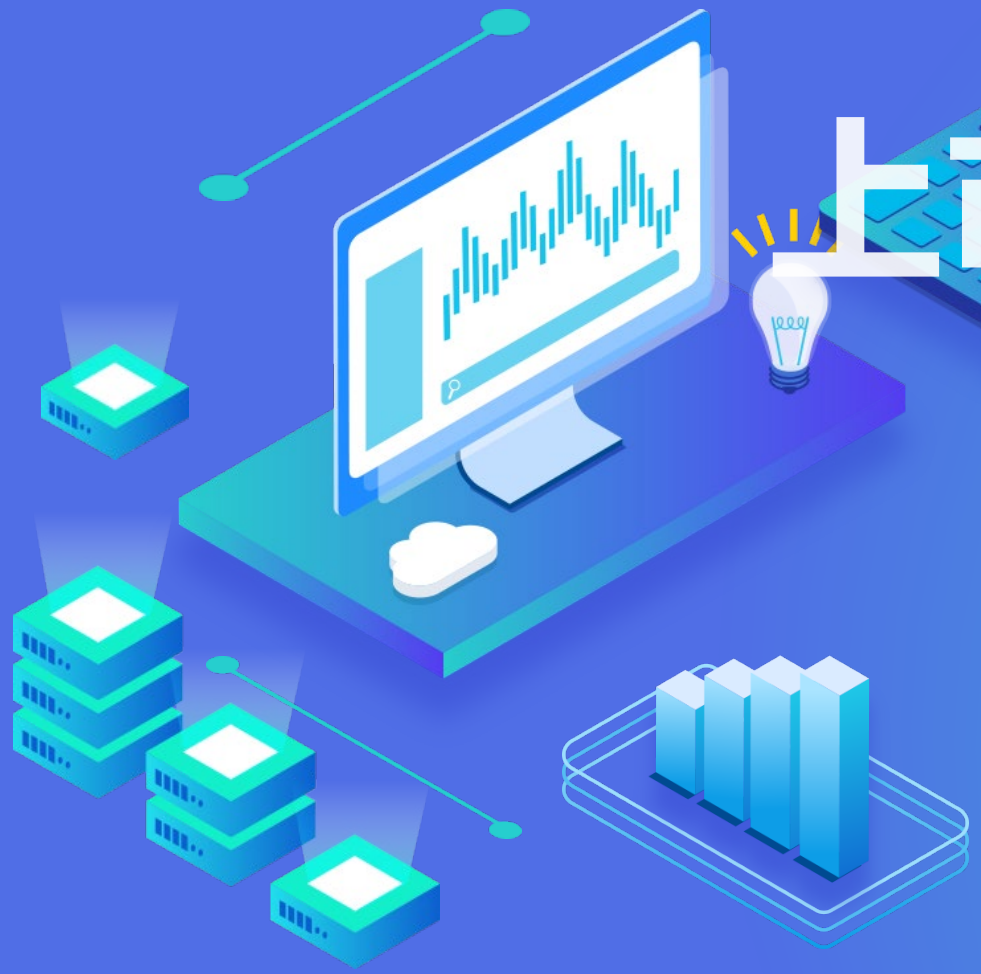
DigQuant



上市公司财务指标与股价的关系

DigQuant
点宽

如何找出与股价息息相关的财务指标?

我们给出各上市公司的各类财务数据，如何判断那些财务指标与股价相关程度最高呢?

证券代码	年份	每股收益EPS(基本)	每股收益EPS(稀释)	每股收益EPS(期末股本摊薄)	每股收益EPS(最新股本摊薄)	每股收益EPS(扣除/基本)	每股收益EPS(扣除/稀释)	每股收益EPS(扣除/期末股本摊薄)	每股收益EPS(扣除/最新股本摊薄)	每股收益EPS(TTM)	每股收益EPS(最新公告)	每股净资产BPS
8425	2015	1.56	1.56	1.53	1.13	1.56	1.56	1.53	1.13	1.53	1.49	11.29
8423	2015	1.64	1.64	1.64	1.56	1.6	1.6	1.59	1.52	1.46	1.71	9.08
8421	2015	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.14	0.35	0.97
8419	2015	-0.06	-0.06	-0.05	-0.05	-0.06	-0.06	-0.06	-0.06	0.04	-0.03	1.09
8417	2015	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.31	0.4	0.38	3.25
8415	2015	0.06	0.06	0.06	0.04	0.05	0.05	0.05	0.03	-0.15	-0.13	1.6
8413	2015	0.08	0.08	0.08	0.07	0.07	0.07	0.07	0.06	0.04	-0.09	1.21
8411	2015	0.5	0.5	0.5	0.31	0	0	0	0	0.45	0.2	2.81
8409	2015	0.03	0.03	0.03	0.03	0	0	0	0	-0.07	0.03	2.64
8407	2015	0.26	0.26	0.26	0.26	0.03	0.03	0.03	0.03	0.24	0.32	3.52
8405	2015	0.3	0.3	0.3	0.2	0.14	0.14	0.14	0.1	0.21	0.24	3.68
8403	2015	0.26	0.26	0.26	0.26	0.19	0.19	0.19	0.19	0.23	0.07	3.55
8401	2015	-0.52	-0.52	-0.52	-0.52	-0.47	-0.47	-0.47	-0.47	-0.35	0.26	1.17
8399	2015	0	0	0	0	0	0	0	0	0	0.01	0.02
8397	2015	-0.12	-0.12	-0.12	-0.03	-0.15	-0.15	-0.15	-0.04	0.13	0.27	3.11
8395	2015	-0.02	-0.02	-0.02	-0.02	-0.05	-0.05	-0.11	-0.11	0.09	0.02	1.1
8393	2015	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.3	3.54
8391	2015	0.19	0.19	0.19	0.19	0.24	0.24	0.24	0.24	0.2	-0.03	2.77
8389	2015	0.15	0.15	0.14	0.1	0.13	0.13	0.1	0.07	0.08	0.09	2.92
8387	2015	0.31	0.31	0.28	0.28	0.29		0.26	0.26	0.32	0.68	5.24
8385	2015	0.45	0.45	0.45	0.38	0.44		0.44	0.36	0.53	0.82	5.48
8383	2015	2.1	2.1	2.1	1.78	1.97	1.97	1.97	1.67	1.99	2.33	15.04

如何找出与股价息息相关的财务指标?

数学建模过程:

- 数据预处理
- 特征工程
- 模型建立
- 评估模型效果

数据预处理

- 缺失值处理
- 标准化处理
- （数据采样）

[2]:
```
#查看缺失值概况
data.isna().sum()
```

Out[2]
```
证券代码                  0
年份                    0
每股收益EPS(基本)          577
每股收益EPS(稀释)         1024
每股收益EPS(期末股本摊薄)      321
                     ...
现金净流量同比增长率           657
总资产同比增长率             658
净利润同比增长率             657
营业总成本同比增长率           657
收盘价                  4115
Length: 161, dtype: int64
```

+ Code + Markdown

[3]:
```
#数据预处理
#定义处理缺失值的函数
def delete_nan(data,percent):
    print('删除80%有缺失值的列')
    data.dropna(axis=1,how='any',thresh=percent*len(data),subset=None,inplace=True)
    print('填补缺失值为0')
    data=data.fillna(0,inplace=True)
#使用该函数删除缺失值
delete_nan(data, 0.8)
#将处理缺失值之后的数据
data.to_csv('案例数据-缺失值处理.csv',encoding = 'utf_8_sig')
```

```
删除80%有缺失值的列
填补缺失值为0
```

数据预处理

- 缺失值处理
- 标准化处理
- （数据采样）

[4]:

```
#数据预处理
#标准化
from sklearn import preprocessing
def standard(data):
    scaler = preprocessing.StandardScaler().fit(data)
    data_x_stand = scaler.transform(data)
    data_x_stand=pd.DataFrame(data_x_stand)
    data_x_stand.columns=data.columns.to_list()
    return data_x_stand

data_stand=standard(data)
print(data_stand.head())
data_stand.to_csv('案例数据-标准化处理.csv',encoding = 'utf_8_sig')
```

```
      证券代码        年份  每股收益EPS(基本)  每股收益EPS(稀释)  每股收益EPS(期末股本摊薄)  \
0  1.731533 -1.414297     1.260580     1.308325         0.365513
1  1.730711 -1.414297     1.350996     1.400881         0.402511
2  1.729889 -1.414297    -0.491222    -0.484951        -0.145737
3  1.729067 -1.414297    -0.570336    -0.565938        -0.165918
4  1.728245 -1.414297    -0.152164    -0.137866        -0.044833

   每股收益EPS(最新股本摊薄)  每股收益EPS(扣除/基本)  每股收益EPS(扣除/稀释)  每股收益EPS(扣除/期末股本摊薄)  \
0         1.153734        1.427432        1.503189           0.687174
1         1.764040        1.475458        1.552812           0.723387
2        -0.435899       -0.433579       -0.419694          -0.230225
3        -0.521058       -0.517625       -0.506534          -0.272474
4        -0.010104       -0.073384       -0.047523          -0.049160
```

特征工程

- 特征选择
- 特征降维

```
from sklearn.feature_selection import SelectKBest

# 数据集划分为特征X和标签y,y是回归
def get_flag(data):
    data_class_y=data['收盘价']
    return data_class_y

y = get_flag(data_stand)
x = data_stand.drop(['收盘价'],axis = 1)
```

+ Code + Markdown

[6]:

```
#相关性检测
from scipy.stats import pearsonr
import numpy as np

def kbest_pearson(x,y,n):
    X, y = x, y
    print(X.shape)
    # 在此定义为计算相关系数
    # 函数输入为特征矩阵和目标向量，输出为二元组（评分，P值）的数组,数组第i项为第i个特征的评分和P值
    pearsonr_way = lambda X, Y: np.array(list(map(lambda x:pearsonr(np.squeeze(x), np.squeeze(Y)), X.T))).T[0]
    #第一个参数为计算评估特征是否好的函数，该函数
    #参数k为选择的特征个数
    i = SelectKBest(pearsonr_way, k=n).fit(X, y).get_support(indices=True)
    # print(X_new.shape)
    print(i)
    #返回所属的列数
    return i

i=kbest_pearson(data_stand,y,20)
data.iloc[:,i].to_csv('kbest筛选指标pearson.csv',encoding = 'utf_8_sig')
```

```
(21067, 153)
[  2   3   5   6   7   8   9  10  11  12  13  14  22  23  24  25  26  28
  29 152]
```

模型建立

- 划分训练集和测试集
- 数据格式处理
- LSTM模型建立
- 评估模型

[8]:
```
#将数据划分为训练集和测试集
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x, y, test_size =0.1)
```

+ Code + Markdown

[9]:
```
#数据格式处理
y_test=pd.DataFrame(y_test)
y_train=pd.DataFrame(y_train)
x_train.reset_index(inplace = True)
x_test.reset_index(inplace = True)
y_train.reset_index(inplace = True)
y_test.reset_index(inplace = True)

need_num = 90
x_train_new = []
y_train_new = []
#每90个数据为一组，作为测试数据，下一个数据为标签
for i in range(need_num, x_train.shape[0]):
    x_train_new.append(x_train[i-need_num: i])
    y_train_new.append(y_train.iloc[i, 0])
#将数据转化为数组
import numpy as np
x_train_new, y_train_new = np.array(x_train), np.array(y_train)
#因为LSTM要求输入的数据格式为三维的，[training_number, time_steps, 1]，因此对数据进行相应转化
x_train_new = np.reshape(x_train_new, (x_train_new.shape[0], x_train_new.shape[1], 1))

x_train_new[0]
y_train_new[0]


need_num = 90
x_test_new = []
y_test_new = []
#每90个数据为一组，作为测试数据，下一个数据为标签
for i in range(need_num, x_test.shape[0]):
    x_test_new.append(x_test[i-need_num: i])
    y_test_new.append(y_test.iloc[i, 0])
#将数据转化为数组
import numpy as np
x_test_new, y_test_new = np.array(x_test), np.array(y_test)
#因为LSTM要求输入的数据格式为三维的，[training_number, time_steps, 1]，因此对数据进行相应转化
x_test_new = np.reshape(x_test_new, (x_test_new.shape[0], x_test_new.shape[1], 1))
```