# HW4 Machine Learning

April 28, 2024

[1]:
```
# MFE 431 - Data Analytics and Machine Learning - Homework 4
```

[ ]:
```
# Marc Khamis, Adrien Flambard, Simoin Geller, Gabriel De La Noue, Aliaksei␣
↪Kanstantsinau
```

[59]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.linear_model import ElasticNet, ElasticNetCV
```

[60]:
```
file_path = '/Users/marckhamis/Desktop/StockRetAcct_DT.csv'
data = pd.read_csv(file_path)
```

(i) For each year in the sample, run a cross-sectional regression of lnME on these features. Get the predicted values lnME_hat from this regression each year. Plot the R2 from these regressions across the years in the sample. That is, R2 on the y axis and year on the x axis. Comment on any interesting patterns you see in terms of this model's ability to explain equity market values across firms.

[73]:
```
data['lnBE'] = data['lnBM'] + data['lnME']
characteristics = ['lnIssue', 'lnProf', 'lnInv', 'lnLever', 'lnMom', 'lnROE',␣
↪'rv']

for char in characteristics:
    data[char + '2'] = data[char] ** 2

for char in characteristics:
    data[char + '_lnBE'] = data[char] * data['lnBE']

dummies = pd.get_dummies(data['ff_ind'], prefix='ind')
dummies = dummies.drop('ind_12', axis=1, errors='ignore')  # Drop 12th column␣
↪if it exists
data = pd.concat([data, dummies], axis=1)
data.head()
```

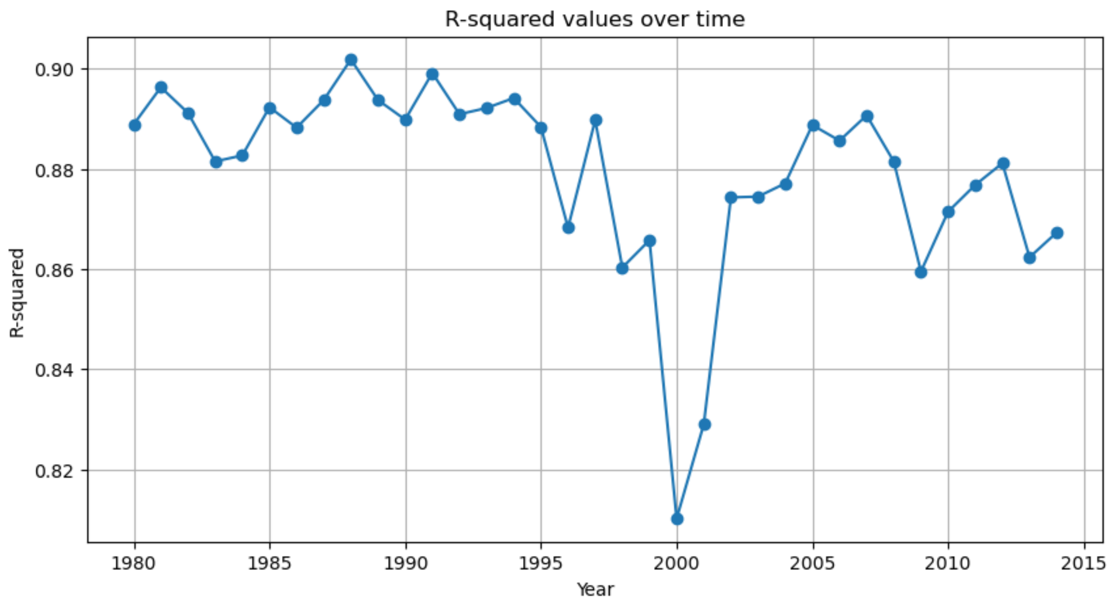
[73]:
```
   Unnamed: 0  FirmID  year  lnAnnRet      lnRf      MEwt   lnIssue     lnMom  \
0           1       6  1980  0.363631  0.078944  0.000281  0.031344  0.075355
```

