In [2]:
```
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 [3]:
```
df = pd.read_csv('crypto-markets.csv')
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 748363 entries, 0 to 748362
Data columns (total 13 columns):
slug            748363 non-null object
symbol          748363 non-null object
name            748363 non-null object
date            748363 non-null object
ranknow         748363 non-null int64
open            748363 non-null float64
high            748363 non-null float64
low             748363 non-null float64
close           748363 non-null float64
volume          748363 non-null float64
market          748363 non-null float64
close_ratio     748363 non-null float64
spread          748363 non-null float64
dtypes: float64(8), int64(1), object(4)
memory usage: 74.2+ MB
```

In [5]:
```
df = df.drop(['symbol', 'market'], axis=1)
```

In [6]:
```
df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')
```

In [7]:
```
df['hlc_average'] = (df['high'] + df['low'] + df['close']) / 3
df['ohlc_average'] = (df['open'] + df['high'] + df['low'] + df['close']) / 4
```

In [8]:
```
df.head()
```

Out[8]:

	slug	name	date	ranknow	open	high	low	close	volume	close_ratio	spread	hlc_average	ohlc_average
0	bitcoin	Bitcoin	2013-04-28	1	135.30	135.98	132.10	134.21	0.0	0.5438	3.88	134.096667	134.3975
1	bitcoin	Bitcoin	2013-04-29	1	134.44	147.49	134.00	144.54	0.0	0.7813	13.49	142.010000	140.1175
2	bitcoin	Bitcoin	2013-04-30	1	144.00	146.93	134.05	139.00	0.0	0.3843	12.88	139.993333	140.9950
3	bitcoin	Bitcoin	2013-05-01	1	139.00	139.89	107.72	116.99	0.0	0.2882	32.17	121.533333	125.9000
4	bitcoin	Bitcoin	2013-05-02	1	116.38	125.60	92.28	105.21	0.0	0.3881	33.32	107.696667	109.8675

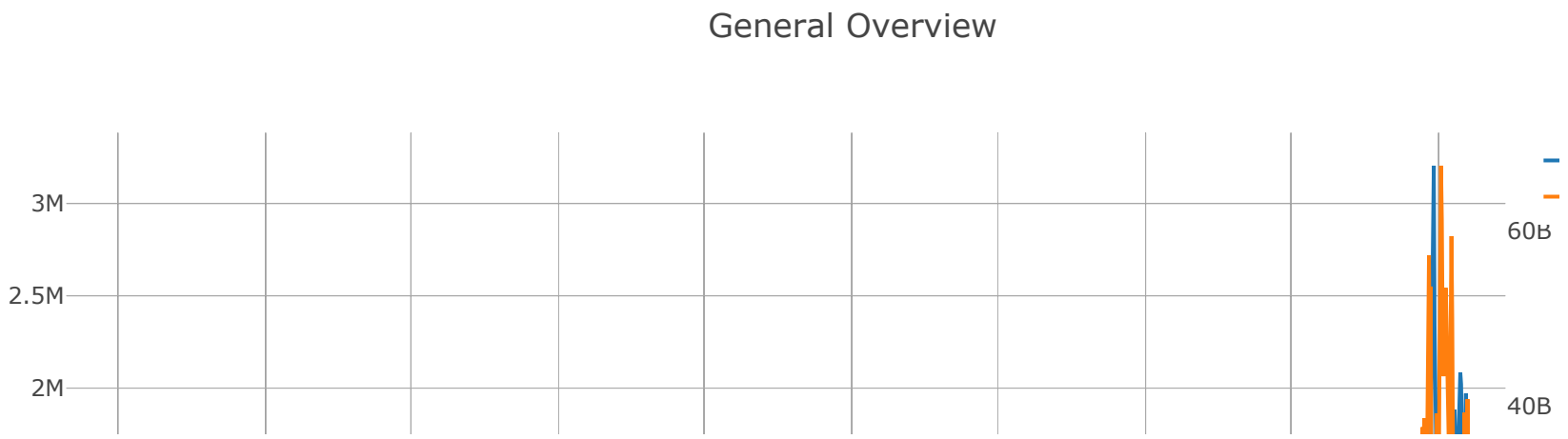
In [9]:
```
groupby = df.groupby('date', as_index=False).sum()
groupby.head()
```

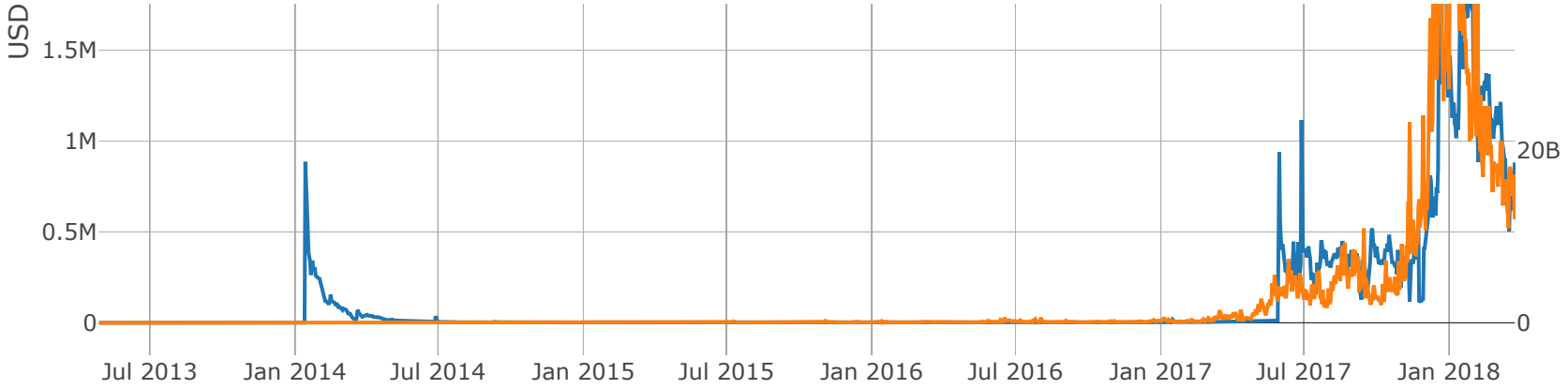
Out[9]:

	date	ranknow	open	high	low	close	volume	close_ratio	spread	hlc_average	ohlc_average
0	2013-04-28	1516	145.957751	146.808723	142.410696	144.953417	0.0	4.0624	4.40	144.724279	145.032647
1	2013-04-29	1516	145.196567	159.254506	144.480055	156.038552	0.0	4.6429	14.78	153.257704	151.242420
2	2013-04-30	1516	155.394106	159.058432	144.701084	150.159450	0.0	2.0204	14.35	151.306322	152.328268
3	2013-05-01	1516	150.110840	151.394123	116.585804	126.571464	0.0	1.6388	34.81	131.517130	136.165558
4	2013-05-02	1516	125.921471	135.907571	99.924710	113.755909	0.0	1.6845	35.98	116.529397	118.877415

In [10]:

```
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')
```





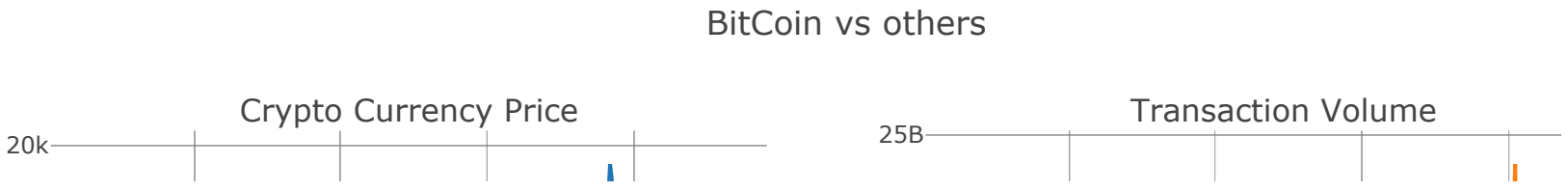
In [11]:
```
df = df[df['date'] >= dt.date(2017, 1, 1)]
```

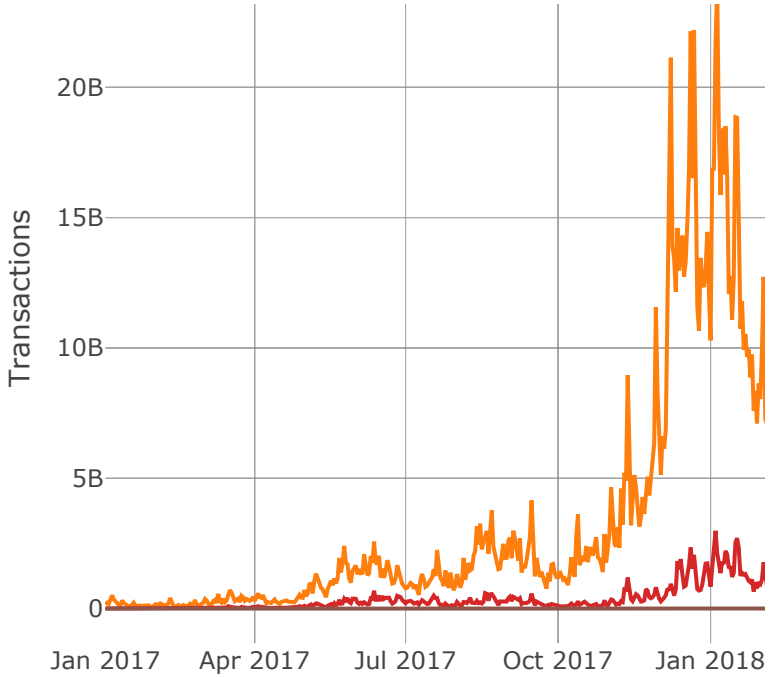
In [12]:
```
bitcoin = df[df['ranknow'] == 1]

