JP Morgan Mini Project 1 - Revisited

In [73]:
```
import pandas as pd
import math
import operator
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#plt.style.use("ggplot")
plt.style.use("dark_background")
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.api import OLS
import warnings
import scipy
warnings.filterwarnings('ignore')
```

In [74]:
```
# loading dataset
nifty = pd.read_csv("NIFTY.csv")
nifty.index = pd.to_datetime(nifty.Date)
nifty.describe()
```

Out[74]:

	Open	High	Low	Close
count	250.000000	250.000000	250.000000	250.000000
mean	13367.213404	13435.472791	13261.676603	13355.313385
std	1706.866327	1713.182699	1694.558284	1706.141755
min	10323.799810	10401.049810	10267.349610	10302.099610
25%	11539.587890	11584.312257	11454.687260	11523.212408
50%	13761.500000	13831.399905	13660.174805	13754.899905
75%	14834.162603	14917.262700	14704.262452	14849.337402
max	15915.349610	15915.650390	15842.400390	15869.250000

In [75]:
```
# loading dataset
sgx_nifty = pd.read_csv("SGX Nifty.csv")
sgx_nifty.index = pd.to_datetime(sgx_nifty.Date)
sgx_nifty.describe()
```

Out[75]:

	Open	High	Low	Close
count	250.000000	250.000000	250.000000	250.000000
mean	2051.822601	2075.947193	2022.966408	2047.625007
std	124.753095	125.641820	122.249624	122.931349
min	1720.000000	1741.000000	1695.550049	1704.099976
25%	1959.399994	1977.450012	1932.000000	1956.375030
50%	2031.750000	2060.575074	2000.625000	2028.024964
75%	2140.762512	2166.500000	2105.500000	2134.025086
max	2325.000000	2369.350098	2310.550049	2324.550049

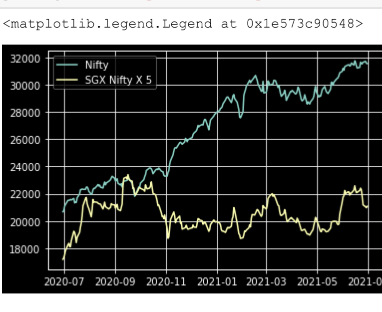
Exploratory Analysis

Vizualizing NIFTY and SGX NIFTY

In [76]:
```
#Calculating mid value
nifty["Mid"]=nifty["High"]+nifty["Low"]
sgx_nifty["Mid"]=sgx_nifty["High"]+sgx_nifty["Low"]
```

In [77]:
```
plt.plot(nifty.index, nifty["Mid"])
plt.plot(sgx_nifty.index, sgx_nifty["Mid"]*5)
plt.legend(["Nifty", "SGX Nifty X 5"])
```

Out[77]: `<matplotlib.legend.Legend at 0x1e573c90548>`

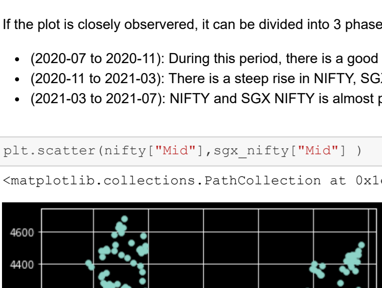


If the plot is closely observed, it can be divided into 3 phases:

- (2020-07 to 2020-11): During this period, there is a good rise in SGX NIFTY towards the middle, and a gradual rise in NIFTY
- (2020-11 to 2021-03): There is a steep rise in NIFTY, SGX NIFTY doesn't vary much, but their smaller fluctions coincides.
- (2021-03 to 2021-07): NIFTY and SGX NIFTY is almost parallel

In [78]:
```
plt.scatter(nifty["Mid"],sgx_nifty["Mid"] )
```

Out[78]: `<matplotlib.collections.PathCollection at 0x1e57382fac8>`

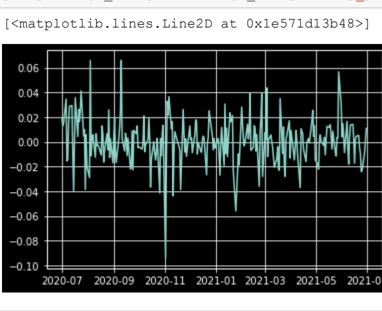


Vizualizing returns for NIFTY and SGX NIFTY

Here Daily price change is assumed to be returns (i.e) ((previous day closing price) - (current day closing price))/(current day closing price)

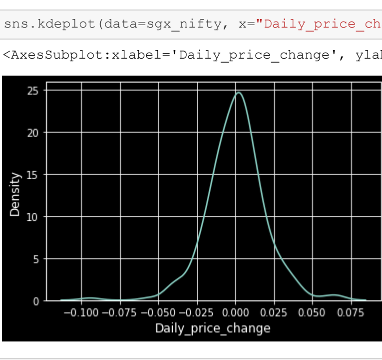
In [79]:
```
sgx_nifty["Daily_price_change"]=sgx_nifty["Close"].diff(periods=1)/sgx_nifty["Close"]
plt.plot(sgx_nifty.index, sgx_nifty["Daily_price_change"])
```

