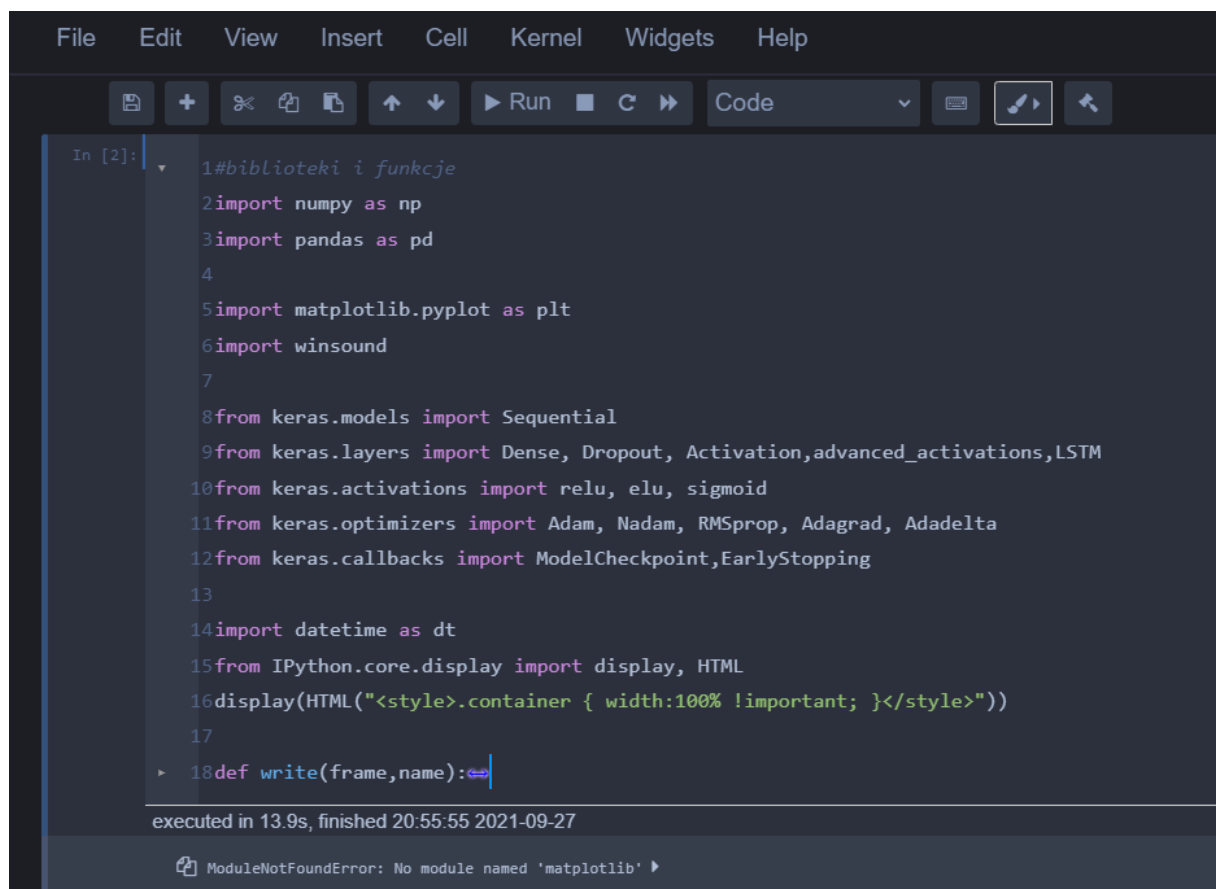


1. Wgrać anaconda nawigator z internetu
2. Uruchomić anaconda prompt
3. Utworzyć środowisko wirtualne z zadaną wersją python'a. W tym celu wpisać w okno konsoli anaconda prompt: **conda create -n tensorflow python=3.6** , po czym Zatwierdzić klikając **y** a potem enter
4. Uruchomić środowisko wirtualne wpisując **conda activate tensorflow**
5. Zainstalować tensorflow komendą **pip install tensorflow** . Zatwierdzić klikając **y** a potem enter

Po wykonaniu na dole pozycja (base) C:\Users\username > powinna zmienić się na (tensorflow) C:\Users\ username >

6. Zainstalować jupyter notebook komendą **conda install jupyter notebook** .Zatwierdzić klikając **y** a potem enter.

Ważne, żeby wykonać to w tym kroku żeby móc sprawdzać, czy środowisko importuje wszystkie moduły z nagłówka skryptów, czy też coś trzeba doinstalować. Najlepiej dla celów sprawdzania otworzyć anaconda nawigator i wybrać docelowo środowisko i GUI oraz równolegle aktywować to samo środowisko przez anaconda prompt. Wtedy z linii komend można dogrywać kolejne pakiety a w GUI w nawigatorze sprawdzać. GUI otwiera się w przeglądarce. Komunikat o braku danego modułu wygląda jak poniżej. Przy zauważeniu trzeba dograć moduł metodą **conda install**. Przy pomyślnym przejściu instalacji w miejscu błędu jest tylko komunikat [Using TensorFlow backend](#). (uwaga, przy wykonywaniu kilka razy pod rząd poprawnie komunikat potrafi zniknąć) Przy krokowym intalowaniu modułów może być potrzeba klikania kernel ->restart and clear output i cell->all output clear, cell0>execution timings->clear



The screenshot shows the Anaconda Jupyter Notebook interface. The top menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for saving, adding cells, undo, redo, running, and other functions. The main area displays a code cell with the following Python code:

```
In [2]: 1#biblioteki i funkcje
        2import numpy as np
        3import pandas as pd
        4
        5import matplotlib.pyplot as plt
        6import winsound
        7
        8from keras.models import Sequential
        9from keras.layers import Dense, Dropout, Activation,advanced_activations,LSTM
       10from keras.activations import relu, elu, sigmoid
       11from keras.optimizers import Adam, Nadam, RMSprop, Adagrad, Adadelata
       12from keras.callbacks import ModelCheckpoint,EarlyStopping
       13
       14import datetime as dt
       15from IPython.core.display import display, HTML
       16display(HTML("<style>.container { width:100% !important; }</style>"))
       17
       18def write(frame,name):
```

Below the code cell, it says "executed in 13.9s, finished 20:55:55 2021-09-27". At the bottom, a message box displays the error: "ModuleNotFoundError: No module named 'matplotlib'" with a small icon of a document with a red X.

7. Zainstalować **keras** komendą **conda install keras** .Zatwierdzić klikając **y** a potem enter. To jest nakładka na tensorflow.
8. Zainstalować **pandas** komendą **conda install pandas** .Zatwierdzić klikając **y** a potem enter.
9. Zainstalować **matplotlib** komendą **conda install matplotlib** .Zatwierdzić klikając **y** a potem enter.
10. Zainstalować **xlrd** komendą **conda install xlrd==1.1.0** .Zatwierdzić klikając **y** a potem enter.
11. Zainstalować **xlrw** komendą **pip install xlrw**

**UWAGA**-w nowszych wersjach tensorflow, np. 2.6.2 może pokazać się komunikat o niezgodności geometrii warstw z wartościami oczekiwanymi.

Należy wtedy w linii:

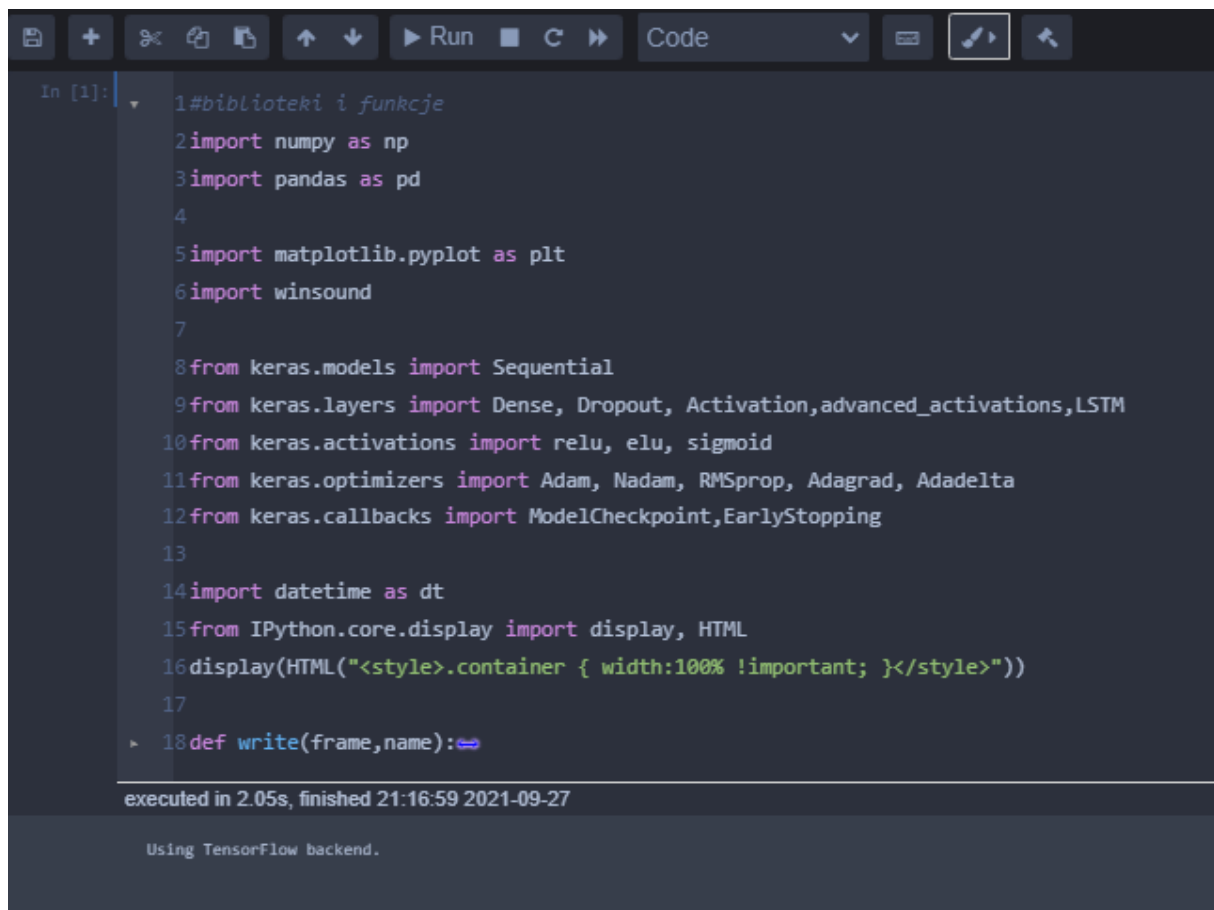
```
history=model.fit(in_tr,  
out_tr,validation_data=[in_val,out_val],epochs=epo,batch_size=128,verbose=1  
,shuffle=True,callbacks=[early_stopping])
```

zamienić [in\_val,out\_val] na (in\_val,out\_val)

Jupyter notebook można otwierać zarówno przez anaconda nawigator jak i z linii komend. W tym drugim przypadku należy zastosować conda activate tensorflow, potem komenda jupyter notebook.

**TO ważne, bo nawigator potrafi się zwieszać/baaardzo długo ładować plus tak jest dużo szybciej otworzyć.**

Sugerowane jest wykonanie skryptu szkielet skryptu (cel->run all) i zobaczenie czy ma się coś podobnego jak na screenach poniżej i czy zapisuje plik 'zzzz.xlsx' w folderze ze skryptem.



The image shows a Jupyter Notebook interface with a dark theme. The top toolbar includes icons for saving, adding cells, zooming, and running code. The code cell is labeled 'In [1]:' and contains 18 lines of Python code. The code imports various libraries including numpy, pandas, matplotlib, keras, and datetime. It also includes a custom HTML display command and a function definition. The output area shows the execution time and date, and a message about the TensorFlow backend.

```
In [1]: 1#biblioteki i funkcje
        2import numpy as np
        3import pandas as pd
        4
        5import matplotlib.pyplot as plt
        6import winsound
        7
        8from keras.models import Sequential
        9from keras.layers import Dense, Dropout, Activation,advanced_activations,LSTM
       10from keras.activations import relu, elu, sigmoid
       11from keras.optimizers import Adam, Nadam, RMSprop, Adagrad, Adadelta
       12from keras.callbacks import ModelCheckpoint,EarlyStopping
       13
       14import datetime as dt
       15from IPython.core.display import display, HTML
       16display(HTML("<style>.container { width:100% !important; }</style>"))
       17
       18def write(frame,name):
```

executed in 2.05s, finished 21:16:59 2021-09-27

Using TensorFlow backend.

```

In [2]: 1#wczytywanie przykładowej ramki danych
2data=pd.read_excel('df_ex.xlsx')
3
4#Normalizacja zmiennych
5data['A']=data['A']/100
6data['B']=data['B']/100
7data['Res']=data['Res']/10000
8
9#podział danych na treningowe/walidacyjne/testowe
10df_tr_in=data.loc[0:74,['A','B']]
11df_val_in=data.loc[75:100,['A','B']]
12df_test_in=data.loc[101:200,['A','B']]
13
14df_tr_out=data.loc[0:74,'Res']
15df_val_out=data.loc[75:100,'Res']
16df_test_out=data.loc[101:200,'Res']
17
18#przeformatowanie danych na postać macierzy 3D do lstm
19tr_samples_num=len(df_tr_in)
20in_tr=df_tr_in.values.reshape(tr_samples_num,1,len(df_tr_in.columns)) #najpierw Liczba próbek, potem Liczba wierszy, potem kolumn
21out_tr=df_tr_out.values.reshape(tr_samples_num,1,1)
22
23val_samples_num=len(df_val_in)
24in_val=df_val_in.values.reshape(val_samples_num,1,len(df_val_in.columns)) #najpierw Liczba próbek, potem Liczba wierszy, potem kolumn
25out_val=df_val_out.values.reshape(val_samples_num,1,1)
26
27test_samples_num=len(df_test_in)
28in_test=df_test_in.values.reshape(test_samples_num,1,len(df_test_in.columns)) #najpierw Liczba próbek, potem Liczba wierszy, potem kolumn
29out_test=df_test_out.values.reshape(test_samples_num,1,1)

```

executed in 35ms, finished 21:17:00 2021-09-27

```

In [5]: 1data

```

executed in 12ms, finished 21:17:36 2021-09-27

	A	B	Res
0	0.90	0.49	0.4410
1	0.09	0.90	0.0810
2	0.59	0.73	0.4307
3	0.35	0.74	0.2590
4	0.75	0.08	0.0600
...	...	...	...
195	0.30	0.38	0.1140
196	0.90	0.88	0.7920
197	0.11	0.11	0.0121
198	0.05	0.58	0.0280
199	0.35	0.35	0.5500

```

21 opt = optim.Adam(model.parameters()) #optimizer
22 #opt=trf(lr=0.001,learning_rate_power=-0.5,initial_accumulator_value=0.1,l1_regularization_strength=0.0,l2_regularization_strength=0.0,l2_shrinkage_regularization_strength=0.0,)
23 model.compile(loss='mae',optimizer=opt)
24
25
26 pat=20
27 epo=200
28
29 early_stopping = EarlyStopping(monitor='val_loss', patience=pat)
30 #instrukcja uczenia modelu
31 history=model.fit(in_tr, out_tr,validation_data=[in_val,out_val],epochs=epo,batch_size=128,verbose=1,shuffle=True,callbacks=[early_stopping])
32 #instrukcja walidacji modelu
33 score = model.evaluate(in_test, out_test, batch_size=128) #instrukcja testowania modelu
34
35 ucz=history.history['loss'][::-1]
36 spr=history.history['val_loss'][::-1]
37 prognosis=model.predict(in_test, batch_size=None, verbose=0, steps=None) #wpisać jak chce się wartości przewidywane uzyskać
38 print(f'MAE test [%] = {score*100} ,uczenie= {ucz*100}, sprawdza={spr*100}')
39
40
41 prognosis2=prognosis.Flatten().tolist()
42 out_test2=out_test.Flatten().tolist()
43 out=pd.DataFrame(list(zip(prognosis2,out_test2)))
44 out.columns=['prognosis','reals']
45
46 write(out,'progniza_iter')
47
48 plt.plot(history.history['loss'])
49 plt.plot(history.history['val_loss'])
50 plt.title('Model loss')
51 plt.ylabel('Loss')
52 plt.xlabel('Epoch')
53 plt.legend(['Train', 'Val'], loc='upper left')
54 plt.show()

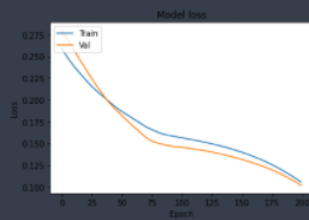
```

executed in 2.25s, finished 21:17:19 2021-09-27

```

Epoch 200/200
75/75 [-----] - 0s 1bus/step - loss: 0.1056 - val_loss: 0.1018
99/99 [-----] - 0s 0us/step
MAE test [%] = 11.00614070892334 ,uczenie= 10.558604170246124, sprawdza=10.101127488613129

```



```

(1): write(data, 'zzzz')

```

executed in 262ms, finished 21:17:03 2021-09-27