Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по предмету:
«Машинное обучение»
По лабораторной работе №8
«Рекуррентные нейронные сети для анализа временных рядов»

Выполнил: Сенькович Дмитрий Сергеевич
магистрант кафедры информатики
группы №858642

Проверил: Стержанов Максим Валерьевич
доцент, кандидат технических наук

Минск 2020

Оглавление

1 Постановка задачи

Требуется построить модели ARIMA и рекуррентную нейронную сеть для предсказания среднемесячного количества точек на солнце на основе данных более 3000 месяцев.

2 Чтение данных. Построение основных характеристик временного ряда

Для начала следует взглянуть на данные:

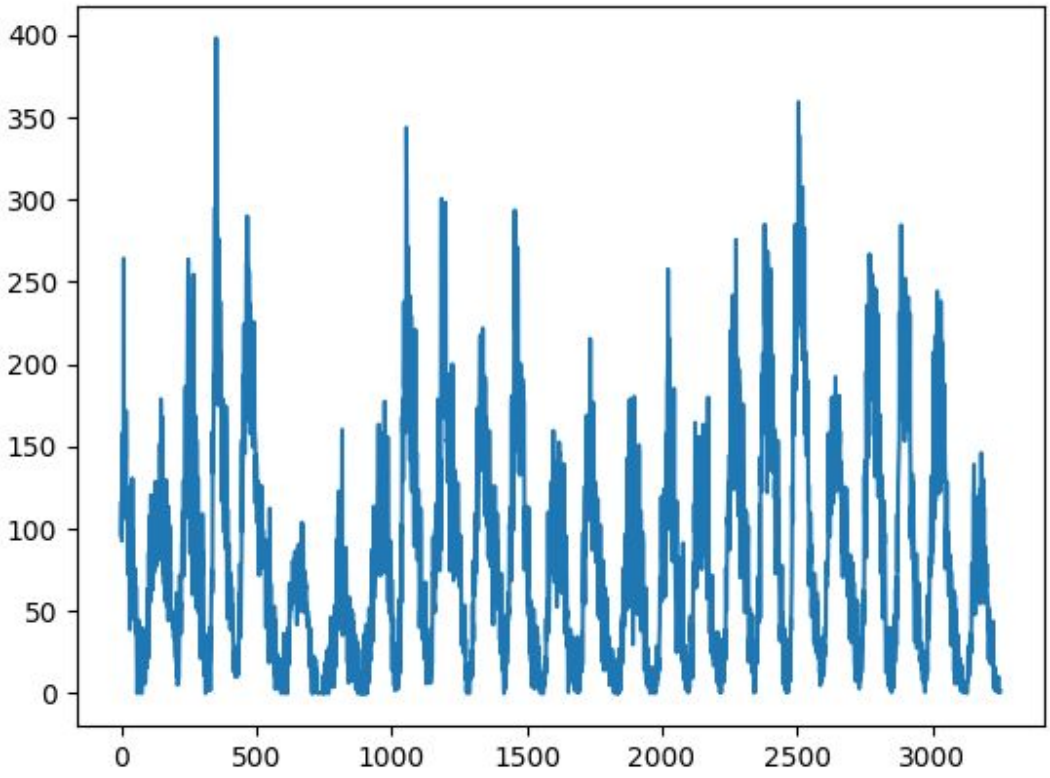

Рисунок 2.1. Исходные данные

Построим графики основных характеристик временного ряда: автокорреляции, частичной автокорреляции, сезонности и тренда. Приведем листинг кода для считывания данных и построения этих графиков:

```
def parser(x):

    return pd.to_datetime(x, format='%Y-%m-%d', errors="ignore")
```

```
if __name__ == "__main__":

    dataset = pd.read_csv(DATASET_FILE_NAME, parse_dates=[0], header=0,
index_col=0, squeeze=True, usecols=[1, 2], date_parser=parser)

    decomposition = seasonal_decompose(dataset, model='additive')

    #decomposition.plot()


    # data

    # pyplot.plot(list(range(0, len(dataset))), dataset)


    # trend

    pyplot.title('Trend')

    pyplot.plot(decomposition.trend)


    # seasonality

    pyplot.title('Seasonality')

    pyplot.plot(decomposition.seasonal)


    # autocorrelation

    autocorrelation_plot(dataset)


    # partial autocorrelation

    tsaplots.plot_pacf(dataset)


    pyplot.show()
```

Приведем соответствующие им графики характеристик ряда:

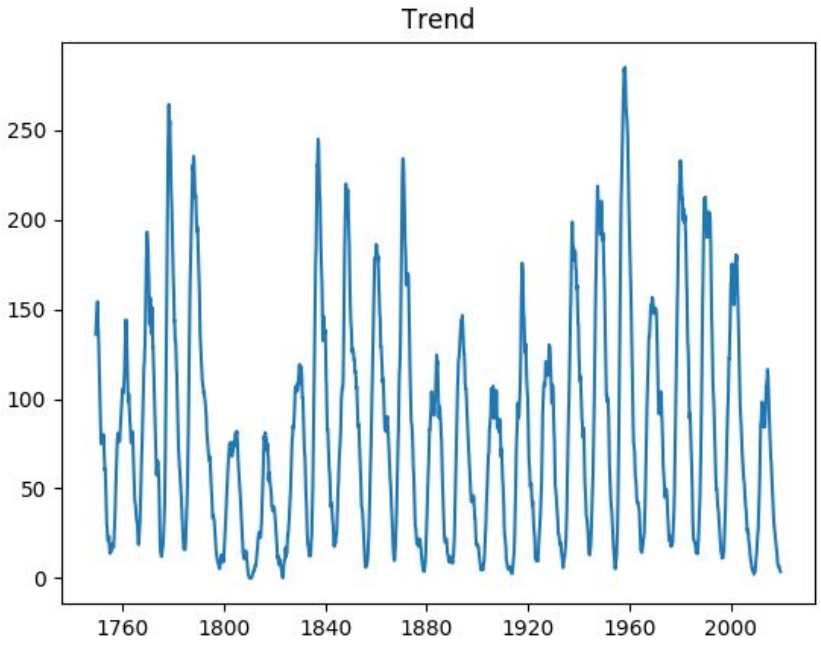

Рисунок 2.2. Линия тренда временного ряда

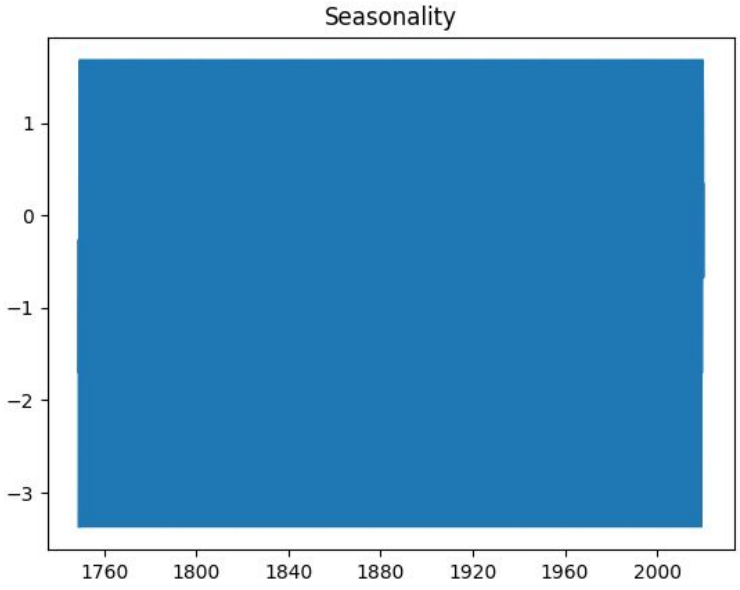

Рисунок 2.3. Сезонность временного ряда

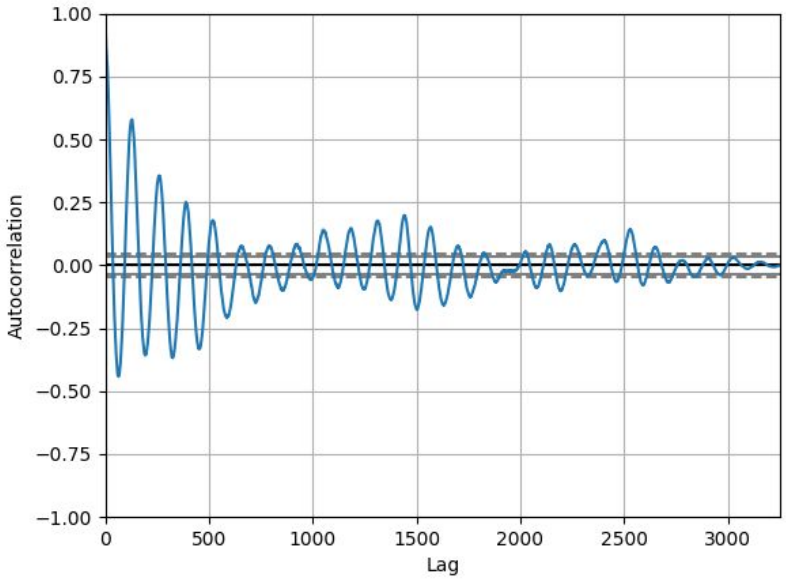

Рисунок 2.4. Автокорреляция

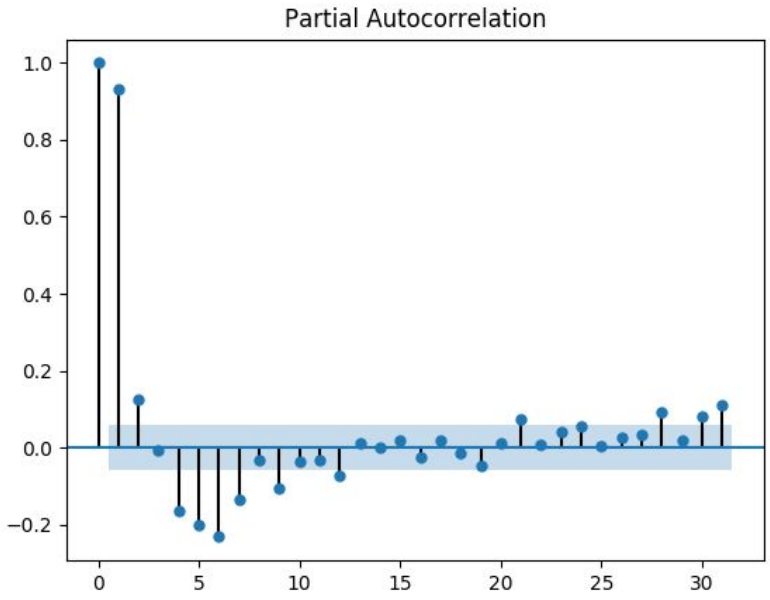

Рисунок 2.5. Частичная автокорреляция

3 Построение модели ARIMA

Найдем параметры (p, d, q) ARIMA из графиков выше. Для начала для наглядности приведем формулу модели:

$$\triangle^d X_t = c + \sum_{i=1}^{p} a_i \triangle^d X_{t-i} + \sum_{j=1}^{q} b_j \varepsilon_{t-j} + \varepsilon_t$$

Здесь p - количество слагаемых в AR части модели, q - в MA части, а d - взятие d раз разностей первого порядка.

Параметр d используется, если временной ряд не стационарен, т.е. не имеет постоянных мат ожидания и дисперсии, на что может указывать нелинейный тренд, как в нашем случае. Обычно d равен 0, 1 или 2. Возьмем 1.

Параметр p определяется по автокорреляции: это пересечение линии графика с верхней границей (на графике прерывистая линия). Покажем увеличенную версию графика для определения параметра:

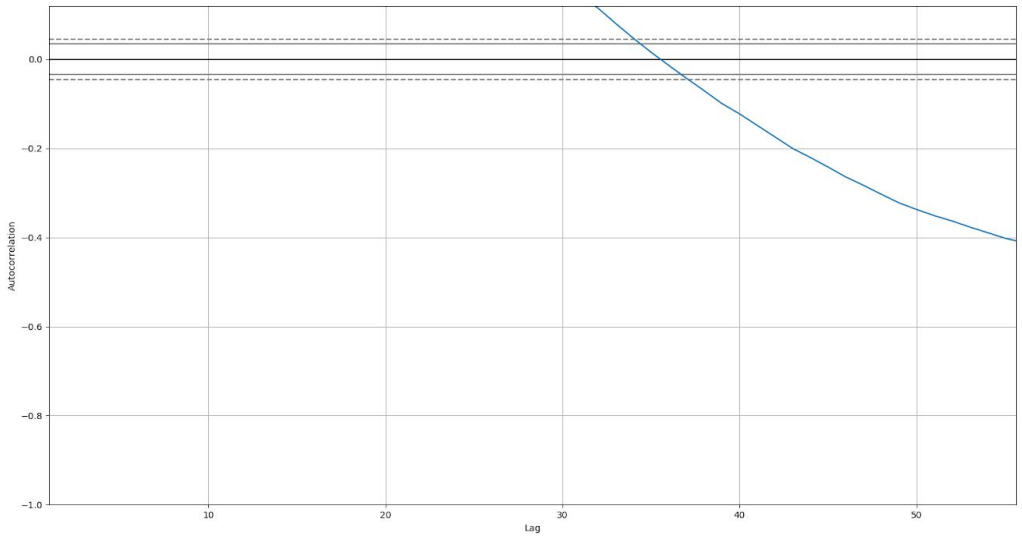

Рисунок 3.1. Увеличенный график автокорреляции

График пересекается с границей в значении примерно 34.

Далее, определим значение q. Значение определяется по графику частичной автокорреляции, а именно по количеству значений вне диапазона (см. рис. 2.5 из предыдущего пункта). Таких значений 29.

Построим модель ARIMA по этим параметрам, заодно предсказывая значения валидационной выборки:

```
    test_split = int(dataset_size / 8)

    train_dataset, validation_dataset = dataset[:-test_split],
dataset[-test_split:]

    print(train_dataset.shape)

    print(validation_dataset.shape)

    history = [x for x in train_dataset]

    predictions = []

    tt = TicToc('learning')

    tt.tic()

    for t in range(len(validation_dataset)):

        # fit model

        model = ARIMA(history, order=(34,1,21))

        model_fit = model.fit(disp=False)

        # one step forecast

        yhat = model_fit.forecast()[0]

        # store forecast and ob

        predictions.append(yhat)

        history.append(validation_dataset[t])

        print(t)

    tt.toc()

    print("Elapsed: %f" % (tt.elapsed / 60))
```

```
# evaluate forecasts

error = mean_squared_error(validation_dataset, predictions)

print('Test MSE: %s' % error)

pyplot.plot(validation_dataset)

pyplot.plot(predictions, color='red')

pyplot.show()
```

Данная модель уже на первом экземпляре работает больше 30 минут (было решено не дожидаться окончания обработки ввиду нецелесообразности использования модели).

Поэтому решено воспользоваться автоматическим подбором параметров ARIMA модели:

```
rs_fit = pm.auto_arima(dataset,

                       n_jobs=-1,

                       error_action='ignore',

                       suppress_warnings=True,

                       stepwise=False, random=True,

                       n_fits=1000)


print(rs_fit.summary())
```

Поиск нашел модель с параметрами (3, 0, 2). Заменим нашу модель на эту и получим следующие результаты:

Время обработки: 17.79 минут

MSE: 469.753.

Приведем график реальных значений и предсказания модели:

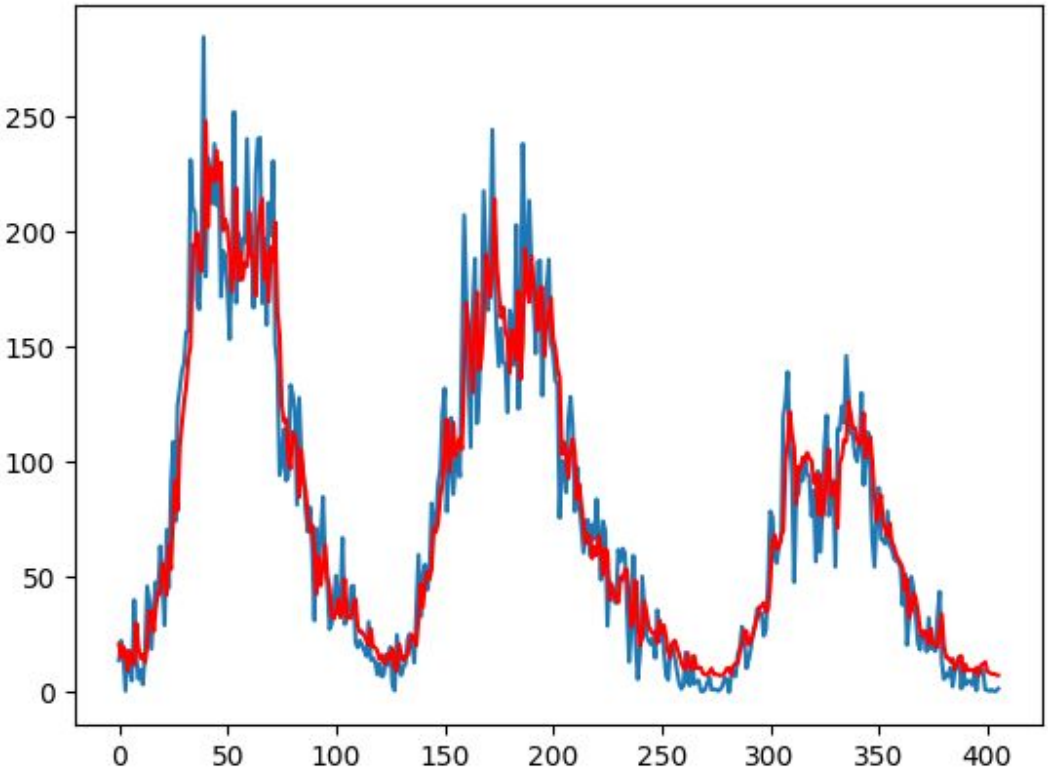

Рисунок 3.2. Предсказания модели ARIMA

4 Построение модели RNN

Покажем, как будем формировать выборку для обучения: весь датасет будет трансформирован в последовательности значений из оригинального ряда, а в качестве значения для обучения будет следующее значение в этом ряду за последовательностью, например, первый экземпляр из выборки будет, скажем, 500 первых элементов оригинального временного ряда, а в качестве значения будет 501-ый:

```
def build_dataset(sequence, steps):

    dataset, labels = [], []

    for i in range(len(sequence)):

        end = i + steps

        if end > len(sequence)-1:

            break

        seq_x, seq_y = sequence[i:end], sequence[end]

        dataset.append(seq_x)

        labels.append(seq_y)

    return np.array(dataset).astype('float32'),
np.array(labels).astype('float32')
```

Сначала покажем лучшую получившуюся (с точки зрения значения mse) модель:

```
def map_predictions(predictions):

    mapped_predictions = []

    for prediction in predictions:

        mapped_predictions.append(prediction[0][0])

    return np.array(mapped_predictions).astype('float32')
```

```
if __name__ == "__main__":

    spots = read_spots()

    #np.random.shuffle(spots)

    dataset, labels = build_dataset(spots, WINDOW_LENGTH)

    print(dataset.shape)

    print(labels.shape)

    train_dataset, validation_dataset, train_labels, validation_labels =
train_test_split(dataset,


labels,


test_size=1 / 8, shuffle=True)

    print(train_dataset.shape)

    print(train_labels.shape)

    print(validation_dataset.shape)

    print(validation_labels.shape)


    t = TicToc('learning')

    t.tic()

    model = Sequential([

        Conv1D(128, 5, strides=1, padding="same", activation="relu"),

        Bidirectional(LSTM(128, return_sequences=True, dropout=0.2)),

        Bidirectional(LSTM(128, return_sequences=True, dropout=0.2)),

        Dense(256, activation='relu'),

        Dropout(0.2),

        Dense(128, activation='relu'),

        Dropout(0.2),

        Dense(1),
```

```
    ])

    model.compile(optimizer='adam', loss='mse')

    model.fit(train_dataset, train_labels, epochs=EPOCHS_NUMBER,
batch_size=BATCH_SIZE)

    t.toc()

    print("Elapsed: %f" % (t.elapsed / 60))

    predictions = model.predict(validation_dataset, verbose=0)

    error = mean_squared_error(validation_labels, map_predictions(predictions))

    print("Test MSE: %s" % error)

    pyplot.plot(validation_labels)

    pyplot.plot(map_predictions(predictions), color='red')

    pyplot.show()
```

WINDOW_SIZE - длина каждого экземпляра выборки. Большинство экспериментов были проведены с рандомизацией выборок и нерелевантны, поэтому покажем только часть, аналогичную с ARIMA моделью:

Window size	Кол-во эпох	Время	MSE	Кол-во Conv1d	Кол-во нейронов Conv1d	Кол-во LSTM слоев	Кол-во нейронов LSTM	Dropout LSTM	Кол-во Dense слоев	Кол-во нейронов Dense	Dropout Dense
500	100	2.95	1177.587	1	64	2	128	-	2	128, 128	0.1
132	100	1.00	1082.296	1	64	2	128	-	2	32, 16	-
64	100	0.58	695.4209	1	64	2	128	-	2	32, 16	-
48	100	0.50	561.	1	64	2	128	-	2	32,	-

			039							16	
32	100	0.37	726.0093	1	64	2	128	-	2	32, 16	-
48	100	0.98	1065.191	1	256	2	256	-	2	256, 128	-
48	100	1.08	586.3977	1	256	2	256	0.2	2	256, 128	0.1
34	200	0.95	486.0885	1	128	2	128	0.2	2	256x2	0.2

Покажем получившийся график предсказания значений временного ряда на валидационной выборки для получившейся RNN:

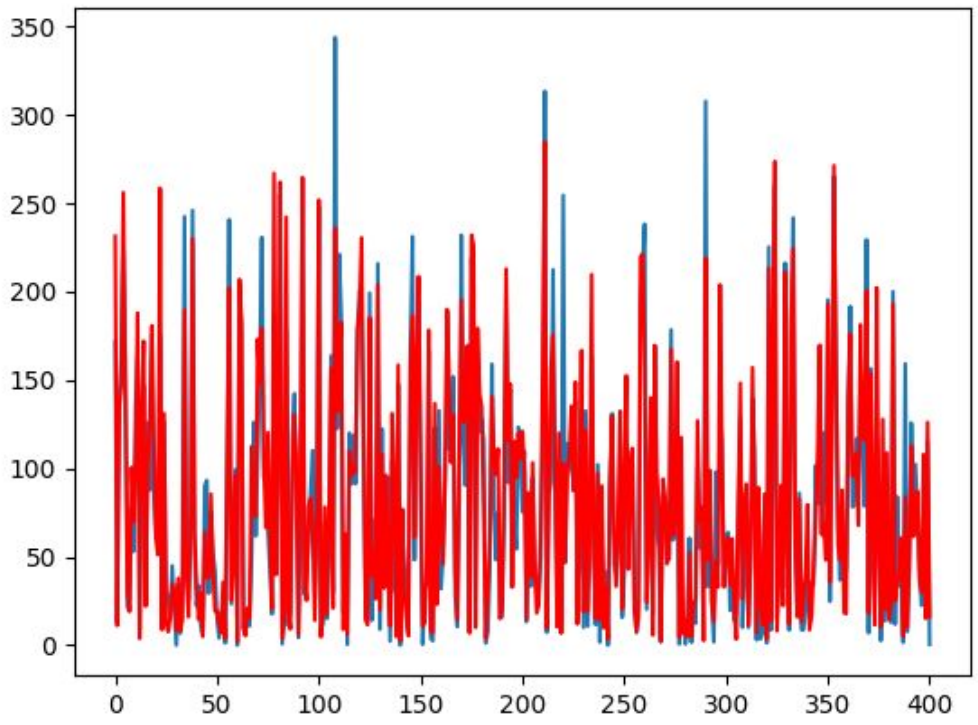

Рисунок 4.1. Предсказания RNN модели