In [2]:

```
!pip install ta
```

```
Collecting ta
  Downloading ta-0.11.0.tar.gz (25 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: numpy in c:\users\aditya kudva\anaconda3\lib\site-packages (from ta)
(1.24.3)
Requirement already satisfied: pandas in c:\users\aditya kudva\anaconda3\lib\site-packages (from ta)
(1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\aditya kudva\anaconda3\lib\site-pack
ages (from pandas->ta) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\aditya kudva\anaconda3\lib\site-packages (from
pandas->ta) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\aditya kudva\anaconda3\lib\site-packages (from pyt
hon-dateutil>=2.8.1->pandas->ta) (1.16.0)
Building wheels for collected packages: ta
  Building wheel for ta (setup.py): started
  Building wheel for ta (setup.py): finished with status 'done'
  Created wheel for ta: filename=ta-0.11.0-py3-none-any.whl size=29422 sha256=0a3d6b67f776f2ea472a7ff4
6b666b280eaa1914cfa6b6c4b7cdad55531d19e8
  Stored in directory: c:\users\aditya kudva\appdata\local\pip\cache\wheels\a1\d7\29\7781cc5eb9a3659d0
32d7d15bdd0f49d07d2b24fec29f44bc4
Successfully built ta
Installing collected packages: ta
Successfully installed ta-0.11.0
```

In [9]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import ta
import warnings
warnings.filterwarnings('ignore')
```

In [5]:

```
df = pd.read_csv('AAPL.csv', parse_dates=['Date'], index_col='Date')
df
```

Out[5]:

	Open Price	High Price	Low Price	Close Price	Adj Close Price	Volume
Date						
2014-05-27	87.982857	89.408569	87.947144	89.375717	80.948952	87216500
2014-05-28	89.431427	89.975716	89.111427	89.144287	80.739334	78870400
2014-05-29	89.692856	90.981430	89.681427	90.768570	82.210480	94118500
2014-05-30	91.139999	92.024284	89.842857	90.428574	81.902557	141005200
2014-06-02	90.565712	90.690002	88.928574	89.807144	81.339699	92337700
...	...	...	...	...	...	...
2020-05-18	313.170013	316.500000	310.320007	314.959991	314.959991	33843100
2020-05-19	315.029999	318.519989	313.010010	313.140015	313.140015	25432400
2020-05-20	316.679993	319.519989	316.519989	319.230011	319.230011	27876200
2020-05-21	318.660004	320.890015	315.869995	316.850006	316.850006	25672200
2020-05-22	315.769989	319.230011	315.350006	318.890015	318.890015	20430600

1510 rows × 6 columns

In [10]:

```
print(df.head())
```

```
            Open Price  High Price  Low Price  Close Price  Adj Close Price  \
Date
2014-05-27   87.982857   89.408569  87.947144    89.375717        80.948952
2014-05-28   89.431427   89.975716  89.111427    89.144287        80.739334
2014-05-29   89.692856   90.981430  89.681427    90.768570        82.210480
2014-05-30   91.139999   92.024284  89.842857    90.428574        81.902557
2014-06-02   90.565712   90.690002  88.928574    89.807144        81.339699

               Volume
Date
2014-05-27   87216500
2014-05-28   78870400
2014-05-29   94118500
2014-05-30  141005200
2014-06-02   92337700
```

In [11]:

```
print(df.columns)
```

```
Index(['Open Price', 'High Price', 'Low Price', 'Close Price',
       'Adj Close Price', 'Volume'],
      dtype='object')
```

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
Open Price         0
High Price         0
Low Price          0
Close Price        0
Adj Close Price    0
Volume             0
dtype: int64
```

In [7]:

```
df.fillna(method='ffill', inplace=True)
```

In [15]:

```
df['SMA_50'] = ta.trend.sma_indicator(df['Close Price'], window=50)
df['SMA_200'] = ta.trend.sma_indicator(df['Close Price'], window=200)
df['RSI'] = ta.momentum.rsi(df['Close Price'], window=14)
df['MACD'] = ta.trend.macd_diff(df['Close Price'])
```

In [16]:

```
df['Future Close'] = df['Close Price'].shift(-1)
df['Target'] = (df['Future Close'] > df['Close Price']).astype(int)
```

In [17]:

```
df.dropna(inplace=True)
```

In [18]:

```
X = df[['SMA_50', 'SMA_200', 'RSI', 'MACD']]
y = df['Target']
```

In [19]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
X_train, X_test, y_train, y_test
```

Out[19]:
```
(              SMA_50     SMA_200        RSI       MACD
 Date
 2015-03-11  118.9348  105.731307  43.638948 -1.284230
 2015-03-12  119.1456  105.906679  49.410402 -1.247728
 2015-03-13  119.3670  106.078907  47.377260 -1.237522
 2015-03-16  119.6584  106.249814  50.823398 -1.099838
 2015-03-17  120.0126  106.432871  55.632029 -0.840376
 ...              ...         ...        ...       ...
 2019-05-02  190.4058  192.164050  67.636408 -0.201529
 2019-05-03  191.2196  192.265550  70.432564  0.003276
 2019-05-06  191.9298  192.355950  63.053939 -0.123348
 2019-05-07  192.5024  192.410850  52.813487 -0.600758
 2019-05-08  193.0738  192.468150  52.872143 -0.910430

 [1048 rows x 4 columns],
                  SMA_50     SMA_200        RSI       MACD
 Date
 2019-05-09  193.590800  192.513700  49.276357 -1.238334
 2019-05-10  194.071400  192.534600  44.038720 -1.644446
 2019-05-13  194.286400  192.489100  32.131856 -2.579975
 2019-05-14  194.542600  192.461350  36.849120 -2.873436
 2019-05-15  194.850400  192.461050  40.285105 -2.785926
 ...                ...         ...        ...       ...
 2020-05-15  273.114000  263.209399  61.940977  1.025197
 2020-05-18  273.632599  263.742049  65.697653  0.996037
 2020-05-19  274.572000  264.287649  63.990111  0.722853
 2020-05-20  275.249800  264.917100  67.073985  0.814189
 2020-05-21  276.078200  265.516350  64.740544  0.586820

 [262 rows x 4 columns],
 Date
 2015-03-11    1
 2015-03-12    0
 2015-03-13    1
 2015-03-16    1
 2015-03-17    1
              ..
 2019-05-02    1
 2019-05-03    0
 2019-05-06    0
 2019-05-07    1
 2019-05-08    0
 Name: Target, Length: 1048, dtype: int32,
 Date
 2019-05-09    0
 2019-05-10    0
 2019-05-13    1
 2019-05-14    1
 2019-05-15    0
              ..
 2020-05-15    1
 2020-05-18    0
 2020-05-19    1
 2020-05-20    0
 2020-05-21    1
 Name: Target, Length: 262, dtype: int32)
```

In [20]:
```python
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[20]:
```
▾         RandomForestClassifier

RandomForestClassifier(random_state=42)
```

In [21]:

```
y_pred = model.predict(X_test)
print(f'Accuracy: {accuracy_score(y_test, y_pred):.2f}')
print('Classification Report:')
print(classification_report(y_test, y_pred))
```

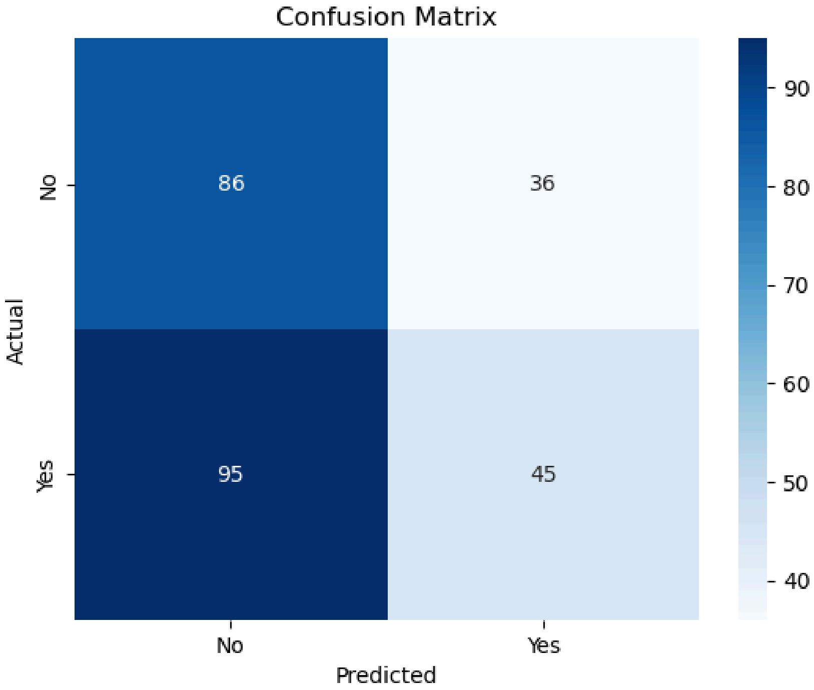
```
Accuracy: 0.50
Classification Report:
              precision    recall  f1-score   support

           0       0.48      0.70      0.57       122
           1       0.56      0.32      0.41       140

    accuracy                           0.50       262
   macro avg       0.52      0.51      0.49       262
weighted avg       0.52      0.50      0.48       262
```

In [22]:

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'], yticklabels=['No', 'Yes']
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



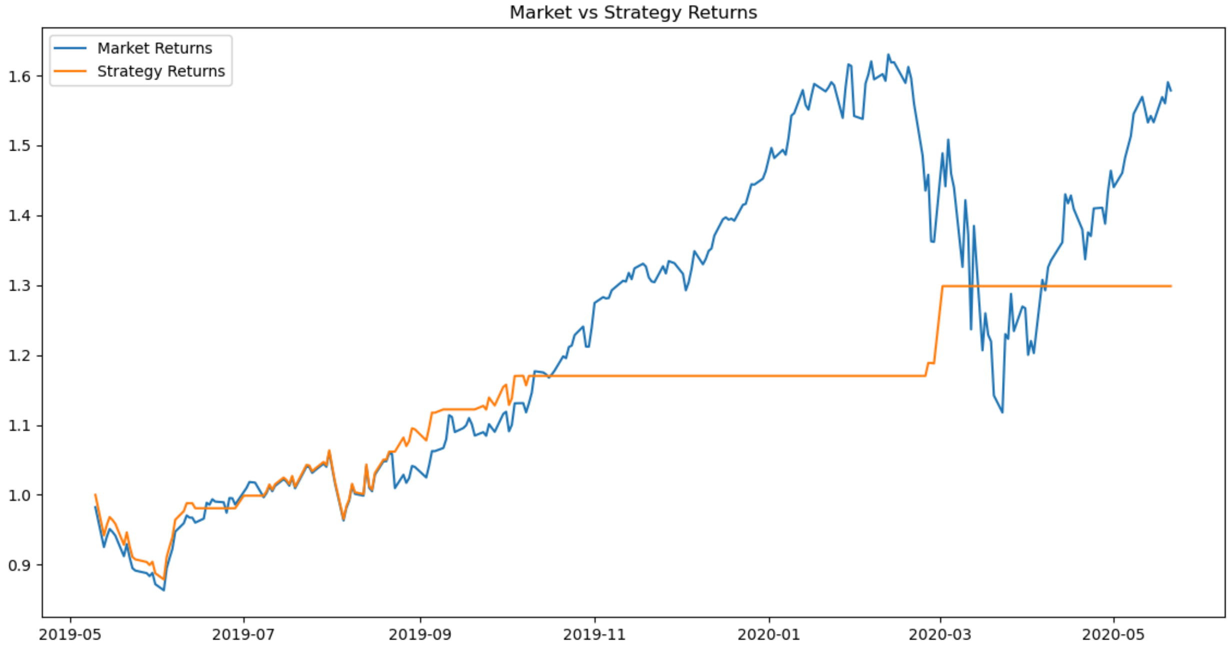
In [23]:

```
test_df = df.iloc[len(X_train):].copy()
test_df['Predicted Signal'] = y_pred
test_df['Strategy Returns'] = test_df['Close Price'].pct_change() * test_df['Predicted Signal'].shift(1
```

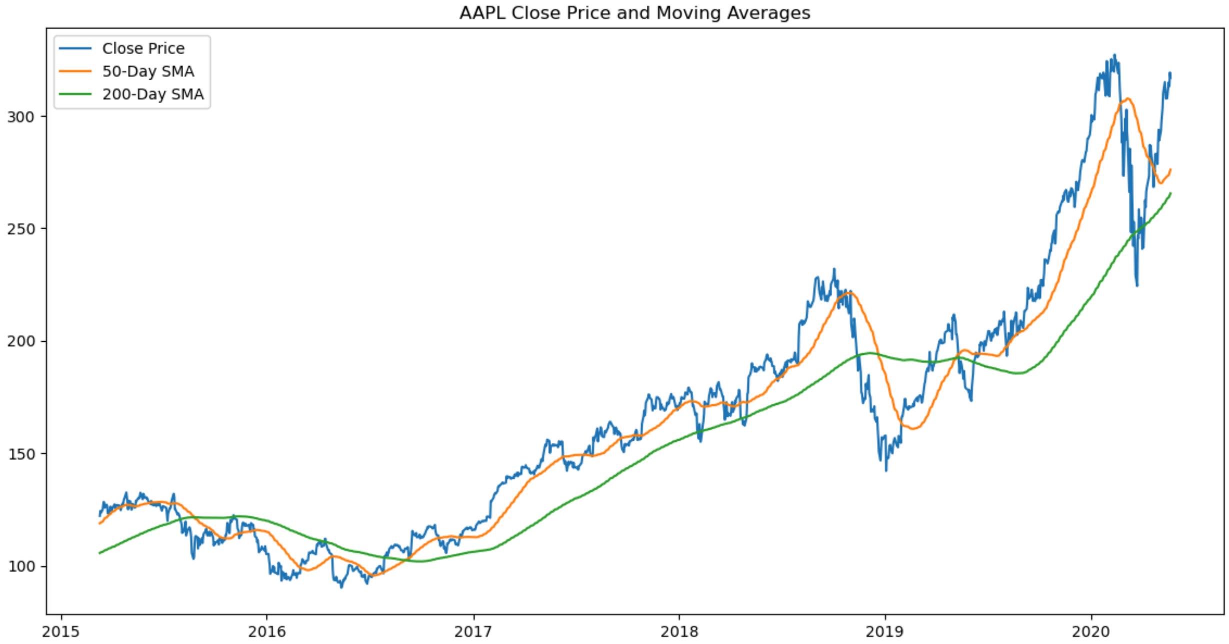
In [24]:

```
test_df['Cumulative Market Returns'] = (1 + test_df['Close Price'].pct_change()).cumprod()
test_df['Cumulative Strategy Returns'] = (1 + test_df['Strategy Returns']).cumprod()
```

```
In [25]: plt.figure(figsize=(14, 7))
         plt.plot(test_df['Cumulative Market Returns'], label='Market Returns')
         plt.plot(test_df['Cumulative Strategy Returns'], label='Strategy Returns')
         plt.legend()
         plt.title('Market vs Strategy Returns')
         plt.show()
```

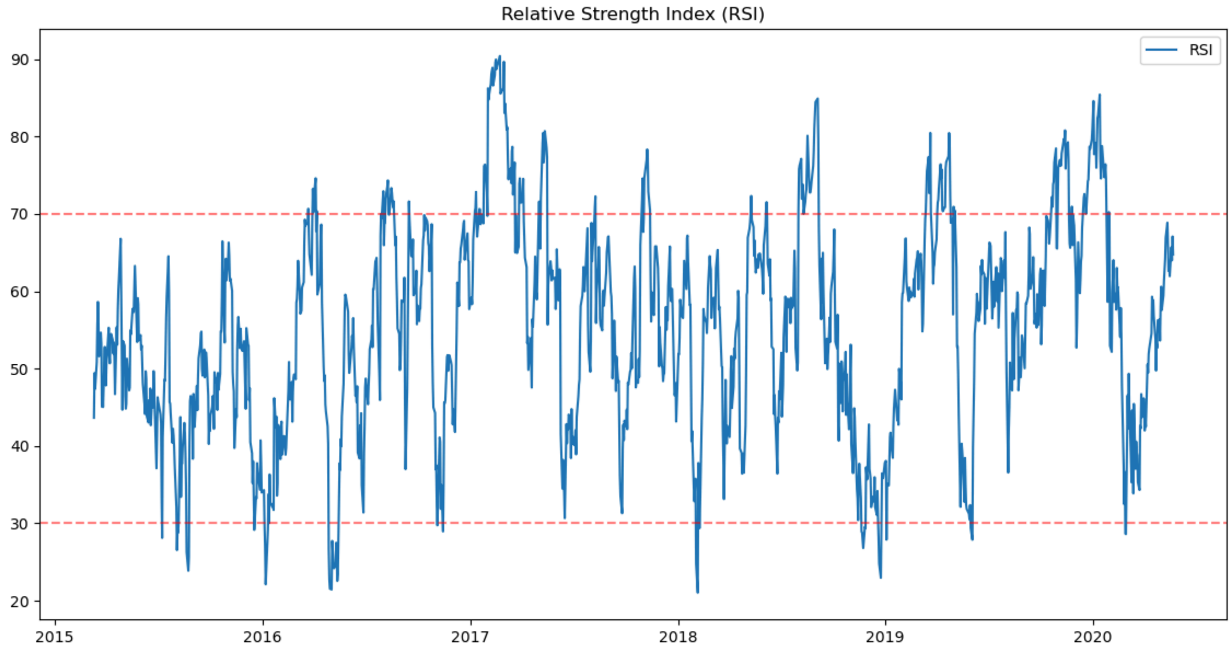


```
In [26]: plt.figure(figsize=(14, 7))
         plt.plot(df['Close Price'], label='Close Price')
         plt.plot(df['SMA_50'], label='50-Day SMA')
         plt.plot(df['SMA_200'], label='200-Day SMA')
         plt.legend()
         plt.title('AAPL Close Price and Moving Averages')
         plt.show()
```



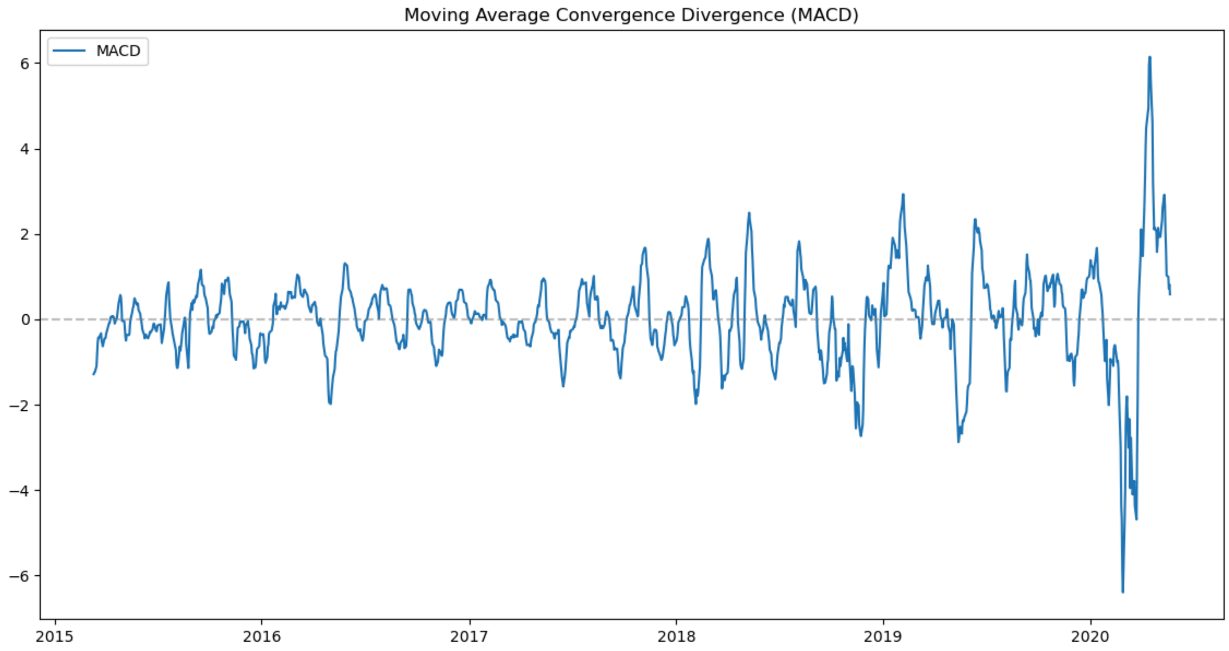
In [27]:

```
plt.figure(figsize=(14, 7))
plt.plot(df['RSI'], label='RSI')
plt.axhline(30, linestyle='--', alpha=0.5, color='red')
plt.axhline(70, linestyle='--', alpha=0.5, color='red')
plt.title('Relative Strength Index (RSI)')
plt.legend()
plt.show()
```



In [28]:

```
plt.figure(figsize=(14, 7))
plt.plot(df['MACD'], label='MACD')
plt.axhline(0, linestyle='--', alpha=0.5, color='grey')
plt.title('Moving Average Convergence Divergence (MACD)')
plt.legend()
plt.show()
```

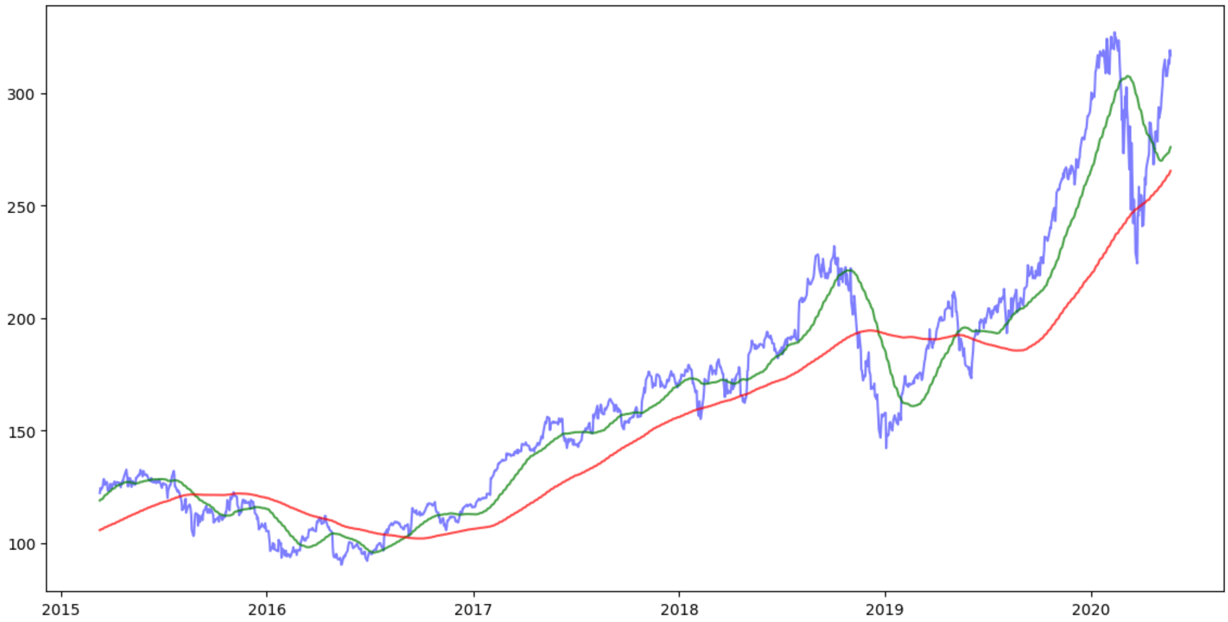


In [29]:

```
import matplotlib.dates as mdates
import matplotlib.ticker as mticker
from matplotlib.dates import DateFormatter
```

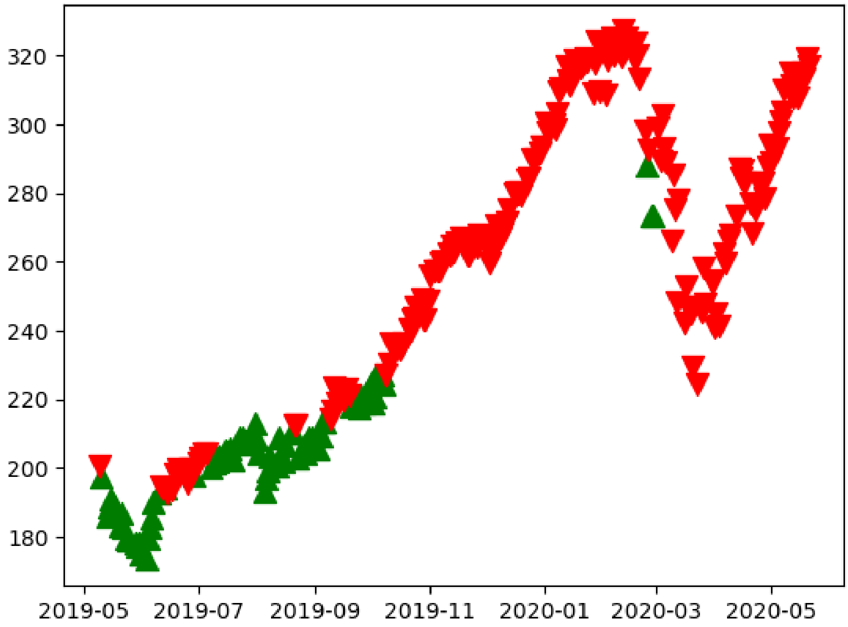
```
In [30]: plt.figure(figsize=(14, 7))
         plt.plot(df.index, df['Close Price'], label='Close Price', color='blue', alpha=0.5)
         plt.plot(df.index, df['SMA_50'], label='50-Day SMA', color='green', alpha=0.7)
         plt.plot(df.index, df['SMA_200'], label='200-Day SMA', color='red', alpha=0.7)
```

```
Out[30]: [<matplotlib.lines.Line2D at 0x1df39976a50>]
```



```
In [31]: buy_signals = test_df[test_df['Predicted Signal'] == 1]
         sell_signals = test_df[test_df['Predicted Signal'] == 0]
         plt.scatter(buy_signals.index, df.loc[buy_signals.index]['Close Price'], marker='^', color='green', alp
         plt.scatter(sell_signals.index, df.loc[sell_signals.index]['Close Price'], marker='v', color='red', alp
```

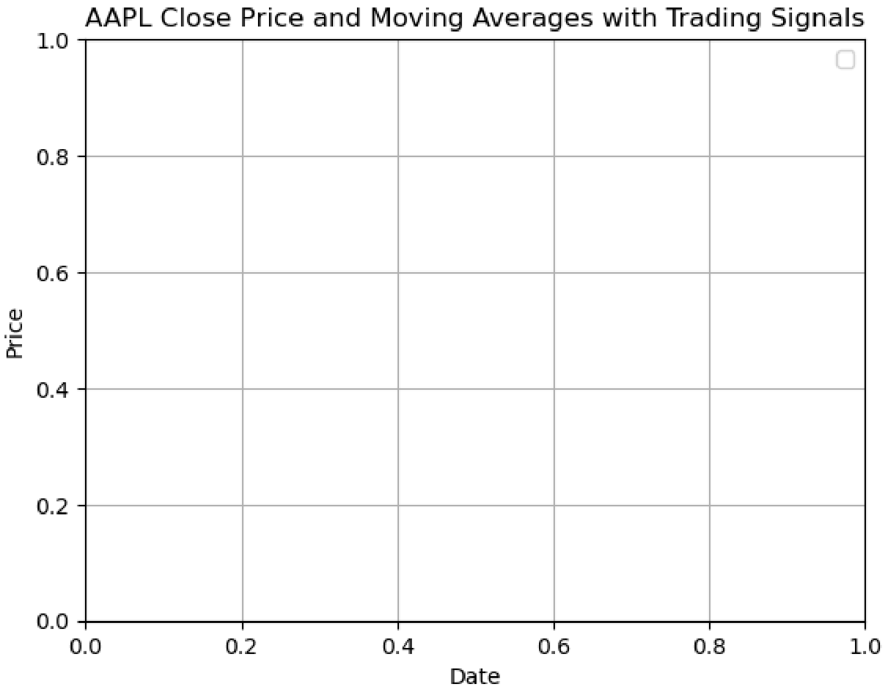
```
Out[31]: <matplotlib.collections.PathCollection at 0x1df39a04290>
```



In [32]:

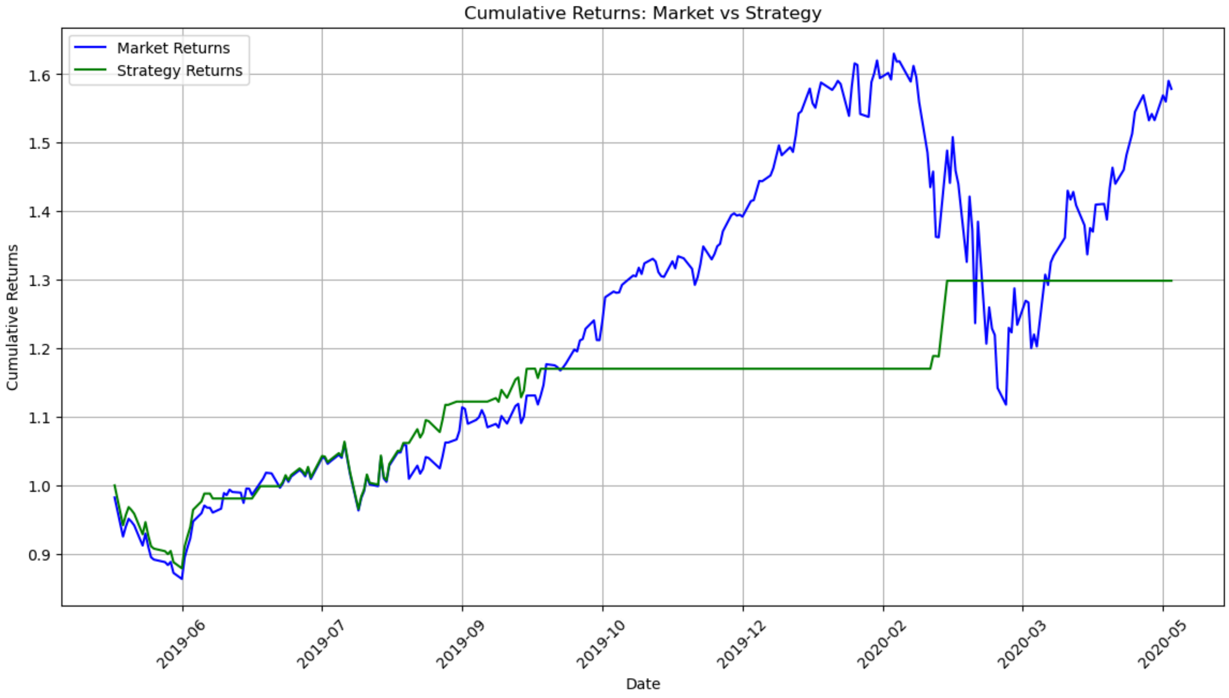
```
plt.legend()
plt.title('AAPL Close Price and Moving Averages with Trading Signals')
plt.xlabel('Date')
plt.ylabel('Price')
plt.grid(True)
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore
are ignored when legend() is called with no argument.
```



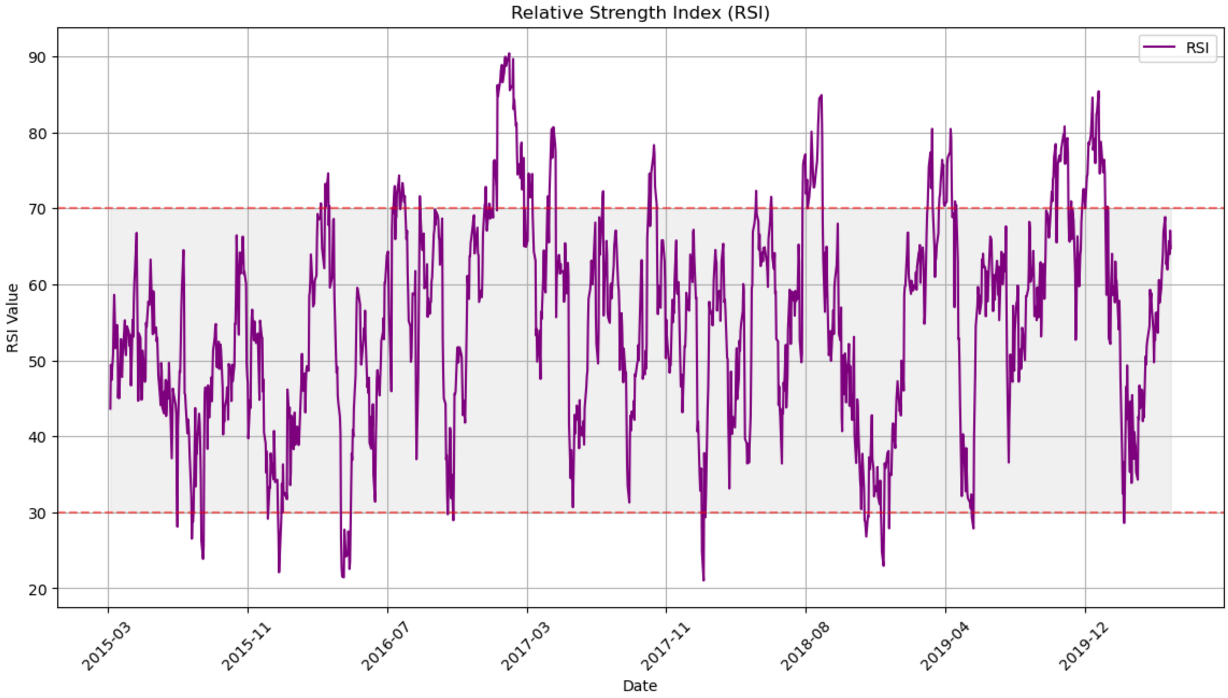
In [33]:

```
fig, ax = plt.subplots(figsize=(14, 7))
ax.plot(test_df.index, test_df['Cumulative Market Returns'], label='Market Returns', color='blue')
ax.plot(test_df.index, test_df['Cumulative Strategy Returns'], label='Strategy Returns', color='green')
ax.set_title('Cumulative Returns: Market vs Strategy')
ax.set_xlabel('Date')
ax.set_ylabel('Cumulative Returns')
ax.legend()
ax.grid(True)
ax.xaxis.set_major_locator(mticker.MaxNLocator(10))
ax.xaxis.set_major_formatter(DateFormatter('%Y-%m'))
plt.xticks(rotation=45)
plt.show()
```



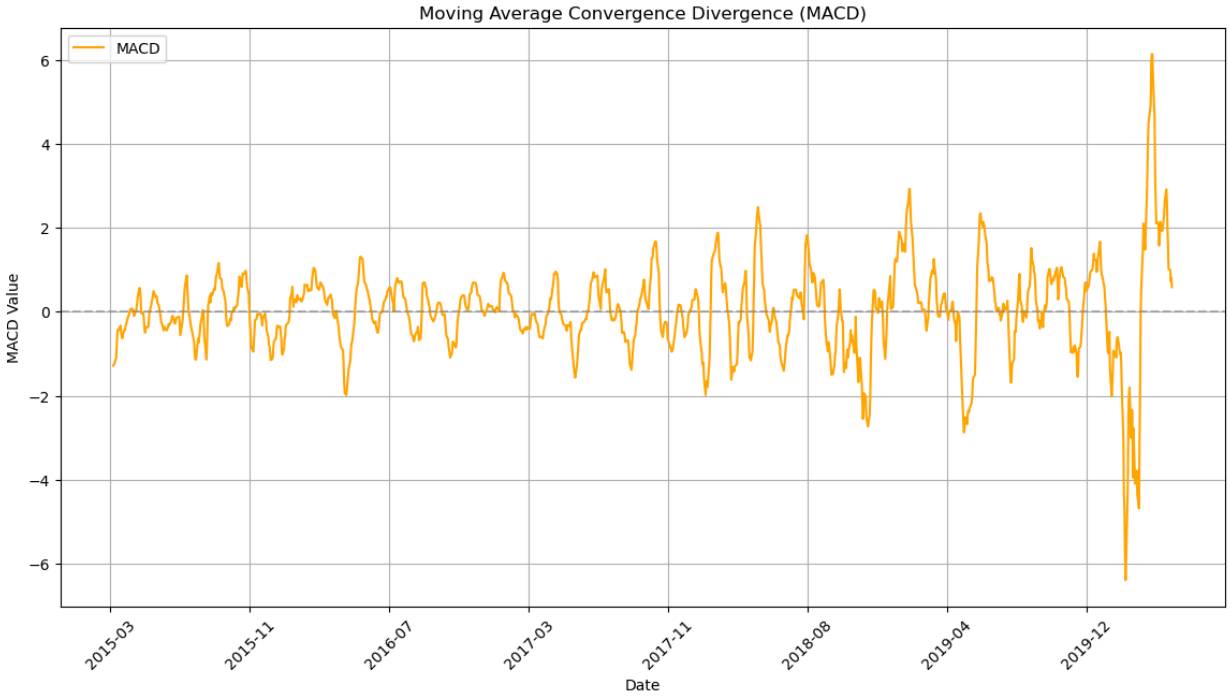
In [34]:

```
fig, ax = plt.subplots(figsize=(14, 7))
ax.plot(df.index, df['RSI'], label='RSI', color='purple')
ax.axhline(30, linestyle='--', alpha=0.5, color='red')
ax.axhline(70, linestyle='--', alpha=0.5, color='red')
ax.fill_between(df.index, y1=30, y2=70, alpha=0.1, color='grey')
ax.set_title('Relative Strength Index (RSI)')
ax.set_xlabel('Date')
ax.set_ylabel('RSI Value')
ax.legend()
ax.grid(True)
ax.xaxis.set_major_locator(mticker.MaxNLocator(10))
ax.xaxis.set_major_formatter(DateFormatter('%Y-%m'))
plt.xticks(rotation=45)
plt.show()
```



In [35]:

```
fig, ax = plt.subplots(figsize=(14, 7))
ax.plot(df.index, df['MACD'], label='MACD', color='orange')
ax.axhline(0, linestyle='--', alpha=0.5, color='grey')
ax.set_title('Moving Average Convergence Divergence (MACD)')
ax.set_xlabel('Date')
ax.set_ylabel('MACD Value')
ax.legend()
ax.grid(True)
ax.xaxis.set_major_locator(mticker.MaxNLocator(10))
ax.xaxis.set_major_formatter(DateFormatter('%Y-%m'))
plt.xticks(rotation=45)
plt.show()
```



In [37]:

```
!pip install mpf
```

```
Collecting mpf

ERROR: pip's dependency resolver does not currently take into account all the packages that are inst
alled. This behaviour is the source of the following dependency conflicts.
tables 3.8.0 requires blosc2~=2.0.0, which is not installed.
tables 3.8.0 requires cython>=0.29.21, which is not installed.
conda 23.7.2 requires ruamel-yaml<0.18,>=0.11.14, but you have ruamel-yaml 0.18.6 which is incompati
ble.
python-lsp-black 1.2.1 requires black>=22.3.0, but you have black 0.0 which is incompatible.


  Obtaining dependency information for mpf from https://files.pythonhosted.org/packages/c6/16/a77a69
be0090883f490e74303f966369176f940331a8ac5904e8977f3510/mpf-0.57.1-py3-none-any.whl.metadata (http
s://files.pythonhosted.org/packages/c6/16/a77a69be0090883f490e74303f966369176f940331a8ac5904e8977f35
10/mpf-0.57.1-py3-none-any.whl.metadata)
  Downloading mpf-0.57.1-py3-none-any.whl.metadata (5.8 kB)
Collecting asciimatics==1.15.0 (from mpf)
  Obtaining dependency information for asciimatics==1.15.0 from https://files.pythonhosted.org/packa
ges/35/bf/9cad857b630c840738003eb24c1adb63490a1024ec40a9dcc3a753300c38/asciimatics-1.15.0-py3-none-a
ny.whl.metadata (https://files.pythonhosted.org/packages/35/bf/9cad857b630c840738003eb24c1adb63490a1
```

In [39]:

```
!pip install mplfinance
```

```
Collecting mplfinance
  Obtaining dependency information for mplfinance from https://files.pythonhosted.org/packages/d7/d9/3
1c436ea7673c21a5bf3fc747bc7f63377582dfe845c3004d3e46f9deee0/mplfinance-0.12.10b0-py3-none-any.whl.meta
data (https://files.pythonhosted.org/packages/d7/d9/31c436ea7673c21a5bf3fc747bc7f63377582dfe845c3004d3
e46f9deee0/mplfinance-0.12.10b0-py3-none-any.whl.metadata)
  Downloading mplfinance-0.12.10b0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: matplotlib in c:\users\aditya kudva\anaconda3\lib\site-packages (from m
plfinance) (3.7.1)
Requirement already satisfied: pandas in c:\users\aditya kudva\anaconda3\lib\site-packages (from mplfi
nance) (1.5.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\aditya kudva\anaconda3\lib\site-packages
(from matplotlib->mplfinance) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\aditya kudva\anaconda3\lib\site-packages (from
matplotlib->mplfinance) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\aditya kudva\anaconda3\lib\site-packages
(from matplotlib->mplfinance) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\aditya kudva\anaconda3\lib\site-packages
(from matplotlib->mplfinance) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\users\aditya kudva\anaconda3\lib\site-packages (from
matplotlib->mplfinance) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\aditya kudva\anaconda3\lib\site-packages (f
rom matplotlib->mplfinance) (23.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\aditya kudva\anaconda3\lib\site-packages (fro
m matplotlib->mplfinance) (9.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\aditya kudva\anaconda3\lib\site-packages
(from matplotlib->mplfinance) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\aditya kudva\anaconda3\lib\site-packag
es (from matplotlib->mplfinance) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\aditya kudva\anaconda3\lib\site-packages (from
pandas->mplfinance) (2022.7)
Requirement already satisfied: six>=1.5 in c:\users\aditya kudva\anaconda3\lib\site-packages (from pyt
hon-dateutil>=2.7->matplotlib->mplfinance) (1.16.0)
Downloading mplfinance-0.12.10b0-py3-none-any.whl (75 kB)
   ---------------------------------------- 0.0/75.0 kB ? eta -:--:--
   -------------------------------------- - 71.7/75.0 kB 2.0 MB/s eta 0:00:01
   ---------------------------------------- 75.0/75.0 kB 1.0 MB/s eta 0:00:00
Installing collected packages: mplfinance
Successfully installed mplfinance-0.12.10b0
```

In [71]:

```
import mplfinance as mpf
import pandas as pd
```

In [72]:

```
df_candle = df[['Open Price', 'High Price', 'Low Price', 'Close Price', 'Volume']].copy()
df_candle.columns = ['Open', 'High', 'Low', 'Close', 'Volume']
```

In [73]:

```
buy_signals = test_df[test_df['Predicted Signal'] == 1].index
sell_signals = test_df[test_df['Predicted Signal'] == 0].index
```

In [74]:

```
buy_prices = df.loc[buy_signals, 'Close Price']
sell_prices = df.loc[sell_signals, 'Close Price']
```

In [75]:

```
buy_prices = buy_prices.reindex(df_candle.index, fill_value=pd.NA)
sell_prices = sell_prices.reindex(df_candle.index, fill_value=pd.NA)
```

In [65]:

```
print(f"df_candle columns: {df_candle.columns}")
print(f"df_candle index length: {len(df_candle.index)}")
```

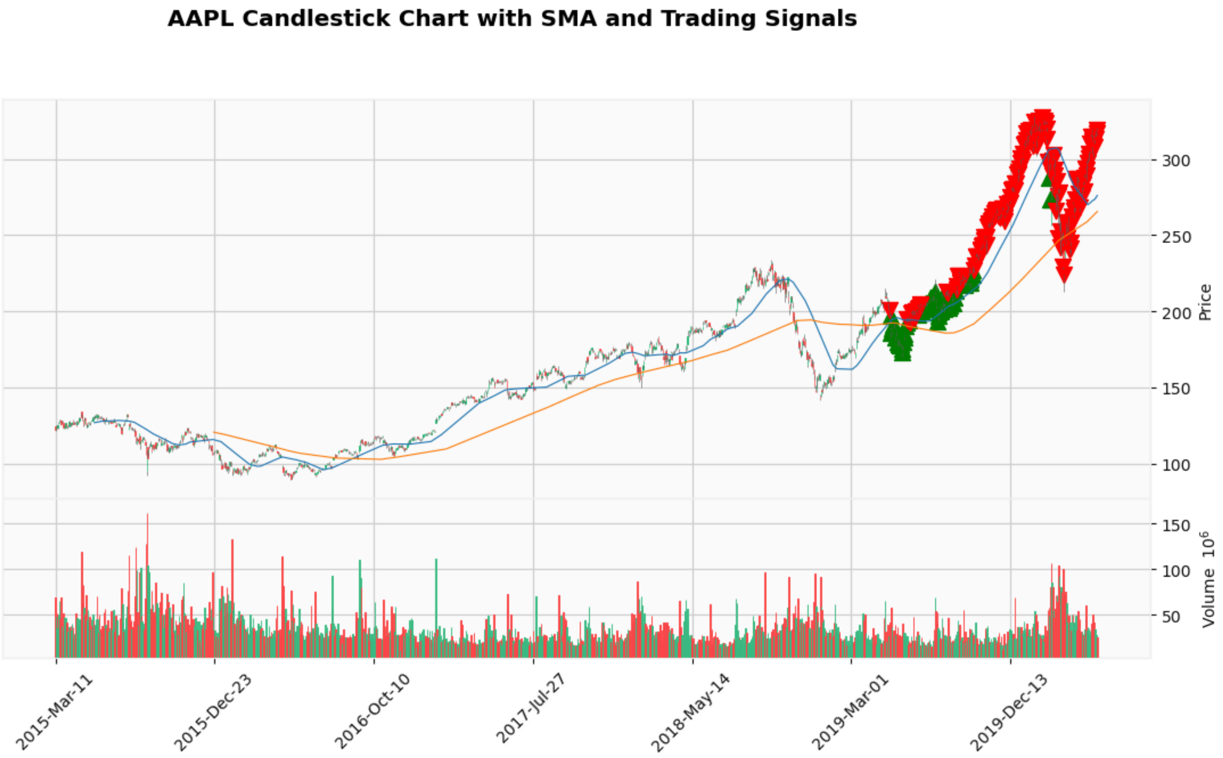
```
df_candle columns: Index(['Open', 'High', 'Low', 'Close', 'Volume'], dtype='object')
df_candle index length: 1310
```

In [76]:

```
addplot = [
    mpf.make_addplot(buy_prices, type='scatter', markersize=100, marker='^', color='g', panel=0),
    mpf.make_addplot(sell_prices, type='scatter', markersize=100, marker='v', color='r', panel=0)
]
```

In [77]:

```
mpf.plot(df_candle, type='candle', addplot=addplot,
         mav=(50, 200), volume=True, title='AAPL Candlestick Chart with SMA and Trading Signals',
         style='yahoo', figsize=(14, 7))
```



In [ ]: