|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

***\_\_*** ***Сравнение моделей авторегрессии и LSTM для \_\_***

***\_\_\_\_\_ предсказания курса доллора \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент \_\_ИУ5-31М\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_М.С. Торжков\_\_\_\_\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_****Ю.Е. Гапанюк\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2023 г.***Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_ИУ5\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_В.И. Терехов\_\_

(И.О.Фамилия)

«\_04\_» \_\_\_\_сентября\_\_\_\_\_\_\_ 2023 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме \_\_ Сравнение моделей авторегрессии и LSTM для предсказания курса доллора \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_ИУ5-31М\_\_

\_\_\_\_\_\_\_\_\_\_\_\_Торжков Максим Сергеевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИССЛЕДОВАТЕЛЬСКАЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_КАФЕДРА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_\_\_\_ нед., 50% к \_\_\_\_ нед., 75% к \_\_\_ нед., 100% к \_\_\_\_ нед.

Техническое задание

\_\_Разработать и сравнить модели ARIMA и LSTM для предсказания значения курса доллора\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на 19 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_04\_» \_\_\_\_сентября\_\_\_\_\_\_\_ 2023 г.

**Руководитель НИР**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Ю.Е. Гапанюк\_\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_М.С. Торжков\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc153722606)

[Основная часть 3](#_Toc153722607)

[ЗАКЛЮЧЕНИЕ 16](#_Toc153722623)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc153722624)

## ВВЕДЕНИЕ

С момента зарождения валютного рынка для прогнозирования значений были разработаны различные методы прогнозирования фондового рынка. Некоторые из этих методов используются для определения будущих движений и включают фундаментальный анализ, технический анализ и смешанный анализ, например, использование статистических методов для моделирования поведения цен и составления прогнозов на будущее. Отдельные модели, такие как ARIMA, использовались для прогнозирования временных рядов из-за их популярности в качестве классических методов прогнозирования. В последнее десятилетие RNN типа LSTM широко использовались для прогнозирования последовательных данных. Механизм, с помощью которого такие сети хранят долгосрочную и краткосрочную информацию, делает их мощными при выполнении прогнозирования исторических данных.

В этой работе мы сравниваем модель ARIMA и LSTM, чтобы выполнить прогнозирование для набора данных по обменному курсу рубля к доллару США.

# 

## Основная часть

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt

import matplotlib.pyplot as plt  
%matplotlib inline  
  
from statsmodels.graphics.tsaplots import plot\_acf  
from statsmodels.graphics.tsaplots import plot\_pacf  
  
from statsmodels.tsa.stattools import adfuller  
from statsmodels.tsa.seasonal import seasonal\_decompose  
from statsmodels.tsa.stattools import acf, pacf  
from statsmodels.tsa.arima.model import ARIMA  
from statsmodels.tsa.arima.model import ARIMAResults  
from sklearn.metrics import mean\_squared\_error as MSE  
  
from keras.models import Sequential  
from keras.layers import Dense  
from keras.optimizers import Adam  
import keras.backend as K  
from keras import layers  
from keras.initializers import Zeros  
from keras.callbacks import EarlyStopping  
from keras.models import load\_model  
  
  
# Set random seed  
np.random.seed(0)

from keras.models import Sequential  
from keras.layers import Dense  
from keras.optimizers import Adam  
import keras.backend as K  
from keras import layers  
from keras.initializers import Zeros

# Set figure size for all matplotlib figures  
fig\_size = plt.rcParams["figure.figsize"]  
fig\_size[0] = 10  
fig\_size[1] = 5

## Импорт датасета

import pandas as pd  
df = pd.read\_csv('USD\_RUB.csv', sep=",")  
df.date = pd.to\_datetime(df.Date)  
df = df.set\_index('Date').interpolate().drop(columns=['Open', 'High', 'Low', 'Change %'])  
print(df.isna().sum())  
df = df.copy().dropna(axis=0)  
rate\_data = df.iloc[::-1]  
rate\_data = rate\_data[3800:5250]

