00_data_prep

September 29, 2021

1 How to prepare the data

We use a simplified version of the data set constructed in Chapter 4, Alpha factor research. It consists of daily stock prices provided by Quandl for the 2010-2017 period and various engineered features.

The decision tree models in this chapter are not equipped to handle missing or categorical variables, so we will apply dummy encoding to the latter after dropping any of the former.

[1]:
```
%matplotlib inline

import warnings
import os
from pathlib import Path
import quandl
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
import graphviz
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor,␣
 ↪export_graphviz, _tree
from sklearn.linear_model import LinearRegression, Ridge, LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪GridSearchCV
from sklearn.metrics import roc_auc_score, roc_curve, mean_squared_error,␣
 ↪precision_recall_curve
from sklearn.preprocessing import Imputer
import statsmodels.api as sm
from scipy.interpolate import interp1d, interp2d
```

[2]:
```
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
```

1.1 Get Data

[3]:
```
with pd.HDFStore('../data/assets.h5') as store:
    print(store.info())
    prices = store['quandl/wiki/prices'].adj_close.unstack('ticker')
    stocks = store['us_equities/stocks']
```

```
<class 'pandas.io.pytables.HDFStore'>
File path: ../data/assets.h5
/engineered_features            frame        (shape->[445640,33])
/fred/assets                    frame        (shape->[4826,5])
/quandl/wiki/prices             frame        (shape->[15389314,12])
/quandl/wiki/stocks             frame        (shape->[1,2])
/sp500/prices                   frame        (shape->[37721,5])
/sp500/stocks                   frame        (shape->[1,7])
/us_equities/stocks             frame        (shape->[6834,6])
```

[4]:
```
shared = prices.columns.intersection(stocks.index)
prices = prices.loc['2010': '2018', shared]
stocks = stocks.loc[shared, ['marketcap', 'ipoyear', 'sector']]
```

[5]:
```
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2412 entries, A to ZUMZ
Data columns (total 3 columns):
marketcap    2407 non-null float64
ipoyear      1065 non-null float64
sector       2372 non-null object
dtypes: float64(2), object(1)
memory usage: 75.4+ KB
```

[6]:
```
prices.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2113 entries, 2010-01-04 to 2018-03-27
Columns: 2412 entries, A to ZUMZ
dtypes: float64(2412)
memory usage: 38.9 MB
```

1.1.1 Create monthly return series

Remove outliers

[8]:
```
returns = prices.resample('M').last().pct_change().stack().swaplevel()
returns = (returns[returns.between(left=returns.quantile(.05),
                                   right=returns.quantile(.95))].
 ↪to_frame('returns'))
```

1.1.2 Lagged Returns

[9]:
```
for t in range(1, 13):
    returns[f't-{t}'] = returns.groupby(level='ticker').returns.shift(t)
returns = returns.dropna()
```

1.1.3 Time Period Dummies

[10]:
```
# returns = returns.reset_index('date')
dates = returns.index.get_level_values('date')
returns['year'] = dates.year
returns['month'] = dates.month
returns = pd.get_dummies(returns, columns=['year', 'month'])
```

[11]:
```
returns.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 171162 entries, (AAON, 2011-02-28 00:00:00) to (ZTS, 2018-03-31
00:00:00)
Data columns (total 33 columns):
returns      171162 non-null float64
t-1          171162 non-null float64
t-2          171162 non-null float64
t-3          171162 non-null float64
t-4          171162 non-null float64
t-5          171162 non-null float64
t-6          171162 non-null float64
t-7          171162 non-null float64
t-8          171162 non-null float64
t-9          171162 non-null float64
t-10         171162 non-null float64
t-11         171162 non-null float64
t-12         171162 non-null float64
year_2011    171162 non-null uint8
year_2012    171162 non-null uint8
year_2013    171162 non-null uint8
year_2014    171162 non-null uint8
year_2015    171162 non-null uint8
year_2016    171162 non-null uint8
year_2017    171162 non-null uint8
year_2018    171162 non-null uint8
month_1      171162 non-null uint8
month_2      171162 non-null uint8
month_3      171162 non-null uint8
month_4      171162 non-null uint8
month_5      171162 non-null uint8
month_6      171162 non-null uint8
month_7      171162 non-null uint8
```

```
month_8      171162 non-null uint8
month_9      171162 non-null uint8
month_10     171162 non-null uint8
month_11     171162 non-null uint8
month_12     171162 non-null uint8
dtypes: float64(13), uint8(20)
memory usage: 20.7+ MB
```

1.1.4 Get stock characteristics

Create age proxy

[12]:
```
stocks['age'] = pd.qcut(stocks.ipoyear, q=5, labels=list(range(1, 6))).
 ↪astype(float).fillna(0).astype(int)
stocks = stocks.drop('ipoyear', axis=1)
```

Create size proxy

[15]:
```
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2412 entries, A to ZUMZ
Data columns (total 3 columns):
marketcap    2407 non-null float64
sector       2372 non-null object
age          2412 non-null int64
dtypes: float64(1), int64(1), object(1)
memory usage: 155.4+ KB
```

[16]:
```
stocks.marketcap.head()
```

[16]:
```
ticker
A       1.960000e+10
AA      8.540000e+09
AAL     1.767000e+10
AAMC    1.104900e+08
AAN     3.200000e+09
Name: marketcap, dtype: float64
```

[18]:
```
stocks['size'] = pd.qcut(stocks.marketcap, q=10, labels=list(range(1, 11)))
stocks = stocks.drop(['marketcap'], axis=1)
```

Create Dummy variables

[19]:
```
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2412 entries, A to ZUMZ
Data columns (total 3 columns):
```

```
sector    2372 non-null object
age       2412 non-null int64
size      2407 non-null category
dtypes: category(1), int64(1), object(1)
memory usage: 139.3+ KB
```

[20]:
```
stocks = pd.get_dummies(stocks,
                        columns=['size', 'age',  'sector'],
                        prefix=['size', 'age', ''],
                        prefix_sep=['_', '_', ''])
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2412 entries, A to ZUMZ
Data columns (total 28 columns):
size_1                    2412 non-null uint8
size_2                    2412 non-null uint8
size_3                    2412 non-null uint8
size_4                    2412 non-null uint8
size_5                    2412 non-null uint8
size_6                    2412 non-null uint8
size_7                    2412 non-null uint8
size_8                    2412 non-null uint8
size_9                    2412 non-null uint8
size_10                   2412 non-null uint8
age_0                     2412 non-null uint8
age_1                     2412 non-null uint8
age_2                     2412 non-null uint8
age_3                     2412 non-null uint8
age_4                     2412 non-null uint8
age_5                     2412 non-null uint8
Basic Industries          2412 non-null uint8
Capital Goods             2412 non-null uint8
Consumer Durables         2412 non-null uint8
Consumer Non-Durables     2412 non-null uint8
Consumer Services         2412 non-null uint8
Energy                    2412 non-null uint8
Finance                   2412 non-null uint8
Health Care               2412 non-null uint8
Miscellaneous             2412 non-null uint8
Public Utilities          2412 non-null uint8
Technology                2412 non-null uint8
Transportation            2412 non-null uint8
dtypes: uint8(28)
memory usage: 164.8+ KB
```

1.1.5 Combine data

[21]:
```
data = (returns
        .reset_index('date')
        .merge(stocks, left_index=True, right_index=True)
        .dropna()
        .set_index('date', append=True))

s = len(returns.columns)
data.iloc[:, s:] = data.iloc[:, s:].astype(int).apply(pd.to_numeric,␣
 ↪downcast='integer')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 171162 entries, (A, 2011-03-31 00:00:00) to (ZUMZ, 2018-02-28
00:00:00)
Data columns (total 61 columns):
returns                  171162 non-null float64
t-1                      171162 non-null float64
t-2                      171162 non-null float64
t-3                      171162 non-null float64
t-4                      171162 non-null float64
t-5                      171162 non-null float64
t-6                      171162 non-null float64
t-7                      171162 non-null float64
t-8                      171162 non-null float64
t-9                      171162 non-null float64
t-10                     171162 non-null float64
t-11                     171162 non-null float64
t-12                     171162 non-null float64
year_2011                171162 non-null uint8
year_2012                171162 non-null uint8
year_2013                171162 non-null uint8
year_2014                171162 non-null uint8
year_2015                171162 non-null uint8
year_2016                171162 non-null uint8
year_2017                171162 non-null uint8
year_2018                171162 non-null uint8
month_1                  171162 non-null uint8
month_2                  171162 non-null uint8
month_3                  171162 non-null uint8
month_4                  171162 non-null uint8
month_5                  171162 non-null uint8
month_6                  171162 non-null uint8
month_7                  171162 non-null uint8
month_8                  171162 non-null uint8
month_9                  171162 non-null uint8
month_10                 171162 non-null uint8
```

```
month_11                171162 non-null uint8
month_12                171162 non-null uint8
size_1                  171162 non-null int8
size_2                  171162 non-null int8
size_3                  171162 non-null int8
size_4                  171162 non-null int8
size_5                  171162 non-null int8
size_6                  171162 non-null int8
size_7                  171162 non-null int8
size_8                  171162 non-null int8
size_9                  171162 non-null int8
size_10                 171162 non-null int8
age_0                   171162 non-null int8
age_1                   171162 non-null int8
age_2                   171162 non-null int8
age_3                   171162 non-null int8
age_4                   171162 non-null int8
age_5                   171162 non-null int8
Basic Industries        171162 non-null int8
Capital Goods           171162 non-null int8
Consumer Durables       171162 non-null int8
Consumer Non-Durables   171162 non-null int8
Consumer Services       171162 non-null int8
Energy                  171162 non-null int8
Finance                 171162 non-null int8
Health Care             171162 non-null int8
Miscellaneous           171162 non-null int8
Public Utilities        171162 non-null int8
Technology              171162 non-null int8
Transportation          171162 non-null int8
dtypes: float64(13), int8(28), uint8(20)
memory usage: 25.3+ MB
```

1.1.6 Store data

[22]:
```
with pd.HDFStore('data.h5') as store:
    store.put('data', data)
```