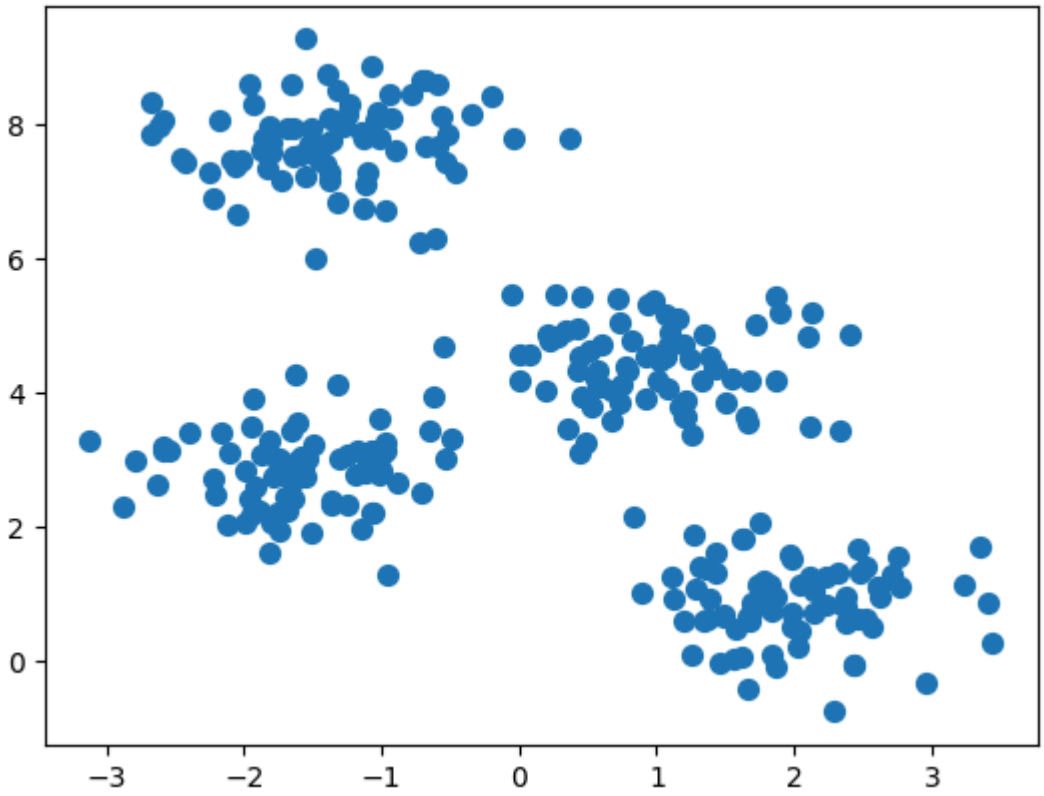
Step 1

K-means

In [1]:
```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
```

In [2]:
```
# Generate synthetic data
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.6, random_state=0)

# Visualize the data
plt.scatter(X[:, 0], X[:, 1], s=50)
plt.show()
```



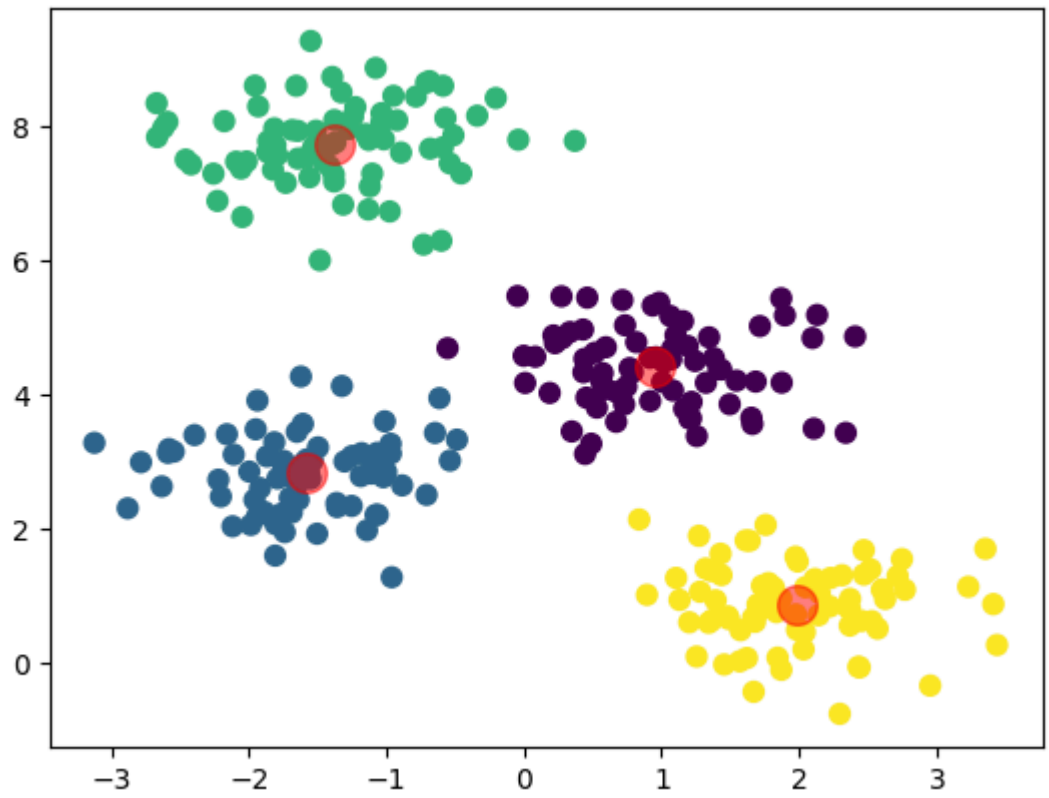
In [3]:
```
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1416: FutureWar
ning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

Out[3]:
```
KMeans
KMeans(n_clusters=4)
```

In [4]:
```
y_kmeans = kmeans.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
```

