

Predicting Employee Attrition using Machine Learning

Harshitha S A (21Z220)

Prathiksha R (21Z235)

Santhoshi R (21Z251)

Aishwariya R (22Z431)

Kamali A (22Z436)

19Z610 – MACHINE LEARNING LABORATORY

report submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF ENGINEERING

BRANCH: Computer Science and Engineering



April 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

TABLE OF CONTENTS

1. Problem Statement	3
2. Dataset Description	3
3. Methodology / Models used.....	4
4. Tools used	5
5. Challenges faced	6
6. Contribution of team mem.....	6
7. ANNEXURE I: Code	7
8. ANNEXURE II: Snapshots of Output	12
9. References	15

1. PROBLEM STATEMENT:

The objective of our project is to develop a web application that leverages machine learning to predict whether an employee is likely to be promoted, based on a range of parameters such as demographic details, performance metrics, and professional attributes. This tool aims to assist human resources departments in making data-driven promotion decisions by offering a predictive insight complemented by an analysis of precision, recall, and accuracy metrics derived from the underlying dataset. By automating the prediction of promotion eligibility, the application endeavors to optimize the promotion process, ensure equitable decision-making, and enhance overall organizational efficiency.

2. DATASET DESCRIPTION:

The dataset comprises **54,808** instances, each representing an individual employee's record within an organization. Each record encapsulates both categorical and numerical attributes:

Categorical Attributes:

- **Department:** The segment of the organization the employee belongs to, with 9 unique departments represented.
- **Region:** The geographical region of the employee's workplace, featuring 34 unique regions.
- **Education:** The highest level of education attained by the employee, with 3 distinct categories.
- **Gender:** The gender of the employee, with two genders present in the dataset.
- **Recruitment Channel:** The channel through which the employee was recruited, with three different channels recorded.

Numerical Attributes:

- **Age:** Reflects the diversity in the workforce with ages ranging from 20 to 60 years.
- **Length of Service:** The tenure of employees in the organization, ranging from 1 to 37 years.
- **KPIs:** Key Performance Indicators, which are binary (0 or 1), indicating whether or not certain targets have been met.
- **Awards Won:** A binary attribute signifying if the employee has won any awards (0 or 1).
- **Average Training Score:** Scores from training sessions, varying between 39 and 99, indicating the skill level post-training.
- **Number of Trainings:** The count of trainings attended, from 1 to 10 sessions.
- **Previous Year Rating:** The performance rating of the previous year, on a scale of 1-5.

The target variable, **is_promoted**, indicates whether an employee was promoted (1) or not (0), which the model predicts.

Dataset link:

- <https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset>
- <https://docs.google.com/spreadsheets/d/1HvEtLnD7LMoKQXK2WDS5q1qKToE3-9e3hsfya27I79Q/edit?usp=sharing>

3. METHODOLOGY / MODELS USED:

The study employs a rigorous machine learning pipeline that encapsulates preprocessing, feature selection, model training, and validation. The methodologies adopted are delineated below:

- **Preprocessing:** Data preprocessing is the first crucial step, involving normalization, encoding of categorical variables through one-hot encoding or label encoding, and handling of missing values. We employ **sklearn.preprocessing** for these tasks, ensuring that the data fed into the models is clean and in a format amenable to machine learning algorithms.
- **Feature Selection:** We utilize both univariate and model-based selection techniques to identify the most predictive features. **SelectKBest** is applied to filter attributes that have the strongest relationship with the target variable. Additionally, we harness feature importance scores from ensemble methods to further refine our feature set.
- **Model Training:** A suite of machine learning models is trained and evaluated. The selection includes:
 - **Logistic Regression:** A baseline model, implemented using **sklearn.linear_model**, provides a probabilistic framework for binary classification.
 - **Decision Trees:** We leverage **sklearn.tree** for constructing decision trees that capture non-linear relationships and interactions between features.
 - **Random Forest:** An ensemble of decision trees from **sklearn.ensemble** that improves prediction robustness through bagging.
 - **Gradient Boosting Machines (GBM):** Advanced ensemble techniques from **sklearn.ensemble**, like GBM, are utilized for their ability to optimize on loss functions directly and handle a variety of data irregularities effectively.
 - **Support Vector Machines (SVM):** Applied for their efficacy in high-dimensional spaces, using **sklearn.svm**.
- **Model Evaluation and Selection:** Each model's performance is evaluated using cross-validation techniques to ensure the models' generalizability. We adopt a range of metrics to assess model efficacy, including Accuracy, Precision, Recall, and the F1 score. The best-performing model is selected based on a balance of these metrics, ensuring a trade-off

between false positives and false negatives, which is critical in the context of employee promotions.

This methodological framework depicted by the block diagram Figure 1 ensures that the study is rooted in robust statistical practices and the predictive model is both accurate and interpretable.

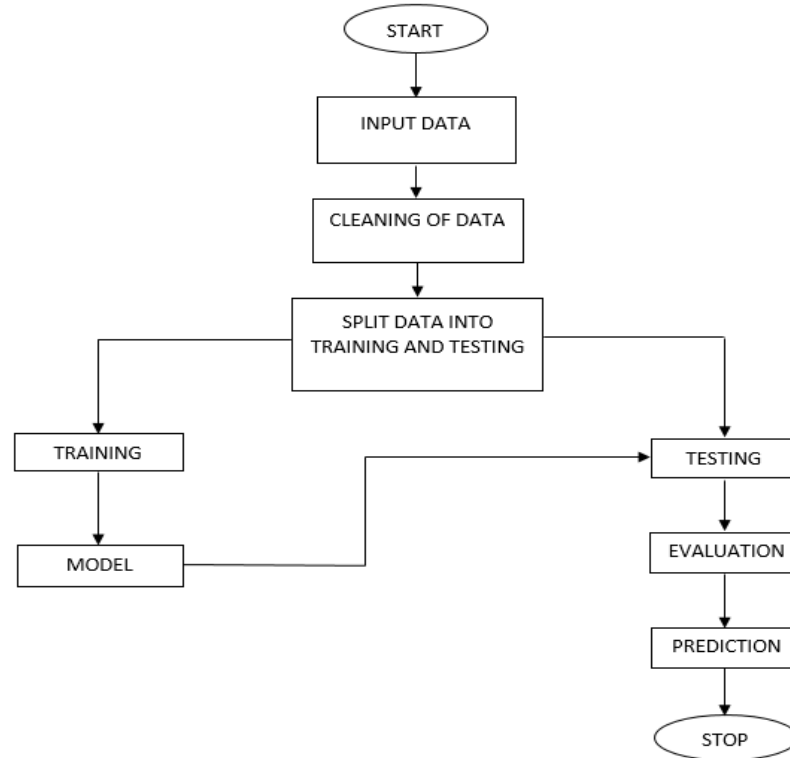


Figure 1 Block diagram of the system

4. TOOLS USED:

Data Manipulation and Analysis:

- **Pandas:** Utilized for data manipulation and analysis; it provided extensive capabilities for data shaping, cleaning, and aggregation.
- **NumPy:** Used for numerical computations, it facilitated efficient processing of arrays and matrices, which are fundamental to machine learning tasks.

Machine Learning Algorithms and Modeling:

- **Scikit-learn:** This library was the cornerstone of our modeling process, providing straightforward access to a wide array of machine learning algorithms, feature selection techniques, and evaluation metrics.
- **XGBoost:** An optimized gradient boosting library that we leveraged for its performance and speed in building ensemble models.
- **LightGBM:** Employed for its efficient implementation of the gradient boosting framework, particularly useful for large datasets.

Model Evaluation:

- **Matplotlib and Seaborn:** These libraries generated visualizations for exploratory data analysis and for representing model evaluation metrics graphically.
- **Scikit-plot:** An extension of Matplotlib, it was used for more advanced visualizations such as ROC curves and confusion matrices.

Web Application Framework:

- **Flask:** A micro web framework for Python that we utilized to deploy the machine learning model as a web application. It allowed us to create a lightweight RESTful API as the backend.
- **HTML/CSS/JavaScript:** Used for the frontend to create an interactive user interface for inputting data and displaying predictions and analytics.

5. CHALLENGES FACED:

- **Data Imbalance:** We used SMOTE to balance our dataset, ensuring fairer promotion predictions.
- **Feature Selection:** We improved our model by carefully selecting and engineering the most impactful features.
- **Model Complexity:** We fine-tuned a Gradient Boosting Machine to be both effective and easy to understand.
- **Overfitting:** We prevented overfitting by applying cross-validation and regularization methods.
- **Deployment:** We resolved scalability issues in our web application by implementing Docker for smoother deployment.

6. CONTRIBUTION OF TEAM MEMBERS:

Roll No.	Name	Contribution
21Z235	Prathiksha R	Led the data preprocessing and handled data balancing.
21Z220	Harshitha S A	Focused on feature engineering and selection.
22Z436	Kamali A	Was in charge of model development and tuning.
21Z251	Santhoshi R	Managed model evaluation and metrics analysis.
22Z431	Aishwarya R	Oversaw the web application development and deployment.

