Back-Testing for Algorithmic Trading Strategies

How to choose a strategy? It would be a good start to test alternative strategies retrospectively, knowing which of these strategies worked best in the past and produced more accurate signals. This is called Back-Test.

In [1]:
```
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import yahoofinancials as yf
from yahoo_fin.stock_info import *
import requests_html
import requests
import ftplib
import ta as ta
import io

from tscv import GapWalkForward
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.model_selection import GridSearchCV,cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.tree import export_graphviz

from six import StringIO

from IPython.display import Image
import pydotplus
from yellowbrick.classifier import ClassificationReport,ConfusionMatrix,ROCAUC
from yellowbrick.model_selection import FeatureImportances
import graphviz
```

Let's get the historical time series data of the stock by specifying the start and end dates

In [2]:
```
history = yf.YahooFinancials('TSLA').get_historical_price_data('2021-01-01', '2021-1
df = pd.DataFrame(history['TSLA']['prices'])
df.head()
```

Out[2]:

	date	high	low	open	close	volume	adjclose	formatted_da
0	1609770600	248.163330	239.063339	239.820007	243.256668	145914600	243.256668	2021-01-
1	1609857000	246.946671	239.733337	241.220001	245.036667	96735600	245.036667	2021-01-
2	1609943400	258.000000	249.699997	252.830002	251.993332	134100000	251.993332	2021-01-
3	1610029800	272.329987	258.399994	259.209991	272.013336	154496700	272.013336	2021-01-

	date	high	low	open	close	volume	adjclose	formatted_da
4	1610116200	294.829987	279.463318	285.333344	293.339996	225166500	293.339996	2021-01-

In [3]:
```
df.drop('date', axis=1, inplace=True)
```

In [4]:
```
df.index = pd.to_datetime(df['formatted_date'])
df.drop('formatted_date', axis=1, inplace=True)
```

Let's change the name of the adjclose variable to 'Price'.

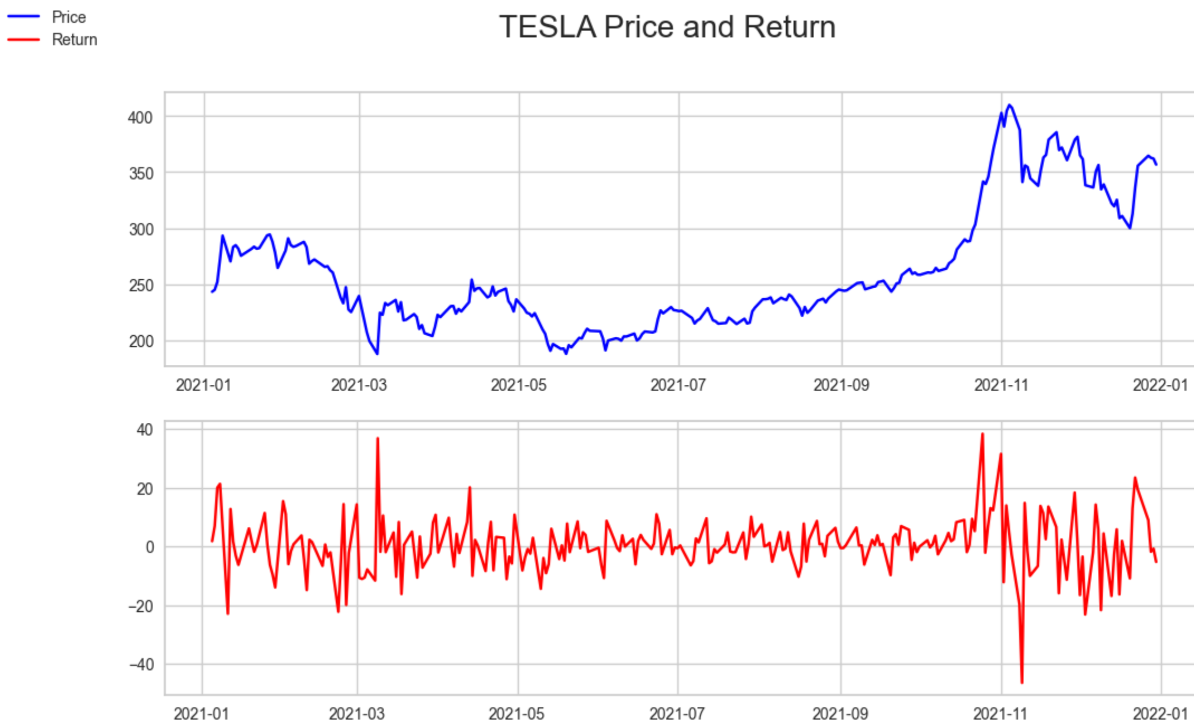
In [5]:
```
df.rename(columns={'adjclose': 'price'}, inplace=True)
```

Let's change the price change to 'return' (daily return) and assign the percentage change in the price to 'return_pct'.

In [6]:
```
df["return"] = df["price"].diff()
df["return_pct"] = df["price"].pct_change()
```

Time Series Graph of Price and Return

In [7]:
```
f, axarr = plt.subplots(2,sharex=False,figsize=(12,7))
f.suptitle('TESLA Price and Return', fontsize=20)
axarr[0].plot(df['price'], color='blue')
axarr[0].grid(True)
axarr[1].plot(df['return'], color='red')
axarr[1].grid(True)
f.legend(['Price', 'Return'], loc='upper left')
plt.show()
```



Three Alternative Strategies

- **1) Price > EMA10 : If the price goes above the 10-day exponential moving average, it is considered a buy signal.**
- **2) EMA10 > EMA30 : When the 10-day exponential moving average rises above the 30-day exponential moving average, it can be considered as a buy signal.**
- **3) MACD > MACDS : The MACD indicator is the difference between the exponential values of the 26 and 12-day moving average. The 9-day exponential moving average of the MACD is called the MACD Signal (MACDS). If the MACD goes above the MACDS, it is considered a buy signal.**

1) Price > EMA10 (s1)

In [8]:

```
df["EMA10"] = ta.trend.ema_indicator(df["price"],10,fillna=True)
```

Let's create Buy-Sell Signals

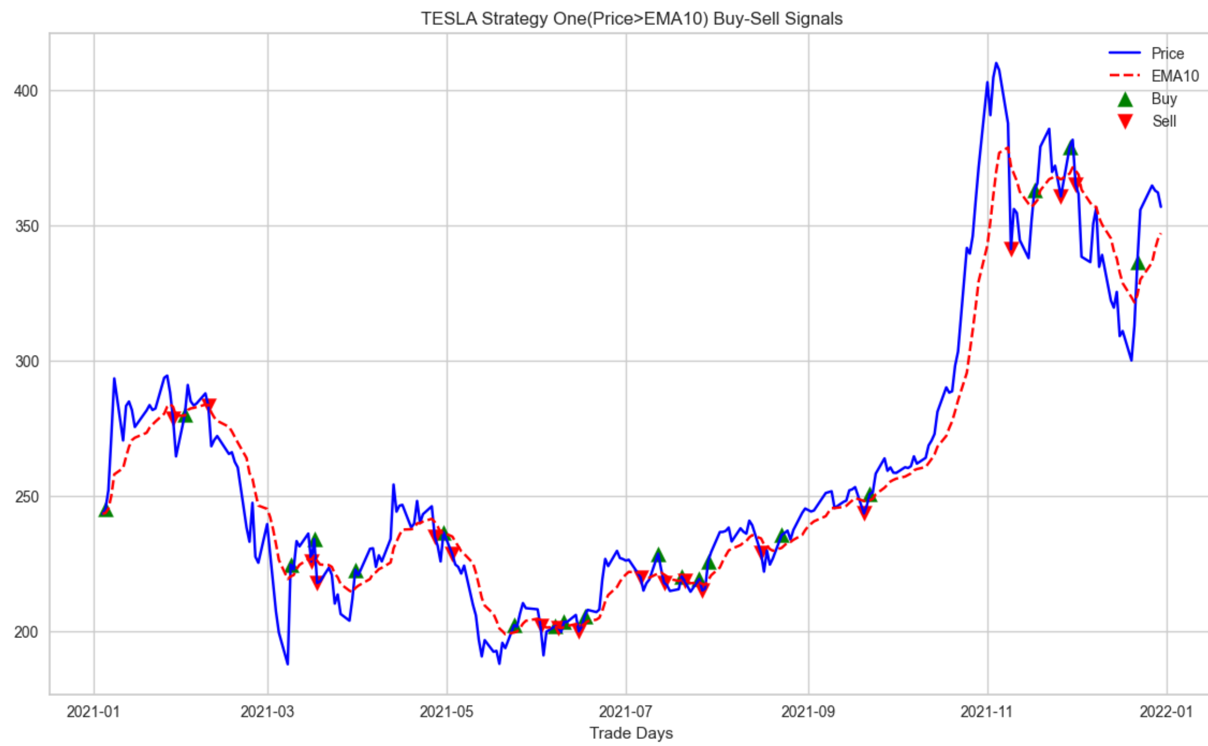
In [9]:

```
df["buy_s1"] = np.where(df["price"] > df["EMA10"], 1, 0)
df["sell_s1"] = np.where(df["price"] < df["EMA10"], 1, 0)
df["buy_s1_ind"] = np.where((df["buy_s1"] > df["buy_s1"].shift(1)),1, 0)
df["sell_s1_ind"] = np.where((df["sell_s1"] > df["sell_s1"].shift(1)),1, 0)
```

In [10]:

```
df["date"] = df.index
```

```
fig1 = plt.figure(figsize=(14,8))
plt.plot(df["price"],label="Price",color='blue')
plt.plot(df["EMA10"],label="EMA10",color='red',linestyle='--')
plt.scatter(df.loc[df["buy_s1_ind"] == 1].index,
            df["price"][df["buy_s1_ind"] == 1], color='green', marker='^', s=100, la

plt.scatter(df.loc[df["sell_s1_ind"] == 1].index,
            df["price"][df["sell_s1_ind"] == 1], color='red', marker='v', s=100, lab

plt.xlabel('Trade Days')
plt.legend(loc='best')
plt.title('TESLA Strategy One(Price>EMA10) Buy-Sell Signals')
plt.show()
```



According to the Strategy One, the profit of the trader who trades with $1000 in the relevant period:

Assuming we allocate 5% of the return as transaction costs, then we will consider 95% of the percentage return.

