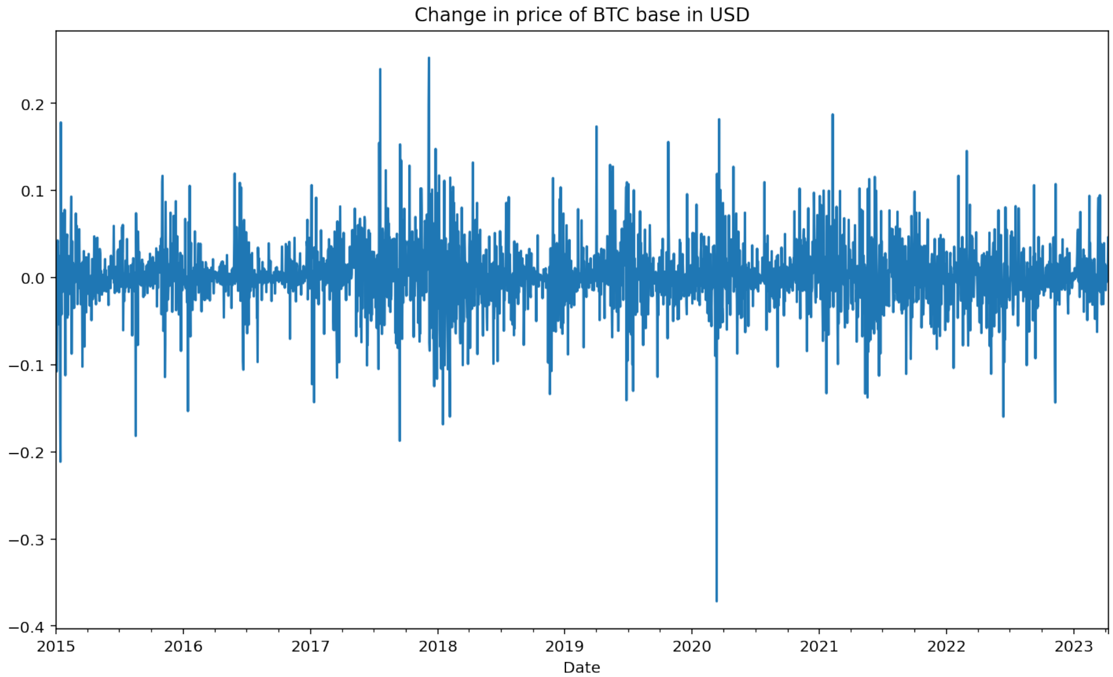
PDFformat

April 12, 2023

[58]:
```
# Import Bitcoin based in USD
tickers = 'BTC-USD '
matana = (
    yf.download(tickers=tickers, progress=False)
    .assign(Date=lambda x: x.index.tz_localize(None))
    .set_index('Date')
    .rename_axis(columns=['Ticker'])
)
returns_1 = matana['Adj Close'].pct_change().loc['2015':]
matana['return']=returns_1
# Plot the return
returns_1.plot(figsize=(12,7))
plt.title('Change in price of BTC base in USD')

ff = (
    pdr.DataReader(
        name='F-F_Research_Data_Factors_daily',
        data_source='famafrench',
        start='1900',
        session=session
    )
)
```



[60]:
```
# The Fama-French Three-Factor Model
brk = (
    yf.download(tickers='BTC-USD', progress=False)
    .assign(
        Date=lambda x: x.index.tz_localize(None),
        Ri=lambda x: x['Adj Close'].pct_change().mul(100)
    )
    .set_index('Date')
    .join(ff[0])
    .assign(RiRF = lambda x: x['Ri'] - x['RF'])
    .rename(columns={'Mkt-RF': 'MktRF'})
    .rename_axis(columns='Variable')
)

model = smf.ols(formula='RiRF ~ MktRF + SMB + HML', data=brk.iloc[:756])
fit = model.fit()
summary = fit.summary()
summary
```

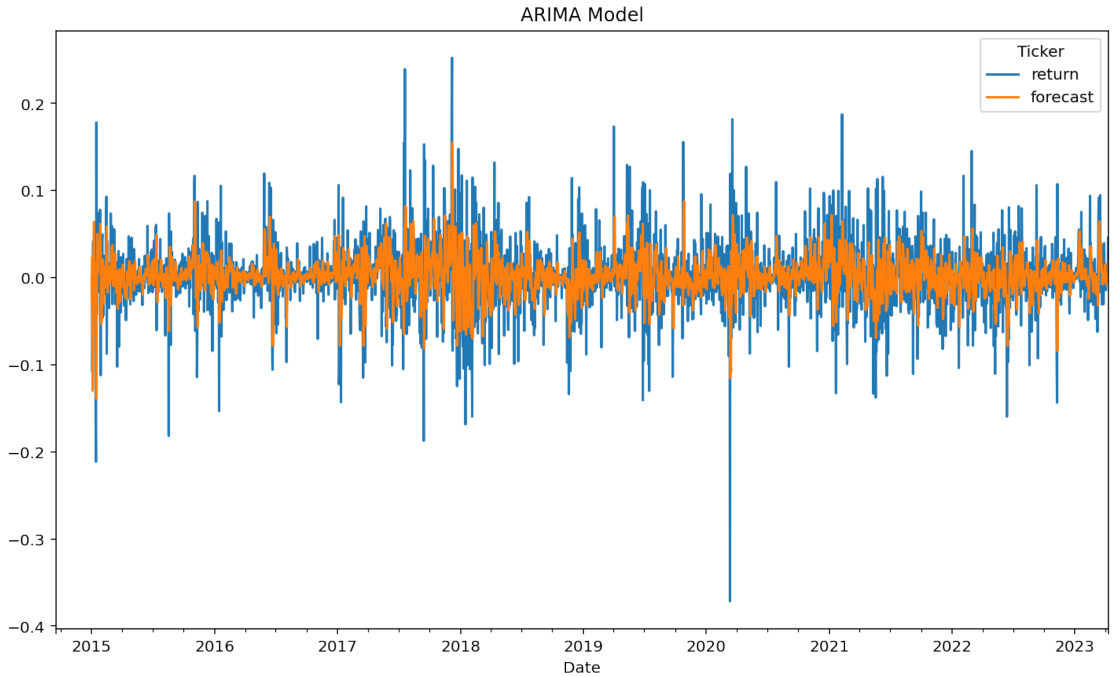
[60]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                   RiRF   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.005
```

```
Method:                 Least Squares   F-statistic:                   0.08239
Date:                Tue, 11 Apr 2023   Prob (F-statistic):              0.970
Time:                        17:11:32   Log-Likelihood:                -1372.4
No. Observations:                 521   AIC:                             2753.
Df Residuals:                     517   BIC:                             2770.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.1010      0.148      0.681      0.496      -0.190       0.393
MktRF         -0.0718      0.158     -0.454      0.650      -0.383       0.239
SMB            0.0727      0.291      0.250      0.803      -0.499       0.644
HML            0.0225      0.290      0.078      0.938      -0.547       0.592
==============================================================================
Omnibus:                      136.317   Durbin-Watson:                   1.915
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1791.164
Skew:                          -0.738   Prob(JB):                         0.00
Kurtosis:                      11.963   Cond. No.                         2.15
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

[61]:
```python
# 1. Model ARIMA
# Order of differencing
result = adfuller(matana['Adj Close'].dropna())
print('ADF Statistic:', result[0])
print('p-value:', result[1])
diff = matana['Adj Close'].diff().dropna()
#plot_acf(diff)
#plot_pacf(diff)
#plt.show()
# Apply model: one autoregressive, one differencing, and one moving average
model = sm.tsa.arima.ARIMA(returns_1, order=(2,2,1))
results = model.fit()
print(results.summary())
matana['forecast']=results.predict()
matana[['return','forecast']].plot(figsize=(12,7))
plt.title('ARIMA Model')
warnings.filterwarnings('ignore')
```

```
ADF Statistic: -1.5051287573506575
p-value: 0.53097270747215
                               SARIMAX Results
```

```
==============================================================================
Dep. Variable:              Adj Close   No. Observations:                 3023
Model:                 ARIMA(2, 2, 1)   Log Likelihood                5149.721
Date:                Tue, 11 Apr 2023   AIC                         -10291.442
Time:                        17:14:03   BIC                         -10267.388
Sample:                    01-01-2015   HQIC                        -10282.793
                         - 04-11-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.6908      0.011    -60.387      0.000      -0.713      -0.668
ar.L2         -0.3459      0.012    -28.661      0.000      -0.370      -0.322
ma.L1         -0.9999      0.098    -10.230      0.000      -1.191      -0.808
sigma2         0.0019      0.000     10.430      0.000       0.002       0.002
================================================================================
===
Ljung-Box (L1) (Q):                  22.83   Jarque-Bera (JB):
3465.30
Prob(Q):                              0.00   Prob(JB):
0.00
Heteroskedasticity (H):               0.98   Skew:
0.15
Prob(H) (two-sided):                  0.76   Kurtosis:
8.24
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```



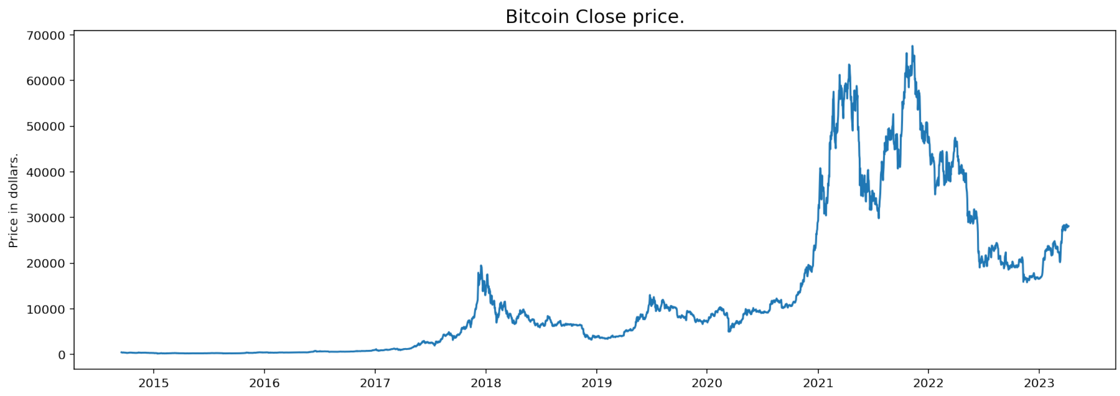
[7]:
```python
# 2. Garch Model
garch11_bitcoin = arch_model(returns_1.dropna()*100, p=1, q=1)
res_bitcoin = garch11_bitcoin.fit(update_freq=10)
print(res_bitcoin.summary())
```

```
Optimization terminated successfully    (Exit mode 0)
            Current function value: 8074.491992409081
            Iterations: 9
            Function evaluations: 60
            Gradient evaluations: 9
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:              Adj Close   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -8074.49
Distribution:                  Normal   AIC:                           16157.0
Method:            Maximum Likelihood   BIC:                           16181.0
                                        No. Observations:                 3018
Date:                Thu, Apr 06 2023   Df Residuals:                     3017
Time:                        21:09:15   Df Model:                            1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|   95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.2247  5.727e-02      3.923  8.731e-05 [  0.112,  0.337]
```

```
                      Volatility Model
==========================================================================
                 coef    std err          t      P>|t|     95.0% Conf. Int.
--------------------------------------------------------------------------
omega          0.6647      0.237      2.806  5.022e-03   [  0.200,  1.129]
alpha[1]       0.1189  2.926e-02      4.063  4.835e-05 [6.154e-02,  0.176]
beta[1]        0.8435  2.936e-02     28.732 1.500e-181   [  0.786,  0.901]
==========================================================================

Covariance estimator: robust
```

[9]:

```python
# 3. Machine Learning model (New)
# Close data
plt.figure(figsize=(15, 5))
plt.plot(matana['Close'])
plt.title('Bitcoin Close price.', fontsize=15)
plt.ylabel('Price in dollars.')
plt.show()
```

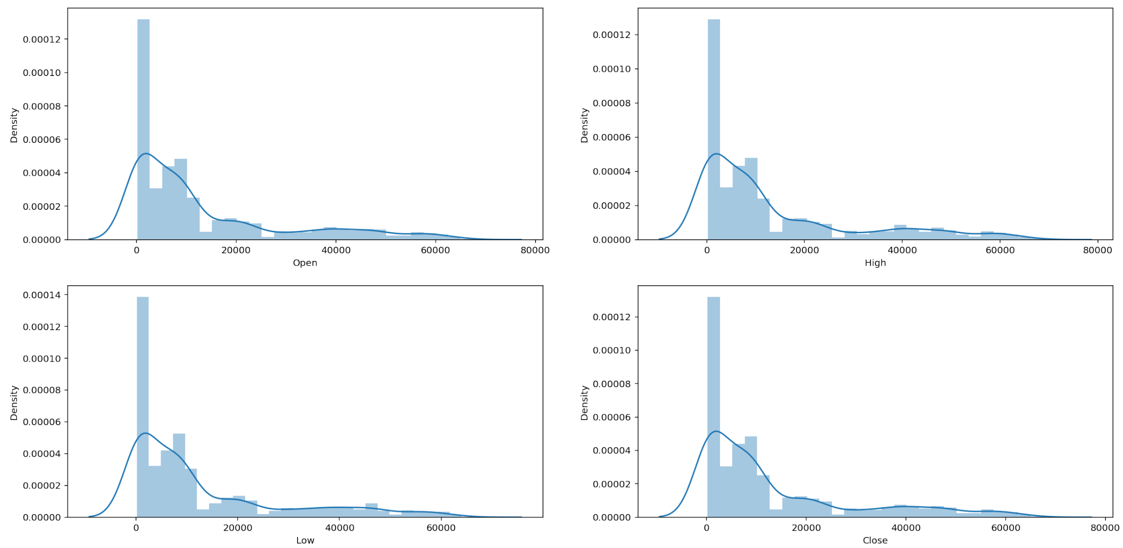


[16]:

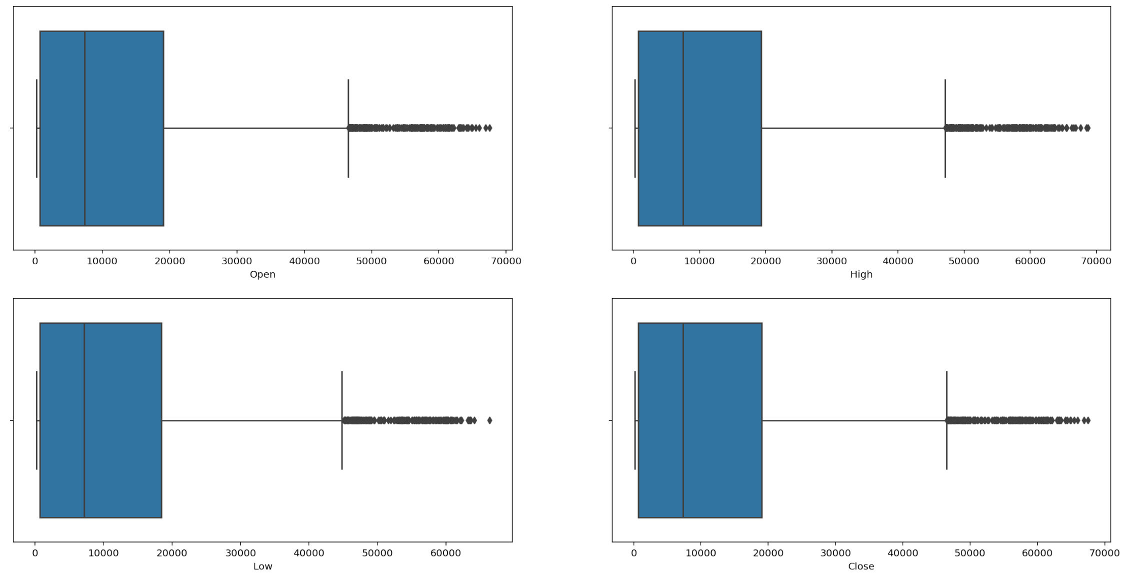
```python
# Distribution plot of the OHLC data
features = ['Open', 'High', 'Low', 'Close']
print('Distribution plot of the OHLC data')

plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(2,2,i+1)
    sb.distplot(matana[col])
plt.show()

# Boxplot of the OHLC data
print('Boxplot of the OHLC data')
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
```

```
    plt.subplot(2,2,i+1)
    sb.boxplot(matana[col])
plt.show()
```

Distribution plot of the OHLC data

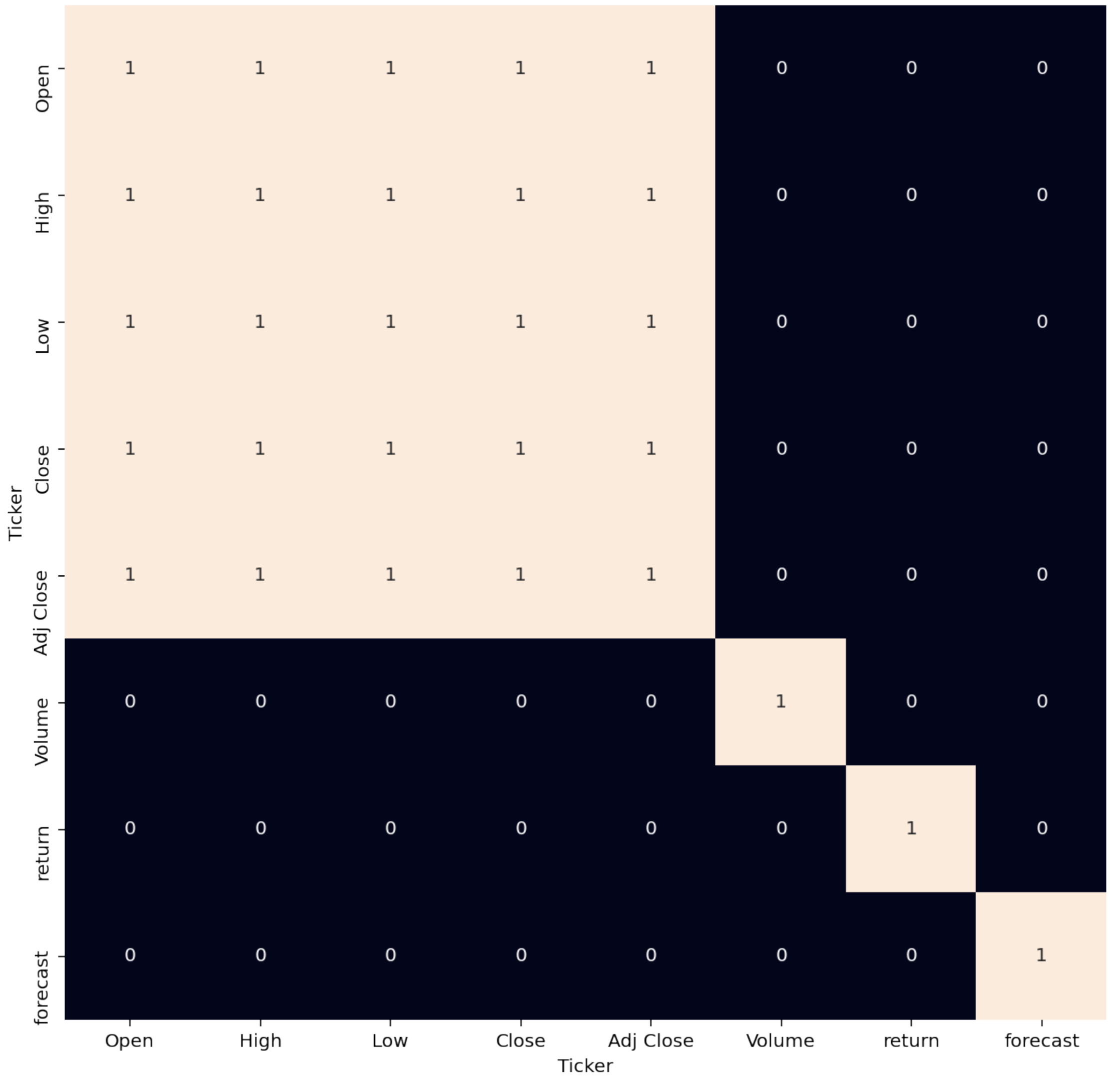


Boxplot of the OHLC data



[17]:
```
# Heat map
plt.figure(figsize=(10, 10))
```

```
# As our concern is with the highly correlated features only so
sb.heatmap(matana.corr() > 0.9, annot=True, cbar=False)
plt.show()
```



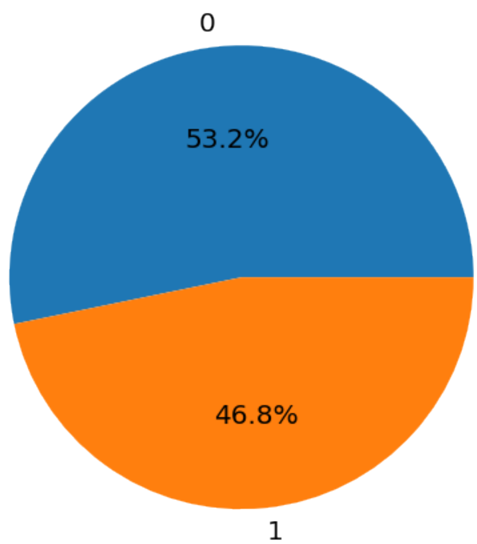
[44]:

```
btc = pd.read_csv('BTC.csv')
```

[50]:

```
splitted = btc['Date'].str.split('/', expand=True)
# split data
btc['year'] = splitted[0].astype('int')
btc['month'] = splitted[1].astype('int')
btc['day'] = splitted[2].astype('int')

# Prepare the training of our model
```

```
btc['is_quarter_end'] = np.where(btc['month']%3==0,1,0)
btc['open-close']  = btc['Open'] - btc['Close']
btc['low-high']  = btc['Low'] - btc['High']
# target is a signal whether to buy or not
btc['target'] = np.where(btc['Close'].shift(-1) > btc['Close'], 1, 0)

# Check No correlated features
plt.pie(btc['target'].value_counts().values,
        labels=[0, 1], autopct='%1.1f%%')
plt.show()
```



[53]:
```
# Training Size
features = btc[['open-close', 'low-high', 'is_quarter_end']]
target = btc['target']

scaler = StandardScaler()
features = scaler.fit_transform(features)

X_train, X_valid, Y_train, Y_valid = train_test_split(
    features, target, test_size=0.1, random_state=2022)
print(X_train.shape, X_valid.shape)
```

```
(2813, 3) (313, 3)
```

[54]:
```
# Apply the model with LogisticRegression, SVC, XGBClassifier
# Performance of different state-of-the-art models.
```

```
models = [LogisticRegression(), SVC(kernel='poly', probability=True),␣
↪XGBClassifier()]

for i in range(3):
    models[i].fit(X_train, Y_train)
    print(f'{models[i]} : ')
    print('Training Accuracy : ', metrics.roc_auc_score(Y_train, models[i].
↪predict_proba(X_train)[:,1]))
    print('Validation Accuracy : ', metrics.roc_auc_score(Y_valid, models[i].
↪predict_proba(X_valid)[:,1]))
    print('\n')
```

```
LogisticRegression() :
Training Accuracy :  0.5296558035487315
Validation Accuracy :  0.48965742784727334


SVC(kernel='poly', probability=True) :
Training Accuracy :  0.4626005389191113
Validation Accuracy :  0.5178235630774262


XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, …) :
Training Accuracy :  0.946875031775891
Validation Accuracy :  0.4970975390401439
```

[ ]: