Fakultet inženjerskih nauka
Univerziteta u Kragujevcu

Tema:

Predviđanje prognoze vremena na osnovu vremenskih serija pomoću LSTM modela

student:
Nikola Mitrevski
400/2021

predmetni profesor:
Vesna Ranković
predmetni asistent:
Tijana Šušteršič

Kragujevac 2021.

Sadržaj:

1 Uvod

Vremenska serija je niz tačaka podataka indeksiranih vremenskim redosledom. Najčešće vremenska serija je niz snimljen u uzastopnim jednako raspoređenim tačkama u vremenu. Podaci o vremenskim serijama su jednostavno merenja ili događaji koji se prate tokom vremena.

U ovom radu biće implementiran primer za predviđanje prognoze vremena na osnovu vremenskih serija pomoću LSTM modela.

2 Uvoz potrebnih biblioteka

```
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
```

3 Uvoz skupa podataka

U ovom radu je korišćen skup podataka o klimi u Jeni(Nemačka), koji je snimio institut Max Planck u periodu od 10.januara.2009 do 31.decembra.2016 i sastoji se od 14 karakteristika(tabela 1). Podaci iz ovog dataset-a su prikupljani na svakih 10 minuta.

Tabela 1 Jena Climate dataset karakteristike

Indeks	Karakteristike	Format	Objašnjenje
1	Date Time	01.01.2009 00:10:00	Referenca datum-vreme
2	p (mbar)	996.52	Jedinica za pritisak izvedena u pascal SI koja se koristi za kvantifikaciju unutrašnjeg pritiska.
3	T (degC)	-8.02	Temperatura u Celzijusima
4	Tpot (K)	265.4	Temperatura u Kelvinima
5	Tdew (degC)	-8.9	Temperatura u Celzijusima u odnosu na vlažnost.
6	rh (%)	93.3	Relativna vlažnost je mera koliko je vazduh zasićen vodenom parom. % određuje količinu vode sadržanu u objektima za prikupljanje.
7	VPmax (mbar)	3.33	Zasićenje pritiska pare
8	VPact (mbar)	3.11	Pritisak pare
9	VPdef (mbar)	0.22	Deficit pritiska pare
10	sh (g/kg)	1.94	Specifična vlažnost
11	H2OC (mmol/mol)	3.12	Koncentracija vodene pare
12	rho (g/m ** 3)	1307.75	Gustina vazduha
13	wv (m/s)	1.03	Brzina vetra
14	max. wv (m/s)	1.75	Maksimalna brzina vetra
15	wd (deg)	152.3	Pravac vetra u stepenima

```
df = pd.read_csv('../binder/jena_climate_2009_2016.csv')
```

4 Definisanje naziva karakteristika i inicijalizacija parametara za obuku modela

```
titles = [
    "Pressure",
    "Temperature",
    "Temperature in Kelvin",
    "Temperature (dew point)",
    "Relative Humidity",
    "Saturation vapor pressure",
    "Vapor pressure",
    "Vapor pressure deficit",
    "Specific humidity",
    "Water vapor concentration",
    "Airtight",
    "Wind speed",
    "Maximum wind speed",
    "Wind direction in degrees",
]

feature_keys = [
    "p (mbar)",
    "T (degC)",
    "Tpot (K)",
    "Tdew (degC)",
    "rh (%)",
    "VPmax (mbar)",
    "VPact (mbar)",
    "VPdef (mbar)",
    "sh (g/kg)",
    "H2OC (mmol/mol)",
    "rho (g/m**3)",
    "wv (m/s)",
    "max. wv (m/s)",
    "wd (deg)",
]

date_time_key = "Date Time"

split_fraction = 0.715
train_split = int(split_fraction * int(df.shape[0]))
step = 6

past = 720
future = 72
learning_rate = 0.001
batch_size = 256
epochs = 10
```

5 Preprocesiranje podataka

Pošto svaka karakteristika ima vrednosti iz različitog opsega, vrši se normalizacija podataka čime se vrednosti karakteristika ograničivaju na opseg od 0 do 1, da bi se dobili najtačniji rezultati.

```
def normalize(data, train_split):
    data_mean = data[:train_split].mean(axis=0)
    data_std = data[:train_split].std(axis=0)
    return (data - data_mean) / data_std
```

Zbog suvišnosti nekih karakteristika, izdvojićemo korisne(Pressure, Temperature, Saturation vapor pressure, Vapor pressure deficit, Specific humidity, Airtight, Wind speed) u poseban skup podataka.

```
selected_features = [feature_keys[i] for i in [0, 1, 5, 7, 8, 10, 11]]
features = df[selected_features]
features.index = df[date_time_key]
features.head()

features = normalize(features.values, train_split)
features = pd.DataFrame(features)
features.head()

train_data = features.loc[0 : train_split - 1]
val_data = features.loc[train_split:]
```

6 Formiranje skupa podataka za obuku

```
start = past + future
end = start + train_split

x_train = train_data[[i for i in range(7)]].values
y_train = features.iloc[start:end][[1]]

sequence_length = int(past / step)

dataset_train = keras.preprocessing.timeseries_dataset_from_array(
    x_train,
    y_train,
    sequence_length=sequence_length,
    sampling_rate=step,
    batch_size=batch_size,
)
```

Funkcija “timeseries_dataset_from_array” uzima niz tačaka podataka prikupljenih u jednakim intervalima, zajedno sa parametrima vremenske serije(dužina niza, razmak između dva niza, itd.) da bi proizvela serije ulaza i željenih izlaza.

7 Formiranje skupa podataka za validaciju

```
x_end = len(val_data) - past - future

label_start = train_split + past + future

x_val = val_data.iloc[:x_end][[i for i in range(7)]].values
y_val = features.iloc[label_start:][[1]]

dataset_val = keras.preprocessing.timeseries_dataset_from_array(
    x_val,
    y_val,
    sequence_length=sequence_length,
    sampling_rate=step,
    batch_size=batch_size,
)
```

8 Definisanje modela

```
for batch in dataset_train.take(1):
    inputs, targets = batch

inputs = keras.layers.Input(shape=(inputs.shape[1], inputs.shape[2]))
lstm_out = keras.layers.LSTM(32)(inputs)
outputs = keras.layers.Dense(1)(lstm_out)

model = keras.Model(inputs=inputs, outputs=outputs)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=learning_rate), loss="mse")
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 120, 7)]          0
_________________________________________________________________
lstm (LSTM)                  (None, 32)                5120
_________________________________________________________________
dense (Dense)                (None, 1)                 33
=================================================================
Total params: 5,153
Trainable params: 5,153
Non-trainable params: 0
_________________________________________________________________
```

Slika 1 Prikaz informacija strukture definisanog LSTM modela

9 Obuka modela

```
path_checkpoint = "model_checkpoint.h5"
es_callback = keras.callbacks.EarlyStopping(monitor="val_loss", min_delta=0, patience=5)

modelckpt_callback = keras.callbacks.ModelCheckpoint(
    monitor="val_loss",
    filepath=path_checkpoint,
    verbose=1,
    save_weights_only=True,
    save_best_only=True,
)

history = model.fit(
    dataset_train,
    epochs=epochs,
    validation_data=dataset_val,
    callbacks=[es_callback, modelckpt_callback],
)
```

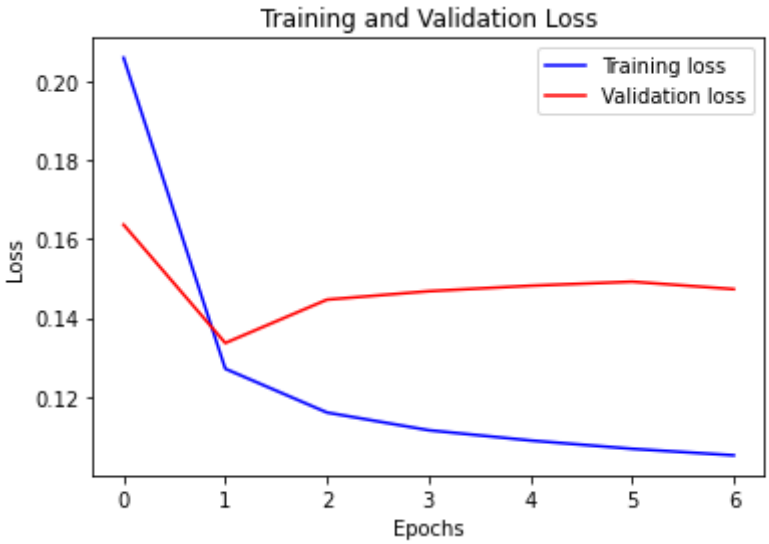
Funkcija ModelCheckpoint se koristi za redovno čuvanje kontrolnih tačaka, a funkcija EarlyStopping za prekidanje obuke modela, kada se gubitak modela više ne poboljšava.

```
Epoch 1/10
1172/1172 [==============================] - ETA: 0s - loss: 0.2059
Epoch 00001: val_loss improved from inf to 0.16357, saving model to model_checkpoint.h5
1172/1172 [==============================] - 101s 86ms/step - loss: 0.2059 - val_loss: 0.1636
Epoch 2/10
1172/1172 [==============================] - ETA: 0s - loss: 0.1271
Epoch 00002: val_loss improved from 0.16357 to 0.13362, saving model to model_checkpoint.h5
1172/1172 [==============================] - 107s 92ms/step - loss: 0.1271 - val_loss: 0.1336
Epoch 3/10
1172/1172 [==============================] - ETA: 0s - loss: 0.1089
Epoch 00005: val_loss did not improve from 0.13362
1172/1172 [==============================] - 110s 94ms/step - loss: 0.1089 - val_loss: 0.1481
Epoch 6/10
 271/1172 [=====>........................] - ETA: 1:12 - loss: 0.1117
```

