using MDI')
ax.set_ylabel('Mean decrease in impurity')
fig.tight_layout()
plt.savefig('FI_SA')
```



In [69]:
```
d = {'Mean_decrease in Impunity':importances, 'names_importances':X2_columns}
forest_importances_2 = pd.DataFrame(data=d)

#calculate standard error
std_error2 = np.std(forest_importances_2['Mean_decrease in Impunity'], ddof=1)
/ np.sqrt(len(forest_importances_2['Mean_decrease in Impunity']))
forest_importances_2['std'] = std_error2

fig2 = px.bar(forest_importances_2, x= 'names_importances', y= 'Mean_decrease
 in Impunity', title='Feature Importance Using MDI: Sentiment Analysis(SA)', e
rror_y='std')
fig2.show()
fig2.write_image('images/fig2.png')
```




# Model 3: Features engineered via the Relative Strength Index, (RSI)

In [70]:
```
# Model 3: Features engineered via the Relative Strength Index, (RSI)
df_model_3 = df_headlines_merge[columns_RSI]
df_model_3.head()
```

Out[70]:

| | sell_points | buy_points | bought_sold | open_class | high_class | low_class | volume_class | label |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **1** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **2** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **3** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| **4** | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

In [71]:
```
#Change label dtype to numeric int64
df_model_3['label'] = pd.to_numeric(df_model_3['label'])
df_model_3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sell_points   285 non-null    int64
 1   buy_points    285 non-null    int64
 2   bought_sold   285 non-null    int64
 3   open_class    285 non-null    int64
 4   high_class    285 non-null    int64
 5   low_class     285 non-null    int64
 6   volume_class  285 non-null    int64
 7   label         285 non-null    int64
dtypes: int64(8)
memory usage: 17.9 KB

C:\Users\egust\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.
py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [72]:
```
#Create the feature data set
X3 = df_model_3
#
X3 = np.array(X3.drop(['label'], 1))
# Target dataset
y3 = np.array(df_model_3['label'])
```

In [73]:
```
#Split the data into 80% training and 20% test
x_train, x_test, y_train, y_test = train_test_split(X3, y3, test_size = 0.2, r
andom_state =0)
```

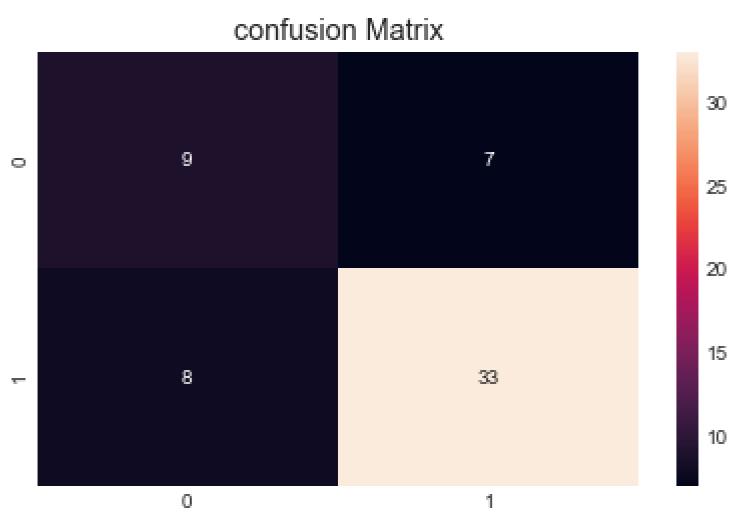
In [74]:
```
#Create and train the model
#Instatiate
LDA = LinearDiscriminantAnalysis()
model = LDA.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [75]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 0.825
Recall Score: 0.8048780487804879
Accuracy Score: 0.7368421052631579
F1 Scorce: 0.8148148148148149
```

In [76]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_LDA_RSI')
plt.show()
```

```
The accuracy is :0.74

AUC is :0.68

Classification Report:
              precision    recall  f1-score   support

           0       0.53      0.56      0.55        16
           1       0.82      0.80      0.81        41

