SEC 1 Homework - 3

January 26, 2024

[15]:
```python
import pandas as pd
```

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

[16]:
```python
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
```

[16]:
```
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
```

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.

[17]:
```python
from pytrends.request import TrendReq
```

[18]:
```
import time
```

[19]:
```
# 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]
```

[20]:
```
dfq = df.resample('Q').mean()
dfq = dfq[dfq.index <= '2023-09-30']
dfq
```

[20]:

date	Phone	Sales	Macbook	Hack	OS	Tech
2004-03-31	80.333333	96.666667	0.000000	94.333333	67.333333	55.000000
2004-06-30	80.333333	91.000000	0.000000	83.666667	69.000000	49.000000
2004-09-30	85.666667	93.000000	0.000000	80.333333	61.333333	48.333333
2004-12-31	82.666667	81.666667	0.000000	80.000000	65.000000	47.666667
2005-03-31	80.333333	87.666667	0.000000	85.333333	62.000000	45.666667
…	…	…	…	…	…	…
2022-09-30	91.000000	50.333333	83.000000	49.000000	80.333333	35.666667
2022-12-31	81.000000	48.333333	77.333333	42.000000	72.666667	36.333333
2023-03-31	76.333333	47.333333	76.000000	43.666667	65.000000	34.666667
2023-06-30	75.333333	49.000000	68.666667	43.333333	72.333333	31.666667
2023-09-30	79.333333	50.666667	78.000000	42.333333	65.000000	34.000000

\

date	Food	Tokyo	Bali
2004-03-31	65.666667	14.666667	34.000000
2004-06-30	59.333333	14.666667	37.333333
2004-09-30	54.333333	14.000000	44.000000
2004-12-31	61.000000	14.333333	35.000000
2005-03-31	62.000000	13.666667	37.000000

```
…                 …          …          …
2022-09-30  97.333333  14.666667  76.000000
2022-12-31  90.666667  14.333333  71.666667
2023-03-31  88.666667  17.000000  80.666667
2023-06-30  92.666667  17.000000  79.333333
2023-09-30  95.333333  17.000000  81.333333

[79 rows x 9 columns]
```

3 2.) Normalize all the X data

[21]:
```python
from sklearn.preprocessing import StandardScaler
```

[22]:
```python
scaler = StandardScaler()
```

[23]:
```python
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.

[24]:
```python
from sklearn.linear_model import Lasso
```

[25]:
```python
Y_04 = Y[Y.index >= '2004-03-31']
```

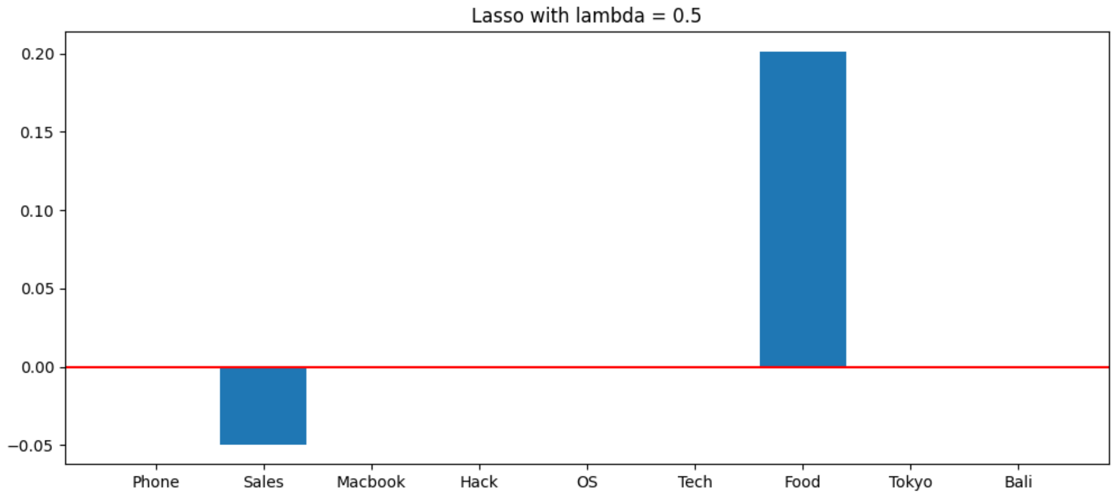
[26]:
```python
lasso = Lasso(alpha = .2) #lower lambda since its penalizing to much
```

[27]:
```python
lasso.fit(X_scaled, Y_04)
```

[27]:
```
Lasso(alpha=0.2)
```

[28]:
```python
coefficients = lasso.coef_
```

[29]:
```python
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?

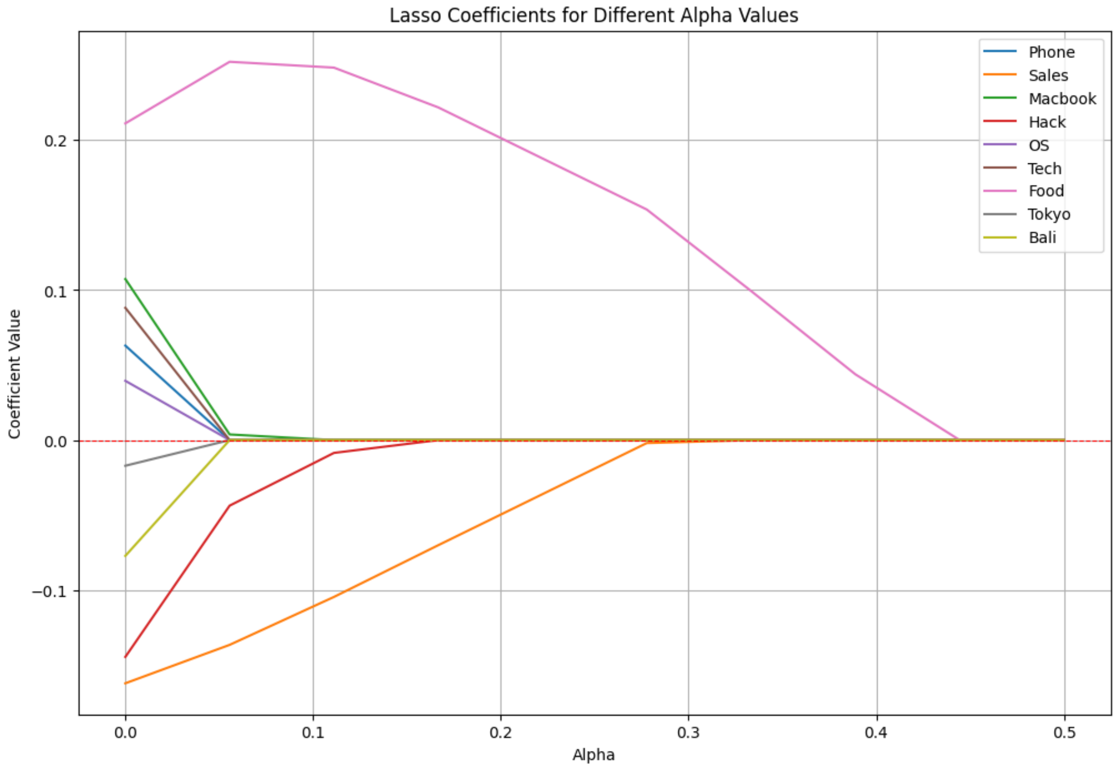
The coefficient does not really make sense. Much more relevant keywords like phone, macbook, os, and tech have 0 values, despite sales is affecting negatively, Food has pretty considerable outcome. Perhaps due to the amount of food bloggers using iPhones and/on social media

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

[39]:
```
import numpy as np

alphas = np.linspace(0, 0.5, 10)

# 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))
```

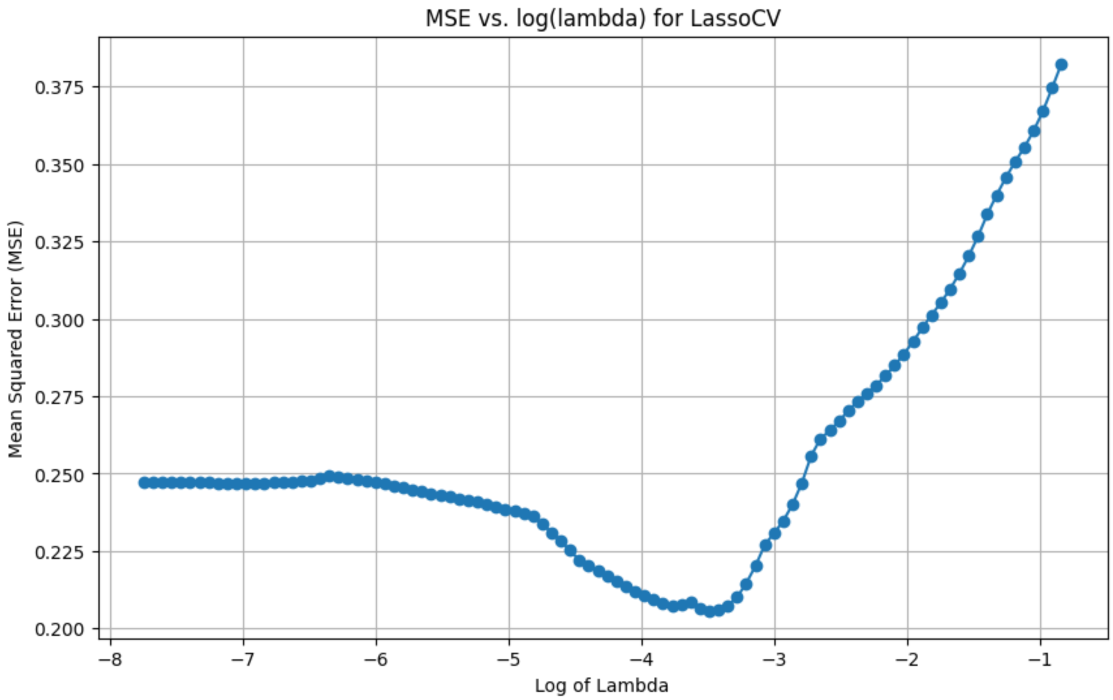
```python
for i in range(len(names)):
    plt.plot(alphas, coefficients_array[i], label=names[i])

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: 1.945e+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?

[38]:
```
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()
```



8.0.1 Approximately between 3 and 4 (3.5)