Price 0  
dtype: int64

<ipython-input-73-8983486cd661>:3: UserWarning: Pandas doesn't allow columns to be created via a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access  
 df.date = pd.to\_datetime(df.Date)

## Визуализация данных

rate\_data.head()

Price  
Date   
Apr 13 2016 66.2975  
Apr 14 2016 66.1361  
Apr 15 2016 66.4050  
Apr 18 2016 66.0847  
Apr 19 2016 65.8547

rate\_data.tail()

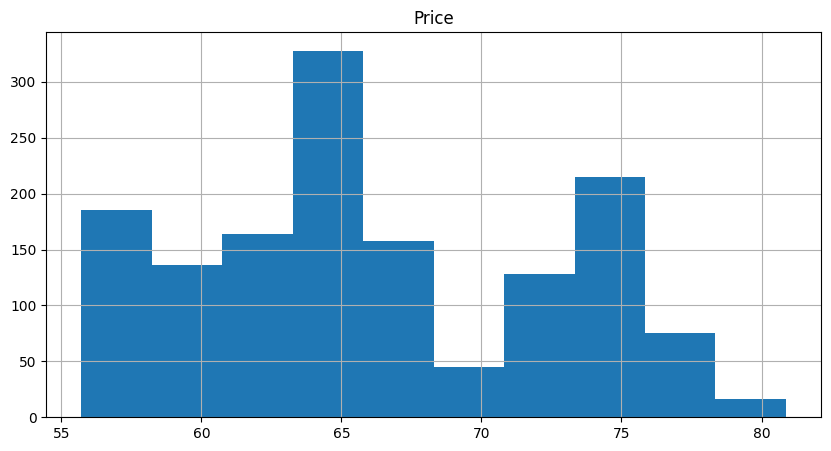
Price  
Date   
Dec 30 2021 74.6539  
Jan 03 2022 74.5508  
Jan 04 2022 75.3700  
Jan 05 2022 76.7857  
Jan 06 2022 76.3130

rate\_data.describe()

Price  
count 1450.000000  
mean 66.162735  
std 6.222516  
min 55.713400  
25% 61.971500  
50% 65.132100  
75% 72.763250  
max 80.869200

Все данные находятся в диапазоне от 55,7 до 80,8 в пределах интервала. Среднее значение и стандартное отклонение близки, что указывает на то, что экстремальных значений либо нет, либо, даже если они есть, они сбалансированы одинаково с обеих сторон данных. По этой причине только квартильных и средних данных недостаточно, чтобы получить представление о распределении данных, необходимо построить гистограмму.

# Plot a histogram  
rate\_data.hist()  
plt.show()



Из гистограммы видно, что значения появляются с наибольшей частотой около 65. Однако существует также огромная частота в диапазоне 73-75. Поскольку ряд является временным рядом, он говорит о том, что значения были постоянными в районе 65 в течение длительного времени, внезапно подскочили примерно до 73, оставались в диапазоне 73-75 в течение некоторого времени.

plt.rcParams["figure.figsize"] = fig\_size  
rate\_data.plot()  
plt.ylabel('Значение USD в рублях')  
plt.grid()



## Анализ стационарности

  Временной ряд называется стационарным, если его среднее значение, дисперсия и структура автокорреляции постоянны относительно временного горизонта ряда. В стационарном временном ряду тенденции не влияют на распознавание закономерностей в данных, поскольку мы начинаем с устранения тренда. Следовательно, прогнозировать стационарный временной ряд легче, чем нестационарный. Практически невозможно получить ряд, который был бы полностью стационарным. По этой причине при анализе временных рядов стационарный ряд относится к слабостационарному ряду, и многие модели прогнозирования основаны на предположении о слабой стационарности. Существуют определенные преобразования, которые реализуются для приведения нестационарного временного ряда в слабостационарную форму.

