

Name: Shunyi Huang

Submission Date: 09/04/2022

```
PS C:\Users\shunyi\OneDrive\Desktop\Data Glacier\Week 4\Deployment-on-Flask-> & C:\Users\shunyi\AppData\Local\Microsoft\Windows\Common-Programs\Python\Python39\python.exe "c:/Users/shunyi/OneDrive/Desktop/Data Glacier/Week 4/Deployment-on-Flask-/flask_app.py"
* Serving Flask app 'flask_app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
```

← → ↻ 🏠 127.0.0.1:5000 110% ☆ 📧

哔哩哔哩弹幕视频网 - ... YouTube University of Wiscon... Tenant Web Access ... Illinois Tollway 我的云端硬盘 - Goog... University of Wiscon... Wolfram|Alpha: Co...

DS salary prediction

Experience level

Employment Type


Job Title

Remote Ratio

Company Size

Employee Residence

Predict

 **Data Glacier**

Your Deep Learning Partner

ML API

127.0.0.1:5000/predict

110%

哔哩哔哩弹幕网... YouTube University of Wisconsin... Tenant Web Access ... Illinois Tollway 我的云端硬盘 - Goog... University of Wisconsin... Wolfram|Alpha: Co...

DS salary prediction

Experience level

Employment Type


Job Title

Remote Ratio

Company Size

Employee Residence

Predict

 **Data Glacier**
Your Deep Learning Partner

The salary will be \$ [20023.08320238]

```

1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[1]:
5
6
7  import pandas as pd
8  import numpy as np
9  import matplotlib.pyplot as plt
10 import seaborn as sns
11 from sklearn.model_selection import train_test_split
12 import pickle
13 import requests
14 import json
15 from statsmodels.stats.outliers_influence import variance_inflation_factor
16 from sklearn.preprocessing import LabelEncoder
17 from sklearn.linear_model import LinearRegression
18 from flask import Flask, request, jsonify, render_template
19 from sklearn.metrics import mean_squared_error
20
21
22
23 # # EDA
24
25 # In[2]:
26
27
28 def eda(df):
29     print(df.head(2),'\n')
30     print(df.info())
31     print(df.describe(),'\n')
32     print(df.isna().sum())
33     print('Number of data point is: ', len(df))
34
35 def box_plot(df, feature):
36     df.plot(y = feature, kind = 'box')
37
38 def heat_map(df):
39     sns.heatmap(df.corr(),annot=True)
40
41 def check_vif(df, features):
42     df = df.drop(features,axis = 1)
43     vif_data = pd.DataFrame()
44     vif_data['feature'] = df.columns
45
46     vif_data['VIF'] = [variance_inflation_factor(df.values, i) for i in range(len(df.columns))]
47     print(vif_data)
48
49
50 # In[3]:
51
52
53 df = pd.DataFrame(pd.read_csv('ds_salaries.csv'))
54 #df.head()
55
56
57 # In[4]:
58
59
60 le = LabelEncoder()
61 df['salary_currency'] = le.fit_transform(df['salary_currency'])
62 df['employee_residence'] = le.fit_transform(df['employee_residence'])
63 df['company_location'] = le.fit_transform(df['company_location'])
64 df['company_size'] = le.fit_transform(df['company_size'])
65 df['experience_level'] = le.fit_transform(df['experience_level'])
66 df['job_title'] = le.fit_transform(df['job_title'])
67
68
69 # In[5]:
70
71
72 df['employment_type'] = le.fit_transform(df['employment_type'])
73 df.head()
74
75
76 # In[6]:
77
78
79 # eda(df)
80 # heat_map(df)
81 # check_vif(df,['work_year','company_location','salary_currency'])
82
83
84 # # Data splitting
85
86 # In[7]:
87 X = df.loc[:, df.columns != 'salary_in_usd']
88 X = X.drop(['Unnamed: 0','work_year','salary','salary_currency','company_location'],axis = 1)
89 y = df['salary_in_usd']
90
91
92 # df = df.drop(['work_year','company_location','salary_currency','salary'],axis = 1)
93 # X = df.loc[:, df.columns != 'salary_in_usd']
94 # X = X.drop(['Unnamed: 0','experience_level','employment_type','employee_residence','remote_ratio','company_size'],axis = 1)
95 # y = df['salary_in_usd']
96 # X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25, random_state = 42)
97

```

File Edit View Language

Python

```
54 #df.head()
55
56
57 # In[4]:
58
59
60 le = LabelEncoder()
61 df['salary_currency'] = le.fit_transform(df['salary_currency'])
62 df['employee_residence'] = le.fit_transform(df['employee_residence'])
63 df['company_location'] = le.fit_transform(df['company_location'])
64 df['company_size'] = le.fit_transform(df['company_size'])
65 df['experience_level'] = le.fit_transform(df['experience_level'])
66 df['job_title'] = le.fit_transform(df['job_title'])
67
68
69 # In[5]:
70
71
72 df['employment_type'] = le.fit_transform(df['employment_type'])
73 df.head()
74
75
76 # In[6]:
77
78
79 # eda(df)
80 # heat_map(df)
81 # check_vif(df, ['work_year', 'company_location', 'salary_currency'])
82
83
84 ## Data splitting
85
86 # In[7]:
87 X = df.loc[:, df.columns != 'salary_in_usd']
88 X = X.drop(['Unnamed: 0', 'work_year', 'salary', 'salary_currency', 'company_location'], axis = 1)
89 y = df['salary_in_usd']
90
91
92 # df = df.drop(['work_year', 'company_location', 'salary_currency', 'salary'], axis = 1)
93 # X = df.loc[:, df.columns != 'salary_in_usd']
94 # X = X.drop(['Unnamed: 0', 'experience_level', 'employment_type', 'employee_residence', 'remote_ratio', 'company_size'], axis = 1)
95 # y = df['salary_in_usd']
96 # X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42)
97
98
99 # In[8]:
100
101
102 X.head()
103
104
105 ## Modeling
106
107 # In[9]:
108
109
110 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42)
111 lm = LinearRegression()
112 lm.fit(X_train, y_train)
113 y_pred = lm.predict(X_test)
114 mean_squared_error(y_test, y_pred)
115
116 # lm = LinearRegression()
117 # lm.fit(X_train, y_train)
118 # y_pred = lm.predict(X_test)
119 # mean_squared_error(y_test, y_pred)
120 # pickle.dump(lm, open('model.pkl', 'wb'))
121
122
123 # In[15]:
124
125 pickle.dump(lm, open('model.pkl', 'wb'))
126 flask_app = Flask(__name__)
127 model = pickle.load(open('model.pkl', 'rb'))
128
129 @flask_app.route('/')
130 def home():
131     return render_template('flask_app.html')
132
133 @flask_app.route('/predict', methods = ['POST'])
134 def predict():
135
136     int_features = [int(x) for x in request.form.values()]
137     final_features = [np.array(int_features)]
138     prediction = model.predict(final_features)
139     return render_template('flask_app.html', prediction_text = 'The salary will be $ {}'.format(prediction))
140
141
142
143 if __name__ == '__main__':
144     from waitress import serve
145     flask_app.run(port=5000, debug = True)
146
147
148
149
150
```

```

<!DOCTYPE html>
<html >
<!-- From https://codepen.io/frytyler/pen/EGdtg-->

<head>
  <meta charset = "UTF-8">
  <title> ML API </title>
  <link href = 'https://fonts.googleapis.com/css?family=Pacifico' rel = 'stylesheet' type = 'text/css'>
  <link href = 'https://fonts.googleapis.com/css?family=Arimo' rel = 'stylesheet' type = 'text/css'>
  <link href = 'https://fonts.googleapis.com/css?family=Hind:300' rel = 'stylesheet' type = 'text/css'>
  <link href = 'https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel = 'stylesheet' type = 'text/css'>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/flask_style.css') }}">

</head>

<body>
  <div class='login'>
    <h1>DS salary prediction</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action = "{{url_for('predict')}}" method = 'POST'>
      <input type = 'text' name = 'experience level' placeholder = 'Experience level' required = 'required' />
      <input type = 'text' name = 'employment type' placeholder = 'Employment Type' required = 'required' />
      <input type = 'text' name = 'job title' placeholder = 'Job Title' required = 'required' />
      <input type = 'text' name = 'remote_ratio' placeholder = 'Remote Ratio' required = 'required' />
      <input type = 'text' name = 'company size' placeholder = 'Company Size' required = 'required' />
      <input type = 'text' name = 'employee residence' placeholder = 'Employee Residence' required = 'required' />

      <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>

    </form>
    <br>
    <br>
    {{ prediction_text }}
  </div>
  
</html>

```