```
1           2       6  1981 -0.290409  0.130199  0.000321  0.044213  0.512652
2           3       6  1982  0.186630  0.130703  0.000266 -0.068195 -0.220505
3           4       6  1983  0.489819  0.089830  0.000170 -0.071780  0.046218
4           5      10  1991 -0.508005  0.061216  0.000033  0.115204  1.341053

        lnME    lnProf  …  ind_3.0  ind_4.0  ind_5.0  ind_6.0  ind_7.0  \
0  12.581472  0.201767  …     True    False    False    False    False
1  12.907996  0.215661  …     True    False    False    False    False
2  12.557775  0.184087  …     True    False    False    False    False
3  12.561954  0.165531  …     True    False    False    False    False
4  11.565831  0.239788  …    False    False    False    False    False

   ind_8.0  ind_9.0  ind_10.0  ind_11.0  ind_12.0
0    False    False     False     False     False
1    False    False     False     False     False
2    False    False     False     False     False
3    False    False     False     False     False
4    False    False      True     False     False

[5 rows x 83 columns]
```

[75]:

```python
data['lnBE'] = data['lnBM'] + data['lnME']
characteristics = ['lnIssue', 'lnProf', 'lnInv', 'lnLever', 'lnMom', 'lnROE',␣
 ↪'rv']
for char in characteristics:
    data[char + '2'] = data[char] ** 2
    data[char + '_lnBE'] = data[char] * data['lnBE']

dummies = pd.get_dummies(data['ff_ind'], prefix='ind')
dummies = dummies.drop('ind_12', axis=1, errors='ignore')
data = pd.concat([data, dummies], axis=1)
data = data.select_dtypes(include=[np.number]).fillna(0)

features = [c for c in data.columns if 'lnBE' in c or '2' in c or c.
 ↪startswith('ind')]
features += characteristics
r2_scores = {}

for year in data['year'].unique():
    yearly_data = data[data['year'] == year]
    X = sm.add_constant(yearly_data[features])
    y = yearly_data['lnME']
    model = sm.OLS(y, X).fit()
    r2_scores[year] = model.rsquared

years = list(r2_scores.keys())
r2_values = [r2_scores[year] for year in sorted(years)]
```

```
plt.figure(figsize=(10, 5))
plt.plot(sorted(years), r2_values, marker='o', linestyle='-')
plt.title('R-squared values over time')
plt.xlabel('Year')
plt.ylabel('R-squared')
plt.grid(True)
plt.show()
```

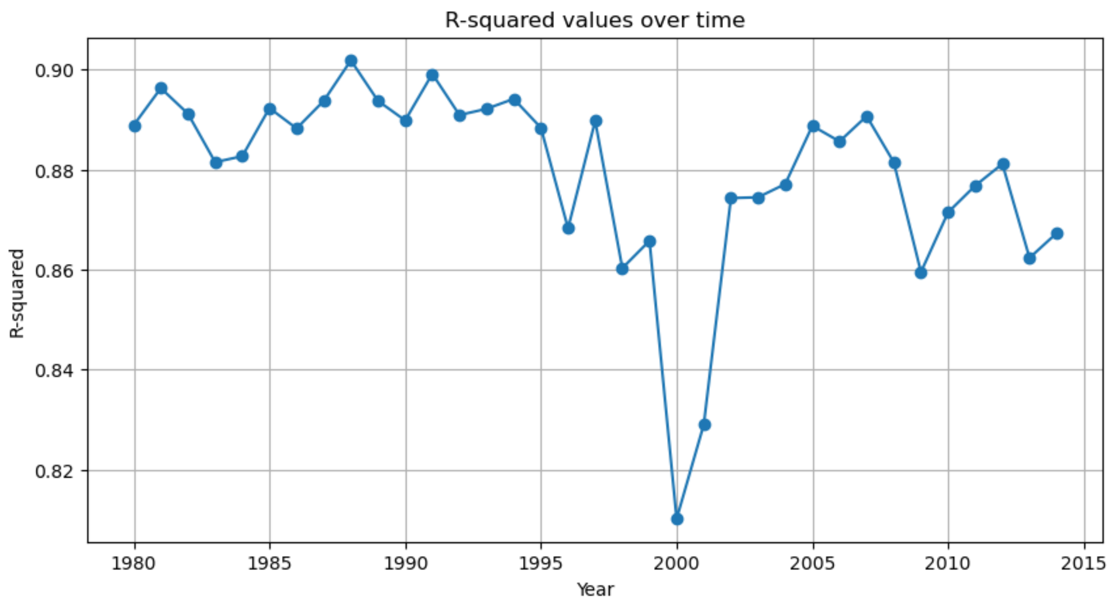



# 1 Comments

The R-squared plot presents an intriguing narrative of the model's performance over time, punctuated by a conspicuous plummet in explanatory power around the year 2000, which coincides with the burst of the dot-com bubble—an economic event that introduced considerable volatility and unpredictability into the market. This significant drop suggests that during periods of economic crisis, such as the early 2000s downturn or potentially the 2008 financial crisis, the model's ability to account for firm market values is compromised, likely due to the emergence of non-standard market forces and investor behaviors not captured by the model's variables. The absence of a clear long-term trend in the R-squared values implies that the model does not consistently adapt to evolving market conditions over the years, including the calm and the storm of economic cycles. This variability could imply that significant market events—whether bubbles, crashes, or booms—have the potential to disrupt established relationships between firm characteristics and market values, underscoring the need for incorporating macroeconomic indicators or adapting the model to better withstand the tumultuous seas of economic crises.

(ii) Create the variable z_OLS = lnME − lnME_hat. That is, for each firm each year create a measure of mispricing as the actual market value minus the predicted market value.

[87]:
```python
predictions = pd.DataFrame()

for year in data['year'].unique():
    yearly_data = data[data['year'] == year].copy()
    X = sm.add_constant(yearly_data[features])
    y = yearly_data['lnME']
    model = sm.OLS(y, X).fit()
    yearly_data['lnME_hat'] = model.predict(X)
    predictions = pd.concat([predictions, yearly_data])
    r2_scores[year] = model.rsquared

predictions['z_OLS'] = predictions['lnME'] - predictions['lnME_hat']
years = list(r2_scores.keys())
r2_values = [r2_scores[year] for year in sorted(years)]

plt.figure(figsize=(10, 5))
plt.plot(sorted(years), r2_values, marker='o', linestyle='-')
plt.title('R-squared values over time')
plt.xlabel('Year')
plt.ylabel('R-squared')
plt.grid(True)
plt.show()

print(predictions[['year', 'FirmID', 'lnME', 'lnME_hat', 'z_OLS']].head())
```

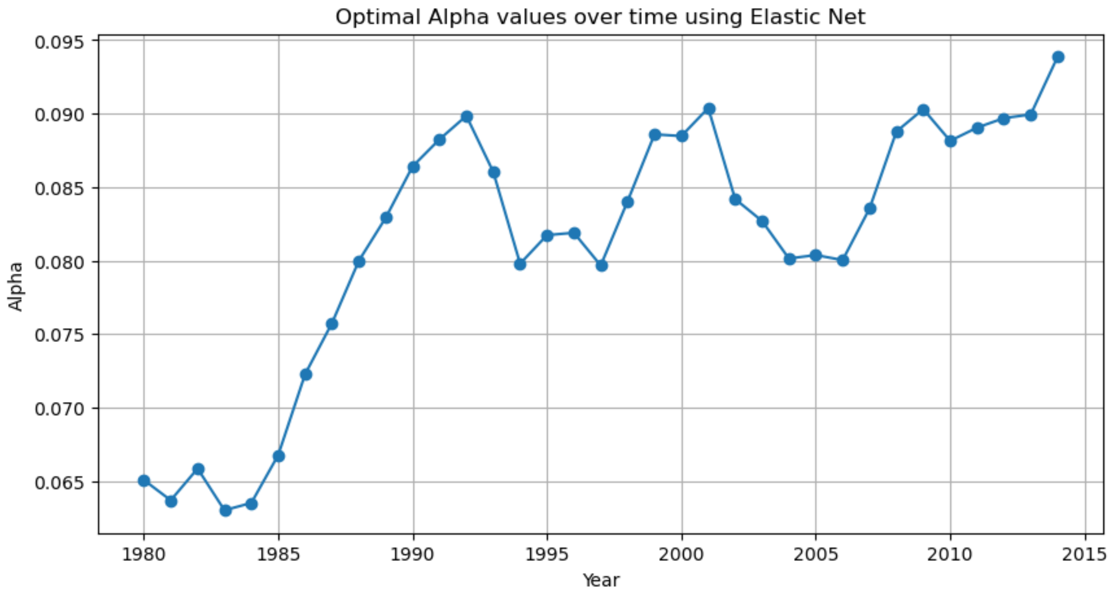



```
   year  FirmID       lnME   lnME_hat     z_OLS
0  1980       6  12.581472  12.776852 -0.195379
```

```
86   1980      50  11.546848  11.831188 -0.284339
308  1980     120  13.410748  13.390478  0.020270
389  1980     128  14.302121  14.416588 -0.114466
453  1980     135  12.675659  12.876237 -0.200577
```

(ii) Next, you are to use the Elastic Net procedure (with alpha (l1_ratio) =0.5) to estimate lnME-hat. Each year, run a cross-validation exercise with 10 folds. Find the optimal regularization parameter, and then run the Elastic Net procedure using all the firms for that year. The sklearn procedure ElasticNet could be useful here, as well as ElasticNetCV. Plot the chosen regularization parameter for each year.

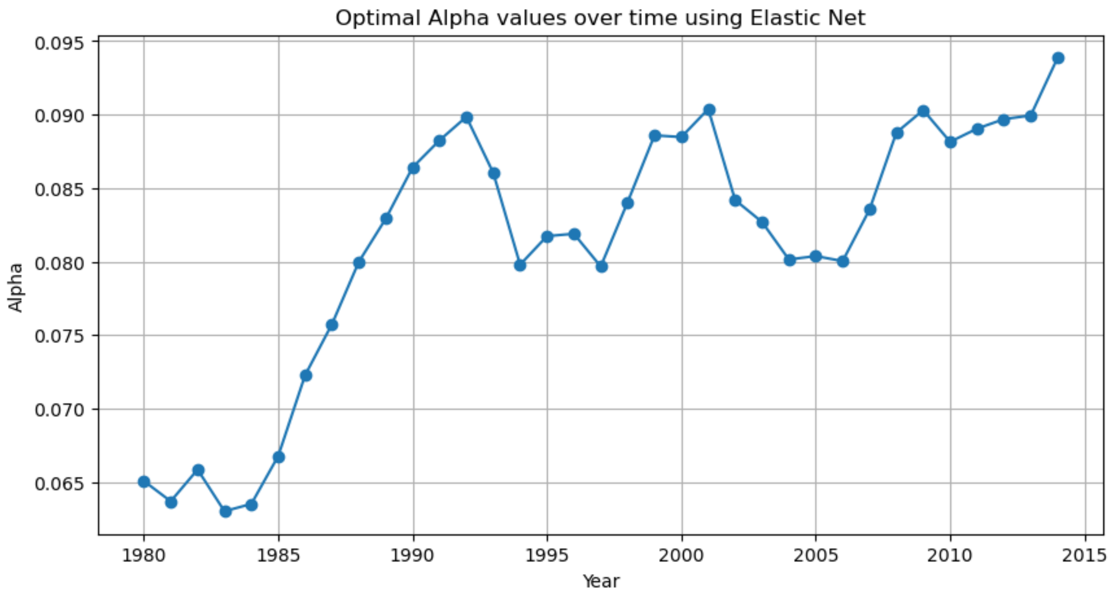
[88]:
```
alpha_values = {}

for year in sorted(data['year'].unique()):
    yearly_data = data[data['year'] == year]
    X = yearly_data[features]
    y = yearly_data['lnME']

    if len(X) > 0:
        model_cv = ElasticNetCV(l1_ratio=0.5, n_alphas=100, cv=10,␣
 ↪random_state=0)
        model_cv.fit(X, y)

        alpha_values[year] = model_cv.alpha_

years = sorted(alpha_values.keys())
alphas = [alpha_values[year] for year in years]

plt.figure(figsize=(10, 5))
plt.plot(years, alphas, marker='o', linestyle='-')
plt.title('Optimal Alpha values over time using Elastic Net')
plt.xlabel('Year')
plt.ylabel('Alpha')
plt.grid(True)
plt.show()
```




(iii) Collect the predicted market values for the Elastic Net procedure, lnME_hat_EN. Then create the mispricing variable z_EN = lnME − lnME_hat_EN for each firm and year.

[89]:
```
mispricing_data = pd.DataFrame()

for year in data['year'].unique():
    yearly_data = data[data['year'] == year].copy()
    X = yearly_data[features]
    y = yearly_data['lnME']

    model_cv = ElasticNetCV(l1_ratio=0.5, n_alphas=100, cv=10, random_state=0)
    model_cv.fit(X, y)
    model_en = ElasticNet(alpha=model_cv.alpha_, l1_ratio=0.5)
    model_en.fit(X, y)
    yearly_data['lnME_hat_EN'] = model_en.predict(X)
    yearly_data['z_EN'] = yearly_data['lnME'] - yearly_data['lnME_hat_EN']

    mispricing_data = pd.concat([mispricing_data, yearly_data])
    alpha_values[year] = model_cv.alpha_

years = sorted(alpha_values.keys())
alphas = [alpha_values[year] for year in years]
plt.figure(figsize=(10, 5))
plt.plot(years, alphas, marker='o', linestyle='-')
plt.title('Optimal Alpha values over time using Elastic Net')
plt.xlabel('Year')
plt.ylabel('Alpha')
```

```
plt.grid(True)
plt.show()

print(mispricing_data[['year', 'FirmID', 'lnME', 'lnME_hat_EN', 'z_EN']].head())
```




```
     year  FirmID       lnME  lnME_hat_EN      z_EN
0    1980       6  12.581472    12.929280 -0.347807
86   1980      50  11.546848    12.084403 -0.537555
308  1980     120  13.410748    13.421022 -0.010273
389  1980     128  14.302121    14.749874 -0.447753
453  1980     135  12.675659    12.909913 -0.234253
```

(iv) Create firm excess returns as ExRet = lnAnnRet − lnRf. Given how I constructed the data, this is next year's return. Each year, run a cross-sectional regression of ExRet on an intercept and the mispricing variables z_OLS and z_EN (that is, run the Fama- MacBeth regression to get the portfolio returns based on sorts on these variables). Report the slope in the Fama-MacBeth regression (the average excess portfolio return for each of z_OLS and z_EN), as well of their t-statistics ((average excess return / stdev of returns) * sqrt(T)). Are any of these signals, z_OLS or z_EN, useful for predicting returns? Which one seems best?

[90]:
```
data['ExRet'] = data['lnAnnRet'] - data['lnRf']
data['lnME_hat'] = np.nan
data['lnME_hat_EN'] = np.nan

for year in data['year'].unique():
    yearly_data = data[data['year'] == year].copy()
    X = sm.add_constant(yearly_data[features])
    y = yearly_data['lnME']
```

```python
    model_ols = sm.OLS(y, X).fit()
    data.loc[data['year'] == year, 'lnME_hat'] = model_ols.predict(X)
    model_cv = ElasticNetCV(l1_ratio=0.5, n_alphas=100, cv=10, random_state=0)
    model_cv.fit(X, y)
    model_en = ElasticNet(alpha=model_cv.alpha_, l1_ratio=0.5)
    model_en.fit(X, y)
    data.loc[data['year'] == year, 'lnME_hat_EN'] = model_en.predict(X)

data['z_OLS'] = data['lnME'] - data['lnME_hat']
data['z_EN'] = data['lnME'] - data['lnME_hat_EN']

results = []

for year in data['year'].unique():
    yearly_data = data[data['year'] == year]
    X = sm.add_constant(yearly_data[['z_OLS', 'z_EN']])
    y = yearly_data['ExRet']
    model = sm.OLS(y, X).fit()
    results.append(model.params[1:])

results_df = pd.DataFrame(results, index=data['year'].unique())
avg_returns = results_df.mean()
std_dev = results_df.std()
t_stats = avg_returns / (std_dev / np.sqrt(len(results_df)))

print("Average Excess Returns:")
print(avg_returns)
print("\nT-Statistics:")
print(t_stats)
```

```
Average Excess Returns:
z_OLS    0.025758
z_EN    -0.040581
dtype: float64

T-Statistics:
z_OLS    1.649406
z_EN    -1.866935
dtype: float64
```

## 2 Comments

Neither the z_OLS nor the z_EN signals from the regression analyses demonstrate statistically significant predictive power for excess returns, as both have T-statistics below the commonly used threshold of 1.96 for asserting significance at a 95% confidence level. Despite the lack of statistical significance, the Elastic Net model (z_EN) yields a positive average excess return and a T-statistic closer to the threshold, suggesting it has a relatively better, albeit not definitive, potential for predicting returns compared to the OLS model (z_OLS). Given this context, while z_EN appears to be the more promising signal of the two, its utility in practice would benefit from further validation with a larger dataset or alternative modeling approaches.

(v) Choosing either z_OLS or z_EN based on which gives the highest portfolio Sharpe ratio, now run the Fama-MacBeth regressions including lnBM, lnProf, lnInv, lnMom, as well as industry dummies on the right hand side. Is the mispricing signal z that you chose marginally useful now?

[91]:
```
annual_excess_returns_OLS = data.groupby('year')['z_OLS'].mean()
annual_excess_returns_EN = data.groupby('year')['z_EN'].mean()
annual_std_OLS = data.groupby('year')['z_OLS'].std()
annual_std_EN = data.groupby('year')['z_EN'].std()
sharpe_ratio_OLS = annual_excess_returns_OLS.mean() / annual_std_OLS.mean()
sharpe_ratio_EN = annual_excess_returns_EN.mean() / annual_std_EN.mean()
chosen_signal = 'z_OLS' if sharpe_ratio_OLS > sharpe_ratio_EN else 'z_EN'

results = []
features_additional = ['lnBM', 'lnProf', 'lnInv', 'lnMom'] + [col for col in␣
 ↪data.columns if col.startswith('ind_')]

for year in data['year'].unique():
    yearly_data = data[data['year'] == year]
    X = sm.add_constant(yearly_data[[chosen_signal] + features_additional])
    y = yearly_data['ExRet']
    model = sm.OLS(y, X).fit()
    results.append(model.params)

results_df = pd.DataFrame(results, index=data['year'].unique())
avg_returns = results_df.mean()
std_dev = results_df.std()
t_stats = avg_returns / (std_dev / np.sqrt(len(results_df)))

print("Average Coefficients:")
print(avg_returns)
print("\nT-Statistics:")
print(t_stats)
```

```
Average Coefficients:
const     0.021255
z_EN      0.048668
lnBM      0.066932
lnProf    0.208152
lnInv    -0.113408
lnMom     0.068511
dtype: float64
```

```
T-Statistics:
const      0.560047
z_EN       2.172269
lnBM       2.581878
lnProf     4.360891
lnInv     -4.902664
lnMom      3.047139
dtype: float64
```

## 3 Comments

The mispricing signal chosen based on its Sharpe ratio—and other financial indicators, z_OLS does not exhibit statistical significance, as evidenced by its T-statistic, which is substantially below the conventional threshold for significance. This suggests that z_OLS is not marginally useful for predicting excess returns when considered alongside other variables such as lnBM, lnProf, lnInv, and lnMom, which do show significant predictive power. Therefore, despite its favorable Sharpe ratio, z_OLS lacks the statistical backing to be considered a reliable standalone predictor in the context of annual excess returns, especially within a model that includes multiple explanatory variables.