# Report on Model Performance and Analysis

## 1. Overview of Findings

Through experimentation with multiple classification algorithms, the XGBoost Classifier emerged as the most effective, achieving an accuracy of **0.95963**. Below, I detail the insights gained from exploring other models and the improvements made along the way.

## 2. KNN (K-Nearest Neighbors) Performance

### Initial Approach

- Initially, KNN was tested with default parameters (`n_neighbors=5`, `weights=uniform`).
- The model struggled with accuracy due to its sensitivity to data scaling and distance metrics.

### Improvements

- Applied Min-Max Scaling to standardize the features, which reduced the bias towards features with higher magnitudes.
- Experimented with hyperparameters:
    - Increasing `n_neighbors` to balance the model's sensitivity to noise.
    - Changing `weights` to `distance`, which helped prioritize closer neighbors in decision-making.
    - Tested various distance metrics (e.g., Euclidean, Manhattan) to find the optimal fit.

### Result

- The score improved incrementally but plateaued at an accuracy of **~0.87**. This was likely due to KNN's limitations in handling high-dimensional data and noisy features.

## 3. Random Forest Classifier Performance

### Initial Approach

- Random Forest performed moderately well from the start, with an accuracy of **~0.91**, leveraging its ability to handle non-linear relationships and feature interactions.

### Improvements

- Increased `n_estimators` (number of trees) to 200, which stabilized predictions.
- Tuned `max_depth` to control overfitting:
    - Shallow trees reduced complexity but impacted accuracy.
    - Deeper trees slightly overfitted but improved the score.
- Applied feature selection based on importance scores generated by Random Forest:
    - Removed less significant features (e.g., features with low variance or high correlation).
    - This reduced noise and improved performance.

**Result**

- Final accuracy: **~0.93**, though Random Forest struggled to capture subtle patterns in the data compared to XGBoost.

## 4. XGBoost Classifier Performance

**Why XGBoost Succeeded**

- **Handling Missing Values:** XGBoost automatically deals with missing data, unlike many other algorithms.
- **Feature Importance:** XGBoost provided insights into the most critical features, allowing for targeted feature selection.
- **Hyperparameter Optimization:**
    - Tuned `learning_rate` (0.004), `max_depth` (3), and `n_estimators` (11000).
    - Added regularization parameters (`lambda` and `alpha`) to prevent overfitting.
- **Boosting Technique:** Incrementally corrected errors from weak learners, making it robust.

**Result**

- Final accuracy: **0.95963**, with strong generalization and robust predictions.

## 5. Impact of Data Transformation Techniques

**Column Filtering**

- Used correlation heatmaps to identify and remove highly correlated features.
- Improved Random Forest and XGBoost performance but had minimal effect on KNN.

**Scaling**

- Standardized features for models sensitive to feature magnitudes (e.g., KNN, Logistic Regression). This was critical for KNN's improvement.

**Feature Engineering**

- Introduced polynomial and interaction features to capture complex relationships. This benefitted Random Forest and XGBoost but increased overfitting in Logistic Regression.

## 6. Key Challenges

**Challenge 1: Feature Selection**

- **Problem:** Some features were redundant or irrelevant, introducing noise.
- **Solution:** Used feature importance from Random Forest and XGBoost to filter less useful features.

**Challenge 2: Overfitting**

- **Problem:** Deep decision trees in Random Forest and XGBoost occasionally overfitted.
- **Solution:** Added regularization techniques and limited `max_depth`.

**Challenge 3: Computational Cost**

- **Problem:** XGBoost's training time increased with hyperparameter tuning.
- **Solution:** Reduced `n_estimators` and used early stopping to halt training when no improvement was observed.

**Challenge 4: Imbalanced Dataset**

- **Problem:** Class imbalance affected model predictions.
- **Solution:** Applied SMOTE (Synthetic Minority Oversampling Technique) to balance the dataset, improving model fairness.

## 8. Conclusion

- **Best Model:** XGBoost Classifier stood out with an accuracy of **0.95963**, owing to its robust learning mechanism and effective handling of feature importance and missing values.
- **Key Learnings:**
    - Preprocessing steps (scaling, feature selection) are crucial for some models (e.g., KNN).
    - Advanced models like XGBoost thrive on hyperparameter tuning and feature engineering.
    - Simpler models like Logistic Regression can serve as good baselines but fail with complex, non-linear data.

This iterative experimentation highlights the importance of understanding the strengths and limitations of each algorithm to achieve optimal results.