Decision Tree Model Report

# 1. Model Architecture

## 1.1 Dataset Description

The dataset used in this report is for predicting pet adoption speed, which is a multiclass classification problem.  
 The target variable is 'AdoptionSpeed' with 5 categories (0 to 4).   
 The features include categorical features such as 'Breed1', 'Breed2', 'Color1', 'Color2', 'Color3', and others.   
 There are numerical features such as 'Age', 'Fee', 'PhotoAmt', etc.   
 Target encoding was used for categorical features such as 'Breed1\_encoded', 'Color1\_encoded', etc.

## 1.2 Decision Tree Model

The Decision Tree Classifier used in this report is a classification model that works by creating a tree structure.  
 The model splits the feature space recursively to make predictions based on the training data. Each internal node represents   
 a decision based on a feature, and each leaf node corresponds to a predicted class. The hyperparameters include 'max\_depth',   
 'min\_samples\_split', and 'min\_samples\_leaf', which help control the complexity and prevent overfitting.  
 The model's performance depends on the depth of the tree and how it splits the data.

# 2. Evaluation Metrics

## 2.1 Accuracy

Accuracy is the proportion of correctly predicted samples out of the total number of samples. It is a commonly used metric   
 for classification problems, but can be misleading in the case of imbalanced datasets.

## 2.2 Confusion Matrix

The confusion matrix is a tool used to evaluate the performance of a classification model by comparing the true   
 labels with the predicted labels. It provides a more detailed view of the model’s accuracy for each class.

## 2.3 Precision, Recall, F1-Score

Precision is the proportion of positive predictions that are correct. Recall is the proportion of actual positives   
 that are correctly identified by the model. F1-score is the harmonic mean of precision and recall, providing a single metric   
 to evaluate both aspects. These are especially important when dealing with imbalanced data.

## 2.4 Macro and Weighted Average

Macro average computes the metric independently for each class and then takes the average, treating all classes equally.   
 Weighted average takes the support (the number of true instances for each class) into account when averaging the metrics.

# 3. Results and Analysis

## 3.1 Confusion Matrix

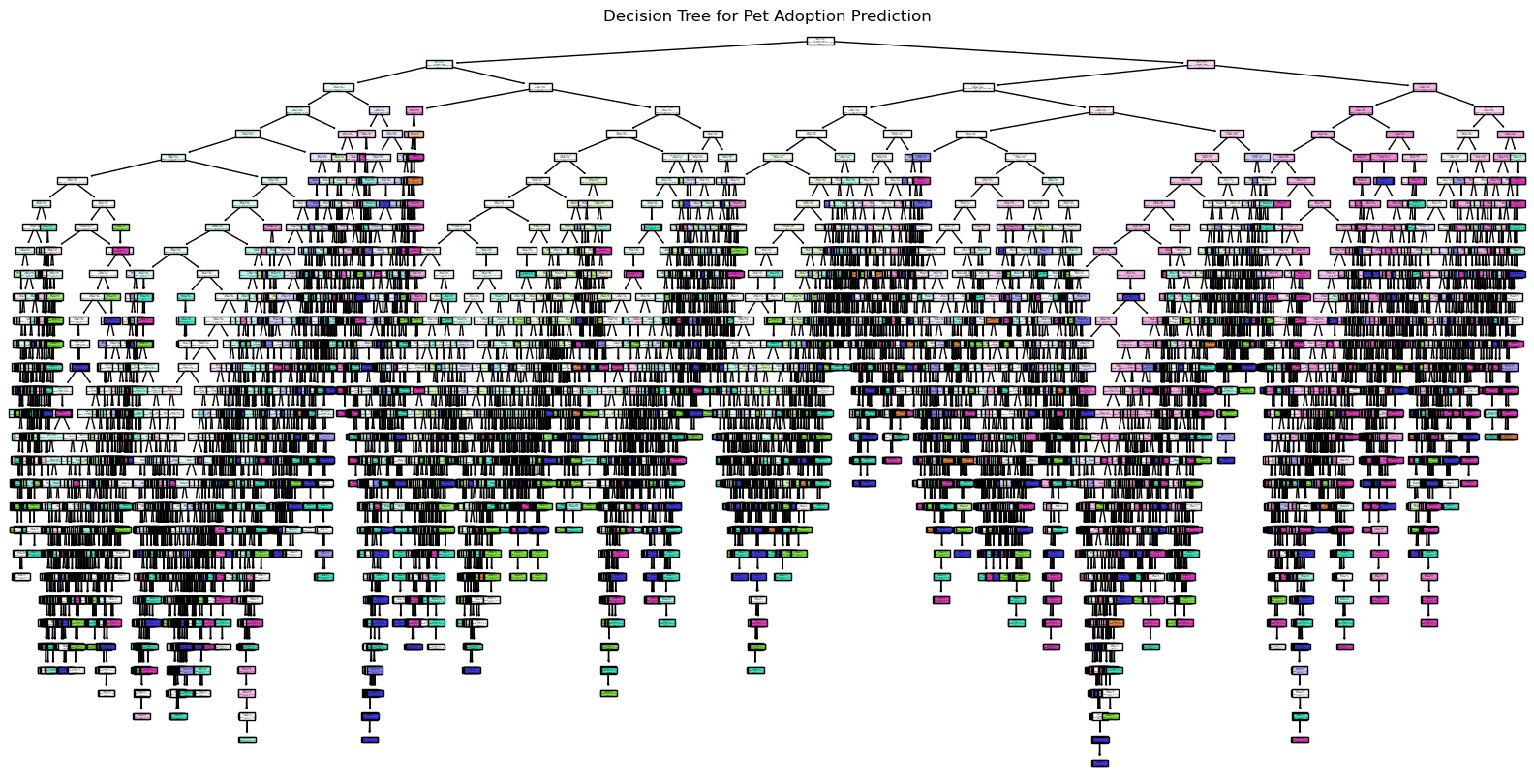
The confusion matrix for the decision tree model is as follows:  
   
 [[ 6 21 16 9 15]  
 [ 18 154 129 86 89]  
 [ 15 117 200 123 113]  
 [ 9 95 135 160 97]  
 [ 15 71 118 125 239]]  
   
 - Class 0 has a poor recognition performance, often misclassified as class 1 and 2.  
 - Class 4, the most frequent class, is predicted more accurately but still has significant confusion with class 3.  
 - Class 2 and 3 show relatively stable predictions, although errors are still prevalent.

## 3.2 Classification Report

The classification report is as follows:  
   
 precision recall f1-score support  
  
 0.0 0.10 0.09 0.09 67  
 1.0 0.34 0.32 0.33 476  
 2.0 0.33 0.35 0.34 568  
 3.0 0.32 0.32 0.32 496  
 4.0 0.43 0.42 0.43 568  
  
 accuracy 0.35 2175  
 macro avg 0.30 0.30 0.30 2175  
 weighted avg 0.35 0.35 0.35 2175  
   
 - Class 0 has low precision and recall, which indicates poor model performance for rare classes.  
 - Class 4 shows higher precision and recall, demonstrating the model’s better ability to predict more frequent classes.  
 - Overall accuracy is 35%, which is a starting point but indicates room for improvement, especially for rare classes.

## 3.3 Results Analysis

The model suffers from class imbalance, especially for class 0, where the number of samples is much lower than other classes.  
 The model tends to predict the majority class more frequently, leading to poor performance on the minority classes.   
 It is recommended to use techniques such as class balancing (e.g., SMOTE or class\_weight='balanced') to address this issue.



## 3.4 Conclusion

The Decision Tree model serves as a solid starting point, but there is significant room for improvement.   
 It is recommended to explore alternative models such as Random Forests or XGBoost, which may provide better performance for this task.  
 Addressing class imbalance and tuning hyperparameters will likely lead to an improved model.