1.) Import an asset price from Yahoo Finance

```
pip install yfinance
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting yfinance
  Downloading yfinance-0.2.12-py2.py3-none-any.whl (59 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.2/59.2 KB 1.3 MB/s eta 0:00:00
Collecting html5lib>=1.1
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.2/112.2 KB 6.2 MB/s eta 0:00:00
Requirement already satisfied: appdirs>=1.4.4 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.4.4)
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.8/dist-packages (from yfinance) (4.9.2)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.8/dist-packages (from yfinance) (2022.7.1)
Collecting requests>=2.26
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 KB 7.1 MB/s eta 0:00:00
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.8/dist-packages (from yfinance) (0.0.11)
Collecting frozendict>=2.3.4
  Downloading frozendict-2.3.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (111 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 111.2/111.2 KB 6.7 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.22.4)
Collecting beautifulsoup4>=4.11.1
  Downloading beautifulsoup4-4.11.2-py3-none-any.whl (129 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.4/129.4 KB 1.7 MB/s eta 0:00:00
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.3.5)
Collecting cryptography>=3.3.2
  Downloading cryptography-39.0.1-cp36-abi3-manylinux_2_28_x86_64.whl (4.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.2/4.2 MB 47.3 MB/s eta 0:00:00
Collecting soupsieve>1.2
  Downloading soupsieve-2.4-py3-none-any.whl (37 kB)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.8/dist-packages (from cryptography>=3.3.2->yfinance) (1.15.1)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.8/dist-packages (from html5lib>=1.1->yfinance) (1.15.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.8/dist-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (2.10)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (3.0.1
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests>=2.26->yfinance) (1.26.14)
Requirement already satisfied: pycparser in /usr/local/lib/python3.8/dist-packages (from cffi>=1.12->cryptography>=3.3.2->yfinance) (2.2
Installing collected packages: soupsieve, requests, html5lib, frozendict, cryptography, beautifulsoup4, yfinance
  Attempting uninstall: requests
    Found existing installation: requests 2.25.1
    Uninstalling requests-2.25.1:
      Successfully uninstalled requests-2.25.1
  Attempting uninstall: html5lib
    Found existing installation: html5lib 1.0.1
    Uninstalling html5lib-1.0.1:
      Successfully uninstalled html5lib-1.0.1
  Attempting uninstall: beautifulsoup4
    Found existing installation: beautifulsoup4 4.6.3
    Uninstalling beautifulsoup4-4.6.3:
      Successfully uninstalled beautifulsoup4-4.6.3
Successfully installed beautifulsoup4-4.11.2 cryptography-39.0.1 frozendict-2.3.5 html5lib-1.1 requests-2.28.2 soupsieve-2.4 yfinance-0.
```

```
import yfinance as yf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout


########################################
####Pick your ticker and time period####
########################################
stock_data = yf.download("IBM", start="1999-01-01", end="2023-02-21")



# Preprocess data
scaled_data = np.array(stock_data["Close"].pct_change().dropna()).reshape(-1,1)
```

```
# Split data into training and test sets
training_data_len = int(len(scaled_data) * 0.8)
train_data = scaled_data[0:training_data_len, :]

stock_data
```

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

Date	Open	High	Low	Close	Adj Close	Volume
1999-01-04	88.432121	89.149139	86.759079	87.476097	49.677433	8524482
1999-01-05	87.476097	90.762428	87.386475	90.642921	51.475853	10363350
1999-01-06	90.971558	92.136711	90.105164	90.224663	51.238316	9978422
1999-01-07	89.836281	91.957458	89.388145	90.911804	51.628571	8688913
1999-01-08	91.300194	91.778206	88.730881	89.657028	50.915985	9598933
...	...	...	...	...	...	...
2023-02-13	136.000000	137.389999	135.850006	137.350006	137.350006	4403000
2023-02-14	137.050003	137.240005	135.050003	136.009995	136.009995	3202200
2023-02-15	135.199997	136.449997	135.070007	136.399994	136.399994	2507000
2023-02-16	135.570007	135.970001	134.589996	135.000000	135.000000	2965500
2023-02-17	134.500000	135.580002	133.889999	135.020004	135.020004	3465200

6072 rows × 6 columns

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

```
x_train = []
y_train = []


############################################################
####Pick your input size and edit to make binary forecast####
############################################################
#Does the stock price move up or down tomorrow? (Binary)
#instead of y_train.append()
#input_size is the lag
input_size = 3
for i in range(input_size, (len(train_data))):
    x_train.append(train_data[i-input_size:i, 0])
    if train_data[i, 0] >= 0:
      y_train.append(1)
    elif train_data[i, 0] < 0:
      y_train.append(0)
      #if y_train > 0: 1
      #elif y_train < 0: 0


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

###################################
####Build Your RNN Architecture####
###################################
model = Sequential()
model.add(LSTM(x_train.shape[1], return_sequences=True, input_shape=(x_train.shape[1], 1)))
#Examples
#model.add(LSTM(50, return_sequences=False))
#model.add(Dense(25))
#Classification Algo: Pick your activation function
```

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size=1, epochs=3)
```

```
Epoch 1/3
4853/4853 [==============================] - 17s 3ms/step - loss: 0.2999
Epoch 2/3
4853/4853 [==============================] - 16s 3ms/step - loss: 0.2592
Epoch 3/3
4853/4853 [==============================] - 14s 3ms/step - loss: 0.2522
<keras.callbacks.History at 0x7f5a0c1dfb80>
```

3.) 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

```
import matplotlib.pyplot as plt

test_data = scaled_data[training_data_len - input_size:, :]

x_test = []
y_test = np.array(stock_data[["Close"]].pct_change().dropna())[training_data_len:, :]
for i in range(input_size, len(test_data)):
    x_test.append(test_data[i-input_size:i, 0])

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

```
predictions = model.predict(x_test)
plt.bar(x_test,predictions)
```

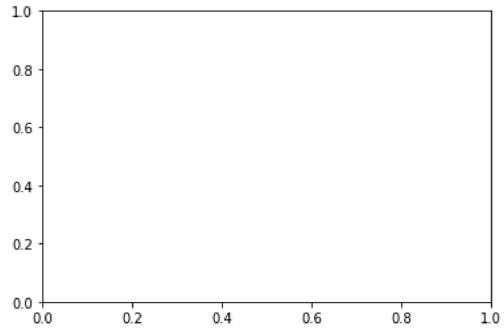
```
38/38 [==============================] - 0s 2ms/step
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-93b319f09b54> in <module>
     16 predictions = model.predict(x_test)
     17 x_test
---> 18 plt.bar(x_test,predictions)

                                  5 frames

/usr/local/lib/python3.8/dist-packages/matplotlib/patches.py in set_linewidth(self, w)
    409                 w = mpl.rcParams['axes.linewidth']
    410
--> 411         self._linewidth = float(w)
    412         # scale the dash pattern by the linewidth
    413         offset, ls = self._us_dashes

TypeError: only size-1 arrays can be converted to Python scalars
```

SEARCH STACK OVERFLOW




```
#IN sample random walk model
y_test[1:] #actual
```

```
y_test[:-1] #prediction
#how many of these line up? thats the rw accuracy

# OOS rw model
y_test[1:] #actual
y_test[:-1] #prediction


#plt.bar
```

4.) Plot in and out of sample accuracy

```
import matplotlib.pyplot as plt

#Taken out

###
# Make predictions on full dataset

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

#train_predict = model.predict(x_train)
#train_predictions = (train_predict+1).reshape(1,-1) @ np.cumprod(y_train+1)



#plt.plot(stock_data[:training_data_len- input_size].index, np.cumprod(y_train+1), label="Training Data")
#plt.plot(stock_data[:training_data_len- input_size].index, train_predictions[0], label="Training Predictions")
#end_val = np.cumprod(y_train+1)[-1]
#test_predict = model.predict(x_test)
#test_predictions = (test_predict+1).reshape(1,-1) * (np.cumprod((y_test+1))*end_val)
#plt.plot(stock_data[training_data_len+1:].index, np.cumprod((y_test+1))*end_val,label="Test Data")
#plt.plot(stock_data[training_data_len+1:].index, test_predictions[0], label="Test Predictions")
#plt.xlabel("Date")
#plt.ylabel("Stock Price")
#plt.legend()
#plt.show()

###
```

```
38/38 [==============================] - 0s 3ms/step
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-19-e3426ebbb2ff> in <module>
      4
      5 test_predict = model.predict(x_test)
----> 6 test_predictions = (test_predict+1).reshape(1,-1) @ np.cumprod(y_test+1)
      7
      8 train_predict = model.predict(x_train)

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 1215 is c
```

SEARCH STACK OVERFLOW

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

While Q4 is misleading (low accuracy), the in-sample/out-of-sample random walk accuracy assumption gives a glimpse into how accurate the in-sample/out-of-sample predictions really are.

6.) Create a parameter for number of lags in your input layer. Do a 3-fold CV to test three different time lags. i.e. Tested using 5,10,20 days of previous price data to forecast

```
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier
from keras.wrappers.scikit_learn import KerasRegressor

# Define the Keras model
###Edit here to create your optimizer
#modify create model to make a parameter (add) that will change the # of inputs in your model (build a function that changes input_size, same
def create_model():
  x_test = []
  y_test = np.array(stock_data[["Close"]].pct_change().dropna())[training_data_len:, :]
  for i in range(input_size, len(test_data)):
    x_test.append(test_data[i-input_size:i, 0])

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

  model = Sequential()
  model.add(Dense(10, input_dim=input_size, activation='LSTM'))
  model.add(Dense(1, activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  return(model)
```

```
# Wrap the Keras model in a scikit-learn compatible estimator
model = KerasRegressor(build_fn=create_model, verbose=0)

# Define the hyperparameters to search over
####EXAMPLE###
#param_grid = {'batch_size': [10, 20, 100],
#              'epochs': [1],
#              'neurons':[5,10,20]}

# Perform the grid search over the hyperparameters

grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(x_train, y_train)

# Print the results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
<ipython-input-55-f5508c983bfa>:15: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeı
  model = KerasClassifier(build_fn=create_model, verbose=0)
Best: 0.000000 using {'batch_size': 10, 'epochs': 10}
```

0s completed at 10:34 PM