In [62]:
```
import yfinance as yf
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

In [63]:
```
# pip install yfinance
```

In [64]:
```
stock_data = yf.download('AAPL', start='2020-01-01', end='2024-01-01')
```

```
[**********100%%**********]  1 of 1 completed
```

In [65]:
```
stock_data = stock_data.dropna()  # Drop rows with missing values
stock_data['Close_Shifted'] = stock_data['Close'].shift(-1)
```

In [66]:
```
stock_data['Daily_Return'] = (stock_data['Close'] - stock_data['Open'])
```

In [67]:
```
stock_data
```

Out[67]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.059425	135480400
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.349144	146322800
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.925636	118387200
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.582672	108872000
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.750244	132079200
...	...	...	...	...	...	...
2023-12-22	195.179993	195.410004	192.970001	193.600006	193.353287	37122800
2023-12-26	193.610001	193.889999	192.830002	193.050003	192.803986	28919300
2023-12-27	192.490005	193.500000	191.089996	193.149994	192.903839	48087700
2023-12-28	194.139999	194.660004	193.169998	193.580002	193.333298	34049900
2023-12-29	193.899994	194.399994	191.729996	192.529999	192.284637	42628800

1006 rows × 8 columns

In [68]:
```
stock_data.corr()
```

Out[68]:

	Open	High	Low	Close	Adj Close	Volume
Open	1.000000	0.999188	0.999027	0.997894	0.997758	-0.644620
High	0.999188	1.000000	0.998884	0.999008	0.998840	-0.635710
Low	0.999027	0.998884	1.000000	0.999073	0.998993	-0.656454
Close	0.997894	0.999008	0.999073	1.000000	0.999861	-0.646689
Adj Close	0.997758	0.998840	0.998993	0.999861	1.000000	-0.648157
Volume	-0.644620	-0.635710	-0.656454	-0.646689	-0.648157	1.000000
Close_Shifted	0.994706	0.995859	0.996203	0.996838	0.996769	-0.645884
Daily_Return	-0.030680	-0.000990	0.002478	0.034227	0.034171	-0.033021

In [69]:
```
stock_data.isnull().sum()
```

Out[69]:
```
Open             0
High             0
Low              0
Close            0
Adj Close        0
Volume           0
Close_Shifted    1
Daily_Return     0
dtype: int64
```

In [70]:
```
stock_data=stock_data.dropna()
stock_data
```

Out[70]:

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	74.059998	75.150002	73.797501	75.087502	73.059425	135480400
2020-01-03	74.287498	75.144997	74.125000	74.357498	72.349144	146322800
2020-01-06	73.447502	74.989998	73.187500	74.949997	72.925636	118387200
2020-01-07	74.959999	75.224998	74.370003	74.597504	72.582672	108872000
2020-01-08	74.290001	76.110001	74.290001	75.797501	73.750244	132079200
...	...	...	...	...	...	...
2023-12-21	196.100006	197.080002	193.500000	194.679993	194.431885	46482500
2023-12-22	195.179993	195.410004	192.970001	193.600006	193.353287	37122800
2023-12-26	193.610001	193.889999	192.830002	193.050003	192.803986	28919300
2023-12-27	192.490005	193.500000	191.089996	193.149994	192.903839	48087700
2023-12-28	194.139999	194.660004	193.169998	193.580002	193.333298	34049900

1005 rows × 8 columns

In [71]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [72]:
```
stock_data.isnull().sum()
```

Out[72]:
```
Open             0
High             0
Low              0
Close            0
Adj Close        0
Volume           0
Close_Shifted    0
Daily_Return     0
dtype: int64
```

In [73]:
```
features = ["Open","High","Low","Close","Adj Close","Volume","Close_Shifted"
for i in range(0,7):
    stock_data[[features[i]]]=scaler.fit_transform(stock_data[[features[i]]]
stock_data
```

```
C:\Users\sneha\AppData\Local\Temp\ipykernel_10380\2896114574.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  stock_data[[features[i]]]=scaler.fit_transform(stock_data[[features[i]]])
C:\Users\sneha\AppData\Local\Temp\ipykernel_10380\2896114574.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  stock_data[[features[i]]]=scaler.fit_transform(stock_data[[features[i]]])
C:\Users\sneha\AppData\Local\Temp\ipykernel_10380\2896114574.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  stock_data[[features[i]]]=scaler.fit_transform(stock_data[[features[i]]])
C:\Users\sneha\AppData\Local\Temp\ipykernel_10380\2896114574.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  stock_data[[features[i]]]=scaler.fit_transform(stock_data[[features[i]]])
C:\Users\sneha\AppData\Local\Temp\ipykernel_10380\2896114574.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  stock_data[[features[i]]]=scaler.fit_transform(stock_data[[features[i]]])
C:\Users\sneha\AppData\Local\Temp\ipykernel_10380\2896114574.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  stock_data[[features[i]]]=scaler.fit_transform(stock_data[[features[i]]])
C:\Users\sneha\AppData\Local\Temp\ipykernel_10380\2896114574.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Loading [MathJax]/extensions/Safe.js

```
ta[[features[i]]]=scaler.fit_transform(stock_data[[features[i]]])
```

Out[73]:

Date	Open	High	Low	Close	Adj Close	Volume	Close_Shifte
2020-01-02	0.120851	0.126496	0.143520	0.133751	0.128204	0.276876	0.12861
2020-01-03	0.122465	0.126461	0.145797	0.128611	0.123242	0.303816	0.13278
2020-01-06	0.116507	0.125373	0.139279	0.132783	0.127269	0.234405	0.13030
2020-01-07	0.127234	0.127022	0.147500	0.130301	0.124873	0.210762	0.13875
2020-01-08	0.122482	0.133233	0.146944	0.138751	0.133030	0.268425	0.15008
...	...	...	...	...	...	...	
2023-12-21	0.986383	0.982175	0.975669	0.975848	0.976070	0.055742	0.96824
2023-12-22	0.979858	0.970455	0.971984	0.968243	0.968535	0.032486	0.96437
2023-12-26	0.968723	0.959788	0.971011	0.964371	0.964698	0.012103	0.96507
2023-12-27	0.960780	0.957051	0.958915	0.965075	0.965395	0.059731	0.96810
2023-12-28	0.972482	0.965192	0.973375	0.968103	0.968395	0.024851	0.96070

1005 rows × 8 columns

In [74]:
```
features = ['Open', 'High', 'Low', 'Close', 'Volume', 'Daily_Return']
X = stock_data[features]
y = stock_data['Close_Shifted']
```

In [75]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

In [76]:
```
X_train,y_train
```

Out[76]:
```
(                Open      High       Low     Close    Volume  Daily_Return
 Date
 2020-04-24  0.087092  0.095635  0.111907  0.103156  0.253721      1.442497
 2020-02-13  0.170408  0.171445  0.192461  0.176915  0.175667      0.169998
 2021-02-11  0.559433  0.556265  0.560437  0.556534  0.099964     -0.769989
 2022-01-05  0.869433  0.863504  0.844558  0.836710  0.175145     -4.690002
 2022-08-04  0.772979  0.772413  0.773580  0.772563  0.078084     -0.199997
 ...              ...       ...       ...       ...       ...           ...
 2020-06-04  0.170762  0.170392  0.187994  0.172426  0.157809     -0.517502
 2021-01-28  0.585106  0.595565  0.580806  0.570335  0.294619     -2.430008
 2023-06-02  0.879504  0.874803  0.876675  0.879170  0.094164     -0.080002
 2021-09-23  0.635674  0.631285  0.642955  0.638918  0.101351      0.180008
 2020-05-29  0.161649  0.162550  0.180504  0.164716  0.321730     -0.327499

 [804 rows x 6 columns],
 Date
 2020-04-24    0.103508
 2020-02-13    0.177056
 2021-02-11    0.558223
 2022-01-05    0.816149
 2022-08-04    0.769324
                  ...
 2020-06-04    0.188586
 2021-01-28    0.534212
 2023-06-02    0.869523
 2021-09-23    0.639551
 2020-05-29    0.171599
 Name: Close_Shifted, Length: 804, dtype: float64)
```

In [77]:
```
y_train.isnull().sum()
```

Out[77]:
```
0
```

In [78]:
```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[78]:
```
▼ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [79]:
```
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
rmse = mse**0.5
mse,rmse
```

Out[79]:
```
(0.0004229827478721637, 0.020566544383346555)
```

In [80]:
```
predictions
```

Out[80]:
```
array([0.85756825, 0.59632229, 0.65454679, 0.83022163, 0.51129192,
       0.77300965, 0.80031401, 0.81243721, 0.54704808, 0.09887197,
       0.96997834, 0.55488443, 0.56326933, 0.65937439, 0.53195344,
       0.63028148, 0.62930976, 0.95001461, 0.4528301 , 0.16482372,
       0.80746248, 0.62271199, 0.42104353, 0.4623233 , 0.69734432,
       0.56386926, 0.57090219, 0.5015734 , 0.60234549, 0.94701357,
       0.59419226, 0.64419527, 0.16553349, 0.49828103, 0.5456715 ,
       0.09392135, 0.86421603, 0.42188305, 0.51472741, 0.41759643,
       0.81268647, 0.84794268, 0.55502245, 0.86285987, 0.47650888,
       0.44242365, 0.6172831 , 0.88782156, 0.66203813, 0.85950253,
       0.54803316, 0.59739892, 0.04683737, 0.60137027, 0.76556832,
       0.66951187, 0.63931488, 0.05103421, 0.16893218, 0.17907108,
       0.916292  , 0.64700055, 0.16032304, 0.74280988, 0.74218739,
       0.74293942, 0.6142053 , 0.45621351, 0.87138558, 0.64033796,
       0.18270125, 0.46288663, 0.6741486 , 0.40880454, 0.55270444,
       0.7287792 , 0.65807102, 0.84512684, 0.05849192, 0.72475716,
       0.77612126, 0.49848645, 0.08244935, 0.56373055, 0.09110585,
       0.85586918, 0.84126817, 0.82005388, 0.42171353, 0.54804444,
       0.65218231, 0.85573751, 0.8191054 , 0.71726653, 0.4935985 ,
       0.4486926 , 0.38300615, 0.54316053, 0.15425951, 0.04548789,
       0.77943211, 0.76931787, 0.43248931, 0.59838823, 0.74870493,
       0.46468428, 0.68527189, 0.95415213, 0.80299471, 0.80710759,
       0.61666345, 0.95574629, 0.63747356, 0.96652595, 0.45219146,
       0.13996462, 0.64464378, 0.76355378, 0.44979668, 0.8263369 ,
       0.28683525, 0.91246853, 0.67996338, 0.82103544, 0.50742831,
       0.77461439, 0.54719948, 0.77577411, 0.61417361, 0.28577271,
       0.59959935, 0.79020894, 0.80378547, 0.59880297, 0.97558962,
       0.85999332, 0.03909995, 0.24515412, 0.8236837 , 0.16051649,
       0.10957329, 0.09992254, 0.55296003, 0.63778408, 0.83217138,
       0.9780696 , 0.55112787, 0.45989966, 0.74536095, 0.47667329,
       0.55445139, 0.91824105, 0.75575192, 0.47536923, 0.77822661,
       0.82620169, 0.81602761, 0.90308657, 0.46676994, 0.04768762,
       0.54976253, 0.82472232, 0.59582798, 0.72492192, 0.73923919,
       0.61410744, 0.64596545, 0.75283892, 0.67941624, 0.4551242 ,
       0.51192916, 0.74093827, 0.8527576 , 0.55438451, 0.22457074,
       0.6705385 , 0.64214902, 0.84689423, 0.51531748, 0.51595824,
       0.74000389, 0.63753201, 0.65096485, 0.55737428, 0.85853505,
       0.6101572 , 0.64688086, 0.30436337, 0.47717569, 0.85796471,
       0.45170209, 0.50065907, 0.63218407, 0.73442709, 0.49107539,
       0.79950217, 0.39805499, 0.69255195, 0.53641769, 0.37958263,
       0.49659513])
```

In [81]:
```
predictions_train = model.predict(X_train)
```

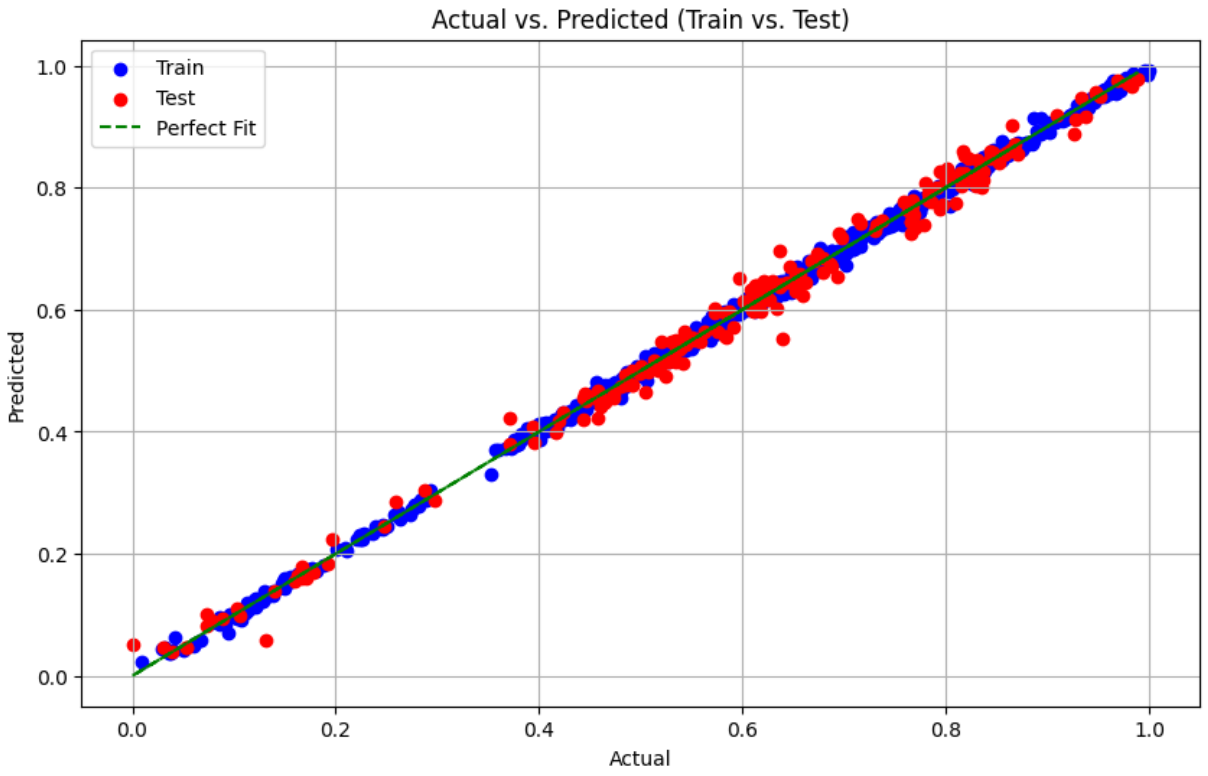
In [82]:
```
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
```

In [83]:
```
r2_train = r2_score(y_train, predictions_train)
print("Train R-squared Score:", r2_train)

r2_test = r2_score(y_test, predictions)
print("Test R-squared Score:", r2_test)
```

```
Train R-squared Score: 0.9990009152144345
Test R-squared Score: 0.9919370970936792
```

In [84]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_train, predictions_train, color='blue', label='Train')
plt.scatter(y_test, predictions, color='red', label='Test')
plt.plot(y_test, y_test, color='green', linestyle='--', label='Perfect Fit')
plt.title('Actual vs. Predicted (Train vs. Test)')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]:
```
```

In [ ]:
```
]
```