7. ANNEXURE I: CODE

App.py:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import
accuracy_score, precision_score, recall_score, f1_score, classification_report
from flask import *
from sklearn.ensemble import VotingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import LinearSVC

app=Flask(__name__)

@app.route('/')
def index():
    return render_template("index.html")

@app.route('/about')
def about():
    return render_template("about.html")

@app.route('/load', methods=["GET", "POST"])
def load():
    global df, dataset
    if request.method == "POST":
        data = request.files['data']
        df = pd.read_csv(data)
        dataset = df.head(100)
        msg = 'Data Loaded Successfully'
        return render_template('load.html', msg=msg)
    return render_template('load.html')

@app.route('/preprocess', methods=['POST', 'GET'])
```

```

def preprocess():
    global x, y, x_train, x_test, y_train, y_test, hvvectorizer,df,data
    if request.method == "POST":
        size = int(request.form['split'])
        size = size / 100
        df = pd.read_csv(r'Employee dataset.csv')
        df.drop('employee_id',axis = 1,inplace = True)
        df.isna().sum()
        df['previous_year_rating'].fillna( df['previous_year_rating'].median(),
inplace=True)
        df['education'].fillna( df['education'].mode()[0], inplace=True)
        df.isna().sum()
        # using the labelEncoder to convert into same datatype
        le = LabelEncoder()
        for i in df.columns:
            if df[i].dtype == 'object':
                df[i] = le.fit_transform(df[i])
        print(df)
        x = df.drop('is_promoted',axis = 1)
        y = df['is_promoted']
        x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.3,random_state=70)
        df.head()
        # describes info about train and test set
        print("Number transactions X_train dataset: ", x_train.shape)
        print("Number transactions y_train dataset: ", y_train.shape)
        print("Number transactions X_test dataset: ", x_test.shape)
        print("Number transactions y_test dataset: ", y_test.shape)
        print(x_train,x_test)
        return render_template('preprocess.html', msg='Data Preprocessed and It Splits
Successfully')
        return render_template('preprocess.html')

@app.route('/model', methods=["POST","GET"])
def model():
    if request.method=="POST":

```



```

global model
s=int(request.form['algo'])
if s==0:
    return render_template('model.html',msg="Choose an algorithm")
elif s==1:
    gnb=GaussianNB()
    gnb.fit(x_train[:500],y_train[:500])
    y_pred=gnb.predict(x_test[:500])
    ac_gnb=accuracy_score(y_pred,y_test[:500])*100
    precision_gnb = precision_score(y_test[:500], y_pred,
average='weighted')*100
    recall_gnb = recall_score(y_test[:500], y_pred, average='weighted')*100
    f1_gnb = f1_score(y_test[:500], y_pred, average='weighted')*100
    msg="The accuracy : "+str(ac_gnb) + str('%')
    msg1="The precision : "+str(precision_gnb) + str('%')
    msg2="The recall : "+str(recall_gnb) + str('%')
    msg3="The f1_score : "+str(f1_gnb) + str('%')
    return render_template("model.html",msg=msg,msg1=msg1,msg2=msg2,msg3=msg3)
elif s==2:
    svs=LinearSVC()
    svs.fit(x_train[:500],y_train[:500])
    y_pred=svs.predict(x_test[:500])
    ac_svs=accuracy_score(y_pred,y_test[:500])*100
    precision_svs = precision_score(y_test[:500], y_pred,
average='weighted')*100
    recall_svs = recall_score(y_test[:500], y_pred, average='weighted')*100
    f1_svs = f1_score(y_test[:500], y_pred, average='weighted')*100
    msg="The accuracy : "+str(ac_svs) + str('%')
    msg1="The precision : "+str(precision_svs) + str('%')
    msg2="The recall : "+str(recall_svs) + str('%')
    msg3="The f1_score : "+str(f1_svs) + str('%')
    return render_template("model.html",msg=msg,msg1=msg1,msg2=msg2,msg3=msg3)
elif s==3:
    from xgboost import XGBClassifier
    xgb = XGBClassifier()
    xgb.fit(x_train, y_train)

```

