Name: Viet Tran
Date: 10/31/2025

Monkey Pox Analysis Report

An ongoing outbreak of monkeypox, a viral disease, was confirmed in May 2022. The initial cluster of cases was found in the United Kingdom, where the first case was detected in London on 6 May 2022 in a patient with a recent travel history from Nigeria. We will use a dataset generated based on a study published recently in the BMJ (Patel et al., 2022) to build three classification models and assess their effectiveness in the medical setting. The dataset contains records of approximately 25,000 patients with a set of clinically relevant features and a target variable indicating if the patient has monkeypox or not. The features included in the dataset are:

1. Patient ID: the unique identifier of each patient record
2. Systemic Illness (categorical): the other illness that the patient has when being diagnosed
3. Rectal Pain (binary)
4. Sore Throat (binary)
5. Penile Oedema (binary)
6. Oral Lesions (binary)
7. Solitary Lesion (binary)
8. Swollen Tonsils (binary)
9. HIV Infection (binary)
10. Sexually Transmitted Infection (binary)
11. MonkeyPox (binary): output class class

1. Data Observation & Preprocessing

The dataset contains 24,875 samples, including 15,825 positive cases and 9,050 negative cases (Figure 1). It has nine variables (excluding the Patient ID), of which eight are binary and one is categorical (Systemic Illness) with four classes. There is an imbalance between the two classes, which may favor positive predictions. However, given the large dataset size, this imbalance is unlikely to have a major effect, so no data balancing method is applied.

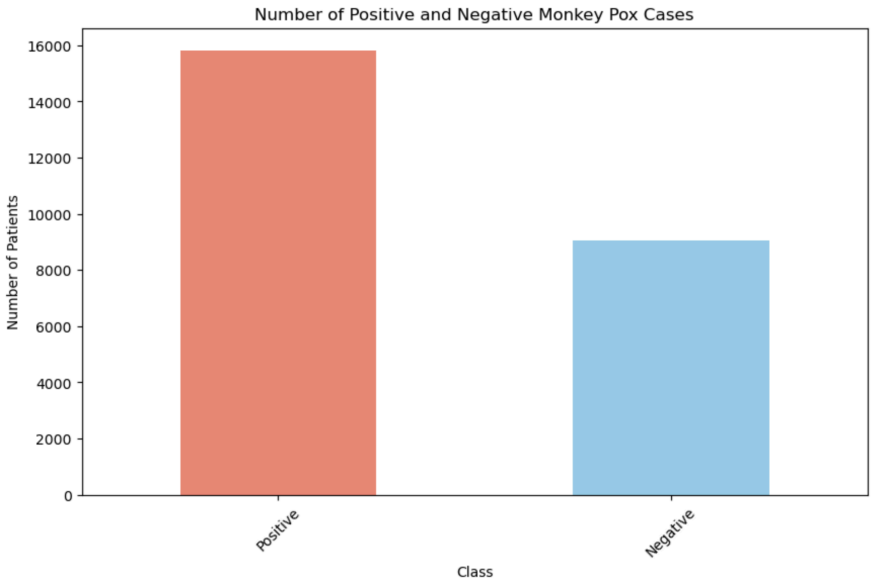

Figure 1. Comparison between Number of Positive and Negative Cases in the Dataset

We apply one-hot encoding to the Systemic Illness variable to convert it into binary features. No normalization is performed since all variables are binary (0 or 1). The Patient ID attribute was excluded from the analysis because it functions merely as a unique identifier for each patient and carries no meaningful relationship with the target variable. Keeping it could introduce noise or cause the model to memorize rather than generalize.

2. Model Training

The dataset was divided into 80% for training and 20% for testing to evaluate model performance. While accuracy provides an overall measure of correctness, in medical context, recall is more important because it reflects the model's ability to identify all positive cases. Missing a positive case may not only lead to an undiagnosed patient but also risk spreading the disease in the community even more. Precision is also emphasized, as it indicates the proportion of predicted positive cases that are truly positive, thereby representing the cost or risk associated with false alarms and unnecessary diagnostic procedures. Therefore, the evaluation focuses not only on accuracy but also on recall, precision, and F1 score to balance the detection of true positive cases against the cost of incorrect positive predictions.

kNN Classifier

Voting Schemes	**Metrics**	**kNN (k = 3)**	**kNN (k = 5)**	**kNN (k = 7)**	**kNN (k = 9)**	**kNN (k = 11)**
Uniform	Recall	74.66%	77.84%	79.76%	80.39%	**80.91%**
	Precision	70.11%	71.00%	70.63%	**71.18%**	71.04%
	F1 Score	72.32%	74.26%	74.92%	75.50%	**75.66%**
	Accuracy	63.04%	65.11%	65.47%	66.27%	**66.33%**

Weighted	Recall	74.66%	77.84%	79.76%	80.39%	**80.91%**
	Precision	70.11%	71.00%	70.63%	**71.18%**	71.04%
	F1 Score	72.32%	74.26%	74.92%	75.50%	**75.66%**
	Accuracy	63.04%	65.11%	65.47%	66.27%	**66.33%**

Table 1. Performance comparison between different number of neighbors and voting schemes for the kNN model

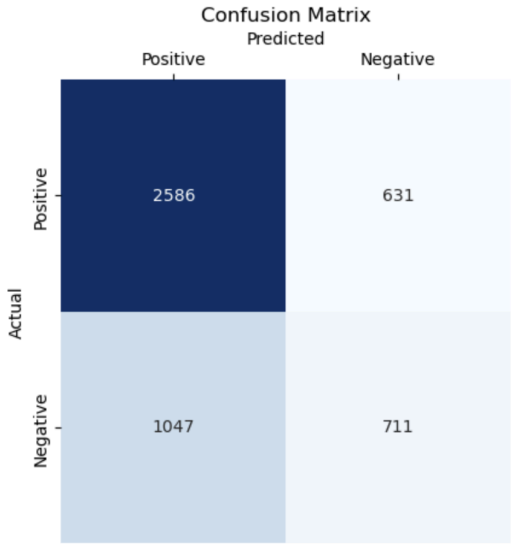

Figure 2. Confusion matrix of kNN with k = 9

Since all features in the dataset are binary, we use the Hamming distance metric for the k-Nearest Neighbors (kNN) model. We fine-tune the number of neighbors k by evaluating values from the set {3, 5, 7, 9, 11} to select the best-performing model. Additionally, we assess model performance using both uniform (unweighted) and distance-weighted voting schemes.

The results show no difference in model performance between uniform and distance-weighted voting schemes (Table 1). This can be explained by the fact that all the features we use are binary. Hence, with Hamming distance, distances are discrete, and the nearest neighbors are often tied or dominated by a single class, so weighting by distance does not substantially alter the predictions. Overall, performance improves as the number of neighbors increases, but increasing k from 9 to 11 does not yield a significant improvement compared to the previous increment. Therefore, we select k = 9 as the optimal number of neighbors for kNN.

Naive Bayes Classifier

Metrics	**Naive Bayes**
Recall	83.46%
Precision	72.82%

F1 Score	77.78%
Accuracy	69.17%

Table 2. Naive Bayes Model's Performance

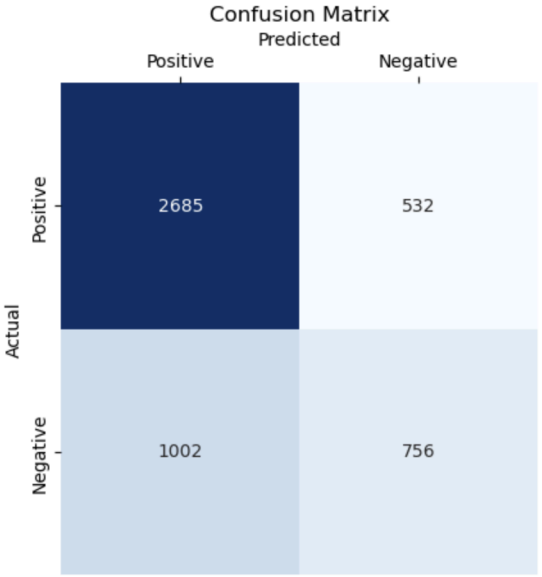

Figure 3. Naive Bayes Model's Confusion Matrix

For Naive Bayes Classifier (NB), we use Bernoulli Naive Bayes model from Python since all the features are binary. The result shows a better performance in all metrics compared to kNN (k = 9).

Decision Tree Classifier

Metrics	**Decision Tree**
Recall	82.25%
Precision	71.40%
F1 Score	76.44%
Accuracy	67.22%

Table 3. Decision Tree Model's Performance

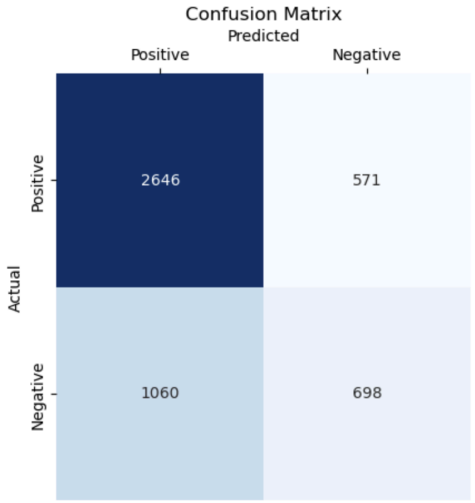

Figure 4. Decision Tree Model's Confusion Matrix

For the Decision Tree Classifier (DT), we do not adjust any configuration. The result also shows a better performance in all metrics compared to kNN (k = 9), but slightly worse than Naive Bayes model.

K-fold Cross Validation

Evaluation Method	**Metrics**	**kNN (k = 9)**	**NB**	**DT**
80-20 train test split	Recall	80.39%	83.46%	82.25%
	Precision	71.18%	72.82%	71.40%
	F1 Score	75.50%	77.78%	76.44%
	Accuracy	66.27%	69.17%	67.22%
5-fold cross validation	Recall	81.34%	84.66%	84.46%
	Precision	70.03%	71.49%	70.16%
	F1 Score	75.26%	77.52%	76.65%
	Accuracy	65.98%	68.76%	67.26%

Table 4. Model Performance Comparison Between Train Test Split and 5-fold Cross Validation

We then perform k-fold cross validation among three models with 5 folds. Table 4 shows a generally better performance when we compare to the 80-20 train test split evaluation method. The precision and overall accuracy is slightly decreased (except for DT's accuracy). However, we can observe an increase in recall which is positive under this medical context in all three models.

3. ROC - AUC Analysis

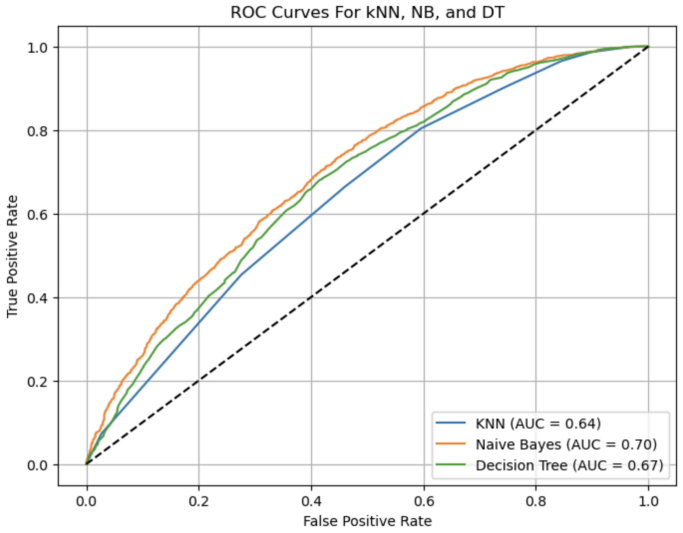

Figure 5. ROC Curve for kNN (blue), NB (orange), and DT (green)

All three ROC curves indicate that the models perform better than random guessing (Figure 5). Naive Bayes consistently shows the highest sensitivity compared to the other two models (Figure 5). Decision Tree performs slightly better than kNN, particularly at moderate false positive rate (FPR) levels (Figure 5).

Overall, Naive Bayes achieves the highest AUC (0.70), followed by Decision Tree (AUC = 0.67) and kNN (AUC = 0.64) (Figure 5). The relatively poor performance of kNN can be attributed to the binary and categorical nature of the dataset, which limits the meaningfulness of distance-based calculations. Therefore, the best classifier for this task is Bernoulli Naive Bayes, as it is specifically suited for binary features. Nonetheless, the overall performance of all models is not outstanding to be used on actual patients. Hence, we can try feature selection to enhance the performance.

4. Feature Selection

Filter Technique - Information Gain

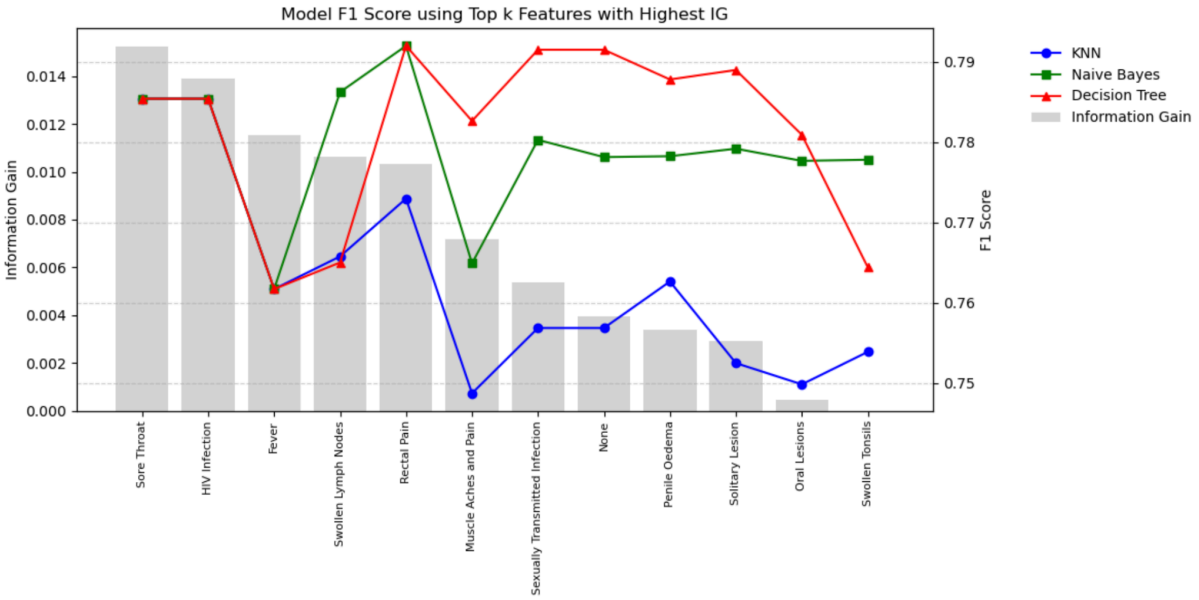

Figure 6. The Information Gain by Each Feature in Decreasing Order and Corresponding F1 Score Performance Using That Feature Subset of kNN, NB, and DT

First, we use information gain (IG) to rank the most discriminating features and select the best subset for each model. To determine the optimal number of features, we plot the F1 score of each model as an evaluation metric of the top k features. F1 score is used because it balances recall and precision, preventing the model from being biased toward predicting all cases as positive when too few features are included. The results indicate that selecting the top 5 features ranked by IG (Sore Throat, HIV Infection, Fever, Swollen Lymph Nodes, and Rectal Pain) yields the best performance across all three models.

Wrapper Technique - Floating Backward Selection

Models	Number of features chosen	Names of features chosen
kNN (k = 9)	4	Rectal Pain, Oral Lesions, HIV Infection, Muscle Aches and Pain
NB	10	Rectal Pain, Sore Throat, Penile Oedema, Oral Lesions, Solitary Lesion, Swollen Tonsils, HIV Infection, Sexually Transmitted Infection, Muscle Aches and Pain, None
DT	11	Rectal Pain, Sore Throat, Oral Lesions, Solitary Lesion, Swollen Tonsils, HIV Infection, Sexually Transmitted Infection, Fever, Muscle Aches and Pain, None, Swollen Lymph Nodes

Table 5. The Number and Names of Features Chosen for kNN, NB, and DT Using FBS

For wrapper technique, we choose the floating backward selection (FBS) method from mlxtend library (similar to BestFirst in Weka). Unlike simple backward selection, this method allows us to trace back the previously chosen features to re-evaluate and trim the features that might adversely affect the performance. The FBS is also less likely to miss features that are only useful in combination compared to forward selection. We choose F1 score similarly to the filter technique as the evaluation metric.

Compared to the filter-based approach, Naive Bayes and Decision Tree selected a significantly larger number of features, whereas kNN required one fewer feature to achieve optimal performance (Table 5). Notably, among the features chosen by kNN, only half overlap with the top five features that yielded the highest information gain. The other features (Oral Lesions and Muscle Aches and Pain) have relatively low IG values. This could be because certain combinations of features may provide additional discriminatory power that is not evident when each feature is evaluated independently. Both Naive Bayes and Decision Tree, however, selected nearly the entire set of features to achieve their best performance.

5. Evaluation on New Feature Subsets

Model	**Metrics**	**No Feature Selection**	**IG-based Selection**	**FBS-based Selection**
kNN (k = 9)	Recall	80.39%	74.01%	**96.33%**
	Precision	**71.18%**	66.92%	66.53%
	F1 Score	75.50%	70.29%	**78.7%**
	Accuracy	66.27%	59.54%	**66.29%**
NB	Recall	83.46%	88.09%	**90.24%**
	Precision	**72.82%**	70.06%	71.26%
	F1 Score	77.78%	78.05%	**79.63%**
	Accuracy	69.17%	67.96%	**70.15%**
DT	Recall	82.25%	**88.09%**	**88.09%**
	Precision	**71.40%**	70.06%	71.06%
	F1 Score	76.44%	78.05%	**78.67%**
	Accuracy	67.22%	67.96%	**69.11%**

Table 6. Comparison between kNN, NB, and DT Performance for Different Feature Selection Methods

The results show a general improvement in model performance when using feature subsets selected by the Floating Backward Selection (FBS) method (Table 6). In particular, recall increases substantially across all models: kNN improves by 16%, NB by 7%, and DT by 6%,

compared to using all features (Table 6). Although precision slightly decreases, the overall balance between recall and precision, as reflected in the F1 score, improves. Accuracy also increases marginally. These results are expected since the FBS procedure uses the F1 score as its optimization criterion, which inherently prioritizes a balance between recall and precision.

In contrast, the Information Gain (IG)-based feature selection method does not lead to any significant improvement. In fact, most performance metrics are lower than those obtained without feature selection. This can be explained by the fact that IG evaluates each feature independently and therefore fails to capture correlations among attributes. For instance, the presence of swollen lymph nodes may be associated with other symptoms such as muscle aches, fever, or sexually transmitted infections. These relationships are typically overlooked by filter techniques.

Hyperparameters	**Values**
Max depth of the tree	3, 5, 7, or None
Minimum samples required to split the node	2, 5, 10
Minimum samples required at a leaf node	1, 2, 5

Table 7. Hyperparameter set used for tuning DT model

Hyperparameters	**Values**
Additive smoothing parameter (alpha)	0.001, 0.01, 0.1, 0.5, 1.0

Table 8. Hyperparameter set used for tuning NB model

Metrics	**DT**		**NB**	
	Before	**After**	**Before**	**After**
Recall	88.09%	**88.87%**	90.24%	90.24%
Precision	71.06%	**71.12%**	71.26%	71.26%
F1 Score	78.67%	**79.01%**	79.63%	79.63%
Accuracy	69.11%	**69.47%**	70.15%	70.15%

Table 9. Performances of DT and NB after Fine Tuning

To further enhance performance, we fine-tuned the NB and DT models using the feature subsets identified by the FBS method. The hyperparameter optimization was conducted using Python's GridSearchCV. In total, 36 combinations of hyperparameters were evaluated for DT and 5 for NB (see Table 7 and Table 8). For the DT model, the best parameter configuration was max_depth = 7, min_samples_leaf = 1, and min_samples_split = 2. For NB, the optimal setting was alpha = 0.5. However, the fine-tuning did not yield a notable improvement. NB performance remained unchanged, and DT showed only marginal gains.

6. Dimension Reduction Using PCA

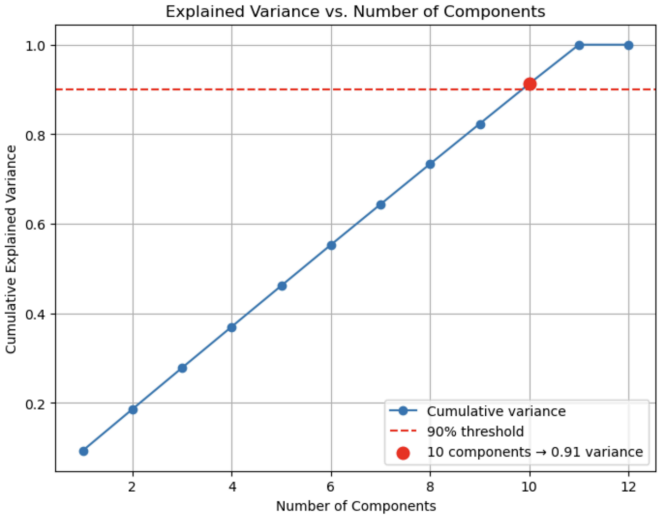

Figure 7. The total variance explained by the number of components

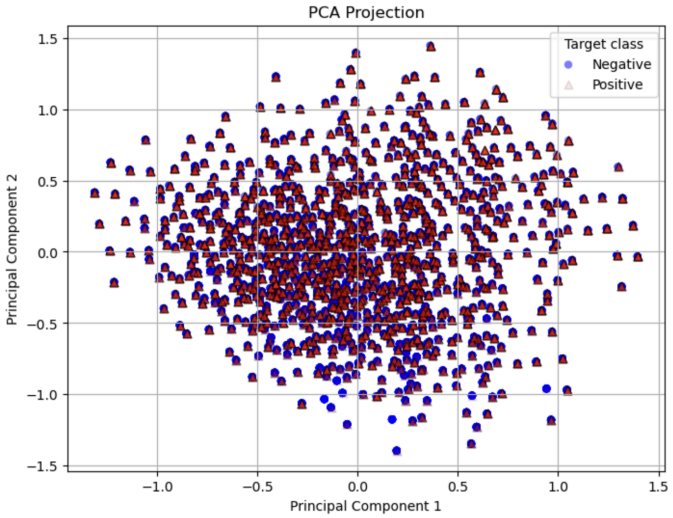

Figure 8. Class separation after reducing to 2 components

Metrics	On PCA-transformed data	On normal data
Recall	**89.31%**	83.46%
Precision	**69.26%**	72.82%
F1 Score	**78.02%**	77.78%
Accuracy	**67.46%**	69.17%
AUC	**0.67**	0.70

Table 10. Naive Bayes Model Performance After Training on PCA-transformed Data Compared to Training on Normal Data.

After applying PCA to the dataset, at least ten principal components were required to explain 90% of the total variance (Figure 7). When projected onto the first two components, the

classes remained highly overlapped, indicating that PCA did not produce clear visual separation between positive and negative samples (Figure 8). The NB model, trained on the PCA-transformed data, achieved a noticeably higher recall (an increase of approximately 6%) compared to the model trained on the original features. However, precision decreased by about 4%, and overall accuracy dropped by roughly 2%. Since recall is prioritized in this medical context, assuming false negatives are costlier than false positives, we can accept this trade-off.

After PCA, the NB model exhibited higher sensitivity but lower discriminative power, as reflected in a lower AUC. This suggests that dimensionality reduction enhanced the model's ability to identify true positive cases but compromised its ability to clearly distinguish between classes. Because PCA converts binary variables into continuous linear combinations, some class-specific patterns were likely smoothed out, resulting in weaker overall ranking performance despite better recall.

While PCA can reduce noise and redundancy, the transformed components are abstract linear mixtures of the original features, making it difficult to attribute model behavior to specific clinical variables. Although PCA can sometimes simplify computation and reveal latent structures, in this case the minimal gain in recall and loss of transparency suggest that dimensionality reduction offers limited practical benefit. For medical decision support, where explainability is essential, retaining the original binary attributes may be preferable despite the slightly lower recall. Additionally, feature selection methods like the wrapper technique could produce a better performance while maintaining the interpretability of the model. Hence, using PCA in this case is not necessary.

Reference

Patel, A., et al. “Clinical features and novel presentations of human monkeypox in a central London centre during the 2022 outbreak: descriptive case series”. British Medical Journal, vol. 378, 2022.