Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Романчук Иван Сергеевич
Должность: Ректор
Дата подписания: 16.04.2024 14:17:48
Уникальный программный ключ:
6319edc2b582ffdacea443f01d5779368d0957ac34f5cd074d8118159045247 9

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ
Заместитель директора по учебно-методической работе Финансово-экономического института
/О.А.Кузьменко/
23.06.2021

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ И ПРИМЕРНЫЕ ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

по дисциплине «Эконометрика и анализ данных»
для обучающихся по направлению подготовки 38.04.01 Экономика
Магистерская программа: Финансовая экономика (Финансомика)
форма обучения заочная

Захарова К.А. Методические рекомендации и примерные задания для выполнения лабораторных работ по дисциплине «Эконометрика и анализ данных» для обучающихся по направлению подготовки 38.04.01«Экономика» Магистерская программа: Финансовая экономика (Финансомика), форма обучения заочная. Тюмень, 2021.

Методические рекомендации и примерные задания для выполнения лабораторных работ опубликованы на сайте ТюмГУ: http://www.utmn.ru/sveden/education/#.

© Тюменский государственный университет, 2021.
© Захарова К.А., 2021.

Краткие методические указания для выполнения лабораторных работ

Владение и применение методов анализа данных в автоматизированных информационных системах становится одной из основных составляющих в профессиональной подготовке магистров экономического направления.

Широкое разнообразие программных продуктов, представленных на современном рынке, способно удовлетворить потребности практически любой сферы деятельности. Однако, многие эксперты стремятся отдать привилегии проверенным программам пакета Microsoft Office, которые надежны, относительно недороги, удобны и понятны, в тоже время совместимы практически со всеми программами для финансово- экономической сферы деятельности, а также программам Gretl, Python.

Предложенные задания дают возможность применения пакета прикладных программ Microsoft Office, IBM SPSS Statistics, Gretl, Python в процессе осуществления управленческой деятельности. Владея основными навыками работы в Microsoft Office, IBM SPSS Statistics, Gretl, Python, экономисты имеют возможность совершенствовать личный аппарат анализа и прогнозирования ситуаций при принятии решений в процессе управления.

Курс освоения приемов работы в программах пакета Microsoft Office, IBM SPSS Statistics, Gretl, Python предусматривает подробный разбор примеров задач и команд, начиная от простейших и заканчивая специализированными, предназначенными исключительно для анализа экономической информации.

Изучение программ Microsoft Excel, Microsoft Access, IBM SPSS Statistics, Gretl и Python начинается с простейших задач, в процессе решения которых студент знакомится с интерфейсом программ и с легкостью осваивает основные приемы работы в программах. Поэтапно задания приобретают экономическую направленность, это позволяет применять полученные знания для решения более сложных экономических задач, в разных областях управленческой сферы деятельности: бухгалтерской, банковской, налоговой, инвестиционной и др.

В лабораторном практикуме затронуты темы решения задач множественной регрессии и временных рядов с помощью Python. Рассмотренные примеры задач дают общее представление о применении эконометрических методов для прогнозирования экономических ситуаций в стране. Этой информации достаточно для того, чтобы специалист мог упростить свою роботу с программами.

Примерные задания для выполнения лабораторных работ.

Лабораторная работа 1: пример решения задачи множественной регрессии с помощью Python

Для примера решения задачи прогнозирования, берём набор данных Energy efficiency из крупнейшего репозитория UCI. В качестве инструментов по традиции будем использовать Python с аналитическими пакетами pandas и scikit-learn.

Описание набора данных и постановка задачи

Дан набор данных, котором описаны следующие атрибуты помещения:

Поле	Описание	Тип
X1	Относительная компактность	FLOAT
X2	Площадь	FLOAT
X3	Площадь стены	FLOAT
X4	Площадь потолка	FLOAT
X5	Общая высота	FLOAT
X6	Ориентация	INT
X7	Площадь остекления	FLOAT
X8	Распределенная площадь остекления	INT
y1	Нагрузка при обогреве	FLOAT
y2	Нагрузка при охлаждении	FLOAT

В нем $X1 \ldots X8$ — характеристики помещения на основании которых будет проводиться анализ, а $y1, y2$ — значения нагрузки, которые надо спрогнозировать.

Предварительный анализ данных

Для начала загрузим наши данные и посмотрим на них:

```
from pandas import read_csv, DataFrame
```

```
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.linear_model import LinearRegression, LogisticRegression
```

```
from sklearn.svm import SVR
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.metrics import r2_score
```

```
from sklearn.cross_validation import train_test_split
```

```
dataset = read_csv('EnergyEfficiency/ENB2012_data.csv',';')
```

```
dataset.head()
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
0	0.98	514.5	294.0	110.25	7	2	0	0	15.55	21.33
1	0.98	514.5	294.0	110.25	7	3	0	0	15.55	21.33
2	0.98	514.5	294.0	110.25	7	4	0	0	15.55	21.33
3	0.98	514.5	294.0	110.25	7	5	0	0	15.55	21.33
4	0.90	563.5	318.5	122.50	7	2	0	0	20.84	28.28

Теперь давайте посмотрим не связаны ли между собой какие-либо атрибуты. Сделать это можно, рассчитав коэффициенты корреляции для всех столбцов.

```
dataset.corr()
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
X1	1.000000e+00	-9.919015e-01	-2.037817e-01	-8.688234e-01	8.277473e-01	0.000000	1.283986e-17	1.764620e-17	0.622272	0.634339
X2	-9.919015e-01	1.000000e+00	1.955016e-01	8.807195e-01	-8.581477e-01	0.000000	1.318356e-16	-3.558613e-16	-0.658120	-0.672999
X3	-2.037817e-01	1.955016e-01	1.000000e+00	-2.923165e-01	2.809757e-01	0.000000	-7.969726e-19	0.000000e+00	0.455671	0.427117
X4	-8.688234e-01	8.807195e-01	-2.923165e-01	1.000000e+00	-9.725122e-01	0.000000	-1.381805e-16	-1.079129e-16	-0.861828	-0.862547
X5	8.277473e-01	-8.581477e-01	2.809757e-01	-9.725122e-01	1.000000e+00	0.000000	1.861418e-18	0.000000e+00	0.889431	0.895785

X6	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000	0.000000e+00	0.000000e+00	-0.002587	0.014290
X7	1.283986e-17	1.318356e-16	-7.969726e-19	-1.381805e-16	1.861418e-18	0.000000	1.000000e+00	2.129642e-01	0.269841	0.207505
X8	1.764620e-17	-3.558613e-16	0.000000e+00	-1.079129e-16	0.000000e+00	0.000000	2.129642e-01	1.000000e+00	0.087368	0.050525
Y1	6.222722e-01	-6.581202e-01	4.556712e-01	-8.618283e-01	8.894307e-01	-0.002587	2.698410e-01	8.736759e-02	1.000000	0.975862
Y2	6.343391e-01	-6.729989e-01	4.271170e-01	-8.625466e-01	8.957852e-01	0.014290	2.075050e-01	5.052512e-02	0.975862	1.000000

Как можно заметить из нашей матрицы, коррелируют между собой следующие столбы (Значение коэффициента корреляции больше 95%):

- y1 --> y2
- x1 --> x2
- x4 --> x5

Теперь давайте выберем, какие столбцы их наших пар мы можем убрать из нашей выборки. Для этого, в каждой паре, выберем столбцы, которые в большей степени оказывают влияние на прогнозные значения Y1 и Y2 и оставим их, а остальные удалим. Как можно заметить и матрицы с коэффициентами корреляции на y1,y2 больше значения оказывают X2 и X5, нежели X1 и X4, таким образом мы можем последние столбцы мы можем удалить.

```
dataset = dataset.drop(['X1','X4'], axis=1)
```

```
dataset.head()
```

Помимо этого, можно заметить, что поля Y1 и Y2 очень тесно коррелируют между собой. Но, т. к. нам надо спрогнозировать оба значения мы их оставляем «как есть». Выбор модели

Отделим от нашей выборки прогнозные значения:

```
trg = dataset[['Y1','Y2']]
```

```
trn = dataset.drop(['Y1','Y2'], axis=1)
```

После обработки данных можно перейти к построению модели. Для построения модели будем использовать следующие методы:

- Метод наименьших квадратов
- Случайный лес

- Логистическую регрессию
- Метод опорных векторов
- Метод ближайших соседей

Оценку будем производить с помощью коэффициента детерминации (R-квадрат). Данный коэффициент определяется следующим образом:

$$R^2 = 1 - \frac{V(y|x)}{V(y)} = 1 - \frac{\sigma^2}{\sigma_y^2},$$

где $V(y|x) = \sigma^2$ — условная дисперсия зависимой величины y по фактору x. Коэффициент принимает значение на промежутке $[0,1]$ и чем он ближе к 1 тем сильнее зависимость.

Ну что же теперь можно перейти непосредственно к построению модели и выбору модели. Давайте поместим все наши модели в один список для удобства дальнейшего анализа:

```
    models = [LinearRegression(), # метод наименьших квадратов
        RandomForestRegressor(n_estimators=100, max_features ='sqrt'), # случайный лес
        KNeighborsRegressor(n_neighbors=6), # метод ближайших соседей
        SVR(kernel='linear'), # метод опорных векторов с линейным ядром
        LogisticRegression() # логистическая регрессия
        ]
```

Итак модели готовы, теперь мы разобьем наши исходные данные на 2 подвыборки: тестовую и обучающую. Кто читал мои предыдущие статьи знает, что сделать это можно с помощью функции train_test_split() из пакета scikit-learn:

```
Xtrn, Xtest, Ytrn, Ytest = train_test_split(trn, trg, test_size=0.4)
```

Теперь, т. к. нам надо спрогнозировать 2 параметра $y1, y2$, надо построить регрессию для каждого из них. Кроме этого, для дальнейшего анализа, можно записать полученные результаты во временный DataFrame. Сделать это можно так:

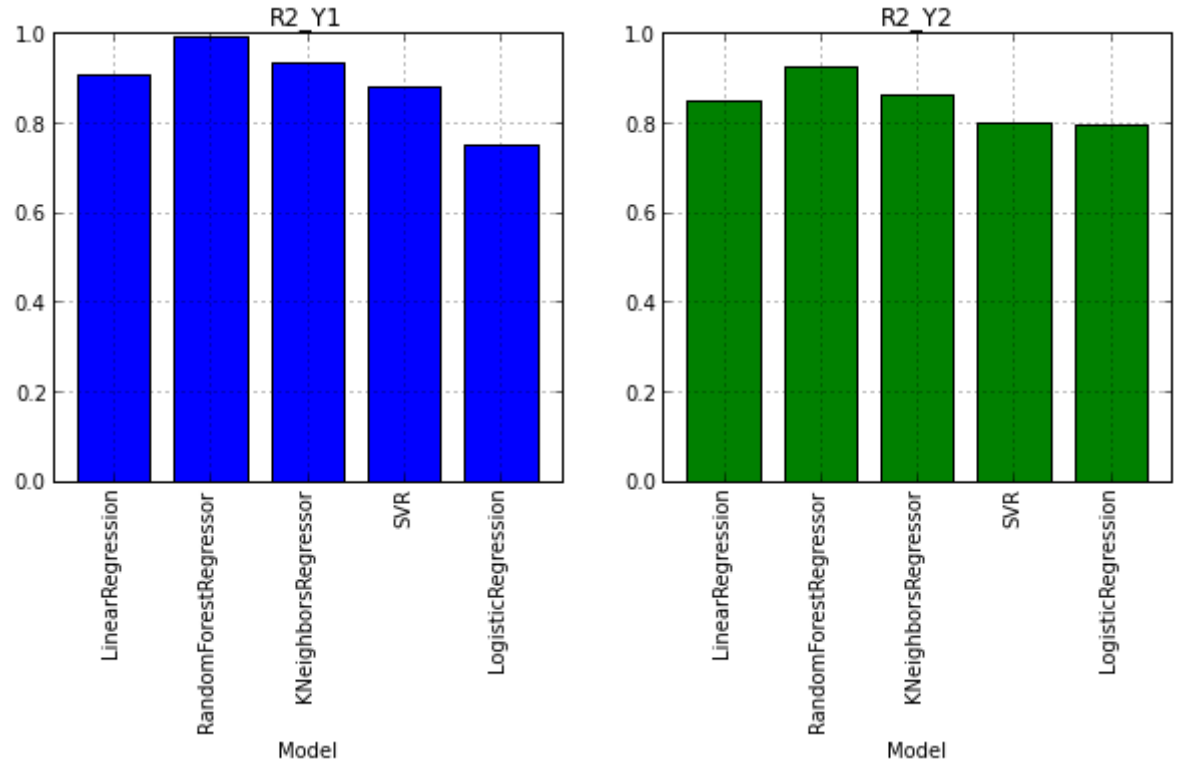
```
    #создаем временные структуры
    TestModels = DataFrame()
    tmp = {}
    #для каждой модели из списка
```

```
    for model in models:
    #получаем имя модели
    m = str(model)
    tmp['Model'] = m[:m.index('(')]
    #для каждого столбцам результирующего набора
    for i inxrange(Ytrn.shape[1]):
    #обучаем модель
    model.fit(Xtrn, Ytrn[:,i])
    #вычисляем коэффициент детерминации
    tmp['R2_Y%s'%str(i+1)] = r2_score(Ytest[:,0], model.predict(Xtest))
    #записываем данные и итоговый DataFrame
    TestModels = TestModels.append([tmp])
    #делаем индекс по названию модели
    TestModels.set_index('Model', inplace=True)
```

Как можно заметить из кода выше, для расчета коэффициента R^2 используется функция r2_score(). Итак, данные для анализа получены. Давайте теперь построим графики и посмотрим какая модель показала лучший результат:

```
fig, axes = plt.subplots(ncols=2, figsize=(10,4))
TestModels.R2_Y1.plot(ax=axes[0], kind='bar', title='R2_Y1')
```

```
TestModels.R2_Y2.plot(ax=axes[1], kind='bar', color='green', title='R2_Y2')
```



Анализ результатов и выводы

Из графиков, приведенных выше, можно сделать вывод, что лучше других с задачей справился метод RandomForest (случайный лес). Его коэффициенты детерминации выше остальных по обоим переменным: $R^2_{y1} \approx 99\,\%, R^2_{y2} \approx 90\,\%$

ля дальнейшего анализа давайте заново обучим нашу модель:

```
model = models[1]
```

```
model.fit(Xtrn, Ytrn)
```

При внимательном рассмотрении, может возникнуть вопрос, почему в предыдущий раз и делили зависимую выборку Ytrn на переменные(по столбцам), а теперь мы этого не делаем. Дело в том, что некоторые методы, такие как RandomForestRegressor, может работать с несколькими прогнозируемыми переменными, а другие (например SVR) могут работать только с одной переменной. Поэтому на при предыдущем обучении мы использовали разбиение по столбцам, чтобы избежать ошибки в процессе построения некоторых моделей. Выбрать модель — это, конечно же, хорошо, но еще неплохо бы обладать информацией, как каждый фактор влиет на прогнозное значение. Для этого у модели есть свойство feature_importances_. С помощью него, можно посмотреть вес каждого фактора в итоговой моделей:

```
model.feature_importances_
```

array([0.40717901, 0.11394948, 0.34984766, 0.00751686, 0.09158358, 0.02992342])

В нашем случае видно, что больше всего на нагрузку при обогреве и охлаждении влияют общая высота и площадь. Их общий вклад в прогнозной модели около 72%. Также

необходимо отметить, что по вышеуказанной схеме можно посмотреть влияние каждого фактора отдельно на обогрев и отдельно на охлаждение, но т. к. эти факторы у нас очень тесно коррелируют между собой ($r = 97\,\%$), мы сделали общий вывод по ним обоим который и был написан выше.

Лабораторная работа 2: анализ временных рядов с помощью PYTHON и модуля STATSMODELS.

Данный модуль предоставляет широкий набор средств и методов для проведения статистического анализа и эконометрики. Я попытаюсь показать основные этапы анализа таких рядов, в заключении мы построим модель ARIMA. Для примера взяты реальные данные по товарообороту одного из складских комплексов Подмосковья.

Загрузка и предварительная обработка данных

Для начала загрузим данные и посмотрим на них:

```
from pandas import read_csv, DataFrame
import statsmodels.api as sm
from statsmodels.iolib.table import SimpleTable
from sklearn.metrics import r2_score
import ml_metrics as metrics
```

In [2]:

```
dataset = read_csv('tovar_moving.csv',';', index_col=['date_oper'],
parse_dates=['date_oper'], dayfirst=True)
dataset.head()
```

	Otgruzka	priemka
date_oper		
2009-09-01	179667	276712
2009-09-02	177670	164999
2009-09-03	152112	189181
2009-09-04	142938	254581

2009-09-05	130741	192486

Итак, как можно заметить функция read_csv(), в данном случем помимо указания параметров, которые задают используемые колонки и индекс, можно заметить еще 3 параметра для работы с датой. Остановимся на них поподробнее. parse_dates задает имена столбцов, которые будут преобразованы в тип DateTime. Стоит отметить, что если в данном столбце будут пустые значения парсинг не удастся и вернется столбец типа object. Чтобы этого избежать надо добавить параметр keep_default_na=False. Заключительный параметр dayfirst указывает функции парсинга, что первое в строке первым идет день, а не наоборот. Если не задать этот параметр, то функция может не правильно преобразовывать даты и путать месяц и день местами. Например 01.02.2013 будет преобразовано в 02-01-2013, что будет неправильно. Выделим в отдельную серию временной ряд со значениями отгрузок:

```
otg = dataset.Otgruzka
```

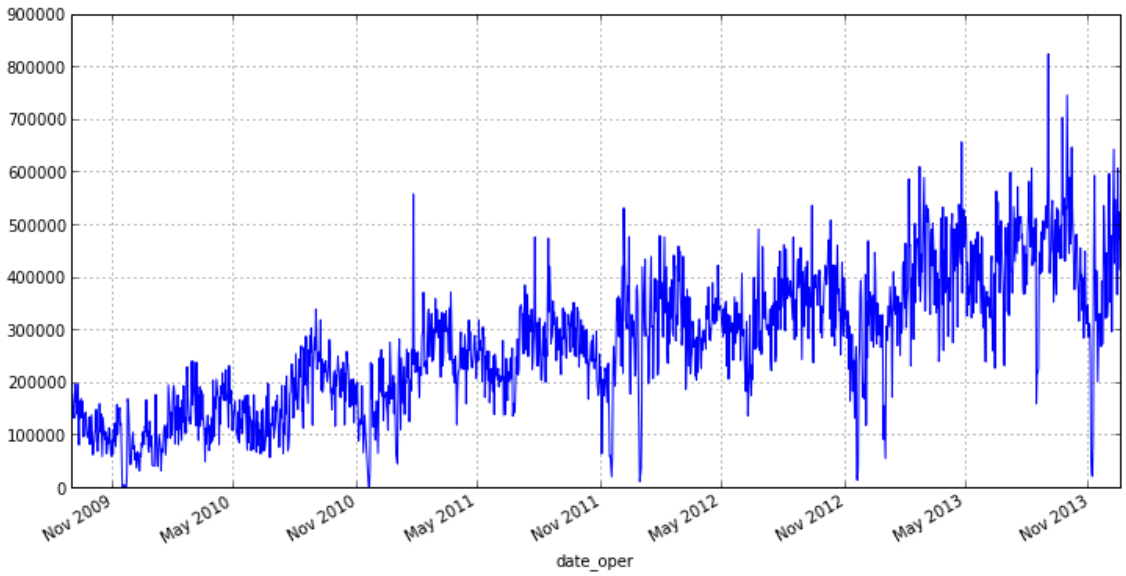
```
otg.head()
```

date_oper	
2009-09-01	179667
2009-09-02	177670
2009-09-03	152112
2009-09-04	142938
2009-09-05	130741
Name: Otgruzka, dtype: int64	

Итак у нас теперь есть временной ряд и можно перейти к его анализу. Анализ временного ряда

Для начала давайте посмортим график нашего ряда:

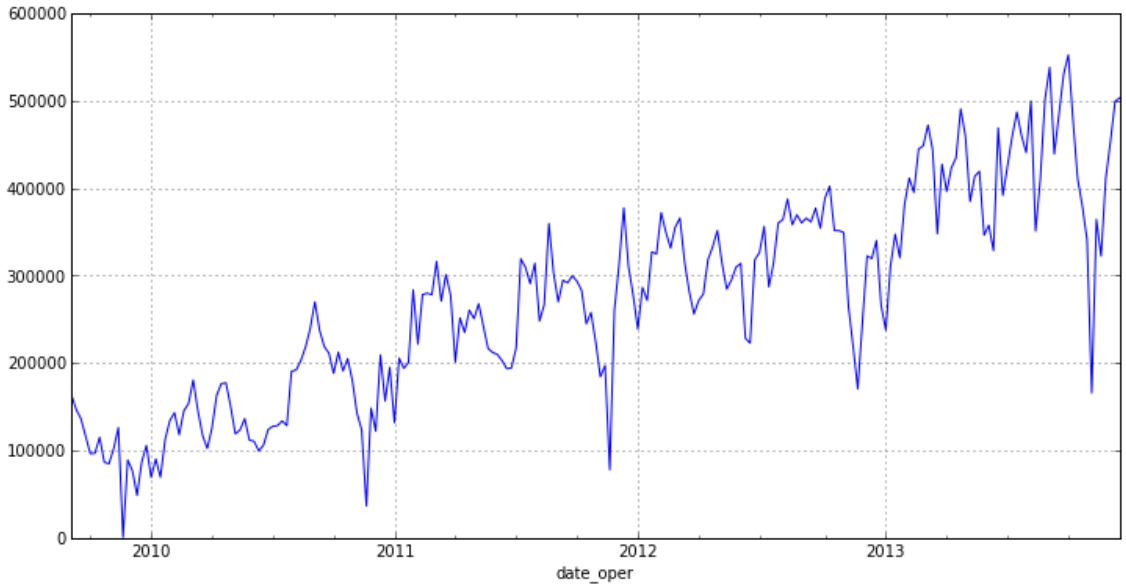
```
otg.plot(figsize=(12,6))
```



Из графика видно, что наш ряд имеет небольшое кол-во выбросов, которые влияют на разброс. Кроме того анализировать отгрузки за каждый день не совсем верно, т.к., например, в конце или начале недели будут дни в которые товара отгружается значительно больше, нежели в остальные. Поэтому есть смысл перейти к недельному интервалу и среднему значению отгрузок на нем, это избавит нас от выбросов и уменьшит колебания нашего ряда. В pandas для этого есть удобная функция resample(), в качестве параметров ей передается период округления и аггрегатная функция:

```
otg = otg.resample('W', how='mean')
```

```
otg.plot(figsize=(12,6))
```



Как можно заметить, новый график не имеет ярких выбросов и имеет ярко выраженный тренд. Из это можно сделать вывод о том, что ряд не является стационарным[1].

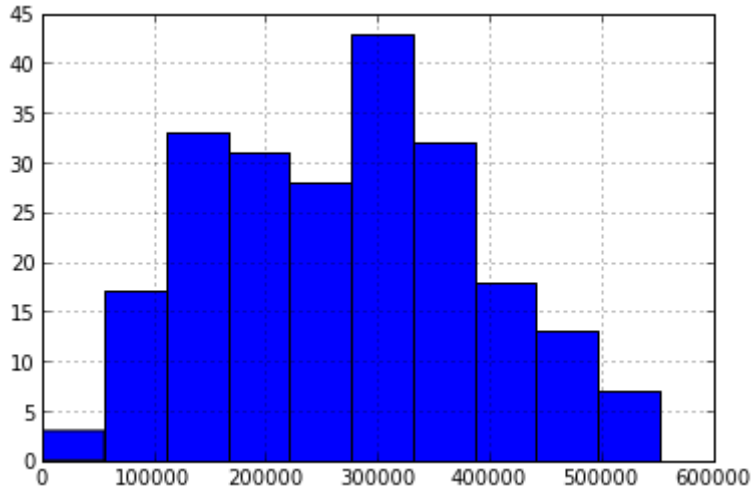
```
itog = otg.describe()
```

```
otg.hist()
```

```
itog
```

count	225
mean	270858.285365
std	118371.082975
min	872.857143
25%	180263.428571
50%	277898.714286
75%	355587.285714
max	552485.142857

dtype: float64



Как можно заметить из характеристик и гистограммы, ряд у нас более менее однородный и имеет относительно небольшой разброс о чем свидетельствует коэффициент вариации: $V = \frac{\sigma}{\overline{x}}$, где σ — среднеквадратическое отклонение, $\overline{x}$ — среднее арифметическое выборки. В нашем случае он равен:

```
print 'V = %f' % (itog['std']/itog['mean'])
```

V = 0.437022

Проведем тест Харки — Бера для определения номарльности распределения, чтобы подтвердить предположение об однородности. Для этого в существует функция

jarque_bera(), которая возвращает значения данной статистики:

```
row =  [u'JB', u'p-value', u'skew', u'kurtosis']
jb_test = sm.stats.stattools.jarque_bera(otg)
a = np.vstack([jb_test])
itog = SimpleTable(a, row)
print itog
```

```
=============================================================
      JB           p-value           skew          kurtosis
-------------------------------------------------------------
5.60453241944 0.0606724103035 0.111910758759 2.25991843713
-------------------------------------------------------------
```

Значение данной статистика свидетельствует о том, нулевая гипотеза о нормальности распределения отвергается с малой вероятностью (probably > 0.05), и, следовательно, наш ряд имеет нормального распределения.
Функция SimpleTable() служит для оформления вывода. В нашем случае на вход ей подается массив значений (размерность не больше 2) и список с названиями столбцов или строк.
Многие методы и модели основаны на предположениях о стационарности ряда, но как было замечено ранее наш ряд таковым скорее всего не является. Поэтому для проверки проверки стационарности давайте проведем обобщенный тест Дикки-Фуллера на наличие единичных корней. Для этого в модуле statsmodels есть функция adfuller():

```
test = sm.tsa.adfuller(otg)
print 'adf: ', test[0]
print 'p-value: ', test[1]
print'Critical values: ', test[4]
if test[0]> test[4]['5%']:
    print 'есть единичные корни, ряд не стационарен'
else:
```

```
    print 'единичных корней нет, ряд стационарен'
```

adf: -1.38835541357
p-value: 0.58784577297
Critical values: {'5%': -2.8753374677799957, '1%': -3.4617274344627398, '10%': -2.5741240890815571}
есть единичные корни, ряд не стационарен

Проведенный тест подтвердил предположения о не стационарности ряда. Во многих случаях взятие разности рядов позволяет это сделать. Если, например, первые разности ряда стационарны, то он называется интегрированным рядом первого порядка. Итак, давайте определим порядок интегрированного ряда для нашего ряда:

```
otg1diff = otg.diff(periods=1).dropna()
```

В коде выше функция diff() вычисляет разность исходного ряда с рядом с заданным смещением периода. Период смещения передается как параметр period. Т.к. в разности первое значение получиться неопределенным, то нам надо избавиться от него для этого и используется метод dropna(). Проверим получившийся ряд на стационарность:

```
test = sm.tsa.adfuller(otg1diff)
print 'adf: ', test[0]
print 'p-value: ', test[1]
print'Critical values: ', test[4]
if test[0]> test[4]['5%']:
    print 'есть единичные корни, ряд не стационарен'
else:
    print 'единичных корней нет, ряд стационарен'
```

adf: -5.95204224907
p-value: 2.13583392404e-07
Critical values: {'5%': -2.8755379867788462, '1%': -3.4621857592784546, '10%': -2.574231080806213}
единичных корней нет, ряд стационарен

Как видно из кода выше получившийся ряд первых разностей приблизился к стационарному. Для полной уверенности разобъем его на несколько промежутков и

убедимся мат. ожидания на разных интервалах:

```
m = otg1diff.index[len(otg1diff.index)/2+1]
```

```
r1 = sm.stats.DescrStatsW(otg1diff[m:])
```

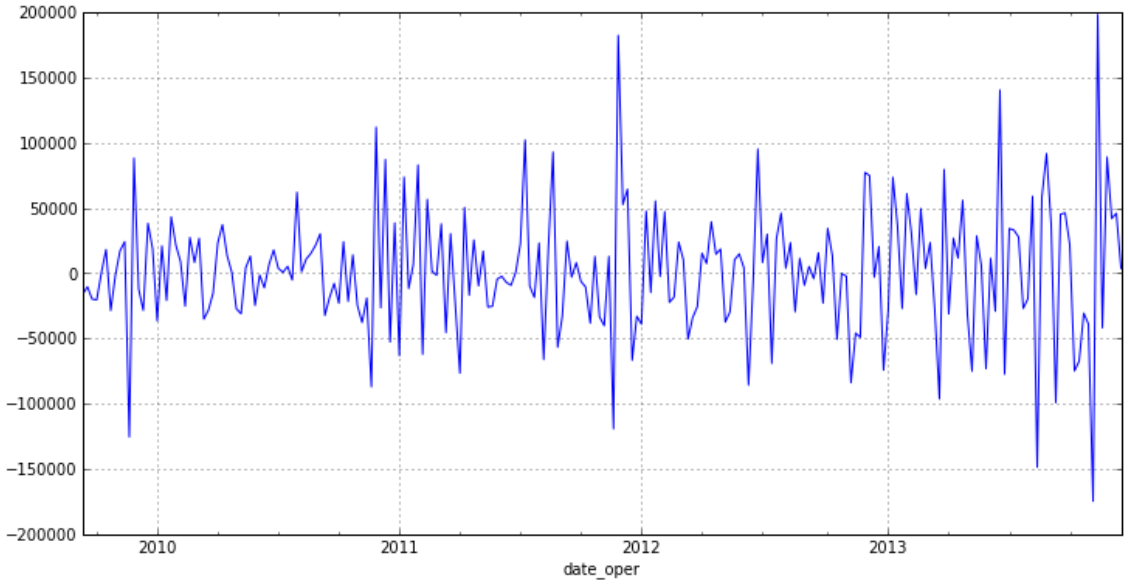
```
r2 = sm.stats.DescrStatsW(otg1diff[:m])
```

```
print 'p-value: ', sm.stats.CompareMeans(r1,r2).ttest_ind()[1]
```

p-value: 0.693072039563

Высокое p-value дает нам возможность утверждать, что нулевая гипотеза о равенстве средних верна, что свидетельствует о стационарности ряда. Осталось убедиться в отсутствии тренда для этого построим график нашего нового ряда:

```
otg1diff.plot(figsize=(12,6))
```



Тренд действительно отсутствует, таким образом ряд первых разностей является стационарным, а наш исходный ряд — интегрированным рядом первого порядка.

Построение модели временного ряда

Для моделирования будем использовать модель ARIMA, построенную для ряда первых разностей. Итак, чтобы построить модель нам нужно знать ее порядок, состоящий из 2-х параметров:

1. p — порядок компоненты AR
2. d — порядок интегрированного ряда
3. q — порядок компонетны MA

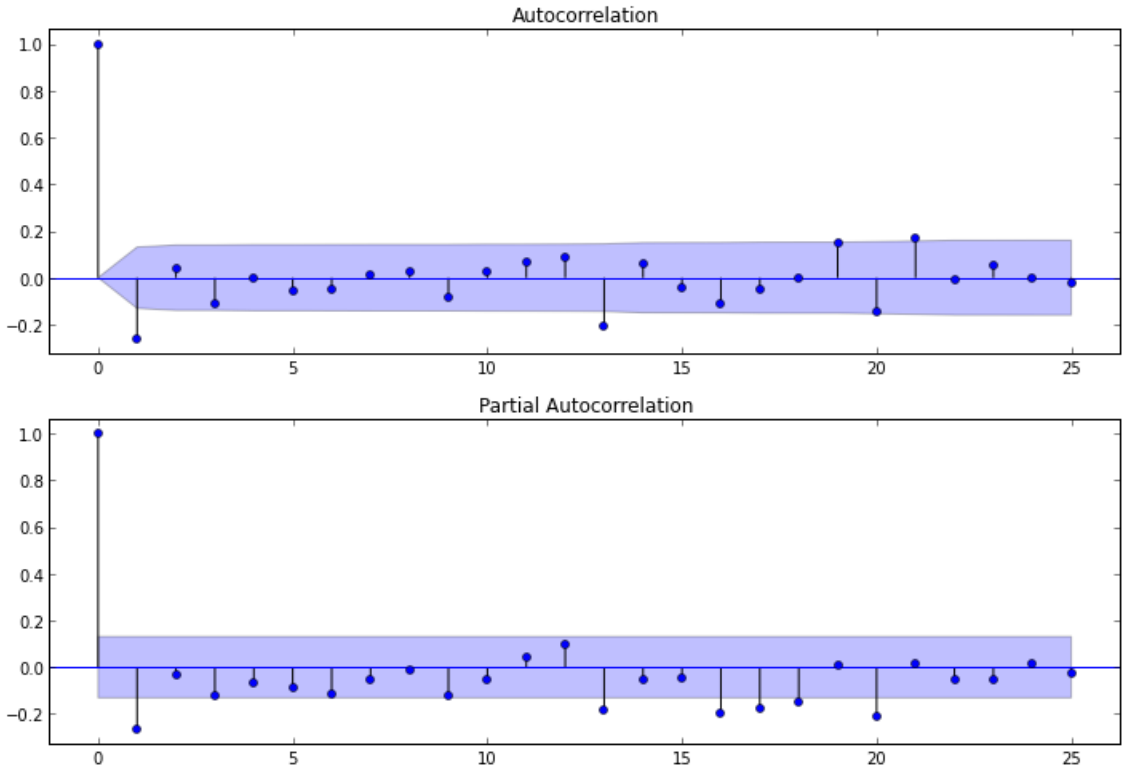
Параметр d есть и он равет 1, осталось определить p и q. Для их определения нам надо изучить авторкорреляционную(ACF) и частично автокорреляционную(PACF) функции для ряда первых разностей.

ACF поможет нам определить q, т. к. по ее коррелограмме можно определить количество автокорреляционных коэффициентов сильно отличных от 0 в модели MA

PACF поможет нам определить p, т. к. по ее коррелограмме можно определить максимальный номер коэффициента сильно отличный от 0 в модели AR.

Чтобы построить соответствующие коррелограммы, в пакете statsmodels имеются следующие функции: plot_acf() и plot_pacf(). Они выводят графики ACF и PACF, у которых по оси X откладываются номера лагов, а по оси Y значения соответствующих функций. Нужно отметить, что количество лагов в функциях и определяет число значимых коэффициентов. Итак, наши функции выглядят так:

```
ig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(otg1diff.values.squeeze(), lags=25, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(otg1diff, lags=25, ax=ax2)
```



После изучения коррелограммы PACF можно сделать вывод, что p = 1, т.к. на ней только 1 лаг сильно отличнен от нуля. По коррелограмме ACF можно увидеть, что q = 1, т.к. после лага 1 значении функций резко падают.
Итак, когда известны все параметры можно построить модель, но для ее построения мы возмем не все данные, а только часть. Данные из части не попавших в модель мы оставим для проверки точности прогноза нашей модели:

```
src_data_model = otg[:'2013-05-26']
```

```
model = sm.tsa.ARIMA(src_data_model, order=(1,1,1), freq='W').fit(full_output=False,
disp=0)
```

Параметр trend отвечает за наличие константы в модели. Выведем информацаю по получившейся модели:

```
print model.summary()
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                   D.y   No. Observations:                  194
Model:                ARIMA(1, 1, 1)   Log Likelihood               -2326.028
Method:                       css-mle   S.D. of innovations          38615.075
Date:                Tue, 24 Dec 2013   AIC                           4660.057
Time:                        02:12:47   BIC                           4673.128
Sample:                    09-13-2009   HQIC                          4665.350
                         - 05-26-2013
==============================================================================
                 coef    std err          z      P>|z|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const       1588.2266    142.728     11.128      0.000      1308.484  1867.969
ar.L1.D.y      0.6660      0.055     12.166      0.000         0.559     0.773
ma.L1.D.y     -1.0000      0.014    -72.214      0.000        -1.027    -0.973
                                    Roots
=============================================================================
                 Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.5015           +0.0000j            1.5015            0.0000
MA.1            1.0000           +0.0000j            1.0000            0.0000
-----------------------------------------------------------------------------
```

Как видно из данной информации в нашей модели все коэффициенты значимые и можно перейти к оценке модели.

Анализ и оценка модели

Проверим остатки данной модели на соответствие «белому шуму», а также проанализируем коррелограму остатков, так как это может нам помочь в определении важных для включения и прогнозирования элементов регрессии.

Итак первое, что мы сделаем это проведем Q-тест Льюнга — Бокса для проверки гипотезы о том, что остатки случайны, т. е. являются «белым шумом». Данный тест проводится на остатках модели ARIMA. Таким образом, нам надо сначала получить остатки модели и построить для них ACF, а затем к получившимся коэффициентам приметить тест. С помощью statsmadels это можно сделать так:

```
q_test = sm.tsa.stattools.acf(model.resid, qstat=True) #свойство resid, хранит остатки
модели, qstat=True, означает что применяем указынный тест к коэф-ам
```

```
print DataFrame({'Q-stat':q_test[1], 'p-value':q_test[2]})
```

Результат

Значение данной статистики и p-values, свидетельствуют о том, что гипотеза о случайности остатков не отвергается, и скорее всего данный процесс представляет «белый шум». Теперь давайте расчитаем коэффициент детерминации R^2, чтобы понять какой процент наблюдений описывает данная модель:

```
pred = model.predict('2013-05-26','2014-12-31', typ='levels')
```

```
trn = otg['2013-05-26':]
```

```
r2 = r2_score(trn, pred[1:32])
```

```
print 'R^2: %1.2f' % r2
```

R^2: -0.03

Среднеквадратичное отклонение[2] нашей модели:

```
metrics.rmse(trn,pred[1:32])
```

80919.057367642512

Средняя абсолютная ошибка[2] прогноза:

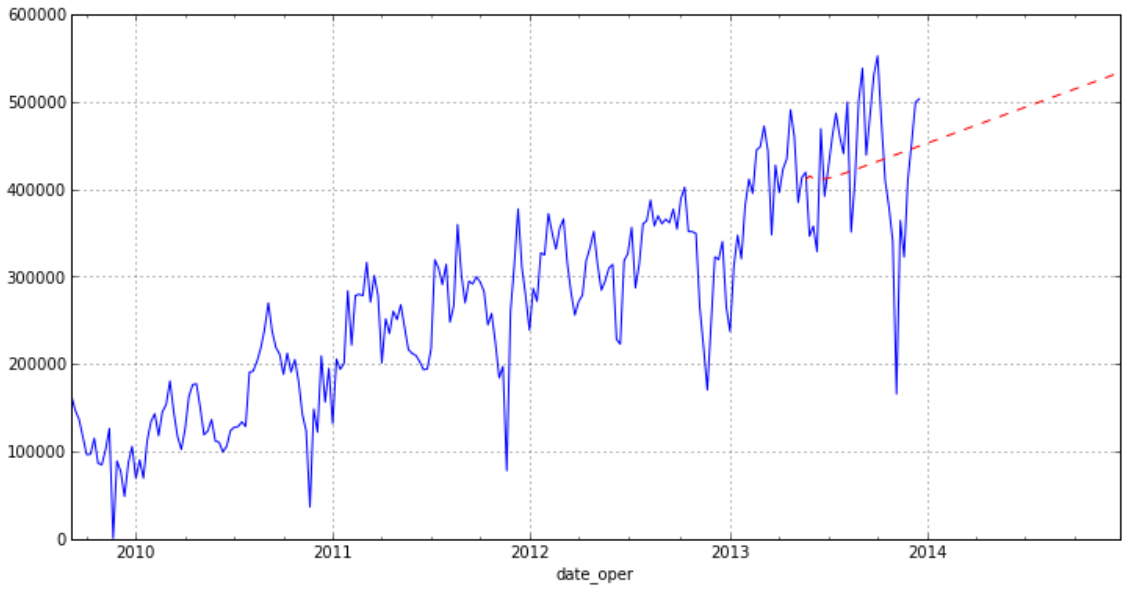
```
metrics.mae(trn,pred[1:32])
```

63092.763277651895

Осталось нарисовать наш прогноз на графике:

```
otg.plot(figsize=(12,6))
```

```
pred.plot(style='r--')
```


600000
500000
400000
300000
200000
100000
0
2010
2011
2012
2013
2014
date_oper