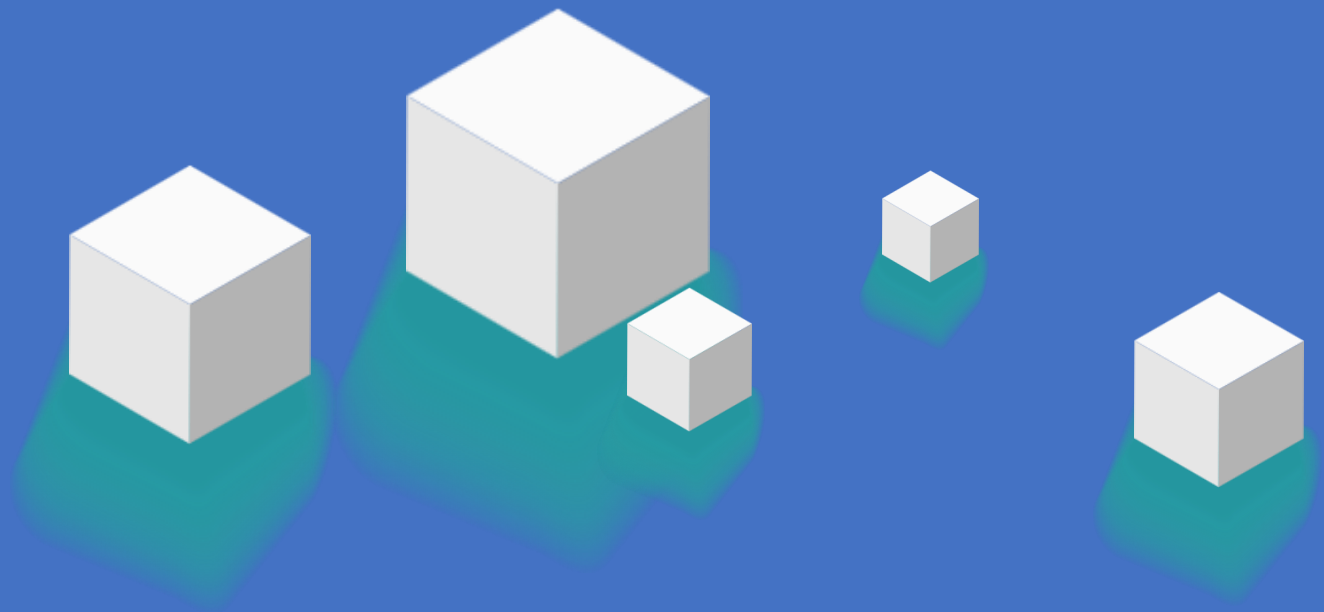
时间序列分析



时间序列模型

你认为金融交易有时间上的规律么？

案例：沪市指数预测

Case: Shanghai Stock Index Forecast

CASE：沪市指数预测

- 沪市指数的历史数据（从1990年12月19日到2024年6月30日），shanghai_index_1990_12_19_to_2024_06_30.csv
- 请你对沪市指数未来3个月（截止到2024年9月30日）的变化进行预测

1990年到2024年沪市指数走势

什么是时间序列模型

What is a time series model

时间序列：

- 建立了观察结果与时间变化的关系，能帮预测未来一段时间内的结果变化情况

时间序列分析与回归分析的区别：

- 在选择模型前，我们需要确定结果与变量之间的关系。回归分析训练得到的是目标变量y与自变量x（一个或多个）的相关性，然后通过新的自变量x来预测目标变量y。而时间序列分析得到的是目标变量y与时间的相关性
- 回归分析擅长的是多变量与目标结果之间的分析，即便是单一变量，也往往与时间无关。而时间序列分析建立在时间变化的基础上，它会分析目标变量的趋势、周期、时期和不稳定因素等。这些趋势和周期都是在时间维度的基础上，是我们要观察的重要特征

1990年到2024年沪市指数走势

时间序列分析

Time Series Analysis

时间序列：

- 按照时间顺序组成的数字序列
- 历史悠久，在中国古代的农业社会中，人们就将一年中不同时间节点和天气的规律总结了下来，形成了二十四节气，也就是从时间序列中观察天气和太阳的规律（只是当时没有时间序列模型和相应工具）
- 时间序列在金融、经济、商业领域拥有广泛的应用
- 机器学习模型，包括AR、 MA、 ARMA、 ARIMA
- 神经网络，时序大模型都可以进行时间序列预测

时间序列分析是金融时序数据中很重要的工具，了解方法后续执行可由技术或通过AI大模型辅助进行

沪市指数走势

时间序列模型

Time Series Analysis

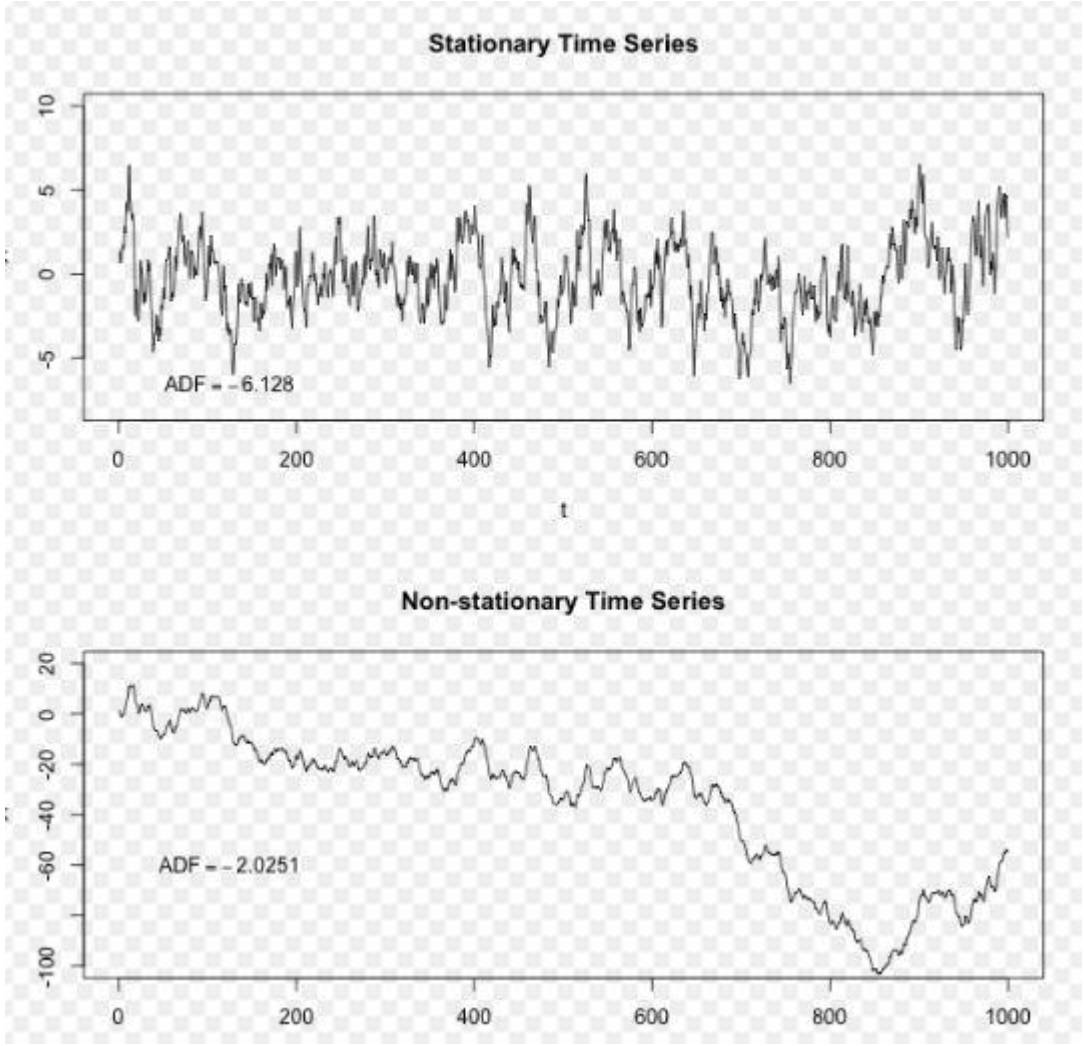
时间序列及分解：

- 平稳序列，stationary series

基本上不存在趋势（Trend）的序列，各观察值基本上在某个固定的水平上波动

- 非平稳序列，non-stationary series

包含趋势、季节性或周期性的序列，可以只有一种成分，也可能是多种成分的组合



时间序列分析

Time Series Analysis

时间序列及分解：

- 趋势（trend）：时间序列在长时期内呈现出来的某种持续上升或持续下降的变动，也称长期趋势
- 季节性（seasonality）：时间序列在一年内重复出现的周期波动。销售旺季，销售淡季，旅游旺季、旅游淡季

季节，可以是任何一种周期性变化，不一定是一年中的四季

含有季节成分的序列可能含有趋势，也可能不含有趋势

- 周期性（cyclicity）：通常是由经济环境的变化引起

不同于趋势变动，不是朝着单一方向的持续运动，而是涨落相间的交替波动

不同于季节变动，季节变动有比较固定的规律，变动周期大多为一年。周期性的循环波动无固定规律，变动周期多在一年以上，且周期长短不一

- 随机性（Irregular），指受偶然因素影响所形成的的不规则波动，在时间序列中无法预估

随机性是不规则波动，除去趋势、周期性、季节性的偶然性波动

因素	举例
长期趋势 Trend(T)	国内生产总值
季节变动 Season(S)	冰淇淋、暖宝宝、羽绒服、裙子等销售
周期性 Cyclic(C)	太阳黑子数量变化
随机性 Irregular(I)	股票市场受到突然的利好、利空等信息的影响，影响股价产生的波动

