Data Generation

```
# !pip install fredapi
# !pip install quandl
# !pip install hmmlearn
# !pip install yfinance
# !pip install openpyxl

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from IPython.display import Image
from fredapi import Fred
import quandl
import yfinance as yf
import plotly.express as px

%matplotlib inline
```

First we combine all the data we received from the excel sheet.

```
basepath = '/data/workspace_files/'
freq = 'W-MON'
```

```
# Squeeze_data
filename = 'Squeeze_Metrics.csv'
df_squeeze = pd.read_csv(basepath+filename)
df_squeeze.index = pd.to_datetime(df_squeeze['date'])
df_squeeze = df_squeeze.drop(columns={'date'})

# Sector_Valuation
filename = 'Sector_Valuation.csv'
df_sector_val = pd.read_csv(basepath+filename)
df_sector_val.index = pd.to_datetime(df_sector_val['date'])
df_sector_val = df_sector_val.drop(columns={'date','Real Estate'})

# Implied Correlation
filename = 'IC.csv'
df_IC = pd.read_csv(basepath+filename)
df_IC.index = pd.to_datetime(df_IC['date'])
df_IC = df_IC.drop(columns={'date'})

# Equity_Risk_Premium
filename = 'Equity_Risk_Premium.csv'
df_ER = pd.read_csv(basepath+filename)
df_ER.index = pd.to_datetime(df_ER['date'])
df_ER = df_ER.drop(columns={'date'})
df_ER['Real_Bond_Yield'] = pd.to_numeric(df_ER['Real_Bond_Yield'])

# Average Investor Holding
filename = 'Avg_Investor_Holding.csv'
df_Avg_Inv_Hol = pd.read_csv(basepath+filename)
df_Avg_Inv_Hol.index = pd.to_datetime(df_Avg_Inv_Hol['date'])
df_Avg_Inv_Hol = df_Avg_Inv_Hol.drop(columns={'date'})

# Skew
filename = 'SKEW.csv'
df_skew = pd.read_csv(basepath+filename)
df_skew.index = pd.to_datetime(df_skew['date'])
df_skew = df_skew.drop(columns={'date'})

# Liquidity
filename = 'IAQF_Liquidity.csv'
df_liquidity = pd.read_csv(basepath+filename)
df_liquidity.index = pd.to_datetime(df_liquidity['date'])
df_liquidity = df_liquidity.drop(columns={'date'})
```

Fred data

```
fred = Fred(api_key='d19731bfe189a749611dfbace3924358')
```

```
rates = {'DFF': 'Fed_Fund_Rate', 'DGS1MO':'Rate_1M','DGS3MO':'Rate_3M','DGS6MO':'Rate_6M','DGS1':'Rate_1Y','DGS2':'Rate_2Y',
           'DGS3':'Rate_3Y','DGS5':'Rate_5Y','DGS7':'Rate_7Y','DGS10':'Rate_10Y'}

VIX = {'VIXCLS':'VIX', 'VXVCLS':'VIX_3M'}

money_credit = {'T10Y2Y': 'Term_Spread',
                    'BAMLH0A0HYM2': 'HY_OAS', 'BAMLC0A0CM':'IG_OAS','BAMLC0A4CBBB':'BBB_OAS' ,
                   'BAMLEMCBPIOAS':'EM_OAS'}

all_vars = {**rates, **VIX, **money_credit}
final_D = pd.DataFrame()

for var in all_vars:
    var_freq = fred.get_series_info(var)
    series_tmp = fred.get_series(var, aggregation_method='eop')
    series_df_tmp = pd.DataFrame(series_tmp, columns=[all_vars[var]])
    if len(final_D)==0:
        final_D = series_df_tmp
    else:
        final_D = pd.concat([final_D,series_df_tmp],axis=1)
```

```
final_D = final_D.loc[pd.to_datetime('1990-01-02'):,]
```

```
index = pd.bdate_range(start='2/1/1990', end='2/14/2022')
final_D = final_D.loc[index.values]
```

```
consumer_sentiment = {'UMCSENT': 'Cons_Sent', 'CPIAUCNS': 'CPI'}
industrial_data = {'AMTMNO':'Manufacturing_New_Order'}
labor_market = {'UNRATE' : 'Unemployement_rate'}
case_shiller = {'CSUSHPINSA': 'Case Shiller'}

all_vars = {**consumer_sentiment,**industrial_data,**labor_market,**case_shiller}
final_M = pd.DataFrame()

for var in all_vars:
    var_freq = fred.get_series_info(var)
    series_tmp = fred.get_series(var, aggregation_method='eop')
    series_df_tmp = pd.DataFrame(series_tmp, columns=[all_vars[var]])
    if len(final_M)==0:
        final_M = series_df_tmp
    else:
        final_M = pd.concat([final_M,series_df_tmp],axis=1).dropna()
```

```
GDP_based_indicators = {'JHDUSRGDPBR': 'GDP_based_Recess_Ind',
                        'NA000334Q': 'GDP'}
all_vars = {**GDP_based_indicators}
final_Q = pd.DataFrame()

for var in all_vars:
    var_freq = fred.get_series_info(var)
    series_tmp = fred.get_series(var, aggregation_method='eop')
    series_df_tmp = pd.DataFrame(series_tmp, columns=[all_vars[var]])
    if len(final_Q)==0:
        final_Q = series_df_tmp
    else:
        final_Q = pd.concat([final_Q,series_df_tmp],axis=1).dropna()
```

```
asset = '^VVIX'
VVIX_data_df = yf.download(asset,
                      start='1990-01-01',
                      end='2021-12-31',
                      progress=False)[['Close']].rename(columns={'Close':'VVIX'})
```

```
final = pd.concat([final_D, final_M, final_Q, df_squeeze, df_sector_val,
                   df_IC, df_ER, df_Avg_Inv_Hol, VVIX_data_df, df_skew, df_liquidity], axis=1)
final = final.loc[pd.to_datetime('1990-01-01'):,]
freq = 'W-MON'
final = final.resample(freq).first()
final = final.ffill()
```

```
filter = final.isna().sum()/final.shape[0]
filter_new = filter[filter<0.70]
filter_new.index

final_new_70 = final[filter_new.index].dropna()
```

```
filter = final.isna().sum()/final.shape[0]
filter_new = filter[filter<0.1]
filter_new.index

final_new_10 = final[filter_new.index].dropna()
```

Data from Yahoo Fin

```
filter = final.isna().sum()/final.shape[0]
filter_new = filter[filter<0.30]
filter_new.index

final_new_30 = final[filter_new.index].dropna()
```

```
def preprocess_spy_vix_data(filename='MFE+230K+HW1.xlsx', basepath = '/data/workspace_files/'):
    """
    Read SPY Index, VIX Index, and Risk free rate from the xlsx file.
    preprocess the data.

    :param filename (str): name of the data file
    :param basepath (str): name of the base path
    :return data (DataFrame): all the data
    """

    data = pd.read_excel(basepath+filename,
                         sheet_name='SP',
                         header=1,
                         index_col=0)
    data.index = pd.to_datetime(data.index)
    data = data.dropna()
    data = data.rename(columns={'S&P Return Index': 'SP500',
                                'Overnight Risk Free Rate': 'RF'})
    data= data.drop(columns={'VIX'})

    data['RF'] = data['RF'] / 100
    return data

spy_vix_data = preprocess_spy_vix_data()
```

```
asset = '^RUA'
russel_data_df = yf.download(asset,
                      start='1990-01-01',
                      end='2021-12-31',
                      progress=False)
```

```
asset = 'IWV'
russel_etf_data_df = yf.download(asset,
                      start='1990-01-01',
                      end='2021-12-31',
                      progress=False)
```

```
spx_pe = quandl.get("MULTPL/SP500_PE_RATIO_MONTH", authtoken="Go75ZA9T38nzzWtKMhZ7", start_date= '1990-01-01',
                      end_date= '2021-12-31')
```

```
spx_price = yf.download('^GSPC',
                      start='1990-01-01',
                      end='2021-12-31',
                      progress=False)
spx_data = pd.merge(spx_price['Close'], spx_pe, left_index = True, right_index = True, how = 'outer')
spx_data['Value'] = spx_data['Value'].shift(1)
spx_data = spx_data.dropna(how='all')
spx_data['Close'].fillna(method = 'ffill', inplace = True)
spx_data['earnings'] = spx_data['Close']/spx_data['Value']
spx_data['earnings'].fillna(method = 'ffill', inplace = True)
spx_data['pe'] = spx_data['Close']/spx_data['earnings']
```

Fama Fench factors

```
ff_data = pd.read_csv('/data/workspace_files/F-F_Research_Data_5_Factors_2x3_daily.csv', skiprows = 3, index_col = 0, parse_dates = Tr
ff_data = ff_data.loc['1990-01-01':]
ff_data.drop('RF', axis = 1, inplace = True)
```

```
# cumulative return
ff_data_w = ((ff_data/100.+1).apply(np.log).resample(freq).sum().apply(np.exp) - 1)*100
```

Volume

```
data_df = pd.concat([russel_data_df, spy_vix_data[['RF']], spx_data['pe']],axis=1).dropna()
#data_df['Volume'] = russel_etf_data_df['Volume']
data_df = data_df.drop(columns=['Volume'])
data_df_W = data_df.resample(freq).first()

# add ff factors
data_df = pd.concat([data_df, ff_data], axis=1).dropna()
data_df_W = pd.concat([data_df_W, ff_data_w], axis=1).dropna()
```

```
# Feature params
 # corresponds to biweekly frequency
ma_period = 10
column_price = 'Close'

#computed daily
data_df['close_open_return'] = (data_df['Open'] - data_df['Close'].shift(1)) / data_df['Open']


close_weekly = data_df[column_price].resample(freq).first()
data_df_W['last_return'] = close_weekly.pct_change() #weekly return
data_df_W['last_return1'] = close_weekly.pct_change(periods=4) #monthly
data_df_W['last_return2'] = data_df[column_price].pct_change(periods=12) #quarterly
data_df_W['close_open_return'] = (data_df[['close_open_return']]+1.).apply(np.log).resample(freq).sum().apply(np.exp) - 1.

#forward looking return
return_period = 1 # future reutrn
data_df_W["future_return"] = close_weekly.pct_change(return_period).shift(-return_period)

def values_deviation(vals):
    """ z-score for volumes and price """
    return (vals[-1] - np.mean(vals)) / np.std(vals)

price_deviation_period = 20
data_df['price_deviation'] = data_df[column_price].rolling(price_deviation_period).apply(values_deviation)
#data_df['volume_deviation'] = data_df['Volume'].rolling(5).apply(values_deviation) #weekly

data_df_W["price_deviation"] = data_df["price_deviation"].resample(freq).first()
#data_df_W["volume_deviation"] = data_df["volume_deviation"].resample(freq).first()

data_df = data_df.replace([np.inf, -np.inf], np.nan)
data_df_W = data_df_W.replace([np.inf, -np.inf], np.nan)
```

```
orig_data_df = data_df.copy()
orig_data_df_W = data_df_W.copy()
```

```
data_df_W = orig_data_df_W[['Close',
                        'RF',
                        'last_return',
                        'last_return1',
                        'last_return2',
                        'close_open_return',
                        'future_return',
                        "pe",
                        "Mkt-RF", 'SMB', 'HML', 'RMW', 'CMA',
                        'price_deviation',
                    ]]
```

```
data_df_W = data_df_W.ffill().dropna()
```

```
# Transoform the data
def transform(data):
    #data['Close'] = data['Close'].diff()
    data['pe'] = data['pe'].apply(np.log).diff()
    data['Fed_Fund_Rate'] = data['Fed_Fund_Rate'].diff()
    data['VIX'] = data['VIX'].apply(np.log).diff()
    data['Cons_Sent'] = data['Cons_Sent'].apply(np.log).diff()
    data['CPI'] = data['CPI'].diff()
    data['Manufacturing_New_Order'] = data['Manufacturing_New_Order'].diff()

    for c in ['Avg. Stock Holding', 'SKEW', 'Consumer Staples',
    'Equity Risk Premium', 'Long_Term_Growth_Forecast',
    'Real_Bond_Yield', 'CAPE', 'GDP', 'Case Shiller',
    'Unemployement_rate', 'Energy', 'Financials', 'Industrials', 'Consumer Discretionary',
    'Information Technology', 'Materials', 'Utilities', 'Communication Services', 'Health Care',
    'VVIX', 'HY_OAS', 'IG_OAS', 'BBB_OAS', 'price', 'gex', 'IC', 'Liquidity (Scaled by GDP)',
    'Rate_1M', 'Rate_3M', 'Rate_6M', 'Rate_1Y', 'Rate_2Y',
    'Rate_3Y', 'Rate_5Y', 'Rate_7Y', 'Rate_10Y', 'Term_Spread',]:
        if c in data.columns:
            data[c] = data[c].diff()

    return data
```

```
data_temp = pd.concat([data_df_W, final_new_10],axis=1).dropna()
data_fut_return = data_temp[['future_return']]
data_df_92 = data_temp.drop(columns={'future_return'})

data_vix = data_temp[['VIX']]
data_df_92 = data_temp.drop(columns={'VIX'})

data_rf = data_temp[['RF']]
data_df_92 = data_temp.drop(columns={'RF'})
data_df_W = data_df_W.drop(columns={'RF'})

data_df_92 = transform(data_df_92)
```

```
data_df_92.columns
```

```
Index(['Close', 'last_return', 'last_return1', 'last_return2',
       'close_open_return', 'future_return', 'pe', 'Mkt-RF', 'SMB', 'HML',
       'RMW', 'CMA', 'price_deviation', 'Fed_Fund_Rate', 'Rate_3M', 'Rate_6M',
       'Rate_1Y', 'Rate_2Y', 'Rate_3Y', 'Rate_5Y', 'Rate_7Y', 'Rate_10Y',
       'VIX', 'Term_Spread', 'Cons_Sent', 'CPI', 'Manufacturing_New_Order',
       'Unemployement_rate', 'Case Shiller', 'GDP_based_Recess_Ind', 'GDP',
       'CAPE', 'Real_Bond_Yield', 'Long_Term_Growth_Forecast',
       'Equity Risk Premium', 'Avg. Stock Holding', 'SKEW',
       'Liquidity (Scaled by GDP)'],
      dtype='object')
```

```
data_df_99 = pd.concat([data_df_W, final_new_30],axis=1).dropna()
data_df_99 = data_df_99.drop(columns={'future_return'})
data_df_99 = transform(data_df_99)
```

```
data_df_11 = pd.concat([data_df_W, final_new_70],axis=1).dropna()
data_df_11 = data_df_11.drop(columns={'future_return'})
data_df_11 = transform(data_df_11)
```

```
from pandas import read_csv
from statsmodels.tsa.stattools import adfuller
def is_stationary(X):
    # Reject the null hypothesis (H0), the data does not have a unit root and is stationary.
    result = adfuller(X)
    return result[1] < 0.05

for col in data_df_92.columns:
    if not is_stationary(data_df_92[col].dropna()):
        print(col)

print("99:")
for col in data_df_99.columns:

    if not is_stationary(data_df_99[col].dropna()):
        print(col)

print("11:")
for col in data_df_11.columns:
    if not is_stationary(data_df_11[col].dropna()):
        print(col)
```

```
Close
99:
Close
11:
Close
```

```
data_close = data_df_92[['Close']]
```

Dump to pickle file

```
pd.to_pickle(data_df_92.dropna(), '/data/workspace_files/data_df92.pickle') #37 columns

pd.to_pickle(data_fut_return.dropna(), '/data/workspace_files/data_fut_return.pickle')
pd.to_pickle(data_vix.dropna(), '/data/workspace_files/data_vix.pickle')
pd.to_pickle(data_rf.dropna(), '/data/workspace_files/data_rf.pickle')
pd.to_pickle(data_close.dropna(), '/data/workspace_files/data_close.pickle')

pd.to_pickle(data_df_99.dropna(), '/data/workspace_files/data_df99.pickle') #51 columns
pd.to_pickle(data_df_11.dropna(), '/data/workspace_files/data_df11.pickle') #58 columns
```

```
data_df_92.columns
```

```
Index(['Close', 'last_return', 'last_return1', 'last_return2',
       'close_open_return', 'future_return', 'pe', 'Mkt-RF', 'SMB', 'HML',
       'RMW', 'CMA', 'price_deviation', 'Fed_Fund_Rate', 'Rate_3M', 'Rate_6M',
       'Rate_1Y', 'Rate_2Y', 'Rate_3Y', 'Rate_5Y', 'Rate_7Y', 'Rate_10Y',
       'VIX', 'Term_Spread', 'Cons_Sent', 'CPI', 'Manufacturing_New_Order',
       'Unemployement_rate', 'Case Shiller', 'GDP_based_Recess_Ind', 'GDP',
       'CAPE', 'Real_Bond_Yield', 'Long_Term_Growth_Forecast',
       'Equity Risk Premium', 'Avg. Stock Holding', 'SKEW',
       'Liquidity (Scaled by GDP)'],
      dtype='object')
```