Лабораторная работа 2. Временные ряды

2.1 Понятие временного ряда

Временной ряд – это последовательность значений, описывающих протекающий во времени процесс, измеренных в последовательные моменты времени, обычно через равные промежутки. В данных задачах по крайней мере одним из входных параметров всегда рассматривается временная метка, а цель состоит в определении состояния системы в последующие временные отсчеты, то есть построенная система должна уметь делать «предсказания».

В данном контексте появляется неоднозначность относительно разделения на обучающую и тестовую выборки. Классическим подходом является отнесение последних по времени отсчетов к тестовой выборке.

Для решения подобных задач используются различные подходы. В частности, можно проводить дополнительную обработку датасета и выделять отдельные признаки для данных, такие как медианное значение количества запросов по дням недели, месяцам или еще каким-то градациям. В данном случае возможно использовать рассмотренные ранее подходы к построению систем принятия решений. Однако, в данной лабораторной работе рассматриваются подходы, основанные на потоковой обработке данных в исходном виде, то есть в зависимости от временных отсчетов.

При работе с временными рядами всегда целесообразно использовать графическое отображение для лучшего понимания данных и тенденций в них.

```
import matplotlib.pyplot as plt

dataset = pd.read_csv('...')
row = train.iloc[1000,:].values #пример выбора значений 1000-ой строки
plt.plot(row)
plt.show()
```

Интегрированные модели, модели авторегрессии и модели скользящего среднего.

Модели авторегресии (autoregression) порядка p AR(p), интегрированные (integrated) модели порядка d I(d) и модели скользящего среднего (moving average) порядка q MA(q) – классические математические модели, имитирующие поведение процессов, наблюдаемых в реальном мире, с течением времени.

Для работы с этими моделями может использоваться класс `ARIMA` из модуля `statsmodels.tsa.arima.model` пакета `statsmodels` (установить который можно при помощи команды `pip install statsmodels`):

```
from statsmodels.tsa.arima.model import ARIMA
model = ARIMA(dataset, order=(p, d, q), seasonal_order=(P, D, Q, s))
fitted = model.fit()
```

Класс ARIMA является обобщенной моделью, включающей в себя модель авторегрессии порядка p, интегрированную модель порядка i, модель скользящего среднего порядка q, а также модель более высокого уровня с сезонной авторегрессией порядка P, сезонной интегрированной моделью порядка I, сезонной моделью скользящего среднего порядка Q, с сезонной периодичностью s. Порядок основной модели задаётся при помощи аргументов `order` в виде кортежа из 3 чисел и `seasonal_order` в виде кортежа из 4 чисел, соответствующих модели SARIMA порядка (p, d, q)x(P, D, Q, s). Таким образом, например, модель порядка (1, 0, 0) является «чистой» авторегрессионной моделью первого порядка, модель (0, 1, 0) задаёт случайное блуждание (т.е. нестационарную интегрированную модель первого порядка), модель (0, 0, 1) является моделью скользящего среднего первого порядка, модель (1, 1, 1)x(1, 1, 1, 10) является комбинированной моделью первого порядка с сезонной составляющей в виде комбинированной модели 1 порядка и сезонным сдвигом в 10 отсчётов. Подбор порядка модели должен осуществляться на основе анализа статистических характеристик временного ряда относительнонекоторого сдвига, как правило, при помощи функции автокорреляции и анализа дифференциальных рядов.

Функция автокорреляции (ACF, autocorrelation function) $C(i)$ определяется как корреляция временного ряда с самим собой, сдвинутым на i отсчётов в прошлое. По определению $C(0) = 1$, а остальные значения заключены в диапазон $[-1; 1]$. Для вывода автокорреляционной функции по заданному временному ряду можно использовать функцию `plot_acf` модуля `statsmodels.graphics.tsaplots` пакета `statsmodels` (рисунок 1).

```
from statsmodels.graphics.tsaplots import plot_acf

fig, axes = plt.subplots(1, 2)

axes[0].plot(dataset)
plot_acf(dataset, ax=axes[1]);
```

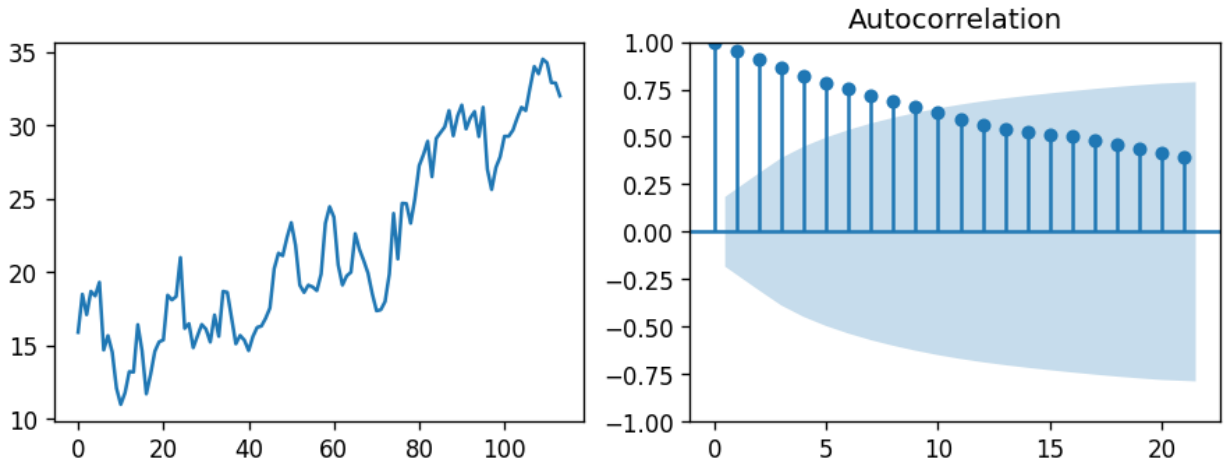

Рисунок 1. Временной ряд и соответствующая ему автокорреляционная функция

Помимо автокорреляционной функции, также для анализа временных рядов используется функция частичной автокорреляции (PACF, partial autocorrelation function). Основным недостатком обычной автокорреляционной функции является тот факт, что автокорреляция порядка i характеризует корреляцию не только между текущим и i-м прошлым отсчётом, но и между всеми другими кратными отсчётами в этом диапазоне. Из-за этого, например, значительная корреляция с запаздыванием на 2 отсчёта может с большой вероятностью повлиять на корреляцию с запаздыванием на 4, 6 и всех последующих кратных сдвигах. Чтобы устранить «фантомные» корреляции в пределах диапазона, функция частичной автокорреляции дополнительно применяет ко всем отсчётам между текущим и прошлым i-м сглаживающую фильтрацию, что позволяет подчеркнуть автокорреляцию сдвига именно на i отсчётов. Также при анализе временных рядов рекомендуется рассматривать корреляционную и автокорреляционную функцию вместе, т.к. различия между ними могут указывать на скрытые зависимости.

Для вывода частичной автокорреляционной функции по заданному временному ряду можно использовать функцию `plot_pacf` модуля `statsmodels.graphics.tsaplots` пакета `statsmodels` (рисунок 2).

