Week 3 - Laboratory

ECON441B

Mauricio Vargas-Estrada
Master in Quantitative Economics
University of California - Los Angeles

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from pytrends.request import TrendReq

# Imputer
from sklearn.impute import KNNImputer
from sklearn.model_selection import train_test_split

# Disable all warnings
import warnings
warnings.filterwarnings('ignore')
```

0.) Clean the Apple Data to get a quarterly series of EPS.

In [ ]:

```
data = pd.read_csv('AAPL_quarterly_financials.csv')
```

In [ ]:

```
# Transposing the data and setting the first row as the column names
data = data.T
data.columns = ['dates'] +  data.iloc[0,:]
data = data.drop(data.index[0:2,])
```

In [ ]:

```
# Converting data.index to datetime
data.index = pd.to_datetime(data.index)
```

In [ ]:

```
# Ordering the data from oldest to newest
data = data.sort_index()
```

In [ ]:

```
# Converting to numeric, but the numbers are strings with commas.
data = data.apply(lambda x: x.str.replace(',',''))
data = data.apply(pd.to_numeric)
```

In [ ]:

```
# Using an nearest neighbor imputer to fill in missing values.
# The 5 nearest neighbors are used to fill in the missing values.
imputer = KNNImputer(n_neighbors=5)
data = pd.DataFrame(
    imputer.fit_transform(data),
    columns=data.columns,
    index=data.index
)
```

In [ ]:

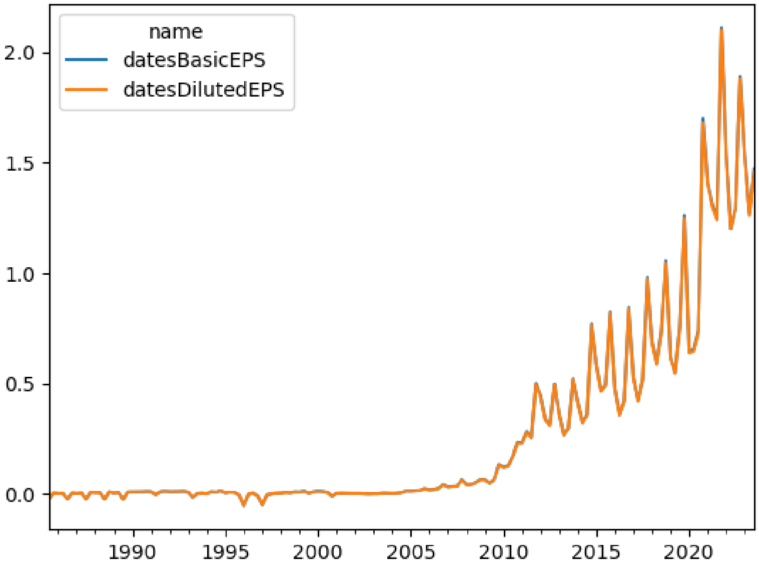
```
# Getting a DataFrame containing the columns that contain the word 'EPS'
eps_name = data.columns.str.contains('EPS')
eps = data.iloc[:,eps_name]
```

In [ ]:

```
eps.plot()
```

Out[ ]:

```
<Axes: >
```



1.) Come up with 6 search terms you think could nowcast earnings. (Different than the ones I used) Add in 3 terms that that you think will not Nowcast earnings. Pull in the gtrends data. Clean it to have a quarterly average.

In [ ]:

```
# Create pytrends object
pytrends = TrendReq(hl='en-US', tz=360)

# Set up the keywords and the timeframe
keywords = [
    'Apple',
    'Macintosh',
    'Apple Stocks',
    'Apple Financial',
    'Apple Earnings',
    'NASDAQ',
    'SP500',
    'GDP',
    'Technology',
    'Linear Regression',
    'Bayes Theorem',
    'UCLA'
]  # Add your keywords here
start_date = '2004-01-01'
end_date = '2024-01-01'

# Create an empty DataFrame to store the results
df = pd.DataFrame()

# Iterate through keywords and fetch data
for keyword in keywords:
    pytrends.build_payload([keyword], cat=0, timeframe=f'{start_date} {end_date}', geo='', gprop='')
    interest_over_time_df = pytrends.interest_over_time()
    df[keyword] = interest_over_time_df[keyword]
```

In [ ]:

```
df = df.resample('Q').sum()
```

In [ ]:

```
y = eps.iloc[:,0]
```

In [ ]:

```
# Subset y into the same time period as X
first_date = max(df.index[0], y.index[0])
last_date = min(df.index[-1], y.index[-1]) + pd.DateOffset(months=3)
```

In [ ]:

```
df = df.loc[first_date:last_date]
y = y.loc[first_date:last_date]
```

2.) Normalize all the X data

```
In [ ]: from sklearn.preprocessing import StandardScaler
```

```
In [ ]: scaler = StandardScaler()
```

```
In [ ]: X_scaled = scaler.fit_transform(df)
```

3.) Run a Lasso with lambda of .5. Plot a bar chart.

