Лабораторная работа №8. Рекуррентные нейронные сети для анализа временных рядов

Данные: Набор данных для прогнозирования временных рядов, который состоит из среднемесячного числа пятен на солнце, наблюдаемых с января 1749 по август 2017. Данные в виде csv-файла можно скачать на сайте Kaggle -> https://www.kaggle.com/robervalt/sunspots/ (https://www.kaggle.com/robervalt/sunspots/)

In [1]:
```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/au
th?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.goog
leusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&r
esponse_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fa
uth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20
https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20ht
tps%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly
(https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6b
n6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_u
ri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=emai
l%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2
f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis
.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fpeopleapi.readonly)

Enter your authorization code:
··········
Mounted at /content/drive
```

In [33]:
```
1 !pip install tensorflow==1.14
```

```
Collecting tensorflow==1.14
  Downloading https://files.pythonhosted.org/packages/de/f0/96fb2e04
12ae9692dbf400e5b04432885f677ad6241c088ccc5fe7724d69/tensorflow-1.14
.0-cp36-cp36m-manylinux1_x86_64.whl
(https://files.pythonhosted.org/packages/de/f0/96fb2e0412ae9692dbf40
0e5b04432885f677ad6241c088ccc5fe7724d69/tensorflow-1.14.0-cp36-cp36m
-manylinux1_x86_64.whl) (109.2MB)
     |████████████████████████████████| 109.2MB 95kB/s
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/pytho
n3.6/dist-packages (from tensorflow==1.14) (1.27.2)
Requirement already satisfied: keras-applications>=1.0.6 in /usr/loc
al/lib/python3.6/dist-packages (from tensorflow==1.14) (1.0.8)
Requirement already satisfied: numpy<2.0,>=1.14.5 in /usr/local/lib/
python3.6/dist-packages (from tensorflow==1.14) (1.18.2)
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/pytho
n3.6/dist-packages (from tensorflow==1.14) (1.12.1)
Requirement already satisfied: protobuf>=3.6.1 in /usr/local/lib/pyt
```

```
hon3.6/dist-packages (from tensorflow==1.14) (3.10.0)
Collecting tensorboard<1.15.0,>=1.14.0
  Downloading https://files.pythonhosted.org/packages/91/2d/2ed263449a078cd9c8a9ba50ebd50123adf1f8cfbea1492f9084169b89d9/tensorboard-1.14.0-py3-none-any.whl (https://files.pythonhosted.org/packages/91/2d/2ed263449a078cd9c8a9ba50ebd50123adf1f8cfbea1492f9084169b89d9/tensorboard-1.14.0-py3-none-any.whl) (3.1MB)
     |████████████████████████████████| 3.2MB 42.3MB/s
Collecting tensorflow-estimator<1.15.0rc0,>=1.14.0rc0
  Downloading https://files.pythonhosted.org/packages/3c/d5/21860a5b11caf0678fbc8319341b0ae21a07156911132e0e71bffed0510d/tensorflow_estimator-1.14.0-py2.py3-none-any.whl (https://files.pythonhosted.org/packages/3c/d5/21860a5b11caf0678fbc8319341b0ae21a07156911132e0e71bffed0510d/tensorflow_estimator-1.14.0-py2.py3-none-any.whl) (488kB)
     |████████████████████████████████| 491kB 50.2MB/s
Requirement already satisfied: google-pasta>=0.1.6 in /usr/local/lib/python3.6/dist-packages (from tensorflow==1.14) (0.2.0)
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==1.14) (0.9.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6/dist-packages (from tensorflow==1.14) (1.1.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==1.14) (1.12.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==1.14) (1.1.0)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.6/dist-packages (from tensorflow==1.14) (0.34.2)
Requirement already satisfied: astor>=0.6.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==1.14) (0.8.1)
Requirement already satisfied: gast>=0.2.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==1.14) (0.3.3)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from keras-applications>=1.0.6->tensorflow==1.14) (2.10.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from protobuf>=3.6.1->tensorflow==1.14) (46.1.3)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow==1.14) (3.2.1)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-packages (from tensorboard<1.15.0,>=1.14.0->tensorflow==1.14) (1.0.1)
Installing collected packages: tensorboard, tensorflow-estimator, tensorflow
  Found existing installation: tensorboard 2.2.0
    Uninstalling tensorboard-2.2.0:
      Successfully uninstalled tensorboard-2.2.0
  Found existing installation: tensorflow-estimator 2.2.0rc0
    Uninstalling tensorflow-estimator-2.2.0rc0:
      Successfully uninstalled tensorflow-estimator-2.2.0rc0
  Found existing installation: tensorflow 2.2.0rc2
    Uninstalling tensorflow-2.2.0rc2:
```

```
    Uninstalling tensorflow-2.2.0rc2:
      Successfully uninstalled tensorflow-2.2.0rc2

Successfully installed tensorboard-1.14.0 tensorflow-1.14.0 tensorfl
ow-estimator-1.14.0
```

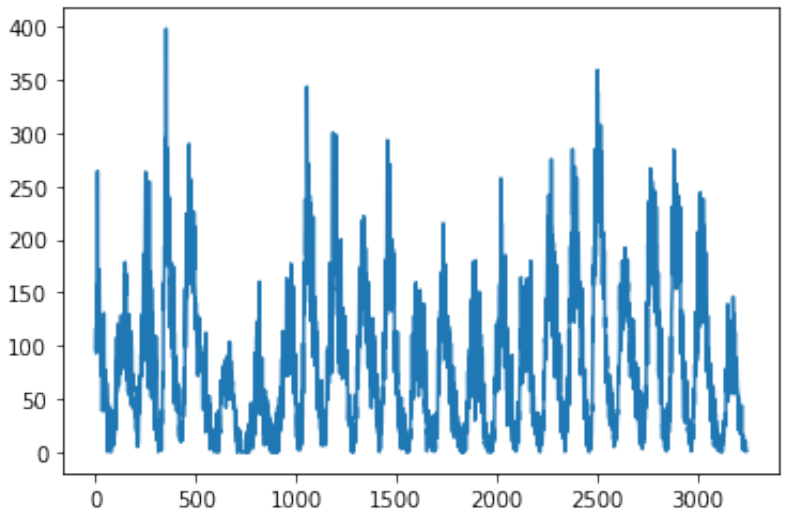
In [0]:

```
import numpy
import matplotlib.pyplot as plt
import pandas
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from pandas.plotting import autocorrelation_plot
from statsmodels.tsa.arima_model import ARIMA
```

Задание 1. Загрузите данные. Изобразите ряд в виде графика. Вычислите основные характеристики временного ряда (сезонность, тренд, автокорреляцию).

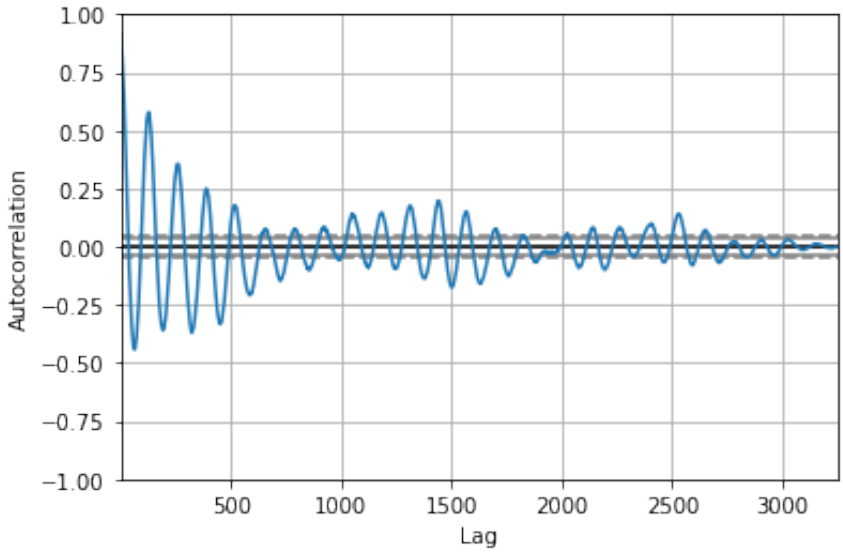
In [11]:

```
dataframe = pandas.read_csv('/content/drive/My Drive/Collab Data/S
plt.plot(dataframe)
plt.show()
```



In [12]:
```
1 autocorrelation_plot(dataframe)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe862419438>



Задание 2. Для прогнозирования разделите временной ряд на обучающую, валидационную и контрольную выборки.

In [13]:
```
1  dataset = dataframe.values
2  dataset = dataset.astype('float32')
3  scaler = MinMaxScaler(feature_range=(0, 1))
4  dataset = scaler.fit_transform(dataset)
5  print(len(dataset))
6
7  train_size = int(len(dataset) * 0.7)
8  validation_size = int(len(dataset) * 0.2)
9  test_size = len(dataset) - train_size - validation_size
10 train, validation, test = dataset[0:train_size,:], dataset[train_s
11 print(len(train), len(validation), len(test))
```

