Koronavirus Dokumentace

Produkt mé práce je jednoduchý Jupyter notebook, který pomocí několika málo pomocných modulů pythonu zprostředkovává jednoduché rozhrání na modelování epidemie koronaviru pomocí knihovny TensorFlow. Projekt využívá dataset Oxfordské Univerzity a umožňuje tvorbu modelu neuronových sítí zpracovávající a předvídající postup epidemie v „singlestep“ (předvídání jeden den do budoucna) a „multistep“ (předvídání delší časové řady) modelů. Projekt mi hlavně sloužil na celkové seznámení s problematikou umělé inteligence, neurovnových sítí a předvídání časových řad. Zkoušel jsem vytvářet postupně složitější modely na předvídání vývoje epidemie. Začal jsem jednoduchým baseline modelem kopírujících poslední hodnotu, přes model jedné vrstvy lineární transformace a několik vrstev hustě spojených neuronů po long short term memory model a lstm s residual wraperem, které se ukázali být nejslibnější.

Instalace

Projekt je jednoduchý Jupyter notebook a pár python souborů zabalených v zipu. K provozu je tedy nutno mít nainstalovaný python a využití jupyter-labu nebo google colab. Využívám několik základních knihoven pythonu jako numpy, pandas, matplotlib a tesorflow.

Obsah řešení

- corona.ipynb – hlavní rozhraní jupyter notebook, vhodné pro zkoušky tvorby nových modelů a jejich srovnání.
- CoronaAI.py – hlavní třída projektu obsluhující celé workflow projektu
- Preprocessor.py – preprocesor pro snadnou úpravu dat
- Window.py – třída reprezentující časové okno řady
- WindowGenerator.py – třída zařizující tvorbu oken
- Models.py – modul obsahující pomocné funkce a třídy modelů

Dataset

https://github.com/OxCGRT/covid-policy-tracker

Data, obsahující celkový počet nakažených, celkový počet mrtvých a různé kategorické údaje popisující ve stupních určitou míru zásahu vlád v dané kategorie, jsou stáhnuta z gitu Oxforské Univerzity, uložena a předzpracována. Následně je možno vytvořit si vlastní modely TF, registrovat je třídě CoronaAI a následně natrénovat a vyhodnotit na nějakém státě.

Dataset obsahuje informace:

- pro každou zemi (či region), vdaný den (v projektu pak přepočítávám na denní přírůstek):
  - celkovém počtu nakažených
  - celkovém počtu zemřelých

Jednotlivá státní opatření se v datasetu řadí do několika kategorií:
https://github.com/OxCGRT/covid-policy-tracker/blob/master/documentation/codebook.md

C. containment and closure policies
zde se řadí v podstatě všechna základní opatření jako zavření pracovišť a škol, zrušení veřejných akcí, omezení shromažďování a omezení veřejné dopravy
V projektu jsem se zabýval v podstatě jen touhle částí dat, jelikož mi přišla relevantní.

E. economic policies
Zde se zaobírá převáženě ekonomickou podporou / finanční pomocí.

H. health system policies
Tato kategorie se zabývá zdravotnictvím, informovaností obyvatelstva, testováním a opatřením jako jsou roušky a respirátory.

V. vaccination policies
Tata data jsou nějaká specifická data přímo k vakcinaci a různým strategiím očkování.

M. miscellaneous policies
Pod M patří asi cokoliv co nepatří jinam.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144153 entries, 0 to 144152
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   CountryName                 144153 non-null  object
 1   RegionName                  48735 non-null   object
 2   Date                        144153 non-null  datetime64[ns]
 3   C1_School closing           139015 non-null  float64
 4   C2_Workplace closing        138924 non-null  float64
 5   C3_Cancel public events     139012 non-null  float64
 6   C4_Restrictions on gatherings  138994 non-null  float64
 7   C5_Close public transport   138831 non-null  float64
 8   ConfirmedCases              134277 non-null  float64
 9   ConfirmedDeaths             134198 non-null  float64
dtypes: datetime64[ns](1), float64(7), object(2)
memory usage: 11.0+ MB
None
```

[4]:
```
corona.country_data("Czech Republic")
```

[4]:

Date	C1_School closing	C2_Workplace closing	C3_Cancel public events	C4_Restrictions on gatherings	C5_Close public transport	ConfirmedCases	ConfirmedDeaths	NewCases	NewDeaths
2020-01-01	0.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN
2020-01-02	0.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN
2020-01-03	0.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN
2020-01-04	0.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN
2020-01-05	0.0	0.0	0.0	0.0	0.0	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
2021-05-23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	335.0	8.0
2021-05-24	NaN	NaN	NaN	NaN	NaN	NaN	NaN	199.0	8.0
2021-05-25	NaN	NaN	NaN	NaN	NaN	NaN	NaN	686.0	13.0
2021-05-26	NaN	NaN	NaN	NaN	NaN	NaN	NaN	655.0	13.0
2021-05-27	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

https://raw.githubusercontent.com/OxCGRT/covid-policy-tracker/master/data/OxCGRT_latest.csv

Základní přehled datasetu ČR:

```
corona.country_data("Czech Republic").describe()
```

	C1_School closing	C2_Workplace closing	C3_Cancel public events	C4_Restrictions on gatherings	C5_Close public transport	C6_Stay at home requirements	C7_Restrictions on internal movement	C8_International travel controls	NewCases	NewDeaths
count	504.000000	504.000000	504.000000	504.000000	504.000000	504.000000	504.000000	504.000000	504.000000	504.000000
mean	1.932540	1.547619	1.331349	2.787698	0.170635	0.829365	0.976190	2.865079	3299.023810	59.851190
std	0.974533	0.770811	0.642536	1.421424	0.376563	0.899884	0.880216	0.540277	4459.283667	74.288405
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-2214.000000	-3.000000
25%	1.000000	1.000000	1.000000	2.000000	0.000000	0.000000	0.000000	3.000000	92.750000	1.000000
50%	2.000000	2.000000	1.000000	4.000000	0.000000	0.000000	1.000000	3.000000	675.500000	10.000000
75%	3.000000	2.000000	2.000000	4.000000	0.000000	2.000000	2.000000	3.000000	5313.250000	117.750000
max	3.000000	3.000000	2.000000	4.000000	1.000000	2.000000	2.000000	4.000000	17773.000000	295.000000

Z popisu dat jsem objevil jistý šum v datech, jelikož NewCases, ani NewDeaths nemůže být nikdy záporné, ačkoliv v datech se takové hodnoty objevily. Tyto hodnoty by se mohu bych mohl zkusit nahradit za nuly (novější hodnoty), zda to bude mít dopady na výsledky běhu.

Train

```
train.describe()
```

	C1_School closing	C2_Workplace closing	C3_Cancel public events	C4_Restrictions on gatherings	C5_Close public transport	C6_Stay at home requirements	C7_Restrictions on internal movement	C8_International travel controls	NewCases	NewDeaths
count	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	1.539735	1.294702	1.089404	2.135762	0.023179	0.572848	0.682119	2.774834	1595.215232	22.761589
std	0.993372	0.886914	0.632944	1.425251	0.150721	0.827070	0.723326	0.683670	3429.801072	54.330746
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-3.000000
25%	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	2.000000	48.000000	0.000000
50%	1.000000	1.000000	1.000000	2.000000	0.000000	0.000000	1.000000	3.000000	131.500000	2.000000
75%	2.000000	2.000000	1.000000	4.000000	0.000000	1.000000	1.000000	3.000000	644.000000	8.000000
max	3.000000	3.000000	2.000000	4.000000	1.000000	2.000000	2.000000	4.000000	15731.000000	295.000000

U trénovacích dat samotných není tento šum tak patrný.

Test

```
test.describe()
```

	C1_School closing	C2_Workplace closing	C3_Cancel public events	C4_Restrictions on gatherings	C5_Close public transport	C6_Stay at home requirements	C7_Restrictions on internal movement	C8_International travel controls	NewCases	NewDeaths
count	101.000000	101.000000	101.000000	101.0	101.000000	101.000000	101.0	101.0	101.000000	101.000000
mean	2.693069	1.851485	1.386139	4.0	0.297030	1.663366	2.0	3.0	7462.623762	133.316832
std	0.463521	0.357383	0.489291	0.0	0.459229	0.474915	0.0	0.0	4064.046906	37.849949
min	2.000000	1.000000	1.000000	4.0	0.000000	1.000000	2.0	3.0	0.000000	0.000000
25%	2.000000	2.000000	1.000000	4.0	0.000000	1.000000	2.0	3.0	4289.000000	107.000000
50%	3.000000	2.000000	1.000000	4.0	0.000000	2.000000	2.0	3.0	7488.000000	137.000000
75%	3.000000	2.000000	2.000000	4.0	1.000000	2.000000	2.0	3.0	9537.000000	157.000000
max	3.000000	2.000000	2.000000	4.0	1.000000	2.000000	2.0	3.0	17773.000000	213.000000

U testovacích dokonce není vůbec. Min obou NewCases a NewDeaths je 0.

Validation

```
val.describe()
```

	C1_School closing	C2_Workplace closing	C3_Cancel public events	C4_Restrictions on gatherings	C5_Close public transport	C6_Stay at home requirements	C7_Restrictions on internal movement	C8_International travel controls	NewCases	NewDeaths
count	101.000000	101.0	101.0	101.000000	101.000000	101.000000	101.000000	101.0	101.000000	101.000000
mean	2.346535	2.0	2.0	3.524752	0.485149	0.762376	0.831683	3.0	4229.980198	97.287129
std	0.607217	0.0	0.0	0.855501	0.502272	0.939665	0.990649	0.0	4576.753712	80.129313
min	1.000000	2.0	2.0	2.000000	0.000000	0.000000	0.000000	3.0	-2214.000000	1.000000
25%	2.000000	2.0	2.0	4.000000	0.000000	0.000000	0.000000	3.0	726.000000	23.000000
50%	2.000000	2.0	2.0	4.000000	0.000000	0.000000	0.000000	3.0	2222.000000	76.000000
75%	3.000000	2.0	2.0	4.000000	1.000000	2.000000	2.000000	3.0	6970.000000	160.000000
max	3.000000	2.0	2.0	4.000000	1.000000	2.000000	2.000000	3.0	16816.000000	278.000000

Vidíme, že největší šum je ve validačních datech, což jsou data časově nejnovější, což dává smysl.

Jak projekt funguje?

Hlavním výsledkem a zároveň uživatelským rozhraním je jupyter notebook, který umožňuje komunikovat s hlavní třídou, *CoronaAI*. Ta například načte data z gitu, zpracuje a uloží si je pod příslušné regiony. V rámci preprocesingu bylo třeba ujasnit si s čím chci pracovat. Původně jsem pracoval pouze s daty z České republiky, ale projekt je schopný pouhou změnou parametru změnit svoje působení.

Preprocessor se pak v pozadí stará rozdělení dat na učící, testovací a validační data, připravení vhodných column transformerů a očištěny od šumu.

Původně jsem různá vládní opatření bral jako kategorická data, vytvořil jsem si one-hot encoder. Nicméně později po konzultaci jsem přešel na reprezentaci opatření pomocí hodnot float, jakožto vhodnější pro neuronové sítě.

Dále projekt nabízí jednoduché prostření pro nastavení vlastního nového modelu a otestování jeho nastavení. Primární motivací pro volbu takového formátu mi byla moje vlastní neznalost a potřeba se seznámit s problematikou a když jsem viděl, že opakuji spousty kódu, dal jsem si za úkol vytvořit jednoduchý framework na práci s touto problematikou. Myšlenka byla taková, že tak si vše dobře ujasním, jak má fungovat.

Testované modely

Máme-l připravený framework, můžeme začít tvořit nějaké modely. Modely můžeme rozdělit do dvou skupin:

I. Single step model
   Jedná se o modely, které na základě nějaké sekvence vstupů předpovídají jednu hodnotu do budoucnosti.
   a. Baseline model
      Jedná se o zcela jednoduchý model, který pouze kopíruje poslední známou hodnotu. Takový model se nám bude hodit pro srovnání ostatních modelů.

```
baseline = Baseline()
corona.register_model("Baseline", baseline, WindowGenerator.single_step_window)
```

   b. Linear model
      Pouze jedna vrstva – lineární transformace

```
linear = tf.keras.Sequential([
    tf.keras.layers.Dense(units=corona.num_features)
])
corona.register_model("Linear", linear, WindowGenerator.single_step_window)
```

   c. Dense model
      Dvě hustě provázané vrstvy neuronů

```
dense = tf.keras.Sequential([
    tf.keras.layers.Dense(units=64, activation='relu'),
    tf.keras.layers.Dense(units=64, activation='relu'),
    tf.keras.layers.Dense(units=corona.num_features)
])
corona.register_model("Dense", dense, WindowGenerator.single_step_window)
```

   d. Multi layers dense
      Experiment s více vrstvami, nemá moc efekt

```
multi_step_dense = tf.keras.Sequential([
    # Shape: (time, features) => (time*features)
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=corona.num_features),
    # Add back the time dimension.
    # Shape: (outputs) => (1, outputs)
    tf.keras.layers.Reshape([1, -1]),
])
corona.register_model("Multistep dense", multi_step_dense, WindowGenerator.conv_window)
```

   e. Conv model
      Test konvuluční sítě

```
conv_model = tf.keras.Sequential([
    tf.keras.layers.Conv1D(filters=32,
                           kernel_size=(CONV_WIDTH,),
                           activation='relu'),
    tf.keras.layers.Dense(units=32, activation='relu'),
    tf.keras.layers.Dense(units=corona.num_features),
])
corona.register_model("Convulation", conv_model, WindowGenerator.conv_window)
```

   f. LSTM model
      Zatím neslibnější model na takové časové řady

```
lstm_model = tf.keras.models.Sequential([
    # Shape [batch, time, features] => [batch, time, lstm_units]
    tf.keras.layers.LSTM(32, return_sequences=True),
    tf.keras.layers.Dense(units=corona.num_features)
])
corona.register_model("LSTM", lstm_model, WindowGenerator.wide_window)
```

g. Residual LSTM
Ukazuje být nejlepší pro tuto řadu single step modelů

```
residual_lstm = ResidualWrapper(
    tf.keras.Sequential([
    tf.keras.layers.LSTM(32, return_sequences=True),
    tf.keras.layers.Dense(
        corona.num_features,
        # The predicted deltas should start small
        # So initialize the output layer with zeros
        kernel_initializer=tf.initializers.zeros())
]))
corona.register_model("Residual LSTM", residual_lstm, WindowGenerator.wide_window)
```

II. Multi step model

Multistep modely předpovídají delší časový úsek do budoucnosti.

a. Baseline last
Kopíruje pořád poslední hodnotu

```
last_baseline = MultiStepLastBaseline(OUT_STEPS)
corona.register_model("Multi Last", last_baseline, WindowGenerator.multi_window)
```

b. Baseline repeat
Kopíruje vstupní sekvenci na výstup

```
repeat_baseline = RepeatBaseline(OUT_STEPS)
corona.register_model("Multi Repeat", repeat_baseline, WindowGenerator.multi_window)
```

c. Linear model

```
multi_linear_model = tf.keras.Sequential([
    # Take the last time-step.
    # Shape [batch, time, features] => [batch, 1, features]
    tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),
    # Shape => [batch, 1, out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*corona.num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, corona.num_features])
])
corona.register_model("Multi Linear", multi_linear_model, WindowGenerator.multi_window)
```

d. Dense model

```
multi_dense_model = tf.keras.Sequential([
    # Take the last time step.
    # Shape [batch, time, features] => [batch, 1, features]
    tf.keras.layers.Lambda(lambda x: x[:, -1:, :]),
    # Shape => [batch, 1, dense_units]
    tf.keras.layers.Dense(512, activation='relu'),
    # Shape => [batch, out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*corona.num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, corona.num_features])
])
corona.register_model("Multi Dense", multi_dense_model, WindowGenerator.multi_window)
```

e. Konvuluční model

```
multi_conv_model = tf.keras.Sequential([
    # Shape [batch, time, features] => [batch, CONV_WIDTH, features]
    tf.keras.layers.Lambda(lambda x: x[:, -CONV_WIDTH:, :]),
    # Shape => [batch, 1, conv_units]
    tf.keras.layers.Conv1D(256, activation='relu', kernel_size=(CONV_WIDTH)),
    # Shape => [batch, 1,  out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*corona.num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, corona.num_features])
])
corona.register_model("Multi Convu", multi_conv_model, WindowGenerator.multi_window)
```

f. LSTM model

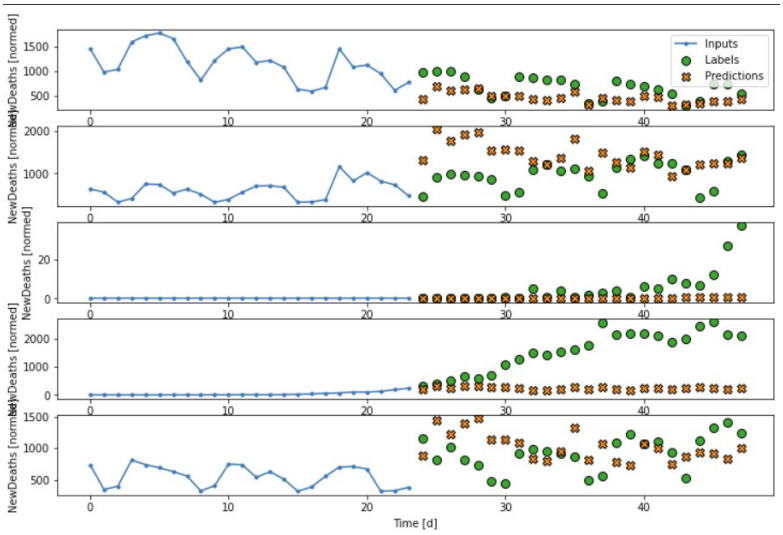
```
multi_lstm_model = tf.keras.Sequential([
    # Shape [batch, time, features] => [batch, lstm_units]
    # Adding more `lstm_units` just overfits more quickly.
    tf.keras.layers.LSTM(32, return_sequences=False),
    # Shape => [batch, out_steps*features]
    tf.keras.layers.Dense(OUT_STEPS*corona.num_features,
                          kernel_initializer=tf.initializers.zeros()),
    # Shape => [batch, out_steps, features]
    tf.keras.layers.Reshape([OUT_STEPS, corona.num_features])
])
corona.register_model("Multi LSTM", multi_lstm_model, WindowGenerator.multi_window)
```

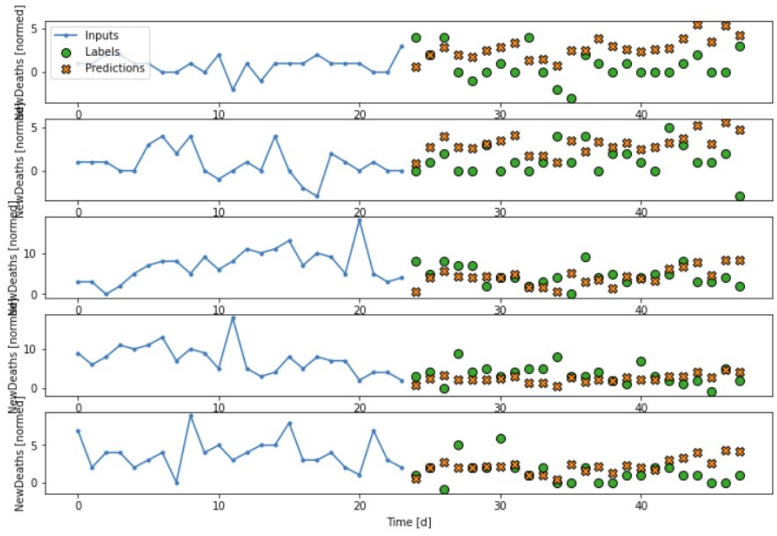
g. Residual LSTM model

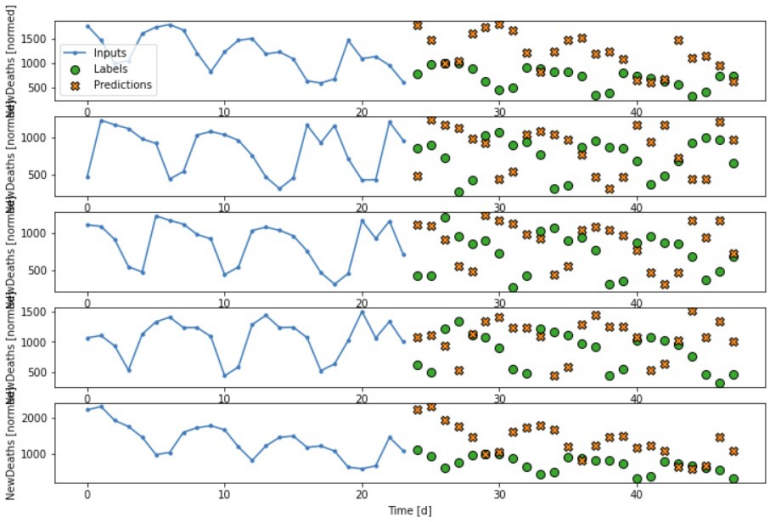
```
residual_lstm_model = ResidualWrapper(
    tf.keras.Sequential([
        # Shape [batch, time, features] => [batch, lstm_units]
        # Adding more `lstm_units` just overfits more quickly.
        tf.keras.layers.LSTM(32, return_sequences=False),

        # Shape => [batch, out_steps*features]
        tf.keras.layers.Dense(OUT_STEPS*corona.num_features,
                              kernel_initializer=tf.initializers.zeros()),
        # Shape => [batch, out_steps, features]
        tf.keras.layers.Reshape([OUT_STEPS, corona.num_features])
    ])
)
corona.register_model("Multi Residual LSTM", residual_lstm_model, WindowGenerator.multi_window)
```

h. Feedback model

Tento model se liší od ostatních multistep modelů tím, že předvídá hodnoty postupně a výstupy pak zpětně dodává jako vstupy pro další predikce.







Vyhodnocení a testování

```
corona.evaluate_models()
```

```
Input shape (batch, time, features): (32, 1, 10)
Output shape (batch, time, features): (32, 1, 10)
4/4 [==============================] - 0s 1000us/step - loss: 1067394.7500 - mean_absolute_error: 258.0170 - root_mean_squared_error: 1033.1479
None
NewDeaths
Epoch 1/100
10/10 [==============================] - 1s 18ms/step - loss: 2200389.4886 - mean_absolute_error: 588.9000 - root_mean_squared_error: 1476.9476 - val_l
oss: 9120433.0000 - val_mean_absolute_error: 2449.2695 - val_root_mean_squared_error: 3020.0054
Epoch 2/100
10/10 [==============================] - 0s 7ms/step - loss: 1704252.0568 - mean_absolute_error: 516.0473 - root_mean_squared_error: 1303.9768 - val_lo
ss: 8726278.0000 - val_mean_absolute_error: 2384.9634 - val_root_mean_squared_error: 2954.0273
Epoch 3/100
10/10 [==============================] - 0s 7ms/step - loss: 1920878.0455 - mean_absolute_error: 539.3498 - root_mean_squared_error: 1384.2847 - val_lo
ss: 8322118.5000 - val_mean_absolute_error: 2317.3105 - val_root_mean_squared_error: 2884.8081
Epoch 4/100
10/10 [==============================] - 0s 6ms/step - loss: 1626517.3068 - mean_absolute_error: 491.2797 - root_mean_squared_error: 1274.0634 - val_lo
ss: 7952925.0000 - val_mean_absolute_error: 2253.6064 - val_root_mean_squared_error: 2820.0928
Epoch 5/100
10/10 [==============================] - 0s 6ms/step - loss: 1585524.7500 - mean_absolute_error: 486.6572 - root_mean_squared_error: 1256.0460 - val_lo
ss: 7597824.5000 - val_mean_absolute_error: 2190.5088 - val_root_mean_squared_error: 2756.4150
Epoch 6/100
10/10 [==============================] - 0s 6ms/step - loss: 1206258.6080 - mean_absolute_error: 416.9385 - root_mean_squared_error: 1087.4197 - val_lo
ss: 7242543.5000 - val_mean_absolute_error: 2125.6985 - val_root_mean_squared_error: 2691.1970
Epoch 7/100
10/10 [==============================] - 0s 8ms/step - loss: 1469332.6705 - mean_absolute_error: 472.1556 - root_mean_squared_error: 1211.0923 - val_lo
ss: 6885535.5000 - val_mean_absolute_error: 2058.5645 - val_root_mean_squared_error: 2624.0303
Epoch 8/100
10/10 [==============================] - 0s 6ms/step - loss: 1421941.1477 - mean_absolute_error: 469.6593 - root_mean_squared_error: 1191.4886 - val_lo
ss: 6583627.0000 - val_mean_absolute_error: 2000.3079 - val_root_mean_squared_error: 2565.8582
Epoch 9/100
10/10 [==============================] - 0s 7ms/step - loss: 1539162.5568 - mean_absolute_error: 490.5460 - root_mean_squared_error: 1230.8103 - val_lo
ss: 6301064.5000 - val_mean_absolute_error: 1944.2908 - val_root_mean_squared_error: 2510.1921
```

Ukázka běhu vyhodnocení (učení) modelů.

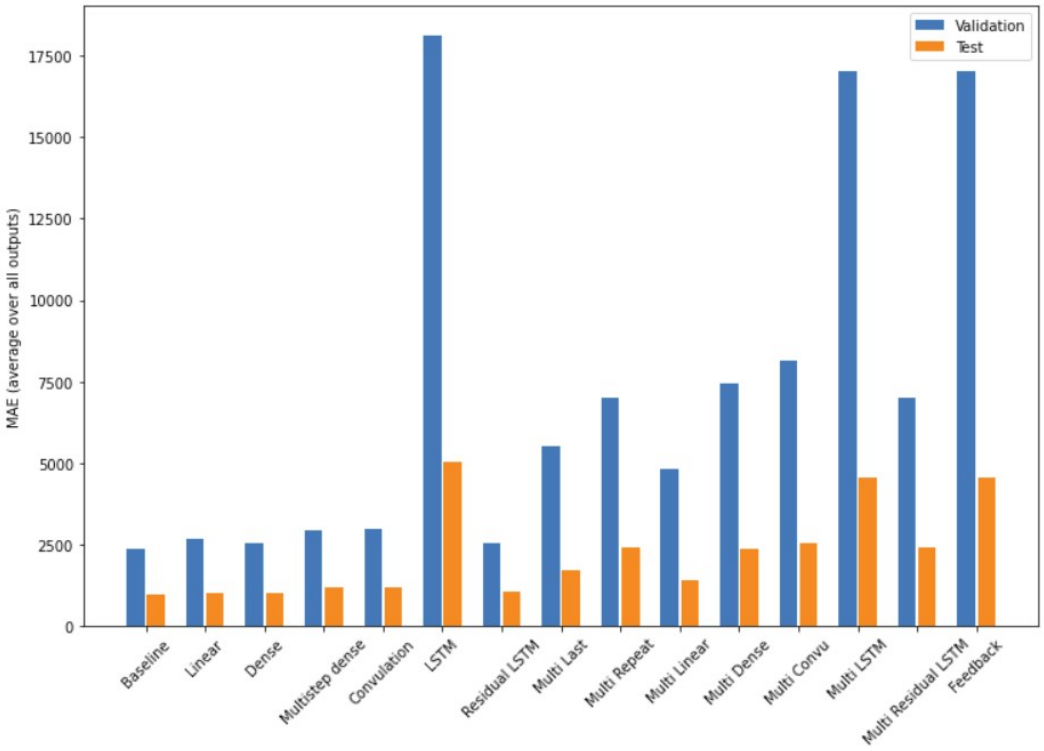
```
corona.evaluate_models()
```

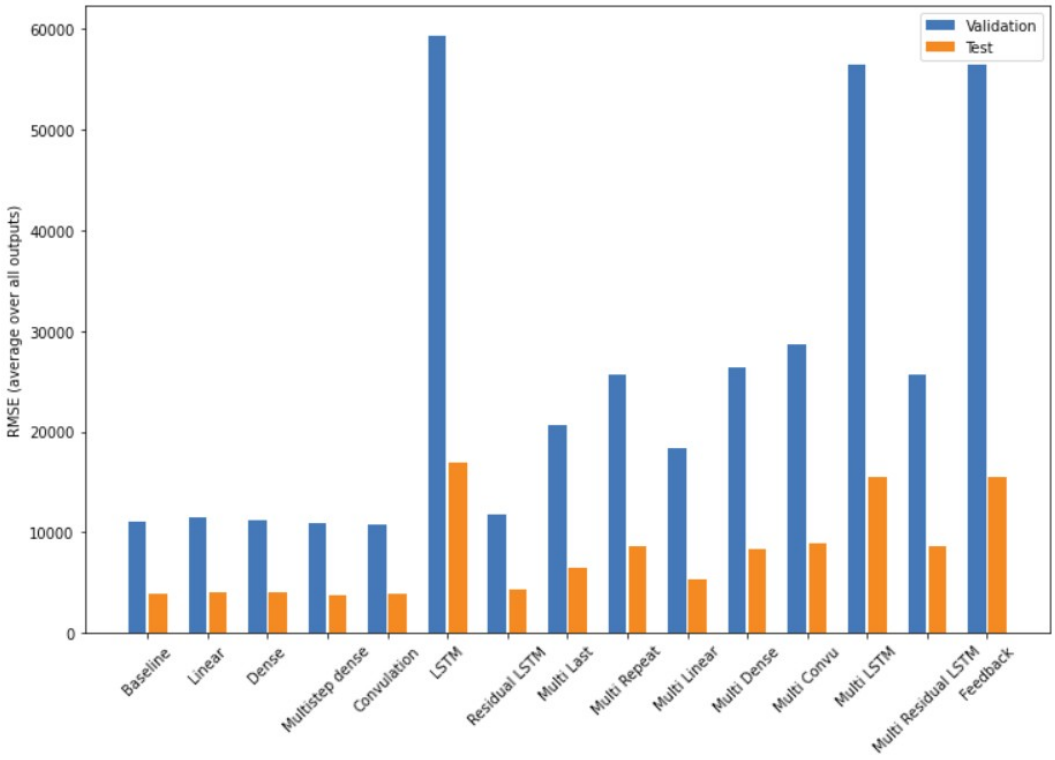
```
Input shape (batch, time, features): (32, 24, 10)
Output shape (batch, time, features): (32, 24, 10)
2/2 [==============================] - 0s 4ms/step - loss: 762436416.0000 - mean_absolute_error: 7666.2378 - root_mean_squared_error: 27612.2480
None
NewDeaths
Epoch 1/100
8/8 [==============================] - 3s 79ms/step - loss: 207190720.0000 - mean_absolute_error: 3873.1854 - root_mean_squared_error: 14386.1020 - val
_loss: 3181421568.0000 - val_mean_absolute_error: 17003.9004 - val_root_mean_squared_error: 56404.0898
Epoch 2/100
8/8 [==============================] - 0s 17ms/step - loss: 206394787.5556 - mean_absolute_error: 3921.0610 - root_mean_squared_error: 14352.5534 - val
_loss: 3181412864.0000 - val_mean_absolute_error: 17003.6914 - val_root_mean_squared_error: 56404.0117
Epoch 3/100
8/8 [==============================] - 0s 14ms/step - loss: 208023873.7778 - mean_absolute_error: 3971.0518 - root_mean_squared_error: 14417.7656 - val
_loss: 3181403136.0000 - val_mean_absolute_error: 17003.5332 - val_root_mean_squared_error: 56403.9336
Epoch 4/100
1/8 [==>.....................] - ETA: 0s - loss: 274703296.0000 - mean_absolute_error: 4055.8955 - root_mean_squared_error: 16574.1758
```

Postupně vyhodnocuje všechny registrované modely.

```
corona.evaluate_models()

Input shape (batch, time, features): (32, 24, 10)
Output shape (batch, time, features): (32, 24, 10)
2/2 [==============================] - 0s 5ms/step - loss: 3180446976.0000 - mean_absolute_error: 16999.0938 - root_mean_squared_error: 56395.4531
None
NewDeaths
Baseline                : 977.7629 (MAE), 3975.5391 (RMSE)
Linear                  : 982.0212 (MAE), 3941.0220 (RMSE)
Dense                   : 1022.5168 (MAE), 3981.7009 (RMSE)
Multistep dense         : 1127.5363 (MAE), 3813.9146 (RMSE)
Convulation             : 1250.5029 (MAE), 4009.3291 (RMSE)
LSTM                    : 5041.4917 (MAE), 16932.8164 (RMSE)
Residual LSTM           : 1058.1995 (MAE), 4265.7573 (RMSE)
Multi Last              : 1706.7347 (MAE), 6418.0547 (RMSE)
Multi Repeat            : 2423.5405 (MAE), 8686.5049 (RMSE)
Multi Linear            : 1424.4211 (MAE), 5353.0210 (RMSE)
Multi Dense             : 2412.7239 (MAE), 8694.8936 (RMSE)
Multi Convu             : 2635.7339 (MAE), 9035.9336 (RMSE)
Multi LSTM              : 4538.9365 (MAE), 15547.7402 (RMSE)
Multi Residual LSTM     : 2424.0891 (MAE), 8687.2344 (RMSE)
Feedback                : 4539.2290 (MAE), 15546.8555 (RMSE)
```

Bohužel se mi zatím nepodařilo dosáhnou převratných výsledků, spíše to pro mě bylo nové bádání a objevování. Nicméně postup mi v průběhu dělal radost. Paradoxně baseline model si vede až moc dobře, dotahuje LSTM s residual wrapperem. Vyhodnocoval jsem pomocí metrik mean average error a root mean squared error. Projekt má mezery v doladění precizní architektury sítě a vhodných parametrů. Jednoduché lineární modely dosahují slušných výsledků ve srovnání s baseline modely.





V závěru vidíme, že pouze multistep Linear model porazil multistep baseline modely na testovacích i validačních datech!