In [1]:
```
# Multicollinearity Analysis
#Libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import yfinance as yf
import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.model_selection import TimeSeriesSplit, cross_val_score
from sklearn.metrics import mean_squared_error
import os

plt.rcParams["figure.figsize"] = (12, 9)
sns.set(style="whitegrid")
```

In [26]:
```
#Dataset
## Tickers: S&P 500 ETF plus ten large components

tickers = ["^GSPC", "AAPL", "MSFT", "GOOG", "META", "AMZN","NVDA", "TSLA", "BRK-
start, end = "2010-01-01", "2025-09-01"

for t in tickers:
    df = yf.download(t, start=start, end=end, interval="3mo", progress=False)
    print("\nTicker:", t)
    print("Columns:", df.columns.tolist())
    print("Head:\n", df.head(2))
```

```
Ticker: ^GSPC
Columns: [('Close', '^GSPC'), ('High', '^GSPC'), ('Low', '^GSPC'), ('Open', '^GSP
C'), ('Volume', '^GSPC')]
Head:
 Price             Close         High          Low         Open        Volume
Ticker           ^GSPC        ^GSPC        ^GSPC        ^GSPC         ^GSPC
Date
2010-01-01  1169.430054  1180.689941  1044.500000  1116.560059  279192470000
2010-04-01  1030.709961  1219.800049  1028.329956  1171.229980  354511440000

Ticker: AAPL
Columns: [('Close', 'AAPL'), ('High', 'AAPL'), ('Low', 'AAPL'), ('Open', 'AAPL'),
('Volume', 'AAPL')]
Head:
 Price          Close      High       Low      Open       Volume
Ticker          AAPL      AAPL      AAPL      AAPL         AAPL
Date
2010-01-01  7.054727  7.129178  5.711327  6.407194  38099247200
2010-04-01  7.550961  8.375914  5.981509  7.127077  47101037200

Ticker: MSFT
Columns: [('Close', 'MSFT'), ('High', 'MSFT'), ('Low', 'MSFT'), ('Open', 'MSFT'),
('Volume', 'MSFT')]
Head:
 Price           Close       High        Low       Open      Volume
Ticker           MSFT       MSFT       MSFT       MSFT        MSFT
Date
2010-01-01  21.930445  23.390477  20.642620  22.926262  3544531400
2010-04-01  17.308956  23.755621  17.263822  22.078134  4710971300

Ticker: GOOG
Columns: [('Close', 'GOOG'), ('High', 'GOOG'), ('Low', 'GOOG'), ('Open', 'GOOG'),
('Volume', 'GOOG')]
Head:
 Price           Close       High        Low       Open      Volume
Ticker           GOOG       GOOG       GOOG       GOOG        GOOG
Date
2010-01-01  14.029052  15.572415  12.863427  15.509087  9030100641
2010-04-01  11.006886  14.788982  11.001198  14.133689  8611573755

Ticker: META
Columns: [('Close', 'META'), ('High', 'META'), ('Low', 'META'), ('Open', 'META'),
('Volume', 'META')]
Head:
 Price           Close       High        Low       Open      Volume
Ticker           META       META       META       META        META
Date
2012-05-01  21.576982  33.245053  21.477597  28.712991  1188100200
2012-08-01  20.980661  24.101422  17.442472  21.368271  3311526900

Ticker: AMZN
Columns: [('Close', 'AMZN'), ('High', 'AMZN'), ('Low', 'AMZN'), ('Open', 'AMZN'),
('Volume', 'AMZN')]
Head:
 Price        Close    High     Low    Open       Volume
Ticker        AMZN    AMZN    AMZN    AMZN         AMZN
Date
2010-01-01  6.7885  6.9095  5.6910  6.8125  11980988000
2010-04-01  5.4630  7.5545  5.3005  6.7900   8946270000
```

```
Ticker: NVDA
Columns: [('Close', 'NVDA'), ('High', 'NVDA'), ('Low', 'NVDA'), ('Open', 'NVDA'), ('Volume', 'NVDA')]
Head:
 Price          Close      High       Low      Open       Volume
Ticker           NVDA      NVDA      NVDA      NVDA         NVDA
Date
2010-01-01   0.398845  0.434604  0.347270  0.424289  37507632000
2010-04-01   0.234035  0.415578  0.234035  0.400679  51575292000

Ticker: TSLA
Columns: [('Close', 'TSLA'), ('High', 'TSLA'), ('Low', 'TSLA'), ('Open', 'TSLA'), ('Volume', 'TSLA')]
Head:
 Price          Close   High       Low      Open      Volume
Ticker           TSLA   TSLA      TSLA      TSLA        TSLA
Date
2010-06-01   1.298667  1.728  0.998667  1.666667  1194210000
2010-09-01   2.355333  2.400  1.300000  1.308000   793632000

Ticker: BRK-B
Columns: [('Close', 'BRK-B'), ('High', 'BRK-B'), ('Low', 'BRK-B'), ('Open', 'BRK-B'), ('Volume', 'BRK-B')]
Head:
 Price           Close       High        Low       Open     Volume
Ticker           BRK-B      BRK-B      BRK-B      BRK-B      BRK-B
Date
2010-01-01   81.269997  83.570000  64.720001  66.000000  841106900
2010-04-01   79.690002  81.949997  68.480003  81.599998  497986800

Ticker: JPM
Columns: [('Close', 'JPM'), ('High', 'JPM'), ('Low', 'JPM'), ('Open', 'JPM'), ('Volume', 'JPM')]
Head:
 Price           Close       High        Low       Open      Volume
Ticker             JPM        JPM        JPM        JPM         JPM
Date
2010-01-01   29.927181  30.796574  24.764324  27.947641  2660845800
2010-04-01   24.512856  32.273139  24.445897  30.150610  3060674400

Ticker: JNJ
Columns: [('Close', 'JNJ'), ('High', 'JNJ'), ('Low', 'JNJ'), ('Open', 'JNJ'), ('Volume', 'JNJ')]
Head:
 Price           Close       High        Low       Open     Volume
Ticker             JNJ        JNJ        JNJ        JNJ        JNJ
Date
2010-01-01   41.002075  41.473725  38.920530  40.693932  708414300
2010-04-01   37.425018  41.949476  36.468163  41.423527  950979900
```

In [49]:
```
#Add SPY and XLK quarterly returns to our DataFrame
for etf in ["SPY", "XLK"]:
    df = yf.download(etf, start="2010-01-01", end="2025-09-01",
                     interval="3mo", progress=False)
    print("\nTicker:", t)
    print("Columns:", df.columns.tolist())
    print("Head:\n", df.head(2))
```

```
Ticker: JNJ
Columns: [('Close', 'SPY'), ('High', 'SPY'), ('Low', 'SPY'), ('Open', 'SPY'), ('V
olume', 'SPY')]
Head:
 Price          Close       High        Low       Open       Volume
Ticker            SPY        SPY        SPY        SPY          SPY
Date
2010-01-01  88.040848  88.921255  78.694975  84.556840  12058443200
2010-04-01  77.991447  92.271998  77.734546  89.007873  17007028100

Ticker: JNJ
Columns: [('Close', 'XLK'), ('High', 'XLK'), ('Low', 'XLK'), ('Open', 'XLK'), ('V
olume', 'XLK')]
Head:
 Price          Close       High        Low       Open     Volume
Ticker            XLK        XLK        XLK        XLK        XLK
Date
2010-01-01  18.671961  18.898287  16.740099  18.704292  718489100
2010-04-01  16.542023  19.590945  16.493370  18.780061  822849300
```

In [50]:

```python
#Quarterly Prices

tickers = [ "^GSPC", "AAPL", "MSFT", "GOOG", "META", "AMZN","NVDA", "TSLA", "BRK
start, end = "2010-01-01", "2025-09-01"

frames = []
for t in tickers:
    df = yf.download(t, start=start, end=end, interval="3mo", progress=False)
    col = next((c for c in ["Adj Close", "Close"] if c in df.columns), None)
    s = df[col].copy()
    s.name = t                    # set the Series name
    frames.append(s)

prices_q = pd.concat(frames, axis=1)
print("Quarterly prices:\n", prices_q.head())
```

```
Quarterly prices:
 Ticker            ^GSPC       AAPL        MSFT        GOOG  META    AMZN  \
Date
2010-01-01  1169.430054  7.054727  21.930445  14.029052   NaN  6.7885
2010-04-01  1030.709961  7.550961  17.308956  11.006886   NaN  5.4630
2010-06-01          NaN       NaN        NaN        NaN   NaN     NaN
2010-07-01  1141.199951  8.518209  18.505398  13.006658   NaN  7.8530
2010-09-01          NaN       NaN        NaN        NaN   NaN     NaN

Ticker          NVDA      TSLA      BRK-B        JPM        JNJ        SPY  \
Date
2010-01-01  0.398845       NaN  81.269997  29.927181  41.002075  88.040848
2010-04-01  0.234035       NaN  79.690002  24.512856  37.425018  77.991447
2010-06-01       NaN  1.298667        NaN        NaN        NaN        NaN
2010-07-01  0.267731       NaN  82.680000  25.512232  39.621323  86.645172
2010-09-01       NaN  2.355333        NaN        NaN        NaN        NaN

Ticker            XLK
Date
2010-01-01  18.671961
2010-04-01  16.542023
2010-06-01        NaN
2010-07-01  18.732578
2010-09-01        NaN
```

