计算机 22A4 2022015232 孙春辉
2025 年 3 月 30 日 星期 yi 第七次作业

Report 7

Part 1: Course exercises

1.1: 时间序列

1.2:   时间戳

1.3:   基础频率

1.4:   重采样

1.5:   Period

2.1:   T

2.2:   T

2.3:   F

2.4:   T

2.5:   F

3.1:   A

3.2:   D

3.3:   D

3.4   B

3.5:   C

4.1: 时间戳表示带时区的特定的日期时间；时间差表示绝对的持续时间；时期是由时间点及其相关频率定义的时间跨度。

4.2: 如果是将高频率数据聚合到低频率，比如将每日采集的频率变成每月采集，则称为降采样；如果将低频率数据转换到高频率数据，比如将每月采集的频率变成每日采集，则称为升采样。

5:

（1）、运行产生 ValueError 异常，主要是因为 date_range()函数中必须传入 start、end、periods、freq 中至少三个参数，而代码中只传入了 start 和 freq 参数。
（2）、运行结果为 5
（3）、运行出现异常，因为日期字符串的格式不能够被正确解析。

Part 2:

Re-implement the codes from the stock prediction and analysis Demo in the text book page 223-228, summarize the results and screenshot the codes in the report.

计算机 22A4 2022015232 孙春辉
2025 年 3 月 30 日 星期 yi 第七次作业

```python
# 查询2022年3月到5月的报警记录
final_alarm_record = final_alarm_record.sort_index()
record_3_5 = final_alarm_record.loc['2022-03-01': '2022-05-31'].sort_index()
record_3_5
```

	OFFENSE_TYPE_ID	OFFENSE_CATEGORY_ID	GEO_LON	GEO_LAT	IS_CRIME	IS_TRAFFIC
REPORTED_DATE						
2022-03-01 00:07:00	traf-other	all-other-crimes	-104.799964	39.798282	1	0
2022-03-01 00:07:00	assault-dv	other-crimes-against-persons	-104.889641	39.752364	1	0
2022-03-01 00:26:00	traffic-accident-hit-and-run	traffic-accident	-104.846448	39.779922	0	1
2022-03-01 00:30:00	criminal-mischief-other	public-disorder	-105.057143	39.654277	1	0
2022-03-01 00:42:00	obstructing-govt-operation	all-other-crimes	-105.039012	39.628846	1	0
...	...	...	...	...	...	...
2022-05-31 23:22:00	vehicular-eluding-no-chase	all-other-crimes	-104.978375	39.782895	1	0
2022-05-31 23:24:00	criminal-trespassing	all-other-crimes	-104.999951	39.753054	1	0
2022-05-31 23:31:00	burglary-residence-by-force	burglary	-105.034769	39.723257	1	0
2022-05-31 23:49:00	traffic-accident	traffic-accident	-104.882790	39.777120	0	1
2022-05-31 23:50:00	theft-shoplift	larceny	-104.897946	39.769688	1	0

21778 rows × 6 columns

```python
# 根据OFFENSE_CATEGORY_ID列分组统计数量，得到每种类别报警记录的数量
result = record_3_5.groupby('OFFENSE_CATEGORY_ID').count()
new_result = result.sort_values(by='OFFENSE_TYPE_ID')
new_result
```

	OFFENSE_TYPE_ID	GEO_LON	GEO_LAT	IS_CRIME	IS_TRAFFIC
OFFENSE_CATEGORY_ID					
murder	15	15	15	15	15
arson	25	25	25	25	25
robbery	271	271	271	271	271
white-collar-crime	301	301	301	301	301
drug-alcohol	318	318	318	318	318
aggravated-assault	541	541	541	541	541
burglary	1051	1051	1051	1051	1051
auto-theft	1175	1175	1175	1175	1175
other-crimes-against-persons	1225	1225	1225	1225	1225
theft-from-motor-vehicle	1842	1842	1842	1842	1842
larceny	2060	2060	2060	2060	2060
public-disorder	2421	2421	2421	2421	2421
all-other-crimes	4605	4605	4605	4605	4605
traffic-accident	5928	5928	5928	5928	5928

```python
# 获取第一条报警记录
new_result.iloc[0:1]
```

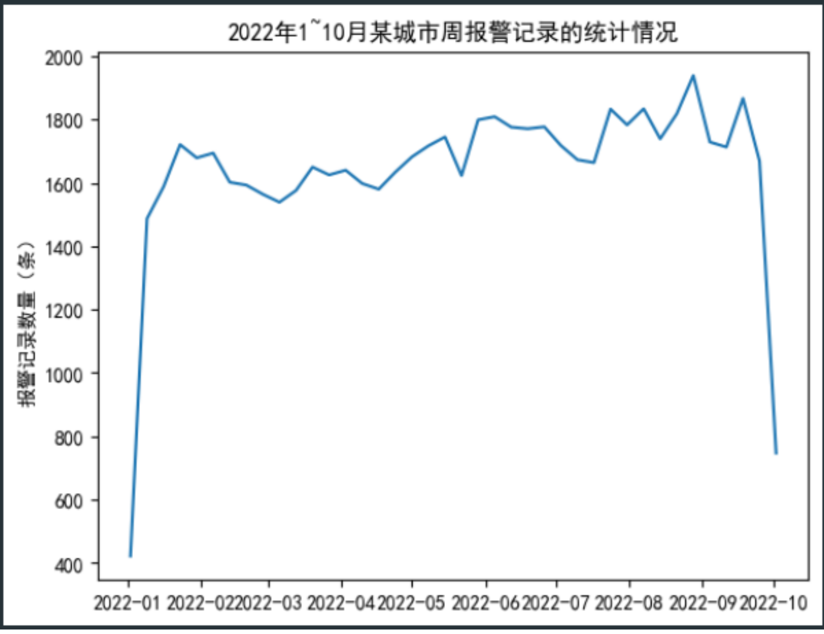
	OFFENSE_TYPE_ID	GEO_LON	GEO_LAT	IS_CRIME	IS_TRAFFIC
OFFENSE_CATEGORY_ID					
murder	15	15	15	15	15

```python
# 获取最后一条报警记录
new_result.iloc[-1:-2:-1]
```

	OFFENSE_TYPE_ID	GEO_LON	GEO_LAT	IS_CRIME	IS_TRAFFIC
OFFENSE_CATEGORY_ID					
traffic-accident	5928	5928	5928	5928	5928

计算机 22A4 2022015232 孙春辉
2025 年 3 月 30 日 星期 yi 第七次作业

```python
# 绘制折线图
import matplotlib.pyplot as plt
# 设置字体为简黑
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.plot(weekly_alarm.index.values, weekly_alarm.values)
plt.title('2022年1~10月某城市周报警记录的统计情况')
plt.ylabel('报警记录数量（条）')
plt.show()
```



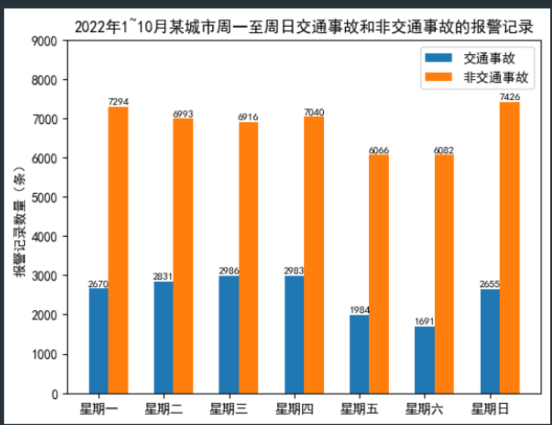

```python
# 筛选出交通事故的记录
left = alarm_record[alarm_record['IS_TRAFFIC'] == 1]['REPORTED_DATE'].dt.weekday.value_counts()
left = left.sort_index()
left
```

```
REPORTED_DATE
0    2670
1    2831
2    2986
3    2983
4    1984
5    1691
6    2655
Name: count, dtype: int64
```

```python
# 筛选出非交通事故的记录
right = alarm_record[alarm_record['IS_CRIME'] == 1]['REPORTED_DATE'].dt.weekday.value_counts()
right = right.sort_index()
right
```

```
REPORTED_DATE
0    7294
1    6993
2    6916
3    7040
4    6066
5    6082
6    7426
Name: count, dtype: int64
```

```python
import numpy as np
x = np.arange(right.size)
bar_width = 0.3    # 柱形的宽度
rect_blue = plt.bar(x, left.values, tick_label=["星期一", "星期二", "星期三", "星期四",
                                               "星期五", "星期六", "星期日"], width=bar_width)
rect_orange = plt.bar(x+bar_width, right.values, width=bar_width)
# 添加标题
plt.title('2022年1~10月某城市周一至周日交通事故和非交通事故的报警记录')
# 设置y轴的标签和刻度范围
plt.ylabel('报警记录数量（条）')
plt.ylim(0, 9000)
# 添加注释文本
def autolabel(rects):
    """在每个柱形上方添加注释文本"""
    for rect in rects:
        rect_height = rect.get_height()
        rect_x = rect.get_x()
        rect_width = rect.get_width()
        plt.text(rect_x + rect_width / 2, rect_height + 30,
                 s='{}'.format(rect_height),
                 ha='center', va='bottom', fontsize=8)
autolabel(rect_blue)
autolabel(rect_orange)
# 添加图例
plt.legend([rect_blue, rect_orange], ['交通事故', '非交通事故'])
plt.show()
```



我们可以根据数据可以看出，在所有的报警记录中，交通事故的报警记录占总报警记录近百分之三十左右，并且在 2 月到 9 月之间，所有的报警记录数量基本稳定在 1600 条每月左右。

Part 3:

从 Moodle 下载‘Report7-AirPassengers.csv’文件并完成以下操作.

1. 读取并显示数据;

```
import numpy as np
import pandas as pd
from datetime import datetime
df = pd.read_csv('./Report7-AirPassengers.csv')
df.index = pd.to_datetime(df.index)
ts = df
print(ts.head())
print(ts.head().index)
```

```
1970-01-01 00:00:00.000000001  1949-02         118
1970-01-01 00:00:00.000000002  1949-03         132
1970-01-01 00:00:00.000000003  1949-04         129
1970-01-01 00:00:00.000000004  1949-05         121
DatetimeIndex([          '1970-01-01 00:00:00',
               '1970-01-01 00:00:00.000000001',
               '1970-01-01 00:00:00.000000002',
               '1970-01-01 00:00:00.000000003',
               '1970-01-01 00:00:00.000000004'],
              dtype='datetime64[ns]', freq=None)
```

2. 查看 1955 年数据;

```
# ts = df['#Passengers']
ts['1955']
# df_1955 = ts.loc['1955']
# df_1955
```

```
Month
1955-01-01    242
1955-02-01    233
1955-03-01    267
1955-04-01    269
1955-05-01    270
1955-06-01    315
1955-07-01    364
1955-08-01    347
1955-09-01    312
1955-10-01    274
1955-11-01    237
1955-12-01    278
Name: #Passengers, dtype: int64
```

3. 对 1950 年 1 月到 8 月数据进行切片;

```
ts['1950-01':'1950-08-31']
```

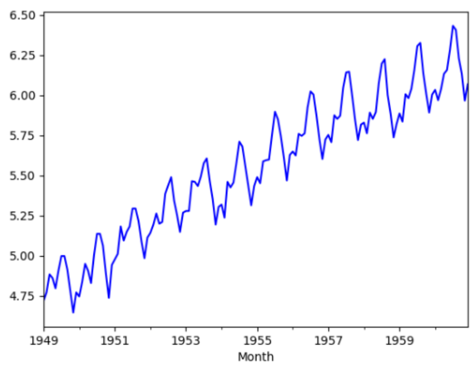
```
Month
1950-01-01    115
1950-02-01    126
1950-03-01    141
1950-04-01    135
1950-05-01    125
1950-06-01    149
1950-07-01    170
1950-08-01    170
Name: #Passengers, dtype: int64
```

4. 利用观察法进行数据平稳性检测;

test_stationary.py

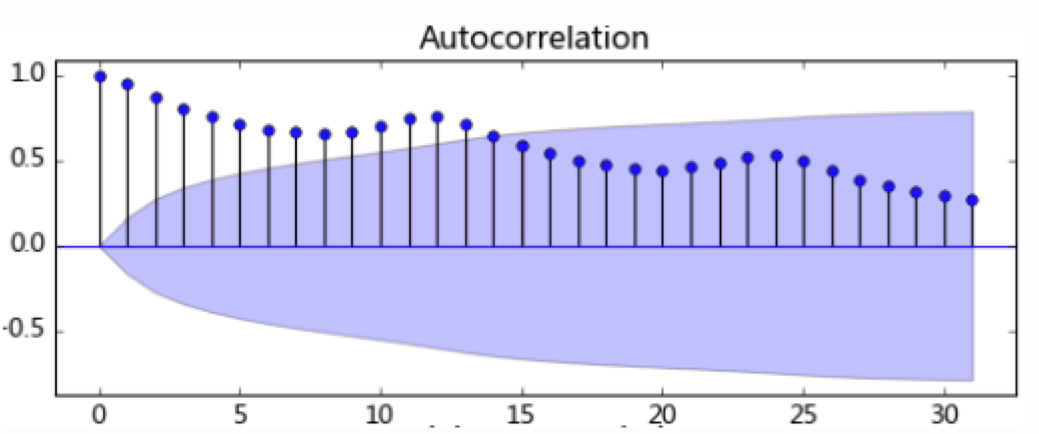
```
from statsmodels.tsa.stattools import adfuller
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# 移动平均图
def draw_trend(timeSeries, size):
    f = plt.figure(facecolor='white')
    # 对 size 个数据进行移动平均
    rol_mean = timeSeries.rolling(window=size).mean()
    # 对 size 个数据进行加权移动平均
    rol_weighted_mean = pd.ewma(timeSeries, span=size)

    timeSeries.plot(color='blue', label='Original')
    rolmean.plot(color='red', label='Rolling Mean')
    rol_weighted_mean.plot(color='black', label='Weighted Rolling
Mean')
    plt.legend(loc='best')
    plt.title('Rolling Mean')
    plt.show()

def draw_ts(timeSeries):
    f = plt.figure(facecolor='white')
    timeSeries.plot(color='blue')
    plt.show()
def testStationarity(ts):
    dftest = adfuller(ts)
    # 对上述函数求得的值进行语义描述
```

```
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    return dfoutput

# 自相关和偏相关图，默认阶数为 31 阶
def draw_acf_pacf(ts, lags=31):
    f = plt.figure(facecolor='white')
    ax1 = f.add_subplot(211)
    plot_acf(ts, lags=31, ax=ax1)
    ax2 = f.add_subplot(212)
    plot_pacf(ts, lags=31, ax=ax2)
    plt.show()
```



5. 利用单位根检验方法对数据进行平稳性检测;

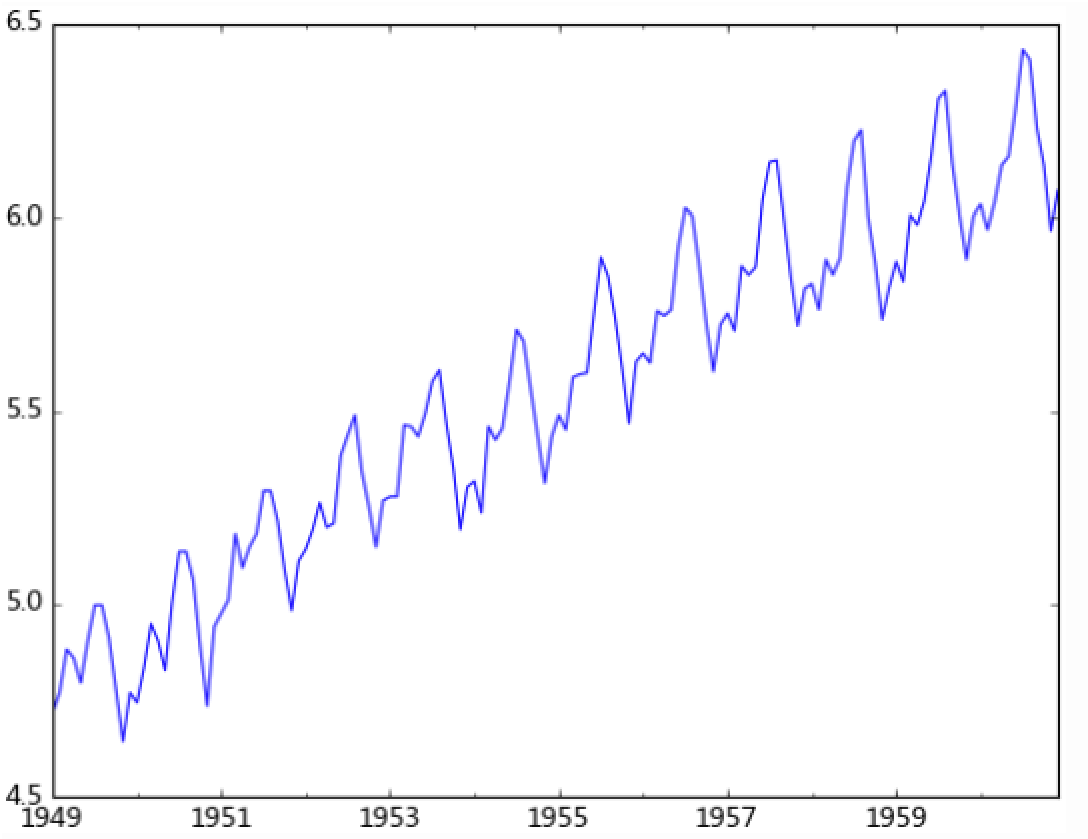
代码同上



6. 对数据进行基于对数变换的平稳性处理;

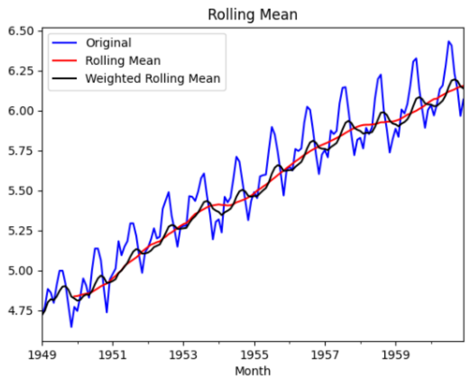
main.py

```
import test_stationarity
import numpy as np
import pandas as pd

df = pd.read_csv(r'D:\schoolzwu\AI\machine-study\data_analyse\3-30\Report7-AirPassengers.csv', encoding='utf-8', index_col='Month')
df.index = pd.to_datetime(df.index)
ts = df['#Passengers']  # 生成 pd.Series 对象

ts_log = np.log(ts)
test_stationarity.draw_ts(ts_log)
```



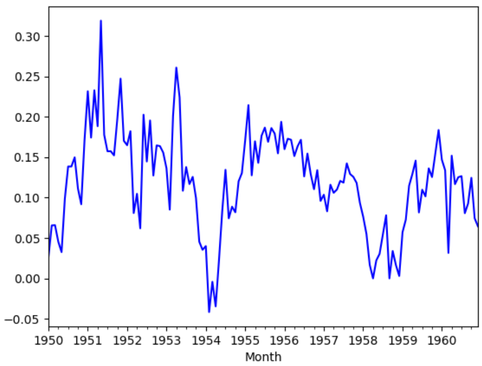
7. 对数据进行基于平滑法的平稳性处理;

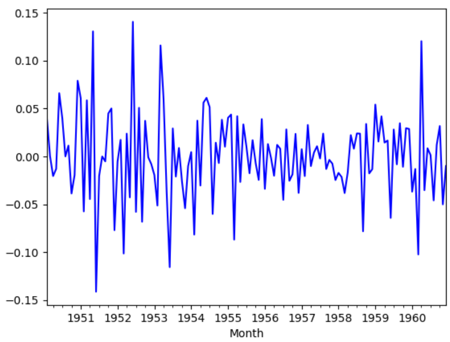
```python
test_stationarity.draw_trend(ts_log, 12)
```



8. 对数据进行基于差分的平稳性处理；

```python
diff_12 = ts_log.diff(12)
diff_12.dropna(inplace=True)
diff_12_1 = diff_12.diff(1)
diff_12_1.dropna(inplace=True)
test_stationarity.draw_ts(diff_12)
print(test_stationarity.testStationarity(diff_12_1))
```



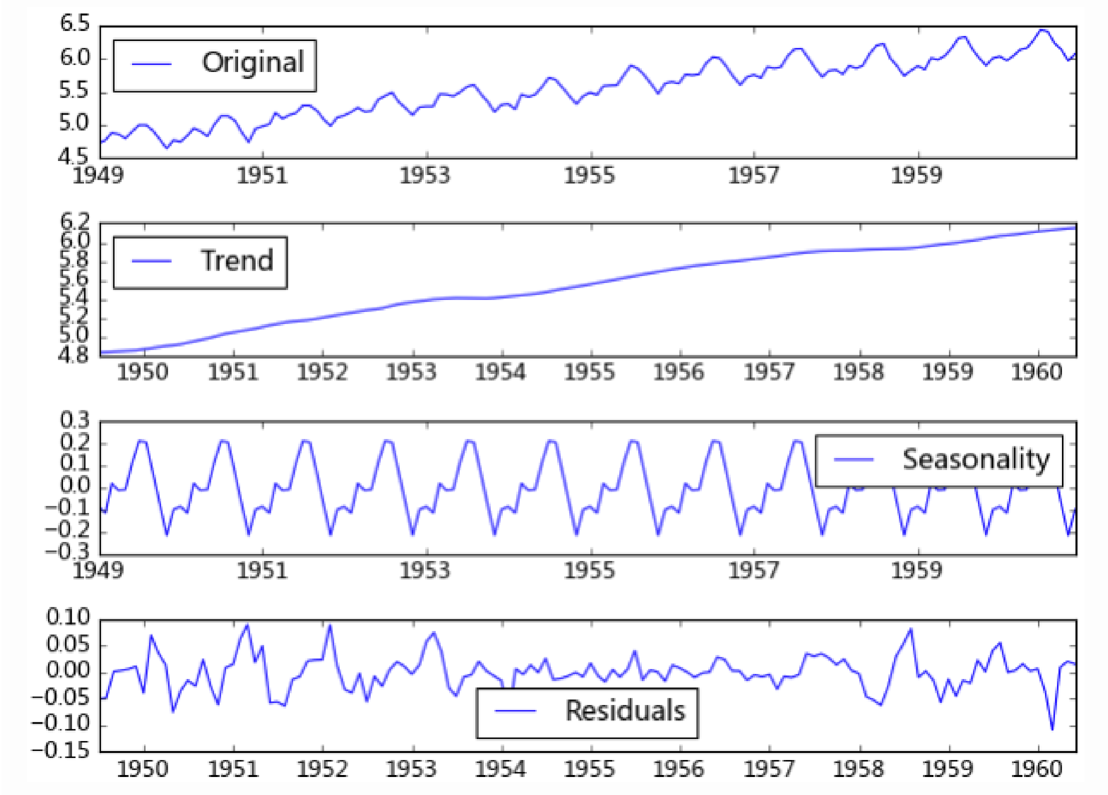



```
Test Statistic                  -4.443325
p-value                          0.000249
#Lags Used                      12.000000
Number of Observations Used    118.000000
Critical Value (1%)             -3.487022
Critical Value (5%)             -2.886363
Cri                               580009
dty  Focus folder in explorer (ctrl + click)
```

9. 对数据进行基于分解的平稳性处理；

```python
import test_stationarity
```

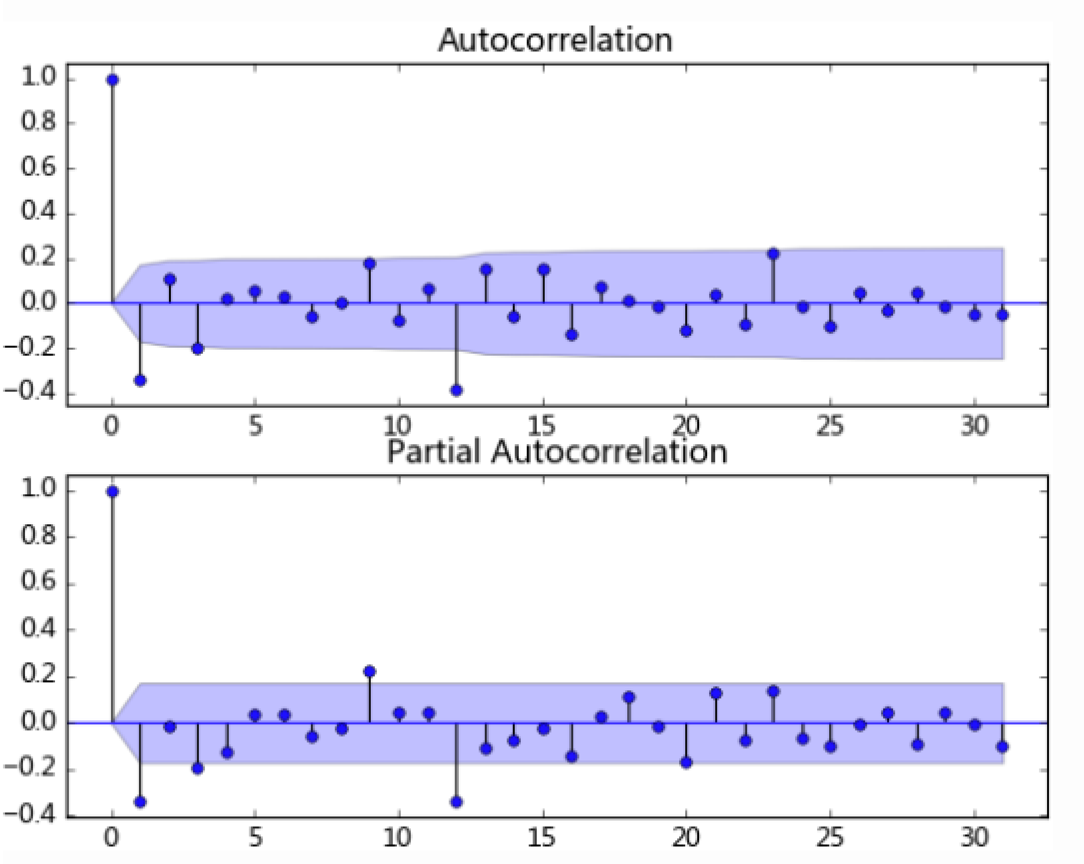
```
import numpy as np
import pandas as pd
from statsmodels.tsa.arima_model import ARMA

df = pd.read_csv(r'D:\schoolzwu\AI\machine-
study\data_analyse\3-30\Report7-AirPassengers.csv',
encoding='utf-8', index_col='Month')
df.index = pd.to_datetime(df.index)
ts = df['#Passengers']  # 生成pd.Series 对象

ts_log = np.log(ts)
# 差分
# diff_12 = ts_log.diff(12)
# diff_12.dropna(inplace=True)
# diff_12_1 = diff_12.diff(1)
# diff_12_1.dropna(inplace=True)
# test_stationarity.draw_ts(diff_12)
# print(test_stationarity.testStationarity(diff_12_1))

# 分解
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(ts_log, model="additive")

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid
```



10. 对数据进行基于 1 阶差分的模型识别;

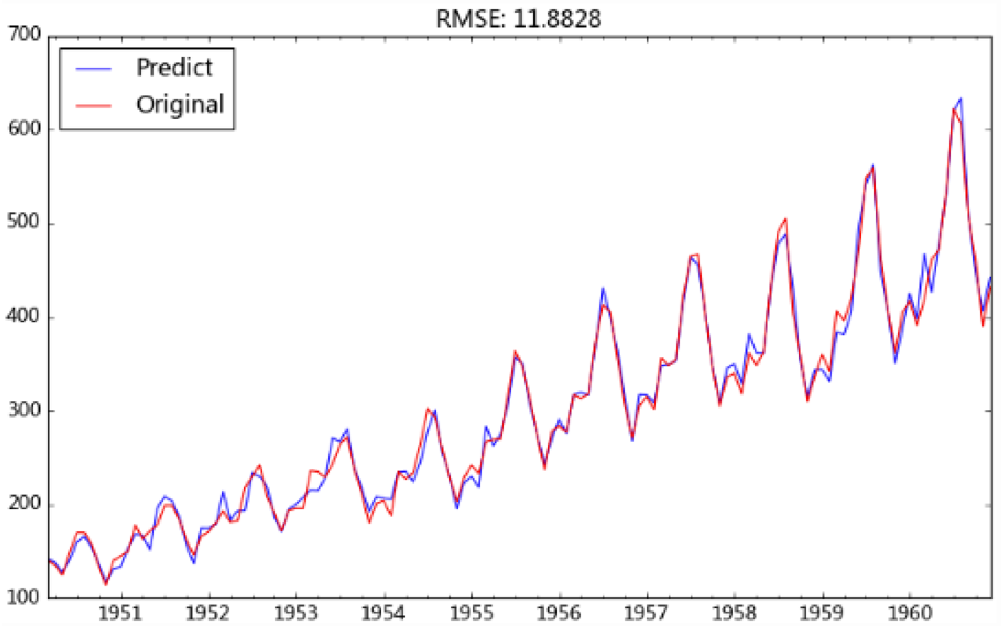
```
rol_mean = ts_log.rolling(window=12).mean()
rol_mean.dropna(inplace=True)
ts_diff_1 = rol_mean.diff(1)
ts_diff_1.dropna(inplace=True)
print(test_stationarity.testStationarity(ts_diff_1))
```

```
PS D:\schoolzwu\AI\machine-study\data_analyse\
Test Statistic                  -2.709577
p-value                          0.072396
#Lags Used                      12.000000
Number of Observations Used    119.000000
Critical Value (1%)             -3.486535
Critical Value (5%)             -2.886151
Critical Value (10%)            -2.579896
dtype: float64
```



11. 对数据进行逆变换还原后进行基于均方根误差的样本拟合;

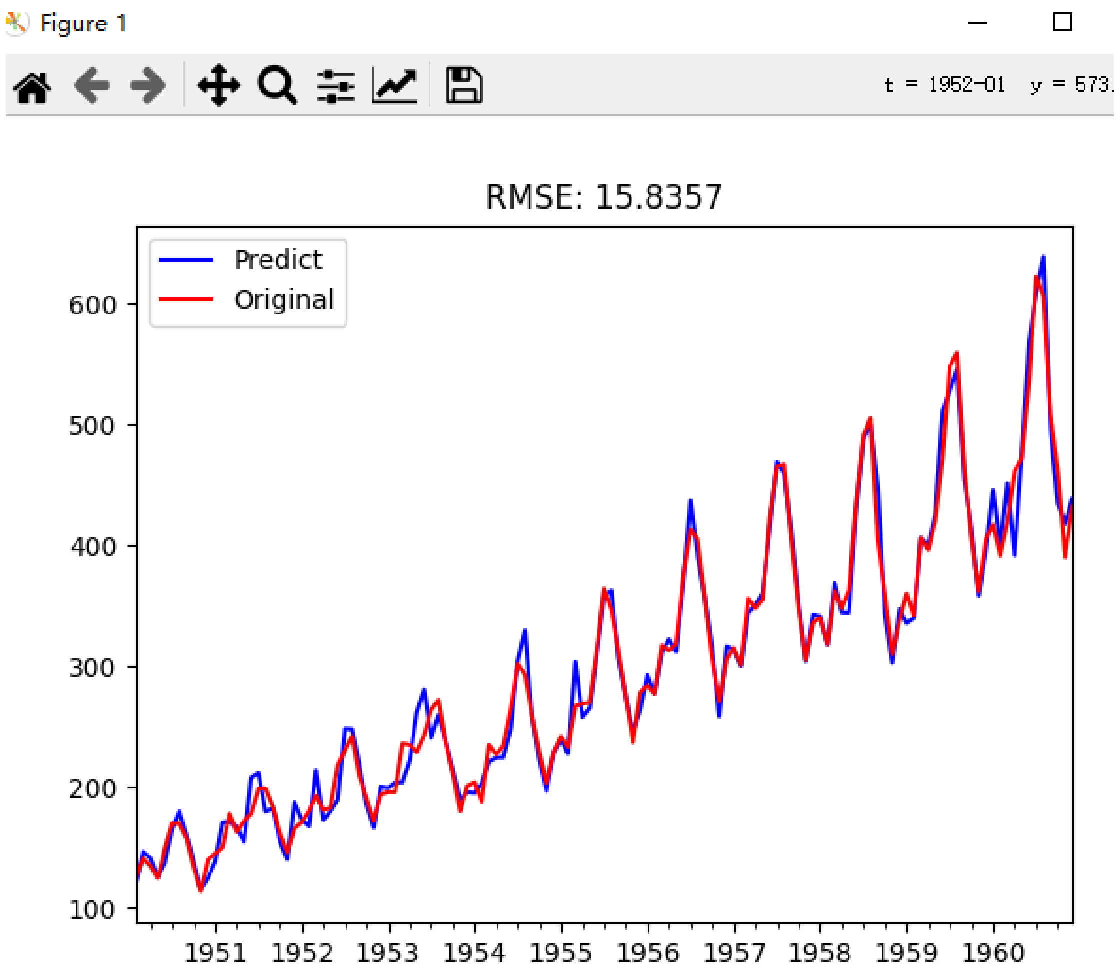
```
ts = ts[log_recover.index]  # 过滤没有预测的记录
plt.figure(facecolor='white')
log_recover.plot(color='blue', label='Predict')
ts.plot(color='red', label='Original')
plt.legend(loc='best')
plt.title('RMSE: %.4f'% np.sqrt(sum((log_recover-
ts)**2)/ts.size))
plt.show()
```



12. 分离差分，完善 ARIMA 模型；

```
model = ARIMA(ts_diff_2, order=(1, 1, 0))
# order=(p,d,q) 其中 p=1 是 AR 阶数，d=1 是差分阶数，q=1 是 MA 阶数
result_arma = model.fit()
predict_ts = result_arma.predict()
# 一阶差分还原
diff_shift_ts = ts_diff_1.shift(1)
diff_recover_1 = predict_ts.add(diff_shift_ts)
# 再次一阶差分还原
rol_shift_ts = rol_mean.shift(1)
diff_recover = diff_recover_1.add(rol_shift_ts)
# 移动平均还原
rol_sum = ts_log.rolling(window=11).sum()
rol_recover = diff_recover*12 - rol_sum.shift(1)
# 对数还原
log_recover = np.exp(rol_recover)
log_recover.dropna(inplace=True)

ts = ts[log_recover.index]  # 过滤没有预测的记录
plt.figure(facecolor='white')
```

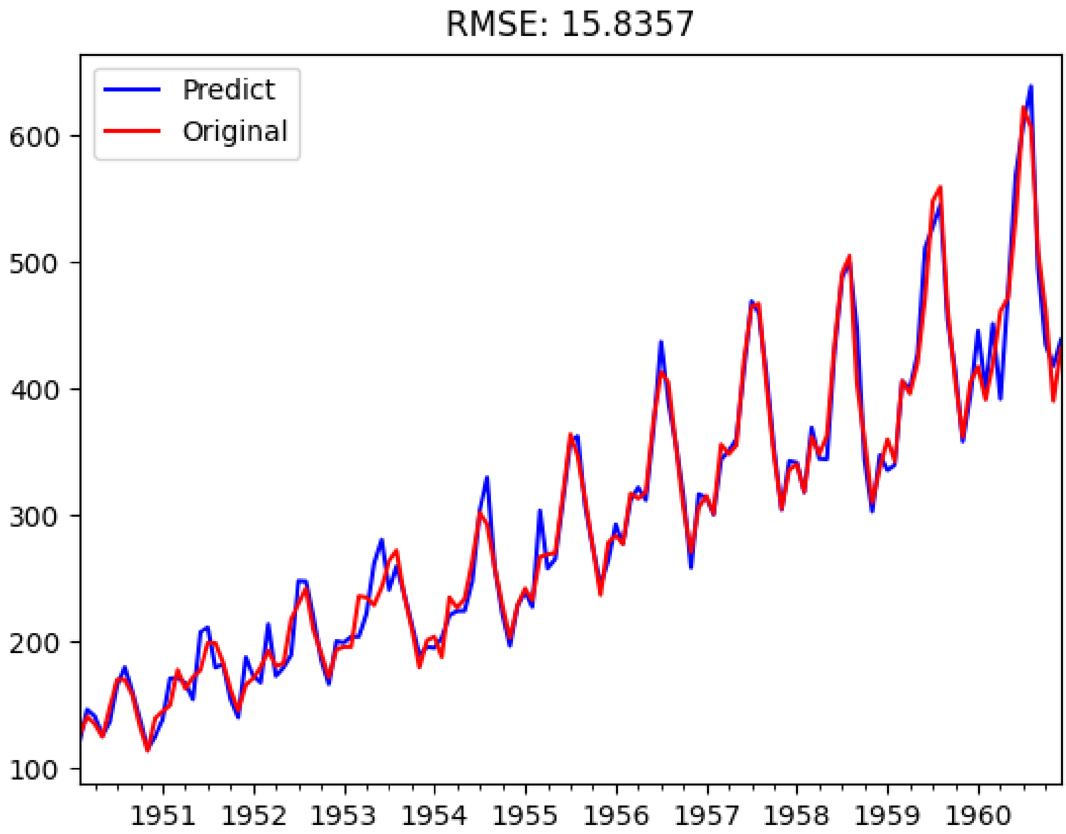
```
log_recover.plot(color='blue', label='Predict')
ts.plot(color='red', label='Original')
plt.legend(loc='best')
plt.title('RMSE: %.4f'% np.sqrt(sum((log_recover-ts)**2)/ts.size))
plt.show()
```



13. 以周为单位对数据进行滚动预测;

```
from dateutil.relativedelta import relativedelta
def _add_new_data(ts, dat, type='day'):
    if type == 'day':
        new_index = ts.index[-1] + relativedelta(days=1)
    elif type == 'month':
        new_index = ts.index[-1] + relativedelta(months=1)
    ts[new_index] = dat
```

```
def add_today_data(model, ts,  data, d, type='day'):
    _add_new_data(ts, data, type)  # 为原始序列添加数据
    # 为滞后序列添加新值
    d_ts = diff_ts(ts, d)
    model.add_today_data(d_ts[-1], type)

def forecast_next_day_data(model, type='day'):
    if model == None:
        raise ValueError('No model fit before')
    fc = model.forecast_next_day_value(type)
    return predict_diff_recover(fc, [12, 1])

ts_train = ts_log[:'1956-12']
ts_test = ts_log['1957-1':]

diffed_ts = diff_ts(ts_train, [12, 1])
forecast_list = []

for i, dta in enumerate(ts_test):
    if i%7 == 0:
        model = arima_model(diffed_ts)
        model.certain_model(1, 1)
    forecast_data = forecast_next_day_data(model, type='month')
    forecast_list.append(forecast_data)
    add_today_data(model, ts_train, dta, [12, 1], type='month')

predict_ts = pd.Series(data=forecast_list, index=ts['1957-1':].index)
log_recover = np.exp(predict_ts)
original_ts = ts['1957-1':]
```



Ref：https://www.cnblogs.com/foley/p/5582358.html（任务实现请参考该网址）