

# CIS 5900

## Group 4

Smit Shiroya  
Kathryn Mak

Monish Phatarpeka  
Pratik Parmar

Sushant Burde  
Swarnim Jambhule

## “Bitcoin Price Prediction using deep learning”

---

### Objectives

**List what your objectives are :** In this hands-on lab, you will learn how to:

- Visualization using kibana
- Build neural network system
- Classification methods
- Train Dataset and predict the price

### Platform Spec

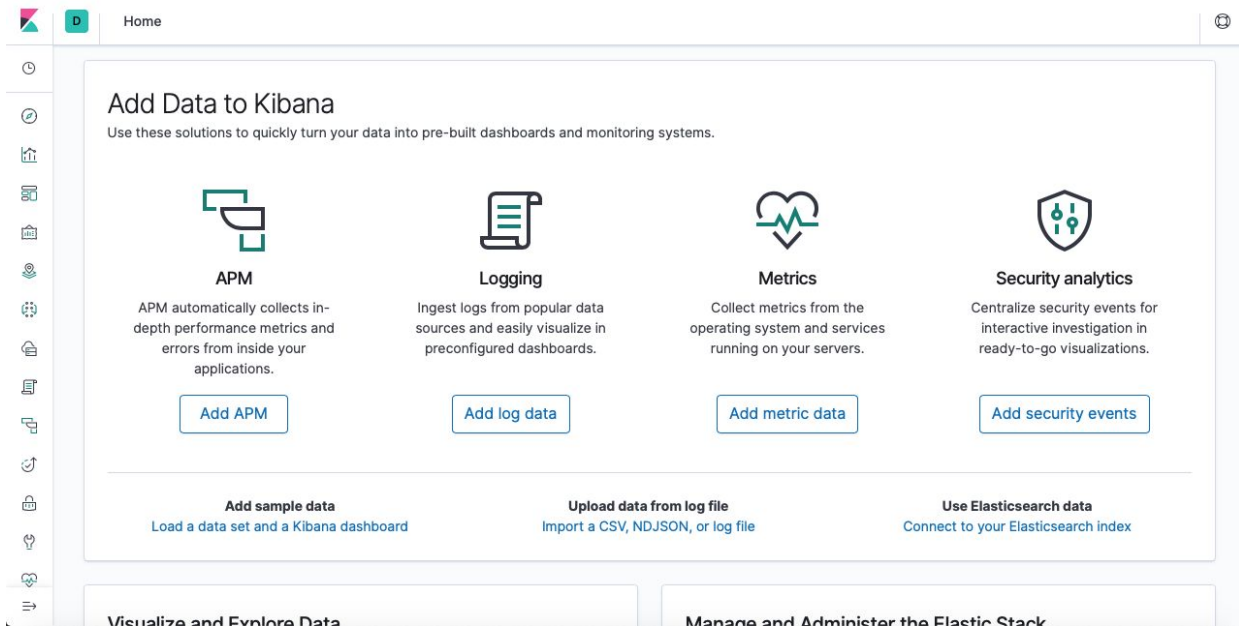
- System : MacOS/Windows
- Total Memory Size: 8GB
- Processor : 2.5 GHz
- Storage : 500Gb
- Memory : 8 GB Ram
- Kibana : Version 7.1
- Jupyter : Version 6.0

### Install and Start Kibana

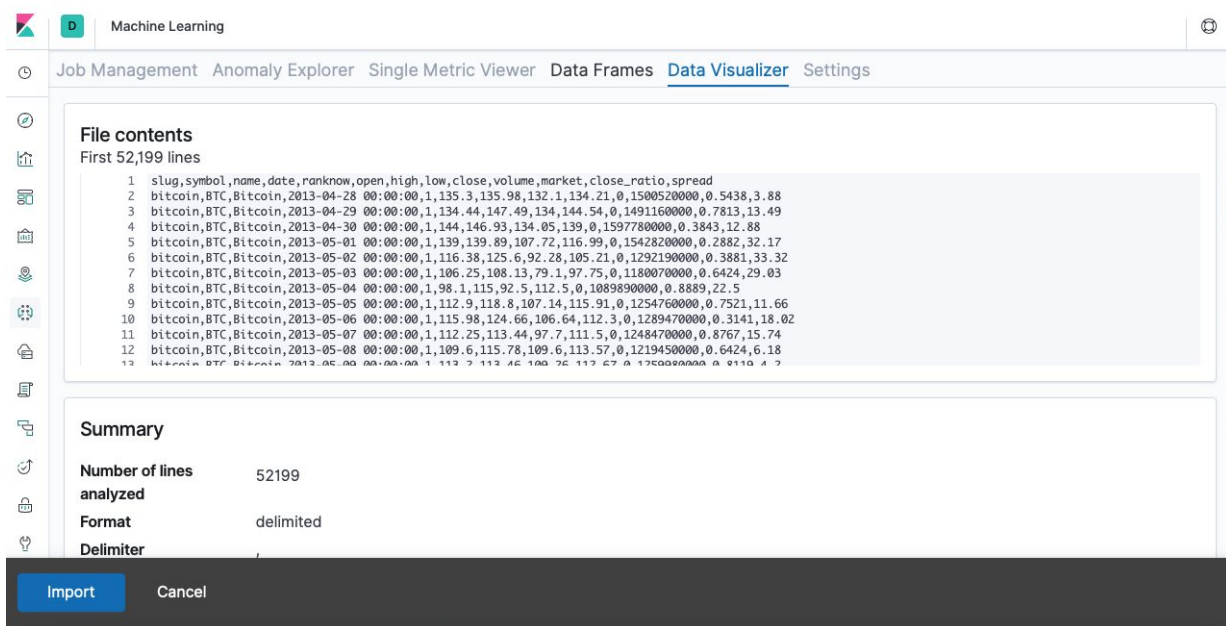
1) To get Kibana up and running:

1. Download the Kibana 4 binary package for your platform.
2. Extract the .zip or tar.gz archive file.
3. Run Kibana from the install directory: bin/kibana (Linux/MacOSX) or bin\kibana.bat(Windows).

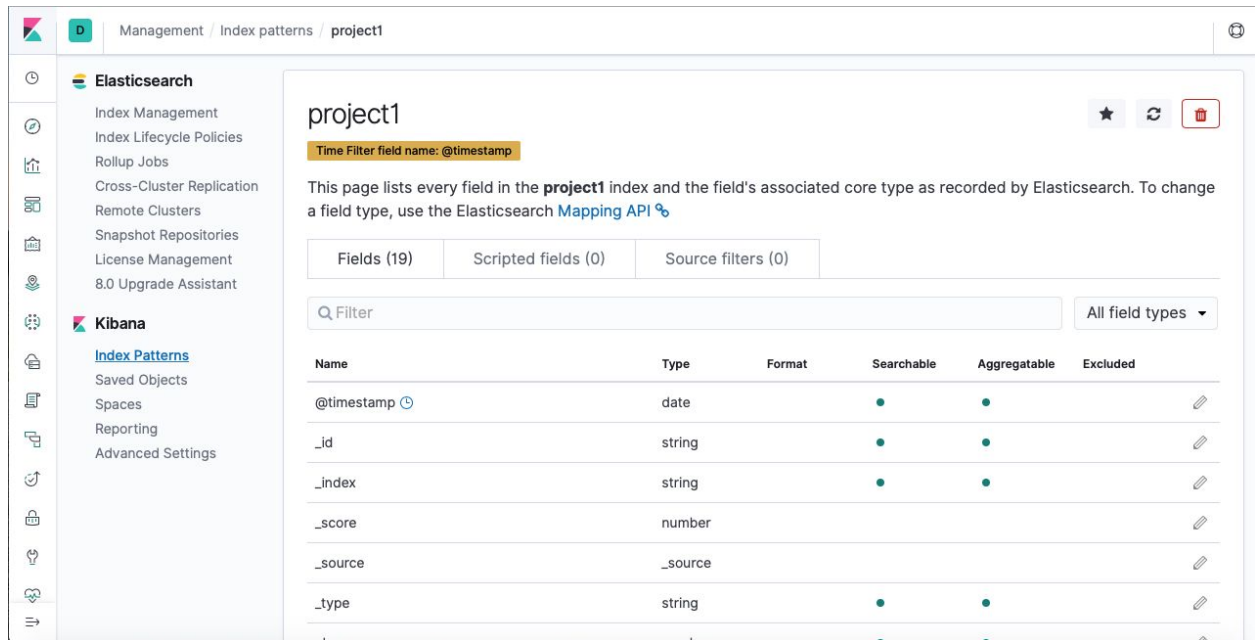
2) Click the kibana icon button and upload data from login (.csv file) for the visualization.



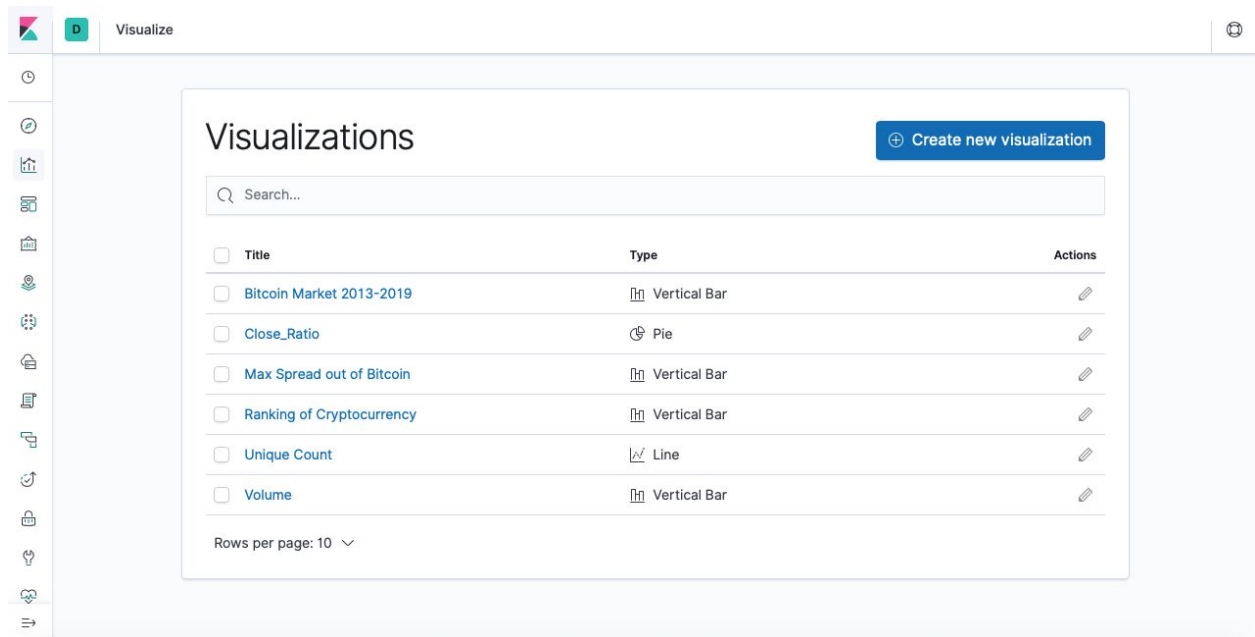
3) After uploading the file click the import button. It will take 2-3 minutes to upload and then give an index pattern name for the data.



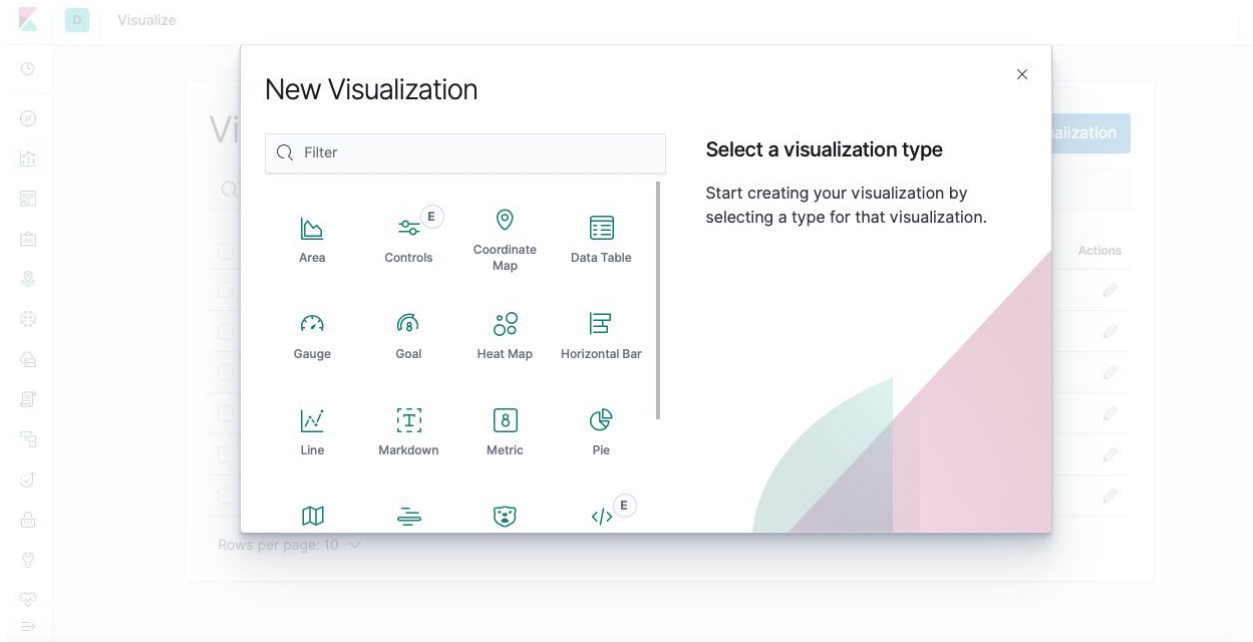
4) In this data the index pattern name is project1.



5) In Kibana page, open Visualize.

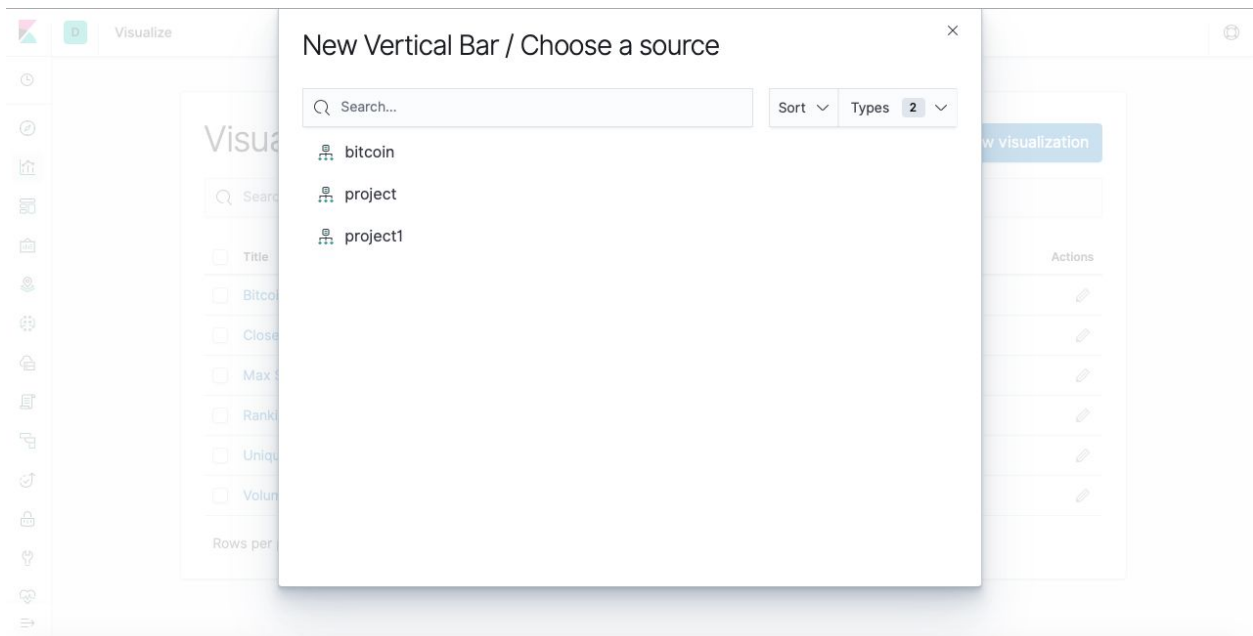


6) Click Create a visualization or the + button. You'll see all the visualization types in Kibana.



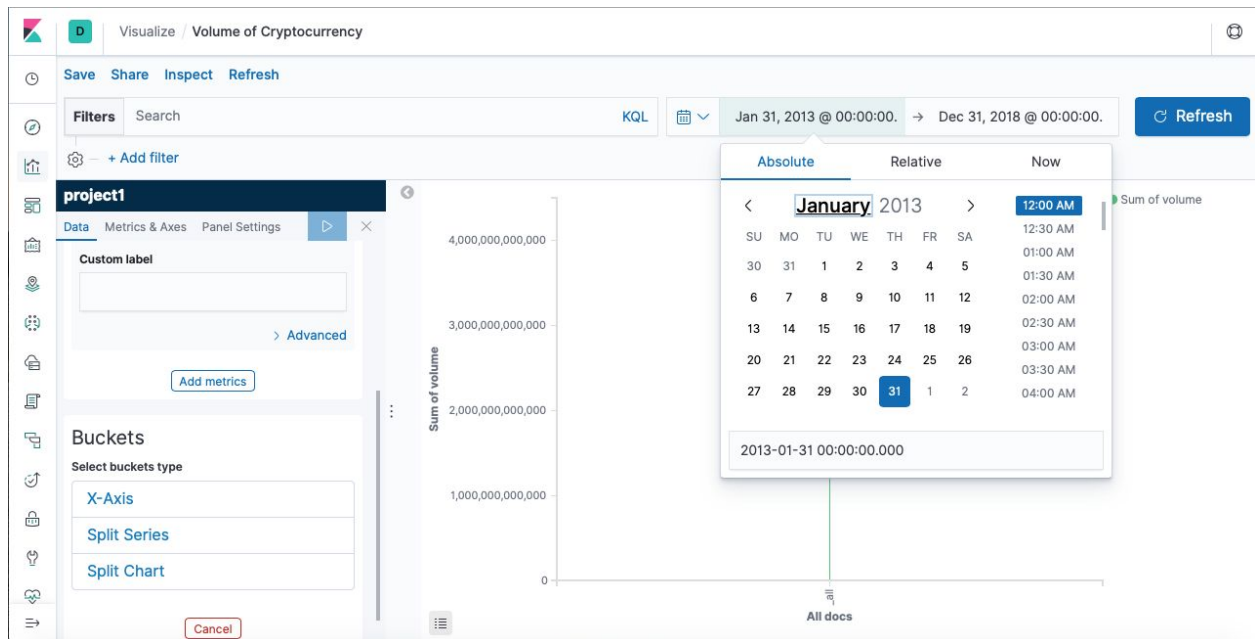
7) Click Line Chart.

8) In New Search, select the project1\* index pattern. We'll use the line chart to get the trend of the cryptocurrency.

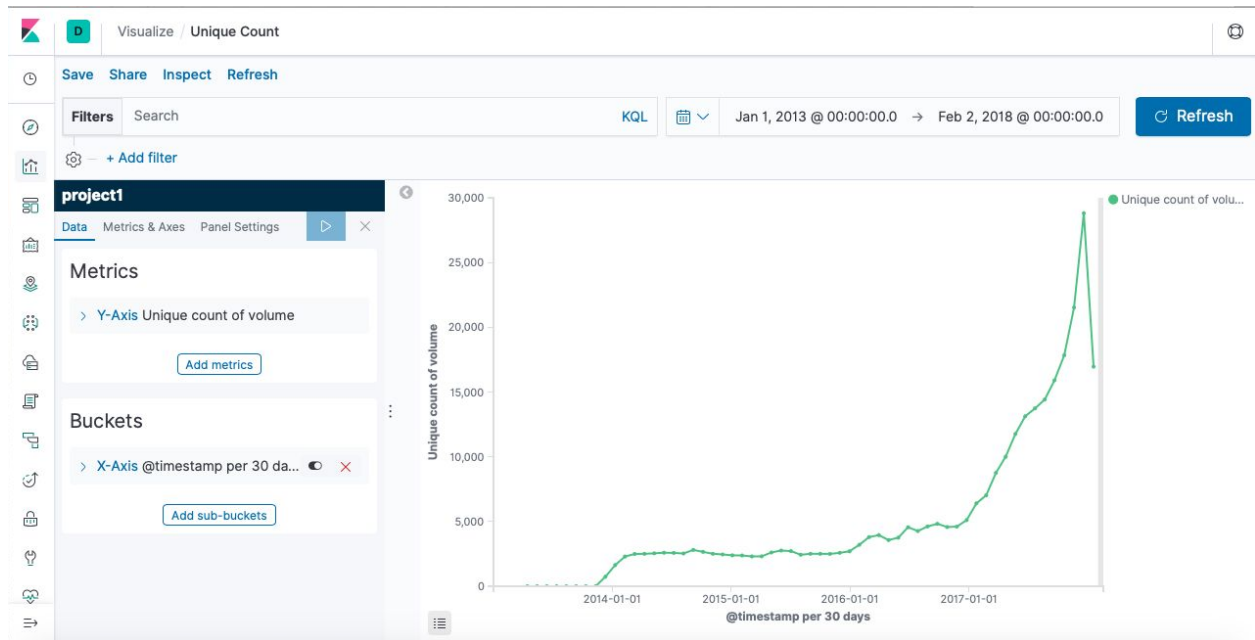


9) Set the Date and Time to get the data from an index pattern.

- In the top menu bar, click the time picker on the far right.
- Click Absolute.
- Set the start time to Jan 31, 2013 and the end time to Dec 31, 2018.
- Click Go.



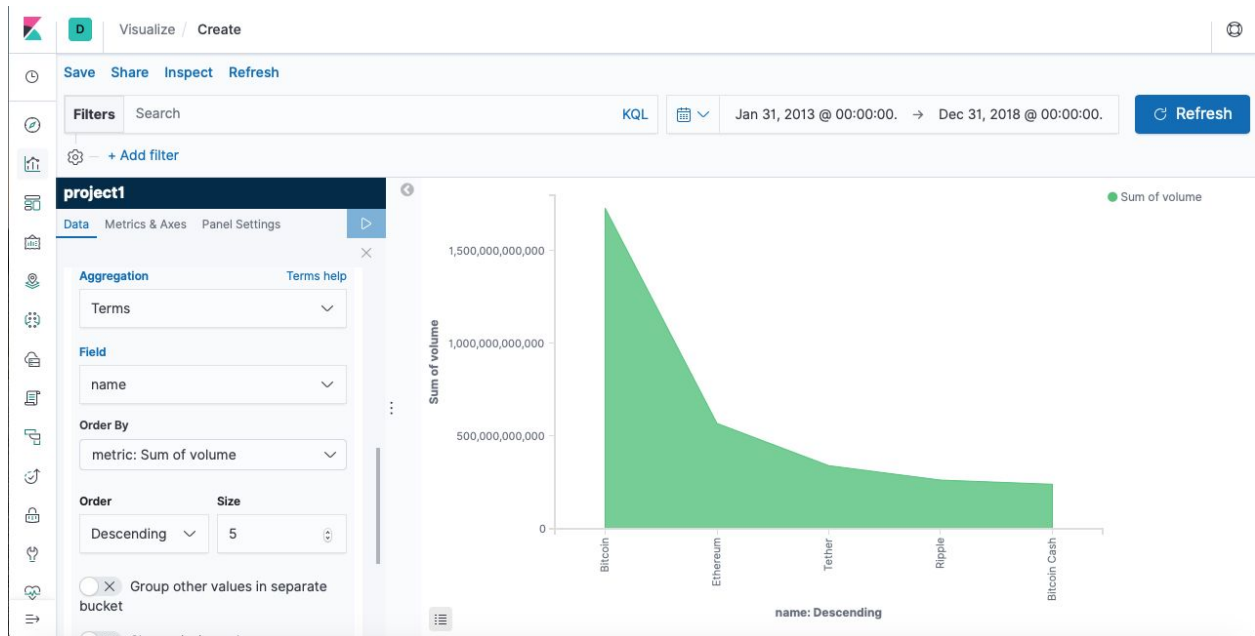
- In the Metrics pane, expand Y-Axis.
- Set Aggregation to Unique Count.
- Set Field to volume.
- Again in the metrics pane, expand X-Axis
- Set Aggregation to Date Histogram.
- Set field to @timestamp.
- Set Min Interval to yearly.
- Click apply button.



- Save the chart as “Unique Count”.

10) In New Search, select the project1\* index pattern. We'll use the vertical bar to get the total volume of the cryptocurrency.

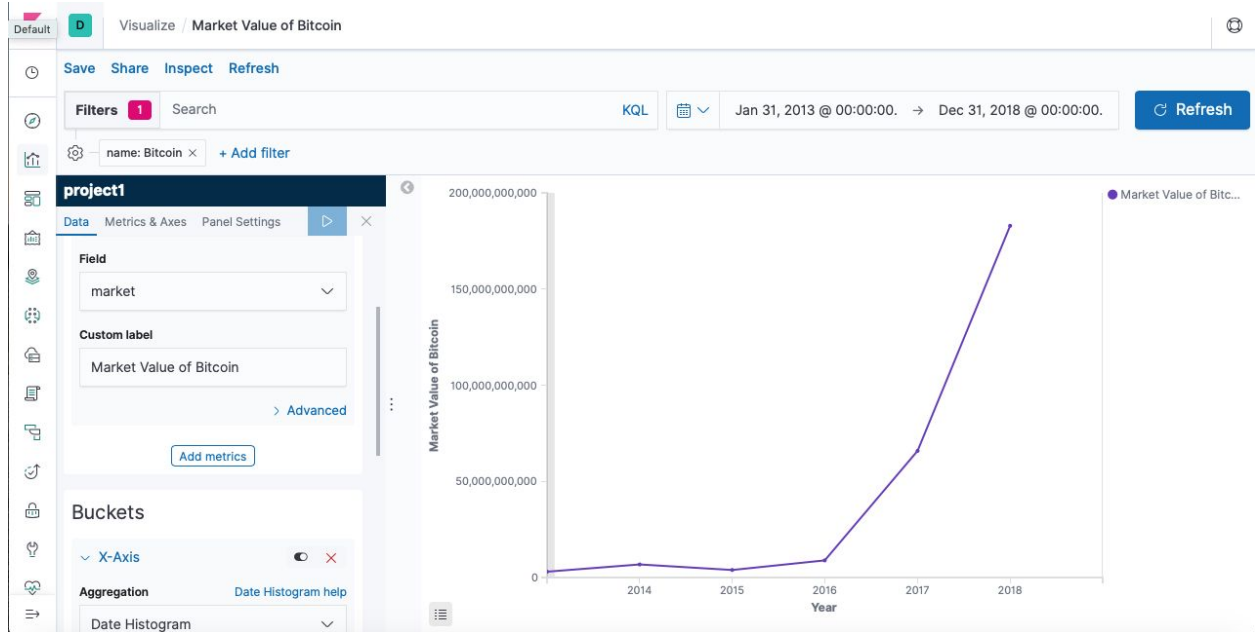
- In the Metrics pane, expand Y-Axis.
- Set Aggregation to sum.
- Set Field to volume.
- Again in the metrics pane, expand X-Axis
- Set Aggregation to terms.
- Set field to name.
- Select order to descending.
- In the top menu bar, click the time picker on the far right.
- Click apply button.



- Save the chart as “Volume of Cryptocurrency”.

11) In New Search, select the project1\* index pattern. We’ll use the **line chart** to get the market value of the bitcoin.

- Repeat the step 5 and 6. That is, to select Visualize menu in the left frame of the Kibana
- Select and create a Line chart and set the search source to project1\*.
- Repeat the step 9 to set the time interval.
- In the Metrics pane, expand Y-Axis.
- Set Aggregation to average.
- Set Field to market.
- Give the custom label name : market
- Again in the metrics pane, expand X-Axis
- Set Aggregation to date histogram.
- Set field to @timestamp.
- Set the minimum interval to yearly
- Give the custom label name : year
- Click apply button.

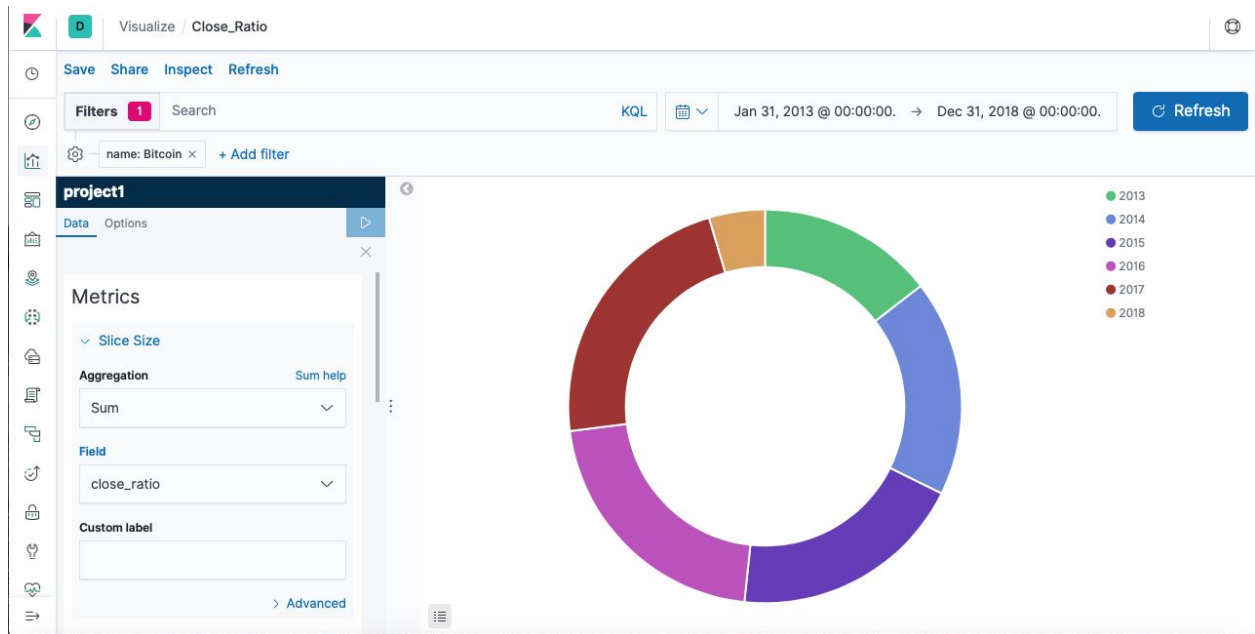


- Save the chart as “Market Value of Bitcoin”.

12) Now we will use pie chart to get an idea about close ratio of bitcoin from the year 2013 to 2018.

- Repeat the step 5 and 6. That is, to select Visualize menu in the left frame of the Kibana
- Select and create a Pie chart and set the search source to project1\*.
- Repeat the step 9 to set the time interval.
- In the Metrics pane.
- Set Aggregation to sum.
- Set Field to Close ratio  $((\text{Close}-\text{Low})/(\text{High}-\text{Low}))$ .
- In the bucket expand split slices.
- Set Aggregation to date histogram.
- Set field to @timestamp.
- Set the minimum interval to yearly.
- Click apply button.

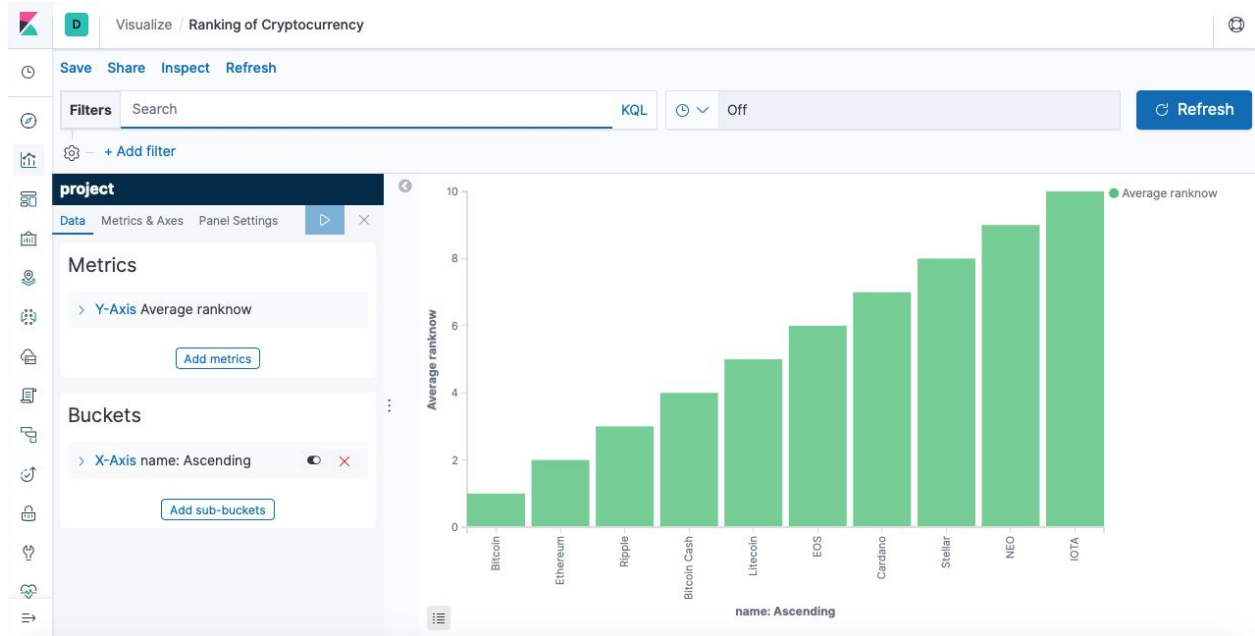




- Save the chart as “Close-Ratio of Bitcoin”.

### 13) Creating a bar chart to get the rank of cryptocurrency.

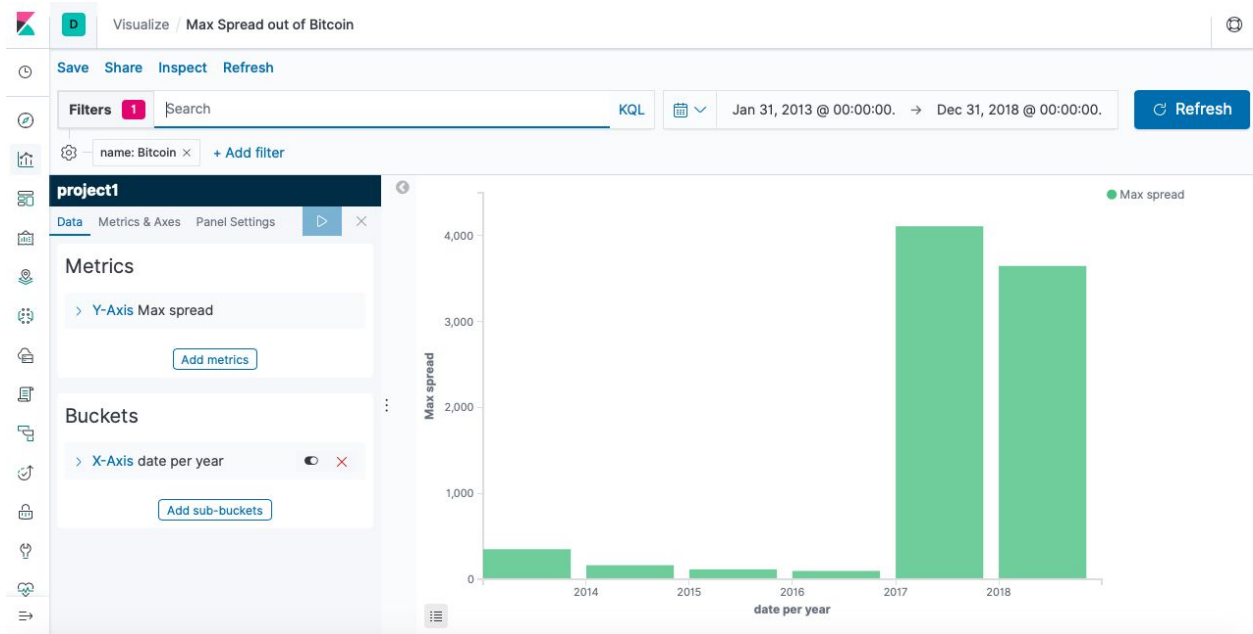
- Repeat the step 5 and 6. That is, to select Visualize menu in the left frame of the Kibana
- Select and create a bar chart and set the search source to project1\*.
- Repeat the step 9 to set the time interval.
- In the Metrics pane , expand Y-axis.
- Set Aggregation to average.
- Set field to ranknow.
- Give the custom label name : Ranking
- In the buckets pane, expand X-axis.
- Set Aggregation to terms.
- Set field to name and change the into Ascending and set the size equal to 10.
- Give the custom label name : Cryptocurrency.
- Click apply button.



- Save the chart as “Ranking of cryptocurrency”.

#### 14) Creating a bar chart to get the max spread out of bitcoin.

- Repeat the step 5 and 6. That is, to select Visualize menu in the left frame of the Kibana
- Select and create a bar chart and set the search source to project1\*.
- Repeat the step 9 to set the time interval.
- In the Metrics pane , expand Y-axis.
- Set Aggregation to Max.
- Set field to spread.
- In the bucket expand X-axis.
- Set Aggregation to date histogram.
- Set field to date.
- Set the minimum interval to yearly.
- Click apply button.



- Save the chart as “Max Spread out of Bitcoin”.

## Apply Deep Learning

- Requirement : System (Mac or Windows) , Jupyter Notebook , Python Libraries (Numpy , Keras, Scikit Learn)
- To install Jupyter we recommend you to install anaconda which is a great tool and contains all the tools like jupyter , tensorflow , numpy ,spyder which we can access just from one software: “<https://www.anaconda.com/distribution/>”
- Note : if you are not using jupyter then you will have to install all the libraries manually which.
- To install library : Open terminal and type command - “**\$ pip install library name**”
- We recommend you to use jupyter because it comes with all the libraries installed.

- 1) Now open anaconda in your desktop and select jupyter from the list. It will run on the local server.
- 2) Type the python command and generate a prediction graph and accuracy data in the output format.

# Output

## Python Command :

### 1) Import libraries :

```
import datetime as dt

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt

from plotly import tools

import plotly.offline as py

py.init_notebook_mode(connected=True)

import plotly.graph_objs as go

import xgboost as xgb

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from sklearn.model_selection import train_test_split

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

import warnings

warnings.filterwarnings("ignore")
```

### 2) Read .CSV file

```
df = pd.read_csv('crypto-markets.csv')

df.info() #gives information about your dataset
```

3) Added a column which is "date" and I converted "Timestamp" columns to date form. Also deleted symbol and market columns and group by according to closing price.

```
df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')
```

```
df = df.drop(['symbol', 'market'], axis=1)
```

```
group=df.groupby("date")
```

```
data=group["close"].mean()
```

```
data.shape
```

```
data.isnull().sum()
```

4) Separating last 50 rows as the test data.

```
close_train=data.iloc[:len(data)-50]
```

```
close_test=data.iloc[len(close_train):]
```

5) Setting our values between 0-1 in order to avoid domination of high values.

```
close_train=np.array(close_train)
```

```
close_train=close_train.reshape(close_train.shape[0],1)
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler=MinMaxScaler(feature_range=(0,1))
```

```
close_scaled=scaler.fit_transform(close_train)
```

6) Now , Choosing each 50 data as x-train and 51th as y-train

```
timestep=50
```

```
x_train=[]
```

```
y_train=[]
```

```
for i in range(timestep,close_scaled.shape[0]):
```

```
    x_train.append(close_scaled[i-timestep:i,0])
```

```
    y_train.append(close_scaled[i,0])
```

```

x_train,y_train=np.array(x_train),np.array(y_train)

x_train=x_train.reshape(x_train.shape[0],x_train.shape[1],1) #reshaped for RNN

print("x_train shape= ",x_train.shape)

print("y_train shape= ",y_train.shape)

```

7) Now it is time to Use LSTM method to predict price

```

from sklearn.metrics import mean_absolute_error

from keras.models import Sequential

from keras.layers import Dense, LSTM, Dropout, Flatten

model=Sequential()

model.add(LSTM(10,input_shape=(None,1),activation="relu")) #relu is activation method

model.add(Dense(1))

model.compile(loss="mean_squared_error",optimizer="adam") #Adam is optimizer algorithm

model.fit(x_train,y_train,epochs=100,batch_size=32) #100epochs means it will run the process 100 times and will train the data

```

8) Keep testing the data

```

inputs=data[len(data)-len(close_test)-timestep:]

inputs=inputs.values.reshape(-1,1)

inputs=scaler.transform(inputs)

x_test=[]

for i in range(timestep,inputs.shape[0]):

    x_test.append(inputs[i-timestep:i,0])

x_test=np.array(x_test)

x_test=x_test.reshape(x_test.shape[0],x_test.shape[1],1)

```

9) Now it is time to predict the value

```
predicted_data=model.predict(x_test)

predicted_data=scaler.inverse_transform(predicted_data)

data_test=np.array(close_test)

data_test=data_test.reshape(len(data_test),1)
```

10) Plotting a graph of true result vs predicted result

```
plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')

plt.plot(data_test,color="r",label="true result")

plt.plot(predicted_data,color="b",label="predicted result")

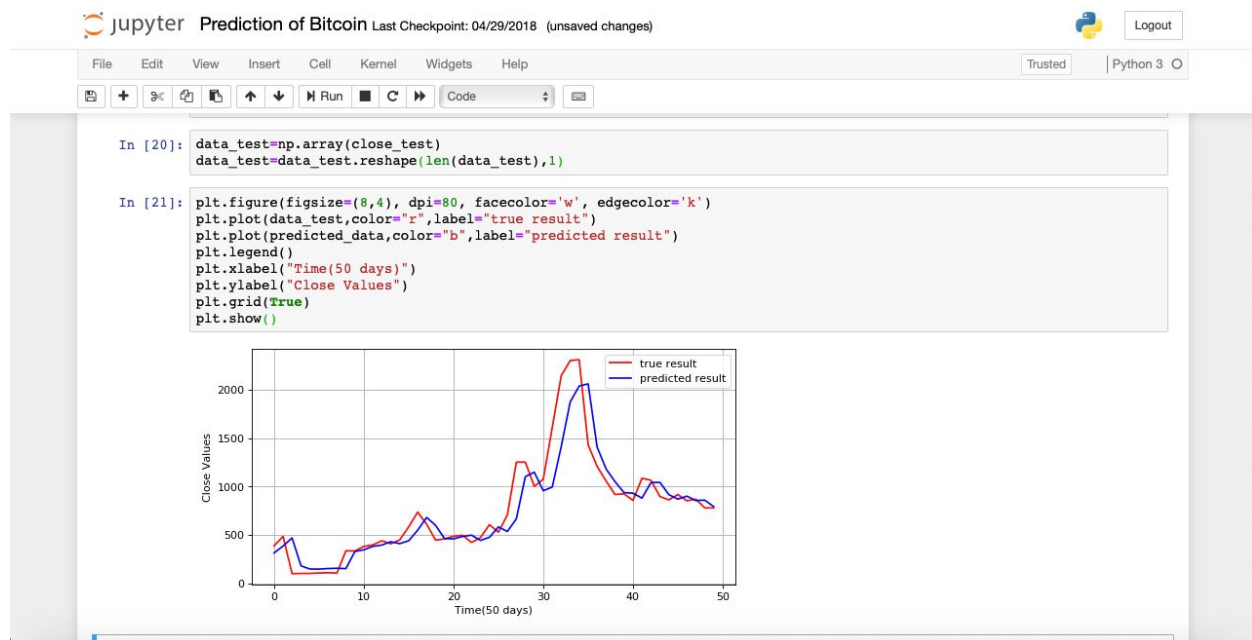
plt.legend()

plt.xlabel("Time(50 days)")

plt.ylabel("Close Values")

plt.grid(True)

plt.show()
```



- By using plot library which is used for creating graph , Red line indicates the true result of the data and the blue line represents the output that we predicted using deep learning.

11) Using different classification it will calculate the accuracy of the model , here we are using 3 regression techniques (Linear regression,Random Forest regression,Gradient Boosting regression ) and will compare the result.

```
classifiers = {
    'LinearRegression': LinearRegression(),
    'Random Forest Regressor': RandomForestRegressor(n_estimators=100, random_state=1),
    'Gradient Boosting Regressor': GradientBoostingRegressor(n_estimators=500)
}
```

```
summary = list()
```

```
for name, clf in classifiers.items():
```

```
    print(name)
```

```
    nada = clf.fit(X_train, y_train)
```



```

print(f'R2: {r2_score(y_test, clf.predict(X_test)):.2f}')

print(f'MAE: {mean_absolute_error(y_test, clf.predict(X_test)):.2f}')

print(f'MSE: {mean_squared_error(y_test, clf.predict(X_test)):.2f}')

print()

summary.append({

    'MSE': mean_squared_error(y_test, clf.predict(X_test)),

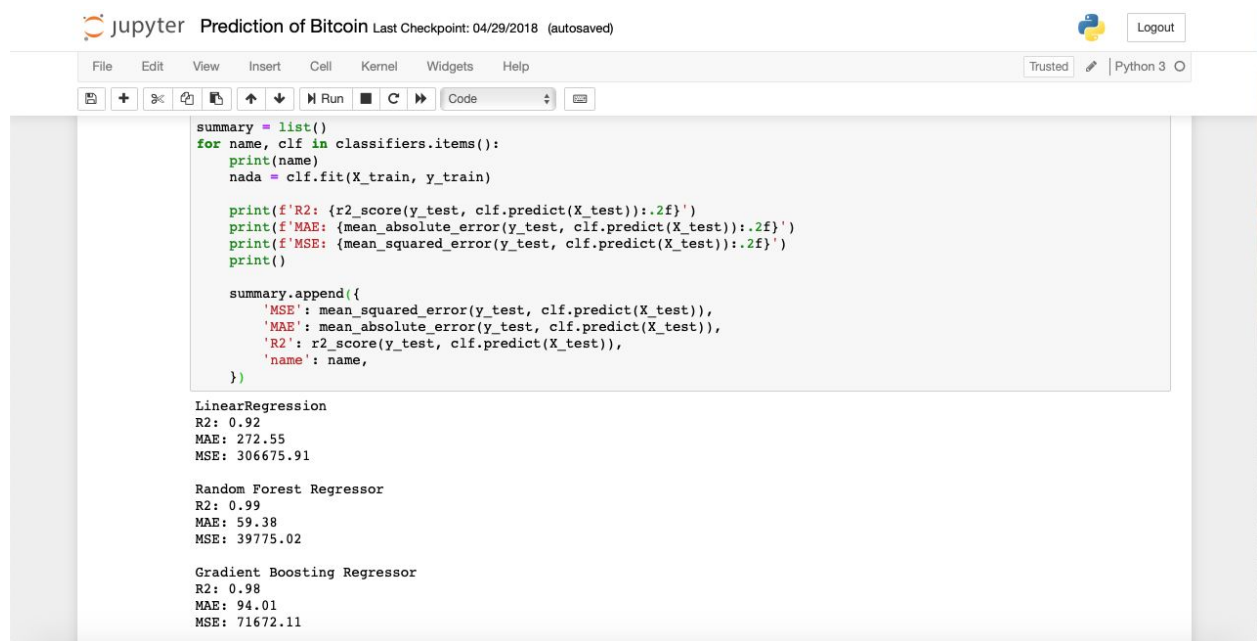
    'MAE': mean_absolute_error(y_test, clf.predict(X_test)),

    'R2': r2_score(y_test, clf.predict(X_test)),

    'name': name,

})

```



Jupyter Prediction of Bitcoin Last Checkpoint: 04/29/2018 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```

summary = list()
for name, clf in classifiers.items():
    print(name)
    nada = clf.fit(X_train, y_train)

    print(f'R2: {r2_score(y_test, clf.predict(X_test)):.2f}')
    print(f'MAE: {mean_absolute_error(y_test, clf.predict(X_test)):.2f}')
    print(f'MSE: {mean_squared_error(y_test, clf.predict(X_test)):.2f}')
    print()

    summary.append({
        'MSE': mean_squared_error(y_test, clf.predict(X_test)),
        'MAE': mean_absolute_error(y_test, clf.predict(X_test)),
        'R2': r2_score(y_test, clf.predict(X_test)),
        'name': name,
    })

```

LinearRegression  
R2: 0.92  
MAE: 272.55  
MSE: 306675.91

Random Forest Regressor  
R2: 0.99  
MAE: 59.38  
MSE: 39775.02

Gradient Boosting Regressor  
R2: 0.98  
MAE: 94.01  
MSE: 71672.11

- To check the accuracy of the predicted data there are 3 methods to calculate value.
  - 1) Linear Regression
  - 2) Random Forest Regressor
  - 3) Gradient Boosting Regressor
- By applying 3 of them as we can see that random forest regressor is the efficient and preferable algorithm for this data which shows the prediction result is 99% accurate.

## References

1. URL of Data Source, <https://www.kaggle.com/jessevent/all-crypto-currencies>
2. URL of your Github, <https://github.com/sburde71/CIS-5900-Group-4>
3. URL of References,
  - <https://skymind.ai/wiki/lstm>
  - <https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714>
  - [https://en.wikipedia.org/wiki/Statistical\\_classification](https://en.wikipedia.org/wiki/Statistical_classification)

---

---