    accuracy                           0.74        57
   macro avg       0.68      0.68      0.68        57
weighted avg       0.74      0.74      0.74        57
```



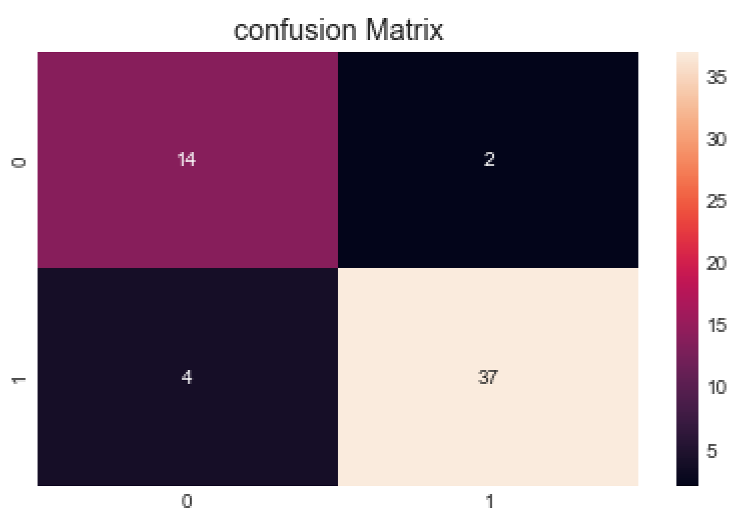

In [77]:
```
#Create and train the model
#Instatiate
rf_classifier = RandomForestClassifier(n_estimators=100)
model = rf_classifier.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [78]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 0.9487179487179487
Recall Score: 0.9024390243902439
Accuracy Score: 0.8947368421052632
F1 Scorce: 0.9249999999999999
```

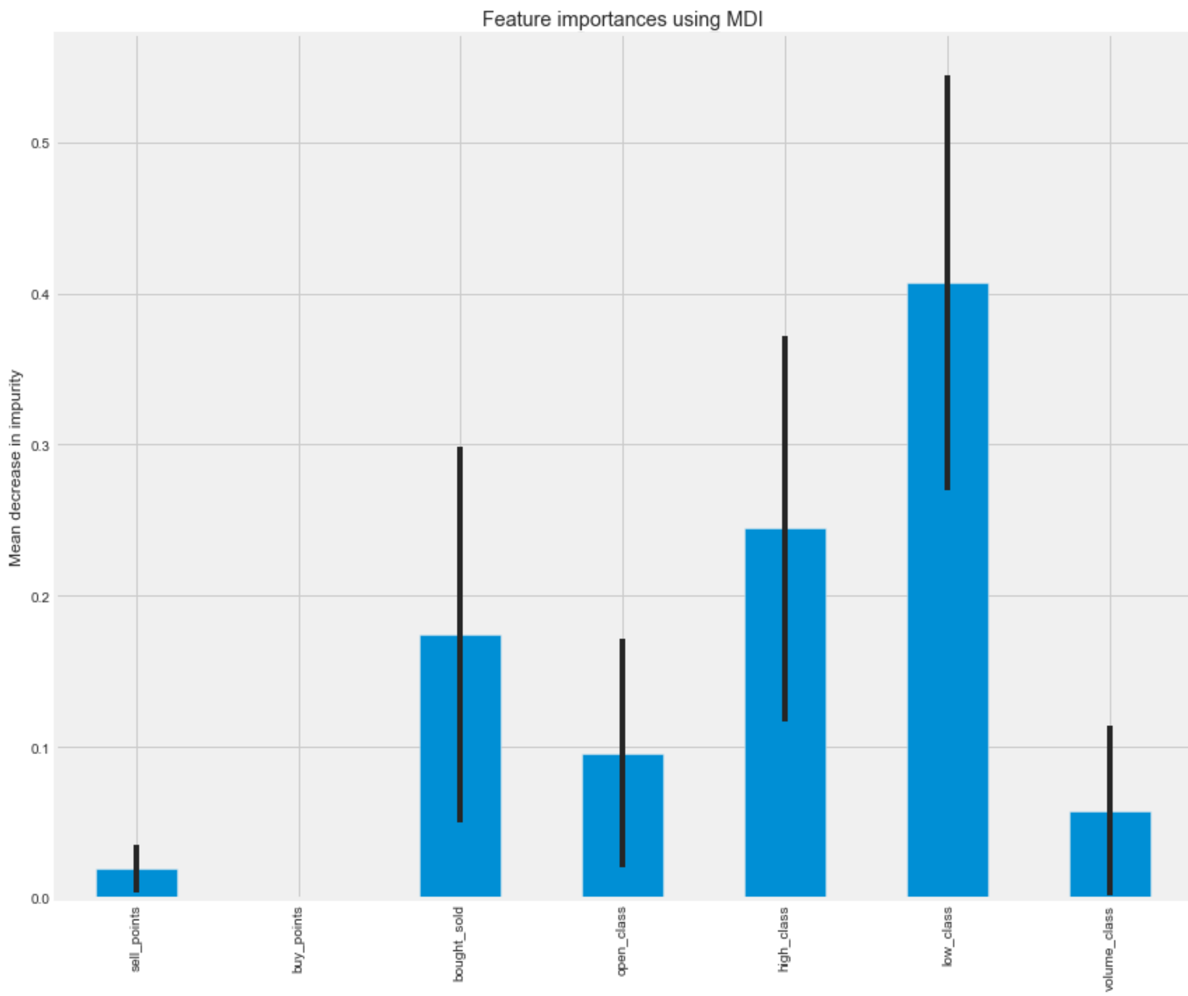
In [79]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_RF_RSI')
plt.show()
```

```
The accuracy is :0.89

AUC is :0.89

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.88      0.82        16
           1       0.95      0.90      0.92        41

    accuracy                           0.89        57
   macro avg       0.86      0.89      0.87        57
weighted avg       0.90      0.89      0.90        57
```

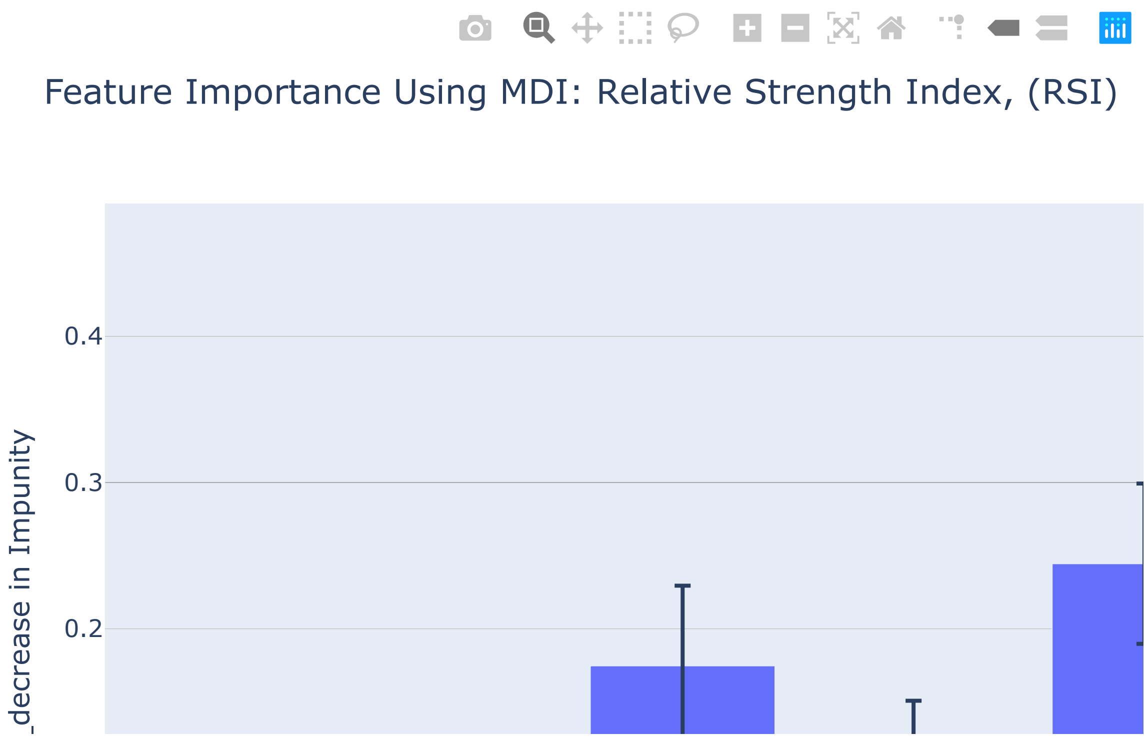



In [80]:
```
start_time = time.time()
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0
)
elapsed_time = time.time() - start_time
```

