A close up of a logo

AI-generated content may be incorrect.

**Salary Predictor Web App**

Machine Learning CBIO313

Name: Ahmed Hamdi

ID: 221001876

Under supervision:

Dr.Mohamed Elsayeh

Machine Learning Project Report

# Abstract

This project focuses on developing a machine learning-based web application to predict whether a salary is high or not based on job-related features such as company rating, founding year, job seniority, company state, and average salary. The model is trained on a cleaned dataset with relevant preprocessing steps, including handling missing values, encoding categorical variables, and performing feature selection. Multiple classification models, including Logistic Regression, Random Forest, and Decision Tree, were trained and evaluated. The best-performing model was then integrated into a Flask web application that provides users with salary prediction based on their input. The final system serves as an interactive and intelligent tool that can support job seekers and HR professionals in evaluating job offers based on historical data and trends.

# Introduction

Machine learning has significantly transformed how predictive insights are extracted from data across various industries. In the domain of employment and human resources, one critical challenge is estimating whether a particular job offer provides a competitive salary. Accurate salary prediction can help individuals make informed career decisions and guide organizations in setting fair compensation benchmarks.

This project aims to build an end-to-end salary prediction system using machine learning techniques and deploy it via a user-friendly web application. The dataset used in this project includes various features such as company ratings, the year the company was founded, job seniority level, geographical state, and average salary. These features are selected for their relevance to salary estimation, as they reflect both organizational and job-specific factors.

We begin by performing exploratory data analysis (EDA) to understand the distribution of key variables and detect any data quality issues. Next, we preprocess the dataset by encoding categorical features and selecting the most informative features. Several classification models are trained, evaluated, and compared based on their accuracy. The most effective model is then serialized and deployed using a Flask backend.

This integrated machine learning system demonstrates the practical application of data science in career and recruitment analytics. It showcases how structured data and predictive models can be combined to provide real-time, intelligent feedback for salary evaluation, improving transparency and decision-making in the job market.

# Methodology

**1.Loading and Preparing the Dataset**  
The dataset is loaded into a Pandas DataFrame from a CSV file. It contains features like company rating, year founded, job seniority level, state, and average salary.  
  
**2. Label Encoding**  
Categorical variables like seniority and state are encoded into integers using label encoding. This ensures consistent feature dimensions for model training and prediction.  
  
**3. Feature and Target Separation**  
The features used for training include Rating, Founded, seniority, state, and avg\_salary. The target variable 'high\_salary' is a binary label indicating whether a salary is considered high (1) or not (0).  
 **4. Model Training**  
A Random Forest Classifier is used due to its robustness and ability to handle both numerical and categorical features. The model is trained on the selected features to predict the high\_salary label.  
  
**5. Saving the Model and Feature Set**  
The trained model is saved using joblib for future inference. Additionally, the list of feature names is saved to ensure correct structure during deployment.  
 **6. Web Application Deployment**  
A Flask app (app.py) is developed to deploy the model via a user-friendly web interface. Users input job details, which are processed and passed to the model for prediction.  
  
**7. Prediction Logic**  
User inputs are structured into a dictionary matching the training feature layout. Missing features are added with zero values, and the dictionary is converted to a DataFrame for prediction.  
  
**8. Error Handling**  
The Flask app includes error-handling logic to ensure that any missing inputs or mismatches in feature names are caught and displayed to the user.  
  
Model Performance Comparison:  
- Logistic Regression: Accuracy = 82%  
-SVC : Accuracy = 84%  
- Random Forest Classifier: Accuracy = 89%  
  
The Random Forest Classifier outperformed the other models and was selected as the final model for deployment due to its higher accuracy and general robustness.  
  
**9. Deployment**

The trained model was serialized using Joblib and deployed using a Flask web application. Users can interact with the model through a user-friendly HTML interface.

# Results and Discussion

The trained model achieved reasonable accuracy and was able to differentiate high salary jobs based on the input features. The Flask app functions effectively, providing immediate predictions based on user inputs. The biggest challenge faced was maintaining consistency between the feature names during training and prediction, which was resolved by using label encoding and saving the exact feature list.

# Visualizations and Interpretation

## 1. Average Salary Distribution

This histogram shows the distribution of average salaries across job listings. We observe that most average salaries cluster between 90 and 140k, with fewer extreme values on either side. This helps in determining salary segmentation thresholds, like using the median to define 'high salary'. A graph of a number of jobs

AI-generated content may be incorrect.

## 2. Job Listings per State

This bar chart represents the number of job listings available in each U.S. state. California has the highest number of listings, followed by Virginia and Massachusetts. This gives insight into the regional concentration of tech opportunities.

A graph of salary distribution

AI-generated content may be incorrect.

## 3. Average Salary by State

The bar chart illustrates how average salaries vary across different states. Delaware and North Carolina lead with the highest average salaries. This analysis reveals geographical trends in salary expectations.

A graph of salary

AI-generated content may be incorrect.

**4. Seniority vs. Average Salary**

The boxplot compares the distribution of average salaries between junior and senior roles. Senior positions generally exhibit a higher average salary and wider salary spread. However, there is some overlap suggesting that other factors also influence salary.

## A graph of a graph showing a difference between a senior and a senior AI-generated content may be incorrect.

**5. Correlation Heatmap**

The heatmap illustrates the correlation between numerical features in the dataset. Strong correlations are found between min\_salary, max\_salary, and avg\_salary. Rating and Founded year show weak correlation with salary, which suggests their limited predictive power.

A chart of a heatmap

AI-generated content may be incorrect.

**Conclusion**

In this project, we successfully developed a machine learning-powered salary prediction system that classifies whether a salary offer is high or not based on multiple job-related features. Through systematic data preprocessing, exploratory analysis, and the evaluation of different classification algorithms—namely Logistic Regression, Random Forest, and Decision Tree—we were able to determine the most effective model based on prediction accuracy. The Random Forest classifier demonstrated the highest accuracy and was selected for deployment.

The integration of the trained model into a Flask web application further enhanced the project’s usability, allowing users to interactively enter job details and receive real-time predictions. This web-based tool can be beneficial for job seekers, HR departments, and analysts aiming to assess salary competitiveness based on historical data.

Overall, this project highlights the practical value of machine learning in addressing real-world problems and emphasizes the importance of end-to-end development—from data handling and model training to deployment and user interaction.