BITCOIN PREDICTION 2021

Predict Bitcoin Price for the next 30 days with our Machine Learnig Model.

Final Project by:

- Charlie Burd

- George Quintanilla

- Emmanuel Martinez

In [1]:
```
## Columbia University 2021
```

Part 1 - DATAFRAME, CLEAN AND PREPARE DATA.

In [2]:
```
# Frist, Import Lib needed for our Bitcoin prediction and ETL.
import pandas as pd
import numpy as np
import time
import datetime
import pandas as pd
import numpy as np

# Import Data and Convert into DF
df = pd.read_csv("./Resources/Bitcoin_Historical_Data/Bitcoin_1_Min_Historical_Data_201
# show the first 5 rows
df.head()
```

Out[2]:

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	1325317920	4.39	4.39	4.39	4.39	0.455581	2.0	4.39
1	1325317980	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1325318040	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1325318100	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1325318160	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [3]:
```
# Convert Timestamp into TimeZone (ET)
df['Timestamp'] = (pd.to_datetime(df['Timestamp'], unit='s')
                     .dt.tz_localize('est')
                     .dt.tz_convert('America/New_York'))
# show the first 5 rows
df.head()
```

Out[3]:

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	2011-12-31 07:52:00-05:00	4.39	4.39	4.39	4.39	0.455581	2.0	4.39
1	2011-12-31 07:53:00-05:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2011-12-31 07:54:00-05:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2011-12-31 07:55:00-05:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2011-12-31 07:56:00-05:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [4]:
```
# show the last 5 rows
df.tail()
```

Out[4]:

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weight
4727772	2020-12-30 23:56:00-05:00	28801.47	28829.42	28785.64	28829.42	0.965221	27804.572129	2880
4727773	2020-12-30 23:57:00-05:00	28829.42	28863.90	28829.42	28857.06	2.368831	68332.350629	2884
4727774	2020-12-30 23:58:00-05:00	28850.49	28900.52	28850.49	28882.82	2.466590	71232.784464	2887
4727775	2020-12-30 23:59:00-05:00	28910.54	28911.52	28867.60	28881.30	7.332773	211870.912660	2889
4727776	2020-12-31 00:00:00-05:00	28893.21	28928.49	28893.21	28928.49	5.757679	166449.709320	2890

In [5]:
```
#Drop NaNs from Dataset.
df.dropna(inplace=True)
# show the first 5 rows
df.head()
```

Out[5]:

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
0	2011-12-31 07:52:00-05:00	4.39	4.39	4.39	4.39	0.455581	2.000000	4.390000
478	2011-12-31 15:50:00-05:00	4.39	4.39	4.39	4.39	48.000000	210.720000	4.390000
547	2011-12-31 16:59:00-05:00	4.50	4.57	4.50	4.57	37.862297	171.380338	4.526411

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Volume_(Currency)	Weighted_Price
548	2011-12-31 17:00:00-05:00	4.58	4.58	4.58	4.58	9.000000	41.220000	4.580000
1224	2012-01-01 04:16:00-05:00	4.58	4.58	4.58	4.58	1.502000	6.879160	4.580000

In [6]:

```
# Clean DataFrame for Prediction utilization.
# Remove Volume Currency from DF
df.drop(['Volume_(Currency)'],1,inplace=True)
# show the first 5 rows
df.head()
```

Out[6]:

	Timestamp	Open	High	Low	Close	Volume_(BTC)	Weighted_Price
0	2011-12-31 07:52:00-05:00	4.39	4.39	4.39	4.39	0.455581	4.390000
478	2011-12-31 15:50:00-05:00	4.39	4.39	4.39	4.39	48.000000	4.390000
547	2011-12-31 16:59:00-05:00	4.50	4.57	4.50	4.57	37.862297	4.526411
548	2011-12-31 17:00:00-05:00	4.58	4.58	4.58	4.58	9.000000	4.580000
1224	2012-01-01 04:16:00-05:00	4.58	4.58	4.58	4.58	1.502000	4.580000

