



Project Report

MACHINE LEARNING APPROACH FOR EMPLOYEE PERFORMANCE PREDICTION

This project uses machine learning to predict employee performance based on data like experience, education, and job role. It helps companies identify top performers, improve productivity, and make fair HR decisions.

Prepared by
BISHAL NASKAR
FUTURE INSTITUTE OF ENGINEERING AND
MANAGEMENT

ACKNOWLEDGEMENT

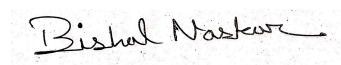
I would like to express my sincere gratitude to Revanth Sir, my mentor and guide, for his invaluable guidance, encouragement, and continuous support throughout the development of this project titled "Machine Learning Approach for Employee Performance Prediction."

His insightful feedback, patience, and motivation have been instrumental in shaping this work and helping me understand both the theoretical and practical aspects of integrating machine learning with real-world web applications.

I would also like to extend my appreciation to all the faculty members and peers who have provided their assistance, suggestions, and moral support during various stages of this project.

Finally, I am deeply thankful to my family and friends for their constant encouragement and understanding, which inspired me to complete this project successfully.

Sincerely,



BISHAL NASKAR
ECE, Future Institute of Engineering and Management

Mentor's Signature

TABLE OF CONTENTS

1. Abstract.....	3
2. Introduction.....	4
3. Objectives.....	5
4. System Architecture Overview.....	6
A. Machine Learning Backend	
B. Flask Web Application Frontend	
5. Technologies Used.....	7
6. Methodology and Implementation.....	8
Step 1 – Data Collection and Preparation	
Step 2 – Model Training	
Step 3 – Flask Application Development	
Step 4 – Folder Structure	
Step 5 – Template and Navigation Design	
7. Result and Observation.....	12
8. Advantages.....	13
9. Limitations.....	13
10. Future Enhancements.....	14
11. Conclusion.....	15
12. References.....	15

ABSTRACT

Employee performance is a vital factor influencing the success and growth of any organization. Traditional performance evaluations are often subjective and time-consuming. The use of machine learning (ML) provides a more objective, data-driven approach to predicting employee performance based on measurable parameters. This project focuses on developing and implementing ML models to predict employee performance using various features such as education, experience, department, satisfaction, and other relevant factors. The results assist organizations in making informed decisions regarding recruitment, promotions, and workforce management.

INTRODUCTION

In today's modern world, every organization depends on its employees for success. Employees play a very important role in deciding how well a company performs. To maintain good productivity, companies must identify which employees are performing well and which employees need improvement.

Traditionally, managers have used manual performance reviews, supervisor feedback, and yearly appraisals to judge employees. However, these methods are often **time-consuming, subjective, and sometimes inaccurate**.

With the help of **Machine Learning (ML)**, organizations can now make **data-driven decisions**. By analyzing past data and work-related factors such as **experience, satisfaction level, training hours, attendance, and promotion history**, a computer model can predict how well an employee is likely to perform in the future.

In this project, a **Machine Learning model** has been developed to predict employee performance levels based on historical data. To make this prediction system more user-friendly, a **Flask web application** has been created. Flask is a lightweight and easy-to-use web framework in Python that allows users to interact with the model through a web browser.

This system allows HR departments or managers to **input employee details** in a simple form and get **instant predictions** about their performance level. The main advantage of this system is that it is **automated, easy to use, and accurate**, helping organizations make better and faster HR decisions.

OBJECTIVES

The main goals and objectives of this project are:

- To create a Machine Learning model that can predict an employee's performance using given input data.
- To design a simple and interactive Flask web application that allows users to input data and view prediction results easily.
- To connect the ML model with the web application so that users can get instant results without any programming knowledge.
- To use real-life HR data and perform preprocessing to clean, encode, and prepare it for model training.
- To evaluate the model's accuracy and improve its performance using appropriate algorithms.
- To provide an easy and cost-effective solution that can be used by any organization to monitor employee productivity.
- To demonstrate full integration between data science and web development.

SYSTEM ARCHITECTURE OVERVIEW

The system developed in this project mainly consists of two parts:

A. Machine Learning Backend

This part is responsible for the **data processing, analysis, and prediction**. It uses Python libraries like `scikit-learn`, `pandas`, and `numpy`.

Steps involved in the backend:

- Data loading and cleaning
- Feature selection and encoding
- Model training and evaluation
- Saving the trained model using `joblib`

Once the model is trained, it is saved as a `.pkl` file (pickle file). This file is later used by Flask to make predictions.

B. Flask Web Application Frontend

This part provides a **user interface (UI)** for entering employee details. Flask connects the user input with the backend model.

When the user enters data and clicks the "Submit" button, Flask sends the data to the model, gets the prediction, and displays it on a result page.

The app has several pages like:

- **Home Page** – Introduction and project overview
- **About Page** – Information about the project and developers
- **Predict Page** – Input form for employee data
- **Submit Page** – Displays predicted results

TECHNOLOGIES USED

Category	Technology / Tool	Purpose
Programming Language	Python 3	For Machine Learning and Flask application
Framework	Flask	For building the web application
Libraries	scikit-learn, pandas, numpy, joblib	For model building, prediction, and data handling
Frontend	HTML, CSS, Jinja2	For designing the web pages
IDE	PyCharm or VS Code	For coding and testing
Environment	Python Virtual Environment	To manage dependencies
Deployment	Localhost	For local testing and demonstration

Each tool and technology was selected because it is open-source, simple to use, and widely supported in Python.

METHODOLOGY AND IMPLEMENTATION

The development of this project followed a **step-by-step approach**, starting from data collection to model deployment.

The steps are explained below:

Step 1: Data Collection

The first step was to collect a dataset that contains information about employees.

The dataset includes features like:

- Age
- Education Level
- Department
- Job Role
- Years of Experience
- Training Hours
- Monthly Working Hours
- Satisfaction Level
- Performance Score
- Absenteeism Days

These features are important for predicting how well an employee might perform.

Step 2: Data Preprocessing

Before training the model, the dataset is cleaned and prepared.

Tasks performed in this stage:

1. Removing missing or incomplete values.
2. Converting categorical data (like job role, department) into numeric form using encoding.
3. Normalizing numerical columns so that all features have similar scales.
4. Dividing data into **training (80%)** and **testing (20%)** sets.

This ensures that the model is trained effectively and tested fairly.

Step 3: Model Training

Once data is ready, a suitable ML algorithm is used to train the model.

Common algorithms include:

- **Random Forest Classifier**
- **Logistic Regression**
- **Decision Tree Classifier**

Example code snippet:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

After training, the model's accuracy is tested.

Once satisfactory performance is achieved, the model is saved using:

```
import joblib
joblib.dump(model, 'model.pkl')
```

Step 4: Flask Application Development

The Flask application connects the web interface with the trained model.

Below is the main Python code (app.py):

```
from flask import Flask, render_template, request
import joblib, numpy as np, pandas as pd, os

app = Flask(__name__)

MODEL_PATH = os.path.join(os.path.dirname(__file__), "model.pkl")
model = joblib.load(MODEL_PATH)

feature_cols = ['age', 'education_level', 'department', 'job_role',
                'years_experience', 'training_hours',
                'avg_monthly_hours',
                'satisfaction_level', 'last_performance_score',
                'absenteeism_days', 'promotion_last_5yrs']

@app.route("/")
```

```

def home():
    return render_template("home.html")

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

@app.route("/predict")
def predict_page():
    return render_template("predict.html")

@app.route("/submit", methods=["POST"])
def submit():
    try:
        data = {}
        for col in feature_cols:
            val = request.form.get(col)
            if col in ['age', 'years_experience', 'training_hours',
                      'avg_monthly_hours', 'satisfaction_level',
                      'last_performance_score', 'absenteeism_days',
                      'promotion_last_5yrs']:
                data[col] = float(val)
            else:
                data[col] = val


















        X = pd.DataFrame([data])
        prediction = model.predict(X)[0]
        return render_template("submit.html", prediction=prediction)

    except Exception as e:
        return render_template("submit.html", error=str(e))

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

```

Step 5: Folder Structure

- ✓  Employee_Performance_Prediction [Project_Demo] C:\
- ✓  Dataset
 -  employee_performance.csv
- ✓  Flask
 - ✓  static
 -  about-us-page.png
 -  employee-performance-improvement.jpeg
 -  plpso-feratures-data-business.jpg
 -  style.css
 - ✓  templates
 -  <> about.html
 -  <> base.html
 -  <> home.html
 -  <> predict.html
 -  <> submit.html
 -  app.py
 -  ? model.pkl

Step 6: Check Web Pages Description

1. **Home Page:**
Shows project title and navigation menu.
2. **About Page:**
Describes the purpose, motivation, and features of the system.
3. **Predict Page:**
Displays a form where users can enter employee details like age, experience, satisfaction level, etc.
4. **Submit Page:**
Displays the predicted result (e.g., *High*, *Medium*, or *Low performance*).

RESULTS AND OBSERVATIONS

After testing the system locally, the application performed well:

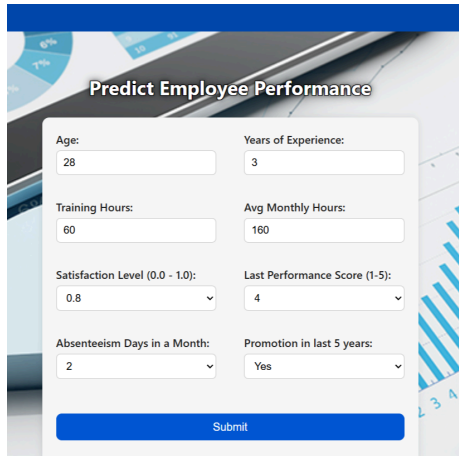
- It provided predictions instantly.
- The interface was user-friendly and simple.
- The model gave logical results according to input data.
- The app ran smoothly on any system with Python installed.

Example result:

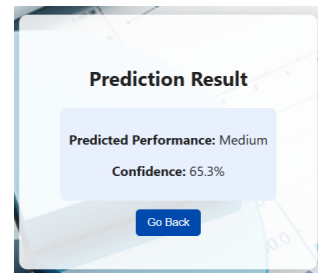
- Input: 28 years, 3 years experience, satisfaction = 0.8
- Output: **Medium Performance**

Result Images:

Input:

A screenshot of a web application titled "Predict Employee Performance". The form contains several input fields: "Age" (text input with value 28), "Years of Experience" (text input with value 3), "Training Hours" (text input with value 60), "Avg Monthly Hours" (text input with value 160), "Satisfaction Level (0.0 - 1.0):" (dropdown menu with value 0.8), "Last Performance Score (1-5):" (dropdown menu with value 4), "Absenteeism Days in a Month:" (dropdown menu with value 2), and "Promotion in last 5 years:" (dropdown menu with value Yes). A blue "Submit" button is at the bottom.

Output:



This proves that the integration of ML with Flask works effectively.

ADVANTAGES

- Predicts employee performance quickly and accurately.
- Saves time and effort for HR teams.
- Reduces bias and improves decision-making.
- Can be updated with new data easily.
- Works on any computer with basic requirements.
- Free to use (open-source tools).
- Easy to expand for future development.

LIMITATIONS

- Model accuracy depends on data quality.
- No database connection for saving user inputs.
- Currently works only on a local server.
- Web design is basic (can be improved).
- Categorical data encoding may not handle all unseen inputs.

FUTURE IMPROVEMENTS

To make this project more powerful and real-world ready, the following improvements can be made:

- Add database integration (MySQL or SQLite) to save prediction history.
- Deploy the app online using platforms like Heroku or AWS.
- Add user authentication (Admin and HR login).
- Improve frontend using Bootstrap or React.
- Add data visualization (graphs, charts for performance trend).
- Include more features such as teamwork, attendance, project success rate, etc.
- Integrate HR **analytics dashboard** for company-wide insights.

CONCLUSION

The **Employee Performance Prediction System** is a successful combination of **Machine Learning and Web Development**. It shows how data science can be applied to human resource management to make better and faster decisions.

The system predicts employee performance levels based on various measurable features and provides instant results through a user-friendly web interface. This project demonstrates how theoretical knowledge can be used to create a **real-world, working application**. It can be further improved to become a complete HR analytics platform for any organization.

REFERENCES

1. Python Official Website – <https://www.python.org>
2. Flask Documentation – <https://flask.palletsprojects.com>
3. scikit-learn Documentation – <https://scikit-learn.org>
4. Pandas Documentation – <https://pandas.pydata.org>
5. NumPy Documentation – <https://numpy.org>
6. HR Employee Performance Dataset (Open Source)