Employee Attrition Prediction: From Data to Deployment

Ashmitha Maduri

Agenda

1. Introduction
2. Problem Definition
3. Path to Predictive Insights
    - Data Collection + Data Cleaning
    - Data Visualization + Data Analysis
    - Feature Engineering
    - Model Building
    - Model Deployment
4. Conclusion

Introduction

Employee attrition, the departure of employees from an organization, poses a critical challenge for businesses globally.

Its implications extend beyond talent loss to encompass productivity setbacks and financial implications.

This project focuses on addressing this challenge through various stages, including data collection, preprocessing, and model deployment.

By utilizing advanced techniques, the aim is to help organizations identify potential attrition cases and implement proactive strategies for employee retention.

Problem Definition

The challenge of employee attrition is impacting businesses across industries. The departure of valuable talent leads to increased costs and reduced productivity.

To address this, a robust machine learning model is needed to predict attrition accurately, enabling proactive retention strategies.

By solving this problem, organizations can optimize workforce management, reduce attrition rates, and achieve sustainable growth in a competitive landscape.

Navigating Employee Attrition: Unveiling the Path to Predictive Insights

Dataset - https://www.kaggle.com/datasets/giripujar/hr-analytics?resource=download

Data Collection
+
Data Cleaning

```
# Feature Engineering
import pandas as pd
import numpy as np
from sklearn import preprocessing

# Exploratory Data Analysis
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# ML model
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score, confusion_matrix
from sklearn.model_selection import GridSearchCV

# Warnings
import warnings
warnings.filterwarnings('ignore')
```

Importing the Libraries

```
data = pd.read_csv("/content/HR_comma_sep.csv")
data
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium
3	0.72	0.87	5	223	5	0	1	0	sales	low
4	0.37	0.52	2	159	3	0	1	0	sales	low
...	...	...	...	...	...	...	...	...	...	...
14994	0.40	0.57	2	151	3	0	1	0	support	low
14995	0.37	0.48	2	160	3	0	1	0	support	low
14996	0.37	0.53	2	143	3	0	1	0	support	low
14997	0.11	0.96	6	280	4	0	1	0	support	low
14998	0.37	0.52	2	158	3	0	1	0	support	low

14999 rows × 10 columns

Loading the Data

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000

Standard Deviation in `average_monthly_hours` is very high at 49.9 this indicates that there is a lot of variance in the observed data around the mean/second quartile.

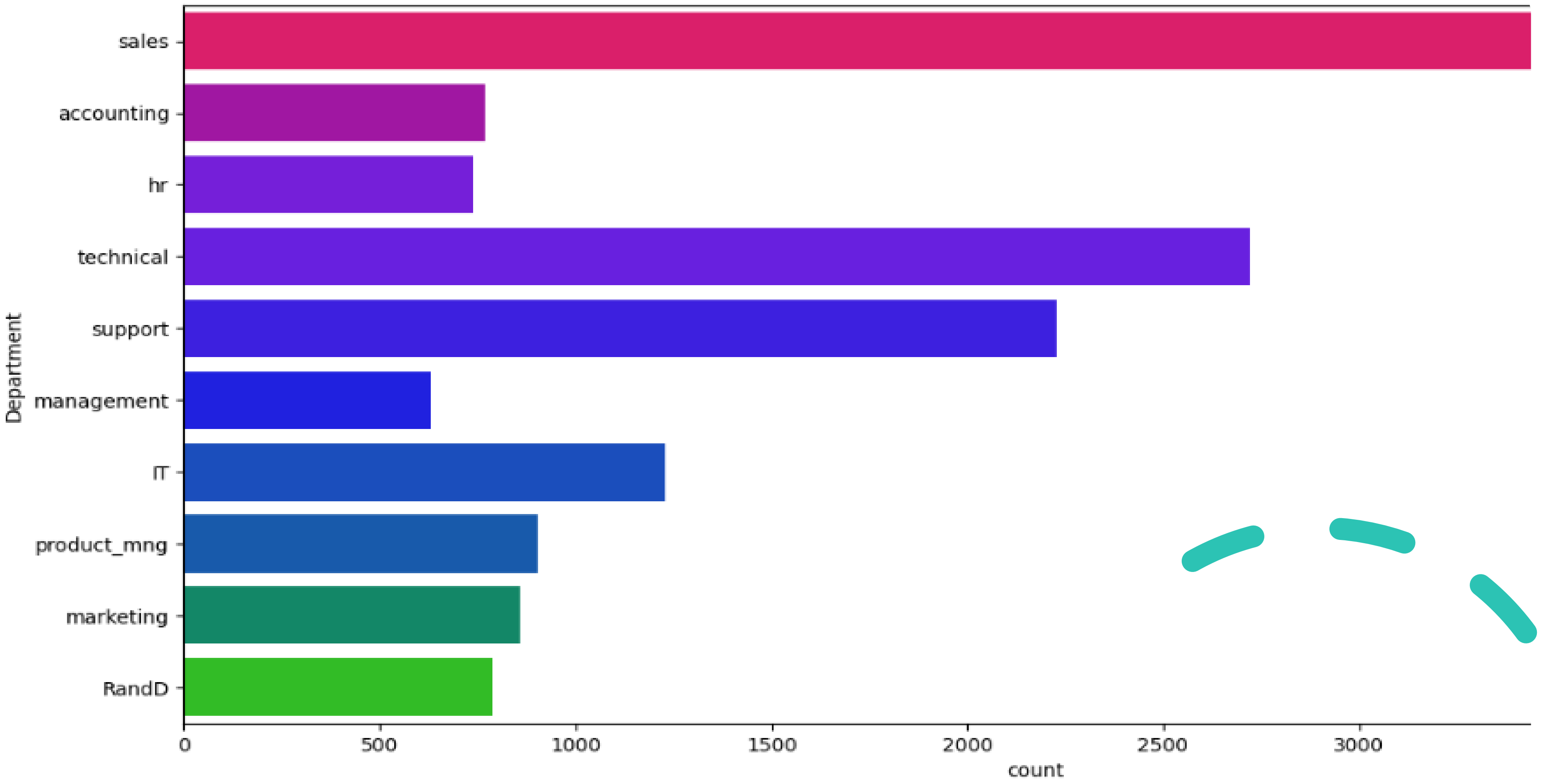
`time_spend_company` has a maximum time of 10 years but its second quartile (mean/50%) explains that 50% of the overall workforce has spent only 3 years in the company and its upper quartile(75%) shows that there is only 25% of the employees who spent more than 4 years.

Some skewness/outliers can be expected as a result.

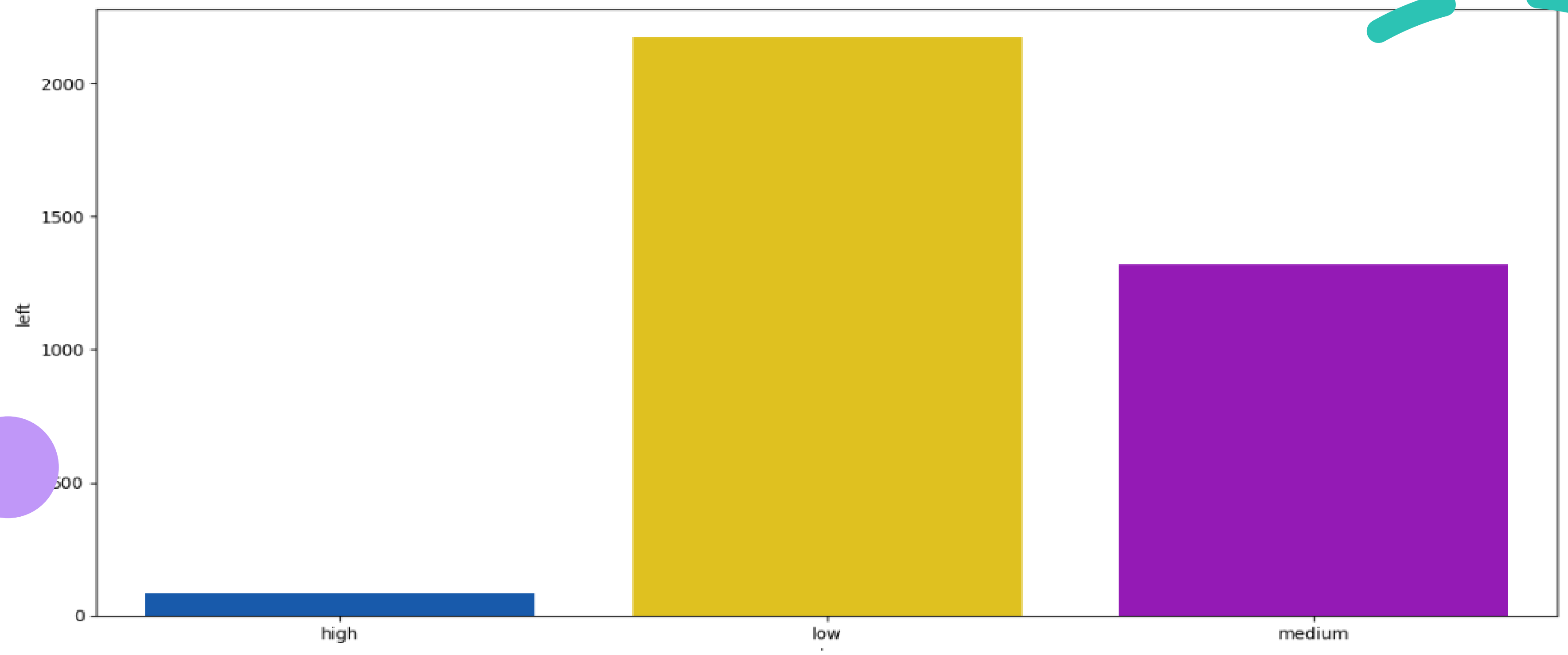
Statistical Analysis

Data Visualization + Data Analysis

(EDA)



Distribution of the Departments

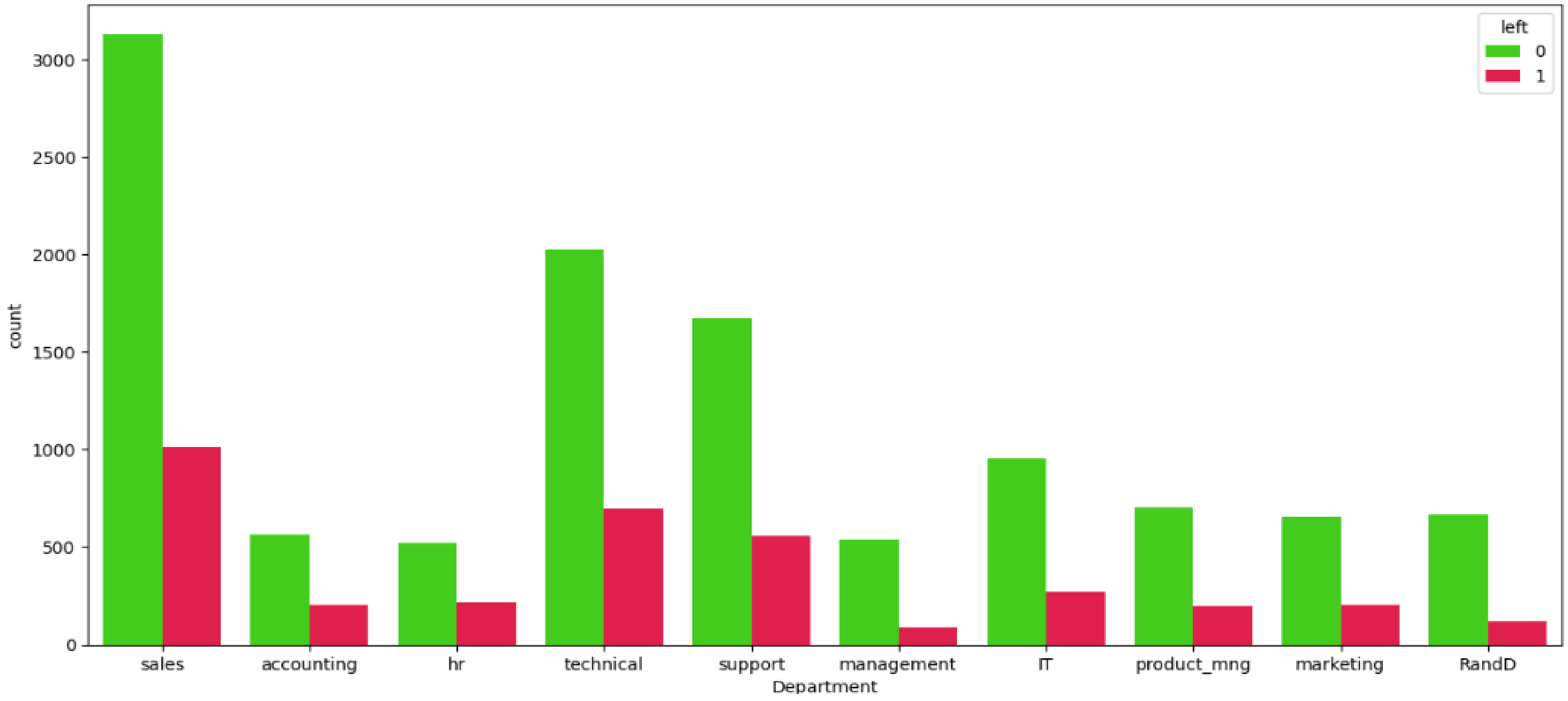


Salary Distribution :

The low to medium range salaried employees tend to leave the company more frequently in comparison with high salaried ones; this could be because the strategic level employees (high salaried) are settled and content with the company unlike (low/medium) salaried ones and therefore this factor could behave as an outlier further.

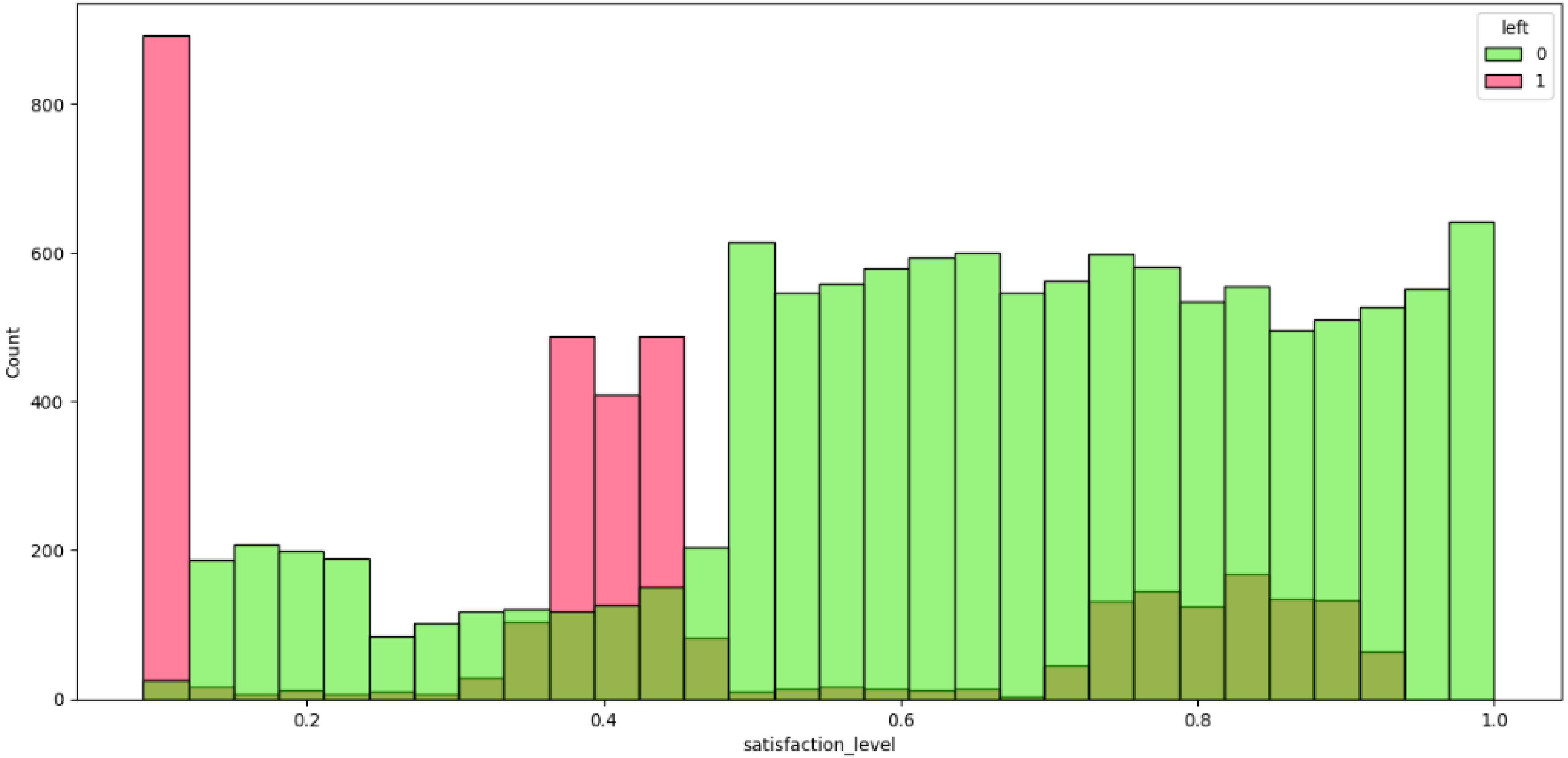
Department wise Attrition :

Out of the total employees who left the company (3571) those who left from Sales, Technical and Support Departments sum up to more than 75% this could be because they are from the functional level of the organisation.



Analysing Satisfaction levels :

It is evident that employees who left the company were not really satisfied and employees who were satisfied didn't leave.



Feature Engineering

Converting categorical features into numerical

1. `Salary`
2. `Departments`

Salary

- 0 - High
- 1 - Low
- 2 - Medium

Departments

- 0 - IT
- 1 - RandD
- 2 - Accounting
- 3 - HR
- 4 - Management
- 5 - Marketing
- 6 - Product Management
- 7 - Sales
- 8 - Support
- 9 - Technical