In [7]:

```
# Clean DataFrame for Prediction utilization.
# Remove Volume Currency from DF
df.drop(['Volume_(BTC)'],1,inplace=True)
# show the first 5 rows
df.head()
```

Out[7]:

	Timestamp	Open	High	Low	Close	Weighted_Price
0	2011-12-31 07:52:00-05:00	4.39	4.39	4.39	4.39	4.390000
478	2011-12-31 15:50:00-05:00	4.39	4.39	4.39	4.39	4.390000
547	2011-12-31 16:59:00-05:00	4.50	4.57	4.50	4.57	4.526411
548	2011-12-31 17:00:00-05:00	4.58	4.58	4.58	4.58	4.580000
1224	2012-01-01 04:16:00-05:00	4.58	4.58	4.58	4.58	4.580000

In [8]:

```
# Rename Timestamp and Weighted_Price
df.rename(columns = {'Timestamp':'Last Update', 'Weighted_Price':'Weighted Price'}, inp
# show the first 5 rows
df.head()
```

Out[8]:

	Last Update	Open	High	Low	Close	Weighted Price
0	2011-12-31 07:52:00-05:00	4.39	4.39	4.39	4.39	4.390000
478	2011-12-31 15:50:00-05:00	4.39	4.39	4.39	4.39	4.390000
547	2011-12-31 16:59:00-05:00	4.50	4.57	4.50	4.57	4.526411

	Last Update	Open	High	Low	Close	Weighted Price
548	2011-12-31 17:00:00-05:00	4.58	4.58	4.58	4.58	4.580000
1224	2012-01-01 04:16:00-05:00	4.58	4.58	4.58	4.58	4.580000

In [9]:
```
# Clean Last Update into ms.
df['Last Update'] = df['Last Update'].values.astype(dtype='datetime64[ms]')
# show the first 5 rows
df.head()
```

Out[9]:

	Last Update	Open	High	Low	Close	Weighted Price
0	2011-12-31 12:52:00	4.39	4.39	4.39	4.39	4.390000
478	2011-12-31 20:50:00	4.39	4.39	4.39	4.39	4.390000
547	2011-12-31 21:59:00	4.50	4.57	4.50	4.57	4.526411
548	2011-12-31 22:00:00	4.58	4.58	4.58	4.58	4.580000
1224	2012-01-01 09:16:00	4.58	4.58	4.58	4.58	4.580000

In [10]:
```
# Esport Clean DF into CSV
df.to_csv (r'./Resources/Bitcoin_Historical_Data/Bitcoin_2011-2020_Clean.csv', index =
```

In [11]:
```
# Let's create a new DF to use for our prediction, removing the Last Update (Timestamp)
wp_df = df.copy(deep=True)
wp_df.head()
```

Out[11]:

	Last Update	Open	High	Low	Close	Weighted Price
0	2011-12-31 12:52:00	4.39	4.39	4.39	4.39	4.390000
478	2011-12-31 20:50:00	4.39	4.39	4.39	4.39	4.390000
547	2011-12-31 21:59:00	4.50	4.57	4.50	4.57	4.526411
548	2011-12-31 22:00:00	4.58	4.58	4.58	4.58	4.580000
1224	2012-01-01 09:16:00	4.58	4.58	4.58	4.58	4.580000

Let's use 2020 Bitcoin Data only.

In [12]:
```
#Let's filter new DataFrame for 2020 year/data only.
wp2020_df = wp_df[(wp_df["Last Update"] >= '2020-10-01')]
wp2020_df.head()
```

Out[12]:

	Last Update	Open	High	Low	Close	Weighted Price
4596436	2020-10-01 00:00:00	10726.44	10732.79	10722.40	10732.50	10729.037066
4596437	2020-10-01 00:01:00	10731.67	10737.98	10731.67	10737.98	10737.846562
4596438	2020-10-01 00:02:00	10732.89	10735.00	10731.05	10731.16	10733.112729

	Last Update	Open	High	Low	Close	Weighted Price
4596439	2020-10-01 00:03:00	10730.44	10732.23	10725.66	10725.66	10730.994184
4596440	2020-10-01 00:04:00	10730.56	10730.57	10722.32	10722.32	10724.952553

In [13]:
```
wp2020_df.tail()
```

Out[13]:

	Last Update	Open	High	Low	Close	Weighted Price
4727772	2020-12-31 04:56:00	28801.47	28829.42	28785.64	28829.42	28806.429798
4727773	2020-12-31 04:57:00	28829.42	28863.90	28829.42	28857.06	28846.441863
4727774	2020-12-31 04:58:00	28850.49	28900.52	28850.49	28882.82	28879.056266
4727775	2020-12-31 04:59:00	28910.54	28911.52	28867.60	28881.30	28893.695831
4727776	2020-12-31 05:00:00	28893.21	28928.49	28893.21	28928.49	28909.166061

In [14]:
```
# Esport Clean 2020 DF into CSV
wp2020_df.to_csv (r'./Resources/Bitcoin_Historical_Data/Bitcoin_Q32020_Clean.csv', inde
```

Let's Clean the 2020 Columns and Data

In [15]:
```
# Remove Volume Currency from DF
wp2020_df.drop(['Last Update'],1,inplace=True)
# show the first 5 rows
wp2020_df.head()
```

```
C:\Users\emman\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWar
ning:
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/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

Out[15]:

	Open	High	Low	Close	Weighted Price
4596436	10726.44	10732.79	10722.40	10732.50	10729.037066
4596437	10731.67	10737.98	10731.67	10737.98	10737.846562
4596438	10732.89	10735.00	10731.05	10731.16	10733.112729
4596439	10730.44	10732.23	10725.66	10725.66	10730.994184
4596440	10730.56	10730.57	10722.32	10722.32	10724.952553

In [16]:
```
# Remove Volume Currency from DF
wp2020_df.drop(['Open'],1,inplace=True)
# show the first 5 rows
wp2020_df.head()
```

```
C:\Users\emman\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWar
```

```
ning:
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/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

Out[16]:

	High	Low	Close	Weighted Price
4596436	10732.79	10722.40	10732.50	10729.037066
4596437	10737.98	10731.67	10737.98	10737.846562
4596438	10735.00	10731.05	10731.16	10733.112729
4596439	10732.23	10725.66	10725.66	10730.994184
4596440	10730.57	10722.32	10722.32	10724.952553

In [17]:

```
# Remove Volume Currency from DF
wp2020_df.drop(['High'],1,inplace=True)
# show the first 5 rows
wp2020_df.head()
```

```
C:\Users\emman\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWar
ning:
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/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

Out[17]:

	Low	Close	Weighted Price
4596436	10722.40	10732.50	10729.037066
4596437	10731.67	10737.98	10737.846562
4596438	10731.05	10731.16	10733.112729
4596439	10725.66	10725.66	10730.994184
4596440	10722.32	10722.32	10724.952553

In [18]:

```
# Remove Volume Currency from DF
wp2020_df.drop(['Low'],1,inplace=True)
# show the first 5 rows
wp2020_df.head()
```

```
C:\Users\emman\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWar
ning:
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/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

Out[18]:

	Close	Weighted Price
4596436	10732.50	10729.037066
4596437	10737.98	10737.846562

	Close	Weighted Price
4596438	10731.16	10733.112729
4596439	10725.66	10730.994184
4596440	10722.32	10724.952553