In [51]:
```
# Quarterly simple returns
returns_q = prices_q.pct_change().dropna()

print("Quarterly returns:\n", returns_q.head())
```

```
Quarterly returns:
 Ticker         ^GSPC      AAPL      MSFT      GOOG      META      AMZN  \
Date
2012-06-01  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2012-07-01  0.057636  0.142295 -0.020750  0.300705  0.000000  0.113729
2012-08-01  0.000000  0.000000  0.000000  0.000000 -0.027637  0.000000
2012-09-01  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2012-10-01 -0.010051 -0.198839 -0.096541 -0.062452  0.000000 -0.013566

Ticker          NVDA      TSLA     BRK-B       JPM       JNJ       SPY  \
Date
2012-06-01  0.000000 -0.033221  0.000000  0.000000  0.000000  0.000000
2012-07-01 -0.034732  0.000000  0.058442  0.140357  0.029911  0.063306
2012-08-01  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2012-09-01  0.000000  0.185835  0.000000  0.000000  0.000000  0.000000
2012-10-01 -0.080960  0.000000  0.017007  0.095272  0.026513 -0.005555

Ticker           XLK
Date
2012-06-01  0.000000
2012-07-01  0.077261
2012-08-01  0.000000
2012-09-01  0.000000
2012-10-01 -0.060358
```

```
C:\Users\OkechPC\AppData\Local\Temp\ipykernel_38808\3376601740.py:2: FutureWarnin
g: The default fill_method='pad' in DataFrame.pct_change is deprecated and will b
e removed in a future version. Either fill in any non-leading NA values prior to
calling pct_change or specify 'fill_method=None' to not fill NA values.
  returns_q = prices_q.pct_change().dropna()
```

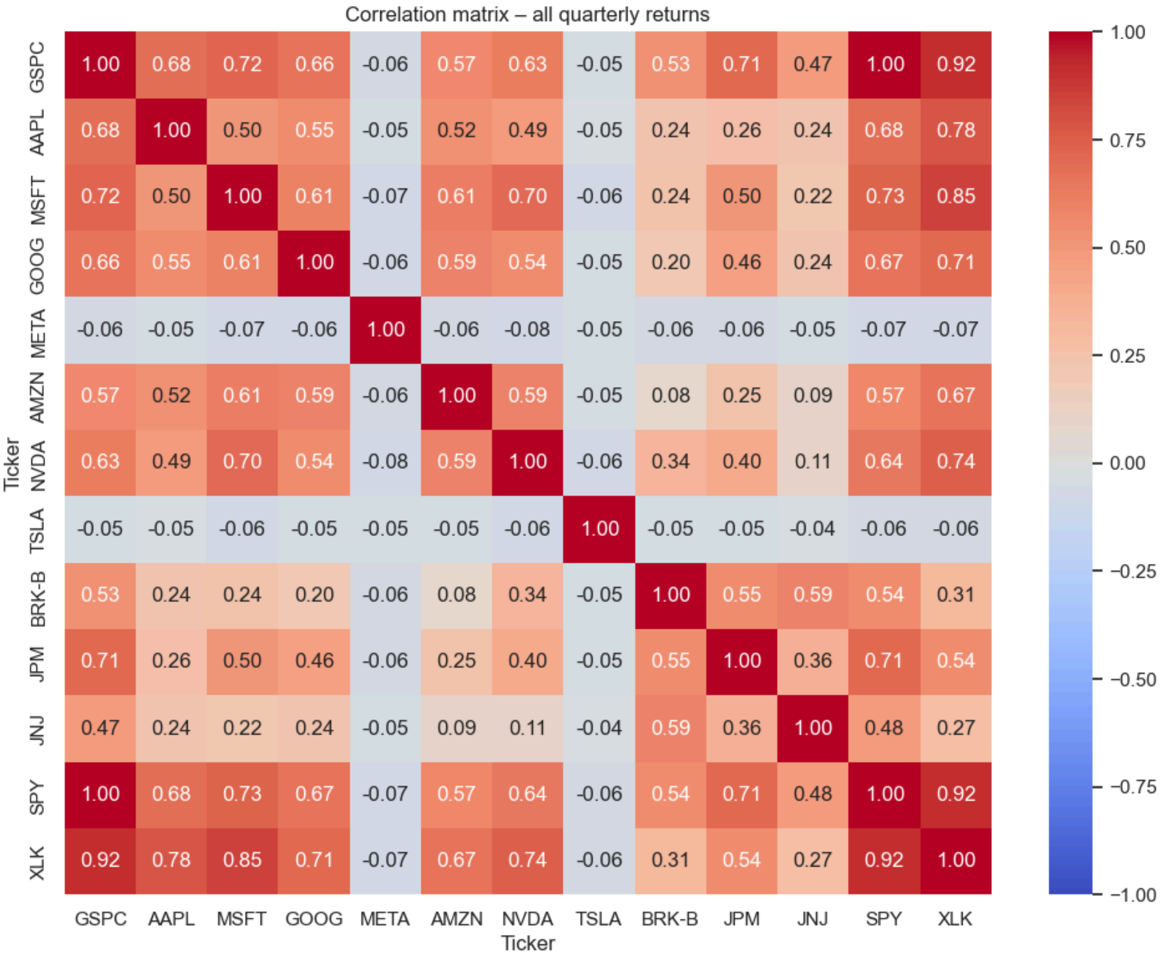
In [53]:
```
#S&P 500's quarterly return as the dependent variable and the others as predicto
#Rename the S&P 500 column

returns_q = returns_q.rename(columns={"^GSPC": "GSPC"})
y = returns_q["GSPC"]
X = returns_q[["AAPL", "MSFT", "GOOG", "META", "AMZN","NVDA", "TSLA", "BRK-B", "

#Correlation plot of all variables

corr_all = returns_q.corr()

plt.figure(figsize=(12,9))
sns.heatmap(corr_all, annot=True, fmt=".2f", cmap="coolwarm",
            vmin=-1, vmax=1)
plt.title("Correlation matrix - all quarterly returns")
plt.show()
```



In [36]:

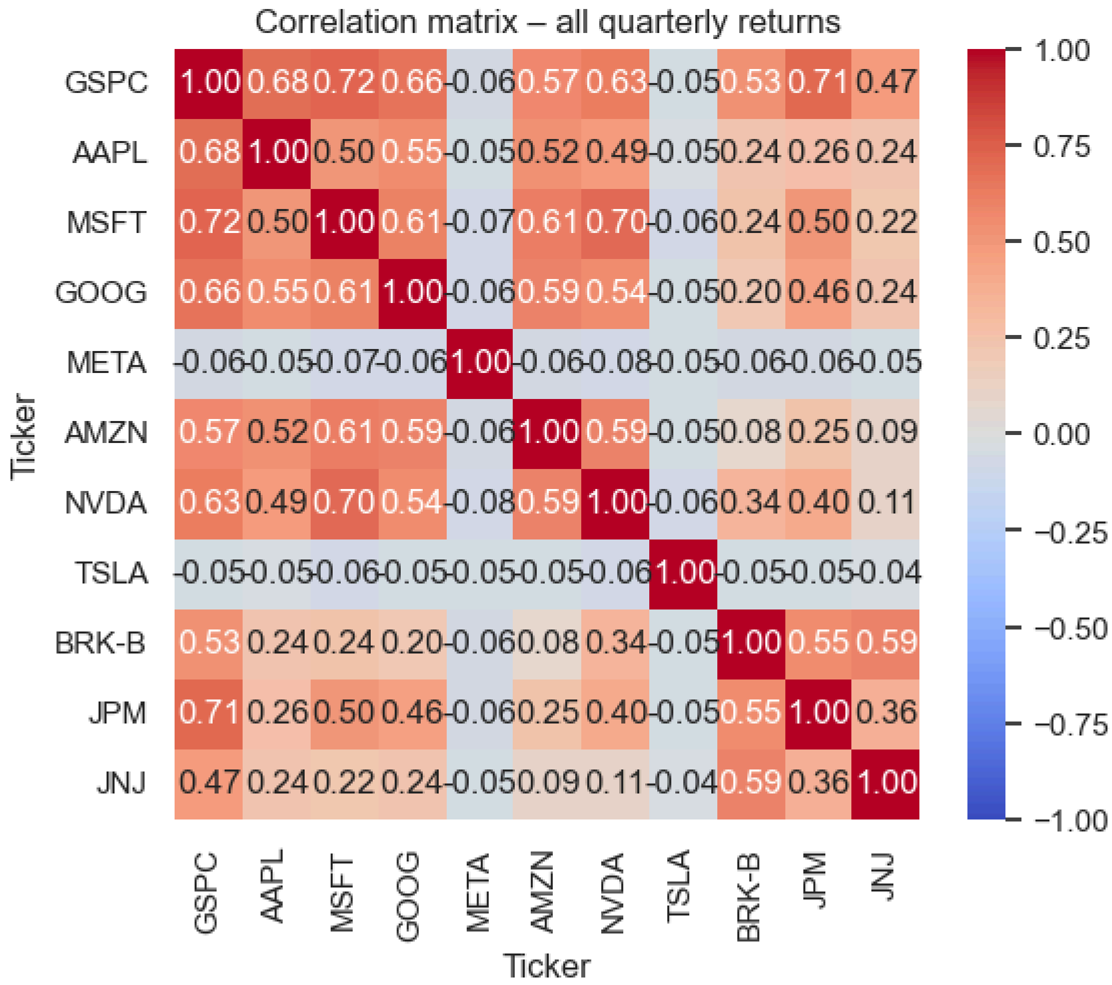
```python
#S&P500's quarterly return as the dependent variable and the others as predictor

y = returns_q["GSPC"]
X = returns_q[["AAPL", "MSFT", "GOOG", "META", "AMZN","NVDA", "TSLA", "BRK-B", "

#Correlation plot of all variables

corr_all = returns_q.corr()

plt.figure(figsize=(6,5))
sns.heatmap(corr_all, annot=True, fmt=".2f", cmap="coolwarm",
            vmin=-1, vmax=1)
plt.title("Correlation matrix – all quarterly returns")
plt.show()
```

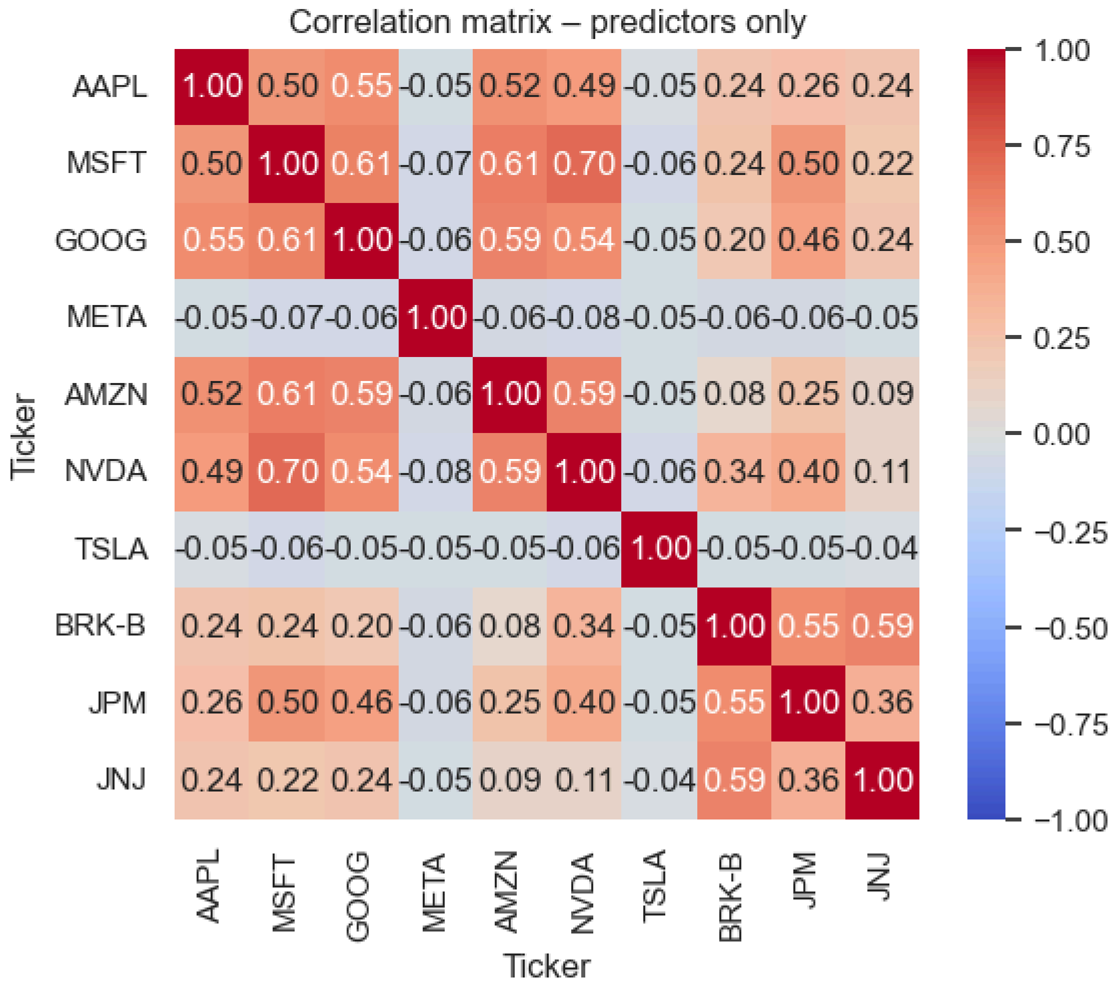


In [37]:

```
# Correlation plot of all independent variables

corr_predictors = X.corr()

plt.figure(figsize=(6,5))
sns.heatmap(corr_predictors, annot=True, fmt=".2f", cmap="coolwarm",
            vmin=-1, vmax=1)
plt.title("Correlation matrix – predictors only")
plt.show()
```



In [54]:

```
#S&P500 Excess Return Regression Model with Independent Variables

returns_q = returns_q.rename(columns={"BRK-B": "BRK_B"})

model_1 = smf.ols(
    formula="GSPC ~ AAPL + MSFT + GOOG + META + AMZN + NVDA + TSLA + BRK_B + JPM
    data=returns_q
).fit()

print(model_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   GSPC   R-squared:                       0.998
Model:                            OLS   Adj. R-squared:                  0.998
Method:                 Least Squares   F-statistic:                     7313.
Date:                Mon, 29 Sep 2025   Prob (F-statistic):          2.55e-196
Time:                        13:52:27   Log-Likelihood:                 781.32
No. Observations:                 159   AIC:                            -1537.
Df Residuals:                     146   BIC:                            -1497.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.0011      0.000     -6.370      0.000      -0.001      -0.001
AAPL          -0.0039      0.003     -1.226      0.222      -0.010       0.002
MSFT          -0.0046      0.005     -0.884      0.378      -0.015       0.006
GOOG          -0.0047      0.003     -1.680      0.095      -0.010       0.001
META           0.0020      0.001      1.648      0.101      -0.000       0.004
AMZN          -0.0045      0.002     -1.886      0.061      -0.009       0.000
NVDA          -0.0026      0.002     -1.693      0.093      -0.006       0.000
TSLA           0.0008      0.001      1.413      0.160      -0.000       0.002
BRK_B         -0.0091      0.005     -1.894      0.060      -0.019       0.000
JPM           -0.0069      0.004     -1.930      0.056      -0.014       0.000
JNJ           -0.0119      0.005     -2.469      0.015      -0.021      -0.002
SPY            1.0286      0.017     59.784      0.000       0.995       1.063
XLK           -0.0016      0.016     -0.100      0.921      -0.033       0.030
==============================================================================
Omnibus:                       44.495   Durbin-Watson:                   2.558
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               71.301
Skew:                          -1.505   Prob(JB):                     3.29e-16
Kurtosis:                       4.306   Cond. No.                         157.
==============================================================================

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

In [41]:
```python
# Parameters with high precision
model_1.summary2().tables[1]
```

Out[41]:

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	-0.004045	0.001525	-2.652182	8.869392e-03	-0.007059	-0.001031
AAPL	0.148737	0.018724	7.943856	4.552458e-13	0.111737	0.185737
MSFT	0.126245	0.031951	3.951200	1.200054e-04	0.063106	0.189384
GOOG	0.038868	0.025373	1.531826	1.277001e-01	-0.011273	0.089008
META	0.007546	0.011127	0.678140	4.987419e-01	-0.014442	0.029533
AMZN	0.048658	0.021131	2.302652	2.269211e-02	0.006900	0.090415
NVDA	0.012014	0.012845	0.935314	3.511501e-01	-0.013370	0.037398
TSLA	0.002915	0.005016	0.581196	5.619926e-01	-0.006996	0.012826
BRK_B	0.083064	0.038901	2.135291	3.438333e-02	0.006192	0.159937
JPM	0.206532	0.025678	8.043155	2.593367e-13	0.155789	0.257274
JNJ	0.135825	0.039593	3.430568	7.807945e-04	0.057585	0.214065

In the above results, we can see Johnson & Johnson and SPY estimates are significant (p-values less than 0.05).

In [55]:

```
# Regression model to check multicollinearity among independent variables

model_AAPL = smf.ols(
    formula="AAPL ~ MSFT + GOOG + META + AMZN + NVDA + TSLA + BRK_B + JPM + JNJ
    data=returns_q
).fit()

print(model_AAPL.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                   AAPL   R-squared:                       0.755
Model:                            OLS   Adj. R-squared:                  0.737
Method:                 Least Squares   F-statistic:                     41.15
Date:                Mon, 29 Sep 2025   Prob (F-statistic):           2.16e-39
Time:                        13:52:39   Log-Likelihood:                 262.97
No. Observations:                 159   AIC:                            -501.9
Df Residuals:                     147   BIC:                            -465.1
Df Model:                          11                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.0036      0.004      0.814      0.417      -0.005       0.012
MSFT          -0.8288      0.117     -7.107      0.000      -1.059      -0.598
GOOG           0.0426      0.073      0.584      0.560      -0.102       0.187
META          -0.0066      0.032     -0.210      0.834      -0.069       0.056
AMZN           0.0679      0.061      1.108      0.270      -0.053       0.189
NVDA          -0.1036      0.039     -2.678      0.008      -0.180      -0.027
TSLA          -0.0026      0.014     -0.180      0.857      -0.031       0.026
BRK_B          0.3762      0.121      3.107      0.002       0.137       0.616
JPM           -0.1221      0.092     -1.329      0.186      -0.304       0.060
JNJ            0.2165      0.123      1.753      0.082      -0.028       0.460
SPY           -1.5939      0.427     -3.733      0.000      -2.438      -0.750
XLK            3.1376      0.320      9.801      0.000       2.505       3.770
==============================================================================
Omnibus:                       47.182   Durbin-Watson:                   2.012
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              448.223
Skew:                          -0.695   Prob(JB):                     4.67e-98
Kurtosis:                      11.107   Cond. No.                         139.
==============================================================================

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

In [56]:
```
# Parameters with 6 significant digits
model_AAPL.summary2().tables[1]
```

Out[56]:

	Coef.	Std.Err.	t	P>|t|	[0.025	0.975]
Intercept	0.003561	0.004377	0.813749	4.171047e-01	-0.005088	0.012211
MSFT	-0.828794	0.116618	-7.106916	4.762701e-11	-1.059258	-0.598329
GOOG	0.042588	0.072978	0.583572	5.604027e-01	-0.101634	0.186810
META	-0.006644	0.031616	-0.210134	8.338538e-01	-0.069125	0.055837
AMZN	0.067852	0.061252	1.107742	2.697827e-01	-0.053197	0.188900
NVDA	-0.103580	0.038677	-2.678061	8.247812e-03	-0.180016	-0.027145
TSLA	-0.002567	0.014249	-0.180128	8.573003e-01	-0.030725	0.025592
BRK_B	0.376202	0.121089	3.106822	2.270256e-03	0.136902	0.615502
JPM	-0.122129	0.091922	-1.328614	1.860342e-01	-0.303790	0.059531
JNJ	0.216453	0.123489	1.752808	8.171981e-02	-0.027591	0.460497
SPY	-1.593886	0.426923	-3.733428	2.694290e-04	-2.437586	-0.750187
XLK	3.137576	0.320116	9.801387	9.155974e-18	2.504953	3.770200

In [57]:

```python
# Add a constant column for the intercept

import statsmodels.api as sm
X_with_const = sm.add_constant(X)

vif = pd.DataFrame()
vif["Variable"] = X_with_const.columns
vif["VIF"] = [variance_inflation_factor(X_with_const.values, i)
              for i in range(X_with_const.shape[1])]

print(vif)
```

```
   Variable        VIF
0     const   1.320004
1      AAPL   4.079076
2      MSFT   6.096314
3      GOOG   2.232055
4      META   1.013497
5      AMZN   2.207639
6      NVDA   3.052805
7      TSLA   1.010518
8     BRK-B   2.936833
9       JPM   3.133554
10      JNJ   2.143507
11      SPY  27.118045
12      XLK  42.098410
```

We can see Microsoft, SPY and XLK Indexes have VIF values larger than 5 indicating severe multicollinearity. Since these variables have severe multicollinearity issues, we need to dig further into this variable and maybe drop them from the model. We can also use Principle Component Analysis.

In [70]:

```python
#PCA(Principal Component Analysis)
#Libraries

from scipy.stats import boxcox
from sklearn import preprocessing
```

```
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
```

In [77]:
```
#Select Ten Variables & Standardize

columns_to_use = ["AAPL", "MSFT", "GOOG", "META", "AMZN",
                  "NVDA", "TSLA", "BRK_B", "JPM", "JNJ", "SPY", "XLK"]

pc = returns_q[columns_to_use]
# Standardize
pca_data = scale(pc)

# PCA
pca = PCA(n_components=12)
pca.fit(pca_data)
```

Out[77]:
```
PCA
PCA(n_components=12)
```

In [78]:
```
# Get proportions of variance and cumulative proportion of variance
pr_var = pca.explained_variance_ratio_
cum_pr = np.cumsum(pca.explained_variance_ratio_)
ind = ["Proportion of variance", "Cumulative proportion of variance"]
cols = ["PC1", "PC2", "PC3", "PC4", "PC5", "PC6", "PC7", "PC8", "PC9", "PC10", "
pd.DataFrame(np.vstack((pr_var, cum_pr)), ind, columns=cols)
```

Out[78]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	P
Proportion of variance	0.467415	0.132597	0.087362	0.078195	0.060061	0.045638	0.040971	0.032
Cumulative proportion of variance	0.467415	0.600012	0.687374	0.765569	0.825630	0.871268	0.912240	0.944

Looking at the proportion of variance and cumulative proportion in the above chart we can see that the first principal component accounts for about 46.7% of the total variance in the data. The first seven principal components explain about 91.2% of the total variance.

In [79]:
```
# Coefficients (Loadings) of 12 Principal Components
pc_res = pd.DataFrame(pca.components_.T, index=list(pc.columns), columns=cols)
pc_res
```

Out[79]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	
AAPL	0.309535	-0.148488	-0.002510	0.043996	0.565727	0.089767	0.581412	-0.05
MSFT	0.353194	-0.154890	0.000112	-0.005376	-0.231410	0.039169	-0.198365	0.60
GOOG	0.326337	-0.148612	-0.000909	0.014233	0.058669	-0.529424	-0.179600	-0.46
META	-0.042809	-0.028931	-0.699319	0.710901	-0.025403	0.026300	-0.033561	-0.00
AMZN	0.296498	-0.345422	0.001024	-0.014688	0.115723	0.067595	-0.501865	-0.31
NVDA	0.326891	-0.167354	0.001577	-0.030023	-0.295965	0.576736	-0.092894	-0.11
TSLA	-0.035934	-0.023634	0.714777	0.696447	-0.022517	0.022907	-0.029133	-0.00
BRK_B	0.210070	0.578836	-0.001791	0.015041	-0.072145	0.430482	0.053073	-0.39
JPM	0.280945	0.327083	-0.002472	0.028709	-0.531885	-0.415462	0.224015	-0.04
JNJ	0.178814	0.554386	-0.002234	0.027275	0.479700	-0.072423	-0.466694	0.27
SPY	0.398953	0.119375	-0.003956	0.056018	0.020096	-0.092397	0.154982	0.09
XLK	0.401069	-0.139737	-0.002466	0.036934	0.046890	0.001573	0.181827	0.24

For $PC1$, the XLK Index excess return has the highest absolute value of the coefficient. This means that the XLK Index excess return has the largest impact on $PC1$. Therefore, we can use the XLK Index excess return as a proxy for $PC1$. Following the same logic, we can use BRK_B excess return as a proxy for $PC2$. We also know that the first seven principal components cover around 91.2% of the data variation. In this case, we can just choose XLK Index excess return, BRK_B excess return, S&P500's revenue growth, TSLA excess return, JNJ excess return and NVDA excess return to run a regression model.

In [76]:

```
# OLS for revised model
model_3 = smf.ols(
    "GSPC ~ XLK + BRK_B + TSLA + JNJ + NVDA",
    data=returns_q,
).fit()
model_3.summary()
```

Out[76]:

OLS Regression Results

Dep. Variable:	GSPC	**R-squared:**	0.925
Model:	OLS	**Adj. R-squared:**	0.922
Method:	Least Squares	**F-statistic:**	377.1
Date:	Mon, 29 Sep 2025	**Prob (F-statistic):**	4.26e-84
Time:	16:31:07	**Log-Likelihood:**	478.36
No. Observations:	159	**AIC:**	-944.7
Df Residuals:	153	**BIC:**	-926.3
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-0.0021	0.001	-1.975	0.050	-0.004	1.02e-06
XLK	0.6658	0.025	26.771	0.000	0.617	0.715
BRK_B	0.1974	0.025	8.048	0.000	0.149	0.246
TSLA	0.0015	0.004	0.416	0.678	-0.006	0.009
JNJ	0.0946	0.028	3.360	0.001	0.039	0.150
NVDA	-0.0366	0.009	-4.068	0.000	-0.054	-0.019

Omnibus:	77.446	**Durbin-Watson:**	2.043
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	442.966
Skew:	-1.686	**Prob(JB):**	6.47e-97
Kurtosis:	10.449	**Cond. No.**	36.1

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

From the result , we can see that the XLK Index has become significant, and the adjusted R^2 has improved to 0.925. By using the information provided by PCA, we can reduce the impact of multicollinearity among independent variables.

In [ ]: