

Employee Attrition Prediction

Introduction

Employee attrition is when an employee leaves the company through any method, including voluntary resignations, layoffs, failure to return from a leave of absence, or even illness or death. Whenever anyone ceases working for the company for any reason *and is not replaced for a long time (if ever), that would be employee attrition.*

There are two main types of employee attrition:

Voluntary attrition: When an employee chooses to leave the company, that is voluntary attrition. This can include any reason an employee leaves on their own accord, whether it's truly voluntary or not. True voluntary terminations, such as resignations for a new job or to move across the country, are the ones you're probably most familiar with. But an employee who leaves due to health reasons or only quits because the work situation is toxic can also fall under voluntary attrition. The company retains the decision not to replace the employee—although there are some times the company would like to replace someone but cannot.

Involuntary attrition: When the company decides to part ways with an employee, this is involuntary attrition. This can be through a position elimination, for example, due to reorganization or layoffs, for cause (such as stealing or fighting), poor performance, or termination when someone abandons their job. (You can argue the last one is a voluntary termination, but the company makes the final call to terminate.) The company then doesn't backfill the position or eliminates it.

Literature Survey

Existing Problem

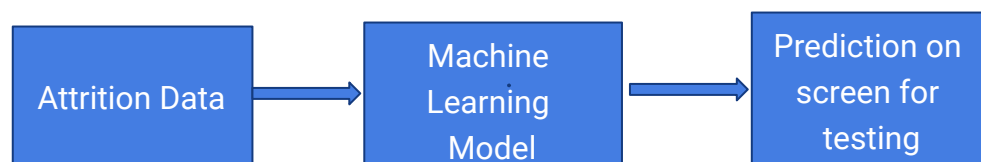
Attrition in human resources refers to the gradual loss of employees over time (resignation, retirement etc.). A major problem in high employee attrition is its cost to an organization. Job postings, hiring processes, paperwork and new hire training are some of the common expenses of losing employees and replacing them. Additionally, regular employee turnover prohibits your organization from increasing its collective knowledge base and experience over time. Human resource analytics (HR analytics) should be implemented so as to overcome this problem.

Proposed solution

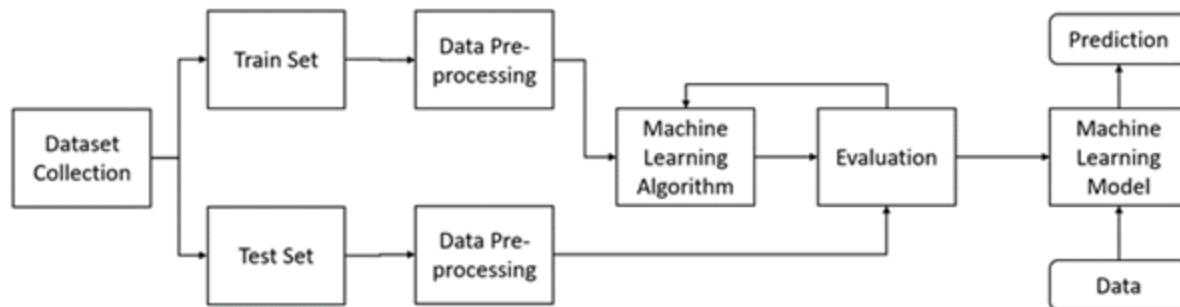
By using Machine learning model we are try to predict which valuable employees are probable to leave the organization subsequently so as to find the areas where the organization is lagging behind. This model can be used by the Human Resource departments of the organizations to form efficient strategies to retain the valuable employees before they start looking for new jobs like by providing a hike in their salary, offering promotions if necessary travel and stay abroad or start the hiring process.

Theoretical Analysis

Block Diagram:



Machine Learning Workflow:



Project Flow :

- User interacts with the UI (User Interface) to enter the current attrition data.
- Entered data are analyzed and predictions are made based on interpretation that whether employee will be attrited or not.
- Predictions are popped onto the UI.

Data Collection

The given data set is related to Taxi Fares. It was taken from the website [kaggle.com](https://www.kaggle.com). The website provides various datasets from various domains.

Data pre-processing

Importing required Libraries :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter as c
from sklearn.preprocessing import LabelEncoder
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
```

- **Pandas:** It is a python library mainly used for data manipulation.
- **NumPy:** This python library is used for numerical analysis.
- **Matplotlib and Seaborn:** Both are the data visualization library used for plotting graph which will help us for understanding the data.
- **Accuracy score:** used in classification type problem and for finding accuracy it is used.
- **R2 Score:** Coefficient of Determination or R^2 is another metric used for evaluating the performance of a regression model. The metric helps us to compare our current model with a constant baseline and tells us how much our model is better.
- **Train_test_split:** used for splitting data arrays into training data and for testing data.
- **joblib:** to serialize your machine learning algorithms and save the serialized format to a file.
- **Scikit-learn (Sklearn)** is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python.

Importing the dataset :

```
[2] data=pd.read_csv(r"/content/drive/MyDrive/WA_Fn-UseC_-HR-Employee-Attrition.csv")
```

```
[3] data.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Ger
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	2	Fei
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	3	I
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	4	I
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	4	Fei
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	1	I

- You might have your data in .csv files, .excel files or .tsv files or something else. But the goal is the same in all cases. If you want to analyse that data using pandas, the first step will be to read it into a data structure that's compatible with pandas.
- Let's load a .csv data file into pandas. There is a function for it, called **read_csv()**. We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).
- Path names on Windows tend to have backslashes in them. But we want them to mean actual backslashes, not special characters.

Label Encoding :

Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with [Machine Learning algorithms](#) too. We need to convert each text category to numbers in order for the machine to process those using mathematical equations.

How should we handle categorical variables? There are Multiple way to handle, but will see one of it is LabelEncoding.

- **Label Encoding** is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

```
[15] le=LabelEncoder()
data1['Attrition']=le.fit_transform(data1['Attrition'])
data1['BusinessTravel']=le.fit_transform(data1['BusinessTravel'])
data1['Department']=le.fit_transform(data1['Department'])
data1['EducationField']=le.fit_transform(data1['EducationField'])
#data1['Gender']=le.fit_transform(data1['Gender'])
data1['JobRole']=le.fit_transform(data1['JobRole'])
data1['MaritalStatus']=le.fit_transform(data1['MaritalStatus'])
data1['OverTime']=le.fit_transform(data1['OverTime'])
```

```
[23] data1.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	HourlyRate	JobLevel	JobRole	MaritalStatus	MonthlyIncome
0	41	1	2	1102	2	1	2	1	94	2	7	2	5993
1	49	0	1	279	1	8	1	1	61	2	6	1	5130
2	37	1	2	1373	1	2	2	4	92	1	2	2	2090
3	33	0	1	1392	1	3	4	1	56	1	6	1	2909
4	27	0	2	591	1	2	1	3	40	1	2	1	3468

Splitting Data into Train and Test:

- When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.
- Now split our dataset into train set and test using train_test_split class from scikit learn library.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

+ Code + Text

```
[111] print(x_train.shape)
      print(x_test.shape)
      print(y_train.shape)
      print(y_test.shape)

(1029, 25)
(441, 25)
(1029, 1)
(441, 1)
```

Model Building

Training and testing the model:

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms or Regression algorithms.

Example: 1. Linear Regression.

2. Logistic Regression.

3. Random Forest Regression / Classification.

4. Decision Tree Regression / Classification.

You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate.

Now we apply Logistic regression algorithm on our dataset.

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In **logistic regression**, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

```
[127] logr=LogisticRegression(max_iter = 500,n_jobs=8)
      logr.fit(x_train,y_train)
      y_test
```

```
▶ y_pred_logr=logr.predict(x_test)
  y_pred_logr
```

```
[129] data['Attrition'].value_counts()
```

```
[131] from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

```
[132] confusion_matrix = confusion_matrix(y_test, y_pred_logr)
      print(confusion_matrix)
```

```
[[366  5]
 [ 58 12]]
```

```
[133] acc_logr=accuracy_score(y_test,y_pred_logr)
      acc_logr
```

```
0.8571428571428571
```

```
[134] print(classification_report(y_test, y_pred_logr))
```

	precision	recall	f1-score	support
0	0.86	0.99	0.92	371
1	0.71	0.17	0.28	70
accuracy			0.86	441
macro avg	0.78	0.58	0.60	441
weighted avg	0.84	0.86	0.82	441

Support Vector Machine

“Support Vector Machine” (SVM) is a supervised [machine learning algorithm](#) which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well


```

svm=SVC()
svm.fit(x_train,y_train)

[136] y_pred_svm=svm.predict(x_test)

[137] from sklearn.metrics import confusion_matrix,classification_report,accuracy_score

[138] confusion_matrix = confusion_matrix(y_test, y_pred_svm)
      print(confusion_matrix)

      [[371  0]
       [ 70  0]]

[139] acc_svm=accuracy_score(y_test,y_pred_svm)
      acc_svm

      0.8412698412698413

[140] print(classification_report(y_test,y_pred_svm))

              precision    recall  f1-score   support

         0       0.84        1.00        0.91        371
         1       0.00        0.00        0.00         70

   accuracy          0.84
  macro avg       0.42        0.50        0.46
 weighted avg       0.71        0.84        0.77

```

K-Nearest Neighbors

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.



```
knn=KNeighborsClassifier()  
knn.fit(x_train,y_train)
```

```
[142] y_pred_knn=knn.predict(x_test)  
y_pred_knn
```

```
[143] from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

```
[144] confusion_matrix(y_test,y_pred_knn)
```

```
array([[355, 16],  
       [ 65,  5]])
```

```
[145] acc_knn=accuracy_score(y_test,y_pred_knn)  
acc_knn
```

```
0.8163265306122449
```

```
[146] print(classification_report(y_test,y_pred_knn))
```

	precision	recall	f1-score	support
0	0.85	0.96	0.90	371
1	0.24	0.07	0.11	70
accuracy			0.82	441
macro avg	0.54	0.51	0.50	441
weighted avg	0.75	0.82	0.77	441

Predict the values:

Once the model is trained, it's ready to make predictions. We can use the **predict** method on the model and pass **x_test** as a parameter to get the output as **pred**.

Notice that the prediction output is an array of real numbers corresponding to the input array.

```
y_pred_logr=logr.predict(x_test)
y_pred_logr
```

```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,  
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,  
       0])
```

Evaluation:

Finally, we need to check to see how well our model is performing on the test data. There are many evaluation techniques are there. For this, we evaluate **accuracy_score** produced by the model.

```
[133] acc_logr=accuracy_score(y_test,y_pred_logr)
      acc_logr
```

```
0.8571428571428571
```

```
print(classification_report(y_test, y_pred_logr))
```

	precision	recall	f1-score	support
0	0.86	0.99	0.92	371
1	0.71	0.17	0.28	70
accuracy			0.86	441
macro avg	0.78	0.58	0.60	441
weighted avg	0.84	0.86	0.82	441

Saving a model:

Model is saved so it can be used in future and no need to train it again.

```
[ ] import pickle
    pickle.dump(logr,open('Attrition.pkl','wb'))
```

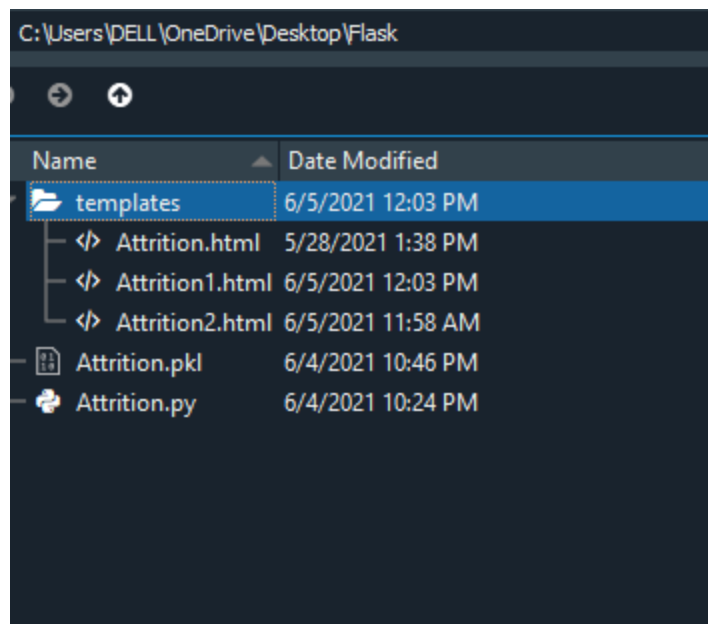
Application Building:

Creating a HTML File, flask application.

- Build python code
- Importing Libraries
- Routing to the html Page
- Showcasing prediction on UI
- Run The app in local browser.

Project Structure:

Create a Project folder that contains files as shown below



- We are building a Flask Application that needs HTML pages stored in the templates folder
- Templates folder contains index.html

Task 1: Importing Libraries

```
from flask import Flask, request, render_template
import pickle
import numpy as np
```

Task 2: Routing to the html Page :

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

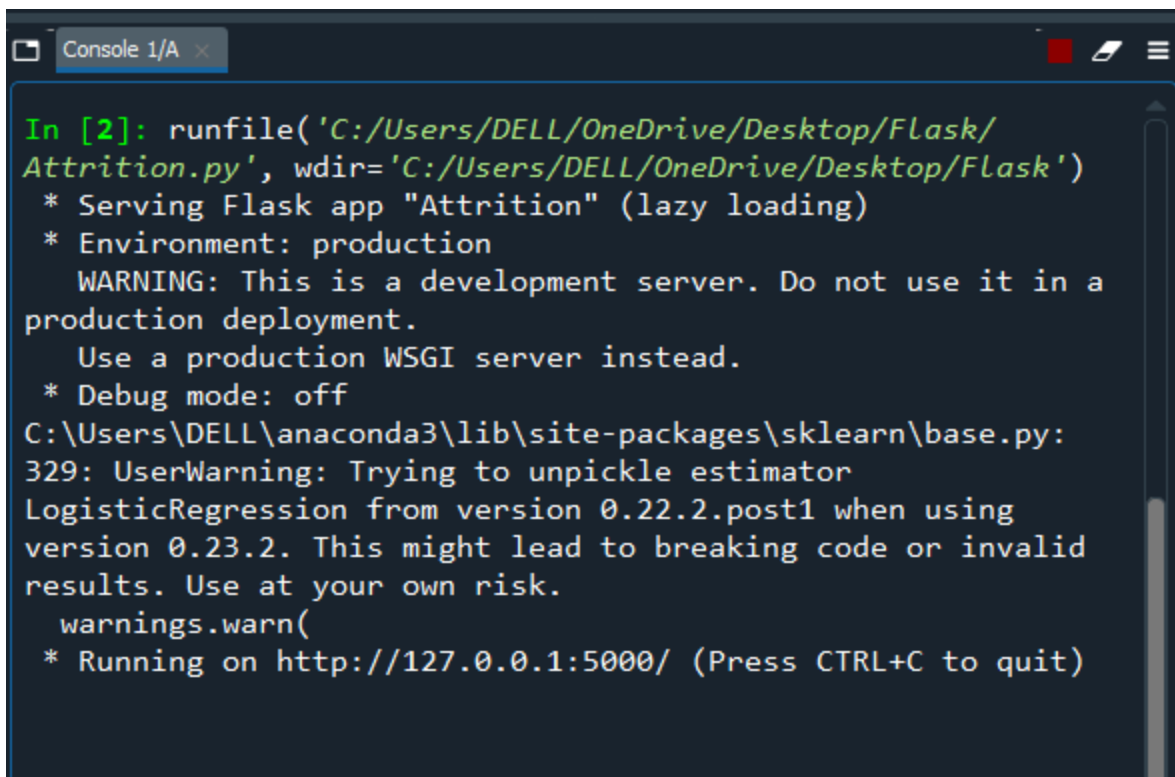
    ag = int(request.form["Age"])
    bt = int(request.form["BusinessTravel"])
    dr = int(request.form["DailyRate"])
    dpt = int(request.form["Department"])
    dfh = int(request.form["DistanceFromHome"])
    edu = int(request.form["Education"])
    ef = int(request.form["EducationField"])
    #gen = request.form["Gender"]
    hr = int(request.form["HourlyRate"])
    jl = int(request.form["JobLevel"])
    jr = int(request.form["JobRole"])
    ms = int(request.form["MaritalStatus"])
    mi = int(request.form["MonthlyIncome"])
    mr = int(request.form["MonthlyRate"])
    ncw = int(request.form["NumCompaniesWorked"])
    ot = int(request.form["OverTime"])
    psh = int(request.form["PercentSalaryHike"])
    pr = int(request.form["PerformanceRating"])
    sol = int(request.form["StockOptionLevel"])
    twh = int(request.form["TotalWorkingYears"])
    ttly = int(request.form["TrainingTimesLastYear"])
    yac = int(request.form["YearsAtCompany"])
    yicr = int(request.form["YearsInCurrentRole"])
    yslp = int(request.form["YearsSinceLastPromotion"])
    ywcm = int(request.form["YearsWithCurrManager"])
    ts = int(request.form["Total_Satisfaction"])
    a=np.array([[ag,bt,dr,dpt,dfh,edu,ef,hr,jl,jr,ms,mi,mr,ncw,ot,psh,pr,sol,twh,ttly,yac,yicr,yslp,ywcm,ts]])
    print(a)
    pred=logr.predict(a)
    if(pred == 0):
        output = "Not Attrited"
        print("Not Attrited")
    else:
        output = "Attrited"
        print("Attrited")
    return render_template('Attrition1.html', prediction_text= output)
```

