Итоговый проект по курсу

«Введение в модели последовательных данных»

Predictive maintenance on pump sensor data

Состав группы:

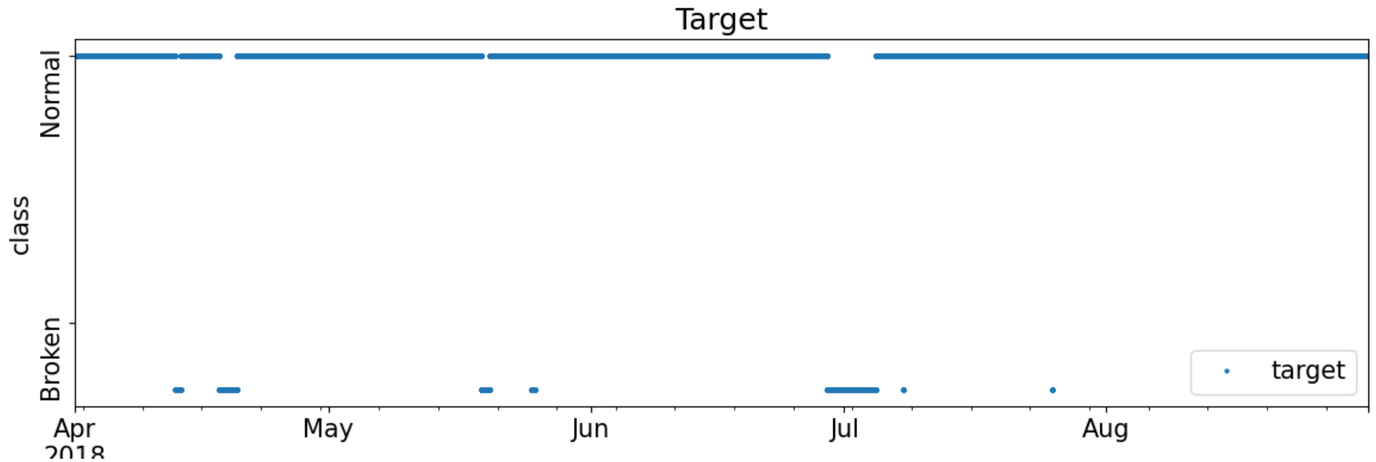
Николаев Олег
Калякин Роман
Кирячёк Владислав
Шарков Никита

20.12.2022

Описание датасета

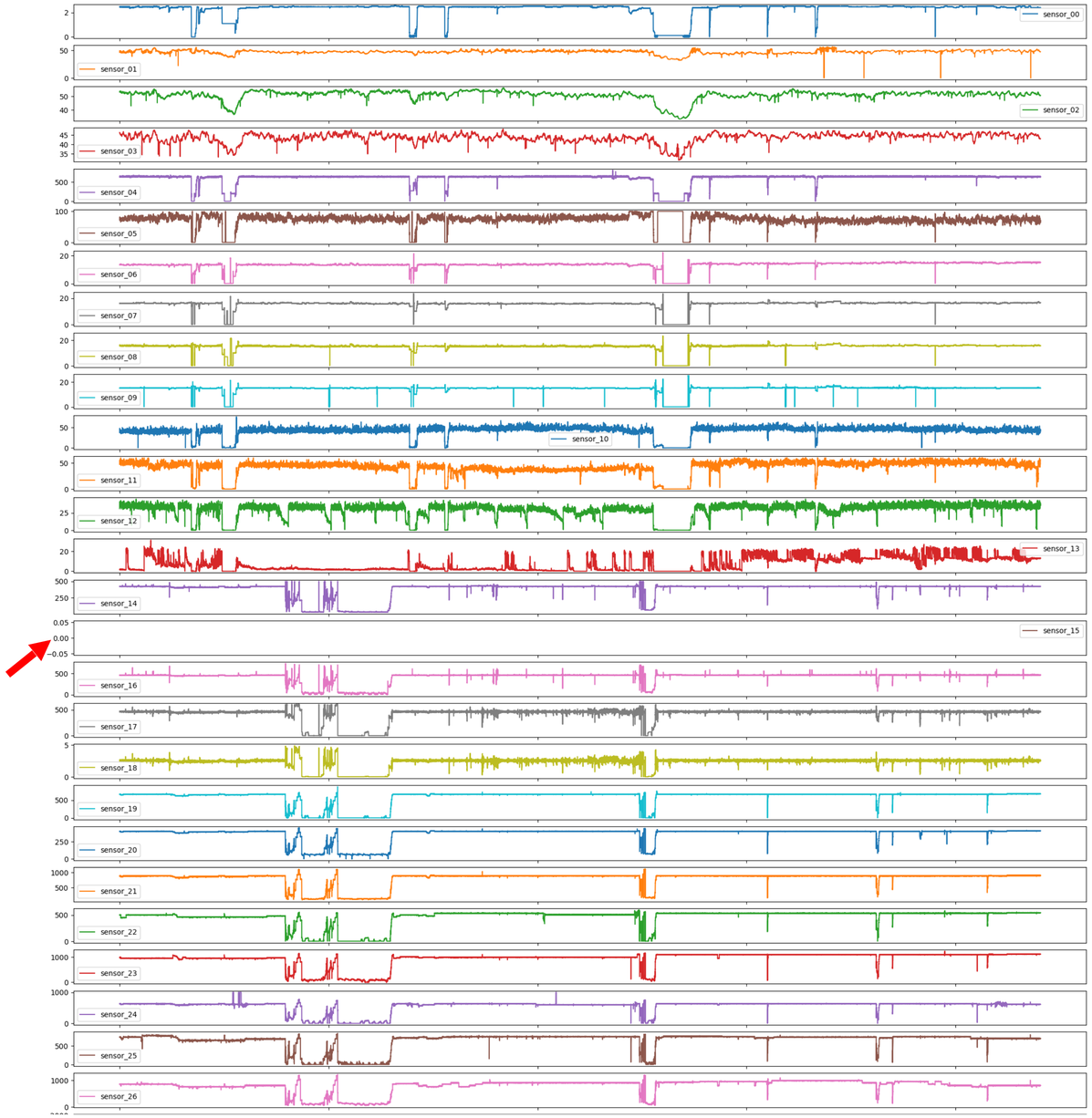
Задача профилактического обслуживания насоса (predictive maintenance)
Датасет: https://ga-data-cases.s3.eu-central-1.amazonaws.com/pump_sensor.zip
Размерность: 220320 x 55
Описание целевого столбца:

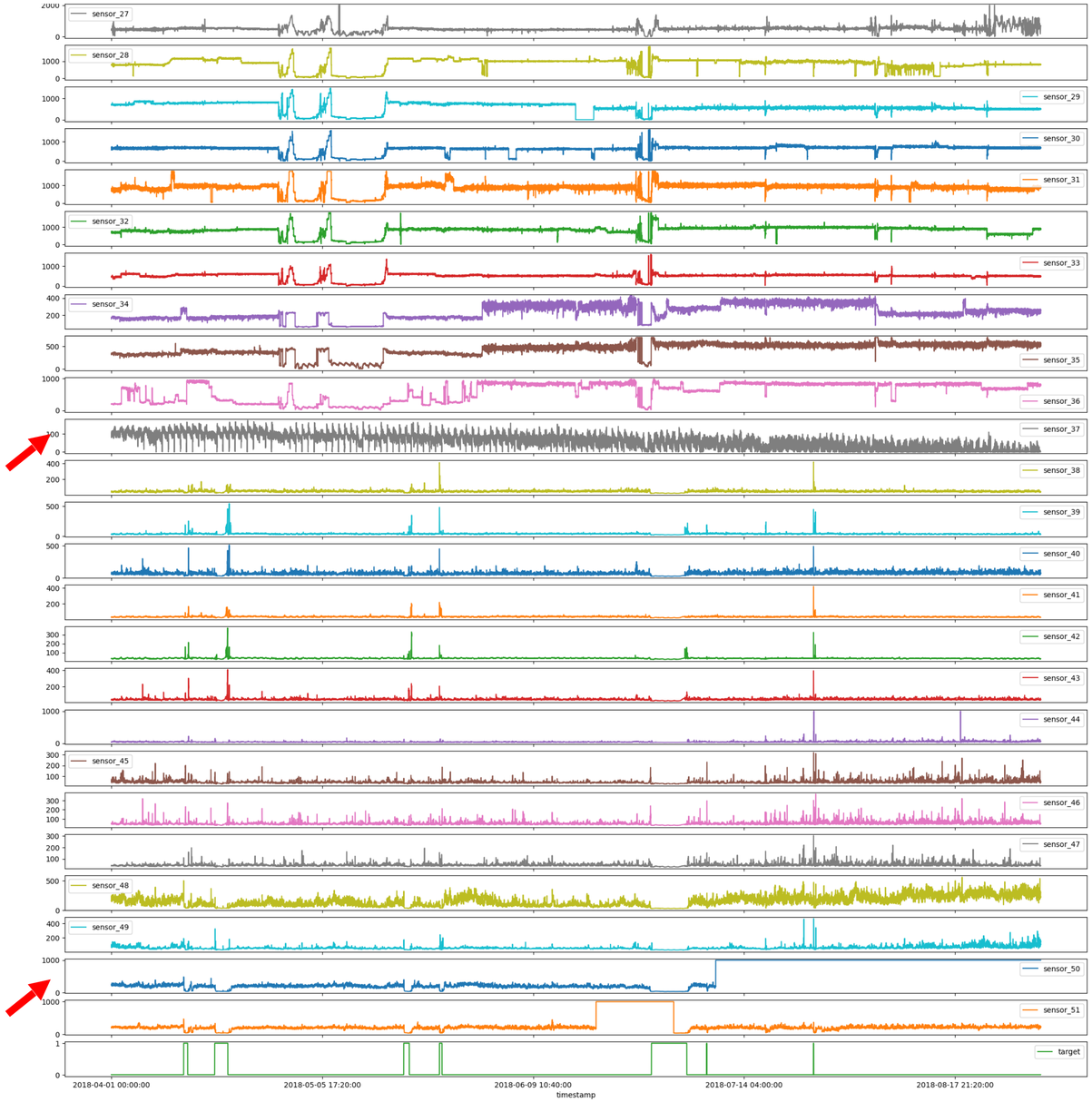
NORMAL	205836
BROKEN	14477



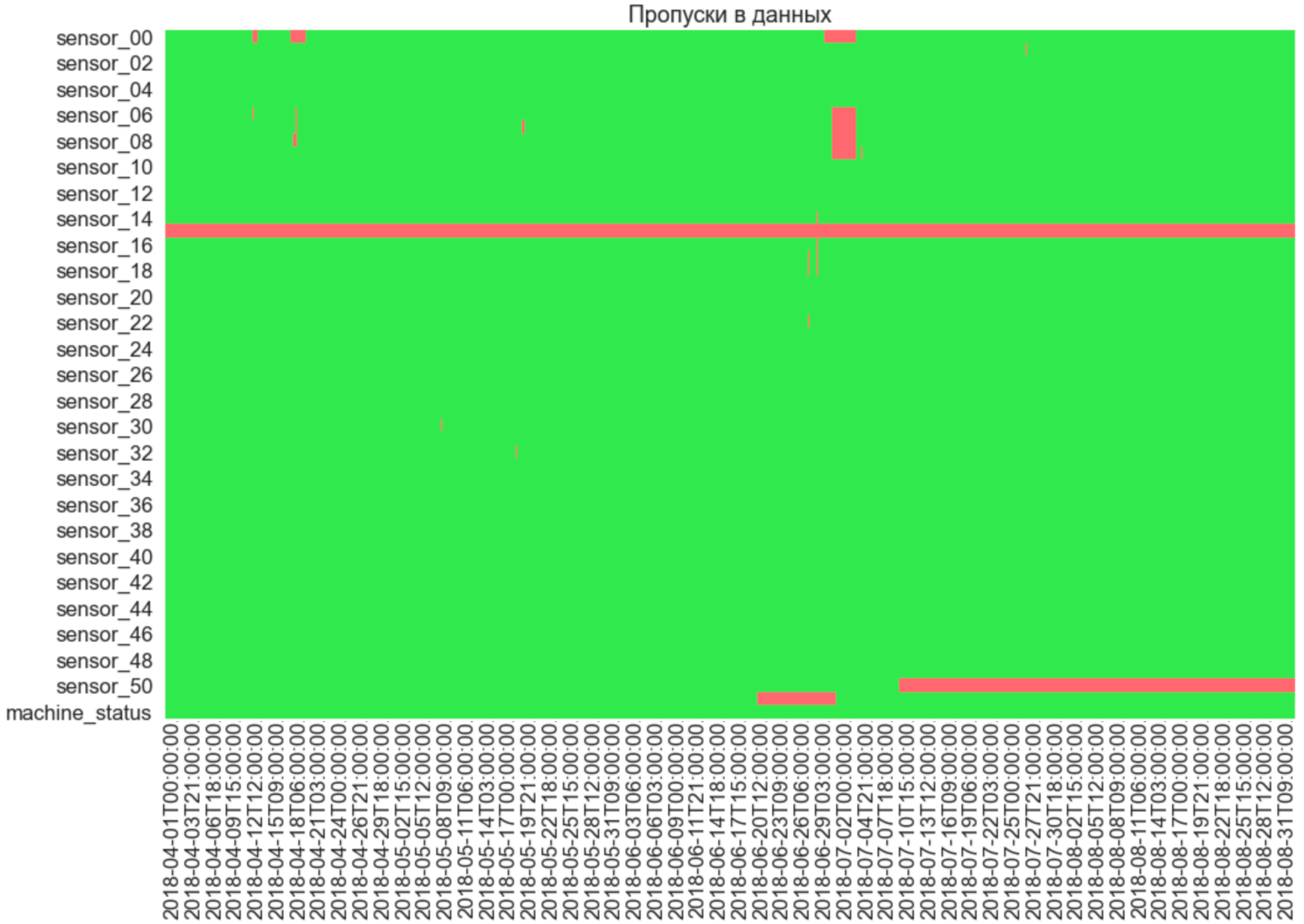
timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	sensor_07	sensor_08	sensor_09	...	sensor_44	sensor_45	sensor_46	sensor_47	sensor_48	sensor_49	sensor_50	sensor_51	machine_status
2018-04-01 00:00:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146	16.13136	15.56713	15.05353	...	39.641200	65.68287	50.92593	38.194440	157.9861	67.70834	243.0556	201.3889	NORMAL
2018-04-01 00:01:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146	16.13136	15.56713	15.05353	...	39.641200	65.68287	50.92593	38.194440	157.9861	67.70834	243.0556	201.3889	NORMAL
2018-04-01 00:02:00	2.444734	47.35243	53.2118	46.397570	638.8889	73.54598	13.32465	16.03733	15.61777	15.01013	...	39.351852	65.39352	51.21528	38.194443	155.9606	67.12963	241.3194	203.7037	NORMAL
2018-04-01 00:03:00	2.460474	47.09201	53.1684	46.397568	628.1250	76.98898	13.31742	16.24711	15.69734	15.08247	...	39.062500	64.81481	51.21528	38.194440	155.9606	66.84028	240.4514	203.1250	NORMAL
2018-04-01 00:04:00	2.445718	47.13541	53.2118	46.397568	636.4583	76.58897	13.35359	16.21094	15.69734	15.08247	...	38.773150	65.10416	51.79398	38.773150	158.2755	66.55093	242.1875	201.3889	NORMAL