Slika 2 Obuka LSTM modela (trajanje 6 epoha)

10 Vizuelizacija gubitka modela

```
def visualize_loss(history, title):
    loss = history.history["loss"]
    val_loss = history.history["val_loss"]
    epochs = range(len(loss))
    plt.figure()
    plt.plot(epochs, loss, "b", label="Training loss")
    plt.plot(epochs, val_loss, "r", label="Validation loss")
    plt.title(title)
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.legend()
    plt.show()

visualize_loss(history, "Training and Validation Loss")
```



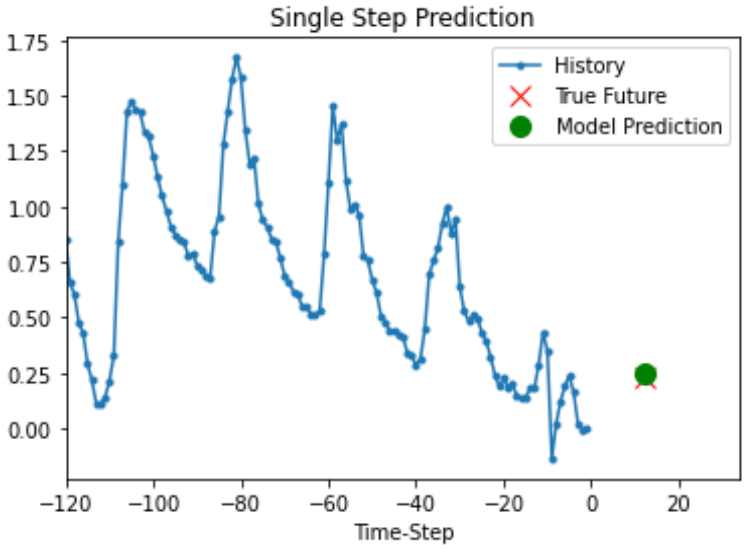
Slika 3 Vizuelizacija gubitka LSTM modela

Sa slike se vidi da posle prve epohe, gubitak prestaje da se smanjuje.

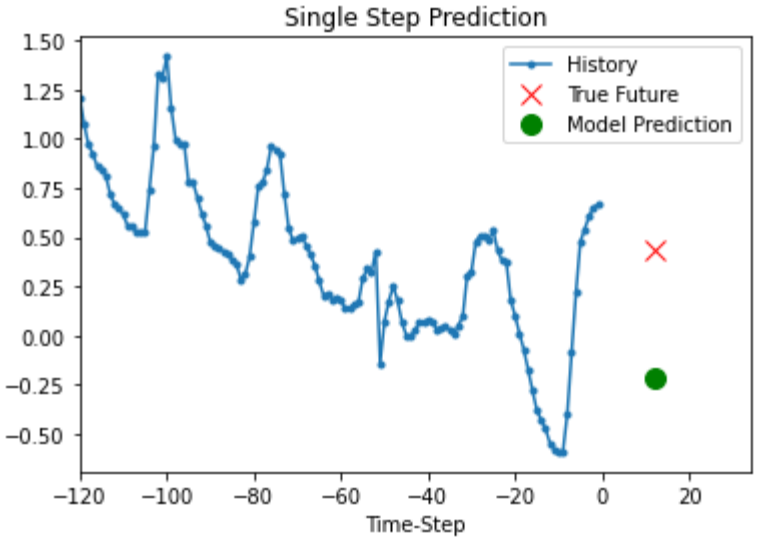
11 Vizualizacija predviđanja(validacije) modela

Predviđanje modela za 5 skupova vrednosti uzetih iz skupa za validaciju.

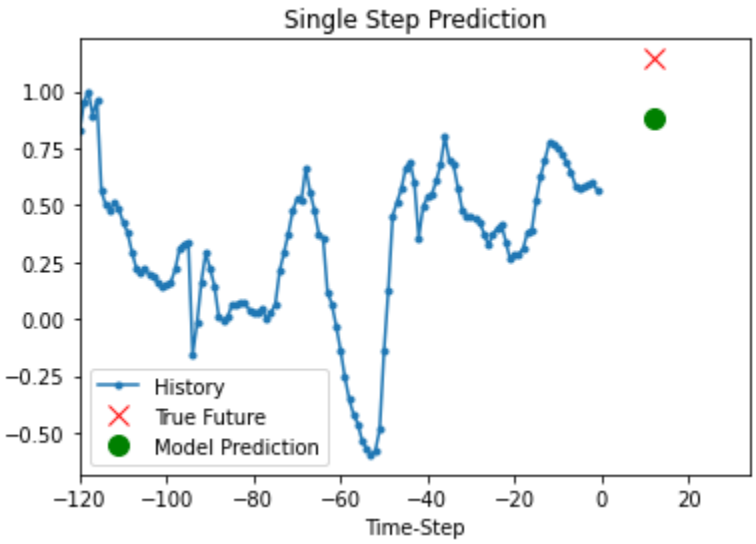
```
def show_plot(plot_data, delta, title):
    labels = ["History", "True Future", "Model Prediction"]
    marker = [".-", "rx", "go"]
    time_steps = list(range(-(plot_data[0].shape[0]), 0))
    if delta:
        future = delta
    else:
        future = 0

    plt.title(title)
    for i, val in enumerate(plot_data):
        if i:
            plt.plot(future, plot_data[i], marker[i], markersize=10, label=labels[i])
        else:
            plt.plot(time_steps, plot_data[i].flatten(), marker[i], label=labels[i])
    plt.legend()
    plt.xlim([time_steps[0], (future + 5) * 2])
    plt.xlabel("Time-Step")
    plt.show()
    return


for x, y in dataset_val.take(5):
    show_plot(
        [x[0][:, 1].numpy(), y[0].numpy(), model.predict(x)[0]],
        12,
        "Single Step Prediction",
    )
```



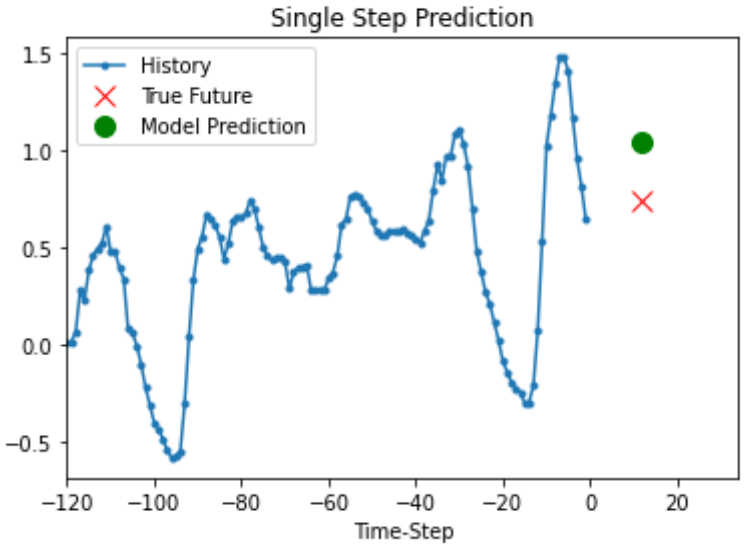
Slika 4 Grafički prikaz validacije LSTM modela za prvi skup podataka



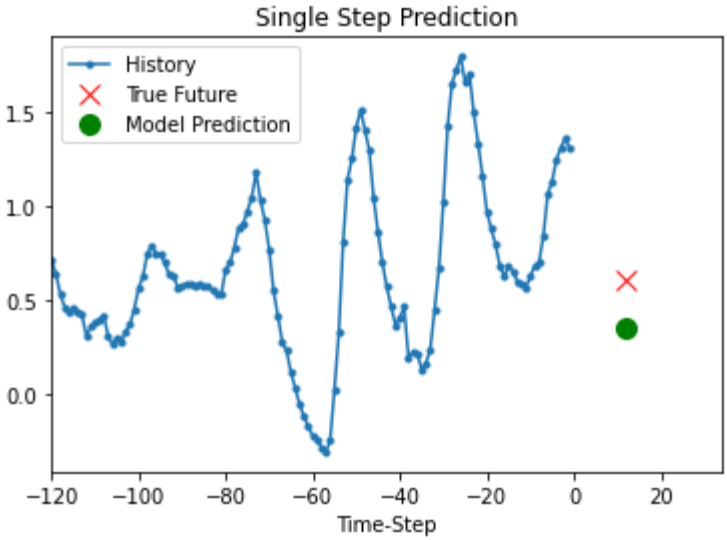
Slika 5 Grafički prikaz validacije LSTM modela za drugi skup podataka



Slika 6 Grafički prikaz validacije LSTM modela za treći skup podataka



Slika 7 Grafički prikaz validacije LSTM modela za četvrti skup podataka



Slika 8 Grafički prikaz validacije LSTM modela za peti skup podataka

12 Literatura

[1] Keras -> Timeseries forecasting for weather prediction, link: https://keras.io/examples/timeseries/timeseries_weather_forecasting/?fbclid=IwAR1iQq-3pVDGqbt8rBQBN5d_qGRFfjJjDzgcxiKUTSRQ8TdgwBmCYOGwlVw, 06.12.2021(10:12).