In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#import data
tesla_data = pd.read_csv(r"C:\Users\dekea\Downloads\archive (13)\tsla_2014_2023.csv")
```

In [3]:
```
tesla_data.head()
```

Out[3]:

	date	open	high	low	close	volume	rsi_7	rsi_14	cci_7	cci_14	sma_50	ema_50	sma_100	ema
0	2014-01-02	9.986667	10.165333	9.770000	10.006667	92826000	55.344071	54.440118	-37.373644	15.213422	9.682107	9.820167	10.494240	9.67
1	2014-01-03	10.000000	10.146000	9.906667	9.970667	70425000	53.742629	53.821521	-81.304471	17.481130	9.652800	9.826069	10.495693	9.68
2	2014-01-06	10.000000	10.026667	9.682667	9.800000	80416500	46.328174	50.870410	-123.427544	-37.824708	9.629467	9.825047	10.496740	9.68
3	2014-01-07	9.841333	10.026667	9.683333	9.957333	75511500	53.263037	53.406750	-84.784651	-20.779431	9.597747	9.830235	10.503407	9.68
4	2014-01-08	9.923333	10.246667	9.917333	10.085333	92448000	58.368660	55.423026	60.799662	43.570559	9.573240	9.840239	10.511147	9.69

In [4]:
```
# Check for missing values
print(tesla_data.isnull().sum())
```

```
date              0
open              0
high              0
low               0
close             0
volume            0
rsi_7             0
rsi_14            0
cci_7             0
cci_14            0
sma_50            0
ema_50            0
sma_100           0
ema_100           0
macd              0
bollinger         0
TrueRange         0
atr_7             0
atr_14            0
next_day_close    0
dtype: int64
```

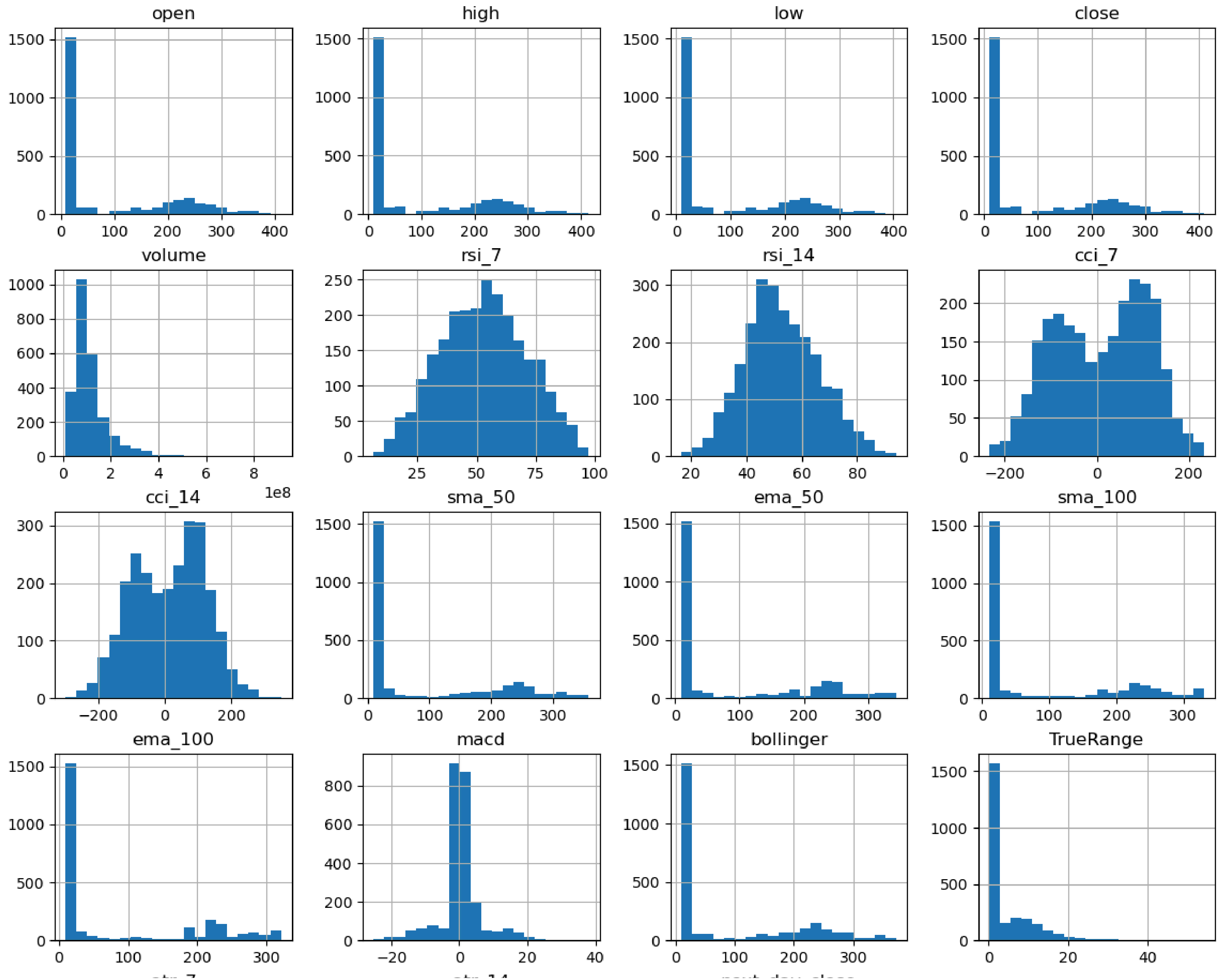
In [5]:
```
tesla_data.describe()
```

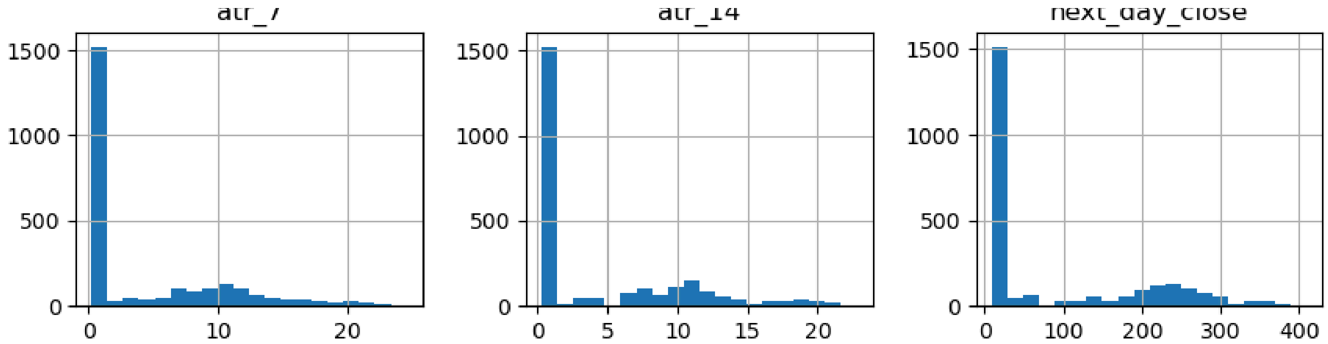
Out[5]:

	open	high	low	close	volume	rsi_7	rsi_14	cci_7	cci_14	sma_50	
count	2516.000000	2516.000000	2516.000000	2516.000000	2.516000e+03	2516.000000	2516.000000	2516.000000	2516.000000	2516.000000	2516
mean	94.098510	96.172733	91.865096	94.072491	1.131986e+08	53.058382	52.862457	9.809933	13.202457	91.810735	91
std	108.593936	111.022486	105.911918	108.500301	7.547433e+07	18.239752	13.352063	100.975002	109.285239	106.581797	106
min	9.366667	9.800000	9.111333	9.289333	1.062000e+07	6.395305	16.564126	-233.333333	-297.930166	9.490973	9
25%	15.763167	16.082168	15.491167	15.814167	6.643185e+07	39.859440	43.595435	-76.876737	-78.543937	15.496080	15
50%	21.801001	22.198334	21.487666	21.877667	9.320775e+07	53.226417	51.621434	19.823624	24.702835	21.563733	21
75%	200.017505	204.525829	194.482498	200.049999	1.323710e+08	65.900330	61.937068	94.426550	99.180514	192.341650	196
max	411.470001	414.496674	405.666656	409.970001	9.140820e+08	97.460910	94.197983	233.333333	350.643337	357.870532	344

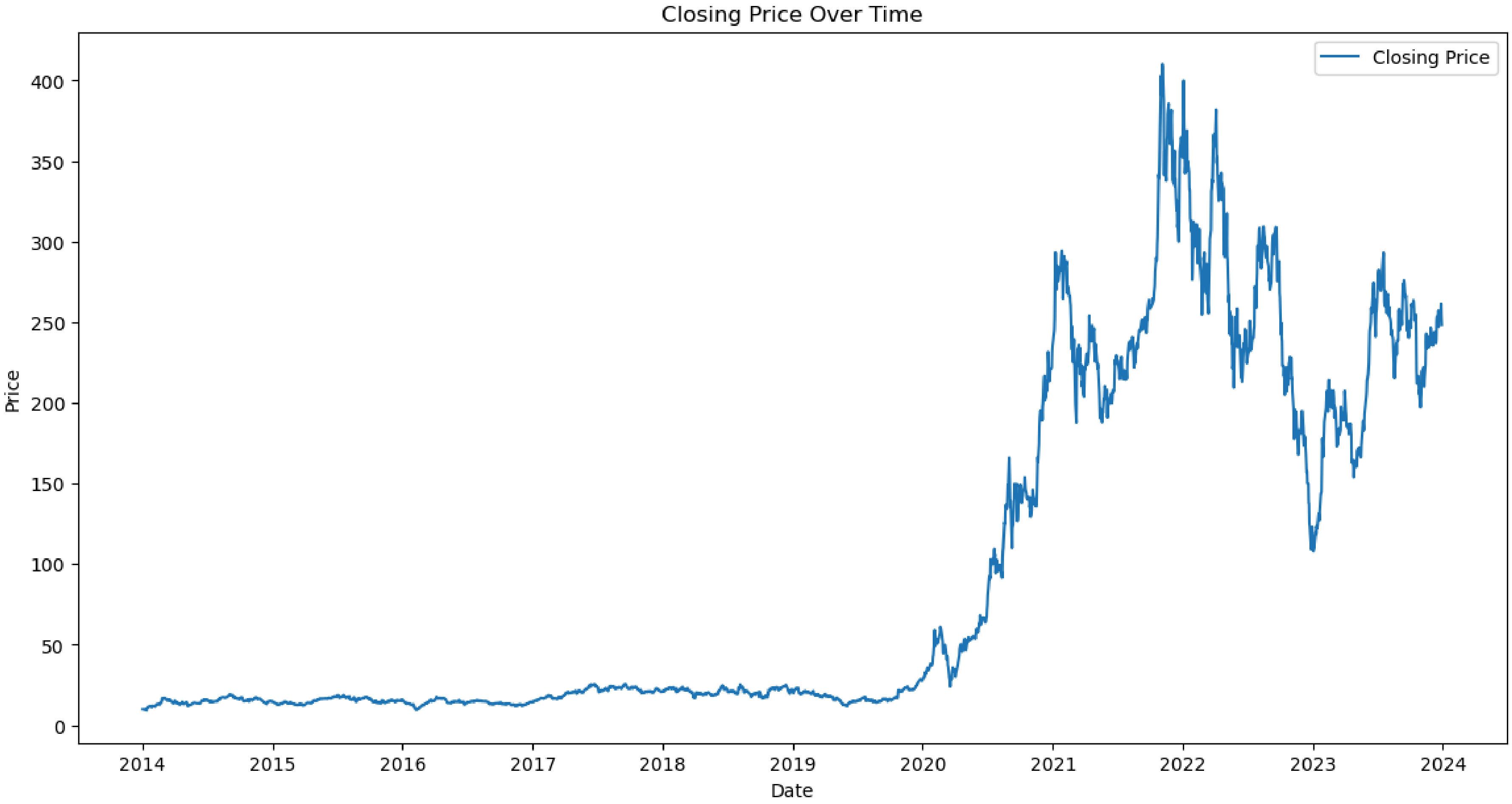
In [6]:

```
# Plot histograms for all numerical features
tesla_data.hist(figsize=(14, 14), bins=20)
plt.show()
```


open
high
low
close
volume
1e8
rsi_7
rsi_14
cci_7
cci_14
sma_50
ema_50
sma_100
ema_100
macd
bollinger
TrueRange


atr_7
atr_14
next_day_close

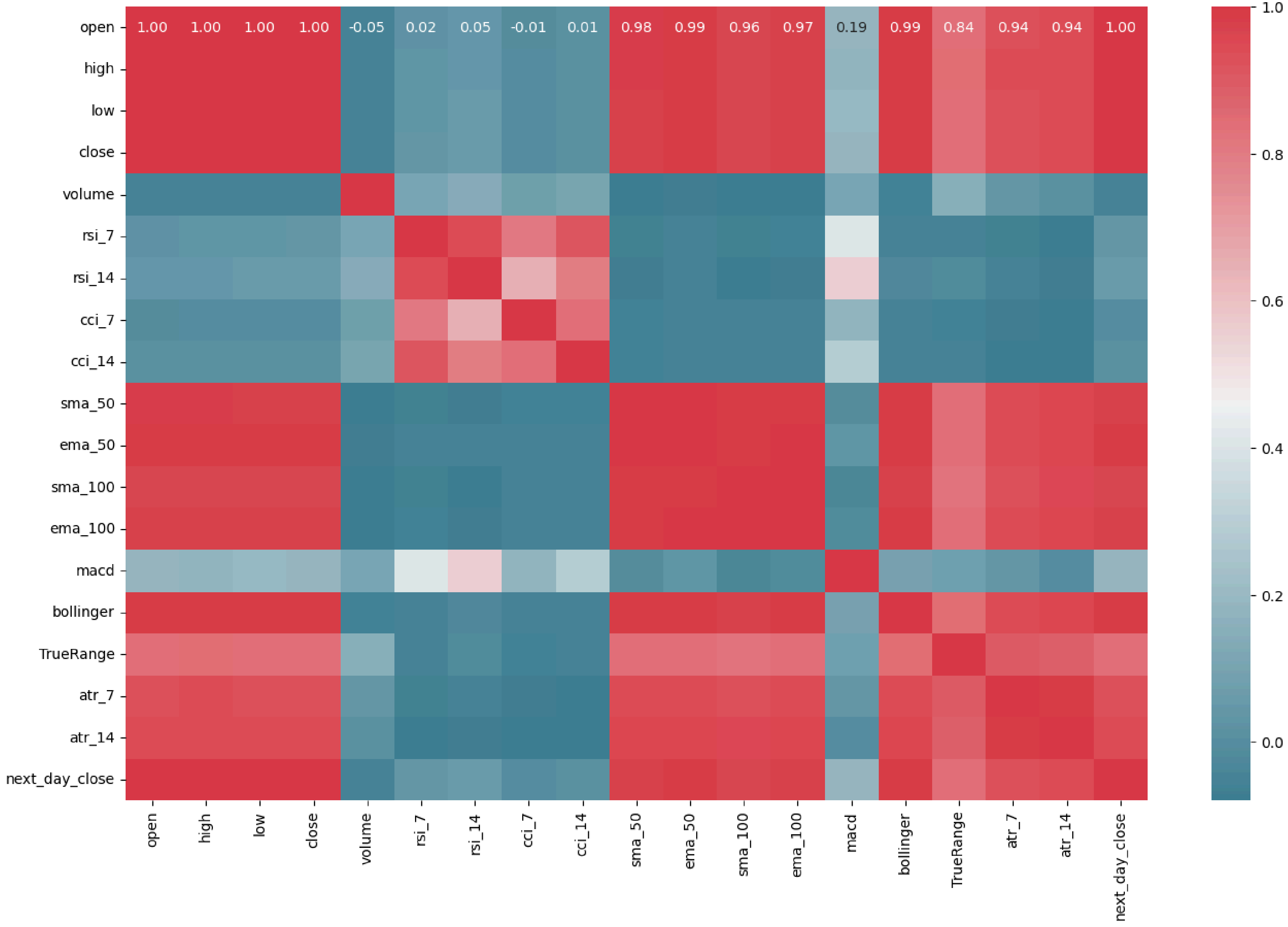
In [7]:
```
# Plot closing price over time
# Ensure the 'date' column is in datetime format
tesla_data['date'] = pd.to_datetime(tesla_data['date'])
plt.figure(figsize=(14, 7))
plt.plot(tesla_data['date'], tesla_data['close'], label='Closing Price')
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Closing Price Over Time')
plt.legend()
plt.show()
```



In [8]:
```
# Define features and target
features = tesla_data[['open', 'high', 'low', 'close', 'volume', 'rsi_7', 'rsi_14', 'cci_7', 'cci_14', 'sma_50', 'ema_
target = tesla_data['next_day_close']
```

In [9]:
```
# Drop the 'date' column as it is not numeric
tesla_data = tesla_data.drop(columns=['date'])
```

In [10]:
```
plt.subplots(figsize=(16, 10))
colormap = sns.diverging_palette(220,10,as_cmap=True)
sns.heatmap(tesla_data.corr(), annot=True, fmt=".2f", cmap=colormap)
plt.show()
```


open
high
low
close
volume
rsi_7
rsi_14
cci_7
cci_14
sma_50
ema_50
sma_100
ema_100
macd
bollinger
TrueRange
atr_7
atr_14
next_day_close
1.00 1.00 1.00 1.00 -0.05 0.02 0.05 -0.01 0.01 0.98 0.99 0.96 0.97 0.19 0.99 0.84 0.94 0.94 1.00
1.0
0.8
0.6
0.4
0.2
0.0

In [11]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

In [12]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
X_train_scaled.shape, X_test_scaled.shape
```

Out[12]:
```
((2012, 18), (504, 18))
```

In [13]:
```
from sklearn.linear_model import LinearRegression

# Initialize the model
model = LinearRegression()

# Train the model
model.fit(X_train_scaled, y_train)
```

Out[13]:
```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [15]:
```
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Make predictions
y_pred = model.predict(X_test_scaled)



# Calculate metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
print(f'Mean Absolute Error: {mae}')
print (f'Coefficients:  ', model.coef_)
```

```
Mean Squared Error: 50.494142206734516
R-squared: 0.9956982654163343
Mean Absolute Error: 5.293769293234429
Coefficients:   [ -8.71301384   6.33910751  24.33301282  86.51421893   0.10255627
   1.05416591  -0.43380552  -0.32188761  -0.14094462  21.10679322
 -49.06750533  12.92493621   0.81159435   0.56213897  14.76116595
   0.34992027   1.80426805  -2.4958959 ]
```

In [16]:
```
# Example: Predict the next day's closing price based on today's data
today_data = tesla_data.iloc[-1][['open', 'high', 'low', 'close', 'volume', 'rsi_7', 'rsi_14', 'cci_7', 'cci_14', 'sma
predicted_price = model.predict(today_data)
print(f'Predicted next day close price: {predicted_price}')
```

```
Predicted next day close price: [10345776.47129408]
```

Regression Analysis Using Decison Tree to compare the model

In [17]:
```
from sklearn.tree import DecisionTreeRegressor

# Train a Decision Tree Regressor
tree_model = DecisionTreeRegressor(random_state=42)
tree_model.fit(X_train_scaled, y_train)

# Predict on the testing set
y_pred_tree = tree_model.predict(X_test_scaled)

# Evaluate the model

mse_tree = mean_squared_error(y_test, y_pred_tree)
mae_tree = mean_absolute_error(y_test, y_pred_tree)
r2_tree = r2_score(y_test, y_pred_tree)

print(f'Mean Squared Error: {mse_tree}')
print(f'R-squared: {r2_tree}')
print(f'Mean Absolute Error: {mae_tree}')
```

```
Mean Squared Error: 103.76416777333037
R-squared: 0.9911600457092964
Mean Absolute Error: 6.6160662599206335
```

In [21]:
```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score

# Initialize the Ridge Regression model
ridge_model = Ridge(alpha=1.0, random_state=42)  # You can adjust the alpha parameter for regularization strength

# Train the Ridge Regression model
ridge_model.fit(X_train_scaled, y_train)

# Predict on the testing set
y_pred_ridge = ridge_model.predict(X_test_scaled)

# Evaluate the model
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
mae_ridge = mean_absolute_error(y_test, y_pred_ridge)
r2_ridge = r2_score(y_test, y_pred_ridge)

mae_ridge, mse_ridge, r2_ridge
```

Out[21]:
```
(5.305062611621865, 51.00654814458296, 0.9956546121479078)
```

In [ ]:
```

```