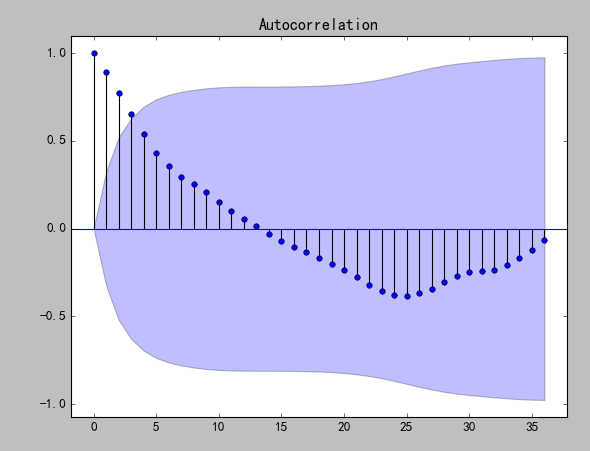
**时间序列ARIMA模型详解：**



    顾名思义，时间序列是时间间隔不变的情况下收集的时间点集合。这些集合被分析用来了解长期发展趋势，为了预测未来或者表现分析的其他形式。但是是什么令时间序列与常见的回归问题的不同？

**有两个原因：**

**1、时间序列是跟时间有关的**。所以基于线性回归模型的假设：观察结果是独立的在这种情况下是不成立的。

**2、随着上升或者下降的趋势，更多的时间序列出现季节性趋势的形式**，如：特定时间框架的具体变化。即：如果你看到羊毛夹克的销售上升，你就一定会在冬季做更多销售。

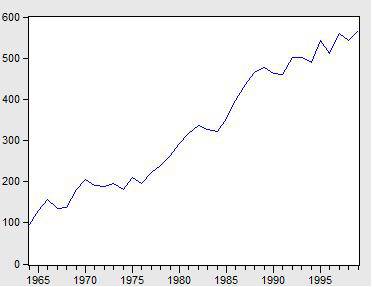
常用的时间序列模型有AR模型、MA模型、ARMA模型和ARIMA模型等。

**一、时间序列的预处理**

拿到一个观察值序列之后，首先要对它的平稳性和纯随机性进行检验，这两个重要的检验称为序列的预处理。根据检验的结果可以将序列分为不同的类型，对不同类型的序列我们会采用不同的分析方法。

先说下什么是平稳，平稳就是围绕着一个常数上下波动且波动范围有限，即有常数均值和常数方差。如果有明显的趋势或周期性，那它通常不是平稳序列。序列平稳不平稳，一般采用三种方法检验：

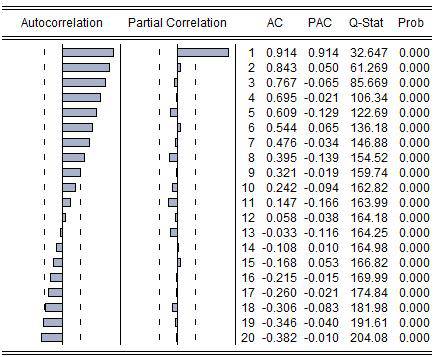
**（1）时序图检验**



看看上面这个图，很明显的增长趋势，不平稳。

（2）自相关系数和偏相关系数

还以上面的序列为例：用SPSS得到自相关和偏相关图。



分析：左边第一个为自相关图（Autocorrelation），第二个偏相关图(Partial Correlation)。

平稳的序列的自相关图和偏相关图要么拖尾，要么是截尾。截尾就是在某阶之后，系数都为 0 ，怎么理解呢，看上面偏相关的图，当阶数为 1 的时候，系数值还是很大， 0.914. 二阶长的时候突然就变成了 0.050. 后面的值都很小，认为是趋于 0 ，这种状况就是截尾。什么是拖尾，拖尾就是有一个缓慢衰减的趋势，但是不都为 0 。

自相关图既不是拖尾也不是截尾。以上的图的自相关是一个三角对称的形式，这种趋势是单调趋势的典型图形，说明这个序列不是平稳序列。

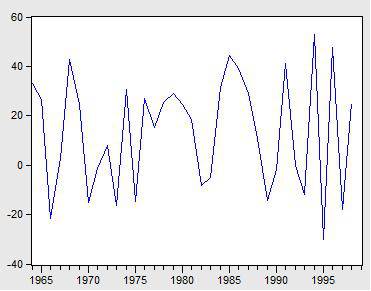
**（3）单位根检验**

单位根检验是指检验序列中是否存在单位根，如果存在单位根就是非平稳时间序列。

**不平稳，怎么办？**

答案是差分，转换为平稳序列。什么是差分？一阶差分指原序列值相距一期的两个序列值之间的减法运算；k阶差分就是相距k期的两个序列值之间相减。如果一个时间序列经过差分运算后具有平稳性，则该序列为差分平稳序列，可以使用ARIMA模型进行分析。

还是上面那个序列，两种方法都证明他是不靠谱的，不平稳的。确定不平稳后，依次进行1阶、2阶、3阶...差分，直到平稳为止。先来个一阶差分，上图:



 从图上看，一阶差分的效果不错，看着是平稳的。

**平稳性检验过后，下一步是纯随机性检验。**

对于纯随机序列，又称白噪声序列，序列的各项数值之间没有任何相关关系，序列在进行完全无序的随机波动，可以终止对该序列的分析。白噪声序列是没有信息可提取的平稳序列。

对于平稳非白噪声序列，它的均值和方差是常数。通常是建立一个线性模型来拟合该序列的发展，借此提取该序列的有用信息。ARMA模型是最常用的平稳序列拟合模型。

**二、平稳时间序列建模**

某个时间序列经过预处理，被判定为平稳非白噪声序列，就可以进行时间序列建模。

**建模步骤：**

（1）计算出该序列的自相关系数（ACF）和偏相关系数（PACF）；

（2）模型识别，也称模型定阶。根据系数情况从AR(p)模型、MA(q)模型、ARMA(p，q)模型、ARIMA（p，d，q）模型中选择合适模型，其中p为自回归项，d为差分阶数，q为移动平均项数。

下面是平稳序列的模型选择：

|  |  |  |
| --- | --- | --- |
| 自相关系数（ACF） | 偏相关系数（PACF） | 选择模型 |
| 拖尾 | p阶截尾 | AR(p) |
| q阶截尾 | 拖尾 | MA(q) |
| p阶拖尾 | q阶拖尾 | ARMA(p，q) |

 ARIMA 是 ARMA 算法的扩展版，用法类似 。

（3）估计模型中的未知参数的值并对参数进行检验；

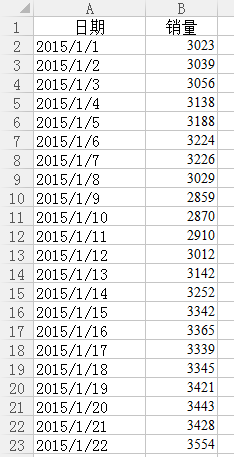
（4）模型检验；

（5）模型优化；

（6）模型应用：进行短期预测。

**三、python实例操作**

以下为某店铺2015/1/1~2015/2/6的销售数据,以此建模预测2015/2/7~2015/2/11的销售数据。



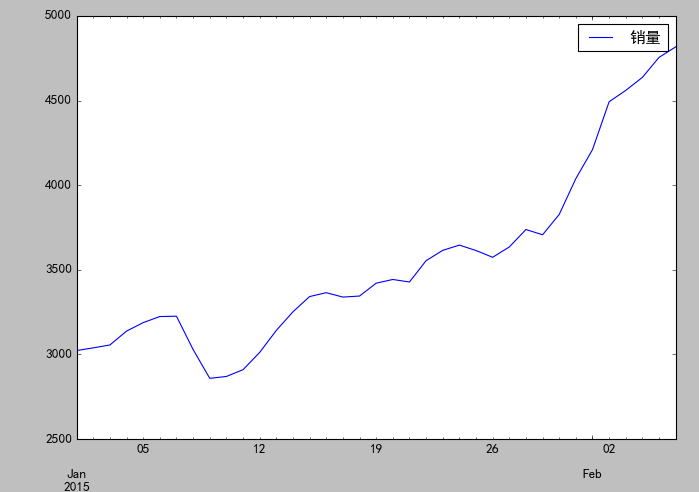
*#-\*- coding: utf-8 -\*-  
#arima时序模型***import**pandas **as**pd  
  
*#参数初始化*discfile = **'E:/destop/text/arima\_data.xls'**forecastnum = 5  
  
*#读取数据，指定日期列为指标，Pandas自动将“日期”列识别为Datetime格式*data = pd.read\_excel(discfile, index\_col = **u'日期'**)  
  
*#时序图***import**matplotlib.pyplot **as**plt

*#用来正常显示中文标签*

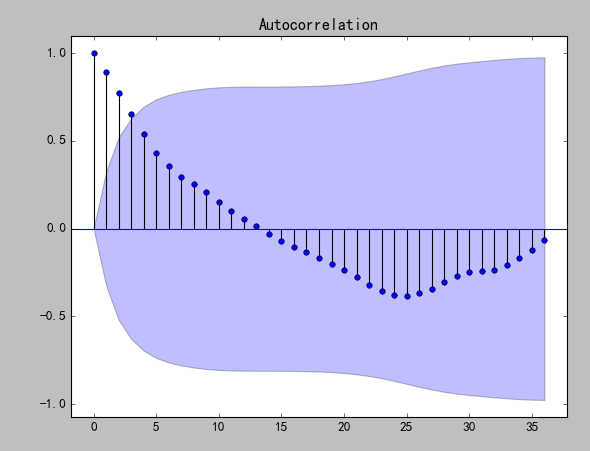
plt.rcParams[**'font.sans-serif'**] = [**'SimHei'**]

*#用来正常显示负号*

plt.rcParams[**'axes.unicode\_minus'**] = False data.plot()  
plt.show()



*#自相关图***from**statsmodels.graphics.tsaplots **import**plot\_acf  
plot\_acf(data).show()



*#平稳性检测***from**statsmodels.tsa.stattools **import**adfuller **as**ADF  
**print**(**u'原始序列的ADF检验结果为：'**, ADF(data[**u'销量'**]))

*#返回值依次为adf、pvalue、usedlag、nobs、critical values、icbest、regresults、resstore*

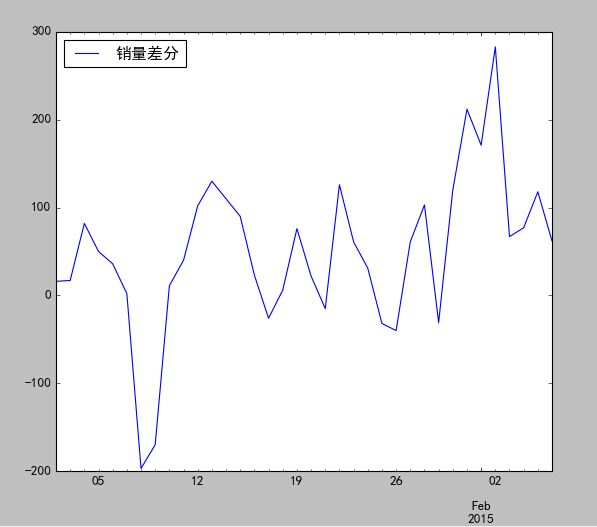
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 原始序列的单位根（adf）检验 | | | | |
| adf | cValue | | | p值 |
| 1% | 5% | 10% |
| 1.81 | -3.7112 | -2.9812 | -2.6301 | 0.9984 |

Pdf值大于三个水平值，p值显著大于0.05，该序列为非平稳序列。

*#差分后的结果*D\_data = data.diff().dropna()  
D\_data.columns = [**u'销量差分'**]

*#时序图*

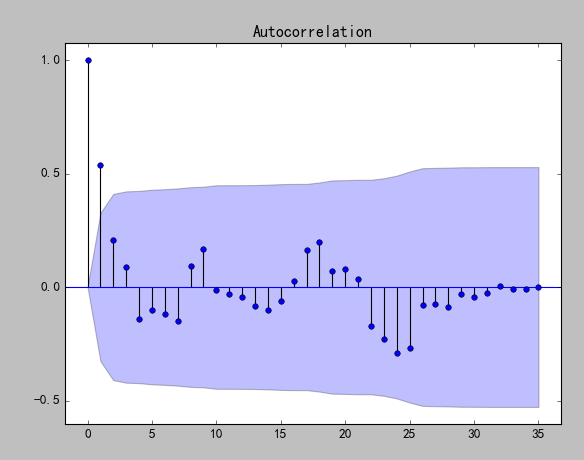
D\_data.plot() plt.show()



*#自相关图*

plot\_acf(D\_data).show()

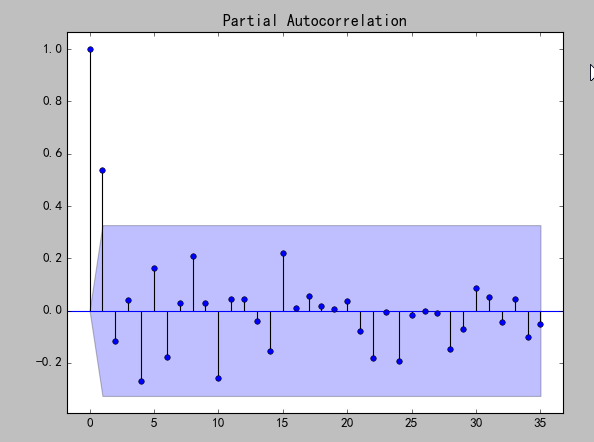
plt.show()

**

**from**statsmodels.graphics.tsaplots **import**plot\_pacf

*#偏自相关图*

plot\_pacf(D\_data).show()

**

***#平稳性检测***

**print**(**u'差分序列的ADF检验结果为：'**, ADF(D\_data[**u'销量差分'**]))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 一阶差分后序列的单位根（adf）检验 | | | | |
| adf | cValue | | | p值 |
| 1% | 5% | 10% |
| -3.15 | -3.6327 | -2.9485 | -2.6130 | 0.0227 |

Pdf值小于两个水平值，p值显著小于0.05，一阶差分后序列为平稳序列。

*#白噪声检验***from**statsmodels.stats.diagnostic **import**acorr\_ljungbox

*#返回统计量和p值*

**print**(**u'差分序列的白噪声检验结果为：'**, acorr\_ljungbox(D\_data, lags=1))

|  |  |
| --- | --- |
| 一阶差分后序列的白噪声检验 | |
| stat | P值 |
| 11.304 | 0.007734 |

P值小于0.05，所以一阶差分后的序列为平稳非白噪声序列。

**from**statsmodels.tsa.arima\_model **import**ARIMA  
*#定阶*

*#一般阶数不超过length/10*

pmax = int(len(D\_data)/10)

*#一般阶数不超过length/10*

qmax = int(len(D\_data)/10)

*#bic矩阵*

bic\_matrix = [] **for**p **in**range(pmax+1):  
  tmp = []  
  **for**q **in**range(qmax+1):

*#存在部分报错，所以用try来跳过报错。*

**try**: tmp.append(ARIMA(data, (p,1,q)).fit().bic)  
    **except**:  
      tmp.append(None)  
  bic\_matrix.append(tmp)  
  
*#从中可以找出最小值*

bic\_matrix = pd.DataFrame(bic\_matrix)  *#先用stack展平，然后用idxmin找出最小值位置。*

p,q = bic\_matrix.stack().idxmin()

**print**(**u'BIC最小的p值和q值为：%s、%s'**%(p,q))

取BIC信息量达到最小的模型阶数，结果p为0，q为1，定阶完成。

*#建立ARIMA(0, 1, 1)模型*

model = ARIMA(data, (p,1,q)).fit()

*#给出一份模型报告*

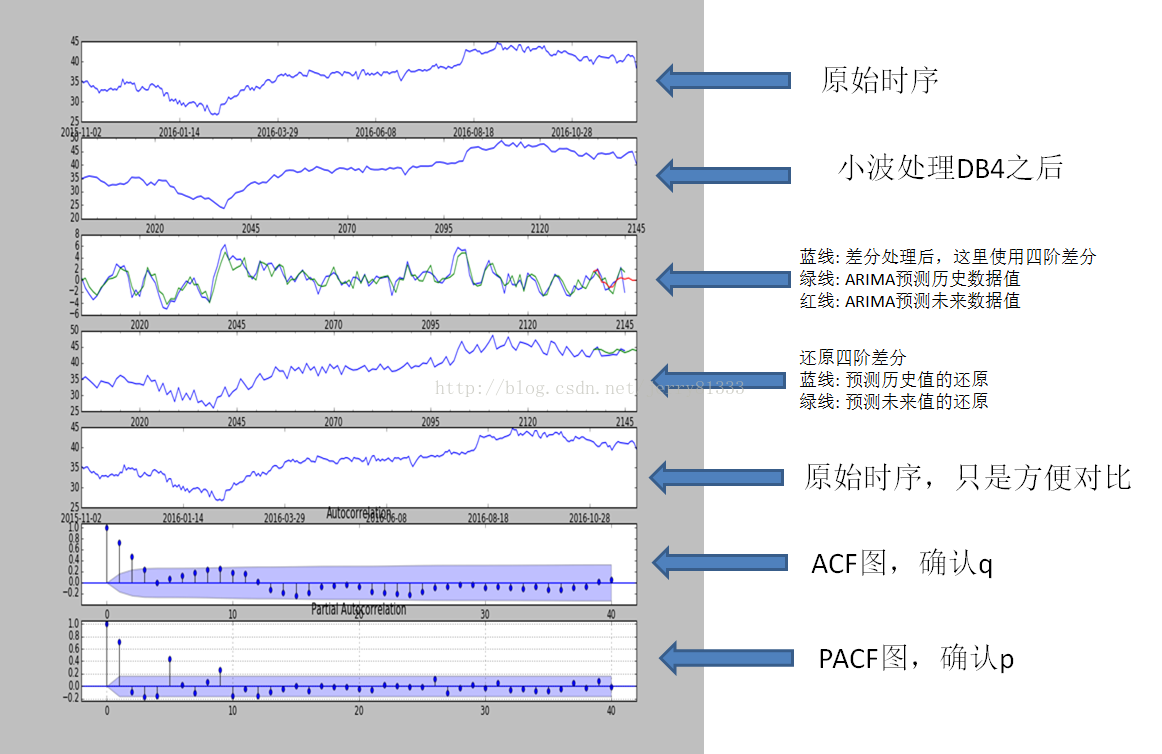
model.summary2()

*#作为期5天的预测，返回预测结果、标准误差、置信区间。*

model.forecast(5)

最终模型预测值如下:

|  |  |  |  |
| --- | --- | --- | --- |
| 2015/2/7 | 2015/2/8 | 2015/2/9 | 2015/2/10 |



1. 判断时间序列是否是平稳白噪声序列，若不是进行平稳化 （2）本实例数据带有周期性，因此先进行一阶差分，再进行144步差分 （3）看差分序列的自相关图和偏自相关图，差分后的而序列为平稳序列 （4）模型定阶，根据aic,bic,hqic （5）预测，确定模型后预测 （6）还原，由于预测时用的差分序列，得到的预测值为差分序列的预测值，需要将其还原

我们将使用“网格搜索”来迭代地探索参数的不同组合。 对于参数的每个组合，我们使用statsmodels模块的SARIMAX()函数拟合一个新的季节性ARIMA模型，并评估其整体质量。 一旦我们探索了参数的整个范围，我们的最佳参数集将是我们感兴趣的标准产生最佳性能的参数。 我们开始生成我们希望评估的各种参数组合：

# Define the p, d and q parameters to take any value between 0 and 2

p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets

pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets

seasonal\_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')

print('SARIMAX: {} x {}'.format(pdq[1], seasonal\_pdq[1]))

print('SARIMAX: {} x {}'.format(pdq[1], seasonal\_pdq[2]))

print('SARIMAX: {} x {}'.format(pdq[2], seasonal\_pdq[3]))

print('SARIMAX: {} x {}'.format(pdq[2], seasonal\_pdq[4]))

OutputExamples of parameter combinations for Seasonal ARIMA...

SARIMAX: (0, 0, 1) x (0, 0, 1, 12)

SARIMAX: (0, 0, 1) x (0, 1, 0, 12)

SARIMAX: (0, 1, 0) x (0, 1, 1, 12)

SARIMAX: (0, 1, 0) x (1, 0, 0, 12)

我们现在可以使用上面定义的参数三元组来自动化不同组合对ARIMA模型进行培训和评估的过程。 在统计和机器学习中，这个过程被称为模型选择的网格搜索（或超参数优化）。

在评估和比较配备不同参数的统计模型时，可以根据数据的适合性或准确预测未来数据点的能力，对每个参数进行排序。 我们将使用AIC （Akaike信息标准）值，该值通过使用statsmodels安装的ARIMA型号方便地返回。 AIC衡量模型如何适应数据，同时考虑到模型的整体复杂性。 在使用大量功能的情况下，适合数据的模型将被赋予比使用较少特征以获得相同的适合度的模型更大的AIC得分。 因此，我们有兴趣找到产生最低AIC值的模型。

下面的代码块通过参数的组合来迭代，并使用SARIMAX函数来适应相应的季节性ARIMA模型。 这里， order参数指定(p, d, q)参数，而seasonal\_order参数指定季节性ARIMA模型的(P, D, Q, S)季节分量。 在安装每个SARIMAX()模型后，代码打印出其各自的AIC得分。

warnings.filterwarnings("ignore") # specify to ignore warning messages

for param in pdq:

for param\_seasonal in seasonal\_pdq:

try:

mod = sm.tsa.statespace.SARIMAX(y,

order=param,

seasonal\_order=param\_seasonal,

enforce\_stationarity=False,

enforce\_invertibility=False)

results = mod.fit()

print('ARIMA{}x{}12 - AIC:{}'.format(param, param\_seasonal, results.aic))

except:

continue

由于某些参数组合可能导致数字错误指定，因此我们明确禁用警告消息，以避免警告消息过载。 这些错误指定也可能导致错误并引发异常，因此我们确保捕获这些异常并忽略导致这些问题的参数组合。

上面的代码应该产生以下结果，这可能需要一些时间：

OutputSARIMAX(0, 0, 0)x(0, 0, 1, 12) - AIC:6787.3436240402125

SARIMAX(0, 0, 0)x(0, 1, 1, 12) - AIC:1596.711172764114

SARIMAX(0, 0, 0)x(1, 0, 0, 12) - AIC:1058.9388921320026

SARIMAX(0, 0, 0)x(1, 0, 1, 12) - AIC:1056.2878315690562

SARIMAX(0, 0, 0)x(1, 1, 0, 12) - AIC:1361.6578978064144

SARIMAX(0, 0, 0)x(1, 1, 1, 12) - AIC:1044.7647912940095

...

...

...

SARIMAX(1, 1, 1)x(1, 0, 0, 12) - AIC:576.8647112294245

SARIMAX(1, 1, 1)x(1, 0, 1, 12) - AIC:327.9049123596742

SARIMAX(1, 1, 1)x(1, 1, 0, 12) - AIC:444.12436865161305

SARIMAX(1, 1, 1)x(1, 1, 1, 12) - AIC:277.7801413828764

我们的代码的输出表明， SARIMAX(1, 1, 1)x(1, 1, 1, 12)产生最低的AIC值为277.78。 因此，我们认为这是我们考虑过的所有模型中的最佳选择。

https://www.howtoing.com/a-guide-to-time-series-forecasting-with-arima-in-python-3/