In [19]:
```
# Remove Volume Currency from DF
wp2020_df.drop(['Close'],1,inplace=True)
# show the first 5 rows
wp2020_df.head()
```

```
C:\Users\emman\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopyWar
ning:
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/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

Out[19]:

	Weighted Price
4596436	10729.037066
4596437	10737.846562
4596438	10733.112729
4596439	10730.994184
4596440	10724.952553

In [20]:
```
#Now we have a clean Weighted Price to predict some Bitcoin Price for the next 30 days.
```

Part 2 - PREPARE DATA FOR BITCOIN PRICE PREDICTION

30 Days Prediction Model

In [21]:
```
# Machine Learning Coding.
# 30 Days Prediction.
```

In [22]:
```
# Create Variable bitCoinPredictionDays to set the days.
bitCoinPredictionDays = 30
```

In [23]:
```
# Create another column shifted 'n'  units up
wp2020_df['Prediction > 30 Days'] = wp2020_df[['Weighted Price']].shift(-bitCoinPredict
# show the first 5 rows
wp2020_df.head()
```

```
<ipython-input-23-e44da41661f7>: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/stable/user_
```

```
guide/indexing.html#returning-a-view-versus-a-copy
  wp2020_df['Prediction > 30 Days'] = wp2020_df[['Weighted Price']].shift(-bitCoinPredic
tionDays)
```

Out[23]:

	Weighted Price	Prediction > 30 Days
4596436	10729.037066	10694.332210
4596437	10737.846562	10693.748406
4596438	10733.112729	10697.376797
4596439	10730.994184	10694.319800
4596440	10724.952553	10702.741126

In [24]:

```
# show the last 5 rows from new dataframe
wp2020_df.to_csv (r'./Resources/Bitcoin_Historical_Data/30_Days_Price_Prediction_Bitcoi
wp2020_df.tail()
```

Out[24]:

	Weighted Price	Prediction > 30 Days
4727772	28806.429798	NaN
4727773	28846.441863	NaN
4727774	28879.056266	NaN
4727775	28893.695831	NaN
4727776	28909.166061	NaN

Part 3 - CREATE AN INDEPENDENT DATASET

In [25]:

```
# Let's Create the independent Dataset.
# Here we will convert our wp_df Dataframe into a Numpy Array and drop the "Prediction
x = np.array(wp2020_df.drop(['Prediction > 30 Days'],1))

# Remove the last 'n' rows where 'n' is the bitCoinPredictionDays
x = x[:len(wp2020_df)-bitCoinPredictionDays]
print(x)
```

```
[[10729.037066]
 [10737.846562]
 [10733.112729]
 ...
 [28839.117804]
 [28830.61964 ]
 [28839.418978]]
```

Part 4 - CREATE A DEPENDENT DATASET

In [26]:

```
# Create the dependent Dataset.
# Convert the DataFrame into a Numpy Array
y = np.array(wp2020_df['Prediction > 30 Days'])

# Get all the values except last 'n' rows
y = y[:-bitCoinPredictionDays]
print(y)
```

```
[10694.33221  10693.748406 10697.376797 ... 28879.056266 28893.695831
 28909.166061]
```

Part 5 - SPLIT DATA FOR MACHINE LEARNING TEST

In [27]:
```
# Import libs, and Split the data into 80% training and 20% testing
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size = 0.2)
```

In [28]:
```
# Set the predictionDays array equal to last 30 rows from the original data set
bitCoinPredictionDays_array = np.array(wp2020_df.drop(['Prediction > 30 Days'],1))[-bit

print(bitCoinPredictionDays_array)
```

```
[[28866.70587 ]
 [28856.483639]
 [28858.865737]
 [28862.944384]
 [28874.291006]
 [28911.544955]
 [28909.842736]
 [28906.612263]
 [28892.668831]
 [28872.926497]
 [28880.63073 ]
 [28879.380747]
 [28859.108745]
 [28856.848249]
 [28845.828559]
 [28879.940696]
 [28884.159258]
 [28857.404737]
 [28838.32483 ]
 [28829.425958]
 [28838.37645 ]
 [28837.207539]
 [28814.411945]
 [28812.73295 ]
 [28810.597267]
 [28806.429798]
 [28846.441863]
 [28879.056266]
 [28893.695831]
 [28909.166061]]
```

Now let's create a Machine Learning Model

Part 6 - CREATE AND TRAIN VECTOR MACHINE MODEL

In [29]:
```
# Start Vector Machine (Regression)
```

In [30]:
```
# We need first to install sklearn if missing: (We already installed - as comment below
#pip install sklearn
```

In [31]:
```
from sklearn.svm import SVR
```

