BIO PROJECT

Symptom and Disease Relationship:

- Symptoms are physical or behavioral changes reported by individuals that indicate potential illnesses.
- Diseases often have overlapping symptoms, making it challenging to diagnose accurately without deeper knowledge.
  Medical diagnostics is
  Using AI, the aim is to automate or assist in this process by analyzing patterns in symptom-disease data.

Usally we have the text data
Converting this text into a structured form (like numerical vectors) is essential for analysis and prediction using machine learning.
Feature Extraction :
vectorization
TF-IDF (Term Frequency-Inverse Document Frequency) captures how important a word is in the dataset by considering how frequently it appears across multiple symptom descriptions.
TF-IDF is a statistical measure used to evaluate how important a word is to a document in a collection

TF(t,d)=Total number of terms in document d/Number of times term t appears in document d

Inverse Document Frequency (IDF):

- Measures how unique or rare a word is across all documents.
- IDF(t)=log(Total Number of documents/ number of documents containing term t)

tf * *idf

- **High TF-IDF Score**: A word is important for a specific document (high TF) but not common across all documents (high IDF).
- **Low TF-IDF Score**: A word is either too common across the corpus or irrelevant in the document.

after this traing and test data is divided
------>

Building and Training Models :

KNN :
KNN is a simple, instance-based algorithm.
It predicts the output for a new data point by finding its `K` closest neighbors in the training data based on distance (e.g., Euclidean distance)
