# Smart Resume Screening System

*Team Members :*

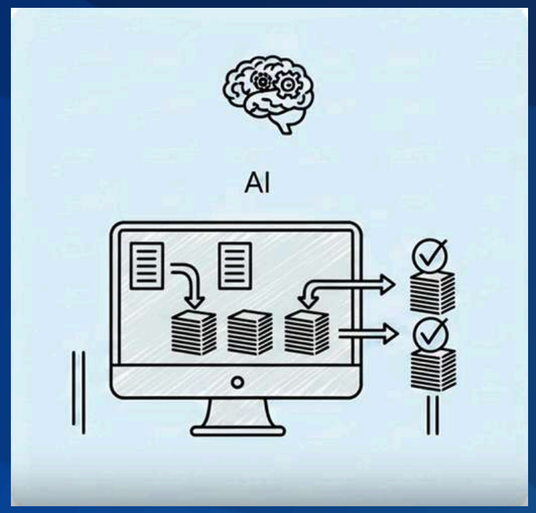
*Ananya Tyagi - UCE2023604*
*Sanisa - UCE2023606*
*Shrishti Chandak - UCE2023609*
*Sakshi Kale - UCE2023629*

02

# The Problem & The Solution :


***The Challenge in Modern Recruitment***

- ***Volume***: *Hundreds of resumes per job posting.*
- ***Time***: *Manual screening is slow, inconsistent, and prone to human bias.*
- ***Efficiency***: *Difficulty identifying the best match quickly.*

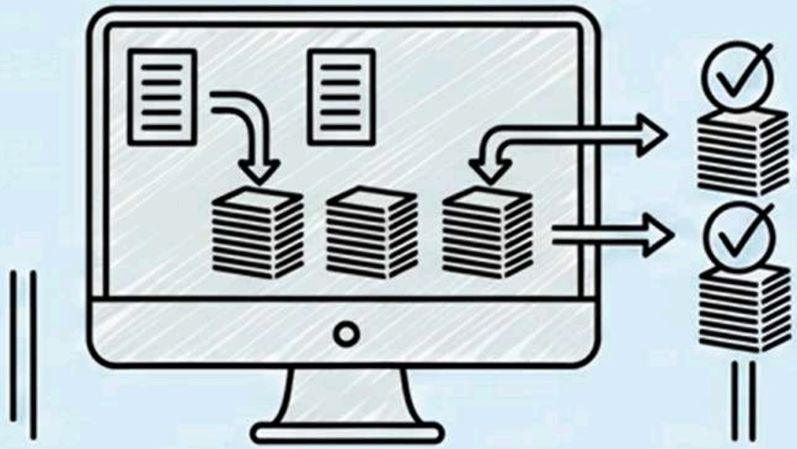



***The Solution: Our AI System***

- *Automated bulk resume processing.*
- *ML-based categorization & ranking.*
- *Advanced talent pool analysis and team formation.*
- ***Key Feature:*** *Training models (like KNN) from scratch for transparency.*

Manual

AI



# 03 System Overview & Architecture

***Core Components (React & ML Backend)***

- ***Frontend:*** *React UI for Bulk Classification, Matching, and Talent Pool Management.*
- ***Backend Logic:*** *Python/ML Script for training, featuring TF-IDF, KNN, and K-Means.*

