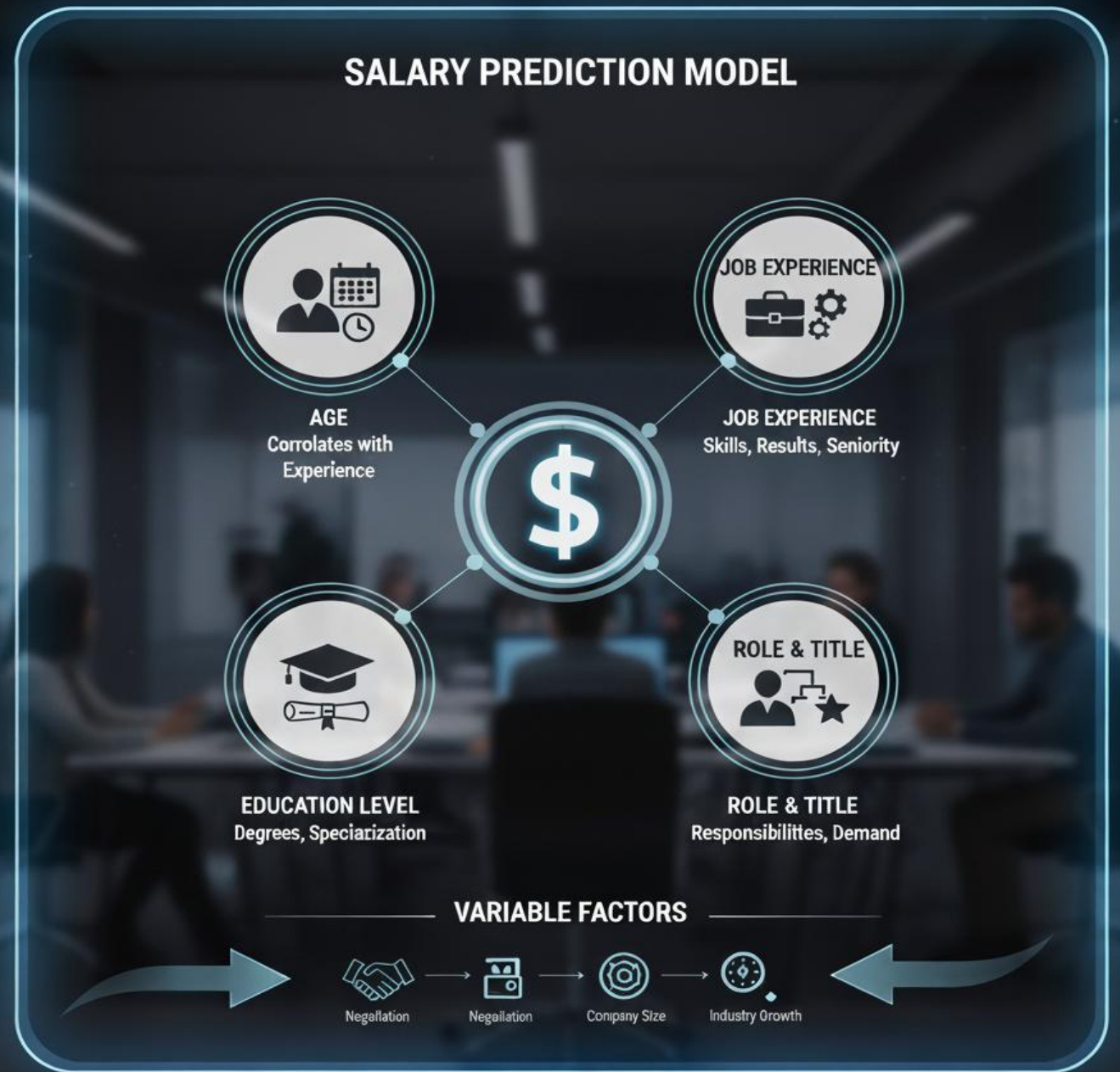


# SALARY PREDICTION USING MACHINE LEARNING

NAME: Srijita Saha

Guide's name: Ms Naina Devi

Department: Computer Science &  
Engineering



# AGENDA

**THIS PRESENTATION GIVES AN OVERVIEW OF THE *SALARY PREDICTION SYSTEM* PROJECT.**

**IT COVERS THE PROJECT'S PURPOSE, TOOLS AND TECHNOLOGIES USED, WORKING PROCESS WITH OUTPUT SCREENSHOTS, APPLICATIONS, CHALLENGES FACED WITH THEIR SOLUTIONS, FUTURE SCOPE, AND THE PROJECT LINK FOR REFERENCE**

- Introduction to the project
- Tools and Technologies used
- Project Working & output
- Applications
- Problems face and solutions
- Future scope
- Project Link



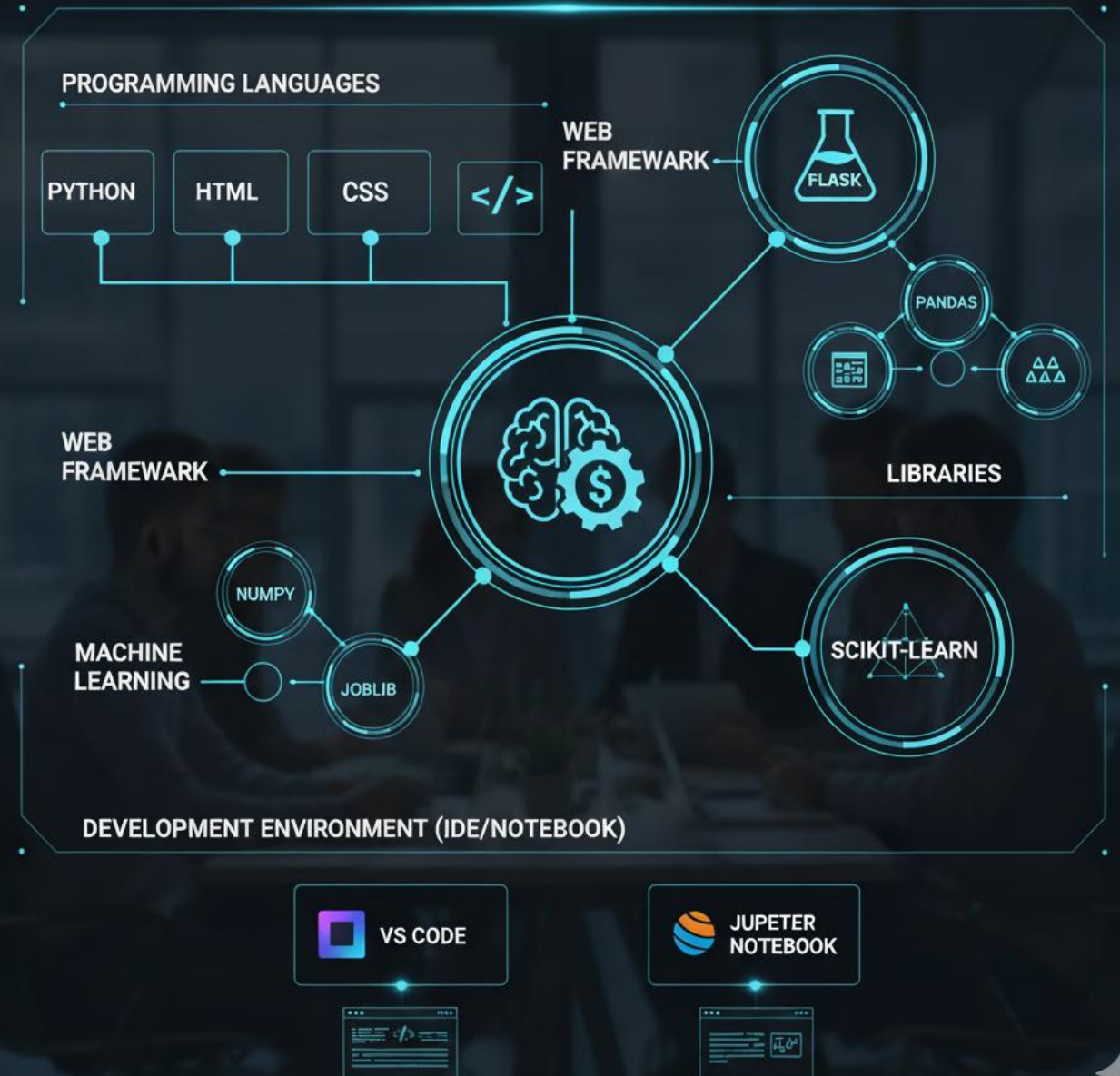
# INTRODUCTION

The *Salary Prediction System* is a machine learning-based web application that predicts an employee's salary based on inputs like job title, education level, years of experience, age, and gender. It helps job seekers and HR professionals estimate fair salaries using data-driven predictions

- Predicts employee salary based on features like job title, experience, education, gender, and age.
- Uses **Machine Learning (Linear Regression/Decision Tree/etc.)** for prediction
- Built with **Flask (backend)** and **HTML/CSS (frontend)** for web deployment

# TOOLS & TECHNOLOGIES USED

- **Languages:** Python, HTML, CSS
- **Frameworks:** Flask, Scikit-learn
- **Libraries:** Pandas, NumPy, Joblib
- **IDE:** VS Code / Jupyter Notebook





# PROJECT WORKING

- **Data Collection:** The dataset contains employee details like *Job Title*, *Age*, *Gender*, *Education Level*, and *Years of Experience* along with their *Salary*.
- **Data Preprocessing:** Missing values are handled, and categorical data (like Job Title, Education, Gender) is encoded into numeric form for model training.
- **Model Training:** A machine learning model (e.g., Linear Regression or Ensemble Model) is trained to learn the relationship between features and salary.
- **Model Saving:** The trained model is saved as a .pkl file using joblib for later use in the Flask app.
- **Web Integration (Flask App):** The Flask backend loads the model and predicts salary based on user inputs entered through the frontend form.
- **Output Display:** The predicted salary is displayed instantly on the web page

# Model Training & Evaluation

```
C:\Users\User\Desktop\salary_predictor>py train_model.py
```

```
First few rows of data:
```

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0
2	45.0	Male	PhD	Senior Manager	15.0	150000.0
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0
4	52.0	Male	Master's	Director	20.0	200000.0


```
Categorical columns: ['Gender', 'Education Level', 'Job Title']
```

```
Numeric columns: ['Age', 'Years of Experience']
```

```
Model Performance:
```

```
RMSE: 17551.14
```

```
R2 Score: 0.87
```

```
Model saved successfully as models/salary_model.pkl 
```

```
C:\Users\User\Desktop\salary_predictor>
```

- The dataset includes key employee attributes such as **Age, Gender, Education Level, Job Title, and Years of Experience.**
- The data was preprocessed and used to train ensemble models like **Random Forest, Gradient Boosting, and XGBoost.**
- The final model achieved:
  - RMSE:** 17,551.14
  - R<sup>2</sup> Score:** 0.87
- The trained model was saved as **salary\_model.pkl** for deployment in the web application.



## FLASK API EXECUTION

```
C:\Users\User\Desktop\salary_predictor>py api.py
* Serving Flask app 'api'
* 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: 854-480-231
127.0.0.1 - - [11/Nov/2025 21:59:56] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [11/Nov/2025 21:59:56] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [11/Nov/2025 22:03:35] "POST /predict HTTP/1.1" 200 -
```

- The Flask server was successfully started at **http://127.0.0.1:5000** in debug mode.
- The API exposed two routes:
  - **/** → Home route to test the connection.
  - **/predict** → Accepts user input and returns the predicted salary.
- A **POST request** was made to **/predict**, and the server responded with **status 200 (Success)** — confirming the API is working properly

# Output Screenshots

## Salary Prediction Form

Predict Salary

## Salary Prediction Form

Predict Salary

## Salary Prediction Form

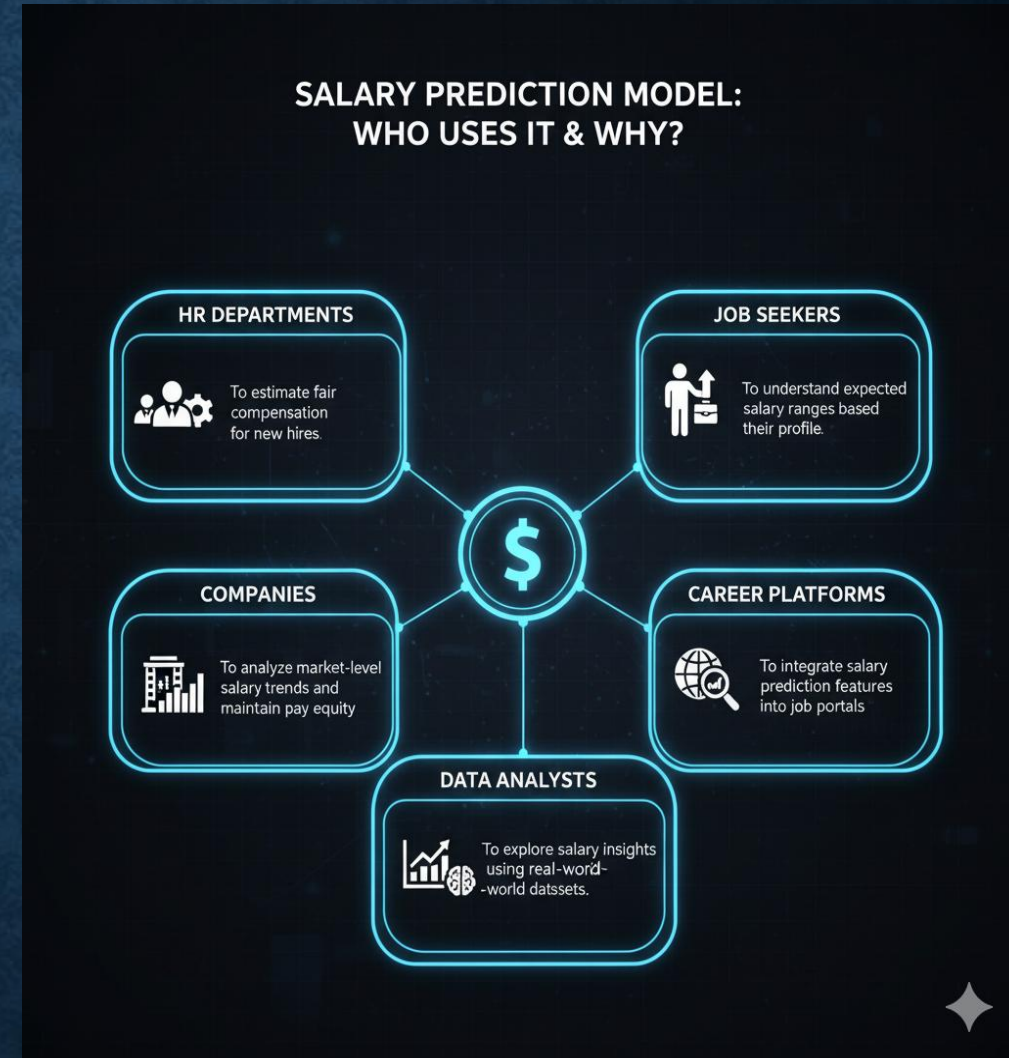
Predict Salary

**Predicted Salary: ₹70,675.00**



# APPLICATIONS

- **HR Departments:** To estimate fair compensation for new hires.
- **Job Seekers:** To understand expected salary ranges based on their profile.
- **Companies:** To analyze market-level salary trends and maintain pay equity.
- **Career Platforms:** To integrate salary prediction features into job portals.
- **Data Analysts:** To explore salary insights using real-world datasets.



# PROBLEMS FACED & SOLUTIONS

## PROBLEMS

- Missing data while processing
- Model not accurate initially
- Integration error (columns mismatch)

## SOLUTIONS

- Used imputation techniques
- Tuned model and selected best features
- Fixed frontend field names to match model columns




# FUTURE SCOPE & PROJECT LINK

Improve accuracy using deep learning models

Build interactive dashboard (e.g., Streamlit)

Integrate live data input

 **Project Link:** [https://github.com/srijitaaa2005/SALARY\\_PREDICTOR](https://github.com/srijitaaa2005/SALARY_PREDICTOR)

THANK YOU