# Week 4: Deployment on Flask

## Name: Marija Babić

## Batch code: LISUM14

## Submission date: 2022-10-22

## Submitted to: https://github.com/marija0408/NLP-Data-Science-Internship/tree/main/Week%205

## 1.Finding the dataset

I have found a simple dataset for linear regression on Kaggle. The link is below:

<https://www.kaggle.com/datasets/karthickveerakumar/salary-data-simple-linear-regression>

It is a small dataset, 30 observations all together, but enough for me to perform a simple model. The dataset has two columns: YearsExperience and Salary. YearsExperience would be independent variable (or X in my code) and Salary would be dependent variable (y in my code).

## 2.Fitting the model and saving the model

The next step is fitting the model. I created ipynb notebook called *Prepare\_and\_save\_regression\_model.ipynb*. After reading the data from the csv file downloaded from Kaggle website, I checked a couple of things.

Graphical user interface, text, email

Description automatically generated

Figure : Reading the data and checking column types

The first thing is checking column types (everything was good, both variables were in float format) and the second thing is describe function which gave me info about number of values, mean, std, min column values, max column values etc. Everything looked good, so the last thing I did to check the data quality was to plot the data to see how it looked. I used the same dataset for article writing so I have checked more things while preparing the article and knew that everything is good from the data side.

Chart, scatter chart

Description automatically generated

Figure : Checking descriptive statistics and plotting the scatterplot

After checking the data quality, it was time to fit the model and save it. I used *LinearRegression* library from *sklearn.linear\_model* to fit the model and *pickle.dump* for saving the model. I also did a test where I checked the model coefficient and intercept to see if it produces the same value as the model.

Graphical user interface, text, application, email

Description automatically generated

Figure : Fitting the model, saving the model and checking the regressor coefficient and intercept

## 3.Api model local

I have created app.py file in my repository Heroku-web-app-api:

Text

Description automatically generated

After running the file through cmd, we get the url which will be used in postman:

Text

Description automatically generated

I copied the url and pasted it into postman. If we go to the route, this is the result:

Graphical user interface, text, application, email

Description automatically generated

We get json response from the web app. If we add predict/ to the route, we can send the years value to the web app:

Graphical user interface, text, application, email

Description automatically generated

We got the predicted salary for the given years of experience.

## 4.HTML model local

I have created the separate model that works a bit different, in the for of the website. I used the index.html that I have found in the resources of the course. This is the app.py in the Heroku-web-app-html repo that I have created:

Text

Description automatically generated

If we run this file in the cmd, we get:

Text

Description automatically generated

If we copy the url into the browser, we get:

Graphical user interface, text

Description automatically generated

The result we get is the same as for the api version.

## 3.Models on Heroku

After testing the files locally, I erased the part of code from both app.py files.

This is before:

Text

Description automatically generated

This is after:

Text

Description automatically generated

I also added a Procfile file to both repos:

Graphical user interface, text, application

Description automatically generated

I pushed all the changes to my remote git repos (each app is in separate repo).

After that, I went to Heroku and clicked on Create new app button:

Graphical user interface, text, application, email

Description automatically generated

We have to choose a unique app name and region:

Graphical user interface, text, application, email, Teams

Description automatically generated

On the Deploy page, after clicking on github, I choose the repo where the files for my web application are:

Graphical user interface, text, application, email

Description automatically generated

And after that, I started the manual deployment:

Text

Description automatically generated

After all the requirements are installed, we get the url of the web app:

Graphical user interface, text, application

Description automatically generated

I did the same thing for the api model.