Визуализация датасета

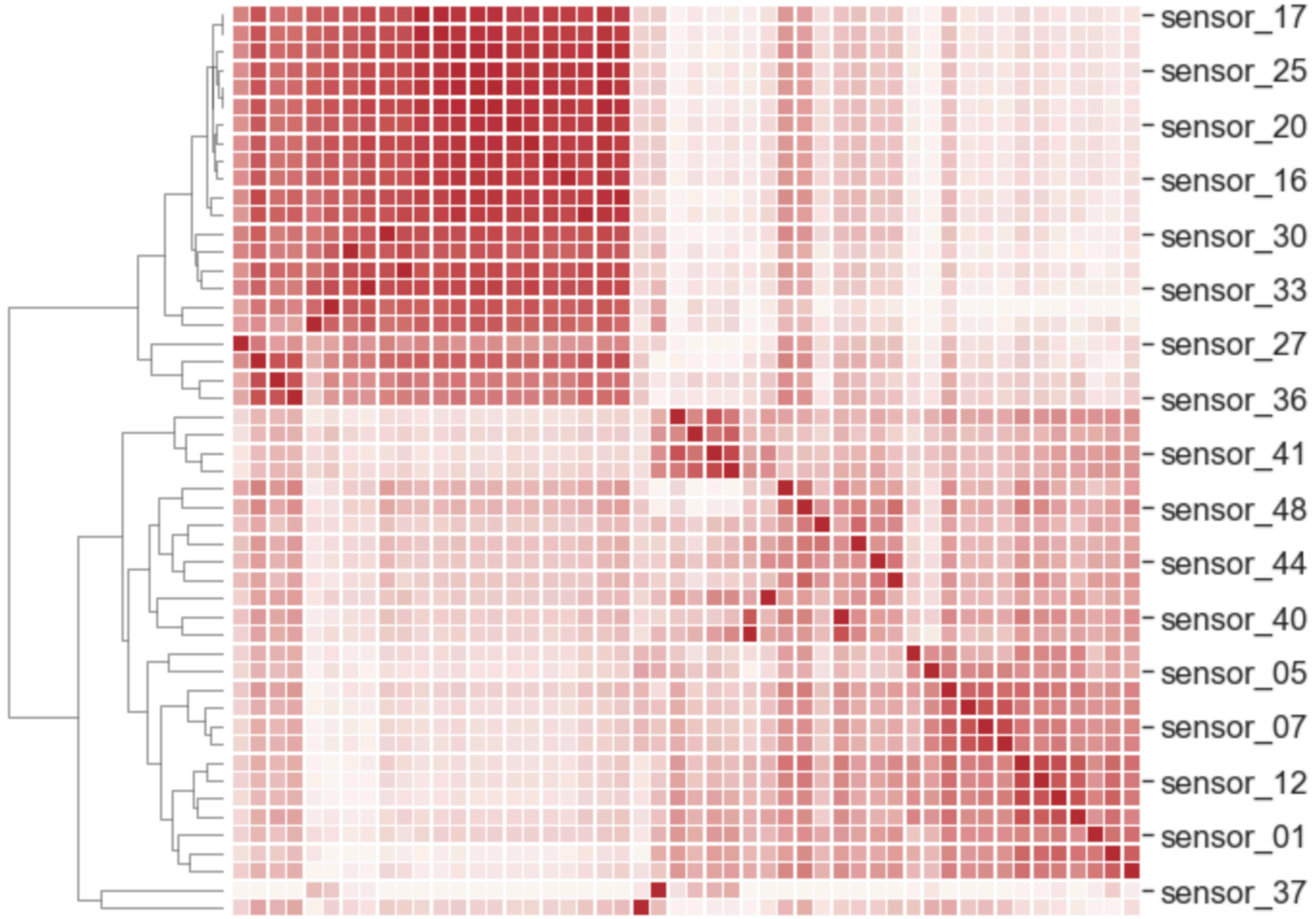


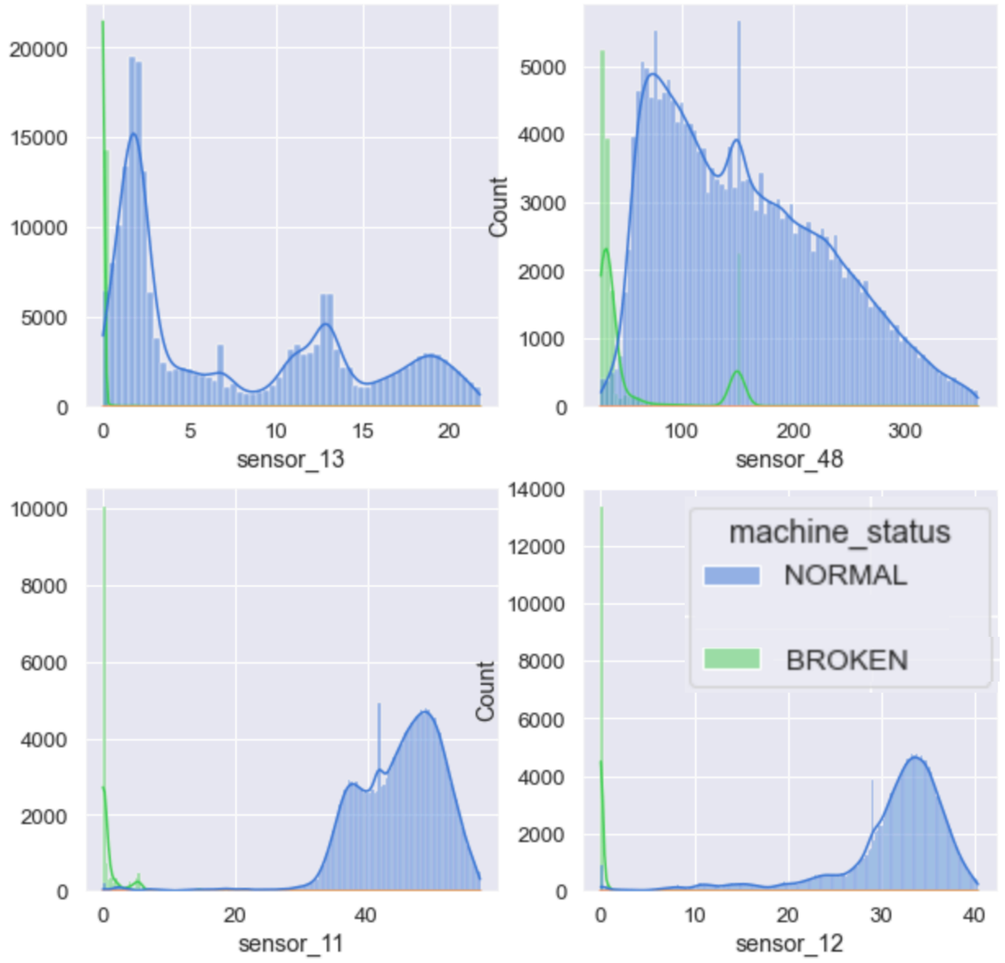


Обработка пропусков



Корреляции, значимые признаки



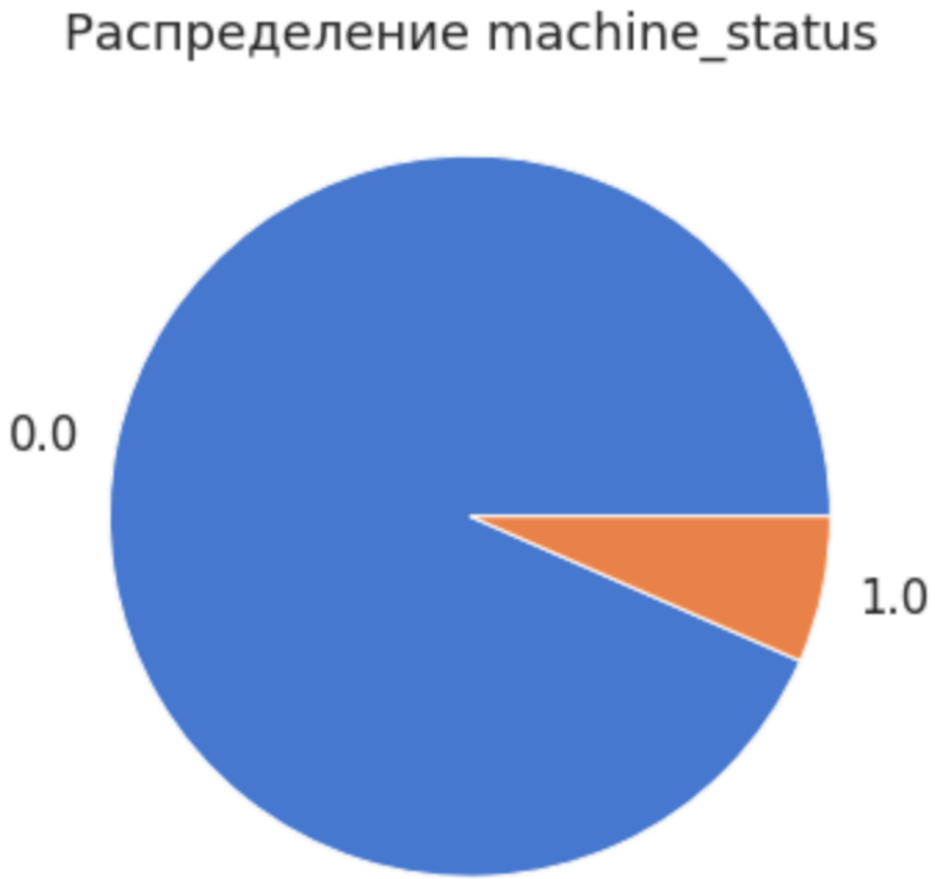


Метрики

- Binary crossentropy

```
model.compile(loss='binary_crossentropy', optimizer='adam')
```

```
class_weight = {0: 1.,
                1: 15.}
```



Подготовка данных

Основные шаги программы:

1)
```
data = data.fillna(method="ffill")
```

2)
```
data['target'] = data['machine_status'].map({'NORMAL': 0, 'BROKEN': 1, 'RECOVERING': 1})
```

3)
```
data = data.drop(labels=['sensor_15', 'sensor_37', 'sensor_50', 'machine_status'], axis=1)
```

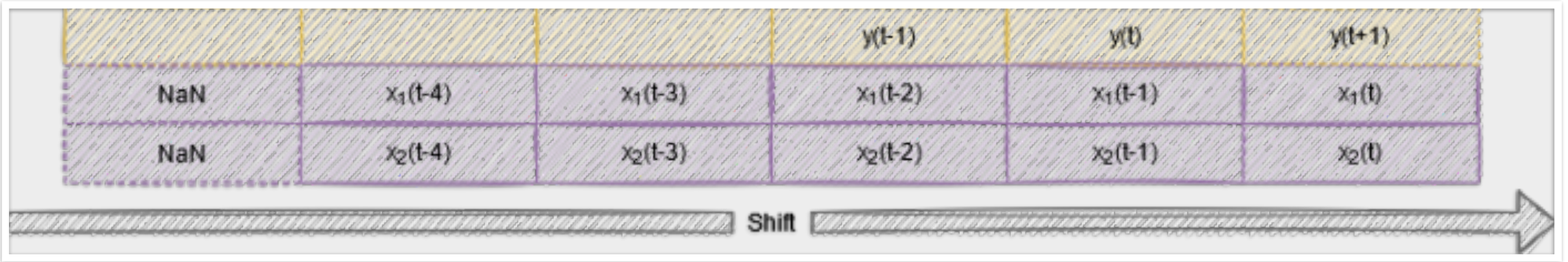
4)
```
data_scaled.iloc[:, :-1] = MinMaxScaler().fit_transform(data.iloc[:, :-1])
```

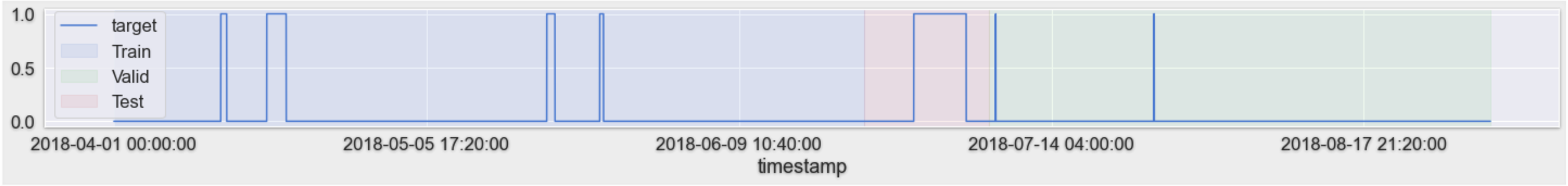
5)
```
data_shift.iloc[:, -1] = data_scaled.iloc[:, -1].shift(-Future)
```

`Future = 60` — параметр сдвига (см. рис. ниже)

Разбиение на обучающий, валидационный и тестовый датасеты:

```
X_train = data_shift.iloc[:120000, :-1].values
y_train = data_shift.iloc[:120000, -1].values
X_valid = data_shift.iloc[140000:, :-1].values
y_valid = data_shift.iloc[140000:, -1].values
X_test = data_shift.iloc[120000:140000, :-1].values
y_test = data_shift.iloc[120000:140000, -1].values
```





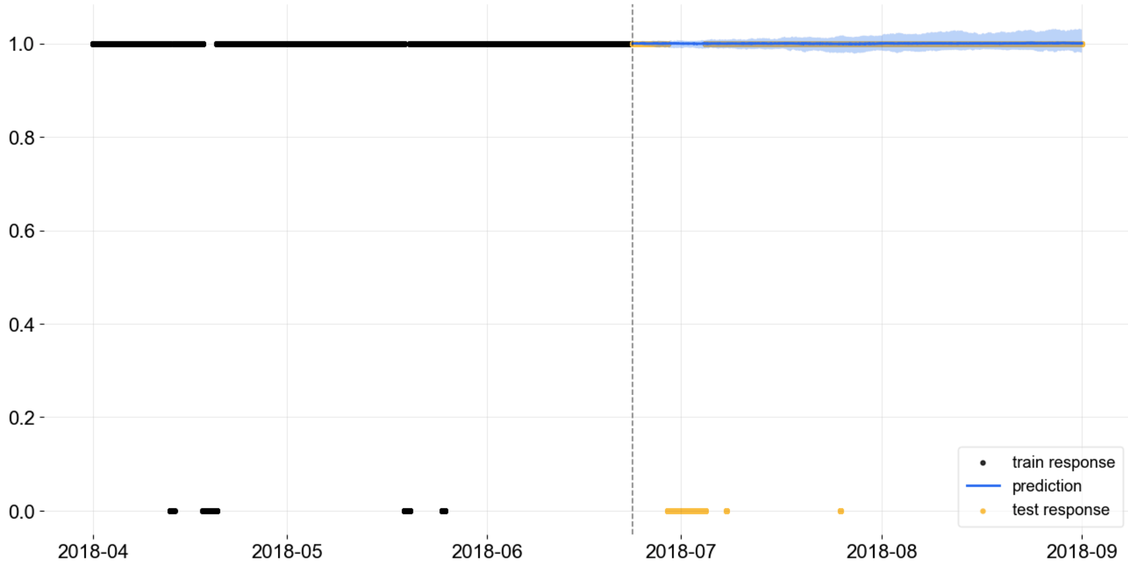
Модели

- ARIMA/Orbit
- RNN
- PCA + Transformer
- LSTM

Обучение Orbit

```python
from orbit.models.dlt import DLT
# инициализируем модель
model = DLT(
    response_col='delta',
    date_col='timestamp',
    regressor_col=list(train.columns[:-1]),
    global_trend_option = 'flat',
    seed=1,
)

# обучаем модель
model.fit(df=train.reset_index())
# делаем прогноз
y_train_pred = model.predict(df=train.iloc[:, n:].reset_index())
y_test_pred = model.predict(df=test.iloc[:, n:].reset_index())
```

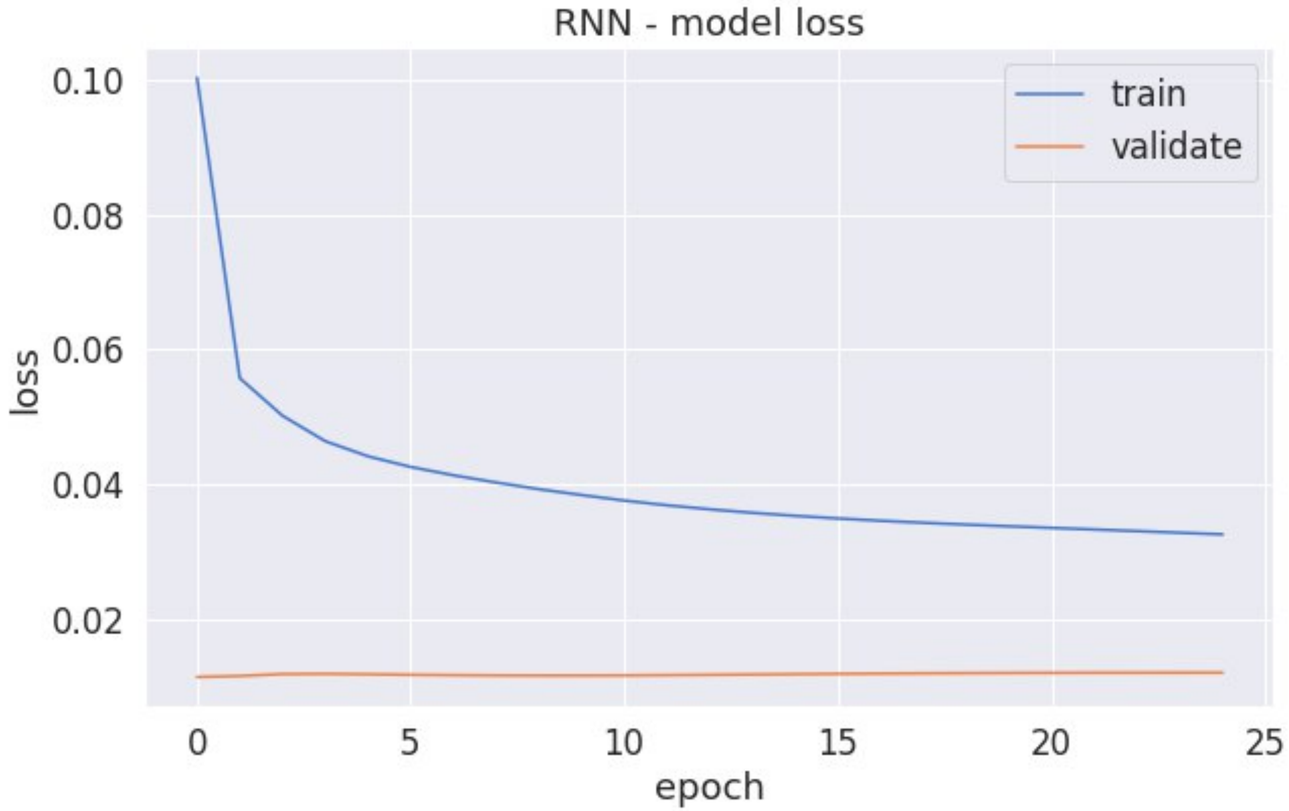
Test



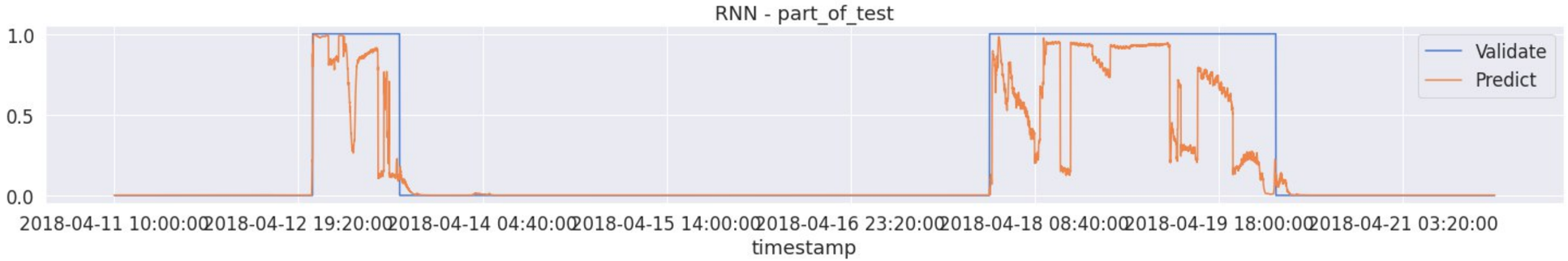
Обучение RNN

```
x = SimpleRNN(16, input_shape=(in_shape[1], in_shape[2]))(inputs)
out = Dense(1, activation='sigmoid')(x)

model = tf.keras.Model(inputs=inputs, outputs=out)

model.compile(loss='binary_crossentropy',
              optimizer='adam')
```

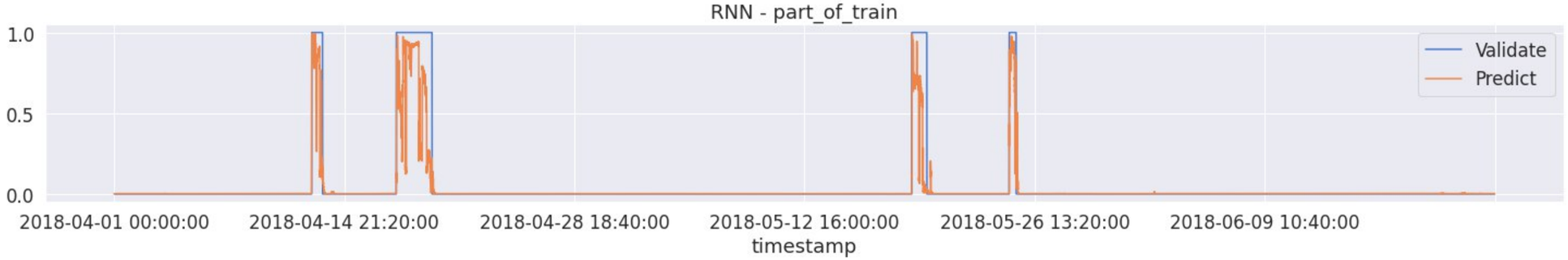
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 1, 49)]           0

 simple_rnn_5 (SimpleRNN)    (None, 16)                1056

 dense_6 (Dense)             (None, 1)                 17

=================================================================
Total params: 1,073
Trainable params: 1,073
Non-trainable params: 0
```



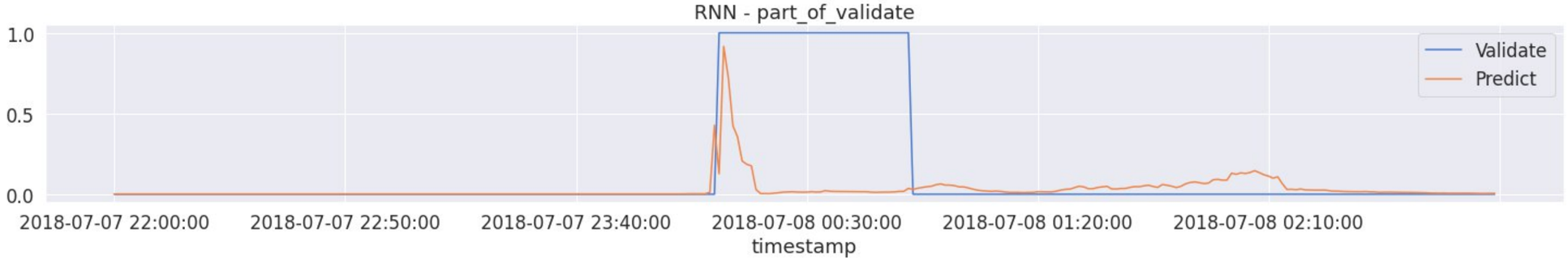
Train



Validate

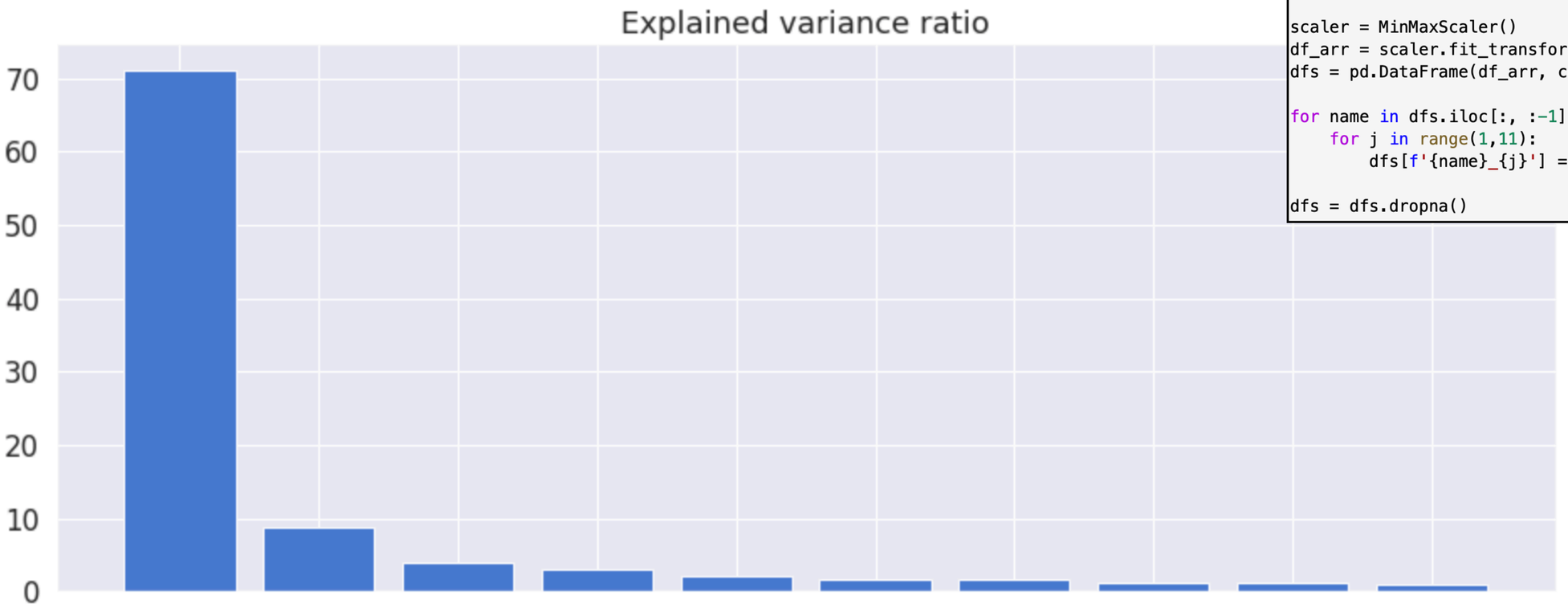


Test



Разложение PCA

```
from sklearn.decomposition import PCA
pca_transformer = PCA()
X2 = pca_transformer.fit_transform(df)

df_pca = pd.DataFrame(X2, columns = [f'PCA_{i}' for i in range(51)], index = df.index)
```



	PCA_0	PCA_1	PCA_2	PCA_3	PCA_4	PCA_5	PCA_6	PCA_7	PCA_8	PCA_9	PCA_10	PCA_11	PCA_12	PCA_13
0	0.711572	0.088474	0.039499	0.031585	0.02181	0.017528	0.015987	0.012595	0.011598	0.0093	0.007112	0.006404	0.0053	0.004052

Итоговые признаки

```
df_pca = df_pca.iloc[:, :10]
df_pca['machine_status'] = df['machine_status']

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df_arr = scaler.fit_transform(df_pca)
dfs = pd.DataFrame(df_arr, columns = df_pca.columns, index = df_pca.index)

for name in dfs.iloc[:, :-1]:
    for j in range(1,11):
        dfs[f'{name}_{j}'] = dfs[name].shift(j)

dfs = dfs.dropna()
```

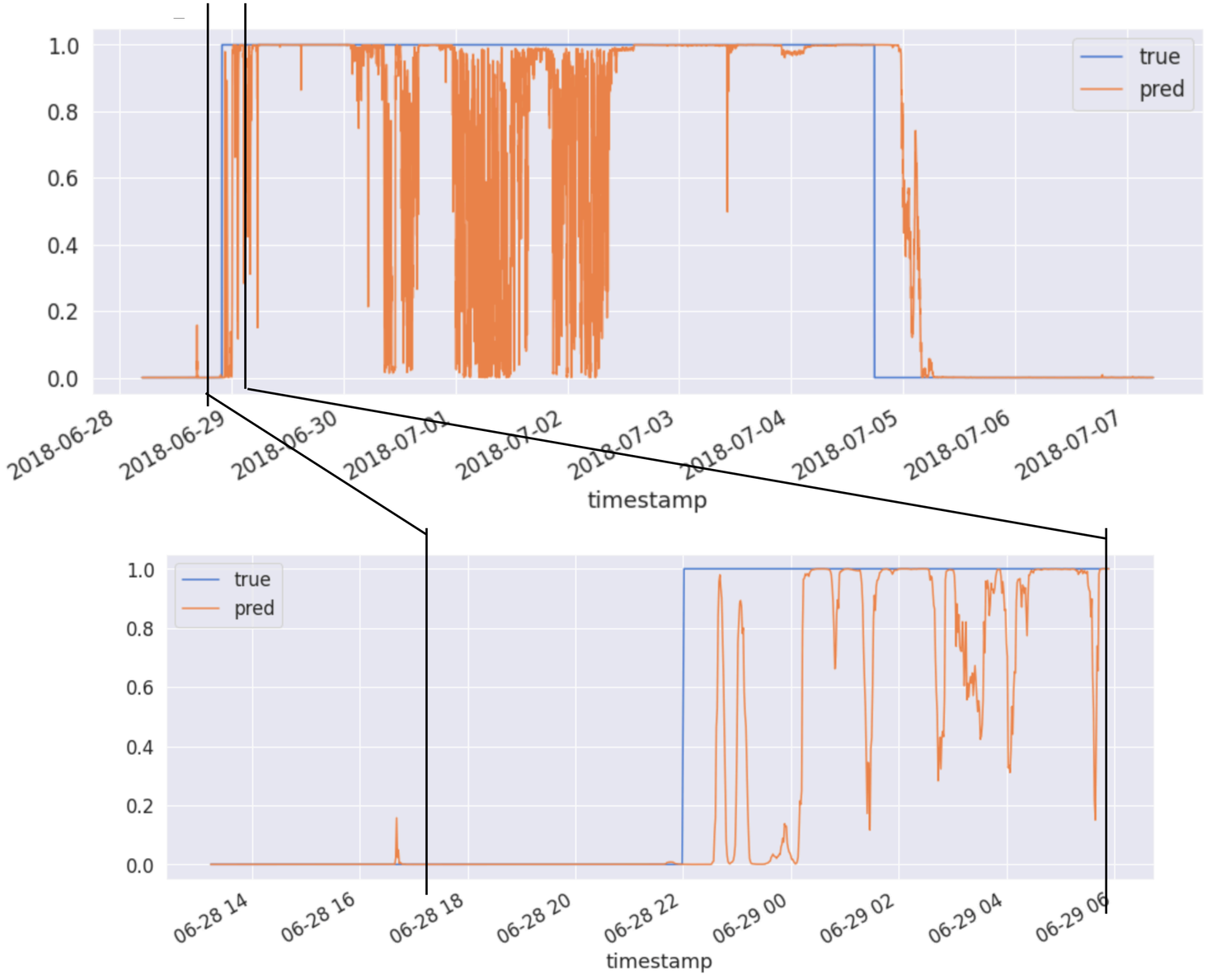
Обучение Transformer

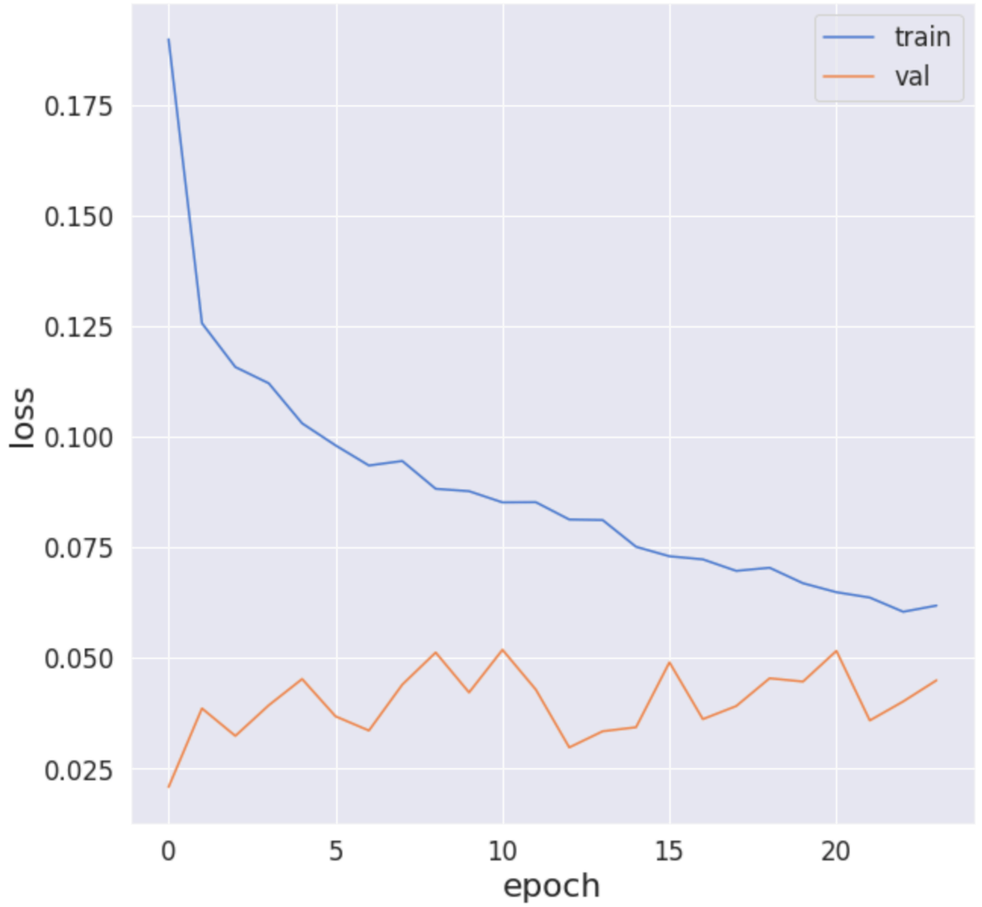
```
model = build_model(
    input_shape,
    head_size=32,
    num_heads=4,
    ff_dim=4,
    num_transformer_blocks=1,
    mlp_units=[128],
    mlp_dropout=0.4,
    dropout=0.25,
)

model.compile(
    loss="sparse_categorical_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=1e-3),
    metrics=["sparse_categorical_accuracy"],
)
model.summary()

history = model.fit(
    x_train,
    y_train,
    validation_split=0.2,
    epochs=10,
    batch_size=256,
    class_weight=class_weight
)
```

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 110, 1)]	0	[]
multi_head_attention_2 (MultiHeadAttention)	(None, 110, 1)	897	['input_3[0][0]', 'input_3[0][0]']
dropout_6 (Dropout)	(None, 110, 1)	0	['multi_head_attention_2[0][0]']
tf.__operators__.add_4 (TFOpLambda)	(None, 110, 1)	0	['dropout_6[0][0]', 'input_3[0][0]']
conv1d_4 (Conv1D)	(None, 110, 4)	8	['tf.__operators__.add_4[0][0]']
dropout_7 (Dropout)	(None, 110, 4)	0	['conv1d_4[0][0]']
conv1d_5 (Conv1D)	(None, 110, 1)	5	['dropout_7[0][0]']
layer_normalization_5 (LayerNormalization)	(None, 110, 1)	2	['conv1d_5[0][0]']
tf.__operators__.add_5 (TFOpLambda)	(None, 110, 1)	0	['layer_normalization_5[0][0]', 'tf.__operators__.add_4[0][0]']
global_average_pooling1d_2 (GlobalAveragePooling1D)	(None, 110)	0	['tf.__operators__.add_5[0][0]']
dense_4 (Dense)	(None, 64)	7104	['global_average_pooling1d_2[0][0]']
dropout_8 (Dropout)	(None, 64)	0	['dense_4[0][0]']
dense_5 (Dense)	(None, 2)	130	['dropout_8[0][0]']

Total params: 8,146

Предсказание Transformer





Обучение LSTM

Описание модели:

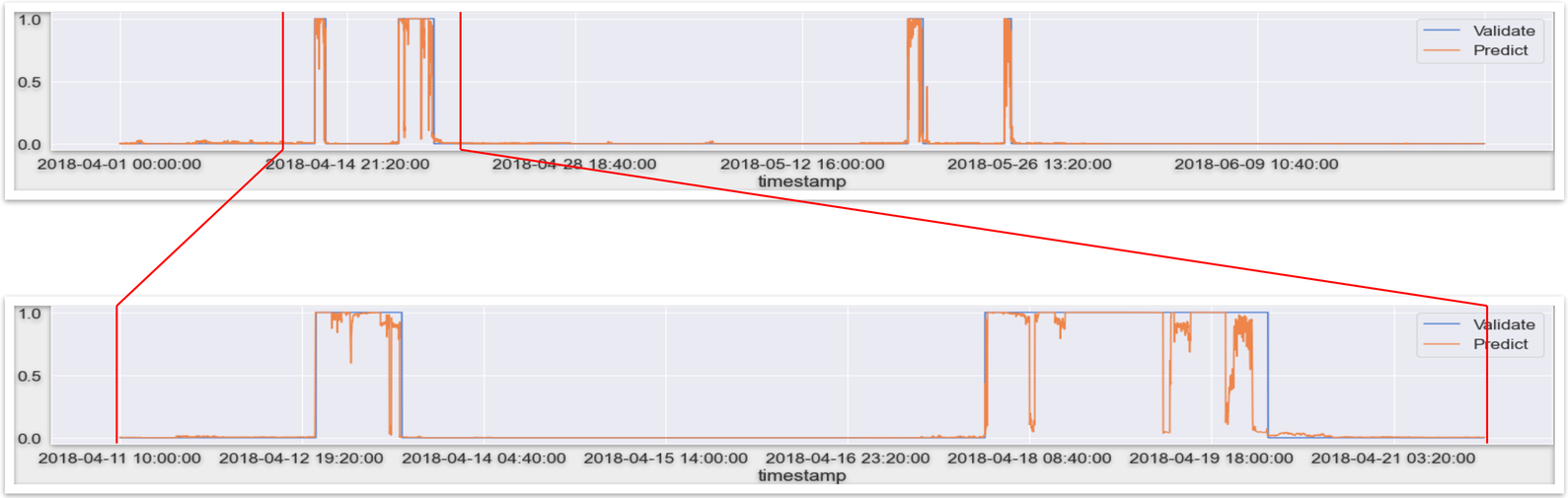
```
inputs= tf.keras.Input(shape=(in_shape[1], in_shape[2]))
x = LSTM(50, activation='relu', input_shape=(in_shape[1], in_shape[2]), return_sequences=True)(inputs)
x = LSTM(50, activation='relu')(x)
out = Dense(1, activation='sigmoid')(x)

model = tf.keras.Model(inputs=inputs, outputs=out)

model.compile(loss='binary_crossentropy', optimizer='adam')
```

Обучение модели:

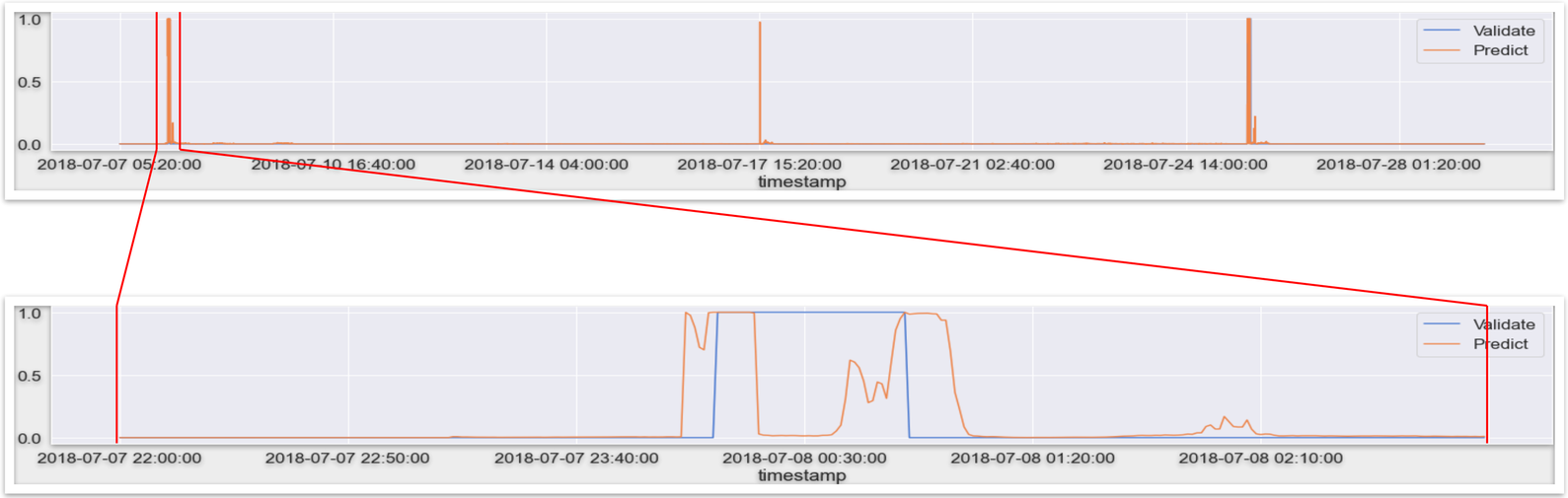
```
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_valid, y_valid), shuffle=False)
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 1, 49)]           0

 lstm (LSTM)                 (None, 1, 50)             20000

 lstm_1 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 40,251
Trainable params: 40,251
Non-trainable params: 0
```

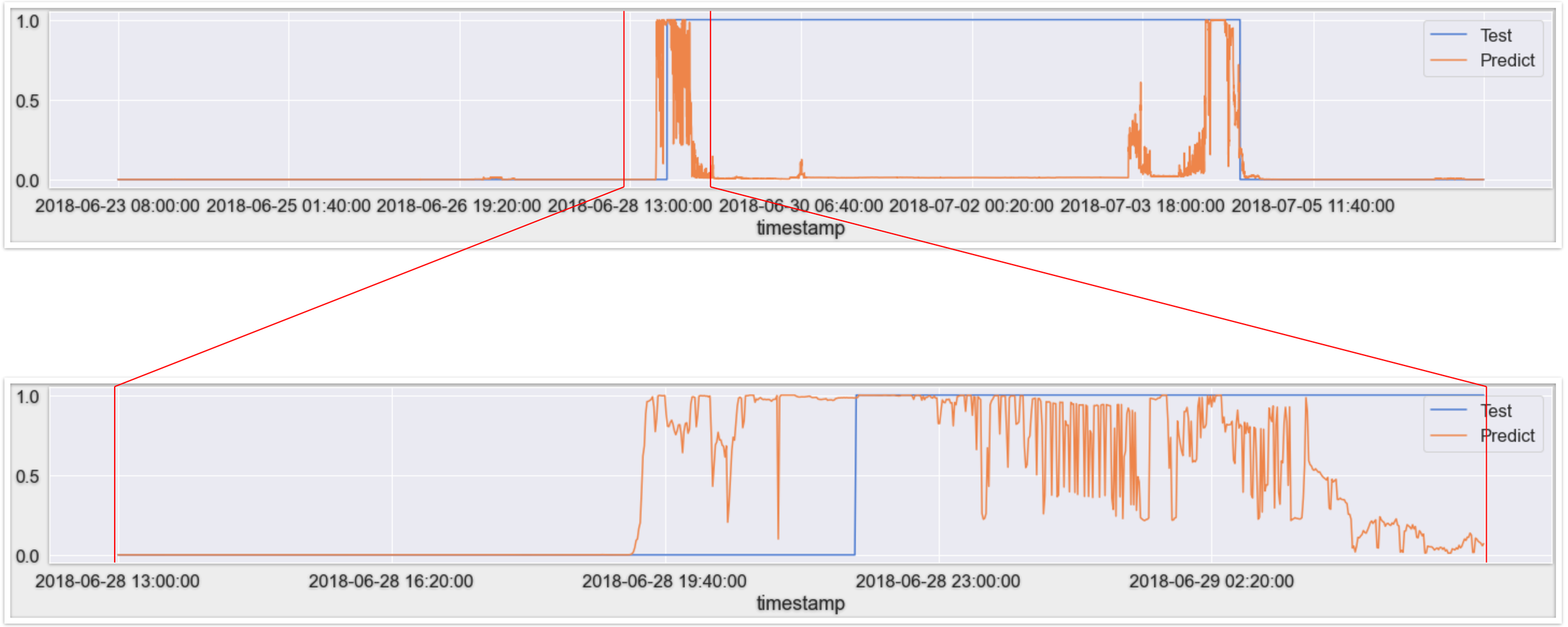
Train



Validate



Предсказание LSTM



Заключение

1. Архитектура сети LSTM показала наилучший результат
2. Сдвиг таргета позволяет предсказать поломку насоса заранее (3 часа)
3. Время обучения LSTM(100 эпох) составило 11 минут.