```
3249
2274 649 326
```

In [0]:
```
1 def create_dataset(dataset, look_back=1):
2     dataX, dataY = [], []
3     for i in range(len(dataset)-look_back-1):
4         a = dataset[i:(i+look_back), 0]
5         dataX.append(a)
6         dataY.append(dataset[i + look_back, 0])
7     return numpy.array(dataX), numpy.array(dataY)
```

In [0]:
```
trainX, trainY = create_dataset(train)
validationX, validationY = create_dataset(validation)
testX, testY = create_dataset(test)
```

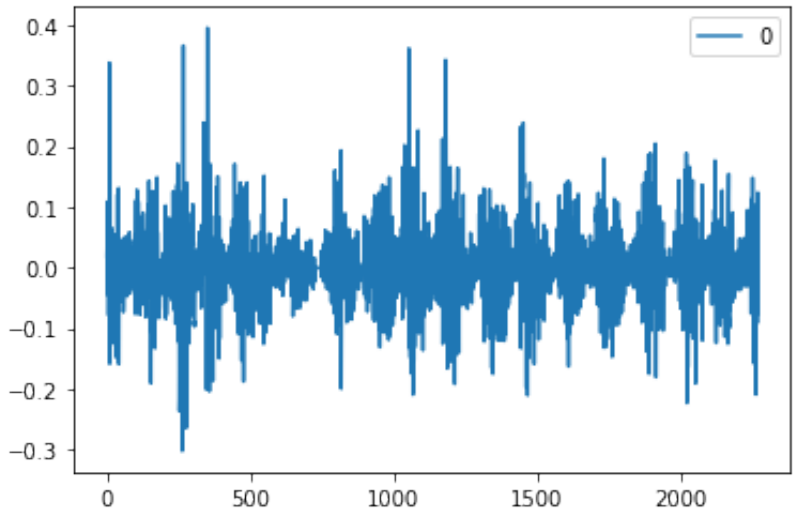
In [0]:
```
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1
validationX = numpy.reshape(validationX, (validationX.shape[0], 1,
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

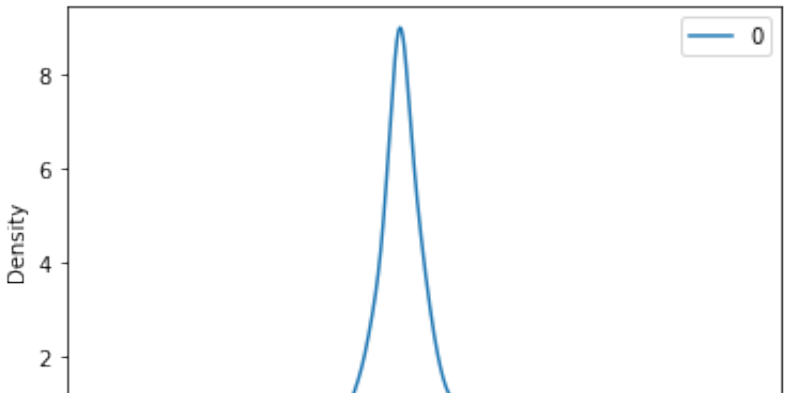
Задание 3. Примените модель ARIMA для прогнозирования значений данного временного ряда.

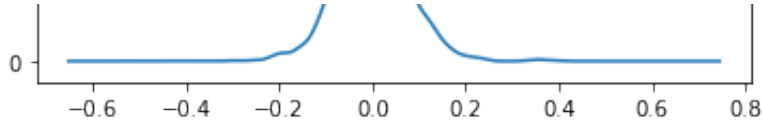
In [19]:
```
model = ARIMA(train, order=(5,1,0))
model_fit = model.fit(disp=0)
print(model_fit.summary())

residuals = pandas.DataFrame(model_fit.resid)
residuals.plot()
plt.show()
residuals.plot(kind='kde')
plt.show()
print(residuals.describe())
```

```
                             ARIMA Model Results
==============================================================================
==========
Dep. Variable:                    D.y   No. Observations:
2273
Model:                 ARIMA(5, 1, 0)   Log Likelihood
3045.325
Method:                       css-mle   S.D. of innovations
0.063
Date:                Mon, 06 Apr 2020   AIC
-6076.650
Time:                        19:10:39   BIC
-6036.548
Sample:                             1   HQIC
-6062.021

==============================================================================
==========
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
const       7.617e-05      0.001      0.124      0.902      -0.001
0.001
ar.L1.D.y     -0.4350      0.021    -20.801      0.000      -0.476
-0.394
```

```
ar.L2.D.y      -0.3022       0.023    -13.327       0.000       -0.347
-0.258
ar.L3.D.y      -0.2199       0.023     -9.526       0.000       -0.265
-0.175
ar.L4.D.y      -0.1225       0.023     -5.397       0.000       -0.167
-0.078
ar.L5.D.y      -0.0807       0.021     -3.854       0.000       -0.122
-0.040
                                    Roots
=============================================================================
=========
                  Real          Imaginary           Modulus
Frequency
-----------------------------------------------------------------------------
---------
AR.1              0.8289           -1.3397j            1.5754
-0.1618
AR.2              0.8289           +1.3397j            1.5754
0.1618
AR.3             -1.6684           -0.0000j            1.6684
-0.5000
AR.4             -0.7534           -1.5571j            1.7298
-0.3217
AR.5             -0.7534           +1.5571j            1.7298
0.3217
-----------------------------------------------------------------------------
---------
```







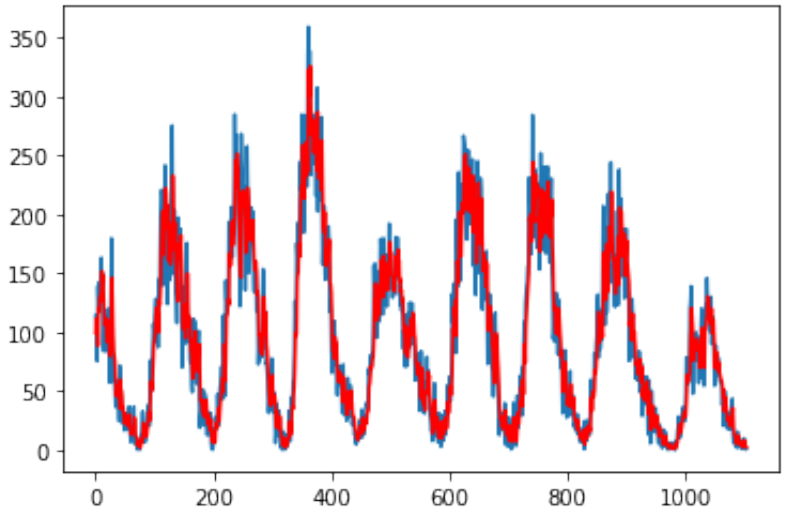
	0
count	2273.000000
mean	0.000016
std	0.063384
min	-0.302310
25%	-0.032105
50%	-0.002130
75%	0.031582
max	0.395665

In [29]:
```
from sklearn.metrics import mean_squared_error

X = dataframe.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
```

```
predicted=99.593578, expected=114.100000
predicted=111.438596, expected=105.200000
predicted=105.534552, expected=112.100000
predicted=107.246697, expected=75.300000
predicted=88.500018, expected=139.200000
predicted=119.917690, expected=122.400000
predicted=118.089734, expected=142.400000
predicted=128.630236, expected=134.300000
predicted=130.644754, expected=128.300000
predicted=125.985576, expected=152.400000
predicted=144.315743, expected=163.400000
predicted=151.895354, expected=139.600000
predicted=142.860535, expected=149.600000
predicted=147.696833, expected=102.300000
predicted=122.111836, expected=83.900000
predicted=106.352766, expected=98.300000
predicted=108.618970, expected=114.800000
predicted=112.110896, expected=104.700000
predicted=106.974992, expected=83.800000
predicted=92.200240, expected=87.900000
```

In [30]:
```
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)

plt.plot(test)
plt.plot(predictions, color='red')
plt.show()
```

```
Test MSE: 623.160
```



Задание 4. Повторите эксперимент по прогнозированию, реализовав рекуррентную нейронную сеть (с как минимум 2 рекуррентными слоями).

Задание 5. Сравните качество прогноза моделей. Какой максимальный результат удалось получить на контрольной выборке?

In [8]:
```
batch_size = 1
look_back = 1

model = Sequential()
model.add(LSTM(4, batch_input_shape=(batch_size, look_back, 1), st
model.add(LSTM(4, batch_input_shape=(batch_size, look_back, 1), st
model.add(Dense(1))


model.summary()

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=50, batch_size=1, verbose=2, vali
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras
/backend/tensorflow_backend.py:66: The name tf.get_default_graph is
deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras
/backend/tensorflow_backend.py:541: The name tf.placeholder is depre
cated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras
/backend/tensorflow_backend.py:4432: The name tf.random_uniform is d
eprecated. Please use tf.random.uniform instead.

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (1, 1, 4)                 96
_________________________________________________________________
lstm_2 (LSTM)                (1, 4)                    144
```

In [18]:
```
testPredict = model.predict(testX, batch_size=batch_size)
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0
print('Test Score: %.2f RMSE' % (testScore))
```

```
Test Score: 22.43 RMSE
```