### Преобразования ряда в слабостационарный

   Слабая стационарность требует, чтобы все корни авторегрессионного запаздывающего многочлена были больше единицы. Если они равны или меньше единицы, наше предположение о стационарности не выполняется. Мы используем расширенный тест Дики-Фуллера (ADF) для проверки стационарности преобразованного ряда. ADF проверяет нулевую гипотезу о том, что ряд имеет единичный корень. Следовательно, после выполнения проверки гипотезы показатели, возвращаемые ADF, которые говорят в нашу пользу (те, которые указывают на стационарность), являются:   1. Высокое абсолютное значение t-статистики (предпочтительно выше критического значения в 1%)   2. Значение 'p-value' очень близко к нулю.

# ADF test on raw data (without transformation)  
  
df\_test = adfuller(rate\_data['Price'], autolag='AIC')  
result = pd.Series(df\_test[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])  
for key,value in df\_test[4].items():  
 result['Critical Value (%s)'%key] = value  
print (result)

Test Statistic -1.171938  
p-value 0.685549  
#Lags Used 3.000000  
Number of Observations Used 1446.000000  
Critical Value (1%) -3.434880  
Critical Value (5%) -2.863541  
Critical Value (10%) -2.567835  
dtype: float64

#### Итог:

1. Абсолютное значение t-статистики намного больше любого критического значения.
2. значение p очень близко к 1, а не к 0. Следовательно, мы принимаем нулевую гипотезу. Другими словами, этот ряд далек от стационарности.

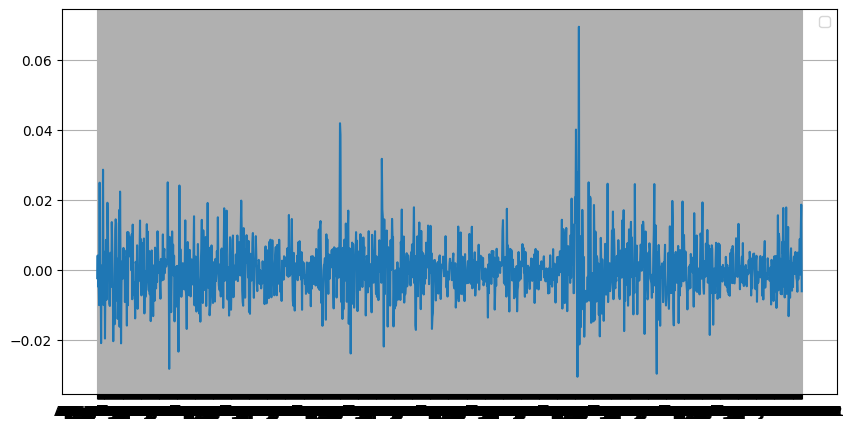
### Приводим к стационарности

# Calculate and plot logarithm of the data  
rate\_data\_log=np.log(rate\_data['Price'])  
plt.rcParams["figure.figsize"] = fig\_size  
rate\_data\_log.plot(label='Log Data')  
plt.legend(loc='best')  
plt.grid()  
plt.show()



# Calculate and plot the first order differnece  
  
rate\_log\_diff = rate\_data\_log - rate\_data\_log.shift()  
rate\_log\_diff.dropna(inplace=True)  
plt.rcParams["figure.figsize"] = fig\_size  
plt.plot(rate\_log\_diff)  
plt.legend('Log Data')  
plt.legend(loc='best')  
plt.grid()  
plt.show()

WARNING:matplotlib.legend:No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



# Perform ADF test  
  
df\_test\_logdiff = adfuller(rate\_log\_diff, autolag='AIC')  
result\_logdiff = pd.Series(df\_test\_logdiff[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])  
for key,value in df\_test\_logdiff[4].items():  
 result\_logdiff['Critical Value (%s)'%key] = value  
print (result\_logdiff)

Test Statistic -23.055448  
p-value 0.000000  
#Lags Used 2.000000  
Number of Observations Used 1446.000000  
Critical Value (1%) -3.434880  
Critical Value (5%) -2.863541  
Critical Value (10%) -2.567835  
dtype: float64

#### Итог:

1. Значение t-статистики намного меньше любого из критических значений.
2. Значение p равно нулю.

## Предсказывание значений

#### Autoregressive Integrated Moving Average (ARIMA)

rate\_data\_log

Date  
Apr 13 2016 4.194152  
Apr 14 2016 4.191715  
Apr 15 2016 4.195772  
Apr 18 2016 4.190937  
Apr 19 2016 4.187451  
 ...   
Dec 30 2021 4.312863  
Jan 03 2022 4.311481  
Jan 04 2022 4.322409  
Jan 05 2022 4.341018  
Jan 06 2022 4.334843  
Name: Price, Length: 1450, dtype: float64

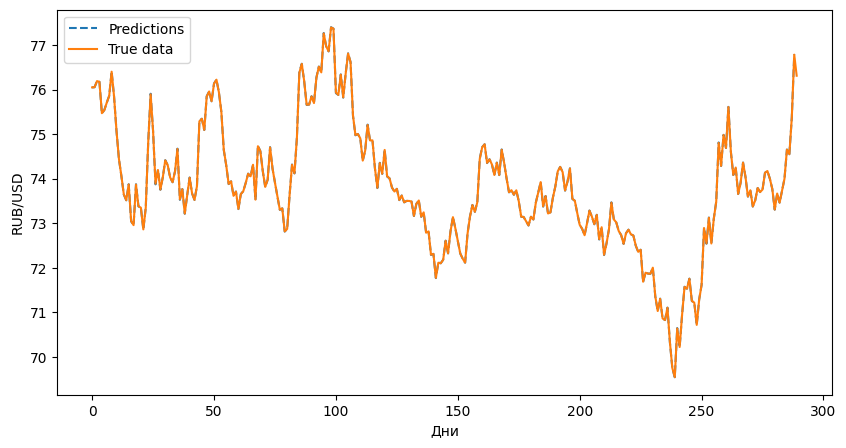
data1 = rate\_data\_log.values  
train\_size = int(len(data1) \* 0.80)  
train = list(data1[0:train\_size])  
train\_ = list(data1[0:train\_size])  
test = list(data1[train\_size:])  
otest = list(data1[train\_size:0])  
  
train\_list = [x for x in train]  
predictions = []  
  
for t in range(len(test)):  
 model = ARIMA(np.array(train),order=(1,1,0))  
 fit = model.fit()  
 forecast = float(fit.forecast()[0])  
 predictions.append(forecast)  
 train.append(float(test[t]))

/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle\_retvals  
 warnings.warn("Maximum Likelihood optimization failed to "  
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607:

model = ARIMA(np.array(train),order=(1,1,0))  
fit = model.fit()  
full\_predictions=float(fit.forecast()[0])

plt.plot(np.exp(predictions)[1:],'--',label='Predictions')  
plt.plot(np.exp(test),label='True data')  
#plt.plot(np.exp(full\_predictions),label='Full prediction')  
plt.ylabel('RUB/USD')  
plt.xlabel('Дни')  
plt.legend()  
plt.plot()  
  
print ('Mean squared error: {0:3f}'.format(MSE(np.exp(test),np.exp(predictions))))

Mean squared error: 0.215888



#Значение завтра  
train = list(data1)  
  
train\_list = [x for x in train]  
predictions = []  
  
model = ARIMA(np.array(train),order=(1,1,0))  
fit = model.fit()  
forecast = float(fit.forecast()[0])  
  
tomorrow\_value = np.exp(forecast)  
  
#Значение сегодня  
  
rate\_data\_log = rate\_data\_log[:-1]  
data1 = rate\_data\_log.values  
train = list(data1)  
  
train\_list = [x for x in train]  
predictions = []  
  
model = ARIMA(np.array(train),order=(1,1,0))  
fit = model.fit()  
forecast = float(fit.forecast()[0])  
today\_value = np.exp(forecast)

rate\_data.tail()

Price  
Date   
Dec 30 2021 74.6539  
Jan 03 2022 74.5508  
Jan 04 2022 75.3700  
Jan 05 2022 76.7857  
Jan 06 2022 76.3130

print ("Реальное значение сегодня - {0:3f}, предсказанное - {1:3f}.".format(rate\_data.values[-1][0],today\_value))  
print ("Предсказанное значение на завтра - {0:3f}.".format(tomorrow\_value))

Реальное значение сегодня - 76.313000, предсказанное - 76.791905.  
Предсказанное значение на завтра - 76.311567.

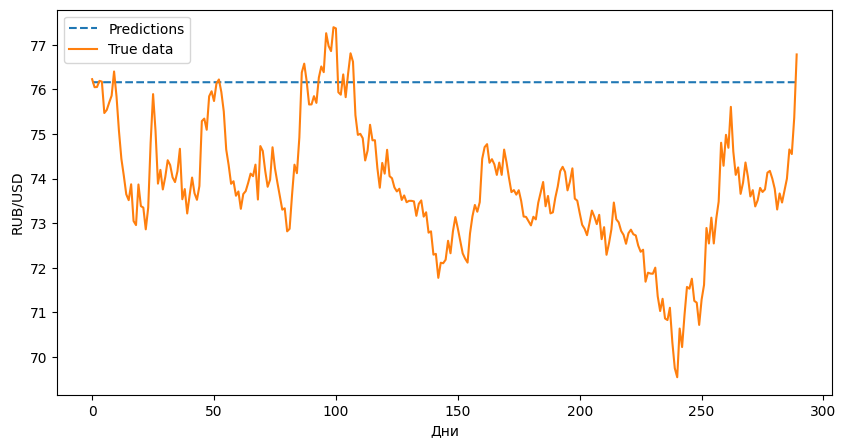
#### Предсказания на все дни в выборке

train\_list = [x for x in train]  
predictions = []  
  
for t in range(len(test)):  
 model = ARIMA(np.array(train),order=(1,1,0))  
 fit = model.fit()  
 forecast = float(fit.forecast()[0])  
 predictions.append(forecast)  
 train.append(float(forecast))

warnings.warn("Maximum Likelihood optimization failed to "  
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle\_retvals  
 warnings.warn("Maximum Likelihood optimization failed to "  
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle\_retvals  
 warnings.warn("Maximum Likelihood optimization failed to "  
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check mle\_retvals  
 warnings.warn("Maximum Likelihood optimization failed to "

# predictions = fit.forecast(len(test))[0]  
plt.plot(np.exp(predictions),'--',label='Predictions')  
plt.plot(np.exp(test),label='True data')  
#plt.plot(np.exp(full\_predictions),label='Full prediction')  
plt.ylabel('RUB/USD')  
plt.xlabel('Дни')  
plt.legend()  
plt.plot()  
  
print ('Mean squared error: {0:3f}'.format(MSE(np.exp(test),np.exp(predictions))))

Mean squared error: 7.443427



Поскольку ARIMA является линейной моделью, прогнозирование с использованием ARIMA на срок более 1 дня совершенно неэффективно. Модель продолжает восходящий тренд до бесконечности.

### LSTM- Recurrent Neural Network (LSTM-RNN)

data1 = rate\_data\_log.values  
train\_size = int(len(data1) \* 0.80)  
train = list(data1[0:train\_size])  
test = list(data1[train\_size:])

def gen\_XY(t\_list,size):  
 '''  
 Generates an autoregressive lagged array.  
  
 Parameters  
 ----------  
 1. t\_list: (list) Data to generate an autoregressive array out of.  
 2. size: (int) Number of lagged values to consider.  
  
 Returns  
 ----------  
 1. X: (np.ndarray) Autoregressive feature array.  
 2. Y.T: (np.ndarray) Column array with values of response variable.  
  
 '''  
 X = np.zeros([1,size])  
 Y = []  
 temp\_list=[]  
 for i in range(len(t\_list)-size):  
 add\_list = t\_list[i:i+size]  
 target = t\_list[i+size]  
 X = np.append(X,np.array([add\_list]),axis=0)  
 Y.append(target)  
  
 X = np.delete(X,0,axis=0)  
 Y = np.array(Y)  
  
 return X,Y.T

X\_train,y\_train = gen\_XY(train,1)  
X\_test,y\_test = gen\_XY(test,1)

X\_tr\_t = X\_train.reshape(X\_train.shape[0], 1, X\_train.shape[1])  
X\_tst\_t = X\_test.reshape(X\_test.shape[0], 1, X\_test.shape[1])

X\_tr\_t.shape

(1158, 1, 1)

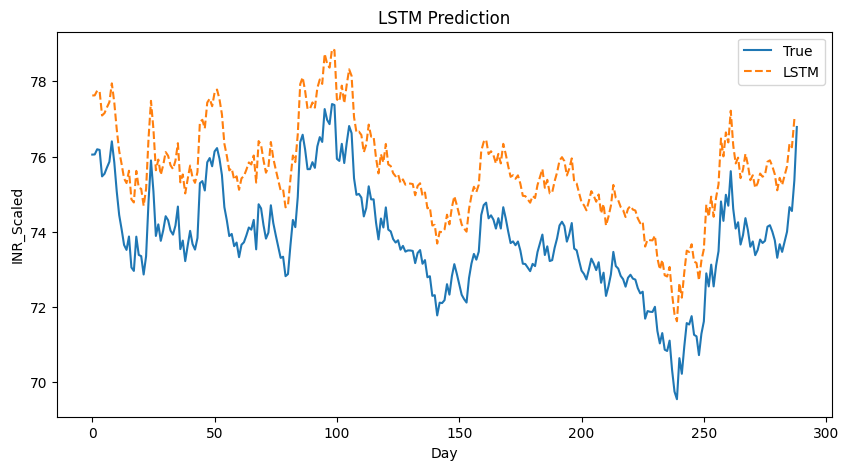
from keras.layers import LSTM  
K.clear\_session()  
model\_lstm = Sequential()  
model\_lstm.add(LSTM(10, input\_shape=(1, X\_train.shape[1]), activation='relu', kernel\_initializer='lecun\_uniform', return\_sequences=False))  
model\_lstm.add(Dense(10,input\_shape=(1, X\_train.shape[1]), activation='relu'))  
  
model\_lstm.add(Dense(1))  
model\_lstm.compile(loss='mean\_squared\_error', optimizer='adam')  
early\_stop = EarlyStopping(monitor='loss', patience=5, verbose=1)  
history\_model\_lstm = model\_lstm.fit(X\_tr\_t, y\_train, epochs=200, batch\_size=1, verbose=1, shuffle=False, callbacks=[early\_stop])

Epoch 1/200  
1158/1158 [==============================] - 6s 4ms/step - loss: 0.5679  
Epoch 2/200  
1158/1158 [==============================] - 3s 3ms/step - loss: 7.3089e-05  
Epoch 3/200  
1158/1158 [==============================] - 4s 3ms/step - loss: 7.6281e-05  
Epoch 4/200  
1158/1158 [==============================] - 5s 4ms/step - loss: 8.2054e-05  
Epoch 5/200  
1158/1158 [==============================] - 4s 3ms/step - loss: 9.5769e-05  
Epoch 6/200  
1158/1158 [==============================] - 3s 3ms/step - loss: 1.2267e-04  
Epoch 7/200  
1158/1158 [==============================] - 4s 3ms/step - loss: 1.3922e-04  
Epoch 7: early stopping

y\_pred\_test\_lstm = model\_lstm.predict(X\_tst\_t)  
y\_train\_pred\_lstm = model\_lstm.predict(X\_tr\_t)  
  
print("The MSE on the Train set is:\t{:0.3f}".format(MSE(np.exp(y\_train), np.exp(y\_train\_pred\_lstm))))  
print("The MSE on the Test set is:\t{:0.3f}".format(MSE(np.exp(y\_test), np.exp(y\_pred\_test\_lstm))))

10/10 [==============================] - 0s 4ms/step  
37/37 [==============================] - 0s 4ms/step  
The MSE on the Train set is: 6.066  
The MSE on the Test set is: 3.274

plt.plot(np.exp(y\_test), label='True')  
plt.plot(np.exp(y\_pred\_test\_lstm)[1:],'--',label='LSTM')  
plt.title("LSTM Prediction")  
plt.xlabel('Day')  
plt.ylabel('INR\_Scaled')  
plt.legend()  
plt.show()



До сих пор мы прогнозировали только прогноз на один день вперед. Естественно, почему вас должно интересовать только прогнозирование стоимости завтрашнего дня? Однако ошибка распространяется очень быстро при прогнозировании во временных рядах, особенно в одномерном сценарии, таком как этот. В некоторых случаях, скажем, если частота повторяется ежегодно, даже прогноз на один шаг вперед может дать много информации. Но в данном случае давайте посмотрим, насколько точен 3-недельный прогноз на основе тестовых данных.

### Пердсказания на несколько дней вперед

data1 = rate\_data\_log.values  
train = list(data1[0:train\_size])  
test = list(data1[train\_size:])

X\_train,y\_train = gen\_XY(train,10)  
# X\_test,y\_test = gen\_XY(test,10)

X\_tr\_t = X\_train.reshape(X\_train.shape[0], 1, X\_train.shape[1])

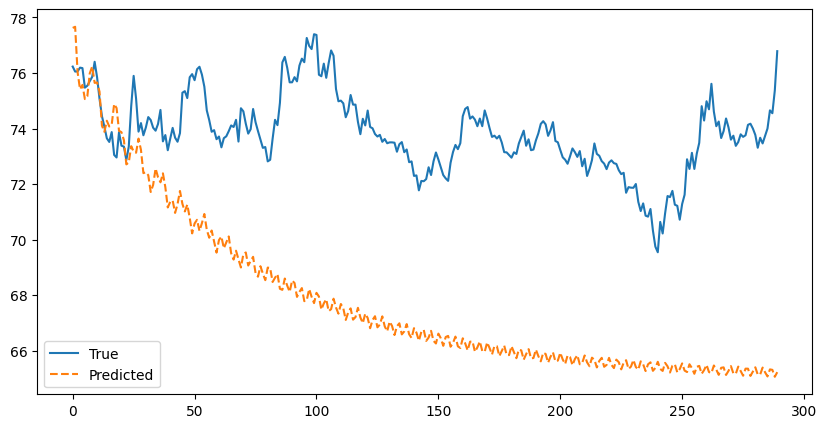
K.clear\_session()  
model\_lstm = Sequential()  
model\_lstm.add(LSTM(5, input\_shape=(1, X\_train.shape[1]), activation='relu', kernel\_initializer='lecun\_uniform', return\_sequences=False))  
model\_lstm.add(Dense(5,input\_shape=(1, X\_train.shape[1]), activation='relu'))  
  
model\_lstm.add(Dense(1))  
model\_lstm.compile(loss='mean\_squared\_error', optimizer='adam')  
early\_stop = EarlyStopping(monitor='loss', patience=5, verbose=1)  
history\_model\_lstm = model\_lstm.fit(X\_tr\_t, y\_train, epochs=200, batch\_size=1, verbose=1, shuffle=False, callbacks=[early\_stop])

Epoch 1/200  
1149/1149 [==============================] - 5s 3ms/step - loss: 0.5509  
Epoch 2/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 5.0929e-04  
Epoch 3/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 5.9634e-04  
Epoch 4/200  
1149/1149 [==============================] - 4s 4ms/step - loss: 6.3731e-04  
Epoch 5/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 6.2071e-04  
Epoch 6/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 5.1321e-04  
Epoch 7/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 3.7912e-04  
Epoch 8/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.9375e-04  
Epoch 9/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.3815e-04  
Epoch 10/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.2673e-04  
Epoch 11/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.2741e-04  
Epoch 12/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.2727e-04  
Epoch 13/200  
1149/1149 [==============================] - 4s 4ms/step - loss: 2.2654e-04  
Epoch 14/200  
1149/1149 [==============================] - 8s 7ms/step - loss: 2.2574e-04  
Epoch 15/200  
1149/1149 [==============================] - 7s 6ms/step - loss: 2.2499e-04  
Epoch 16/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.2432e-04  
Epoch 17/200  
1149/1149 [==============================] - 5s 4ms/step - loss: 2.2372e-04  
Epoch 18/200  
1149/1149 [==============================] - 6s 5ms/step - loss: 2.2319e-04  
Epoch 19/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.2273e-04  
Epoch 20/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.2233e-04  
Epoch 21/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.2200e-04  
Epoch 22/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.2173e-04  
Epoch 23/200  
1149/1149 [==============================] - 4s 4ms/step - loss: 2.2152e-04  
Epoch 24/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.2137e-04  
Epoch 25/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.2128e-04  
Epoch 26/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.2125e-04  
Epoch 27/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.2128e-04  
Epoch 28/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.2138e-04  
Epoch 29/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.2153e-04  
Epoch 30/200  
1149/1149 [==============================] - 3s 3ms/step - loss: 2.2176e-04  
Epoch 31/200  
1149/1149 [==============================] - 4s 3ms/step - loss: 2.2205e-04  
Epoch 31: early stopping

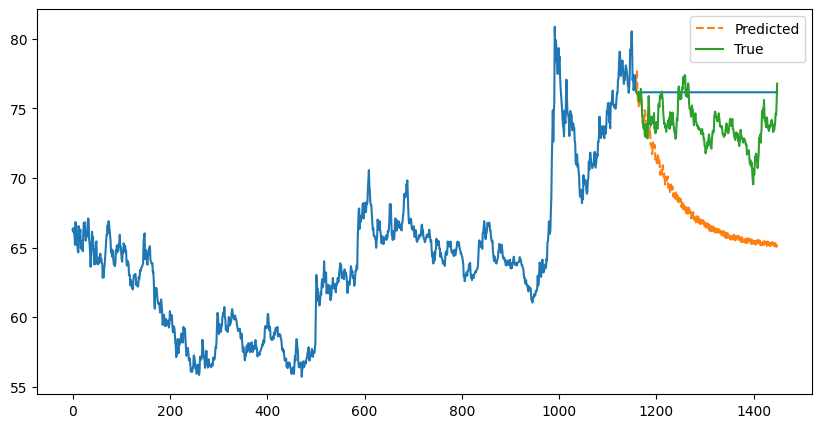
test\_array = X\_tr\_t[-1][0]  
pred = []  
forecast\_length = len(test)  
for i in range(forecast\_length):  
 y\_train\_pred\_lstm = model\_lstm.predict(test\_array.reshape(1,1,10))  
 pred.append(y\_train\_pred\_lstm[0][0])  
 test\_array = np.append(test\_array[1:],y\_train\_pred\_lstm[0][0])

1/1 [==============================] - 0s 22ms/step  
1/1 [==============================] - 0s 25ms/step  
1/1 [==============================] - 0s 22ms/step

plt.plot(np.exp(test),label='True')  
plt.plot(np.exp(pred),'--',label='Predicted')  
plt.legend()  
plt.show()



plt.plot(np.exp(train))  
plt.plot(range(train\_size,len(data1)),np.exp(pred),'--',label='Predicted')  
plt.plot(range(train\_size,len(data1)),np.exp(test),label='True')  
plt.legend()  
plt.show()  
MSE(np.exp(test),np.exp(pred))



43.19630690967806

## ЗАКЛЮЧЕНИЕ

В результате сравнения моделей было сделано несколько выводов:

1. ARIMA отлично показывает предсказание на 1 день. Поскольку ARIMA является линейной моделью, прогнозирование с использованием ARIMA на срок более 1 дня совершенно неэффективно. Модель продолжает восходящий тренд до бесконечности.
2. LSTM немного хуже показывает себя на предсказаниях в течение 1го дня, чем ARIMA
3. LSTM показывает неплохой результат на промежутке до 20 дней
4. При окне большем 20 дней модель LSTM также показывает слабый результат

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1)             Карта методов scikit - https://scikit-learn.org/stable/tutorial/machine\_learning\_map/index.html

2)             Данные по курсу RUB/USD - https://www.moex.com/en/index/rtsusdcur.aspx?tid=2601

3)             Курс лекций Разработка нейронных сетей - https://github.com/iu5git/Deep-learning