Week 1 Lab - due by 10th July, 2022 (11:59 pm CDT)

[Week 1 Lecture Link Here]

Objective: to perform Exploratory Data Analysis (EDA) on a multiple-asset portfolio

Setup and Loading Packages

Setup and Loading Packages

In [87]:
```
%matplotlib inline
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import scipy.stats as stats
from datetime import datetime, timedelta
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from matplotlib.ticker import FuncFormatter
import pandas_datareader as pdr
```

Months following the COVID-19 pandemic recovery, the stock market seems to slow down on its ralley and the cryptocurrency market continues to show volatility. As a quantitative analyst and an investor, you want to understand the empirical behaviors of the assets before building a predictive model and investing in them, since you believe that this can give you a statistical edge in your portfolio. You have several assets in mind and would like to conduct an initial analysis on their historical performances to see if they are a good makeup for your portfolio.

Please complete the following problems to perform full EDA on your stock selection.

Problem 1: Preliminary Visualization

a) Select 3-5 assets of your preference, then specify their ticker(s), start and end dates of their price data you want to explore.

Notice that any assets can be selected, and not just stocks. For example, cryptocurrency and foreign exchange instruments can be suggested as well. Some relatively new cryptocurrencies (e.g., Solana, USD Coin) only have complete data dating back to three or four years ago, so setting the duration of data further back than these dates may result in inaccurate representation of their relationships.

In [100…
```
symbolList =  ['NFLX','TSLA','NVDA','TWTR'] # asset ticker symbols
START_DATE = '2015-07-08' # asset data start date
END_DATE = '2022-07-08' # asset data end date
```

Run the following code chunk to extract the adjusted close prices and compute log returns of Apple's stock from Yahoo Finance.

PLEASE DO NOT CHANGE THIS CODE !!!

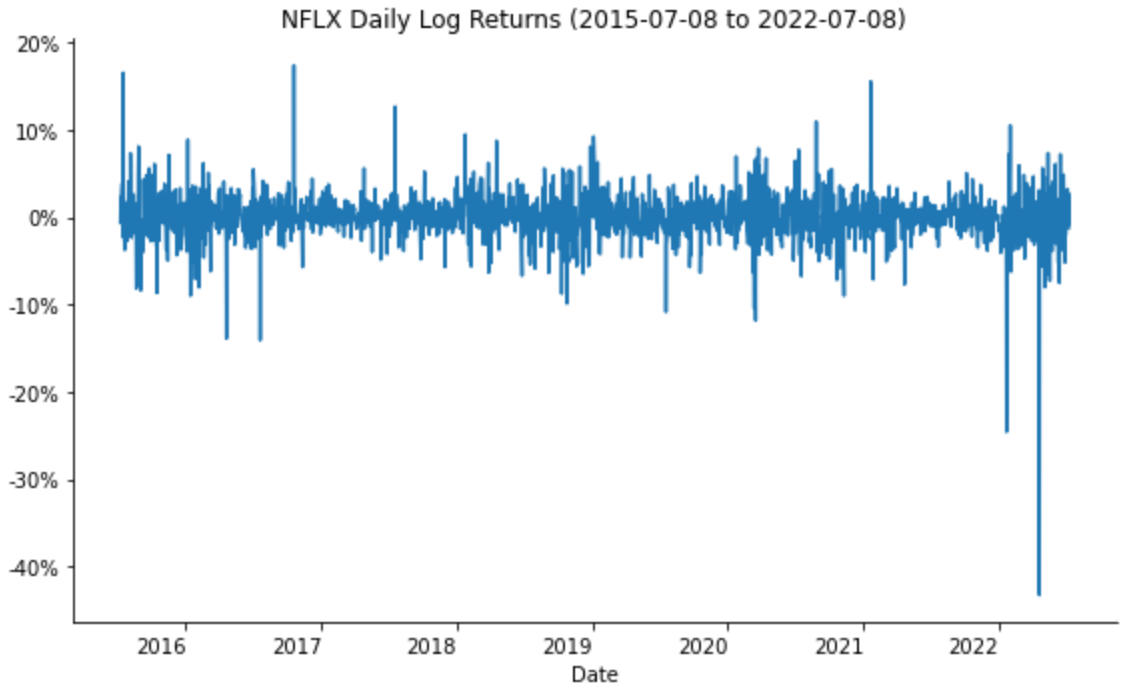
In [101…
```
stockPxList = pdr.DataReader(symbolList , 'yahoo', START_DATE, END_DATE)['Adj Close'] #
# converting prices to log returns and removing NaN values
stockLogRetList = np.log(stockPxList).diff().dropna()
```

b) Please write a code piece to perform 2 visualizaitons on the assets' log returns you extracted from a).

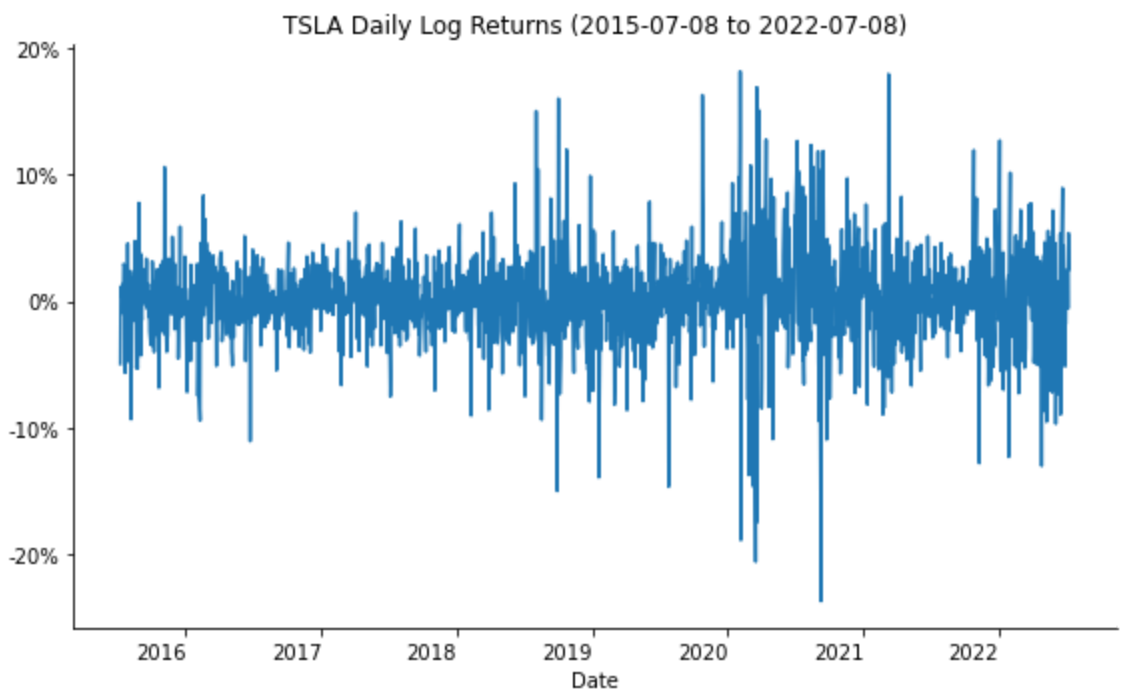
Since there are at least 3 assets, plots such as time plot, scatter plot, box plot, and histogram would have to be plotted more than once to evaluate each asset.

Visualization #1:

In [102…
```
## Log returns

#1 Netflix
ax = stockLogRetList['NFLX'].plot(figsize=(8, 5),
                      title=symbolList[0]+' Daily Log Returns '+ '(' + START_DATE + ' to
ax.yaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.0%}'.format(y)))
sns.despine()
plt.tight_layout();
```

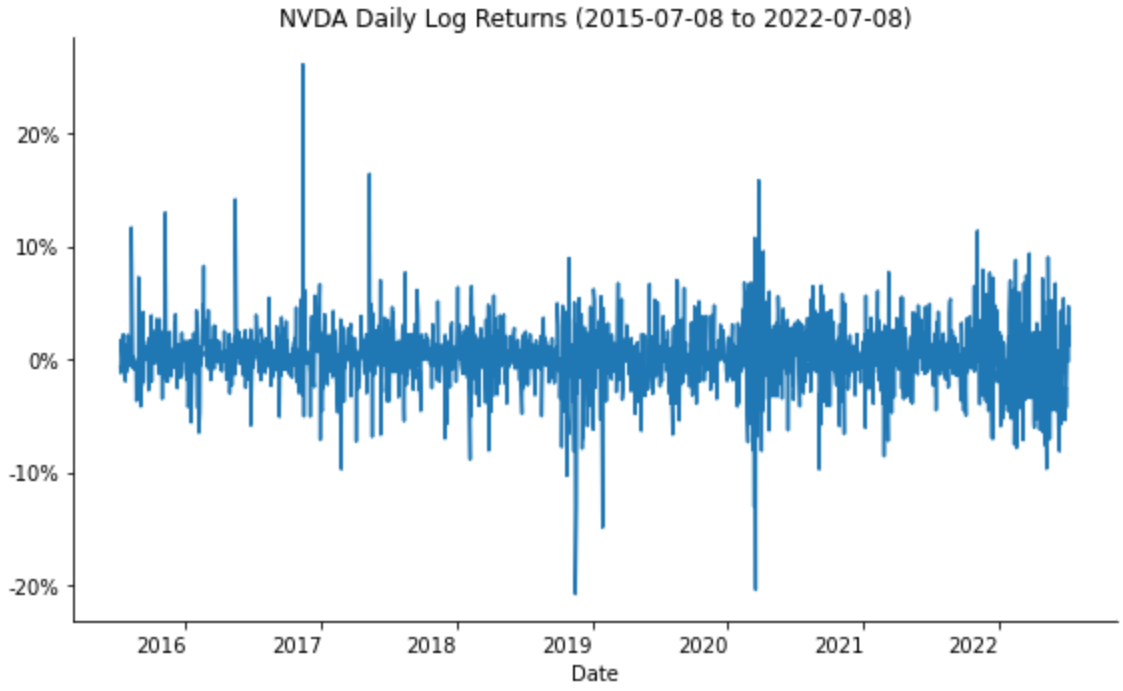


In [103…
```
#2 TSLA
ax = stockLogRetList['TSLA'].plot(figsize=(8, 5),
                      title=symbolList[1]+' Daily Log Returns '+ '(' + START_DATE + ' to
ax.yaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.0%}'.format(y)))
sns.despine()
plt.tight_layout();
```



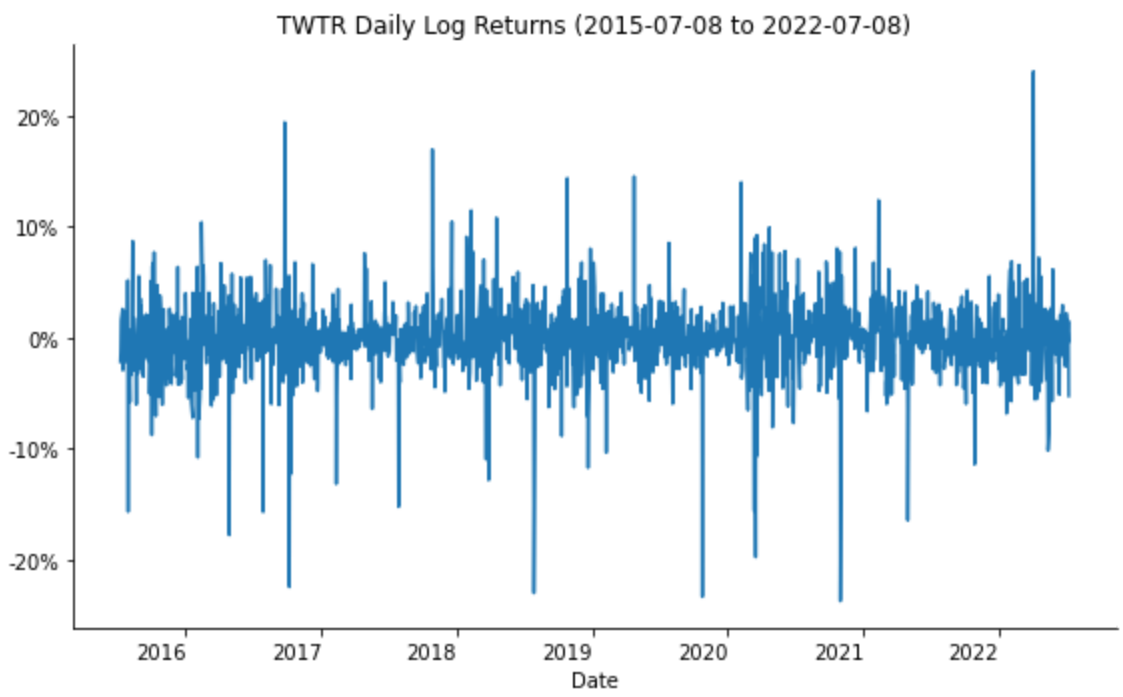
In [104…

```
#3 NVDA
ax = stockLogRetList['NVDA'].plot(figsize=(8, 5),
                        title=symbolList[2]+' Daily Log Returns '+ '(' + START_DATE + ' to
ax.yaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.0%}'.format(y)))
sns.despine()
plt.tight_layout();
```



In [105…

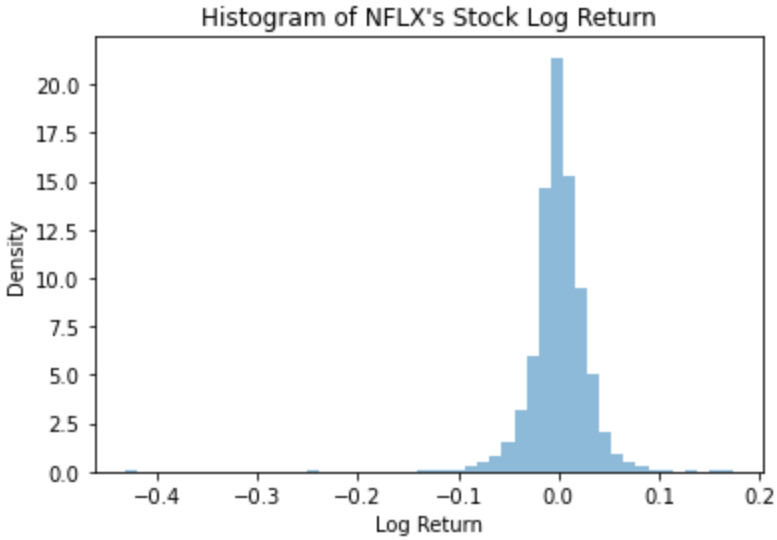
```
#4 TWTR
ax = stockLogRetList['TWTR'].plot(figsize=(8, 5),
                        title=symbolList[3]+' Daily Log Returns '+ '(' + START_DATE + ' to
ax.yaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.0%}'.format(y)))
sns.despine()
plt.tight_layout();
```



Visualization #2:

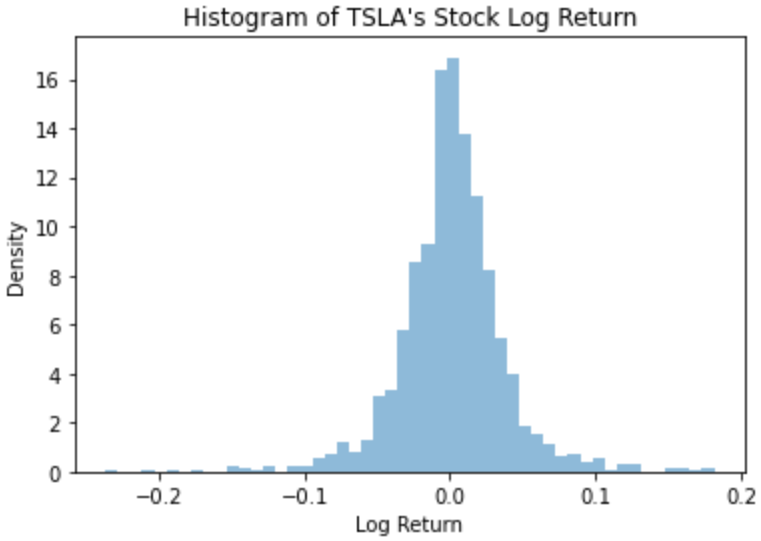
In [106…
```
#1 NFLX
_, bins, _= plt.hist(stockLogRetList['NFLX'], bins=50, density=1, alpha=0.5)
plt.title("Histogram of " + symbolList[0] + "'s Stock Log Return")
plt.xlabel("Log Return")
plt.ylabel("Density")
```

Out[106]:
```
Text(0, 0.5, 'Density')
```



In [107…
```
#2 TSLA
_, bins, _= plt.hist(stockLogRetList['TSLA'], bins=50, density=1, alpha=0.5)
plt.title("Histogram of " + symbolList[1] + "'s Stock Log Return")
plt.xlabel("Log Return")
plt.ylabel("Density")
```

Out[107]:
```
Text(0, 0.5, 'Density')
```

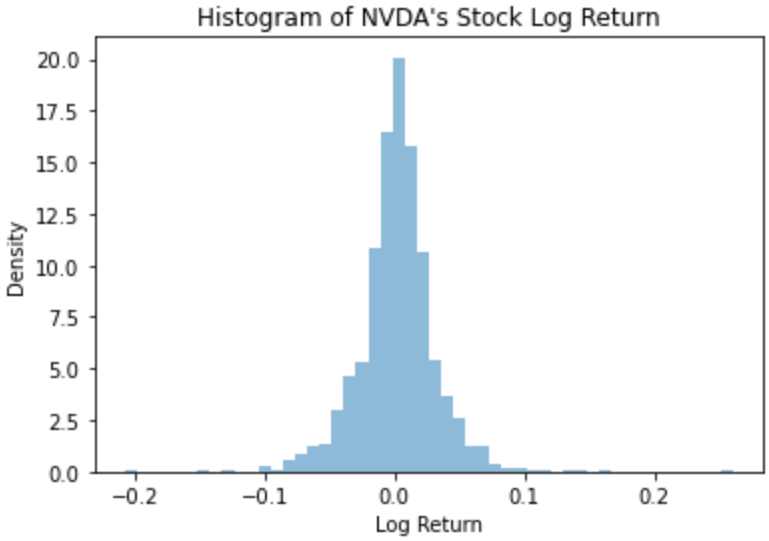


In [108…

```
#3 NVDA
_, bins, _= plt.hist(stockLogRetList['NVDA'], bins=50, density=1, alpha=0.5)
plt.title("Histogram of " + symbolList[2] + "'s Stock Log Return")
plt.xlabel("Log Return")
plt.ylabel("Density")
```

Out[108]:

```
Text(0, 0.5, 'Density')
```

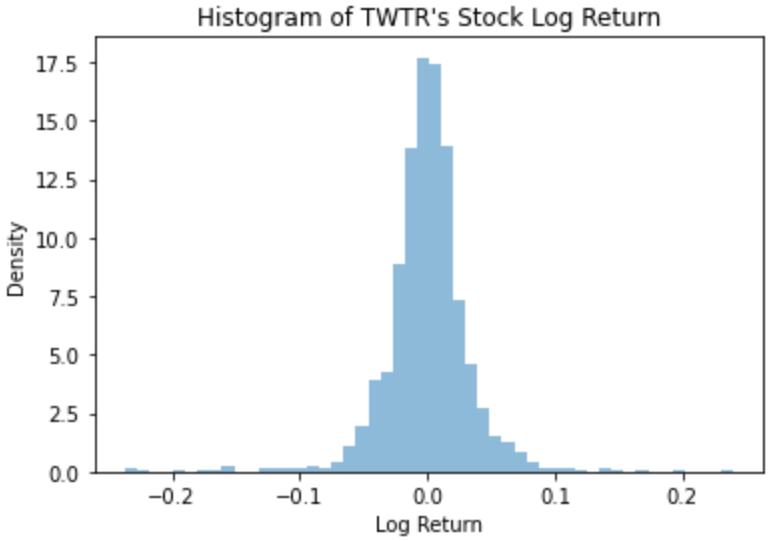


In [109…

```
#4 TWTR
_, bins, _= plt.hist(stockLogRetList['TWTR'], bins=50, density=1, alpha=0.5)
plt.title("Histogram of " + symbolList[3] + "'s Stock Log Return")
plt.xlabel("Log Return")
plt.ylabel("Density")
```

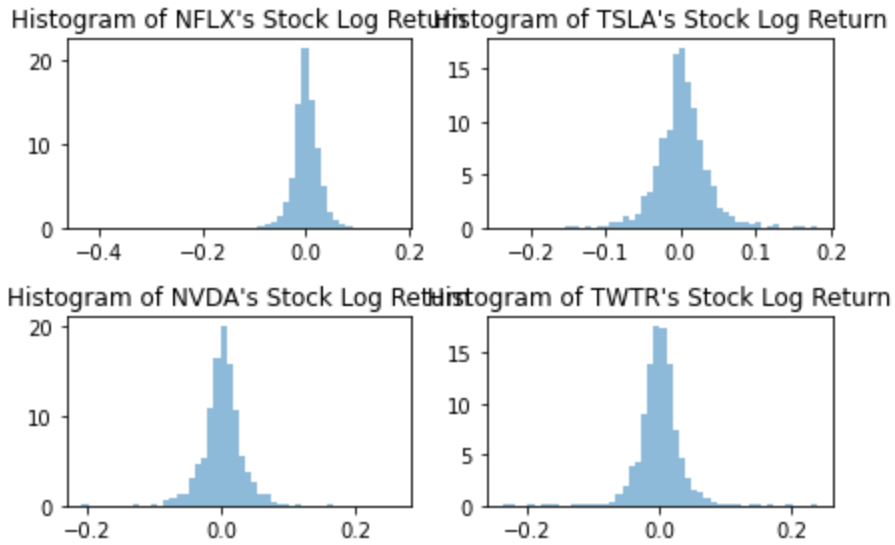
Out[109]:

```
Text(0, 0.5, 'Density')
```



In [110…

```
#OVERALL
fig,((ax0,ax1),(ax2,ax3))=plt.subplots(nrows=2,ncols=2)

ax0.hist(stockLogRetList['NFLX'], bins=50, density=1, alpha=0.5)
ax0.set_title("Histogram of " + symbolList[0] + "'s Stock Log Return")

ax1.hist(stockLogRetList['TSLA'], bins=50, density=1, alpha=0.5)
ax1.set_title("Histogram of " + symbolList[1] + "'s Stock Log Return")

ax2.hist(stockLogRetList['NVDA'], bins=50, density=1, alpha=0.5)
ax2.set_title("Histogram of " + symbolList[2] + "'s Stock Log Return")

ax3.hist(stockLogRetList['TWTR'], bins=50, density=1, alpha=0.5)
ax3.set_title("Histogram of " + symbolList[3] + "'s Stock Log Return")

fig.tight_layout()
```



c) Interpret the visualizations that you performed above. What can you say about them?

Netflix's log returns is centered around 10% to -10% (daily log return graph), only occassionally exceeding out this general limit (around 10 times), but there is one time when drastically decreased 40% (in 2022). Tesla is centured around 15% to -15%, but there is only one time exceeding 20% down. Nvidia is centured around 10% to -10%, and generally stays that way, it only has 3 instances of extending that limit. Twitter is mostly centured around 10% to -10%, but is periodically drops more than 20% or increases more than 20%

Of these four Tesla is the one that usually has greater returns / greater losses (as it fluctuates wider and the density is greater on the tails than other stocks I choice) Nvidia should be the more or less stable choice, as it fluctuates low.

Problem 2: Preliminary Normality Testing

You realized that within the date range you specified, there may be some days when the assets make big directional swings, hence skewing the data or thickening the probabilities of extreme values. To keep your minds in peace, you decided to perform normality testing to understand how your assets' distribution compare to what's condered 'normal'.

(e.g., If your date range spans the COVID-19 pandemic, you may see more extreme tail values or outliers in your log returns, which deviates from a normal distribution because the market fluctuates a lot during this time.)

a) Please write a code piece to perform 1 normality test on the assets' returns you extracted from problem 1.

In [111…
```
shapiro_test=stats.shapiro(stockLogRetList['TWTR'])
shapiro_test.pvalue
```

Out[111]:
```
1.1529121171114077e-34
```

b) Interpret the result you obtained from the normality test you chose in part a). What can you say about it?

Because the p-value is significantly lower than the usual 0.05 assumed, the null hypothesis (that the log returns of Twitter Stock is from a normal distribution) is rejected. TWTR's log return is not a normal distribution.

Problem 3: Preliminary Pre-processing

Imbalanced labels is a classification predictive modeling problem where the distribution of examples across the classes is not equal. For example, we may collect measurements of cats and have 80 samples of one cat species and 20 samples of a second cat species. This represents an example of an imbalanced classification problem. A 50-50 or a near-50-50 sample species would form a balanced classification problem.

As a quantitative analyst, you are curious as to how the list of assets you chose above helps predict the direction of another asset. But before diving into the modeling portion, you want to investigate any label imbalance problems.

Please read this blog before jumping into this question: https://machinelearningmastery.com/what-is-imbalanced-classification/

a) Specify the ticker of the asset whose direction you are interested in predicting. This stock shall be different than the ones you chose in problem 1.

In [112…
```
SYMBOL = 'META' # asset ticker symbol
```

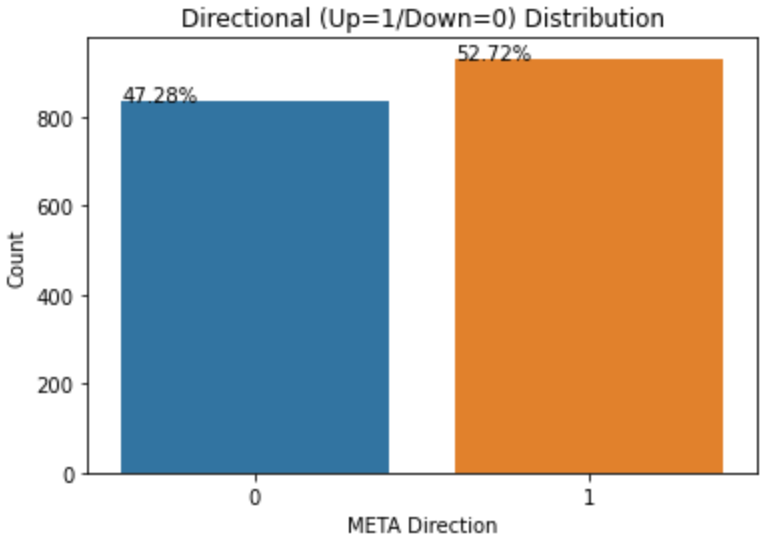
Run the following code chunk. This will binarize the returns for the asset that you're trying to predict over the period specified above. In other words, the asset's price will be transformed into 0's and 1's - 0 if price did not go up, 1 if price went up. A bar plot is produced to show the label distribution. For instance, there

should be one bar showing how many days the stock goes up and another showing how many days the stock goes down.

PLEASE DO NOT CHANGE THIS CODE CHUNK!!!

FEATURES = symbolList.copy() stockPx = pdr.get_data_yahoo(SYMBOL, START_DATE, END_DATE)['Adj Close'] # storing adjusted stock prices into a variable stockPx01 = (stockPx.pct_change().dropna() > 0).astype(int) # visualize directional label distribution ax = sns.countplot(x = stockPx01) plt.title('Directional (Up=1/Down=0) Distribution') plt.xlabel(SYMBOL + ' Direction') plt.ylabel('Count') total = len(stockPx01) for p in ax.patches: percentage = '{:.2f}%'.format(100 * p.get_height()/total) x_coord = p.get_x() y_coord = p.get_y() + p.get_height()+0.02 ax.annotate(percentage, (x_coord, y_coord))

In [113…
```
FEATURES = symbolList.copy()
stockPx = pdr.get_data_yahoo(SYMBOL, START_DATE, END_DATE)['Adj Close'] # storing adjust
stockPx01 = (stockPx.pct_change().dropna() > 0).astype(int)
# visualize directional label distribution
ax = sns.countplot(x = stockPx01)
plt.title('Directional (Up=1/Down=0) Distribution')
plt.xlabel(SYMBOL + ' Direction')
plt.ylabel('Count')
total = len(stockPx01)
for p in ax.patches:
        percentage = '{:.2f}%'.format(100 * p.get_height()/total)
        x_coord = p.get_x()
        y_coord = p.get_y() + p.get_height()+0.02
        ax.annotate(percentage, (x_coord, y_coord))
```



b) Are the lables balanced or imbalanced? Why?

The labels are mostly balanced, with only a slight imbalance, as the ratio of distribution of up and down is 48% : 52%, very close to a balance (50%:50%).

c) How do you think they can affect our prediction? (Hint: think about what the training data will look like)

As the algorithm for predictive modeling usually assumes an equal number of each class, an imbalance will cause too few of test cases in one class of the training set, which result in a poor predictive strength of that class.

d) Please suggest one way to handle imbalanced data?

Use SMOTE (Synthetic Minority Over-sampling Technique) to give some data from the majority class to the minority class.

e) What is are the features in this problem?

The list of assets I choose before (Nvidia,Tesla,Twitter,and Netflix)

f) Please write a one-line code to split the data into 80% training set and 20% testing set.

In [114…
```
X_train,X_test,y_train,y_test=train_test_split(stockPxList[1:],stockPx01,test_size=0.2,r
```

In [115…
```
stockPxList[1:].info()
stockPx01.info()
```

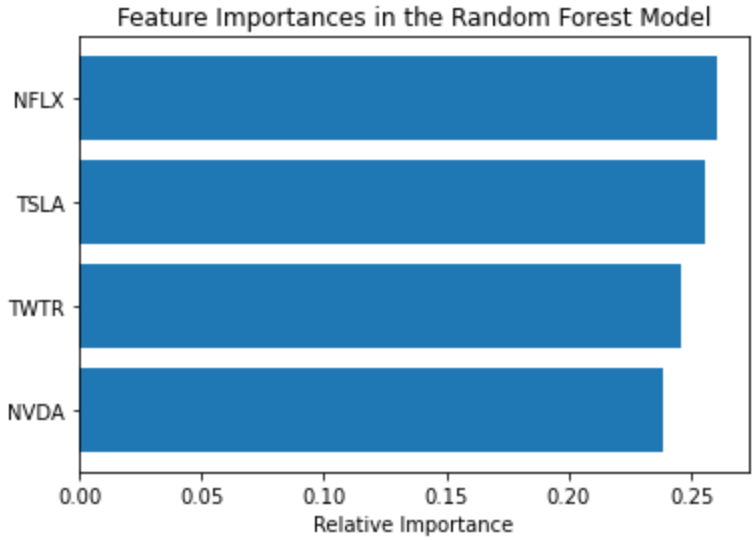
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1764 entries, 2015-07-08 to 2022-07-08
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   NFLX    1764 non-null   float64
 1   TSLA    1764 non-null   float64
 2   NVDA    1764 non-null   float64
 3   TWTR    1764 non-null   float64
dtypes: float64(4)
memory usage: 68.9 KB
<class 'pandas.core.series.Series'>
DatetimeIndex: 1764 entries, 2015-07-08 to 2022-07-08
Series name: Adj Close
Non-Null Count  Dtype
--------------  -----
1764 non-null   int32
dtypes: int32(1)
memory usage: 20.7 KB
```

g) Please write a code piece to visualize the feature importance ranking of with a bar plot. How are the features ranked by their importance scores?

(Hint: you would need to split the data first in part e) before computing the importance scores here)

In [116…
```
from sklearn.ensemble import RandomForestRegressor
rf_model = RandomForestRegressor(random_state=0)
rf_model.fit(X_train, y_train)
importances=rf_model.feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances in the Random Forest Model')
plt.barh(range(len(indices)), importances[indices], align='center')
plt.yticks(range(len(indices)), [FEATURES[i] for i in indices])
plt.xlabel('Relative Importance')
```

Out[116]:
```
Text(0.5, 0, 'Relative Importance')
```



Netflix is most important in predicting whether Meta's stock goes up or down, followed by Tesla, then Twitter, finally Nvidia, which is least important in predicting Meta.