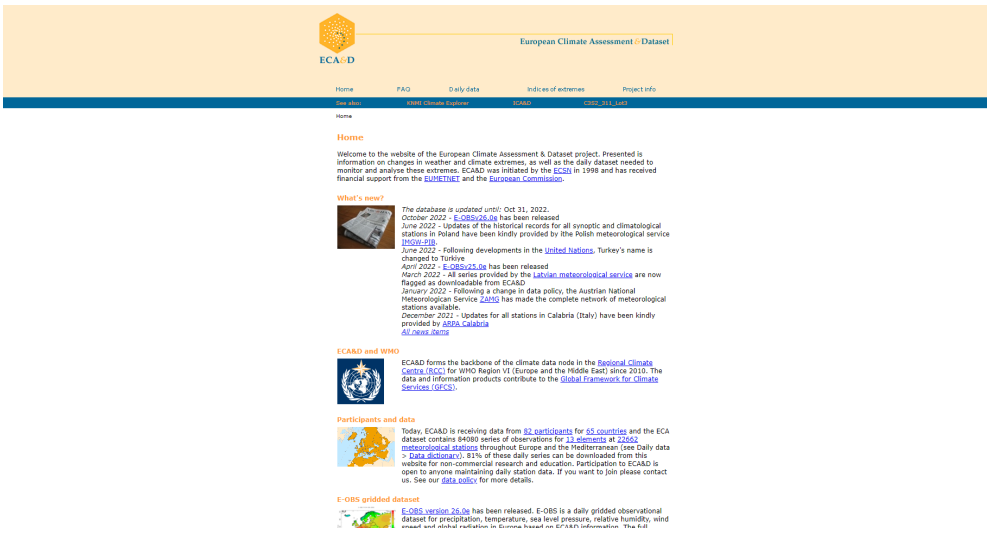
Temperature prediction - Slovakia

Alena Stracenská[1] and Ondrej Šima[2]

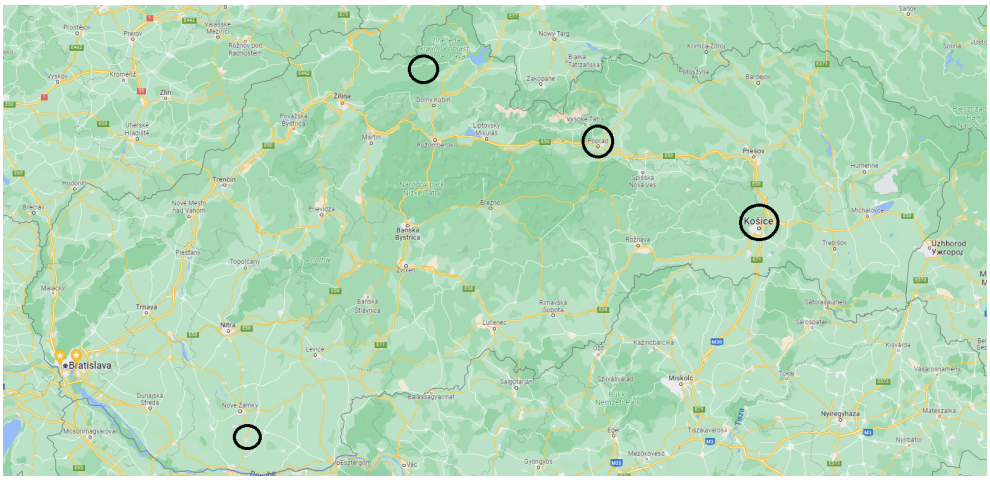
[1]stracensk@gmail.com
[2]osima1@student.euba.sk

1 Predikcia teplôt na Slovensku z historických údajov

V tomto projekte sme sa zamerali na predikciu počasia, a to konkrétne priemerných teplôt z historických údajov od roku 1951 po rok 2021. Ukážeme si, či platil trend zvyšovania teploty počas uvedených rokov a zároveň si ukážeme, ako by mohli priemerné teploty rásť až do roku 2055. Dáta sme prevzali v surovom stave zo stránky ecad.eu.

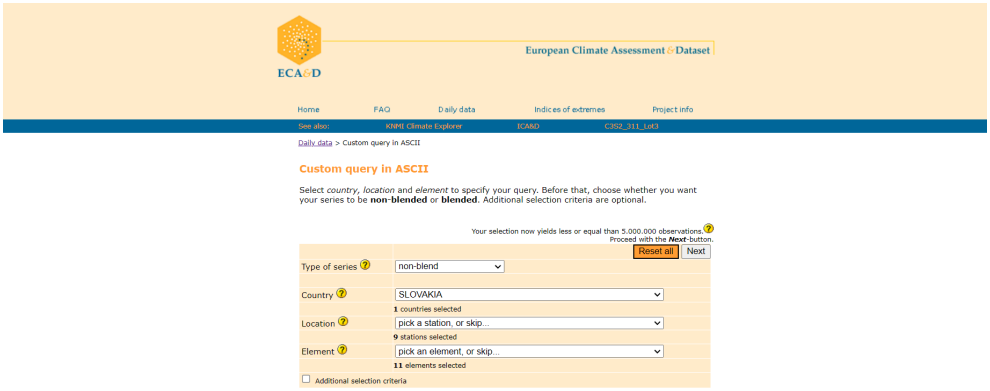


Obrázok č. 1: Zdrojová stránka dát.



Obrázok č. 2: Mapa miest, v ktorých sa nachádzajú meteostanice.

Takto sme si zvolili jednotlivé dáta zo Slovenska za všetky meteostanice a stiahli si ich.

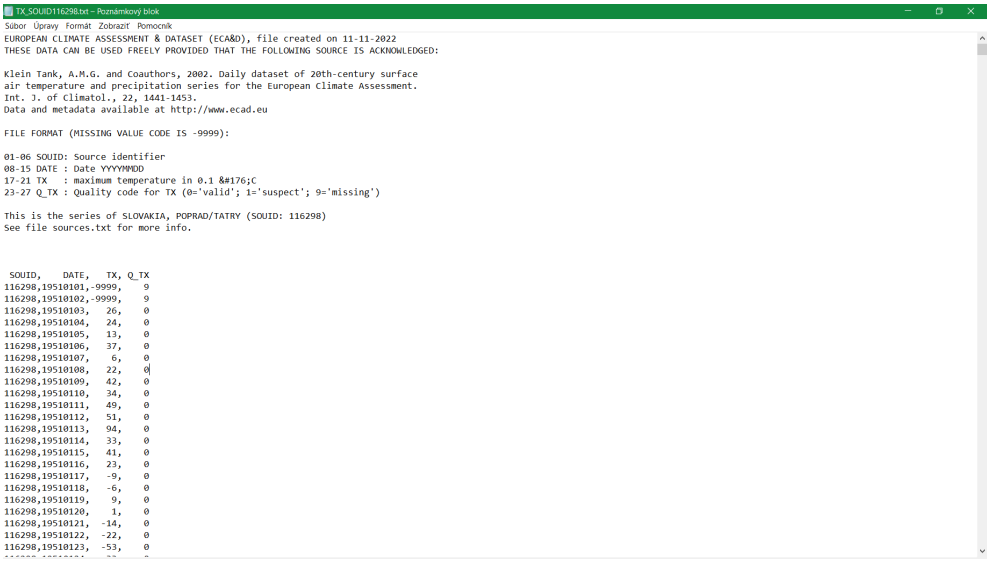


Obrázok č. 3: Náhľad na zvolené údaje pri sťahovaní datasetu.

Keďže dáta, s ktorými sme chceli pracovať boli prístupné len v štyroch meteostaniciach:

- Hurbanovo (skratka: HUR)
- Košice (skratka: KE)
- Poprad (skratka: PP)
- Oravská Lesná (skratka: ORL)

Tak zo stiahnutých súborov sme vytiahli dáta týkajúce sa týchto staníc pre maximálnu, minimálnu a priemernú teplotu a dátum. Stiahnuté zipko obsahovalo len súbory s príponou **.txt** a vyzerali takto:



Obrázok č. 4: Obsah .txt súboru.

Keďže úprava priamo v Pythone by bola mierne komplikovaná, rozhodli sme sa dáta spracovať priamo v Exceli a dať ich všetky do jedného súboru. Súbor je dostupný na stiahnutie tu: vsetky_udaje_sprocesovane_nofix.csv.

1.1 Importovanie používaných knižníc

[1]:
```
#import pouzitych kniznic
import os as os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as sk
import math
```

1.2 Ignorovanie warningov v celkom kóde

[2]:
```
#ignorovanie warningov
import warnings
warnings.filterwarnings('ignore')
```

1.3 Importovanie .csv filu a nastavenie max. počtu riadkov

Funkcia **pd.set_option**, nám umožní nastaviť max. počet riadkov, ktoré chceme zobraziť, a tie si následne korigujeme vo funkcii **head**.

[3]:
```
#import csv filu
df = pd.read_csv(r"C:/Users/strac/Desktop/5_rocnik_AS/ZS_22_23/ML/projekt_final/
 ↪vsetky_udaje_sprocesovane_nofix.csv", sep = ";")
#nastavenie maximalneho zobrazenia riadkov pre pandas, ktore si nasledne vieme␣
 ↪korigovat ich mnozstvom v heade
pd.set_option('display.max_rows', 30000)
df.head(5)
```

[3]:

	DATE	HUR_MEAN_TMP	KE_MEAN_TMP	ORL_MEAN_TEMP	PP_MEAN_TEMP	\
0	19510101	0	-9999	-9999	-9999	
1	19510102	33	-9999	-9999	-9999	
2	19510103	47	13	28	-8	
3	19510104	23	23	-6	0	
4	19510105	15	10	1	-20	

	HUR_MIN_TEMP	KE_MIN_TEMP	ORL_MIN_TEMP	PP_MIN_TEMP	HUR_MAX_TEMP	\
0	-22	-9999	-9999	-9999	-6	
1	4	-9999	-9999	-9999	-4	
2	19	-34	-7	-40	15	
3	11	-25	-17	-38	-19	
4	-7	-7	-40	-40	-22	

	KE_MAX_TEMP	ORL_MAX_TEMP	PP_MAX_TEMP
0	-9999	-9999	-9999
1	-9999	-9999	-9999
2	61	50	26
3	56	48	24
4	29	8	13

2 Čistenie dát

Ako môžeme vidieť vyššie, hodnoty -9999 pri teplotách boli chýbajúce, tie sme nahradili hodnotou NaN. Následne sme zmenili stĺpec **DATE**, ktorý bol typu string na objekt **datetime**, pre lepšiu a jednoduchšiu manipuláciu pri tvorbe grafov. Následne sme vynásobili teploty * 0.1, keďže boli hodnoty reprezentované ako dekadická (*decimal*) hodnota s pevným (v datasete skrytým) exponentom. Prvú časť čistenia dát teda máme za sebou.

[4]:
```
#nahradime zle hodnoty (-9999) prazdnou hodnotou (NaN)
df.replace(-9999,np.nan,inplace=True)

#uprava datumu (string) na objekt (datetime)
df["DATE"]=pd.to_datetime(df["DATE"],format="%Y%m%d")

#vynasobenie priemernych, max, min teplot, lebo zdrojova stranka udajov nacitavala␣
 ↪udaje len v celych cislach
df.loc[:, df.columns != "DATE"]=df.loc[:, df.columns != "DATE"]*0.1

df.head(5)
```

[4]:

	DATE	HUR_MEAN_TMP	KE_MEAN_TMP	ORL_MEAN_TEMP	PP_MEAN_TEMP	\
0	1951-01-01	0.0	NaN	NaN	NaN	
1	1951-01-02	3.3	NaN	NaN	NaN	
2	1951-01-03	4.7	1.3	2.8	-0.8	
3	1951-01-04	2.3	2.3	-0.6	0.0	
4	1951-01-05	1.5	1.0	0.1	-2.0	

	HUR_MIN_TEMP	KE_MIN_TEMP	ORL_MIN_TEMP	PP_MIN_TEMP	HUR_MAX_TEMP	\
0	-2.2	NaN	NaN	NaN	-0.6	
1	0.4	NaN	NaN	NaN	-0.4	
2	1.9	-3.4	-0.7	-4.0	1.5	
3	1.1	-2.5	-1.7	-3.8	-1.9	
4	-0.7	-0.7	-4.0	-4.0	-2.2	

	KE_MAX_TEMP	ORL_MAX_TEMP	PP_MAX_TEMP
0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	6.1	5.0	2.6
3	5.6	4.8	2.4
4	2.9	0.8	1.3

V tomto kroku sme pomocou interpolácie nahradili chýbajúce hodnoty. Napríklad si môžeme všimnúť, že údaje k teplotám za prvé 2 dni roku 1951 chýbali, takže cez interpoláciu sme doplnili hodnoty z dňa 3.1.1951 do predošlých dvoch prvých januárových dní. Prečo sme to tak spravili ? Ak by sme chýbajúce hodnoty nahradili len priemerom, tak teplota napr. 11.2 stupňa 1.1.1951 by zrejme nedávala zmysel, preto sme sa rozhodli použiť interpoláciu, ktorá pre nás predstavovala lepšie riešenie, než úplne skreslené číslo z priemeru.

[5]:
```
#linearna interpolacia neplatnych hodnot, okrem datumov, max 5 za sebou iducich,
#obojsmerne (doplni hodnotu aj ked pred nou hodnota este neexistovala)
for stlpec in df.drop('DATE',axis=1).columns:
    df[stlpec].interpolate(method='linear',limit=5,limit_direction='both',inplace=True)


df.head(5)
```

[5]:
```
         DATE  HUR_MEAN_TMP  KE_MEAN_TMP  ORL_MEAN_TEMP  PP_MEAN_TEMP  \
0 1951-01-01           0.0          1.3            2.8          -0.8
1 1951-01-02           3.3          1.3            2.8          -0.8
2 1951-01-03           4.7          1.3            2.8          -0.8
3 1951-01-04           2.3          2.3           -0.6           0.0
4 1951-01-05           1.5          1.0            0.1          -2.0

   HUR_MIN_TEMP  KE_MIN_TEMP  ORL_MIN_TEMP  PP_MIN_TEMP  HUR_MAX_TEMP  \
0          -2.2         -3.4          -0.7         -4.0          -0.6
1           0.4         -3.4          -0.7         -4.0          -0.4
2           1.9         -3.4          -0.7         -4.0           1.5
3           1.1         -2.5          -1.7         -3.8          -1.9
4          -0.7         -0.7          -4.0         -4.0          -2.2

   KE_MAX_TEMP  ORL_MAX_TEMP  PP_MAX_TEMP
0          6.1           5.0          2.6
1          6.1           5.0          2.6
2          6.1           5.0          2.6
3          5.6           4.8          2.4
4          2.9           0.8          1.3
```

[6]:
```
df.tail(5)
```

[6]:
```
             DATE  HUR_MEAN_TMP  KE_MEAN_TMP  ORL_MEAN_TEMP  PP_MEAN_TEMP  \
26201 2022-09-26           NaN          NaN            NaN           NaN
26202 2022-09-27           NaN          NaN            NaN           NaN
26203 2022-09-28           NaN          NaN            NaN           NaN
26204 2022-09-29           NaN          NaN            NaN           NaN
26205 2022-09-30           NaN          NaN            NaN           NaN

       HUR_MIN_TEMP  KE_MIN_TEMP  ORL_MIN_TEMP  PP_MIN_TEMP  HUR_MAX_TEMP  \
26201           NaN          NaN           NaN          NaN          23.9
26202           NaN          NaN           NaN          NaN          25.0
26203           NaN          NaN           NaN          NaN          25.6
26204           NaN          NaN           NaN          NaN          27.2
26205           NaN          NaN           NaN          NaN          26.3

       KE_MAX_TEMP  ORL_MAX_TEMP  PP_MAX_TEMP
26201          NaN           NaN          NaN
26202          NaN           NaN          NaN
26203          NaN           NaN          NaN
26204          NaN           NaN          NaN
26205          NaN           NaN          NaN
```

Následne sme pri prechádzaní dataframe objavili, že od dátumu 1.1.2021 zdroj datasetu nedisponuje údajmi, tak sme sa ich rozhodli odstrániť a zároveň sme nastavili teploty na maximálne dve desatinné miesta.

[7]:
```
#odstranenie hodnot NaN od datumu 1.1.2021, dovod: stranka nedisponuje od tohto datumu␣
 ↪hodnotami od SHMU
#fyi pre riadky plati n-1, preto v kode od 31.12.2020
df.drop(df[(df['DATE'] > '2020-12-31')].index, inplace=True)

#nastavenie kazdej hodnoty dataframe na max 2 des. miesta
pd.set_option('float_format', '{:.2f}'.format)
```

```
df.tail(5)
```

[7]:

	DATE	HUR_MEAN_TMP	KE_MEAN_TMP	ORL_MEAN_TEMP	PP_MEAN_TEMP	\
25563	2020-12-27	-0.90	0.30	-2.50	-2.40	
25564	2020-12-28	6.20	3.20	1.90	1.50	
25565	2020-12-29	6.80	7.20	1.50	2.10	
25566	2020-12-30	5.60	4.50	1.30	0.90	
25567	2020-12-31	1.10	4.50	-0.20	0.30	

	HUR_MIN_TEMP	KE_MIN_TEMP	ORL_MIN_TEMP	PP_MIN_TEMP	HUR_MAX_TEMP	\
25563	-6.30	-6.40	-13.20	-11.20	1.40	
25564	0.60	1.20	0.70	-0.50	2.60	
25565	3.50	2.10	0.60	0.30	2.70	
25566	3.80	2.30	-0.20	-1.20	1.80	
25567	-0.70	1.80	-1.20	-1.40	2.80	

	KE_MAX_TEMP	ORL_MAX_TEMP	PP_MAX_TEMP
25563	2.70	1.80	0.50
25564	5.30	2.80	2.50
25565	9.00	4.20	4.30
25566	8.00	2.40	3.70
25567	5.90	2.00	2.00

V tomto kroku sme si zistili, koľko nulových resp. NaN hodnot ešte máme v datasete. Vidíme, že ich je 51. Dané množstvo predstavuje 51 dní, čo nie je také zlé. Napriek tomu sme sa rozhodli ich odstrániť.

[8]:
```
#zistenie nulovych hodnot
df.isnull().sum()
```

[8]:
```
DATE              0
HUR_MEAN_TMP      0
KE_MEAN_TMP       0
ORL_MEAN_TEMP     0
PP_MEAN_TEMP      0
HUR_MIN_TEMP      0
KE_MIN_TEMP       0
ORL_MIN_TEMP     51
PP_MIN_TEMP       0
HUR_MAX_TEMP      0
KE_MAX_TEMP       0
ORL_MAX_TEMP     51
PP_MAX_TEMP       0
dtype: int64
```

Dataset máme očistený a pripravený na použitie.

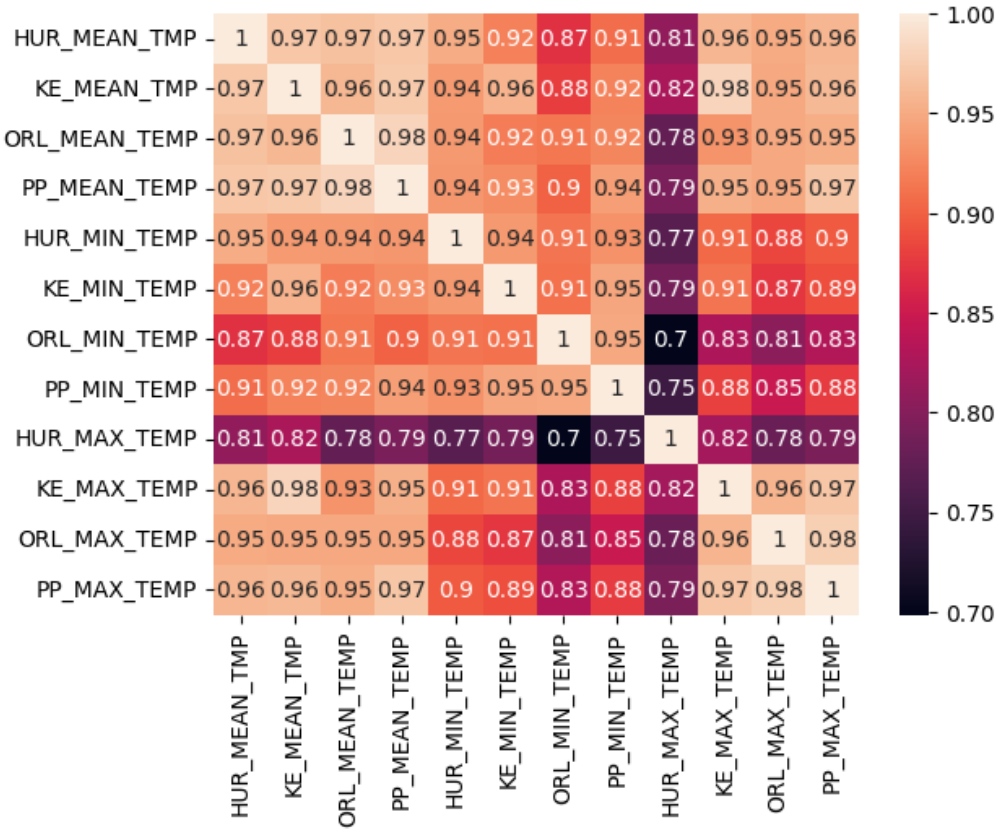
[9]:
```
#odstranenie zvysnych riadkov s nulovymi hodnotami
df.dropna(inplace = True)

#overenie, ci sa riadky dropli
df.isnull().sum()
```

[9]:
```
DATE              0
HUR_MEAN_TMP      0
KE_MEAN_TMP       0
ORL_MEAN_TEMP     0
PP_MEAN_TEMP      0
HUR_MIN_TEMP      0
KE_MIN_TEMP       0
ORL_MIN_TEMP      0
PP_MIN_TEMP       0
HUR_MAX_TEMP      0
KE_MAX_TEMP       0
ORL_MAX_TEMP      0
PP_MAX_TEMP       0
dtype: int64
```

3 Použitie očisteného datasetu

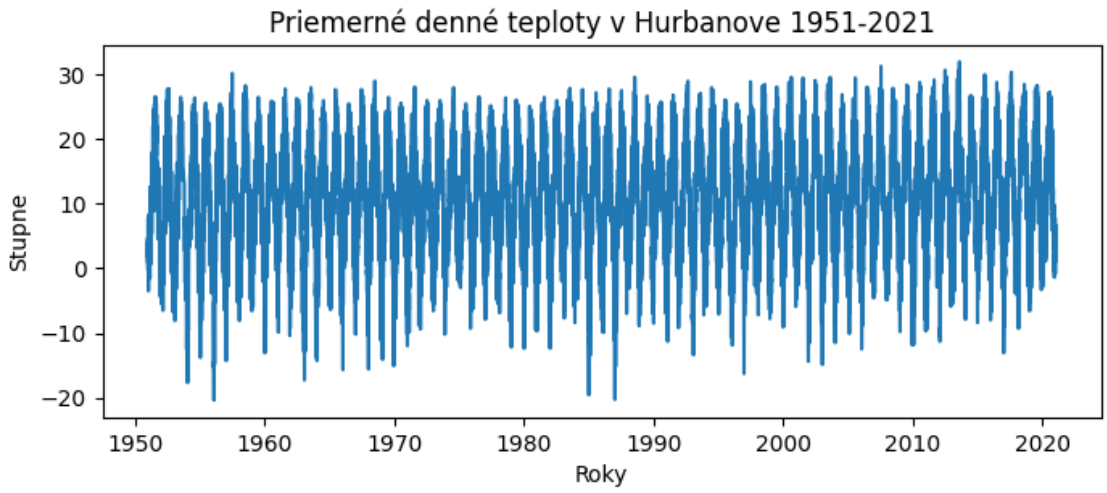
Z očisteného datasetu sme spravili korelačnú maticu. Vidíme, že teploty vysoko korelujú, čo značí, že je multikolinealita vysoká a mali by sme ich vylúčiť, avšak my s nimi potrebujeme pracovať. Čiže korelačná matica nám slúži pre informačné účely.

[10]:
```
#zobrazenie korelacnej matice
korelacnaMatica = df.corr()
sns.heatmap(korelacnaMatica, annot = True)
plt.show()
```



Z datasetu sme si zobrazili, ako vyzerali priemerné denné teploty v Hurbanove za posledných 70 rokov. Graf je strašne neprehľadný, ale slúži aspoň pre ilustráciu maxím a miním teplôt. To isté platilo aj pre ostatné tri mestá a to Košice, Poprad a Oravskú Lesnú, preto sme ich grafy zakomentovali (v prípade potreby sa dajú jednoducho odkomentovať).

[11]:
```
#vseobecny prehlad o priemernych dennych teplotach za roky 1951-2021 vo vsetkych␣
↪mestach
#graf priemernych dennych teplot 1951 - 2021 pre Hurbanovo
from matplotlib.pyplot import figure
plt.figure(figsize=(8,3))
plt.plot(df["DATE"], df["HUR_MEAN_TMP"])
#plt.legend(loc = 'center left', bbox_to_anchor = (1, 0.5))
plt.title("Priemerné denné teploty v Hurbanove 1951-2021")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()
'''
#graf priemernych dennych teplot 1951 - 2021 pre Oravsku Lesnu
plt.figure(figsize=(8,3))
plt.plot(df["DATE"], df["PP_MEAN_TEMP"])
plt.title("Priemerné denné teploty v Poprade 1951-2021")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()

#graf priemernych dennych teplot 1951 - 2021 pre Kosice
plt.figure(figsize=(8,3))
plt.plot(df["DATE"], df["KE_MEAN_TMP"])
plt.title("Priemerné denné teploty v Košiciach 1951-2021")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()

#graf priemernych dennych teplot 1951 - 2021 pre Oravsku Lesnu
plt.figure(figsize=(8,3))
plt.plot(df["DATE"], df["ORL_MEAN_TEMP"])
plt.title("Priemerné denné teploty v Oravskej Lesnej 1951-2021")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()
'''
```



[11]:
```
'\n#graf priemernych dennych teplot 1951 - 2021 pre Oravsku
Lesnu\nplt.figure(figsize=(8,3))\nplt.plot(df["DATE"],
df["PP_MEAN_TEMP"])\nplt.title("Priemerné denné teploty v Poprade
1951-2021")\nplt.xlabel("Roky")\nplt.ylabel("Stupne")\nplt.show()\n\n#graf
priemernych dennych teplot 1951 - 2021 pre
Kosice\nplt.figure(figsize=(8,3))\nplt.plot(df["DATE"],
df["KE_MEAN_TMP"])\nplt.title("Priemerné denné teploty v Košiciach
1951-2021")\nplt.xlabel("Roky")\nplt.ylabel("Stupne")\nplt.show()\n\n#graf
priemernych dennych teplot 1951 - 2021 pre Oravsku
Lesnu\nplt.figure(figsize=(8,3))\nplt.plot(df["DATE"],
df["ORL_MEAN_TEMP"])\nplt.title("Priemerné denné teploty v Oravskej Lesnej
1951-2021")\nplt.xlabel("Roky")\nplt.ylabel("Stupne")\nplt.show()\n'
```

V tomto kroku sme si spravili ročné priemery priemernej, maximálnej a minimálnej teploty pre všetky mestá za posledných 70 rokov. Priemerné ročné teploty sme následne zobrazili na grafe.

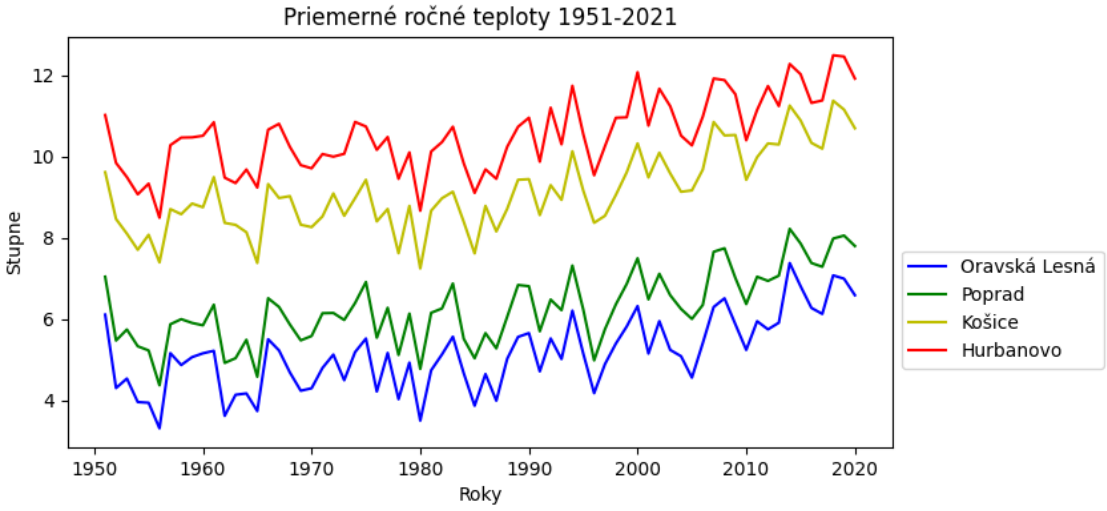
[12]:
```
##skusime rocne priemery
rocne=df.groupby(df.DATE.dt.year).mean()
hurbanovo = rocne[['HUR_MEAN_TMP', 'HUR_MAX_TEMP', 'HUR_MIN_TEMP']]
kosice = rocne[['KE_MEAN_TMP', 'KE_MAX_TEMP', 'KE_MIN_TEMP']]
poprad = rocne[['PP_MEAN_TEMP', 'PP_MAX_TEMP', 'PP_MIN_TEMP']]
kosice = rocne[['ORL_MEAN_TEMP', 'ORL_MAX_TEMP', 'ORL_MIN_TEMP']]
#prediktory
X3=rocne.index
X3=X3.values.reshape((len(X3), 1))
rocne.tail()
```

[12]:

DATE	HUR_MEAN_TMP	KE_MEAN_TMP	ORL_MEAN_TEMP	PP_MEAN_TEMP	HUR_MIN_TEMP	\
2016	11.32	10.34	6.28	7.38	6.44	
2017	11.38	10.20	6.13	7.29	6.05	
2018	12.50	11.38	7.08	7.98	7.50	
2019	12.46	11.16	7.00	8.06	7.30	
2020	11.92	10.70	6.59	7.80	6.95	

```
      KE_MIN_TEMP  ORL_MIN_TEMP  PP_MIN_TEMP  HUR_MAX_TEMP  KE_MAX_TEMP  \
DATE
2016         5.64          1.44         2.34         15.12        14.97
2017         5.17          1.12         1.87         14.74        15.01
2018         6.73          2.03         2.55         15.70        16.38
2019         6.45          2.06         2.73         15.21        16.53
2020         5.98          1.57         2.56         15.41        15.70

      ORL_MAX_TEMP  PP_MAX_TEMP
DATE
2016         11.92        13.01
2017         11.79        13.06
2018         13.04        13.83
2019         12.81        14.30
2020         12.37        13.62
```

[13]:
```
plt.figure(figsize=(8,4))
plt.plot(rocne.index, rocne["ORL_MEAN_TEMP"], 'b', label = "Oravská Lesná")
plt.plot(rocne.index, rocne["PP_MEAN_TEMP"], 'g', label = "Poprad")
plt.plot(rocne.index, rocne["KE_MEAN_TMP"], 'y', label = "Košice")
plt.plot(rocne.index, rocne["HUR_MEAN_TMP"],'r', label = "Hurbanovo")
plt.legend(loc = 'best', bbox_to_anchor = (1, 0.5))
plt.title("Priemerné ročné teploty 1951-2021")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()
```



3.1 Lineárny regresný model

Najskôr sme vyskúšali tvorbu lineárnej regresie len čisto pre 1 lokalitu. Ako nezávislú premennú sme vzali **DATE** a ako závislé premenné sme vzali **HUR_MEAN_TMP**, **HUR_MAX_TEMP** a **HUR_MIN_TEMP**. Následne sme rozdelili dáta na trénovacie 75% a testovacie 25%, potom sme špecificky použili funkciu **make_pipeline** pomocou ktorej bolo možné spojiť normalizáciu dát (kvôli veľkým

teplotným výkyvom) a inicializáciu lineárnej regresie. Následne sme pomocou **lm.fit** natrénovali dáta.

[14]:
```
#najskor vyskusame len cisto regresiu samotnej teploty pre 1 lokalitu
#targetova/cielova/predikovana hodnota
Y = df[['HUR_MEAN_TMP', 'HUR_MAX_TEMP', 'HUR_MIN_TEMP']]
#prediktor
X=df[['DATE']]
```

[15]:
```
#trenovanie a testovanie X a Y
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.25,␣
 ↪random_state = 42)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

lm=make_pipeline(
    StandardScaler(),
    LinearRegression(),
)


lm.fit(X_train,y_train)
```

```
(19137, 1)
(6380, 1)
(19137, 3)
(6380, 3)
```

[15]:
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('linearregression', LinearRegression())])
```

Použitie **MAE**, **RMSE** a **MSE** na otestovanie odchýlok modelu. Pre všetky 3 platí, že čím nižšia hodnota, tým lepšie.

Vidíme, že dáta majú dosť veľkú odchýlku. V ideálnom prípade by mala byť čo najnižšia. Mali by sme brať ale do úvahy, že rozpätie denných teplôt v závislosti na ročné obdobie môže kolísať od cca -20 až po +35 stupňov.

[16]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
print("Mean absolute error linearneho modelu / trenovacie:
 ↪",round(mean_absolute_error(y_train,lm.predict(X_train)),5))
print("Mean absolute error linearneho modelu / testovacie:
 ↪",round(mean_absolute_error(y_test,lm.predict(X_test)),5))
```

```
Mean absolute error linearneho modelu / trenovacie: 7.47687
Mean absolute error linearneho modelu / testovacie: 7.47807
```

[17]:
```
print("Mean square error linearneho modelu / trenovacie:
 ↪",round(mean_squared_error(y_train,lm.predict(X_train)),5))
print("Mean square error linearneho modelu / testovacie:
 ↪",round(mean_squared_error(y_test,lm.predict(X_test)),5))
```

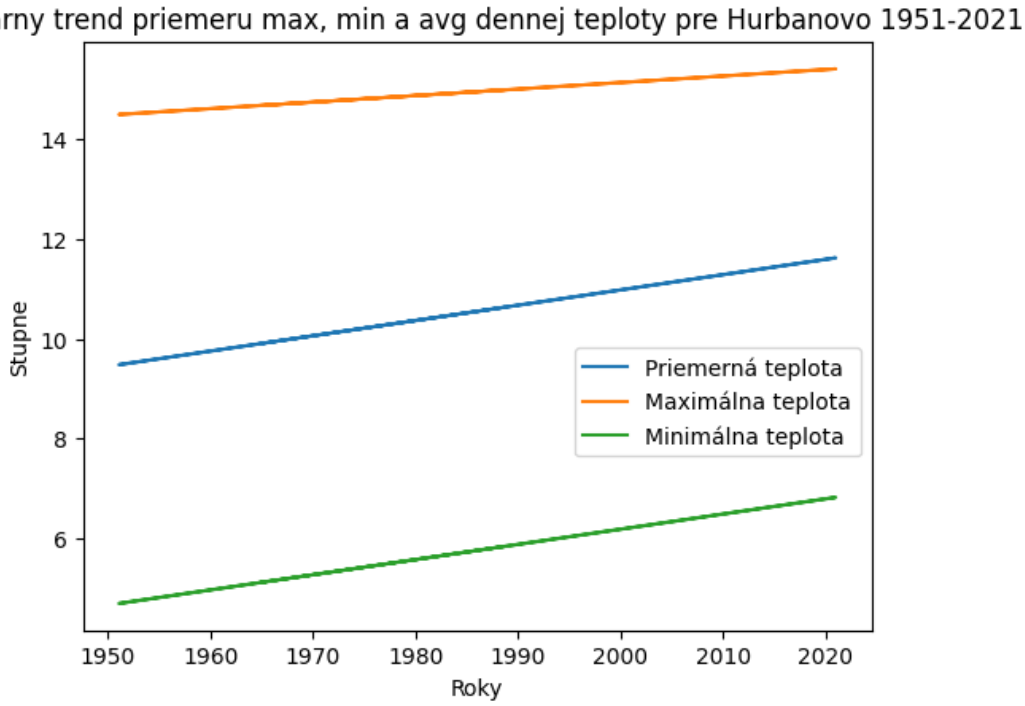
```
Mean square error linearneho modelu / trenovacie: 78.162
Mean square error linearneho modelu / testovacie: 77.96308
```

[18]:
```
print("Root mean square error linearneho modelu / trenovacie:",round(math.
 ↪sqrt(mean_squared_error(y_train,lm.predict(X_train))),5))
print("Root mean square error linearneho modelu / testovacie:",round(math.
 ↪sqrt(mean_squared_error(y_test,lm.predict(X_test))),5))
```

```
Root mean square error linearneho modelu / trenovacie: 8.84093
Root mean square error linearneho modelu / testovacie: 8.82967
```

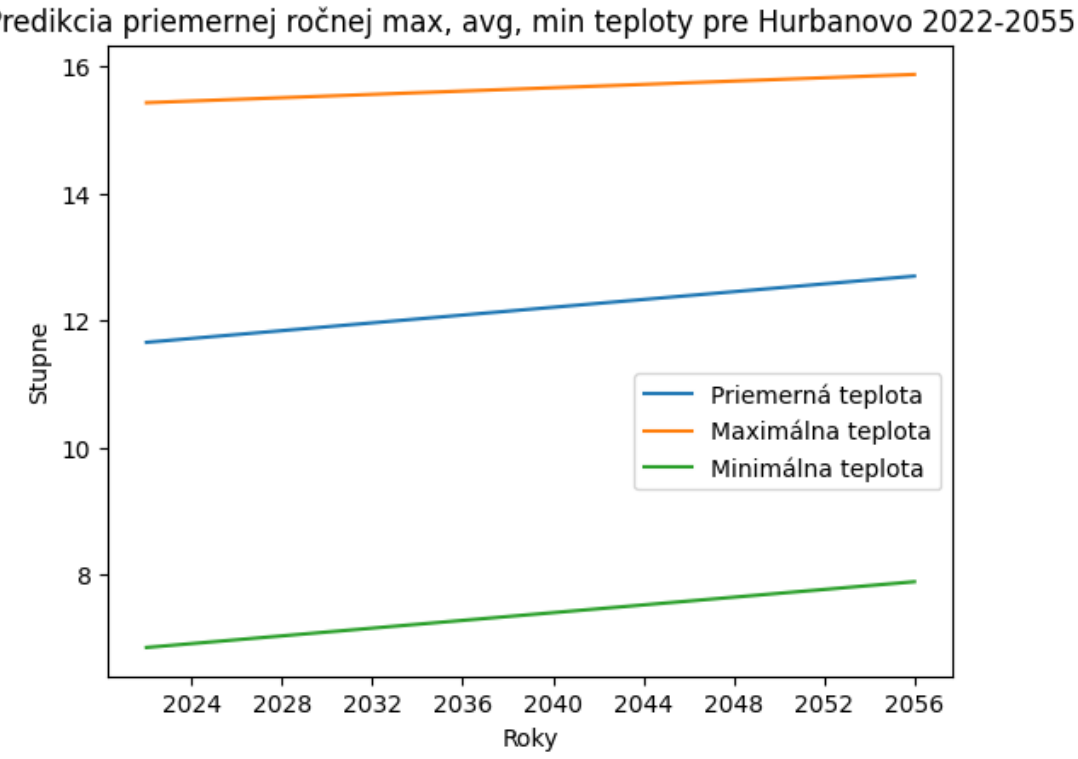
V nasledujúcom grafe sme sme zobrazili lineárny trend priemeru maximálnej, minimálnej a priemernej dennej teploty za roky 1951 až 2021 v Hurbanove. Na natrénovanom modeli sme predikovali testovacie dáta. Následne v grafe môžeme vidieť, že trend teploty je rastúci počas posledných 70 rokov.

[19]:
```
##linearny trend priemernych dennych teplot pocas rokov 1951-2021 s natrenovanym␣
 ↪modelom pre test data
plt.plot(X_test, lm.predict(X_test), label=['Priemerná teplota', 'Maximálna teplota',␣
 ↪'Minimálna teplota'])
plt.legend(loc = 'best', bbox_to_anchor = (1, 0.5))
plt.title("Lineárny trend priemeru max, min a avg dennej teploty pre Hurbanovo␣
 ↪1951-2021")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()
```



Následne sme predikovali, ako by mohol pokračovať rastúci trend teplôt pre roky 2022 až 2055 pre Hurbanovo. Môžeme vidieť, že trend globálneho otepľovania je reálny a bude sa diať aj v nasledujúcich rokoch. Do úvahy samozrejme neberieme CO2, energetickú krízu etc. Zaujímavé je ale vidieť, že priemerná minimálna teplota rastie rýchlejšie ako priemerná maximálna teplota, z čoho možno usúdiť, že za pár desaťročí nebude v južných oblastiach Slovenska (vrátane BA) žiaden sneh.

[20]:
```
##predikcia linearneho modelu pre roky 2022-55
roky=pd.date_range(start="2022-01-01",end="2055 -12-31")
roky=roky.values.reshape((len(roky), 1))
plt.plot(roky, lm.predict(roky),label=['Priemerná teplota', 'Maximálna teplota',␣
↪'Minimálna teplota'])
plt.legend(loc = 'best', bbox_to_anchor = (1, 0.5))
plt.title("Predikcia priemernej ročnej max, avg, min teploty pre Hurbanovo 2022-2055")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()
```



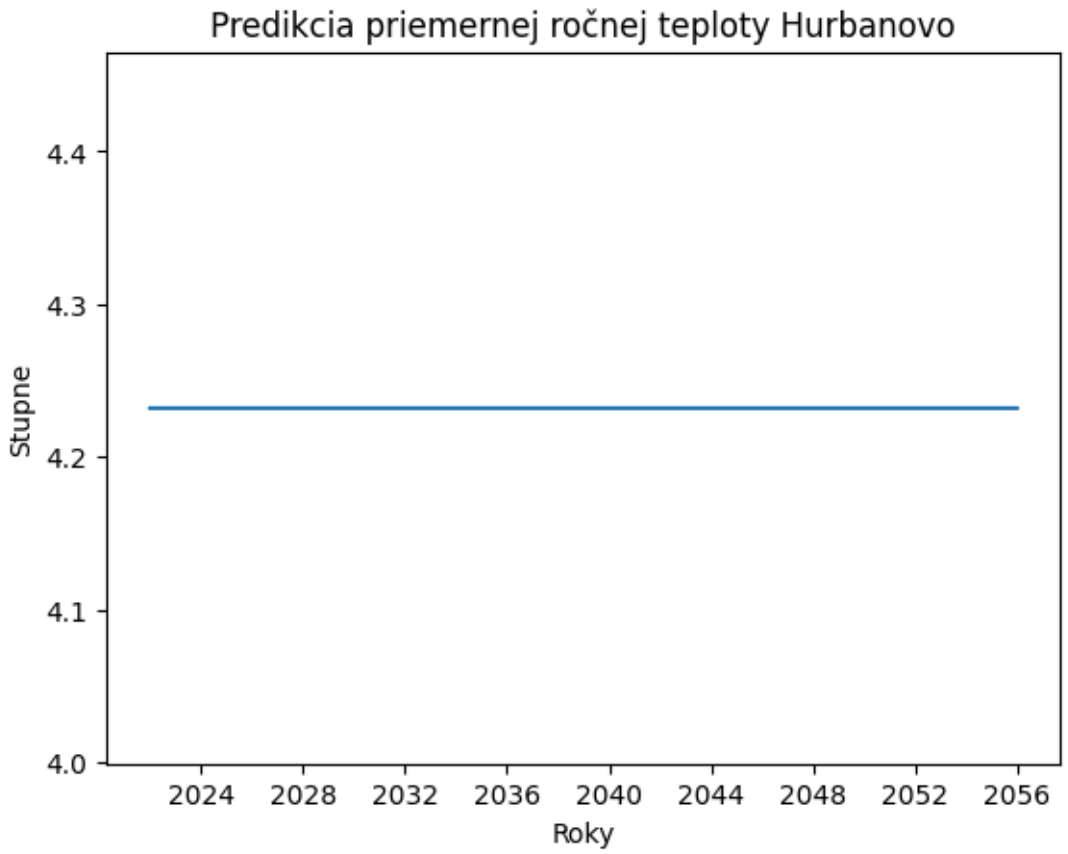
3.2 Random forest regressor

Tentokrát sme použili algoritmus random forest regressor. Ako nezávislú premennú sme určili **DATE** a ako závislú premennú len priemernú teplotu **HUR_MEAN_TMP** v Hurbanove. Aplikovali sme na random forest normalizáciu dát a nastavili sme forestu parametre. Skúšali sme ich viacero, no tieto nám dali najlepšie výsledky. Odchýlky už predsa vyšli lepšie, no stále to nie je ideálne. Na zobrazenom grafe môžeme vidieť, že predpovedať priemernú teplotu v Hurbanove len podľa dátumu bol omyl.

[21]:
```
##skusme to iste s random forestom
##nemozeme "predpovedat" 3 premenne naraz tak to spravime len s priemernou teplotou
Y1=df['HUR_MEAN_TMP']
X1=df[['DATE']]
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1,Y1, test_size = 0.25,␣
↪random_state = 42)

from sklearn.pipeline import make_pipeline
from sklearn.ensemble import RandomForestRegressor

forestModel = make_pipeline(
    StandardScaler(),
    RandomForestRegressor(
                            n_estimators = 1000,
                            max_depth = 500,
                            n_jobs = -1)
)

forestModel.fit(X_train1, y_train1)

from sklearn.metrics import mean_absolute_error, mean_squared_error
print("Mean absolute error forest modelu / trenovacie:
↪",round(mean_absolute_error(y_train1,forestModel.predict(X_train1)),5))
print("Mean absolute error forest modelu / testovacie:
↪",round(mean_absolute_error(y_test1,forestModel.predict(X_test1)),5))

print("Mean square error forest modelu / trenovacie:
↪",round(mean_squared_error(y_train1,forestModel.predict(X_train1)),5))
print("Mean square error forest modelu / testovacie:
↪",round(mean_squared_error(y_test1,forestModel.predict(X_test1)),5))

print("Root mean square error forest modelu / trenovacie:",round(math.
↪sqrt(mean_squared_error(y_train1,forestModel.predict(X_train1))),5))
print("Root mean square error forest modelu / testovacie:",round(math.
↪sqrt(mean_squared_error(y_test1,forestModel.predict(X_test1))),5))
```

```
Mean absolute error forest modelu / trenovacie: 0.6135
Mean absolute error forest modelu / testovacie: 1.65794
Mean square error forest modelu / trenovacie: 0.64732
Mean square error forest modelu / testovacie: 4.70242
Root mean square error forest modelu / trenovacie: 0.80456
Root mean square error forest modelu / testovacie: 2.16851
```

[22]:
```
## predikcia priemernej teploty 2022-2055
plt.plot(roky, forestModel.predict(roky))
plt.title("Predikcia priemernej ročnej teploty Hurbanovo")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()
##zle!
```



Teraz sa pozrieme na závislosť priemernej od maximálnej a minimálnej teploty v Hurbanove. Cez random forest natrénujeme model a vidíme odchýlky, ktoré sú tentokrát lepšie. Budeme teda predikovať priemernú teplotu na základe maximálnej a minimálnej teploty.

[23]:
```
##skusime zavislost priemernej od max/min teploty
###targetova/cielova/predikovana hodnota

Y2 = df['HUR_MEAN_TMP']
###prediktory
X2 = df[['HUR_MAX_TEMP', 'HUR_MIN_TEMP']]
##split ako minule
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2,Y2, test_size = 0.25,␣
↪random_state = 42)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

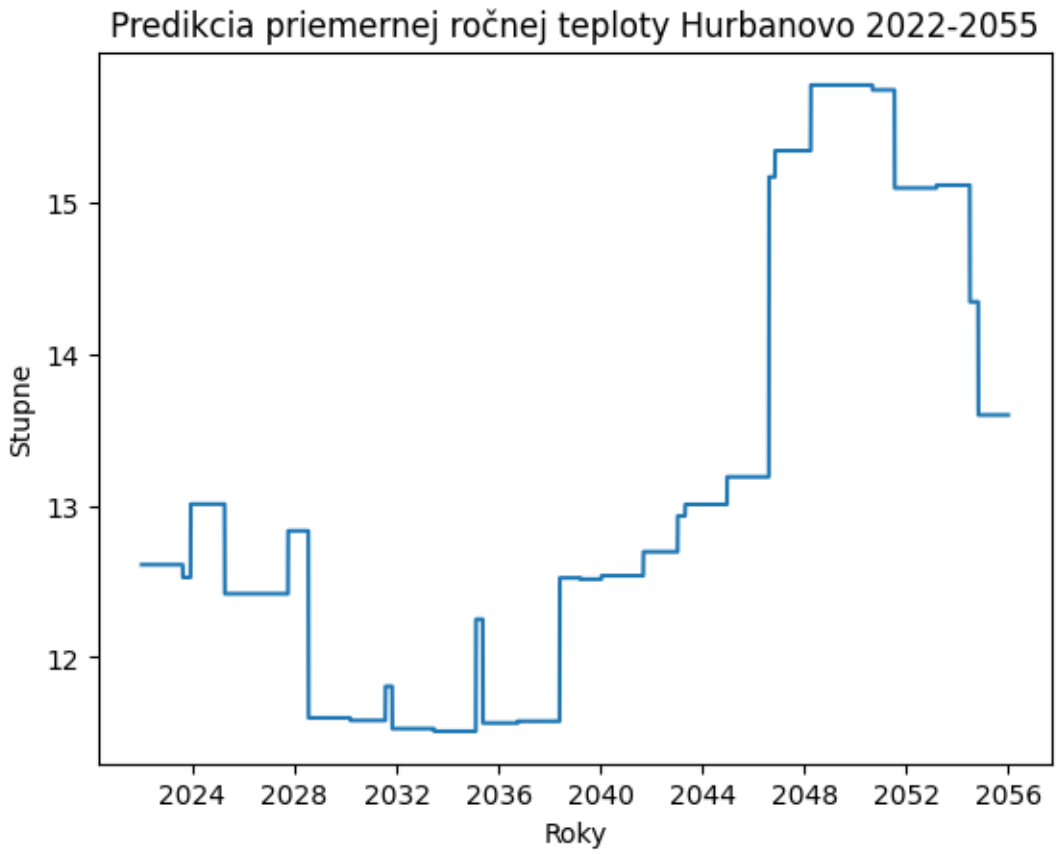
```
(19137, 1)
(6380, 1)
(19137, 3)
(6380, 3)
```

[24]:
```
forestModel2 = make_pipeline(
    StandardScaler(),
    RandomForestRegressor(
                            n_estimators = 1000,
                            max_depth = 500,
                            n_jobs = -1)
)

forestModel2.fit(X_train2, y_train2)

print("Mean absolute error forest modelu 2 / trenovacie:
 ↪",round(mean_absolute_error(y_train2,forestModel2.predict(X_train2)),5))
print("Mean absolute error forest modelu 2 / testovacie:
 ↪",round(mean_absolute_error(y_test2,forestModel2.predict(X_test2)),5))

print("Mean square error frest modelu 2 / trenovacie:
 ↪",round(mean_squared_error(y_train2,forestModel2.predict(X_train2)),5))
print("Mean square error forest modelu 2 / testovacie:
 ↪",round(mean_squared_error(y_test2,forestModel2.predict(X_test2)),5))

print("Root mean square error forest modelu 2 / trenovacie:",round(math.
 ↪sqrt(mean_squared_error(y_train2,forestModel2.predict(X_train2))),5))
print("Root mean square error forest modelu 2 / testovacie:",round(math.
 ↪sqrt(mean_squared_error(y_test2,forestModel2.predict(X_test2))),5))
```

```
Mean absolute error forest modelu 2 / trenovacie: 0.98047
Mean absolute error forest modelu 2 / testovacie: 2.21456
Mean square error frest modelu 2 / trenovacie: 1.67032
Mean square error forest modelu 2 / testovacie: 7.69057
Root mean square error forest modelu 2 / trenovacie: 1.29241
Root mean square error forest modelu 2 / testovacie: 2.77319
```

Pomocou lineárnej regresie sme predikovali maximálnu a minimálnu aj priemernú teplotu a výsledky lineárnej regresie použijeme, aby sme cez forest model vedeli predikovať priemernú ročnú teplotu pre Hurbanovo pre roky 2022-2055. Graf priemernej ročnej teploty pre Hurbanovo môžeme vidieť nižšie. Je však dosť rozbitý.

[25]:
```
linear_predikcia=lm.predict(roky)
forest_predikcia=forestModel2.predict(np.delete(linear_predikcia,0,axis=1))
plt.plot(roky, forest_predikcia)
plt.title("Predikcia priemernej ročnej teploty Hurbanovo 2022-2055")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()
plt.show()
```



Vyskúšajme iný prístup, a to zoskupiť si roky a z jednotlivých teplôt spraviť priemery pre každý rok.

[26]:
```
##iny uhol pohladu -> skusime rocne priemery
rocne=df.groupby(df.DATE.dt.year).mean()
Y3_HUR = rocne[['HUR_MEAN_TMP', 'HUR_MAX_TEMP', 'HUR_MIN_TEMP']]
Y3_KE = rocne[['KE_MEAN_TMP', 'KE_MAX_TEMP', 'KE_MIN_TEMP']]
Y3_PP = rocne[['PP_MEAN_TEMP', 'PP_MAX_TEMP', 'PP_MIN_TEMP']]
Y3_ORL = rocne[['ORL_MEAN_TEMP', 'ORL_MAX_TEMP', 'ORL_MIN_TEMP']]
#prediktory
X3=rocne.index
X3=X3.values.reshape((len(X3), 1))
X_train_HUR, X_test_HUR, y_train_HUR, y_test_HUR = train_test_split(X3,Y3_HUR,␣
↪test_size = 0.25, random_state = 42)
X_train_KE, X_test_KE, y_train_KE, y_test_KE = train_test_split(X3,Y3_KE, test_size =␣
↪0.25, random_state = 42)
X_train_PP, X_test_PP, y_train_PP, y_test_PP = train_test_split(X3,Y3_PP, test_size =␣
↪0.25, random_state = 42)
X_train_ORL, X_test_ORL, y_train_ORL, y_test_ORL = train_test_split(X3,Y3_ORL,␣
↪test_size = 0.25, random_state = 42)
rocne.head()
```

[26]:
```
      HUR_MEAN_TMP  KE_MEAN_TMP  ORL_MEAN_TEMP  PP_MEAN_TEMP  HUR_MIN_TEMP  \
DATE
```

```
1951         11.02         9.62          6.11          7.04          6.62
1952          9.85         8.47          4.30          5.47          5.01
1953          9.50         8.11          4.54          5.75          4.53
1954          9.07         7.71          3.96          5.33          4.43
1955          9.34         8.08          3.94          5.23          4.83

      KE_MIN_TEMP  ORL_MIN_TEMP  PP_MIN_TEMP  HUR_MAX_TEMP  KE_MAX_TEMP  \
DATE
1951         4.46          0.36         2.17         15.38        14.95
1952         3.16         -1.27         0.14         14.79        13.13
1953         2.82         -1.40         0.55         14.59        13.03
1954         2.55         -1.55         0.34         14.89        12.80
1955         3.26         -1.57         0.26         14.58        12.80

      ORL_MAX_TEMP  PP_MAX_TEMP
DATE
1951         12.05        12.21
1952          9.81        10.61
1953         10.96        11.02
1954         10.39        10.37
1955         10.04        10.36
```

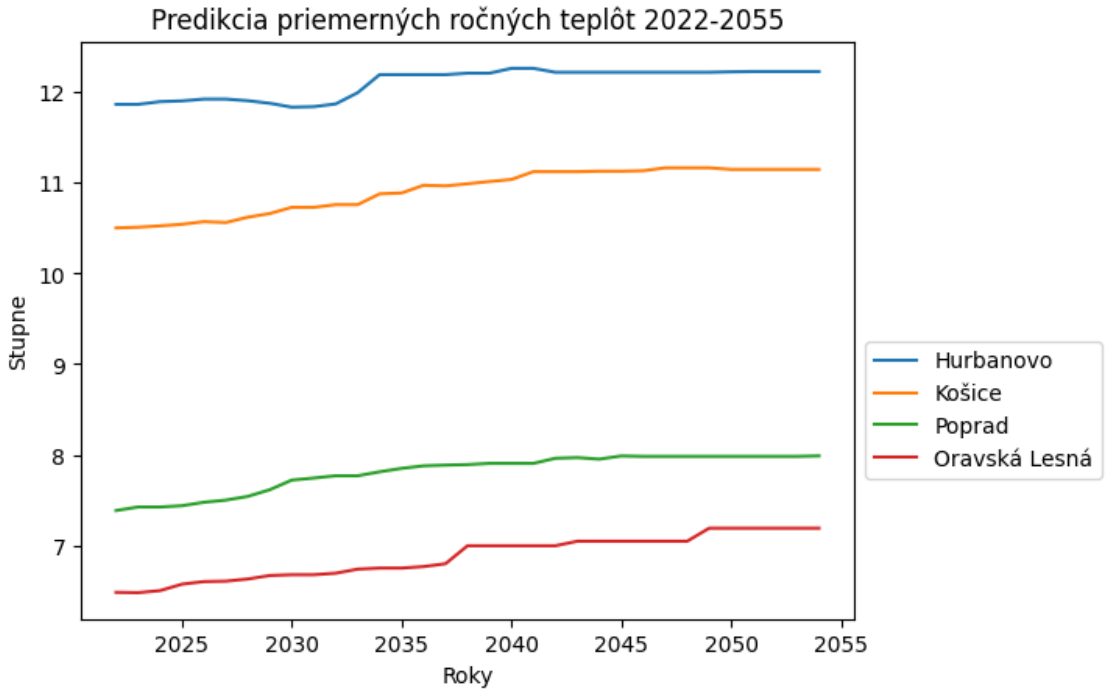
V nasledujúcom kóde robíme to isté, čo vyššie. Pre lepšiu ukážku to tu znova nakopírujem.

Pomocou lineárnej regresie sme predikovali maximálnu a minimálnu aj priemernú teplotu a výsledky lineárnej regresie použijeme, aby sme cez forest model vedeli predikovať priemernú ročnú teplotu pre všetky 4 mestá v rozmedzí rokov 2022-2055. Graf priemernej ročnej teploty pre všetky mestá vidieť nižšie. Na grafe môžeme vidieť, ako teplota pomaly, ale isto rastie aj v nasledujúcich desaťročiach.

[27]:
```
lm3_HUR=make_pipeline(
    StandardScaler(),
    LinearRegression(),
)

lm3_KE=make_pipeline(
    StandardScaler(),
    LinearRegression(),
)

lm3_PP=make_pipeline(
    StandardScaler(),
    LinearRegression(),
)

lm3_ORL=make_pipeline(
    StandardScaler(),
    LinearRegression(),
)

lm3_HUR.fit(X_train_HUR,y_train_HUR)
lm3_KE.fit(X_train_KE,y_train_KE)
lm3_PP.fit(X_train_PP,y_train_PP)
lm3_ORL.fit(X_train_ORL,y_train_ORL)
```

```
#predikovane hodnoty
Y4_HUR = rocne['HUR_MEAN_TMP']
Y4_KE = rocne['KE_MEAN_TMP']
Y4_PP = rocne['PP_MEAN_TEMP']
Y4_ORL = rocne['ORL_MEAN_TEMP']

###prediktory
X4_HUR = rocne[['HUR_MAX_TEMP', 'HUR_MIN_TEMP']]
X4_KE = rocne[['KE_MAX_TEMP', 'KE_MIN_TEMP']]
X4_PP = rocne[['PP_MAX_TEMP', 'PP_MIN_TEMP']]
X4_ORL = rocne[['ORL_MAX_TEMP', 'ORL_MIN_TEMP']]

##split ako minule
X_train4_HUR, X_test4_HUR, y_train4_HUR, y_test4_HUR = train_test_split(X4_HUR,Y4_HUR,␣
 ↪test_size = 0.25, random_state = 42)
X_train4_KE, X_test4_KE, y_train4_KE, y_test4_KE = train_test_split(X4_KE,Y4_KE,␣
 ↪test_size = 0.25, random_state = 42)
X_train4_PP, X_test4_PP, y_train4_PP, y_test4_PP = train_test_split(X4_PP,Y4_PP,␣
 ↪test_size = 0.25, random_state = 42)
X_train4_ORL, X_test4_ORL, y_train4_ORL, y_test4_ORL = train_test_split(X4_ORL,Y4_ORL,␣
 ↪test_size = 0.25, random_state = 42)

forestModel3_HUR = make_pipeline(
    StandardScaler(),
    RandomForestRegressor(
                            n_estimators = 1000,
                            max_depth = 500,
                            n_jobs = -1)
)

forestModel3_KE = make_pipeline(
    StandardScaler(),
    RandomForestRegressor(
                            n_estimators = 1000,
                            max_depth = 500,
                            n_jobs = -1)
)

forestModel3_PP = make_pipeline(
    StandardScaler(),
    RandomForestRegressor(
                            n_estimators = 1000,
                            max_depth = 500,
                            n_jobs = -1)
)

forestModel3_ORL = make_pipeline(
    StandardScaler(),
    RandomForestRegressor(
                            n_estimators = 1000,
                            max_depth = 500,
                            n_jobs = -1)
```

```
)
forestModel3_HUR.fit(X_train4_HUR, y_train4_HUR)
forestModel3_KE.fit(X_train4_KE, y_train4_KE)
forestModel3_PP.fit(X_train4_PP, y_train4_PP)
forestModel3_ORL.fit(X_train4_ORL, y_train4_ORL)

roky2=np.arange(2022,2055,1)
roky2=roky2.reshape(len(roky2),1)

linear2_HUR=lm3_HUR.predict(roky2)
linear2_KE=lm3_KE.predict(roky2)
linear2_PP=lm3_PP.predict(roky2)
linear2_ORL=lm3_ORL.predict(roky2)

plt.plot(roky2, forestModel3_HUR.predict(np.delete(linear2_HUR,0,axis=1)), label =␣
↪"Hurbanovo")
plt.plot(roky2, forestModel3_KE.predict(np.delete(linear2_KE,0,axis=1)), label =␣
↪"Košice")
plt.plot(roky2, forestModel3_PP.predict(np.delete(linear2_PP,0,axis=1)), label =␣
↪"Poprad")
plt.plot(roky2, forestModel3_ORL.predict(np.delete(linear2_ORL,0,axis=1)), label =␣
↪"Oravská Lesná")
plt.legend(loc = 'best', bbox_to_anchor = (1, 0.5))
plt.title("Predikcia priemerných ročných teplôt 2022-2055")
plt.xlabel("Roky")
plt.ylabel("Stupne")
plt.show()
```



Presnosť modelu zmeráme prostredníctvom R^2 metriky. Použijeme funkciu **.score**. .score dokumentacia .score behavior.

[28]:
```
print("R² train data Hurbanovo: ",forestModel3_HUR.score(X_train4_HUR, y_train4_HUR))
print("R² train data Košice: ",forestModel3_KE.score(X_train4_KE, y_train4_KE))
print("R² train data Poprad: ", forestModel3_PP.score(X_train4_PP, y_train4_PP))
print("R² train data Oravská Lesná: ", forestModel3_ORL.score(X_train4_ORL,␣
↪y_train4_ORL))
```

```
R² train data Hurbanovo:  0.9778732921560519
R² train data Košice:  0.9941799718526932
R² train data Poprad:  0.9938391060472845
R² train data Oravská Lesná:  0.982734936777464
```

[29]:
```
print("R² test data Hurbanovo: ",forestModel3_HUR.score(X_test4_HUR, y_test4_HUR))
print("R² test data Košice: ",forestModel3_KE.score(X_test4_KE, y_test4_KE))
print("R² test data Poprad: ", forestModel3_PP.score(X_test4_PP, y_test4_PP))
print("R² test data Oravská Lesná: ", forestModel3_ORL.score(X_test4_ORL, y_test4_ORL))
```

```
R² test data Hurbanovo:  0.8464309390734385
R² test data Košice:  0.9662277680364888
R² test data Poprad:  0.9574597738940757
R² test data Oravská Lesná:  0.8785886515551937
```