РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра информационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

Дисциплина: Методы машинного обучения

Студент: Кузнецов Юрий Владимирович

Группа: НФИбд 01-20

Москва 2023

Вариант 16

1. Считайте котировки акции с указанным ниже тикером за 2018-2019 год: MDB Mongodb Inc Cl A
2. Показатель акции: дневная доходность, способ визуализации – диаграмма размаха
3. Прогнозирование стоимости акции через 12 дней по данным за предыдущие 25 дней.
4. Показатель качества R2
5. Техника борьбы с исчезающими градиентами: Функция активации без насыщения

Задание:

В соответствии с индивидуальным заданием, указанным в записной книжке команды, выполните следующие работы:

1. При помощи модуля pandas_datareader считайте котировки указанной в индивидуальным задании акции за указанный период времени.
2. Визуализируйте котировки акции (столбец `Adj Close`) за весь период на графике. Подпишите оси и рисунок.
3. Вычислите и визуализируйте заданный показатель акции в соответствии с индивидуальным заданием.
4. Сформируйте обучающую, тестовую и валидационные выборки для обучения нейронной сети в соответствии с индивидуальным заданием.
5. Постройте нейронную сеть MLP с нормализующим слоем и одним плотным скрытым слоем из 16 нейронов для прогнозирования стоимости акции и обучите ее на обучающей выборке. Оцените качество прогнозирования при помощи заданного показателя качества на тестовой выборке.
6. Примените указанную в индивидуальном задании технику решения проблемы исчезающих градиентов и постройте нейронную сеть MLP с нормализующим слоем и тремя плотными скрытыми слоями из 16 нейронов для прогнозирования стоимости акции и обучите ее на обучающей выборке. Оцените качество прогнозирования при помощи заданного показателя качества для тестовой выборки.
7. Постройте рекуррентную нейронную сеть с нормализующим слоем и одним скрытым слоем LSTM из 16 нейронов для прогнозирования стоимости акции и обучите ее на обучающей выборке. Оцените качество прогнозирования при помощи заданного показателя качества на тестовой выборке.
8. Визуализируйте кривые обучения для трех построенных моделей на одном рисунке в зависимости от эпохи обучения, подписывая оси и рисунок и создавая легенду. Используйте для визуализации относительную ошибку (ошибку обучения, деленную на начальную ошибку на первой эпохе).
9. Визуализируйте весь набор данных и прогнозы трех построенных моделей для обучающей и тестовой выборок на одном рисунке (ось X – даты, ось Y – стоимость акции), подписывая оси и рисунок и создавая легенду.

Решение

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf

from sklearn.metrics import mean_squared_error


from pandas_datareader import data as pdr
import yfinance as yfin
import datetime as dt

yfin.pdr_override()
```

1. Загрузим котировки акции компании DIS Walt Disney Company за 2020 и 2021 года:

```
data = pdr.get_data_yahoo('MDB',
                          start=dt.datetime(2018, 1, 1),
                          end=dt.datetime(2019, 12, 31))
data = data.dropna()
data.head()
```

```
[**********100%**********]  1 of 1 completed
```

Date	Open	High	Low	Close	Adj Close	Volume
2018-01-02	29.930000	30.080	28.855000	29.250000	29.250000	231600
2018-01-03	29.309999	29.420	28.940001	29.150000	29.150000	256700
2018-01-04	29.290001	29.490	28.900000	29.049999	29.049999	198900
2018-01-05	29.090000	29.135	28.590000	29.049999	29.049999	205600
2018-01-08	29.090000	29.090	28.209999	28.809999	28.809999	205400