The predicted class is typically determined by majority voting among the neighbors.
it also makes no assumptions

Application in Symptom Matching:

- Symptoms of a disease are often similar across cases.
- KNN works well here because it finds the most similar symptom descriptions (neighbors) in the dataset to classify a new symptom input.

drawbacks :
for large data and sensitive to the noise data
Each tree is trained on a randomly chosen **subset of data (rows)** and a **subset of features (columns)**.
RF :

that combines multiple **decision trees**.
Each decision tree is trained on a random subset of the data and features.
Diseases are often associated with complex combinations of symptoms.
Random Forest can measure the importance of features, helping identify which symptoms are most significant for predicting diseases

Model optimization :

grid optimization - a method for hyperparameter tuning to improve model performance
Grid Search systematically tests combinations of hyperparameters
Evaluate the model on all possible combinations of these values using cross-validation.
Machine learning models like Random Forest can perform suboptimally with default hyperparameters. that s why we use

Application to Random Forest:
these are the hyper parameters : They control the model's behavior and can significantly affect its performance.

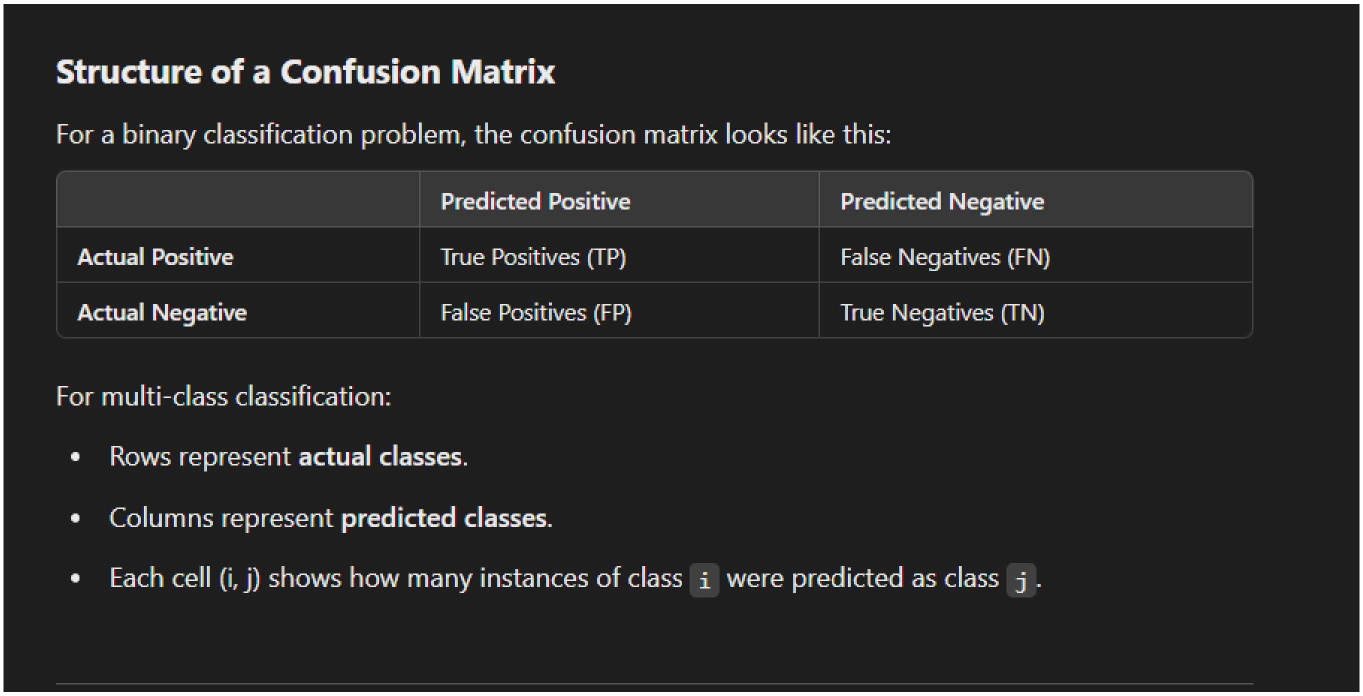
- Example:
  - Number of Trees (`n_estimators`): More trees generally improve performance but increase computation.
  - Maximum Depth of Trees (`max_depth`): Controls how deep each tree grows; deeper trees capture more
  KNN serves as a straightforward method, while Random Forest, optimized with Grid

Search,is the solution .
This workflow combines biology and AI to provide a system for predicting diseases, offering a scalable solution for healthcare diagnostics.

Precision:

- Definition: **(True Positives) / (True Positives + False Positives)**.
  Recall:
- Definition: **(True Positives) / (True Positives + False Negatives)**.
- Meaning: The ability of the model to correctly identify all instances of a disease.
  1-Score:
- Definition: Harmonic mean of precision and recall: **(2 × Precision × Recall) / (Precision + Recall)**
  Accuracy:
- Definition: **(Correct Predictions) / (Total Predictions)**.
  -**Support**:
- Definition: The number of actual instances for each disease in the dataset.

Confusion Matrix :
A **confusion matrix** is a table used to evaluate the performance of a classification model. It summarizes the model's predictions compared to the actual labels in the dataset. For multi-class problems like this one, the confusion matrix includes multiple rows and columns, one for each class.



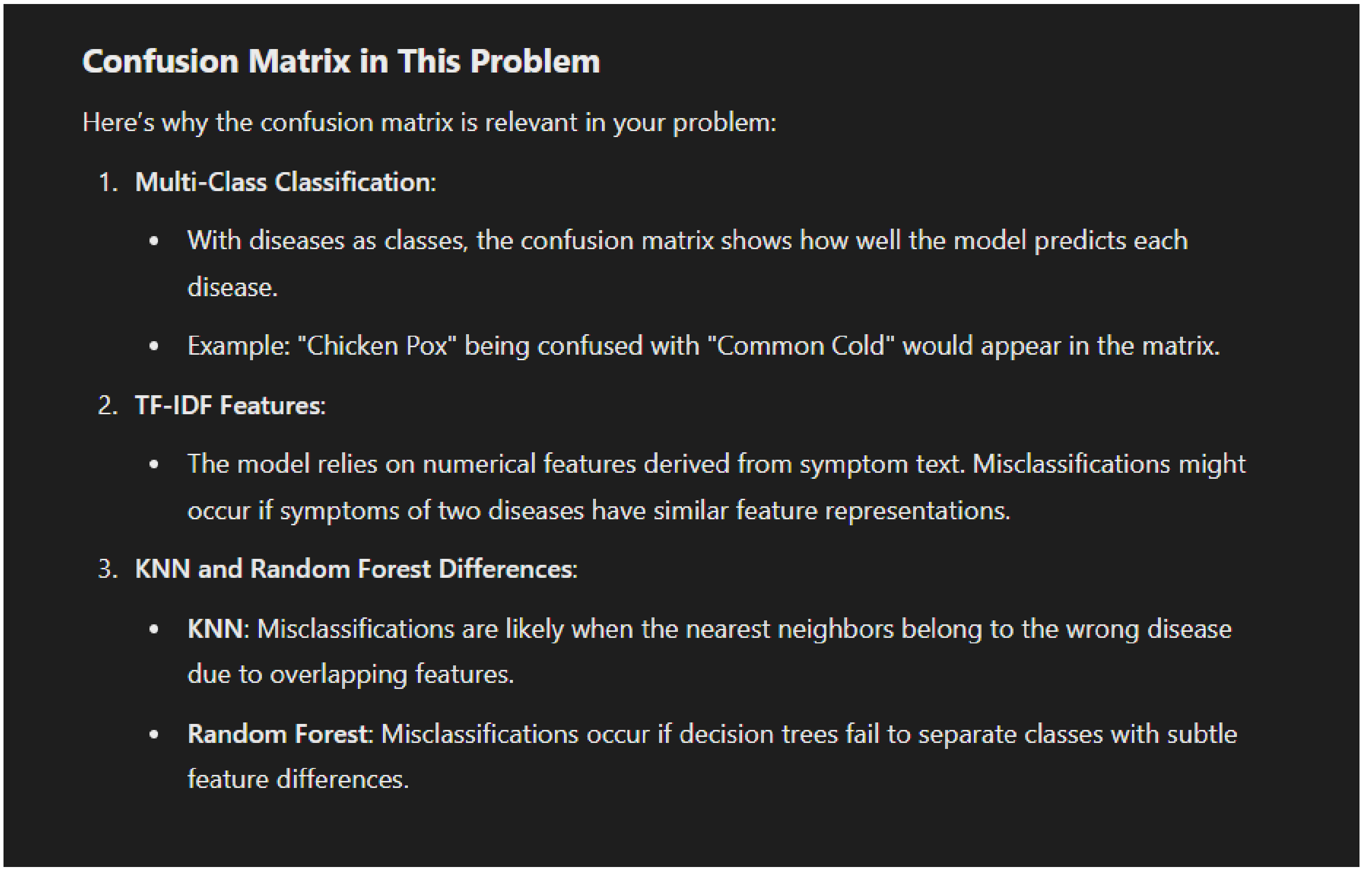
Structure of a Confusion Matrix

For a binary classification problem, the confusion matrix looks like this:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positives (TP)	False Negatives (FN)
Actual Negative	False Positives (FP)	True Negatives (TN)

For multi-class classification:

- Rows represent **actual classes**.
- Columns represent **predicted classes**.
- Each cell (i, j) shows how many instances of class `i` were predicted as class `j`.

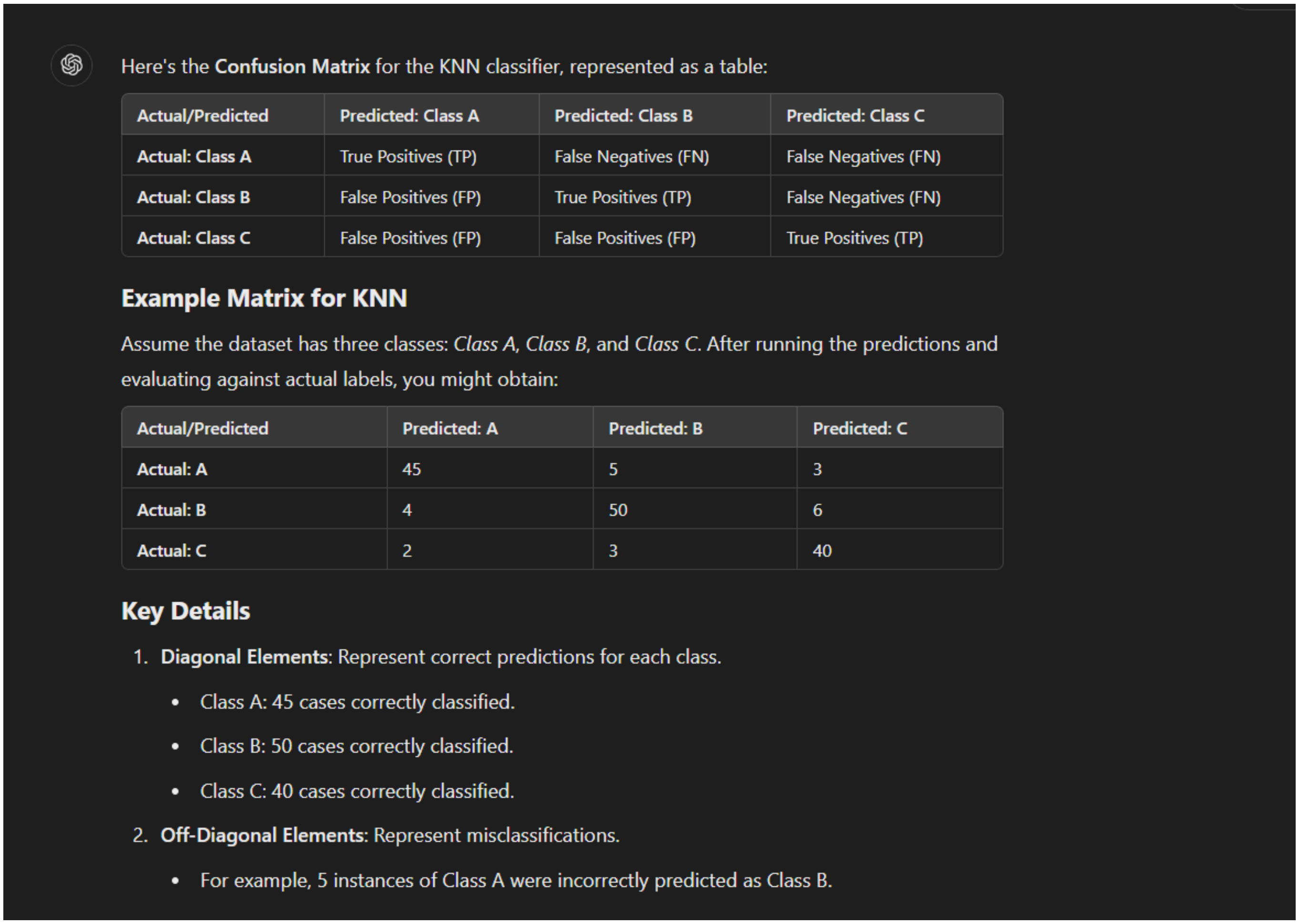


Confusion Matrix in This Problem

Here’s why the confusion matrix is relevant in your problem:

1. **Multi-Class Classification**:
    - With diseases as classes, the confusion matrix shows how well the model predicts each disease.
    - Example: "Chicken Pox" being confused with "Common Cold" would appear in the matrix.
2. **TF-IDF Features**:
    - The model relies on numerical features derived from symptom text. Misclassifications might occur if symptoms of two diseases have similar feature representations.
3. **KNN and Random Forest Differences**:
    - **KNN**: Misclassifications are likely when the nearest neighbors belong to the wrong disease due to overlapping features.
    - **Random Forest**: Misclassifications occur if decision trees fail to separate classes with subtle feature differences.

---->



Here's the **Confusion Matrix** for the KNN classifier, represented as a table:

Actual/Predicted	Predicted: Class A	Predicted: Class B	Predicted: Class C
Actual: Class A	True Positives (TP)	False Negatives (FN)	False Negatives (FN)
Actual: Class B	False Positives (FP)	True Positives (TP)	False Negatives (FN)
Actual: Class C	False Positives (FP)	False Positives (FP)	True Positives (TP)

Example Matrix for KNN

Assume the dataset has three classes: *Class A*, *Class B*, and *Class C*. After running the predictions and evaluating against actual labels, you might obtain:

Actual/Predicted	Predicted: A	Predicted: B	Predicted: C
Actual: A	45	5	3
Actual: B	4	50	6
Actual: C	2	3	40

Key Details

1. **Diagonal Elements**: Represent correct predictions for each class.
   - Class A: 45 cases correctly classified.
   - Class B: 50 cases correctly classified.
   - Class C: 40 cases correctly classified.
2. **Off-Diagonal Elements:** Represent misclassifications.
   - For example, 5 instances of Class A were incorrectly predicted as Class B.