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 [13]:

```
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 ]
```



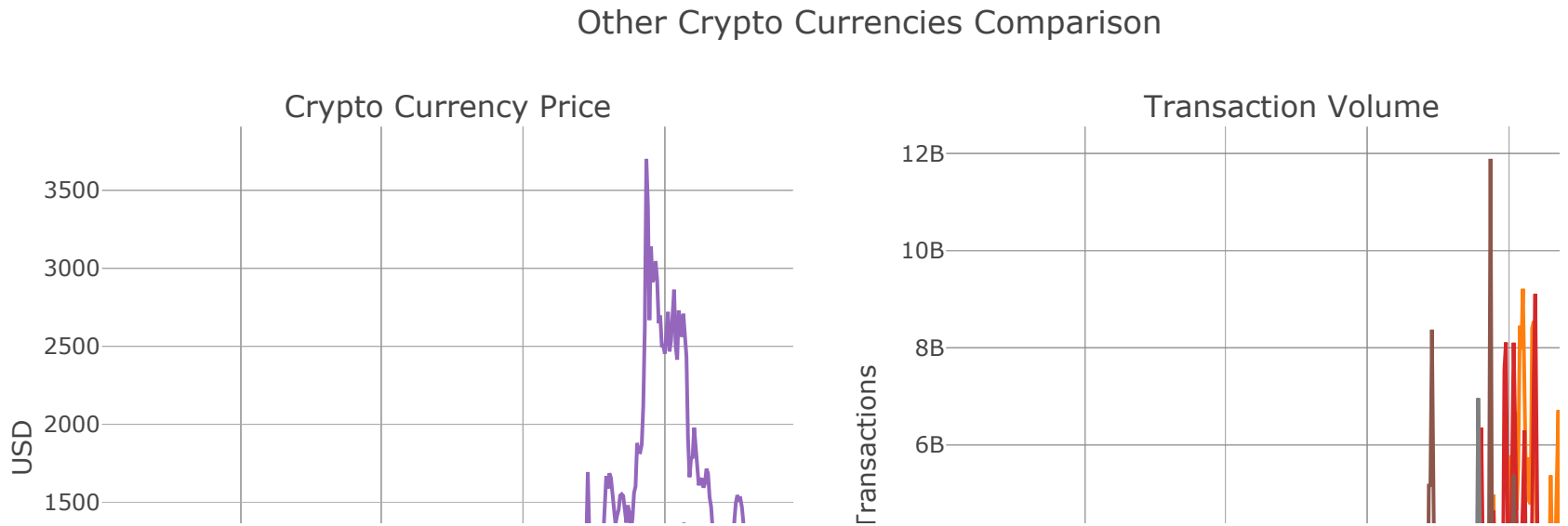


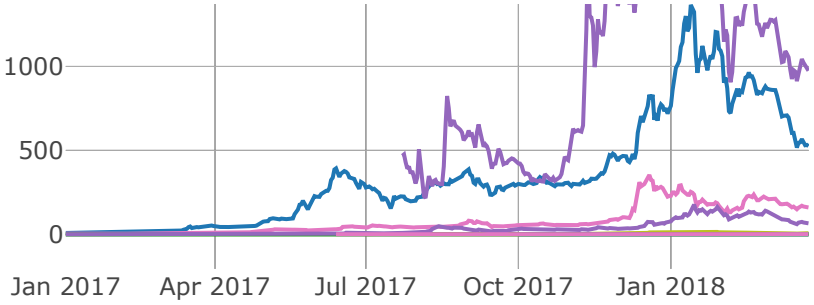
In [14]:
```
top9 = df[(df['ranknow'] >= 2) & (df['ranknow'] <= 10)]
top9.name.unique()
```

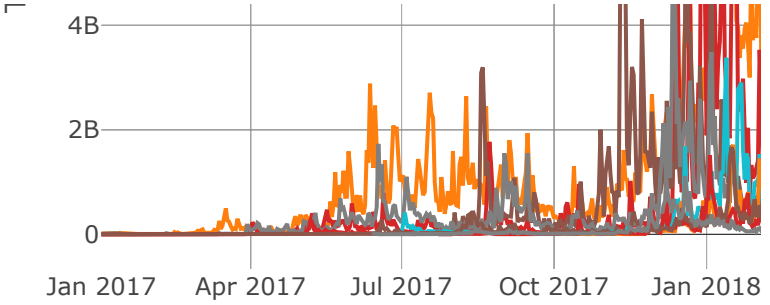
Out[14]:
```
array(['Ethereum', 'Ripple', 'Bitcoin Cash', 'Litecoin', 'EOS', 'Cardano',
       'Stellar', 'NEO', 'IOTA'], dtype=object)
```

In [15]:
```
fig = tools.make_subplots(rows=1, cols=2, subplot_titles=(
    'Crypto Currency Price', 'Transaction Volume'
))

for name in top9.name.unique():
    crypto = top9[top9['name'] == name]
    trace0 = go.Scatter(x=crypto['date'], y=crypto['hlc_average'], name=name)
    fig.append_trace(trace0, 1, 1)

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

fig['layout'].update(title='Other Crypto Currencies Comparison')
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='other-crypto-currencies-comparison')
```

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







In [16]:
```
summary = top9.groupby('name', as_index=False).mean()
summary.sort_values('close', ascending=True)
```

Out[16]:

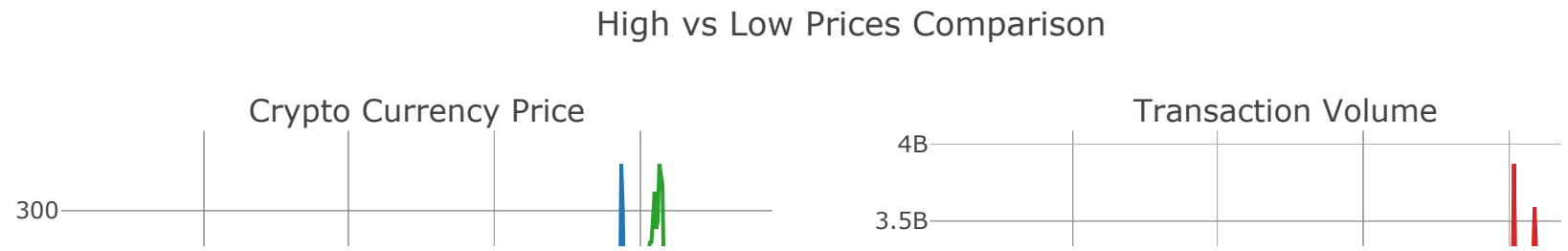
	name	ranknow	open	high	low	close	volume	close_ratio	spread	hlc_average	ohlc_aver
8	Stellar	8	0.108276	0.117521	0.098131	0.108728	6.592691e+07	0.506016	0.018775	0.108127	0.108
1	Cardano	7	0.288167	0.312111	0.260880	0.288909	2.295742e+08	0.509769	0.050739	0.287300	0.287
7	Ripple	3	0.398245	0.428918	0.367117	0.399760	5.835651e+08	0.495210	0.061715	0.398599	0.398
4	IOTA	10	1.384238	1.493196	1.257350	1.385273	1.044280e+08	0.544600	0.235839	1.378606	1.380
2	EOS	6	4.542335	4.886912	4.170982	4.559202	3.014290e+08	0.496236	0.716082	4.539032	4.539
6	NEO	9	33.668843	36.031880	30.854013	33.783129	1.033951e+08	0.538299	5.177906	33.556341	33.584
5	Litecoin	5	76.809800	81.161114	72.134833	77.101024	4.302964e+08	0.536351	9.026281	76.798990	76.801
3	Ethereum	2	349.163808	365.533140	330.962339	350.166147	1.251831e+09	0.555100	34.570802	348.887209	348.956
0	Bitcoin Cash	4	1106.709472	1194.996057	1021.766301	1107.688902	9.770481e+08	0.466754	173.229756	1108.150420	1107.790

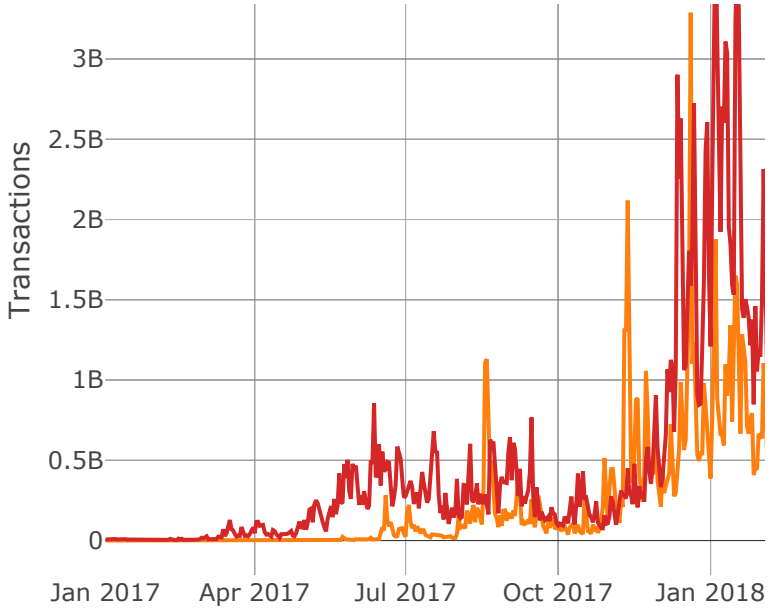
In [17]:
```
low_price = top9[top9['ranknow'].isin([4, 6, 7, 9])]
low_price = low_price.groupby('date', as_index=False).mean()

high_price = top9[top9['ranknow'].isin([2, 3, 5, 8, 10])]
high_price = high_price.groupby('date', as_index=False).mean()
```

In [18]:

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

trace0 = go.Scatter(x=low_price['date'], y=low_price['hlc_average'], name='Low Price')
fig.append_trace(trace0, 1, 1)

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

trace2 = go.Scatter(x=high_price['date'], y=high_price['hlc_average'], name='High Price')
fig.append_trace(trace2, 1, 1)

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

fig['data'][0].update(yaxis='y3')
fig['layout'].update(title='High vs Low Prices Comparison')
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)
fig['layout']['yaxis3'] = {
    'anchor': 'x1', 'domain': [0.0, 1.0], 'nticks': 6,
    'overlaying': 'y1', 'side': 'right', 'showgrid': False
}

py.iplot(fig, filename='high-vs-low-prices-comparison')
```

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





In [73]:
```
currency = df[df['name'] == 'Bitcoin'].copy()
currency.tail()
```

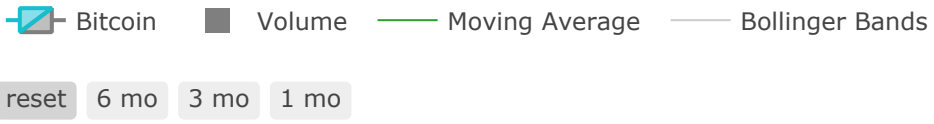
Out[73]:

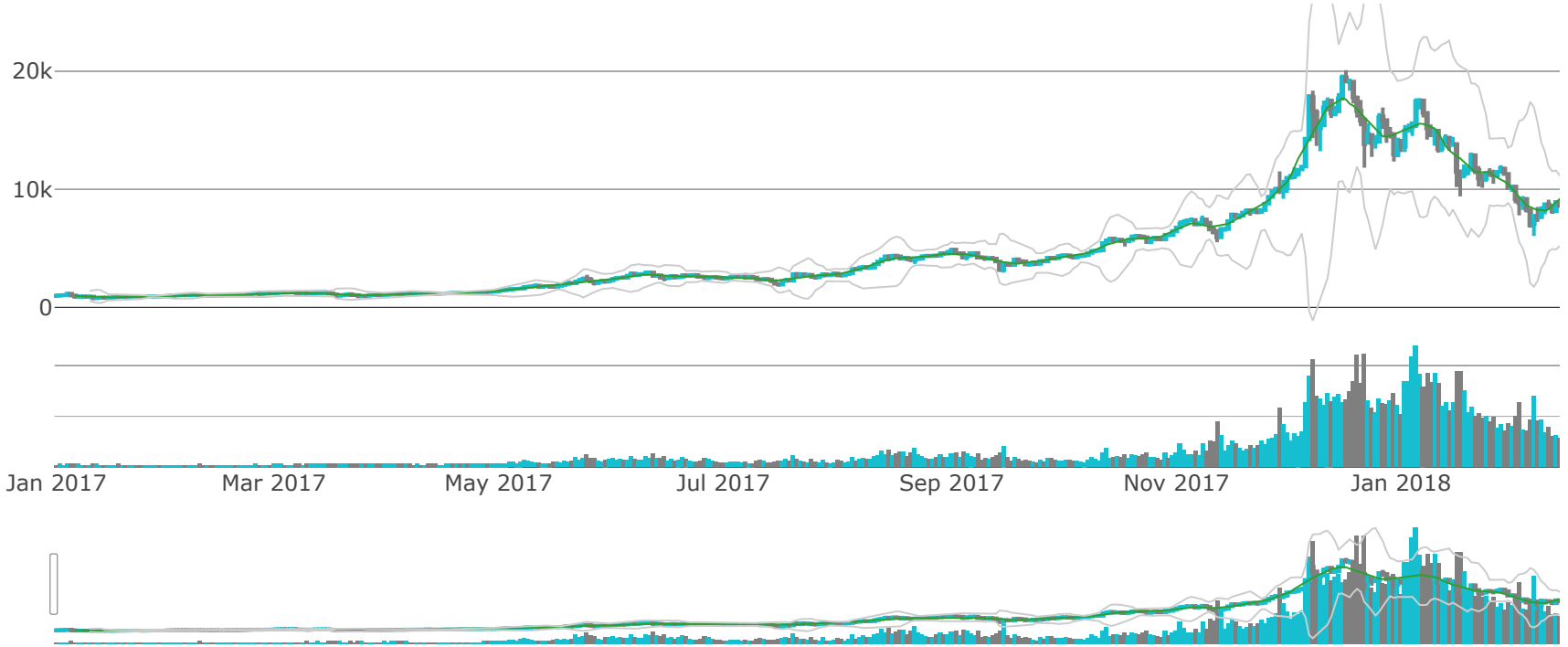
	slug	name	date	ranknow	open	high	low	close	volume	close_ratio	spread	hlc_average	ohlc_average
1788	bitcoin	Bitcoin	2018-03-21	1	8937.48	9177.37	8846.33	8929.28	6.043130e+09	0.2506	331.04	8984.326667	8972.6150
1789	bitcoin	Bitcoin	2018-03-22	1	8939.44	9100.71	8564.90	8728.47	5.530390e+09	0.3053	535.81	8798.026667	8833.3800
1790	bitcoin	Bitcoin	2018-03-23	1	8736.25	8879.62	8360.62	8879.62	5.954120e+09	1.0000	519.00	8706.620000	8714.0275
1791	bitcoin	Bitcoin	2018-03-24	1	8901.95	8996.18	8665.70	8668.12	5.664600e+09	0.0073	330.48	8776.666667	8807.9875
1792	bitcoin	Bitcoin	2018-03-25	1	8612.81	8682.01	8449.10	8495.78	4.569880e+09	0.2004	232.91	8542.296667	8559.9250

In [63]:
```
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 [64]:
```
currency['target'] = currency['close'].shift(-30)
```

In [65]:
```
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[65]:
```
((335, 12), (84, 12))
```

In [66]:
```
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 [67]:
```
currency = currency.drop('target', axis=1)
```

In [68]:
```
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,
    })
```

```
LinearRegression
R2: 0.72
MAE: 1619.68
MSE: 5460702.93

Random Forest Regressor
R2: 0.97
MAE: 381.27
MSE: 518337.29

Gradient Boosting Regressor
R2: 0.97
MAE: 440.16
MSE: 536444.53
```

In [69]:

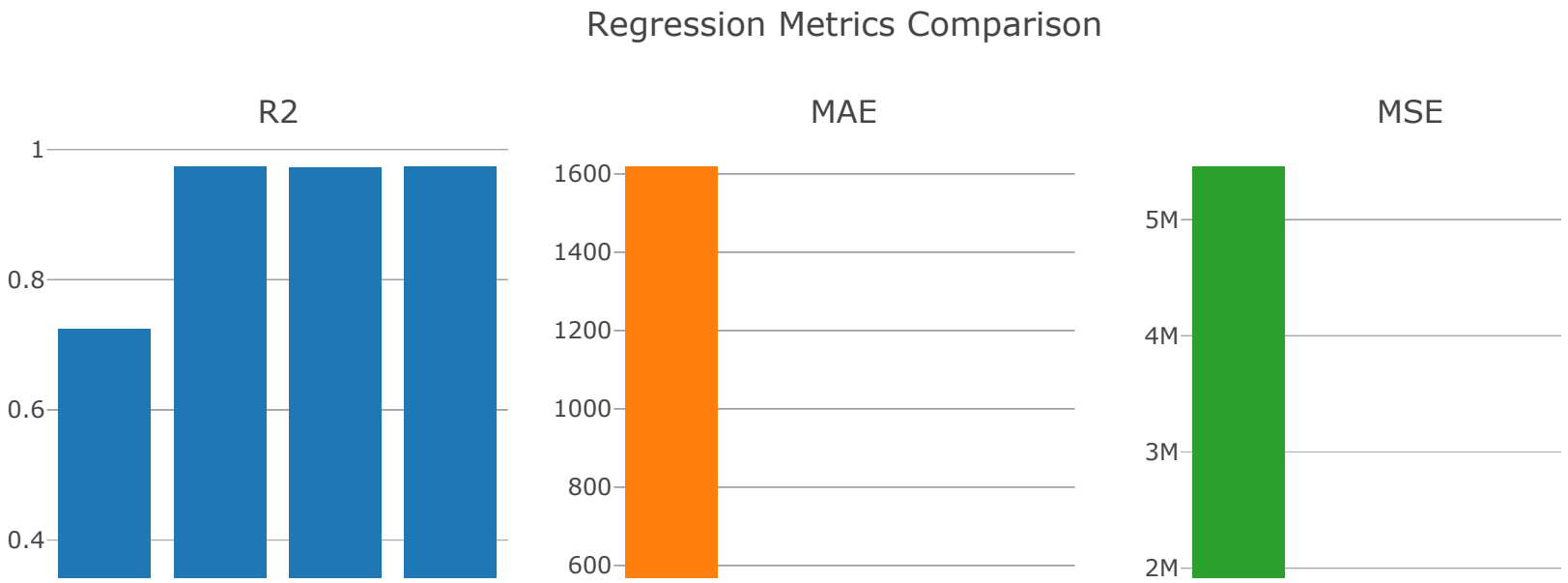
```
dtrain = xgb.DMatrix(X_train.values, y_train.values)
dtest = xgb.DMatrix(X_test.values)

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',
})
```