```
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=200, alpha=0.5)
plt.show()
```



PCA

In [5]:
```
import numpy as np
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import pandas_datareader.data as web
import scipy as sp
from datetime import datetime
from sklearn.preprocessing import StandardScaler
import seaborn as sns
```

In [6]:
```
# downloading the data
start = datetime(2024, 1, 1)
end = datetime(2024, 9, 10)
data = [
    "DEXINUS",
    "DEXHKUS",
    "DEXVZUS",
    "DEXBZUS",
    "DEXCHUS",
    "DEXJPUS",
    "DEXMXUS",
    "DEXCAUS",
    "DEXKOUS",
    "DEXSFUS",
]
data = web.DataReader(data, "fred", start, end).dropna(how="all").ffill()
```

In [7]:
```
data
```

Out[7]:

DATE	DEXINUS	DEXHKUS	DEXVZUS	DEXBZUS	DEXCHUS	DEXJPUS	DEXMXUS	DEXCAUS	DE
2024-01-02	83.31	7.8150	35.8856	4.8943	7.1426	141.89	17.0140	1.3310	
2024-01-03	83.30	7.8078	35.7773	4.9239	7.1497	143.55	17.0560	1.3358	
2024-01-04	83.21	7.8081	35.8597	4.9143	7.1589	144.59	17.0480	1.3355	
2024-01-05	83.16	7.8092	35.9444	4.8744	7.1450	144.52	16.9141	1.3331	
2024-01-08	83.13	7.8065	35.9165	4.8745	7.1540	143.89	16.8030	1.3361	
...	...	...	...	...	...	...	...	...	
2024-09-04	83.95	7.7969	36.5475	5.6403	7.1119	144.31	19.8161	1.3519	
2024-09-05	83.96	7.7958	36.5520	5.5924	7.0921	143.43	19.9926	1.3515	
2024-09-06	83.91	7.7945	36.5929	5.5884	7.0876	142.13	20.0399	1.3561	
2024-09-09	83.93	7.7969	36.6041	5.6042	7.1133	142.92	19.8845	1.3560	
2024-09-10	83.95	7.7970	36.5937	5.6483	7.1205	142.28	20.0831	1.3607	

175 rows × 10 columns

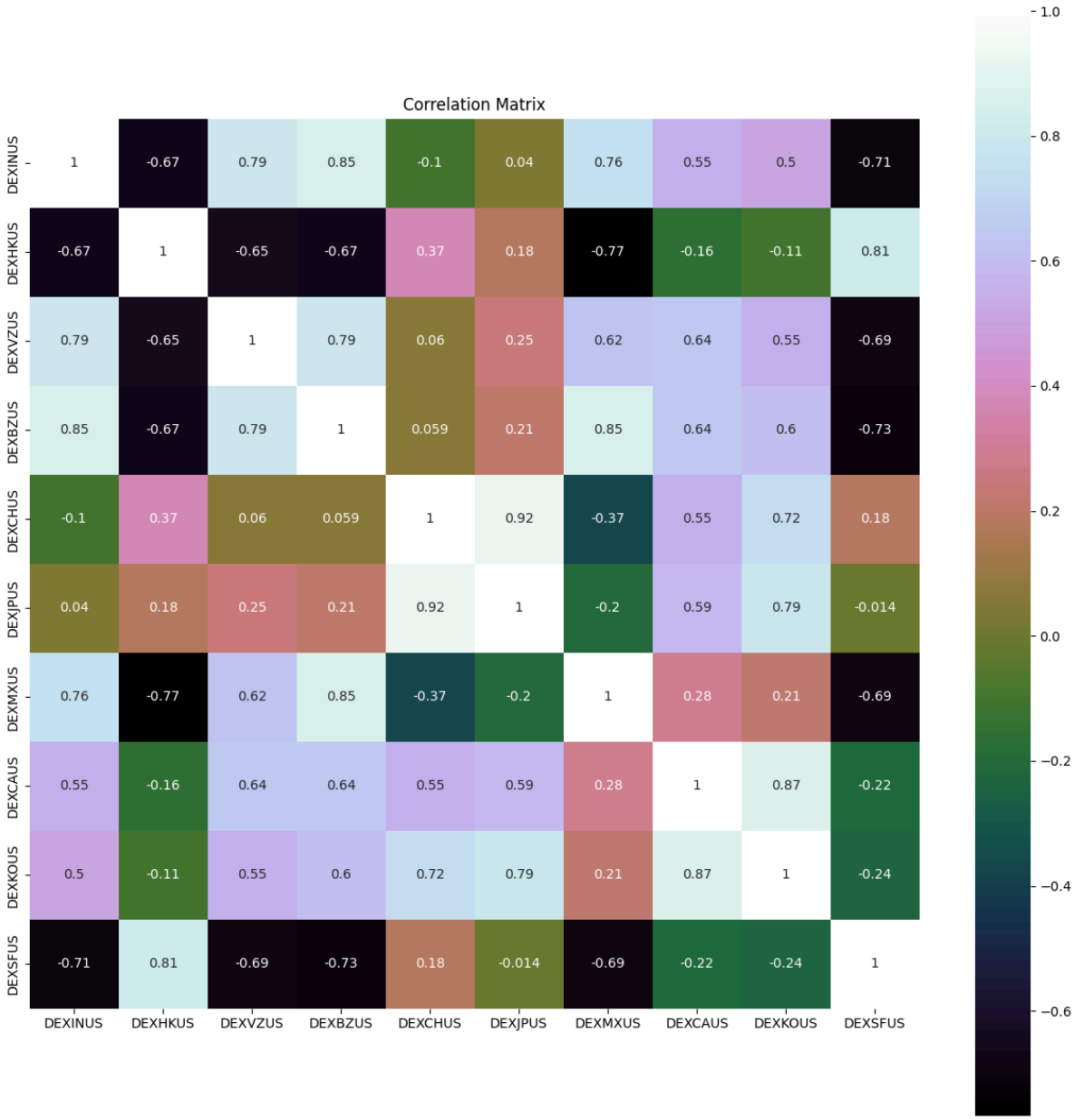
In [8]:
```
df = data.copy()
df.head(10)
```

Out[8]:

DATE	DEXINUS	DEXHKUS	DEXVZUS	DEXBZUS	DEXCHUS	DEXJPUS	DEXMXUS	DEXCAUS	DE
2024-01-02	83.31	7.8150	35.8856	4.8943	7.1426	141.89	17.0140	1.3310	
2024-01-03	83.30	7.8078	35.7773	4.9239	7.1497	143.55	17.0560	1.3358	
2024-01-04	83.21	7.8081	35.8597	4.9143	7.1589	144.59	17.0480	1.3355	
2024-01-05	83.16	7.8092	35.9444	4.8744	7.1450	144.52	16.9141	1.3331	
2024-01-08	83.13	7.8065	35.9165	4.8745	7.1540	143.89	16.8030	1.3361	
2024-01-09	83.09	7.8158	35.8487	4.8956	7.1675	144.35	16.9360	1.3384	