Out[79]: `[<matplotlib.lines.Line2D at 0x1e571d13b48>]`



In [80]:
```
sns.kdeplot(data=sgx_nifty, x="Daily_price_change")
```

Out[80]: `<AxesSubplot:xlabel='Daily_price_change', ylabel='Density'>`

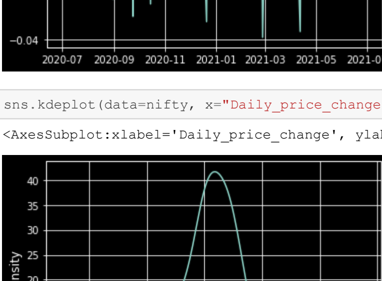


In [81]:
```
nifty["Daily_price_change"]=nifty["Close"].diff(periods=1)/nifty["Close"]
plt.plot(nifty.index, nifty["Daily_price_change"])
```

Out[81]: `[<matplotlib.lines.Line2D at 0x1e5720dc708>]`

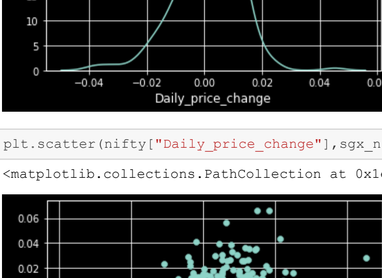
In [82]:
```
sns.kdeplot(data=nifty, x="Daily_price_change")
```

Out[82]: `<AxesSubplot:xlabel='Daily_price_change', ylabel='Density'>`



In [83]:
```
plt.scatter(nifty["Daily_price_change"],sgx_nifty["Daily_price_change"] )
```

Out[83]: `<matplotlib.collections.PathCollection at 0x1e5739a9bc8>`



In [84]:
```
# Import dependencies
import plotly
import plotly.graph_objs as go

# Configure Plotly to be rendered inline in the notebook.
plotly.offline.init_notebook_mode()

# Configure the trace.
trace = go.Scatter3d(
    x=nifty["Daily_price_change"][1:],  # <-- Put your data instead
    y=sgx_nifty["Daily_price_change"][1:] ,  # <-- Put your data instead
    z=nifty.index.values[1:],  # <-- Put your data instead
    mode='markers'
)

# Configure the layout.
layout = go.Layout(
    margin={'l': 0, 'r': 0, 'b': 0, 't': 0}
)

data = [trace]

plot_figure = go.Figure(data=data, layout=layout)
print("x:","Nifty Daily Price Change")
print("y:","SGX Nifty Daily Price Change")
print("z:","Time Daily Price Change")


# Render the plot.
plotly.offline.iplot(plot_figure)
```

```
x: Nifty Daily Price Change
y: SGX Nifty Daily Price Change
z: Time Daily Price Change
```

Attempt 1: Using Linear Regression on Returns

In [85]:
```
OLS(nifty["Daily_price_change"][1:],sgx_nifty[["Daily_price_change"]][1:]).fit().summary()
```

Out[85]:

OLS Regression Results

Dep. Variable:	Daily_price_change	**R-squared (uncentered):**	0.295
Model:	OLS	**Adj. R-squared (uncentered):**	0.292
Method:	Least Squares	**F-statistic:**	103.9
Date:	Sun, 15 Aug 2021	**Prob (F-statistic):**	1.32e-20
Time:	15:57:35	**Log-Likelihood:**	818.07
No. Observations:	249	**AIC:**	-1634.
Df Residuals:	248	**BIC:**	-1631.
Df Model:	1		
Covariance Type:	nonrobust		

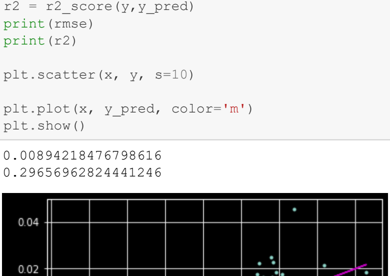
	coef	std err	t	P>|t|	[0.025	0.975]
Daily_price_change	0.3107	0.030	10.194	0.000	0.251	0.371

Omnibus:	10.510	**Durbin-Watson:**	1.838
Prob(Omnibus):	0.005	**Jarque-Bera (JB):**	21.531
Skew:	-0.082	**Prob(JB):**	2.11e-05
Kurtosis:	4.431	**Cond. No.**	1.00

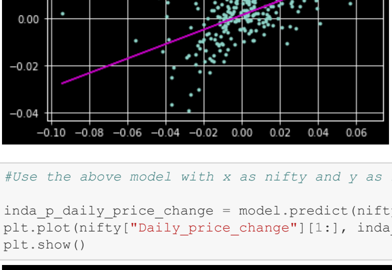
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [86]:
```
model = LinearRegression()
x=sgx_nifty[["Daily_price_change"]][1:]
y=nifty["Daily_price_change"][1:]
model.fit(x,y )
y_pred = model.predict(x)

rmse = np.sqrt(mean_squared_error(y,y_pred))
r2 = r2_score(y,y_pred)
print(rmse)
print(r2)

plt.scatter(x, y, s=10)

plt.plot(x, y_pred, color='m')
plt.show()
```

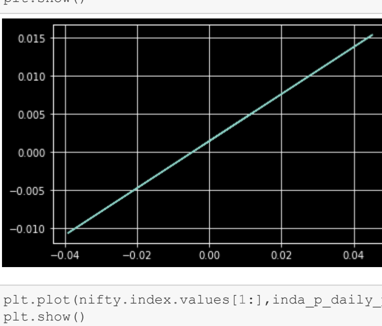
```
0.00894218476798616
0.29656962824441246
```



In [87]:
```
#Use the above model with x as nifty and y as inda.p

inda_p_daily_price_change = model.predict(nifty[["Daily_price_change"]][1:])
plt.plot(nifty["Daily_price_change"][1:], inda_p_daily_price_change)
plt.show()
```



In [88]:
```
plt.plot(nifty.index.values[1:],inda_p_daily_price_change)
plt.show()
```



In [89]:
```
# Standard deviation of inda.p
inda_p_daily_price_change.std()
```

Out[89]: `0.0032833560260221474`

In [90]:
```
inda_p_daily_price_change.mean()
```

Out[90]: `0.0019343795412585592`

Since linear regression was used to fit the model, correlation between Nifty and Inda.P is 1

In [91]:
```
nifty["Daily_price_change"][1:].std()
```

Out[91]: `0.010683336663063283`

Assumptions and Approach:

- Standard Deviation of Inda.P returns = 0.0033
- Standard Deviation of Nifty returns = 0.0107
- Covariance = Correlation X sigma1 X sigma2 = 1x0.0033x0.0107

In [92]:
```
0.0033*0.0107
```

Out[92]: `3.531e-05`

Covariance = 3.531x10^(-5)

Attempt 2: Using Standard deviation and Correlation:

Q1

- **Assumption:** Inda.P and Nifty have same correlation as Nifty and SGX Nifty = rho
- **Given:**
  - Standard Deviation for Inda.P returns = 1.5 x 0.01
  - Closing price at 30th June 2021 = USD 110
- **Approach:**
  - Find standard deviation of Inda.P Daily Price Change = sigma1 = 1.5 x 0.01
  - Find standard deviation of Nifty Daily Price Change = sigma2
  - Calculation rho
  - Covariance of Inda.P and Nifty = rho x sigma1 x sigma2

In [93]:
```
#Pearson correlation for Nifty and SGX Nifty
rho=nifty["Daily_price_change"][1:].corr(sgx_nifty["Daily_price_change"][1:])
rho
```

Out[93]: `0.5445820675016873`

In [94]:
```
#Standard deviation of Inda.P Daily Price Change (i.e) sigma1
sigma1=1.5*0.01
sigma1
```

Out[94]: `0.015`

In [95]:
```
#Standard deviation of Nifty Daily Price Change (i.e) sigma2
sigma2=nifty["Daily_price_change"].std()
sigma2
```

Out[95]: `0.010683336663063283`

In [96]:
```
#Covariance of Inda.P and Nifty
covariance=rho*sigma1*sigma2
covariance
```

Out[96]: `8.726930351681369e-05`

Q2

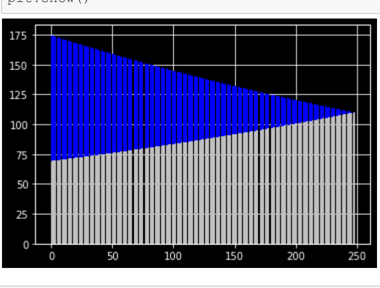
- **Assumptions:**
  - Mean of Daily Price Change for Inda.P is 0
- **Given:**
  - Standard Deviation for Daily Price Change for Inda.P = sigma1 = 1.5x0.01
  - Closing price at 30th June 2021 = USD 110
  - z=1.96 for 95% CI
- **Approach:**
  - Calculate Confidence Interval for Daily Price Change of Inda.P using:
$$\mathrm{CI} = \bar{X} \pm (z \times \sigma_{\bar{X}})$$
$$\sigma_{\bar{X}} = \frac{s}{\sqrt{N}}$$
  - We can now back track to find closing price of Inda.P at 95% confidence interval.

In [97]:
```
#Confidence interval for daily price change of Inda.P
N=nifty["Daily_price_change"].count()
CI=(1.96*sigma1)/math.sqrt(N)
CI
```

Out[97]: `0.0018631492965419652`

In [98]:
```
#Range of Closing prices
Inda_P_low=[]
Inda_P_high=[]
Inda_P_low.insert(0,110)
Inda_P_high.insert(0,110)
for i in range(1,N):
    Inda_P_low.insert(0,Inda_P_low[0]-Inda_P_low[0]*CI)
    Inda_P_high.insert(0,Inda_P_high[0]+Inda_P_high[0]*CI)
```

In [100]:
```
plt.bar(list(range(N)), Inda_P_high, color='b')#bottom=Inda_P_low,
plt.bar(list(range(N)), Inda_P_low, color='#c0c0c0')

plt.show()
```



In [ ]: