Group Work Project # 1

Data Collection & Processing

For our handbook we are going to use 5 years of daily return data from different assets such as NVDIA, AMD, META, TESLA, AMAZON, S&P 500,Dow Jones Index, Tresuery yield index 30yrs.

> We are using Yahoo Finance API to to collect our data. After that we are calculating the daily returns and drpoping missing data and properfly formatting it.

In [1]:
```
!pip install yfinance
```

```
Requirement already satisfied: yfinance in /usr/local/lib/python3.10/dist-packages
(0.2.37)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.10/dist-pac
kages (from yfinance) (1.5.3)
Requirement already satisfied: numpy>=1.16.5 in /usr/local/lib/python3.10/dist-pac
kages (from yfinance) (1.25.2)
Requirement already satisfied: requests>=2.31 in /usr/local/lib/python3.10/dist-pa
ckages (from yfinance) (2.31.0)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.10/di
st-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in /usr/local/lib/python3.10/dist-packa
ges (from yfinance) (4.9.4)
Requirement already satisfied: appdirs>=1.4.4 in /usr/local/lib/python3.10/dist-pa
ckages (from yfinance) (1.4.4)
Requirement already satisfied: pytz>=2022.5 in /usr/local/lib/python3.10/dist-pack
ages (from yfinance) (2023.4)
Requirement already satisfied: frozendict>=2.3.4 in /usr/local/lib/python3.10/dist
-packages (from yfinance) (2.4.0)
Requirement already satisfied: peewee>=3.16.2 in /usr/local/lib/python3.10/dist-pa
ckages (from yfinance) (3.17.1)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /usr/local/lib/python3.1
0/dist-packages (from yfinance) (4.12.3)
Requirement already satisfied: html5lib>=1.1 in /usr/local/lib/python3.10/dist-pac
kages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-pac
kages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.10/dist-packages
(from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-pack
ages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.1
0/dist-packages (from pandas>=1.3.0->yfinance) (2.8.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.
10/dist-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pack
ages (from requests>=2.31->yfinance) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dis
t-packages (from requests>=2.31->yfinance) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dis
t-packages (from requests>=2.31->yfinance) (2024.2.2)
```

In [2]:
```
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms
import statsmodels.api as sm
import scipy.stats as stats
from scipy.stats import norm
from scipy.stats import skewnorm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression




plt.rcParams["figure.figsize"] = (12, 9)
```

In [3]:
```
def assets(tickers):
  for i in tickers:
    asset_df = yf.download(tickers, start='2019-03-06', end='2024-03-06', progress=
    asset_df = asset_df['Adj Close'].diff().dropna()
    asset_df.columns = tickers

  asset_df.reset_index(inplace=True)

  return asset_df
```

In [4]:
```
tickers = ['NVDA', 'AMD', 'META', 'TSLA', 'AMZN', '^GSPC', '^DJI', '^TYX' ]
```

In [5]:
```
main_df = assets(tickers)
```

In [6]:
```
main_df
```

Out[6]:

	Date	NVDA	AMD	META	TSLA	AMZN	^GSPC	^DJI	^T
0	2019-03-07	-0.330000	-2.150002	-3.376404	-0.692776	0.023333	-200.230469	-22.520020	-0.0
1	2019-03-08	-0.070000	-0.257500	0.469498	0.342667	0.503334	-22.990234	-5.859863	-0.0
2	2019-03-11	0.949999	2.490997	2.467392	2.607170	0.452000	200.640625	40.229980	0.0
3	2019-03-12	0.530001	0.124001	-0.149857	0.342651	-0.504000	-96.220703	8.219971	-0.0
4	2019-03-13	-0.110001	0.885498	1.448471	1.514641	0.373333	148.230469	19.399902	0.0
...	...	...	...	...	...	...	...	...	
1253	2024-02-28	-1.460007	-0.379990	-3.029999	-10.379517	2.309998	-23.390625	-8.420410	-0.0
1254	2024-02-29	15.990005	3.599991	6.110016	14.489319	-0.159988	47.371094	26.510254	-0.0
1255	2024-03-01	10.110001	1.460007	12.169983	31.668518	0.759995	90.988281	40.810059	-0.0
1256	2024-03-04	2.720001	-0.639999	-4.109985	29.578613	-14.500000	-97.550781	-6.129883	0.0
1257	2024-03-05	-0.229996	-3.460007	-7.970001	7.309998	-7.399994	-404.636719	-52.300293	-0.0

1258 rows × 9 columns

In [7]:
```
main_df.columns
```

Out[7]:
```
Index(['Date', 'NVDA', 'AMD', 'META', 'TSLA', 'AMZN', '^GSPC', '^DJI', '^TYX'], dt
ype='object')
```

In [8]:
```
main_df.rename(columns= {'^GSPC' : 'GSPC', '^DJI' : 'DJI', '^TYX' : 'TYX'}, inplace
print(main_df.columns)
```

```
Index(['Date', 'NVDA', 'AMD', 'META', 'TSLA', 'AMZN', 'GSPC', 'DJI', 'TYX'], dtype
='object')
```

The Four challenges that we have picked are :

- **Multicollinearity** [Team Member 1]
- **Skewness** [Team Member 2]
- **Sensitivity to outliers** [Team Member 3]
- **Overfitting** [Team Member 1,2,3]

> **Throughout this handbook we are going to address:-**

1. Definition: Technical definition using formulas or equations
2. Description: Written explanation (1–2 sentences)
3. Demonstration: Numerical example using real-world data (or simulated data if not found)

4. Diagram: Visual example using real-world data (using same data as above)
5. Diagnosis: How to recognize or test that the problem exists
6. Damage: Clear statement of the damaged caused by the problem
7. Directions: Suggested models that can address this

Multicollinearity

Defination

Multicollinearity is a phenomenon when two or more variables which are independent are highly correlated to each other. which can cause estimation issue to our regression coefficient and make our model unreliable (Et. al.).

Let's take an example of a linear regression in which we have "n" indepnedent variable and "c" observation and one dependent variable :

$$\mathrm{Y}_c = \beta_0 + \beta_1\mathrm{X1} + \beta_2\mathrm{X2} + \ldots + \beta_c\mathrm{Xn} + \epsilon_c$$

where:

- Y is the dependent variable for c obersvations. ('NVDA' in our case)
- X...X_n is the independent varaible for c obersvations. ('AMD', 'META', 'TSLA', 'AMZN', '^GSPC', '^DJI', '^TYX' in our case)
- β_0...β_c are the regression coefficient.
- ϵ_c is the idiosyncratic error observations.

So, Multicollinearity happens when there are high correlation between the independent varibles which in this case are X terms.

Description

As a quant on the derivatives desk we have to work with lots of financial data which is in time series and as the data is random it is not in our control how the data is generated. And as we cannot control how financial data is generated, many times multiple financial variables move together and when two or more variables moves together we call it multicollinerity.

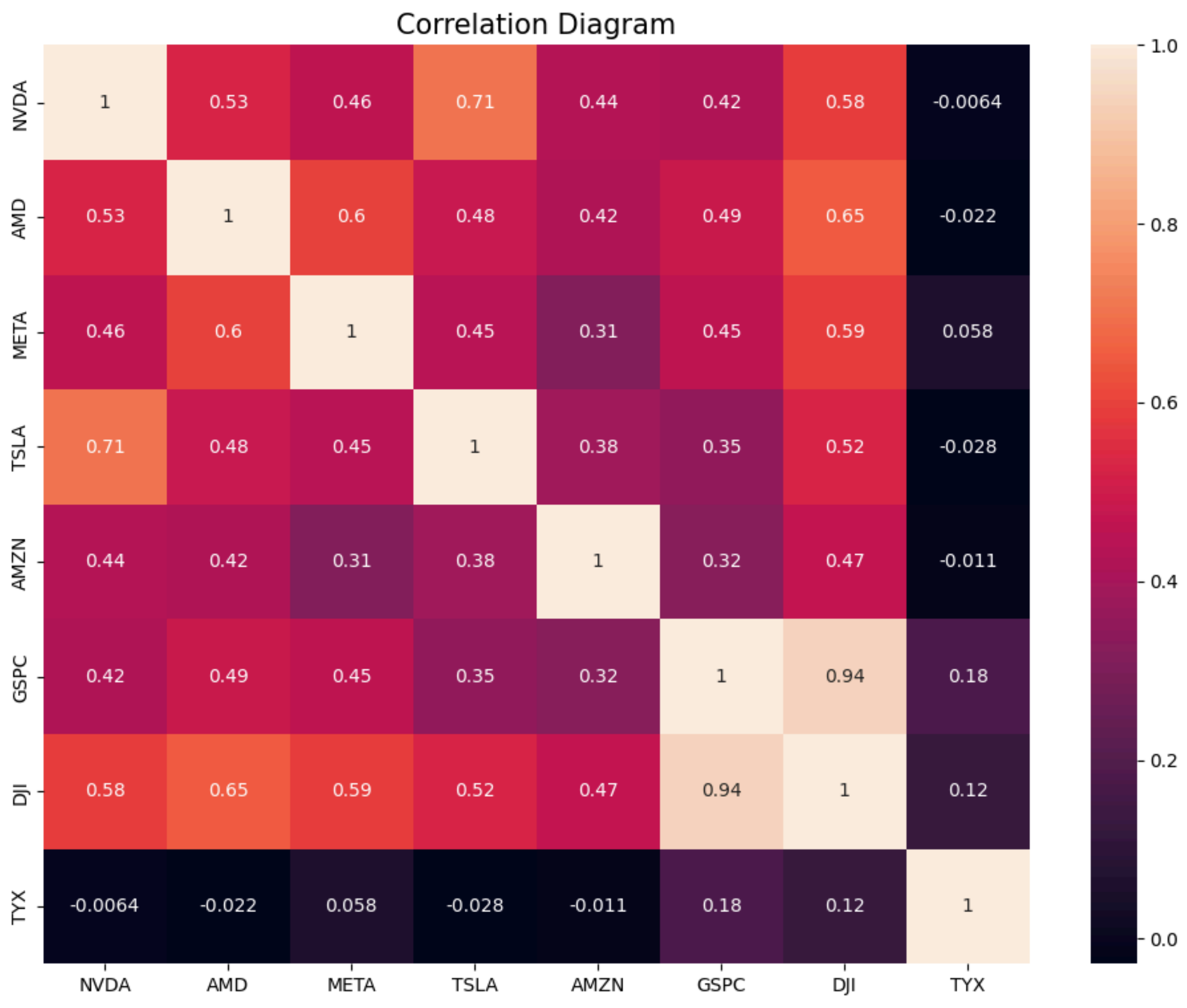
In our case we are going to find if there is multicollinerity between NVDIA, TESLA, META, AMAZON, AMD, S&P500, Dow jones, 30yr US tresury daily returns. Due to current volatile rally in NVDIA we want to analyze it's peers as well as some tech company with broader market index to understand the NVDIA market volatilty better to hedge against any risk.

Demonstration

Now we will try to is there any multicollinerity between out fianncial variables. to achieve this we are goint to plot correlation matrix which is the best way to find multicollinerity, A corellation matrix provides us all the correlations of all our variables.

Let's look how NVDA daily return model get affected by multicollinerity:

In [9]:
```
# Correlation plot of all the variables
data = main_df[
    [
        'NVDA',
        'AMD',
        'META',
        'TSLA',
        'AMZN',
        'GSPC',
        'DJI',
        'TYX',
    ]
]
c = data.corr()
sns.heatmap(c, annot=True)
plt.title("Correlation Diagram", fontsize=15)
plt.show()
```



	NVDA	AMD	META	TSLA	AMZN	GSPC	DJI	TYX
NVDA	1	0.53	0.46	0.71	0.44	0.42	0.58	-0.0064
AMD	0.53	1	0.6	0.48	0.42	0.49	0.65	-0.022
META	0.46	0.6	1	0.45	0.31	0.45	0.59	0.058
TSLA	0.71	0.48	0.45	1	0.38	0.35	0.52	-0.028
AMZN	0.44	0.42	0.31	0.38	1	0.32	0.47	-0.011
GSPC	0.42	0.49	0.45	0.35	0.32	1	0.94	0.18
DJI	0.58	0.65	0.59	0.52	0.47	0.94	1	0.12
TYX	-0.0064	-0.022	0.058	-0.028	-0.011	0.18	0.12	1

Diagram

As you can see we have plotted correaltion diagram of our variables now we will try dignose any issue in the next section.

Diagnosis

Now let's analyze the diagram :

1. We can see in the matrix , NVDA daily returns have a positive correlation with AMD, TSLA and DJI (Dow jones Index) daily returns, which mean these three vraibales can expreail NVDA daily return pretty good.

2. No we will analyze independent variables, TSLA, AMZN, TYX have very low correaltion with other independent varaibles.

3. After analysis we can see that GSCP and DJI daily retruns , AMD daily returns META and DJI daily returns are highly correlated.

Damage

Now we will try to understand what damage will be done to our model if we have multicollinearity :

- If our independent varibales are highly correlated, the std errors, variances and covariances of the coefficients from the regression may become large and due to that the confidence intervals for coefficient estimates will be wide making our estimates less precise.

- Our estimates are less likely to be significant as we have high std errors. we may get high R^2 and Adjusted R^2 because our model is good. but the collinerity issue makes it harder to isolate the individual impact from the independent varibales in the model.

- As we discussed it is harder to seperate each independent varibles impact in the model, our model can still be used to predict if the new data have same colliner issue as the sample data on which is model is build.

To understand further we will run our model and see how multicollinearity affects our model results.

First we will run a simple test regression model just for comparision purpose in this model we will only use three independent varaible i.e GSCP, DJI, TYX to understant its effect on NVDA daily returns.

In [10]:
```
test_model = smf.ols(
    "NVDA ~ GSPC + DJI + TYX",
    data=main_df,
).fit()

test_model.summary()
```

Out[10]:

OLS Regression Results

Dep. Variable:	NVDA	**R-squared:**	0.485
Model:	OLS	**Adj. R-squared:**	0.484
Method:	Least Squares	**F-statistic:**	394.3
Date:	Wed, 13 Mar 2024	**Prob (F-statistic):**	2.34e-180
Time:	13:29:35	**Log-Likelihood:**	-2730.5
No. Observations:	1258	**AIC:**	5469.
Df Residuals:	1254	**BIC:**	5490.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.0472	0.060	0.788	0.431	-0.070	0.165
GSPC	-0.0094	0.001	-18.287	0.000	-0.010	-0.008
DJI	0.1061	0.004	26.938	0.000	0.098	0.114
TYX	-0.0993	1.065	-0.093	0.926	-2.189	1.990

Omnibus:	280.489	**Durbin-Watson:**	2.021
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2024.422
Skew:	0.833	**Prob(JB):**	0.00
Kurtosis:	8.987	**Cond. No.**	6.39e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.39e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Now we will run NVDA Daily return model with all Seven Independent varaibles :

In [11]:
```
# NVDA daily return Regression Model with Seven Independent Variables

model = smf.ols(
    "NVDA ~ AMD + META + TSLA + AMZN + GSPC + DJI + TYX",
    data=main_df,
).fit()

model.summary()
```

Out[11]:

OLS Regression Results

Dep. Variable:	NVDA	**R-squared:**	0.592
Model:	OLS	**Adj. R-squared:**	0.589
Method:	Least Squares	**F-statistic:**	258.6
Date:	Wed, 13 Mar 2024	**Prob (F-statistic):**	7.64e-238
Time:	13:29:35	**Log-Likelihood:**	-2585.3
No. Observations:	1258	**AIC:**	5187.
Df Residuals:	1250	**BIC:**	5228.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0182	0.054	-0.339	0.735	-0.123	0.087
AMD	0.0282	0.029	0.964	0.335	-0.029	0.086
META	-0.0032	0.011	-0.283	0.777	-0.025	0.019
TSLA	0.1675	0.009	17.923	0.000	0.149	0.186
AMZN	0.0192	0.009	2.223	0.026	0.002	0.036
GSPC	-0.0041	0.001	-6.370	0.000	-0.005	-0.003
DJI	0.0505	0.006	8.116	0.000	0.038	0.063
TYX	0.1096	0.954	0.115	0.909	-1.762	1.981

Omnibus:	231.106	**Durbin-Watson:**	1.977
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2226.997
Skew:	0.548	**Prob(JB):**	0.00
Kurtosis:	9.425	**Cond. No.**	6.41e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.41e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [12]:
```
test_model.summary2().tables[1]
```

Out[12]:

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.047215	0.059952	0.787542	4.311137e-01	-0.070403	0.164832
GSPC	-0.009397	0.000514	-18.287075	2.077461e-66	-0.010405	-0.008389
DJI	0.106091	0.003938	26.938497	1.726485e-126	0.098365	0.113818
TYX	-0.099326	1.065160	-0.093250	9.257196e-01	-2.189018	1.990365

In [13]:
```
# Parameters with high precision
model.summary2().tables[1]
```

Out[13]:

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.018185	0.053645	-0.338991	7.346731e-01	-0.123430	0.087059
AMD	0.028205	0.029253	0.964182	3.351414e-01	-0.029185	0.085595
META	-0.003159	0.011153	-0.283218	7.770568e-01	-0.025039	0.018722
TSLA	0.167543	0.009348	17.922736	4.136514e-64	0.149203	0.185882
AMZN	0.019158	0.008619	2.222685	2.641537e-02	0.002248	0.036068
GSPC	-0.004132	0.000649	-6.370326	2.645370e-10	-0.005405	-0.002860
DJI	0.050476	0.006219	8.116181	1.141432e-15	0.038275	0.062677
TYX	0.109620	0.954033	0.114902	9.085415e-01	-1.762063	1.981303

In the abover results we can see that we have 2 models one is "test_model" which is a regression with 3 independent varaible and all are broader market assets like S&P 500, Dow jones index, 30yr US tresurey Index and the other model which is "model" contains more independent varaibles which are peer of NVDA or from a similar sector.

Now we will see the comparision between the results of these 2 models to uderstand the effect of multicollinerity :

- In the above reults summary table we can see that in test_model DJI coefficients are (0.106091) which is significant in the model but in the main model i.e "model" it is just (0.050476) which is not significant at all.
- In the results of "model" we can see all the p-values are larger than 0.05 which means no independent varaibles estimates are significant.
- The R^2 in second model is (0.592) which is higher than first model (0.485) this is because we have more independent value and the adjusted R^2 of the first model also higher (0.589) compared to first model's adjusted R^2 (0.484)

One more statstical method that we use is **variance inflation factor(VIF)** which can be used to determine if there is any multicollinerity issue in independent varaible or not (Tay).

- **VIF** Formula :

> $$\text{VIF} = \frac{1}{1 - R^2}$$

- when VIF is 1 and R^2 is 0, this means there is no multicollinerity in this independent variable.
- The higher the multicollinerity the more the VIF value.
- VIF between 1 to 5 have no severe multicollinerity.
- VIF higher than 5 shows have there is severe multicollineity.

Now let's implet this into our example :

In [14]:
```
labels = [

        'NVDA',
        'AMD',
        'META',
        'TSLA',
        'AMZN',
        'GSPC',
        'DJI',
        'TYX',
]

variables = model.model.exog
vif = [variance_inflation_factor(variables, i) for i in range(variables.shape[1])]

pd.DataFrame(vif[1:], labels[1:], columns=["VIF"])
```

Out[14]:

	VIF
AMD	2.459770
META	1.882379
TSLA	1.842356
AMZN	1.556022
GSPC	18.698611
DJI	28.708614
TYX	1.067021

Here we can see GSPC and DJI have higher value than 5, which suggest that these varaibles have multicollinerity issue.

Directions

As we discussed the damage due to multicollinearity, In this section we will talk about the directions that we should take to counter these issues :

> **Direction 1** : We can drop independent variables which have high correlation, the thumb rule is to select correlation between 0.8 and 0.9. we can drop any one of the corrrelated value and see if the model shows any imporvement. but this appraoch has some issue as it only considers relationship between two variable at one time, so if we have multicollinerity with more than two varibale there will be issue using this method.

Note: In our case we have high multicollinerity between only 2 varibales so we can we this approach in our case.

> **Direction 2** : Next approach we can take is to define one of the independent variables in the model as dependent variables and run a regression with the

> other independent variables. we can try to find if any of the independet varaibles can be used to explain other independent variable.
>
> **Direction 3** : At last our third approach which is one of the most affective approach is pricipal component analysis.

Skewness

Definition:

Skewness refers to the asymmetry of a probability distribution (Sharma). It describes how the data points are distributed around the center of the distribution (Mudholkar and Hutson).

Formula for Skewness: Fisher - Pearson Standarized Moment Coefficient:

$$\sum_{i=1}^{n} \frac{(X_i - \bar{X})^3/n}{s_X^3}$$

where, $\bar{X}$ = Mean of X, S = Standard Deviation of X

Description:

A distribution can have three types of skewness:

- Right (Positive) Skew: In a right-skewed distribution, the tail on the right-hand side (positive values) is longer or fatter than the tail on the left-hand side. Most of the data points are concentrated on the left-hand side of the distribution, with a few large values pulling the mean to the right.
  Mean > Median > Mode
- Left (Negative) Skew: In a left-skewed distribution, the tail on the left-hand side (negative values) is longer or fatter than the tail on the right-hand side.Most of the data points are concentrated on the right-hand side of the distribution, with a few small values pulling the mean to the left.
  Mean < Median < Mode
- Zero Skew: When a distribution has zero skew, it is symmetrical. Its left and right sides are mirror images. Normal distributions have zero skew.
  Mean = Median = Mode

Demonstration:

In [15]:
```
test_model = smf.ols(
    "NVDA ~ GSPC + DJI + TYX",
    data=main_df,
).fit()
test_model.summary()
```

Out[15]:

OLS Regression Results

Dep. Variable:	NVDA	**R-squared:**	0.485
Model:	OLS	**Adj. R-squared:**	0.484
Method:	Least Squares	**F-statistic:**	394.3
Date:	Wed, 13 Mar 2024	**Prob (F-statistic):**	2.34e-180
Time:	13:29:35	**Log-Likelihood:**	-2730.5
No. Observations:	1258	**AIC:**	5469.
Df Residuals:	1254	**BIC:**	5490.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.0472	0.060	0.788	0.431	-0.070	0.165
GSPC	-0.0094	0.001	-18.287	0.000	-0.010	-0.008
DJI	0.1061	0.004	26.938	0.000	0.098	0.114
TYX	-0.0993	1.065	-0.093	0.926	-2.189	1.990

Omnibus:	280.489	**Durbin-Watson:**	2.021
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2024.422
Skew:	0.833	**Prob(JB):**	0.00
Kurtosis:	8.987	**Cond. No.**	6.39e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.39e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

From the above model summary, we can notice that the skewness is 0.833, which implies that it is moderately right (positively) skewed.

In [16]:
```python
fitted_values = test_model.fittedvalues
residuals = test_model.resid
```

In [17]:
```python
shapiro_test = stats.shapiro(residuals)
print(
    "Shapiro W: {0} \nShapiro p-value {1}".format(
        shapiro_test.statistic, shapiro_test.pvalue
    )
)
```

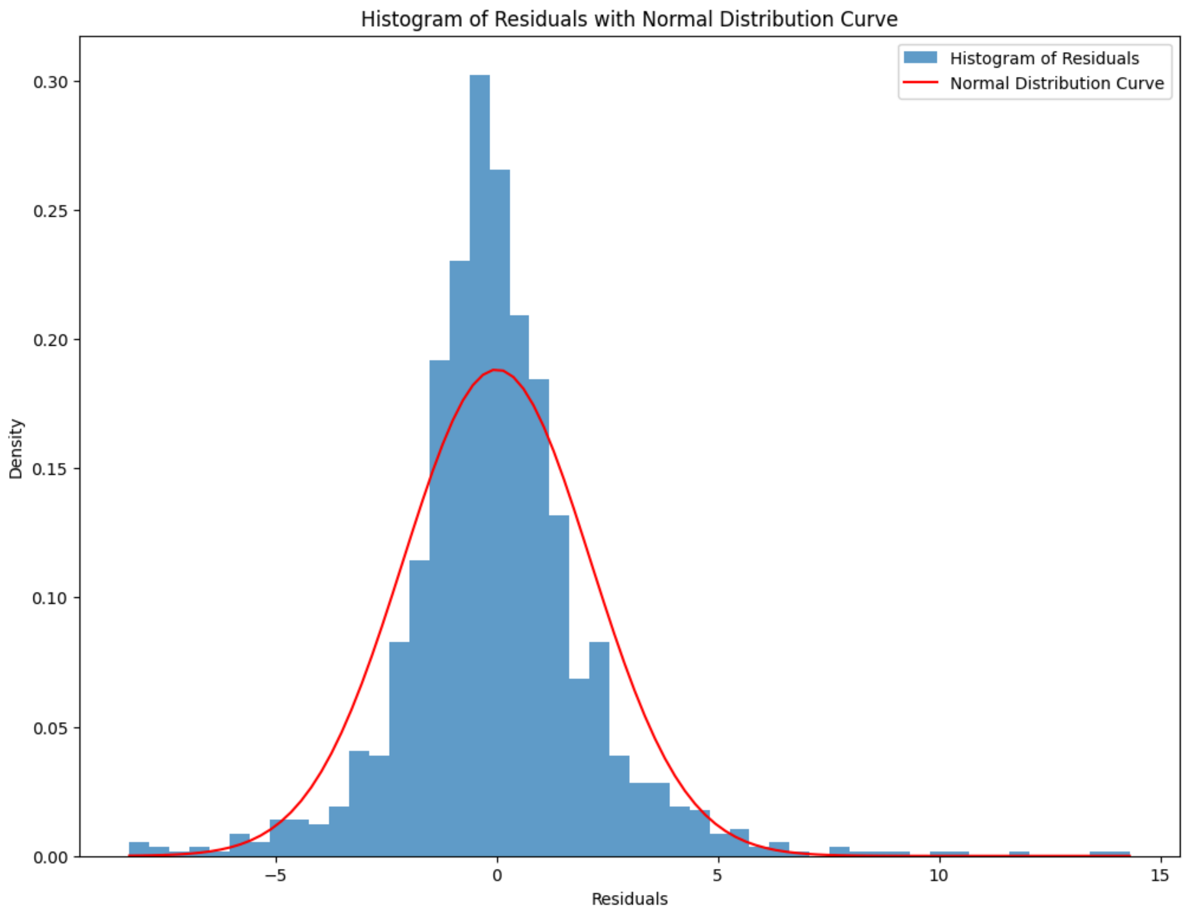
```
Shapiro W: 0.9232006669044495
Shapiro p-value 8.000548907699045e-25
```

As per the Normality Test using the Shapiro Test, the p-value is way less than 0.05, thus we can reject null hypotheses. So, they are not normally distributed

Diagram:

In [18]:
```
def HistogPlotNormal(residuals, label):
    mu = np.mean(residuals)
    sigma = np.std(residuals)

    x = np.linspace(min(residuals), max(residuals), 100)
    y = norm.pdf(x, loc=mu, scale=sigma)

    plt.hist(residuals, bins=50, density=True, alpha=0.7, label="Histogram of Resic
    plt.plot(x, y, 'r-', label="Normal Distribution Curve")

    plt.xlabel("Residuals")
    plt.ylabel("Density")
    plt.title("Histogram of Residuals with Normal Distribution Curve")
    plt.legend()

    plt.show()
```

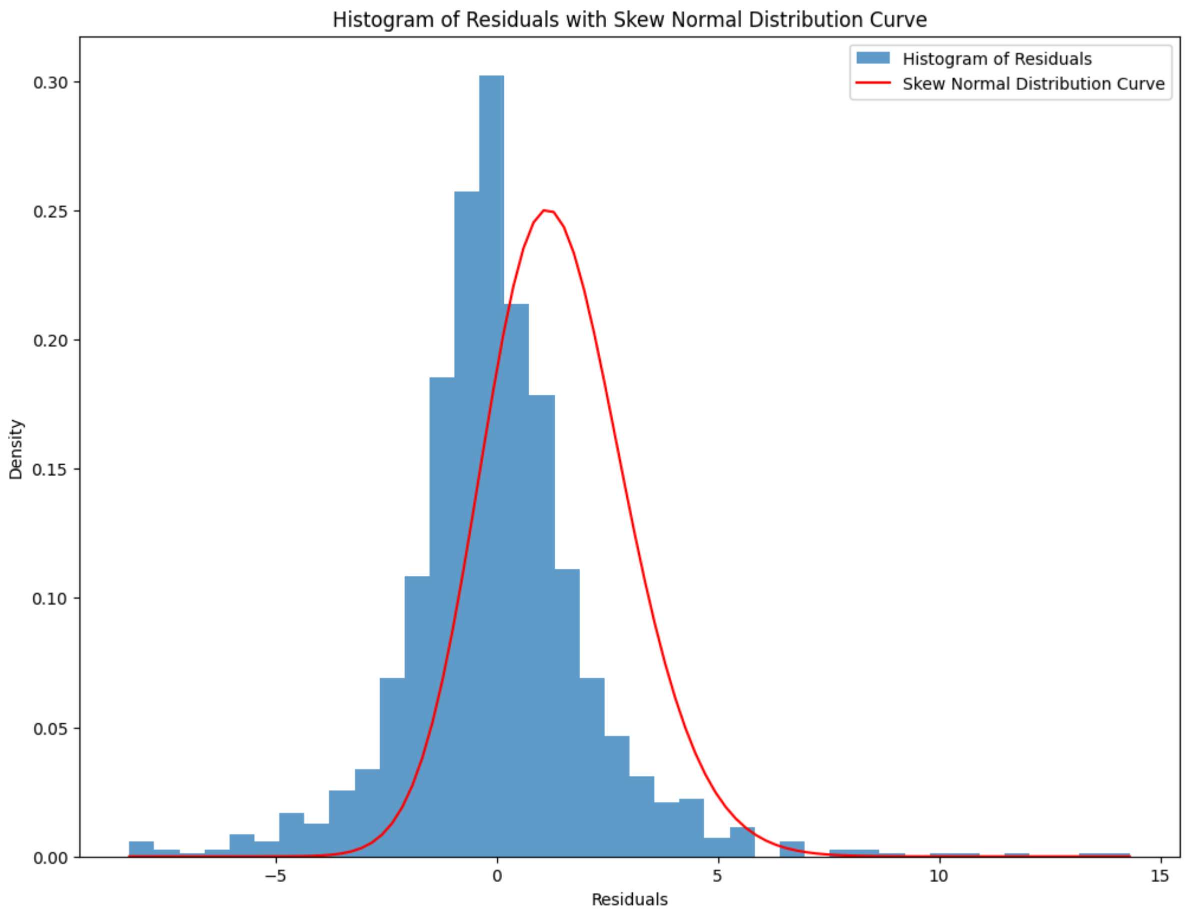
In [19]:
```
HistogPlotNormal(residuals, "Normal Disribution - Residuals from OLS Model")
```



From the above graph, we can see that the tail on the right side is longer, indicating that it is right skewed.

In [20]:
```
def HistogPlot(residuals, label):
    mu = np.mean(residuals)
    sigma = np.std(residuals)
    alpha = skewnorm.fit(residuals)[0]

    x = np.linspace(min(residuals), max(residuals), 100)
    y = skewnorm.pdf(x, alpha, loc=mu, scale=sigma)
    plt.hist(residuals, bins=40, density=True, alpha=0.7, label="Histogram of Resid
    plt.plot(x, y, 'r-', label="Skew Normal Distribution Curve")
    plt.xlabel("Residuals")
    plt.ylabel("Density")
    plt.title("Histogram of Residuals with Skew Normal Distribution Curve")
    plt.legend()

    plt.show()
```

In [21]:
```
HistogPlot(residuals, "Residuals from OLS Model")
```



Diagnosis:

There are various ways to check if Skewness exists in the data:

1. Visual Inspection - Plotting histograms, plots to check if it is skewed by analyzing the plot and checking for asymmetry
2. Using the statistical packages -
   For example: As used above,
   test_model = smf.ols("NVDA ~ GSPC + DJI + TYX", data=main_df,).fit()
   test_model.summary()
   The summary gives the skewness coefficient which can be used to analyze.
   If skewness > 0, right skewed
   skewness < 0, left skewed
   skewness = 0, no skewness

Other ways includes checking the Normality:

1. Q -Q Plot Analysis
2. Shapiro Test for Normality on the residuals: If p- value is greater than 0.05, we can reject the Null hypothesis of normality, if not, it is normally distributed.

This indicates if the distribution is normally distributed, if not, we can conclude that it is asymmetrical.

Damage:

1. Impact on statistical models and tests: Generally, the models and tests assume the data distribution to be normal, skewness violates this assumptions, thus, impacting the results. To address skewness, we need to use skew-normal, skew-t distribution.
2. Outliers is one of the reason behind the skewness. Thus, the data distribution is distorted due to presence of skewness, making the analysis challenging (Wikipedia Contributors).

Directions:

To address the mentioned damage caused by Skewness, we can do the following:

1. Skew-Normal or Skew - t Distributions Model : We can use these distributions to handle the skewness effectively, without any transformation.
2. Log transformation: Transform skewed distribution to a normal distribution. It compresses the tail and makes the distribution symmetrical.
3. Remove outliers: Outliers that are irrelavant to the analysis can be removed.
4. Normalize: By using Mix-Max Scaling, scaling data to 0 to 1 range.
5. Box Cox transformation (Lai)

Sensitivity to outliers

Definition

Some data points are extreme that do not confirm to the distribution pattern of majority of the data, are called outliers. In measurement of centrality, mode and median are not sensitive to outliers but mean is highly sensitive and statistical measures based on mean score are thus highly sensitive too (“Sensitivity to Outliers”). Ignoring outliers can lead to significant incorrect estimation (Chambers et al.). Outliers that can influence the model performance are influential points.

Description

An outlier shows the deviation from the general trend of the other points. In that case the regression model will not make correct estimation. In OLS (Orinary Least Square) outliers will pull the regression line closer to outlier. We will examine the outlier in scatter plot

Figure 2: Histogram, Correlation, and Scatter Plot Graph Matrix For Independent Variables and Dependent Variable

In [22]:
```
# Function to calculate correlation coefficient between two arrays


def corr(x, y, **kwargs):
    # Calculate the value
    coef = np.corrcoef(x, y)[0][1]
    # Make the label
    label = r"$\rho$ = " + str(round(coef, 2))

    # Add the label to the plot
    ax = plt.gca()
    ax.annotate(label, xy=(0.3, 0.15), size=20, xycoords=ax.transAxes)
```

In [23]:
```
# Create the default pairplot
grid = sns.pairplot(
    main_df, vars=['NVDA', 'AMD', 'META', 'TSLA', 'AMZN', 'GSPC', 'DJI', 'TYX'], he
)

# Map a scatter plot and Pearson correlation coefficient to the upper triangle
grid = grid.map_upper(plt.scatter)
grid = grid.map_upper(corr)

# Map a histogram to the diagonal
grid = grid.map_diag(plt.hist)

# Map a density plot to the lower triangle
grid = grid.map_lower(sns.kdeplot)
```

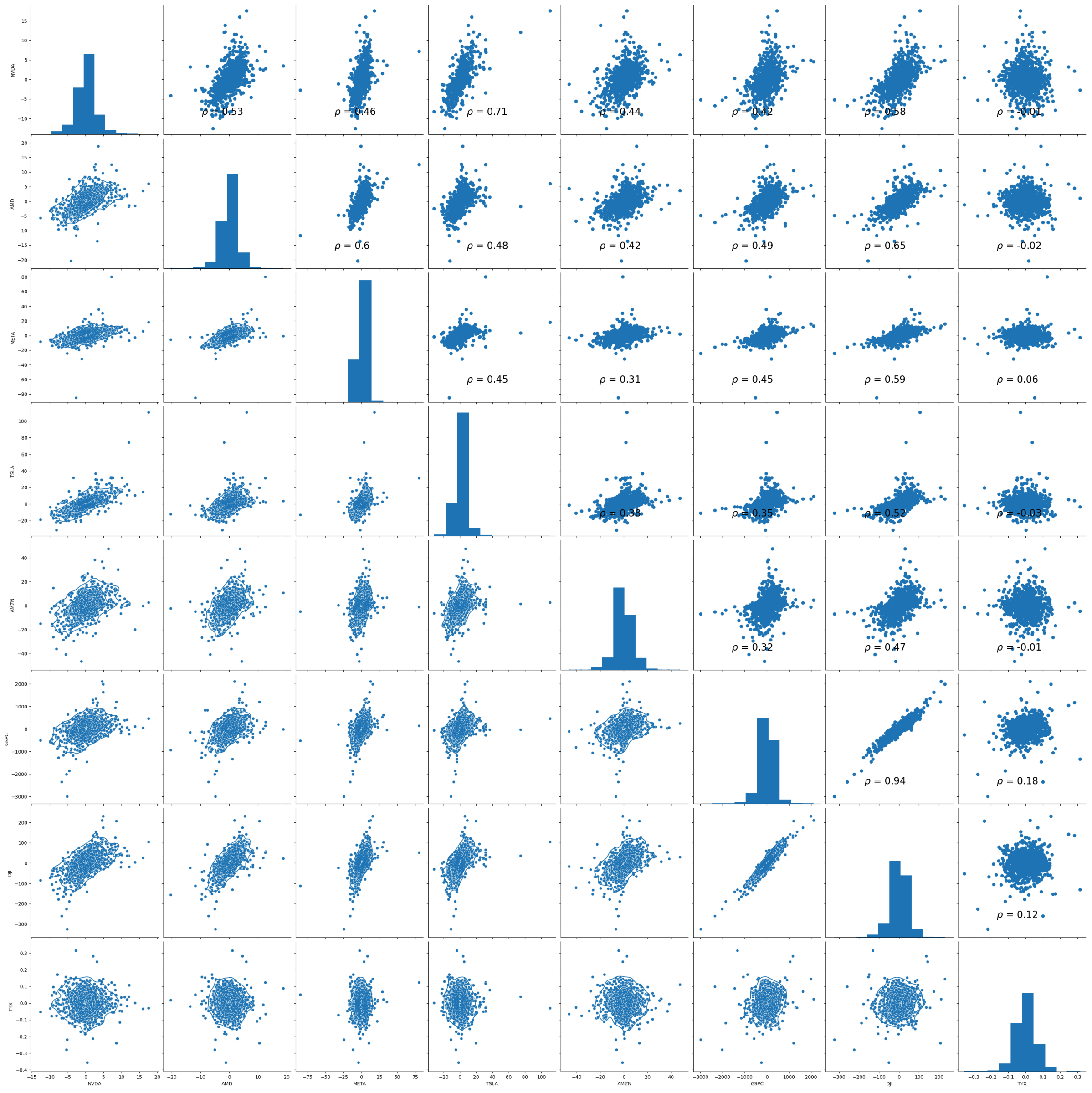


Figure 2 provides a summary of the relationships of variables in NVDA stock analysis. The matrix is split into three parts.

The lower left triangle of the matrix shows the scatter plots for all two-way combinations of the eight variables. The graphs of the diagonal of the matrix are the histograms of the eight variables. The upper right triangle of the matrix provides the correlation values of all two-way combinations of the eight variables.

The scatter plot NVDA and AMD, NVDA and TSLA, NVDA and DJI is high correlated (>.5). The data points also make oval shape more or less. These signs suggest these variables are good choices for modeling NVDA. NVDA is negatively correllated with TYX.

Demonstration

Now we will try to identify the influential points. We need to identify influential points in the dataset and study why they are extreme (a special event or erroneous data input) because they can potentially alter the regression result. Then, we will run one model with the influential point and one without the influential point. Afterwards, we compare the model results and evaluate which results fit our modeling goal the best.

We use Cook's distance (Cook's D) to identify influential points in the dataset. Cook's D is a metric to calculate the prediction difference from the model with a data point and without a data point.

Figure 3: Influence Plot on Test Model

In [24]:
```
# Influence plot on Test Model
fig = sm.graphics.influence_plot(test_model, criterion="cooks", alpha=0.5)
fig.tight_layout(pad=1.0)
```

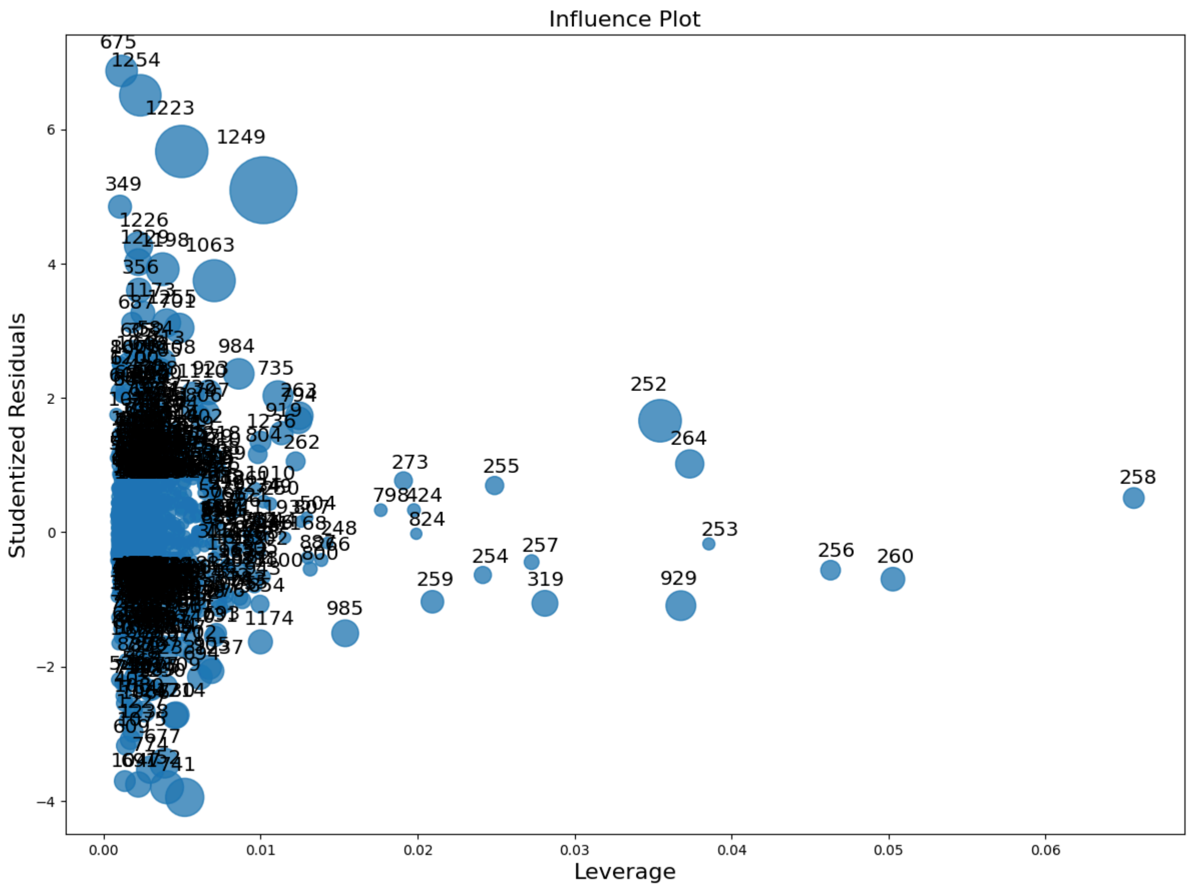
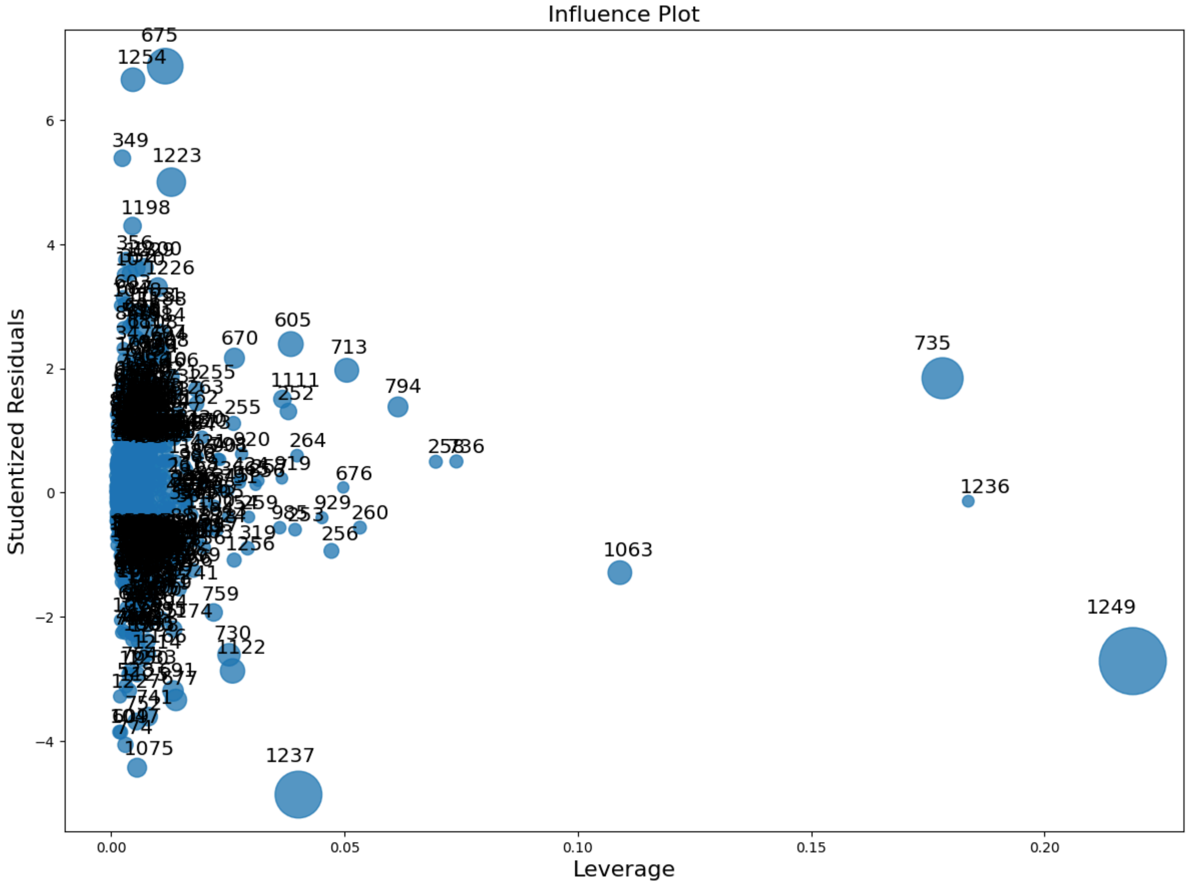


Figure 3: Influence Plot on Complete Model

In [25]:
```
# Influence plot
fig = sm.graphics.influence_plot(model, criterion="cooks", alpha=0.5)
fig.tight_layout(pad=1.0)
```



In [26]:
```
# Finding most influential points in Test Model
influence = test_model.get_influence()
inf_sum = influence.summary_frame().sort_values("cooks_d", ascending=False)
inf_sum.head(30)
```

Out[26]:

	dfb_Intercept	dfb_GSPC	dfb_DJI	dfb_TYX	cooks_d	standard_resid	hat_diag	dffits_inte
1249	0.122188	-0.356034	0.447566	-0.042983	0.065312	5.040159	0.010179	0.51
1223	0.153389	-0.198187	0.151932	0.335555	0.039164	5.597817	0.004974	0.39
252	0.056370	0.110292	-0.110418	-0.305689	0.025260	1.658075	0.035449	0.31
1063	0.093814	-0.280337	0.287955	0.114421	0.024604	3.724472	0.007045	0.31
1254	0.175114	-0.198589	0.224969	-0.084065	0.023956	6.398281	0.002335	0.30
741	-0.124584	-0.133342	0.192441	0.102212	0.019971	-3.922988	0.005164	-0.28
752	-0.113082	0.066439	-0.014392	0.111200	0.014434	-3.775055	0.004035	-0.24
1198	0.101131	-0.194473	0.209452	0.069233	0.014286	3.892430	0.003758	0.23
675	0.196772	0.128152	-0.117557	-0.027134	0.013028	6.742327	0.001145	0.22
984	0.061810	-0.142966	0.160115	-0.108940	0.012024	2.353252	0.008610	0.21
735	0.066310	0.134805	-0.176908	0.038977	0.011559	2.028813	0.011108	0.21
677	-0.098394	-0.001162	0.036233	-0.167475	0.011408	-3.422693	0.003880	-0.21
929	-0.025961	0.052842	-0.102288	0.134709	0.011391	-1.092466	0.036774	-0.21
701	0.076110	-0.130170	0.166818	0.062570	0.011109	3.030527	0.004815	0.21
264	0.028274	0.147623	-0.095409	-0.034090	0.010029	1.016979	0.037339	0.20
1226	0.114304	-0.019916	0.070383	-0.052910	0.009979	4.232756	0.002223	0.19
1255	0.082029	-0.135949	0.156337	-0.056391	0.009724	3.106225	0.004015	0.19
263	0.054084	-0.000517	-0.016896	-0.169515	0.009466	1.732155	0.012462	0.19
774	-0.106152	0.011052	0.040935	0.034408	0.009198	-3.518092	0.002964	-0.19
985	-0.035432	0.174752	-0.182850	-0.034089	0.008825	-1.502849	0.015389	-0.18
1229	0.108390	-0.135327	0.126609	0.092779	0.008807	3.995788	0.002202	0.18
794	0.055377	0.073416	-0.124006	0.018402	0.008715	1.664953	0.012419	0.18
730	-0.083385	-0.130237	0.132739	0.122270	0.008511	-2.713238	0.004603	-0.18
1214	-0.081610	-0.129820	0.142189	-0.064956	0.008417	-2.721548	0.004525	-0.18
319	-0.032351	0.116651	-0.070736	0.023016	0.008095	-1.057997	0.028112	-0.17
691	-0.109248	0.058652	-0.012998	-0.009525	0.007710	-3.734362	0.002207	-0.17
1237	-0.054825	0.103589	-0.085300	-0.140464	0.007360	-2.063382	0.006867	-0.17
923	0.066561	0.098873	-0.135269	-0.007845	0.007282	2.079466	0.006691	0.17
356	0.104363	0.090861	-0.074257	-0.116368	0.007196	3.581175	0.002239	0.16
694	-0.060855	-0.121535	0.095165	-0.048156	0.007146	-2.153022	0.006129	-0.16

In [27]:

```python
# Finding most influential points in the complete model
influence = model.get_influence()
inf_sum = influence.summary_frame().sort_values("cooks_d", ascending=False)
inf_sum.head(30)
```

Out[27]:

	dfb_Intercept	dfb_AMD	dfb_META	dfb_TSLA	dfb_AMZN	dfb_GSPC	dfb_DJI	dfb_TYX
1249	0.027159	-0.009237	-0.014015	-1.388982	0.141085	-0.364942	0.413849	0.012499
1237	-0.081911	0.021518	0.532956	-0.701923	0.253686	0.140682	-0.107005	-0.377065
735	0.068271	0.038824	-0.813029	0.074668	0.007087	-0.096913	0.149374	0.094435
675	0.161805	-0.077861	-0.123298	0.385085	-0.574107	-0.001271	0.048345	-0.016275
1223	0.109652	-0.046049	-0.225278	0.379334	-0.017291	-0.064124	0.024287	0.312891
605	0.064351	-0.466741	0.107579	-0.017291	0.056197	-0.135060	0.165861	-0.049253
1122	-0.046347	0.061479	-0.001958	-0.379609	-0.104729	-0.017293	0.057511	-0.140977
713	0.057010	0.101939	0.019404	0.062445	0.424169	0.197563	-0.223296	0.105670
1063	-0.002762	0.091759	0.035735	-0.414930	0.048152	-0.029661	0.045174	-0.044205
1254	0.160566	0.119574	0.009181	0.253770	-0.153207	-0.038362	0.008689	-0.078505
730	-0.079461	-0.102930	-0.015546	0.031657	0.354345	-0.014406	-0.005076	0.113957
677	-0.106945	0.119306	0.023557	0.126793	-0.273397	0.021232	-0.010156	-0.157376
691	-0.102083	0.051077	0.199541	0.158041	0.228806	0.248558	-0.253945	-0.018609
670	0.061507	-0.098649	0.130640	0.024648	0.324139	0.117564	-0.127606	0.009833
794	0.040478	-0.280576	0.153630	0.028038	0.100267	0.016104	-0.021962	-0.006035
1226	0.067407	-0.036932	0.027109	0.277498	-0.066838	0.071427	-0.058225	-0.041460
1075	-0.103010	0.083919	0.071216	-0.225317	-0.090316	0.000893	-0.000043	-0.082115
741	-0.121970	0.079831	-0.028382	0.155214	0.088573	0.027072	-0.028090	0.099484
1200	0.112846	-0.087600	-0.163463	-0.182598	-0.106607	-0.172261	0.214964	0.003629
1111	0.047146	0.279328	-0.055916	0.045542	-0.013469	0.128945	-0.150857	-0.053275
1198	0.105758	-0.071808	0.065026	0.030284	-0.075436	-0.172477	0.154658	0.069111
759	-0.058951	-0.262929	0.078975	0.000647	0.040836	-0.066970	0.091743	-0.105486
752	-0.117982	0.030887	0.057547	0.116473	-0.006140	0.119671	-0.093528	0.106317
1229	0.082539	-0.024497	0.047609	0.185201	-0.064456	-0.028541	0.000640	0.081158
984	0.064863	-0.091437	-0.043268	0.013430	-0.032917	-0.165634	0.166691	-0.119267
349	0.152036	0.139012	-0.069501	0.032590	-0.136284	-0.020148	-0.003649	-0.000560
252	0.042570	0.029167	0.028894	0.036288	0.039501	0.108597	-0.104098	-0.240824
1174	-0.058438	0.064261	0.069810	-0.039918	-0.052208	-0.005904	-0.031346	0.181021
1188	0.093184	0.083801	0.041220	-0.179823	-0.087169	-0.018208	0.031861	-0.060036
1131	0.086891	0.008359	-0.012501	-0.116882	-0.165329	-0.084452	0.097681	0.107462

Figure 3 and 4 are influence plot. It helps us visualize the Cook's D among all data points. On the horizontal axis of the plot is hat value. On the vertical axis of the plot is studentized residuals. The bubbles in the plot indicate the size of the Cook's D for each point. The bigger the bubble, the higher the Cook's D for a data point.

From the influence plot, we can see that Point 1249 has the highest Cook's D values. Among the points, Point 1249 has a Cook's D of .256777 which is less than 1, so there is no influential point in either the test model or complete model.

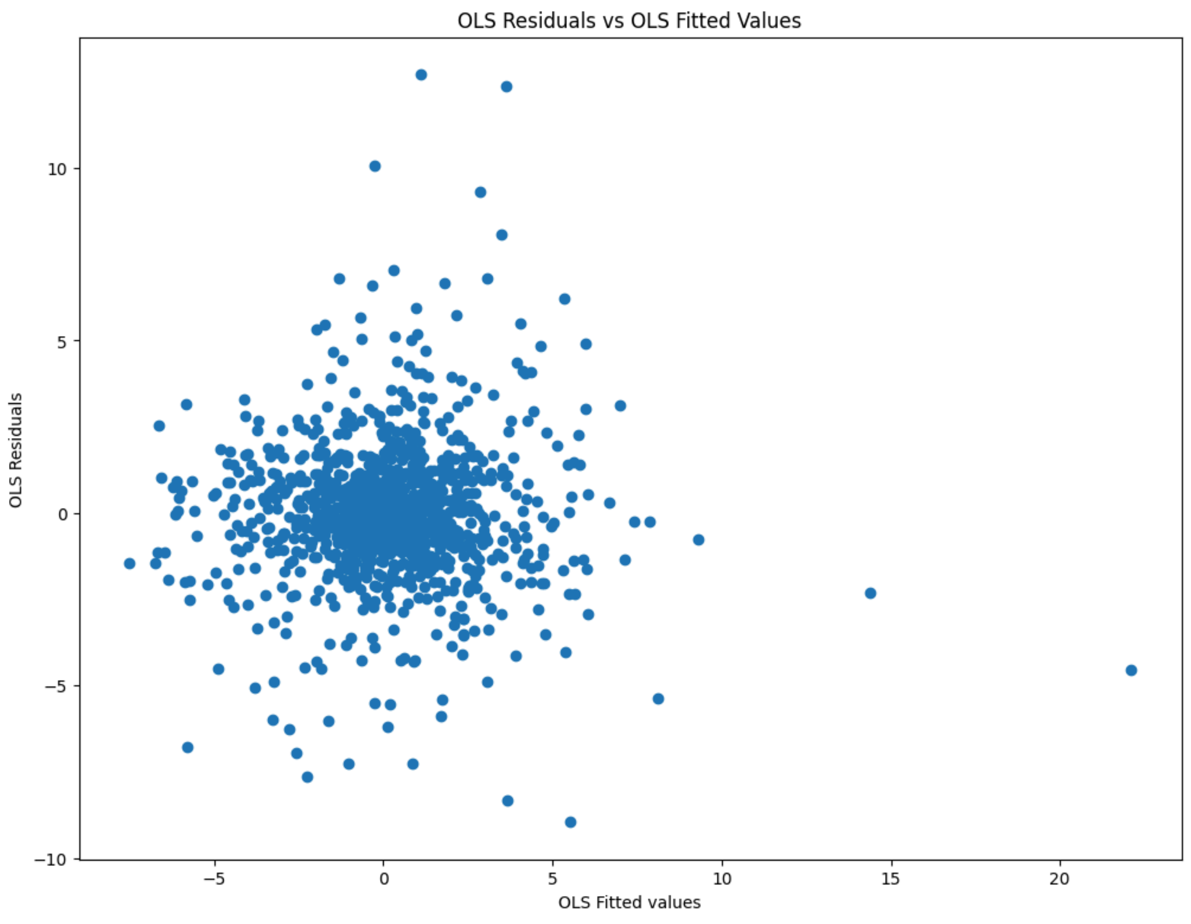
Diagram

As you can see we have plotted scatter plot diagram and influence plot diagram of our variables now we will to find heteroskedasticity isse.

For deep analysis we will plot scatter plot for OLS Fitted Value and OLS Residuals to check presence of heteroskedasticity issue

Figure 5: Scatter Plot for OLS Fitted Values and OLS Residuals

In [28]:
```
# Scatter Plot for OLS Fitted Values and OLS Residuals

# Fitted values and residuals
model_fitted_y = model.fittedvalues
model_resid_y = model.resid

# Plot
plt.scatter(x=model_fitted_y, y=model_resid_y)
plt.title("OLS Residuals vs OLS Fitted Values")
plt.xlabel("OLS Fitted values")
plt.ylabel("OLS Residuals")
plt.show()
```



In [29]:
```
# Breusch-Pagan Test
name = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]
```

```
test = sms.het_breuschpagan(model.resid, model.model.exog)
pd.DataFrame(test, index=name, columns=[""])
```

Out[29]:

Lagrange multiplier statistic	4.231576e+01
p-value	4.520372e-07
f-value	6.215746e+00
f p-value	3.557169e-07

Diagnosis

Figure 5 shows that there is more variation in residuals when fitted values are positive than when fitted values are negative. The scatter plot indicates there might be a heteroskedasticity issue. We run a Breusch-Pagan test to double-check the hypothesis.

The test result from Figure 6 also confirms the existence of heteroskedasticity because the p-value is less than 0.05. Based on the above information, we will run the model with **weighted least square**, with the weight generated.

Damage

We will analyse the damage by the presence of extreme events. The first model is built by "Team member 1 Multicollinerity : Step 2".

In [30]:
```
# NVDA daily return Regression Model with Seven Independent Variables

model = smf.ols(
    "NVDA ~ AMD + META + TSLA + AMZN + GSPC + DJI + TYX",
    data=main_df,
).fit()

model.summary()
```

Out[30]:

OLS Regression Results

Dep. Variable:	NVDA	**R-squared:**	0.592
Model:	OLS	**Adj. R-squared:**	0.589
Method:	Least Squares	**F-statistic:**	258.6
Date:	Wed, 13 Mar 2024	**Prob (F-statistic):**	7.64e-238
Time:	13:31:04	**Log-Likelihood:**	-2585.3
No. Observations:	1258	**AIC:**	5187.
Df Residuals:	1250	**BIC:**	5228.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-0.0182	0.054	-0.339	0.735	-0.123	0.087
AMD	0.0282	0.029	0.964	0.335	-0.029	0.086
META	-0.0032	0.011	-0.283	0.777	-0.025	0.019
TSLA	0.1675	0.009	17.923	0.000	0.149	0.186
AMZN	0.0192	0.009	2.223	0.026	0.002	0.036
GSPC	-0.0041	0.001	-6.370	0.000	-0.005	-0.003
DJI	0.0505	0.006	8.116	0.000	0.038	0.063
TYX	0.1096	0.954	0.115	0.909	-1.762	1.981

Omnibus:	231.106	**Durbin-Watson:**	1.977
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2226.997
Skew:	0.548	**Prob(JB):**	0.00
Kurtosis:	9.425	**Cond. No.**	6.41e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.41e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [31]:

```
# WLS regression result

# Add Absolute residuals and fitted values to main_df columns
main_df["abs_residuals"] = np.abs(model.resid)
main_df["fitted_values"] = model.fittedvalues

# Fit OLS model with absolute residuals and fitted values
model_temp = smf.ols("abs_residuals ~ fitted_values", data=main_df).fit()

# Compute weights and add it to the main_df column
weights = model_temp.fittedvalues
weights = weights**-2
main_df["weights"] = weights
```

```
# Fit WLS model
Y = main_df["NVDA"]
X = main_df[['AMD', 'META', 'TSLA', 'AMZN', 'GSPC', 'DJI', 'TYX']]
X = sm.add_constant(X)  # add a intercept point

model_WLS = sm.WLS(Y, X, main_df["weights"]).fit()
model_WLS.summary()
```

Out[31]:

WLS Regression Results

Dep. Variable:	NVDA	**R-squared:**	0.624
Model:	WLS	**Adj. R-squared:**	0.622
Method:	Least Squares	**F-statistic:**	296.1
Date:	Wed, 13 Mar 2024	**Prob (F-statistic):**	3.89e-260
Time:	13:31:04	**Log-Likelihood:**	-2574.9
No. Observations:	1258	**AIC:**	5166.
Df Residuals:	1250	**BIC:**	5207.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0245	0.053	-0.462	0.644	-0.129	0.080
AMD	0.0199	0.028	0.701	0.483	-0.036	0.075
META	-0.0130	0.010	-1.254	0.210	-0.033	0.007
TSLA	0.1913	0.011	18.092	0.000	0.171	0.212
AMZN	0.0197	0.008	2.372	0.018	0.003	0.036
GSPC	-0.0040	0.001	-6.408	0.000	-0.005	-0.003
DJI	0.0493	0.006	8.118	0.000	0.037	0.061
TYX	0.0220	0.911	0.024	0.981	-1.766	1.810

Omnibus:	164.363	**Durbin-Watson:**	1.967
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1616.802
Skew:	0.182	**Prob(JB):**	0.00
Kurtosis:	8.542	**Cond. No.**	6.85e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.85e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In the abover results we can see that we have 2 models one is OLS which is a Ordinary Least Square regression with 7 independent varaible and all are broader market assets which are peer of NVDA or from a similar sector and a WLS which is Weighted Least Square model.

Now we will see the comparision between the results of these 2 models to uderstand the effect of heteroskedasticity which explains outlier events :

- In the results of OLS and WLS we can see that the p-values of 'AMD', 'META' and 'TYX' are larger than 0.05 which means these independent varaibles estimates are not significant.
- The R^2 in WLS model is (0.624) which is higher than OLS model (0.592) this is because we have more independent value and the adjusted R^2 of the OLS model also higher (0.622) compared to OLS model's adjusted R^2 (0.589)

Direction

Handling outliers in regression analysis is crucial to ensure robust and accurate model performance.

In our model we did not identified specific problem of outliers as Cooks's Distance is less than 1 in each case, so outliers are not present. However outliers are always required to be handled effectively and directions are:

Direction 1: We can exclude outliers from the dataset if outliers are not important datapoints in the study.

Direction 2: Rather than excluding outliers, we consider using robust regression methods which is less sensitive to outliers.

Ways to address heteroscedasticity:

Direction 3: Transform the Dependent Variable: One approach is to transform the dependent variable. Common transformations include taking the logarithm of the dependent variable. This can help stabilize the variance and make it more consistent across different levels of the predictor variables.

Direction 4: Weighted Least Squares Regression (WLS): WLS assigns different weights to each observation based on their variance. By giving more weight to observations with lower variance, WLS accounts for heteroscedasticity.

Overfitting

Definition

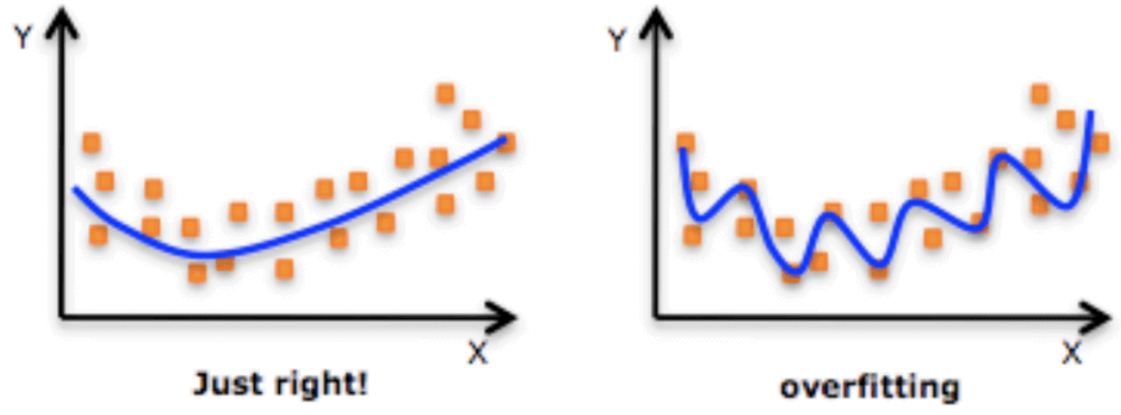
The tendency to fit trainingging data too closely to training set makes it complex where model captures noises and idiosyncrasies. Such a model works well with training data but fails on testing data because of overfitting. Such a model memorises training data and loses focus from underlying patterns and loses its ability to genralize.

In finance this problem arises where model (e.g. Stock market prediction model) works well with historic data but is unable to provide accuracy on data in future market conditions (Melanie).

Description

While working with financial data we usually have a lot of variables, which makes our data high dimesional an when on that high dimensional data we run regression and the model predict the training data accurately but fails on testing or new data.

This ability to predict model correctly i.e not only the relationship between endogenous and exogenous variable but also idiosyncratic random error is termed as **overfitting**.



Source: Quora

Here in the image we can see that there are two models with same data points, the left model is simple model with one variables but on the right we can we more than one variables which makes the model complecated and the models also captures noise and as we can it is predection the data accurately hence there is overfitting in right model.

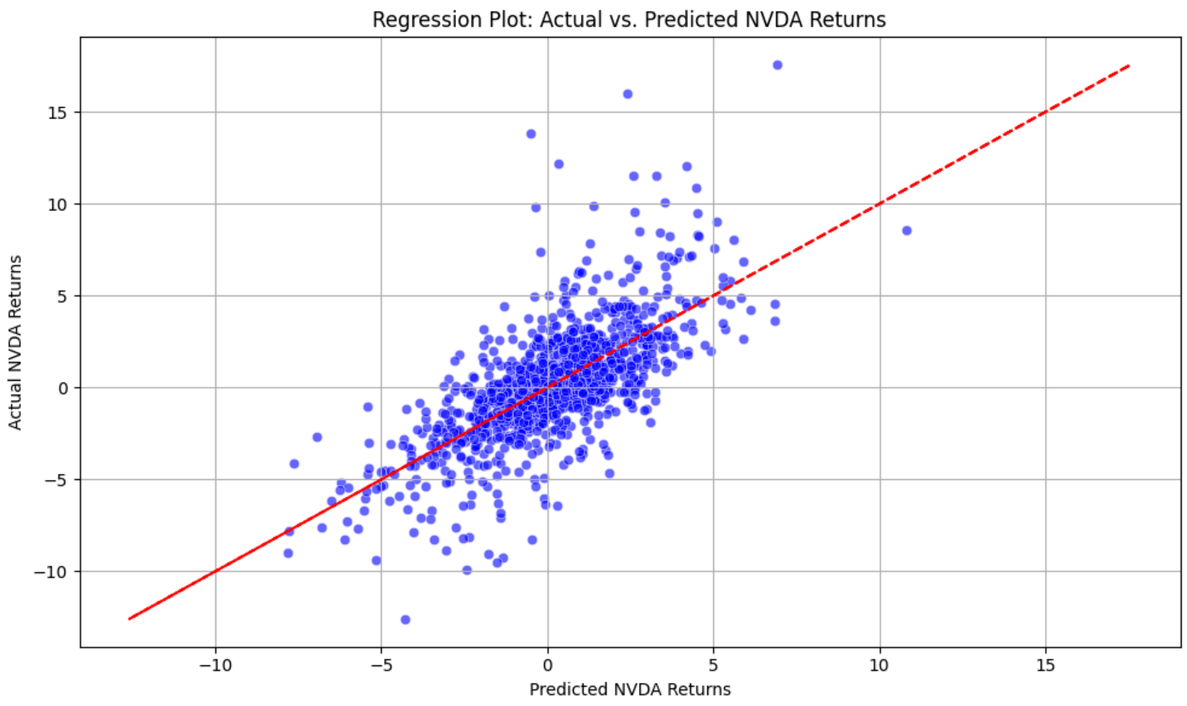
Demonstration

Now let's see the real life example of our data that we are using which is NVDIA, TESLA, META, AMAZON, AMD, S&P500, Dow jones, 30yr US tresury daily returns.

First we will use model build by me in (Team member 1 Multicollinerity : Step 2), we will plot the regression plot for that model :

```
In [32]: # Get the predicted values
predicted_values = test_model.predict()

# Create a scatter plot of actual vs. predicted values
plt.figure(figsize=(10, 6))
sns.scatterplot(x=predicted_values, y=main_df['NVDA'], color='blue', alpha=0.6)

# Add labels and title
plt.xlabel('Predicted NVDA Returns')
plt.ylabel('Actual NVDA Returns')
plt.title('Regression Plot: Actual vs. Predicted NVDA Returns')
```

```
# Add a diagonal line for reference
plt.plot(main_df['NVDA'], main_df['NVDA'], color='red', linestyle='--')

# Show plot
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [33]:
```
# Get the predicted values
predicted_values = model.predict()

# Create a scatter plot of actual vs. predicted values
plt.figure(figsize=(10, 6))
sns.scatterplot(x=predicted_values, y=main_df['NVDA'], color='blue', alpha=0.6)

# Add labels and title
plt.xlabel('Predicted NVDA Returns')
plt.ylabel('Actual NVDA Returns')
plt.title('Regression Plot: Actual vs. Predicted NVDA Returns')

# Add a diagonal line for reference
plt.plot(main_df['NVDA'], main_df['NVDA'], color='red', linestyle='--')

# Show plot
plt.grid(True)
plt.tight_layout()
plt.show()
```

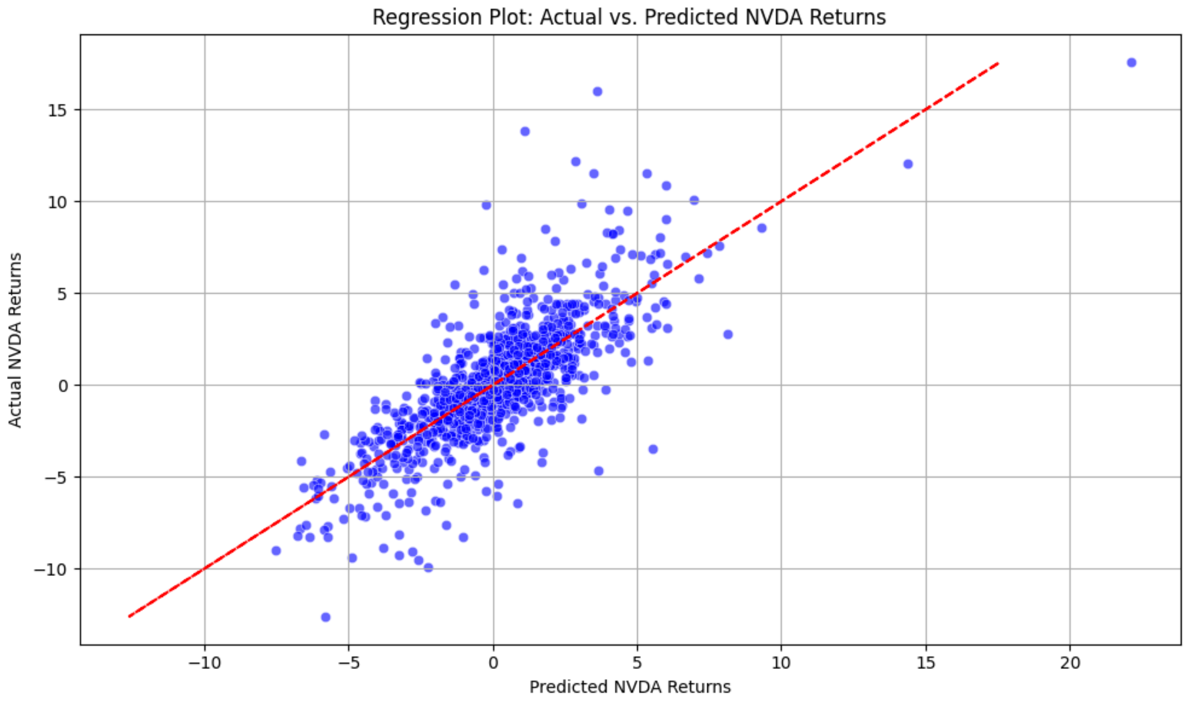


Diagram:

Here in the above first figure we can see a simple regression from our test_model model now in the second figure we can use model written by (Team member 3 : Sensitivity to outliers : step 2), Now we will plot the regression for this model Here we can see in these above diagram for the second figure that this regression is fitting the data more accuretly, to verify this we need to do diagnosis which we will do in diagnosis section and and if there is any overfitting in our data we will discuss the damage done due to overfitting as well as direction how to further improve it.

Diagnosis

To daignose the issue of overfitting we will use cross-validation technique. we will perform cross validation on both the models and asses thir performance on test data. we will compare the mean cross-validation score as well as standard deviation of cross-validation scores and see if there is any overfitting issue :

In [34]:
```
X = main_df[['GSPC', 'DJI', 'TYX']]  # Independent variables for model 1
y = main_df['NVDA']  # Dependent variable for both models

# Initialize the models
model_1 = LinearRegression()
model_2 = LinearRegression()

# Perform cross-validation for model 1
cv_scores_model_1 = cross_val_score(model_1, X, y, cv=5)  # Change cv value as need

# Define independent variables for model 2
X_model_2 = main_df[['AMD', 'META', 'TSLA', 'AMZN', 'GSPC', 'DJI', 'TYX']]

# Perform cross-validation for model 2
cv_scores_model_2 = cross_val_score(model_2, X_model_2, y, cv=5)  # Change cv value
```

In [35]:
```
# Print mean and standard deviation of cross-validation scores for models
print("Model 1 - Mean Cross-Validation Score:", np.mean(cv_scores_model_1))
print("Model 1 - Standard Deviation of Cross-Validation Scores:", np.std(cv_scores_


print("Model 2 - Mean Cross-Validation Score:", np.mean(cv_scores_model_2))
print("Model 2 - Standard Deviation of Cross-Validation Scores:", np.std(cv_scores_
```

```
Model 1 - Mean Cross-Validation Score: 0.4321363211602347
Model 1 - Standard Deviation of Cross-Validation Scores: 0.12443832808885182
Model 2 - Mean Cross-Validation Score: 0.5195691479487055
Model 2 - Standard Deviation of Cross-Validation Scores: 0.15183405371229233
```

Here we can see Mean and std dev. of cross-validation scores are higher in model_2 compared to model_1, which concludes that there is overfitting issue in the model_2. Now in the next section we will understand what damages it can do to our model.

Damage

The major damage that overfitting do to a model is that, it gives the model a perfect fit on the training data and when the model is used on the testing dataset the model predection of the endogenous varibale from testing data perform very poor.

The overfitted model try to model all the possible data in the training data set from the good, the bad and the ugly which not only includes signal but also idiosyncratic error. And the result of all this is that the model becomes garbage.

Directions

Now that we have understood what is overfitting, How is this an issue, Why does it happen, what damages it can do, now we will discuss the further directions in this section on how to mitigate this issue of overfitting:

> Direction 1: Regularization Techniques
>
> - Penalized Regression : We can use Penalized Regression techniques like Ridge Regression, LASSO (Least Absolute Shrinkage and Selection Operator) Regression.
>
> Direction 2: Simplify the model
>
> - As we see model_1 has three input variables which are independent to each other incontrast to model_2 which has seven input variables (leads to multicolliniearity) the standard deviation of cross-validation score is less in model_1. The efficiency of simpler model is higher.

These regression techniques introduce penalty functions in the regression model which removes overfitting by shrinking the regression coefficients towards zero, it the value is too large. the penalty functions try to reduce the impact of an independent varibale in the model if the coefficient of the exogenous variables is too large.

References

Chambers, Ray, et al. "Outlier robust small area estimation." Journal of the Royal Statistical Society Series B: Statistical Methodology 76.1 (2014): 47-69.

Et. al., Alhassan Umar Ahmad,. “A Study of Multicollinearity Detection and Rectification under Missing Values.” Turkish Journal of Computer and Mathematics Education (TURCOMAT), vol. 12, no. 1S, 11 Apr. 2021, pp. 399–418, https://doi.org/10.17762/turcomat.v12i1s.1880. Accessed 7 Oct. 2021.

Lai, Cheryl. “Study Notes: Handling Skewed Data for Machine Learning Models.” Medium, 4 Nov. 2020, reinec.medium.com/my-notes-handling-skewed-data-5984de303725.

Melanie. “Overfitting: What Is It? How Can I Avoid It?” Data Science Courses | DataScientest, 23 Sept. 2023, datascientest.com/en/overfitting-what-is-it-how-can-i-avoid-it. Accessed 13 Mar. 2024.

"Team member 1 Multicollinerity : Step 2", OLS Model: Group_5338_Group Work Project 1 M3

Mudholkar, Govind S., and Alan D. Hutson. “The Epsilon–Skew–Normal Distribution for Analyzing Near-Normal Data.” Journal of Statistical Planning and Inference, vol. 83, no. 2, Feb. 2000, pp. 291–309, https://doi.org/10.1016/s0378-3758(99)00096-8. Accessed 28 Apr. 2021.

“Sensitivity to Outliers.” Cloud.sowiso.nl, cloud.sowiso.nl/courses/theory/116/1746/26513/en. Accessed 10 Mar. 2024.

Sharma, Abhishek. “What Is Skewness in Statistics? | Statistics for Data Science.” Analytics Vidhya, 5 July 2020, www.analyticsvidhya.com/blog/2020/07/what-is-skewness-statistics/.

Tay, Richard. "Correlation, variance inflation and multicollinearity in regression model." Journal of the Eastern Asia Society for Transportation Studies 12 (2017): 2006-2015.

Wikipedia Contributors. “Skewness.” Wikipedia, Wikimedia Foundation, 22 Nov. 2019, en.wikipedia.org/wiki/Skewness.