```
le = preprocessing.LabelEncoder()
data['Salary'] = le.fit_transform(data['salary'])
data['Departments'] = le.fit_transform(data['Department'])
```

```
data = data.drop(['salary', 'Department'], axis = 1)
```

```
print(le.classes_)
```

```
['IT' 'RandD' 'accounting' 'hr' 'management' 'marketing' 'product_mng'
 'sales' 'support' 'technical']
```

```
data.Departments.unique()
```

```
array([7, 2, 3, 9, 8, 4, 0, 6, 5, 1])
```

Ensuring all the columns have the same data type

```
data.dtypes
```

```
satisfaction_level       float64
last_evaluation          float64
number_project             int64
average_montly_hours       int64
time_spend_company         int64
Work_accident              int64
left                       int64
promotion_last_5years      int64
Salary                     int64
Departments                int64
dtype: object
```

```
for col in data.columns:
    if data[col].dtype == 'int64':
        data[col] = data[col].astype('float64')
```

```
data.dtypes
```

```
satisfaction_level       float64
last_evaluation          float64
number_project           float64
average_montly_hours     float64
time_spend_company       float64
Work_accident            float64
left                     float64
promotion_last_5years    float64
Salary                   float64
Departments              float64
dtype: object
```

Scaling + Splitting

```
from sklearn import preprocessing
```

```
Scaler = preprocessing.MinMaxScaler()
```

```
data[['satisfaction_level', 'promotion_last_5years',
      'last_evaluation','average_montly_hours',
      'Work_accident', 'time_spend_company']] = Scaler.fit_transform(data[['satisfaction_level',
                                                                          'promotion_last_5years',
                                                                          'last_evaluation',
                                                                          'average_montly_hours',
                                                                          'Work_accident',
                                                                          'time_spend_company']])
```

```
X = data.drop('left', axis = 1)
y = data['left']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 33)
```

MODEL BUILDING

Training the ML Model

```python
classifiers = {
    'Logistic Regression': (LogisticRegression(),
                            {'C': [0.1, 1, 10]}),


    'SVM': (SVC(),
            {
                'C': [0.1, 1, 10],
             'kernel': ['linear', 'rbf']
             }
            ),


    'Random Forest': (RandomForestClassifier(),
                      {
                          'n_estimators': [20, 60, 90],
                       'max_depth': [3, 5, None]
                       }
                      ),


    'Gradient Boosting': (GradientBoostingClassifier(),
                          {
                              'n_estimators': [50, 150, 240],
                           'learning_rate': [0.1, 0.01, 0.001]
                           }
                          ),


    'XGB Classifier' : (XGBClassifier(),
                        {
                            'max_depth': [2, 4, 6, 8, 10],
                         'learning_rate': [0.1, 0.01, 0.001],
                         'n_estimators': [50, 150, 200]
                         }
                        ),
```

```python
    'XGB Classifier' : (XGBClassifier(),
                        {
                            'max_depth': [2, 4, 6, 8, 10],
                         'learning_rate': [0.1, 0.01, 0.001],
                         'n_estimators': [50, 150, 200]
                         }
                        ),


    'AdaBoost': (AdaBoostClassifier(),
                 {
                     'n_estimators': [70, 170, 270],
                  'learning_rate': [0.1, 0.01, 0.001]
                  }
                 ),


    'Decision Tree Classifier': (DecisionTreeClassifier(),
                                 {
                                     'max_depth': [2, 4, 6, 8, 10],
                                  'min_samples_split': [2, 5, 10, 15, 20],
                                  'min_samples_leaf': [1, 2, 3, 4, 5]
                                  }
                                 )
}
```

Conducting GridSearchCV on each classifier to identify the optimal model with the best hyperparameters

```
for name, (classifier, param_grid) in classifiers.items():
    print(name)
    grid_search = GridSearchCV(classifier, param_grid, cv=5)
    grid_search.fit(X_test, y_test)
    print('Best parameters:', grid_search.best_params_)
    print('Best score:', grid_search.best_score_)
    print('------------------')
```

```
Logistic Regression
Best parameters: {'C': 0.1}
Best score: 0.7843333333333333
------------------
SVM
Best parameters: {'C': 10, 'kernel': 'rbf'}
Best score: 0.8916666666666666
------------------
Random Forest
Best parameters: {'max_depth': None, 'n_estimators': 90}
Best score: 0.9789999999999999
------------------
Gradient Boosting
Best parameters: {'learning_rate': 0.1, 'n_estimators': 150}
Best score: 0.97
------------------
XGB Classifier
Best parameters: {'learning_rate': 0.1, 'max_depth': 8, 'n_estimators': 150}
Best score: 0.9756666666666666
------------------
AdaBoost
Best parameters: {'learning_rate': 0.1, 'n_estimators': 270}
Best score: 0.9443333333333334
------------------
Decision Tree Classifier
Best parameters: {'max_depth': 6, 'min_samples_leaf': 1, 'min_samples_split': 20}
Best score: 0.9736666666666667
------------------
```

```
suitable_models = {
    'Logistic Regression' : LogisticRegression(C = 0.1),
    'XG Boost' : XGBClassifier(learning_rate = 0.1,
                               max_depth = 8,
                               n_estimators = 150),
    'Random Forest' : RandomForestClassifier(n_estimators = 90),
    'Ada Boost' : AdaBoostClassifier(learning_rate = 0.1,
                                     n_estimators = 270),
    'Gradient Boost' : GradientBoostingClassifier(learning_rate = 0.1,
                                                  n_estimators = 240),
    'Support Vector Machine' : SVC(C = 10,
                                   kernel = 'rbf'),
    'Decision Tree Classifier' : DecisionTreeClassifier(max_depth = 6,
                                                        min_samples_leaf = 1,
                                                        min_samples_split = 20)
}
```

	Model	Accuracy	Recall	Precision	F1 Score
0	Logistic Regression	0.773	0.261	0.580	0.360
1	XG Boost	0.986	0.952	0.992	0.972
2	Random Forest	0.991	0.969	0.996	0.982
3	Ada Boost	0.953	0.878	0.927	0.901
4	Gradient Boost	0.977	0.931	0.976	0.953
5	Support Vector Machine	0.912	0.789	0.843	0.815
6	Decision Tree Classifier	0.974	0.916	0.975	0.945

Evaluating the Models

- \`Accuracy\`: Random Forest produced the best test accuracy with its score at 99.1%
- \`Recall\`: Random Forest is the strongest by far with a score of 96.6%. XG Boost is second.
- \`Precision\`: Random Forest is a clear winner followed by XG Boost, respectively.
- \`F1 Score\`: Random Forest is the best amongst all.

Saving + Downloading the Best Model

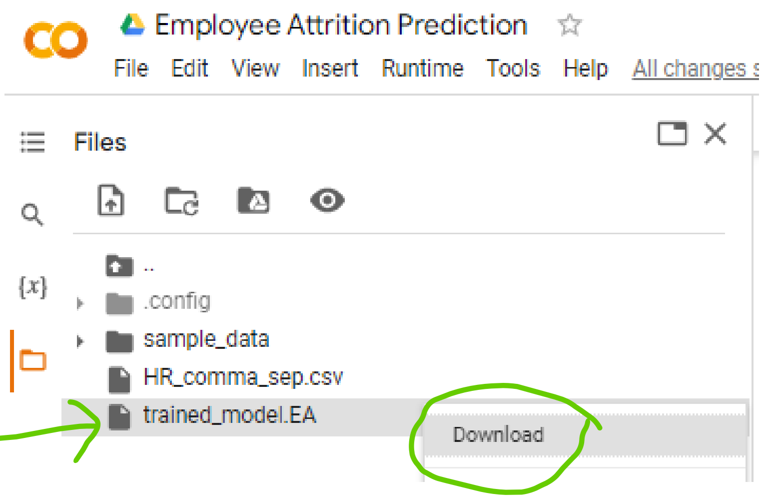
```python
classifier = RandomForestClassifier(n_estimators=90)
classifier.fit(X_train, y_train)
```

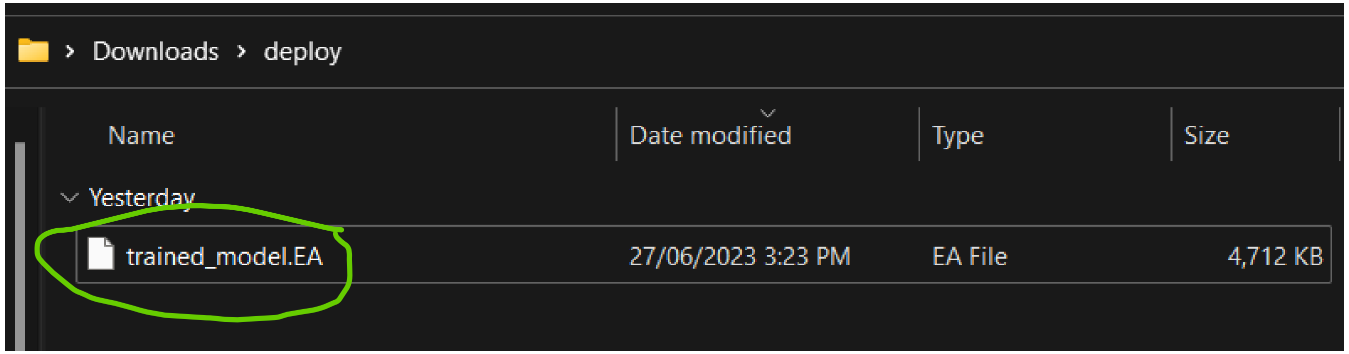
```
▾ RandomForestClassifier
RandomForestClassifier(n_estimators=90)
```

```python
import pickle
```

```python
filename = 'trained_model.EA'
pickle.dump(classifier, open(filename, 'wb'))
```

```python
loaded_model = pickle.load(open("trained_model.EA", 'rb'))
```

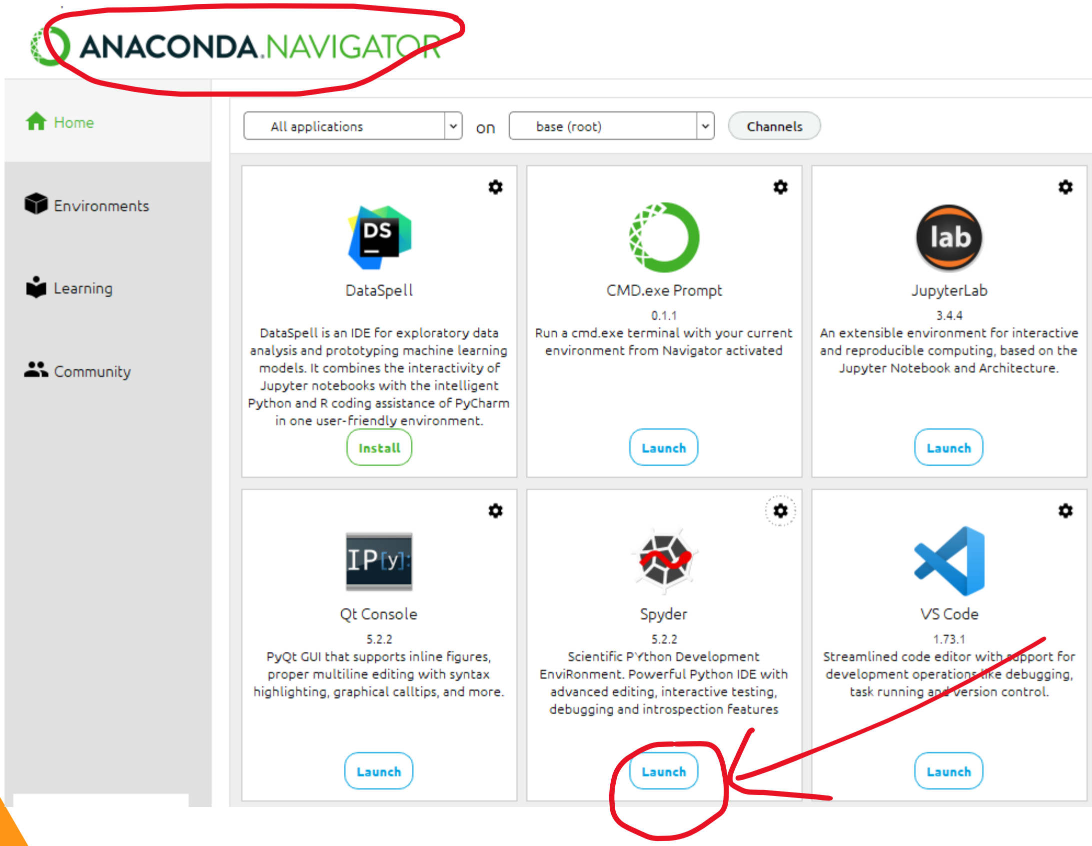


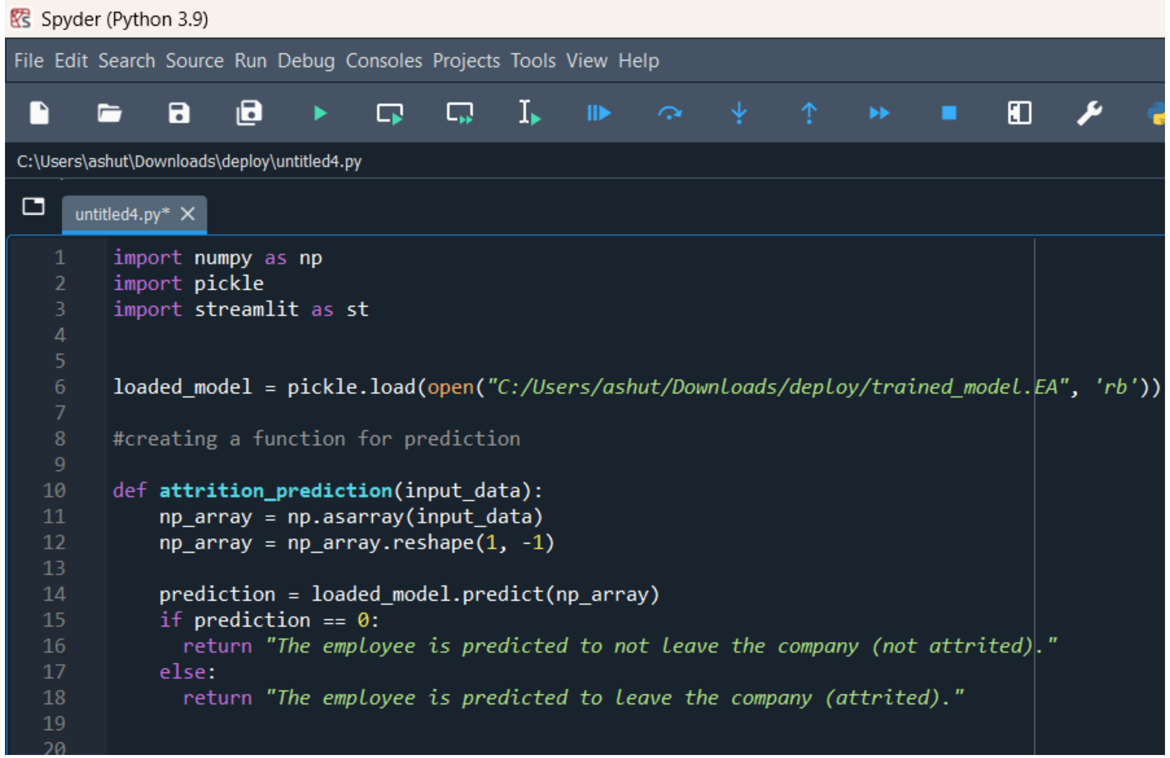


MODEL DEPLOYMENT

Streamlit

Deploying the Model using Streamlit



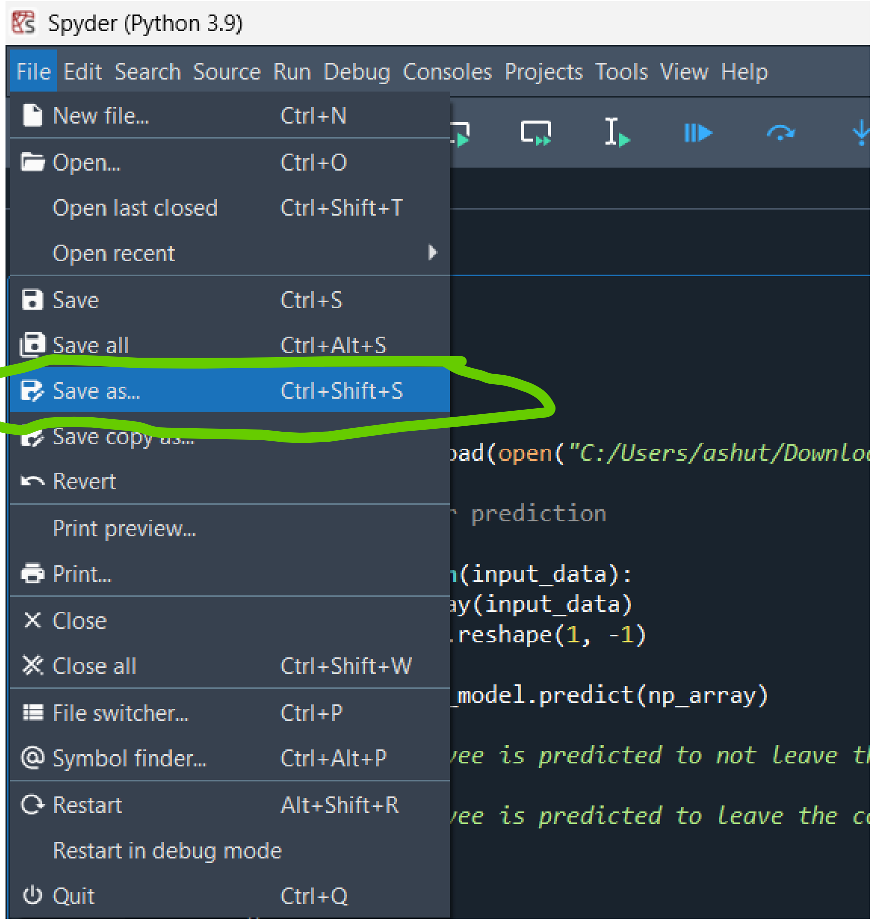


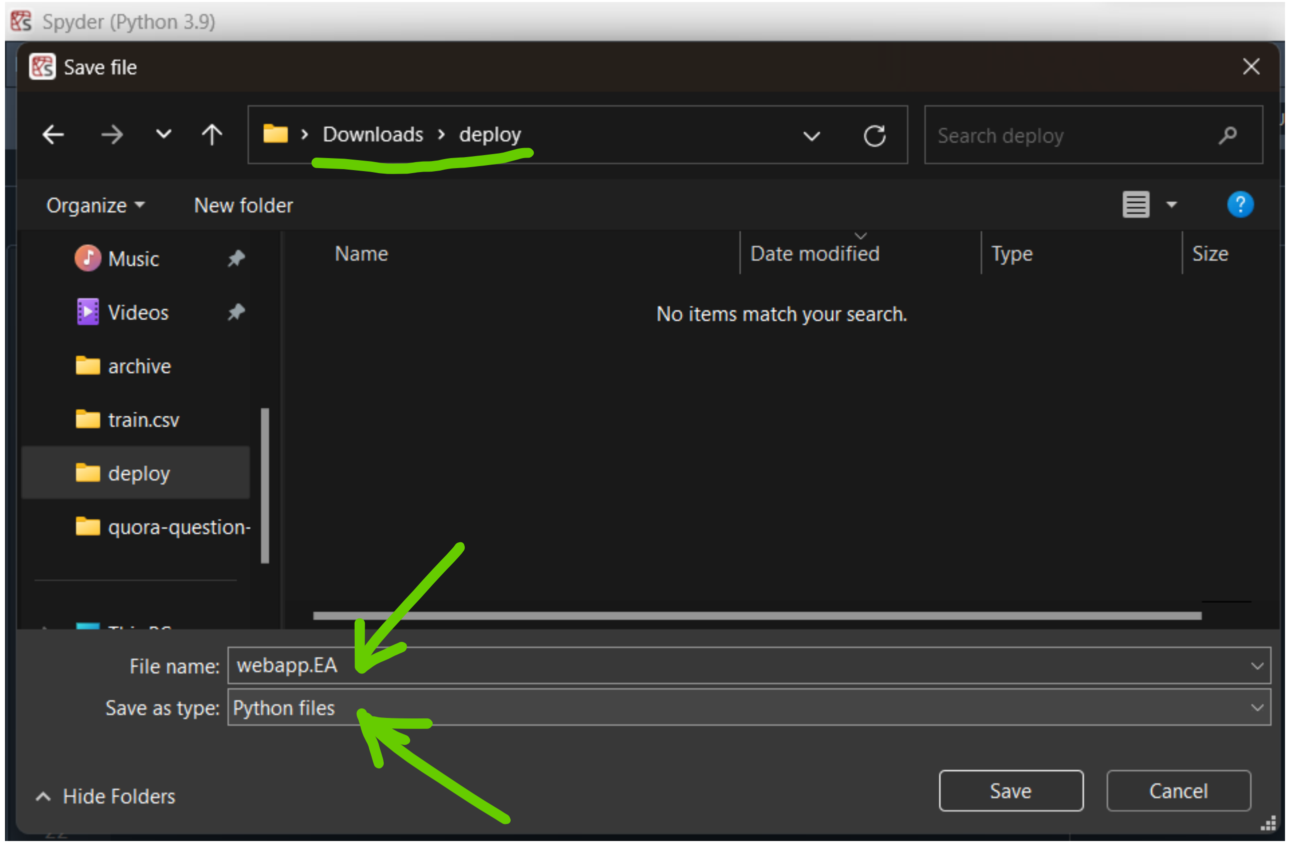
```python
def main():

    # Title
    st.title("Employee Attrition Prediction Web Application")

    # Input Data
    satisfaction_level = st.text_input('Enter employee satisfaction level:')
    last_evaluation = st.text_input("Enter employee's last evaluation:")
    number_project = st.text_input("Number of projects:")
    average_monthly_hours = st.text_input("Employee's average monthly hours:")
    time_spend_company = st.text_input("Enter years at company:")
    Work_accident = st.text_input("Work accident? (1 for Yes, 0 for No):")
    promotion_last_5years = st.text_input("Promotion in the last 5 years? (1 for Yes, 0 for No):")
    Salary = st.text_input("Salary level (0 for High, 1 for Low, 2 for Medium):")
    Departments = st.text_input("Enter the department code (0 for IT, 1 for RandD, 2 for Accounting, 3 for HR, 4 for Management, 5

    # Code for Prediction
    analysis = ''

    if st.button("Employee churn prediction result"):
        if '' in [satisfaction_level, last_evaluation, number_project, average_monthly_hours, time_spend_company, Work_accident, p
            analysis = "Please fill in all the input fields."
        else:
            input_data = [float(satisfaction_level), float(last_evaluation), float(number_project), float(average_monthly_hours),
            analysis = attrition_prediction(input_data)

    st.success(analysis)


