Razvoj aplikacije „Temp Prediction“

Babić, Ivan

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Ivan Babić

Razvoj aplikacije „Temp Prediction“

Završni rad

Pula, Rujan 2021.

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Ivan Babić

Razvoj aplikacije „Temp Prediction“

Završni rad

JMBAG: 0242004929
Studijski smjer: Preddiplomski studij Informatike

Predmet: Modeliranje i Simulacija
Znanstveno područje: Društvene Znanosti
Mentor: izv. Prof. dr. sc. Darko Etinger

Pula, Rujan 2021.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Babić Ivan kandidat za sveučilišnog prvostupnika informatike, ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student Ivan Bc

U Puli, 19.09.2020 godine

IZJAVA

o korištenju autorskog djela

Ja, Babić Ivan dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom Razvoj aplikacije " Temp Prediction " koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Sadržaj:

Table of Contents

Sažetak

U današnje doba klimatske promjene utječu na svačiji život te floru i faunu koja nas okružuje upravo iz tog razloga odlučio sam se iskušati u izradi jedne aplikacije koja nam omogućava predviđanje rasta temperature na temelju zabilježenih podataka.

Sama aplikacija sastoji se od skupa podataka u obliku .csv datoteke izvornog koda koji izrađuje model podataka putem strojnog učenja te korisničkog sučelja koji je izrađen u tkinteru.

Skup podataka ima prosječnu mjesečnu temperaturu od godina 1743. do 2013. Skup podataka ima 8599212 redaka i 7 kolumni, datum naziva kolumni, prosječnu temperaturu, prosječnu temperaturu nesigurnost, grad, državu, zemljopisnu širinu i dužinu. Postoji ukupno 3448 gradova svijeta i 159 zemalja

Model podataka koji je izrađen strojnim učenjem točnije algoritmom Random forest regression

Te korisničkim sučeljem koji je izrađen u tkinteru gdje korisnik može dati naziv grada, naziv zemlje, mjesec i godinu za predviđanje temperature

Motivacija

Klimatske promjene utječu na čitavo naše okruženje, na nas kao što i na našu okolinu povezane su za niz elementarnih nepogoda, pojavom novih vrsta bolesti, te gljivičnih te oboljenja. Sve su to pojave koje su povezane naglim porastom temperatura, te tim putem utječu na floru, faunu pa čak i ekonomiju.

Želio sam izraditi ovaj projekt jer bavio sam se poljoprivredom dugi niz godina te iz godine u godinu primjećivao sam štete na usjevima koje su posljedica visokim temperaturama postaju sve češća pojava kao što su se češće počele pojavljivat i druge elementarne pogode, te sam se već tada krenuo raspitivat u kojoj mjeri je porast temperatura utječe na učestalost tih pojava, iako nisam nikakav stručnjak tog područja po mom skromnom mišljenju siguran sam da naprosto mora postojati nekakva korelacija sa time što se događa u prirodi i naglim porastom prosječnih temperatura.

Upravo te promjene koje viđam u okolišu su me potaknule na izradu ove aplikacije jer želim biti u mogućnosti prognozirati prosječnu mjesečnu temperaturu te vidjeti vlastitim očima koliko taj porast utječe na naš okoliš.

Iako u ova aplikacija kao takva nije u mogućnosti mjeriti izravan utjecaj temperature na prirodu, ona je u stanju prognozirati porast temperature na temelju zabilježenih podataka.

1.Time series data preparation

Podaci vremenskih serija mogu se izraziti kao nadzirano učenje. S obzirom na niz brojeva za skup vremenskih serija, možemo restrukturirati podatke tako da izgledaju kao problem nadziranog učenja.

To možemo učiniti korištenjem prethodnih vremenskih koraka kao ulaznih varijabli, a sljedeći vremenski korak kao izlaznu varijablu. Ovaj skup vremenskih serija možemo restrukturirati kao nadzirani problem učenja korištenjem vrijednosti u prethodnom vremenskom koraku za predviđanje vrijednosti na sljedeći vremenski korak.

Možemo koristiti funkciju shift () u Pandas-u za automatsko stvaranje novih okvira problema vremenskih serija s obzirom na željenu duljinu ulaznih i izlaznih sekvenci. Donja funkcija će uzeti vremensku seriju kao vremenski niz NumPy polja s jednom ili više stupaca i pretvoriti ga u nadzirani problem učenja s navedenim brojem ulaza i izlaza

1.1.Time Series analiza

Time series analysis predstavljaju niz promatranja snimljenih u pravilnim vremenskim intervalima.

Predviđanje vremenskih serija ima ogroman komercijalni značaj jer po potrebi može predstavljati elemente koje su važne za poslovanje poput potražnje i prodaje, broja posjetitelja web stranice, cijene dionica itd.

Ovisno o učestalosti promatranja, vremenski niz obično može biti satni, dnevni, tjedni, mjesečni, tromjesečni i godišnji. Ponekad možete imati sekunde i vremenske serije po minuti, primjerice, broj klikova i korisničke posjete svake minute itd.

Vremenski niz je potrebno analizirati zato što je to pripremni korak prije nego što krenemo u razvijanje prognozu serije.

Time series analysis uključuje razumijevanje različitih aspekata o prirođenoj prirodi serije, tako da ste bolje informirani o stvaranju smislenih i točnih prognoza. Budući da u ovoj aplikaciji bavimo se svjetskom temperaturom, uzimamo je u obzir kao niz stacionarnih podataka.

Transformacija podataka: Naš skup podataka ima kategorijske stupce poput naziva grada i zemlje, encoder oznaka koristi se za pretvaranje kategorijskih obilježja u matematičke brojeve.

2.Random Forest

Random Forest popularan je i učinkovit algoritam strojnog učenja.

Široko se koristi za klasifikacijske i regresijske probleme s prediktivnim modeliranjem sa strukturiranim (tabličnim) skupovima podataka, npr. podataka kako izgledaju u proračunskoj tablici ili tablici baze podataka.

Slučajna šuma može se koristiti i za predviđanje vremenskih serija, iako zahtijeva da se skup vremenskih serija prvo transformira u nadzirani problem učenja. Također zahtijeva upotrebu specijalizirane tehnike za procjenu modela koja se naziva validacija hodom naprijed.
Ansambl slučajnih šuma
Slučajna šuma je skup algoritama stabla odlučivanja.

To je proširenje bootstrap agregacije (vrećice) stabala odlučivanja i može se koristiti za probleme klasifikacije i regresije.

U pakiranju se stvara niz stabala odluka gdje se svako stablo stvara iz različitog početnog uzorka skupa podataka za obuku. Bootstrap uzorak je uzorak skupa podataka za obuku gdje se primjer može pojaviti više puta u uzorku. To se naziva "uzorkovanje sa zamjenom".

Spakiranje je učinkovit algoritam ansambla jer se svako stablo odlučivanja uklapa u malo drugačiji skup podataka za obuku, a zauzvrat ima nešto drugačije performanse. Za razliku od normalnih modela stabla odlučivanja, poput stabala klasifikacije i regresije (CART), stabla koja se koriste u ansamblu nisu obrezana, što ih čini pomalo prilagođenim skupu podataka za obuku. To je poželjno jer pomaže učiniti svako stablo drugačijim i imati manje korelirana predviđanja ili pogreške predviđanja.

Predviđanja iz stabala su prosječna za sva stabla odlučivanja, što rezultira boljim performansama od bilo kojeg pojedinačnog stabla u modelu.

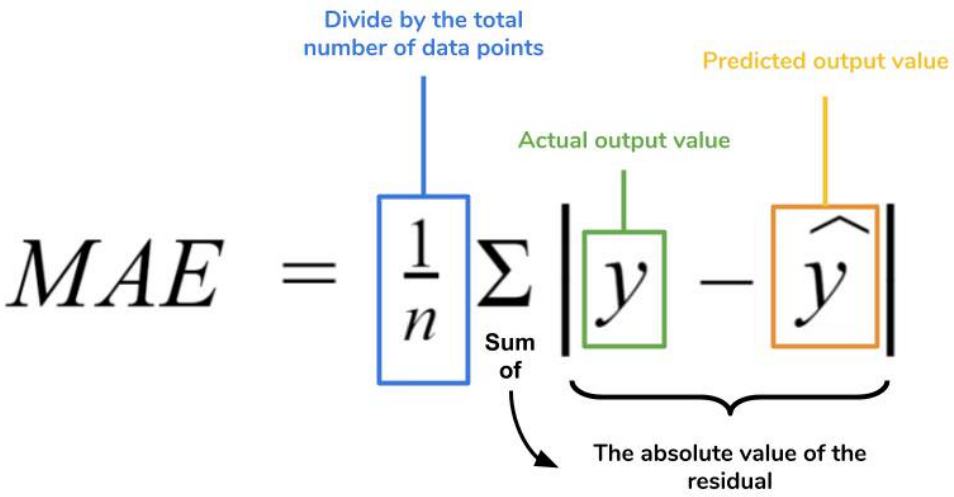
Predviđanje za problem regresije je prosjek predviđanja za stabla u ansamblu. Predviđanje o klasifikacijskom problemu je većina glasova za oznaku razreda preko stabala u ansamblu.

Regresija: Predviđanje je prosječno predviđanje na stablima odluka.
Klasifikacija: Predviđanje je oznaka klase većine glasova predviđena u stablima odluka.
Slučajna šuma uključuje izgradnju velikog broja stabala odluka iz uzoraka početnog programa iz skupa podataka za obuku, poput vrećica.

Učinak je da su predviđanja, a zauzvrat i pogreške predviđanja, napravljena od svakog stabla u ansamblu više različita ili manje povezana. Kada se predviđanja iz ovih manje korelirani stabala prosječno izrade za predviđanje, to često rezultira boljim učinkom od stabala odluka u vrećicama.

Kad koristimo algoritam slučajnih šuma za rješavanje regresijskih problema, koristimo srednju apsolutnu pogrešku (**M**ean **A**bsolute **E**rror) kako bi se vaši podaci granali sa svakog čvora.

1.



[1] *Izvor: https://imgur.com/19LNbyQ*

3.Razumijevajući serije za nadziranje

S obzirom na niz brojeva za skup vremenskih serija, možemo restrukturirati podatke tako da izgledaju kao nadzirani problem učenja. To možemo učiniti korištenjem prethodnih vremenskih koraka kao ulaznih varijabli, a sljedeći vremenski korak koristiti kao izlaznu varijablu.

Ova funkcija u biti stvara nove varijable uvođenjem zaostajanja u izvorne serije, mi trenutno imamo 2 ulazne varijable 'prosječna mjesečna temperatura' i 'grad-zemlja' (uzimajući datum kao indeks) također n_in = 2 i n_out = 1 znači, 2 koraci u prošlom vremenu koriste se za predviđanje 1 korak ispred.

Pomoću ove gornje metode možemo samo prognozirati u ograničenom vremenskom razdoblju, recimo do 2020. ili 2025. Na primjer, želimo napraviti predviđanje za neki mjesec 2020. godine, što je 7 godina unaprijed. za to nam je potrebno najmanje 7 godina prošlih podataka za obuku. Štoviše, bit će 7x12 = 84 varijable što će uzrokovati velike veličine modela i imati velike računalne troškove.

Druga metoda daje fleksibilnost za predviđanje beskonačne budućnosti. Vježbamo na 4 - godišnjim podacima s n_in = 2 i n_out = 1 radeći samo predviđanje u jednom vremenskom koraku. Sada korisnik daje samo 2 unosa Datum za koji je potrebno prognozirati te grad i državu

Sada nam je potrebna ova metoda za izvođenje, ne možemo proslijediti jedan uzorak u series_to_supervised () jer će ga pretvoriti u NAN radeći pomak (zaostajanje) pa nam je potrebno nekoliko uzoraka kako bismo izbjegli problem s NAN -om, pa uzimamo 36 uzoraka iz izvornih podataka.

[2]

```
# transform a time series dataset into a supervised learning dataset
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = pd.DataFrame(data)
    cols = list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
    # put it all together
    agg = pd.concat(cols, axis=1)
    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg.values
```

[2] *Izvor: https://github.com/ivbabic/Temp_predict*

4.Filtriranje podataka

3

```
# load the dataset
series = pd.read_csv('/content/drive/MyDrive/weather_dataset/GlobalLandTemperaturesByCity.csv', header=0, index_col=0)
```

Nakon što smo podigli csv datoteku u idućem koraku izvršavamo pretvaranje svih kategoričnih naziva u mala slova i ispuštanje svih vrijednosti koje nedostaju ako su dostupne u skupu podataka.

4

```
series['City_Country'] = (series['City']+series['Country']).str.lower()
series['City'] = series['City'].str.lower()
series['Country'] = series['Country'].str.lower()
series.dropna(inplace=True)
d = series
```

U idućem koraku filtriramo set podataka po državama koje ćemo upotrijebiti u modelu, ti koraci su bolje poznati pod nazivom čišćenje podataka, što je nužan korak kako bi naš krajnji model podataka na bio što ispravniji te precizniji u svojim ishodima.

Postupak čišćenja podataka se sastoji od otklanjanja duplih vrijednosti, ispravak strukturalnih grešaka te otklanjanje ili popunjavanje praznih vrijednosti.

5

```
combined_data = pd.DataFrame()
for cont in country:
  data = series[series['Country']==cont]

  combined_data = combined_data.append(data)

combined_data
```

dt	AverageTemperature	AverageTemperatureUncertainty	City	Country	Latitude	Longitude
1743-11-01	7.478	1.866	aix en provence	france	44.20N	4.47E
1744-04-01	11.596	2.044	aix en provence	france	44.20N	4.47E
1744-05-01	13.287	1.791	aix en provence	france	44.20N	4.47E
1744-06-01	17.675	1.733	aix en provence	france	44.20N	4.47E
1744-07-01	20.056	1.825	aix en provence	france	44.20N	4.47E

[3] *Izvor: Izvor: https://github.com/ivbabic/Temp_predict*
[4] *Izvor: Izvor: https://github.com/ivbabic/Temp_predict*
[5] *Izvor: Izvor: https://github.com/ivbabic/Temp_predict*

U našem slučaju mi filtriramo podatke prema onim zemljama nad kojim želimo izvršiti prognozu, preciznije Francuska, Italija, Mađarska, Austrija, Švicarska, Hrvatska, Bosna i Hercegovina, Srbija, Rumunjska i Grčka i određeni niz godina jer inače model podataka koji dobijemo postane prevelik za RAM memoriju računala zatim otklanjamo one podatke koji nam nisu korisni i nužni za izvršiti prognozu.

6

```
series = series[series.index>="2000-01-01"] ### taking only past 13 years data
last_month = series[series.index>="2013-07-01"]
# droping "City","Country",,'AverageTemperatureUncertainty','Latitude','Longitude' features
series.drop(['City','Country','AverageTemperatureUncertainty','Latitude','Longitude'], axis = 1, inplace = True)
le = LabelEncoder()
# considering only one feature "City_Country", and tranforming categociral feature using label encoder
series['City_Country'] = le.fit_transform(series['City_Country'])
print(series.tail())
values = series.values
```

Nakon što smo filtrirali podatke ubacujemo ih u nadzirano učenje.

7

```
# transform the time series data into supervised learning
data = series_to_supervised(values, n_in=2)
```

[6] *Izvor: https://github.com/ivbabic/Temp_predict*
[7] *Izvor: https://github.com/ivbabic/Temp_predict*

5.Razdvajanje podatka u skup za treniranje te skup za testiranje

Razdvajanje podatka u skup za treniranje te skup za testiranje je proces kojim dijelimo podatke u dva različita univarijatna skupa, skup za treniranje te skup za testiranje.

Te kao što i samo ime kaže u skupu za treniranje podataka treniramo podatke, dok u skupu za testiranje ih testiramo.

Svrha Razdvajanja podataka u 2 skupine je kako bi se izbjegao overfitting[11] podataka te što više poboljšala preciznost predviđanja.

8

```
# split dataset into train/test sets
def train_test_split(data, n_test):
    return data[:-n_test, :], data[-n_test:, :]
```

9

```
train, test = train_test_split(data, last_month.shape[0])
trainX, trainy = train[:, 1:], train[:, 0]
testX, testy = test[:, 1:], test[:, 0]
```

6.Model fitting

Uklapanje modela označava onaj dio procesa u strojnom učenju u kojem odabrani model učenja generalizira podatke slične onima kojima je obučen.

Uklapanje modela odnosi se na podešavanje parametra unutar modela radi poboljšanja preciznosti u predviđanju, te uključuje procjenu parametra koja na bolje odgovara postojećem skupu podataka, a zatim rješenja uklapa u model podataka.

10

```
model = RandomForestRegressor(n_estimators=1000)
model.fit(trainX, trainy)
yhat = model.predict(testX)
```

U prikazanom kodu izvršavamo RandomForestRegressor i uklapamo model nad kojim je izvršen algoritam.

[8] *Izvor: https://github.com/ivbabic/Temp_predict*
[9] *Izvor: https://github.com/ivbabic/Temp_predict*
[10] *Izvor: https://github.com/ivbabic/Temp_predict*

Nakon što smo izvršili sve potrebne radnje te napon dobili model podataka koji ćemo koristiti za prognoziranje spašavamo model u zasebnu datoteku te ga podižemo po potrebi.
[11]

```
import pickle

saved_model = pickle.dump(model, open("/content/drive/MyDrive/weather_dataset/europian_country_few.pickle.dat", "wb"))

#loading save model
rf_saved_model = pickle.load(open("/content/drive/MyDrive/weather_dataset/europian_country_few.pickle.dat", "rb"))
```

[12]

```
print("Please choose among these country and city only for temperature prediction")
for cont in country:
  if cont in d['Country'].unique():
    print(cont, d[d['Country']==cont]['City'].unique())
    print()
```

U donjoj slici imamo prikaz filtriranih gradova te država koji su prisutni u modelu.

[13]

```
austria ['graz' 'innsbruck' 'linz' 'salzburg' 'vienna']

switzerland ['basel' 'bern' 'geneva' 'lausanne' 'zurich']

croatia ['rijeka' 'split' 'zagreb']

bosnia and herzegovina ['banja luka' 'mostar' 'sarajevo' 'tuzla' 'zenica']

serbia ['belgrade' 'kragujevac' 'nis' 'novi sad' 'pristina' 'prizren']

romania ['arad' 'bacau' 'baia mare' 'botosani' 'braila' 'brasov' 'bucharest'
 'buzau' 'cluj napoca' 'constanta' 'craiova' 'drobeta turnu severin'
 'focsani' 'galati' 'iasi' 'oradea' 'piatra neamt' 'pitesti' 'ploiesti'
 'ramnicu valcea' 'satu mare' 'sibiu' 'suceava' 'targu mures' 'timisoara']

greece ['athens' 'iráklion' 'kallithéa' 'lárisa' 'pátrai' 'peristérion'
 'thessaloníki']
```

[11] *Izvor: https://github.com/ivbabic/Temp_predict*
[12] *Izvor: https://github.com/ivbabic/Temp_predict*
[13] *Izvor: https://github.com/ivbabic/Temp_predict*

7.Input podaci

Da bi aplikacija odradila nekakvu prognozu zahtijeva nekoliko ulaznih podataka ili inputa sa strane korisnika a ti podaci su sljedeći naziv grada, naziv države, godina te mjesec.

14

```
######## INPUT from USER ####################
#model only train on dataset from 2009 to 2013
# for temperature forcasting we should pass month as year later than 2013
# month should be two digits number,  for example : 12, 11, 01, 05 etc.
city = input("Enter the name of the City name :").lower()
country = input( "Enter the name of Country : ").lower()
year = input("Enter the year :")
month = input( "Enter the month : ")
y = int(year)
m = int(month)
```

```
Enter the name of the City name :rijeka
Enter the name of Country : croatia
Enter the year :2021
Enter the month : 09
```

Na temelju unesenih podataka te podataka prijašnjih i idućih godina stvaramo indeks podataka koji koristimo u svrhe nadzirano učenja. Ovaj postupak nam je bitan kako bismo u drugom trenutku zaključili stupanj odstupanja našeg modela.

15.1

```
###################### creating INDEX for input month and year
###################### extracting previous as well as next years data for supervised learning purpose
#test = pd.DataFrame(columns  =['dt','AverageTemperature','City_Country'])
test = pd.DataFrame()
flag = 0
if y<=2012:
  ind = year+'-'+month+'-01'
  test = pd.DataFrame()
  for year in (y-1,y,y+1):
    year = str(year)
    for i in range(1,13):
      if i<10:
        date = year+'-0'+str(i)+'-01'
      else:
        date = year+'-'+str(i)+'-01'
      same_dates = d.loc[date]
      # intermediate = same_dates[(same_dates['City']==city) & (same_dates['Country']==country)]
      # if intermediate.shape[0]>1:
      #   print(intermediate.groupby('dt').mean())
      test = test.append(same_dates[(same_dates['City']==city) & (same_dates['Country']==country)],ignore_index=True)
```

14 *Izvor:* https://github.com/ivbabic/Temp_predict

15.1 *Izvor:* https://github.com/ivbabic/Temp_predict

15.2

```
else:
  flag = 1
  ind = year+'-'+month+'-01'
  for year in range(2013,y+2):
    year_art = str(2012-(year-2013))
    year = str(year)

    for i in range(1,13):
      if i<10:
        date = year+'-0'+str(i)+'-01'
        date_art = year_art+'-0'+str(i)+'-01'

      else:
        date = year+'-'+str(i)+'-01'
        date_art = year_art+'-'+str(i)+'-01'

      same_dates = d.loc[date_art]
      test = test.append(same_dates[(same_dates['City']==city) & (same_dates['Country']==country)],ignore_index=True)
      #test = test.append({'dt': date, 'AverageTemperature': 0, 'City_Country': city+country},ignore_index=True)
  #test.set_index('dt', inplace=True)
test.drop(['City','Country','AverageTemperatureUncertainty','Latitude','Longitude'], axis = 1, inplace = True)
#test.drop_duplicates(inplace = True)
print(test)
```

U idućem koraku još jednom izvršavamo pretvaranje kategorijskih značajki unesenih vrijednosti u matematičku veličinu pomoću label kodera koji dodaje variabilama numeričke veličine, dobivene vrijednosti prebacujemo u serije za nadziranje te izvršavamo treniranje.

16

```
test['City_Country'] = le.fit_transform(test['City_Country'])
values = test.values
xtest = series_to_supervised(values,n_in=2)
testX, testy = xtest[:, 1:], xtest[:, 0]
y_pred = rf_saved_model.predict(testX)
```

15.2 *Izvor:* https://github.com/ivbabic/Temp_predict

16 *Izvor:* https://github.com/ivbabic/Temp_predict

Nakon što smo imamo sve potrebne vrijednosti unesene sa strane korisnika aplikacije iznosi prognozu prosječne mjesečnu temperature iz modela podataka, te uspoređuje sa realnom temperaturom i mjeri odstupanje od prognoziranog.

17

```
if flag==0:
  from sklearn.metrics import mean_absolute_error
  print("Predicted Monthly Temperature of"+ind+" is {:.3f}".format(y_pred[m+11]))
  print("Actual Monthly Temperature of"+ind+" is {:.3f}".format(testy[m+11]))
  print("Mean absolute Error of entire year is {:.3f} :".format(mean_absolute_error(testy, y_pred)))
  # plot expected vs predicted
  pyplot.plot(testy, label='Expected')
  pyplot.plot(y_pred, label='Predicted')
  pyplot.legend()
  pyplot.show()
else:
  print("Predicted Monthly Temperature of"+ind+" is {:.3f}".format(y_pred[m+11]))
```

```
Predicted Monthly Temperature of2021-09-01 is 17.015
```

[17] *Izvor: Izvor: https://github.com/ivbabic/Temp_predict*

8.Korisničko sučelje

U ovom projektu za izradu korisničkog sučelja koristili smo Tkinter što je jedna od najčešće korištenih knjižnica za izradu korisničkog sučelja u pythonu. Kako su Tk i Tkinter dostupni na većini Unix platformi, kao i na Windows sustavu, razvoj GUI aplikacija s Tkinterom postaje brži i jednostavniji.

Kako bismo koristili naš model kroz tkinter najprije moramo podignuti model koji smo ranije spasili.

18

```
from tkinter import *
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pickle

#Loading save model
rf_saved_model = pickle.load(open("europian_country_few.pickle.dat", "rb"))
```

19

```
root = Tk()
root.configure(background='#6CA6AD')

def getvals():
    print(f"City Name: {City_name_v.get()}\nCountry name: {Country_name_v.get()}\nDate: {Date_v.get()} ")

def clear():
    City_name_entry.delete(0, END)
    Country_name_entry.delete(0, END)
    Year_entry.delete(0, END)
    Month_entry.delete(0, END)
    Temp.delete(0, END)
    return None
```

U idućem koraku podižemo csv datoteku koju ćemo koristiti u svrhe nadzirano učenja, odnosno u svrhu uspoređivanja podatka sa onima koji su istrenirani u modelu.

20

```
series = pd.read_csv('GlobalLandTemperaturesByCity.csv',
                     header=0, index_col=0)
series['City_Country'] = (series['City'] + series['Country']).str.lower()
series['City'] = series['City'].str.lower()
series['Country'] = series['Country'].str.lower()
series.dropna(inplace=True)
d = series
```

[18] *Izvor: https://github.com/ivbabic/Temp_predict*
[19] *Izvor: https://github.com/ivbabic/Temp_predict*
[20] *Izvor: https://github.com/ivbabic/Temp_predict*

Korisničko sučelje sastavljeno je od relativno jednostavnog koda te nije ništa previše zahtjevno, sastoji se od dva dijela odnosno prethodno navedenih variabili koje korisnik je dužan unijeti te datoteke iz koje vučemo dodatne vrijednosti potrebne za stvaranje indeksa podataka.

21

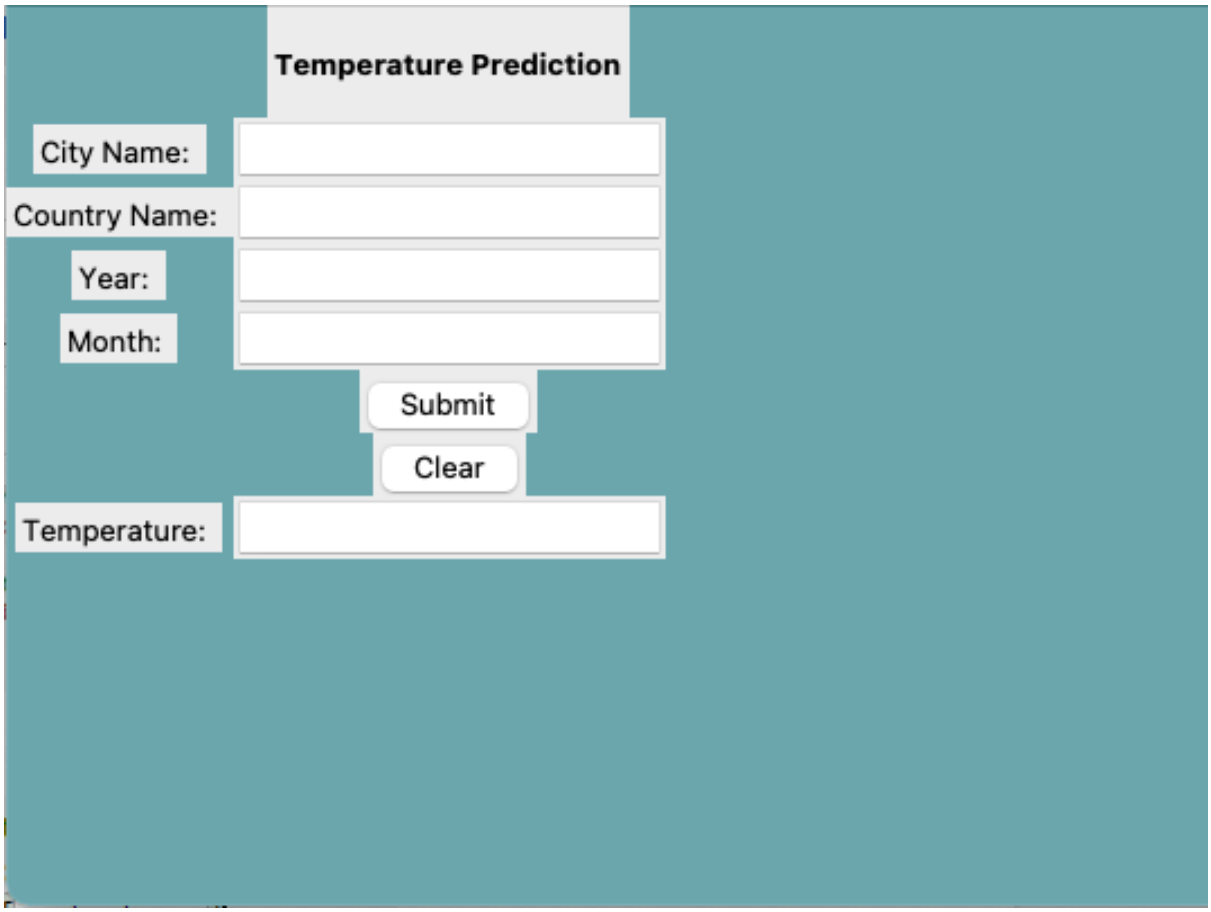
```
root.geometry("800x400")
Label(root, text="Temperature Prediction", pady=15, font="Century 13 bold").grid(row=0, column=3)

City_name = Label(root, text="City Name: ")
Country_name = Label(root, text="Country Name: ")
Year_name = Label(root, text="Year: ")
Month_name = Label(root, text="Month: ")
Temp_label = Label(root, text="Temperature: ")

City_name.grid(row=1, column=2)
Country_name.grid(row=2, column=2)
Year_name.grid(row=3, column=2)
Month_name.grid(row=4, column=2)
Temp_label.grid(row=7, column=2)

City_name_v = StringVar()
Country_name_v = StringVar()
Year_v = StringVar()
Month_v = StringVar()
Temp_v = StringVar()
```

22

```
City_name_entry = Entry(root, textvariable=City_name_v)
Country_name_entry = Entry(root, textvariable=Country_name_v)
Year_entry = Entry(root, textvariable=Year_v)
Month_entry = Entry(root, textvariable=Month_v)

Temp = Entry(root, textvariable=Temp_v).grid(row=7, column=3)

City_name_entry.grid(row=1, column=3)
Country_name_entry.grid(row=2, column=3)
Year_entry.grid(row=3, column=3)
Month_entry.grid(row=4, column=3)
Button(text="Submit", command=process).grid(row=5, column=3)
Button(text="Clear", command=clear).grid(row=6, column=3)

root.mainloop()
```

[21] *Izvor:* https://github.com/ivbabic/Temp_predict
[22] *Izvor:* https://github.com/ivbabic/Temp_predict

23



[23] *Izvor:*https://github.com/ivbabic/Temp_predict

Zaključak

U današnje doba klimatske promjene utječu na čitavo naše okruženje, na nas kao što i na našu okolinu povezane su za niz elementarnih nepogoda te utječu na nas iako zbog trenutne situacije sa Covid-19 malo smo zapustili ovu tematiku smatram da kao društvo trebali bismo se više pozabaviti tim specifičnim problemom, i što prije pronaći prihvatljivo rješenje.

Algoritmi za strojno učenje te umjetna inteligencija nam mogu doista pomoći predvidjeti posljedice naših odluka kao društvo te nam ukazati alternativan i možda prihvatljiviji smjer.

Kod aplikacije dostupan je na sljedećem linku: https://github.com/ivbabic/Temp_predict

Popis slika:

Popis literature:

1. Salvador, José & Oliveira, João & Breternitz, Mauricio. (2020). REINFORCEMENT LEARNING: A LITERATURE REVIEW (September 2020). 10.13140/RG.2.2.30323.76327.
2. A. Singh, N. Thakur and A. Sharma, "A review of supervised machine learning algorithms," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 1310-1315.
3. Introduction to Machine Learning with Python: A Guide for Data Scientists Paperback – Nov. 1 2016, Andreas C. Müller, Sarah Guido.
4. Machine Learning For Absolute Beginners: A Plain English Introduction *Oliver Theobold (2017)*
5. Feature Engineering and Selection: A Practical Approach for Predictive Models (Chapman & Hall/CRC Data Science Series) 1st Edition

by Max Kuhn (Author), Kjell Johnson (Author)

6. Practical Time Series Analysis Prediction with Statistics and Machine Learning, Aileen Nielsen 2020