

Name: Deployment on Flask

Batch code: LISUM20

Submission date: 24.04.2023

Submitted to: Data Glacier

Prepared by: Daria Lazorenko

1. Data Intake

We used a simple Salary dataset that contains two columns: "YearsExperience" and "Salary". We loaded the dataset into a pandas DataFrame and split it into training and testing datasets.

2. Model training

We used linear regression to predict the salary based on the years of experience. We used scikit-learn library to train our model.

3. Save the model

We used pickle to save our trained model to a file.

4. Deploy the model on Flask

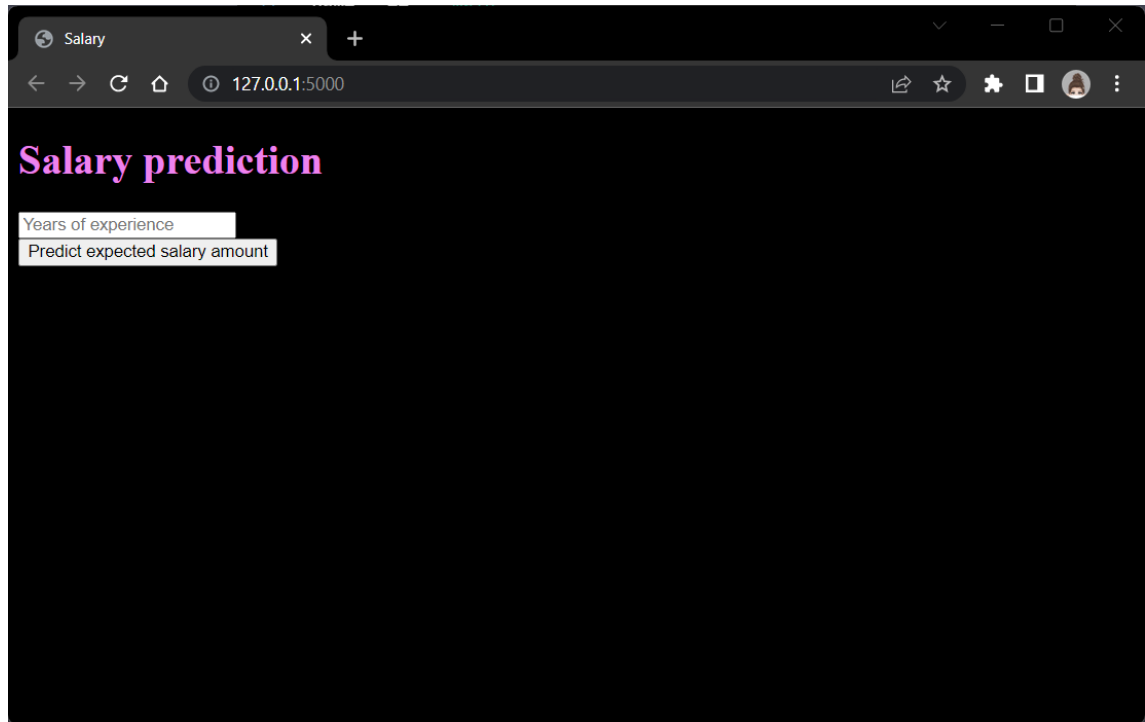
We created a Flask app and loaded the trained model using pickle to create a RESTful API that takes a "YearsExperience" input and returns the predicted salary.

```
1  import numpy as np
2  from flask import Flask, request, render_template
3  import pickle
4
5  #Create an app object using the Flask class.
6  app = Flask(__name__)
7
8  #Load the trained model. (Pickle file)
9  model = pickle.load(open('models/model.pkl', 'rb'))
10
11 #use the route() decorator to tell Flask what URL should trigger our function.
12 @app.route('/')
13 def home():
14     return render_template('index.html')
15
16 @app.route('/predict',methods=['POST'])
17 def predict():
18
19     int_features = [float(x) for x in request.form.values()]
20     features = [np.array(int_features)]
21     prediction = model.predict(features)
22
23     output = round(prediction[0], 2)
24
25     return render_template('index.html', prediction_text='Expected salary is {}'.format(output))
26
27 if __name__ == "__main__":
28     app.run()
```

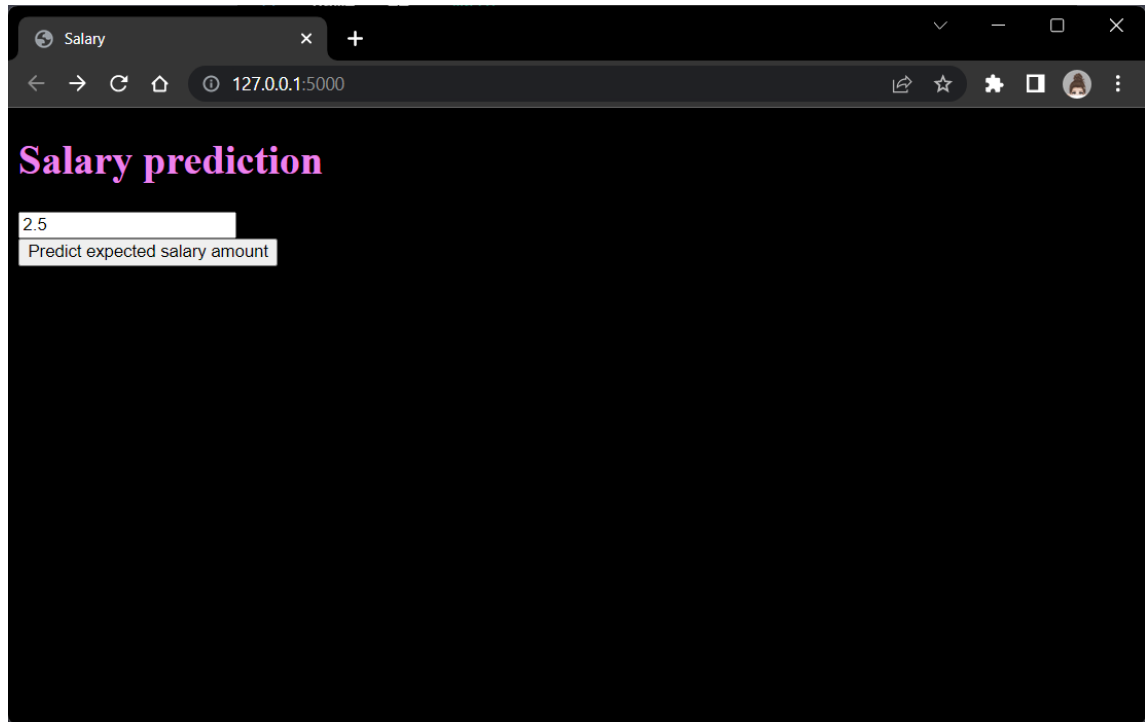
Graphics 1: handle POST requests and return the results

5. Create a simple web interface for a deployed machine learning model using HTML and Flask.

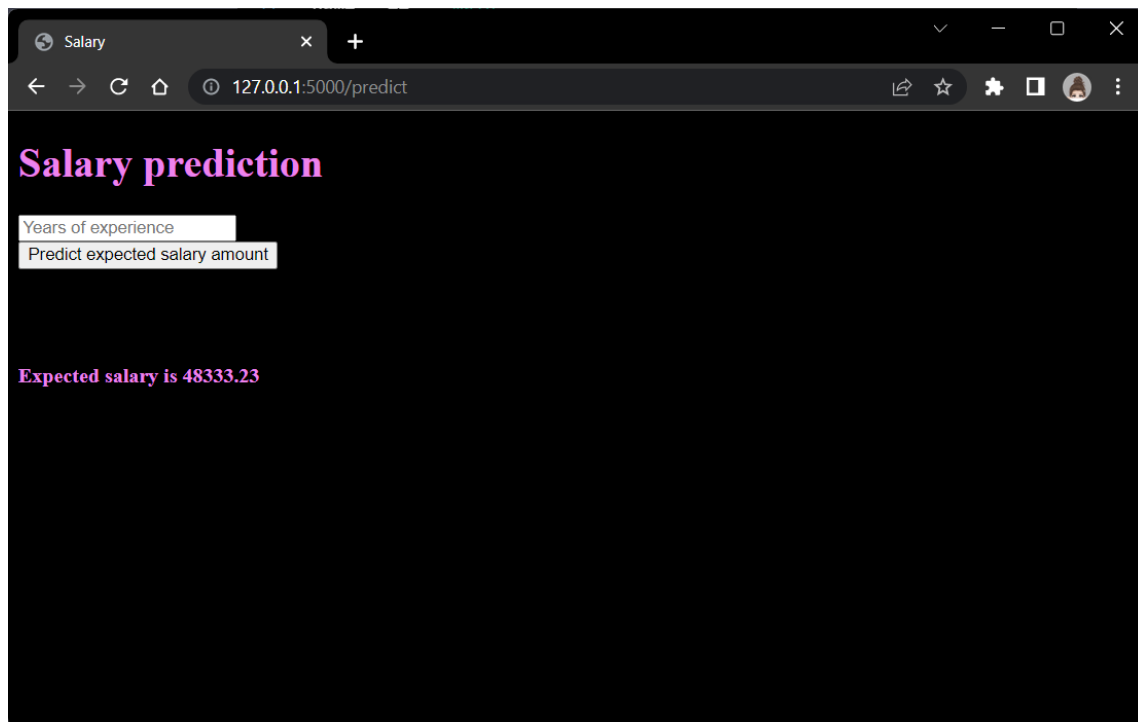
Users can input data into a form, which is processed by the Flask route and passed through the model. The prediction is then displayed on a new HTML page.



A screenshot of a web browser window with a dark theme. The browser's address bar shows the URL '127.0.0.1:5000'. The page title is 'Salary'. The main heading is 'Salary prediction' in a large, pink, serif font. Below the heading, there are two input fields: the first is labeled 'Years of experience' and the second is labeled 'Predict expected salary amount'. Both fields are currently empty.



A screenshot of the same web browser window. The 'Years of experience' input field now contains the number '2.5'. The 'Predict expected salary amount' field remains empty. The rest of the page, including the heading and browser interface, is identical to the previous screenshot.



6. Conclusion

In this project, we used a simple linear regression Salary dataset to predict the salary based on the years of experience. We trained our model using scikit-learn and saved it using pickle. We deployed the model on Flask to create a RESTful API that takes a "YearsExperience" input and returns the predicted salary. Our model can now be accessed by any client that sends a POST request to the Flask app.