



Data Glacier

Your Deep Learning Partner

Deployment of a Machine Learning Model using Flask on Heroku

Alireza Ehiaei

11-July-2021

TABLE OF CONTENTS

01	Introduction
02	Dataset
03	Preparing data
04	Autoregression model
05	Deployment the model using Flask
06	Deployment the model on Heroku

Introduction

In this research, a times series auto regression model for estimating the population of a country or region in a chosen time period.

First, The data preparing process is mentioned, then the model is introduced, flask app and its deployment on Heroku are provided.

The goal of this research is to get the experience of deployment of a Flask app on Heroku.

Dataset

- Data set includes data of population, both sexes, of regions, subregions, countries or areas around the world from 1950 to 2020 including 289 rows.
- The data source is:
<https://population.un.org/wpp/Download/Standard/Population/>.
- Data and code are uploaded at: https://github.com/Alireza-Ehiaei/Data_Sciences/tree/main/Machine_learning/Deploy_on_Heroku.
- The web app created in this project:
<https://estimation-ml.herokuapp.com/>

Process

First, we have a block of codes for downloading and preparing data, then some codes for the auto regression model, and the creating Flask app.

Finally, the steps of deploying the app on Heroku using git is explained.

Preparing data

In order to be able to run auto regression model we need time series data format preferably by indexing time, so that each column has population of one country. So, a function with name 'panel' is defined to convert data from their initial format into the desired format. The code is in next page and converts the initial format below:

Index	Variant	Region, subregion	Notes	Country code	Type	Parent code	1950	1951	1952	1953	1954
1	Estimates	WORLD		900	World	0	2 536 431	2 584 034	2 630 862	2 677 609	2 724 841
2	Estimates	UN developed areas		1803	Label/Separate	900
3	Estimates	More developed areas		901	Developments	1803	814 819	824 004	833 720	843 788	854 060
4	Estimates	Less developed areas		902	Developments	1803	1 721 612	1 760 031	1 797 142	1 833 822	1 870 781
5	Estimates	Least developed countries		941	Developments	902	195 428	199 180	203 015	206 986	211 133
6	Estimates	Less developed countries		934	Developments	902	1 526 184	1 560 850	1 594 126	1 626 836	1 659 651
7	Estimates	Less developed regions, excluding		948	Developments	1803	1 157 420	1 179 933	1 203 963	1 229 440	1 256 301
8	Estimates	Land-locked developing		1636	Special other	1803	103 803	105 870	108 079	110 423	112 894
9	Estimates	Small Island Developing		1637	Special other	1803	23 771	24 209	24 685	25 187	25 710
10	Estimates	World Bank income groups		1802	Label/Separate	900
11	Estimates	High income		1500	Income Group	1802	604 000	700 000	711 500	720 100	730 500

Preparing data

```
app = Flask(__name__)
#### Creating time series panel data
|
def get_df_name(df):
    name = [x for x in globals() if globals()[x] is df][0]
    return name

def panel(df):

    name = get_df_name(df)
    df = df.drop(columns=['Index', 'Type', 'Variant', 'Notes', 'Country code', 'Parent code'])

    # remove all rows with NaN cells
    df = df.dropna(axis=0)

    df = df.rename(columns={'Region, subregion, country or area': 'country'})
    # rows having '...' are not deleted, delete by:
    #df = df.loc[~((df['country']=='...') | (df['year']=='...') | (df['Region, subregion, country or area']

    df['country'] = df['country'].replace({'Iran, Islamic Rep.': 'Iran'})
    dft = df.T
    new_header = dft.iloc[0]
    dft = dft[1:]
    dft.columns = new_header

    dft = dft.loc[:, ~(dft == '...').any()]

    for col in range(len(dft.columns)):
        dft.iloc[:, col] = dft.iloc[:, col].str.replace(' ', '')

    return dft

### Reading data
pop = pd.read_csv(r"https://raw.githubusercontent.com/Alireza-Ehiaei/Machine_learning/master/WPP2019_POP_F

panel = panel(pop)
```

Preparing data

Data is ready to be used in model:

[illegible]

Autoregression model

In this research, the AutoReg model is used to learn from previous data of the population of each country in order to build a forecasting model. Since each country has its own population dynamics, it may have a different autoregression model, so if we save the model of one country with pickle and use it to forecast the population of other countries, the estimations won't be reliable. In this research the function named 'autoreg' is defined to build the model and will be called for each country in order to build the model separately.

Autoregression model

autoreg model:

```
def autoreg(df, list_country, a, b):

    from statsmodels.tsa.ar_model import AutoReg
    from random import random
    #import pickle
    import json

    yhat = []
    yhat= pd.DataFrame(yhat)
    # contrived dataset
    data = panel[panel.columns[panel.columns.isin(list_country)]]
    data = data.astype(int).reset_index(drop=True)
    # fit model
    for col in data.columns:
        model_AutoReg = AutoReg(data.loc[:,col], lags=1,old_names=False)
        model_fit = model_AutoReg.fit()

    #if there is just one model (one time series) save the model with pickle and call it in app:
    # pickle.dump(model_AutoReg , open('model.pkl','wb'))

    # make prediction
    yhat.loc[:,col] = model_fit.predict(a,b)
```

Autoregression model

The auto regression model can be tested before creating Flask app.

```
: 1 autoreg(panel,['WORLD', 'More developed regions'], 80,85)
```

For example, the prediction for the first two columns for the time periods 2030 – 2035 is:

	WORLD	More developed regions
years		
2030	8,745,733,000	1,304,051,000
2031	8,844,029,000	1,306,841,000
2032	8,942,922,000	1,309,584,000
2033	9,042,415,000	1,312,279,000
2034	9,142,512,000	1,314,927,000
2035	9,243,217,000	1,317,530,000

Deployment the model using Flask

After importing libraries as well Flask, an instance of the *Flask* class is defined at tope of the code to be able to call the app on heroku. for this app the index.html, layout.html and view.html files are saved in templates folder and css file in static folder both whitin the main folder that includes the main file (app.py).

In the first part of app, 'index' function is called that asks from the user to enter the set of countries and years through the index.html file.

```
@app.route('/')
def index():
    return render_template(
        'index.html',
        data= list(panel.columns))
```

Deployment the model using Flask

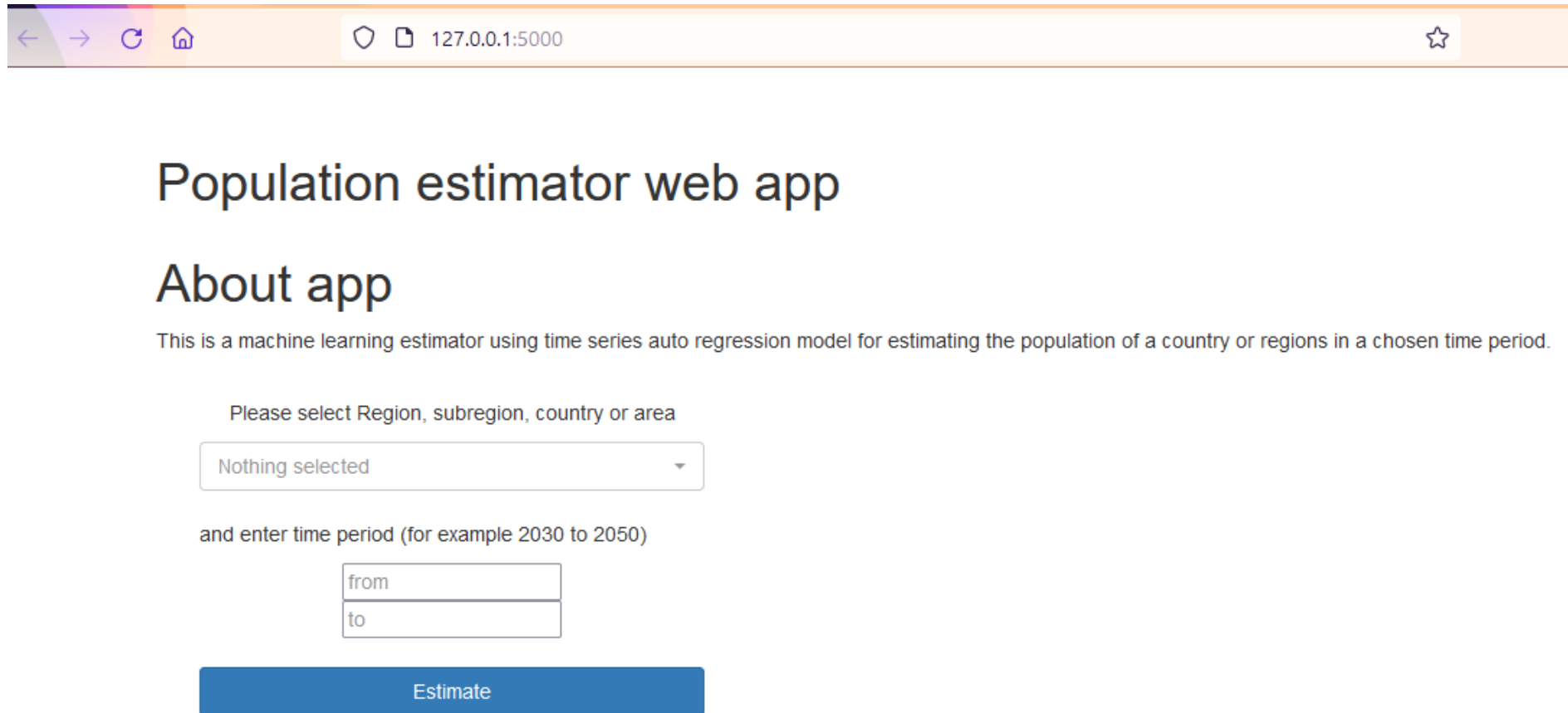
index.html file:

```
<div class="container">

  <div class="col-md-4" style="text-align: center">
    <form action="{{ url_for('test') }}" method="post" id="multiple_select_form" :
    <br />
    <p>Please select Region, subregion, country or area. </p>
    <p>(sometimes Firefox does not allow multi selection, please choose another !
    <select multiple name="countries" class="form-control selectpicker" data-live-
      {% for o in data %}
        <option value="{{ o }}">{{ o }}</option>
      {% endfor %}
    </select>
    <br /><br />
    <p>and enter time period (for example 2030 to 2050)</p>
    <input type="text" name="from" placeholder="from" required="required" /><br>
    <input type="text" name="to" placeholder="to" required="required" /><br>
    <br />
    <button type="submit" class="btn btn-primary btn-block">Estimate</button>
  </form>
</div>
```

Deployment the model using Flask

The page that is created locally by the codes above:



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000'. The page content includes a title 'Population estimator web app', a subtitle 'About app', and a description: 'This is a machine learning estimator using time series auto regression model for estimating the population of a country or regions in a chosen time period.' Below the text, there is a dropdown menu with the placeholder text 'Nothing selected' and a label 'Please select Region, subregion, country or area'. Underneath the dropdown, there is a label 'and enter time period (for example 2030 to 2050)' followed by two input fields labeled 'from' and 'to'. At the bottom of the form is a blue button labeled 'Estimate'.

Population estimator web app

About app

This is a machine learning estimator using time series auto regression model for estimating the population of a country or regions in a chosen time period.

Please select Region, subregion, country or area

Nothing selected

and enter time period (for example 2030 to 2050)

from




to

Estimate

Deployment the model using Flask

Then the data entered by the user will be sent to the test route of the app and the autoregression model will be executed on the selected data and the result will be returned based on settings defined in layout.html and view.html and style.css.

For example, for the first two columns from 2030 to 2035 are:

   127.0.0.1:5000/test		
<h2>Estimation</h2>		
Estimation of your selected data is:		
	WORLD	More developed regions
years		
2030	8,745,733,000	1,304,051,000
2031	8,844,029,000	1,306,841,000
2032	8,942,922,000	1,309,584,000
2033	9,042,415,000	1,312,279,000
2034	9,142,512,000	1,314,927,000
2035	9,243,217,000	1,317,530,000

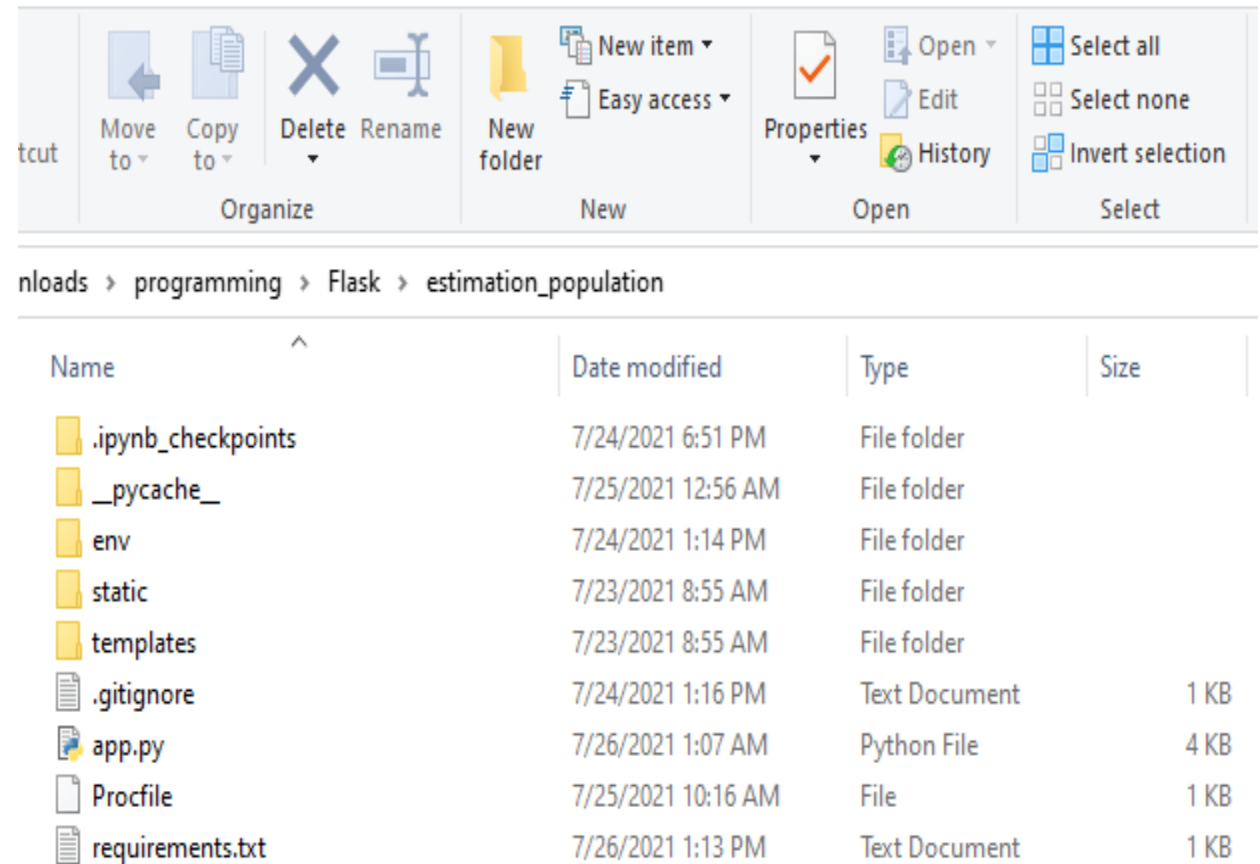
Deployment the flask app on Heroku

Heroku is a cloud platform as a service supporting several programming languages. Developers use Heroku to deploy, manage, and scale modern apps.

There are different ways to deploy an app on Heroku, it can be done using heroku website and connecting it to the github pages that all files and template and static folders are uploaded. We used git in this project.

Deployment the flask app on Heroku

First, the command prompt is applied and the path of the folder in personal devices (laptop) including all files is sent to it, then a virtual environment is created, and by using gunicorn in Procfile file Heroku receives the order of executing the app. Requirements.txt determine the libraries the should be installed by the app that Heroku needs to apply, the folder looks like this:



The screenshot shows a Windows File Explorer window with the address bar displaying the path: `nloads > programming > Flask > estimation_population`. The ribbon at the top includes tabs for 'Organize', 'New', 'Open', and 'Select'. The main area displays a list of files and folders with columns for Name, Date modified, Type, and Size.

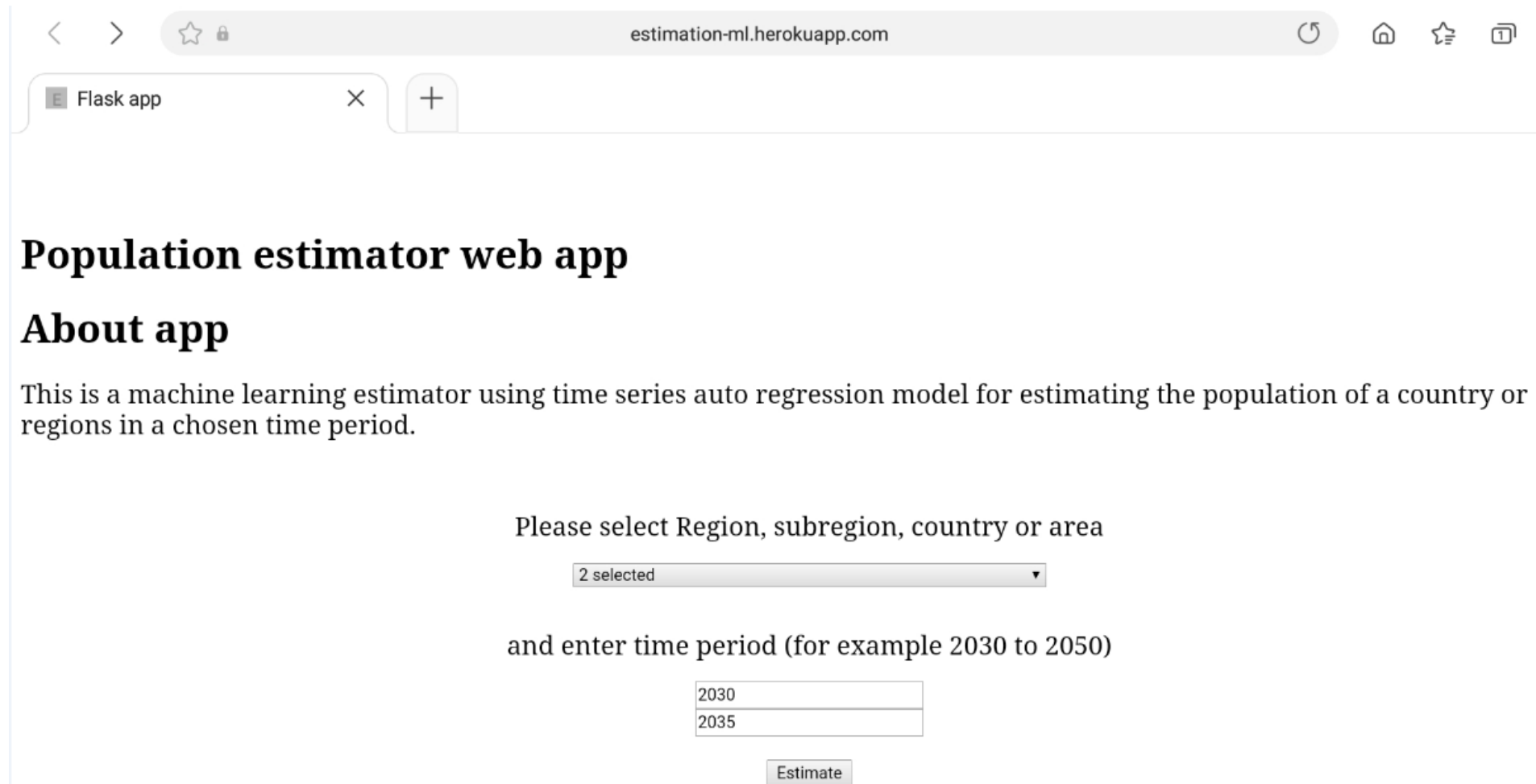
Name	Date modified	Type	Size
.ipynb_checkpoints	7/24/2021 6:51 PM	File folder	
__pycache__	7/25/2021 12:56 AM	File folder	
env	7/24/2021 1:14 PM	File folder	
static	7/23/2021 8:55 AM	File folder	
templates	7/23/2021 8:55 AM	File folder	
.gitignore	7/24/2021 1:16 PM	Text Document	1 KB
app.py	7/26/2021 1:07 AM	Python File	4 KB
Procfile	7/25/2021 10:16 AM	File	1 KB
requirements.txt	7/26/2021 1:13 PM	Text Document	1 KB

Deployment the flask app on Heroku

In the command prompt, we login to Heroku account that we should create it on the Heroku website, then an app is created on Heroku called estimation-ml, then add and commit all files and their changes to the remote git and by pushing them to the Heroku, the app will be created and is accessible through internet for others. The first page of the app and the result for the first two regions on Heroku are provided in the following slides.

Note, Firefox and internet explorer sometimes do not execute Heroku apps properly, please choose another internet browser.

Deployment the the flask app on Heroku



< > ☆ 🔒 estimation-ml.herokuapp.com ↻ 🏠 ☆ 📄

Flask app × +

Population estimator web app

About app

This is a machine learning estimator using time series auto regression model for estimating the population of a country or regions in a chosen time period.

Please select Region, subregion, country or area

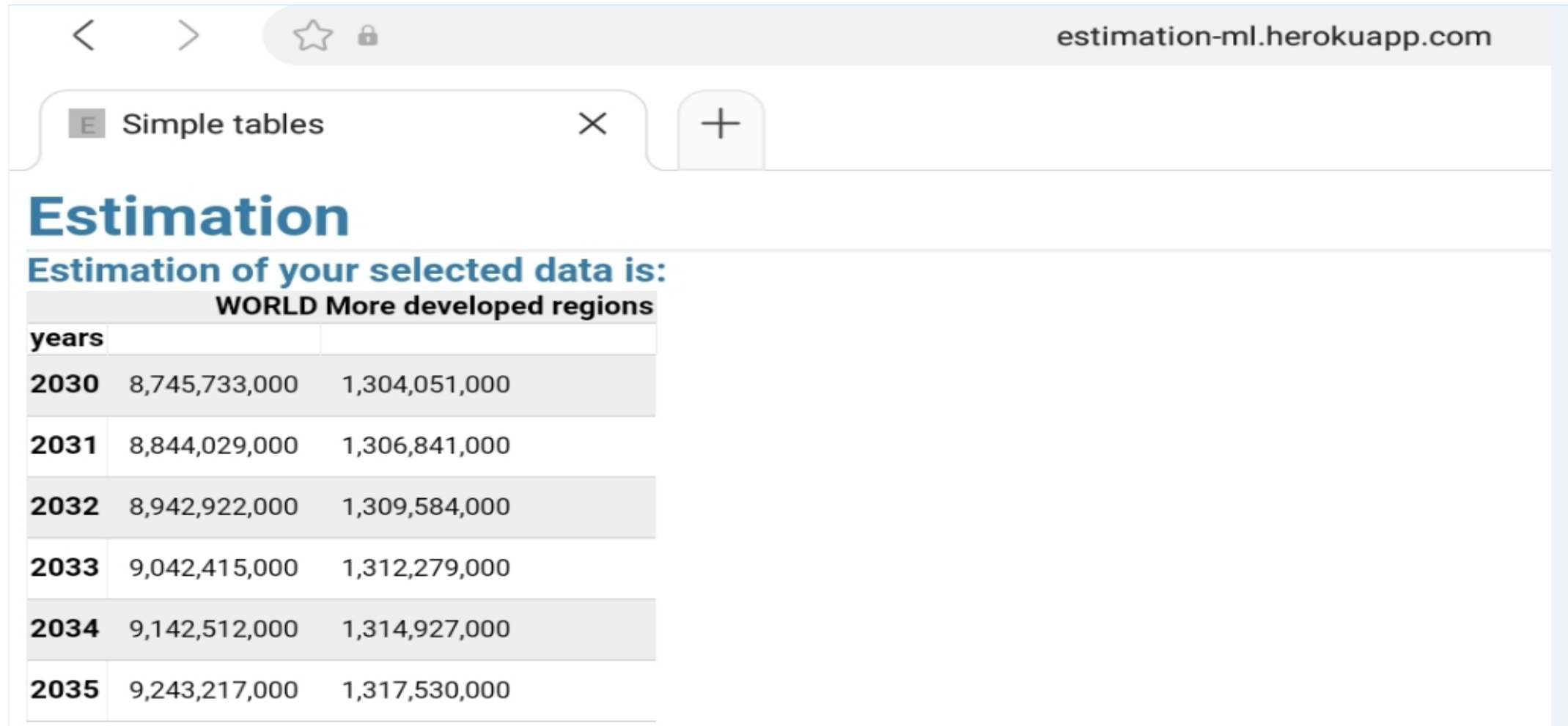
2 selected ▼

and enter time period (for example 2030 to 2050)

2030
2035

Estimate

Deployment the flask app on Heroku



estimation-ml.herokuapp.com

Simple tables

Estimation

Estimation of your selected data is:

WORLD More developed regions		
years		
2030	8,745,733,000	1,304,051,000
2031	8,844,029,000	1,306,841,000
2032	8,942,922,000	1,309,584,000
2033	9,042,415,000	1,312,279,000
2034	9,142,512,000	1,314,927,000
2035	9,243,217,000	1,317,530,000

Thank You