SEC 1 Homework - 3

January 25, 2024

[104]:
```
import pandas as pd
```

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

[105]:
```
data = pd.read_csv('/Users/adrianonggowarsito/Desktop/AAPL_quarterly_financials.
  ↪csv', index_col=0, header=None).T
y = data.reindex(index=data.index[::-1])
y.set_index('name', inplace=True)
y.drop(y.tail(1).index,inplace=True)
y.index = pd.to_datetime(y.index)
Y = y['BasicEPS']
Y = Y.fillna(0.)
Y
```

[105]:
```
name
1985-09-30      0.0
1985-12-31    0.004
1986-03-31    0.002
1986-06-30    0.002
1986-09-30      0.0
              …
2022-09-30     1.29
2022-12-31     1.89
2023-03-31     1.53
2023-06-30     1.27
2023-09-30     1.47
Name: BasicEPS, Length: 153, dtype: object
```

[106]:
```
df
```

[106]:

	Phone	Sales	Macbook	Hack	OS	Tech	Food	Tokyo	Bali
date									
2004-01-01	79	100	0	97	71	60	62	15	34
2004-02-01	79	92	0	82	66	56	62	14	33
2004-03-01	75	93	0	85	70	56	62	15	34
2004-04-01	77	96	0	90	68	56	63	14	36
2004-05-01	78	94	0	76	68	49	63	15	36

```
…             …    …       …    …  ..   …    …    …    …
2023-09-01    80   52      79   40  77   39   93   19   77
2023-10-01    81   50      81   40  75   37   90   18   70
2023-11-01    75   54      90   35  72   42   92   18   71
2023-12-01    77   47      68   46  65   40   92   19   73
2024-01-01    82   51      83   49  66   40   91   21   85

[241 rows x 9 columns]
```

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

[107]:
```
from pytrends.request import TrendReq
```

[108]:
```
import time
```

[109]:
```
# Create pytrends object
pytrends = TrendReq(hl='en-US', tz=360)

# Set up the keywords and the timeframe
keywords = ['Phone', 'Sales', 'Macbook', 'Hack', 'OS', 'Tech', 'Food', 'Tokyo',␣
 ↪'Bali']  # 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:
    time.sleep(5)
    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]
```

[110]:
```
dfq = df.resample('Q').mean()
dfq = dfq[dfq.index <= '2023-09-30']
dfq
```

[110]:
```
                Phone      Sales    Macbook       Hack         OS       Tech  \
date
2004-03-31  77.666667  95.000000   0.000000  88.000000  69.000000  57.333333
```

```
2004-06-30  79.333333  96.000000   0.000000  83.666667  72.000000  52.000000
2004-09-30  83.333333  96.000000   0.000000  79.000000  69.000000  51.333333
2004-12-31  84.333333  86.000000   0.000000  80.000000  69.666667  50.666667
2005-03-31  83.000000  94.000000   0.000000  83.000000  66.666667  50.000000
…                 …          …          …          …          …          …
2022-09-30  93.000000  50.000000  83.000000  47.333333  83.666667  37.000000
2022-12-31  79.666667  49.000000  74.333333  41.000000  76.333333  39.000000
2023-03-31  79.666667  50.333333  76.666667  44.000000  71.666667  38.333333
2023-06-30  75.333333  52.333333  68.000000  43.333333  77.333333  34.333333
2023-09-30  82.333333  50.666667  76.666667  41.000000  72.333333  37.666667

                 Food      Tokyo       Bali
date
2004-03-31  62.000000  14.666667  33.666667
2004-06-30  59.666667  14.666667  36.000000
2004-09-30  53.000000  15.000000  41.333333
2004-12-31  61.000000  14.666667  35.333333
2005-03-31  62.666667  14.000000  37.666667
…                 …          …          …
2022-09-30  93.333333  15.333333  75.000000
2022-12-31  88.333333  15.333333  72.000000
2023-03-31  91.333333  19.666667  83.000000
2023-06-30  91.000000  18.333333  77.666667
2023-09-30  93.000000  17.666667  84.000000

[79 rows x 9 columns]
```

3 2.) Normalize all the X data

[111]:
```
from sklearn.preprocessing import StandardScaler
```

[112]:
```
scaler = StandardScaler()
```

[113]:
```
X_scaled = scaler.fit_transform(dfq)
```

4 3.) Import data. Train, Test, Holdout (80%,15%,5%)

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

[136]:
```
from sklearn.linear_model import Lasso
```

[137]:
```
Y_04 = Y[Y.index >= '2004-03-31']
```

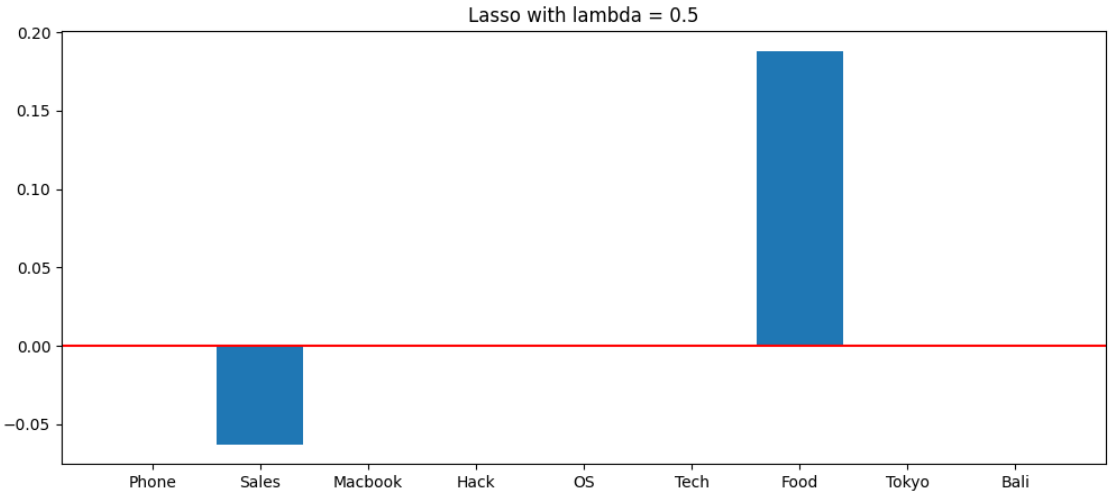
[138]:
```
lasso = Lasso(alpha = .2) #lower lambda since its penalizing to much
```

[139]:
```
lasso.fit(X_scaled, Y_04)
```

[139]:
```
Lasso(alpha=0.2)
```

[140]:
```
coefficients = lasso.coef_
```

[141]:
```
import matplotlib.pyplot as plt

names = ['Phone', 'Sales', 'Macbook', 'Hack', 'OS', 'Tech', 'Food', 'Tokyo',␣
 ↪'Bali']

plt.figure(figsize=(12,5))
plt.title('Lasso with lambda = 0.5')
plt.bar(range(len(coefficients)), coefficients, tick_label=names)
plt.axhline(0, color = 'red')
plt.show()
```



6 5.) Do these coefficient magnitudes make sense?

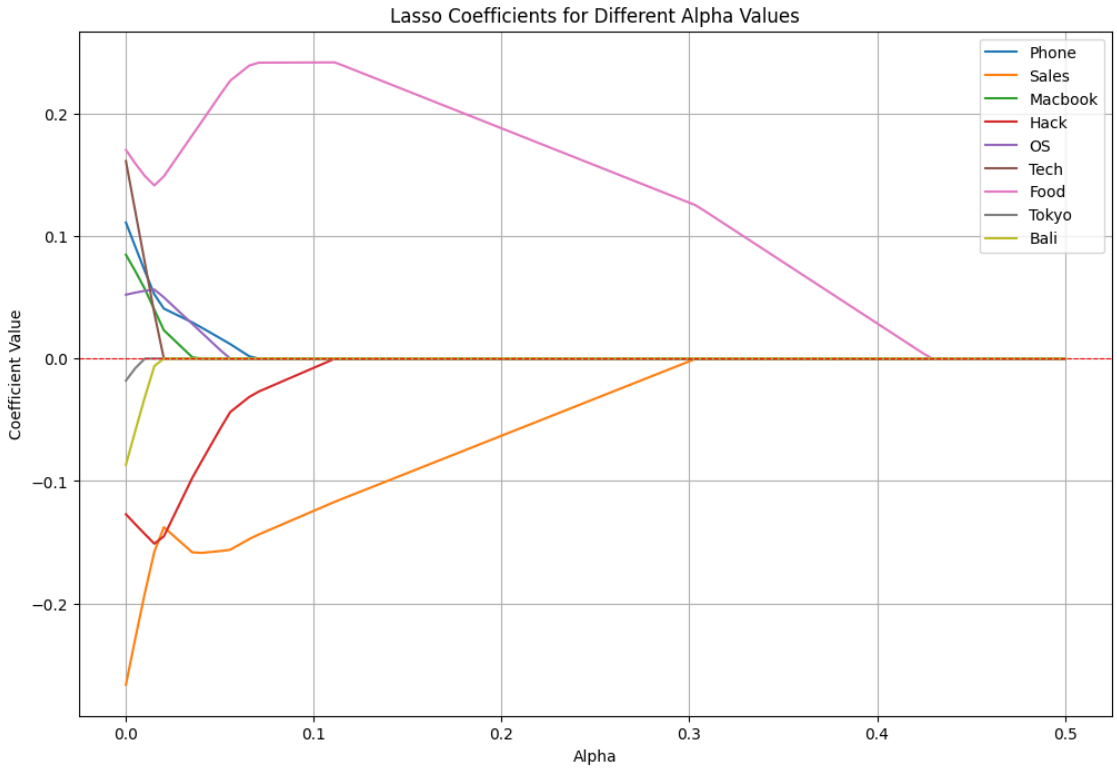
The coefficient does not really make sense. Much more relevant keywords like phone, macbook, os, and tech have 0 values, despite sales is affecting negatively, Food has pretty considerable outcome. Perhaps due to the amount of food bloggers using iPhones and/on social media

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

[143]:
```
import numpy as np

alphas = np.linspace(0, 0.5, 100)

# List to store coefficients for each alpha
coefficients_list = []

# Loop through different alpha values
for alpha in alphas:
    lasso = Lasso(alpha=alpha)
    lasso.fit(X_scaled, Y_04)
    coefficients = lasso.coef_
    coefficients_list.append(coefficients)

# Transpose the coefficients list for easier plotting
coefficients_array = np.array(coefficients_list).T

# Plotting
plt.figure(figsize=(12, 8))
for i in range(len(names)):
    plt.plot(alphas, coefficients_array[i], label=names[i])

plt.xlabel('Alpha')
plt.ylabel('Coefficient Value')
plt.title('Lasso Coefficients for Different Alpha Values')
plt.axhline(0, color='red', linestyle='--', linewidth=0.8)
plt.legend()
plt.grid(True)
plt.show()
```

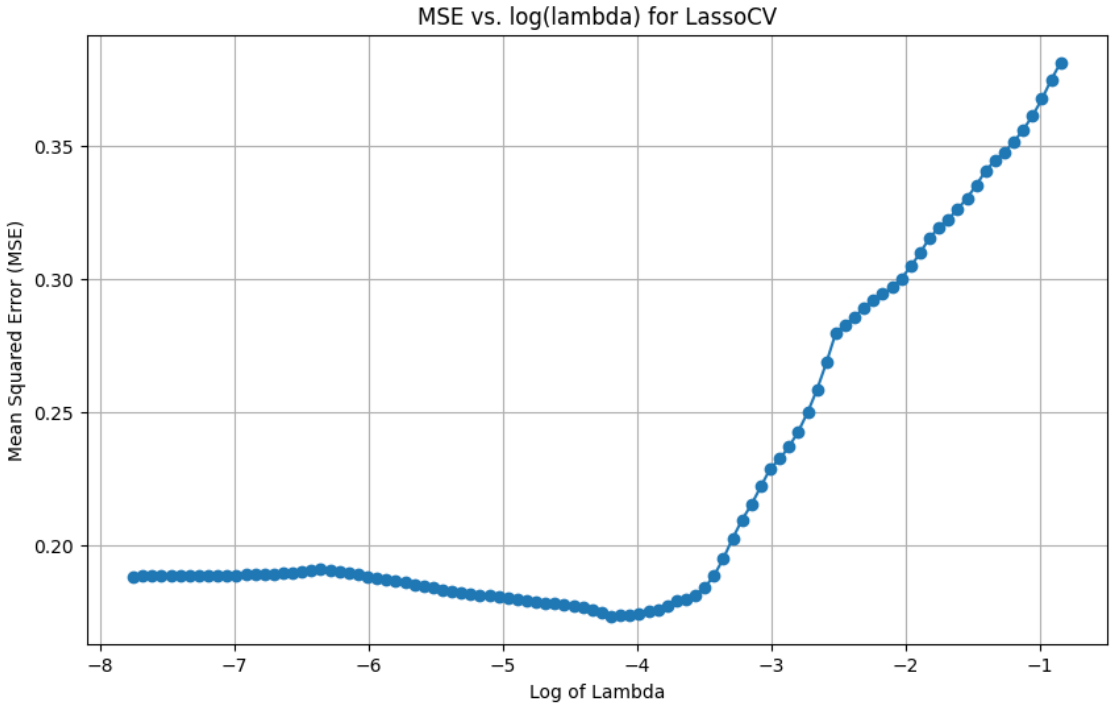
```
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-
packages/sklearn/base.py:1151: UserWarning: With alpha=0, this algorithm does
not converge well. You are advised to use the LinearRegression estimator
  return fit_method(estimator, *args, **kwargs)
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-
packages/sklearn/linear_model/_coordinate_descent.py:628: UserWarning:
Coordinate descent with no regularization may lead to unexpected results and is
discouraged.
  model = cd_fast.enet_coordinate_descent(
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-
packages/sklearn/linear_model/_coordinate_descent.py:628: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisation. Duality
gap: 1.758e+00, tolerance: 2.085e-03 Linear regression models with null weight
```

```
for the l1 regularization term are more efficiently fitted using one of the
solvers implemented in sklearn.linear_model.Ridge/RidgeCV instead.
  model = cd_fast.enet_coordinate_descent(
```



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

[144]:
```
from sklearn.linear_model import LassoCV
modCV = LassoCV().fit(X_scaled,Y_04)

alphas = modCV.alphas_
mse_values = modCV.mse_path_.mean(axis=1)

# Plotting
plt.figure(figsize=(10, 6))
plt.plot(np.log(alphas), mse_values, marker='o')
plt.xlabel('Log of Lambda')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('MSE vs. log(lambda) for LassoCV')
plt.grid(True)
plt.show()
```


MSE vs. log(lambda) for LassoCV
Mean Squared Error (MSE)
0.35
0.30
0.25
0.20
−8
−7
−6
−5
−4
−3
−2
−1
Log of Lambda