```
data.tail()
```

Date	Open	High	Low	Close	Adj Close	Volume
2019-12-23	130.750000	133.289993	130.110001	132.360001	132.360001	641700
2019-12-24	132.389999	133.078995	130.473999	132.550003	132.550003	501100
2019-12-26	132.919998	137.442993	132.309998	135.020004	135.020004	705900
2019-12-27	135.520004	135.850006	132.630005	134.130005	134.130005	441100
2019-12-30	133.880005	134.660004	128.453003	128.929993	128.929993	904800

2. Визуализируем котировки акции (столбец Adj Close) за весь период на графике:

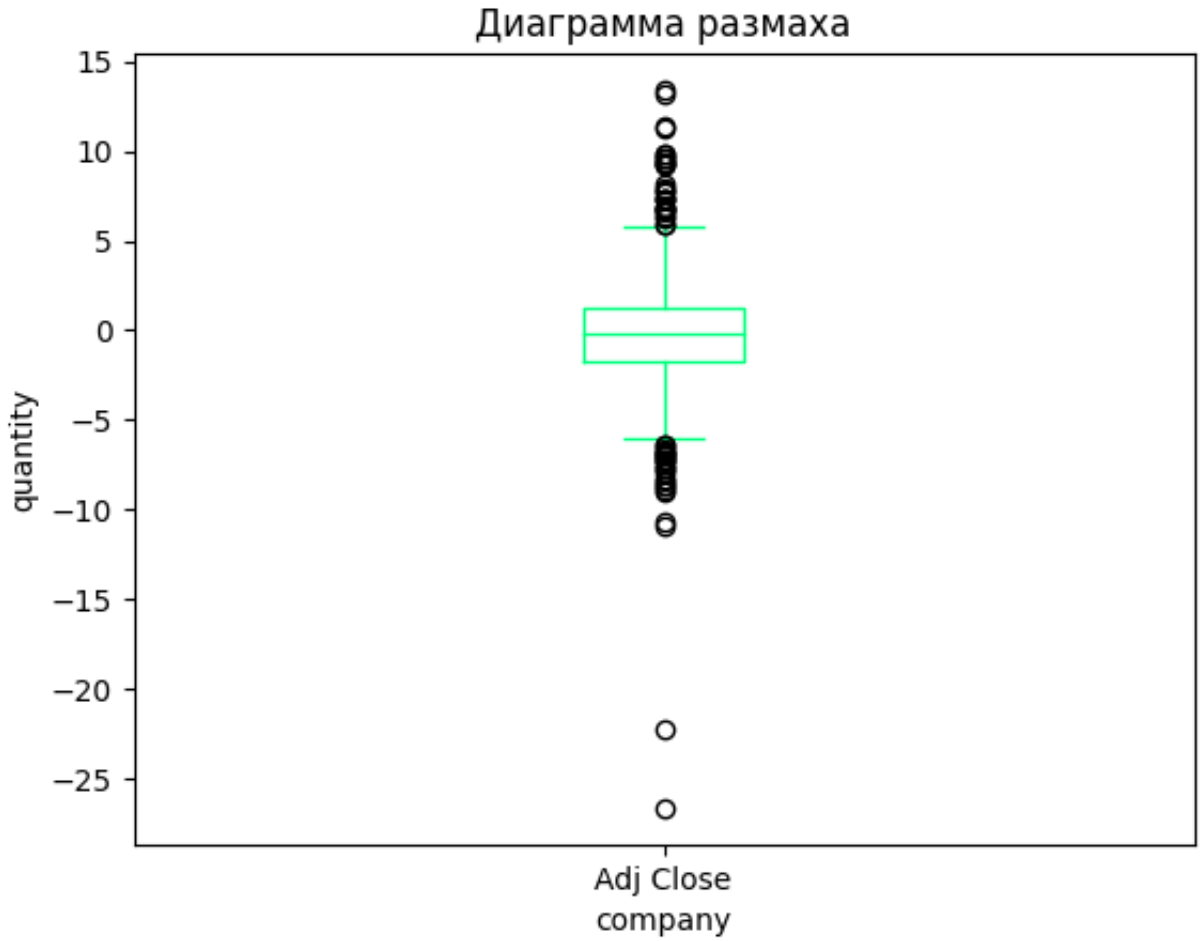
```
data['Adj Close'].plot.line(grid=True,title='Котировки акций компании MDB за 201
```

```
<Axes: title={'center': 'Котировки акций компании MDB за 2018-2019 гг.'},
xlabel='Date', ylabel='Price'>
```

3. Вычислим и визуализируем дневной доход с помощью диаграммы размаха:

```
(data[['Adj Close']].shift(1) - data[['Adj Close']]).plot.box(color='springgreen
plt.title('Диаграмма размаха')
plt.xlabel('company')
plt.ylabel('quantity')
```

```
Text(0, 0.5, 'quantity')
```



4. Сформируем обучающую, тестовую и валидационные выборки для обучения нейронной сети:

```
num_train_samples = int(0.5 * len(data))
num_val_samples = int(0.25 * len(data))
num_test_samples = len(data) - num_train_samples - num_val_samples
print("num_train_samples:", num_train_samples)
print("num_val_samples:", num_val_samples)
print("num_test_samples:", num_test_samples)
```

```
num_train_samples: 251
num_val_samples: 125
num_test_samples: 126
```

```
sampling_rate = 1
sequence_length = 25
delay = sampling_rate * (sequence_length + 12 - 1)
batch_size = 64

train_dataset = tf.keras.utils.timeseries_dataset_from_array(
    data[:-delay],
    targets=data['Adj Close'][delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=0,
    end_index=num_train_samples - delay)

val_dataset = tf.keras.utils.timeseries_dataset_from_array(
    data[:-delay],
    targets=data['Adj Close'][delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=num_train_samples - delay,
    end_index=num_train_samples + num_val_samples - delay)

test_dataset = tf.keras.utils.timeseries_dataset_from_array(
    data[:-delay],
    targets=data['Adj Close'][delay:],
    sampling_rate=sampling_rate,
    sequence_length=sequence_length,
    shuffle=True,
    batch_size=batch_size,
    start_index=num_train_samples + num_val_samples - delay)
```

5. Построим нейронную сеть MLP с нормализующим слоем и одним плотным скрытым слоем из 16 нейронов для прогнозирования стоимости акции и обучим ее на обучающей выборке:

```
X = data
feature_normalizer = tf.keras.layers.Normalization(axis=None,input_shape=(sequen
feature_normalizer.adapt(X)


inputs = tf.keras.Input(shape=(sequence_length, data.shape[-1]))
x = tf.keras.layers.Flatten()(inputs)
x = feature_normalizer(x)
x = tf.keras.layers.Dense(16, activation="relu")(x)
outputs = tf.keras.layers.Dense(1)(x)
model1 = tf.keras.Model(inputs, outputs)

model1.compile(optimizer="rmsprop", loss="mse", metrics=["mse"])
history1 = model1.fit(train_dataset,
                      epochs=20,
                      validation_data=val_dataset,

                     )
```

```
Epoch 1/20
3/3 [==============================] - 1s 145ms/step - loss: 3442.4109 - ms
Epoch 2/20
3/3 [==============================] - 0s 30ms/step - loss: 3327.4458 - mse
Epoch 3/20
3/3 [==============================] - 0s 38ms/step - loss: 3230.6184 - mse
Epoch 4/20
3/3 [==============================] - 0s 30ms/step - loss: 3148.3862 - mse
Epoch 5/20
3/3 [==============================] - 0s 31ms/step - loss: 3073.7739 - mse
Epoch 6/20
3/3 [==============================] - 0s 32ms/step - loss: 3003.4800 - mse
Epoch 7/20
3/3 [==============================] - 0s 39ms/step - loss: 2935.7451 - mse
Epoch 8/20
3/3 [==============================] - 0s 39ms/step - loss: 2869.4983 - mse
Epoch 9/20
3/3 [==============================] - 0s 38ms/step - loss: 2804.1589 - mse
Epoch 10/20
3/3 [==============================] - 0s 38ms/step - loss: 2739.3118 - mse
Epoch 11/20
3/3 [==============================] - 0s 32ms/step - loss: 2674.7537 - mse
Epoch 12/20
3/3 [==============================] - 0s 42ms/step - loss: 2610.4644 - mse
Epoch 13/20
3/3 [==============================] - 0s 35ms/step - loss: 2546.2244 - mse
Epoch 14/20
3/3 [==============================] - 0s 40ms/step - loss: 2481.9219 - mse
Epoch 15/20
3/3 [==============================] - 0s 38ms/step - loss: 2417.9136 - mse
Epoch 16/20
3/3 [==============================] - 0s 38ms/step - loss: 2353.8135 - mse
Epoch 17/20
3/3 [==============================] - 0s 38ms/step - loss: 2289.5181 - mse
Epoch 18/20
3/3 [==============================] - 0s 38ms/step - loss: 2225.4690 - mse
Epoch 19/20
3/3 [==============================] - 0s 32ms/step - loss: 2161.5303 - mse
Epoch 20/20
3/3 [==============================] - 0s 38ms/step - loss: 2097.7002 - mse
```

Оценим качество прогнозирования при помощи RMSE на тестовой выборке:

```
print(test_dataset)
```

```
<_BatchDataset element_spec=(TensorSpec(shape=(None, None, 6), dtype=tf.flo
```

```
print(val_dataset)
```

```
<_BatchDataset element_spec=(TensorSpec(shape=(None, None, 6), dtype=tf.flo
```

```
import numpy as np
from sklearn.metrics import r2_score

# Assuming you have predictions and actual values for the test set
y_pred = model1.predict(test_dataset)
y_true = np.array(data['Adj Close'][delay + num_train_samples + num_val_samples:
```

```
2/2 [==============================] - 0s 6ms/step
```

```
print(len(y_pred), len(y_true))
```

```
102 90
```

```
y_pred = y_pred[:len(y_true)]
```

```
r2 = r2_score(y_true, y_pred)
print("R2 Score:", r2)
```

```
R2 Score: -138.44711197535537
```

Из отрицательного значения R2 видим, что модель неадекватна для предсказания данных или что данные имеют очень низкую структуру и не могут быть эффективно моделированы.

```
print(f"MSE на тестовом наборе: {model1.evaluate(test_dataset)[1]:.2f}")
```

```
2/2 [==============================] - 0s 13ms/step - loss: 14052.8574 - ms
MSE на тестовом наборе: 14052.86
```

6. Применим функцию активации без насыщения для решения проблемы исчезающих градиентов и построим нейронную сеть MLP с нормализующим слоем и тремя плотными скрытыми слоями из 16 нейронов для прогнозирования стоимости акции и обучим ее на обучающей выборке.

```
inputs = tf.keras.Input(shape=(sequence_length, data.shape[-1]))
x = tf.keras.layers.Flatten()(inputs)
x = feature_normalizer(x)
x = tf.keras.layers.Dense(16, activation="relu")(x)
x = tf.keras.layers.Activation(tf.keras.activations.linear)(x)  # Функция актива
outputs = tf.keras.layers.Dense(1)(x)
model2 = tf.keras.Model(inputs, outputs)

model2.compile(optimizer="rmsprop", loss="mse", metrics=["mse"])
history2 = model2.fit(train_dataset,
                      epochs=20,
                      validation_data=val_dataset)
```

```
Epoch 1/20
3/3 [==============================] - 2s 213ms/step - loss: 3341.9753 - ms
Epoch 2/20
3/3 [==============================] - 0s 86ms/step - loss: 3208.7021 - mse
Epoch 3/20
3/3 [==============================] - 0s 88ms/step - loss: 3102.6763 - mse
Epoch 4/20
3/3 [==============================] - 0s 83ms/step - loss: 3012.0479 - mse
Epoch 5/20
3/3 [==============================] - 0s 64ms/step - loss: 2932.4829 - mse
Epoch 6/20
3/3 [==============================] - 0s 57ms/step - loss: 2859.6868 - mse
Epoch 7/20
3/3 [==============================] - 0s 50ms/step - loss: 2790.8926 - mse
Epoch 8/20
3/3 [==============================] - 1s 169ms/step - loss: 2724.8569 - ms
Epoch 9/20
3/3 [==============================] - 0s 34ms/step - loss: 2660.5757 - mse
Epoch 10/20
3/3 [==============================] - 0s 38ms/step - loss: 2597.2747 - mse
Epoch 11/20
3/3 [==============================] - 0s 39ms/step - loss: 2534.7742 - mse
Epoch 12/20
3/3 [==============================] - 0s 38ms/step - loss: 2473.0166 - mse
Epoch 13/20
3/3 [==============================] - 0s 29ms/step - loss: 2411.7588 - mse
Epoch 14/20
3/3 [==============================] - 0s 32ms/step - loss: 2350.8074 - mse
Epoch 15/20
3/3 [==============================] - 0s 29ms/step - loss: 2290.3269 - mse
Epoch 16/20
3/3 [==============================] - 0s 38ms/step - loss: 2230.1609 - mse
Epoch 17/20
3/3 [==============================] - 0s 38ms/step - loss: 2170.2202 - mse
Epoch 18/20
3/3 [==============================] - 0s 40ms/step - loss: 2110.5984 - mse
Epoch 19/20
3/3 [==============================] - 0s 39ms/step - loss: 2051.4355 - mse
Epoch 20/20
3/3 [==============================] - 0s 30ms/step - loss: 1992.7631 - mse
```

Оценим качество прогнозирования при помощи MSE на тестовой выборке:

```
import numpy as np
from sklearn.metrics import r2_score

# Assuming you have predictions and actual values for the test set
y_pred = model2.predict(test_dataset)
y_true = np.array(data['Adj Close'][delay + num_train_samples + num_val_samples:

y_pred = y_pred[:len(y_true)]

r2 = r2_score(y_true, y_pred)
print("R2 Score:", r2)
```

```
2/2 [==============================] - 0s 6ms/step
R2 Score: -134.26935169828522
```

```
print(f"MSE на тестовом наборе: {model2.evaluate(test_dataset)[1]:.2f}")
```

```
2/2 [==============================] - 0s 7ms/step - loss: 13634.5781 - mse
MSE на тестовом наборе: 13634.58
```

7. Построим рекуррентную нейронную сеть с нормализующим слоем и одним скрытым слоем LSTM из 16 нейронов для прогнозирования стоимости акции и обучим ее на обучающей выборке:

```
inputs = tf.keras.Input(shape=(sequence_length, data.shape[-1]))
x = feature_normalizer(inputs)
x = tf.keras.layers.LSTM(16, activation="relu")(x)
outputs = tf.keras.layers.Dense(1)(x)
model3 = tf.keras.Model(inputs, outputs)

model3.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001), loss="ms
history3 = model3.fit(train_dataset,
                      epochs=20,
                      validation_data=val_dataset,

                     )
```

```
Epoch 1/20
3/3 [==============================] - 4s 151ms/step - loss: 3467.1118 - ms
Epoch 2/20
3/3 [==============================] - 0s 39ms/step - loss: 3460.8867 - mse
Epoch 3/20
3/3 [==============================] - 0s 39ms/step - loss: 3454.2915 - mse
Epoch 4/20
3/3 [==============================] - 0s 46ms/step - loss: 3447.1978 - mse
Epoch 5/20
3/3 [==============================] - 0s 41ms/step - loss: 3439.6433 - mse
Epoch 6/20
3/3 [==============================] - 0s 40ms/step - loss: 3431.2839 - mse
Epoch 7/20
3/3 [==============================] - 0s 48ms/step - loss: 3422.1509 - mse
Epoch 8/20
3/3 [==============================] - 0s 40ms/step - loss: 3411.6689 - mse
Epoch 9/20
3/3 [==============================] - 0s 40ms/step - loss: 3399.5356 - mse
Epoch 10/20
3/3 [==============================] - 0s 39ms/step - loss: 3384.5647 - mse
Epoch 11/20
3/3 [==============================] - 0s 48ms/step - loss: 3364.8281 - mse
Epoch 12/20
3/3 [==============================] - 0s 43ms/step - loss: 3337.2473 - mse
Epoch 13/20
3/3 [==============================] - 0s 46ms/step - loss: 3291.4299 - mse
Epoch 14/20
3/3 [==============================] - 0s 48ms/step - loss: 3204.0518 - mse
Epoch 15/20
3/3 [==============================] - 0s 47ms/step - loss: 2994.0144 - mse
Epoch 16/20
3/3 [==============================] - 0s 88ms/step - loss: 2527.1882 - mse
Epoch 17/20
3/3 [==============================] - 0s 44ms/step - loss: 1566.9692 - mse
Epoch 18/20
3/3 [==============================] - 0s 45ms/step - loss: 751.0750 - mse:
Epoch 19/20
3/3 [==============================] - 0s 47ms/step - loss: 950.4987 - mse:
Epoch 20/20
3/3 [==============================] - 0s 42ms/step - loss: 611.3481 - mse:
```

Оценим качество прогнозирования при помощи MSE на тестовой выборке:

```
import numpy as np
from sklearn.metrics import r2_score

y_pred = model3.predict(test_dataset)
y_true = np.array(data['Adj Close'][delay + num_train_samples + num_val_samples:

y_pred = y_pred[:len(y_true)]

r2 = r2_score(y_true, y_pred)
print("R2 Score:", r2)
```

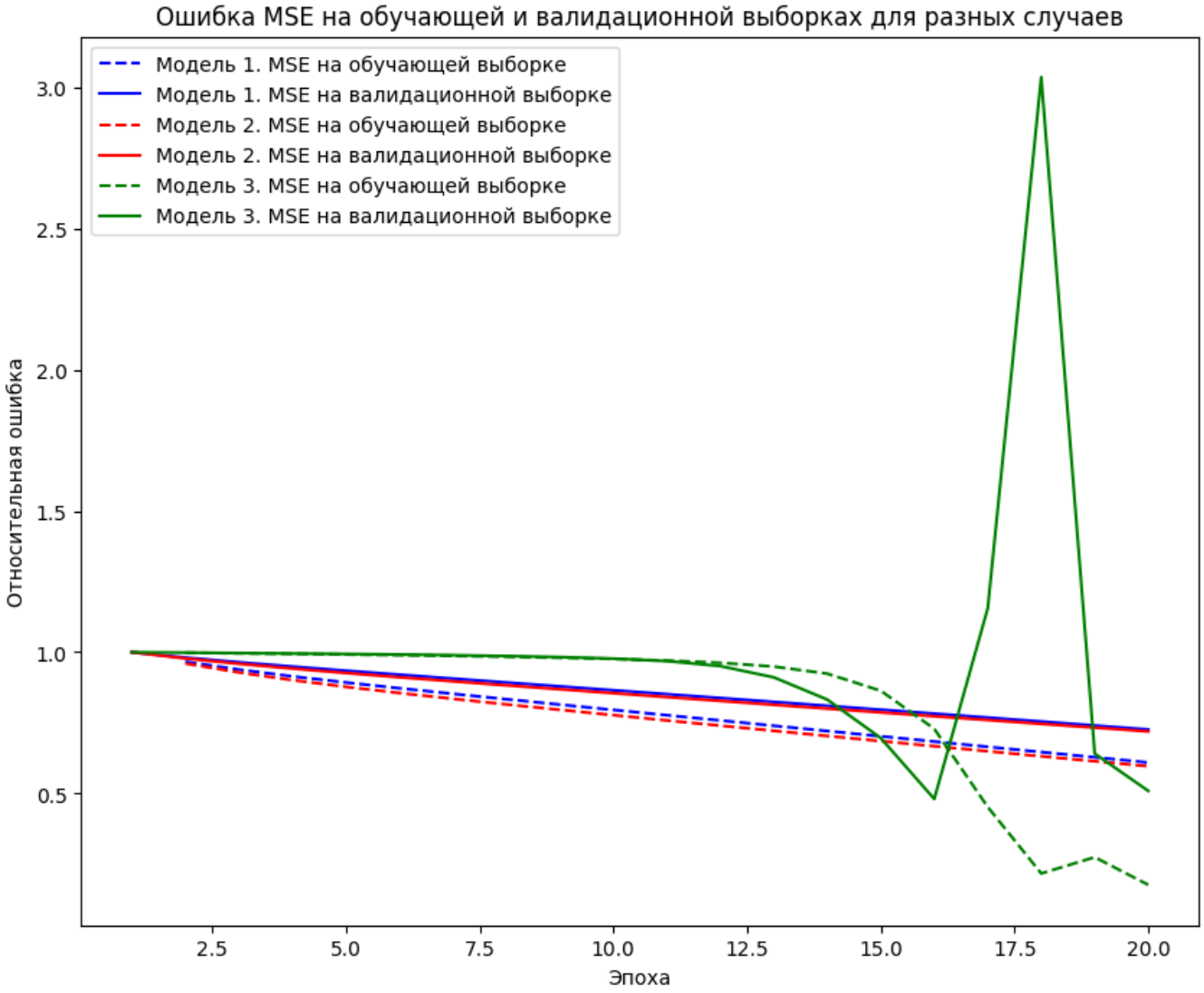
```
2/2 [==============================] - 0s 12ms/step
R2 Score: -27.292441624583454
```

```
print(f"Test MSE: {model3.evaluate(test_dataset)[1]:.2f}")
```

```
2/2 [==============================] - 0s 10ms/step - loss: 2856.1838 - mse
Test MSE: 2856.18
```

8. Визуализируем кривые обучения для трех построенных моделей на одном рисунке в зависимости от эпохи обучения, подписывая оси и рисунок и создавая легенду. Используем для визуализации относительную ошибку:

```
loss1 = np.array(history1.history["mse"]) / history1.history["mse"][0]
val_loss1 = np.array(history1.history["val_mse"]) / history1.history["val_mse"][

loss2 = np.array(history2.history["mse"]) / history2.history["mse"][0]
val_loss2 = np.array(history2.history["val_mse"]) / history2.history["val_mse"][

loss3 = np.array(history3.history["mse"]) / history3.history["mse"][0]
val_loss3 = np.array(history3.history["val_mse"]) / history3.history["val_mse"][

epochs = range(1, len(loss1) + 1)
plt.figure(figsize=(10,8))
plt.plot(epochs[1:], loss1[1:], "b--", label="Модель 1. MSE на обучающей выборке
plt.plot(epochs, val_loss1, "b", label="Модель 1. MSE на валидационной выборке")
plt.plot(epochs[1:], loss2[1:], "r--", label="Модель 2. MSE на обучающей выборке
plt.plot(epochs, val_loss2, "r", label="Модель 2. MSE на валидационной выборке")
plt.plot(epochs[1:], loss3[1:], "g--", label="Модель 3. MSE на обучающей выборке
plt.plot(epochs, val_loss3, "g", label="Модель 3. MSE на валидационной выборке")
plt.title("Ошибка MSE на обучающей и валидационной выборках для разных случаев")
plt.legend()
plt.xlabel('Эпоха')
plt.ylabel('Относительная ошибка')
plt.show()
```



9. Визуализируем весь набор данных и прогнозы трех построенных моделей для обучающей и тестовой выборок на одном рисунке:

```
ds_data = data['Adj Close']

look_back = 25
```

```
def prepare_for_plot(model):
  trainPredict = model.predict(train_dataset)
  valPredict = model.predict(val_dataset)
  testPredict = model.predict(test_dataset)

  point1 = len(trainPredict) + look_back
  point2 = point1 + len(valPredict) + look_back + 12

  trainPredictPlot = ds_data.copy()
  trainPredictPlot[:] = np.nan
  trainPredictPlot[look_back:point1] = trainPredict.reshape(-1)

  valPredictPlot = ds_data.copy()
  valPredictPlot[:] = np.nan
  valPredictPlot[point1 + look_back + 12: point2] = valPredict.reshape(-1)

  testPredictPlot = ds_data.copy()
  testPredictPlot[:] = np.nan
  testPredictPlot[point2 + look_back + 12:point2 + look_back + len(testPredict)

  return trainPredictPlot[:], valPredictPlot[:], testPredictPlot[:]
```

```
print(test_dataset)
```

```
    <_BatchDataset element_spec=(TensorSpec(shape=(None, None, 6), dtype=tf.flo
```

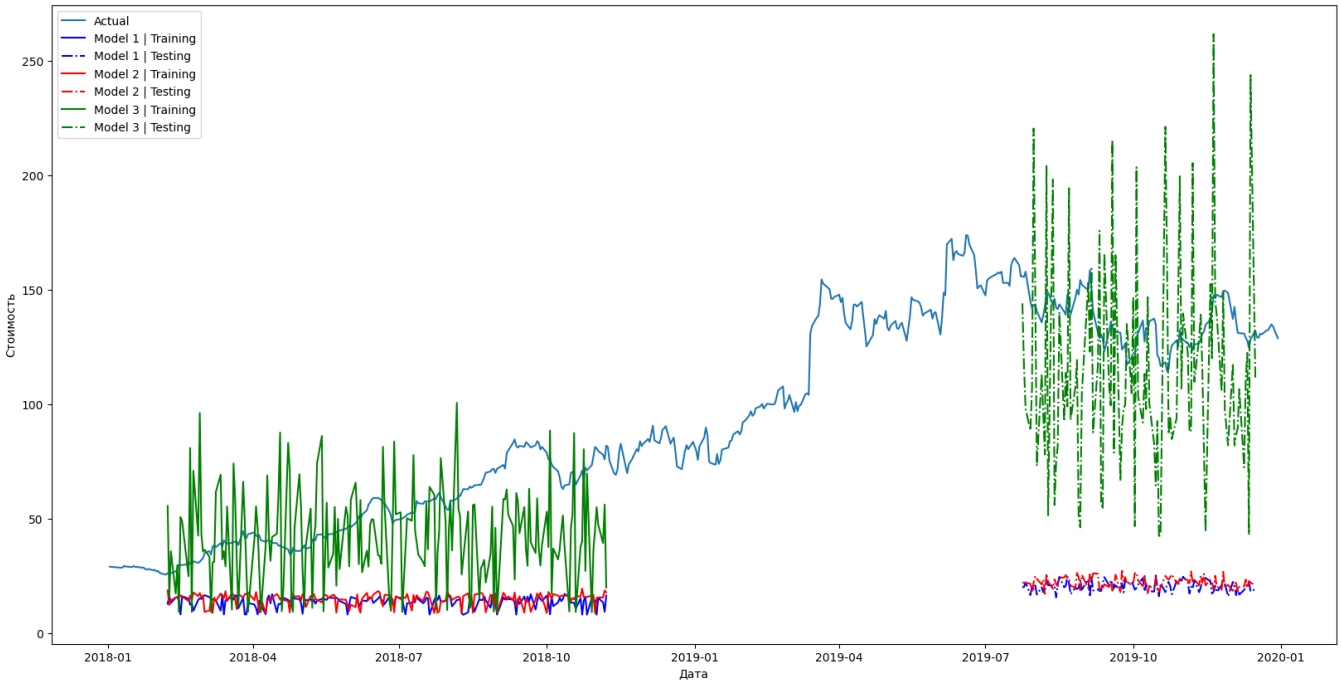
```
trainPredictPlot, valPredictPlot, testPredictPlot = prepare_for_plot(model1)

print("Длина trainPredictPlot:", len(trainPredictPlot))
print("Длина valPredictPlot:", len(valPredictPlot))
print("Длина testPredictPlot:", len(testPredictPlot))
```

```
    3/3 [==============================] - 0s 3ms/step
    2/2 [==============================] - 0s 5ms/step
    2/2 [==============================] - 0s 8ms/step
    Длина trainPredictPlot: 502
    Длина valPredictPlot: 502
    Длина testPredictPlot: 502
```

```
trainPredictPlot1, valPredictPlot1, testPredictPlot1 = prepare_for_plot(model1)
trainPredictPlot2, valPredictPlot2, testPredictPlot2 = prepare_for_plot(model2)
trainPredictPlot3, valPredictPlot3, testPredictPlot3 = prepare_for_plot(model3)
```

```
3/3 [==============================] - 0s 5ms/step
2/2 [==============================] - 0s 6ms/step
2/2 [==============================] - 0s 6ms/step
3/3 [==============================] - 0s 4ms/step
2/2 [==============================] - 0s 6ms/step
2/2 [==============================] - 0s 6ms/step
3/3 [==============================] - 0s 6ms/step
2/2 [==============================] - 0s 8ms/step
2/2 [==============================] - 0s 7ms/step
```

```
plt.figure(figsize=(20,10))
plt.plot(ds_data, label='Actual')
plt.plot(trainPredictPlot1, 'b', label='Model 1 | Training')
plt.plot(testPredictPlot1, 'b-.', label='Model 1 | Testing')
plt.plot(trainPredictPlot2, 'r', label='Model 2 | Training')
plt.plot(testPredictPlot2, 'r-.', label='Model 2 | Testing')
plt.plot(trainPredictPlot3, 'g', label='Model 3 | Training')
plt.plot(testPredictPlot3, 'g-.', label='Model 3 | Testing')
plt.xlabel('Дата')
plt.ylabel('Стоимость')
plt.legend();
```


Actual
Model 1 | Training
Model 1 | Testing
Model 2 | Training
Model 2 | Testing
Model 3 | Training
Model 3 | Testing
Стоимость
Дата
0
50
100
150
200
250
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01