```

y_pred = xgb.predict(x_test)
ac_xgb = accuracy_score(y_test, y_pred)
precision_xgb = precision_score(y_test, y_pred, average='weighted') * 100
recall_xgb = recall_score(y_test, y_pred, average='weighted') * 100
f1_xgb = f1_score(y_test, y_pred, average='weighted') * 100
msg = "The accuracy: " + str(ac_xgb) + '%'
msg1 = "The precision: " + str(precision_xgb) + '%'
msg2 = "The recall: " + str(recall_xgb) + '%'
msg3 = "The f1_score: " + str(f1_xgb) + '%'
return render_template("model.html", msg=msg, msg1=msg1, msg2=msg2,
msg3=msg3)
elif s==4:
    gnb = GaussianNB()
    svb = LinearSVC()
    vot = VotingClassifier(estimators=[
        ('gnb', gnb),
        ('svb', svb)
    ], voting='hard')
    vot.fit(x_train[:500], y_train[:500])
    y_pred = vot.predict(x_test[:500])
    ac_vot = accuracy_score(y_pred, y_test[:500]) * 100
    precision_vot = precision_score(y_test[:500], y_pred, average='weighted')
* 100
    recall_vot = recall_score(y_test[:500], y_pred, average='weighted') * 100
    f1_vot = f1_score(y_test[:500], y_pred, average='weighted') * 100
    msg="The accuracy :"+str(ac_vot) + str('%')
    msg1="The precision :"+str(precision_vot) + str('%')
    msg2="The recall : "+str(recall_vot) + str('%')
    msg3="The f1_score :"+str(f1_vot) + str('%')
    return render_template("model.html",msg=msg,msg1=msg1,msg2=msg2,msg3=msg3)
return render_template("model.html")

@app.route('/prediction' , methods=["POST","GET"])
def prediction():

```

```

if request.method=="POST":
    f1=float(request.form['department'])
    f2=float(request.form['region'])
    f5=float(request.form['education'])
    f6=float(request.form['gender'])
    f7=float(request.form['recruitment_channel'])
    f8=float(request.form['no_of_trainings'])
    f9=int(request.form['age'])
    f10=float(request.form['previous_year_rating'])
    f11=float(request.form['length_of_service'])
    f12=float(request.form['KPIs'])
    f13=float(request.form['awards_won?'])
    f14=float(request.form['avg_training_score'])

    lee=[f1,f2,f5,f6,f7,f8,f9,f10,f11,f12,f13,f14]
    print(lee)

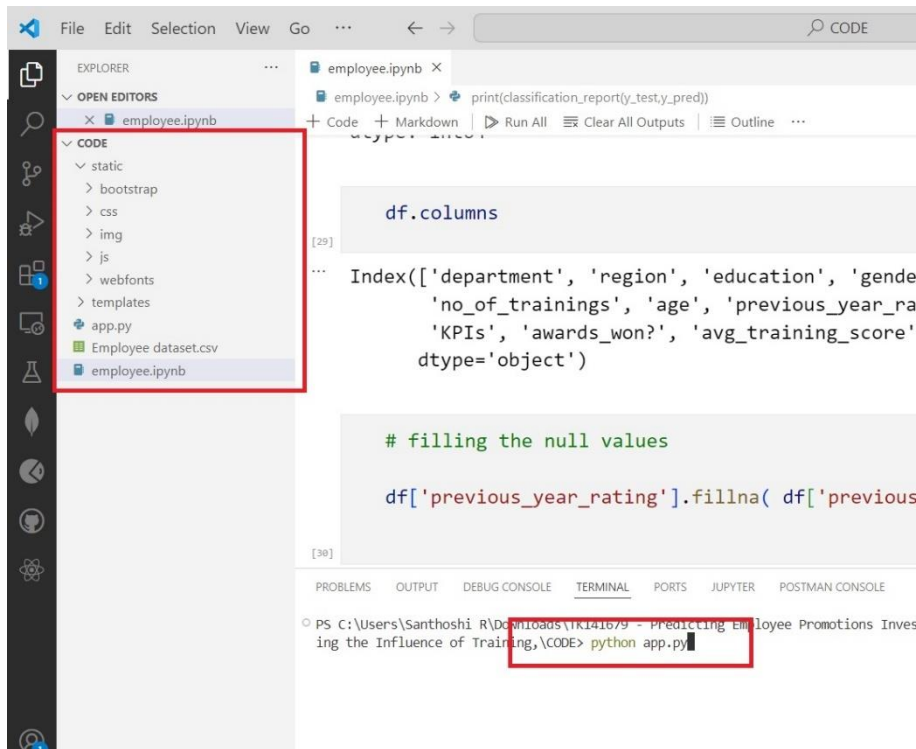
    from sklearn.naive_bayes import GaussianNB
    model=GaussianNB()
    model.fit(x_train,y_train)
    result=model.predict([lee])
    print(result)
    if result==0:
        msg="not promoted"
        return render_template('prediction.html', msg=msg)
    else:
        msg="promoted"
        return render_template('prediction.html', msg=msg)
    return render_template("prediction.html")

if __name__=="__main__":
    app.run(debug=True)

```

Google colab file that contain code for data preprocessing:

8. ANNEXURE II: SNAPSHOTS OF THE OUTPUT:



```
df.columns
```

```
[29] ... Index(['department', 'region', 'education', 'gender', 'no_of_trainings', 'age', 'previous_year_rating', 'KPIs', 'awards_won?', 'avg_training_score', 'dtype='object'])
```

```
# filling the null values
```

```
df['previous_year_rating'].fillna(df['previous
```

```
[30]
```

```
PS C:\Users\Santhoshi R\Downloads\TK141679 - Predicting Employee Promotions Investigating the Influence of Training,\CODE> python app.py
```

```
PS C:\Users\Santhoshi R\Downloads\TK141679 - Predicting Employee Promotions Investigating the Influence of Training,\CODE> python app.py
```

```
* Serving Flask app 'app'
```

```
* Debug mode: on
```

```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
```

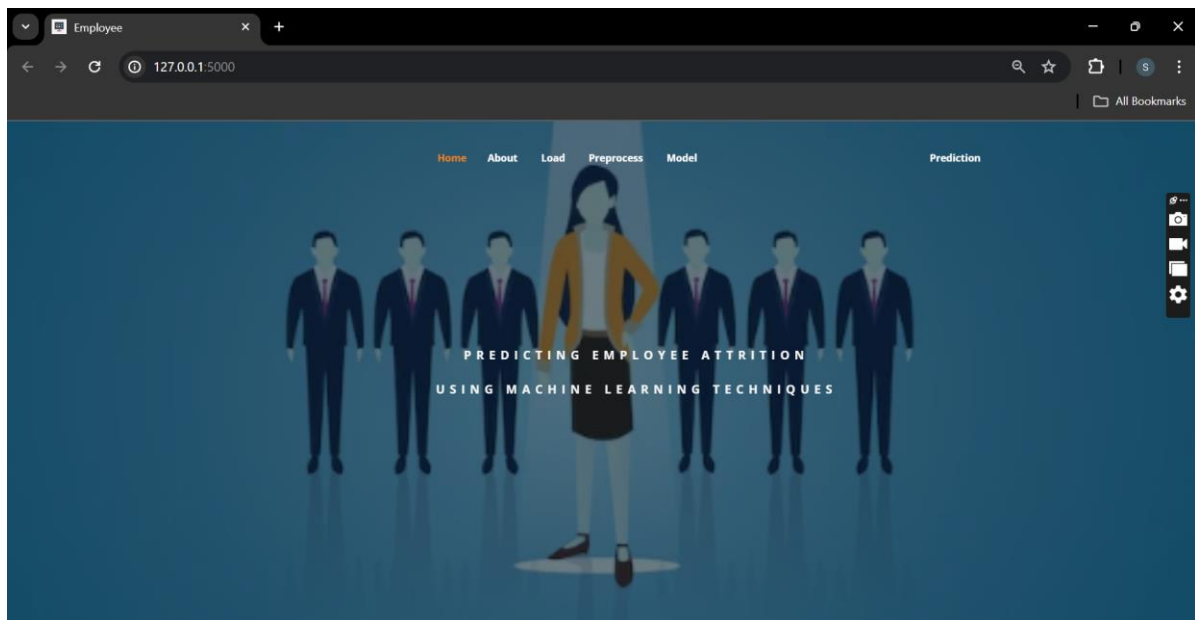
```
* Running on http://127.0.0.1:5000
```

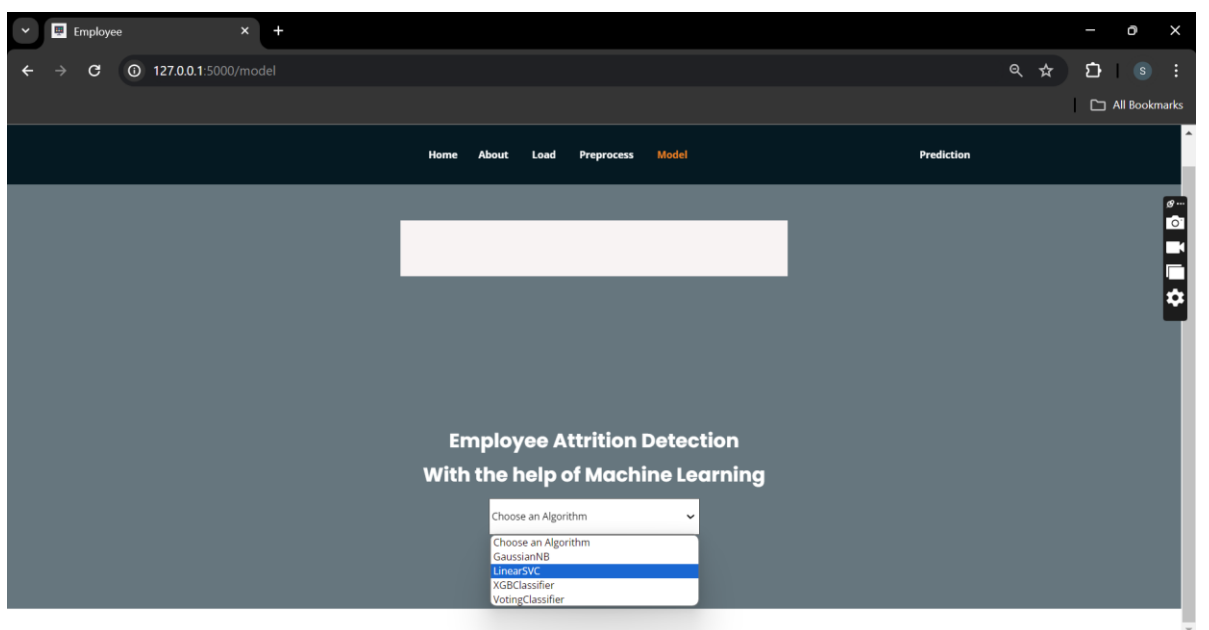
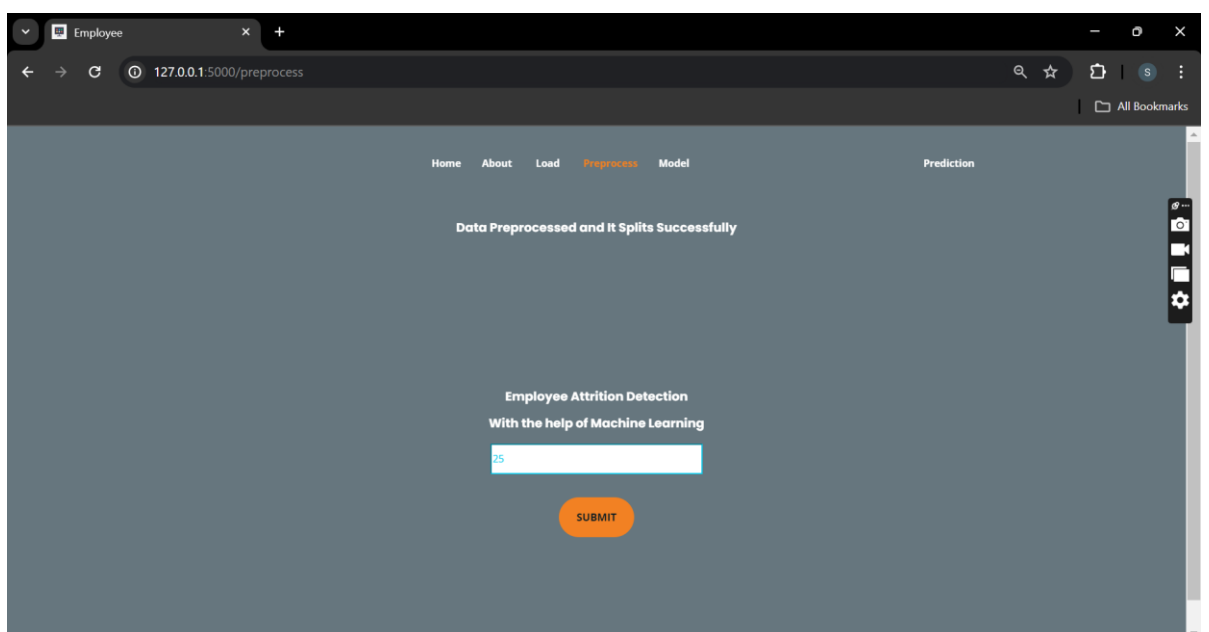
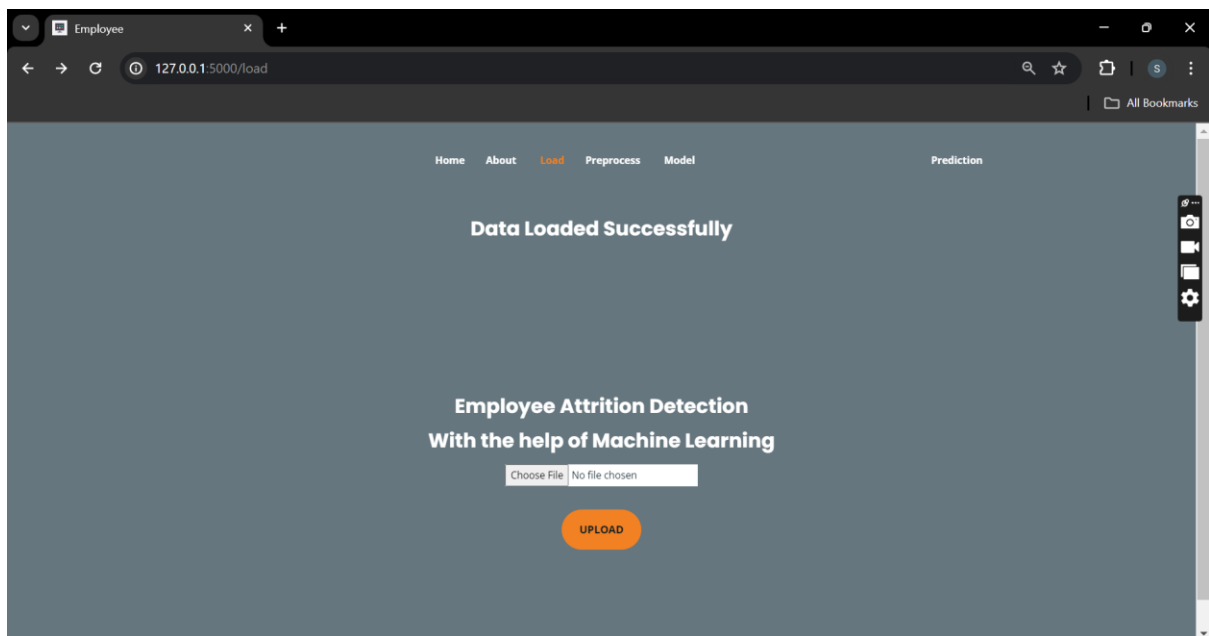
```
Press CTRL+C to quit
```

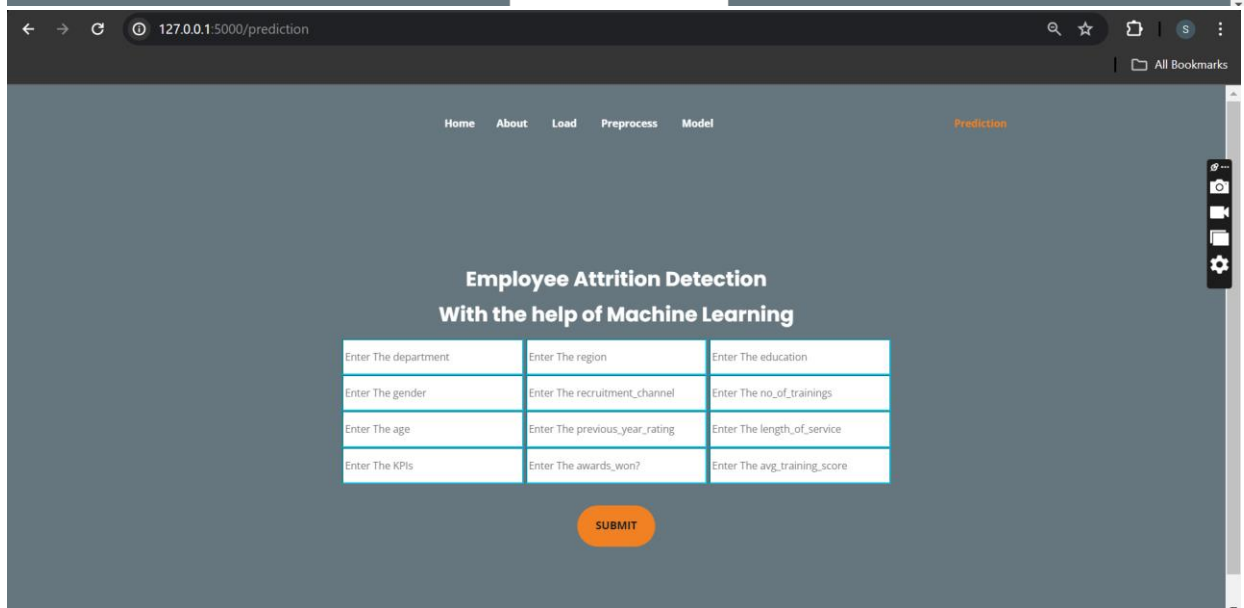
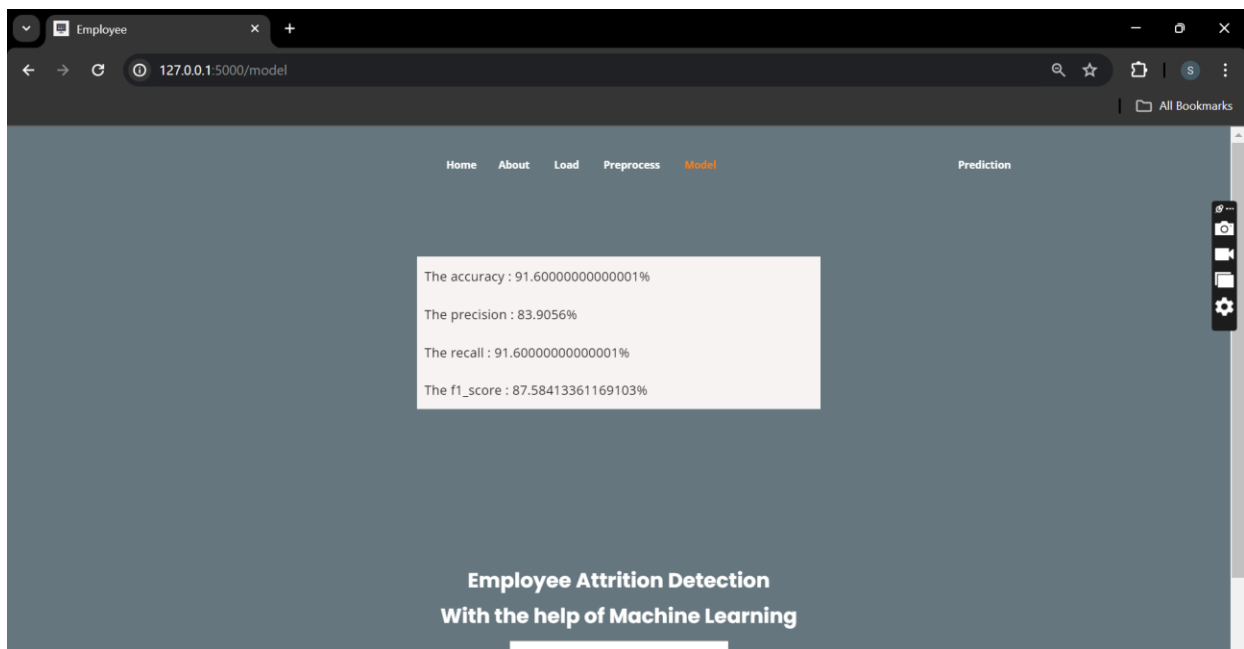
```
* Restarting with stat
```

```
* Debugger is active!
```

```
* Debugger PIN: 137-030-780
```







9. REFERENCES:

- [1] Vasantha Lakshmi, C. Patvardhan, An optical character recognition system for printed Telugu text . Pattern Anal Appl 7: pp. 190-204, 2004
- [2] M. Wenying and D. Zuchun, “A Digital Character Recognition Algorithm Based on the Template Weighted Match Degree”, International Journal of Smart Home, Vol. 7, No. 3, pp. 53-60, 2013.
- [3] Singh, R. and Kaur, M.. OCR for Telugu Script Using BackPropagation Based Classifier, International Journal of Information Technology and Knowledge Management, Vol. 2(2), pp.639-643, 2010
- [4] S. Arora, D. Bhattacharjee, M. Nasipuri, D. K. Basu, M. Kundu, Application of statistical features in handwritten Devanagari character recognition, International Journal of Recent Trends in Engineering, vol. 2, 2009, pp. 40–42.
- [5] Majid Ziaratban, Karim Faez, Farhad Faradji, Language based feature extraction using template matching in Farsi/Arabic handwritten numeral recognition, Ninth International Conference on Document Analysis and Recognition, vol. 1, 2007, pp. 297–301
- [6] <https://blog.perceptyx.com/employee-attribution-analytics>
- [7] <https://medium.com/@melissaonwuka/python-pandas-for-employee-attribution-c1aa86b5e26f>
- [8] <https://www.kaggle.com/code/adevvenugopal/employee-attribution-prediction-using-ml>
- [9] <https://www.analyticsvidhya.com/blog/2021/11/employee-attribution-prediction-a-comprehensive-guide/>
- [10] <https://medium.com/analytics-vidhya/predict-employee-attribution-a34e2c5a972d>