时间序列工具（statsmodels）

Time Series Tools (statsmodels)

statsmodels工具：

- statsmodels工具包提供统计计算，包括描述性统计以及统计模型的估计和推断

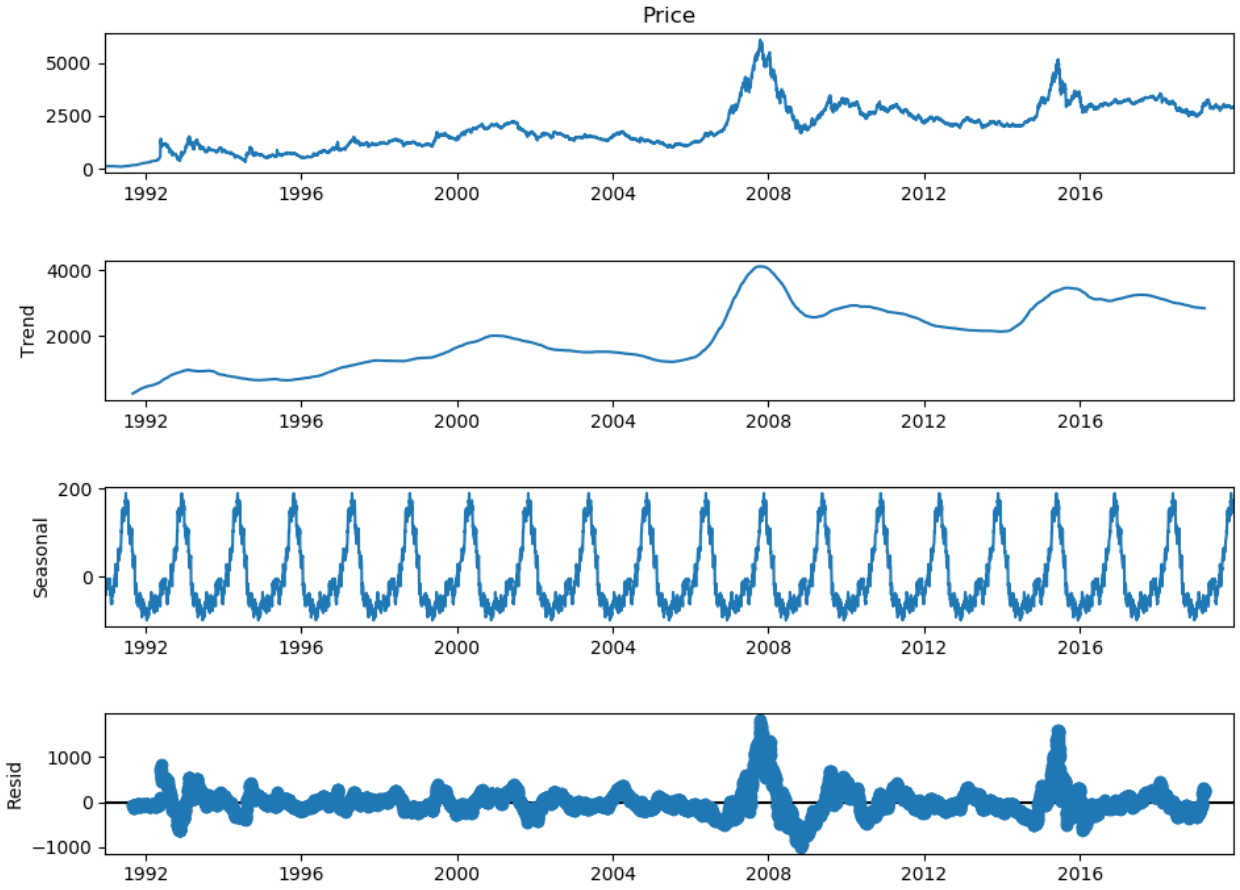
statsmodels主要包括如下子模块：

- 回归模型：线性回归，广义线性模型，线性混合效应模
- 方差分析（ANOVA）
- 时间序列分析：AR，ARMA，ARIMA等

时间序列工具（statsmodels）

Time Series Tools (statsmodels)

```
import statsmodels.api as sm

# 数据加载

data = pd.read_csv('shanghai_index_1990_12_19_to_2020_03_12.csv',
usecols=['Timestamp', 'Price'])

data.Timestamp = pd.to_datetime(data.Timestamp)

data = data.set_index('Timestamp')

data['Price'] = data['Price'].apply(pd.to_numeric, errors='ignore')

# 进行线性插补缺漏值

data.Price.interpolate(inplace=True)

# 返回三个部分 trend（趋势），seasonal（季节性）和residual (残留

result = sm.tsa.seasonal_decompose(data.Price, period=250)

result.plot()

plt.show()
```



时间序列模型

Time Series Model

AR模型：

- Auto Regressive，中文叫自回归模型
- 认为过去若干时刻的点通过线性组合，再加上白噪声就可以预测未来某个时刻的点
- 日常生活环境中就存在白噪声，在数据挖掘的过程中，可以把它理解为一个期望为0，方差为常数的纯随机过程
- AR模型存在一个阶数p，称为AR(p)模型，也叫作p阶自回归模型。指的是通过这个时刻点的前p个点，通过线性组合再加上白噪声来预测当前时刻点的值
- AR是线性时间序列分析模型中最简单的模型，通过前面部分的数据与后面部分的数据之间的相关关系来建立回归方程：

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + u_t$$

AR(p)，表示p阶的自回归过程，ϕ 为自回归系数

u_t 表示白噪声，是时间序列中的数值的随机波动。这些波动会相互抵消，即累计为0

- 如果只有一个时间记录点时，则为AR(1)，即一阶自回归过程：

$$x_t = \phi_1 x_{t-1} + u_t$$

时间序列模型

Time Series Model

MA模型：

- Moving Average，中文叫做滑动平均模型
- 与AR模型大同小异，AR模型是历史时序值的线性组合，MA是通过历史白噪声进行线性组合来影响当前时刻点
- MA模型中的历史白噪声是通过影响历史时序值，从而间接影响到当前时刻点的预测值
- MA模型存在一个阶数q，称为MA(q)模型，也叫作q阶移动平均模型
- AR和MA模型都存在阶数，在AR模型中，用p表示，在MA模型中用q表示，这两个模型大同小异，与AR模型不同的是MA模型是历史白噪声的线性组合
- MA模型，通过前面通过将一段时间序列中白噪声序列进行加权和，可以得到移动平均方程：

$$x_t = u_t + \phi_1 u_{t-1} + \phi_2 u_{t-2} + \ldots + \phi_q u_{t-q}$$

- MA(q)表示q阶移动平均过程，ϕ为移动回归系数，u_t 为不同时间点的白噪声
- Xt 为第t天的股票价格，而Ut为第t天的新闻影响，当天的股票价格受当天的新闻影响，也受昨天的新闻影响（但影响力要弱些，所以要乘上系数）

时间序列模型

Time Series Model

ARMA模型：

- Auto Regressive Moving Average，中文叫做自回归滑动平均模型
- AR模型和MA模型的混合，相比AR模型和MA模型，它有更准确的估计
- ARMA模型存在p和q两个阶数，称为ARMA(p,q)模型：

$$x_t = u_t + \phi_1 u_{t-1} + \phi_2 u_{t-2} + \ldots + \phi_q u_{t-q} + \vartheta_1 x_{t-1} + \vartheta_2 x_{t-2} + \ldots + \vartheta_p x_{t-p}$$

- 自回归模型结合了两个模型的特点，AR解决当前数据与后期数据之间的关系，MA则可以解决随机变动，即噪声问题

时间序列模型

Time Series Model

ARIMA模型：

- Auto Regressive Integrated Moving Average模型，中文叫差分自回归滑动平均模型，也叫求合自回归滑动平均模型
- 相比于ARMA，ARIMA多了一个差分的过程，作用是对不平稳数据进行差分平稳，在差分平稳后再进行建模
- ARIMA的原理和ARMA模型一样。相比于ARMA(p,q)的两个阶数，ARIMA是一个三元组的阶数(p,d,q)，称为ARIMA(p,d,q)模型，其中d是差分阶数
- AR，MA是ARMA的特殊形式，而ARMA是ARIMA的特殊形式

ARIMA模型步骤：

- Step1，观察时间序列数据，是否为平稳序列
- Step2，对于非平稳时间序列要先进行d阶差分运算，化为平稳时间序列
- Step3，使用ARIMA（p, d, q）模型进行训练拟合，找到最优的(p, d, q)，及训练好的模型
- Step4，使用训练好的ARIMA模型进行预测，并对差分进行还原

ARIMA 用差分将不平稳数据先变得平稳，再用ARMA模型

时间序列模型

Time Series Model

关于差分：

- 差分=序列之间做差值，目的是为了得到平稳的序列，也就是去掉前面数值的影响
- 一次差分为序列之间做一次差值，二次差分为在一次差分的基础上在做一次差分
- $f(x) = x^2$ 若x=[1,4,9,16,25….]（x有二次趋势）

一次差分的结果为[4-1,9-4,16-9,25-16…] = [3,5,7,9,11…]，此时x序列仍不平稳，有一次上升的趋势

再做一次差分为[2,2,2,2…]，此时x为平稳序列

ARMA工具

Time Series Tool ARMA

ARMA工具：

- from statsmodels.tsa.arima_model import ARMA
- ARMA(endog,order,exog=None)
- endog：endogenous variable，代表内生变量，又叫非政策性变量，它是由模型决定的，不被政策左右，可以说是我们想要分析的变量，或者说是我们这次项目中需要用到的变量
- order：代表是p和q的值，也就是ARMA中的阶数
- exog：exogenous variables，代表外生变量。外生变量和内生变量一样是经济模型中的两个重要变量。相对于内生变量而言，外生变量又称作为政策性变量，在经济机制内受外部因素的影响，不是我们模型要研究的变量

如果我们想要创建ARMA(7,0)模型，可以写成：

ARMA(data,(7,0))，其中data是我们想要观察的变量，(7,0)代表(p,q)的阶数。

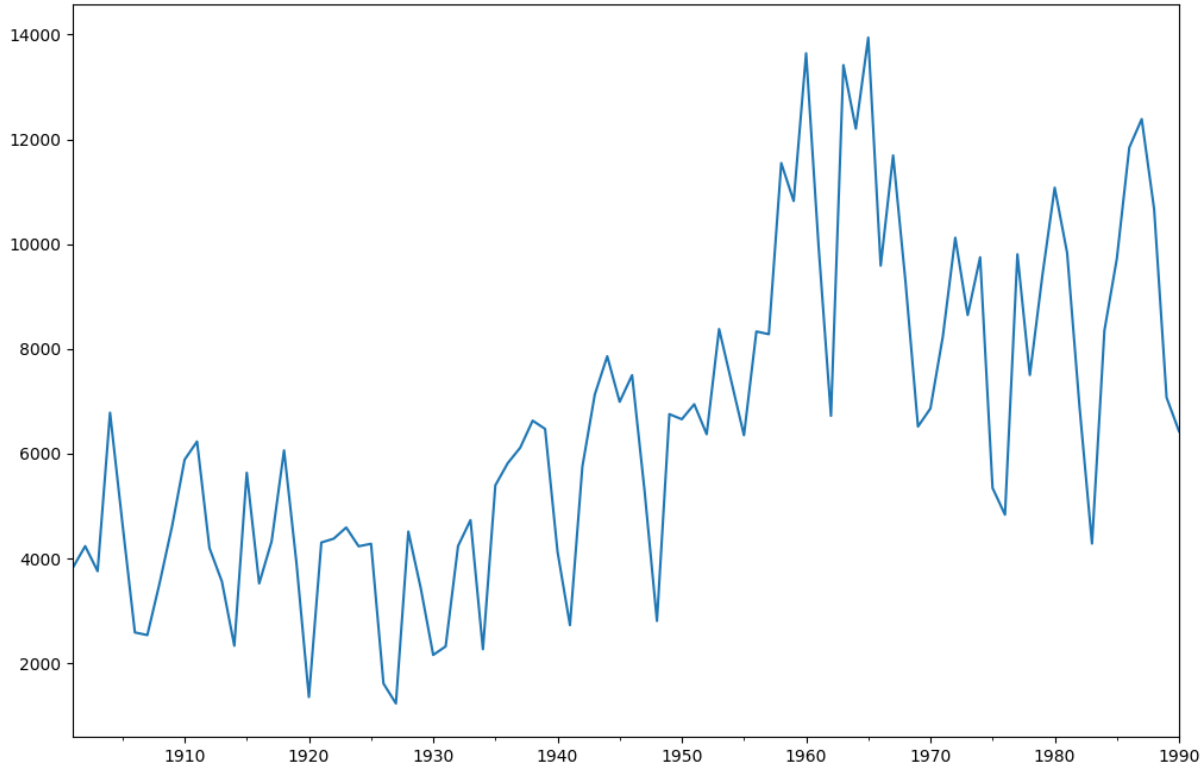
fit函数，进行拟合

predict(start, end)函数，进行预测，其中start为预测的起始时间，end为预测的终止时间

ARMA工具

Time Series Tool ARMA

```
# 用ARMA进行时间序列预测

from statsmodels.tsa.arima_model import ARMA

# 创建数据

data = [3821, 4236, 3758, 6783, 4664, 2589, 2538, 3542, 4626, 5886, 6233,
4199, 3561, 2335, 5636, 3524, 4327, 6064, 3912, 1356, 4305, 4379, 4592,
4233, 4281, 1613, 1233, 4514, 3431, 2159, 2322, 4239, 4733, 2268, 5397,
5821, 6115, 6631, 6474, 4134, 2728, 5753, 7130, 7860, 6991, 7499, 5301,
2808, 6755, 6658, 6944, 6372, 8380, 7366, 6352, 8333, 8281, 11548, 10823,
13642, 9973, 6723, 13416, 12205, 13942, 9590, 11693, 9276, 6519, 6863,
8237, 10122, 8646, 9749, 5346, 4836, 9806, 7502, 9387, 11078, 9832, 6886,
4285, 8351, 9725, 11844, 12387, 10666, 7072, 6429]

data=pd.Series(data)

data_index = sm.tsa.datetools.dates_from_range('1901','1990')
```

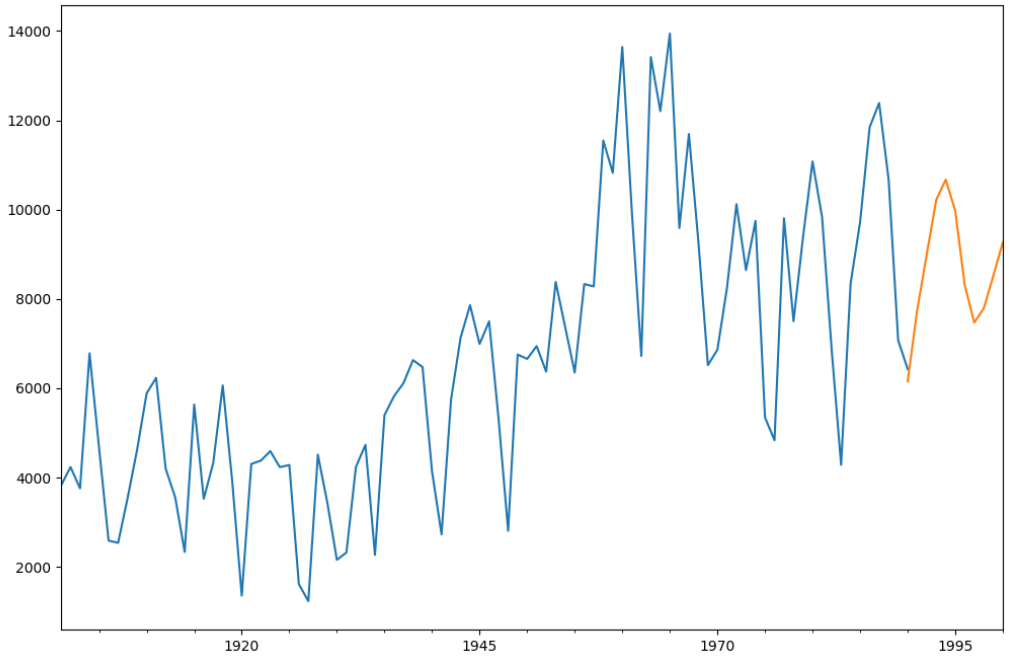


ARMA工具

Time Series Tool ARMA

```
# 绘制数据图

data.index = pd.Index(data_index)

data.plot(figsize=(12,8))

plt.show()

# 创建ARMA模型# 创建ARMA模型

arma = ARMA(data,(7,0)).fit()

print('AIC: %0.4lf' %arma.aic)

# 模型预测

predict_y = arma.predict('1990', '2000')

# 预测结果绘制

fig, ax = plt.subplots(figsize=(12, 8))

ax = data.ix['1901':].plot(ax=ax)

predict_y.plot(ax=ax)

plt.show()
```

```
At iterate   35    f=  8.87823D+00    |proj g|=  6.55707D-03

At iterate   40    f=  8.87776D+00    |proj g|=  1.28608D-04

At iterate   45    f=  8.87776D+00    |proj g|=  1.24345D-06

At iterate   50    f=  8.87776D+00    |proj g|=  1.08358D-05

At iterate   55    f=  8.87776D+00    |proj g|=  5.32907D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    8     58     89      1     0     0   1.243D-06   8.878D+00
  F =   8.8777593158679711

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

 Warning:  more than 10 function and gradient
   evaluations in the last line search.  Termination
   may possibly be caused by a bad search direction.

 Cauchy                time 0.000E+00 seconds.
 Subspace minimization time 0.000E+00 seconds.
 Line search           time 0.000E+00 seconds.

 Total User time 0.000E+00 seconds.

AIC: 1615.9967
```



AIC准则，也叫作赤池消息准则，是衡量统计模型拟合好坏的一个标准，数值越小代表模型拟合得越好

案例：沪市指数预测（ARMA）

Case: Shanghai Stock Index Forecast (ARMA)

使用ARMA工具对沪市指数进行预测：

- Step1，数据加载&探索

按照不同的时间尺度（天，月，季度，年）可以将数据压缩，得到不同尺度的数据，然后做可视化呈现。

df_month = df.resample('M').mean()

- Step2，模型选择&训练，在给定范围内，选择最优的超参数

创建ARMA时间序列模型。我们并不知道p和q取什么值时，可以给它们设置一个区间范围，比如都是range(0,3)，然后计算不同模型的AIC数值，选择最小的AIC数值对应的那个ARMA模型

- Step3，模型预测，可视化呈现

用这个最优的ARMA模型预测未来3个月的沪市指数走势，并将结果做可视化呈现。

1990年到2024年沪市指数走势

案例：沪市指数预测（ARMA）

Case: Shanghai Stock Index Forecast (ARMA)

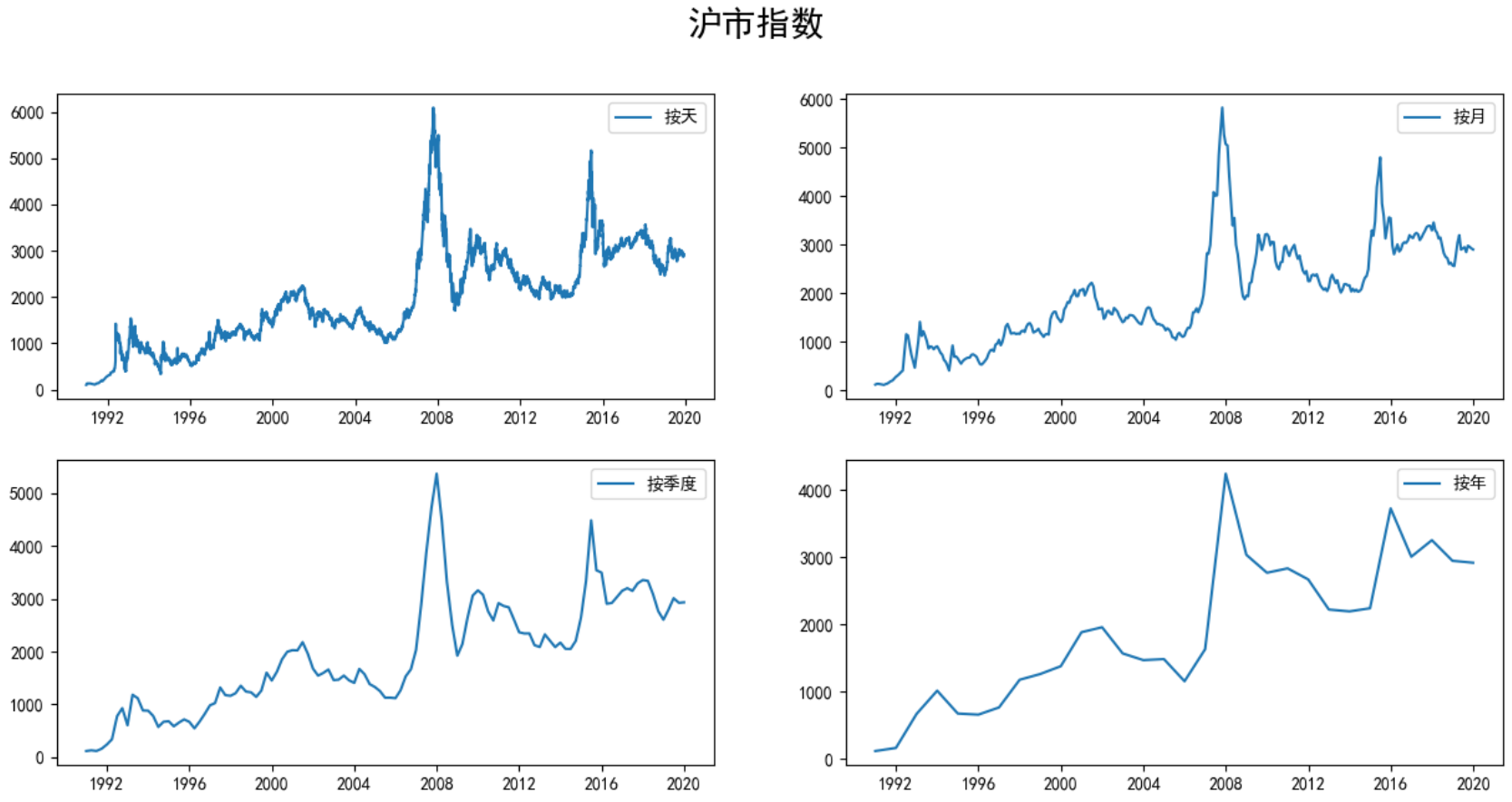
```
# 数据加载
df = pd.read_csv('./shanghai_index_1990_12_19_to_2024_06_30.csv')
df = df[['Timestamp', 'Price']]
# 将时间作为df的索引
df.Timestamp = pd.to_datetime(df.Timestamp)
df.index = df.Timestamp
# 数据探索
print(df.head())
# 按照月，季度，年来统计
df_month = df.resample('M').mean()
print(df_month)
df_Q = df.resample('Q-DEC').mean()
df_year = df.resample('A-DEC').mean()
```

```
            Timestamp   Price
Timestamp
1990-12-19 1990-12-19   99.98
1990-12-20 1990-12-20  104.39
1990-12-21 1990-12-21  109.13
1990-12-24 1990-12-24  114.55
1990-12-25 1990-12-25  120.25
                Price
Timestamp
1990-12-31   116.990000
1991-01-31   132.628182
1991-02-28   131.887778
1991-03-31   126.011429
1991-04-30   118.426818
...                 ...
2019-08-31  2847.063264
2019-09-30  2978.383790
2019-10-31  2954.832456
2019-11-30  2923.774700
2019-12-31  2900.789287

[349 rows x 1 columns]
```

案例：沪市指数预测（ARMA）

Case: Shanghai Stock Index Forecast (ARMA)

```
# 按照天，月，季度，年来显示沪市指数的走势

fig = plt.figure(figsize=[15, 7])

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签

plt.suptitle('沪市指数', fontsize=20)

plt.subplot(221)

plt.plot(df.Price, '-', label='按天')

plt.legend()

plt.subplot(222)

plt.plot(df_month.Price, '-', label='按月')

……

plt.plot(df_Q.Price, '-', label='按季度')

……

plt.plot(df_year.Price, '-', label='按年')

plt.legend()

plt.show()
```



案例：沪市指数预测（ARMA）

Case: Shanghai Stock Index Forecast (ARMA)

```
# 设置参数范围
ps = range(0, 3)
qs = range(0, 3)
parameters = product(ps, qs)
parameters_list = list(parameters)
# 寻找最优ARMA模型参数，即best_aic最小
results = []
best_aic = float("inf") # 正无穷
for param in parameters_list:
    try:
        model = ARMA(df_month.Price,order=(param[0], param[1])).fit()
    except ValueError:
        print('参数错误:', param)
        continue
    aic = model.aic
    if aic < best_aic:
        best_model = model
        best_aic = aic
        best_param = param
    results.append([param, model.aic])
print('最优模型: ', best_model.summary())
```

最优模型:

ARMA Model Results

Dep. Variable:	Price	No. Observations:	349
Model:	ARMA(2, 2)	Log Likelihood	-2271.637
Method:	css-mle	S.D. of innovations	161.441
Date:	Thu, 12 Dec 2019	AIC	4555.275
Time:	16:31:17	BIC	4578.405
Sample:	12-31-1990 - 12-31-2019	HQIC	4564.482

	coef	std err	z	P>|z|	[0.025	0.975]
const	1873.2337	491.673	3.810	0.000	909.573	2836.894
ar.L1.Price	0.4987	0.130	3.844	0.000	0.244	0.753
ar.L2.Price	0.4659	0.128	3.634	0.000	0.215	0.717
ma.L1.Price	0.8587	0.123	7.000	0.000	0.618	1.099
ma.L2.Price	0.3721	0.062	5.979	0.000	0.250	0.494

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0245	+0.0000j	1.0245	0.0000
AR.2	-2.0949	+0.0000j	2.0949	0.5000
MA.1	-1.1540	-1.1644j	1.6394	-0.3743
MA.2	-1.1540	+1.1644j	1.6394	0.3743

案例：沪市指数预测（ARMA）

Case: Shanghai Stock Index Forecast (ARMA)

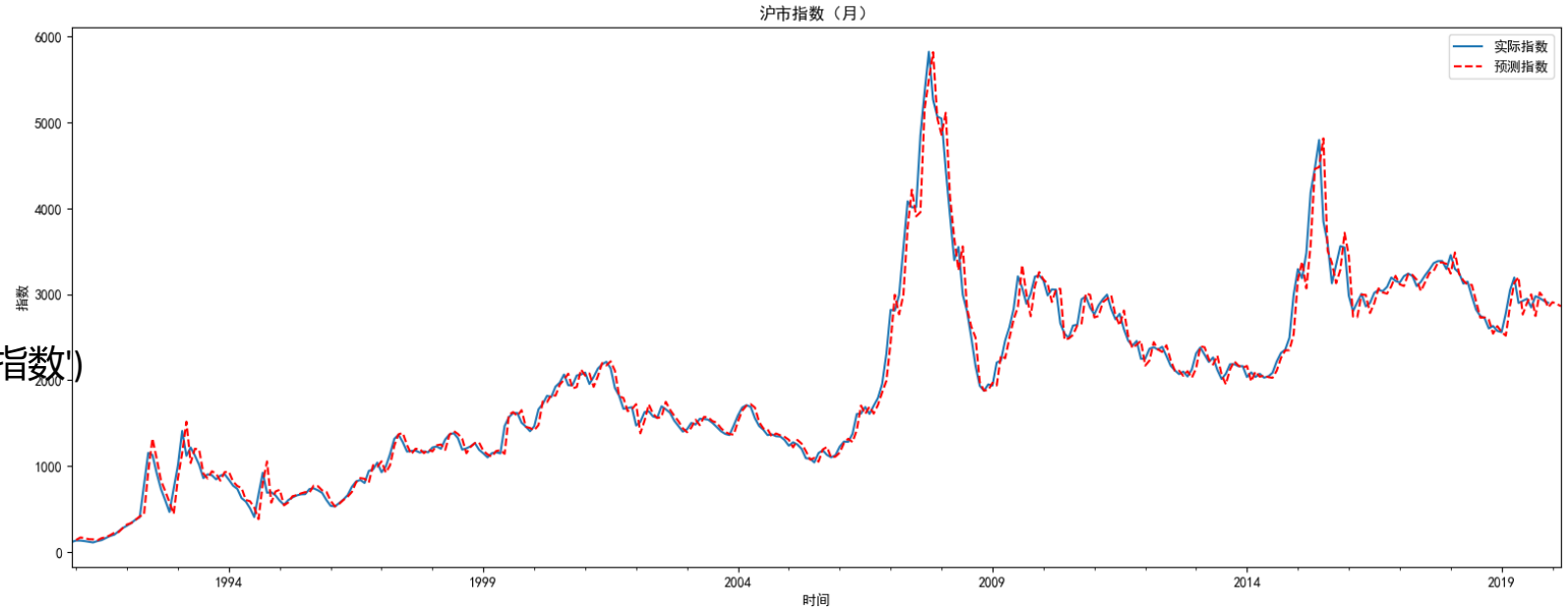
```
# 设置future_month，需要预测的时间date_list
future_month = 3
last_month = pd.to_datetime(df_month2.index[len(df_month2)-1])
date_list = []
for i in range(future_month):
    # 计算下个月有多少天
    year = last_month.year
    month = last_month.month
    if month == 12:
        month = 1
        year = year+1
    else:
        month = month + 1
    next_month_days = calendar.monthrange(year, month)[1]
    #print(next_month_days)
    last_month = last_month + timedelta(days=next_month_days)
    date_list.append(last_month)
print('date_list=', date_list)
```

```
date_list= [Timestamp('2020-04-30 00:00:00', freq='M'), Timestamp('2020-05-31 00:00:00', freq='M'), Timestamp('2020-06-30 00:00:00', freq='M')]
```

案例：沪市指数预测（ARMA）

Case: Shanghai Stock Index Forecast (ARMA)

```
# 添加未来要预测的3个月
future = pd.DataFrame(index=date_list, columns= df_month.columns)
df_month2 = pd.concat([df_month2, future])
df_month2['forecast'] = best_model.predict(start=0, end=len(df_month2))
# 第一个元素不正确，设置为NaN
df_month2['forecast'][0] = np.NaN
print(df_month2)
# 沪市指数预测结果显示
plt.figure(figsize=(30,7))
df_month2.Price.plot(label='实际指数')
df_month2.forecast.plot(color='r', ls='--', label='预测指数')
plt.legend()
plt.title('沪市指数（月）')
plt.xlabel('时间')
plt.ylabel('指数')
plt.show()
```



案例：沪市指数预测（ARMA）

Case: Shanghai Stock Index Forecast (ARMA)

```
# 设置参数范围
ps = range(0, 5)
qs = range(0, 5)
ds = range(1, 2) #使用1阶差分
for param in parameters_list:
    try:
        model = sm.tsa.statespace.SARIMAX(df_month.Price,
                        order=(param[0], param[1], param[2]),
                        enforce_stationarity=False,
                        enforce_invertibility=False).fit()
    except ValueError:
        print('参数错误:', param)
        continue
    aic = model.aic
    if aic < best_aic:
    ……
# 输出最优模型
print('最优模型: ', best_model.summary())
```

最优模型:　　Statespace Model Results

Dep. Variable:	Price	No. Observations:	349
Model:	SARIMAX(1, 1, 4)	Log Likelihood	-2230.429
Date:	Sat, 14 Dec 2019	AIC	4472.859
Time:	22:15:50	BIC	4495.885
Sample:	12-31-1990	HQIC	4482.031
	- 12-31-2019		
Covariance Type:	opg		

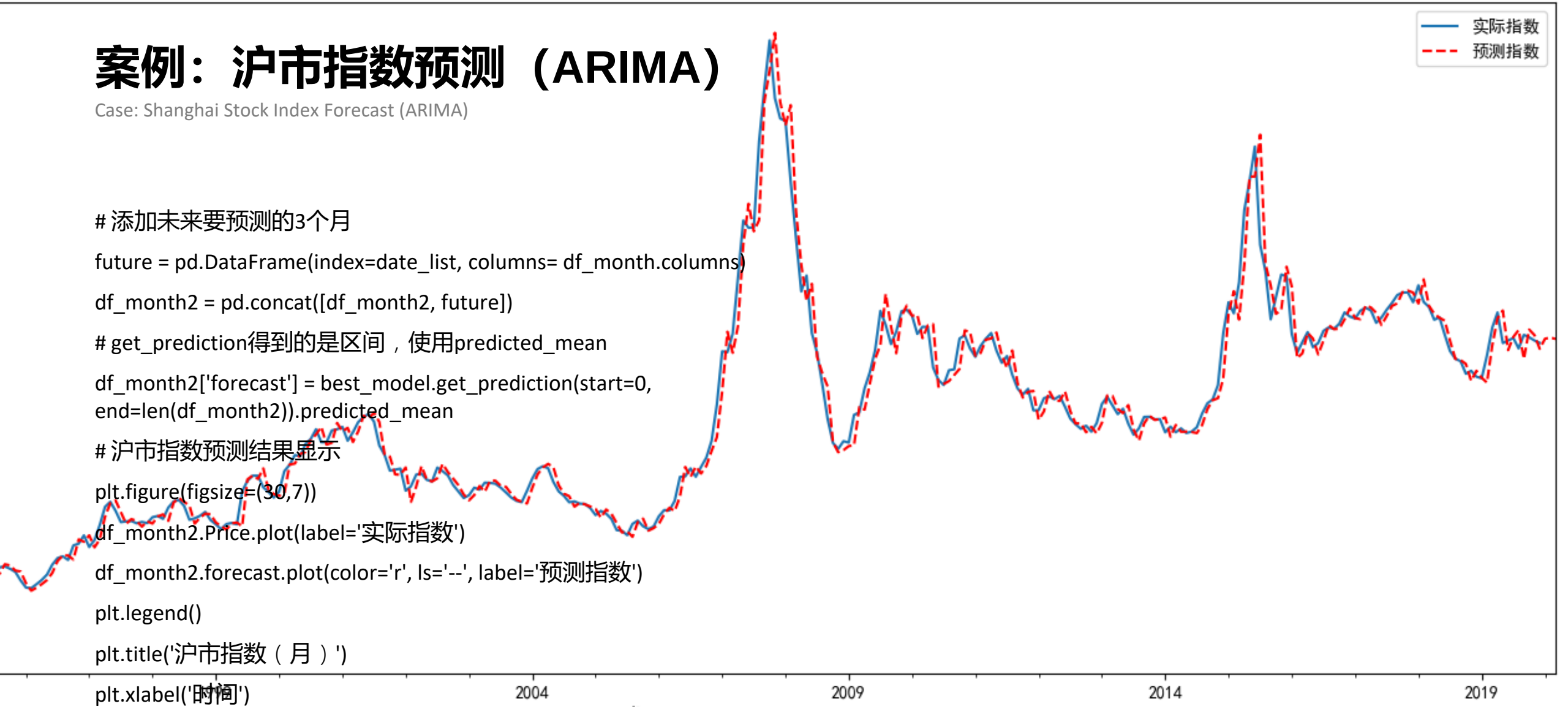
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5689	0.162	3.521	0.000	0.252	0.886
ma.L1	-0.2316	0.163	-1.420	0.156	-0.551	0.088
ma.L2	-0.0426	0.058	-0.732	0.464	-0.157	0.071
ma.L3	-0.1654	0.045	-3.654	0.000	-0.254	-0.077
ma.L4	0.2216	0.031	7.128	0.000	0.161	0.283
sigma2	2.603e+04	943.910	27.579	0.000	2.42e+04	2.79e+04

Ljung-Box (Q):	44.60	Jarque-Bera (JB):	1130.74
Prob(Q):	0.28	Prob(JB):	0.00
Heteroskedasticity (H):	2.23	Skew:	-0.84
Prob(H) (two-sided):	0.00	Kurtosis:	11.73

案例：沪市指数预测（ARIMA）

Case: Shanghai Stock Index Forecast (ARIMA)

```
# 添加未来要预测的3个月
future = pd.DataFrame(index=date_list, columns= df_month.columns)
df_month2 = pd.concat([df_month2, future])
# get_prediction得到的是区间，使用predicted_mean
df_month2['forecast'] = best_model.get_prediction(start=0,
end=len(df_month2)).predicted_mean
# 沪市指数预测结果显示
plt.figure(figsize=(30,7))
df_month2.Price.plot(label='实际指数')
df_month2.forecast.plot(color='r', ls='--', label='预测指数')
plt.legend()
plt.title('沪市指数（月）')
plt.xlabel('时间')
plt.ylabel('指数')
plt.show()
```



资金流入流出

你能对业务量突然上涨/下降的情况进行预测么

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

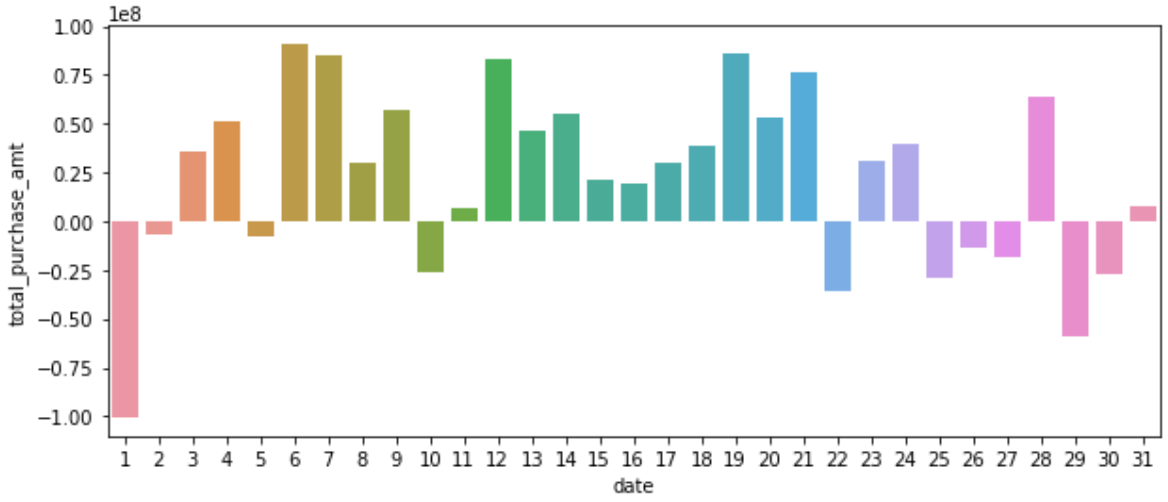
CASE：资金流入流出预测

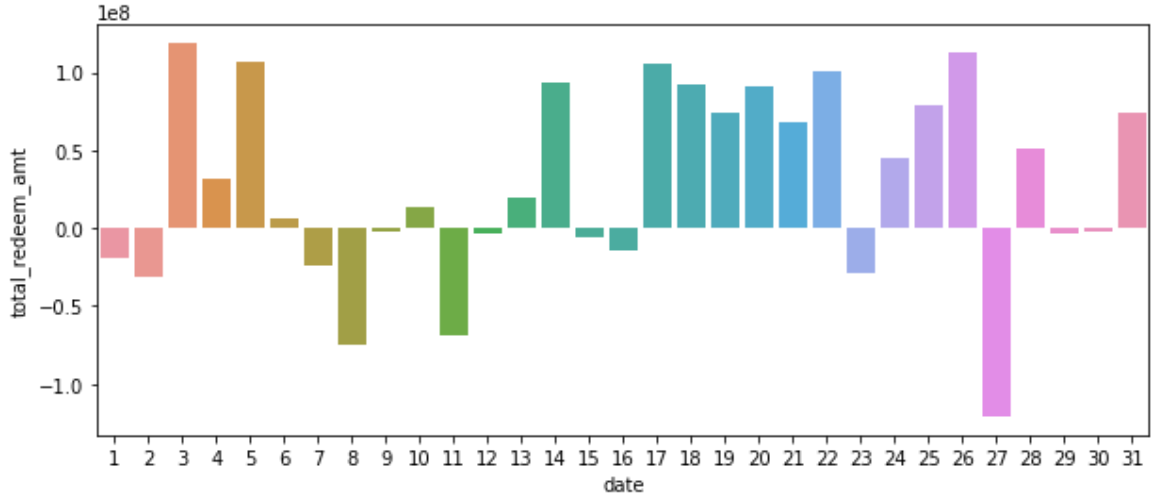
- https://tianchi.aliyun.com/competition/entrance/231573/information
- 数据集一共包括4张表：用户基本信息数据、用户申购赎回数据、收益率表和银行间拆借利率表

2.8万用户，284万行为数据，294天拆解利率，427天收益率

2013-07-01到2014-08-31，预测2014年9月的申购和赎回

Thinking：如果能对未来30天的资金流入流出预测准确，对货币类理财产品有怎样的价值?





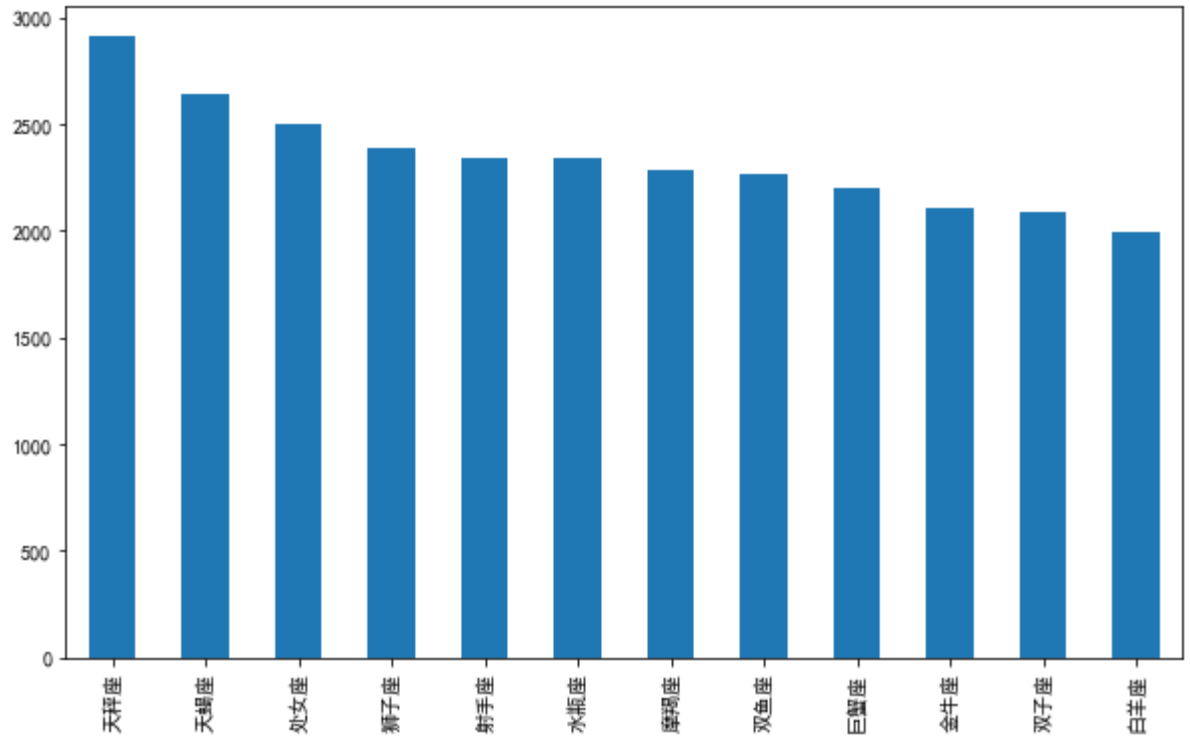
案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

- 用户信息表，user_profile_table

总共随机抽取了约3万用户，主要包含了用户的性别、城市和星座，其中部分用户在 2014 年 9 月份第一次出现，这些用户只在测试数据中

列名	类型	含义	示例
user_id	bigint	用户 ID	1234
Sex	bigint	用户性别（1：男，0：女）	0
City	bigint	所在城市	6081949
constellation	string	星座	射手座



案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

用户申购赎回数据表 user_balance_table

- 数据包括了 20130701 至 20140831 申购和赎回信息，字段包括用户操作时间和操作记录，其中操作记录包括申购和赎回两个部分
- 金额的单位是分，即0.01元
- 如果用户今日消费总量为0，即consume_amt=0，同时四个category字段为空
- 数据经过了脱敏，同时保证了：

今日余额 = 昨日余额 + 今日申购 - 今日赎回，不会出现负值

列名	类型	含义	示例
user_id	bigint	用户id	1234
report_date	string	日期	20140407
tBalance	bigint	今日余额	109004
yBalance	bigint	昨日余额	97389
total_purchase_amt	bigint	今日总购买量=直接购买+收益	21876
direct_purchase_amt	bigint	今日直接购买量	21863
purchase_bal_amt	bigint	今日支付宝余额购买量	0
purchase_bank_amt	bigint	今日银行卡购买量	21863
total_redeem_amt	bigint	今日总赎回量=消费+转出	10261
consume_amt	bigint	今日消费总量	0
transfer_amt	bigint	今日转出总量	10261
tftobal_amt	bigint	今日转出到支付宝余额总量	0
tftocard_amt	bigint	今日转出到银行卡总量	10261
share_amt	bigint	今日收益	13
category1	bigint	今日类目1消费总额	0
category2	bigint	今日类目2消费总额	0
category3	bigint	今日类目3消费总额	0
category4	bigint	今日类目4消费总额	0

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

收益率表 mfd_day_share_interest

- 收益表为余额宝在 14 个月内的收益率表

列名	类型	含义	示例
mfd_date	string	日期	20140102
mfd_daily_yield	double	万份收益，即1万块钱的收益。	1.5787
mfd_7daily_yield	double	七日年化收益率（%）	6.307

上海银行间同业拆放利率表 mfd_bank_shibor

- 银行间拆借利率表是14个月期间银行之间的拆借利率（皆为年化利率）

列名	类型	含义	示例
mfd_date	String	日期	20140102
Interest_O_N	Double	隔夜利率（%）	2.8
Interest_1_W	Double	1周利率（%）	4.25
Interest_2_W	Double	2周利率（%）	4.9
Interest_1_M	Double	1个月利率（%）	5.04
Interest_3_M	Double	3个月利率（%）	4.91
Interest_6_M	Double	6个月利率（%）	4.79
Interest_9_M	Double	9个月利率（%）	4.76
Interest_1_Y	Double	1年利率（%）	4.78

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

收益计算方式

- 主要基于实际余额宝收益计算方法，进行了简化

1）收益计算的时间不再是会计日，而是自然日，以0点为分隔（0点之前算昨天，0点之后算今天）

2）收益的显示时间，即实际将第一份收益打入用户账户的时间，以周一转入周三显示为例，如果用户在周一存入10000元，即1000000分，那么这笔金额是周一确认，周二是开始产生收益，在周三将周二产生的收益打入到用户的账户中，此时用户的账户中显示的是1000110分

转入时间	首次显示收益时间
周一	周三
周二	周四
周三	周五
周四	周六
周五	下周二
周六	下周三
周天	下周三

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

提交结果表 tc_comp_predict_table

字段	类型	含义	示例
report_date	bigint	日期	20140901
purchase	bigint	申购总额	40000000
redeem	bigint	赎回总额	30000000

每一行数据是一天对申购、赎回总额的预测值，输出2014年9月每天的预测，共30行。 purchase 和 redeem 都是金额数据，精确到分

输出示意：

20140901	40000000	30000000
20140902	40000000	30000000
20140903	40000000	30000000

评估指标：

1）计算测试集上每天的申购及赎回与实际的误差

每日申购相对误差(真实值z_i，预测值为$\hat{z}_i$)：

$$\text{Purchase}_i = \frac{|z_i - \hat{z}_i|}{z_i}$$

每日赎回相对误差(真实值y_i，预测值为$\hat{y}_i$)：

$$\text{Redeem}_i = \frac{|y_i - \hat{y}_i|}{y_i}$$

2）误差与得分之间的计算公式不公布，但保证单调递减

第 i 天的申购误差Purchasei=0 ，这一天的得分为10分；

当Purchasei > 0.3，得分为0

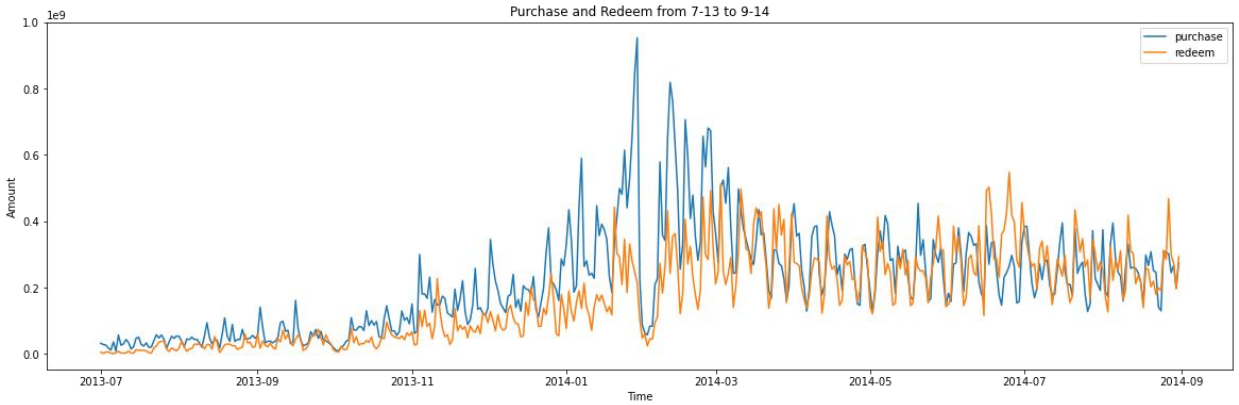
3) 总积分 = 申购预测得分*45% + 赎回预测得分*55%

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

- 数据探索EDA

1）每日总购买与赎回量的时间序列图



2）STL分解，将时序图拆分为：Trend + Seasonal + Residual

- 方法1：采用时间序列进行预测

Step1，平稳性检测 adfuller

Step2，采用 ARIMA模型

Step3，模型训练集与预测

- 方法2：基于时序规则的挖掘

Step1，获得周期因子（weekday）

Step2，计算base

Step3，使用base * 周期因子进行预测

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

- read_csv中的日期格式解析

pd.read_csv('user_balance_table.csv', parse_dates = ['report_date'])

设置parse_dates参数，将时间字符串转换为日期格式

- DataFrame.diff()函数

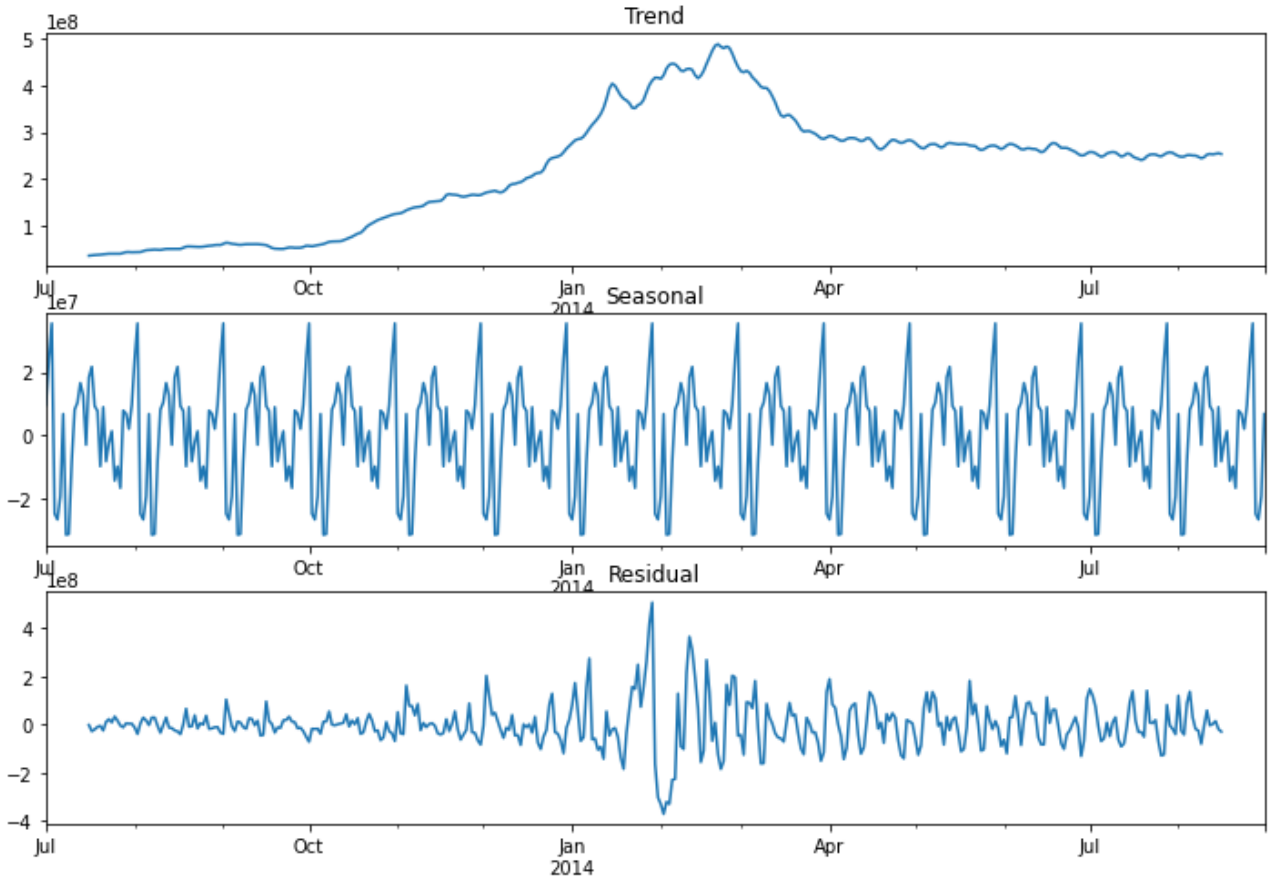
用来将数据进行某种移动之后与原数据进行比较得出的差异数据

- DataFrame.shift()函数

可以把数据移动指定的位数

periods=-1 往上移动或往左移动

periods=1 往下移动或往右移动



案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

平稳性检测（ADF检测）：

- 在使用时间序列模型时（比如 ARMA、ARIMA），需要时间序列是平稳的，所以第一步都需要进行平稳性检验，常用的统计检验方法为ADF检验（也称为单位根检验）
- ADF检验，就是判断序列是否存在单位根，如果序列平稳，就不存在单位根，否则，就会存在单位根
- ADF检验的 H0 假设就是存在单位根，如果得到的显著性检验统计量小于三个置信度（10%，5%，1%），则对应有（90%，95，99%）的把握来拒绝原假设

```
from statsmodels.tsa.stattools import adfuller

t=adfuller(df_p['total_purchase_amt'])
```

```
(-1.5898802926313507, 0.4886749751375928, 18, 408, {'1%': -3.446479704252724, '5%': -2.8686500930967354, '10%': -2.5705574627547096}, 15960.28197033403)
```

输出结果依次为：

t-statistic, p-value, usedlag, nobs

critical-value：测试统计数据的临界值为1%，5%和10%

AIC

如何确定该序列能否平稳：

主要看1%、%5、%10不同程度拒绝原假设的统计值和ADF Test result的比较，如果ADF Test result同时小于1%、5%、10%即说明非常好地拒绝原假设（原假设是不稳定的，因此证明是平稳的）

这里，adf结果为-1.58988， 大于三个level的统计值，无法拒绝原假设（原假设是不平稳的），需要进行一阶差分后，再进行检验

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

- 时间序列预测

1）针对购买purchase建模

ARIMA(purchase,order=(7,1,5)).fit()

model.predict('2014-09-01', '2014-09-30',typ='levels')

2）针对赎回redeem建模

ARIMA(redeem,order=(7,1,5)).fit()

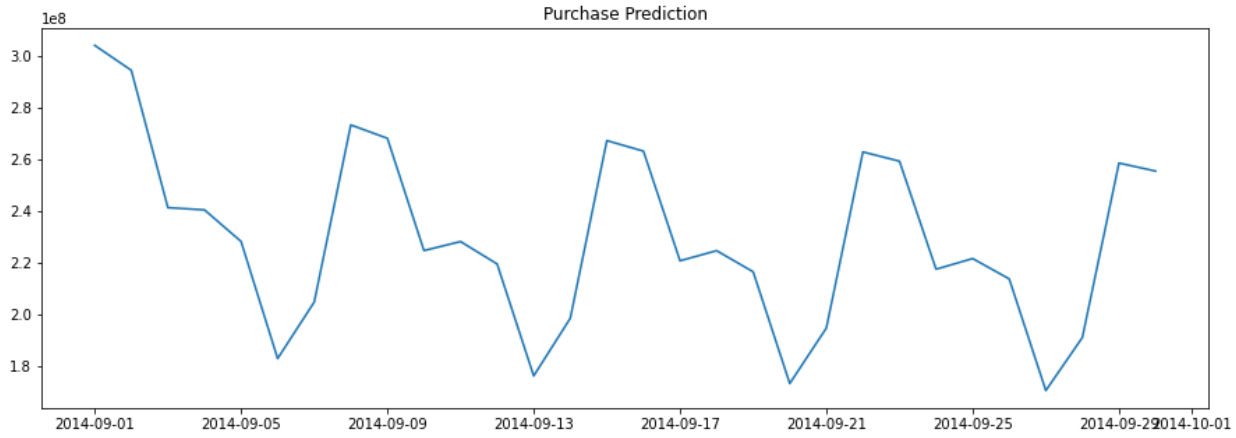
model.predict('2014-09-01', '2014-09-30',typ='levels')

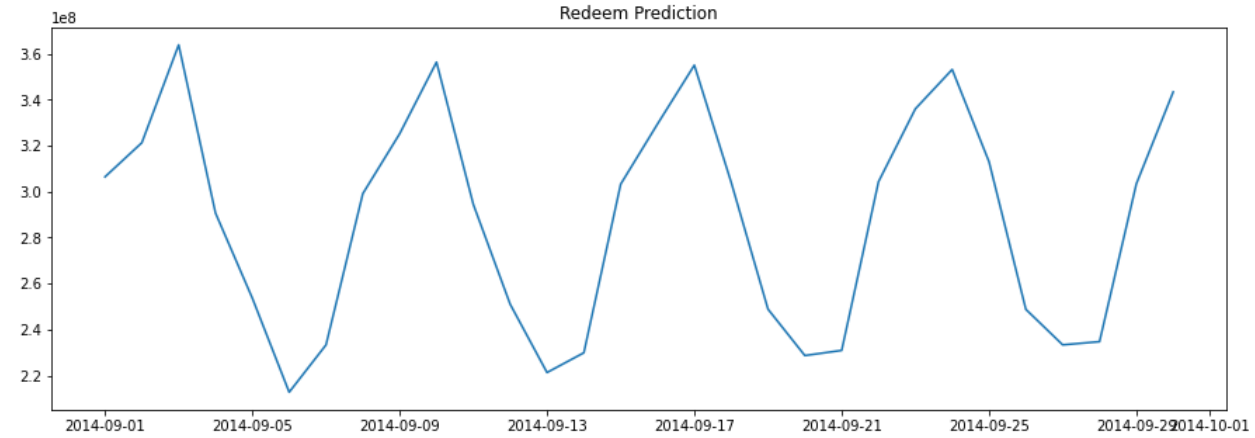
Thinking：模型预测准确性如何？

1）过于简单，实际情况并不是

2）周一到周日的特征规律没有利用

3）没有考虑特殊时间，比如节日，利率波动节点





案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

时间序列规则：

- 选择特征

可以用简单的统计量来作为特征，从中提取出有用的信息

1）中位数：居于中间位置的数，较为稳健

2）均值：当分布符合正态分布时，可以代表整体特征

3）临近数据：离待测数据越近的数据对其影响越大

基于周期因子的时间序列预测

- 很多数据都具有明显的周期性，比如客流量，支付等
- 需要确定周期长度，比如一周7天，一个月30天，结合STL分解 (Seasonal and Trend decomposition) 观察周期变化，缺点是没有考虑到节假日、突发事件等情况

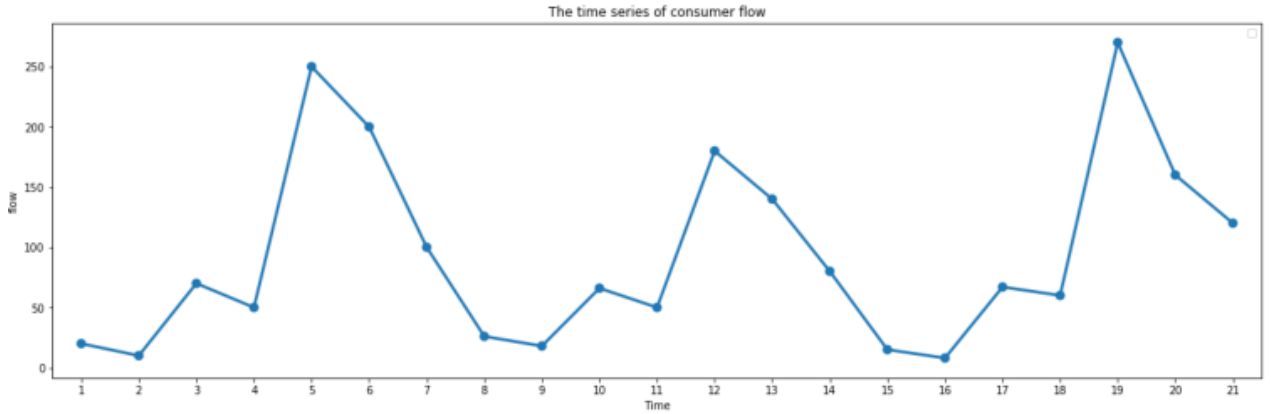
案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

基于周期因子的时间序列预测

- 假设给任务是根据前三周的数据预测第四周每天的客流量

	周一	周二	周三	周四	周五	周六	周日	周均值
第一周	20	10	70	50	250	200	100	100
第二周	26	18	66	50	180	140	80	80
第三周	15	8	67	60	270	160	120	100



Step 1，获得周期因子（weekday）

获得星期几的均值，再除以整体均值

	周一	周二	周三	周四	周五	周六	周日
第一周	20	10	70	50	250	200	100
第二周	26	18	66	50	180	140	80
第三周	15	8	67	60	270	160	120
均值	20.33	12	67.67	53.33	233.33	166.67	100
因子	0.22	0.13	0.73	0.57	2.50	1.79	1.07

Step 2，计算base

Step3，使用base * 周期因子进行预测

假设base=100，可以得到第四周的客流量

	周一	周二	周三	周四	周五	周六	周日
第四周	22	13	73	57	250	179	107

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

预测下个月每一天的情况：

- 如果想预测下个月每天的流量情况，可以基于每月的规律（1-30号的平均流量）* 周期因子

Step1，计算周期因子（weekday）

Step2，计算每日（1号-30号）均值，即1号的平均流量，2号的平均流量 …

Step3，统计星期几（weekday）在每日（day）出现的频次

Step4，基于周期因子获得加权均值，得到每日的base（去掉周期因子的影响）

Step5，根据每日的base和周期因子进行预测

案例：资金流入流出预测

CASE: Prediction of capital inflow and outflow

- 观察特殊日期

清明节，2014年4月5-7日

五一，2014年5月1-3日

六一，2014年5月31-6月2日

中秋节：2014年9月6-8日

国庆节：2014年10月1-7日


Thank You
Using data to solve problems