if __name__ == '__main__':
    main()
```

Console 2/A

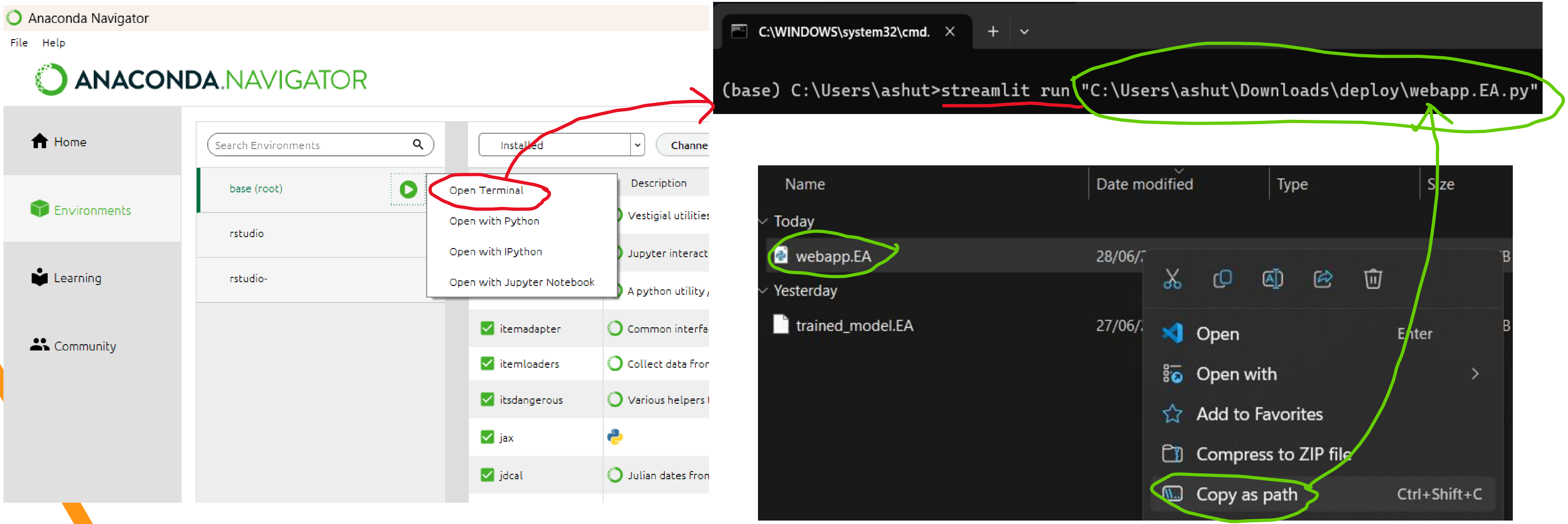
```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/ashut/Downloads/deploy/webapp.EA.py', wdir='C:/Users/ashut/
Downloads/deploy')
2023-06-28 02:29:05.031
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run C:\Users\ashut\Downloads\deploy\webapp.EA.py [ARGUMENTS]
```

Saving the file

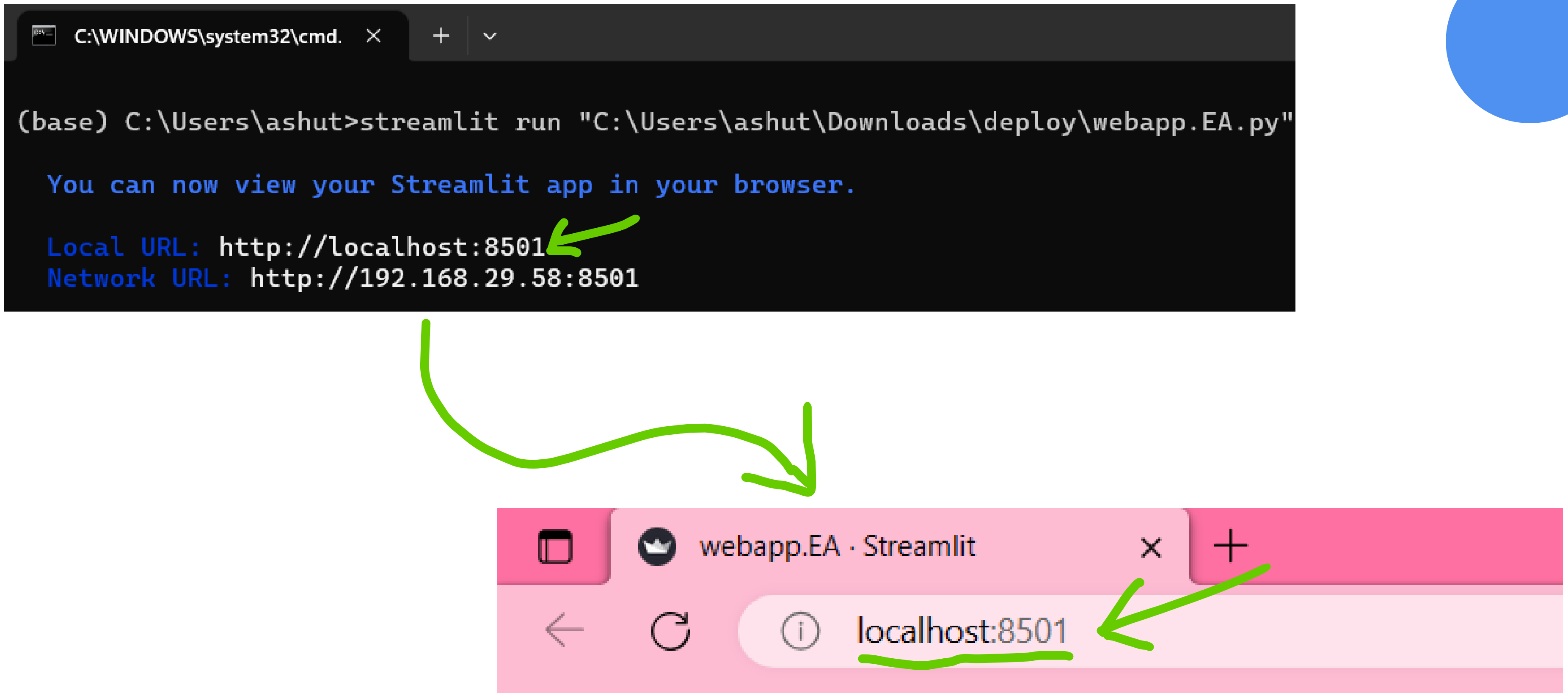




Opening the Terminal for Deployment



Viewing the Web Application



Employee Attrition Prediction Web Application

Filling the inputs

Enter employee satisfaction level:

0.38

Enter employee's last evaluation:

0.53

Number of projects:

2

Employee's average monthly hours:

157

Enter years at company:

3

Work accident? (1 for Yes, 0 for No):

0

Promotion in the last 5 years? (1 for Yes, 0 for No):

1

Salary level (0 for High, 1 for Low, 2 for Medium):

0

Enter the department code (0 for IT, 1 for RandD, 2 for Accounting, 3 for HR, 4 for Management, 5 for Marketing, 6 for Product Management, 7 for Sales, 8 for Support, 9 for Technical):

7

Clicking on this to view the result

Employee churn prediction result

The employee is predicted to leave the company (attrited).

Here's the result

Conclusion

- The developed employee attrition prediction model holds significant value for businesses and organizations in mitigating employee turnover and its associated costs.
- By using this model, organizations can proactively identify employees who are at a higher risk of leaving and implement targeted retention strategies.
- The insights provided by the model can assist decision-makers in making informed human resource management decisions, such as improving employee satisfaction, providing appropriate incentives, or addressing issues within specific departments.

Overall, the model can help organizations optimize their workforce management practices, reduce attrition rates, enhance productivity, and create a more stable and engaged workforce.