**PROJECT EXPLANATION:**

**Name:Shriya Sahu  
Gitam University,Vizag**



**Project Objective:**The goal is to build a machine learning system for **Company X** that automatically predicts the **expected salary (CTC)** of new job applicants based on various features like experience, education, specialization, location, etc. This aims to:

* Reduce human bias in compensation decisions
* Standardize pay across similar profiles
* Minimize manual judgment during salary negotiation

The objective of this exercise is to build a model, using historical data, that will determine the salary to be offered to an employee, minimizing manual judgment in the selection process. The approach aims to be robust and eliminate any discrimination in salary among employees with similar profiles.



**ML Workflow Summary:**  
1. **Data Cleaning**

* Removed or imputed missing values.
* Converted categorical "Yes/No" values into numeric (0/1) where required.
* Ensured correct data types for numerical and categorical fields.

**2.Preprocessing**

* Categorical columns were encoded using OneHotEncoding.
* Features and target variable were split into X and y.

3.**Model Building**

* Trained four regression models:
  + Linear Regression
  + Decision Tree Regressor
  + Random Forest Regressor
  + XGBoost Regressor
* Evaluated each model using **R² Score**.
* Saved the best-performing model using joblib.

4.**Flask Web Interface**

* Created a web page using HTML and Flask to collect new candidate data.
* Submitted data is passed to the ML model for prediction.
* Predicted Expected CTC is displayed on the page.

**Project Folder Structure that I implemented In VSCode:**

| **Folder/File** | **Description** |
| --- | --- |
| 1\_data\_preparation/ | Folder containing original and cleaned datasets |
| expected\_ctc.csv | Original dataset |
| cleaned\_expected\_ctc.csv | Cleaned dataset used for model training |
| 2\_model\_training/ | Folder for model building and training code |
| model\_training.ipynb | Jupyter Notebook with preprocessing, training, and model evaluation |
| input\_template.csv | Template row for proper input format used in the Flask app |
| 3\_flask\_app/ | Flask web app folder |
| app.py | Flask backend application file |
| model.pkl | Trained machine learning model saved using joblib |
| templates/ | HTML template folder |
| index.html | Frontend form for input and predicted CTC display |

### **Model Comparison (Based on R² Score):**

During the model training phase, several machine learning algorithms were used to predict the Expected\_CTC based on various features like experience, education, industry, and other candidate attributes. The models evaluated included **Linear Regression**, **Decision Tree Regressor**, **Random Forest Regressor**, and **XGBoost Regressor**.

After training all models and comparing their performance using the **R² score** (a measure of how well the model explains the variation in the target variable), the **Random Forest Regressor** showed the highest accuracy. This means it was the best at predicting the expected salary compared to the actual values in the test set.

### **Here's why the Random Forest performed best:**

* It can effectively handle both **numerical and categorical data**.
* It works well even when the relationship between input features and the target variable is **non-linear**.
* Since it’s an **ensemble model**, it combines multiple decision trees to reduce overfitting and improve generalization.
* It doesn't require much tuning to achieve good results on structured or tabular data like this.

Other models like **Linear Regression** did not perform as well because they assume a linear relationship between the input features and the target, which doesn't hold in most real-world HR and salary datasets. The **Decision Tree Regressor** did better than Linear Regression but was more prone to overfitting. **XGBoost**, while powerful, did not significantly outperform Random Forest in this case and requires more fine-tuning of parameters.

Thus, the **Random Forest Regressor** was selected as the final model and saved using joblib for use in the Flask web application to predict the salary for new applicants.

**SNIPPET OF MY CODE IN VS CODE:**