In [81]:
```
X3_columns = df_model_3.columns[0:7].to_list()
forest_importances = pd.Series(importances , index= X3_columns)

fig, ax = plt.subplots(figsize=(12,10))
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title('Feature importances using MDI')
ax.set_ylabel('Mean decrease in impurity')
fig.tight_layout()
plt.savefig('FI_RSI')
```



In [82]:
```
d3 = {'Mean_decrease in Impunity':importances, 'names_importances':X3_columns}
forest_importances_3 = pd.DataFrame(data=d3)

#calculate standard error
std_error3 = np.std(forest_importances_3['Mean_decrease in Impunity'], ddof=1)
/ np.sqrt(len(forest_importances_3['Mean_decrease in Impunity']))
forest_importances_3['std'] = std_error3

fig3 = px.bar(forest_importances_3, x= 'names_importances', y= 'Mean_decrease
 in Impunity', title='Feature Importance Using MDI: Relative Strength Index,
 (RSI)',  error_y='std')
fig3.show()
fig3.write_image('images/fig3.png')
```




# Model 4: Features engineered via the Historical Data(HD)

In [83]:
```
# Model 4: Features engineered via the Historical Data(HD)
df_model_4 = df_headlines_merge[columns_hd]
df_model_4.head()
```

Out[83]:

| | open_prev | high_prev | low_prev | volume_prev | label |
|---|---|---|---|---|---|
| **0** | 2436.01 | 2483.98 | 2168.12 | 42089937660 | 0 |
| **1** | 2436.01 | 2483.98 | 2168.12 | 42089937660 | 0 |
| **2** | 2436.01 | 2483.98 | 2168.12 | 42089937660 | 0 |
| **3** | 2436.01 | 2483.98 | 2168.12 | 42089937660 | 0 |
| **4** | 2436.01 | 2483.98 | 2168.12 | 42089937660 | 0 |

In [84]:
```
df_model_4['label'] = pd.to_numeric(df_model_4['label'])
df_model_4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   open_prev    285 non-null    float64
 1   high_prev    285 non-null    float64
 2   low_prev     285 non-null    float64
 3   volume_prev  285 non-null    int64  
 4   label        285 non-null    int64  
dtypes: float64(3), int64(2)
memory usage: 11.3 KB

C:\Users\egust\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.
py:1: SettingWithCopyWarning: 


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [85]:
```
#Create the feature data set
X4 = df_model_4
X4 = np.array(X4.drop(['label'], 1))
#Target dataset
y4 = np.array(df_model_4['label'])
```

In [86]:
```
#Split the data into 80% training and 20% test
x_train, x_test, y_train, y_test = train_test_split(X4, y4, test_size = 0.2, r
andom_state =0)
```

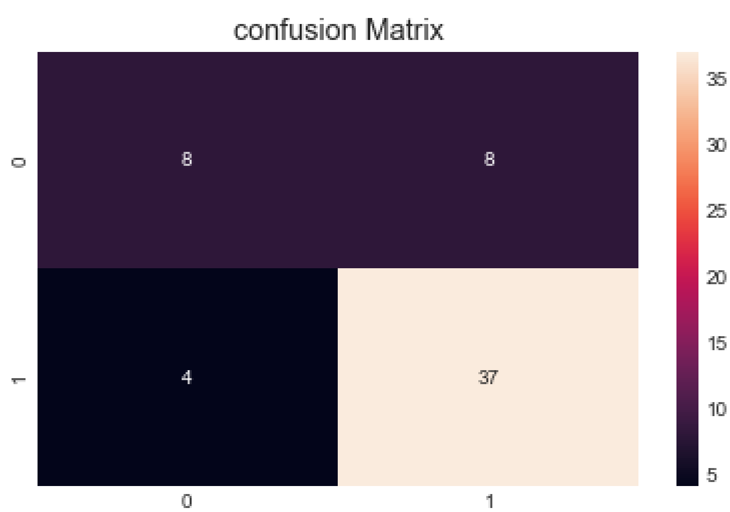
In [87]:
```
#Create and train the model
#Instatiate
LDA = LinearDiscriminantAnalysis()
model = LDA.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [88]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 0.8222222222222222
Recall Score: 0.9024390243902439
Accuracy Score: 0.7894736842105263
F1 Scorce: 0.8604651162790697
```

In [89]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_LDA_HD')
plt.show()
```

```
The accuracy is :0.79

AUC is :0.7

Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.50      0.57        16
           1       0.82      0.90      0.86        41

    accuracy                           0.79        57
   macro avg       0.74      0.70      0.72        57
weighted avg       0.78      0.79      0.78        57
```



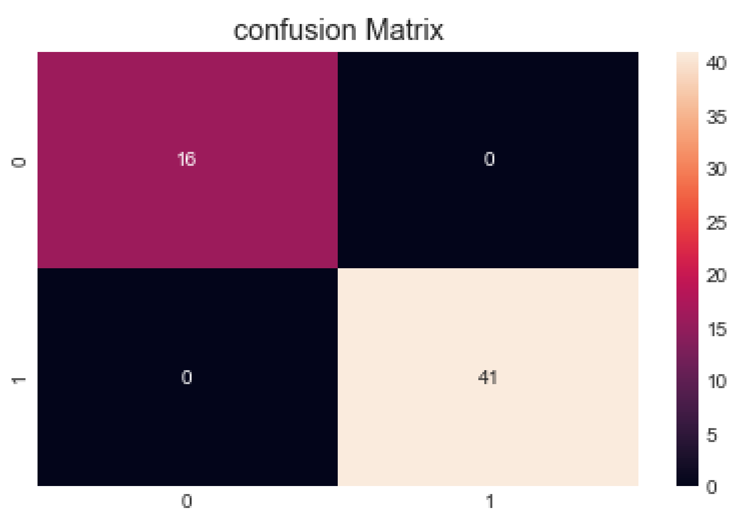

In [90]:
```
#Create and train the model
#Instatiate
rf_classifier = RandomForestClassifier(n_estimators=100)
model = rf_classifier.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [91]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 1.0
Recall Score: 1.0
Accuracy Score: 1.0
F1 Scorce: 1.0
```

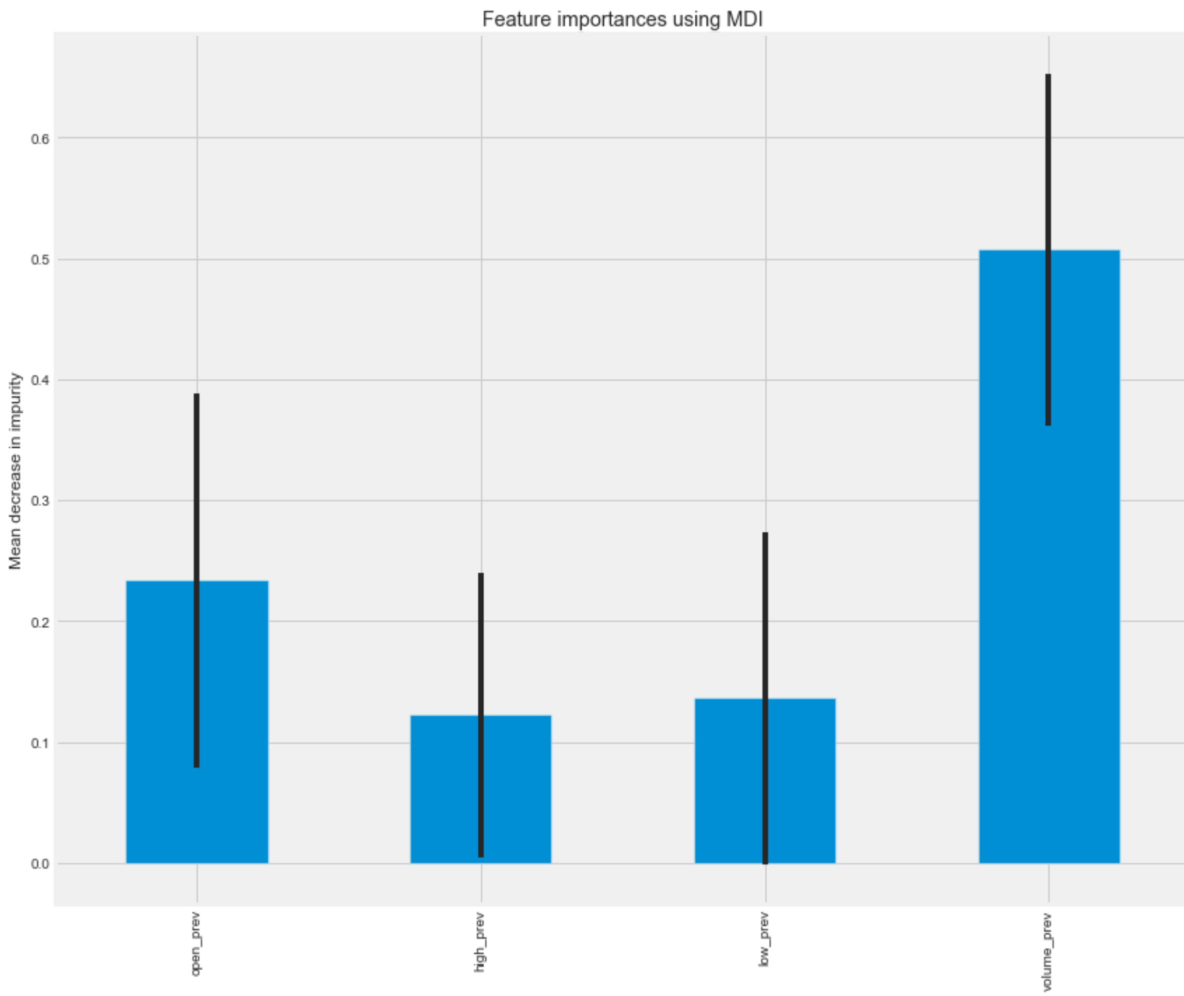
In [92]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_RF_HD')
plt.show()
```

```
The accuracy is :1.0

AUC is :1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        41

    accuracy                           1.00        57
   macro avg       1.00      1.00      1.00        57
weighted avg       1.00      1.00      1.00        57
```

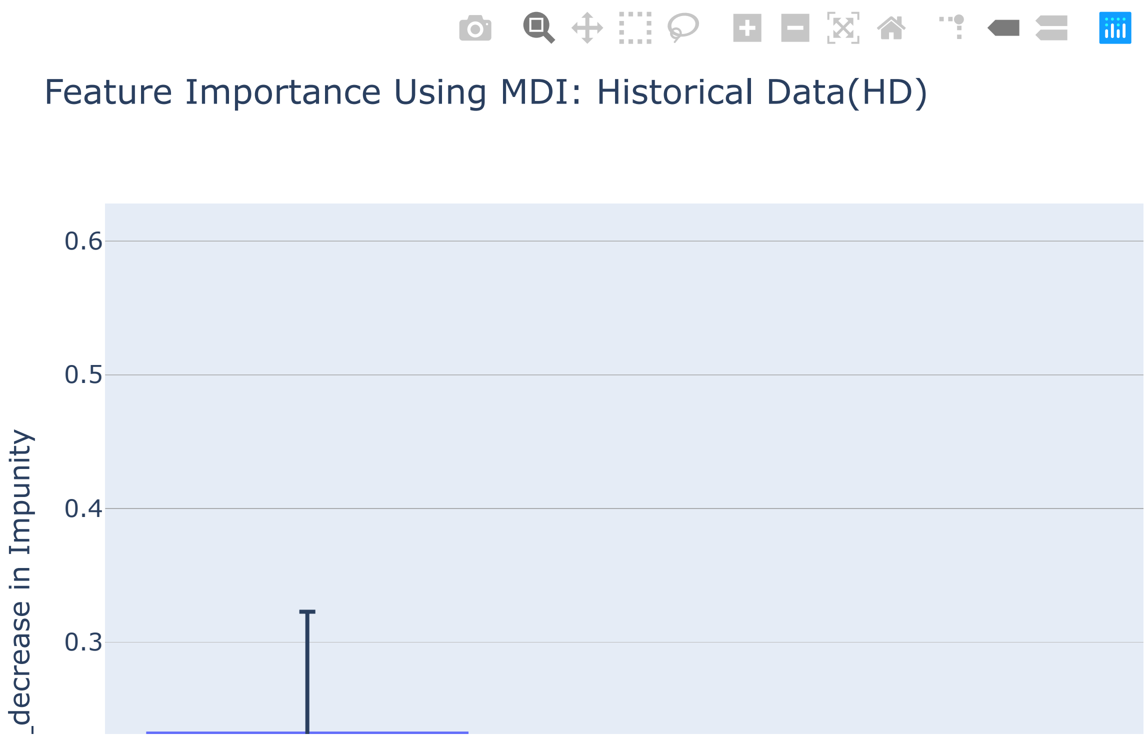



In [93]:
```
start_time = time.time()
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0
)
elapsed_time = time.time() - start_time
```

In [94]:
```
X4_columns = df_model_4.columns[0:4].to_list()
forest_importances = pd.Series(importances , index= X4_columns)

fig, ax = plt.subplots(figsize=(12,10))
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title('Feature importances using MDI')
ax.set_ylabel('Mean decrease in impurity')
fig.tight_layout()
plt.savefig('FI_HD')
```



In [95]:
```
d4 = {'Mean_decrease in Impunity':importances, 'names_importances':X4_columns}
forest_importances_4 = pd.DataFrame(data=d4)

#calculate standard error
std_error4 = np.std(forest_importances_4['Mean_decrease in Impunity'], ddof=1)
/ np.sqrt(len(forest_importances_4['Mean_decrease in Impunity']))
forest_importances_4['std'] = std_error4

fig4 = px.bar(forest_importances_4, x= 'names_importances', y= 'Mean_decrease
 in Impunity', title='Feature Importance Using MDI: Historical Data(HD)', erro
r_y='std' )
fig4.show()
fig4.write_image('images/fig4.png')
```




## Model 5: Features engineered via the SA and RSI

In [96]:
```
# Model 5: Features engineered via the SA and RSI
df_model_5 = df_headlines_merge[columns_sent_RSI]
df_model_5.head()
```

Out[96]:

| | id | Subjectivity | Polarity | Comp | positive | negative | neutural | sell_points | buy_points | boug |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0.250000 | 0.250000 | 0.2023 | 0.107 | 0.000 | 0.893 | 0 | 0 | |
| **1** | 1 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 | 0 | |
| **2** | 2 | 0.770833 | 0.054167 | -0.3612 | 0.126 | 0.335 | 0.539 | 0 | 0 | |
| **3** | 3 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 | 0 | |
| **4** | 3 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 | 0 | |

In [97]:
```
df_model_5['label'] = pd.to_numeric(df_model_5['label'])
df_model_5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            285 non-null    int32  
 1   Subjectivity  285 non-null    float64
 2   Polarity      285 non-null    float64
 3   Comp          285 non-null    float64
 4   positive      285 non-null    float64
 5   negative      285 non-null    float64
 6   neutural      285 non-null    float64
 7   sell_points   285 non-null    int64  
 8   buy_points    285 non-null    int64  
 9   bought_sold   285 non-null    int64  
 10  open_class    285 non-null    int64  
 11  high_class    285 non-null    int64  
 12  low_class     285 non-null    int64  
 13  volume_class  285 non-null    int64  
 14  label         285 non-null    int64  
dtypes: float64(6), int32(1), int64(8)
memory usage: 32.4 KB

C:\Users\egust\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.
py:1: SettingWithCopyWarning: 


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [98]:
```
#Create the feature data set
X5 = df_model_5
X5 = np.array(X5.drop(['label'], 1))
# Target dataset
y5 = np.array(df_model_5['label'])
```

In [99]:
```
#Split the data into 80% training and 20% test
x_train, x_test, y_train, y_test = train_test_split(X5, y5, test_size = 0.2, r
andom_state =0)
```

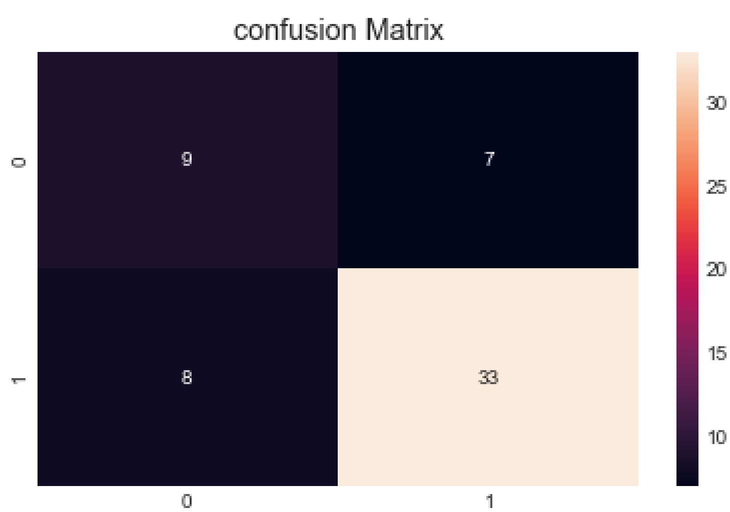
In [100]:
```
#Create and train the model
#Instatiate
LDA = LinearDiscriminantAnalysis()
model = LDA.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [101]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 0.825
Recall Score: 0.8048780487804879
Accuracy Score: 0.7368421052631579
F1 Scorce: 0.8148148148148149
```

In [102]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_LDA_RSI_RA')
plt.show()
```

```
The accuracy is :0.74

AUC is :0.68

Classification Report:
              precision    recall  f1-score   support

           0       0.53      0.56      0.55        16
           1       0.82      0.80      0.81        41

    accuracy                           0.74        57
   macro avg       0.68      0.68      0.68        57
weighted avg       0.74      0.74      0.74        57
```




In [103]:
```
#Create and train the model
#Instatiate
rf_classifier = RandomForestClassifier(n_estimators=100)
model = rf_classifier.fit(x_train, y_train)
pred = model.predict(x_test)
```

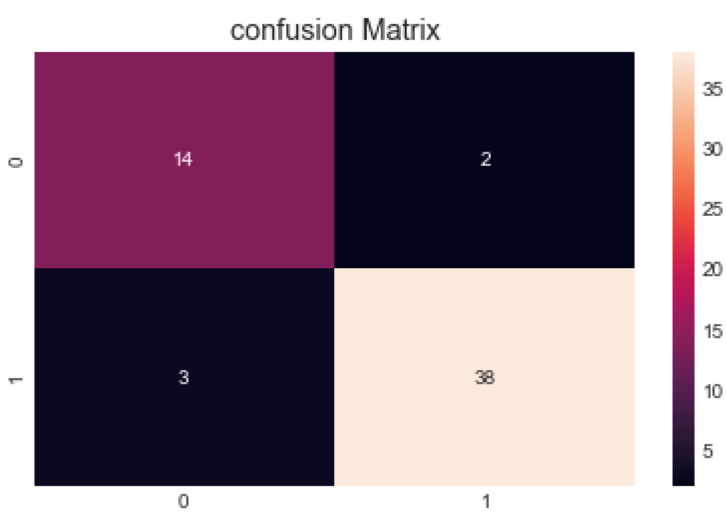
In [104]:

```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 0.95
Recall Score: 0.926829268292683
Accuracy Score: 0.9122807017543859
F1 Scorce: 0.9382716049382716
```

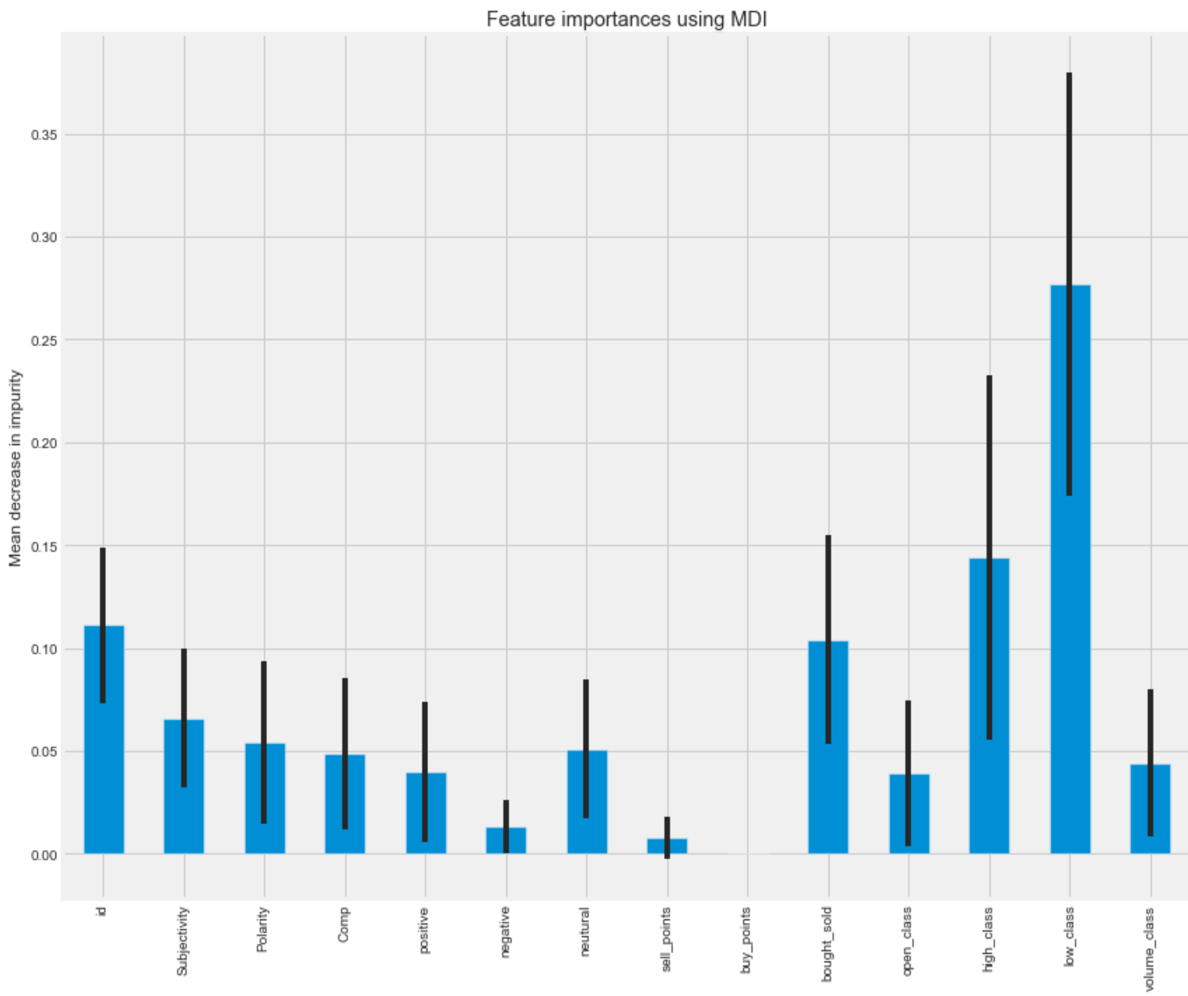
In [105]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_RF_RSI_SA')
plt.show()
```

```
The accuracy is :0.91

AUC is :0.9

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.88      0.85        16
           1       0.95      0.93      0.94        41

    accuracy                           0.91        57
   macro avg       0.89      0.90      0.89        57
weighted avg       0.91      0.91      0.91        57
```




In [106]:
```
start_time = time.time()
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0
)
elapsed_time = time.time() - start_time
```

In [107]:
```
X5_columns = df_model_5.columns[0:14].to_list()
forest_importances = pd.Series(importances , index= X5_columns)

fig, ax = plt.subplots(figsize=(12,10))
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title('Feature importances using MDI')
ax.set_ylabel('Mean decrease in impurity')
fig.tight_layout()
plt.savefig('FI_SA_RSI')
```



In [108]:
```
d5 = {'Mean_decrease in Impunity':importances, 'names_importances':X5_columns}
forest_importances_5 = pd.DataFrame(data=d5)

#calculate standard error
std_error5 = np.std(forest_importances_5['Mean_decrease in Impunity'], ddof=1)
/ np.sqrt(len(forest_importances_5['Mean_decrease in Impunity']))
forest_importances_5['std'] = std_error5

fig5 = px.bar(forest_importances_5, x= 'names_importances', y= 'Mean_decrease
 in Impunity', title='Feature Importance Using MDI:SA and RSI', error_y='std')
fig5.show()
fig5.write_image('images/fig5.png')
```


## Model 6: Features engineered via the SA and RSI and HD

In [109]:
```
# Model 6: Features engineered via the SA and RSI and HD
df_model_6 = df_headlines_merge[columns_all]
df_model_6.head()
```

Out[109]:

| | id | Subjectivity | Polarity | Comp | positive | negative | neutural | sell_points | buy_points | boug |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0.250000 | 0.250000 | 0.2023 | 0.107 | 0.000 | 0.893 | 0 | 0 | |
| **1** | 1 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 | 0 | |
| **2** | 2 | 0.770833 | 0.054167 | -0.3612 | 0.126 | 0.335 | 0.539 | 0 | 0 | |
| **3** | 3 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 | 0 | |
| **4** | 3 | 0.000000 | 0.000000 | 0.0000 | 0.000 | 0.000 | 1.000 | 0 | 0 | |

In [110]:
```
df_model_6['label'] = pd.to_numeric(df_model_6['label'])
df_model_6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 19 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   id            285 non-null    int32  
 1   Subjectivity  285 non-null    float64
 2   Polarity      285 non-null    float64
 3   Comp          285 non-null    float64
 4   positive      285 non-null    float64
 5   negative      285 non-null    float64
 6   neutural      285 non-null    float64
 7   sell_points   285 non-null    int64  
 8   buy_points    285 non-null    int64  
 9   bought_sold   285 non-null    int64  
 10  open_class    285 non-null    int64  
 11  high_class    285 non-null    int64  
 12  low_class     285 non-null    int64  
 13  volume_class  285 non-null    int64  
 14  open_prev     285 non-null    float64
 15  high_prev     285 non-null    float64
 16  low_prev      285 non-null    float64
 17  volume_prev   285 non-null    int64  
 18  label         285 non-null    int64  
dtypes: float64(9), int32(1), int64(9)
memory usage: 41.3 KB

C:\Users\egust\anaconda3\envs\learn-env\lib\site-packages\ipykernel_launcher.
py:1: SettingWithCopyWarning: 


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [111]:
```
#Create the feature data set
X6 = df_model_6
X6 = np.array(X6.drop(['label'], 1))
#Target dataset
y6 = np.array(df_model_6['label'])
```

In [112]:
```
#Split the data into 80% training and 20% test
x_train, x_test, y_train, y_test = train_test_split(X6, y6, test_size = 0.2, r
andom_state =0)
```

In [113]:
```
#Create and train the model
#Instatiate
LDA = LinearDiscriminantAnalysis()
model = LDA.fit(x_train, y_train)
pred = model.predict(x_test)
```

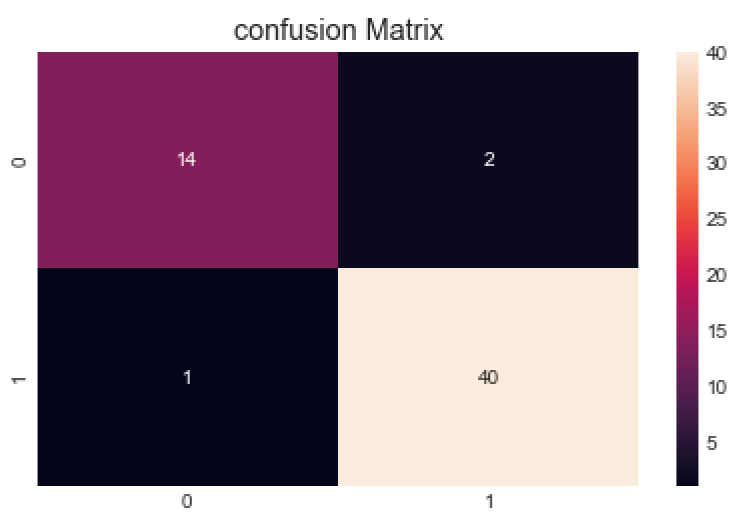
In [114]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 0.9523809523809523
Recall Score: 0.975609756097561
Accuracy Score: 0.9473684210526315
F1 Scorce: 0.963855421686747
```

In [115]:

```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_LDA_all')
plt.show()
```

```
The accuracy is :0.95

AUC is :0.93

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.88      0.90        16
           1       0.95      0.98      0.96        41

    accuracy                           0.95        57
   macro avg       0.94      0.93      0.93        57
weighted avg       0.95      0.95      0.95        57
```




In [116]:

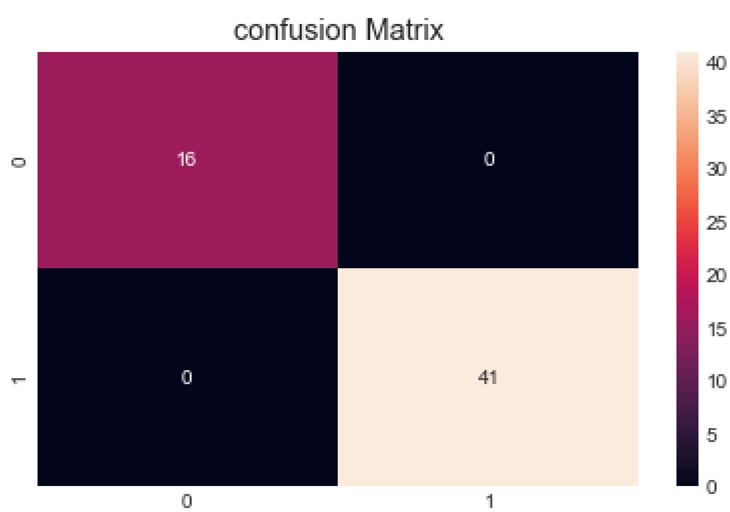
```
#Create and train the model
#Instatiate
rf_classifier = RandomForestClassifier(n_estimators=100)
model = rf_classifier.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [117]:
```
#Check Performance Metrics
performance_metrics= print_metrics(y_test, pred)
```

```
Precision Score: 1.0
Recall Score: 1.0
Accuracy Score: 1.0
F1 Scorce: 1.0
```

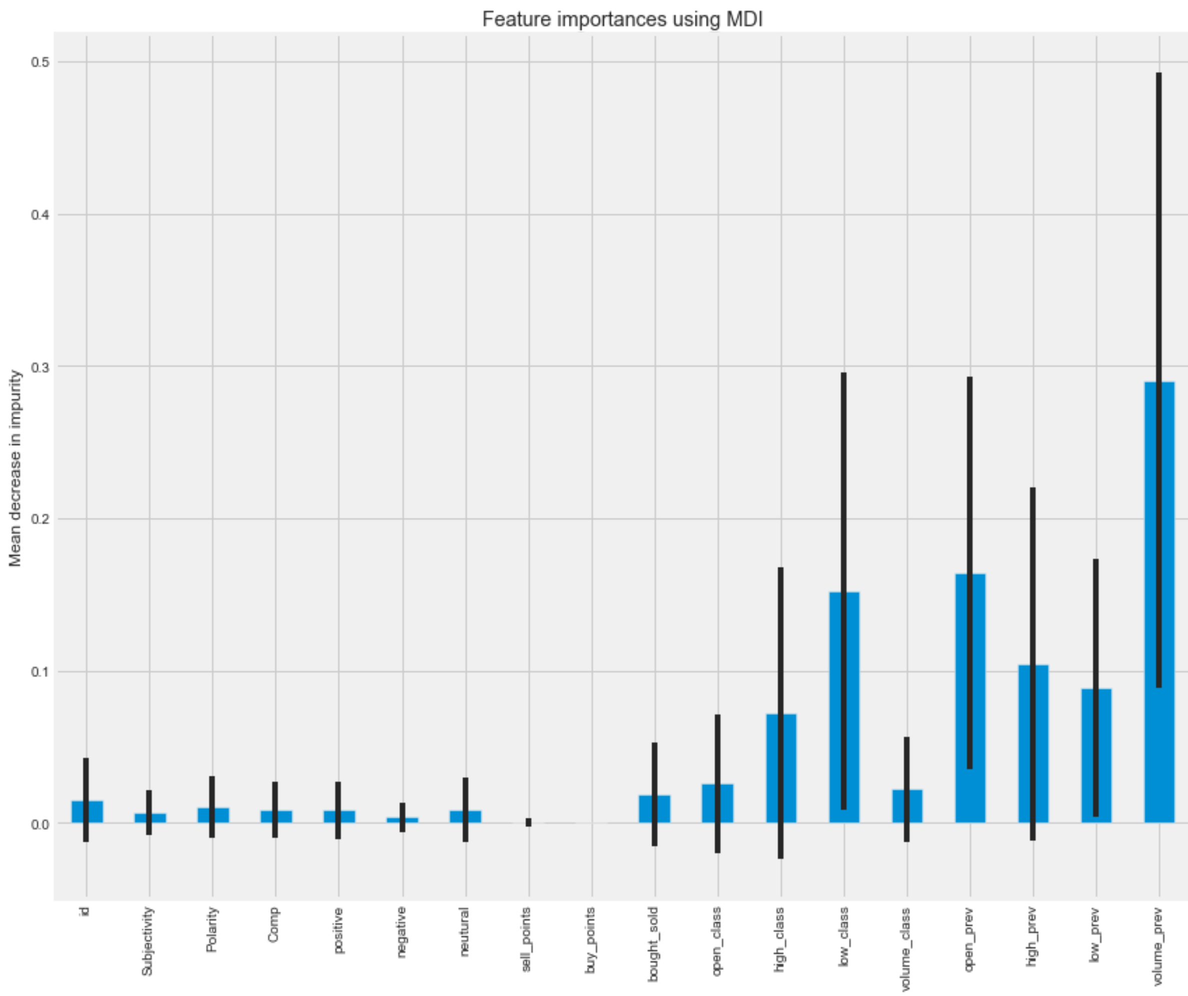
In [118]:
```
#Check the accuracy for prediction
acc= accuracy_score(y_test, pred)

#Check the AUC for predictions
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, pred)
roc_auc = auc(false_positive_rate, true_positive_rate)

#Print your accuracy_score, classification report, and confusion matrix
print('The accuracy is :{0}'.format(round(acc,2)))
print('\nAUC is :{0}'.format(round(roc_auc, 2)))
print('\nClassification Report:')
print(metrics.classification_report(y_test, pred))

sns.heatmap(confusion_matrix(y_test, pred), annot=True)
plt.title("confusion Matrix")
plt.savefig('confusion_matrix_RF_all')
plt.show()
```

```
The accuracy is :1.0

AUC is :1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       1.00      1.00      1.00        41

    accuracy                           1.00        57
   macro avg       1.00      1.00      1.00        57
weighted avg       1.00      1.00      1.00        57
```




In [119]:
```
start_time = time.time()
importances = model.feature_importances_
std = np.std([tree.feature_importances_ for tree in model.estimators_], axis=0
)
elapsed_time = time.time() - start_time
```

In [120]:
```
X6_columns = df_model_6.columns[0:18].to_list()
forest_importances = pd.Series(importances , index= X6_columns)

fig, ax = plt.subplots(figsize=(12,10))
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title('Feature importances using MDI')
ax.set_ylabel('Mean decrease in impurity')
fig.tight_layout()
plt.savefig('FI_all')
```



In [121]:
```
d6 = {'Mean_decrease in Impunity':importances, 'names_importances':X6_columns}
forest_importances_6 = pd.DataFrame(data=d6)

#calculate standard error
std_error6 = np.std(forest_importances_6['Mean_decrease in Impunity'], ddof=1)
/ np.sqrt(len(forest_importances_6['Mean_decrease in Impunity']))
forest_importances_6['std'] = std_error6

fig6 = px.bar(forest_importances_6, x= 'names_importances', y= 'Mean_decrease
 in Impunity', title='Feature Importance Using MDI:SA and RSI and HD', error_y
='std')
fig6.show()
fig6.write_image('images/fig6.png')
```

# Conclusion

In conclusion, the combination of engineered features from the historical data set and the features from a sentiment analysis provide a great foundation to models to predict whether the Ethereum market will close higher or lower than the previous day. Looking into just the months leading up to these all-time highs and before the crash our models provide actionable recommendation to help investors possibly make more informed investments.

## Recommendations

1. Look into the new centers that provide the strongest correlations to the market. Keeping them in your daily reading list will be your best use of time in trying to get information on Ethereum.
2. Utilizing the RSI is useful specifically the bought and sold feature as it provides you better prediction power to determining if the market will close higher or lower than the previous day.
3. One feature to keep in mind is monitoring the low historical data. This provides you with the lowest the coin is being traded at.
4. The strongest feature was the volume value of the previous day. The trading volume is a techincal indicatior that represents the overall activity of a security or market. This can be use to legitmize a it's price action, which can then help investors in their decision to either buy or sell.

## Future Work

The model is a great baseline that will be used to explore the top altcoins as well as bitcoin. Also, I will explore if the important feature remains important across all the other cryptocurrency. I would also like to extend the time to a whole year to gather more data to add into the model. By adding more data points, we will be able to see if the model remains a great model or if there is a significant decrease in the metrics that were used to evaluate the performance of the model.

In [ ]:

```
```