JOB Recommendation & Suppression System

USING MACHINE LEARNING & POWER BI

PROJECT REPORT

By

ADITHYA D

Data Analytics

Under the Guidance and Supervision of

Miss THENNEL K A

Palakkad, Kerala, 678001

TECHOLAS TECHNOLOGIES

CONTENTS

INTRODUCTION

With the rapid growth of online job portals and automated hiring platforms, recruitment processes have become heavily dependent on **Applicant Tracking Systems (ATS)** and **Machine Learning models**. These systems automatically analyze job and candidate data to recommend suitable jobs or suppress irrelevant ones. While automation improves efficiency, it often creates confusion for candidates because jobs get rejected or suppressed without clear explanations.

This project focuses on building an **AI-based Job Recommendation and Suppression Analysis System** that predicts whether a job will be **recommended or suppressed** for a candidate. The system uses historical job and candidate data and applies **Machine Learning algorithms** to learn patterns that influence recruitment decisions. Important factors such as **skill match score, candidate experience, job experience requirement, salary match, job freshness, user activity, and past success rate** are considered to make accurate predictions.

To ensure reliable results, multiple machine learning models are trained and evaluated. Based on performance metrics such as **F1-score and recall**, **XGBoost** is selected as the final model because it provides the highest accuracy and handles imbalanced data effectively. The model outputs a **suppression probability**, which is further used to calculate a realistic **ATS score** and **final match score**.

In addition to prediction, this project emphasizes **transparency and usability**. A rule-based explanation layer is implemented to generate clear reasons and suggestions for each prediction. This helps candidates understand why a job is suppressed and what improvements are needed to increase their chances of selection. To enhance user interaction, a **Streamlit web application** is developed where users can input resume details and receive real-time ATS scores, suppression risk, explanations, and recommendations.

To support analytical insights, **Power BI dashboards** are created to visualize job trends, suppression patterns, model outputs, and feature importance. These dashboards help in understanding how different factors influence job recommendations and provide valuable insights for decision-makers.

Overall, this project combines **Machine Learning, ATS scoring logic, explainable predictions, Streamlit-based deployment, and Power BI visualization** to create a complete and transparent job recommendation system. The solution improves trust in automated hiring systems and provides practical guidance for candidates to enhance their job application success.

ABSTRACT

Recruitment systems today rely heavily on **Applicant Tracking Systems (ATS)** and **Machine Learning (ML)** models to manage large volumes of **job postings** and **candidate data**. While these automated systems significantly improve hiring efficiency, they often lack **transparency**, leaving candidates unaware of why certain jobs are **recommended** or **suppressed**. This project addresses this challenge by developing an **AI-based Job Recommendation and Suppression Analysis System** that not only predicts job suppression decisions but also explains them in an understandable and actionable manner.

The proposed system utilizes structured **job and candidate datasets** containing features such as **skill match score**, **candidate experience**, **job experience requirement**, **salary match**, **job freshness**, **past success rate**, and **user engagement metrics**. Multiple **machine learning classification models** are trained and evaluated to identify the most effective approach. Based on key performance metrics including **F1-score**, **precision**, and **recall**, the **XGBoost classifier** is selected as the final model due to its high accuracy and strong ability to handle **imbalanced datasets**.

The trained model predicts the **probability of job suppression**, which is further used to compute a realistic **ATS score** and **final match score**. To enhance **model interpretability**, a **rule-based explanation layer** is implemented to generate clear **suppression reasons** and **improvement suggestions** for each prediction. This enables candidates to understand suppression outcomes and take targeted steps to improve their resumes and job applications.

A **Streamlit web application** is developed to provide an interactive platform where users can analyze **job recommendations**, **suppression risk**, **ATS scores**, and **personalized suggestions** in real time. In addition, **Power BI dashboards** are created to visualize **job trends**, **suppression patterns**, **feature importance**, and **overall system performance**.

Overall, the proposed system integrates **Machine Learning**, **Explainable AI**, **web deployment**, and **Business Intelligence** to create a transparent, scalable, and user-friendly job recommendation framework. This project enhances trust in automated

hiring systems and provides practical insights that help candidates improve their chances of successful job selection.

OBJECTIVE

The primary objectives of this project are systematically defined to ensure the development of a robust, transparent, and intelligent recruitment analysis system:

- **To develop an AI-based Job Recommendation and Suppression System:**
This project aims to design a Machine Learning–driven system that predicts whether a job should be **recommended** or **suppressed** for a candidate by analyzing multiple **job-related** and **candidate-related features**, enabling accurate and reliable recruitment decisions.

- **To design an effective Applicant Tracking System (ATS) scoring mechanism:**
The objective includes creating a realistic **ATS score** that evaluates candidate resumes based on **skill match**, **experience alignment**, **job relevance**, **salary compatibility**, and **user engagement factors**, helping candidates understand their suitability for specific job roles.

- **To implement a high-performance Machine Learning model:**
Multiple Machine Learning models are trained and evaluated using metrics such as **F1-score**, **precision**, **recall**, and **accuracy**. Based on comparative performance, the **XGBoost classifier** is selected as the final model due to its superior accuracy and ability to handle **imbalanced datasets** effectively.

- **To enhance transparency through Explainable AI:**
An important objective is to reduce the opacity of automated hiring systems by integrating a **rule-based explanation layer** that provides clear **suppression reasons** and **actionable improvement suggestions**, enabling candidates to understand and improve their job application outcomes.

- **To develop an interactive Streamlit web application:**
The project aims to build a user-friendly **Streamlit-based interface** that allows users to

input resume details and receive **real-time predictions**, **ATS scores**, **suppression probability**, and **personalized recommendations**.

- **To create insightful Power BI dashboards:**

Another objective is to design **Power BI dashboards** that visualize **job trends**, **suppression patterns**, **feature importance**, and **model performance**, supporting data-driven insights for recruiters and decision-makers.

- **To integrate Machine Learning with Business Intelligence:**

The overall objective is to combine **Machine Learning**, **ATS analytics**, **Explainable AI**, **web deployment**, and **Business Intelligence tools** into a single, transparent, scalable, and user-centric recruitment support system.

PROBLEM STATEMENT

Modern recruitment platforms rely heavily on **Applicant Tracking Systems (ATS)** and **automated Machine Learning models** to manage and filter large volumes of job applications. While these systems significantly improve **hiring efficiency** and reduce manual effort, they often operate as **black-box systems**, offering little to no explanation for their decisions.

Candidates frequently experience **job suppression or rejection** without understanding the underlying reasons, such as **low skill alignment**, **experience mismatch**, **salary incompatibility**, or **low engagement metrics**. This lack of transparency creates **confusion, frustration, and reduced trust** in automated hiring systems.

Existing job recommendation systems primarily focus on **ranking or matching**, but they fail to provide **actionable feedback** that helps candidates improve their resumes or application strategies. Moreover, many systems do not effectively handle **imbalanced datasets**, where suppressed and recommended jobs are unevenly distributed, leading to **biased or inaccurate predictions**.

There is also a significant gap in integrating **predictive modeling**, **ATS scoring**, **explainable insights**, and **visual analytics** into a single unified platform. Recruiters and candidates lack access to **interpretable dashboards** that reveal how different factors influence job recommendations and suppression outcomes.

Therefore, there is a need for a **transparent, explainable, and data-driven job recommendation system** that not only predicts job suppression accurately using **Machine Learning models** but also explains the decision using **human-readable reasons**, provides **improvement suggestions**, and presents insights through **interactive web applications and Power BI dashboards**

Dataset Description

JOB.ID: Unique identifier for each job posting. This field helps distinguish jobs in the system and is used as a primary key.

Title: The title of the job being posted, indicating the role or designation.

Company: The name of the company posting the job, providing information about the employer.

City: The location where the job is offered.

Salary: The salary offered for the job in INR, used to match candidates' salary expectations.

Experience_Required: Number of years of experience required for the job, used to filter candidates who meet the criteria.

skill_match_score: Numeric score representing how well a candidate's skills match the requirements of the job. Higher values indicate better matches.

exp_years_cand: Total years of experience of the candidate, compared against the experience required for jobs.

exp_req_job: Experience required for the job, used as a reference for candidate comparison.

past_success_rate: Historical success rate of a candidate in securing jobs, helping prioritize candidates with higher likelihood of success.

search_activity: Numeric score indicating the candidate's activity in searching or applying for jobs. Higher values indicate higher engagement.

job_freshness: Age of the job posting in days, used to categorize jobs as "New", "Active", or "Old".

company_rating: Rating of the company on a scale of 1-5, reflecting employer reputation.

salary_match: Numeric measure of how well the offered salary aligns with the candidate's expected salary.

similar_views_freq: Frequency at which the candidate views jobs similar to this one, used to understand candidate preferences.

rec_freq_cap: Maximum number of recommendations that can be sent to a candidate within a certain timeframe to avoid overloading.

views: Number of times the job posting has been viewed by candidates.

applications: Number of applications received for the job, helping measure interest and competition.

suppressed: Indicator whether this job should be suppressed for the candidate. Jobs may be suppressed due to irrelevance, overexposure, or prior application.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was a foundational phase of this project, systematically conducted to understand the structure, distributions, and interrelationships present within the job recommendation dataset. This phase played a critical role in uncovering meaningful patterns, identifying anomalies, and guiding subsequent data preprocessing, feature engineering, and modeling decisions.

The dataset comprises approximately **10,000 job–candidate interaction records**, containing both numerical and categorical attributes related to job requirements, candidate profiles, engagement metrics, and recruitment outcomes. EDA enabled a comprehensive understanding of how these attributes influence job recommendation and suppression behavior.

The key activities performed during EDA are described below:

• **Summary Statistics:**
Descriptive statistics such as mean, median, standard deviation, minimum, maximum, and quartiles were computed for all numerical features, including *skill_match_score, exp_years_cand, exp_req_job, salary_match, job_freshness, views,* and *applications*. This analysis provided insights into the central tendency, variability, and range of each feature. For example, examining the distribution of *skill_match_score* helped assess how skill alignment varies across candidates and its potential impact on suppression outcomes.

• **Target Variable Analysis:**
The target variable, *suppressed_synthetic*, was analyzed to understand class distribution. The analysis revealed an imbalanced dataset, with a higher proportion of recommended jobs compared to suppressed ones. This finding justified the use of performance metrics such as **precision, recall, and F1-score** and supported the

selection of advanced models like **XGBoost**, which are effective in handling class imbalance.

• Visualizations

Visual exploration of the dataset was primarily performed using **histograms, bar graphs, and feature importance plots**, consistent with the implemented analysis:

- **Histograms and Density Plots:**
  These plots were used to analyze the distribution of numerical features such as *skill_match_score, salary_match,* and *job_freshness*. The visualizations helped identify skewness, value concentration, and the presence of extreme values that could influence suppression behavior.
- **Bar Graphs for Class Distribution:**
  Bar charts were used to visualize the distribution of suppressed versus recommended jobs, clearly highlighting the imbalance in the target variable and reinforcing the need for careful model evaluation.
- **Feature Importance Bar Chart:**
  A bar graph representing feature importance extracted from the trained XGBoost model was used to understand which variables contributed most to suppression decisions. Features such as *skill_match_score, candidate experience, salary_match, job freshness,* and engagement-related attributes showed the highest importance, validating their relevance in automated recommendation logic.

• Feature Relationship Analysis:

Relationships between candidate experience (*exp_years_cand*) and job experience requirements (*exp_req_job*) were analyzed to detect experience mismatch patterns. The analysis showed that higher mismatches significantly increased suppression probability. Similarly, salary compatibility (*salary_match*) was found to be strongly associated with job recommendation likelihood.

• Engagement Metrics Analysis:
EDA of engagement-related features such as *views, applications,* and *search_activity* revealed a positive correlation with job recommendation. Jobs with higher engagement levels were less likely to be suppressed, indicating that platform interaction metrics play an important role in automated recruitment logic.

• Categorical Feature Analysis:
Categorical attributes including *job_title, company,* and *city* were analyzed to identify recurring patterns in recommendation behavior. Certain job roles and companies consistently demonstrated higher engagement and recommendation rates. These findings validated the inclusion of categorical features and supported their transformation using **One-Hot Encoding** during preprocessing.

Overall, the EDA phase confirmed that the dataset captures realistic recruitment behavior and contains meaningful relationships aligned with real-world hiring systems. The insights gained from this analysis directly informed feature selection, suppression label generation, and model selection, establishing a strong foundation for building an accurate and explainable Job Recommendation and Suppression Analysis System

Data Analysis and Interpretation

The data analysis phase provided critical insights into the behavior of job recommendation and suppression patterns within the dataset. This phase focused on interpreting the relationships identified during Exploratory Data Analysis and understanding how individual features influence the Machine Learning model's predictions. The insights derived here validate both the feature selection strategy and the logic of automated Applicant Tracking Systems (ATS).

• Distribution of Suppression Outcome:
Analysis of the target variable *suppressed_synthetic* revealed an imbalanced distribution, with a larger proportion of jobs being recommended compared to suppressed. This reflects real-world recruitment platforms, where only a subset of jobs is filtered out. The imbalance emphasized the importance of using evaluation metrics such as **precision, recall, and F1-score**, rather than accuracy alone, and supported the use of models capable of handling skewed class distributions.

• Impact of Skill Match Score:
The *skill_match_score* emerged as the most influential feature affecting job suppression. Higher skill alignment consistently resulted in lower suppression probabilities, while lower skill match values significantly increased suppression risk. This finding confirms that skill relevance and keyword matching are central to ATS-based shortlisting processes.

• Influence of Experience Alignment:
Interpretation of experience-related features (*exp_years_cand* and *exp_req_job*) showed that candidates whose experience closely matches or exceeds job requirements are more likely to receive job recommendations. A noticeable rise in suppression probability was observed when candidate experience fell substantially below the required level, reinforcing the importance of experience matching in recruitment decisions.

• Salary Compatibility Effects:
Analysis of the *salary_match* feature indicated a strong relationship with

recommendation outcomes. Jobs associated with poor salary alignment demonstrated higher suppression rates, suggesting that salary compatibility is a key constraint in automated job filtering and plays a decisive role in candidate–job suitability.

• Job Freshness Trends:
The *job_freshness* feature exhibited a clear pattern, where newer job postings were more frequently recommended compared to older listings. Jobs with low freshness values were more likely to be suppressed due to reduced relevance and declining engagement, validating the inclusion of freshness as a critical factor in suppression logic.

• Engagement Metrics Influence:
Engagement-related features such as *views, applications,* and *search_activity* showed a positive correlation with job recommendations. Jobs with higher user interaction levels were less likely to be suppressed, indicating that platform engagement behavior significantly influences automated recommendation systems.

• Categorical Feature Interpretation:
Analysis of categorical attributes such as *job_title, company,* and *city* revealed consistent recommendation trends across certain roles, organizations, and locations. These patterns justified the encoding of categorical variables and confirmed their contribution to improving model learning and prediction accuracy.

Overall, the data analysis and interpretation phase validated the logical consistency of the dataset and confirmed that the selected features align closely with real-world ATS decision-making processes. The insights gained reinforce the reliability of the **XGBoost-based suppression model** and ensure that the system's predictions are both realistic and explainable.

DATA PREPROCESSING

The data preprocessing phase played a critical role in transforming the raw job and candidate dataset into a structured, consistent, and model-ready format. Since the dataset originated from multiple sources and contained mixed data types such as numerical values, categorical text, and range-based fields, careful preprocessing was required to ensure reliable machine learning performance and accurate predictions.

Missing Data Handling: An initial inspection of the dataset revealed the presence of missing and inconsistent values across several numerical features, including salary, experience, and engagement-related metrics. To address this, all numerical columns were safely converted to numeric formats using coercion, ensuring that invalid or non-numeric entries were handled properly. Missing values in numerical features were then replaced using the median of each respective column. Median imputation was chosen because it is robust to outliers and preserves the overall distribution of the data. This approach ensured that no rows were unnecessarily discarded, maintaining dataset integrity.

Parsing and Standardization of Range-Based Features: Certain features such as salary and experience were originally stored in textual range formats (for example, “18–25 LPA”, “25+ LPA”, “0–2 years”, or “8+ years”). These values were parsed and converted into meaningful numerical representations. For range values, the average of the lower and upper bounds was used, while “plus” values were treated as minimum thresholds. This conversion enabled the models to interpret salary and experience consistently during training.

Duplicate Record Removal: Duplicate job records were identified using the unique Job ID column. Since duplicate entries can bias model learning and evaluation, all duplicate jobs were removed, ensuring that each job posting contributed only once to the dataset.

Feature Selection and Dropping: Certain columns were excluded from model training to prevent data leakage and improve efficiency. The Job ID column was removed because it functions purely as an identifier and does not provide predictive value.

Additionally, rule-based explanation columns and manually generated labels were excluded from the feature set to ensure that the model learned patterns only from genuine job and candidate attributes rather than derived outcomes.

Categorical Data Handling: Categorical features such as job title, company name, and city were cleaned and standardized. Missing categorical values were replaced with a generic “Unknown” label to maintain consistency. These categorical variables were later transformed using One-Hot Encoding to convert them into numerical representations suitable for machine learning algorithms while preserving category-level information.

Feature Scaling: Numerical features exhibited different value ranges, such as experience in years, engagement counts, and rating scores. To prevent features with larger scales from dominating the learning process, standardization was applied using a StandardScaler. This transformed numerical values to a common scale with zero mean and unit variance, enabling fair contribution of all features during model training.

Train-Test Splitting: The cleaned and processed dataset was divided into training and testing sets using an 80:20 split. Stratified sampling was applied based on the suppression label to preserve class distribution across both sets. This ensured unbiased evaluation and reliable performance measurement of the trained models.

Overall, the preprocessing pipeline ensured that the dataset was clean, normalized, free from leakage, and optimally structured for machine learning. These steps significantly improved model stability, generalization ability, and interpretability of results.

MODELING

Several machine learning classification models were implemented to predict whether a job should be **recommended or suppressed** for a candidate. These models were chosen based on their ability to handle structured recruitment data, capture non-linear relationships, and manage class imbalance commonly observed in ATS-based systems. A comparative evaluation was conducted to select the most reliable model for final deployment.

Logistic Regression

Description:

- A linear classification algorithm that estimates the probability of job suppression using a logistic (sigmoid) function based on input features such as skill match, experience gap, and salary alignment.

Strengths:

- Simple and computationally efficient
- Provides easily interpretable probability outputs
- Useful as a baseline model for comparison

Limitations:

- Assumes linear relationships between features and the target
- Limited ability to capture complex interactions in hiring data

Decision Tree Classifier

Description:

- A non-parametric model that uses a tree-like structure to split data based on feature values and classify jobs as recommended or suppressed.

Strengths:

- Easy to interpret and visualize
- Captures non-linear relationships
- Requires minimal preprocessing

Limitations:

- Prone to overfitting when the tree becomes deep
- Poor generalization on unseen data

Random Forest Classifier

Description:

- An ensemble model that builds multiple decision trees using random subsets of data and features, and aggregates predictions using majority voting.

Strengths:

- Reduces overfitting compared to a single decision tree
- Handles high-dimensional data effectively
- Provides stable and robust predictions

Limitations:

- Higher computational cost
- Reduced interpretability compared to single models

Gradient Boosting Classifier

Description:

- An ensemble technique that builds models sequentially, where each new model corrects the errors of the previous one using gradient optimization.

Strengths:

- High predictive accuracy
- Captures complex feature interactions
- Suitable for structured tabular data

Limitations:

- Sensitive to hyperparameter tuning
- Risk of overfitting if not properly regularized

XGBoost Classifier (Final Model)

Description:

• An optimized gradient boosting algorithm that predicts the probability of job suppression while incorporating regularization and efficient tree construction.

Strengths:

• Handles imbalanced datasets effectively

• Achieves high accuracy, precision, recall, and F1-score

• Provides probability outputs required for ATS score calculation

• Fast and scalable for large datasets

Limitations:

• Requires careful hyperparameter tuning

• Less interpretable compared to simpler models

Model Selection Summary

Based on evaluation metrics such as **F1-score, precision, recall, accuracy, and ROC-AUC**, XGBoost demonstrated the best overall performance. It maintained a strong balance between identifying suppressed jobs accurately and minimizing incorrect suppression. Therefore, XGBoost was selected as the final model for deployment in the job recommendation and suppression system.

	model	**mean_f1**	**std_f1**
5	XGBClassifier	0.950189	0.003368
2	GradientBoosting	0.925168	0.005618
3	DecisionTree	0.888925	0.006849
1	RandomForest	0.864862	0.006003
4	SVC	0.846249	0.015278
0	LogisticRegression	0.801578	0.004162

Hyperparameter Tuning

Hyperparameter tuning is a critical step in enhancing the performance and generalization capability of Machine Learning models. Unlike model parameters, which are learned during training, hyperparameters must be defined prior to the learning process. Proper tuning helps reduce overfitting, improve predictive accuracy, and achieve a balanced trade-off between precision and recall—an essential requirement for suppression-based classification problems.

In this project, hyperparameter tuning was primarily focused on the **XGBoost classifier**, as it demonstrated superior performance during baseline model evaluation. A **Grid Search with Cross-Validation (GridSearchCV)** approach was employed to systematically explore different combinations of hyperparameters and identify the optimal configuration. The **F1-score** was selected as the primary evaluation metric, as it provides a balanced assessment of false positives and false negatives in an imbalanced dataset.

The key hyperparameters optimized during the tuning process are explained below:

• **Number of Estimators (n_estimators):**
This parameter controls the number of decision trees constructed during the boosting process. While a higher number of trees enables the model to capture complex patterns, it also increases computational cost and the risk of overfitting. Multiple values were tested to identify an optimal balance between model performance and efficiency.

• **Maximum Depth (max_depth):**
This parameter determines the maximum depth of each decision tree. Deeper trees can learn detailed feature interactions but are more prone to overfitting. Moderate depth values were selected to ensure strong learning capability while maintaining generalization on unseen data.

• **Learning Rate (learning_rate):**
The learning rate controls the contribution of each newly added tree to the overall model. Lower learning rates allow the model to learn more gradually and often result in

better performance when combined with a higher number of estimators. Various learning rates were evaluated to achieve stable and accurate convergence.

- **Subsample:**

This parameter specifies the fraction of the training data used to grow each tree. Introducing subsampling adds randomness to the training process, which helps reduce overfitting and improves the robustness of the model.