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plt.rcParams.update({'figure.figsize':(10,10), 'figure.dpi':90})
fig, axes = plt.subplots(3, 1)
axes[0].plot(dataset)
plot_acf(dataset, ax=axes[1]);
plot_pacf(dataset, ax=axes[2]);
```

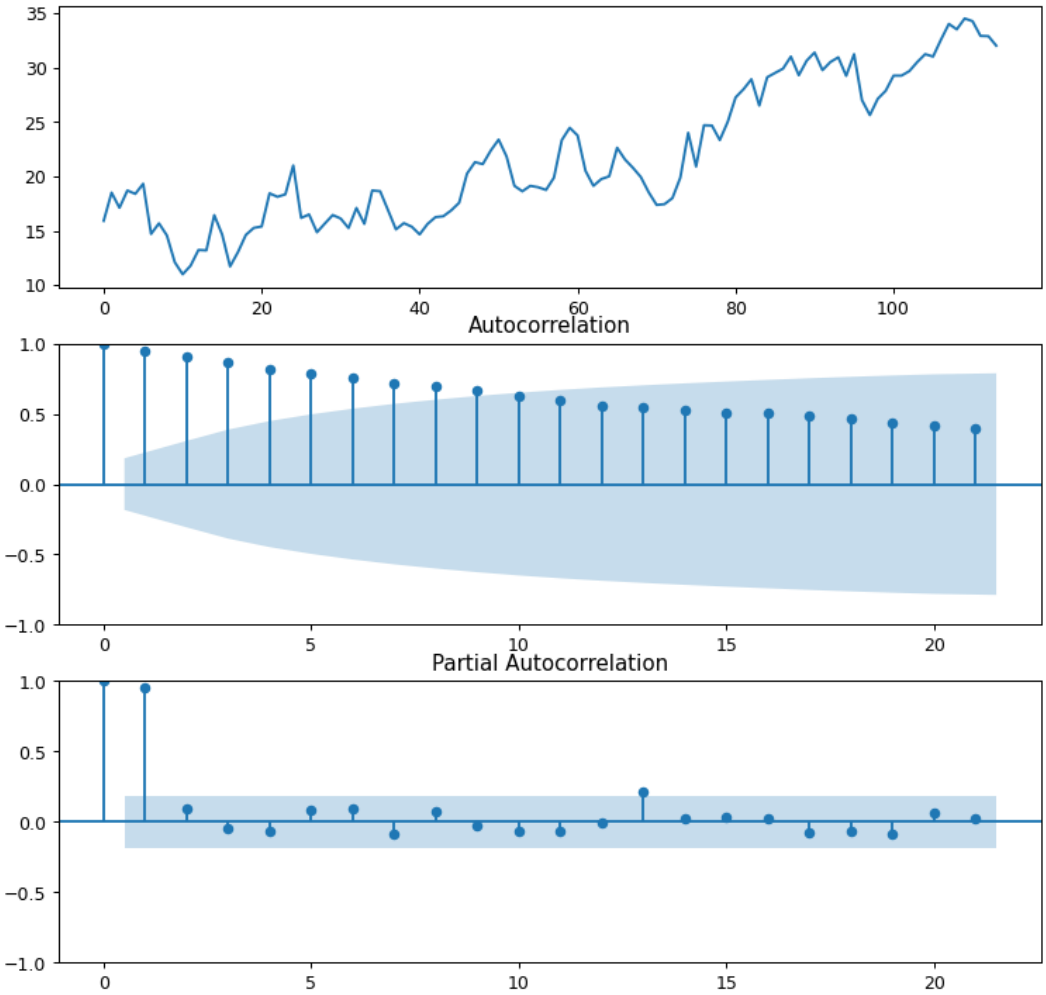

Рисунок 2. Временной ряд, автокорреляционная функция и частичная автокорреляционная функция

Можно выделить следующие этапы идентификации модели:

1. Определить стационарности или нестационарности по результатам визуализации временного ряда;
2. Построить автокорреляционную и частичную автокорреляционную функции, определить стационарности или нестационарности;
3. Если ряд демонстрирует нестационарное поведение, увеличить порядок интегрирования на 1 и построить автокорреляционную и частичную автокорреляционную функцию частичных разностей первого порядка. Повторять этот шаг до тех пор, пока ряд из частичных разностей не станет стационарным;
4. Оценить параметры по полученным графикам автокорреляционной и частичной автокорреляционной функции;
5. Построить модель и оценить её поведение на временном ряду.

6. Если модель плохо описывает ряд, но в нём могут наблюдаться сезонные колебания, оценить целесообразность использования модели SARIMA.

Для анализа стационарности может использоваться как график самого временного ряда, так и совместный анализ автокорреляционной и частичной автокорреляционной функции. Для рассматриваемого случая можно заметить, что временной ряд демонстрирует характерно нестационарное поведение.

Из корреляционной функции можно заметить, что во временном ряду наблюдается значительная корреляция с самим собой на протяжении большого количества сдвигов. Это может указывать на то, что корреляция отсчёта с отставанием на 1 вводит «фантомные» автокорреляции на всех последующих отсчётах. Эту гипотезу подтверждает анализ функции частичной автокорреляции – на ней статистически значимая корреляция наблюдается только для отставания на 1 отсчёт.

Функция частичной автокорреляции может использоваться для того, чтобы определить порядок авторегрессионной модели. По сути, значение в i-м отсчёте функции частичной автокорреляции соответствует значению коэффициента авторегрессии i-го порядка. В рассматриваемом случае можно заметить, что для исследуемой функции наблюдается близкая к 1 корреляция со сдвигом 1, но не на других сдвигах. Т.к. значение этой корреляции близко к 1, это свдительествует о нестационарности ряда.

Основная сложность авторегрессионных моделей – их способность формировать нестационарные ряды в случаях, когда коэффициенты авторегрессии близки к 1. Для рассматриваемого на рисунке 2 случая можно заметить, что имеет место именно такая зависимость – т.е. при подборе параметров авторегрессии первого порядка с большой вероятностью будет получен коэффициент авторегрессии, близкий к 1, и в результате моделирования такого временного ряда будет получен ряд, близкий к случайному блужданию, который, скорее всего, крайне плохо будет описывать исследуемую функцию из-за его нестабильности.

Рассмотрим частичные разности первого порядка (рисунок 3). Для их вычисления можно использовать метод `diff()` класса `pandas.DataFrame`. Следует отметить, что полученные разности будут иметь ту же размерность, однако самое первое значение будет равнятся `NaN`, поэтому для дальнейшего анализа можно отбросить первый отсчёт в полученных разностях:

```
dataset_diff = dataset.diff().iloc[1:]
```

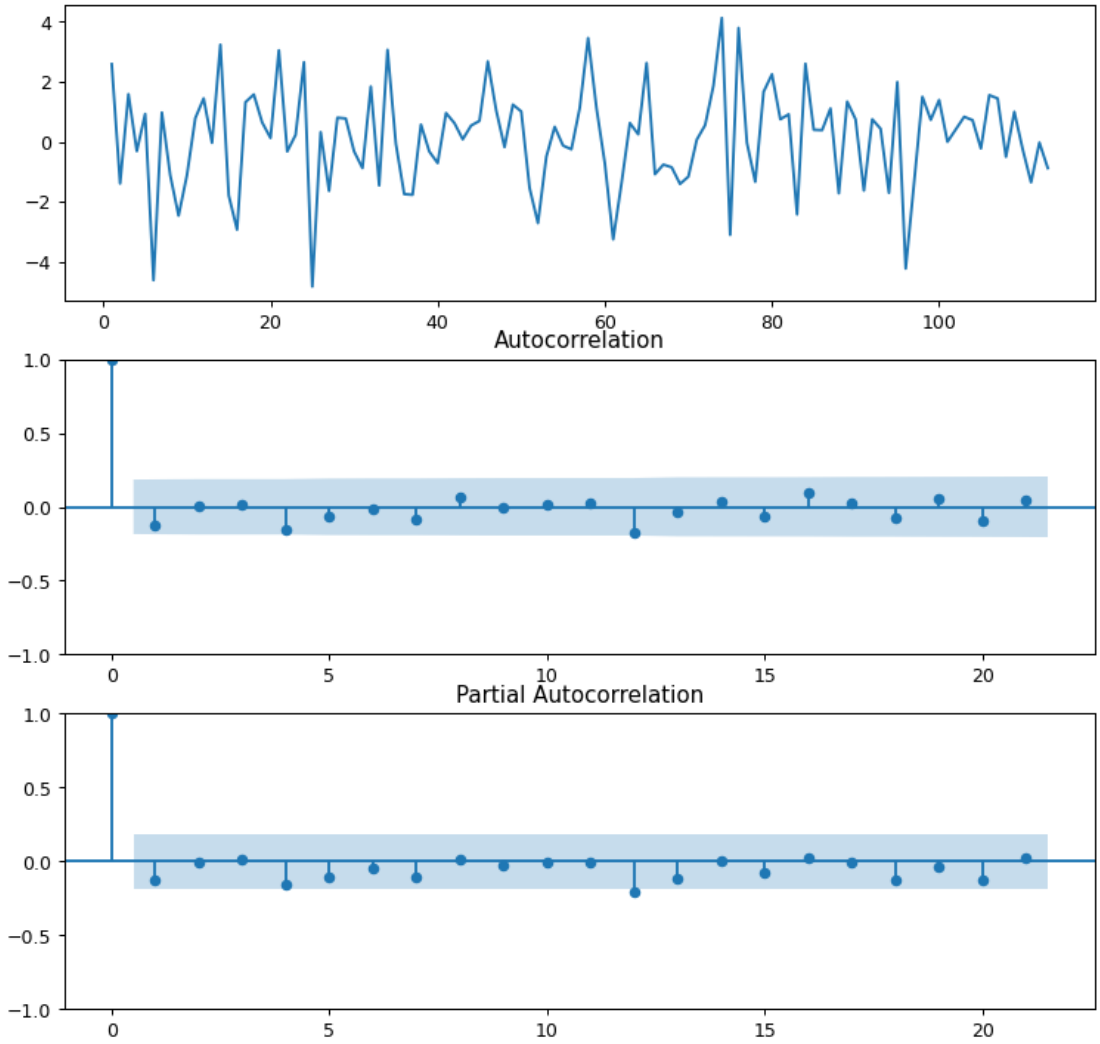

Рисунок 3. Временной ряд, автокорреляционная функция и частичная автокорреляционная функция

Полученный ряд из частичных разностей демонстрирует стационарное поведение, поэтому этап определения степени нестационарности можно остановить, и зафиксировать значение параметра порядка интегрирования d итоговой модели равным 1. Таким образом, **если в функции частичной автокорреляции наблюдается коэффициент для первых k сдвигов, близкий по модулю к 1, ряд является нестационарным – необходимо повысить порядок интегрирования d и проанализировать ряд частичных разностей первого порядка.**

С другой стороны, если коэффициенты для первых k сдвигов характеризуются значительными пиками, но не близки к 1, это может свидетельствовать об авторегрессионном характере зависимости. Таким образом**, если в функции частичной автокорреляции наблюдаются**

положительные коэффициенты для первых k сдвигов (и резкий «обвал» на $(k+1)$-м сдвиге), удаленные по значению и от 0, и от 1, временной ряд может быть описан авторегрессионной моделью порядка k (AR(k)).

Наконец, порядок модели скользящего среднего позволяет регулировать то, насколько модель «сглаживается» от отсчёта к отсчёту и соответствует отрицательной автокорреляции. Таким образом**, если в функции частичной автокорреляции наблюдаются отрицательные коэффициенты для первого сдвига, значительные по модулю коэффициенты первых k сдвигов и резкий «обвал» на $(k+1)$-м сдвиге, временной ряд может быть описан моделью скользящего среднего порядка k (MA(k)).**

Наконец, **если в моделях наблюдается периодичность, частота которой распространяется на большое количество отсчётов (как правило, более 10), вместо использования модели высокого порядка (например, ARIMA(10,1,10)) рекомендуется использовать сезонную модель SARIMA с показателем сезонности, соответствующем периоду.**

Ниже приведен пример получения результата с помощью обученной модели (переменная `fitted`) на количество временных отсчетов, соответствующее тестовой выборке, в также вывод на графике полученного и ожидаемого результатов:

```
fc = fitted.get_forecast(len(row[training_time_steps:]))
conf = fc.conf_int(alpha=0.05) # 95% confidence

fc_series = pd.Series(fc.predicted_mean, index=row[training_time_steps:].index)
lower_series = pd.Series(conf.iloc[:,0],index=row[training_time_steps:].index)
upper_series = pd.Series(conf.iloc[:, 1], index=row[training_time_steps:].index)

plt.figure(figsize=(12,5), dpi=200)
plt.plot(row[:training_time_steps], label='training')
plt.plot(row[training_time_steps:], label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index, lower_series, upper_series, color='k',
alpha=.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

2.2 Сети долгой краткосрочной памяти (LSTM - long short-term memory)

Одним из наиболее популярных нейросетевых подходов к обработке временных рядов являются сети долгой краткосрочной памяти (LSTM). В отличие от обычного нейрона прямого распространения, LSTM-нейрон

представляет собой ячейку, содержащую внутреннее состояние, т.е. способную формировать активацию на выходе не только в зависимости от поданного на вход значения, но и от поданных на вход значений на предыдущих этапах работы системы.

Библиотека Keras предоставляет широкие возможности для работы с сетями долгой краткосрочной памяти. Keras предоставляет унифицированный интерфейс над несколькими «бэкэндами» (backend) – реализациями работы с вычислениями в описательном виде, позволяющие запускать их как на центральном процессоре, так и на видеокартах. Самый популярный бэкэнд, который используется в Keras по умолчанию – TensorFlow, который необходимо установить для корректной работы Keras (`pip install tensorflow`). Саму библиотеку Keras устанавливать не нужно, так как после версии Tensorflow 2.0 она была интегрирована. После этого необходимо проимпортировать необходимые в дальнейшем функции, как показано в примерах далее.

Создание модели осуществляется с помощью класса Sequential, который позволяет задать произвольную конфигурацию последовательных слоев нейронной сети любой сложности. Подробнее особенности работы с моделями на основе данной архитектуры рассматривались в предыдущей лабораторной работе. Пример создания модели:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers, losses, metrics, optimizers


regressor = Sequential([
    layers.LSTM(units = 32, activation = 'relu', input_shape = (1, sequence_length)),
    layers.Dense(1)
])
```

Особенность использования LSTM для прогнозирования временных рядов состоит в необходимости специальной подготовки обучающей выборки. Как было показано в предыдущей работе, отдельным входным элементом для LSTM слоя должен выступать не единичный вектор, а некоторая их последовательность, по состоянию которой может быть оценен выходной результат. При анализе временных рядов целесообразно для обучающей выборки формировать короткие подпоследовательности из исходного временного ряда и ожидать, что по таким подпоследовательностям модель сможет предсказать следующее значение. Другими словами, для обучающей выборки из n элементов можно выбрать некоторую ширину окна t и сформировать (n – t) подпоследовательностей – от 1 до t-1, от 2 до t, от 3 до t+1

и т.д. до подпоследовательности n-t до n. Ожидаемым выходным значением для них будут соответственно значения элементов t, t+1, t+2 и т.д. Код соответствующей процедуры формирования входных последовательностей и ожидаемых значений представлен ниже.

```
sequence_length = 30
X = []
y = []
time_steps = len(row)
for i in range(time_steps - sequence_length):
    sequence = row[i:(i+sequence_length)]
    X.append(sequence)
    y.append(row[i+sequence_length])

X = np.array(X)
y = np.array(y)
```

После формирования данных выборки в нужном виде следует разделить их на обучающую и тестовую. При анализе временных рядов данные не перемешиваются, а подаются последовательно. Первая часть выборки становится обучающей, а вторая тестовой. Например, выделить 30% данных для тестовой выборки можно следующим образом:

```
import math

time_steps = len(X)
training_time_steps = math.floor(time_steps * 0.7)
X_train = X[:training_time_steps]
X_test = X[training_time_steps:]
y_train = y[:training_time_steps]
y_test = y[training_time_steps:]
```

На вход LSTM-слой ожидает данные в виде трехмерной матрицы. Пример приведения данных к нужному виду:

```
X_train_sequences = X_train.reshape(X_train.shape[0], 1, X_train.shape[1])
X_test_sequences = X_test.reshape(X_test.shape[0], 1, X_test.shape[1])
```

Для получения описания созданной модели можно использовать функцию summary(). Функция compile() преобразует описание модели в набор инструкций для «бэкэнда» и готовит тем самым модель к использованию. Для обучения сети используется функция fit.

```
from livelossplot import PlotLossesKerasTF

regressor.compile(loss='mean_squared_error',          optimizer='adam',
metrics=['mae'])
history  =  regressor.fit(X_train_sequences,  y_train,  batch_size=10,
epochs=300, verbose=0, callbacks=[PlotLossesKerasTF()])
```

Анализ результатов в данной лабораторной работе предлагается провести визуально, отобразив ожидаемый и предсказанный временные ряды на одном графике. Пример такого отображения:

```
plt.figure(figsize=(12,5), dpi=200)
plt.plot(range(training_sequences),           row[:training_sequences],
label='training')
plt.plot(range(training_sequences, len(row)), row[training_sequences:],
label='test')
plt.plot(range(sequence_length,         len(row)),         y_predicted,
label='forecast')
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

Следует также отметить, что архитектуры на основе LSTM для анализа временных рядов крайне чувствительны к выбросам, поэтому в качестве дополнительного этапа предварительной обработки рекомендуется использовать нормализацию, например, MinMaxScaler из модуля sklearn.preprocessing.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
series_matrix = row.reshape(len(row), 1)
scaler.fit(series_matrix)
normalized_matrix = scaler.transform(series_matrix)
```

Для преобразования данных к исходному виду можно использовать функцию `inverse.transform`. Пример применения этой функции к данным перед их отображением на графике:

```
plt.plot(scaler.inverse_transform(y.reshape(len(y), 1)))
```

Задача 1. Охота за экзопланетами в глубоком космосе

Данные описывают изменение потока (интенсивности света) нескольких тысяч звезд. Представленные данные очищены и получены из наблюдений, сделанных космическим телескопом NASA Kepler. Если наблюдать за звездой в течение нескольких месяцев или лет, может наблюдаться закономерное «затухание» потока (интенсивности света). Это свидетельствует о том, что вокруг звезды может вращаться тело.

№ варианта	Используемые данные (название колонки)
1	FLUX.1
2	FLUX.2
3	FLUX.3
4	FLUX.4
5	FLUX.5
6	FLUX.6
7	FLUX.7
8	FLUX.8
9	FLUX.9

Источник данных:
https://www.kaggle.com/datasets/keplersmachines/kepler-labelled-time-series-data

Задача 2. Анализ климатических условий

В датасете представлены наблюдения за изменением температуры, влажности, скорости ветра и атмосферного давления в Индии с 1 января 2013 по 24 апреля 2017 года.

№ варианта	Используемые данные (название колонки)
1	meantemp
2	humidity
3	wind_speed
4	meanpressure
5	meantemp
6	humidity

Источник данных:

https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data

Задача 3. Планирование продаж

В датасете представлены данные о продажах 5 разных продуктов и температура воздуха в эти дни. Периодичность наблюдений для разных продуктов не совпадает.

№ варианта	Используемые данные (название колонки)
1	ProductP1
2	ProductP2
3	ProductP3
4	ProductP4
5	ProductP6
6	temperature

Источник данных:
https://www.kaggle.com/datasets/soumyadiptadas/products-sales-timeseries-data

Задача 4. Посещение веб-сайта

В датасете представлены данные о посещении одного из веб-сайтов с 14 сентября 2014 по 19 августа 2020. В качестве наблюдаемых признаков предлагается количество загрузок страницы, количество уникальных посещений, количество первых посещений и количество вернувшихся

№ варианта	Используемые данные (название колонки)
1	Page.Loads
2	Unique.Visits
3	First.Time.Visits
4	Returning.Visits
5	Page.Loads
6	Unique.Visits

Источник данных:
https://www.kaggle.com/datasets/bobnau/daily-website-visitors

Задача 5. Стоимость акций

В датасете представлены данные фондового рынка, а именно цена акций при открытии, максимальная цена за день, минимальная цена за день и количество проданных акций. Для анализа используйте файл AABA_2006-01-01_to_2018-01-01.csv.

№ варианта	Используемые данные (название колонки)
1	Open
2	High
3	Low
4	Close
5	Volume
6	Open

Источник данных:
https://www.kaggle.com/datasets/szrlee/stock-time-series-20050101-to-20171231

Задание к лабораторной 2.

1. Получите задание в соответствии с вашим вариантом (номер задачи зависит от номера группы, а исследуемые данные от варинта).
2. Проанализируйте датасет. Опишите его характеристики: имеющиеся колонки, размер и т.д.
3. Проведите анализа временного ряда (доставшегося вам по варианту), подберите параметры и решите задачу прогнозирования с помощью моделей ARIMA/SARIMA. Обоснуйте полученные результаты.
4. Решите задачу прогнозирования временного ряда (по варианту) с помощью модели LSTM. Подберите параметры модели, обоснуйте полученные результаты.
5. . Решите задачу прогнозирования с помощью модели LSTM для всего набора данных (по номеру группы). Подберите параметры модели, обоснуйте полученные результаты.
6. Попробуйте улучшить работу методов с помощью изменения различных параметров, дополнительной обработки датасета и т.д.
7. Представьте ваше исследование в виде Блокнота JupyterLab, содержащего все пункты задания.