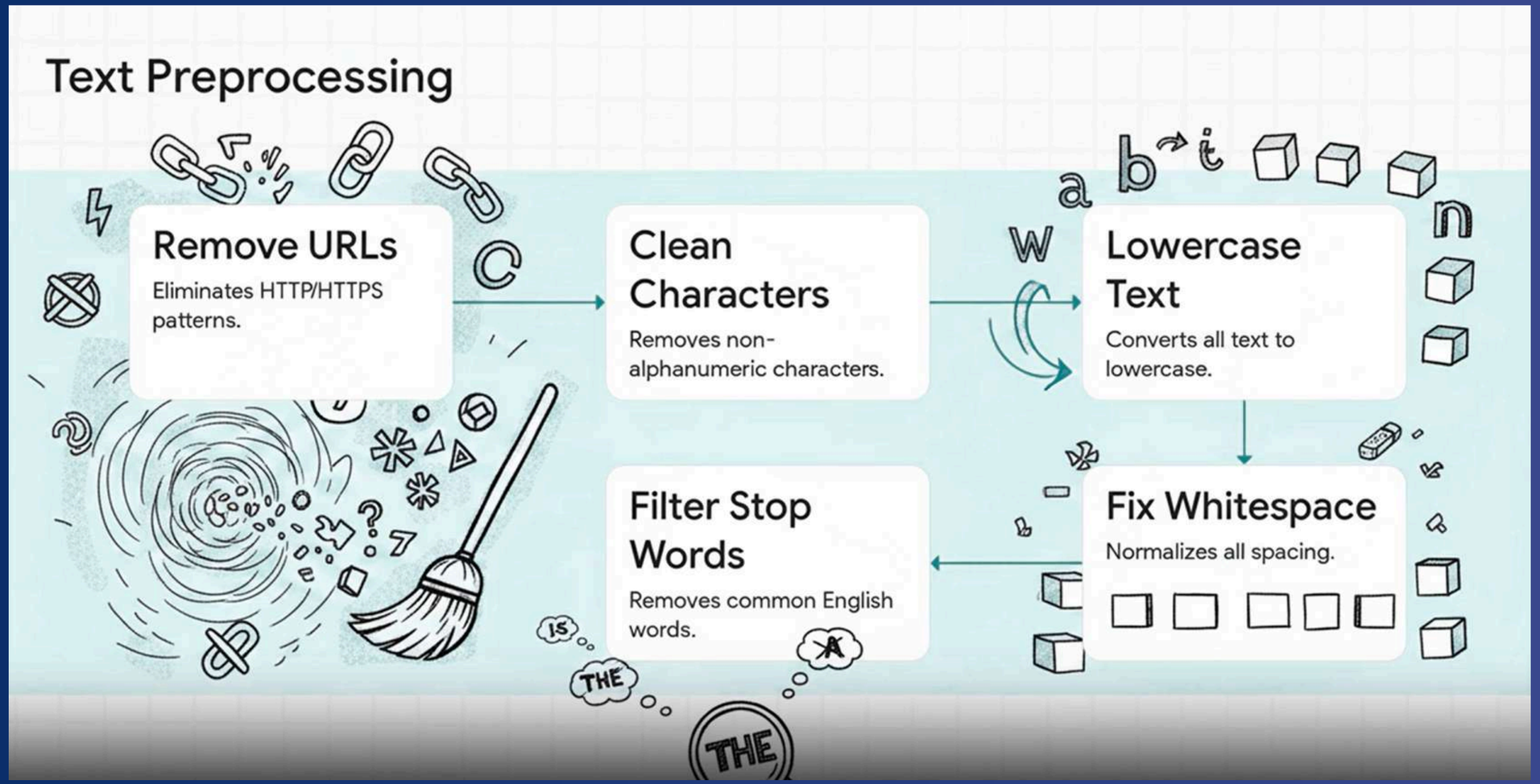
- ***Architecture Flow:***
- *Resume Text Input*
- ***Text Preprocessing***
- ***Feature Extraction (TF-IDF)***
- ***ML Model (KNN/K-Means)***
- *Results (Category, Score, Cluster).*

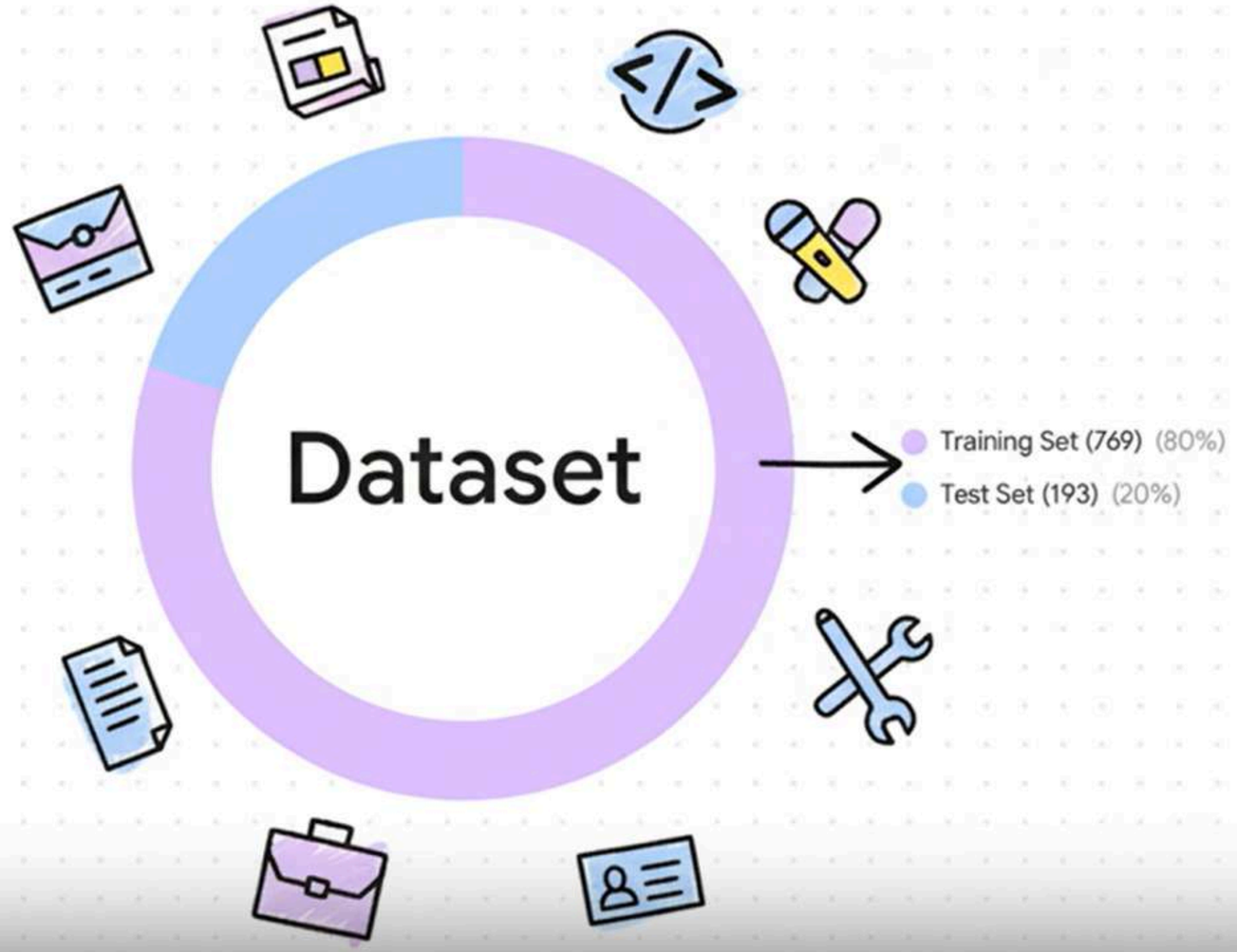
04

# NLP Step 1: Text Preprocessing

- **Goal:** Clean and Normalize the Resume Data

- Process (cleanText function):
- Lowercasing all text.
- Removal of URLs, special characters (punctuation), and multiple spaces**(Tokenization).**
- Removal of Stop Words (e.g., 'the', 'a', 'is').
- **Why it Matters:** A clean dataset is crucial for accurate feature extraction. Garbage In, Garbage Out (GIGO).


Text Preprocessing
Remove URLs
Eliminates HTTP/HTTPS patterns.
Clean Characters
Removes non-alphanumeric characters.
Lowercase Text
Converts all text to lowercase.
Fix Whitespace
Normalizes all spacing.
Filter Stop Words
Removes common English words.
IS
A
THE
THE




The AI is trained on real-world resumes, 80/20 for testing.

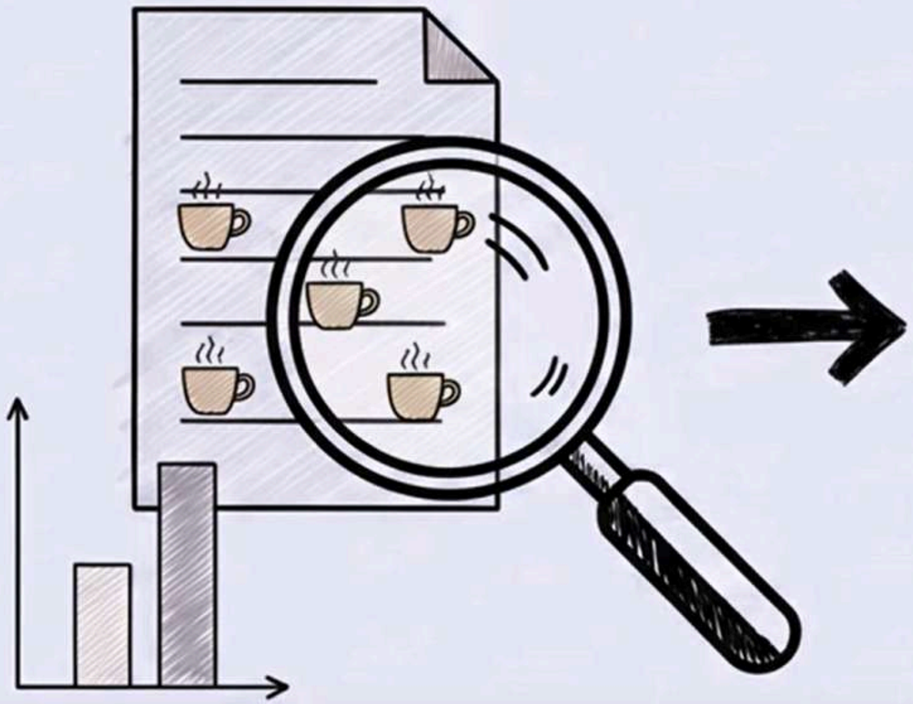
05

# NLP Step 2: Feature Extraction (TF-IDF)

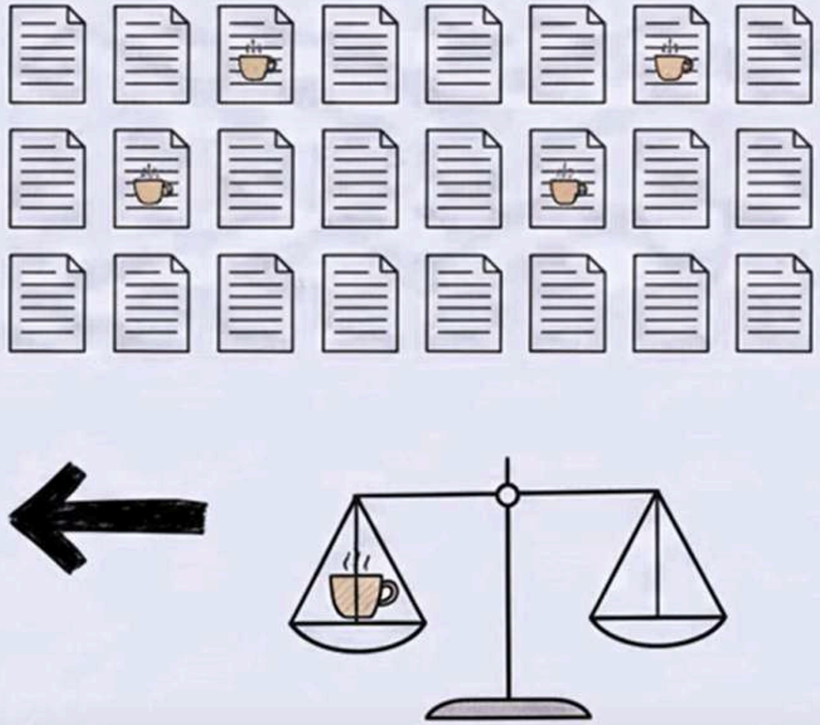
**Converting Text to Numerical Vectors**

- **Algorithm: Term Frequency-Inverse Document Frequency (TF-IDF)**
- **TF (Term Frequency):** How often a word appears in one resume.
- **IDF (Inverse Document Frequency):** How rare a word is across all resumes.
- **Formula Focus:** High TF-IDF score means the word is frequent in a specific resume but rare across the entire dataset—i.e., it's a powerful, distinguishing keyword (e.g., 'TensorFlow' vs. 'work').
- Code Implementation: computeTFIDF function calculates TF, DF, and IDF to create the final vector representations.

## Term Frequency (TF):



## Inverse Document Freq. (IDF):



05

# Title: The NLP Pipeline: How We Create Smart Features

## Our Optimized NLP Pipeline

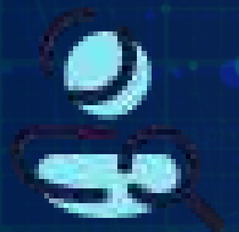
- 1. POS Tagging: We first identify the grammatical role of every word (e.g., Noun, Verb).
- 2. Lemmatization: Using the POS tag, we reduce words to their dictionary root. This groups developing, developed, and developer into one strong feature: develop.
- 3. TF-IDF Vectorization: We convert this clean, lemmatized text into numerical vectors that emphasize the importance of rare skills over common words.
- 4. Duplicate Removal: We found and removed 369 duplicate resumes to prevent skewed results and ensure data quality.

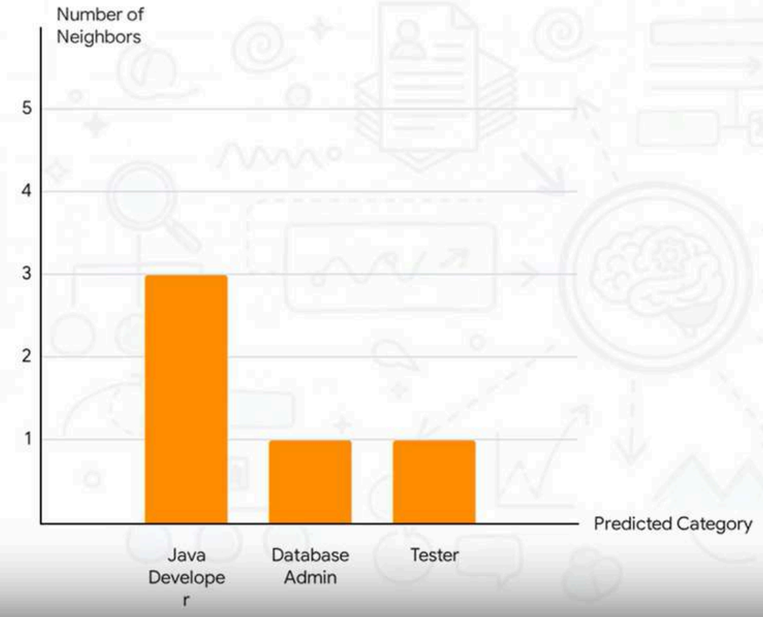
06

# Bulk Classification: The KNN Model

**Automatic Categorization**

- **Algorithm: K-Nearest Neighbors (KNN)**
  - **How it Works:** A new resume's TF-IDF vector is classified by finding the K=5 closest (most similar) resumes in the training data (which already have known categories).
  - **Similarity Metric:** Cosine Similarity (angle between vectors).
- **Function:** knnClassify is used for Bulk Classification of resumes into 25 categories (e.g., 'Data Science', 'Java Developer').



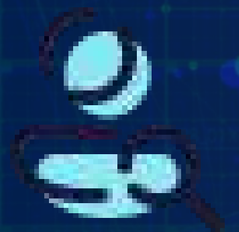



The system performs a majority vote on the categories of the 5 nearest neighbors.

06

# Model Results & Key Findings (SVC)

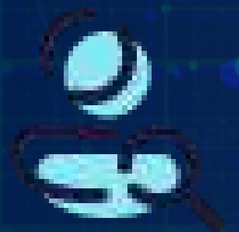
- Model Comparison: We benchmarked three models on our optimized data:
- KNN: Slow and less accurate (79% Accuracy).
- Multinomial NB: Fast but very inaccurate (41% Accuracy).
- LinearSVC : Achieved the highest accuracy (97.9%) and is extremely fast at prediction.
- Key Risk: Algorithmic Bias:
- Our model learns historical patterns, not just skills.
- Example: It learned to associate institutes like "COEP Pune" with the "Java Developer" label, creating an unfair regional bias for candidates from Maharashtra.



08

# Talent Pool Management & Parsing

- **From Resumes to Data Points**
- **Functionality:** Takes all classified resumes and enriches them with extracted features using Heuristics (rule-based parsing).
- **Key Extracted Features (extract... functions):**
- **Skills: (python, react, docker)** via keyword matching.
- **Experience: (e.g., '5 years exp')** via regex.
- **Estimated Salary:** A heuristic calculation based on (Experience *Bonus) + (Skill Count * Bonus).
- **Analytics:** Provides overall insights like Average Experience, Total Budget (potential cost of pool), and unique skills inventory.



09

# Unsupervised Learning: K-Means Clustering

Discovering Hidden Groups within the Talent Pool

Algorithm: K-Means Clustering (Unsupervised Learning).

Goal: Group candidates based on inherent skill similarity, even across different job categories.

Process:

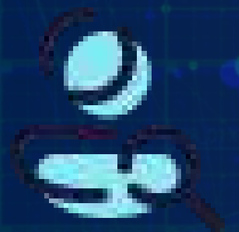
- Uses TF-IDF vectors of skills as input features.
- Calculates cluster centroids (K=3 is a common starting point).
- Assigns members to the nearest centroid.

Outcome: Clusters like Junior Level, Mid Level and Senior Level enabling better talent strategy.

10

# Strategic Team Building (Greedy Algorithm)

- Optimizing Team Composition under Constraints
- Algorithm: Greedy Heuristic
- Input: Required Roles (e.g., 'Java Developer, Data Scientist') and a Max Budget.
- Logic: Iteratively selects the best-qualified available candidate for each required role, ensuring the Total Cost remains below the Max Budget.
- Benefits: Quickly finds a high-confidence team while managing financial constraints, maximizing budget utilization.



11

# Model Performance & Evaluation Metrics

Measuring the KNN Classifier

Key Metrics:

- Accuracy: Overall correctness (82%).
- Precision: How many predicted positives were correct-reducing false positive (81%).
- Recall: How many actual positives were found-reducing false negatives (80%).
- F1 Score: Balance of Precision and Recall (80%).

Conclusion: The KNN model provides robust classification for resume pre-screening.

# Implementation Details & Dataset

## Technical Specifications

- Frontend: React
- Backend Logic:  Python ML using native functions for vector math (for demonstration).
- Feature Size: TF-IDF vectorizer built with 5000 unique words/bigrams.

## Training Dataset (Simulated):

- Total Resumes : 900
- Unique Job Categories: 3-5 (e.g., Data Science, HR, Sales).
- Train/Test Split: 80/20.

# Technical Deep Dive: Cosine Similarity

## The Math Behind Matching

- Concept: Measures the cosine of the angle between two non-zero vectors in a high-dimensional space.
- Formula:

$$\text{Similarity}(\mathbf{A}, \mathbf{B}) = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \cdot ||\mathbf{B}||}$$

A = Job Description TF-IDF Vector.

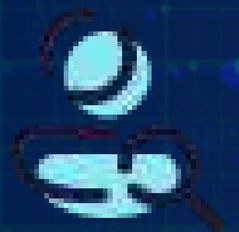
B = Resume TF-IDF Vector.

- Result: A score from 0 (no similarity) to 1 (perfect match). This is the basis for our Bulk Matching scores.

# Future Enhancements

Expanding Capabilities

- Advanced NLP: Implement a pre-trained BERT or Word2Vec model instead of raw TF-IDF for semantic matching.
- Team Builder Optimization: Replace Greedy Heuristics with a Linear Programming or Genetic Algorithm model for true optimal solutions.
- Bias Detection: Introduce algorithms to check for and mitigate bias in salary estimation or candidate ranking based on demographics (Ethical AI).
- Full ML Deployment: Connect the React frontend to a Flask/Django API hosting the saved pickle models for production-level speed.



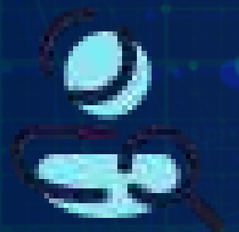
# Conclusion

System Impact

- Efficiency: Drastically reduces manual screening time.
- Quality: Ensures candidates are ranked objectively based on technical similarity to the job description.
- Strategy: Provides strategic insights for proactive talent management (Clustering).

Key Takeaways:

- The system uses TF-IDF for vectorization.
- KNN for classification and Cosine Similarity for matching.
- K-Means for unsupervised skill grouping.




Thank You!