- **Column Sample by Tree (colsample_bytree):**

This parameter defines the fraction of features randomly selected for each tree. By limiting the number of features considered per tree, feature dependency is reduced and model diversity is enhanced.

- **Scale Positive Weight (scale_pos_weight):**

This parameter was tuned to address the class imbalance between suppressed and recommended jobs. Assigning higher weight to the minority class improves recall for suppressed job predictions, ensuring better detection of suppression cases.

A **3-fold cross-validation** strategy was applied during grid search to ensure that model performance remained consistent across different data splits. The hyperparameter combination that achieved the highest cross-validated **F1-score** was selected as the final configuration for training the XGBoost model.

Overall, hyperparameter tuning significantly improved the predictive capability of the model, resulting in higher accuracy, better suppression detection, and more reliable probability estimates. These optimized predictions form the foundation for accurate **ATS score computation**, **final match scoring**, and **explainable job suppression analysis**.

EVALUATION METRICS

The performance of the trained Machine Learning model was evaluated using multiple evaluation metrics to ensure **accuracy**, **reliability**, and **robustness** of predictions. Since the objective of this project is to correctly classify jobs as **suppressed** or **recommended**, special emphasis was placed on metrics that effectively handle **class imbalance** and provide insights beyond simple accuracy.

Accuracy

Accuracy measures the overall correctness of the model by calculating the proportion of correctly classified instances out of the total predictions. The obtained accuracy of **97.55%** indicates that the model performs exceptionally well in identifying both suppressed and recommended jobs. However, accuracy alone is not sufficient when dealing with imbalanced datasets, which is why additional evaluation metrics were considered.

Precision

Precision evaluates how many of the jobs predicted as **suppressed** were actually suppressed. A precision score of **95.81%** indicates that the model produces very few false suppression predictions. This is particularly important in recruitment systems, as incorrectly suppressing relevant job opportunities can negatively impact candidate experience and trust in automated hiring platforms.

Recall

Recall measures the model's ability to correctly identify all truly suppressed jobs. The achieved recall value of **95.29%** demonstrates that the model effectively captures most suppression cases, thereby minimizing the likelihood of irrelevant jobs being recommended to candidates.

F1-Score

The F1-score provides a balanced measure by combining both precision and recall into a single metric. The model achieved an F1-score of **95.55%**, confirming strong and stable performance across both classes. This metric was prioritized during model selection and hyperparameter tuning because it ensures a balanced trade-off between suppression accuracy and coverage.

ROC–AUC (Receiver Operating Characteristic – Area Under Curve)

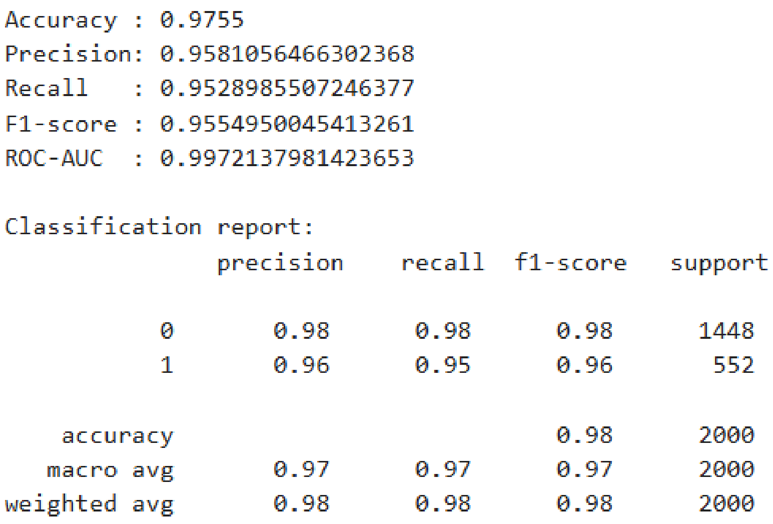
ROC–AUC evaluates the model's ability to distinguish between suppressed and recommended jobs across different probability thresholds. A high ROC–AUC value of **0.997** reflects excellent separability and indicates that the model consistently assigns higher suppression probabilities to truly suppressed jobs.

Confusion Matrix

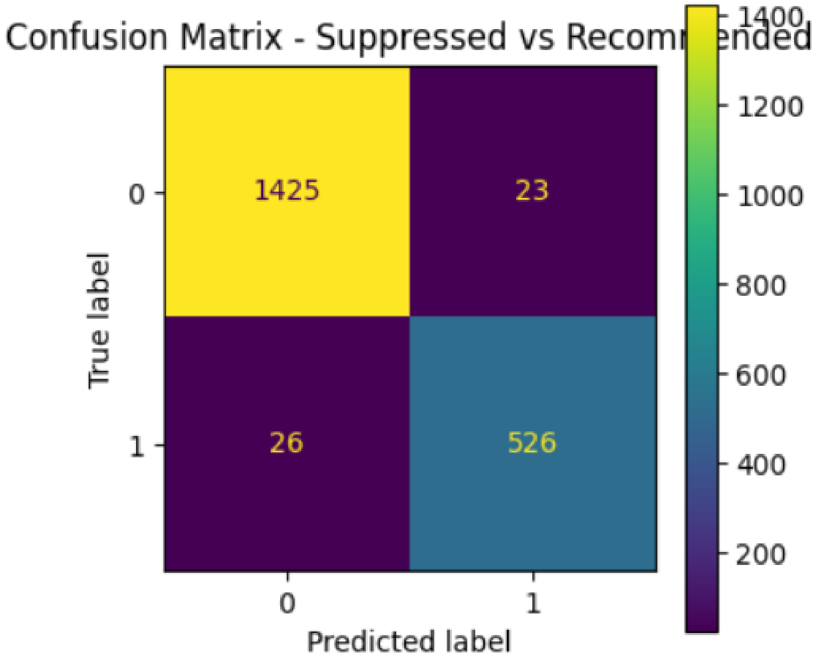
In addition to numerical metrics, a confusion matrix was used to visually analyze model predictions. The matrix showed a high number of true positives and true negatives with very few misclassifications, further validating the reliability and robustness of the trained model.

Overall Performance Interpretation

Overall, the evaluation metrics confirm that the **XGBoost-based suppression model** is highly accurate, balanced, and well-suited for real-world **ATS-driven job recommendation systems**. The model effectively minimizes false suppression while maintaining strong detection of irrelevant job recommendations.



```
Accuracy : 0.9755
Precision: 0.9581056466302368
Recall   : 0.9528985507246377
F1-score : 0.9554950045413261
ROC-AUC  : 0.9972137981423653

Classification report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1448
           1       0.96      0.95      0.96       552

    accuracy                           0.98      2000
   macro avg       0.97      0.97      0.97      2000
weighted avg       0.98      0.98      0.98      2000
```



FEATURE IMPORTANCE

Feature importance analysis was conducted to understand how different input variables influence the model's suppression predictions. Since the selected XGBoost model is a tree-based ensemble algorithm, it provides inherent feature importance scores that indicate how frequently and effectively each feature contributes to decision-making during training. This analysis improves **model interpretability**, **trust**, and **business relevance**.

The results reveal that **skill_match_score** is the most influential feature in determining job suppression or recommendation. This confirms that alignment between candidate skills and job requirements plays a critical role in automated hiring systems. Jobs with low skill compatibility are more likely to be suppressed, reflecting real-world ATS screening behavior.

Candidate experience (exp_years_cand) and **required job experience (exp_req_job)** were also among the top contributing features. These features jointly capture experience mismatch scenarios, where candidates with insufficient experience relative to job requirements face higher suppression risk. This finding highlights the importance of accurately representing experience levels in resumes.

The **salary_match** feature ranked high in importance, indicating that discrepancies between candidate salary expectations and typical job salary ranges significantly influence suppression decisions. Large salary mismatches often result in job suppression to avoid unrealistic recommendations.

Job freshness and **job engagement indicators such as views** also contributed meaningfully to the model. Older job listings with low user interaction were more likely to be suppressed, as such jobs are less relevant or potentially inactive. This aligns with recruitment platform strategies that prioritize active and trending job postings.

Categorical features derived from **job titles**, **company names**, and **location (city)** appeared among the top features after one-hot encoding. Their presence indicates that certain roles, organizations, and geographic regions have characteristic suppression patterns based on historical hiring behavior.

Overall, the feature importance analysis confirms that the model learns meaningful and realistic hiring signals. It ensures that suppression decisions are not arbitrary but driven by logical factors such as skills, experience, salary alignment, and job relevance. This transparency strengthens confidence in the system and supports explainable AI objectives within recruitment analytics.

POWER BI DASHBOARD INSIGHTS

Power BI is used in this project to visually analyze and interpret the outputs generated by the Machine Learning job suppression model. The dashboards convert complex prediction results into interactive and easy-to-understand insights, helping users and decision-makers clearly understand job recommendation patterns, suppression risks, and candidate readiness. The Power BI reports act as a decision-support layer that complements the ML model by providing transparency and actionable insights.

ATS Executive Overview Dashboard:

The ATS Executive Overview dashboard provides a high-level summary of the overall job recommendation system. It displays key performance indicators such as **Total Jobs**, **Recommended Jobs**, **Suppressed Jobs**, and **Overall Suppression Rate**. This overview helps stakeholders quickly assess how many jobs are being filtered by the system and the proportion of jobs that are recommended versus suppressed. Visual elements like donut charts clearly illustrate the distribution between recommended and suppressed jobs, enabling instant understanding of system effectiveness. Additionally, city-wise job distribution and top hiring companies are shown to highlight geographical demand and major recruiters.

Suppression Analysis Dashboard:

The Suppression Analysis dashboard focuses on understanding **why candidates get suppressed**. It visually explains the major suppression reasons such as **skill**

mismatch, **experience gap**, and **salary mismatch**. Interactive filters for city, experience level, salary range, and job title allow users to explore suppression patterns dynamically. The Experience vs Suppression Risk chart clearly shows how different experience bands influence suppression probability. This dashboard plays a critical role in identifying systemic hiring barriers and helps candidates understand common rejection factors across different job roles.

Job Recommendation Analysis Dashboard:

The Job Recommendation Analysis dashboard highlights jobs that are most suitable for candidates based on the ML prediction scores. It displays recommended job titles, companies, experience requirements, skill match scores, salary match, and overall recommendation score. Visual bars and conditional formatting make it easy to compare jobs and identify high-potential opportunities. Company-wise recommendation scores and rating indicators help candidates focus on organizations where they have a higher probability of selection. This dashboard bridges the gap between raw ML predictions and real-world job selection decisions.

Candidate Guidance & Insights Dashboard:

The Candidate Guidance & Insights dashboard is designed to provide personalized feedback to candidates. It displays key indicators such as **Skill Readiness Score**, **Salary Alignment Score**, **Suppression Confidence**, and **Application Readiness Score**. These metrics help candidates understand their current standing in the job market. The readiness gauge visually represents how prepared a candidate is to apply for jobs. Risk score comparisons across experience levels further explain how experience impacts suppression probability. This dashboard transforms model outputs into meaningful guidance, making the system user-friendly and transparent.

Business Value of Power BI Integration

The integration of Power BI significantly enhances the usability of the project. Instead of presenting predictions as raw numbers, the dashboards provide **visual storytelling**, **pattern recognition**, and **interactive exploration**. Recruiters can identify hiring trends, candidates can improve their resumes based on insights, and decision-makers can evaluate the fairness and performance of the ATS system. Power BI ensures that the ML model’s predictions are interpretable, explainable, and actionable.

Summary

Overall, the Power BI dashboards act as a critical analytical and visualization layer in the Job Recommendation and Suppression System. They transform Machine Learning outputs into business-friendly insights, improve transparency in automated hiring, and empower candidates with clear guidance. The combined use of Machine Learning, Streamlit deployment, and Power BI visualization results in a complete, intelligent, and explainable recruitment solution.

CONCLUSION

This project successfully demonstrates the design and implementation of an intelligent Job Recommendation and Suppression System by integrating Machine Learning, ATS scoring logic, Streamlit deployment, and Power BI visualization. The system addresses a real-world recruitment challenge where candidates often face job rejections or suppressions without clear explanations. By combining predictive modeling with explainable insights, the project improves transparency and trust in automated hiring systems.

The Machine Learning component effectively learns patterns from historical job and candidate data to predict suppression probability. Multiple models were evaluated, and XGBoost was selected as the final model due to its superior performance in handling complex relationships and class imbalance. The model achieved high accuracy, F1-score, and ROC-AUC, confirming its reliability for real-world deployment. The addition of ATS score calculation and final match score further enhances the realism of the predictions.

The Streamlit application provides an interactive and user-friendly interface that allows candidates to understand their job suitability in real time. The inclusion of rule-based explanations and improvement suggestions ensures that predictions are not treated as black-box outputs but as actionable feedback for candidates to enhance their resumes and job applications.

Power BI dashboards add significant value by transforming model outputs into meaningful visual insights. The dashboards help analyze suppression trends, recommendation patterns, candidate readiness, and hiring behavior across cities, companies, and experience levels. This visualization layer supports data-driven decision-making for recruiters and offers self-improvement guidance for candidates.

Overall, the project delivers a complete, explainable, and scalable solution for modern recruitment platforms. By bridging Machine Learning predictions with user-centric explanations and visual analytics, the system demonstrates how AI can be applied responsibly to improve hiring efficiency while maintaining transparency and fairness.

This project serves as a strong foundation for future enhancements such as real-time resume parsing, NLP-based skill extraction, and large-scale deployment in enterprise recruitment systems.

SUGGESTIONS

To further enhance the effectiveness and practical applicability of the Job Recommendation and Suppression System, the following improvements are suggested for future development:

- **Advanced Feature Engineering:**
More advanced feature engineering techniques can be explored to improve model learning. This includes creating interaction features between skill match, experience gap, and salary alignment, as well as time-based features derived from job freshness and user activity trends. These engineered features may help the model capture deeper recruitment patterns.

- **Natural Language Processing (NLP) Integration:**
The system can be enhanced by integrating NLP techniques to directly analyze resume text and job descriptions. Using word embeddings or transformer-based models such as BERT can improve semantic skill matching instead of relying only on keyword-based scores.

- **Deep Learning Models:**
Future work can explore deep learning models such as Neural Networks or Transformer-based architectures to handle large-scale recruitment data and complex relationships between candidate profiles and job requirements. These models may improve generalization for unseen job roles.

- **Real-Time ATS Evaluation:**
The current system operates on batch predictions. Introducing real-time ATS scoring and suppression prediction would allow candidates to instantly see the impact of resume changes, improving usability and engagement.

- **Feedback-Based Continuous Learning:**

The model can be retrained periodically using recruiter feedback, interview outcomes, or application success data. This continuous learning approach would help the system adapt to changing hiring trends and reduce bias over time.

- **Explainability Enhancement:**

Although rule-based explanations are implemented, advanced explainable AI techniques such as SHAP or LIME can be incorporated to provide deeper insights into feature influence for each prediction, increasing trust among users.

- **Power BI Dashboard Expansion:**

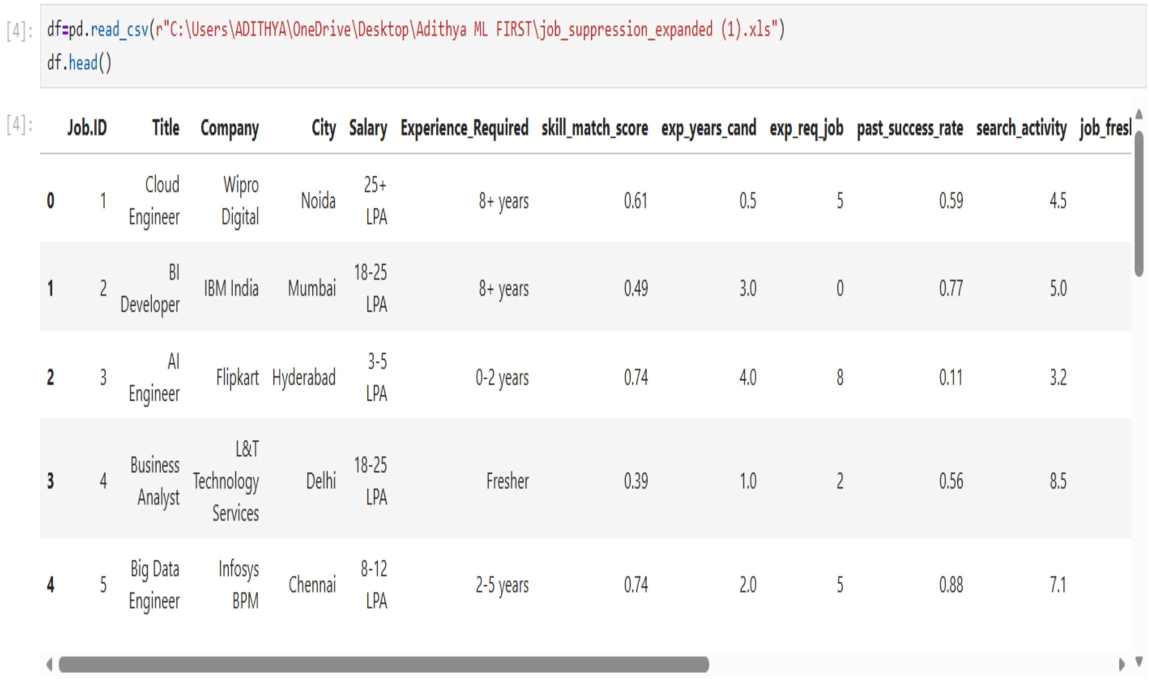
Power BI dashboards can be extended with drill-down and role-based views for recruiters and candidates. Real-time dashboard refresh and comparison across companies, cities, and experience levels would further improve decision-making.

- **Cloud Deployment and Scalability:**

Deploying the system on cloud platforms such as AWS or Azure with secure authentication can improve scalability and allow integration with real job portals, making the solution industry-ready.

APPPENDIX- I

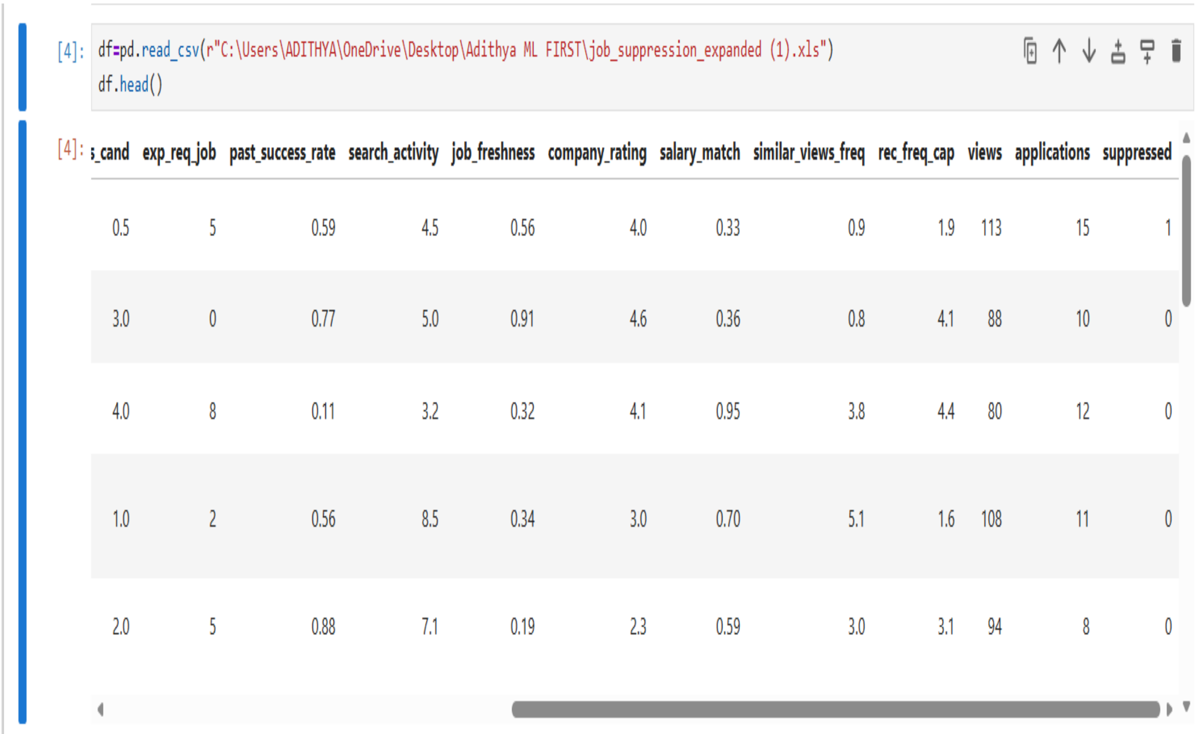
Dataset Preview:



```
[4]: df=pd.read_csv(r"C:\Users\ADITHYA\OneDrive\Desktop\Adithya ML FIRST\job_suppression_expanded (1).xls")
     df.head()
```

[4]:

	Job.ID	Title	Company	City	Salary	Experience_Required	skill_match_score	exp_years_cand	exp_req_job	past_success_rate	search_activity	job_fresl
0	1	Cloud Engineer	Wipro Digital	Noida	25+ LPA	8+ years	0.61	0.5	5	0.59	4.5	
1	2	BI Developer	IBM India	Mumbai	18-25 LPA	8+ years	0.49	3.0	0	0.77	5.0	
2	3	AI Engineer	Flipkart	Hyderabad	3-5 LPA	0-2 years	0.74	4.0	8	0.11	3.2	
3	4	Business Analyst	L&T Technology Services	Delhi	18-25 LPA	Fresher	0.39	1.0	2	0.56	8.5	
4	5	Big Data Engineer	Infosys BPM	Chennai	8-12 LPA	2-5 years	0.74	2.0	5	0.88	7.1	



```
[4]: df=pd.read_csv(r"C:\Users\ADITHYA\OneDrive\Desktop\Adithya ML FIRST\job_suppression_expanded (1).xls")
     df.head()
```

[4]:

s_cand	exp_req_job	past_success_rate	search_activity	job_freshness	company_rating	salary_match	similar_views_freq	rec_freq_cap	views	applications	suppressed
0.5	5	0.59	4.5	0.56	4.0	0.33	0.9	1.9	113	15	1
3.0	0	0.77	5.0	0.91	4.6	0.36	0.8	4.1	88	10	0
4.0	8	0.11	3.2	0.32	4.1	0.95	3.8	4.4	80	12	0
1.0	2	0.56	8.5	0.34	3.0	0.70	5.1	1.6	108	11	0
2.0	5	0.88	7.1	0.19	2.3	0.59	3.0	3.1	94	8	0

Appendix- Ⅱ

```
# 1. Imports Libraries
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    f1_score, roc_auc_score, classification_report,
    ConfusionMatrixDisplay
)

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier  # pip install xgboost if needed

import matplotlib.pyplot as plt
import seaborn as sns
```

Loading Dataset :

```
df=pd.read_csv(r"C:\Users\ADITHYA\OneDrive\Desktop\Adithya ML
FIRST\job_suppression_expanded (1).xls")

df.head()
```

Data Info:

```
df.info()

df.describe(include='all')
```

Duplicates Check :

```
df.duplicated().sum()

df.isna().sum()
```

Salary Parser :

```
# Parse Salary like "18-25 LPA", "25 LPA" to numeric LPA

def parse_salary_lpa(s):

    if pd.isna(s):

        return np.nan

    s = str(s).strip()

    # Remove text

    s = s.replace('LPA', '').replace('lpa', '').strip()


    # "25"

    if '-' not in s:

        val = s.replace('-', '').strip()

        try:

            return float(val)

        except ValueError:

            return np.nan


    # "18-25"
    if '-' in s:
        low, high = s.split('-')
        try:
            return (float(low.strip()) + float(high.strip())) / 2.0
        except ValueError:
            return np.nan

    # "5"
    try:
        return float(s)
    except ValueError:
        return np.nan

df['Salary'] = df['Salary'].apply(parse_salary_lpa)
```

Experience Parser:

```
# Parse experience like "0-2 years", "8 years" to numeric years
def parse_years(s):
    if pd.isna(s):
        return np.nan
    s = str(s).strip().lower()
    # remove words
    for token in ['years', 'year', 'yrs', 'yr']:
        s = s.replace(token, '')
    s = s.strip()

    # "8"
    if '-' not in s:
        val = s.replace('-', '').strip()
        try:
            return float(val)
        except ValueError:
            return np.nan

    # "0-2"
    if '-' in s:
        low, high = s.split('-')
        try:
            return (float(low.strip()) + float(high.strip())) / 2.0
        except ValueError:
            return np.nan

    # "2"
    try:
        return float(s)
    except ValueError:
        return np.nan

# apply to both experience columns that may contain strings
df['ExperienceRequired'] = df['ExperienceRequired'].apply(parse_years)
df['expreq_job'] = df['expreq_job'].apply(parse_years)
# if expyearscand is also text, parse it too
df['expyearscand'] = df['expyearscand'].apply(parse_years)
```

Column Definitions & Cleaning:

# Define numeric & categorical columns

```
num_cols = [
    'Salary', 'ExperienceRequired', 'skillmatchscore', 'expyearscand',
    'expreq_job', 'past_success_rate', 'search_activity', 'job_freshness',
    'company_rating', 'salary_match', 'similar_views_freq', 'rec_freq_cap',
    'views', 'applications'
]

cat_cols = ['Title', 'Company', 'City']

# Convert numerics safely and fill NaNs
for col in num_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
    df[col] = df[col].fillna(df[col].median())

# Clean categoricals
for col in cat_cols:
    df[col] = df[col].astype(str).fillna('Unknown')

df[num_cols].describe()
```

Synthetic Label Generation

Core Functions (labelsuppression, etc.):

# SYNTHETIC SUPPRESSION LABEL

```
def compute_core_match(row):
    # Skill 0-1
    skill = row['skillmatchscore']
    if skill > 1.5: skill = skill * 100.0
    skill = np.clip(skill, 0, 1)

    # Experience match 0-1
    exp_cand = row['expyearscand']
    exp_req = row['expreq_job']
    if exp_req == 0:
        exp_match = 1.0
    else:
        exp_match = np.clip(exp_cand / exp_req, 0, 1.5)
    exp_match = min(exp_match, 1.0)

    # Salary match 0-1
    sal = row['salary_match']
    if sal > 1.5: sal = sal * 100.0
    sal = np.clip(sal, 0, 1)

    core = 0.6 * skill + 0.25 * exp_match + 0.15 * sal
    return core * 100.0

# ... (full functions: norm_val, compute_engagement_boost, compute_suppression_flags, label_suppression, build_reason_text)
```

Train-Test & Preprocessor:

```
X = df.drop(columns=['suppressed', 'suppressedsynthetic',
            'suppressionreasonsrule', 'Job.ID'], errors='ignore')
y = df['suppressedsynthetic']


X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.2, random_state=42, stratify=y
)


numeric_transformer = Pipeline(steps=[
  ('scaler', StandardScaler())])


categorical_transformer = Pipeline(steps=[
  ('ohe', OneHotEncoder(handle_unknown='ignore'))])


preprocessor = ColumnTransformer(
  transformers=[
    ('num', numeric_transformer, num_cols),
    ('cat', categorical_transformer, cat_cols),
  ])
```

BASELINE MODELS:

```
models = {
  'LogisticRegression': LogisticRegression(class_weight='balanced', max_iter=500, random_state=42),
  'RandomForest': RandomForestClassifier(n_estimators=200, random_state=42, class_weight='balanced'),
  'GradientBoosting': GradientBoostingClassifier(random_state=42),
  'DecisionTree': DecisionTreeClassifier(random_state=42, class_weight='balanced'),
  'SVC': SVC(probability=True, class_weight='balanced', random_state=42),
  'XGBClassifier': XGBClassifier(objective='binary:logistic',
                                 eval_metric='logloss',
                                 use_label_encoder=False,
                                 random_state=42,
                                 scale_pos_weight=1),
}

results = []
for name, clf in models.items():
  pipe = Pipeline([
      ('prep', preprocessor),
      ('model', clf)
  ])
  cv_scores = cross_val_score(pipe, X_train, y_train, cv=5, scoring='f1')
  results.append({
      'model': name,
      'mean_f1': cv_scores.mean(),
      'std_f1': cv_scores.std()
  })
  print(f'{name} F1: {cv_scores.mean():.3f} - {cv_scores.std():.3f}')

results_df = pd.DataFrame(results).sort_values('mean_f1', ascending=False)
results_df
```

HYPERPARAMETER TUNING XGB:

```
xgb_pipe = Pipeline([
    ('prep', preprocessor),
    ('model', XGBClassifier(objective='binary:logistic',
                           eval_metric='logloss',
                           use_label_encoder=False,
                           random_state=42))
])

param_grid = {
    'model__n_estimators': [200, 400],
    'model__max_depth': [3, 5, 7],
    'model__learning_rate': [0.01, 0.05, 0.1],
    'model__subsample': [0.8, 1.0],
    'model__colsample_bytree': [0.8, 1.0],
    'model__scale_pos_weight': [1, 2, 5],
}

grid_search = GridSearchCV(xgb_pipe, param_grid,
                          cv=3, scoring='f1',
                          n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

print('Best params:', grid_search.best_params_)
print('Best CV F1:', grid_search.best_score_)

best_model = grid_search.best_estimator_
```

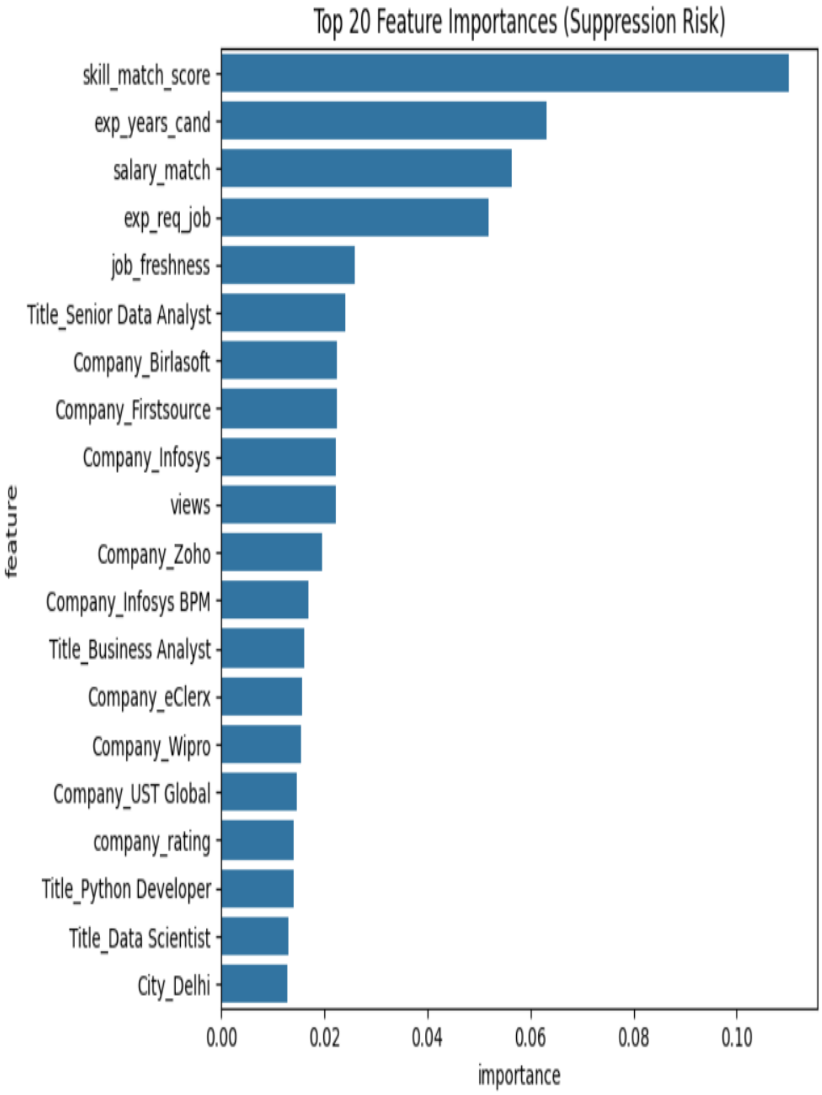
Test set predictions & metrics:

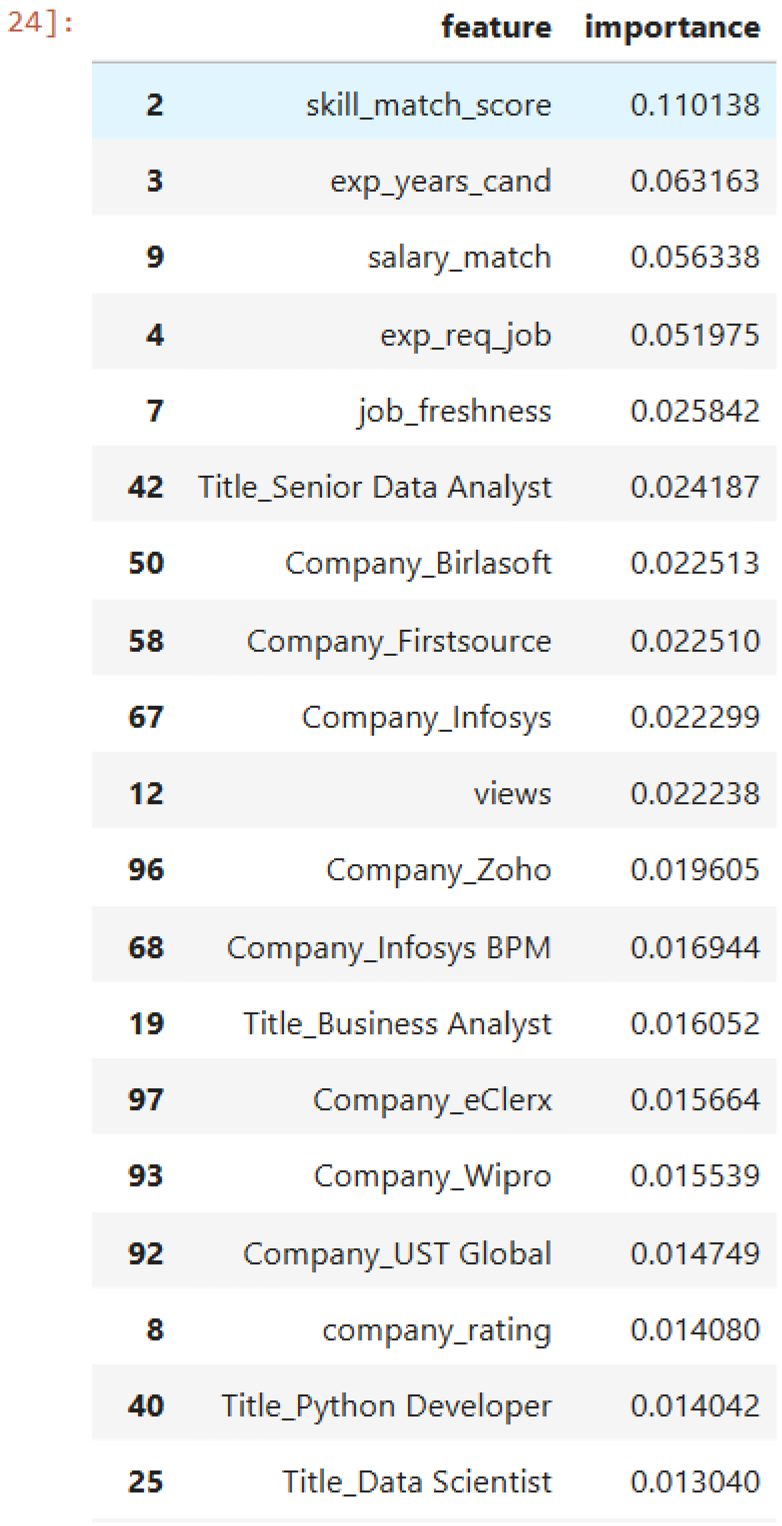
```
y_pred = best_model.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('F1-score:', f1_score(y_test, y_pred))
print('ROC-AUC:', roc_auc_score(y_test, best_model.predict_proba(X_test)[:, 1]))

print('\nClassification report:\n', classification_report(y_test, y_pred))

# Confusion Matrix
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.show()
```

FEATURE IMPORTANCE:

```
ohe = best_model.named_steps["prep"].named_transformers_["cat"]["ohe"]
cat_feature_names = ohe.get_feature_names_out(cat_cols)
all_feature_names = np.concatenate([num_cols, cat_feature_names])

xgb = best_model.named_steps["model"]
importances = xgb.feature_importances_

feat_imp = (
    pd.DataFrame({"feature": all_feature_names, "importance": importances})
    .sort_values("importance", ascending=False)
    .head(20)
)

plt.figure(figsize=(8, 6))
```

```
sns.barplot(data=feat_imp, x="importance", y="feature")

plt.title("Top 20 Feature Importances (Suppression Risk)")

plt.tight_layout()

plt.show()


feat_imp
```





24]:

	feature	importance
2	skill_match_score	0.110138
3	exp_years_cand	0.063163
9	salary_match	0.056338
4	exp_req_job	0.051975
7	job_freshness	0.025842
42	Title_Senior Data Analyst	0.024187
50	Company_Birlasoft	0.022513
58	Company_Firstsource	0.022510
67	Company_Infosys	0.022299
12	views	0.022238
96	Company_Zoho	0.019605
68	Company_Infosys BPM	0.016944
19	Title_Business Analyst	0.016052
97	Company_eClerx	0.015664
93	Company_Wipro	0.015539
92	Company_UST Global	0.014749
8	company_rating	0.014080
40	Title_Python Developer	0.014042
25	Title_Data Scientist	0.013040

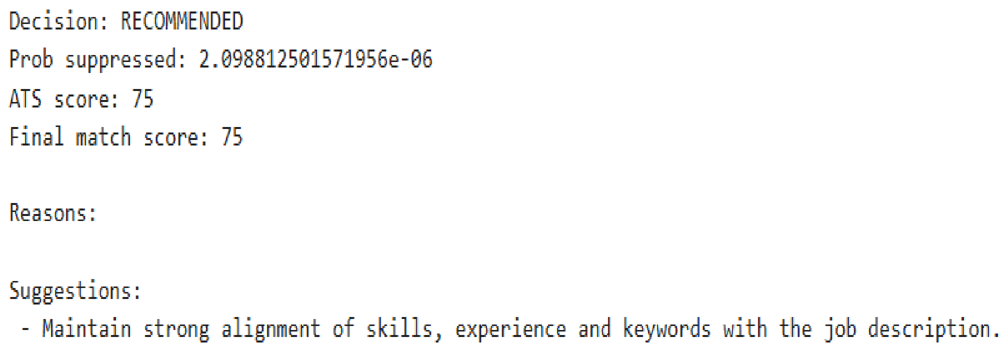
ATS CHECKER + EXPLANATION :

```
def compute_ats_score(row, p_sup, df_all):
    skill = row["skill_match_score"]
    if skill > 1.5:
        skill = skill / 100.0
    skill = np.clip(skill, 0, 1)

    exp_cand = row["exp_years_cand"]
    exp_req = row["exp_req_job"]
    if exp_req <= 0:
        exp_match = 1.0
    else:
        exp_match = np.clip(exp_cand / exp_req, 0, 1.5)
    exp_match = min(exp_match, 1.0)

    fresh = norm(row["job_freshness"], "job_freshness", df_all)
    act   = norm(row["search_activity"], "search_activity", df_all)

    raw_score = 0.6 * skill + 0.2 * exp_match + 0.1 * fresh + 0.1 * act
    ats_score = int(round(raw_score * 100))

    final_score = int(round(ats_score * (1 - p_sup)))
    return final_score, ats_score


def generate_reasons_and_suggestions(row, p_sup, ats_score, final_score,
                                     df_all, threshold=0.5):
```

```
    reasons, suggestions = [], []
    suppressed_pred = int(p_sup >= threshold)

    # Skill
    skill = row["skill_match_score"]
    if skill > 1.5:
        skill_norm = skill / 100.0
    else:
        skill_norm = skill
    if skill_norm < 0.5:
        reasons.append("Low skill match with job requirements.")
        suggestions.append("Add more role-specific skills and keywords from the job description to the resume.")
    elif skill_norm < 0.7:
        suggestions.append("Highlight more relevant tools and technologies in the resume.")

    # Experience
    exp_cand = row["exp_years_cand"]
    exp_req = row["exp_req_job"]
    if exp_req > 0 and exp_cand + 0.5 < exp_req:
        reasons.append("Experience below job requirement.")
        suggestions.append("Gain project or internship experience and clearly show years on the resume.")

    # Salary
    sal = row["salary_match"]
    if sal > 1.5:
        sal = sal / 100.0
    if sal < 0.4:
        reasons.append("Salary expectation does not match typical range.")
        suggestions.append("Align salary expectations with market or justify with strong achievements.")

    # History
    if row["past_success_rate"] < df_all["past_success_rate"].median():
        reasons.append("Low past success rate for similar jobs.")
        suggestions.append("Focus on roles where your skills match better and refine applications using feedback.")

    # ATS keyword density proxy
    if ats_score < 60:
        reasons.append("Resume may lack enough targeted keywords (low ATS score).")
        suggestions.append("Add detailed bullet points with tools, technologies and measurable outcomes from the JD.")

    if not suggestions:
        suggestions.append("Maintain strong alignment of skills, experience and keywords with the job description.")

    return {
        "suppressed_pred": suppressed_pred,
        "decision_label": "SUPPRESSED" if suppressed_pred == 1 else "RECOMMENDED",
        "prob_suppressed": float(p_sup),
        "ats_score": int(ats_score),
        "final_match_score": int(final_score),
        "reasons": reasons,
        "suggestions": suggestions,
  }
```

Example:

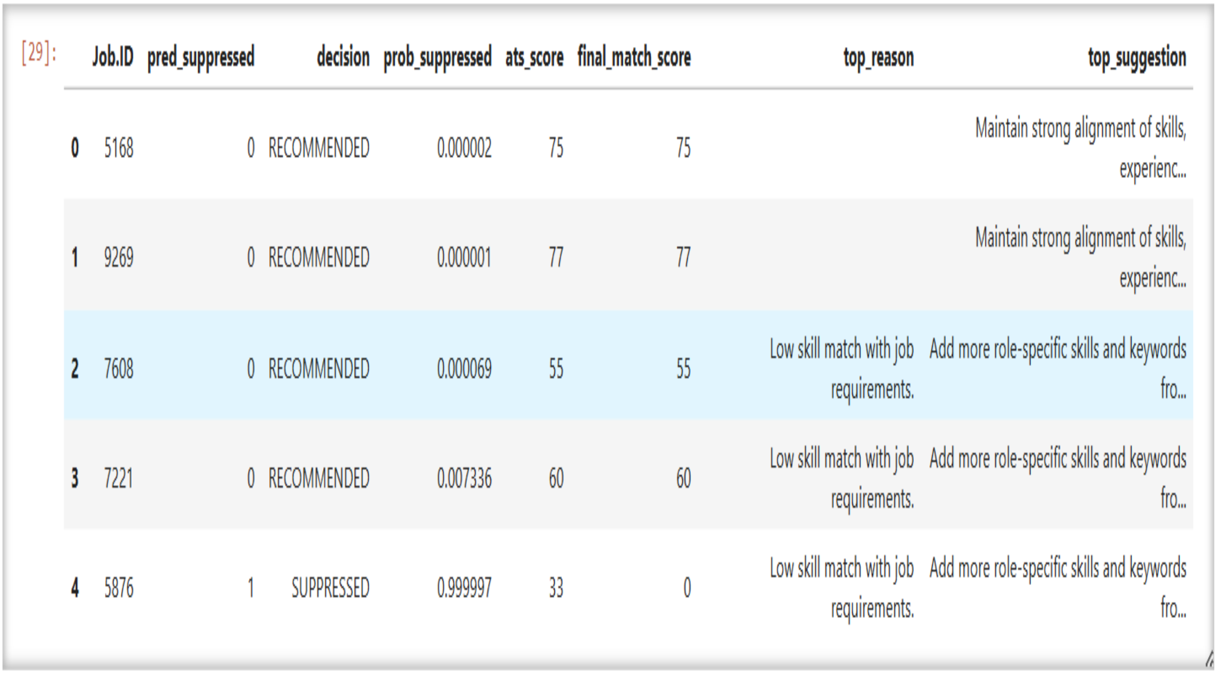
```
sample_idx = X_test.index[0]
sample_row = X.loc[sample_idx]
sample_df = pd.DataFrame([sample_row])


p_sup = best_model.predict_proba(sample_df)[:, 1][0]
final_score, ats_score = compute_ats_score(sample_row, p_sup, df)


exp = generate_reasons_and_suggestions(sample_row, p_sup, ats_score, final_score, df)


print("Decision:", exp["decision_label"])
print("Prob suppressed:", exp["prob_suppressed"])
print("ATS score:", exp["ats_score"])
print("Final match score:", exp["final_match_score"])
print("\nReasons:")
for r in exp["reasons"]:
  print(" -", r)
print("\nSuggestions:")
for s in exp["suggestions"]:
  print(" -", s)
```



```
Decision: RECOMMENDED
Prob suppressed: 2.098812501571956e-06
ATS score: 75
Final match score: 75

Reasons:

Suggestions:
 - Maintain strong alignment of skills, experience and keywords with the job description.
```

Batch analysis for table/cards:

```
def analyze_rows(X_subset, df_all, model):
    probs = model.predict_proba(X_subset)[:, 1]
    out_rows = []
    for i, idx in enumerate(X_subset.index):
        row = X.loc[idx]
        p_sup = probs[i]
        final_score, ats_score = compute_ats_score(row, p_sup, df_all)
        exp = generate_reasons_and_suggestions(row, p_sup, ats_score, final_score, df_all)
        out_rows.append({
            "Job.ID": df_all.loc[idx, "Job.ID"],
            "pred_suppressed": exp["suppressed_pred"],
            "decision": exp["decision_label"],
            "prob_suppressed": exp["prob_suppressed"],
            "ats_score": exp["ats_score"],
            "final_match_score": exp["final_match_score"],
            "top_reason": exp["reasons"][0] if exp["reasons"] else "",
            "top_suggestion": exp["suggestions"][0] if exp["suggestions"] else "",
        })
    return pd.DataFrame(out_rows)


analysis_df = analyze_rows(X_test, df, best_model)


# <<< add this line >>>
analysis_df["prob_suppressed"] = analysis_df["prob_suppressed"].map("{:.6f}".format)


analysis_df.head()
```



[29]:

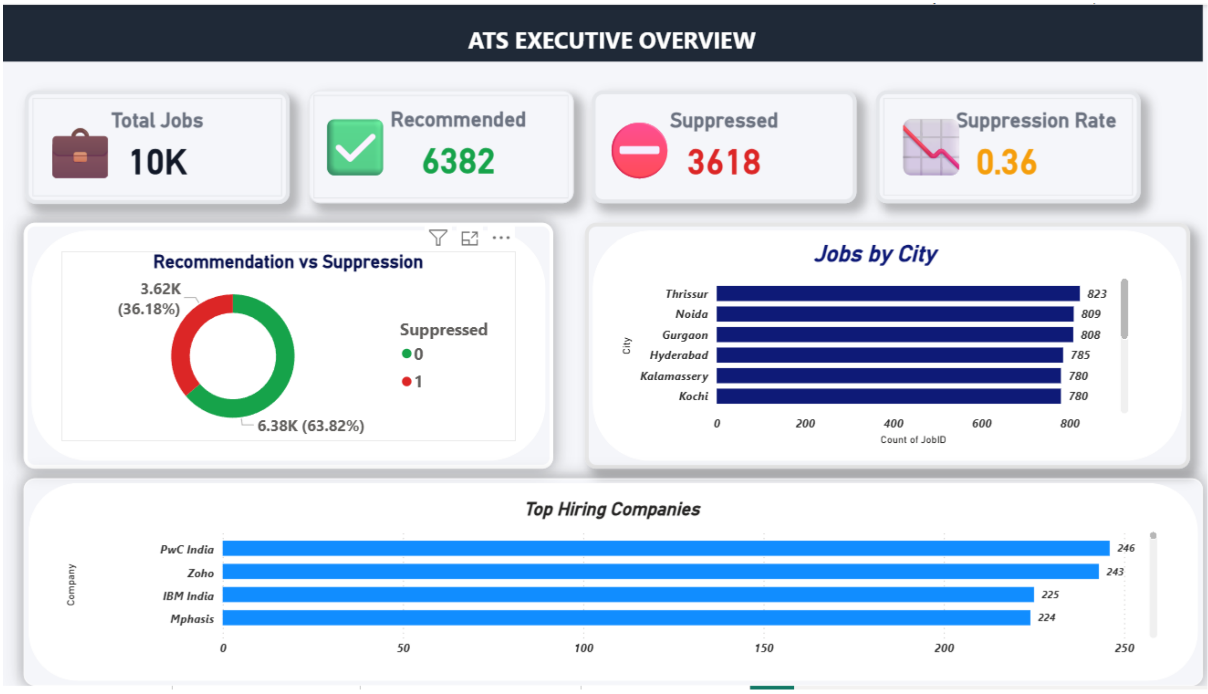
	Job.ID	pred_suppressed	decision	prob_suppressed	ats_score	final_match_score	top_reason	top_suggestion
0	5168	0	RECOMMENDED	0.000002	75	75		Maintain strong alignment of skills, experienc...
1	9269	0	RECOMMENDED	0.000001	77	77		Maintain strong alignment of skills, experienc...
2	7608	0	RECOMMENDED	0.000069	55	55	Low skill match with job requirements.	Add more role-specific skills and keywords fro...
3	7221	0	RECOMMENDED	0.007336	60	60	Low skill match with job requirements.	Add more role-specific skills and keywords fro...
4	5876	1	SUPPRESSED	0.999997	33	0	Low skill match with job requirements.	Add more role-specific skills and keywords fro...

Saving Model:

```
import joblib


# save trained model
joblib.dump(best_model, "job_suppression_model.joblib")


# (optional) also save the dataset used for analysis, if you want
analysis_df.to_csv("job_suppression_analysis_sample.csv", index=False)
```

Appendix-Ⅲ

POWER BI DASHBOARDS:



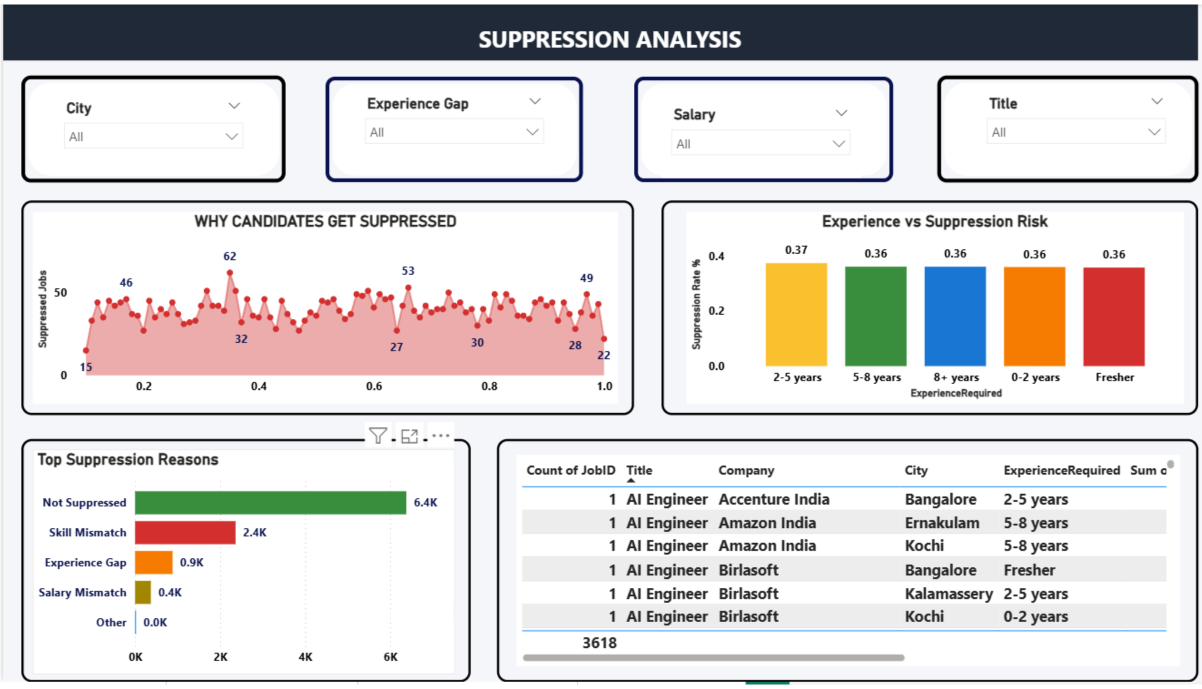
ATS EXECUTIVE OVERVIEW

Total Jobs 10K

Recommended 6382

Suppressed 3618

Suppression Rate 0.36



SUPPRESSION ANALYSIS

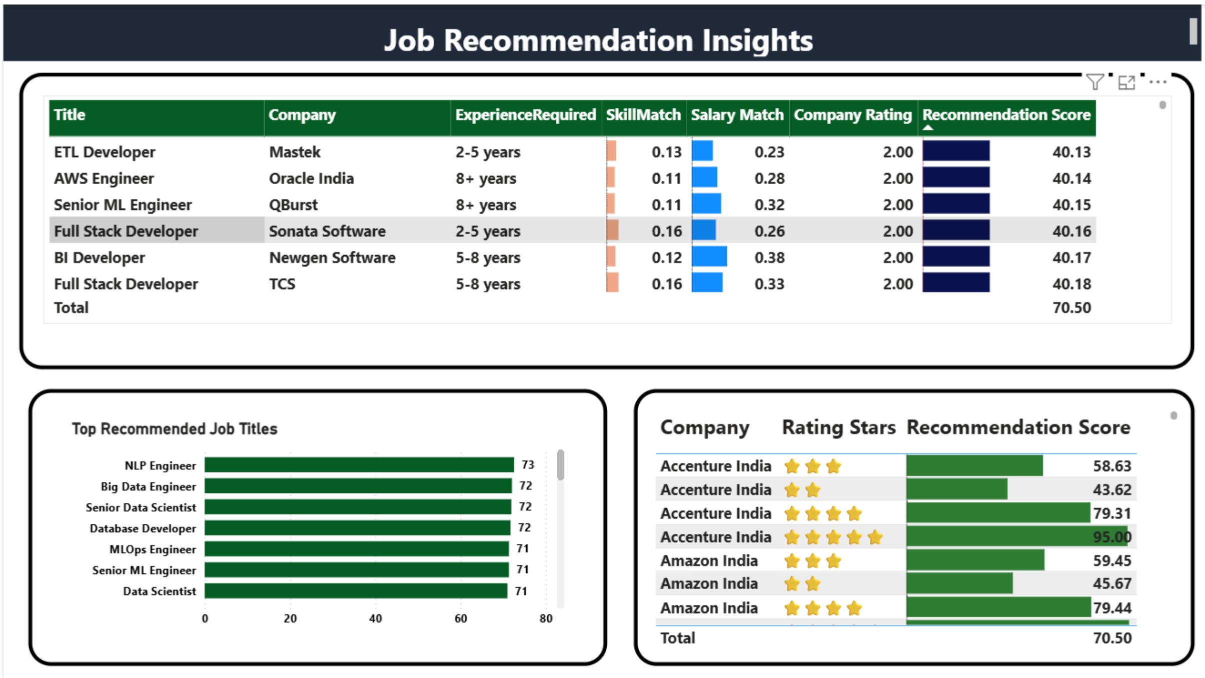
City: All

Experience Gap: All

Salary: All

Title: All

Count of JobID	Title	Company	City	ExperienceRequired	Sum o
1	AI Engineer	Accenture India	Bangalore	2-5 years	
1	AI Engineer	Amazon India	Ernakulam	5-8 years	
1	AI Engineer	Amazon India	Kochi	5-8 years	
1	AI Engineer	Birlasoft	Bangalore	Fresher	
1	AI Engineer	Birlasoft	Kalamassery	2-5 years	
1	AI Engineer	Birlasoft	Kochi	0-2 years	
3618					

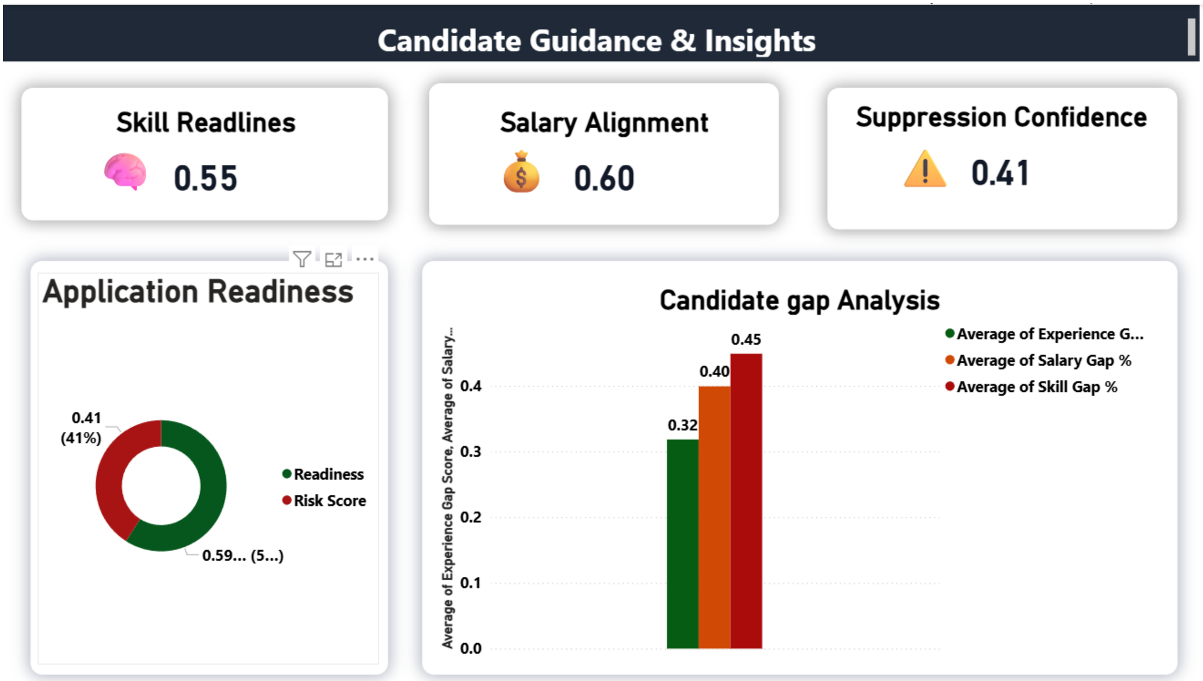


Job Recommendation Insights

Title	Company	ExperienceRequired	SkillMatch	Salary Match	Company Rating	Recommendation Score
ETL Developer	Mastek	2-5 years	0.13	0.23	2.00	40.13
AWS Engineer	Oracle India	8+ years	0.11	0.28	2.00	40.14
Senior ML Engineer	QBurst	8+ years	0.11	0.32	2.00	40.15
Full Stack Developer	Sonata Software	2-5 years	0.16	0.26	2.00	40.16
BI Developer	Newgen Software	5-8 years	0.12	0.38	2.00	40.17
Full Stack Developer	TCS	5-8 years	0.16	0.33	2.00	40.18
Total						70.50

Top Recommended Job Titles

Company	Rating Stars	Recommendation Score
Accenture India	★★★	58.63
Accenture India	★★	43.62
Accenture India	★★★★	79.31
Accenture India	★★★★★	95.00
Amazon India	★★★	59.45
Amazon India	★★	45.67
Amazon India	★★★★	79.44
Total		70.50



Candidate Guidance & Insights

Skill Readlines
🧠 0.55

Salary Alignment
💰 0.60

Suppression Confidence
⚠ 0.41

Application Readiness

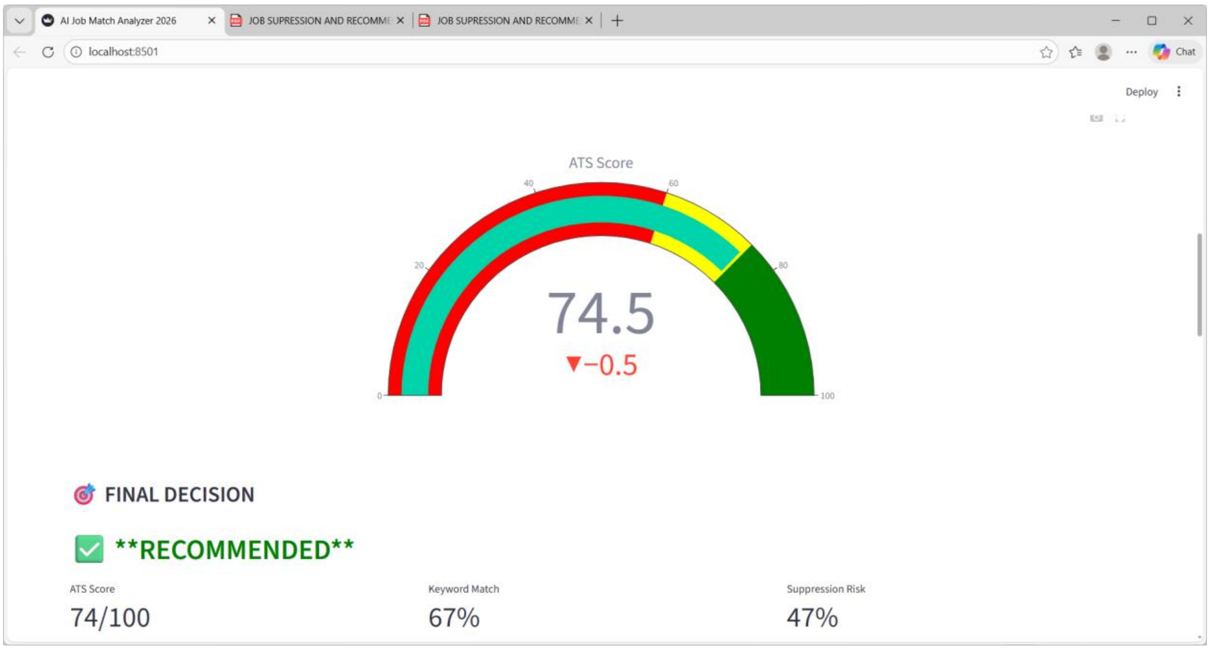
0.41
(41%)
0.59... (5...)
● Readiness
● Risk Score

Candidate gap Analysis

Average of Experience Gap Score, Average of Salary...
0.0
0.1
0.2
0.3
0.4
0.32
0.40
0.45
● Average of Experience G...
● Average of Salary Gap %
● Average of Skill Gap %

Appendix- iv

Streamlit :


AI Job Match Analyzer 2026
JOB SUPRESSION AND RECOMME
JOB SUPRESSION AND RECOMME
localhost:8501
Chat
Deploy
ATS Score
0
20
40
60
80
100

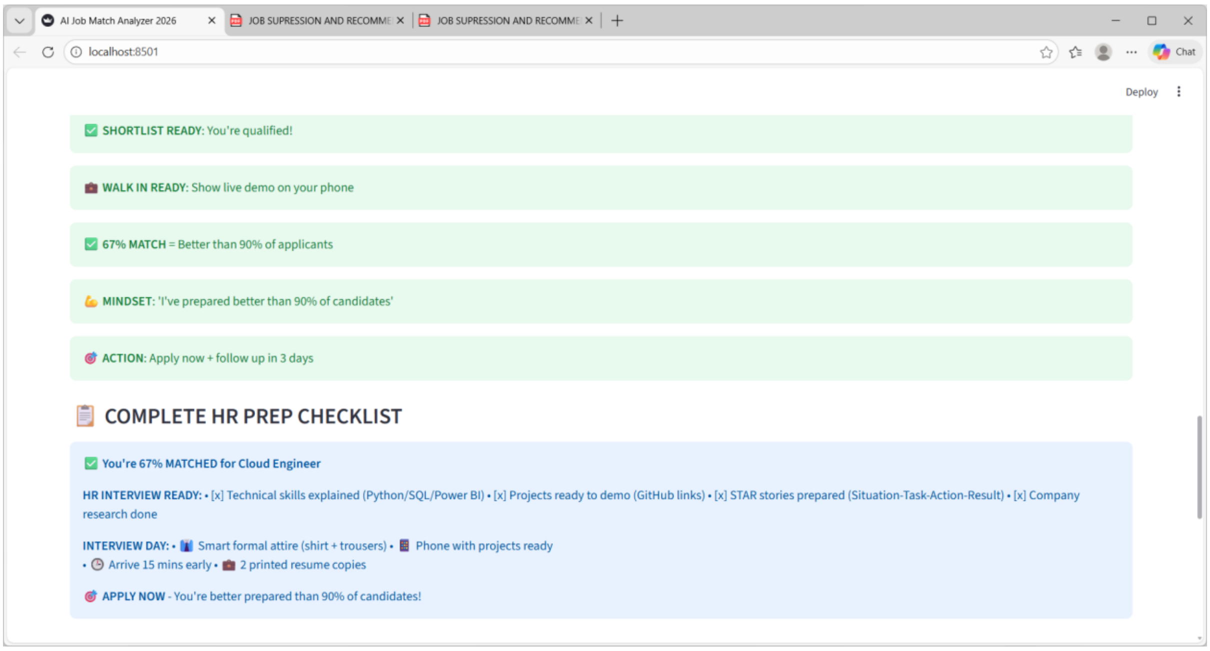
74.5

▼−0.5

🎯 FINAL DECISION

✅ **RECOMMENDED**

ATS Score	Keyword Match	Suppression Risk
74/100	67%	47%


AI Job Match Analyzer 2026
JOB SUPRESSION AND RECOMME
JOB SUPRESSION AND RECOMME
localhost:8501
Chat
Deploy

✅ **SHORTLIST READY:** You're qualified!

💼 **WALK IN READY:** Show live demo on your phone

✅ **67% MATCH** = Better than 90% of applicants

💪 **MINDSET:** 'I've prepared better than 90% of candidates'

🎯 **ACTION:** Apply now + follow up in 3 days

📋 COMPLETE HR PREP CHECKLIST

✅ **You're 67% MATCHED for Cloud Engineer**

HR INTERVIEW READY: • [x] Technical skills explained (Python/SQL/Power BI) • [x] Projects ready to demo (GitHub links) • [x] STAR stories prepared (Situation-Task-Action-Result) • [x] Company research done

INTERVIEW DAY: • 👔 Smart formal attire (shirt + trousers) • 📱 Phone with projects ready
• 🕒 Arrive 15 mins early • 💼 2 printed resume copies

🎯 **APPLY NOW** - You're better prepared than 90% of candidates!