SP500_Index_Research

December 25, 2019

1 Index Research Based on Shiller Dataset

1.1 Purpose and Background

Purpose: evaluate the predictive power of major price and economic indicators on S&P 500 returns
Dataset source: http://www.econ.yale.edu/~shisller/data.htm

- Named as: FinancialMarketData.xlsx

1.2 Import Packages, Import & Explore Data

[59]:
```
# %% packages
# can use "conda info --envs" to check default environment in terminal

import pandas as pd
import matplotlib
import matplotlib.pyplot as plt

# for datetime plotting
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

# more plotting
import seaborn as sns

# machine learning & numpy
import numpy as np
from sklearn import linear_model # for predictive model
from sklearn.model_selection import train_test_split #for splitting
from yellowbrick.regressor import ResidualsPlot

# XGBoost

# file export
import openpyxl
```

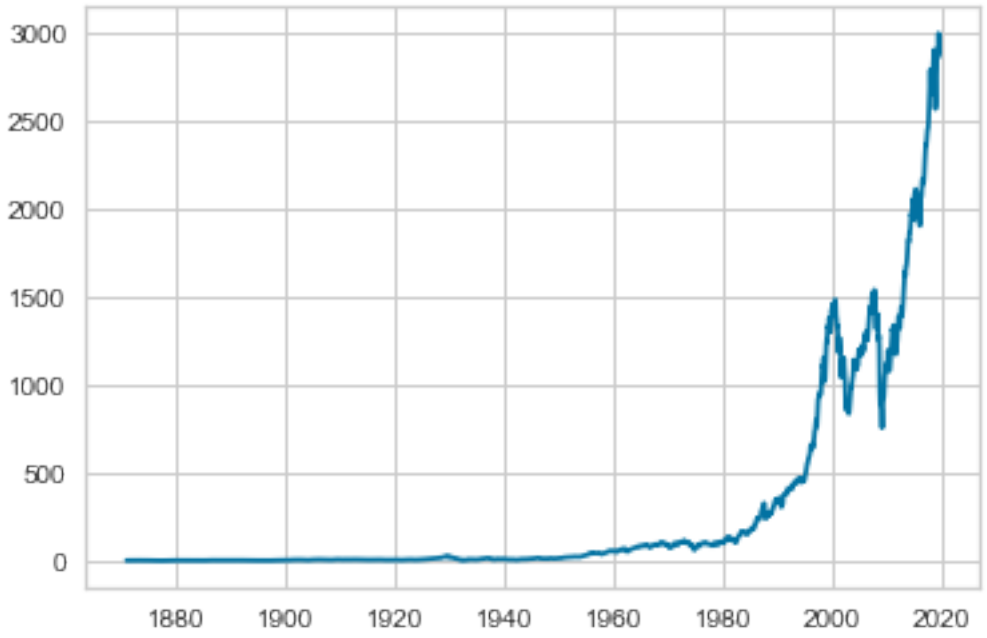
[60]:
```
# %% import & explore data
df = pd.read_excel('SP500_Index_Research_FinancialMarketData.xlsx',
```

```
                    sheet_name = "YearlyMacro")
print(df.dtypes)
```

```
DateFmt                 float64
Price                   float64
Dividend                float64
Earnings                float64
CPI                     float64
DateFraction            float64
RateGS10                float64
RealSP500               float64
RealDividend            float64
RealEarnings            float64
CAPE                    float64
Source                   object
LastUpdated      datetime64[ns]
Date             datetime64[ns]
dtype: object
```

[61]:
```
# plot data
# fix for using IPython/Spyder
# %matplotlib qt5
fig, ax  = plt.subplots()
ax.plot(df['DateFraction'],df['Price'])

# ax1 = ax.twinx()
# ax1.plot(summary['DGS10'], color = 'r')
```

[61]: [<matplotlib.lines.Line2D at 0x2cf7e5cc630>]



1.3 Hypothesis A: Predict S&P 500 10-yr Price Based on Macro Factors

Let: + y = target price + e = adjusted earnings + r = 10-yr treasury rate + g = growth rate
Hypothesize that returns follow: $y = e/r * (1 + g)$
This is based on dividend discount model (assuming dividends are like earnings)
Now, create the necessary variables to test this:

[62]:
```
# %% create columns to complete analysis

# future returns (dependent variable)
df['SP500FwdYr01'] = df['Price'].shift(-12) #next year
df['SP500FwdYr01Returns']  = df['SP500FwdYr01'] /df['Price'] - 1

df['SP500FwdYr10'] = df['Price'].shift(-120) #next 10 yrs
df['SP500FwdYr10Returns']  = (df['SP500FwdYr10'] /df['Price'])**(.1) - 1

# pick independent variables
df['InflationTrailing5yrFactor'] = df['CPI'] / df['CPI'].shift(60) #check␣
 ↪previous 5 years
df['PERatio'] = df['Price'] / df['Earnings']
df['Earnings10yr'] = df['Earnings'].rolling(window = 120).mean()  # mean of␣
 ↪last 10 yrs
df['Earnings10yrGrowthRate'] = (df['Earnings10yr']/df['Earnings10yr'].
 ↪shift(120))**(.1)-1 # last 10 yrs
# TODO try different growth rate
df['Earnings10yrAdj'] = df['Earnings10yr'] *␣
 ↪(1+df['Earnings10yrGrowthRate'])**5 \
    * df['InflationTrailing5yrFactor']
df['PERatio10yrAdj'] =  df['Price'] / df['Earnings10yrAdj']

# such as target price
df['Targetprice1yr']  = df['Earnings'] /(df['RateGS10']/100)
df['Targetprice1yrReturn'] = df['Targetprice1yr'] /df['Price']  -1
df['Targetprice10yr']  = df['Earnings10yrAdj'] /(df['RateGS10']/100) \
    * (1+ df['Earnings10yrGrowthRate'])**10 # includes growth
df['Targetprice10yrReturn'] = (df['Targetprice10yr'] /df['Price'])**(.1)  -1

#trim file
df = df.dropna()

#check results
print(df.head())
```