2024-01-10	83.01	7.8196	35.8273	4.8929	7.1711	145.68	16.9880	1.3379	
2024-01-11	83.03	7.8186	35.8538	4.8838	7.1670	146.03	16.9867	1.3439	
2024-01-12	82.92	7.8195	35.8730	4.8536	7.1665	144.86	16.8510	1.3399	
2024-01-16	83.05	7.8263	35.8845	4.9163	7.1874	147.01	17.1499	1.3472	

In [9]:
```
# correlation
correlation = df.corr()
plt.figure(figsize=(15, 15))
plt.title("Correlation Matrix")
plt.suptitle(
    "Correlation Matrix", fontweight="bold", horizontalalignment="right"
)
sns.heatmap(correlation, vmax=1, square=True, annot=True, cmap="cubehelix");
```

Correlation Matrix

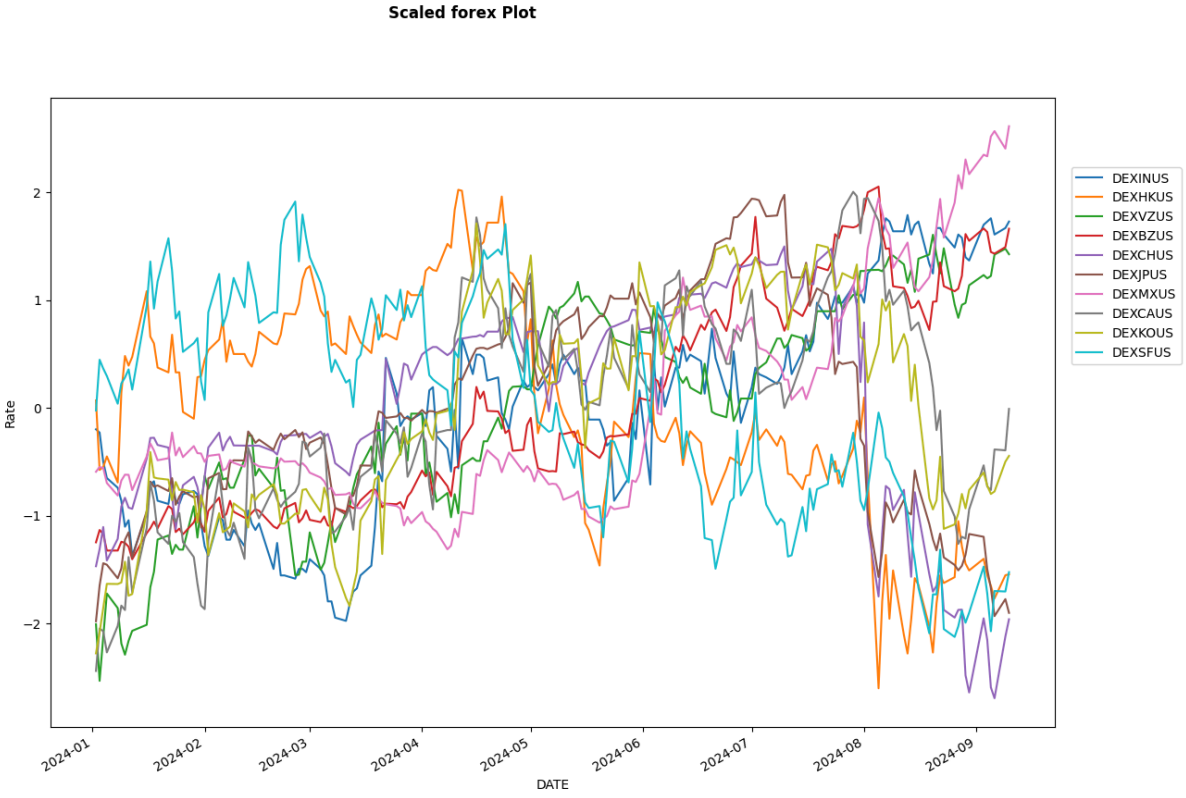


In [10]:
```
scaler = StandardScaler().fit(df)
rescaleddf = pd.DataFrame(scaler.fit_transform(df), columns=df.columns, index=df.ir
# summarize transformed data
df.dropna(how="any", inplace=True)
rescaleddf.dropna(how="any", inplace=True)
rescaleddf.head(2)
```

Out[10]:

DATE	DEXINUS	DEXHKUS	DEXVZUS	DEXBZUS	DEXCHUS	DEXJPUS	DEXMXUS	DEXCAUS	DE
2024-01-02	-0.199610	0.068718	-2.007573	-1.247631	-1.469197	-1.979937	-0.593190	-2.440494	-2
2024-01-03	-0.229723	-0.576470	-2.532519	-1.133455	-1.311090	-1.649009	-0.549333	-2.047673	-2

In [11]:
```
rescaleddf.plot(figsize=(14, 10))
plt.ylabel("Rate")
plt.legend(bbox_to_anchor=(1.01, 0.9), loc=2)
plt.suptitle(
    "Scaled forex Plot", fontweight="bold", horizontalalignment="right"
)
plt.show()
```



In [12]:
```
n_components = 5  # We'll use 5 principal components for illustration
pca = PCA(n_components=n_components)
results = pca.fit(rescaleddf)
```

In [13]:
```
results
```

Out[13]:
```
PCA
PCA(n_components=5)
```

In [14]:
```
loadings = results.components_
num_pc = results.n_features_
pc_list = ["PC" + str(i) for i in list(range(1, num_pc + 1))]
loadings_df = pd.DataFrame.from_dict(dict(zip(pc_list, loadings)))
loadings_df["variable"] = df.columns.values
loadings_df = loadings_df.set_index("variable")
loadings_df
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/deprecation.py:101: FutureWa
rning: Attribute `n_features_` was deprecated in version 1.2 and will be removed i
n 1.4. Use `n_features_in_` instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[14]:

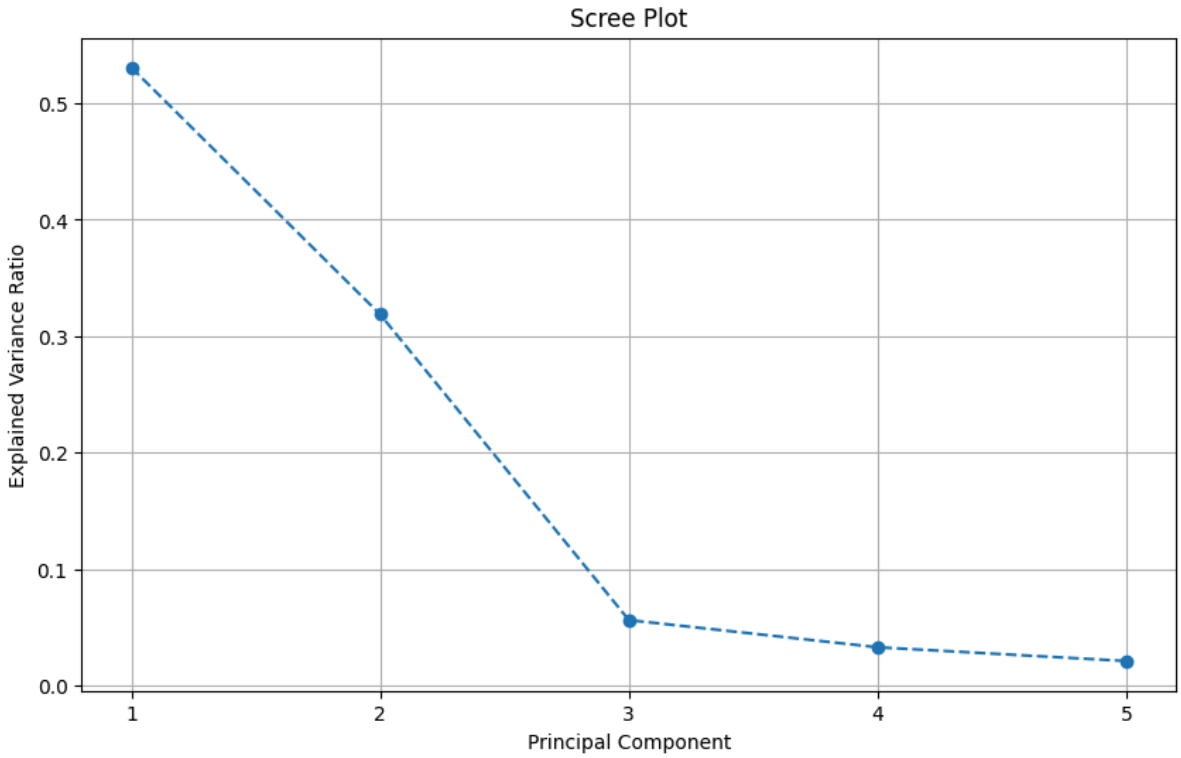
variable	PC1	PC2	PC3	PC4	PC5
DEXINUS	0.391460	0.094065	0.241619	0.158113	-0.595147
DEXHKUS	-0.318341	-0.289571	0.362323	-0.044625	-0.566592
DEXVZUS	0.389324	-0.001434	-0.018646	0.640274	0.204510
DEXBZUS	0.414428	0.025043	0.125106	-0.367672	-0.004384
DEXCHUS	0.043006	-0.537200	-0.233027	-0.155935	-0.008504
DEXJPUS	0.120326	-0.500211	-0.390240	-0.130705	0.074773
DEXMXUS	0.342078	0.261919	0.208786	-0.566297	0.186905
DEXCAUS	0.300232	-0.323493	0.436355	0.224028	0.237293
DEXKOUS	0.287529	-0.396362	0.104611	-0.120897	-0.087705
DEXSFUS	-0.340495	-0.193711	0.585503	-0.031281	0.422295

