In [80]:

```
import datetime as dt

import numpy as np
import pandas as pd

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
```

In [81]:

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

In [82]:

```
df.head()
```

Out[82]:

	slug	symbol	name	date	ranknow	open	high	low	close	volume
0	bitcoin	BTC	Bitcoin	4/28/13	1	135.30	135.98	132.10	134.21	0.0
1	bitcoin	BTC	Bitcoin	4/29/13	1	134.44	147.49	134.00	144.54	0.0
2	bitcoin	BTC	Bitcoin	4/30/13	1	144.00	146.93	134.05	139.00	0.0
3	bitcoin	BTC	Bitcoin	5/1/13	1	139.00	139.89	107.72	116.99	0.0
4	bitcoin	BTC	Bitcoin	5/2/13	1	116.38	125.60	92.28	105.21	0.0

```
In [118]: df['date']=pd.to_datetime(df['date'], errors='raise')
```

```
Exception ignored in: <bound method DMatrix.__del__ of <xgboost.core.DMatrix object at 0x1a34c57470>
>
Traceback (most recent call last):
  File "/Users/Christina/anaconda3/lib/python3.6/site-packages/xgboost/core.py", line 366, in __del_
_
    if self.handle is not None:
AttributeError: 'DMatrix' object has no attribute 'handle'
```

```
In [142]: df = df.drop(['symbol'], axis=1)
```

```
In [143]: df['hlc_average'] = (df['high'] + df['low'] + df['close']) / 3
          df['ohlc_average'] = (df['open'] + df['high'] + df['low'] + df['close']) / 4
```

```
In [144]: df.head()
```

Out[144]:

	slug	name	date	ranknow	open	high	low	close	volume	hlc_average	ohlc_average
1344	bitcoin	Bitcoin	2017-01-01	1	963.66	1003.08	958.70	998.33	147775000.0	986.703333	980.9425
1345	bitcoin	Bitcoin	2017-01-02	1	998.62	1031.39	996.70	1021.75	222185000.0	1016.613333	1012.1150
1346	bitcoin	Bitcoin	2017-01-03	1	1021.60	1044.08	1021.60	1043.84	185168000.0	1036.506667	1032.7800
1347	bitcoin	Bitcoin	2017-01-04	1	1044.40	1159.42	1044.40	1154.73	344946000.0	1119.516667	1100.7375
1348	bitcoin	Bitcoin	2017-01-05	1	1156.73	1191.10	910.42	1013.38	510199000.0	1038.300000	1067.9075

In [184]:

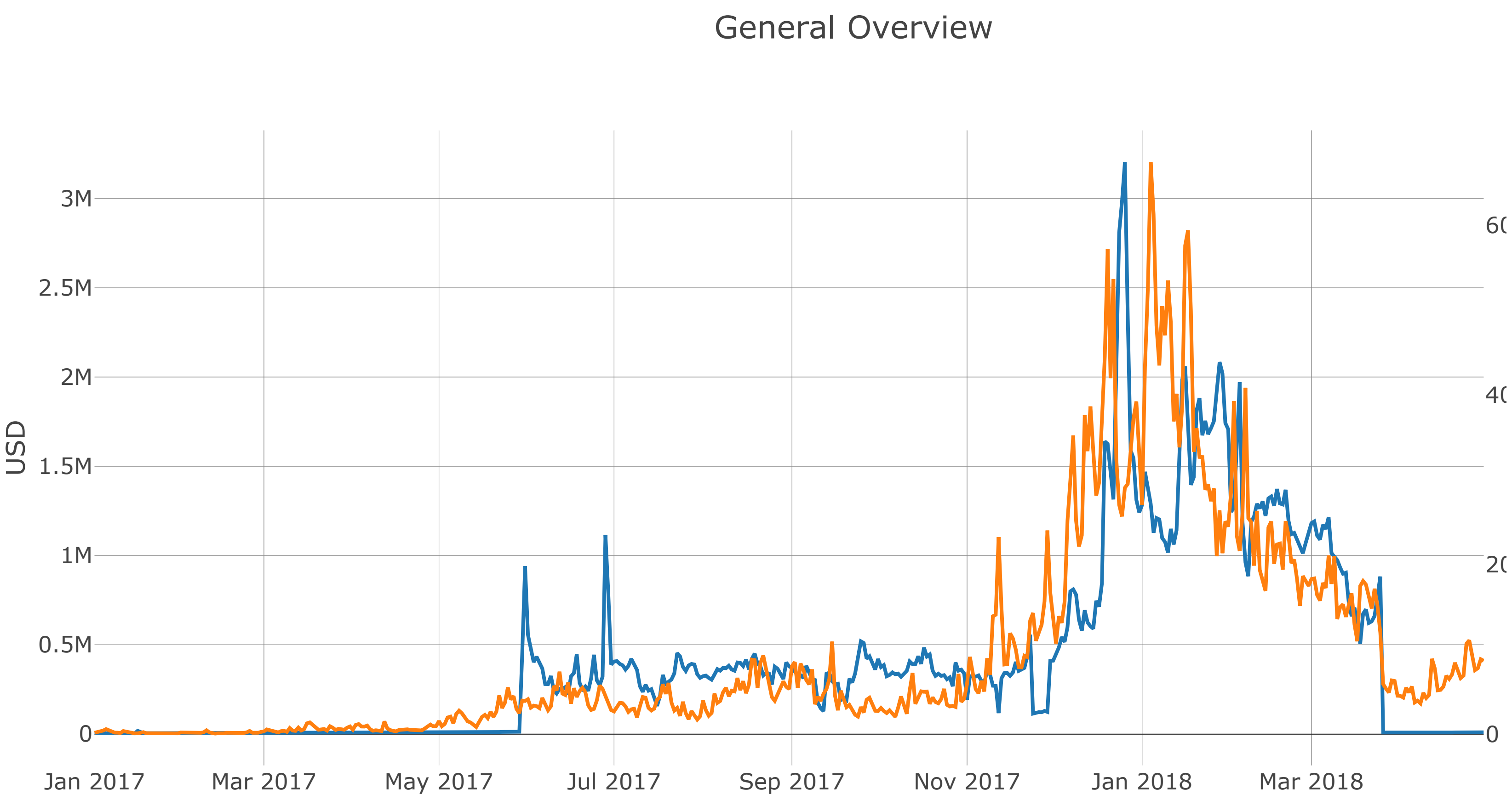
```
groupby = df.groupby('date', as_index=False).sum()
groupby.head()
```

Out[184]:

	date	ranknow	open	high	low	close	volume	hlc_average	ohlc_average
0	2017-01-01	417403	3616.32	3817.89	3533.44	3802.314385	193646542.0	3717.881462	3692.491096
1	2017-01-02	432118	3822.81	4023.32	3682.31	3799.611917	282900763.0	3835.080639	3832.012979
2	2017-01-03	439714	3795.94	3954.52	3728.24	3850.622806	266717601.0	3844.460935	3832.330701
3	2017-01-04	450766	3853.40	4412.57	3774.55	4308.159530	445046388.0	4165.093177	4087.169882
4	2017-01-05	444446	4310.55	4487.38	3377.70	3726.016149	620833004.0	3863.698716	3975.411537

In [146]:

```
trace0 = go.Scatter(
    x=groupby['date'], y=groupby['hlc_average'],
    name='HLC Average'
)

trace1 = go.Scatter(
    x=groupby['date'], y=groupby['volume'],
    name='Volume', yaxis='y2'
)

data = [trace0, trace1]
layout = go.Layout(
    title='General Overview',
    yaxis={
        'title': 'USD',
        'nticks': 10,
    },
    yaxis2={
        'title': 'Transactions',
        'nticks': 5,
        'showgrid': False,
        'overlaying': 'y',
        'side': 'right'
    }
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='time-series-overview')
```



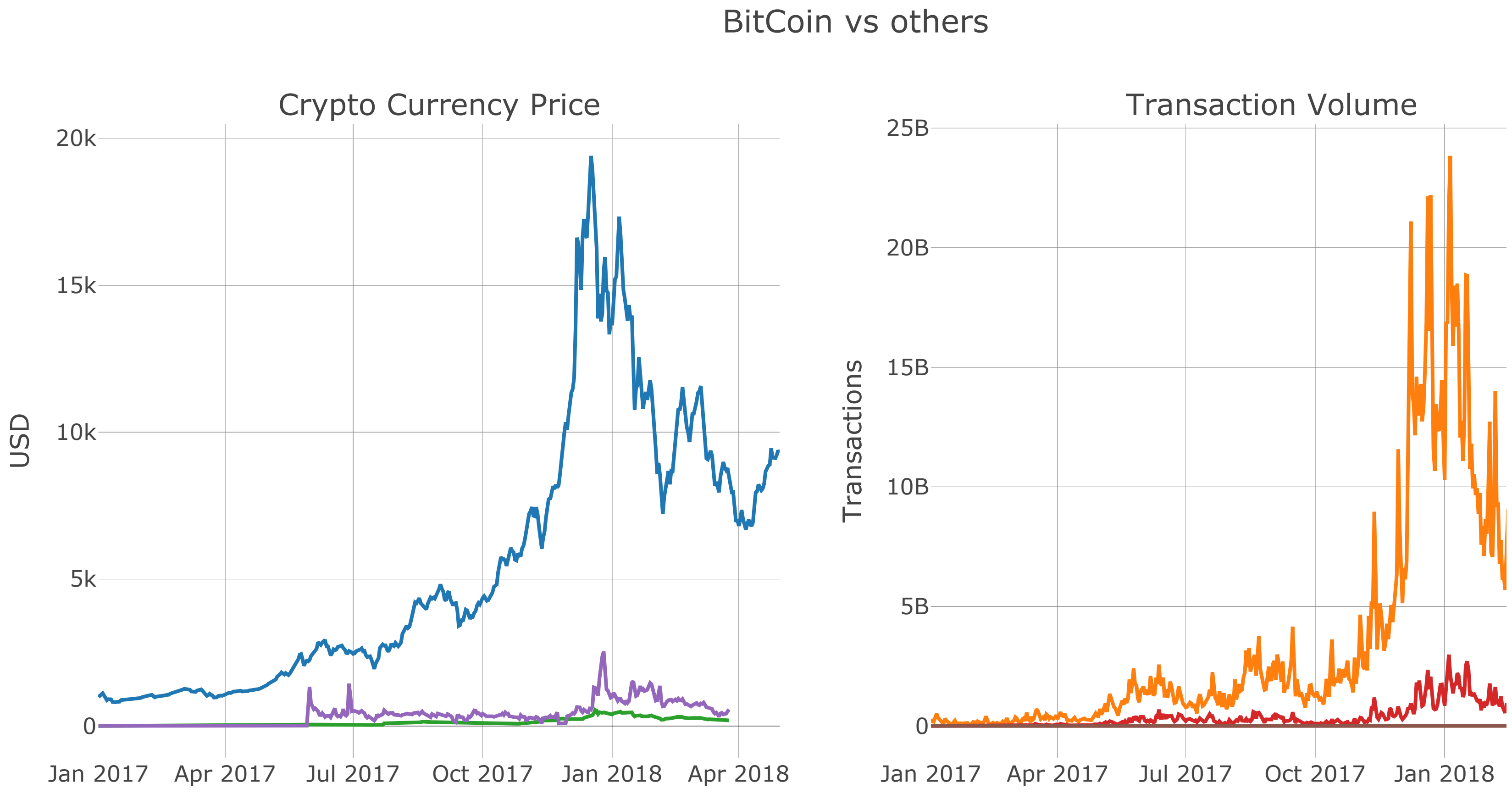
```
In [147]: df = df[df['date'] >= dt.date(2017, 1, 1)]
```

```
In [148]: bitcoin = df[df['ranknow'] == 1]

          others = df[(df['ranknow'] > 1) & (df['ranknow'] <= 10)]
          others = others.groupby('date', as_index=False).mean()

          minor = df[df['ranknow'] > 10]
          minor = minor.groupby('date', as_index=False).mean()
```

In [149]:

```
fig = tools.make_subplots(rows=1, cols=2, subplot_titles=(
    'Crypto Currency Price', 'Transaction Volume'
))

trace0 = go.Scatter(x=bitcoin['date'], y=bitcoin['hlc_average'], name='Bitcoin')
fig.append_trace(trace0, 1, 1)

trace1 = go.Scatter(x=bitcoin['date'], y=bitcoin['volume'], name='Bitcoin')
fig.append_trace(trace1, 1, 2)


trace2 = go.Scatter(x=others['date'], y=others['hlc_average'], name='Others')
fig.append_trace(trace2, 1, 1)

trace3 = go.Scatter(x=others['date'], y=others['volume'], name='Others')
fig.append_trace(trace3, 1, 2)

trace4 = go.Scatter(x=minor['date'], y=minor['hlc_average'], name='Minor ones')
fig.append_trace(trace4, 1, 1)

trace5 = go.Scatter(x=minor['date'], y=minor['volume'], name='Minor ones')
fig.append_trace(trace5, 1, 2)

fig['layout'].update(title='BitCoin vs others')
fig['layout'].update(showlegend=False)
fig['layout']['yaxis1'].update(title='USD')
fig['layout']['yaxis2'].update(title='Transactions')
fig['layout']['xaxis1'].update(nticks=6)
fig['layout']['xaxis2'].update(nticks=6)

py.iplot(fig, filename='bitcoin-vs-others')
```

```
This is the format of your plot grid:
[ (1,1) x1,y1 ]  [ (1,2) x2,y2 ]
```



```
In [150]: top9 = df[(df['ranknow'] >= 2) & (df['ranknow'] <= 10)]
          top9.name.unique()

Out[150]: array(['Ethereum', 'Ripple', 'Bitcoin Cash', 'Litecoin', 'EOS', 'Cardano',
                 'Stellar', 'NEO', 'IOTA'], dtype=object)
```

In [170]:

```
currency = df[df['name'] == 'Bitcoin'].copy()
currency.tail()
```

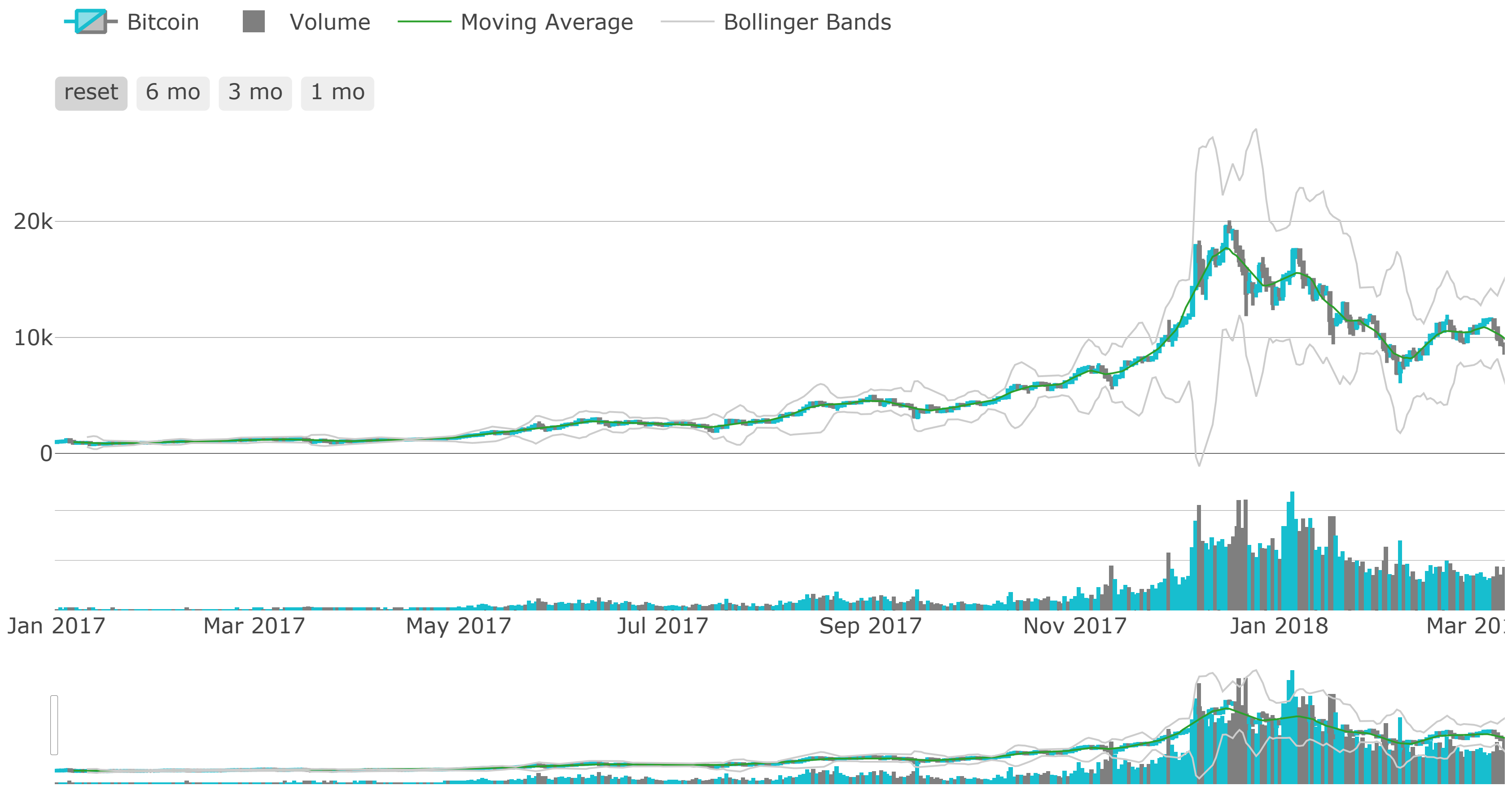
Out[170]:

	slug	name	date	ranknow	open	high	low	close	volume	hlc_average	ohlc_average
1824	bitcoin	Bitcoin	2018-04-26	1	8867.32	9281.51	8727.09	9281.51	8.970560e+09	9096.703333	9039.3575
1825	bitcoin	Bitcoin	2018-04-27	1	9290.63	9375.47	8987.05	8987.05	7.566290e+09	9116.523333	9160.0500
1826	bitcoin	Bitcoin	2018-04-28	1	8939.27	9412.09	8931.99	9348.48	7.805480e+09	9230.853333	9157.9575
1827	bitcoin	Bitcoin	2018-04-29	1	9346.41	9531.49	9193.71	9419.08	8.853000e+09	9381.426667	9372.6725
1828	bitcoin	Bitcoin	2018-04-30	1	9426.11	9477.14	9166.81	9240.55	8.673920e+09	9294.833333	9327.6525

In [171]:

```
increasing_color = '#17BECF'
decreasing_color = '#7F7F7F'

data = []

layout = {
    'xaxis': {
        'rangeselector': {
            'visible': True
        }
    },
    # Adding a volume bar chart for candlesticks is a good practice usually
    'yaxis': {
        'domain': [0, 0.2],
        'showticklabels': False
    },
    'yaxis2': {
        'domain': [0.2, 0.8]
    },
    'legend': {
        'orientation': 'h',
        'y': 0.9,
        'yanchor': 'bottom'
    },
    'margin': {
        't': 40,
        'b': 40,
        'r': 40,
        'l': 40
    }
}

# Defining main chart
trace0 = go.Candlestick(
    x=currency['date'], open=currency['open'], high=currency['high'],
    low=currency['low'], close=currency['close'],
    yaxis='y2', name='Bitcoin',
    increasing=dict(line=dict(color=increasing_color)),
    decreasing=dict(line=dict(color=decreasing_color)),
)

data.append(trace0)
```

```
# Adding some range buttons to interact
rangeselector = {
    'visible': True,
    'x': 0,
    'y': 0.8,
    'buttons': [
        {'count': 1, 'label': 'reset', 'step': 'all'},
        {'count': 6, 'label': '6 mo', 'step': 'month', 'stepmode': 'backward'},
        {'count': 3, 'label': '3 mo', 'step': 'month', 'stepmode': 'backward'},
        {'count': 1, 'label': '1 mo', 'step': 'month', 'stepmode': 'backward'},
    ]
}

layout['xaxis'].update(rangeselector=rangeselector)

# Setting volume bar chart colors
colors = []
for i, _ in enumerate(currency['date']):
    if i != 0:
        if currency['close'].iloc[i] > currency['close'].iloc[i-1]:
            colors.append(increasing_color)
        else:
            colors.append(decreasing_color)
    else:
        colors.append(decreasing_color)

trace1 = go.Bar(
    x=currency['date'], y=currency['volume'],
    marker=dict(color=colors),
    yaxis='y', name='Volume'
)

data.append(trace1)

# Adding Moving Average
def moving_average(interval, window_size=10):
    window = np.ones(int(window_size)) / float(window_size)
    return np.convolve(interval, window, 'same')

trace2 = go.Scatter(
    x=currency['date'][5:-5], y=moving_average(currency['close'])[5:-5],
    yaxis='y2', name='Moving Average',
    line=dict(width=1)
)

data.append(trace2)

# Adding boilinger bands
def bollinger_bands(price, window_size=10, num_of_std=5):
    rolling_mean = price.rolling(10).mean()
    rolling_std = price.rolling(10).std()
    upper_band = rolling_mean + (rolling_std * 5)
    lower_band = rolling_mean - (rolling_std * 5)
    return upper_band, lower_band

bb_upper, bb_lower = bollinger_bands(currency['close'])

trace3 = go.Scatter(
    x=currency['date'], y=bb_upper,
    yaxis='y2', line=dict(width=1),
    marker=dict(color='#ccc'), hoverinfo='none',
    name='Bollinger Bands',
    legendgroup='Bollinger Bands'
)
data.append(trace3)

trace4 = go.Scatter(
    x=currency['date'], y=bb_lower,
    yaxis='y2', line=dict(width=1),
    marker=dict(color='#ccc'), hoverinfo='none',
    name='Bollinger Bands', showlegend=False,
    legendgroup='Bollinger Bands'
)
data.append(trace4)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='Bitcoin-candlestick')
```



```
In [172]: currency['target'] = currency['close'].shift(-30)
```

In [173]:

```
X = currency.dropna().copy()
X['year'] = X['date'].apply(lambda x: x.year)
X['month'] = X['date'].apply(lambda x: x.month)
X['day'] = X['date'].apply(lambda x: x.day)
X = X.drop(['date', 'slug', 'name', 'ranknow', 'target'], axis=1)

y = currency.dropna()['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

X_train.shape, X_test.shape
```

Out[173]:

```
((364, 10), (91, 10))
```

In [174]:

```
forecast = currency[currency['target'].isnull()]
forecast = forecast.drop('target', axis=1)

X_forecast = forecast.copy()
X_forecast['year'] = X_forecast['date'].apply(lambda x: x.year)
X_forecast['month'] = X_forecast['date'].apply(lambda x: x.month)
X_forecast['day'] = X_forecast['date'].apply(lambda x: x.day)
X_forecast = X_forecast.drop(['date', 'slug', 'name', 'ranknow'], axis=1)
```

In [175]:

```
currency = currency.drop('target', axis=1)
```

In [176]:

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

In [178]:

```
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: 0.75
MAE: 1549.59
MSE: 4973478.57

Random Forest Regressor
R2: 0.98
MAE: 353.63
MSE: 346883.67

Gradient Boosting Regressor
R2: 0.97
MAE: 454.02
MSE: 577889.31
```

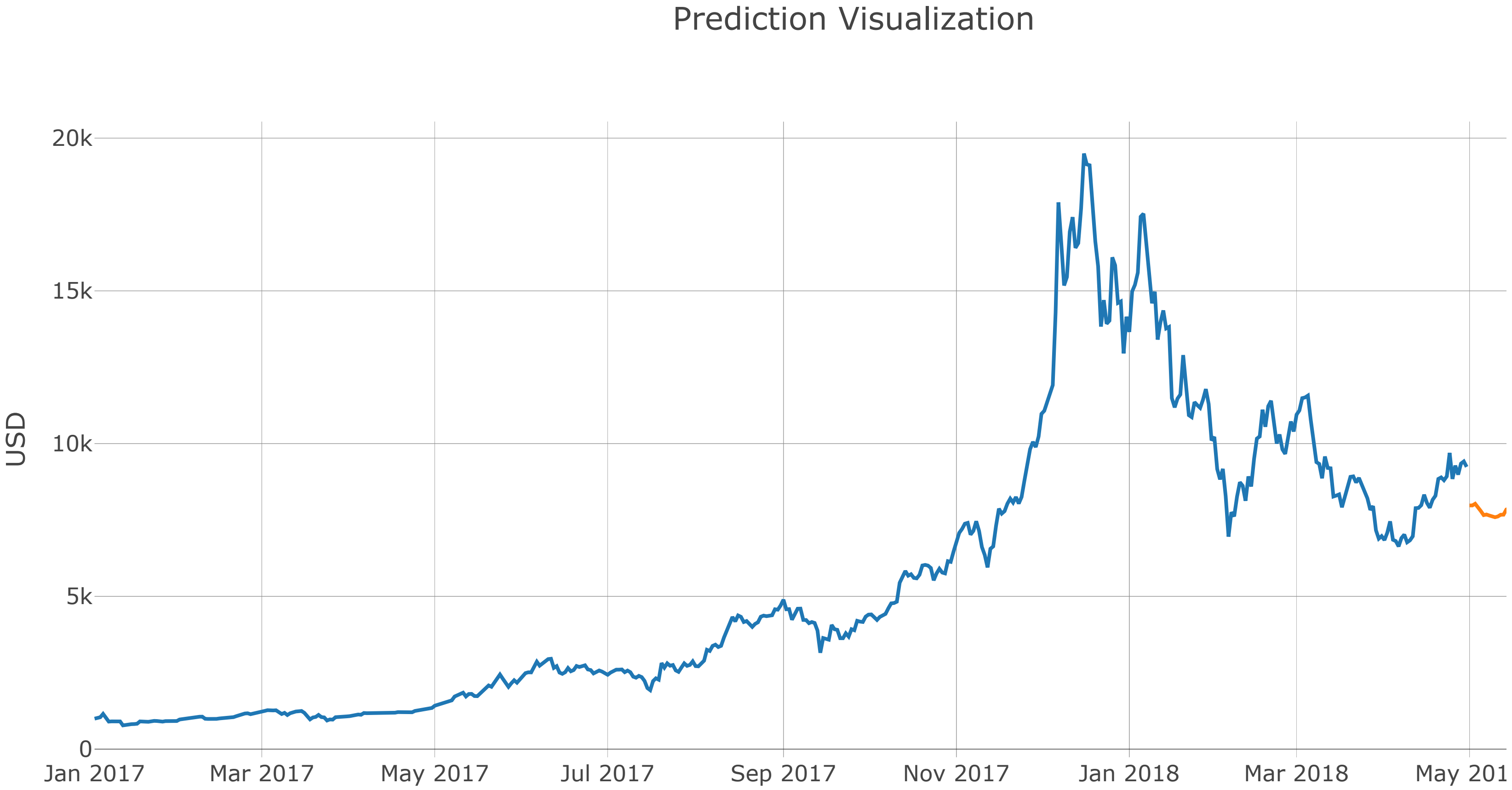
In [179]:

```
dtrain = xgb.DMatrix(X_train.values, y_train.values)
dtest = xgb.DMatrix(X_test.values)

param = {
    'max_depth': 10,
    'eta': 0.3
}
num_round = 20
bst = xgb.train(param, dtrain, num_round)
# make prediction
print('XGBoost')
print(f'R2: {r2_score(y_test, bst.predict(dtest)):.2f}')
print(f'MAE: {mean_absolute_error(y_test, bst.predict(dtest)):.2f}')
print(f'MSE: {mean_squared_error(y_test, bst.predict(dtest)):.2f}')

summary.append({
    'MSE': mean_squared_error(y_test, bst.predict(dtest)),
    'MAE': mean_absolute_error(y_test, bst.predict(dtest)),
    'R2': r2_score(y_test, bst.predict(dtest)),
    'name': 'XGBoost',
})
```

```
XGBoost
R2: 0.99
MAE: 306.62
MSE: 240639.56
```

In [183]:

```
clf = RandomForestRegressor(n_estimators=100, random_state=1)
clf.fit(X_train, y_train)
target = clf.predict(X_forecast)

final = pd.concat([currency, forecast])
final = final.groupby('date').mean()

day_one_forecast = currency.iloc[-1].date + dt.timedelta(days=1)
date = pd.date_range(day_one_forecast, periods=30, freq='D')
predictions = pd.DataFrame(target, columns=['target'], index=date)
final = final.append(predictions)
final.index.names = ['date']
final = final.reset_index()

trace0 = go.Scatter(
    x=final['date'], y=final['close'],
    name='Close'
)

trace1 = go.Scatter(
    x=final['date'], y=final['target'],
    name='Target'
)

data = [trace0, trace1]
layout = go.Layout(
    title='Prediction Visualization',
    yaxis={
        'title': 'USD',
        'nticks': 10,
    },
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='prediction-visualization')
```


Prediction Visualization
USD
20k
15k
10k
5k
0
Jan 2017
Mar 2017
May 2017
Jul 2017
Sep 2017
Nov 2017
Jan 2018
Mar 2018
May 201