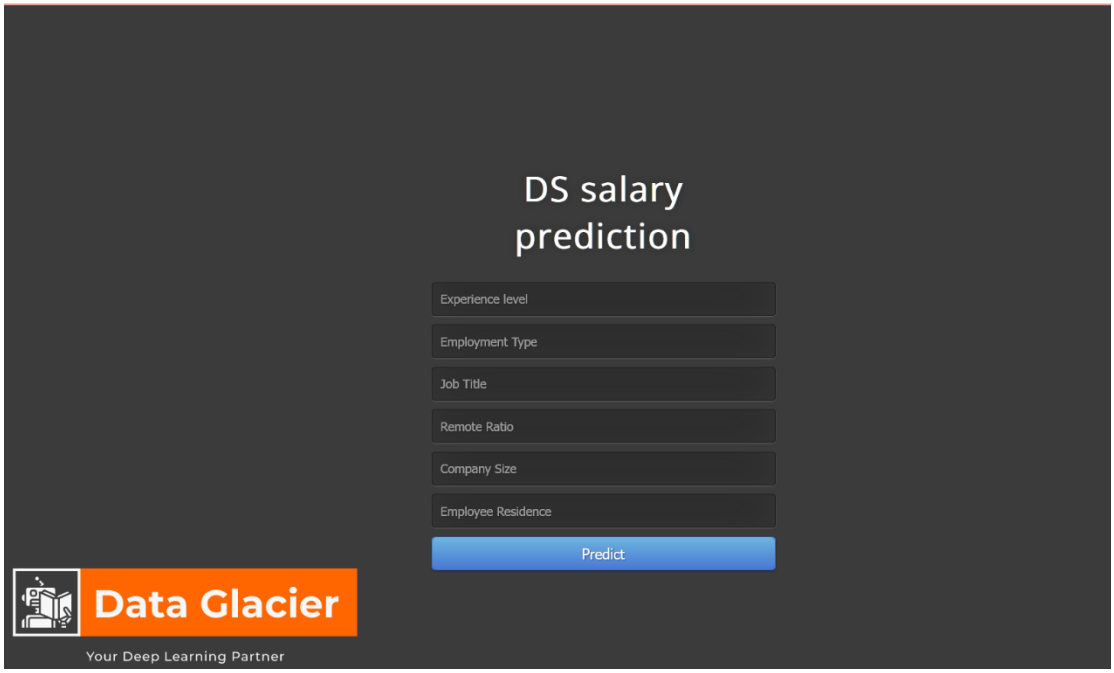
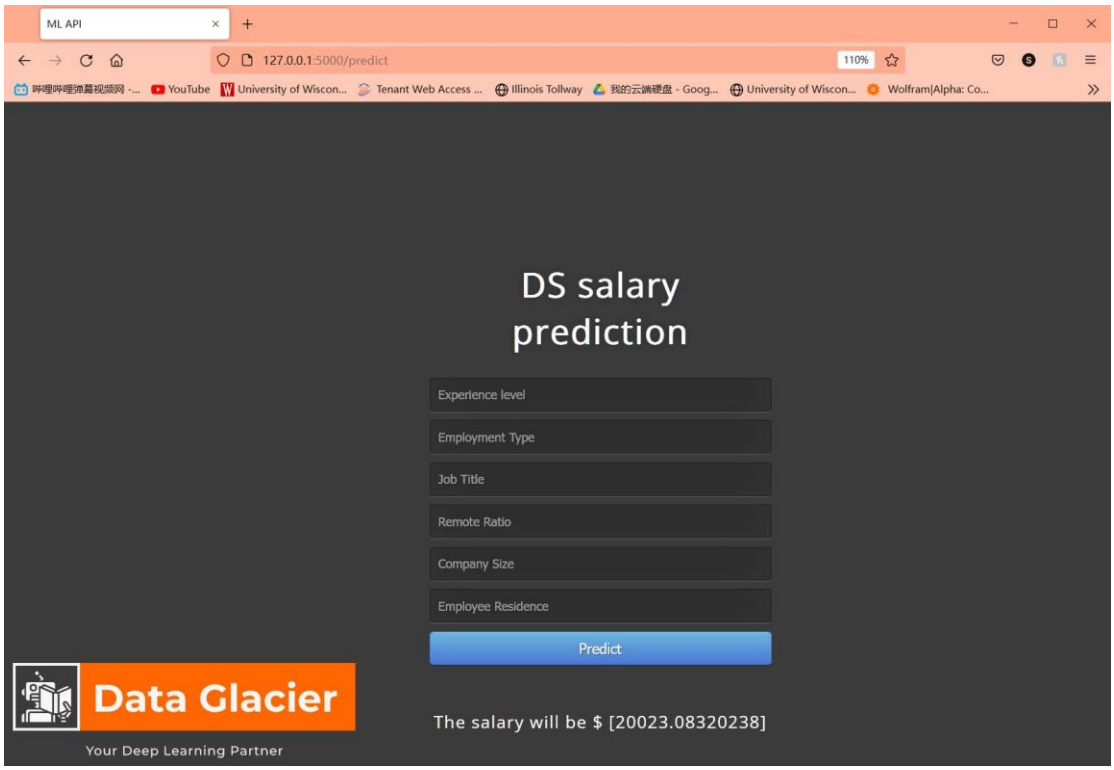
Name: Shunyi Huang
Submission Date: 09/04/2022





jupyter flask_app.py 4 小时前 Logout

File Edit View Language Python

```python
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import pickle
import requests
import json
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from flask import Flask, request, jsonify, render_template
from sklearn.metrics import mean_squared_error



# # EDA

# In[2]:


def eda(df):
      print(df.head(2),'\n')
      print(df.info())
      print(df.describe(),'\n')
      print(df.isna().sum())
      print('Number of data point is: ', len(df))

def box_plot(df, feature):
    df.plot(y = feature, kind = 'box')

def heat_map(df):
      sns.heatmap(df.corr(),annot=True)

def check_vif(df,features):
      df = df.drop(features,axis = 1)
      vif_data = pd.DataFrame()
      vif_data['feature'] = df.columns

      vif_data['VIF'] = [variance_inflation_factor(df.values, i) for i in range(len(df.columns))]
      print(vif_data)


# In[3]:


df = pd.DataFrame(pd.read_csv('ds_salaries.csv'))
#df.head()


# In[4]:


le = LabelEncoder()
df['salary_currency'] = le.fit_transform(df['salary_currency'])
df['employee_residence'] = le.fit_transform(df['employee_residence'])
df['company_location'] = le.fit_transform(df['company_location'])
df['company_size'] = le.fit_transform(df['company_size'])
df['experience_level'] = le.fit_transform(df['experience_level'])
df['job_title'] = le.fit_transform(df['job_title'])


# In[5]:


df['employment_type'] = le.fit_transform(df['employment_type'])
df.head()


# In[6]:


# eda(df)
# heat_map(df)
# check_vif(df,['work_year','company_location','salary_currency'])


# # Data splitting

# In[7]:
X = df.loc[:, df.columns != 'salary_in_usd']
X = X.drop(['Unnamed: 0','work_year','salary','salary_currency','company_location'],axis = 1)
y = df['salary_in_usd']


# df = df.drop(['work_year','company_location','salary_currency','salary'],axis = 1)
# X = df.loc[:, df.columns != 'salary_in_usd']
# X = X.drop(['Unnamed: 0','experience_level','employment_type','employee_residence','remote_ratio','company_size'],axis = 1)
# y = df['salary_in_usd']
# X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25, random_state = 42)
```

jupyter flask_app.py 4 小时前 Logout

File Edit View Language Python

```
#df.head()


# In[4]:


le = LabelEncoder()
df['salary_currency'] = le.fit_transform(df['salary_currency'])
df['employee_residence'] = le.fit_transform(df['employee_residence'])
df['company_location'] = le.fit_transform(df['company_location'])
df['company_size'] = le.fit_transform(df['company_size'])
df['experience_level'] = le.fit_transform(df['experience_level'])
df['job_title'] = le.fit_transform(df['job_title'])


# In[5]:


df['employment_type'] = le.fit_transform(df['employment_type'])
df.head()


# In[6]:


# eda(df)
# heat_map(df)
# check_vif(df,['work_year','company_location','salary_currency'])


# # Data splitting

# In[7]:
X = df.loc[:, df.columns != 'salary_in_usd']
X = X.drop(['Unnamed: 0','work_year','salary','salary_currency','company_location'],axis = 1)
y = df['salary_in_usd']


# df = df.drop(['work_year','company_location','salary_currency','salary'],axis = 1)
# X = df.loc[:, df.columns != 'salary_in_usd']
# X = X.drop(['Unnamed: 0','experience_level','employment_type','employee_residence','remote_ratio','company_size'],axis = 1)
# y = df['salary_in_usd']
# X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25, random_state = 42)


# In[8]:


X.head()


# # Modeling

# In[9]:


X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.25, random_state = 42)
lm = LinearRegression()
lm.fit(X_train, y_train)
y_pred = lm.predict(X_test)
mean_squared_error(y_test,y_pred)

# lm = LinearRegression()
# lm.fit(X_train, y_train)
# y_pred = lm.predict(X_test)
# mean_squared_error(y_test,y_pred)
# pickle.dump(lm, open('model.pkl','wb'))


# In[15]:

pickle.dump(lm, open('model.pkl','wb'))
flask_app = Flask(__name__)
model = pickle.load(open('model.pkl','rb'))

@flask_app.route('/')
def home():
      return render_template('flask_app.html')

@flask_app.route('/predict',methods = ['POST'])
def predict():

      int_features = [int(x) for x in request.form.values()]
      final_features = [np.array(int_features)]
      prediction = model.predict(final_features)
      return render_template('flask_app.html',prediction_text = 'The salary will be $ {}'.format(prediction))



if __name__ == '__main__':
      from waitress import serve
      flask_app.run(port=5000,debug = True)
```

```
<!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 InputFOr 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>
    <img src="/static/images/Original.svg" style="width: 400px;position: absolute;bottom: 10px;left: 10px;" alt="Company Log
</body>
</html>
```