Task 3: Main Function :

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

Activity 3: Run the application

- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type “python app.py” command.
- It will show the local host where your app is running on <http://127.0.0.1:5000/>
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page.



```
Console 1/A x
In [2]: runfile('C:/Users/DELL/OneDrive/Desktop/Flask/Attrition.py', wdir='C:/Users/DELL/OneDrive/Desktop/Flask')
* Serving Flask app "Attrition" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression from version 0.22.2.post1 when using version 0.23.2. This might lead to breaking code or invalid results. Use at your own risk.
  warnings.warn(
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Output Screen:

Enter Age	Choose a BusinessTravel:
<input type="text"/>	<input type="text" value="SELECT"/>
Enter DailyRate	Choose a Department:
<input type="text"/>	<input type="text" value="SELECT"/>
Enter Distance From Home	Enter Education
<input type="text"/>	<input type="text"/>
Choose a Education Field:	Enter Hourly Rate
<input type="text" value="SELECT"/>	<input type="text"/>
Enter Job Level	Choose a Job Role
<input type="text"/>	<input type="text" value="SELECT"/>
Choose a Marital Status	Enter Monthly Income
<input type="text" value="SELECT"/>	<input type="text"/>
Enter Monthly Rate	Enter Num Companies Worked
<input type="text"/>	<input type="text"/>
Choose Over Time	Enter Percent Salary Hike
<input type="text" value="SELECT"/>	<input type="text"/>
Enter Performance Rating	Enter Stock Option Level
<input type="text"/>	<input type="text"/>
Enter Total Working Years	Enter Training Times Last Year
<input type="text"/>	<input type="text"/>
Enter Years At Company	Enter Years In Current Role
<input type="text"/>	<input type="text"/>
Enter Years Since Last Promotion	Enter Years With Curr Manager
<input type="text"/>	<input type="text"/>
Enter Total Satisfaction	
<input type="text"/>	

Enter Age	Choose a BusinessTravel:
<input type="text"/>	<input type="text" value="SELECT"/>
Enter DailyRate	Choose a Department:
<input type="text"/>	<input type="text" value="SELECT"/>
Enter Distance From Home	Enter Education
<input type="text"/>	<input type="text"/>
Choose a Education Field:	Enter Hourly Rate
<input type="text" value="SELECT"/>	<input type="text"/>
Enter Job Level	Choose a Job Role
<input type="text"/>	<input type="text" value="SELECT"/>
Choose a Marital Status	Enter Monthly Income
<input type="text" value="SELECT"/>	<input type="text"/>
Enter Monthly Rate	Enter Num Companies Worked
<input type="text"/>	<input type="text"/>
Choose Over Time	Enter Percent Salary Hike
<input type="text" value="SELECT"/>	<input type="text"/>
Enter Performance Rating	Enter Stock Option Level
<input type="text"/>	<input type="text"/>
Enter Total Working Years	Enter Training Times Last Year
<input type="text"/>	<input type="text"/>
Enter Years At Company	Enter Years In Current Role
<input type="text"/>	<input type="text"/>
Enter Years Since Last Promotion	Enter Years With Curr Manager
<input type="text"/>	<input type="text"/>
Enter Total Satisfaction	
<input type="text"/>	

Submit

Output:Not Attrited

Findings and Suggestions

Through Exploratory Data Analysis,

- The Accuracy for logistic regression is 85.7%.
- The Accuracy for svm classifier is 84.1%.

Conclusion

Predictive Attrition Model helps in not only taking preventive measures but also into making better hiring decisions. Deriving trends in the candidate's performance out of past data is important in order to predict the future trends, as well as to board new employees. Moreover, HR can use the employee data to predict attrition, the possible reasons behind it and can take appropriate measures to prevent it.

We live in a data-driven world – from something as trivial as weather updates to complexity behind GPS navigations, data is constantly being generated every second in every field, which leaves us to decide how to turn it around for our advantage in our respective domains.

Reference

- 1. www.kaggle.com
- 2. www.quora.com
- 3. www.wikipedia.com