```
# Create and Train the Support Vector Machine (Regression) using radial basis function
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.00001)
svr_rbf.fit(xtrain, ytrain)
```

Out[31]:
```
SVR(C=1000.0, gamma=1e-05)
```

Now let's Test the Machine Learning Model.

Part 7 - TEST MODEL

In [32]:
```
# Create a test.
svr_rbf_confidence = svr_rbf.score(xtest,ytest)
print('SVR_RBF accuracy :',svr_rbf_confidence)
```

```
SVR_RBF accuracy : 0.999592485242336
```

In [34]:
```
# print the predicted values
svm_prediction = svr_rbf.predict(xtest)
print(svm_prediction)
print()
print(ytest)
```

```
[10890.34130352 15619.5211721  13829.13119734 ... 12924.31558854
 23664.24685555 16681.45042132]

[10880.307555 15702.175944 13816.064961 ... 12933.64086  23704.590696
 16654.915259]
```

In [37]:
```
#svm_prediction.to_csv (r'./Resources/Bitcoin_Historical_Data/SVM_ML_Bitcoin_Q32020_Cle
```

In [38]:
```
#ytest.to_csv (r'./Resources/Bitcoin_Historical_Data/YTest_ML_Bitcoin_Q32020_Clean.csv'
```

In [41]:
```
# Print the model predictions for the next 30 days
svm_prediction = svr_rbf.predict(bitCoinPredictionDays_array)
print(svm_prediction)
print()
```

```
[28830.52789956 28827.79445355 28828.53530909 28829.66519512
 28831.63243233 28817.06838128 28818.72411394 28821.55041624
 28829.49518056 28831.50376553 28831.76548899 28831.80362233
 28828.60772343 28827.91137049 28823.92907783 28831.79065563
 28831.47138588 28828.08742043 28820.85996275 28817.09275443
 28820.88160859 28820.39030659 28810.9958013  28810.36425488
 28809.58082227 28808.1221527  28824.17056057 28831.80812257
 28829.12264871 28819.34993761]
```

In [53]:
```
prediction_df = pd.DataFrame(svm_prediction)

prediction_df.head()
```

Out[53]:

	0

	0
0	28830.527900
1	28827.794454
2	28828.535309
3	28829.665195
4	28831.632432

In [54]:
```
#Print the actual price for bitcoin for last 30 days
print(prediction_df.tail(bitCoinPredictionDays))
```

```
               0
0   28830.527900
1   28827.794454
2   28828.535309
3   28829.665195
4   28831.632432
5   28817.068381
6   28818.724114
7   28821.550416
8   28829.495181
9   28831.503766
10  28831.765489
11  28831.803622
12  28828.607723
13  28827.911370
14  28823.929078
15  28831.790656
16  28831.471386
17  28828.087420
18  28820.859963
19  28817.092754
20  28820.881609
21  28820.390307
22  28810.995801
23  28810.364255
24  28809.580822
25  28808.122153
26  28824.170561
27  28831.808123
28  28829.122649
29  28819.349938
```

In [57]:
```
# Esport Final ML Results for 2020 DF into CSV
prediction_df.to_csv (r'./Resources/Bitcoin_Historical_Data/FINAL_PREDICTION_ML_Q3Bitco
```

In [ ]:
```
# Final Prediction
```

In [ ]:
```
# Columbia University
# by Emmanuel Martinez, Charlie Burd and George Quintanilla
# Q1 2021 Final Project
```