In [11]:
```
df["value_s1"] = 1000*(1+(np.where(df["buy_s1"]==1,
                                   0.95*df["return_pct"],0)).cumsum())
```

Back-Test Report (s1)

In [12]:
```
print("************ Descriptive Statistics ************")
print("Period",len(df),"days")
print("Highest Daily Loss ",100*round(df["return_pct"].min(),2),"%")
print("Highest Daily Return ",100*round(df["return_pct"].max(),2),"%")
print("Standard Deviation of Return ",100*round(df["return_pct"].std(),2),"%")
```

```
print("Total Potential Return ",100*(round(sum(np.where((df["return_pct"]>0),df["ret
print("Total Potential Loss ",100*(round(sum(np.where((df["return_pct"]<0),df["retur
print("Net Return ",100*df["return_pct"].sum().round(2),"%")

print("************* MODEL PERFORMANCE *************")

print("Return Captured by the Model ",100*sum(np.where((df["buy_s1"]==1),df["return_
print("Loss Maintained by the Model ",100*sum(np.where((df["sell_s1"]==1),df["return
print("**************************************************")
```

```
************* Descriptive Statistics *************
Period 251 days
Highest Daily Loss  -12.0 %
Highest Daily Return  20.0 %
Standard Deviation of Return  3.0 %
Total Potential Return  336.0 %
Total Potential Loss  -283.0 %
Net Return  53.0 %
************* MODEL PERFORMANCE *************
Return Captured by the Model  200.0 %
Loss Maintained by the Model  -147.0 %
**************************************************
```

In [13]:

```
print("************* REPORT *************")
print("The end-of-period price of the stock, which was $",df["price"][0].round(2),
    "at the beginning of the period, became $",df["price"][-1].round(2),"with %",
    (100*(df["price"][-1]-df["price"][0])/df["price"][0]).round(2),"change","The moc
    100*(sum(np.where((df["buy_s1"]==1),df["return_pct"],0))/sum(np.where((df["retur
                                                                       df["return_pct"]
    "of the total positive return.The investment of $1000 at the beginning of the pe
    df["value_s1"][-1].round(2),
    "on the first",len(df),"days.")
```

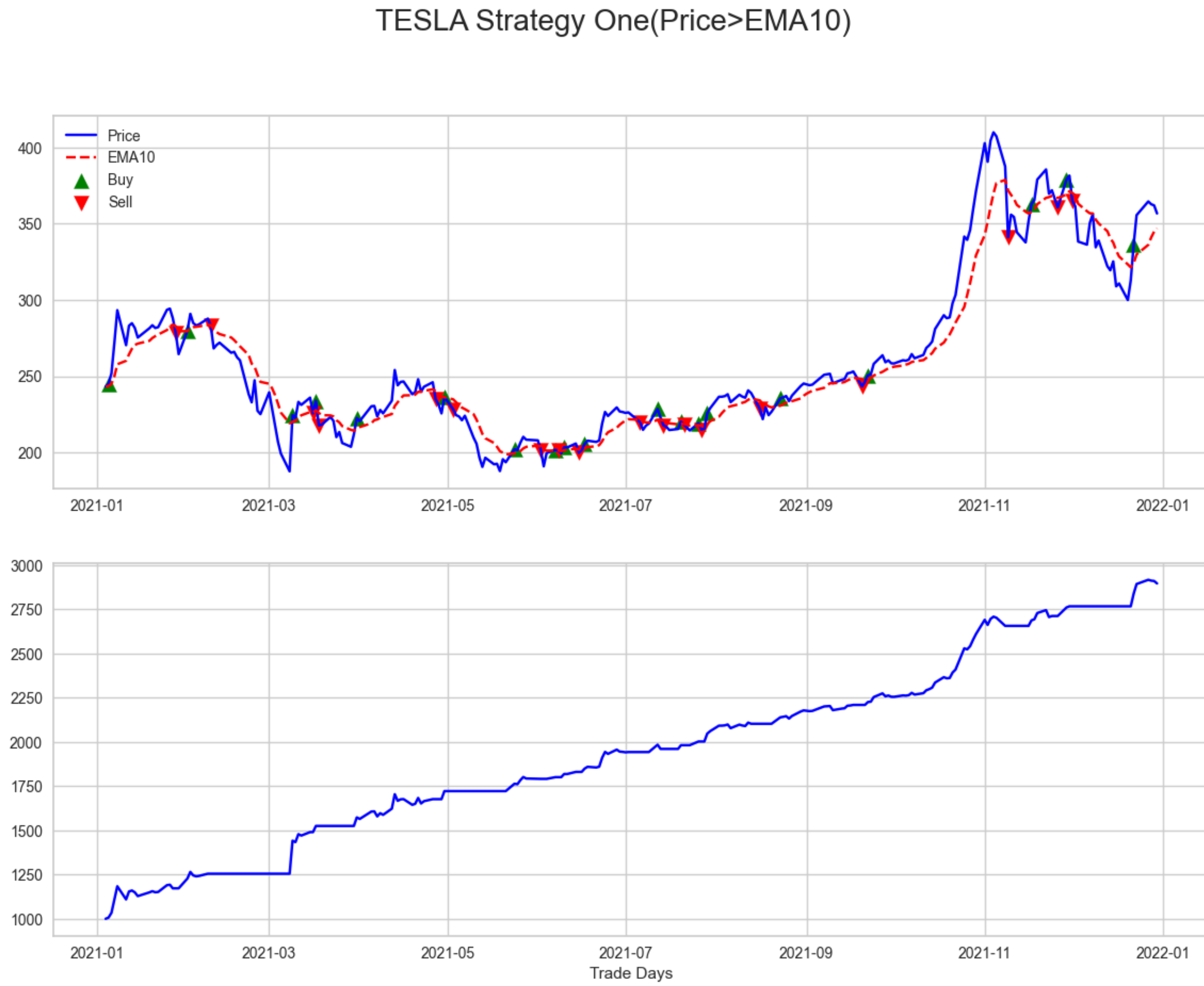
```
************* REPORT *************
The end-of-period price of the stock, which was $ 243.26 at the beginning of the per
iod, became $ 356.78 with % 46.67 change The model captured % 59.0 of the total posi
tive return.The investment of $1000 at the beginning of the period became $ 2896.76
on the first 251 days.
```

The sum of the percentage gains on the days when the stock increased was 336%, and the sum of the percentage loss on the days when the stock decreased was 283%. For the price > EMA10 strategy, it correctly saw 59% of the potential positive gain. On the days when the price correctly indicated that the price would increase and gave a buy signal, the return was 200%. However, someone who bought this stock at the beginning of the period and sold it at the end of the period made a profit of 46%, but according to the signals, the one who traded made a profit of 189.67% for $ 1000.

In [14]:

```
df["date"] = df.index
f,axarr = plt.subplots(2,sharex=False,figsize=(14,10))
f.suptitle('TESLA Strategy One(Price>EMA10)', fontsize=20)
axarr[0].plot(df["price"],label="Price",color='blue')
axarr[0].plot(df["EMA10"],label="EMA10",color='red',linestyle='--')
axarr[0].scatter(df.loc[df["buy_s1_ind"] == 1].index,
                df.loc[df["buy_s1_ind"] == 1,"price"].values, color='green', marker=
axarr[0].scatter(df.loc[df["sell_s1_ind"] == 1].index,
                df.loc[df["sell_s1_ind"] == 1,"price"].values, color='red', marker='
```

```python
axarr[0].legend(loc='best')
axarr[1].plot(df["value_s1"],label="Algorithmic Gain",color='blue')
plt.grid(True)
plt.xlabel('Trade Days')
plt.show()
```



The algorithmic payoff exceeds the potential payoff. Why? Because the investor often earned additional income by selling high and buying low. The cumulative sum of these is greater than the potential gain. Let's Back-Test the same implementation in the other two strategies.

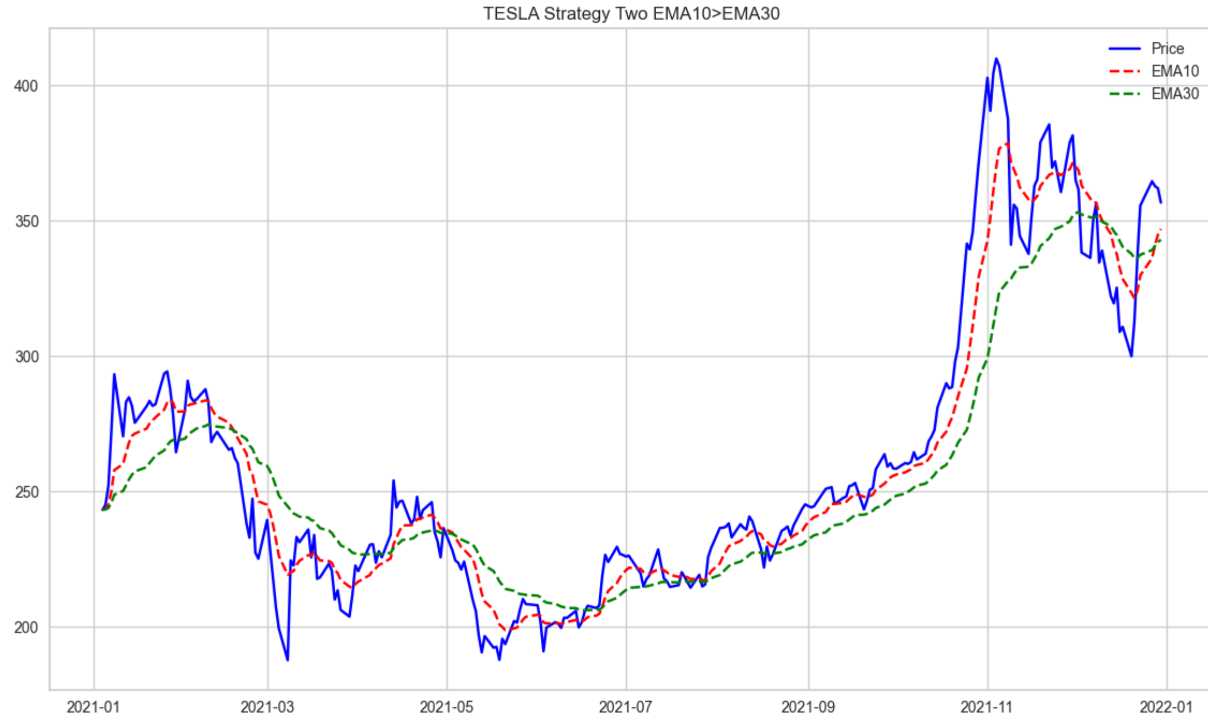
2) EMA10 > EMA30 (s2)

In [15]:

```python
df["EMA30"] = ta.trend.ema_indicator(df["price"],30,fillna=True)
```

In [16]:

```python
fig1 = plt.figure(figsize=(14,8))
plt.plot(df["price"],label="Price",color='blue')
plt.plot(df["EMA10"],label="EMA10",color='red',linestyle='--')
plt.plot(df["EMA30"],label="EMA30",color='green',linestyle='--')
plt.legend(loc='best')
plt.title('TESLA Strategy Two EMA10>EMA30')
plt.show()
```

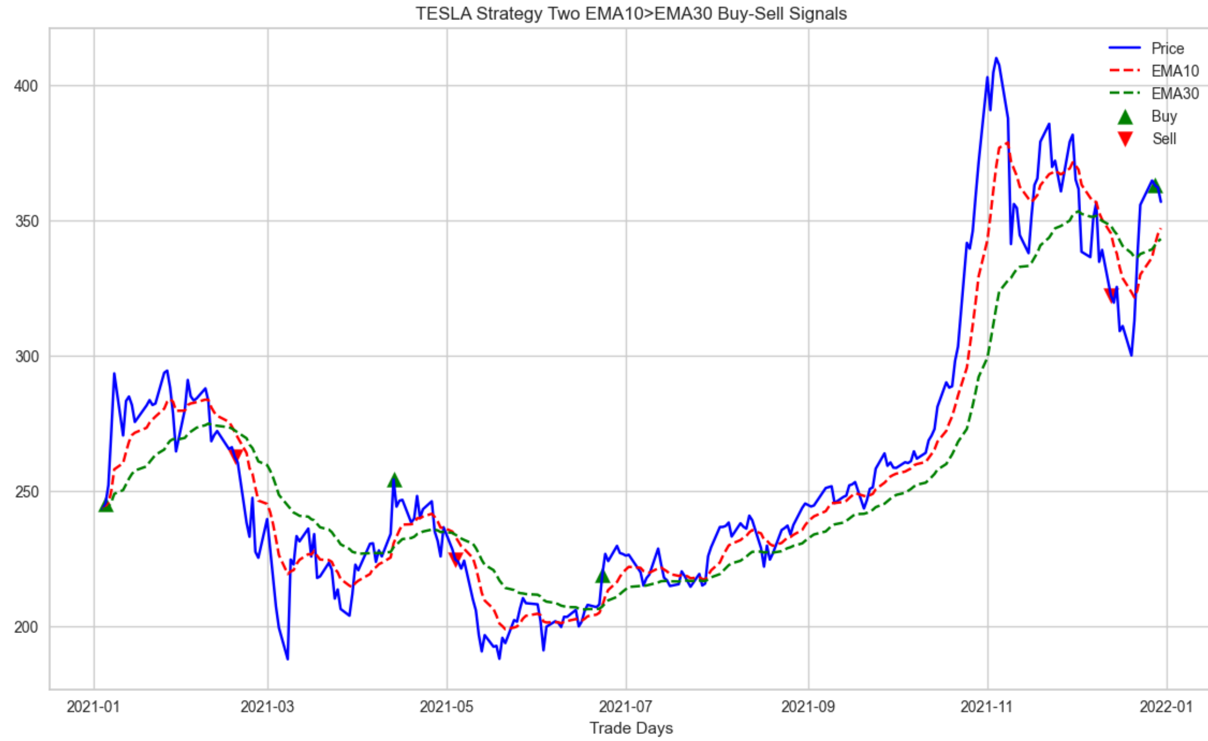


In [17]:

```
df["buy_s2"] = np.where((df["EMA10"] > df["EMA30"]), 1, 0)
df["sell_s2"] = np.where((df["EMA10"] < df["EMA30"]), 1, 0)
df["buy_s2_ind"] = np.where((df["buy_s2"] > df["buy_s2"].shift(1)),1, 0)
df["sell_s2_ind"] = np.where((df["sell_s2"] > df["sell_s2"].shift(1)),1, 0)
```

In [18]:

```
df["date"] = df.index

fig1 = plt.figure(figsize=(14,8))
plt.plot(df["price"],label="Price",color='blue')
plt.plot(df["EMA10"],label="EMA10",color='red',linestyle='--')
plt.plot(df["EMA30"],label="EMA30",color='green',linestyle='--')
plt.scatter(df.loc[df["buy_s2_ind"] == 1].index,
            df["price"][df["buy_s2_ind"] == 1], color='green', marker='^', s=100, la
plt.scatter(df.loc[df["sell_s2_ind"] == 1].index,
            df["price"][df["sell_s2_ind"] == 1], color='red', marker='v', s=100, lab

plt.xlabel('Trade Days')
plt.legend(loc='best')
plt.title('TESLA Strategy Two EMA10>EMA30 Buy-Sell Signals')
plt.show()
```



Trading Gain according to Strategy Two

In [19]:

```
df["value_s2"] = 1000*(1+(np.where(df["buy_s2"]==1,
                                    0.95*df["return_pct"],0)).cumsum())
```

Back-Test Report (s2)

In [20]:

```
print("************* Descriptive Statistics *************")
print("Period",len(df),"days")
print("Highest Daily Loss ",100*round(df["return_pct"].min(),2),"%")
print("Highest Daily Return ",100*round(df["return_pct"].max(),2),"%")
print("Standard Deviation of Return ",100*round(df["return_pct"].std(),2),"%")
print("Total Potential Return ",100*(round(sum(np.where((df["return_pct"]>0),df["ret
print("Total Potential Loss ",100*(round(sum(np.where((df["return_pct"]<0),df["retur
print("Net Return ",100*df["return_pct"].sum().round(2),"%")

print("************* MODEL PERFORMANCE *************")

print("Return Captured by the Model ",100*sum(np.where((df["buy_s2"]==1),df["return_
print("Loss Maintained by the Model ",100*sum(np.where((df["sell_s2"]==1),df["returr
print("**************************************************")
```

```
************* Descriptive Statistics *************
Period 251 days
Highest Daily Loss  -12.0 %
Highest Daily Return  20.0 %
Standard Deviation of Return  3.0 %
Total Potential Return  336.0 %
Total Potential Loss  -283.0 %
Net Return  53.0 %
************* MODEL PERFORMANCE *************
```

```
Return Captured by the Model  61.0 %
Loss Maintained by the Model  -8.0 %
**************************************************
```

In [21]:

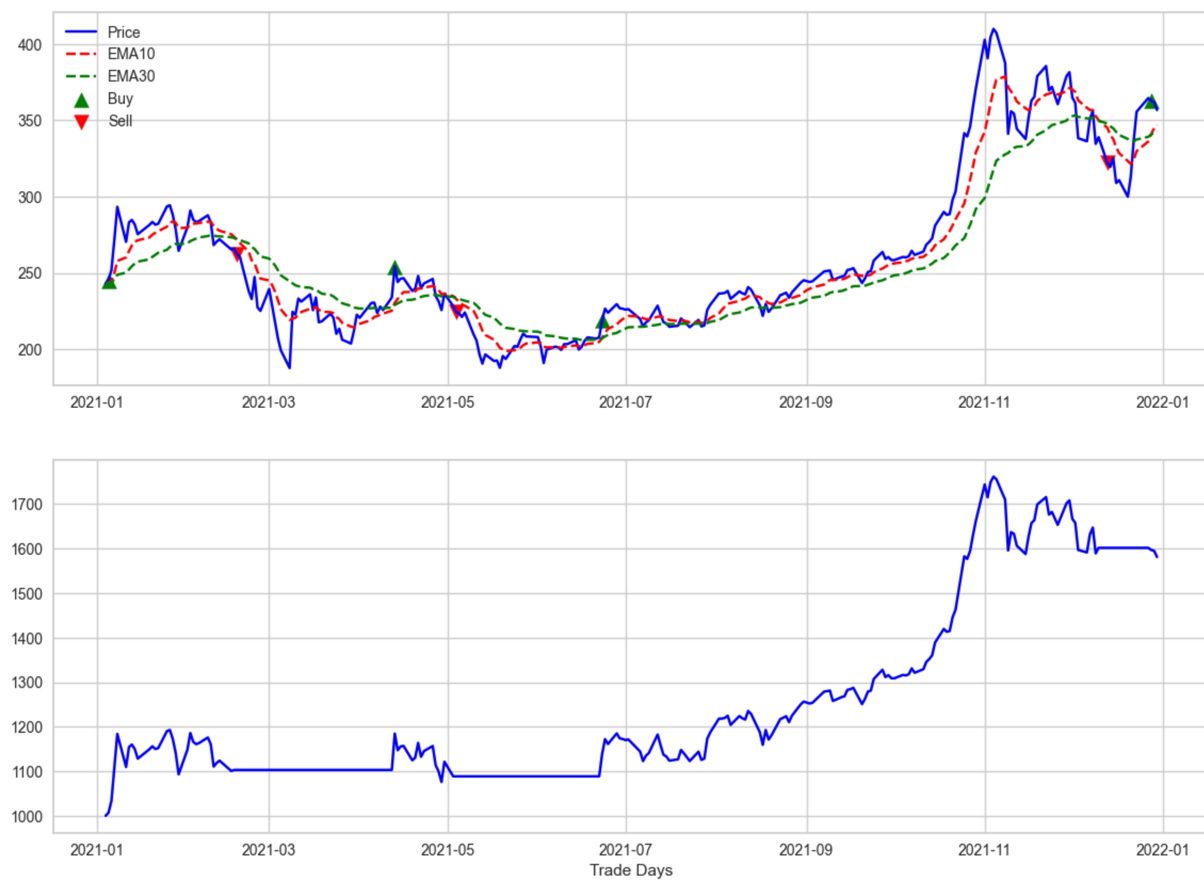
```
print("************* REPORT *************")
print("The end-of-period price of the stock, which was $",df["price"][0].round(2),
    "at the beginning of the period, became $",df["price"][-1].round(2),"with %",
    (100*(df["price"][-1]-df["price"][0])/df["price"][0]).round(2),"change","The moc
    100*(sum(np.where((df["buy_s2"]==1),df["return_pct"],0))/sum(np.where((df["retur
                                                                          df["return_pct"]
    "of the total positive return.The investment of $1000 at the beginning of the pe
    df["value_s2"][-1].round(2),
    "on the first",len(df),"days.")
```

```
************* REPORT *************
The end-of-period price of the stock, which was $ 243.26 at the beginning of the per
iod, became $ 356.78 with % 46.67 change The model captured % 18.0 of the total posi
tive return.The investment of $1000 at the beginning of the period became $ 1581.24
on the first 251 days.
```

In [22]:

```
f,axarr = plt.subplots(2,sharex=False,figsize=(14,10))
f.suptitle('TESLA Strategy Two(EMA10>EMA30)', fontsize=20)
axarr[0].plot(df["price"],label="Price",color='blue')
axarr[0].plot(df["EMA10"],label="EMA10",color='red',linestyle='--')
axarr[0].plot(df["EMA30"],label="EMA30",color='green',linestyle='--')
axarr[0].scatter(df.loc[df["buy_s2_ind"] == 1].index,
                df.loc[df["buy_s2_ind"] == 1,"price"].values, color='green', marker=
axarr[0].scatter(df.loc[df["sell_s2_ind"] == 1].index,
                df.loc[df["sell_s2_ind"] == 1,"price"].values, color='red', marker='

axarr[0].legend(loc='best')
axarr[1].plot(df["value_s2"],label="Algorithmic Gain",color='blue')
plt.grid(True)
plt.xlabel('Trade Days')
plt.show()
```

TESLA Strategy Two(EMA10>EMA30)



3) MACD > MACDS (s3)

In [23]:

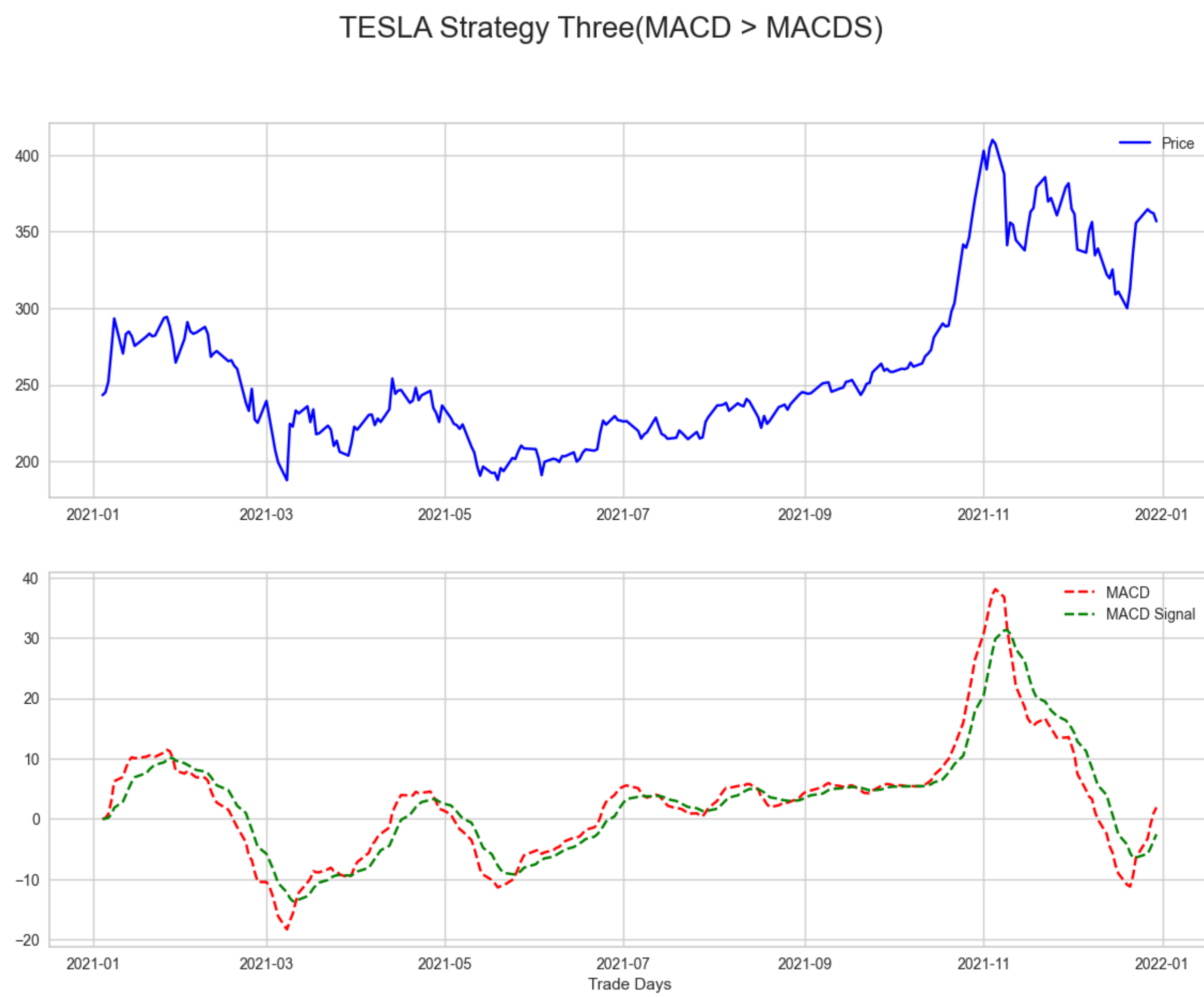
```
df["MACD"] = ta.trend.macd(df["price"],fillna=True,window_fast=12,window_slow=26)
df["MACD_signal"] = ta.trend.macd_signal(df["price"],
                    window_fast=12,window_slow=26,window_sign=9,
                    fillna=True)
```

In [24]:

```
df["buy_s3"] = np.where((df["MACD"] > df["MACD_signal"]), 1, 0)
df["sell_s3"] = np.where((df["MACD"] < df["MACD_signal"]), 1, 0)
df["buy_s3_ind"] = np.where((df["buy_s3"] > df["buy_s3"].shift(1)),1, 0)
df["sell_s3_ind"] = np.where((df["sell_s3"] > df["sell_s3"].shift(1)),1, 0)
```

In [25]:

```
f,axarr = plt.subplots(2,sharex=False,figsize=(14,10))
f.suptitle('TESLA Strategy Three(MACD > MACDS)', fontsize=20)
axarr[0].plot(df["price"],label="Price",color='blue')
axarr[0].legend(loc='best')
axarr[0].grid(True)
axarr[1].plot(df["MACD"],label="MACD",color='red',linestyle='--')
axarr[1].plot(df["MACD_signal"],label="MACD Signal",color='green',linestyle='--')


axarr[1].legend(loc='best')
plt.xlabel('Trade Days')
plt.show()
```



Trading Gain according to Strategy Three

In [26]:

```
df["value_s3"] = 1000*(1+(np.where(df["buy_s3"]==1,
                                   0.95*df["return_pct"],0)).cumsum())
```

Back-Test Report (s3)

In [27]:

```
print("************* Descriptive Statistics *************")
print("Period",len(df),"days")
print("Highest Daily Loss ",100*round(df["return_pct"].min(),2),"%")
print("Highest Daily Return ",100*round(df["return_pct"].max(),2),"%")
print("Standard Deviation of Return ",100*round(df["return_pct"].std(),2),"%")
print("Total Potential Return ",100*(round(sum(np.where((df["return_pct"]>0),df["ret
print("Total Potential Loss ",100*(round(sum(np.where((df["return_pct"]<0),df["retur
print("Net Return ",100*df["return_pct"].sum().round(2),"%")

print("************* MODEL PERFORMANCE *************")

print("Return Captured by the Model ",100*sum(np.where((df["buy_s3"]==1),df["return_
print("Loss Maintained by the Model ",100*sum(np.where((df["sell_s3"]==1),df["returr
print("*****************************************************")
```

```
************* Descriptive Statistics *************
Period 251 days
Highest Daily Loss  -12.0 %
Highest Daily Return  20.0 %
Standard Deviation of Return  3.0 %
Total Potential Return  336.0 %
Total Potential Loss  -283.0 %
Net Return  53.0 %
************* MODEL PERFORMANCE *************
Return Captured by the Model  72.0 %
Loss Maintained by the Model  -19.0 %
**************************************************
```

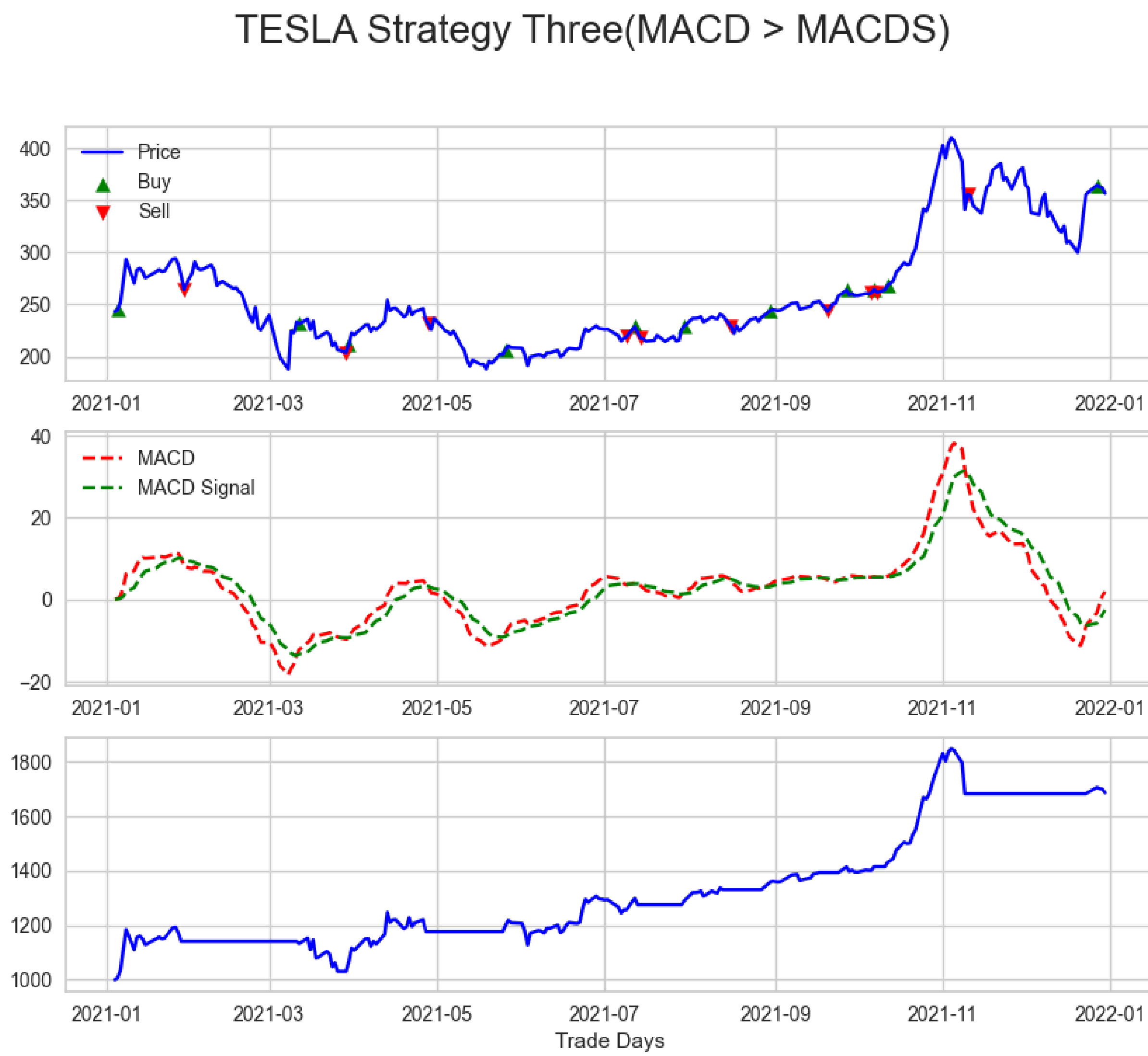
In [28]:

```
print("************* REPORT *************")
print("The end-of-period price of the stock, which was $",df["price"][0].round(2),
    "at the beginning of the period, became $",df["price"][-1].round(2),"with %",
    (100*(df["price"][-1]-df["price"][0])/df["price"][0]).round(2),"change","The moc
    100*(sum(np.where((df["buy_s3"]==1),df["return_pct"],0))/sum(np.where((df["retur
                                                                       df["return_pct"]

    "of the total positive return.The investment of $1000 at the beginning of the pe
    df["value_s3"][-1].round(2),
    "on the first",len(df),"days.")
```

```
************* REPORT *************
The end-of-period price of the stock, which was $ 243.26 at the beginning of the per
iod, became $ 356.78 with % 46.67 change The model captured % 21.0 of the total posi
tive return.The investment of $1000 at the beginning of the period became $ 1683.92
on the first 251 days.
```

In [29]:

```
f,axarr = plt.subplots(3,sharex=False,figsize=(10,8))
f.suptitle('TESLA Strategy Three(MACD > MACDS)', fontsize=20)
axarr[0].plot(df["price"],label="Price",color='blue')
axarr[0].scatter(df.loc[df["buy_s3_ind"] == 1].index,
                df.loc[df["buy_s3_ind"] == 1,"price"].values, color='green', marker=
axarr[0].scatter(df.loc[df["sell_s3_ind"] == 1].index,
                df.loc[df["sell_s3_ind"] == 1,"price"].values, color='red', marker='
axarr[0].legend(loc='best')
axarr[1].plot(df["MACD"],label="MACD",color='red',linestyle='--')
axarr[1].plot(df["MACD_signal"],label="MACD Signal",color='green',linestyle='--')

axarr[1].legend(loc='best')
axarr[2].plot(df["value_s3"],label="Algorithmic Gain",color='blue')
plt.grid(True)
plt.xlabel('Trade Days')
plt.show()
```



Back-Testing is actually a method that we will use in choosing the current strategy. It is natural for price movements to deviate from the direction the strategy is pointing. Since the calculated indicators are created from the movements of the stock, they are very dependent on the price formation. There are many factors that affect the price. Although it recommends the Back-Test Price>EMA10 strategy for 2021, it may not perform the same for 2022 or the first days of 2023. The only strategy can be misleading.

So how can there be a solution to this situation? Answer: Regression Toward The Mean

- **Let's say we create a stronger signal by combining the buy and sell signals pointed out by the three indicators. Buy when two of the three indicators give a buy signal, and sell when it gives a sell signal.**

In [30]:

```
df["BUY"] = np.where((df["buy_s1"]+df["buy_s2"]+df["buy_s3"])>=2,1,0)
df["SELL"] = np.where((df["sell_s1"]+df["sell_s2"]+df["sell_s3"])>=2,1,0)
df["BUY_ind"] = np.where((df["BUY"] > df["BUY"].shift(1)),1, 0)
df["SELL_ind"] = np.where((df["SELL"] > df["SELL"].shift(1)),1, 0)
```

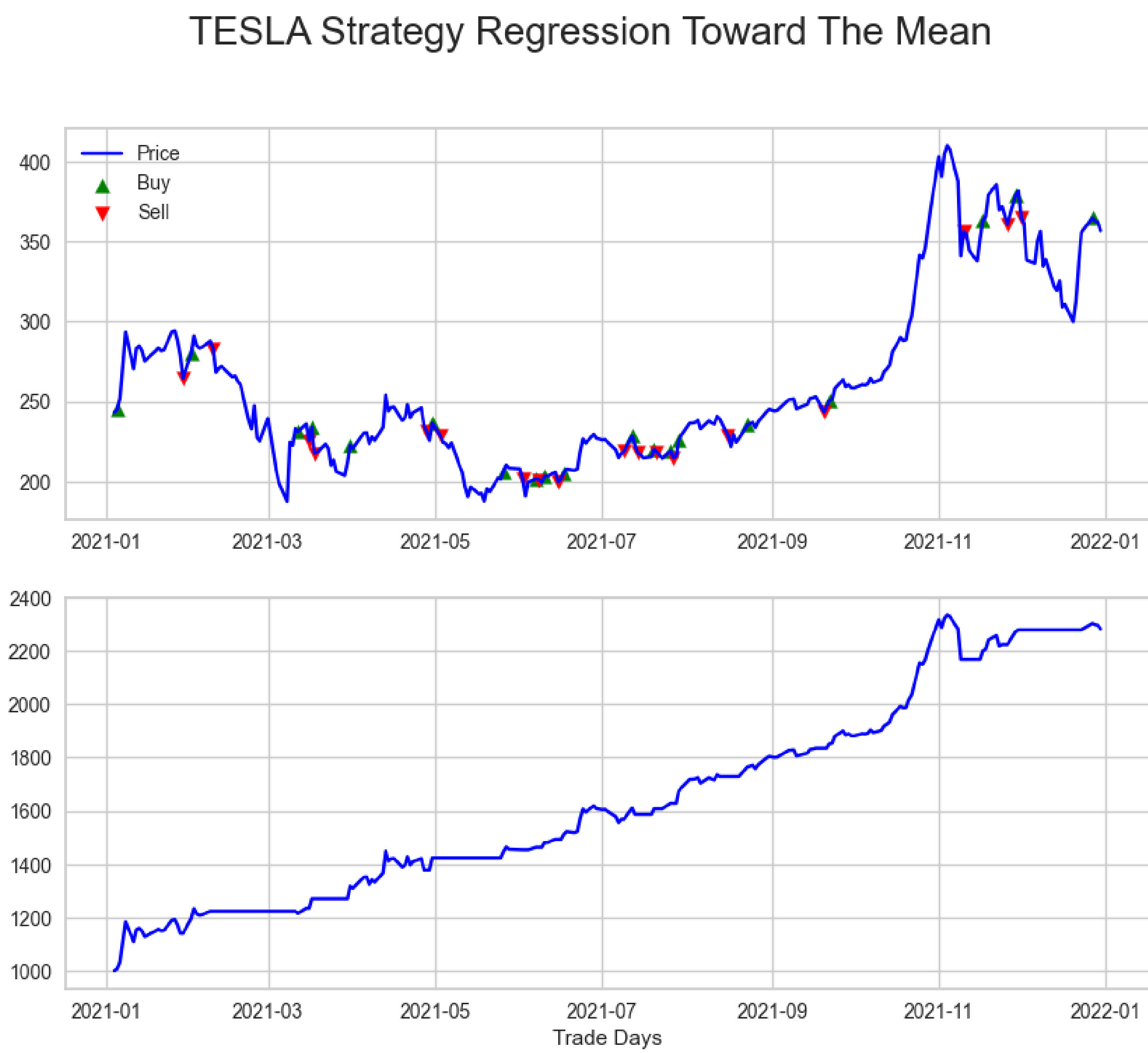
In [31]:
```
df["VALUE"] = 1000*(1+(np.where(df["BUY"]==1,
                                0.95*df["return_pct"],0)).cumsum())
```

In [32]:
```
print("************* Descriptive Statistics *************")
print("Period",len(df),"days")
print("Highest Daily Loss ",100*round(df["return_pct"].min(),2),"%")
print("Highest Daily Return ",100*round(df["return_pct"].max(),2),"%")
print("Standard Deviation of Return ",100*round(df["return_pct"].std(),2),"%")
print("Total Potential Return ",100*(round(sum(np.where((df["return_pct"]>0),df["ret
print("Total Potential Loss ",100*(round(sum(np.where((df["return_pct"]<0),df["retur
print("Net Return ",100*df["return_pct"].sum().round(2),"%")

print("************* MODEL PERFORMANCE *************")

print("Return Captured by the Model ",100*sum(np.where((df["BUY"]==1),df["return_pct
print("Loss Maintained by the Model ",100*sum(np.where((df["SELL"]==1),df["return_pc
print("**************************************************")

print("************* REPORT *************")
print("The end-of-period price of the stock, which was $",df["price"][0].round(2),
    "at the beginning of the period, became $",df["price"][-1].round(2),"with %",
    (100*(df["price"][-1]-df["price"][0])/df["price"][0]).round(2),"change","The moc
    100*(sum(np.where((df["BUY"]==1),df["return_pct"],0))/sum(np.where((df["return_p
                                                                    df["return_pct"]
    "of the total positive return.The investment of $1000 at the beginning of the pe
    df["VALUE"][-1].round(2),
    "on the first",len(df),"days.")
```

```
************* Descriptive Statistics *************
Period 251 days
Highest Daily Loss  -12.0 %
Highest Daily Return  20.0 %
Standard Deviation of Return  3.0 %
Total Potential Return  336.0 %
Total Potential Loss  -283.0 %
Net Return  53.0 %
************* MODEL PERFORMANCE *************
Return Captured by the Model  135.0 %
Loss Maintained by the Model  -82.0 %
**************************************************
************* REPORT *************
The end-of-period price of the stock, which was $ 243.26 at the beginning of the per
iod, became $ 356.78 with % 46.67 change The model captured % 40.0 of the total posi
tive return.The investment of $1000 at the beginning of the period became $ 2283.04
on the first 251 days.
```

In [33]:
```
f,axarr = plt.subplots(2,sharex=False,figsize=(10,8))
f.suptitle('TESLA Strategy Regression Toward The Mean', fontsize=20)
axarr[0].plot(df["price"],label="Price",color='blue')
axarr[0].scatter(df.loc[df["BUY_ind"] == 1].index,
                df.loc[df["BUY_ind"] == 1,"price"].values, color='green', marker='^'

axarr[0].scatter(df.loc[df["SELL_ind"] == 1].index,
                df.loc[df["SELL_ind"] == 1,"price"].values, color='red', marker='v',
axarr[0].legend(loc='best')
axarr[1].plot(df["VALUE"],label="Algorithmic Gain",color='blue')
plt.grid(True)
```

```
plt.xlabel('Trade Days')
plt.show()
```



Algorithmic Trading Model and Decision Tree for Buy-Sell Signals

Let's create a class variable for positive and negative daily earnings. This variable will be our target variable. 0 will represent negative returns and 1 will represent positive returns.

In [34]:

```
df["target_cls"] = np.where(df["return"]>0,1,0)
```

In [35]:

```
cnt = pd.value_counts(df["target_cls"], sort = True)
cnt.plot(kind = 'bar', color=["red","green"])
plt.title("Target Class Distribution")
plt.xlabel("Target Class")
plt.ylabel("Frequency")
plt.show()
```

In [36]:
```
print("Negative Return $",sum(df["return"]<0))
print("Positive Return $",sum(df["return"]>0))
```

```
Negative Return $ 115
Positive Return $ 135
```

In [37]:
```
df["p_ema10"] = np.where(df["price"]>df["EMA10"],1,0)
df["ema10_ema30"] = np.where(df["EMA10"]>df["EMA30"],1,0)
df["macd_macds"] = np.where(df["MACD"]>df["MACD_signal"],1,0)
```

In [38]:
```
df.dropna(inplace=True)
```

In [39]:
```
predictors = ["p_ema10","ema10_ema30","macd_macds"]
```

In [40]:
```
X = df[predictors]
y = df["target_cls"]
```

In [41]:
```
cv = GapWalkForward(n_splits=5, gap_size=0, test_size=50)

dt = DecisionTreeClassifier()

param_grid = {"max_depth": np.arange(3, 30),
              "min_samples_split": range(10, 500,20),
              "criterion": ["gini", "entropy"]}
```

```
gs = GridSearchCV(dt, param_grid, cv=cv,n_jobs=-1,verbose=1).fit(X, y)
```

```
Fitting 5 folds for each of 1350 candidates, totalling 6750 fits
```

In [42]:
```python
gs.best_params_
```

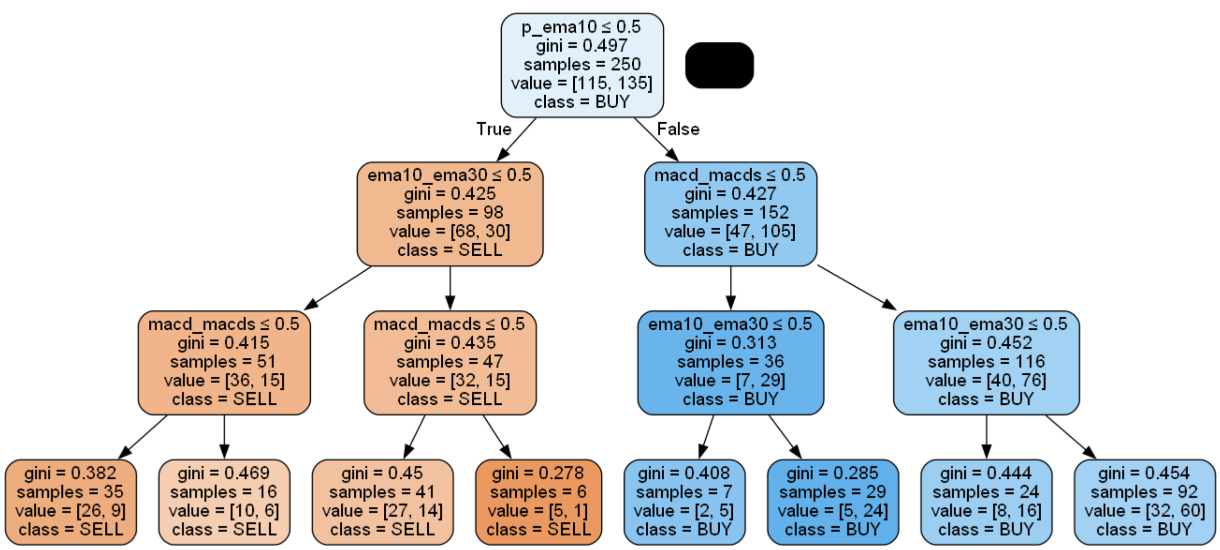
Out[42]:
```
{'criterion': 'gini', 'max_depth': 3, 'min_samples_split': 10}
```

In [43]:
```python
final_clf = gs.best_estimator_
```

In [44]:
```python
scores = cross_val_score(estimator=final_clf, X=X, y=y, cv=5)

print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

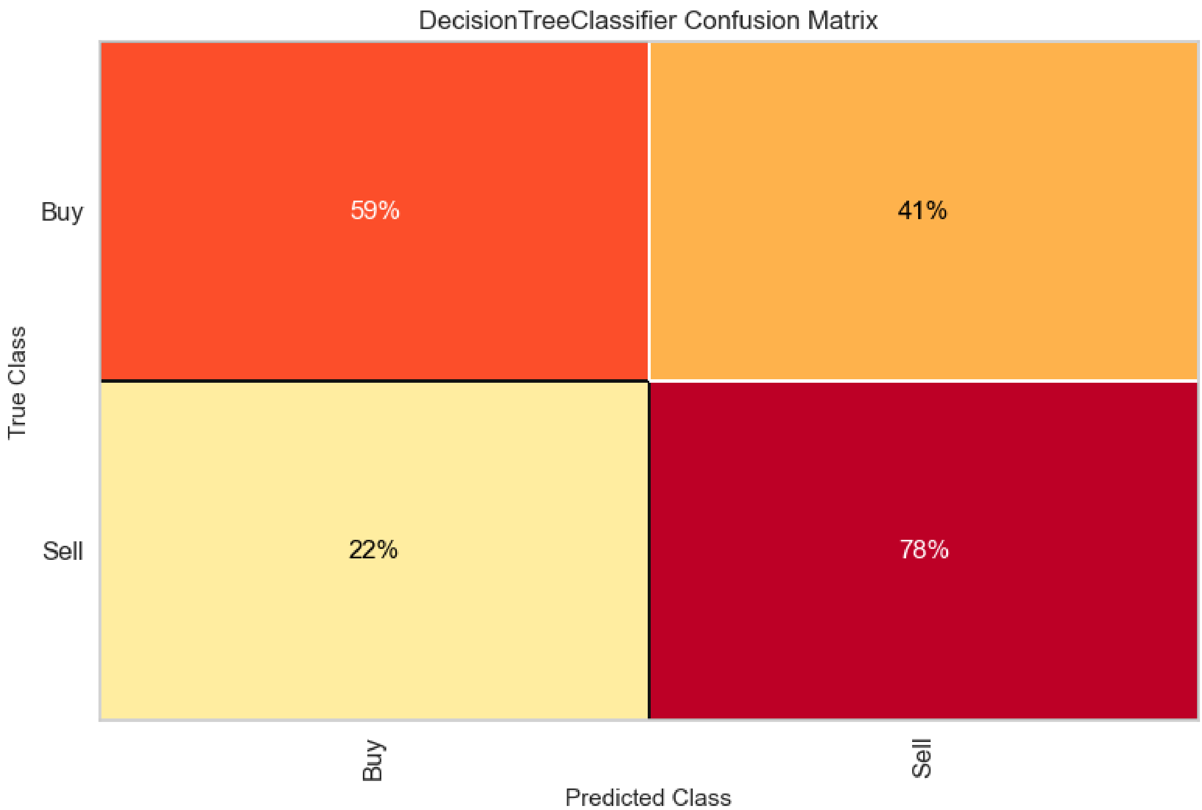
```
Accuracy: 0.67 (+/- 0.10)
```

In [45]:
```python
features = predictors
classes = {0:"SELL",1:"BUY"}

dot_data = StringIO()
export_graphviz(final_clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = features,class_names=classes
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tesla.png')

display(Image('tesla.png'))
```



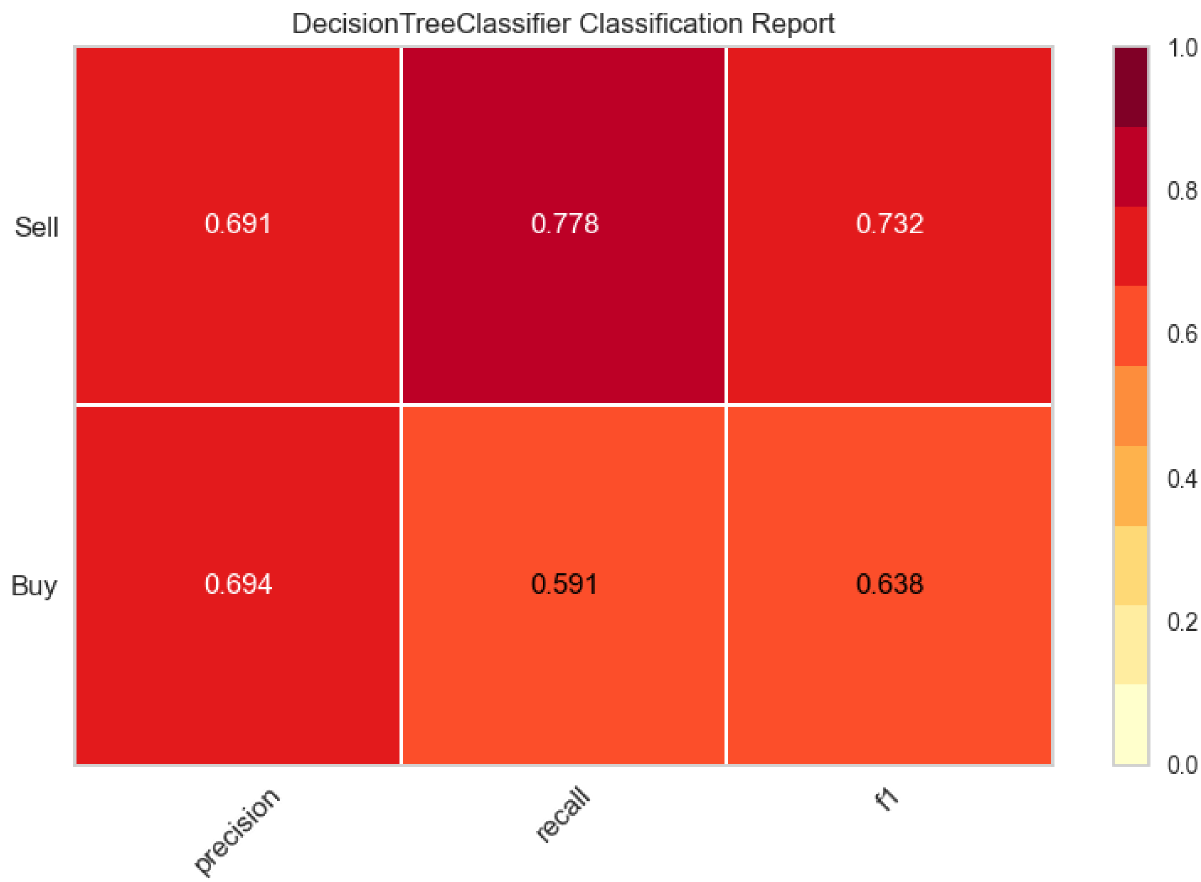
Model Performance

In [46]:
```python
category = ['Buy', 'Sell']

cm = ConfusionMatrix(final_clf, classes=category, percent=True)

cm.fit(X, y)
cm.score(X, y)
cm.poof()
```



Out[46]: <AxesSubplot: title={'center': 'DecisionTreeClassifier Confusion Matrix'}, xlabel='Predicted Class', ylabel='True Class'>

In [47]:
```
cr = ClassificationReport(final_clf, classes=category)

cr.fit(X, y)
cr.score(X, y)
cr.poof()
```

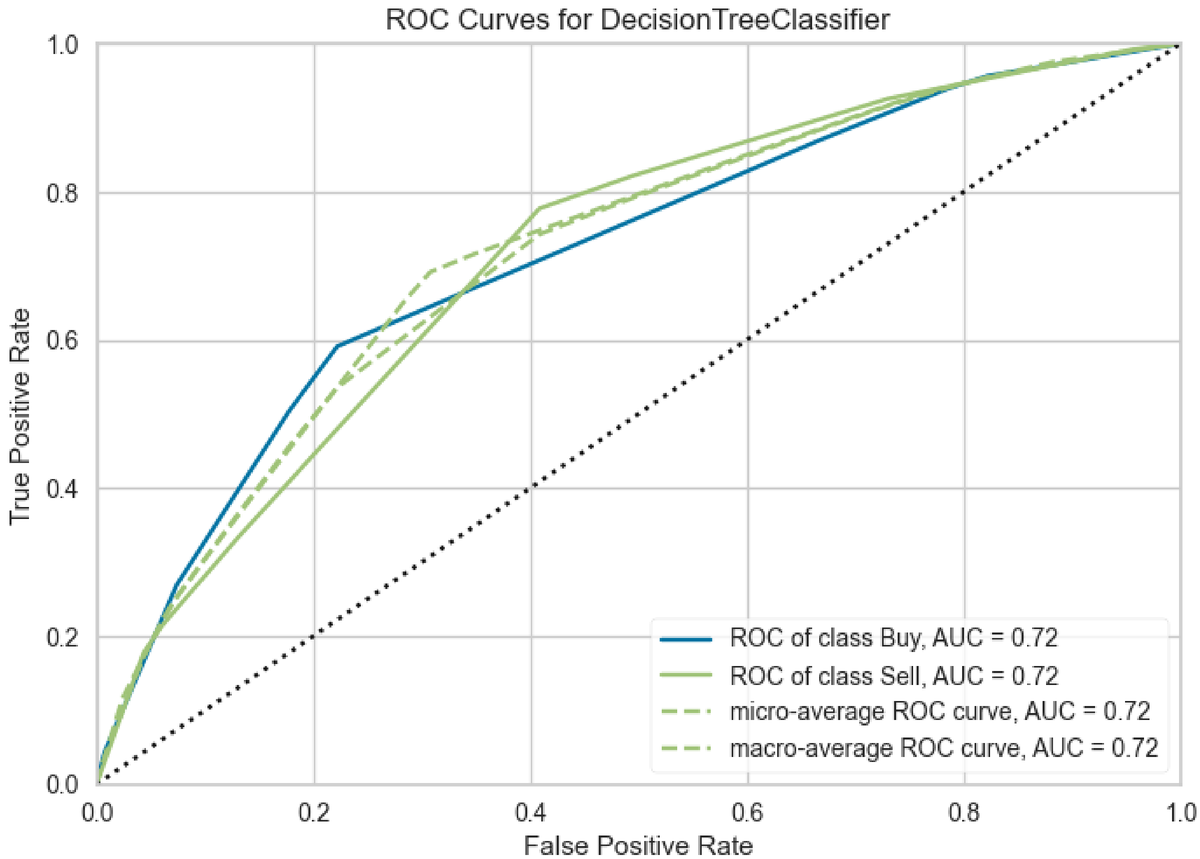


Out[47]: `<AxesSubplot: title={'center': 'DecisionTreeClassifier Classification Report'}>`

The model captures the sell signals better. This was due to the fact that there was some difference between the number of buy and sell signals in the data set.

In [48]:

```
rc = ROCAUC(final_clf, classes=category)

rc.fit(X, y)
rc.score(X, y)
rc.poof()
```



Out[48]:
```
<AxesSubplot: title={'center': 'ROC Curves for DecisionTreeClassifier'}, xlabel='Fal
se Positive Rate', ylabel='True Positive Rate'>
```

Retrospective Prediction

In [49]:
```
df["prediction_signal"] = final_clf.predict(X)
```

In [50]:
```
print("Accuracy of the model is ",accuracy_score(df["target_cls"],df["prediction_sig
```

```
Accuracy of the model is  0.692
```

Back-Test for Decision Tree

In [51]:
```
df["buy_dt"] = np.where((df["prediction_signal"]==1) &
                        (df["prediction_signal"].shift(1)==0),1,0)

df["sell_dt"] = np.where((df["prediction_signal"]==0) &
                         (df["prediction_signal"].shift(1)==1),1,0)

df["buy_dt_ind"] = np.where((df["buy_dt"] > df["buy_dt"].shift(1)),1,0)

df["sell_dt_ind"] = np.where((df["sell_dt"] > df["sell_dt"].shift(1)),1,0)
```

Buy-Sell Signals

In [52]:
```
fig1 = plt.figure(figsize=(14,8))
plt.plot(df["price"],label="Price",color='blue')
plt.scatter(df.loc[df["buy_dt_ind"] == 1].index,
            df.loc[df["buy_dt_ind"] == 1,"price"].values, color='green', marker='^',

plt.scatter(df.loc[df["sell_dt_ind"] == 1].index,
            df.loc[df["sell_dt_ind"] == 1,"price"].values, color='red', marker='v',

plt.legend(loc='best')
plt.grid(True)
plt.xlabel('Trade Days')
plt.title('TESLA \n Decision Tree Classifier \n Buy-Sell Signals', fontsize=20)
plt.show()
```

The profit of the trader who trades in the period related to the Decision Tree Model:

In [53]:
```
df["value_dt"] = 1000*(1 + (np.where(df["buy_dt"]==1,
                                      0.95*df["return_pct"],0)).cumsum())
```

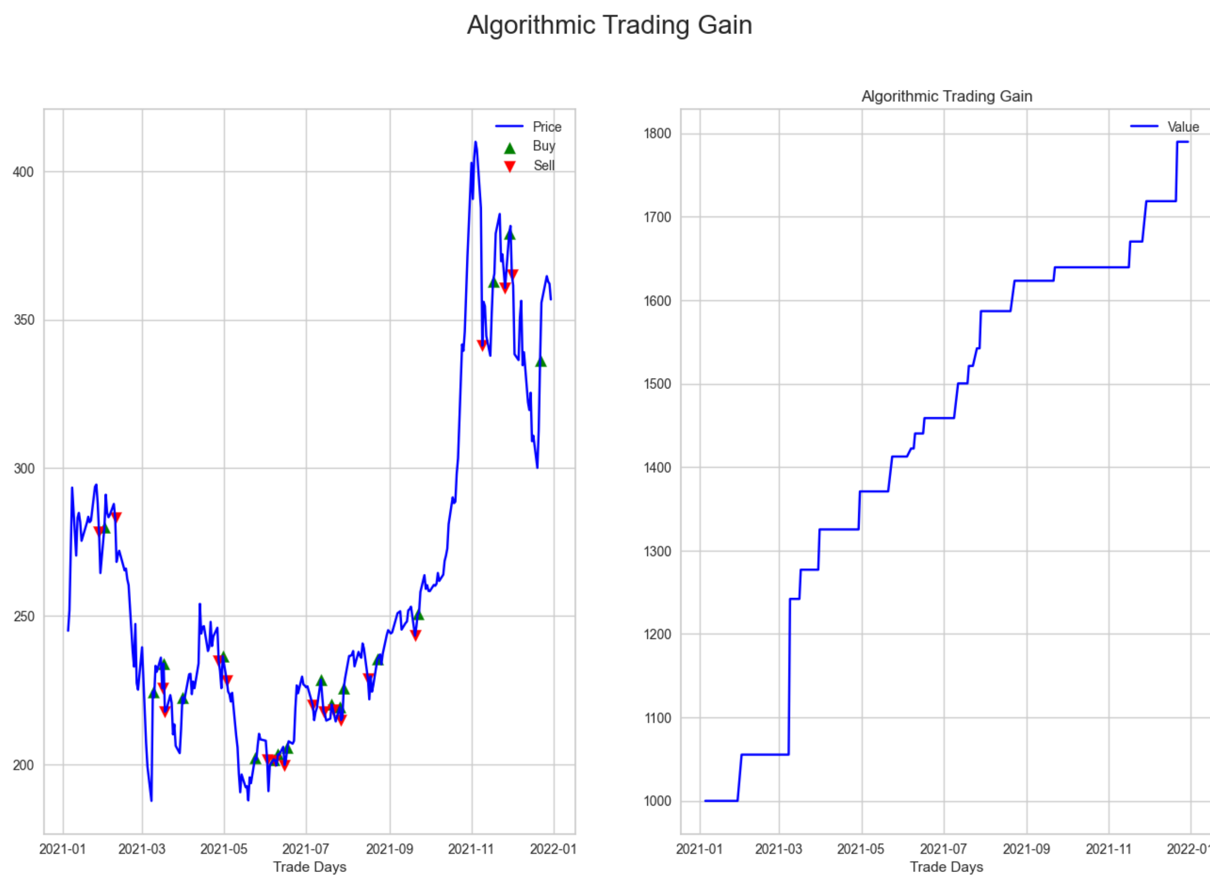
In [54]:
```
print("************* Descriptive Statistics *************")
print("Period",len(df),"days")
print("Highest Daily Loss ",100*round(df["return_pct"].min(),2),"%")
print("Highest Daily Return ",100*round(df["return_pct"].max(),2),"%")
print("Standard Deviation of Return ",100*round(df["return_pct"].std(),2),"%")
print("Total Potential Return ",100*(round(sum(np.where((df["return_pct"]>0),df["ret
```

```
print("Total Potential Loss ",100*(round(sum(np.where((df["return_pct"]<0),df["retur
print("Net Return ",100*df["return_pct"].sum().round(2),"%")

print("************* MODEL PERFORMANCE *************")

print("Return Captured by the Model ",100*sum(np.where((df["buy_dt"]==1),df["return_
print("Loss Maintained by the Model ",100*sum(np.where((df["sell_dt"]==1),df["returr
print("**************************************************")

print("************* REPORT *************")
print("The end-of-period price of the stock, which was $",df["price"][0].round(2),
    "at the beginning of the period, became $",df["price"][-1].round(2),"with %",
    (100*(df["price"][-1]-df["price"][0])/df["price"][0]).round(2),"change","The moc
    100*(sum(np.where((df["buy_dt"]==1),df["return_pct"],0))/sum(np.where((df["retur
                                                                 df["return_pct"]
    "of the total positive return.The investment of $1000 at the beginning of the pe
    df["value_dt"][-1].round(2),
    "on the first",len(df),"days.")
```

```
************* Descriptive Statistics *************
Period 250 days
Highest Daily Loss  -12.0 %
Highest Daily Return  20.0 %
Standard Deviation of Return  3.0 %
Total Potential Return  336.0 %
Total Potential Loss  -283.0 %
Net Return  53.0 %
************* MODEL PERFORMANCE *************
Return Captured by the Model  83.0 %
Loss Maintained by the Model  -66.0 %
**************************************************
************* REPORT *************
The end-of-period price of the stock, which was $ 245.04 at the beginning of the per
iod, became $ 356.78 with % 45.6 change The model captured % 25.0 of the total posit
ive return.The investment of $1000 at the beginning of the period became $ 1789.78 o
n the first 250 days.
```

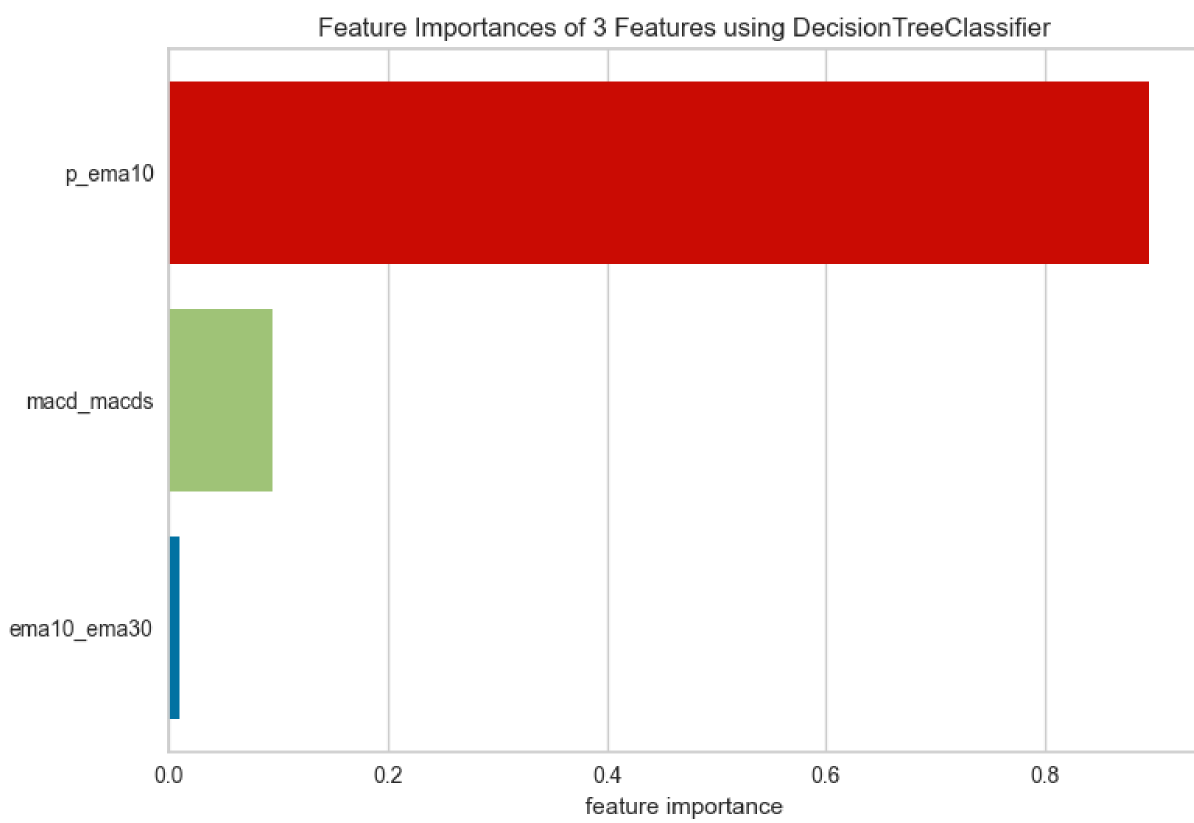
In [55]:

```
f,axarr = plt.subplots(1,2,figsize=(16,10))
f.suptitle('Algorithmic Trading Gain', fontsize=20)
axarr[0].plot(df["price"],label="Price",color='blue')
axarr[0].scatter(df.loc[df["buy_dt_ind"] == 1].index,
            df.loc[df["buy_dt_ind"] == 1,"price"].values, color='green', marker='^',

axarr[0].scatter(df.loc[df["sell_dt_ind"] == 1].index,
            df.loc[df["sell_dt_ind"] == 1,"price"].values, color='red', marker='v',

axarr[0].legend(loc='best')
axarr[0].grid(True)
axarr[0].set_xlabel('Trade Days')

axarr[1].plot(df["value_dt"],label="Value",color='blue')
axarr[1].legend(loc='best')
axarr[1].grid(True)
axarr[1].set_title('Algorithmic Trading Gain')
axarr[1].set_xlabel('Trade Days')
plt.show()
```



In [56]:

```
viz = FeatureImportances(final_clf, classes=category, relative=False)

viz.fit(X, y)
viz.poof()
```



Out[56]:
```
<AxesSubplot: title={'center': 'Feature Importances of 3 Features using DecisionTree Classifier'}, xlabel='feature importance'>
```

Clearly, two of the three strategies seem to be important, while the strategy EMA10>EMA30 has a negligible effect. The MACD>MACDS strategy does not seem to have a significant impact on this model either. When Price>EMA10 gives a buy/sell signal, the other two strategies don't matter much from a decision point of view.