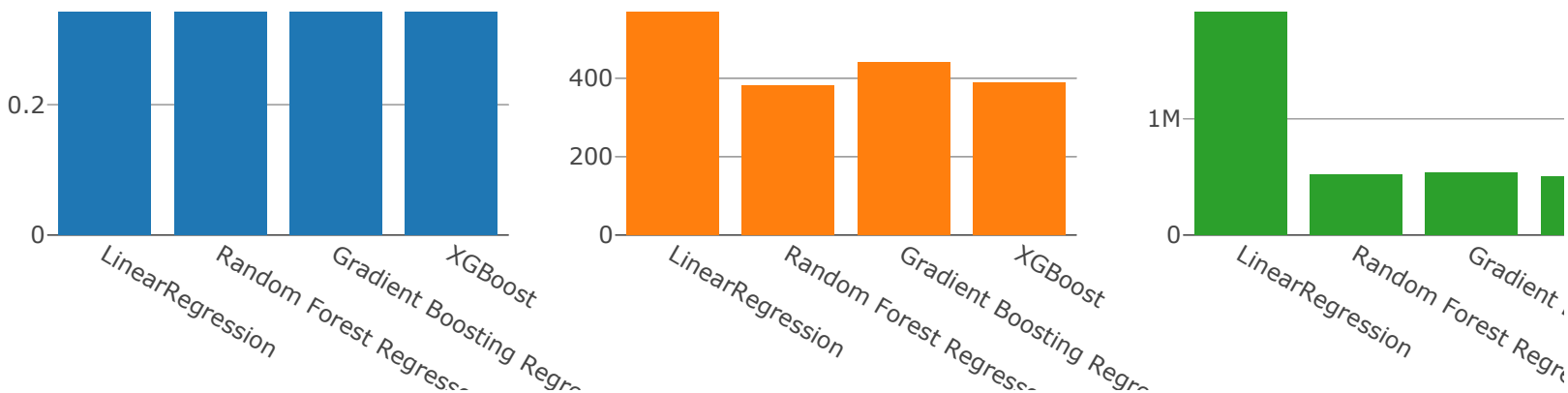
```
XGBoost
R2: 0.97
MAE: 388.64
MSE: 507747.86
```

In [70]:

```
summary = pd.DataFrame(summary)

fig = tools.make_subplots(rows=1, cols=3, subplot_titles=(
    'R2', 'MAE', 'MSE'
))

trace0 = go.Bar(x=summary['name'], y=summary['R2'], name='R2')
fig.append_trace(trace0, 1, 1)

trace1 = go.Bar(x=summary['name'], y=summary['MAE'], name='MAE')
fig.append_trace(trace1, 1, 2)

trace2 = go.Bar(x=summary['name'], y=summary['MSE'], name='MSE')
fig.append_trace(trace2, 1, 3)

fig['layout'].update(title='Regression Metrics Comparison')
fig['layout'].update(showlegend=False)

py.iplot(fig, filename='regression-metrics-comparison')
```

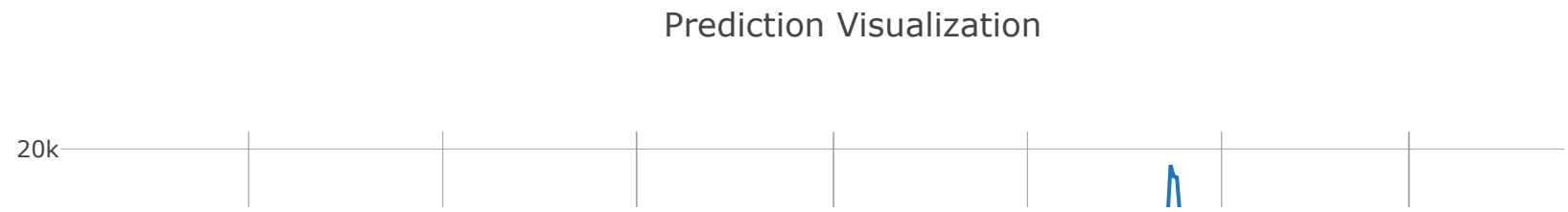
```
This is the format of your plot grid:
[ (1,1) x1,y1 ]  [ (1,2) x2,y2 ]  [ (1,3) x3,y3 ]
```




0.2
0
LinearRegression
Random Forest Regress
Gradient Boosting Regr
XGBoost
400
200
0
LinearRegression
Random Forest Regress
Gradient Boosting Regr
XGBoost
1M
0
LinearRegression
Random Forest Regr
Gradient

In [72]:

```
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')
```




USD
15k
10k
5k
0
Jan 2017
Mar 2017
May 2017
Jul 2017
Sep 2017
Nov 2017
Jan 2018
Mar 2018

In [ ]: