In [22]:
```
import pandas as pd
import numpy as np
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
import matplotlib.pyplot as plt
```

In [15]:
```
import yfinance as yf
import numpy as np

# Download stock data for AAPL
stock_data = yf.download("FDX", start="1990-01-01", end="2022-02-21")

# Calculate daily percentage change in closing price
scaled_data = np.array(stock_data["Close"].pct_change().dropna()).reshape

# Split data into training and test sets using sequential split
training_size = int(len(scaled_data) * 0.8)
training_data = scaled_data[:training_size]
testing_data = scaled_data[training_size:]

```

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

In [16]:
```
training_data
```

Out[16]:
```
array([[ 0.01298701],
       [-0.00769231],
       [-0.02583979],
       ...,
       [ 0.01009557],
       [-0.0056262 ],
       [ 0.02509488]])
```

In [17]:
```
testing_data
```

Out[17]:
```
array([[-0.02837659],
       [-0.00093564],
       [-0.02802865],
       ...,
       [-0.00603298],
       [-0.02026112],
       [-0.00953781]])
```

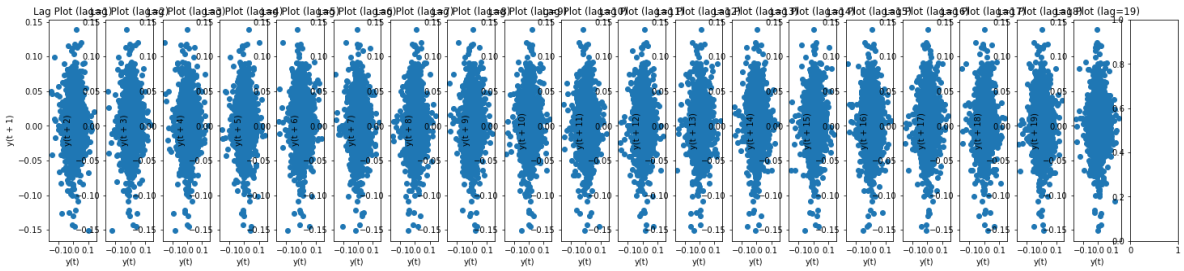
In [18]:
```
training_size
```

Out[18]:
```
6477
```

2.) Create your x_train/y_train data so that your RNN uses percentage change data to make a

binary forecast where the stock moves up or down the next day Build an RNN Architecture accordingly

In [29]:
```
scaled_data_series = pd.Series(scaled_data.squeeze())

fig, axs = plt.subplots(1, 20, figsize=(25, 5))

for x in range(1, 20):
    pd.plotting.lag_plot(scaled_data_series, lag=x, ax=axs[x-1])
    axs[x-1].set_title(f"Lag Plot (lag={x})")

plt.show()

```



In [31]:
```
x_train = []
y_train = []

input_size = 30
for i in range(input_size, len(train_data)):
    x_train.append(train_data[i-input_size:i, 0])
    y_train.append(train_data[i, 0])

y_train = np.array(y_train)
threshold = 0.0
y_train = np.where(y_train > threshold, 1, 0)

x_train = np.array(x_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

```

In [36]:

```
###################################
####Build Your RNN Architecture####
###################################

model = Sequential()
model.add(LSTM(x_train.shape[1], return_sequences=True, input_shape=(x_tra
model.add(LSTM(50, return_sequences=False))
model.add(Dense(1, activation='sigmoid'))


model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(x_train, y_train, batch_size=32, epochs=20)

```

```
Epoch 1/20
202/202 [==============================] - 8s 16ms/step - loss: 0.2498
Epoch 2/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2497
Epoch 3/20
202/202 [==============================] - 3s 17ms/step - loss: 0.2497
Epoch 4/20
202/202 [==============================] - 3s 17ms/step - loss: 0.2497
Epoch 5/20
202/202 [==============================] - 3s 17ms/step - loss: 0.2497
Epoch 6/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2497
Epoch 7/20
202/202 [==============================] - 3s 15ms/step - loss: 0.2496
Epoch 8/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2497
Epoch 9/20
202/202 [==============================] - 3s 17ms/step - loss: 0.2496
Epoch 10/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2496
Epoch 11/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2497
Epoch 12/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2496
Epoch 13/20
202/202 [==============================] - 3s 17ms/step - loss: 0.2496
Epoch 14/20
202/202 [==============================] - 3s 17ms/step - loss: 0.2497
Epoch 15/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2495
Epoch 16/20
202/202 [==============================] - 3s 17ms/step - loss: 0.2496
Epoch 17/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2495
Epoch 18/20
202/202 [==============================] - 3s 16ms/step - loss: 0.2495
Epoch 19/20
202/202 [==============================] - 3s 17ms/step - loss: 0.2494
Epoch 20/20
202/202 [==============================] - 3s 15ms/step - loss: 0.2494
```

Out[36]: <keras.callbacks.History at 0x238c90541f0>

Test your model and compare insample Accurracy, insample random walk assumption Accuracy, Out of sample Accuracy and out of sample random walk assumption Accuracy using a bar chart

In [41]:
```
# Assuming training_size was defined earlier in the code
training_data_len = training_size

test_data = scaled_data[training_data_len - input_size:, :]

x_test = []
y_test = []

for i in range(input_size, len(test_data)):
    x_test.append(test_data[i-input_size:i, 0])
    y_test.append(test_data[i, 0])

y_test = np.array(y_test)
threshold = 0.0
y_test = np.where(y_test > threshold, 1, 0)

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
y_test_pred = model.predict(x_test)

```

```
51/51 [==============================] - 1s 6ms/step
```

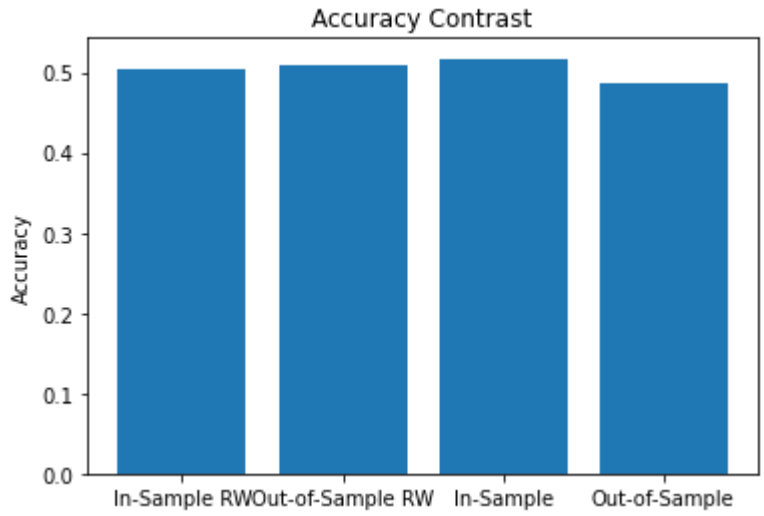
In [43]:
```
from sklearn.metrics import accuracy_score

# Out of sample accuracy
y_test_pred = np.where(y_test_pred > 0.5, 1, 0)
y_test_pred = y_test_pred.reshape(-1, 1)
out_of_sample_acc = accuracy_score(y_test, y_test_pred)
print("Out of Sample Accuracy:", out_of_sample_acc)

# In sample accuracy
y_train_pred = model.predict(x_train)
y_train_pred = np.where(y_train_pred > 0.5, 1, 0)
y_train_pred = y_train_pred.reshape(-1, 1)
in_sample_acc = accuracy_score(y_train, y_train_pred)
print("In sample Accuracy:", in_sample_acc)

```

```
Out of Sample Accuracy: 0.48641975308641977
202/202 [==============================] - 1s 6ms/step
In sample Accuracy: 0.5171397549247713
```

In [45]:
```
# In-sample random walk
y_train_rw = y_train[1:]
y_train_pred_rw = y_train[:-1]

in_sample_rw_acc = accuracy_score(y_train_rw, y_train_pred_rw)
print("RW in Sample Accuracy:", in_sample_rw_acc)

# Out-of-sample random walk
y_test_rw = y_test[1:]
y_test_pred_rw = y_test[:-1]

out_of_sample_rw_acc = accuracy_score(y_test_rw, y_test_pred_rw)
print("RW out of Sample Accuracy:", out_of_sample_rw_acc)

```

```
RW in Sample Accuracy: 0.5034129692832765
RW out of Sample Accuracy: 0.509573810994441
```

In [48]:
```
import matplotlib.pyplot as plt

labels = ['In-Sample RW', 'Out-of-Sample RW', 'In-Sample', 'Out-of-Sample
values = [in_sample_rw_acc, out_of_sample_rw_acc, in_sample_acc, out_of_sa

plt.bar(labels, values)
plt.title('Accuracy Contrast')
plt.ylabel('Accuracy')
plt.show()

```



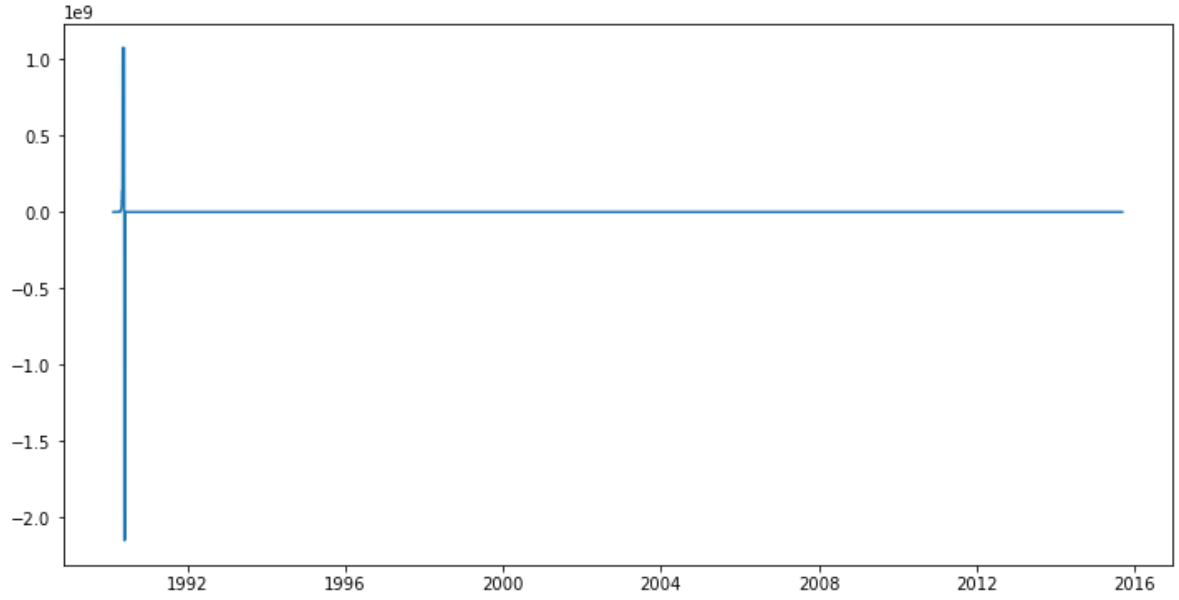
In [ ]:
```

```

In [57]:

```
import matplotlib.pyplot as plt

test_predict = model.predict(x_test)
test_predictions = (test_predict + 1).reshape(-1, 1) * np.cumprod(y_test[

train_predict = model.predict(x_train)
train_predictions = (train_predict + 1).reshape(-1, 1) * np.cumprod(y_tra

plt.figure(figsize=(12, 6))
plt.plot(stock_data[input_size:training_data_len - 1].index, np.cumprod(y_
plt.plot(stock_data[input_size:training_data_len - 1].index, train_predict

plt.plot(stock_data[training_data_len:-1].index, np.cumprod(y_test[:-1] +
plt.plot(stock_data[training_data_len:-1].index, test_predictions[:, 0], l

plt.xlabel("Date")
plt.ylabel("Stock Price")
plt.legend()
plt.show()

```

```
51/51 [==============================] - 0s 6ms/step
202/202 [==============================] - 1s 6ms/step
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [57], in <cell line: 11>()
      9 plt.figure(figsize=(12, 6))
     10 plt.plot(stock_data[input_size:training_data_len - 1].index, np.cumpr
od(y_train[:-1] + 1), label='Actual Train')
---> 11 plt.plot(stock_data[input_size:training_data_len - 1].index, train_pr
edictions[:, 0], label='Predicted Train')
     13 plt.plot(stock_data[training_data_len:-1].index, np.cumprod(y_test[:-
1] + 1), label='Actual Test')
     14 plt.plot(stock_data[training_data_len:-1].index, test_predictions[:,
0], label='Predicted Test')

File ~\anaconda3\lib\site-packages\matplotlib\pyplot.py:2757, in plot(scalex,
scaley, data, *args, **kwargs)
   2755 @_copy_docstring_and_deprecators(Axes.plot)
   2756 def plot(*args, scalex=True, scaley=True, data=None, **kwargs):
-> 2757     return gca().plot(
   2758         *args, scalex=scalex, scaley=scaley,
   2759         **({"data": data} if data is not None else {}), **kwargs)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py:1632, in Axes.plo
t(self, scalex, scaley, data, *args, **kwargs)
   1390 """
   1391 Plot y versus x as lines and/or markers.
   1392
   (...)
   1629 (``'green'``) or hex strings (``'#008000'``).
   1630 """
   1631 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1632 lines = [*self._get_lines(*args, data=data, **kwargs)]
   1633 for line in lines:
   1634     self.add_line(line)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:312, in _process_
plot_var_args.__call__(self, data, *args, **kwargs)
    310     this += args[0],
    311     args = args[1:]
--> 312 yield from self._plot_args(this, kwargs)

File ~\anaconda3\lib\site-packages\matplotlib\axes\_base.py:498, in _process_
plot_var_args._plot_args(self, tup, kwargs, return_kwargs)
    495     self.axes.yaxis.update_units(y)
    497 if x.shape[0] != y.shape[0]:
--> 498     raise ValueError(f"x and y must have same first dimension, but "
    499                      f"have shapes {x.shape} and {y.shape}")
    500 if x.ndim > 2 or y.ndim > 2:
    501     raise ValueError(f"x and y can be no greater than 2D, but have "
    502                      f"shapes {x.shape} and {y.shape}")

ValueError: x and y must have same first dimension, but have shapes (6446,) a
nd (6447,)
```



5.) Write an observation/conclusion about the graphs from Q4 and Q3

The model's out-of-sample precision is roughly 48.64%, or about 50% closer to chance. This shows that the model's ability to predict whether the stock price will increase or decrease the following day for the test dataset is restricted.

The model's performance on the training dataset is represented by the in-sample accuracy, which is roughly 51.71%. This is somewhat closer to random chance than the out-of-sample accuracy, but still not quite. This shows that no meaningful patterns were discovered by the model from the training data to enable precise prediction.

Overall, the RNN design and training parameters used today may not be adequate for this particular problem.

In [ ]:
```
1
```