**📑 Summary Report: AI Smart Resume Screener**

**🔹 Objective**

The notebook aims to build an **automated resume screening tool** that compares job descriptions (JDs) with candidate resumes and predicts whether the resume is a **good match (1)** or **not a match (0)**.

**🔹 Dataset**

* A **dummy dataset** is created manually with pairs of job descriptions and resumes.
* Each pair is labelled:
  + **1 → Match (resume fits the job requirements)**
  + **0 → Not a match (resume doesn’t fit well)**
* Example:
  + JD: *Looking for a Python developer with ML and AWS.*
  + Resume: *Python engineer skilled in ML and AWS.*
  + Label: **1**

**🔹 Data Preprocessing**

* **Text Cleaning:**
  + Lowercasing
  + Removing punctuation and numbers
  + Removing stop words
* **Feature Engineering:**
  + Cleaned JD and Resume text combined.
  + TF-IDF Vectorization used to extract numerical features from text.

**🔹 Machine Learning Models**

Two models were implemented:

1. **Random Forest Classifier**
   * Trained on TF-IDF features.
   * Produced an **accuracy ≈ 50%** on test data.
   * Classification report showed **imbalanced performance** (precision = 0 for some classes).
2. **Decision Tree Classifier**
   * Also gave **accuracy ≈ 50%**.
   * Precision and recall values were low for the "No Match" class.

⚠️ Both models **did not perform well** due to:

* Very **small dataset** (only ~10 examples).
* Overfitting on limited training data.

**🔹 Evaluation Metrics**

* Accuracy: **~0.5 (50%)**
* Precision: ~0.5 (but sometimes 0 due to missing predictions for some labels)
* Recall: Poor for "No Match" cases.
* Confusion Matrix showed imbalance.

**🔹 Predictions on New Data**

The model tested with unseen JD–Resume pairs:

* Example:
  + JD: *Looking for ML Engineer with Python & TensorFlow.*
  + Resume: *Engineer skilled in Python, TensorFlow, AI research.*
  + **Predicted Match: 1 (Correct)**
* Another case wrongly predicted as a match, showing **model weakness**.

**🔹 Strengths**

✅ Automated preprocessing and text cleaning  
✅ TF-IDF for feature extraction  
✅ Implemented multiple ML models  
✅ Pipeline includes evaluation and predictions

**🔹 Limitations**

❌ Very **small dataset** (only dummy examples) → poor generalization  
❌ Accuracy stuck at ~50% (like random guessing)  
❌ No advanced NLP techniques (e.g., word embeddings, BERT)  
❌ Class imbalance not addressed

**🔹 Output:-**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**#Logistic Regression**

**A screenshot of a computer

AI-generated content may be incorrect.**

**#Randomforest**

**A screenshot of a computer

AI-generated content may be incorrect.**

**#decisiontree classifier**

**A screen shot of a computer

AI-generated content may be incorrect.**

**#predictions**

**A computer screen shot of a program

AI-generated content may be incorrect.**