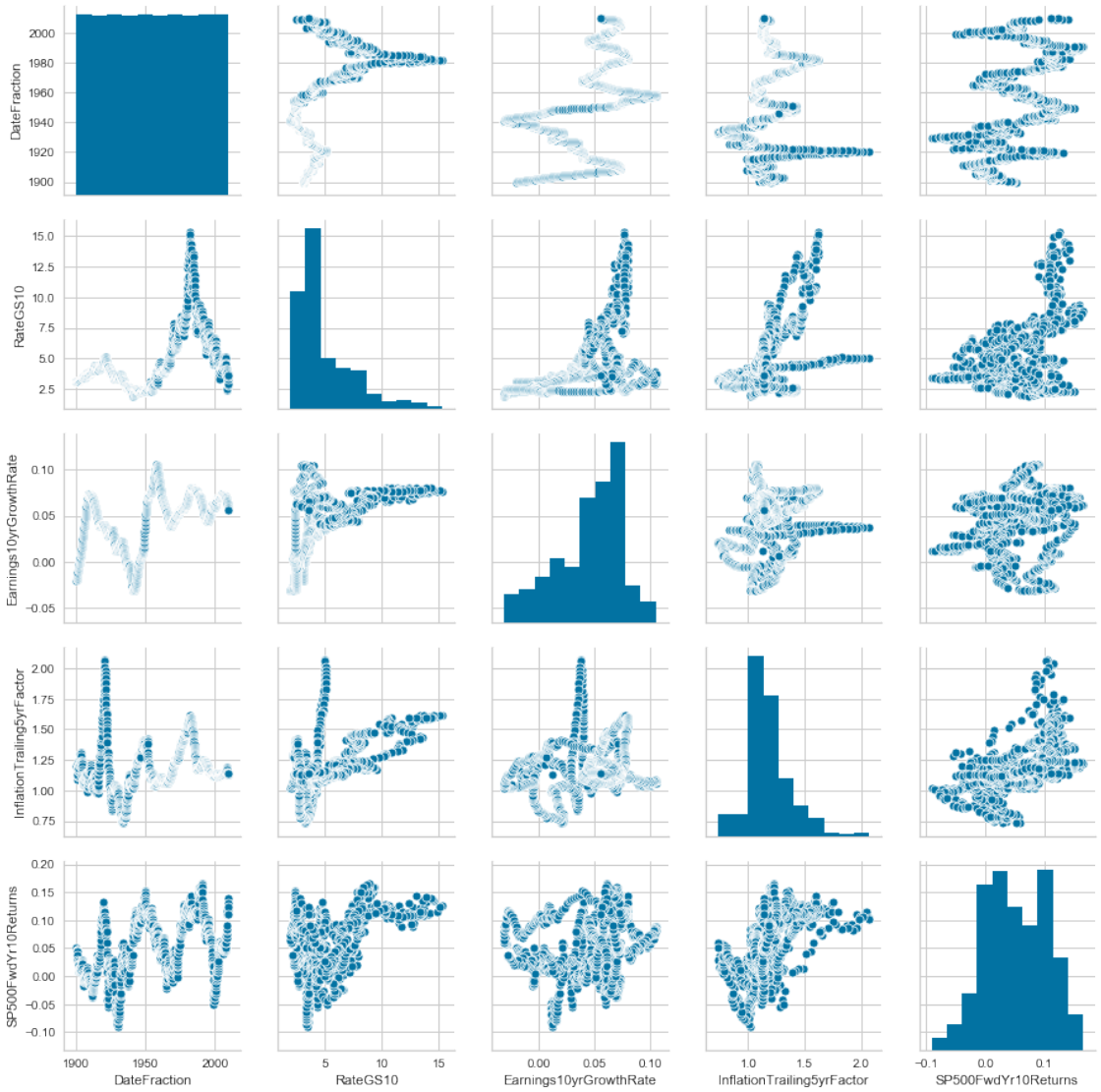
```
     DateFmt  Price  Dividend  Earnings       CPI  DateFraction  RateGS10  \
348  1900.01   6.10    0.2175      0.48  7.897091   1900.041667  3.150000
```

```
349  1900.02   6.21    0.2250      0.48  7.992232   1900.125000  3.145833
350  1900.03   6.26    0.2325      0.48  7.992232   1900.208333  3.141667
351  1900.04   6.34    0.2400      0.48  7.992232   1900.291667  3.137500
352  1900.05   6.04    0.2475      0.48  7.801942   1900.375000  3.133333

      RealSP500  RealDividend  RealEarnings  ...  InflationTrailing5yrFactor  \
348  197.883703      7.055689     15.571177  ...                       1.202898
349  199.053975      7.212101     15.385814  ...                       1.217390
350  200.656664      7.452504     15.385814  ...                       1.217390
351  203.220967      7.692907     15.385814  ...                       1.166667
352  198.326872      8.126805     15.761076  ...                       1.123287

       PERatio Earnings10yr Earnings10yrGrowthRate  Earnings10yrAdj  \
348  12.708333     0.295257               -0.020151         0.320793
349  12.937500     0.296771               -0.019424         0.327535
350  13.041667     0.298292               -0.018673         0.330476
351  13.208333     0.299819               -0.017899         0.319586
352  12.583333     0.301354               -0.017100         0.310538

     PERatio10yrAdj  Targetprice1yr  Targetprice1yrReturn  Targetprice10yr  \
348       19.015406       15.238095              1.498048         8.308159
349       18.959794       15.258278              1.457050         8.557286
350       18.942379       15.278515              1.440657         8.711989
351       19.838157       15.298805              1.413061         8.502910
352       19.450120       15.319149              1.536283         8.340674

     Targetprice10yrReturn
348               0.031377
349               0.032582
350               0.033604
351               0.029788
352               0.032800

[5 rows x 28 columns]
```

1.3.1 Plot Result Using This Formula

[63]:
```
# %% 4 plot data
sns.pairplot(df[['DateFraction', 'RateGS10','Earnings10yrGrowthRate',
                 'InflationTrailing5yrFactor' ,
                 'SP500FwdYr10Returns']])
```

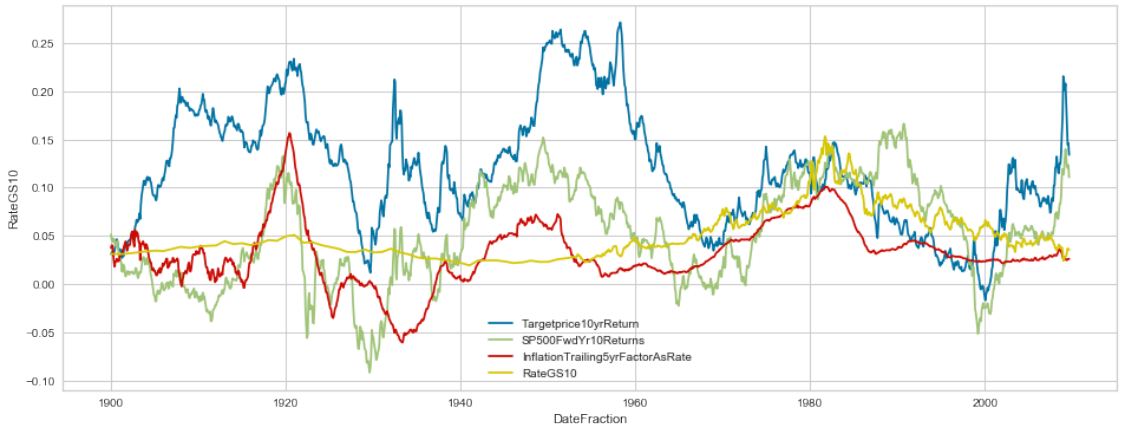
[63]:
```
<seaborn.axisgrid.PairGrid at 0x2cf7e18add8>
```



There are many relationships here, but nothing that clearly jumps out

[64]:
```
plt.figure(figsize=(16,6))
#fig3, ax3  = plt.subplots(figsize=(16, 6))
sns.lineplot(df['DateFraction'],df['Targetprice10yrReturn'], color = 'b')
#ax3 = plt.twinx()
sns.lineplot(df['DateFraction'],df['SP500FwdYr10Returns'], color = 'g')
sns.lineplot(df['DateFraction'],df['InflationTrailing5yrFactor']**.2-1, color =␣
↪'r')
sns.lineplot(df['DateFraction'],df['RateGS10']/100, color = 'y')
plt.legend(labels=['Targetprice10yrReturn','SP500FwdYr10Returns',
                   'InflationTrailing5yrFactorAsRate', 'RateGS10' ])
```

[64]:
```
<matplotlib.legend.Legend at 0x2cf7e3f4080>
```



The blue estimated line is not a good fit for the green actual returns as there is a large spread between the the blue and green lines

1.3.2 Export Data for Potential Further Analysis

[65]:
```
# export data
df.to_excel('SP500_Index_Research_Results.xlsx', sheet_name = 'sheet1',)
```

1.4 Hypothesis B: Use Linear Model to Predict Future 10-yr Factors

[66]:
```
#prep data
X_all = df.loc[:,['DateFraction','PERatio10yrAdj',
            'InflationTrailing5yrFactor', 'RateGS10']]

selected = ['PERatio10yrAdj',
            'InflationTrailing5yrFactor', 'RateGS10']

X_all_selected = X_all[selected]

y_all = df.loc[:,'SP500FwdYr10Returns']

X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, test_size=0.
 ↪33, random_state=42)

X_train_selected = X_train[selected]
X_test_selected = X_test[selected]
```

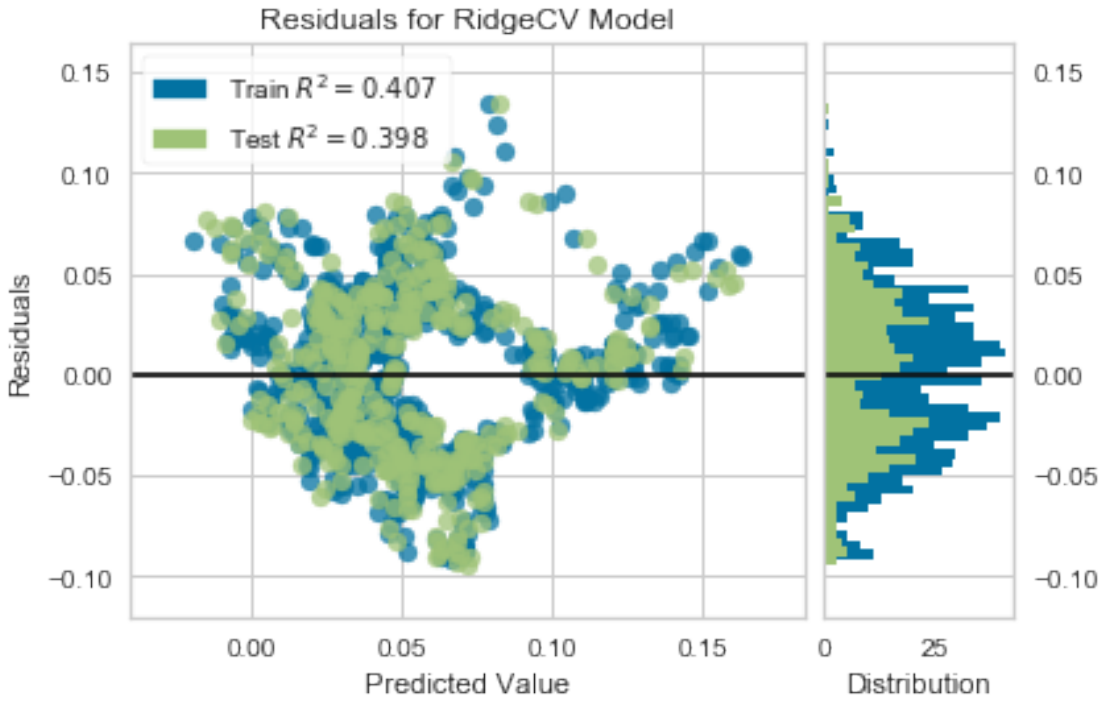
[67]:
```
#train classifier
clf = linear_model.RidgeCV(alphas=[1e-3, 1e-2, 1e-1, 1]).fit(X_train_selected,␣
 ↪y_train)

params = clf.get_params(deep=True)
```

```
#evaluate on test
print("default (R^2) score:" + np.array2string(clf.
 ↪score(X_test_selected,y_test)))
print("intercept: " + np.array2string(clf.intercept_))
print("coefficients: " + np.array2string(clf.coef_))
```

```
default (R^2) score:0.39750241
intercept: -0.03185082
coefficients: [-0.00306141  0.09323249  0.00269951]
```

[68]:
```
#predict using classifier
y_pred_B = clf.predict(X_all_selected)
```

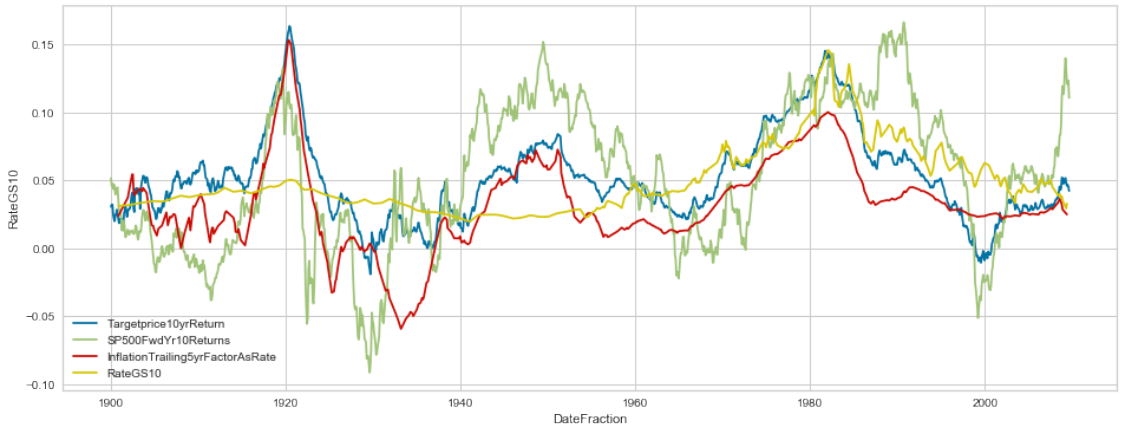
[69]:
```
# residual plots
visualizer = ResidualsPlot(clf)
visualizer.fit(X_train_selected, y_train)  # Fit the training data to the␣
 ↪visualizer
visualizer.score(X_test_selected, y_test)  # Evaluate the model on the test␣
 ↪data
visualizer.poof()                 # Draw/show/poof the data
```



[69]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2cf7e42c2b0>
```

[70]:
```
#plot all results, including train & test
plt.figure(figsize=(16,6))
#fig3, ax3  = plt.subplots(figsize=(16, 6))

sns.lineplot(X_all['DateFraction'],y_pred_B, color = 'b')
#ax3 = plt.twinx()
sns.lineplot(X_all['DateFraction'],y_all, color = 'g')
sns.lineplot(X_test['DateFraction'],X_all['InflationTrailing5yrFactor']**.2-1,␣
 ↪color = 'r')
sns.lineplot(X_test['DateFraction'],X_all['RateGS10']/100, color = 'y')
plt.legend(labels=['Targetprice10yrReturn','SP500FwdYr10Returns',
                   'InflationTrailing5yrFactorAsRate', 'RateGS10' ])
```

[70]:
```
<matplotlib.legend.Legend at 0x2cf00be1908>
```



There is still a (smaller) spread between green and blue lines

1.5 Conclusion

Both hypotheses (A: simple relationship using macro factors such as earnings growth, rates and inflation and B: linear model) do not sufficiently explain rolling 10-year market returns

1.6 Future Work

Hyphothesis C: Use XGBoost for Non-Linear Relationships

Incorporate Corporate bonds