**My**

**First Machine Learning API:**

**Employee Attrition**

**By**

**Sambit Nath**

**About the API**

The intention of this project is to build a simple API that is in the form of a web application. The application takes various parameters of an employee such as working hours, salary, satisfaction level etc. and the ML algorithm that is present at the back end, predicts whether the employee will stay or leave the company.

**Prerequisites (python libraries and frameworks):**

Pandas (for Machine Leraning Model), Scikit Learn, Flask (for API)

**Project files involved:**

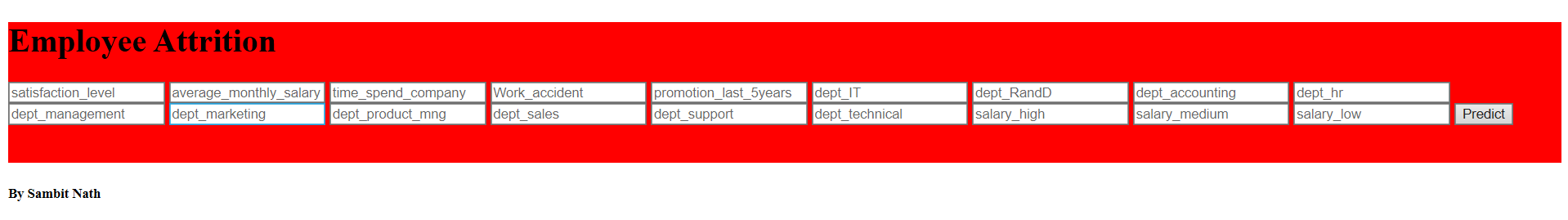
**employee\_attrition2.py** : This file contains the code which fits the input data in a ML algorithm and predicts the class which the employee belongs to. ‘1’ if the employee stays, ‘0’ if the employee leaves the company. The pickle function converts the code into a serialized object.

**flask\_app.py :** This file contains the code which first imports the serialized object (converted through pickle previously). The input values taken from the user by requests.form and sends the predicted outcome for rendering to GUI.html.

**GUI.html:** This file is the frontend html page which acts as an interface to the user where the input parameters are taken and the predicted result is displayed.

**How does the API work?**

1. The login page of the API where the user inputs the details of the employee. The screenshot of the page is as shown below:

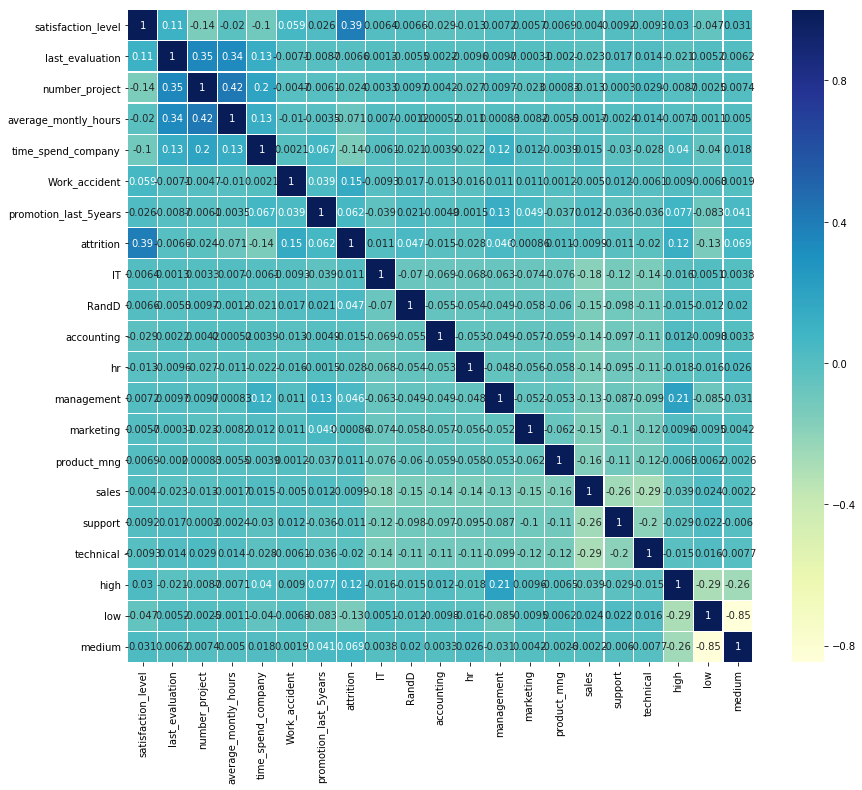


1. The inputs are written and the user clicks on the submit button. The algorithm present in the model.py file predicts the output and presents to the user (circled in the figure below). The screenshot is shown below:

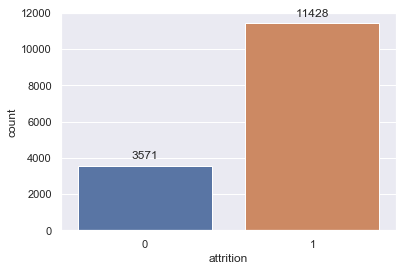


**Data preprocessing, visualization and model creation**

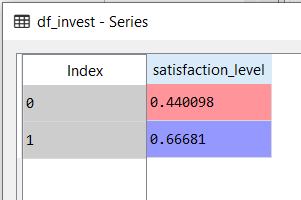
1. The data is comprised of two csv files of which one consists of the information about the employees who would be leaving the company and the ones who would be staying. The two files are merged into one for fitting the data into the algorithm.
2. We first check for null values in our dataset if there is any. None were found in this case.
3. It can be seen that the employee ID column is insignificant for the model, hence it is deleted.
4. The department and the salary column have non numerical values, so they are one hot encoded. Now we have the relevant sparse matrix in our dataset.
5. Now we plot a heat map graph to visualize the correlation between the variables (both independent and dependent). On observing it is found that there is high correlation of the ‘number\_projects’ and ‘last\_evaluation’ columns with other independent variables, hence they are dropped.



1. We check for outliers if there is any. None were found for all continuous variables.
2. Now we check if the data is imbalanced or not. The positive to negative classification ratio is 1:3 approximately. This seems to be a normal scenario.



1. Previously from the heat map it was observed that there is high correlation between the independent variable ‘satisfaction\_level’ and the target variable ‘attrition’, so in order to investigate further we frame a table showing the ‘attrition’ of the employees grouped on the basis of the ‘satisfatcion\_level’ average.



1. Next we split the dataset into training and testing sets and fit it into three different model and find their corresponding confusion matrices to check their accuracy. The accuracy is found to be maximum in the Decision tree classifier.
2. In order to further improve the model I have applied a cross validation technique to check the accuracy of the model. It came around 96% which is high. If we want to improve further we can apply hyperparameter tuning techniques like grid search.

**Working structure**

The ‘employee\_attrition2’ file is first run. In this file the data is preprocessed and the model is trained and finally the accuracy is checked and model validation is done. The file is then converted into a serialized object by using the ‘pickle’ function.

The ‘flask\_api’ file consists of the end point URL where the API is being hosted. This is done by using the flask micro framework of python.

Now when we need to run the application we first run the file ‘employee\_attrition2’ and go to the URL [http://localhost:5000](http://localhost:5000/) which runs the application on port 5000.

flask\_app.py (takes in data from the user through requests.form and sends the predicted class of the employee to the user interface

employee\_attrition2.py (sends serialized object by using pickle function)

GUI.html (the user interface of the API)