# 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%204

## 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.Deployment on flask

The third and the last step is deployment on Flask. I downloaded the Flask Code from LISUM14 Resources to use it as a guideline. I had to change app.py and index.html a bit so it would work on my model.

In app.py I had to change a couple of things:

* Cast input features to float because my model expects floats
* Add one variable, *input\_feature,* which I used in the output sentence
* Change the output sentence
* Add some parameters to *app.run* because otherwise the code wouldn’t run on my computer. I changed the host to localhost and port to 9874

Text

Description automatically generated

Figure : App.py for my model

In index.html I changed the *input type* to be *YearsExperience*:

Text

Description automatically generated

Figure : Changed the input type

After all of the changes written above, I was ready to start my app. I ran cmd from the Week 4 folder and ran the script:  
Text

Description automatically generated

Figure : Running app.py from Week 4 folder

I copied the url and pasted it to the Chrome browser:

Graphical user interface

Description automatically generated

Figure : The app interface

I wanted to test the app. In my *Prepare\_and\_save\_regression\_model.ipynb* notebook there is a part of code where I test the prediction of one value of years of experience. The output was:

Graphical user interface, text, application

Description automatically generated

Figure : Model prediction of Salary for 3 years of experience

I will insert the same value in the app to see if the results match. This is the screen before I clicked Predict button:

Graphical user interface

Description automatically generated

Figure : Checking the prediction

This is the screen after I clicked the Predict button:

Text

Description automatically generated

Figure : The prediction result

We can see that this result matches the one I got in the python notebook.

The app is running properly and gives the correct prediction value of the saved model.