DECEMBER 2022

STOCK PRICE PREDITOR
&
PORTFOLIO OPTIMIZATION

PRESENTED BY

MOHAMED HEKAL

TABLE OF CONTENTS

01 Definition

02 Analysis

03 Methodology

04 Results

05 Conclusion

DEFINITION

Overview

Investment firms, hedge funds, and even individuals have been using financial models to understand market behavior better and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process.

People need money to have what they need to live and thrive, but what do you need to be happy? Also money but a lot of it. One of the ways to get a lot of money is through investment, and why invest? Investing is an effective way to put your money to work and create potential wealth. Investing allows your money to outpace inflation and increase in value.
You may need a lot of money to retire early, retire with more money, or make a big purchase like buying a house, buying your dream car, or traveling the world, all of which means happiness.

Problem Statement

Financial modeling tries to predict the future price of the stocks by learning their historical price over years. So, in this project, I will build a stock price predictor by using a machine learning algorithm that takes daily trading data from the yahoo finance website over a certain date range as input (Adj Close), and outputs projected estimates for a given query date. The system will predict the adjusted closing price for the selected stock and for different time periods, as well as portfolio optimization taking the amount of money you intend to invest in the market and making a mix of stocks that maximizes the Sharpe Ratio, minimizes volatility or maximizes return using Efficient Frontier.

Metrics

Mean square error will be used for evaluating the price prediction model. The smallest the mean square error the better the model performance.
For the portfolio optimization backtest will be great but I will not implement it, I will use the Portfolio Visualizer website to evaluate the method and if the portfolio performs well against S&P 500 Index

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

ANALYSIS

The data used for this project is the adjusted closing price of 193 NASDAQ companies and these large and large companies in the market in various sectors.
The data used for analysis is from 2020-01-01 to 2022-01-01 and is used to analyze how companies and sectors perform during the pandemic.
we have a file from the NASDAQ website :
NASDAQ_screener.csv and this file contain all 193 tickers, their names, and in which sector, used for downloading the data by its ticker from yahoo finance.
The downloaded dataset is of the following form :

	AAPL	ABMD	ABNB	ADBE	ADI	ADP	ADSK	AEP	AKAM	ALGN	...	WBA	WBD	WD
Date														
2020-01-02	73.561539	168.809998	168.809998	334.429993	113.905510	160.128036	187.830002	84.466591	87.639999	283.679993	...	52.089615	32.220001	167.4600
2020-01-03	72.846367	166.820007	166.820007	331.809998	111.900337	159.789581	184.949997	84.376213	87.239998	280.440002	...	52.089615	32.029999	168.4400
2020-01-06	73.426826	179.039993	179.039993	333.709991	110.585663	160.005814	187.119995	84.656380	87.550003	285.880005	...	52.539272	31.959999	169.4900
2020-01-07	73.081490	180.350006	180.350006	333.390015	113.101555	158.069107	187.500000	84.674461	90.199997	283.059998	...	52.274773	32.070000	172.9499
2020-01-08	74.257111	178.690002	178.690002	337.869995	114.123055	159.554550	189.949997	84.421410	91.400002	286.000000	...	49.224159	32.110001	178.7100

So, in order to visualize and see how each sector performs during the pandemic, I do some data preprocessing to group by data by sector and then take the average for all tickers in each sector.
The preprocessing consists of 3 steps:

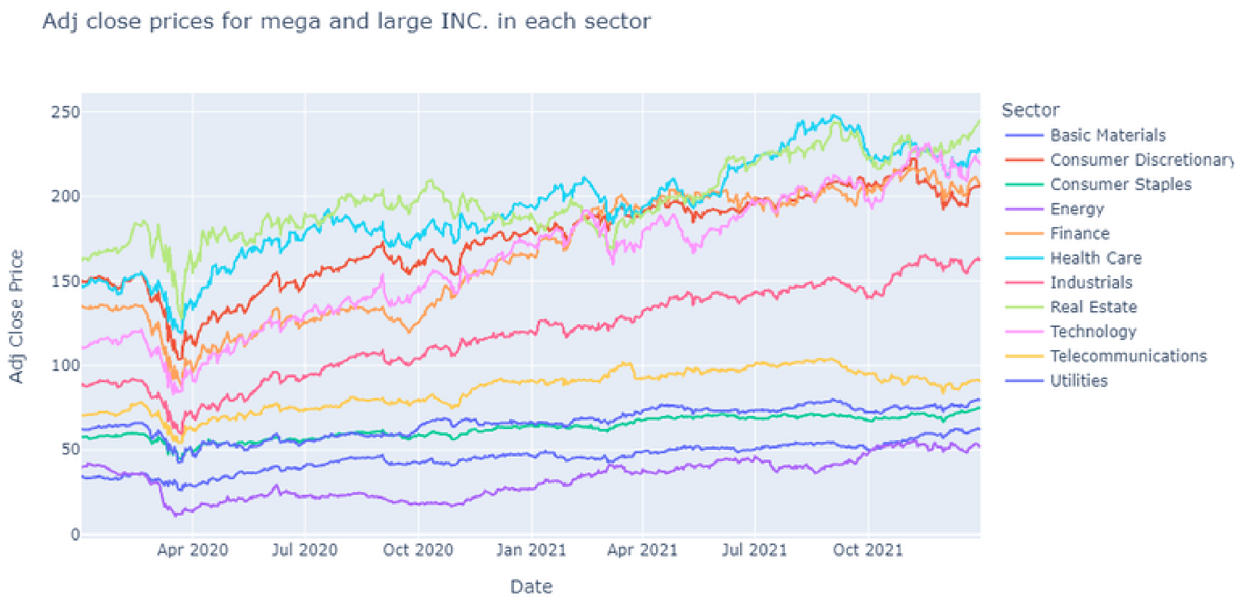
- Transpose the data to make tickers as the index
- Merge it with the NASDAQ_screener file to assign each ticker to its sector