```
In [ ]: from sklearn.linear_model import Lasso
```

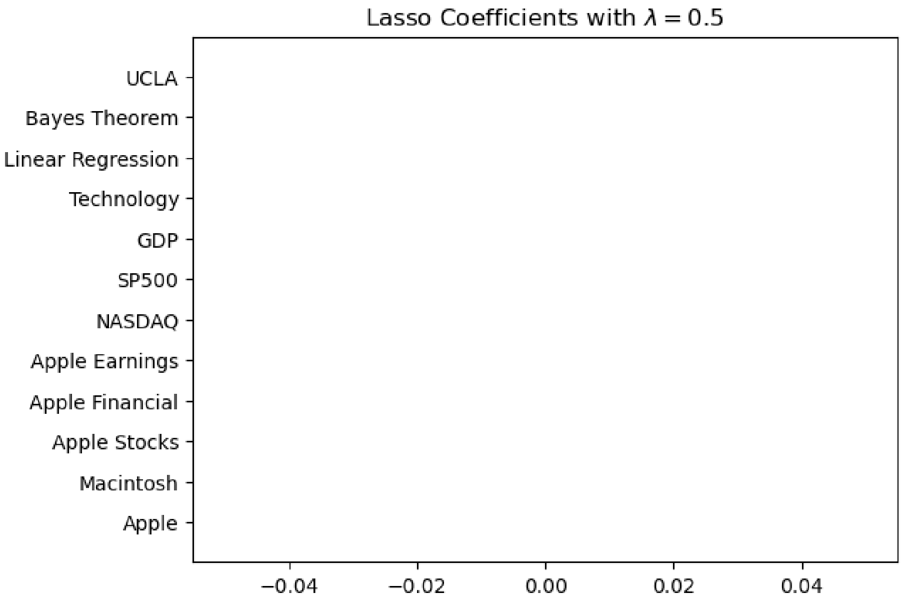
```
In [ ]: lasso = Lasso(alpha=0.5)
```

```
In [ ]: lasso_result = lasso.fit(X_scaled,y)
```

4.) Do these coefficient magnitudes make sense?

```
In [ ]: plt.barh(df.columns, lasso_result.coef_)
        plt.title('Lasso Coefficients with $\lambda = 0.5$')
        plt.plot()
```

```
Out[ ]: []
```



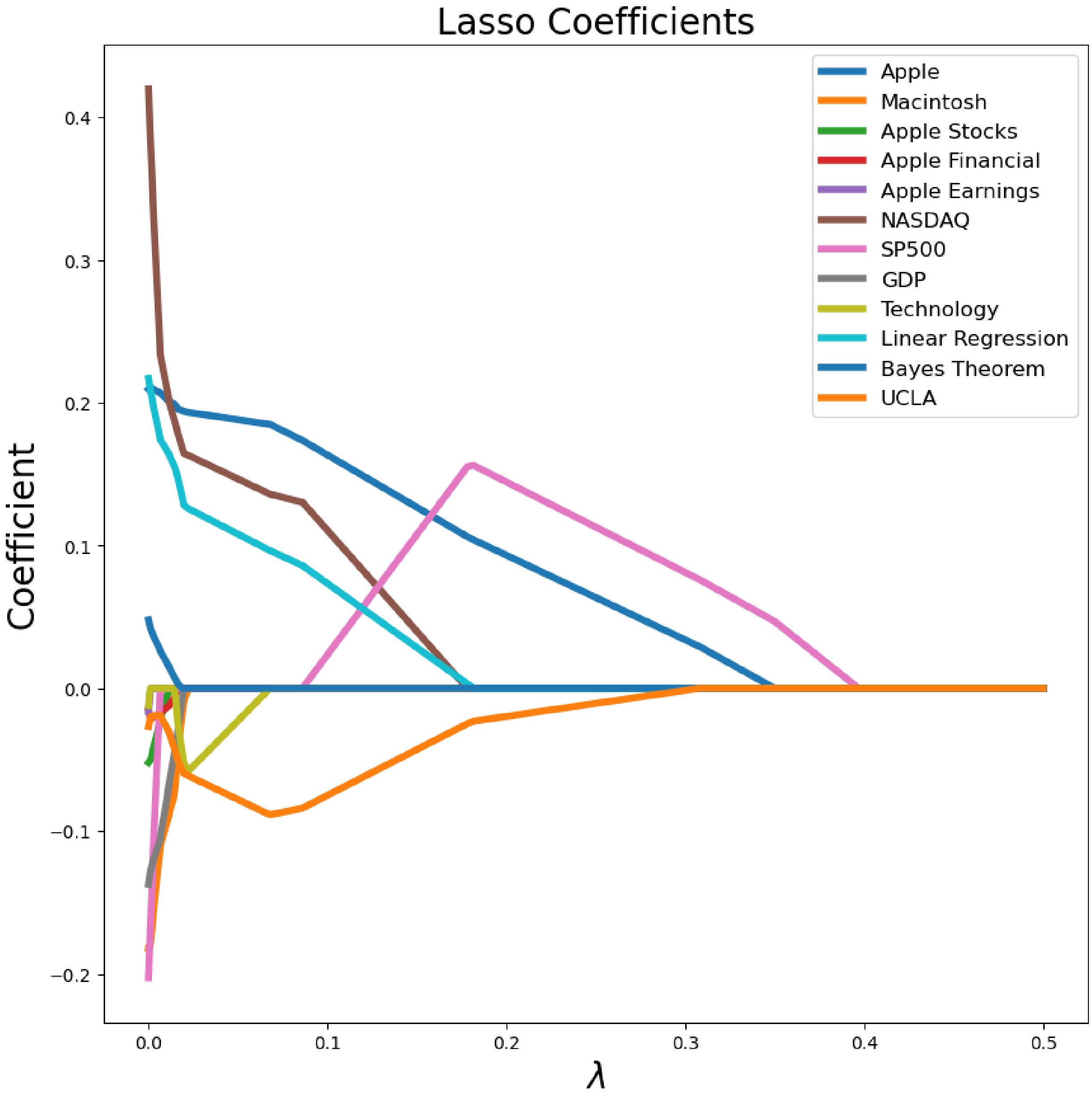
Using a $\lambda = 0.5$ all the coefficients are zero. Meaning that we are penalizing the model excessively.

5.) Run a for loop looking at 10 different Lambdas and plot the coefficient magnitude for each.

```
In [ ]: l_sim = np.linspace(0, 0.5, 1_000)
        coef = np.nan * np.ones((len(l_sim), X_scaled.shape[1]))
        for i, l in enumerate(l_sim):
            lasso = Lasso(alpha=l)
            lasso_result = lasso.fit(X_scaled,y)
            coef[i,:] = lasso_result.coef_
```

```
In [ ]: plt.figure(figsize=(10,10))
        for i in range(coef.shape[1]):
            plt.plot(
                l_sim, coef[:,i],
                label=df.columns[i],
                linewidth=4
            )
        plt.legend(
            df.columns,
            fontsize=12,
        )
        plt.title('Lasso Coefficients', fontsize=20)
        plt.xlabel('$\lambda$', fontsize=20)
        plt.ylabel('Coefficient', fontsize=20)
```

```
Out[ ]: Text(0, 0.5, 'Coefficient')
```



6.) Run a cross validation. What is your ideal lambda?

```
In [ ]: from sklearn.linear_model import LassoCV
```

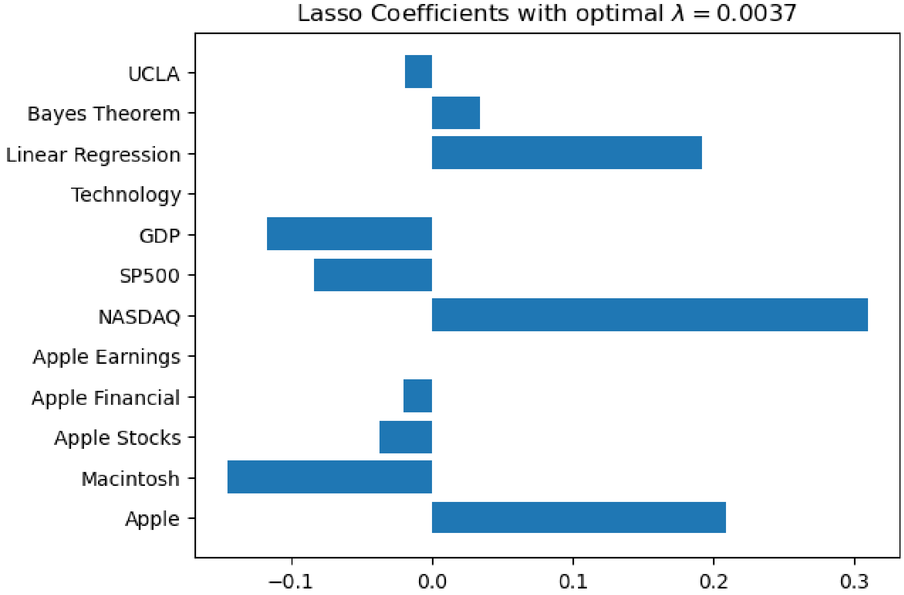
```
In [ ]: modCV = LassoCV(cv=5).fit(X_scaled,y)
```

```
In [ ]: opt_l = modCV.alpha_
        opt_coef = modCV.coef_
        print(f'Optimal lambda: {np.round(opt_l, 5)}')

        Optimal lambda: 0.0037
```

```
In [ ]: plt.barh(df.columns, opt_coef)
        plt.title(f'Lasso Coefficients with optimal $\lambda = {np.round(opt_l, 5)}$')
        plt.plot()
```

```
Out[ ]: []
```


Lasso Coefficients with optimal $\lambda = 0.0037$
UCLA
Bayes Theorem
Linear Regression
Technology
GDP
SP500
NASDAQ
Apple Earnings
Apple Financial
Apple Stocks
Macintosh
Apple
−0.1
0.0
0.1
0.2
0.3