In [15]:
```
loadings_df.index
```

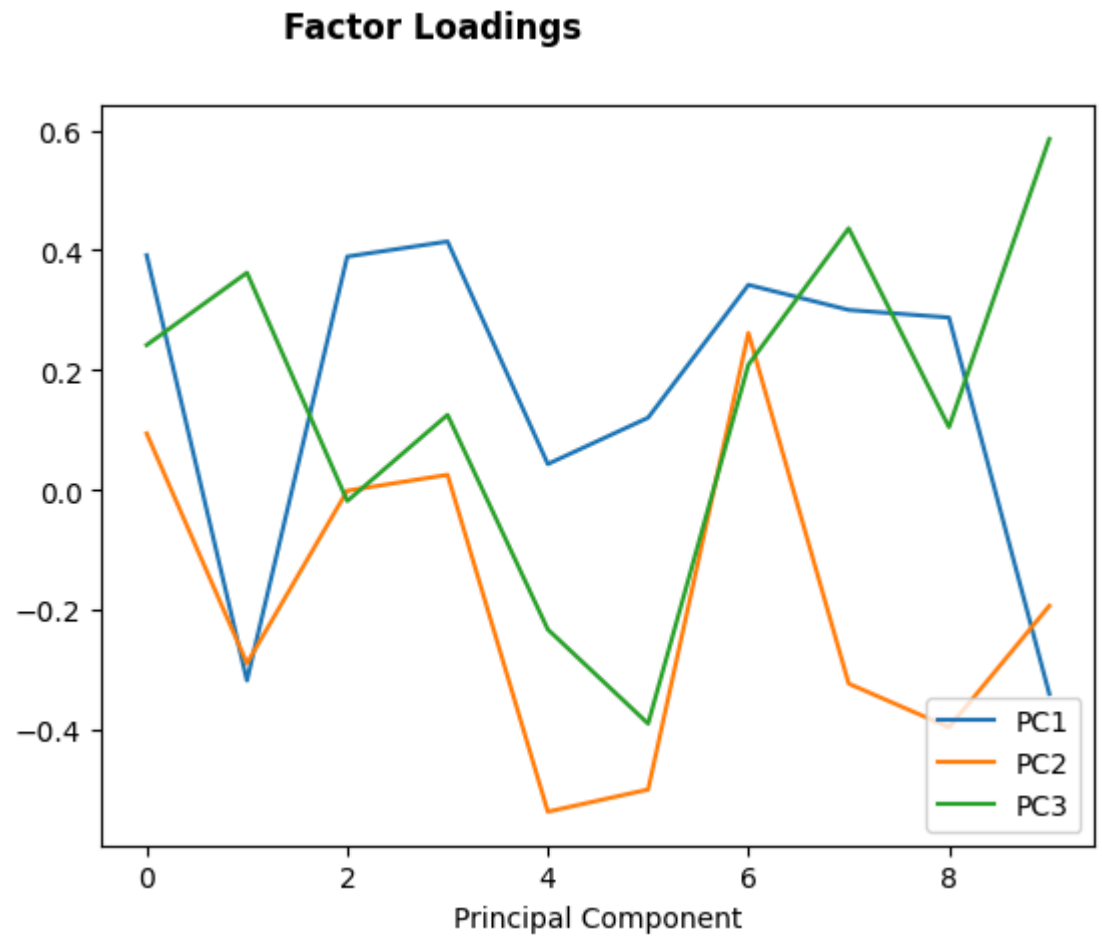
Out[15]:
```
Index(['DEXINUS', 'DEXHKUS', 'DEXVZUS', 'DEXBZUS', 'DEXCHUS', 'DEXJPUS',
       'DEXMXUS', 'DEXCAUS', 'DEXKOUS', 'DEXSFUS'],
      dtype='object', name='variable')
```

In [16]:
```
explained_variance_ratio = results.explained_variance_ratio_
```

In [17]:
```
# Plot the scree plot
plt.figure(figsize=(10, 6))
plt.plot(range(1, n_components + 1), explained_variance_ratio, marker='o', linestyl
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance Ratio')
plt.xticks(range(1, n_components + 1))
plt.grid(True)
plt.show()
```



In [18]:
```
plt.plot(results.components_[0:3].T)
plt.xlabel("Principal Component")
plt.suptitle(
    "Factor Loadings", fontweight="bold", horizontalalignment="right"
)
plt.legend(["PC1", "PC2", "PC3"], loc="lower right")
plt.show()
```



In [19]:
```
!pip install pca
```

```
Requirement already satisfied: pca in /usr/local/lib/python3.10/dist-packages (2.
0.7)
Requirement already satisfied: datazets in /usr/local/lib/python3.10/dist-packages
(from pca) (0.1.9)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packa
ges (from pca) (0.14.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packag
es (from pca) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (f
rom pca) (1.26.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-pack
ages (from pca) (1.3.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (f
rom pca) (1.13.1)
Requirement already satisfied: colourmap>=1.1.15 in /usr/local/lib/python3.10/dist
-packages (from pca) (1.1.17)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(from pca) (2.1.4)
Requirement already satisfied: scatterd>=1.3.7 in /usr/local/lib/python3.10/dist-p
ackages (from pca) (1.3.7)
Requirement already satisfied: adjusttext in /usr/local/lib/python3.10/dist-packag
es (from pca) (1.2.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages
(from scatterd>=1.3.7->pca) (0.13.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages
(from datazets->pca) (2.32.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->pca) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-pack
ages (from matplotlib->pca) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist
-packages (from matplotlib->pca) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist
-packages (from matplotlib->pca) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-p
ackages (from matplotlib->pca) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-pac
kages (from matplotlib->pca) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->pca) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib->pca) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-pack
ages (from pandas->pca) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-pa
ckages (from pandas->pca) (2024.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-pac
kages (from scikit-learn->pca) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/d
ist-packages (from scikit-learn->pca) (3.5.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-pack
ages (from statsmodels->pca) (0.5.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (fro
m patsy>=0.5.6->statsmodels->pca) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.
10/dist-packages (from requests->datazets->pca) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pack
ages (from requests->datazets->pca) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dis
t-packages (from requests->datazets->pca) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dis
t-packages (from requests->datazets->pca) (2024.8.30)
```

In [20]:
```
from pca import pca
```

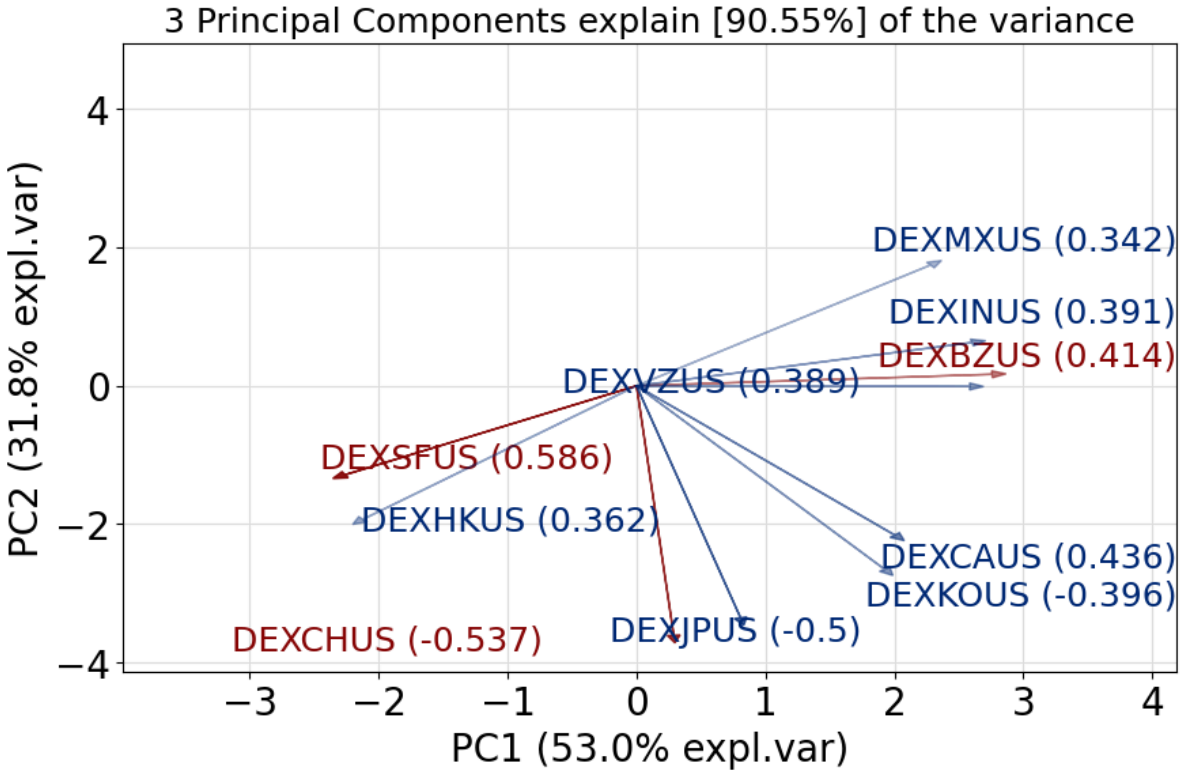
In [21]:
```
model = pca(n_components=3)

# Fit transform
results = model.fit_transform(rescaleddf)
```

```
[pca] >Extracting column labels from dataframe.
[pca] >Extracting row labels from dataframe.
[pca] >The PCA reduction is performed on the [10] columns of the input dataframe.
[pca] >Fit using PCA.
[pca] >Compute loadings and PCs.
[pca] >Compute explained variance.
[pca] >Outlier detection using Hotelling T2 test with alpha=[0.05] and n_component
s=[3]
[pca] >Multiple test correction applied for Hotelling T2 test: [fdr_bh]
[pca] >Outlier detection using SPE/DmodX with n_std=[3]
```

In [22]:
```
# Make biplot with the number of features
print("\033[1m" + "Biplot without the scores" + "\033[0m")
fig, ax = model.biplot(cmap=None, label=False, legend=False, figsize=(10, 6))
```

Biplot without the scores

```
[pca]> [WARNING]: De parameter <label> is deprecated and will not be supported in
future version.
[pca] >Plot PC1 vs PC2 with loadings.
[scatterd]> WARNING use the standardized verbose status. The status [1-6] will be
deprecated in future versions.
[scatterd] >INFO> Create scatterplot
```



In [22]:
```
```

Lasso Regression

In [36]:
```
# Import libraries
import numpy as np
```

```
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
```

Importing the data

In [37]:
```
df = yf.download(["AAPL","AMZN","CSCO","GOOGL","IBM","MSFT","NVDA"],start="2010-01-
df.head(3)
```

```
[**********************100%**********************]  7 of 7 completed
```

Out[37]:

Ticker	AAPL	AMZN	CSCO	GOOGL	IBM	MSFT	NVDA
Date							
2010-01-04 00:00:00+00:00	6.454504	6.6950	16.601274	15.645692	75.353798	23.347319	0.423952
2010-01-05 00:00:00+00:00	6.465663	6.7345	16.527321	15.576794	74.443512	23.354860	0.430143
2010-01-06 00:00:00+00:00	6.362819	6.6125	16.419737	15.184124	73.959923	23.211536	0.432894

In [38]:
```
# adjust date format
df["Date"] = pd.to_datetime(df.index)
df["Date"] = df["Date"].dt.date
df.set_index("Date",inplace=True)
df.head(3)
```

Out[38]:

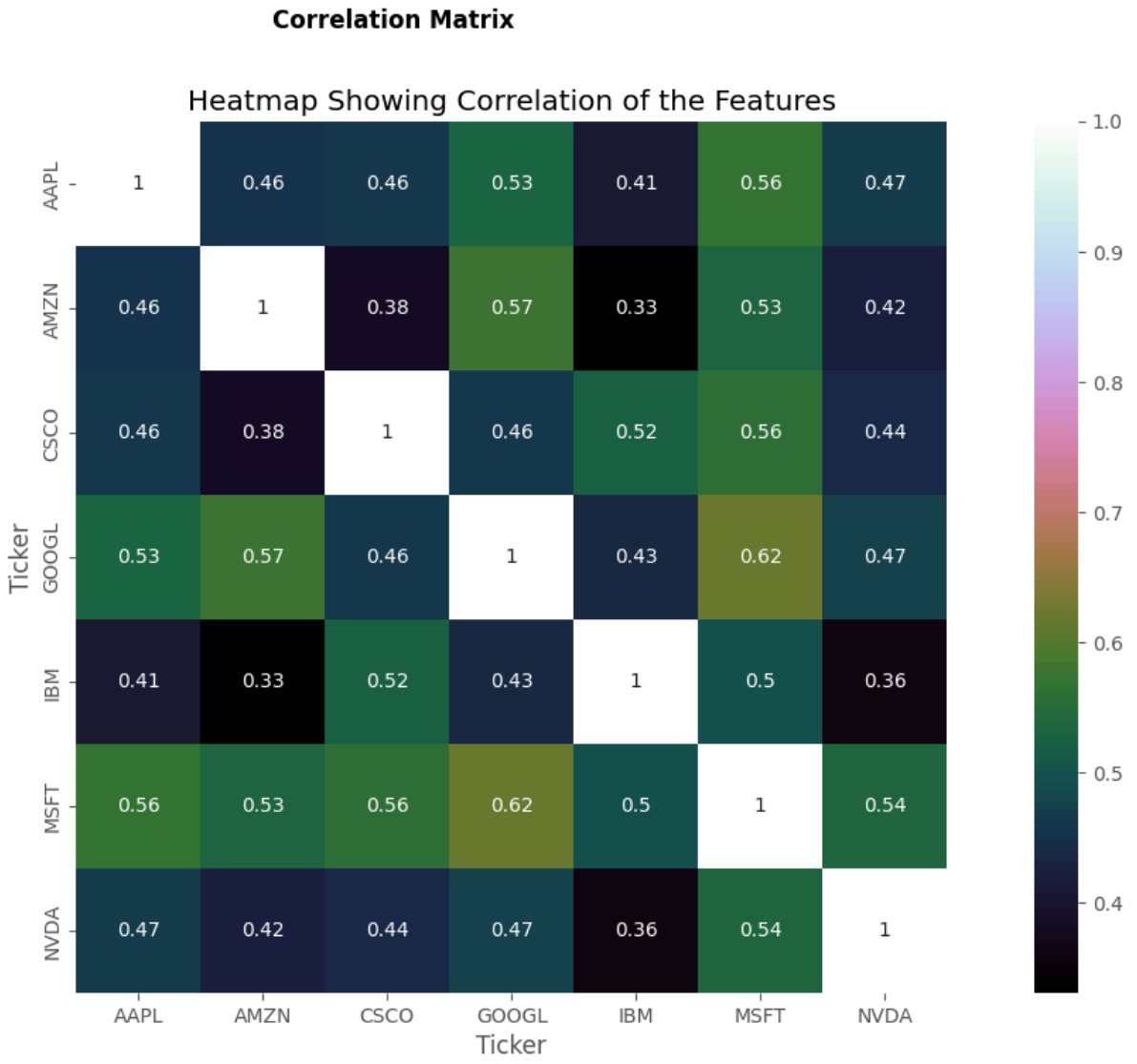
Ticker	AAPL	AMZN	CSCO	GOOGL	IBM	MSFT	NVDA
Date							
2010-01-04	6.454504	6.6950	16.601274	15.645692	75.353798	23.347319	0.423952
2010-01-05	6.465663	6.7345	16.527321	15.576794	74.443512	23.354860	0.430143
2010-01-06	6.362819	6.6125	16.419737	15.184124	73.959923	23.211536	0.432894

In [39]:
```
# compute returns
df = df.pct_change()
df = df.dropna()
df.tail()
```

Out[39]:

Ticker	AAPL	AMZN	CSCO	GOOGL	IBM	MSFT	NVDA
Date							
2021-12-23	0.003644	0.000184	0.012189	0.003425	0.006782	0.004472	0.008163
2021-12-27	0.022975	-0.008178	0.018304	0.006738	0.007579	0.023186	0.044028
2021-12-28	-0.005767	0.005844	0.001735	-0.008245	0.007674	-0.003504	-0.020133
2021-12-29	0.000502	-0.008555	0.006768	-0.000218	0.005429	0.002051	-0.010586
2021-12-30	-0.006578	-0.003289	-0.005316	-0.003099	0.004199	-0.007691	-0.013833

In [40]:
```
corr = df.corr()
plt.figure(figsize=(16, 8))
plt.title("Heatmap Showing Correlation of the Features")
plt.suptitle(
    "Correlation Matrix", fontweight="bold", horizontalalignment="center"
)
sns.heatmap(corr, vmax=1, square=True, annot=True, cmap="cubehelix");
```



In [41]:
```
# define target variable (NVDA) as y and features (independent variables) as X
X, y = df.iloc[:, 0:-1], df.iloc[:, -1]
print(X.shape, y.shape)
```

```
(3019, 6) (3019,)
```

In [42]:
```
# split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=int(len(y) * 0.5), shuffle=False
)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(1510, 6) (1509, 6) (1510,) (1509,)
```

In [43]:
```
print(X)
```

```
Ticker          AAPL      AMZN      CSCO     GOOGL       IBM      MSFT
Date
2010-01-05  0.001729  0.005900 -0.004455 -0.004404 -0.012080  0.000323
2010-01-06 -0.015906 -0.018116 -0.006509 -0.025209 -0.006496 -0.006137
2010-01-07 -0.001849 -0.017013  0.004505 -0.023280 -0.003461 -0.010400
2010-01-08  0.006649  0.027077  0.005299  0.013331  0.010035  0.006897
2010-01-11 -0.008822 -0.024041 -0.002839 -0.001512 -0.010470 -0.012720
...              ...       ...       ...       ...       ...       ...
2021-12-23  0.003644  0.000184  0.012189  0.003425  0.006782  0.004472
2021-12-27  0.022975 -0.008178  0.018304  0.006738  0.007579  0.023186
2021-12-28 -0.005767  0.005844  0.001735 -0.008245  0.007674 -0.003504
2021-12-29  0.000502 -0.008555  0.006768 -0.000218  0.005429  0.002051
2021-12-30 -0.006578 -0.003289 -0.005316 -0.003099  0.004199 -0.007691

[3019 rows x 6 columns]
```

In [44]:
```
# lasso regression
lasso = Lasso(alpha=0.0001)
lasso.fit(X_train, y_train)
y_pred_Lasso = lasso.predict(X_train)
y_pred_test_lasso = lasso.predict(X_test)
```

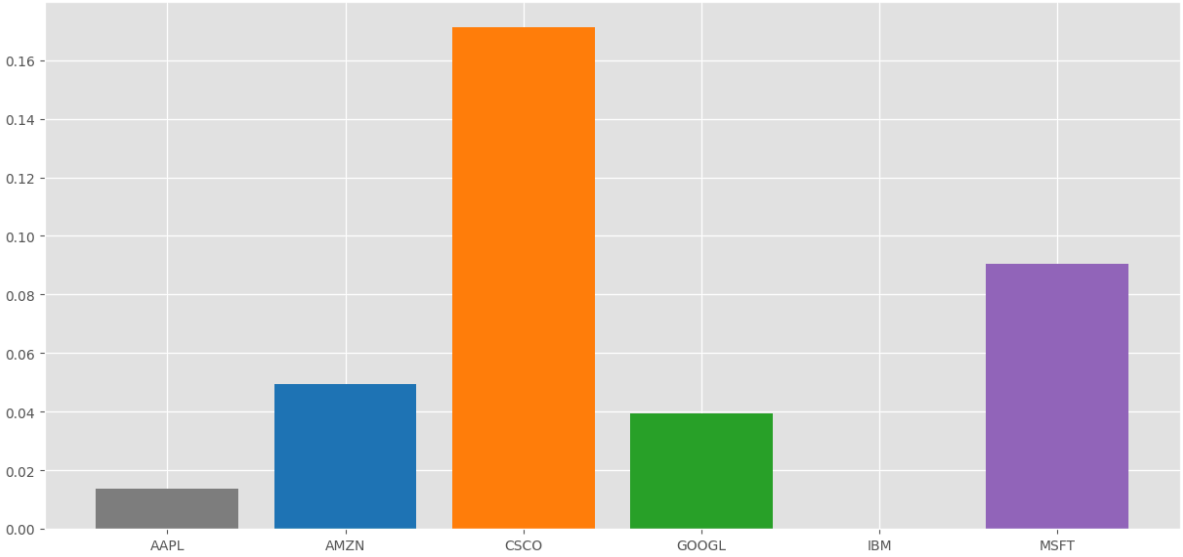
In [45]:
```
# Calculate Mean Squared Error
mean_squared_error = np.mean((y_pred_test_lasso - y_test) ** 2)
print("Mean squared error on test set", mean_squared_error)
```

```
Mean squared error on test set 0.0007216038266250512
```

In [46]:
```
lasso_coeff = pd.DataFrame()
lasso_coeff["Columns"] = X_train.columns
lasso_coeff["Coefficient Estimate"] = pd.Series(lasso.coef_)

print(lasso_coeff)
```

```
  Columns  Coefficient Estimate
0    AAPL              0.013826
1    AMZN              0.049321
2    CSCO              0.171237
3   GOOGL              0.039495
4     IBM              0.000000
5    MSFT              0.090649
```

In [47]:
```
# plotting the coefficient score
fig, ax = plt.subplots(figsize=(15, 7))

color = [
    "tab:gray",
    "tab:blue",
    "tab:orange",
    "tab:green",
    "tab:red",
    "tab:purple",
    "tab:brown",
]

ax.bar(
    lasso_coeff["Columns"], lasso_coeff["Coefficient Estimate"], color=color
)

plt.style.use("ggplot")
plt.show()
```



In [48]:
```
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error


# Assuming 5 folds
k = 5  # Number of folds
kf = KFold(n_splits=k, shuffle=True, random_state=42)

mse_scores = []
for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    lasso.fit(X_train, y_train)
    y_pred = lasso.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    mse_scores.append(mse)

average_mse = sum(mse_scores) / k
print("Average Mean Squared Error:", average_mse)
```

```
Average Mean Squared Error: 0.0005197931455025336
```