		2020-01-02	2020-01-03	2020-01-06	2020-01-07	2020-01-08	2020-01-09	2020-01-10	2020-01-13	2020-01-14	2020-01-15	...	2021-12-17	2021-12-20	2021-12-21	2021-12-22
Sector	Symbol														
Technology	AAPL	73.561546	72.846359	73.426811	73.081505	74.257118	75.834389	76.005829	77.629639	76.581383	76.253204	...	170.152466	168.770493	171.991791	174.626495
Health Care	ABMD	168.809998	166.820007	179.039993	180.350006	178.690002	183.600006	189.059998	168.100006	172.729996	177.919998	...	315.549988	316.230011	332.200012	343.399994
Consumer Discretionary	ABNB	168.809998	166.820007	179.039993	180.350006	178.690002	183.600006	189.059998	168.100006	172.729996	177.919998	...	157.910004	157.229996	165.660004	169.289993
Technology	ADBE	334.429993	331.809998	333.709991	333.390015	337.869995	340.450012	339.809998	345.630005	344.630005	342.940002	...	556.640015	549.770020	557.520020	563.979980
	ADI	113.905487	111.900352	110.585663	113.101562	114.123039	114.123039	112.146255	112.609718	113.054253	111.143692	...	166.949066	165.192337	168.882477	169.088562
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Utilities	XEL	57.911705	58.190212	58.106667	57.985973	57.930279	58.060242	58.153076	58.830788	58.923626	59.851997	...	65.183388	65.300140	64.984711	65.366753
Technology	ZBRA	259.140015	256.049988	258.010010	256.470001	247.639999	246.500000	246.270004	248.580002	248.070007	247.800003	...	588.520020	570.780029	580.210022	580.429993
	ZI	259.140015	256.049988	258.010010	256.470001	247.639999	246.500000	246.270004	248.580002	248.070007	247.800003	...	62.730000	61.320000	64.580002	64.900002
	ZM	68.720001	67.279999	70.320000	71.900002	72.550003	72.620003	73.089996	74.029999	73.160004	76.940002	...	199.740005	197.970001	199.419998	193.130005

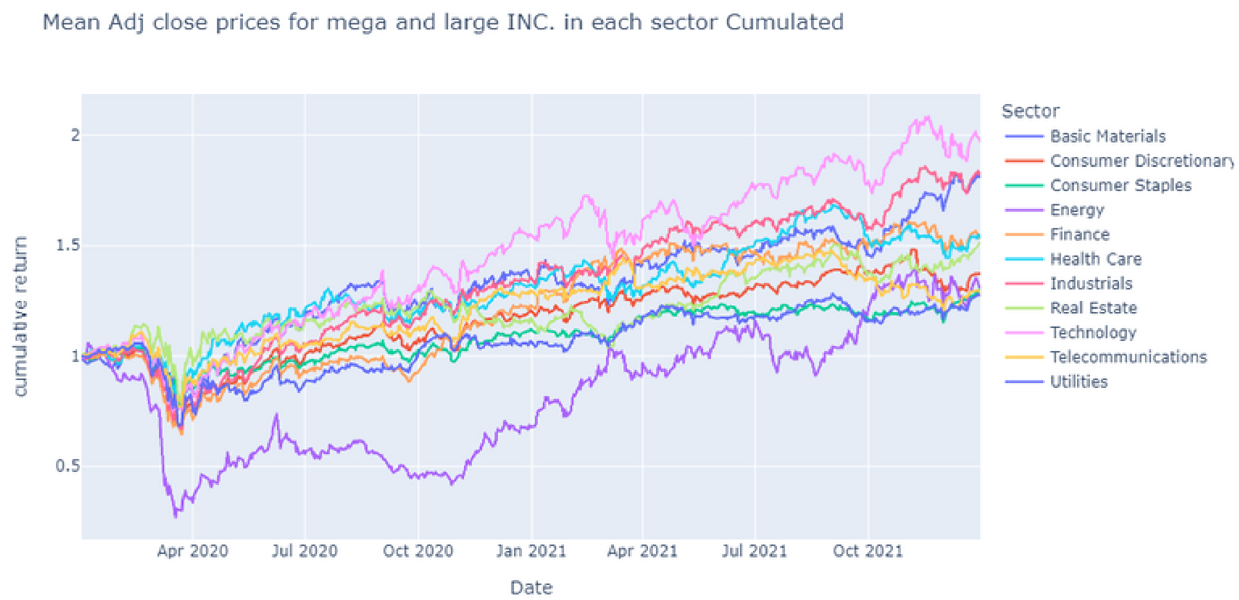
- Then group by sector and take the mean then transpose it again

2020-01-02	34.411755	150.009759	57.741906	39.725860	135.046585	147.254161	88.918388	161.905279	110.985927	70.424459	62.365625
2020-01-03	33.801716	149.395973	57.585885	40.071862	134.402880	146.086201	88.144113	163.403870	110.158006	70.415781	62.107422
2020-01-06	33.274864	149.475932	57.783164	40.141175	133.681833	147.908884	87.636535	163.240372	110.734639	70.652268	62.011530
2020-01-07	33.210163	149.849660	57.220908	42.011181	133.747865	147.654255	87.867168	162.007652	111.362158	70.806174	62.079539
2020-01-08	33.459721	150.514730	56.766306	41.336972	133.497770	148.942633	88.162891	162.926027	112.056666	70.908054	62.073946
...	...	...	...	...	...	...	...	...	...	...	...
2021-12-27	62.074051	205.875998	73.945442	53.746246	211.448817	226.758233	162.369727	241.019228	223.817161	91.164978	79.231415
2021-12-28	62.366943	205.453546	74.375756	53.521850	210.456668	225.649232	162.372701	241.798194	221.048634	91.419141	79.409990
2021-12-29	62.894154	205.907762	74.871976	53.013877	208.890593	227.965054	163.472283	242.726641	221.036981	91.380869	79.844394
2021-12-30	62.298603	205.951681	74.663324	52.124077	207.229121	227.991423	162.086330	244.383500	220.172799	91.160958	79.574097

Visualizing this data

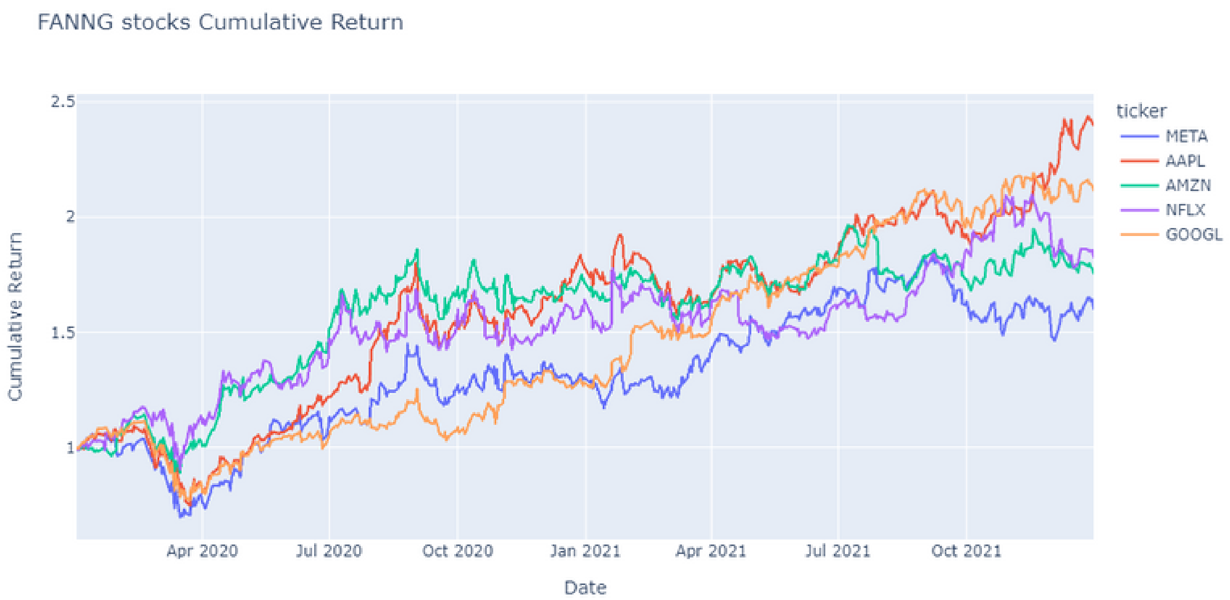


Calculating cumulative return by divide all data with the first row we get this



All sectors have been affected by the pandemic, but there are some sectors that have been affected the most, such as the energy sector and this makes sense because all countries were in lockdown. The energy sector recovered after one year, and most sectors recovered after about 6 months. Health care, real estate, and basic materials recovered more quickly than others, and after one year, the technology sector is booming with the industrial and basic materials sectors. And as we can see, basic materials, utilities, and customer staples were less affected.

And for the FANNG stocks



Meta, Google, and Apple suffered more than Netflix and Amazon in the market. Netflix and Amazon recovered quickly, and both companies are in the customer discretionary sector.

Algorithms and Techniques

Long-term memory will be used to predict the stock price. LSTM is the type of recurrent neural network which is used to find out the order dependencies in sequence, and because the stock price is ordered by date and the old price somehow affects the future price, so LSTM will be useful because it contains gate memory that stores the previous information. The processed data is fed into the LSTM model to predict the price of the chosen stock. In portfolio optimization, the PyPortfolioOpt library will be used to efficiently implement the portfolio optimization method. The diversified portfolio will be a pandas dataframe containing the company name, stock ticker, and the number of shares to buy.

Benchmark

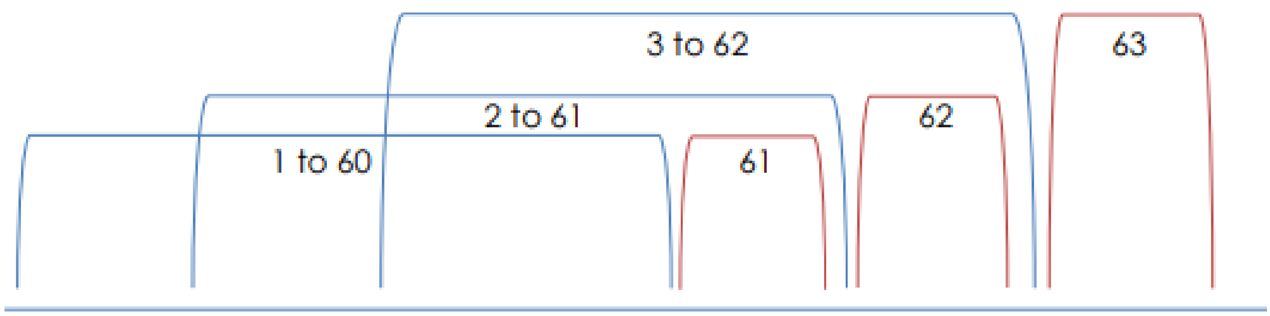
For this project, I will use the basic machine learning model of regression (linear regression) as a benchmark to see how the linear regression model differs from the LSTM model and measure the performance by the difference between the actual value and the predicted value for each model using the mean squared error.

METHODOLOGY

Data preprocessing

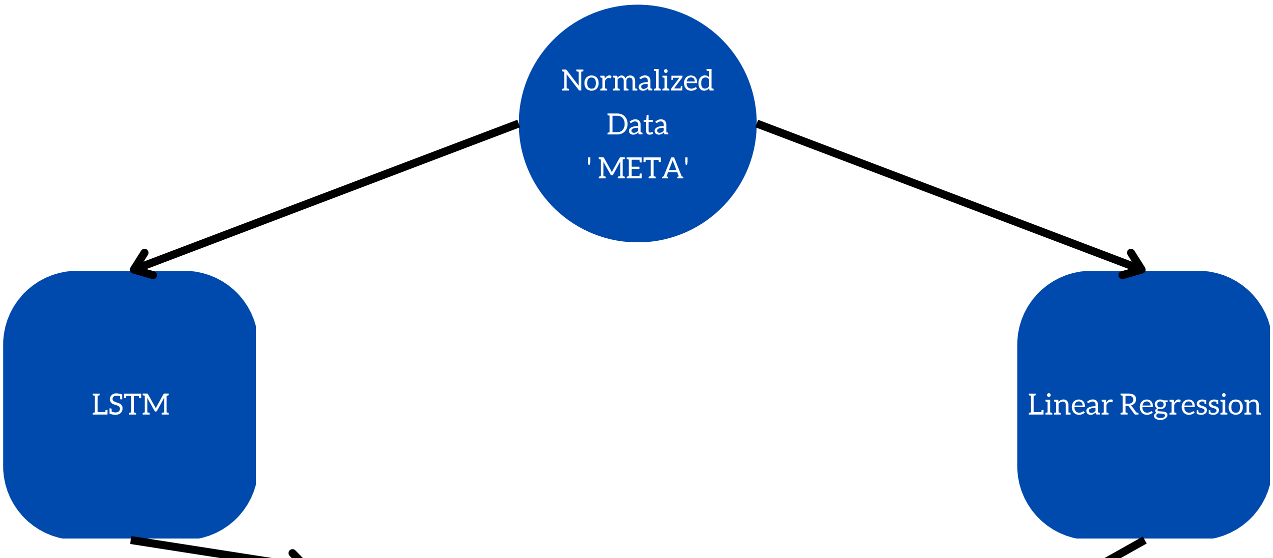
The method used to pre-process data obtained from Yahoo Finance for all 193 indices obtained from NASDAQ, uses 60 days as features for predicting the day 61 price.

Date	AAPL	META	AMZN	NFLX
2020-01-02	70.36	80.55	55.66	65.12
....	......	.....	....	....
2020-03-01	75.65	85.66	60.68	90.12
2020-03-02	76.00	90.12	59.66	96.2



After processing the data and getting X (independent features) and y (dependent feature) I split the data into data to train the model and data to test it
The percentage of training data is 75% and test data is 25%.
Next, I normalized the data using the MinMaxScaler function from Scikit-Learn Lib which makes all numbers between 0, 1
Now the data is ready to be passed to the models.

Models

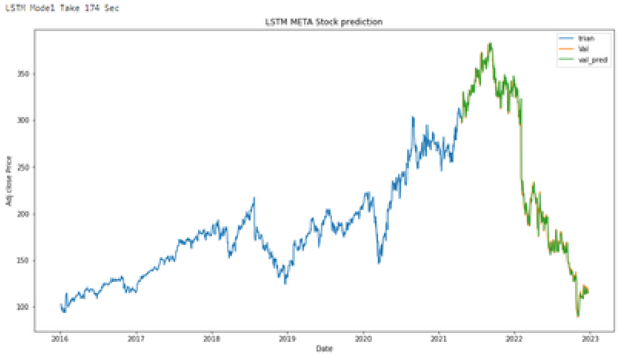



```python
def build_LSTM(df, ticker):
    X, y = data_prep(df,ticker)
    X_train, X_test, y_train, y_test = split_data(X,y)
    X_train, X_test, y_train,y_test, scaler = scale_data(X_train, X_test, y_train,y_test)
    # reshape the data to be 3D for LSTM model
    X_train = np.reshape(X_train, (X_train.shape[0],X_train.shape[1],1))
    X_test = np.reshape(X_test , (X_test.shape[0],X_test.shape[1],1))

  #Build LSTM model
    model=Sequential()
    model.add(LSTM(128,return_sequences=True,input_shape=(X_train.shape[1],1)))
    model.add(LSTM(64))
    model.add(Dense(50))
    model.add(Dense(1))
    # compile the model and train
    model.compile(optimizer='adam', loss = 'mean_squared_error')
    #earlyStop=EarlyStopping(monitor="val_loss",verbose=2,mode='min',patience=5)
    #model.fit(X_train,y_train,epochs=100,batch_size=32,validation_data=(X_test,y_test),callb
    print('Training....')
    tic = time.time()
    model.fit(X_train,y_train,epochs=20,batch_size=8,verbose=0)
    tac = time.time()
    predictions = model.predict(X_test)
    predictions = scaler.inverse_transform(predictions.reshape(-1,1))
    y_test1 = scaler.inverse_transform(y_test)
    MSE = mean_squared_error(y_test1, predictions)
    print('MSE : ', MSE)
    print(f'LSTM Model Take {math.ceil(tac-tic)} Sec')
    return predictions, X_train.shape[0],LSTM,X_test,y_test,scaler
```

```python
def build_lr(df, ticker):
    X, y = data_prep(df,ticker)
    X_train, X_test, y_train, y_test = split_data(X,y)
    X_train, X_test, y_train,y_test, scaler = scale_data(X_train, X_test, y_train,y_test)
    LR = LinearRegression()
    tic = time.time()
    LR.fit(X_train, y_train)
    tac = time.time()
    predictions = LR.predict(X_test)
    predictions = scaler.inverse_transform(predictions.reshape(-1,1))
    y_test1 = scaler.inverse_transform(y_test)
    MSE = mean_squared_error(y_test1, predictions)
    print('MSE : ', MSE)
    print(f'LR Model Take {math.ceil(tac-tic)} Sec')
    #print(X_train.shape,y_train.shape,X_test.shape,y_test.shape)
    return predictions, X_train.shape[0], LR,X_test,y_test,scaler
```

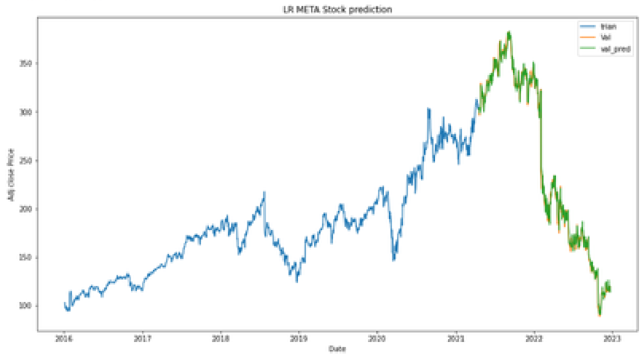
```python
predictions,split_index,LSTM,X_test,y_test,scaler = build_LSTM(FAANG,'meta')
plot_result(FAANG,'meta',predictions ,split_index,'LSTM')
```

```
Training....
14/14 [==============================] - 1s 41ms/step
MSE :  57.16863528743344
LSTM Model Take 174 Sec
```




```python
predictions,split_index,LR,X_test,y_test,scaler = build_lr(FAANG,'meta')
plot_result(FAANG,'meta',predictions ,split_index,'LR')
```

```
MSE :  61.69505302036034
LR Model Take 1 Sec
```



Repeating this for FANNG stocks and getting the result (Time and MSE)

Refine LSTM Model

I try to refine hyperparameters to get lower errors, I refine LSTM in two ways with different structures.

	Layer1	Layer2	Layer3	Layer4	epochs	Batch_size	Early_stop
LSTM	LSTM(128)	LSTM(64)	Dense(50)	Dense(1)	20	8	No
LSTM Refine1	LSTM(128)	LSTM(64)	Dense(50)	Dense(1)	15	4	yes
LSTM Refine2	LSTM(120)	LSTM(60)	Dense(50)	Dense(1)	50	32	No

Results

	META	AAPL	AMZN	NFLX	GOOGL	Time_Sec(Mean)
LR	62.54	9.24	13.95	182.83	6.66	< 1
LSTM	57.17	9.53	15.37	220.91	6.13	182
LSTM Refine 1	58.17	9.21	13.15	nan	nan	286
LSTM Refine 2	58.44	10.31	15.06	220.57	6.88	148

The results were great. The LSTM model performs better than the LR model on META and GOOG and has almost the same performance on AAPL and is very close together, but for NFLX LR performs better than LSTM. The LR model takes less than 1 second to train the model and forecast and is 182 times faster than the LSTM model.
LSTM refine 1 works better than LSTM but takes longer, while LSTM refine 2 is almost the same as LSTM but gets less time.

After training the models and getting the results, it's time to predict the future
To make a forecast for a month, for example, I need to forecast 30 days in the future for a particular stock, and because we use 60 days as features, the last 60 days in the test data are still unused and we need them to make a prediction.
So, I define a function that takes the test data and filters the last 60 days' data, and use it to make a forecast for the next day I used the forecast day as a feature to predict the day after, etc eventually returning the 30-day forecast.

```
def prediction_future(X_test,y_test,time_step=60):
    temp1 = X_test[-1][1:].tolist()
    predictions = y_test[-1].tolist()
    temp1.extend(predictions)

    for i in range(30):
        if len(temp1)==time_step:
            temp1 = np.array(temp1).reshape(-1,time_step)
            predic = LR.predict(temp1)
            temp1 = temp1.tolist()
            predic = predic[0].tolist()
            temp1[0].extend(predic)
            predictions.extend(predic)
            print('done')
        else:
            temp1 = temp1[0][1:]
            temp1 = np.array(temp1).reshape(-1,time_step)
            predic = LR.predict(temp1)
            temp1 = temp1.tolist()
            predic = predic[0].tolist()
            temp1[0].extend(predic)
            predictions.extend(predic)
    return predictions
```

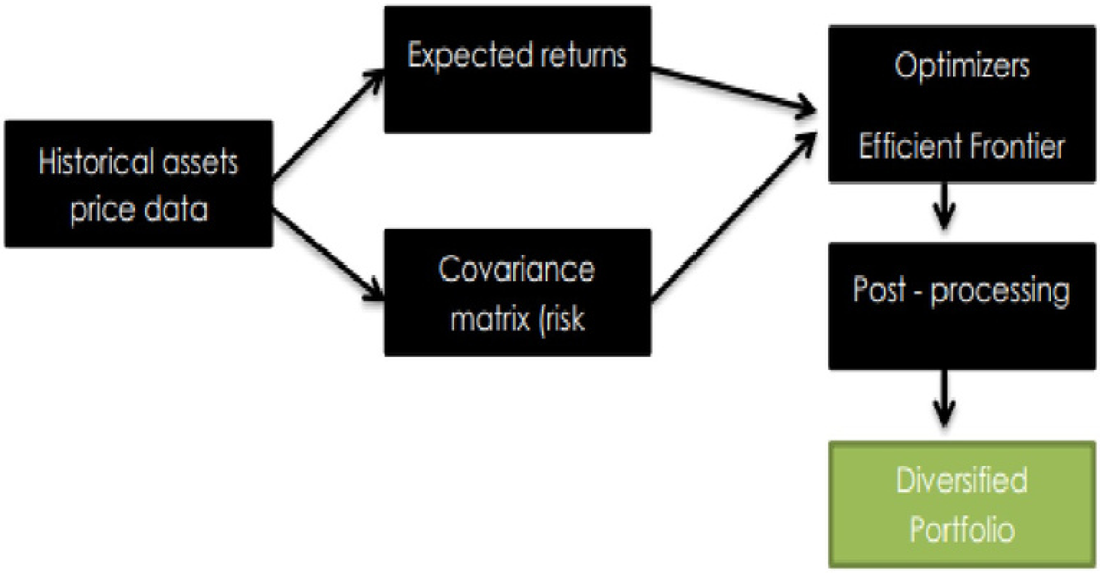
to print the result I build another function to print and calculate the percentages.

```
def show_result(predictions,ticker):
    week = float(predictions[7] / predictions[0] *100-100)
    two_week = float(predictions[14] / predictions[0] *100-100)
    month = float(predictions[-1] / predictions[0] *100-100)
    print(f'regarding to price today {round(float(predictions[0]),2)}$ for {ticker}')
    print(f'predicted stock value for 7 days  is  : {round(week,2)} %')
    print(f'predicted stock value for 14 days is  : {round(two_week,2)} %')
    print(f'predicted stock value for 30 days is  : {round(month,2)} %')
```

```
show_result(predictions,'ZBRA')
```

```
regarding to price today 248.22$ for ZBRA
predicted stock value for 7 days  is  : 0.61 %
predicted stock value for 14 days is  : -0.57 %
predicted stock value for 30 days is  : -2.27 %
```

Portfolio optimization



I'm creating a function that takes the adjusted closed data and does 1 of 3 things using the PyPortfolioOpt lib to build a portfolio with a maximum Sharpe ratio, minimizing risk or maximizing return, by calculating the expected return and covariance matrix and passing it to the effective frontier.

```python
def port_opt(df):
    intention = int(input("input\n1 for maximun sharpe ratio\n2 for minimun risk\n3 for maximum return\n"))
    total_value = int(input('How much would you invest $'))
    print()
    mu = mean_historical_return(df)
    S = risk_models.sample_cov(df)
    ef = EfficientFrontier(mu, S)
    ef.add_objective(objective_functions.L2_reg, gamma=1)
    if intention == 1:
        ef.max_sharpe()
    elif intention == 2 :
        ef.min_volatility()
    else:
        ef._max_return()
    cleaned_weights = ef.clean_weights()
    ef.portfolio_performance(verbose=True)
    print()
    latest_prices = get_latest_prices(df)
    da = DiscreteAllocation(cleaned_weights, latest_prices, total_portfolio_value=total_value)
    allocation, leftover = da.greedy_portfolio()
    name = pd.read_csv('nasdaq_screener.csv')[['Symbol','Name']]
    name.columns = ['Ticker','Name']
    alloc = pd.DataFrame(allocation.items(),columns=['Ticker','Numbers_of_shares'])
    alloc['Allocation'] = round(alloc['Numbers_of_shares'] / alloc['Numbers_of_shares'].sum(),4)*100
    alloc = pd.merge(alloc,name,how='left',on='Ticker')

    if len(alloc) > 0:
        print('Orderd Allocation\n',alloc.to_string())
        print('Left over : $',int(leftover))

    else:
        print('Low money')
        port_opt(df)
```

Build 3 portfolios

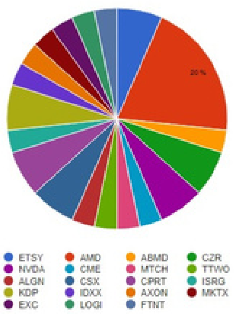
maximun sharpe ratio

```python
port_opt(data)
```

```
input
1 for maximun sharpe ratio
2 for minimun risk
3 for maximum return
1
How much would you invest $2000

Expected annual return: 40.4%
Annual volatility: 18.0%
Sharpe Ratio: 2.14

Orderd Allocation
   Ticker  Numbers_of_shares  Allocation                                     Name
0    ETSY                  2        6.67                    Etsy Inc. Common Stock
1     AMD                  6       20.00      Advanced Micro Devices Inc. Common Stock
2    ABMD                  1        3.33                  ABIOMED Inc. Common Stock
3     CZR                  2        6.67     Caesars Entertainment Inc. Common Stock
4    NVDA                  2        6.67           NVIDIA Corporation Common Stock
5     CME                  1        3.33        CME Group Inc. Class A Common Stock
6    MTCH                  1        3.33              Match Group Inc. Common Stock
7    TTWO                  1        3.33  Take-Two Interactive Software Inc. Common Stock
8    ALGN                  1        3.33          Align Technology Inc. Common Stock
9     CSX                  2        6.67              CSX Corporation Common Stock
10   CPRT                  2        6.67             Copart Inc. (DE) Common Stock
11   ISRG                  1        3.33        Intuitive Surgical Inc. Common Stock
12    KDP                  2        6.67          Keurig Dr Pepper Inc. Common Stock
13   IDXX                  1        3.33       IDEXX Laboratories Inc. Common Stock
14   AXON                  1        3.33           Axon Enterprise Inc. Common Stock
15   MKTX                  1        3.33      MarketAxess Holdings Inc. Common Stock
16    EXC                  1        3.33           Exelon Corporation Common Stock
17   LOGI                  1        3.33   Logitech International S.A. Ordinary Shares
18   FTNT                  1        3.33                  Fortinet Inc. Common Stock
Left over : $ 5
```

Max_Sharpe

Ticker	Name	Allocation
ETSY	Etsy, Inc.	6.67%
AMD	Advanced Micro Devices, Inc.	20.00%
ABMD	ABIOMED, Inc.	3.33%
CZR	Caesars Entertainment Inc	6.67%
NVDA	NVIDIA Corporation	6.67%
CME	CME Group Inc.	3.33%
MTCH	Match Group, Inc.	3.33%
TTWO	Take-Two Interactive Software, Inc.	3.33%
ALGN	Align Technology, Inc.	3.33%
CSX	CSX Corporation	6.67%
CPRT	Copart, Inc.	6.67%
ISRG	Intuitive Surgical, Inc.	3.33%
KDP	Keurig Dr Pepper Inc	6.67%
IDXX	IDEXX Laboratories, Inc.	3.33%
AXON	Axon Enterprise, Inc.	3.33%
MKTX	MarketAxess Holdings, Inc.	3.33%
EXC	Exelon Corp	3.33%
LOGI	Logitech International S.A.	3.33%
FTNT	Fortinet, Inc.	3.35%



Save portfolio »

Performance Summary

Portfolio	Initial Balance	Final Balance	CAGR	Stdev	Max. Drawdown	Sharpe Ratio	Sortino Ratio	Market Correlation
Max_Sharpe	$2,000	$3,222	61.09%	18.49%	-4.58%	2.60	10.06	0.80
SPDR S&P 500 ETF Trust	$2,000	$2,624	31.22%	12.90%	-6.38%	2.03	3.83	1.00

Portfolio Growth

Logarithmic scale   Inflation adjusted

minimun risk

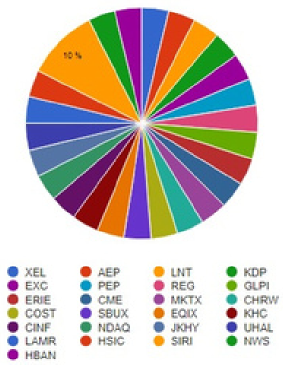
```
port_opt(data)
```

```
input
1 for maximun sharpe ratio
2 for minimun risk
3 for maximum return
2
How much would you invest $2000

Expected annual return: 12.5%
Annual volatility: 12.2%
Sharpe Ratio: 0.86

Orderd Allocation
    Ticker  Numbers_of_shares  Allocation                                       Name
0      XEL                  1        3.33              Xcel Energy Inc. Common Stock
1      AEP                  1        3.33  American Electric Power Company Inc. Common Stock
2      LNT                  1        3.33         Alliant Energy Corporation Common Stock
3      KDP                  1        3.33             Keurig Dr Pepper Inc. Common Stock
4      EXC                  1        3.33              Exelon Corporation Common Stock
5      PEP                  1        3.33                   PepsiCo Inc. Common Stock
6      REG                  1        3.33        Regency Centers Corporation Common Stock
7     GLPI                  1        3.33   Gaming and Leisure Properties Inc. Common Stock
8     ERIE                  1        3.33     Erie Indemnity Company Class A Common Stock
9      CME                  1        3.33           CME Group Inc. Class A Common Stock
10    MKTX                  1        3.33          MarketAxess Holdings Inc. Common Stock
11    CHRW                  1        3.33       C.H. Robinson Worldwide Inc. Common Stock
12    COST                  1        3.33       Costco Wholesale Corporation Common Stock
13    SBUX                  1        3.33            Starbucks Corporation Common Stock
14    EQIX                  1        3.33               Equinix Inc. Common Stock REIT
15     KHC                  1        3.33             The Kraft Heinz Company Common Stock
16    CINF                  1        3.33    Cincinnati Financial Corporation Common Stock
17    NDAQ                  1        3.33                    Nasdaq Inc. Common Stock
18    JKHY                  1        3.33       Jack Henry & Associates Inc. Common Stock
19    UHAL                  1        3.33                        Amerco Common Stock
20    LAMR                  1        3.33    Lamar Advertising Company Class A Common Stock
21    HSIC                  1        3.33               Henry Schein Inc. Common Stock
22    SIRI                  3       10.00          Sirius XM Holdings Inc. Common Stock
23     GEN                  1        3.33                Gen Digital Inc. Common Stock
24    HBAN                  1        3.33   Huntington Bancshares Incorporated Common Stock
25     NWS                  1        3.33       News Corporation Class B Common Stock
26    NWSA                  1        3.33       News Corporation Class A Common Stock
27    LSCC                  1        3.33   Lattice Semiconductor Corporation Common Stock
Left over : $ 5
```

Min_Risk

Ticker	Name	Allocation
XEL	Xcel Energy Inc	3.75%
AEP	American Electric Power Company Inc	3.75%
LNT	Alliant Energy Corp	3.75%
KDP	Keurig Dr Pepper Inc	3.75%
EXC	Exelon Corp	3.75%
PEP	Pepsico Inc	3.75%
REG	Regency Centers Corp	3.75%
GLPI	Gaming and Leisure Properties, Inc.	3.75%
ERIE	Erie Indemnity Company	3.75%
CME	CME Group Inc.	3.75%
MKTX	MarketAxess Holdings, Inc.	3.75%
CHRW	C.H. Robinson Worldwide, Inc.	3.75%
COST	Costco Wholesale Corporation	3.75%
SBUX	Starbucks Corporation	3.75%
EQIX	Equinix, Inc.	3.75%
KHC	The Kraft Heinz Company	3.75%
CINF	Cincinnati Financial Corporation	3.75%
NDAQ	Nasdaq, Inc.	3.75%
JKHY	Jack Henry & Associates, Inc.	3.75%
UHAL	Amerco	3.75%
LAMR	Lamar Advertising Company	3.75%
HSIC	Henry Schein, Inc.	3.75%
SIRI	Sirius XM Holdings Inc.	10.00%
NWS	News Corporation	3.75%
HBAN	Huntington Bancshares Incorporated	3.75%

Save portfolio »



Since the Portfolio Visualizer free account only accepts 25 indicators for backtesting, so I chose the first 25 and adjusted the allocation percentages.

Performance Summary

Portfolio	Initial Balance	Final Balance	CAGR	Stdev	Max. Drawdown	Sharpe Ratio	Sortino Ratio	Market Correlation
Min_Risk	$2,000	$2,540	27.00%	8.22%	-1.48%	2.73	13.30	0.80
SPDR S&P 500 ETF Trust	$2,000	$2,624	31.22%	12.90%	-6.38%	2.03	3.83	1.00

Portfolio Growth

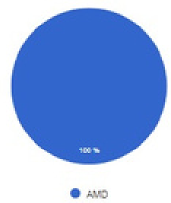
maximum return

```
port_opt(data)
```

```
input
1 for maximun sharpe ratio
2 for minimun risk
3 for maximum return
3
How much would you invest $2000

Expected annual return: 88.7%
Annual volatility: 69.8%
Sharpe Ratio: 1.24

Orderd Allocation
   Ticker  Numbers_of_shares  Allocation                                 Name
0    AMD                108       100.0  Advanced Micro Devices Inc. Common Stock
Left over : $ 6
```

Ticker	Name	Allocation
AMD	Advanced Micro Devices, Inc.	100.00%

Save portfolio »



Performance Summary

Portfolio	Initial Balance	Final Balance	CAGR	Stdev	Max. Drawdown	Sharpe Ratio	Sortino Ratio	Market Correlation
Max_Return	$2,000	$4,969	148.43%	38.49%	-7.82%	2.56	11.07	0.59
SPDR S&P 500 ETF Trust	$2,000	$2,624	31.22%	12.90%	-6.38%	2.03	3.83	1.00

Portfolio Growth

By using the PyPortfolioOpt library to build a diversified portfolio and using the portfolio visualizer site for backtesting the portfolio using the adjusted close price data of 193 tickers for mega and large companies from 2016 to 2018 for making a portfolio and backtest this portfolio for 2019, we found

- The Max Sharpe ratio portfolio achieved 21% more than expected with the same risk in the backtest and with a close Sharpe ratio
- The Min Risk Portfolio achieved a double return from 12.5% as expected to 27% and lower risk than expected and with a higher Sharpe ratio
- The Max Return portfolio from 88% expected, it achieved 148% return with lower risk from 70% to 38% and Sharpe ratio 2.56 in the backtest but still very risky

Justification

- We can see that we can predict the future prices of stocks just by looking at their historical prices, there are many ML models that can predict stock prices.
- Sometimes we found LSTM outperforms the LR model but LR is generally better on this type of data, only the adjusted closing price has been used with these preprocessing techniques and LR is better because of less error and less time <1 second. LSTM can work better on other data types such as using all the features from the dataset such as open, high, low, and close prices.
- And in portfolio optimization, we get satisfactory results from all types of portfolios, and the Max Returns portfolio yielded a very good return, but this is not a diversified portfolio, so it's very risky to invest all your money in one stock or even one sector.
- A diversified portfolio reduces risk through a variety of stocks that are not correlated.

CONCLUSION

Exploring the data, I see that there are some sectors that have been affected less, in other words, they are non-cyclical and therefore relatively stable in both strong or weak economies such as basic materials, utilities, consumer staples, customer discretionary, where the energy sector fell during the pandemic, the other sector Like technology thrives.

Reflection

Set up environment
- installing necessary libraries(pandas, NumPy, Matplotlib, Sci-kit-learn, Keras, PyPortfolioOpt, pandas-DataReader, yfinance)

Download the data
- downloading the adjusted close price for 193 tickers from yahoo finance API and the date between today whenever it is and 6 years before.

Preprocessing the data
- take 60 days as independent features and the day after as a dependent feature from day 2 to 61 as X and y is 62 and so on.
- split the data into 75% training data and 25% as testing data.
- Normalize the data with MinMaxScaler().

- **Develop a benchmark model (LR).**
- **Develop an LSTM model.**
- **Refine the LSTM model.**
- **Plot the result, analyzing and describing it.**
- **Build A portfolio optimization method**

The thing I found interesting is that the linear regression model works better than expected, sometimes LR outperforms the LSTM model in MSE and certainly in time. Time is very important because if I want to make this project dynamic for users, training and predicting a new indicator in just a second is very powerful.

Improvements

- Try to build more models and get fewer errors than LR and LSTM, and try different preprocessing techniques for data and use more features than the adjusted close price
- Build a web application that facilitates the process for users to access the model
- Improve the portfolio optimization to let the user exclude some sectors from processing and if he wants to build a portfolio in specific stocks