

## Week 5: Cloud and API deployment

Name: Marija Babić

Batch code: LISUM14

Submission date: 2022-10-25

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

### 1. Finding the dataset

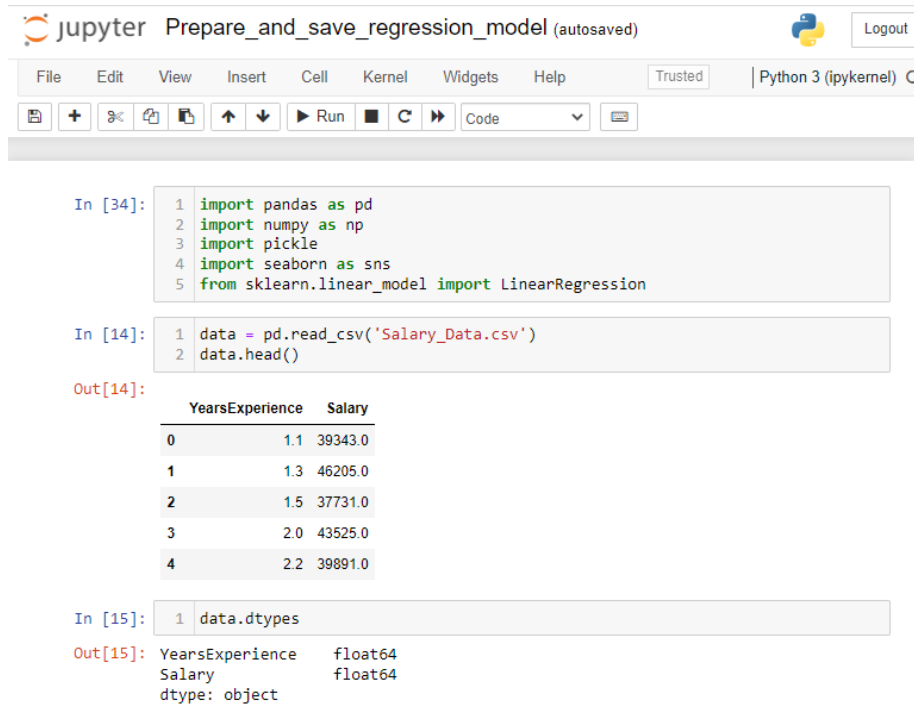
I have found a simple dataset for linear regression on Kaggle. The link is below:

<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.



The image shows a Jupyter Notebook interface with the title "Prepare\_and\_save\_regression\_model (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook contains three input cells and two output cells.

```
In [34]: 1 import pandas as pd
         2 import numpy as np
         3 import pickle
         4 import seaborn as sns
         5 from sklearn.linear_model import LinearRegression
```

```
In [14]: 1 data = pd.read_csv('Salary_Data.csv')
         2 data.head()
```

```
Out[14]:
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
In [15]: 1 data.dtypes
```

```
Out[15]: YearsExperience    float64
         Salary           float64
         dtype: object
```

Figure 1: 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.

```
In [16]: 1 data.describe()
```

```
Out[16]:
```

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

```
In [35]: 1 fig = sns.scatterplot(data = data ,x ='YearsExperience',y ='Salary')
2 fig.set_xlabel('Years of experience')
3 fig.set_ylabel('Salary')
```

```
Out[35]: Text(0, 0.5, 'Salary')
```

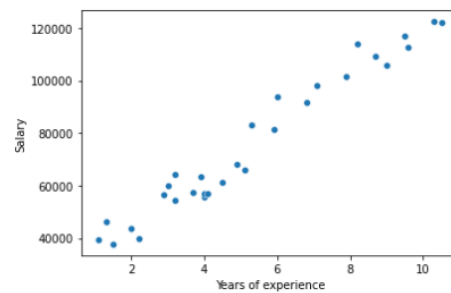


Figure 2: Checking descriptive statistics and plotting the scatterplot

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.

```
In [44]: 1 X = data.iloc[:, :1].values
        2 y = data.iloc[:, -1]
        3
        4 regressor = LinearRegression()
        5 regressor.fit(X, y)
        6 pickle.dump(regressor, open('model.pkl', 'wb'))
```

```
In [45]: 1 regressor.coef_
```

```
Out[45]: array([9449.96232146])
```

```
In [46]: 1 regressor.intercept_
```

```
Out[46]: 25792.20019866871
```

```
In [47]: 1 regressor.predict([[3]])
```

```
Out[47]: array([54142.08716303])
```

```
In [48]: 1 #check result
        2 3*regressor.coef_[0] + 25792.20019866871
```

```
Out[48]: 54142.08716303393
```

Figure 3: 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:

```
app.py X
C:\Users\Marija\Desktop> cd repos\Heroku-web-app-api & python app.py
1 import numpy as np
2 from flask import Flask, request, render_template, jsonify
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open('model.pkl', 'rb'))
7
8 @app.route('/', methods = ['GET', 'POST'])
9 def home():
10     data = 'hello world'
11     return jsonify({'data': data})
12
13 @app.route('/predict/')
14 def predict():
15
16     years = request.args.get('years')
17     float_features = [float(years)]
18     final_features = [np.array(float_features)]
19     prediction = model.predict(final_features)
20
21     input_feature = float_features[0]
22     output = round(prediction[0], 2)
23
24     return jsonify({'Salary': str(output)})
25
26 if __name__ == "__main__":
27     app.run(debug=True, host='localhost', port=9874)
28
```

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

```
C:\Users\Marija\Desktop\repos\Heroku-web-app-api>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://localhost:9874
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 123-521-818
```

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

The screenshot shows a web browser's developer tools interface. The top bar indicates a GET request to `http://localhost:9874`. The main panel displays the response body in JSON format, showing a successful 200 OK status with a response time of 13 ms and a size of 193 B. The response data is:

```
{  "data": "hello world"}
```

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:

Postman interface showing a GET request to `http://localhost:9874/predict/?years=3`. The request is successful, returning a 200 OK status with a response time of 22 ms and a body size of 192 B. The response body is displayed in JSON format:

```
1 {
2   "Salary": "54142.09"
3 }
```

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:

```

app.py 2 X
C:\Users\Marija\Desktop\repos\Heroku-web-app-html > app.py > ...
1  import numpy as np
2  from flask import Flask, request, render_template, jsonify
3  import pickle
4
5  app = Flask(__name__)
6  model = pickle.load(open('model.pkl', 'rb'))
7
8  @app.route('/', methods = ['GET', 'POST'])
9  def home():
10     return render_template('index.html')
11
12  @app.route('/predict/', methods=['POST'])
13  def predict():
14
15
16     float_features = [float(x) for x in request.form.values()]
17     final_features = [np.array(float_features)]
18     prediction = model.predict(final_features)
19
20     input_feature = float_features[0]
21     output = round(prediction[0], 2)
22
23     return render_template('index.html', prediction_text=f'Salary for {input_feature} years of experience should be ${output}')
24
25  if __name__ == "__main__":
26     app.run(debug=True, host='localhost', port=9874)
27
28

```

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

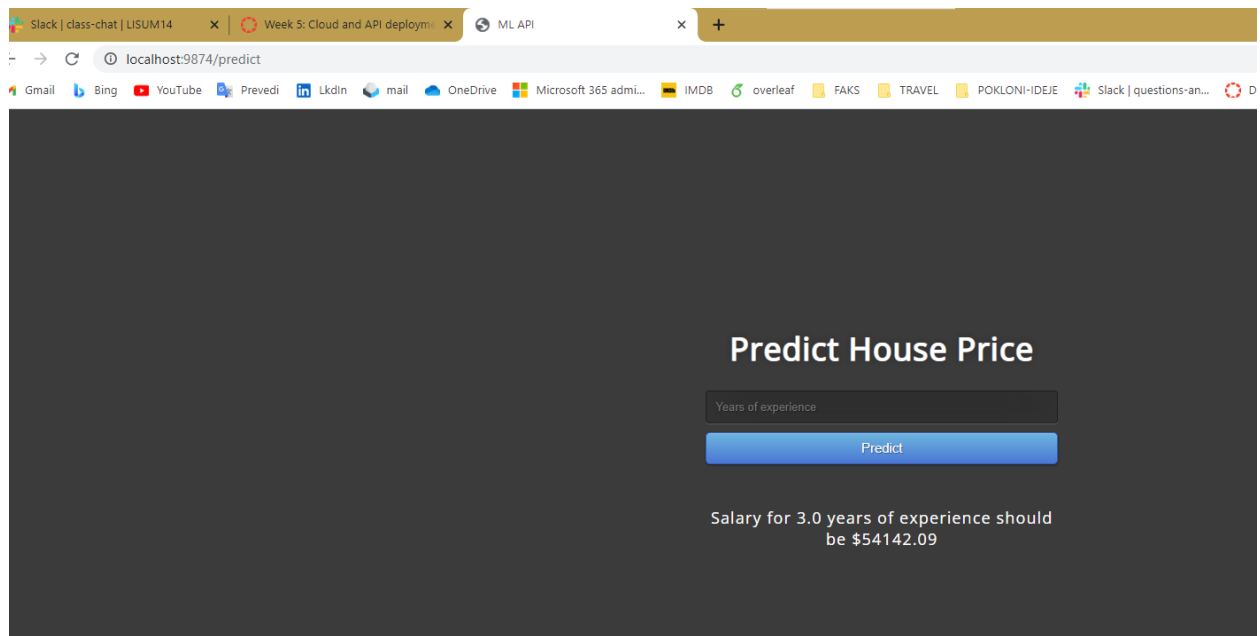
```

C:\Users\Marija\Desktop\repos\Heroku-web-app-html>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production
* Running on http://localhost:9874
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 123-521-818

```

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





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:

```
if __name__ == "__main__":  
    app.run(debug=True, host='localhost', port=9874)
```

This is after:

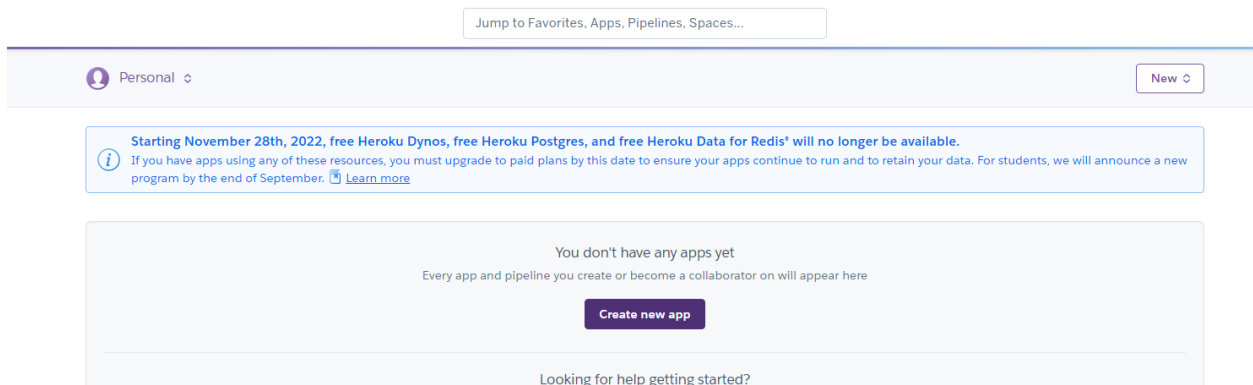
```
if __name__ == "__main__":  
    app.run(debug=True)
```

I also added a Procfile file to both repos:

```
Procfile - Notepad
File Edit Format View Help
web: gunicorn app:app
```

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:



We have to choose a unique app name and region:

A screenshot of the "Create New App" form in Heroku. The form is titled "Create New App" and has a light gray background. It contains the following fields and buttons: 1. "App name" label followed by a text input field containing "mb-html-web-app" and a small "..." button to its right. 2. "Choose a region" label followed by a dropdown menu showing "United States" with a US flag icon and a dropdown arrow. 3. An "Add to pipeline..." button. 4. A purple "Create app" button at the bottom.

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

The screenshot shows the Heroku Deploy page for the application 'mb-html-web-app'. The top navigation bar includes 'Personal', 'mb-html-web-app', and buttons for 'Open app' and 'More'. The main section is titled 'Add this app to a pipeline' and 'Add this app to a stage in a pipeline to enable additional features'. Below this, there are options for 'Deployment method' (Heroku Git, GitHub, Container Registry) and a 'Connect to GitHub' section. The 'Connect to GitHub' section includes a search bar for repositories, a list of repositories (marija0408/Linear-Regression-in-R and marija0408/Power-BI-Reports), and 'Connect' buttons. A 'Manual deploy' section is also visible, showing a dropdown for 'main' and a 'Deploy Branch' button.

And after that, I started the manual deployment:

The screenshot shows the Heroku Manual deploy page. The 'Manual deploy' section is active, showing the current state of a branch to this app. The 'Deploy a GitHub branch' section is also visible, showing the current state of the branch you specify below. The 'Choose a branch to deploy' dropdown is set to 'main', and the 'Deploy Branch' button is visible. Below this, the 'Build main a3a4781a' section shows the deployment process, including the output of the build and the deployment to Heroku. The output shows the app is launched and released, with a URL provided: https://mb-html-web-app.herokuapp.com/. The 'Autoscroll with output' checkbox is checked, and a 'View build log' link is present.

I did the same thing for the api model.