In [1]:
```
# Menu
# 1. Load data and manipulate data
        #1.1 Melt date Column
        #1.2 Change date to to_datetime and set index

#2. Calculate weighted YTM
        #2.1 Calculate daily average YTM weighted by market cap.
        #2.2 Plot YTM
        #2.3 Group by YTM by different rating.
#3. Load and manipulate Macro data
        #3.1 Change date to to_datetime and set index
        #3.2 Merge all macro data and Average YTM into one big dataframe.
#4. Run regression
        #4.1 Perform linear regression of YTM and each factor.
        #4.2 Summary statistics (level of significance )
        #4.3 Plot resule
        #4.4 Conclusion for results.
#5. Perform PCA and split into train sample and test sample
        #5.1 Find cross correlation of macro factors.
        #5.2 Define insample data (prior to 2019). Out of sample data (2020)
        #5.3 Standardize all factors (x-mean)/sd
        #5.4 Perform linear regression of YTM to all factors.
        #5.5 Perform PCA
        #5.6 Use result to predict both train and target data
        #5.7 Plot the result
        #5.8 Compute RMSE and R^2
        #5.9 Run correlation between actual targets and predicted targets
        #5.10 Conclusion
```

1. Load data and manipulate data

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import itertools
import statsmodels.api as sm
from sklearn import linear_model
from sklearn.decomposition import PCA
from IPython.display import display, HTML


price_bond=pd.read_csv("price.csv", index_col = False)
usgdp=pd.read_csv("us_gdp.csv", index_col = False)
china_ir=pd.read_csv("China_ir.csv", index_col = False)
china_gdp=pd.read_csv("China_gdp.csv", index_col = False)
ytm_bond=pd.read_csv("ytm.csv", index_col = False)
df_ir_gold_fx=pd.read_csv("IR_GOLD_FX.csv", index_col = False)
```

1.1 Melt date columns

In [3]:
```
price_bond_melt = pd.melt(price_bond,
                          id_vars=['Issuer Name', 'Ticker', 'Cpn', 'Maturity', 'Series', 'BBG Composite', 'Maturity Type',
       'Curr', 'Maturity (Years from Today)', 'Amount Issued', 'ISIN'],
                          var_name='date',
                          value_name='price')
price_bond_melt.head()
```

Out[3]:

	Issuer Name	Ticker	Cpn	Maturity	Series	BBG Composite	Maturity Type	Curr	Maturity (Years from Today)	Amount Issued	ISIN	date	price
0	China Construction Bank Corp	CCB	2.45	6/24/30	NaN	BBB+	CALLABLE	USD	9.965777	2000000000	XS2140531950	5/17/13	NaN
1	Tencent Holdings Ltd	TENCNT	2.39	6/3/30	REGS	A+	CALLABLE	USD	9.908282	2250000000	US88032XAU81	5/17/13	NaN
2	China Evergrande Group	EVERRE	8.75	6/28/25	NaN	B	CALLABLE	USD	4.977413	4680476000	XS1627599654	5/17/13	NaN
3	Fortune Star BVI Ltd	FOSUNI	5.25	3/23/22	NaN	NR	CALLABLE	USD	1.711157	1400000000	XS1581103428	5/17/13	NaN
4	CNPC Global Capital Ltd	CNPCCH	1.35	6/23/25	NaN	A+	CALLABLE	USD	4.963723	900000000	XS2179917906	5/17/13	NaN

1.2 Change date to to_datetime and set index

In [4]:
```
price_bond_melt['date'] = pd.to_datetime(price_bond_melt['date'])
```

In [5]:
```
price_bond_melt['Maturity'] = pd.to_datetime(price_bond_melt['Maturity'])
```

In [6]:
```
df_price=price_bond_melt.set_index(['date', 'Ticker'])
```

In [7]:
```
ytm_bond_melt=pd.melt(ytm_bond,id_vars=['Issuer Name', 'Ticker', 'Cpn', 'Maturity', 'Series', 'BBG Composite',
       'Maturity Type', 'Curr', 'Maturity (Years from Today)', 'Amount Issued',
       'ISIN'],
                          var_name='date',
                          value_name='YTM')
```

In [8]:
```
ytm_bond_melt['date'] = pd.to_datetime(ytm_bond_melt['date'])
```

In [9]:
```
ytm_bond_melt['Maturity'] = pd.to_datetime(ytm_bond_melt['Maturity'])
```

In [10]:
```
df_ytm=ytm_bond_melt.set_index(['date'])
```

2.1 Calculate daily average YTM weighted by market cap.

In [11]:
```
df_ytm.insert(11,'ytm*amount_issued',df_ytm['YTM']*df_ytm['Amount Issued'],False)
```

In [12]:
```
df_ytm_notna=df_ytm[df_ytm['YTM'].notna()]
df_ytm_notna["Amount Issued"].resample("D").sum()
df_ytm_notna["ytm*amount_issued"].resample("D").sum()
average_ytm=df_ytm_notna["ytm*amount_issued"].resample("D").sum()/df_ytm_notna["Amount Issued"].resample("D").sum()
df_average_ytm= pd.DataFrame (average_ytm,columns=['average_ytm'])
```

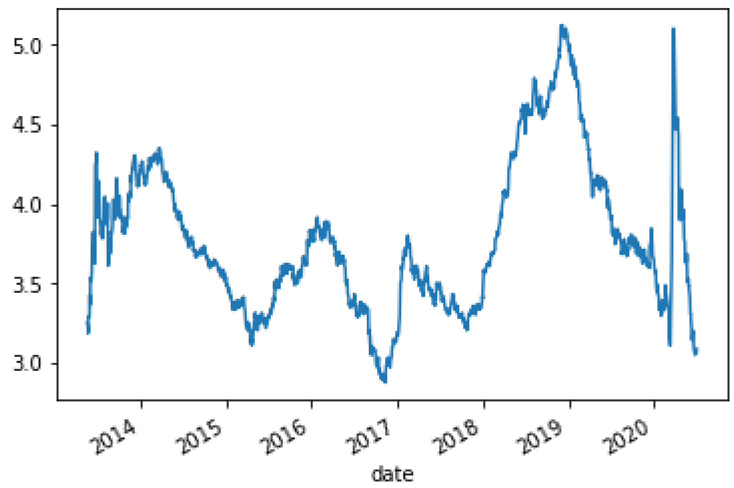
By Year

In [13]:
```
average_ytm=average_ytm[average_ytm.notna()]
average_ytm.resample('A').mean()
```

Out[13]:
```
date
2013-12-31    3.918084
2014-12-31    3.908764
2015-12-31    3.433289
2016-12-31    3.411479
2017-12-31    3.432866
2018-12-31    4.403926
2019-12-31    4.055304
2020-12-31    3.640566
Freq: A-DEC, dtype: float64
```

2.2 Plotting daily YTM

In [14]:
```
average_ytm.plot()
plt.show()
```



2.3 Group bonds by its credit rating

In [15]:
```
df_ytm_notna.groupby(['BBG Composite']).mean()
```

Out[15]:

BBG Composite	Cpn	Maturity (Years from Today)	Amount Issued	ytm*amount_issued	YTM
A	2.331450	1.268071	9.943317e+08	2.927581e+09	2.938142
A+	3.523856	4.953242	1.416602e+09	4.490539e+09	3.162012
A-	3.939469	3.157292	1.229645e+09	4.429988e+09	3.617594
B	8.740786	2.555625	1.997995e+09	2.238203e+10	11.203310
BB	4.593258	NaN	4.638772e+09	1.518265e+10	3.312127
BB+	4.596899	1.990418	1.242300e+09	5.518684e+09	4.814812
BBB	4.732306	4.807780	2.119979e+09	8.113059e+09	3.814009
BBB+	4.717803	3.501088	1.255642e+09	5.032559e+09	3.898274
BBB-	5.167806	5.989043	1.130106e+09	4.940838e+09	4.367691
NR	4.144603	3.338582	1.392700e+09	6.480583e+09	4.290655

In [16]:
```
rating_ytm=df_ytm_notna.groupby(['BBG Composite','date']).sum()['ytm*amount_issued']/df_ytm_notna.groupby(['BBG Composite','date'
```

In [17]:
```
rating_ytm.loc[(('A','A+'),),]
```

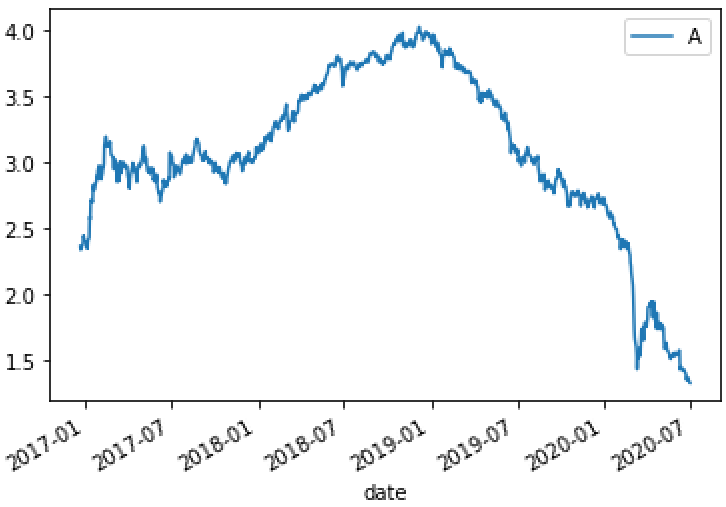
Out[17]:
```
BBG Composite  date
A              2016-12-21    2.362000
               2016-12-22    2.336000
               2016-12-23    2.365000
               2016-12-26    2.367000
               2016-12-27    2.406000
                               ...
A+             2020-06-25    1.769444
               2020-06-26    1.754333
               2020-06-29    1.745412
               2020-06-30    1.753102
               2020-07-01    1.756554
Length: 2778, dtype: float64
```

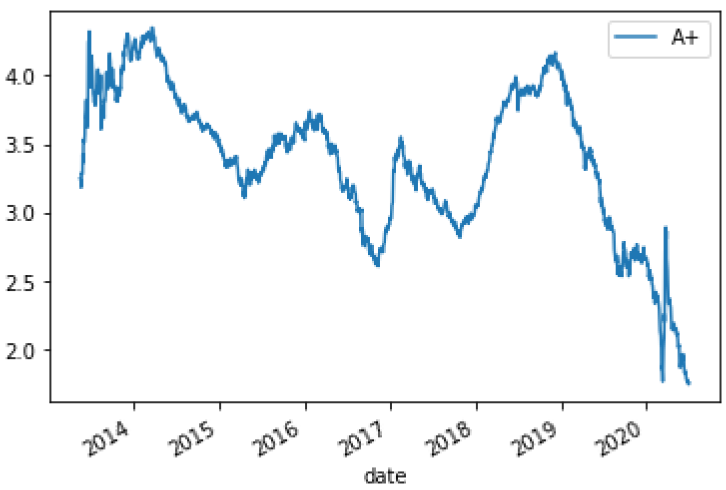
In [18]:
```
rating_ytm['B']
```

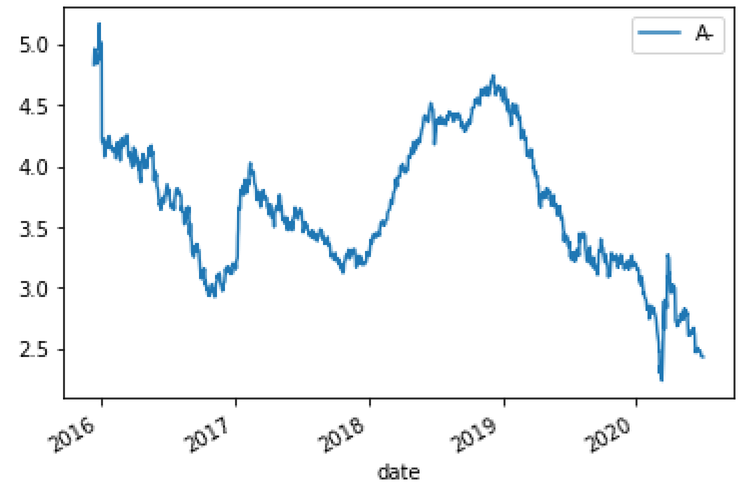
Out[18]:
```
date
2017-06-14     7.146000
2017-06-15     7.160000
2017-06-16     7.174000
2017-06-19     7.180000
2017-06-20     7.190000
                ...
2020-06-25    12.694518
2020-06-26    12.808846
2020-06-29    12.914994
2020-06-30    13.159123
2020-07-01    13.225449
Length: 795, dtype: float64
```

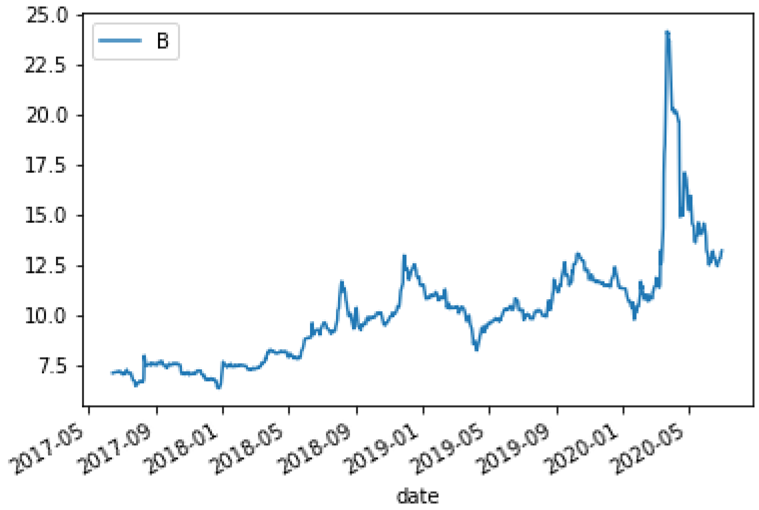
YTM Plots

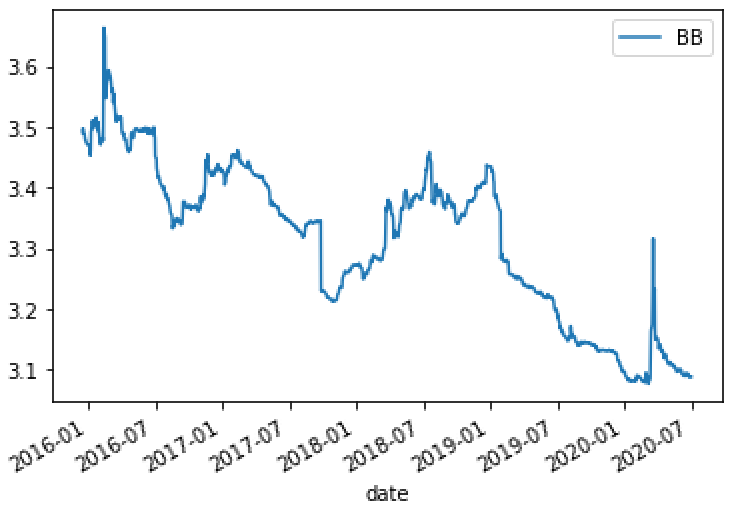
In [19]:
```
rating_ytm['A'].plot(label='A')
plt.legend()
plt.show()
rating_ytm['A+'].plot(label='A+')
plt.legend()
plt.show()
rating_ytm['A-'].plot(label='A-')
plt.legend()
plt.show()
rating_ytm['B'].plot(label='B')
plt.legend()
plt.show()
rating_ytm['BB'].plot(label='BB')
plt.legend()
plt.show()
rating_ytm['BB+'].plot(label='BB+')
plt.legend()
plt.show()
rating_ytm['BBB'].plot(label='BBB')
plt.legend()
plt.show()
rating_ytm['BBB+'].plot(label='BBB+')
plt.legend()
plt.show()
rating_ytm['BBB-'].plot(label='BBB-')
plt.legend()
plt.show()
rating_ytm['NR'].plot(label='NR')

plt.legend()
plt.show()
```

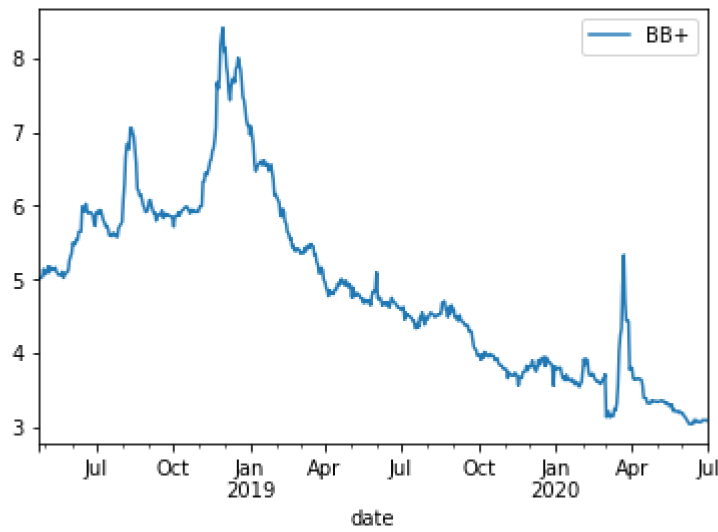


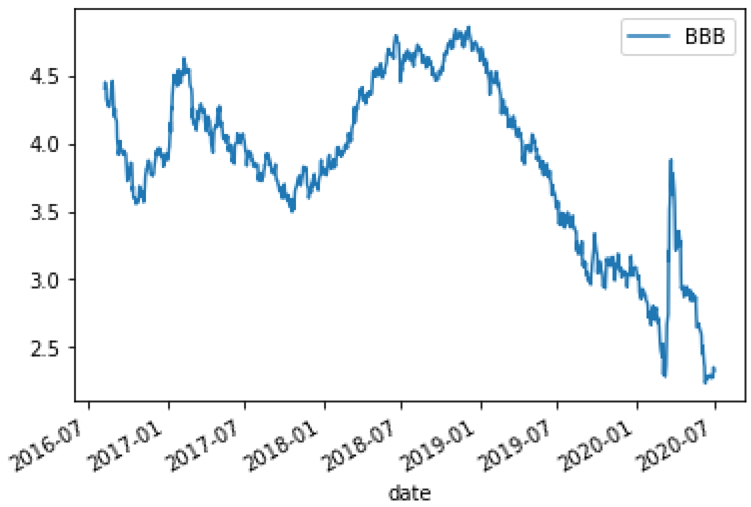


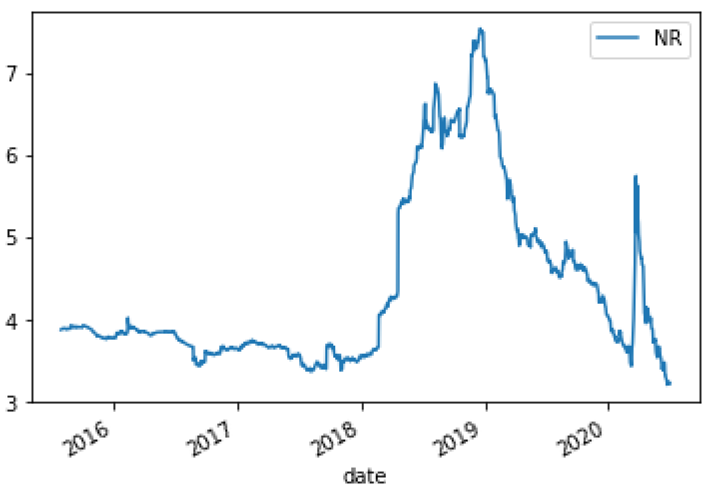

A-
5.0
4.5
4.0
3.5
3.0
2.5
2016
2017
2018
2019
2020
date


B
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
2017-05
2017-09
2018-01
2018-05
2018-09
2019-01
2019-05
2019-09
2020-01
2020-05
date


BB
3.6
3.5
3.4
3.3
3.2
3.1
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
date


BB+
8
7
6
5
4
3
Jul
Oct
Jan
2019
Apr
Jul
Oct
Jan
2020
Apr
Jul
date


BBB
4.5
4.0
3.5
3.0
2.5
2016-07
2017-01
2017-07
2018-01
2018-07
2019-01
2019-07
2020-01
2020-07
date



3.1 Change date to to_datetime and set index

In [20]:
```
df_ir_gold_fx['date'] = pd.to_datetime(df_ir_gold_fx['date'])
```

In [21]:
```
df_ir_gold_fx=df_ir_gold_fx.set_index(['date'])
```

In [22]:
```
df_ir_gold_fx=df_ir_gold_fx.fillna(method='ffill')
```

In [23]:
```
df_average_ytm= pd.DataFrame (average_ytm,columns=['average_ytm'])
```

In [24]:
```
usgdp['date'] = pd.to_datetime(usgdp['date'])
us_gdp=usgdp.set_index(['date'])
```

In [25]:
```
china_ir['date'] = pd.to_datetime(china_ir['date'])
china_ir=china_ir.set_index(['date'])
```

In [26]:
```
china_gdp['date'] = pd.to_datetime(china_gdp['date'])
china_gdp=china_gdp.set_index(['date'])
```

In [27]:
```
us_gdp_china_ir=pd.merge(us_gdp,china_ir,how='outer', left_index=True, right_index=True)
```

In [28]:
```
us_gdp_china_ir_gdp=pd.merge(us_gdp_china_ir,china_gdp,how='outer', left_index=True, right_index=True)
```

In [29]:
```
us_gdp_china_ir_gdp=us_gdp_china_ir_gdp.fillna(method='ffill')
```

3.2 Merge all macro data and Average YTM into one big dataframe.

In [30]:
```
macro=pd.merge(df_ir_gold_fx,us_gdp_china_ir_gdp, how='outer', left_index=True, right_index=True).fillna(method='ffill')
```

In [31]:
```
df_ytm_ir_gold_fx=pd.merge(df_average_ytm,macro, how='inner', left_index=True, right_index=True)
```

Convet China GDP to USD and Caculate daily return

In [32]:
```
df_ytm_ir_gold_fx.insert(8,'China_gdp(USD_billion)',df_ytm_ir_gold_fx['China_gdp(100million_rmb)']/df_ytm_ir_gold_fx['USD/CNY']/1
```

In [33]:
```
df_ytm_ir_gold_fx.insert(4,'gold_return',np.log(df_ytm_ir_gold_fx['Gold(USD/Ounce)']/df_ytm_ir_gold_fx['Gold(USD/Ounce)'].shift(1
df_ytm_ir_gold_fx.insert(6,'fx_return',np.log(df_ytm_ir_gold_fx['USD/CNY']/df_ytm_ir_gold_fx['USD/CNY'].shift(1)),False)
df_ytm_ir_gold_fx.insert(8,'US_gdp_growth',np.log(df_ytm_ir_gold_fx['US_gdp_nominal(billion)']/df_ytm_ir_gold_fx['US_gdp_nominal(
df_ytm_ir_gold_fx.insert(12,'China_gdp_growth',np.log(df_ytm_ir_gold_fx['China_gdp(100million_rmb)']/df_ytm_ir_gold_fx['China_gdp
```

In [34]:
```
df_ytm_macro=df_ytm_ir_gold_fx.dropna()
df_ytm_macro
```

Out[34]:

date	average_ytm	Benchmark interest rate	Treasure bond yields	Gold(USD/Ounce)	gold_return	USD/CNY	fx_return	US_gdp_nominal(billion)	US_gdp_growth	China_ir	China_
2013-05-20	3.230000	0.25	-0.31	1354.75	-0.010281	6.1998	0.000016	16383.0	0.0	3.35	
2013-05-21	3.184000	0.25	-0.34	1360.75	0.004419	6.1911	-0.001404	16383.0	0.0	3.35	
2013-05-22	3.285000	0.25	-0.24	1408.50	0.034489	6.1904	-0.000113	16383.0	0.0	3.35	
2013-05-23	3.289000	0.25	-0.24	1380.50	-0.020080	6.1947	0.000694	16383.0	0.0	3.35	
2013-05-24	3.284000	0.25	-0.26	1390.25	0.007038	6.1867	-0.001292	16383.0	0.0	3.35	
...	...	...	...	...	...	...	...	...	...	...	
2020-06-25	3.061947	0.25	-0.65	1756.55	-0.005394	7.0555	0.000000	18977.4	0.0	2.20	
2020-06-26	3.056591	0.25	-0.68	1747.60	-0.005108	7.0555	0.000000	18977.4	0.0	2.20	
2020-06-29	3.055808	0.25	-0.70	1771.60	0.013640	7.0808	0.003579	18977.4	0.0	2.20	
2020-06-30	3.077582	0.25	-0.68	1768.10	-0.001978	7.0795	-0.000184	18977.4	0.0	2.20	
2020-07-01	3.084522	0.25	-0.68	1771.05	0.001667	7.0710	-0.001201	18977.4	0.0	2.20	

1850 rows × 13 columns

4.1 Perform linear regression of YTM and each factor.

4.2 Summary statistics (level of significance)

4.3 Plot results

In [35]:
```
ir_ols=sm.OLS(df_ytm_macro['average_ytm'],sm.add_constant(df_ytm_macro['Benchmark interest rate'])).fit()
ir_ols.summary()
```

Out[35]:

OLS Regression Results

Dep. Variable:	average_ytm	**R-squared:**	0.216
Model:	OLS	**Adj. R-squared:**	0.216
Method:	Least Squares	**F-statistic:**	509.7
Date:	Thu, 04 Feb 2021	**Prob (F-statistic):**	7.19e-100
Time:	21:11:25	**Log-Likelihood:**	-1025.0
No. Observations:	1850	**AIC:**	2054.
Df Residuals:	1848	**BIC:**	2065.
Df Model:	1		
Covariance Type:	nonrobust		

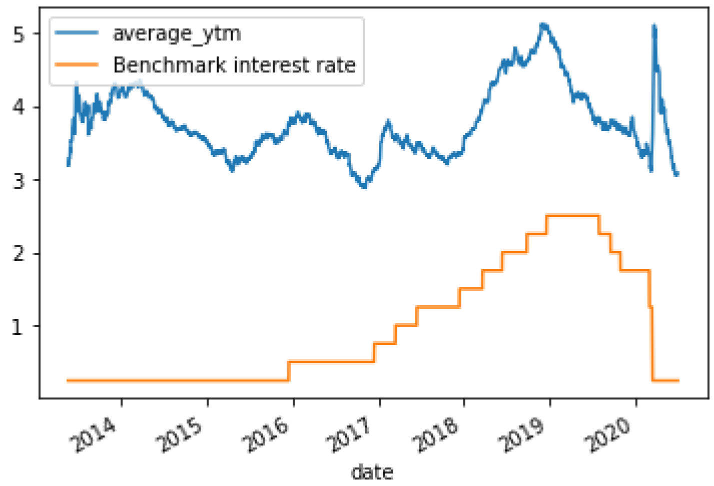
	coef	std err	t	P>|t|	[0.025	0.975]
const	3.5134	0.015	230.304	0.000	3.484	3.543
Benchmark interest rate	0.2737	0.012	22.576	0.000	0.250	0.298

Omnibus:	106.880	**Durbin-Watson:**	0.009
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	76.887
Skew:	0.395	**Prob(JB):**	2.01e-17
Kurtosis:	2.390	**Cond. No.**	2.85

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [36]:
```
df_ytm_macro[['average_ytm','Benchmark interest rate']].plot()
```

Out[36]: <AxesSubplot:xlabel='date'>



Linear regression for YTM and US interest rate is significant.

In [37]:
```
gold_ols=sm.OLS(df_ytm_macro['average_ytm'],sm.add_constant(df_ytm_macro['Gold(USD/Ounce)'])).fit()
gold_ols.summary()
```

Out[37]:

OLS Regression Results

Dep. Variable:	average_ytm	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	-0.000
Method:	Least Squares	**F-statistic:**	0.2800
Date:	Thu, 04 Feb 2021	**Prob (F-statistic):**	0.597
Time:	21:11:25	**Log-Likelihood:**	-1250.2
No. Observations:	1850	**AIC:**	2504.
Df Residuals:	1848	**BIC:**	2515.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	3.8348	0.109	35.171	0.000	3.621	4.049
Gold(USD/Ounce)	-4.427e-05	8.37e-05	-0.529	0.597	-0.000	0.000

Omnibus:	133.098	**Durbin-Watson:**	0.006
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	162.391
Skew:	0.726	**Prob(JB):**	5.46e-36
Kurtosis:	3.014	**Cond. No.**	1.28e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.28e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [38]:
```
fx_ols=sm.OLS(df_ytm_macro['average_ytm'],sm.add_constant(df_ytm_macro['USD/CNY'])).fit()
fx_ols.summary()
```

Out[38]:

OLS Regression Results

Dep. Variable:	average_ytm	**R-squared:**	0.004
Model:	OLS	**Adj. R-squared:**	0.003
Method:	Least Squares	**F-statistic:**	6.982
Date:	Thu, 04 Feb 2021	**Prob (F-statistic):**	0.00830
Time:	21:11:25	**Log-Likelihood:**	-1246.8
No. Observations:	1850	**AIC:**	2498.
Df Residuals:	1848	**BIC:**	2509.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	3.2051	0.217	14.777	0.000	2.780	3.630
USD/CNY	0.0875	0.033	2.642	0.008	0.023	0.152

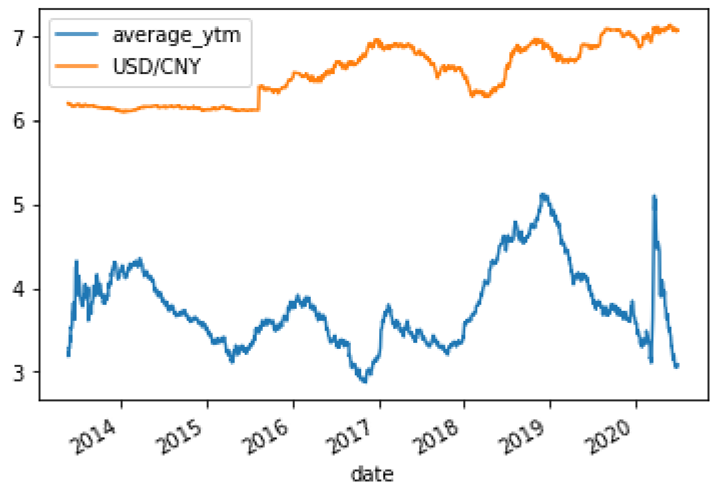
Omnibus:	113.048	**Durbin-Watson:**	0.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	133.415
Skew:	0.655	**Prob(JB):**	1.07e-29
Kurtosis:	2.881	**Cond. No.**	132.

/

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [39]:
```
df_ytm_macro[['average_ytm','USD/CNY']].plot()
```

Out[39]: <AxesSubplot:xlabel='date'>



Linear regression for YTM and foreign exchange rate is significant

In [40]:
```
us_gdp_ols=sm.OLS(df_ytm_macro['average_ytm'],sm.add_constant(df_ytm_macro['US_gdp_nominal(billion)'])).fit()
us_gdp_ols.summary()
```

Out[40]:

OLS Regression Results

Dep. Variable:	average_ytm	**R-squared:**	0.036
Model:	OLS	**Adj. R-squared:**	0.036
Method:	Least Squares	**F-statistic:**	69.89
Date:	Thu, 04 Feb 2021	**Prob (F-statistic):**	1.22e-16
Time:	21:11:25	**Log-Likelihood:**	-1216.0
No. Observations:	1850	**AIC:**	2436.
Df Residuals:	1848	**BIC:**	2447.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.8818	0.227	8.289	0.000	1.437	2.327
US_gdp_nominal(billion)	0.0001	1.27e-05	8.360	0.000	8.12e-05	0.000

Omnibus:	98.023	**Durbin-Watson:**	0.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	100.714
Skew:	0.535	**Prob(JB):**	1.35e-22
Kurtosis:	2.596	**Cond. No.**	3.74e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.74e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Linear regression for YTM and US GDP is significant

In [41]:
```
china_ir_ols=sm.OLS(df_ytm_macro['average_ytm'],sm.add_constant(df_ytm_macro['China_ir'])).fit()
china_ir_ols.summary()
```

Out[41]:

OLS Regression Results

Dep. Variable:	average_ytm	**R-squared:**	0.015
Model:	OLS	**Adj. R-squared:**	0.014
Method:	Least Squares	**F-statistic:**	27.54
Date:	Thu, 04 Feb 2021	**Prob (F-statistic):**	1.71e-07
Time:	21:11:25	**Log-Likelihood:**	-1236.6
No. Observations:	1850	**AIC:**	2477.
Df Residuals:	1848	**BIC:**	2488.
Df Model:	1		
Covariance Type:	nonrobust		

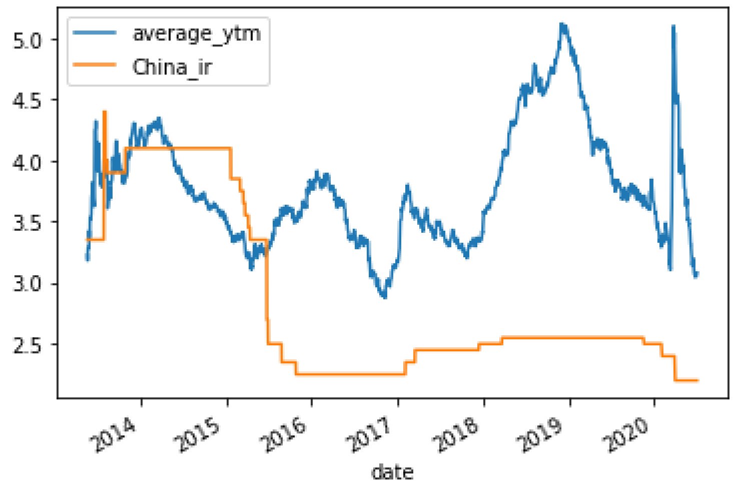
	coef	std err	t	P>|t|	[0.025	0.975]
const	3.5434	0.046	77.133	0.000	3.453	3.633
China_ir	0.0820	0.016	5.248	0.000	0.051	0.113

Omnibus:	175.985	Durbin-Watson:	0.007
Prob(Omnibus):	0.000	Jarque-Bera (JB):	227.057
Skew:	0.848	Prob(JB):	4.96e-50
Kurtosis:	3.257	Cond. No.	13.7

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [42]:
```
df_ytm_macro[['average_ytm','China_ir']].plot()
```

Out[42]: <AxesSubplot:xlabel='date'>



Linear regression for YTM and China's interest rate is significant

In [43]:
```
china_gdp_ols=sm.OLS(df_ytm_macro['average_ytm'],sm.add_constant(df_ytm_macro['China_gdp(USD_billion)'])).fit()
china_gdp_ols.summary()
```

Out[43]:

OLS Regression Results

Dep. Variable:	average_ytm	R-squared:	0.038
Model:	OLS	Adj. R-squared:	0.038
Method:	Least Squares	F-statistic:	73.39
Date:	Thu, 04 Feb 2021	Prob (F-statistic):	2.21e-17
Time:	21:11:25	Log-Likelihood:	-1214.3
No. Observations:	1850	AIC:	2433.
Df Residuals:	1848	BIC:	2444.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.2037	0.068	47.218	0.000	3.071	3.337
China_gdp(USD_billion)	0.0002	2.48e-05	8.567	0.000	0.000	0.000

Omnibus:	91.906	Durbin-Watson:	0.007
Prob(Omnibus):	0.000	Jarque-Bera (JB):	91.276
Skew:	0.501	Prob(JB):	1.51e-20
Kurtosis:	2.577	Cond. No.	1.71e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.71e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Linear regression for YTM and China's GDP is significant

4.4 Conclusion: The Chinese Offshore USD Bonds YTM are positively correlatated with macroeconomic factors such as US and China's interest rate, US and China's GDP and USD/CNY exchange rate. The regressions are statistically significant according to t-stats, p-value and R-square. However, there are no significant correlation between these bonds YTM and gold price and growth of the macroeconomic factors.

5.1 Find cross correlation of macro factors.

In [44]:
```
df_ytm_macro[['Benchmark interest rate','USD/CNY','US_gdp_nominal(billion)','China_ir','China_gdp(USD_billion)']].corr().style.ba
```

Out[44]:

	Benchmark interest rate	USD/CNY	US_gdp_nominal(billion)	China_ir	China_gdp(USD_billion)
Benchmark interest rate	1.000000	0.602782	0.809376	-0.442251	0.793949
USD/CNY	0.602782	1.000000	0.841694	-0.746211	0.680304
US_gdp_nominal(billion)	0.809376	0.841694	1.000000	-0.705358	0.856223
China_ir	-0.442251	-0.746211	-0.705358	1.000000	-0.598405
China_gdp(USD_billion)	0.793949	0.680304	0.856223	-0.598405	1.000000

5.2 Define insample data (prior to 2019). Out of sample data (2020)

In [45]:
```
df_train=df_ytm_macro[df_ytm_macro.index<'2019'][['average_ytm','Benchmark interest rate','USD/CNY','US_gdp_nominal(billion)','Ch
df_test=df_ytm_macro[df_ytm_macro.index>'2019'][['average_ytm','Benchmark interest rate','USD/CNY','US_gdp_nominal(billion)','Chi
df_train
```

Out[45]:

date	average_ytm	Benchmark interest rate	USD/CNY	US_gdp_nominal(billion)	China_ir	China_gdp(USD_billion)
2013-05-20	3.230000	0.25	6.1998	16383.0	3.35	1860.422917
2013-05-21	3.184000	0.25	6.1911	16383.0	3.35	1863.037263
2013-05-22	3.285000	0.25	6.1904	16383.0	3.35	1863.247932
2013-05-23	3.289000	0.25	6.1947	16383.0	3.35	1861.954574
2013-05-24	3.284000	0.25	6.1867	16383.0	3.35	1864.362261
...	...	...	...	...	...	...
2018-12-25	5.066437	2.50	6.8919	18783.5	2.55	3121.037740
2018-12-26	5.058272	2.50	6.8845	18783.5	2.55	3124.392476
2018-12-27	5.023874	2.50	6.8894	18783.5	2.55	3122.170291
2018-12-28	5.002768	2.50	6.8632	18783.5	2.55	3134.089055
2018-12-31	5.000827	2.50	6.8632	18783.5	2.55	3411.066266

1460 rows × 6 columns

5.3 Standardize all factors (x-mean)/sd

In [46]:
```
df_z_train=(df_train-df_train.mean())/df_train.std()
df_z_test=(df_test-df_train.mean())/df_train.std()
```

In [47]:
```
df_z_ytm_macro=(df_ytm_macro-df_ytm_macro.mean())/df_ytm_macro.std()
```

In [48]:
```
df_z_ytm_macro
```

Out[48]:

date	average_ytm	Benchmark interest rate	Treasure bond yields	Gold(USD/Ounce)	gold_return	USD/CNY	fx_return	US_gdp_nominal(billion)	US_gdp_growth	China_ir	Ch
2013-05-20	-1.150686	-0.884249	-2.285680	0.441786	-1.151312	-1.034197	-0.029525	-1.724545	-0.091864	0.701649	
2013-05-21	-1.247373	-0.884249	-2.379079	0.487153	0.472866	-1.060263	-0.792692	-1.724545	-0.091864	0.701649	
2013-05-22	-1.035083	-0.884249	-2.067750	0.848201	3.795258	-1.062360	-0.098945	-1.724545	-0.091864	0.701649	
2013-05-23	-1.026675	-0.884249	-2.067750	0.636487	-2.233936	-1.049477	0.334896	-1.724545	-0.091864	0.701649	
2013-05-24	-1.037185	-0.884249	-2.130016	0.710209	0.762206	-1.073446	-0.732517	-1.724545	-0.091864	0.701649	
...	...	...	...	...	...	...	...	...	...	...	
2020-06-25	-1.503914	-0.884249	-3.344201	3.479882	-0.611334	1.529559	-0.038192	1.307748	-0.091864	-0.933085	
2020-06-26	-1.515171	-0.884249	-3.437599	3.412209	-0.579788	1.529559	-0.038192	1.307748	-0.091864	-0.933085	
2020-06-29	-1.516819	-0.884249	-3.499865	3.593679	1.491629	1.605360	1.885023	1.307748	-0.091864	-0.933085	
2020-06-30	-1.471052	-0.884249	-3.437599	3.567214	-0.233887	1.601465	-0.136846	1.307748	-0.091864	-0.933085	
2020-07-01	-1.456464	-0.884249	-3.437599	3.589520	0.168802	1.575998	-0.683682	1.307748	-0.091864	-0.933085	

1850 rows × 13 columns

5.4 Perform linear regression of YTM to all factors.

In [50]:
```
train_y=df_z_train['average_ytm']
train_x=df_z_train[['Benchmark interest rate','USD/CNY','US_gdp_nominal(billion)','China_ir','China_gdp(USD_billion)']]
test_y=df_z_test['average_ytm']
test_x=df_z_test[['Benchmark interest rate','USD/CNY','US_gdp_nominal(billion)','China_ir','China_gdp(USD_billion)']]
model=sm.OLS(train_y,(train_x)).fit()
print(model.summary())
```

```
                                 OLS Regression Results                                
=======================================================================================
Dep. Variable:            average_ytm   R-squared (uncentered):                   0.625
Model:                            OLS   Adj. R-squared (uncentered):              0.623
Method:                 Least Squares   F-statistic:                              484.2
Date:                Thu, 04 Feb 2021   Prob (F-statistic):                  1.93e-306
Time:                        21:13:07   Log-Likelihood:                         -1355.9
No. Observations:                1460   AIC:                                      2722.
Df Residuals:                    1455   BIC:                                      2748.
Df Model:                           5                                                  
Covariance Type:            nonrobust                                                  
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
Benchmark interest rate     1.6073      0.043     37.537      0.000       1.523       1.691
USD/CNY                    -0.0946      0.028     -3.378      0.001      -0.150      -0.040
US_gdp_nominal(billion)    -0.9115      0.062    -14.729      0.000      -1.033      -0.790
China_ir                    0.0553      0.034      1.647      0.100      -0.011       0.121
China_gdp(USD_billion)     -0.2587      0.034     -7.528      0.000      -0.326      -0.191
==============================================================================
Omnibus:                       45.123   Durbin-Watson:                   0.021
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               37.381
Skew:                          -0.317   Prob(JB):                     7.63e-09
Kurtosis:                       2.540   Cond. No.                         9.29
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

5.5 Perform PCA

In [51]:
```
pca = PCA(0.95)
pca.fit_transform(train_x)
```

Out[51]:
```
array([[-2.60801231, -0.57282551, -0.41036672],
       [-2.61792301, -0.55753185, -0.38622622],
       [-2.61872005, -0.55630107, -0.38428354],
       ...,
       [ 3.62685239,  1.14432736, -0.90057066],
       [ 3.60179695,  1.19362797, -0.82356478],
       [ 3.92977239,  1.41571   , -0.52870391]])
```

In [52]:
```
pca_model=sm.OLS(train_y,pca.fit_transform(train_x)).fit()
print(pca_model.summary())
```

```
                                 OLS Regression Results                                
=======================================================================================
Dep. Variable:            average_ytm   R-squared (uncentered):                   0.457
Model:                            OLS   Adj. R-squared (uncentered):              0.456
Method:                 Least Squares   F-statistic:                              409.1
Date:                Thu, 04 Feb 2021   Prob (F-statistic):                  1.01e-192
Time:                        21:13:13   Log-Likelihood:                         -1625.1
No. Observations:                1460   AIC:                                      3256.
Df Residuals:                    1457   BIC:                                      3272.
Df Model:                           3                                                  
Covariance Type:            nonrobust                                                  
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1             0.0784      0.010      8.019      0.000       0.059       0.098
x2             0.6999      0.024     28.751      0.000       0.652       0.748
x3            -0.6889      0.038    -18.338      0.000      -0.763      -0.615
==============================================================================
Omnibus:                    33427.759   Durbin-Watson:                   0.011
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              114.075
Skew:                          -0.021   Prob(JB):                     1.69e-25
Kurtosis:                       1.631   Cond. No.                         3.84
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

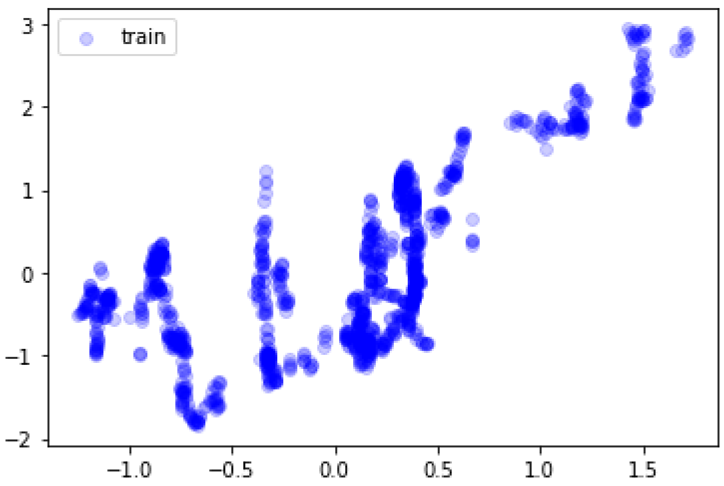
5.6 Use result to predict both train and target data

In [53]:
```
train_predict=pca_model.predict(pca.fit_transform(train_x))
test_predict=pca_model.predict(pca.fit_transform(test_x))
```

5.7 Plot the result

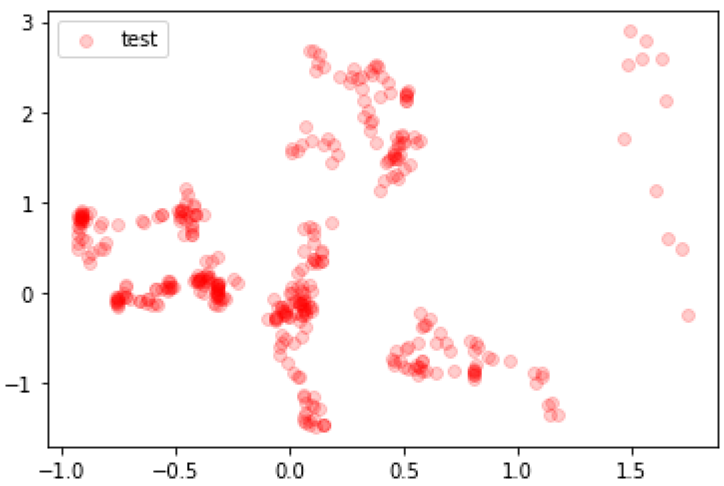
In [54]:
```
plt.scatter(train_predict, train_y, alpha=0.2, color='b', label='train')

plt.legend()
```

Out[54]: <matplotlib.legend.Legend at 0x7f87ff8c1130>



In [55]:
```
plt.scatter(test_predict, test_y, alpha=0.2, color='r', label='test')
plt.legend()
```

Out[55]: <matplotlib.legend.Legend at 0x7f87ff4a7520>



5.8 Compute RMSE and R^2

In [56]:
```
rmse_train=np.sqrt(((train_predict - train_y) ** 2).mean())
rmse_train
```

Out[56]: 0.736504640236427

In [57]:
```
rmse_test=np.sqrt(((test_predict - test_y) ** 2).mean())
rmse_test
```

Out[57]: 1.189837490971203

5.9 Run correlation between actual targets and predicted targets

In [58]:
```
sm.OLS(test_y,test_predict).fit().summary()
```

Out[58]:

OLS Regression Results

Dep. Variable:	average_ytm	**R-squared (uncentered):**	0.002
Model:	OLS	**Adj. R-squared (uncentered):**	-0.000
Method:	Least Squares	**F-statistic:**	0.8394
Date:	Thu, 04 Feb 2021	**Prob (F-statistic):**	0.360
Time:	21:13:20	**Log-Likelihood:**	-579.82
No. Observations:	390	**AIC:**	1162.
Df Residuals:	389	**BIC:**	1166.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	0.0872	0.095	0.916	0.360	-0.100	0.274

Omnibus:	14.875	**Durbin-Watson:**	0.013
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	14.048
Skew:	0.413	**Prob(JB):**	0.000890
Kurtosis:	2.574	**Cond. No.**	1.00

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [59]:
```
sm.OLS(test_y,test_predict).fit().summary()
```

Out[59]:

OLS Regression Results

Dep. Variable:	average_ytm	**R-squared (uncentered):**	0.002

Model:	OLS	**Adj. R-squared (uncentered):**	-0.000
Method:	Least Squares	**F-statistic:**	0.8394
Date:	Thu, 04 Feb 2021	**Prob (F-statistic):**	0.360
Time:	21:13:21	**Log-Likelihood:**	-579.82
No. Observations:	390	**AIC:**	1162.
Df Residuals:	389	**BIC:**	1166.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
x1	0.0872	0.095	0.916	0.360	-0.100	0.274

Omnibus:	14.875	**Durbin-Watson:**	0.013
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	14.048
Skew:	0.413	**Prob(JB):**	0.000890
Kurtosis:	2.574	**Cond. No.**	1.00

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

5.10 Conclusion:

The factors in the models have high correlations with each other, so PCA is used to reduce multicolinearity. The PCA Model give an estimation of YTM above with high p value (0.36) and low R^2 (0).

Author: Nan Li