PROJECT OVERVIEW

Candidate Acceptance Probability Prediction

PROJECT OVERVIEW

- **Objective**: Build a machine learning-based web application that predicts the probability of a candidate being accepted based on several recruitment metrics.

- **Tools & Technologies**: Python, Flask, Scikit-learn, Pandas, Google Colab, ngrok.

- **Purpose**: Streamline the recruitment process by providing recruiters with insights into a candidate's acceptance probability in percentage form.

PROBLEM STATEMENT

- **Context**: Recruitment is often a lengthy process that includes subjective opinions, which may lead to inconsistent hiring decisions. Leveraging data-driven approaches can help improve the efficiency and objectivity of hiring.
- **Challenge**: How to predict the likelihood of a candidate being accepted, based on various inputs such as age, interview scores, skill levels, etc.

HOW IT WORKS?

- **Training:** Machine Learning will learn from candidate’s data history and then create a prediction model.

- **Predict**: Machine Learning predict the likelihood of a candidate being accepted, based on various inputs such as age, interview scores, skill levels, etc.

- **Deployment:** Machine Learning module can be deploy into internal system for a better user experience

ESTIMATED RESULT

- Enhanced Decision-Making
- Efficiency
- Scalable and Consistent Results

TECHNICAL EXPLANATION

DATA COLLECTION & FEATURE ENGINEERING

- https://www.kaggle.com/datasets/rabieelkharoua/predicting-hiring-decisions-in-recruitment-data/data
- Inputs Used:Age, Gender, Education Level, Years of Experience, Previous Companies, Distance from Company, Interview Score, Skill Score, Personality Score, Recruitment Strategy.

- Feature Engineering:Categorized continuous variables like Age, Interview Score, Skill Score, and Personality Score into bins to create meaningful categories.

DATA PREPROCESSING

- **Data Cleaning**: Ensured that the input dataframe only contained the necessary columns before making predictions.

- **Scaling**: Applied scaling to columns like Age, Experience Years, Distance from Company, and scores (Interview, Skill, Personality).

- **One-Hot Encoding**: Added one-hot encoded columns for categorized features.

MODEL BUILDING

- **Final Model**: Random Forest Classifier (accuracy 90%, & recall 82%).
- **Hyperparameter Tuning**: Conducted tuning using GridSearchCV to optimize model performance.
- **Performance Metrics**: Evaluated model using accuracy and recall scores to minimalize potential talent lost.

DEPLOYMENT

- **Application**: Built a simple web application using Flask.

- **Functionality**: Recruiters input candidate data through a web form, and the app returns the probability of the candidate being accepted (displayed as a percentage).

- **ngrok for Deployment**: Used ngrok to expose the locally running Flask application to the internet.

- **Model Integration**: Integrated the machine learning model in the Flask app and deployed it using Google Colab

KEY LEARNINGS

- **Model Deployment**: Experience in deploying machine learning models to the web.
- **Web Development**: Integration of HTML forms with Python Flask for seamless user interaction.
- **End-to-End ML Workflow**: Understanding the complete cycle from data preprocessing to model building, hyperparameter tuning, and deployment.

FUTURE IMPROVEMENTS

- Enhancing the web app UI/UX.
- Adding more robust model explainability.
- Expanding the model to integrate feedback loops to improve predictions over time.