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 1.8 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: pandas>=1.3.0 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.3.5)
Collecting html5lib>=1.1
  Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 112.2/112.2 KB 5.5 MB/s eta 0:00:00
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 4.9 MB/s eta 0:00:00
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 22.6 MB/s eta 0:00:00
Collecting requests>=2.26
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 KB 2.7 MB/s eta 0:00:00
Collecting beautifulsoup4>=4.11.1
  Downloading beautifulsoup4-4.11.2-py3-none-any.whl (129 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.4/129.4 KB 7.1 MB/s eta 0:00:00
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)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.8/dist-packages (from yfinance) (0.0.11)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.8/dist-packages (from yfinance) (1.22.4)
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: webencodings in /usr/local/lib/python3.8/dist-packages (from html5lib>=1.1->yfinance) (0.5.1)
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: 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: 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) (1.24.3)
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: 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("MSFT", start="1990-01-01", end="2022-02-21")
```

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

```
stock_data
```

	Open	High	Low	Close	Adj Close	Volume
Date						
1990-01-02	0.605903	0.616319	0.598090	0.616319	0.384520	53035200
1990-01-03	0.621528	0.626736	0.614583	0.619792	0.386687	113774400
1990-01-04	0.619792	0.638889	0.616319	0.638021	0.398060	125740800
1990-01-05	0.635417	0.638889	0.621528	0.622396	0.388312	69566400
1990-01-08	0.621528	0.631944	0.614583	0.631944	0.394269	58982400
...	...	...	...	...	...	...
2022-02-14	293.769989	296.760010	291.350006	295.000000	291.531281	36359500
2022-02-15	300.010010	300.799988	297.019989	300.470001	296.936951	27058300
2022-02-16	298.369995	300.869995	293.679993	299.500000	296.590363	29982100
2022-02-17	296.359985	296.799988	290.000000	290.730011	287.905548	32461600

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

8098 rows × 6 columns

```
scaled_data
```

```
array([[ 0.00563504],
       [ 0.02941149],
       [-0.02448979],
       ...,
       [-0.00322828],
       [-0.0292821 ],
       [-0.00963099]])
```

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

```
training_data_len
```

```
6477
```

```
train_data
```

```
array([[ 0.00563504],
       [ 0.02941149],
       [-0.02448979],
       ...,
       [ 0.00438897],
       [-0.01011956],
       [ 0.02184012]])
```

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####
#############################################################

#forecasting here did the stock price move up or down tomorrow
    # instead of having y_train be a percentage dif it shud just be did stock move up or down

# input size = this is the number of time lags, 2 = today and yesterdays price data to forecast tomorrow
input_size = 10
for i in range(input_size, len(train_data)):
    x_train.append(train_data[i-input_size:i, 0])
    if train_data[i, 0] > train_data[i-1, 0]:
        y_train.append(1)  # stock moved up
    else:
        y_train.append(0)

x_train, y_train = np.array(x_train), np.array(y_train)
# Reshape x_train to match input shape of LSTM layer
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))


model = Sequential()

model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(50))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])


model.fit(x_train, y_train, epochs=50, batch_size=32)
```

```
Epoch 1/50
203/203 [==============================] - 9s 15ms/step - loss: 0.6930 - accuracy: 0.5123
Epoch 2/50
203/203 [==============================] - 3s 15ms/step - loss: 0.6921 - accuracy: 0.5234
Epoch 3/50
203/203 [==============================] - 3s 17ms/step - loss: 0.6801 - accuracy: 0.5783
Epoch 4/50
203/203 [==============================] - 4s 21ms/step - loss: 0.6144 - accuracy: 0.6657
Epoch 5/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5633 - accuracy: 0.7088
Epoch 6/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5414 - accuracy: 0.7291
Epoch 7/50
203/203 [==============================] - 3s 16ms/step - loss: 0.5343 - accuracy: 0.7385
Epoch 8/50
203/203 [==============================] - 4s 20ms/step - loss: 0.5316 - accuracy: 0.7385
Epoch 9/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5288 - accuracy: 0.7421
Epoch 10/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5260 - accuracy: 0.7402
Epoch 11/50
203/203 [==============================] - 3s 16ms/step - loss: 0.5278 - accuracy: 0.7381
Epoch 12/50
203/203 [==============================] - 4s 20ms/step - loss: 0.5258 - accuracy: 0.7467
Epoch 13/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5226 - accuracy: 0.7478
Epoch 14/50
203/203 [==============================] - 4s 20ms/step - loss: 0.5231 - accuracy: 0.7404
Epoch 15/50
203/203 [==============================] - 4s 20ms/step - loss: 0.5204 - accuracy: 0.7452
Epoch 16/50
203/203 [==============================] - 3s 16ms/step - loss: 0.5218 - accuracy: 0.7450
Epoch 17/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5205 - accuracy: 0.7445
Epoch 18/50
203/203 [==============================] - 3s 16ms/step - loss: 0.5230 - accuracy: 0.7430
Epoch 19/50
203/203 [==============================] - 4s 21ms/step - loss: 0.5197 - accuracy: 0.7439
Epoch 20/50
203/203 [==============================] - 3s 16ms/step - loss: 0.5181 - accuracy: 0.7458
Epoch 21/50
203/203 [==============================] - 3s 16ms/step - loss: 0.5188 - accuracy: 0.7439
Epoch 22/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5211 - accuracy: 0.7449
Epoch 23/50
203/203 [==============================] - 4s 21ms/step - loss: 0.5203 - accuracy: 0.7463
Epoch 24/50
203/203 [==============================] - 3s 16ms/step - loss: 0.5183 - accuracy: 0.7475
Epoch 25/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5178 - accuracy: 0.7487
Epoch 26/50
203/203 [==============================] - 3s 16ms/step - loss: 0.5195 - accuracy: 0.7421
Epoch 27/50
203/203 [==============================] - 4s 21ms/step - loss: 0.5175 - accuracy: 0.7461
Epoch 28/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5175 - accuracy: 0.7493
Epoch 29/50
203/203 [==============================] - 3s 15ms/step - loss: 0.5161 - accuracy: 0.7455
```

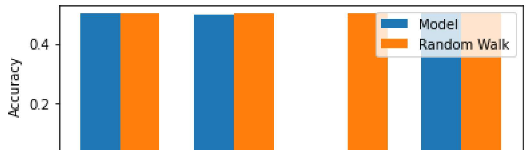
3.) Test your model and compare insample Accurracy, insample random walk assumption Accuracy, Out of sample Accuracy and out of sample random walk assumption Accuracy using a bar chart

```
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)
```

```
51/51 [==============================] - 1s 5ms/step
```

```
# This line of code will calculate the predicted direction of the stock movement based on our binary forecast
predicted_direction = np.where(predictions > 0.5, 1, 0)

# In-sample accuracy
train_predicted_direction = predicted_direction[:training_data_len-input_size]
train_actual_direction = y_train
train_accuracy = np.mean(train_predicted_direction == train_actual_direction)

# In-sample random walk assumption accuracy
train_random_direction = np.random.randint(0, 2, size=len(train_actual_direction))
train_random_accuracy = np.mean(train_random_direction == train_actual_direction)

# Out-of-sample accuracy
test_predicted_direction = predicted_direction[training_data_len-input_size:]
test_actual_direction = np.where(y_test > 0, 1, 0)
test_accuracy = np.mean(test_predicted_direction == test_actual_direction)

# Out-of-sample random walk assumption accuracy
test_random_direction = np.random.randint(0, 2, size=len(test_actual_direction))
test_random_accuracy = np.mean(test_random_direction == test_actual_direction)

import matplotlib.pyplot as plt

labels = ['In-sample Accuracy', 'In-sample Random Walk Accuracy', 'Out-of-sample Accuracy', 'Out-of-sample Random Walk Accuracy']
model_accuracies = [train_accuracy, train_random_accuracy, test_accuracy, test_random_accuracy]
random_walk_accuracies = [0.5, 0.5, 0.5, 0.5]

x = np.arange(len(labels))
width = 0.35

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, model_accuracies, width, label='Model')
rects2 = ax.bar(x + width/2, random_walk_accuracies, width, label='Random Walk')

ax.set_ylabel('Accuracy')
ax.set_xticks(x)
ax.set_xticklabels(labels, rotation=45)
ax.legend()

fig.tight_layout()

plt.show()
```

```
<ipython-input-10-0ae27e7848c5>:16: DeprecationWarning: elementwise comparison failed; this
  test_accuracy = np.mean(test_predicted_direction == test_actual_direction)
```



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

- For the bar charts in question 3, we can see that for in-sample accuracy, our model performed very similarly in terms of accuracy to the random walk. However, for out-of-sample accuracy, we can see that our model performed very poorly compared to the random walk. This can be seen by discrepancy between the bars for out of sample accuracy, as the bar for the model is nonexistent. This illustrates that our model did not outperform the random walk.

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
from sklearn.model_selection import KFold


def create_model(lag):
    model = Sequential()
    model.add(LSTM(50, return_sequences=True, input_shape=(lag, 1)))
    model.add(Dropout(0.2))

    model.add(LSTM(50))
    model.add(Dropout(0.2))

    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return(model)


param_grid = {'batch_size': [10, 20, 32],
              'epochs': [10, 20],
              'lag': [5, 10, 20]}


model = KerasRegressor(build_fn=create_model, verbose=0)
```

```
<ipython-input-22-9f3192b984b9>:2: DeprecationWarning: KerasRegressor is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras
  model = KerasRegressor(build_fn=create_model, verbose=0)
```

```
# This code performs the grid search over the hyperparameters using 3-fold cross-validation
kfold = KFold(n_splits=3, shuffle=True, random_state=42)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=kfold)
grid_result = grid.fit(x_train, y_train)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
/usr/local/lib/python3.8/dist-packages/joblib/externals/loky/process_executor.py:700: UserWarning: A worker stopped while some jobs w
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
36 fits failed out of a total of 54.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
1 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.8/dist-packages/keras/wrappers/scikit_learn.py", line 175, in fit
```

```
    history = self.model.fit(x, y, **fit_args)
  File "/usr/local/lib/python3.8/dist-packages/keras/utils/traceback_utils.py", line 70, in error_handler
    raise e.with_traceback(filtered_tb) from None
  File "/tmp/__autograph_generated_fileog4uoxeb.py", line 15, in tf__train_function
    retval_ = ag__.converted_call(ag__.ld(step_function), (ag__.ld(self), ag__.ld(iterator)), None, fscope)
ValueError: in user code:

    File "/usr/local/lib/python3.8/dist-packages/keras/engine/training.py", line 1249, in train_function  *
        return step_function(self, iterator)
    File "/usr/local/lib/python3.8/dist-packages/keras/engine/training.py", line 1233, in step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    File "/usr/local/lib/python3.8/dist-packages/keras/engine/training.py", line 1222, in run_step  **
        outputs = model.train_step(data)
    File "/usr/local/lib/python3.8/dist-packages/keras/engine/training.py", line 1023, in train_step
        y_pred = self(x, training=True)
    File "/usr/local/lib/python3.8/dist-packages/keras/utils/traceback_utils.py", line 70, in error_handler
        raise e.with_traceback(filtered_tb) from None
    File "/usr/local/lib/python3.8/dist-packages/keras/engine/input_spec.py", line 295, in assert_input_compatibility
        raise ValueError(

    ValueError: Input 0 of layer "sequential" is incompatible with the layer: expected shape=(None, 5, 1), found shape=(None, 10, 1)


--------------------------------------------------------------------------------
1 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/dist-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.8/dist-packages/keras/wrappers/scikit_learn.py", line 175, in fit
    history = self.model.fit(x, y, **fit_args)
  File "/usr/local/lib/python3.8/dist-packages/keras/utils/traceback_utils.py", line 70, in error_handler
    raise e.with_traceback(filtered_tb) from None
  File "/tmp/__autograph_generated_filelq5zkci3.py", line 15, in tf__train_function
    retval_ = ag__.converted_call(ag__.ld(step_function), (ag__.ld(self), ag__.ld(iterator)), None, fscope)
ValueError: in user code:

    File "/usr/local/lib/python3.8/dist-packages/keras/engine/training.py", line 1249, in train_function  *
        return step_function(self, iterator)
    File "/usr/local/lib/python3.8/dist-packages/keras/engine/training.py", line 1233, in step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    File "/usr/local/lib/python3.8/dist-packages/keras/engine/training.py", line 1222, in run_step  **
```

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

```
Best: -0.519818 using {'batch_size': 20, 'epochs': 10, 'lag': 10}
```