TECH 27 - Final Project Submission “Advanced ML for Dog Breed Categorization with Multi-Model Deployment for Robust Real-World Applications”

Progress	0
Status	Done

1. Project Title:

“Advanced ML for Dog Breed Categorization with Multi-Model Deployment for Robust Real-World Applications”

2. Solo Final Project:

By Kiris (Kulkunya Prayarach)

(Contribution Share: Human Intelligence – 70%, AI/ML – 30%)

👴 Human Responsibilities (strategy, oversight, and accountability)

Humans remain the **decision-makers and accountability holders**, leading with skills that require judgment, context, and critical insight. Core responsibilities include:

- **Design & Organization**: structuring workflows, frameworks, and processes.
- **Selection & Validation**: ensuring data and outputs are relevant, accurate, and trustworthy.
- **Critical Thinking & Fact-Finding**: interpreting results, identifying gaps, and verifying information.
- **Oversight & Governance**: approval, adjustment, correction, and ethical calibration of AI outcomes.
- **Decision-Making & Accountability**: holding final responsibility for results, strategies, and impact.
- **Prompt Engineering & Monitoring**: framing precise instructions and continuously verifying system performance.

🤖 AI Delegated Tasks (efficiency, automation, and support)

AI serves as a **supporting assistant**, executing tasks that are repetitive, mechanical, or computationally intensive, always under human guidance. Delegated roles include:

- **Automation of Routine Work**: handling general operational or labor-intensive tasks.
- **Data & File Processing**: unzipping files, preprocessing data, and managing inputs.
- **Drafting & Execution**: generating initial coding scripts, running programs, and adjusting analysis per human queries.
- **Content Support**: producing draft images, correcting grammar, and checking typos.

👤✨🤖 Together : create a balanced ecosystem where human judgment directs AI precision.

3. Intuitions and AI/ML Curiosities:

“In the 21st century, AI and Machine Learning (AI/ML) have emerged as transformative forces, redefining how humanity thinks, learns, predicts, and engages with the world. The synergy between human intelligence and AI/ML is not merely additive. It is multiplicative, unlocking unprecedented potential to transform complex ideas into practical, real-world solutions.“

While quantitative analysis remains a core pillar of data science, the growing importance of qualitative analysis, particularly in image recognition, has captured the attention of AI and data science experts worldwide. From everyday applications such as facial authentication and bioscan, to mission-critical domains like cybersecurity, autonomous vehicles, advanced healthcare diagnostics, 3D imaging, and next-generation biometric systems, the demand for accurate, reliable, and ethical models is accelerating.

In addressing these multi-dimensional challenges, I have applied advanced AI/ML methodologies and cutting-edge tools to create an innovative dog-breed recognition system. Far from a niche use case, this solution serves as a foundational steppingstone for broader image and data analysis applications requiring high precision and dependability.

4. Project Milestones and End Goals:

By fusing deep technical expertise with AI-driven intelligence, this project seeks to deliver timely, mission-critical insights that advance the development of supervised machine learning models across a wide range of algorithms. The work emphasizes accuracy, performance evaluation, hyperparameter optimization, and cross-validation to ensure the resulting architectures are not only robust and well-structured but also establish a solid foundation for applying AI/ML to the complex qualitative data.

Beyond demonstrating technical mastery, this initiative establishes a practical framework for addressing real-world challenges with measurable impact, while paving the way for future advancements through deeper, smarter, faster, and more powerful algorithms. It bridges the gap from science fiction to tangible human progress, enabling breakthroughs in areas such as bioscan metrics, cancer detection, cybersecurity protection, and the emerging era of AGI.

🎯 END GOALS:

- **Best Model Identification** — Data-driven evaluation to determine the most effective models
- **Deployment-Ready Outputs** — Optimized and saved models prepared for real-world deployment
- **Interactive Interface** — User-friendly tools for model selection, testing, and experimentation
- **Innovative Prediction Capabilities** — Robust performance, including handling unseen or out-of-sample data
- **Responsible AI Features** — Models capable of responding with “unknown/unsure” for ambiguous inputs
- **Comprehensive Documentation** — Full performance analysis with clear rationale and transparent reporting

5. Core Thinking with Mind Map Visualization:

This project develops a comprehensive machine learning pipeline for dog breed prediction from images, with a focus on evaluating model generalization to completely unseen breeds—such as Shiba Inu (used as the unseen sample). The objective is to advance model accuracy, reliability, and responsible AI practices.”

🔗 Harnessing AI/ML for Conceptual Mind Maps and Workflow Storytelling

👉 click https://concise-mind-canvas-91.lovable.app/

attachment:19cdcb35-70f3-47e0-9540-91dfcd4addbc:TECH_27_mindmap_explanation.mp4

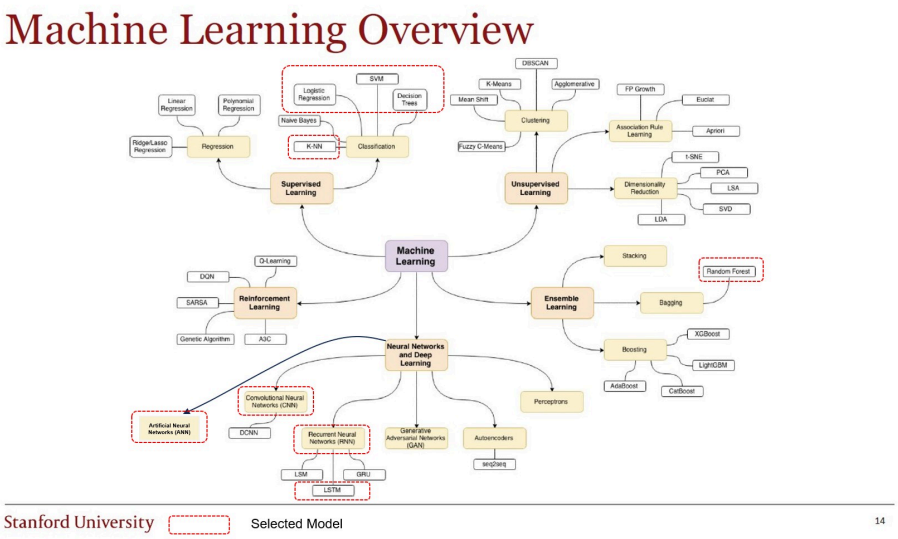
6. AI/ML Milestones for Full Stake End-to-End Predictive Dog Breed Categorization

🐕 6.1 Methodologies

Built and deployed Multi-Algorithm Implementation for Dog Breed Categorizations from basic supervised learning → gearing ensemble learning → stepping to Neural nets/Deep learning for 9 core models or 11-sub modulars including

- 4-traditioanl classification models : K-NN, Logistic Regression, SVM, Decision tree
- 1- Ensemble Learning model : Random Forest
- 4- Neural Networks and Deep learning models : ANN, RNN, CNN (with sub-3 modules standard CNN, simplified advanced CNN, optional label smoothing CNN), and LSTM

Figure 1: Selected ML algorithms for dog breed project



Source: TECH 27 Lecture

🤖 6.2 Feature Engineering Strategies

I. 📻 Image Preprocessing

- **Resize & Normalize**
  - feed 1,200 random samples and Normalize pixel values to `[0,1]` or `[-1,1]` (depending on the model).
- **Cosmetic Plots**
  - Used Dark Theme but maintained white background for image print
  - Convert to grayscale if feeding into classical ML to reduce dimensionality.
- **Noise Reduction**
  - Apply Gaussian blur or denoising filters to handle low-quality images.

II. 🦮 Core Machine Learning Tools

- **Basic Python Tools**
  - Used numpy, matplotlib, PIL, scipy, random, pickle, pandas, seaborn
- **Scikit-learn (sklearn)**
  - Used for classical ML/Ensemble models including **Logistic Regression, K-NN, Random Forest, SVM**, **DP**
  - Provided the `predict_proba`, `decision_function`, and cross-validation utilities.
- **Neural Network Frameworks** (depending on dataset & session)
  - **Keras / TensorFlow** (for CNN, ANN, RNN, LSTM when working on deep image models).
  - For the Kaggle dog dataset, some sessions were PCA + sklearn models, others CNNs.

III. 📍 Comprehensive Calibration Utilities, Cross Validations, and Hyperparameter

- Used **calibratedclassiferCV** to all models
- Deployed Hyperparameters for logistic regression model
- Plot Calibrated accuracy and cross-validation scores for all model
- Plot DT Improvement using **Grid Search**, Top 10 Feature Importance, and DT performance vs. Max Depth for comparing basic DT vs. improved DT accuracy
- Created Hyperparameter for logistic Regression since the model's accuracy is outperformed the other models by using calibrated variant
- Custom **temperature scaling** function to improve probability calibration for Logistic Regression.

IV. 🚨 Setup Advanced Date Augmentation for challenging NNs and DP models

- Setup advanced data augmentation/ Create Data Generator for training (CNN model)
- Using advanced techniques by flips, crops, color jitter, rotation)
- Create data generators for training

V. 🎇 Conduct Comprehensive data pre-processing and preparation

☐ Load Dataset A & B from Stanford dog data and Kaggle Dog Breeds and upload to google drive and shared drive

☐ Pre-process data by readin data and Upziping files from google drive in both folders and save all images into main folders "Stanford Dog" and "Kaggle Dog Breeds" and keep sharing drive for working on the next step

☐ Normalize image pixel values to [0,1]

☐ Shuffle training data

☐ Random selected 1,200 images (1,000 train/200 test)

☐ Create validation split dataset for train/test 80-20 for both dataset

☐ Match labels with images for both dataset

☐ Display 5 sample images from the dataset using matplotlib.pyplot.imshow

7. 📂 Two Datasets:

- **Dataset A 🦮 (Stanford Dog Data):**

👉 http://vision.stanford.edu/aditya86/ImageNetDogs/

- **Dataset B 🦮 (Kaggle Dog Breeds):**

👉 https://www.kaggle.com/code/dansbecker/exercise-intro-to-dl-for-computer-vision/input

8. 🎯 Random Selected 1,200 samples (1,000/train, 200 test) covering entire 120 breeds:

- **Database A** : Random shuffle selected 1,000 /train and 2,000/test covering entire 120 breeds (From Total 44,000 images with 120 breeds)

Figure 2 : 5 image visualization with label and normalized image - Dataset A

- **Database B** : Random Shuffle selected 1,000/train and 2,000/test covering entire 120 breeds (From total 20,579 images 10,222 train + 10,357 test sets)

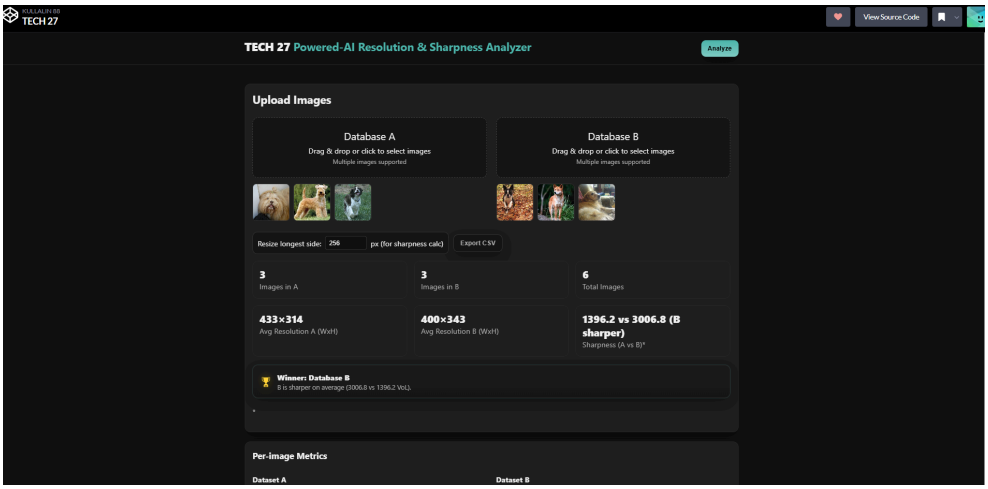
Figure 3 : 5 image visualization with label and normalized image - Dataset B

9. 💡 Innovated Image Resolution Detection- Dataset A & B

🔗 https://codepen.io/KULLALIN-88/full/PwPOpXB

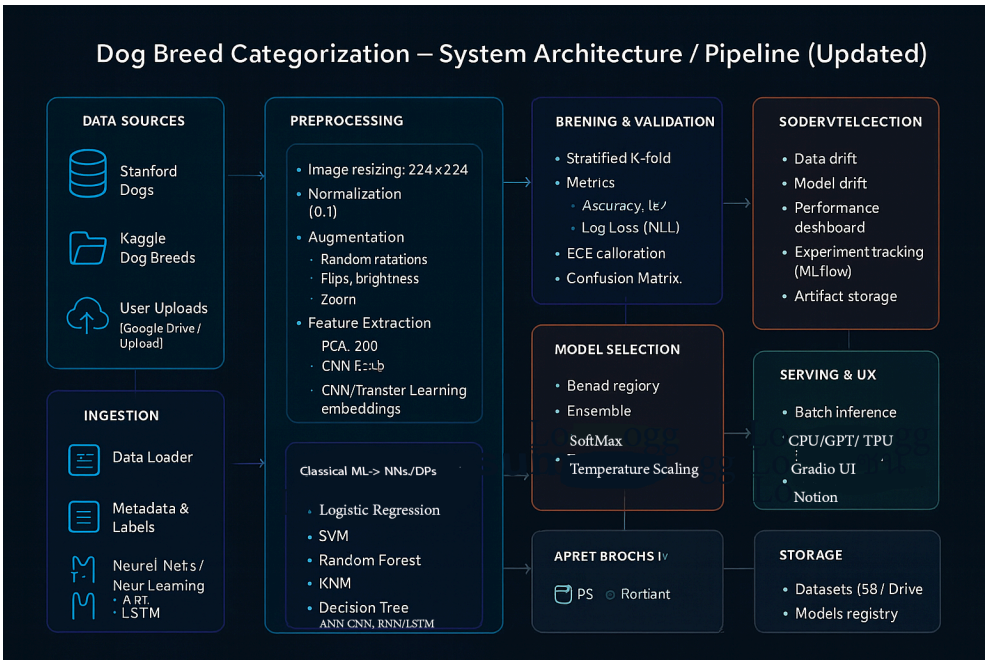
Innovative and adaptive thoughts in the Codepen.ai html deployment for Image resolution comparison for determining the quality and sharpness of dataset A vs. B in resolution scheme. The impressive outcomes confirm that “dataset B (Kaggle Dog Breeds) beats dataset A (Stanford dog) in Sharper and Higher resolution of image quality.

Figure 4: Image Resolution detection with html deployment



10. Block Diagrams & ML Architectures

Figure 5 : Dog Breed Block Diagram & System Architecture



Contribution share (50% Human, 50% AI)

🔷 10.1 Loading data to google drive and upzipped file images in goolgle drive and shared to colab and pre-processing data

- **Stanford Dogs, Kaggle Dog Breeds, and User Uploads** are the main inputs.

- These provide both *labeled datasets* (images with breed names) and *unlabeled images* for testing.
- Having multiple sources ensures diversity, which is crucial for generalization.

🔷 10.2 Ingestion Layer

- **Data Loader**: Handles importing images into the pipeline.
- **Metadata & Labels**: Reads associated breed names and structures the dataset.
- This stage ensures consistent file structures and mappings (image ↔ label).

🔷 10.3 Preprocessing

- **Image Resizing**: Standardize all images (e.g., 224×224 pixels).
- **Normalization**: Scale pixel values to [0,1] for stable training.
- **Data Augmentation**: Apply random rotations, flips, brightness changes, etc., to reduce overfitting.
- **Feature Extraction**:
  - PCA (dimensionality reduction).
  - CNN-based embeddings (transfer learning).

This step prepares the dataset so both ML and Deep Learning models can work effectively.

🔷 10.4 Modeling Layer

Classical ML Models

- Logistic Regression, SVM, Random Forest, K-NN, Decision Trees.
- Typically trained on **features extracted by PCA**

Neural Nets / Deep Learning Models

- **ANN**: Fully connected feedforward networks.
- **CNN**: Convolutional networks specialized for image features.
- **RNN / LSTM**: Sometimes used if sequence data or temporal augmentation is added (less common in static images).

This dual approach lets you compare simpler, explainable models vs. more powerful deep learning architectures.

🔷 10.5 Training & Validation

- **Stratified K-Fold Validation**: Ensures balanced class distribution in splits.
- **Metrics Used**: Accuracy, Top-5 Accuracy, Log Loss, ROC-AUC, ECE (Expected Calibration Error) and Brier score.
- **Calibration**: Important since probability estimates should reflect true uncertainty.

🔷10.6 Model Selection

- Compare performance across models by equally weighted performance indicators from metrics use in section 5.
- Ensemble or soft-voting methods may combine strengths of ML + DL.
- Temperature scaling may be applied for calibration.

🔷 10.7 Serving & Monitoring

- **Serving**: batch inference, CPU/GPU/TPU when it reaches max then downgraded to CPU
- **UX Layer**: Gradio UI interface for end users to upload images and see predicted breeds.
- **Monitoring**:

- Detect **data drift** (incoming images differ from training data).
- Detect **model drift** (performance degradation over time).
- Use MLflow or similar for experiment tracking and artifact storage.

🔷 10.8 Storage

- Store datasets (Google Drive, colab folder, github).
- Store trained models, logs, and metrics in a **model registry** for reproducibility.

🔷 10.9 Deployments

- Github, Colab, and Notion for front-end display

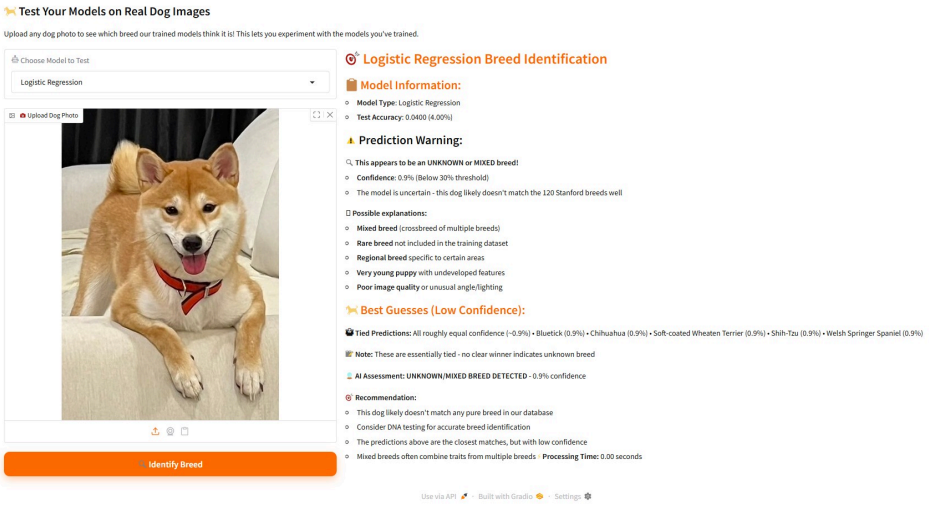
11. 📈 Render Gradio UI Interface for comprehensive visualization

- Use basic Plot tools but cosmetic by keep dark theme for professional setting
- Render Gradio with drop down for classification analysis including summary page, K-NN, Logistic, Logistic-temperature scaling (for hyperparameter), SVM, Random Forest, and DT
- Automated Basic model Recommendation by checking mode's accuracy across classification accuracies and report model calibration, cross validation, model improvement and best classification
- Advanced Interface for Upload images for predictive model with dropdown for model selection and display the analysis outcomes such as confidence, ranking breed prediction with probability/Chance, Limitation & Concerns for allowing Model to say "Unsure" or "Unknown"

12. 🎯 Implemented Live Prediction for Unseen image Prediction (Shiba Inu breed)

- Using training/testing dataset A and B and implemented Unseen breed from training/testing dataset to predict "Shiba inu"
- **Integrating responsible solution by adding control features in allowing models to say "unsure" with breed prediction if unknown data/image and say how the model is confident about "Unseen image of Japanese Shiba breed" - Innovative with responsive solution**

Figure 6: Innovative with responsive solution (Unseen Dog Breed Prediction)



13. 🔮 Feature Engineering models for Part 3 (Dataset A) and 4 (Dataset B) - Classification and Ensemble Algorithms

Major experimental conducted, Model Evaluation, Optimization, Calibration, Hyperparameter methods:

- Progressive steps from linear Logistic Regression classifier → K-NN → SVM → Decision Tree → Random Forest Analysis
- Train both datasets across various models
- Report Test Accuracy → Display Confusion Metrix → Annotate misclassifcations
- Executed Full range of Advanced Evaluation solutions beyond basic approach using log loss, convergence, Expected Calibration Error (ECE), Brier Score and Confusion Metrics
- Implemented wide range of Model Improvement, Cross Validation, and Hyperparameter analysis for solid outcomes using various approahces such as SoftMax prob (Logistic), K Search, (K-NN), GridSearch (SVM and DT),
- Render user-friendly interface: Gradio UI in comparison and Deployment
- Devops AI/ML meets practicality by deployment unseen dog breed prediction using classification + single ensemble learning
- Take a way Bias and Responsible Creation by integrating function to allow models for saying “Unsure” or “Unknown” in proper manners.

Model 1: Logistic Regression Classifier

- Train Logistic model by using train dataset A and dataset B
- Identify 2 digits where the classifier perform poorly
- Plot Confusion matrix and annotate misclassifications
- Evaluate the model on test set using using variety powerful tools such as Accuracy Score, Log loss, Expected Calibration Error (ECE) for calibration reliability like well-calibrated→ confidence matches accuracy, Brier Score for mean squared error (MSE) between predicted prob and actual outcomes, and Confusion Metric)

- Executed SoftMax Probability inspection by pinking 5 test samples that were misclassified by Logistic Regression → Print the true label and predicted label → print the full SoftMax Prob Vector

Model 2 : K-NN Model

- Train a K-Nearest Neighbors classification (with K = 5 for base line for both datasets A and B)
- Assigned solver “saga” and multiclass = “multinormial, n_jobs = -1 for linear logistic regression model
- Report Test Accuracy
- Improve model by using cross validation using optimizing K (K search)
- Create a bar chart to compare the test accuracies of all three/five classifiers

Model 3: SVM model

- Train both datasets with SVM using kernel ‘rbf’, prob = true, random_state = 42 seeds to fix training outcomes
- control parameters_grid by n_estimator such as [75, 100 150] → max_depth [10,20, None], and min sample_split such as [2,5]
- Estimate scoring accuracy
- Model Calibration in handling Cross validation using “sigmoid” method, cv = 3 and n_split such as = 3 and 2, and None respectively, and using CV-safe
- Plot and Print display accuracy

Model 4: Decision Tree model

- Train both datasets with DT
- Estimate Accuracy for DT
- Improved model by using GridSearch for CV improvement with max_depth such as [10,20,30, None] min_sample split such as [2,5,10] and min_sample_leaf such as [1,2,4]
- Plot and Print display accuracy

Model 5: Random Forest Architecture

- Train both datasets with RF for n_estimators = 100 on the same training set
- Deploy cross validation/Hyperparameter/Opitimized parameters using max_depth, min_samples,_split, n_esimator / Validation using GridSearchCV
- Plot and report test accuracy

14. 🔮 Feature Engineering models for Part 5 (Dataset A) and 6 (Dataset B) - Neural Nets/Deep Learning Algorithms

Major experimental conducted, Model Evaluation, Optimization, Calibration, Hyperparameter methods:

- Progressive Complexity from ANN → CNN → RNN/LSTM
- Implement Hidden Units Doubling by : Smart small, double for complexity, reduce for output
- Activation Strategy : ReLU (hidden) in general + Sigmoid and Tanh (LSTM gates) → SoftMax (output)

- Regularization Pyramid: Multiple techniques stacked for optimal performance
- Comprehensive Evaluation: Beyond Accuracy to include calibration and confidence
- Render user-friendly interface: Gradio UI in comparison and Deployment
- Devops AI/ML meets practicality by deployment unseen dog breed prediction using NNs and DP models
- Considered Bias and Responsible outcomes by integrating functions in allow modeling to say “Unsure” or “Unknown” under proper manners.

Model 6 : ANN Architecture

- Train Configuration for Both datasets
- Hidden layer strategy 128 → 256 → 128 (doubling then reducing)
- Activation Function: ReLU for hidden layer
- SoftMax for output Regularization: L1/L2 Penalties + Dropout (0.3)
- Batch Normalization Architecture: Input → Flatten → 5 Dense layers (base line) → Output

Model 7 : CNN Architectures with three sub modules (Standard CNN : optimized 3-block (50 epochs max), simplified advanced CNN: 1 residual block (40 epochs max), optimal label smoothing CNN: only if time permit (20 epochs)

- Setup advanced data augmentation (flips, crops, color jitter, rotation)
- Create data generators for training
- Reduced model Complexity: 3 conv blocks instead of 4
- Larger batch sizes : 64 instead of 32
- Applied Convergence with Higher learning rate: 2e-3 for faster convergence
- Reduced Epochs: 20 -50 instead of 100 for base line
- Aggressive early stopping: Patient 3-5
- Limit time consumption : skip if time > 20 minutes

Model 8: RNN model

- Convert 2D images to temporal sequence (16 time steps)
- Hidden Units: 128 → 256 → 128 doubling strategy
- Activation: ReLU for hidden states
- Regularization: Dropout + Recurrent Dropout + BatchNorm
- RNN with Label Smoothing

Model 9: LSTM Architecture

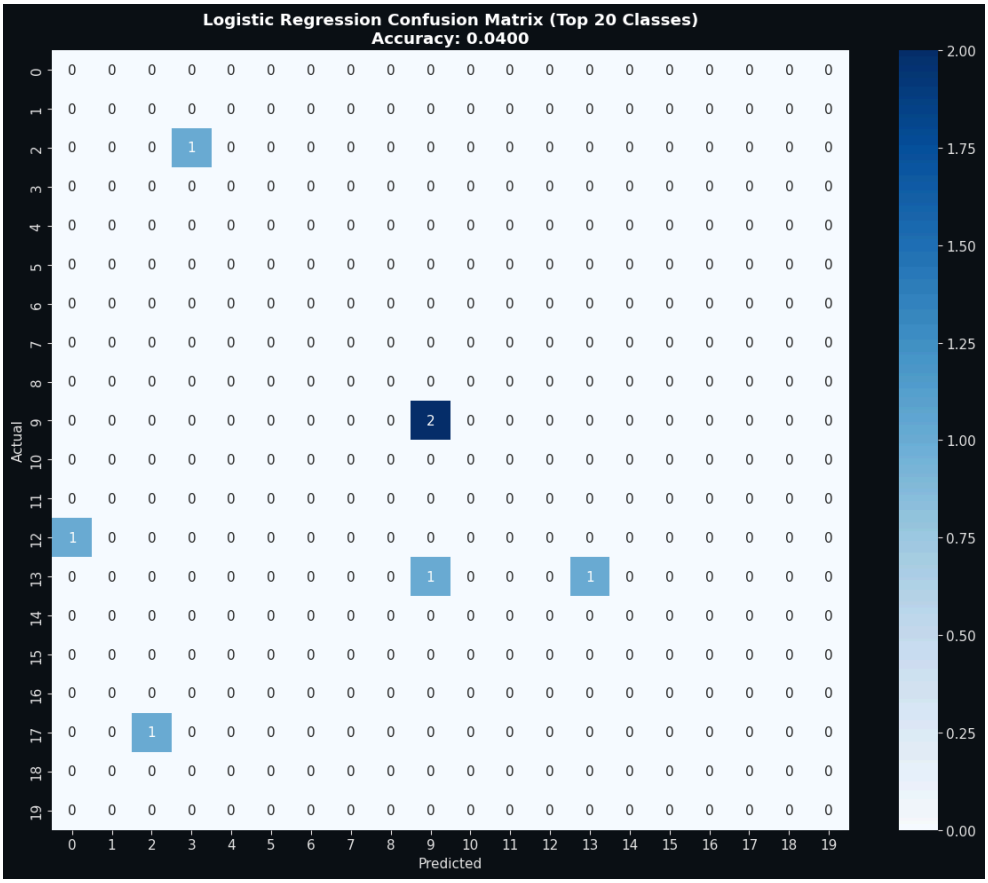
- Hidden Units: 128 → 256 → 128 doubling strategy
- Advanced Activation: Tanh (cell state) + Sigmoid (gates) per request
- Advanced Memory cells for long-term dependencies
- Sequential processing → Image →Temporal sequence conversion

- Gate Activation : Sigmoid + Tanh for LSTM as specifically requested
  - Recurrent regularization Dropout on both input and recurrent connections
  - Memory management LSTM handles long-term dependencies better than RNN

15. 🤖 Main Outcomes : Classification & Random Forest Learning prediction, Accuracy, additional model performance , Hyperparameter tunes, Cross Validation, and Unseen Dog Prediction - Data set A (Part 3 in Colab)

15.1 Confusion Matrix for Logistic Regression

Figure 7: Logistic Regression Confusion Matrix



15.2 Model Calibration, Cross Validation and Hyperparameter tuning

Figure 8: Logistic Regression (Temperature Scaled) with Calibration

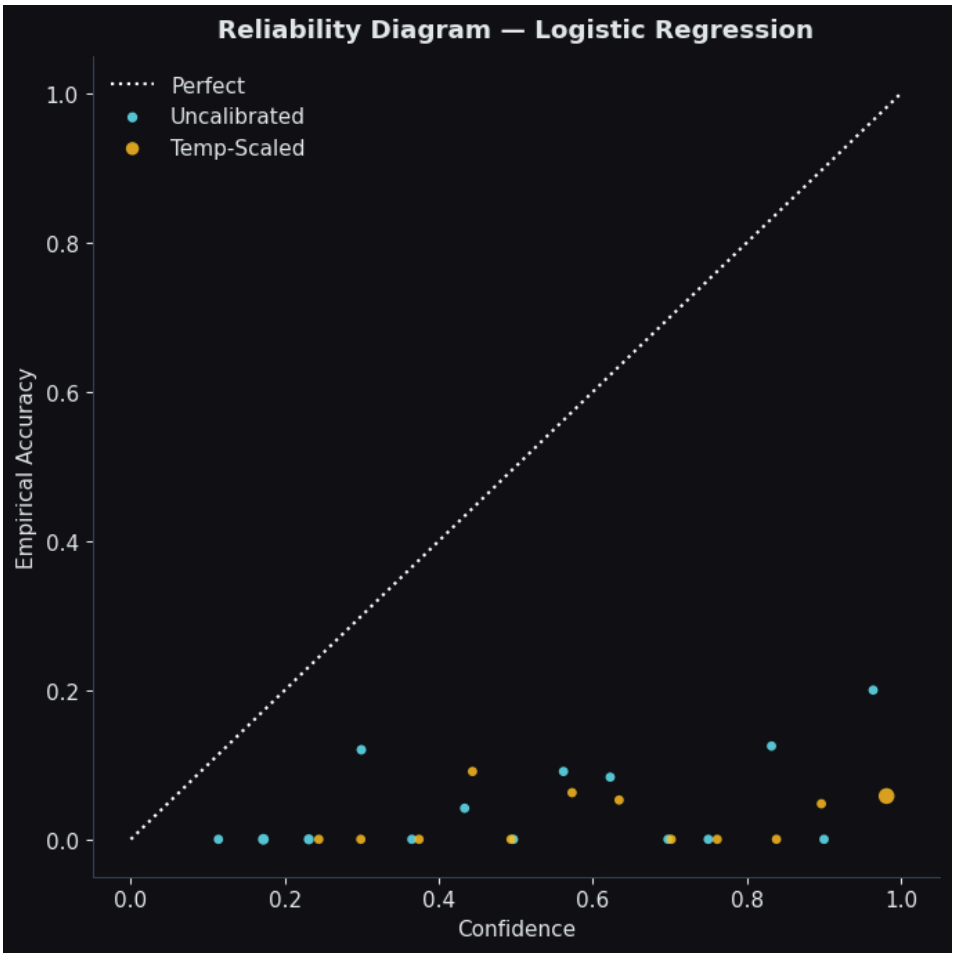


Figure 9: Logistic Regression Before vs After Calibration

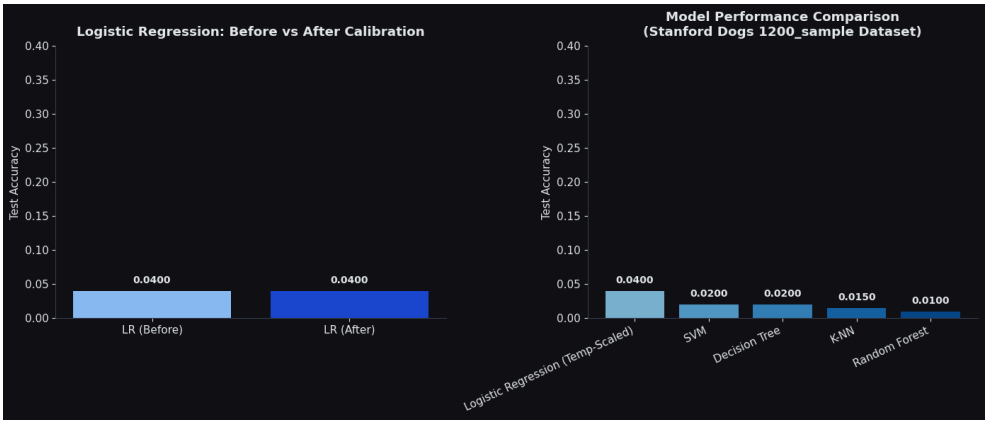


Figure 10: Random Forest Calibration

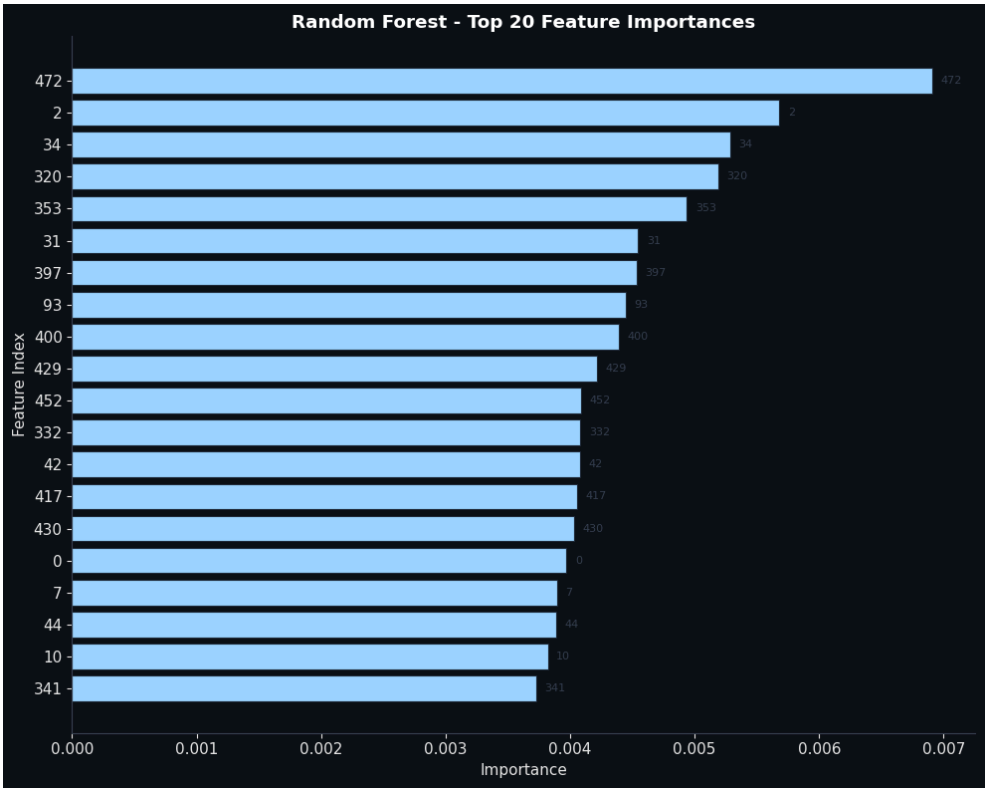


Figure 11: SVM Calibration

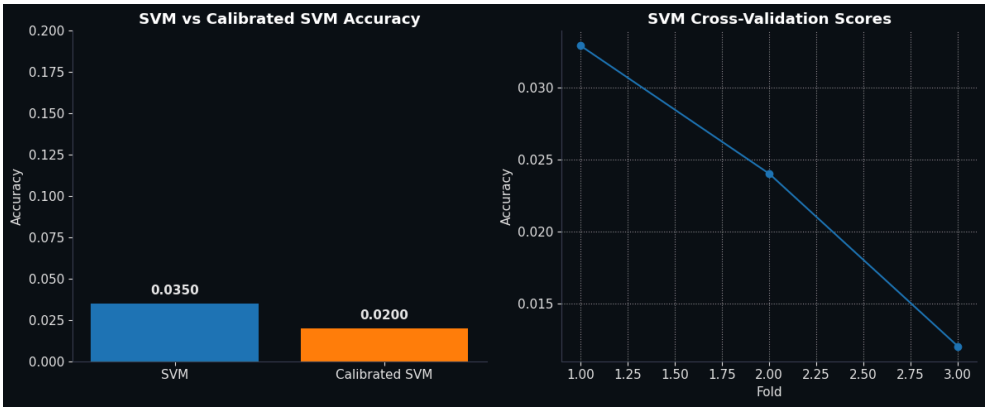
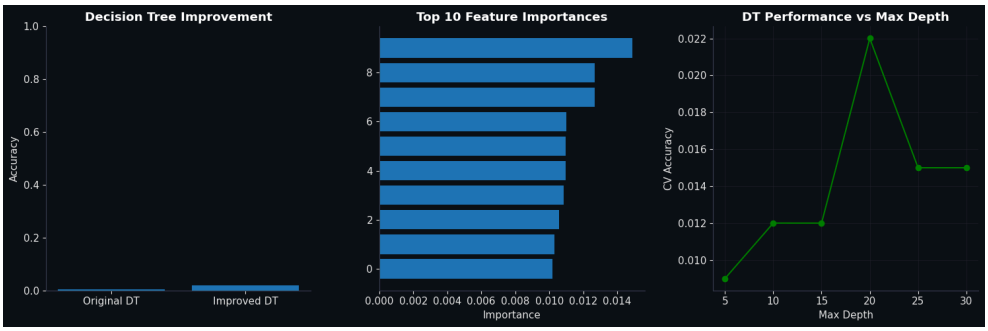
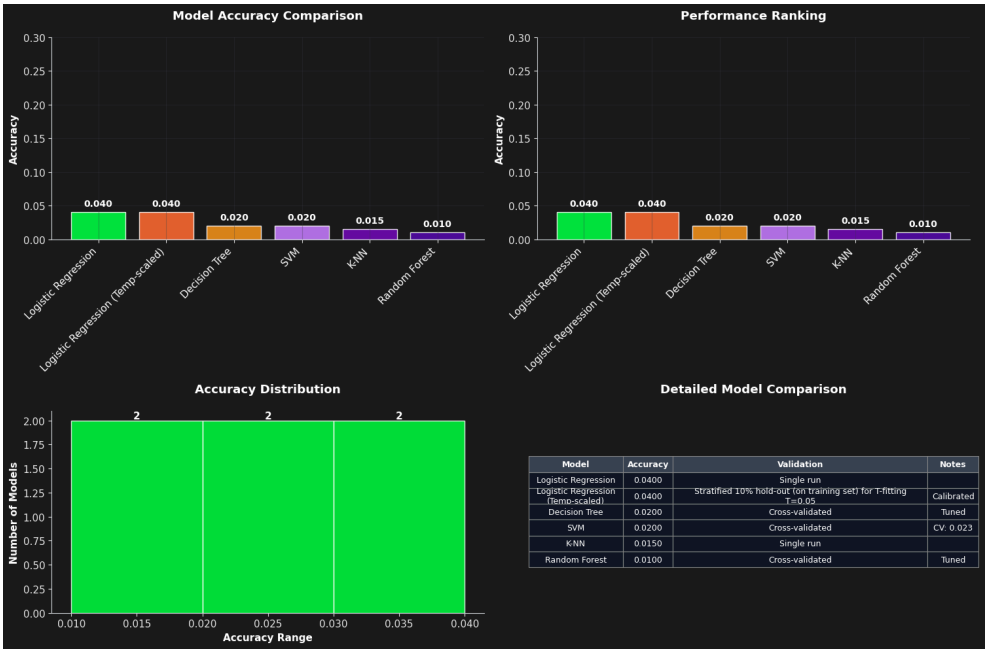


Figure 12: Decision Tree Calibration



15.3 **Model Performance Comparison and Detailed Validation methods**

Figure 13: Model Performance Metrics - Dataset A



Model	Accuracy	Validation	Notes
Logistic Regression	0.0400	Single run	
Logistic Regression (Temp-scaled)	0.0400	Stratified 10% hold-out (on training set) for T-fitting	Calibrated
Decision Tree	0.0200	Cross-validated	Tuned
SVM	0.0200	Cross-validated	CV. 0.023
KNN	0.0150	Single run	
Random Forest	0.0100	Cross-validated	Tuned

15.4 Gradio UI Visualization with Unseen 🐕 breed Prediction

Figure 14: Gradio UI for Model Analysis Summary pages

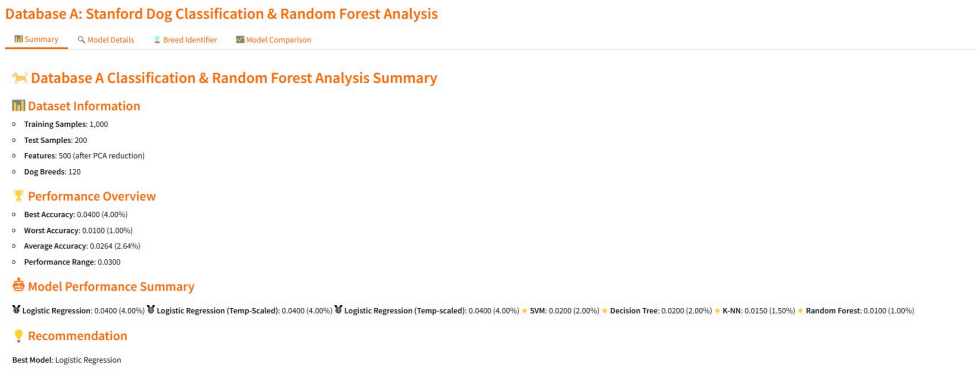


Figure 15: Gradio UI for Specific detailed Model Analysis with Dropdown

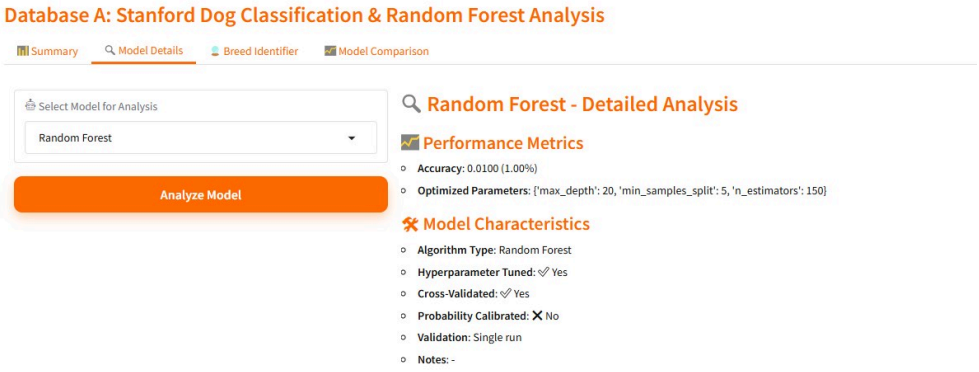
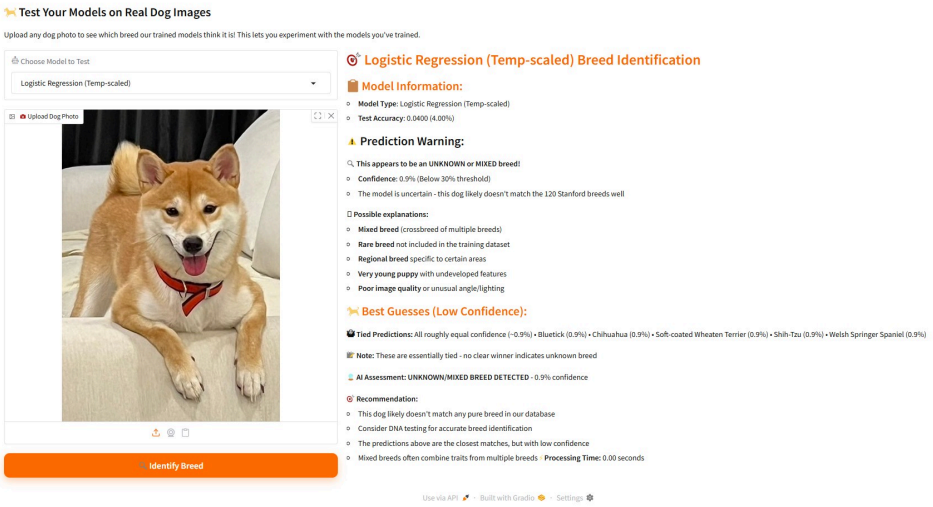


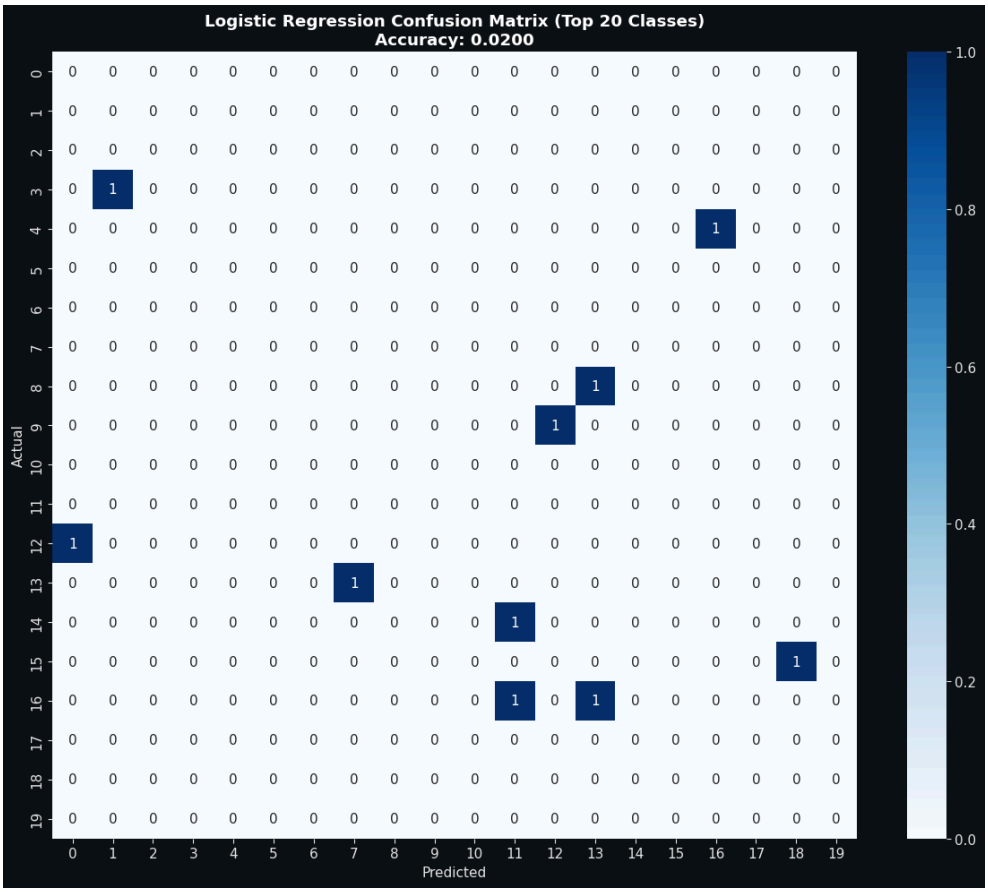
Figure 16: Gradio UI in Unseen Dog (Shiba inu) Prediction using Classification & RF



16. 🤖 Main Outcomes : Classification & Random Forest prediction, Accuracy, additional model performance , Hyperparameter tunes, Cross Validation, and Unseen Dog Prediction - Data set B (Part 4 in Colab)

16.1 Confusion Matrix for Logistic Regression

Figure 17: Logistic Regression Confusion Matrix -Dataset B



16.2 **Model Calibration, Cross Validation and Hyperparameter tuning**

Figure 18: **Logistic Regression with Temp Scaled -Dataset B**

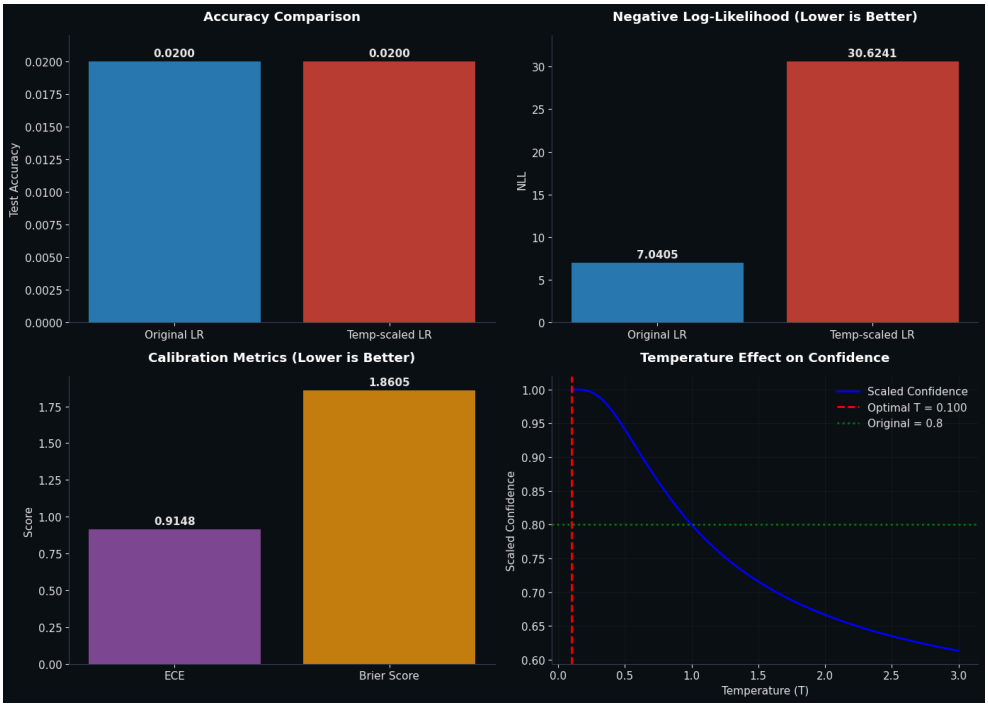


Figure 19: Random Forest Cross Validation- Dataset B

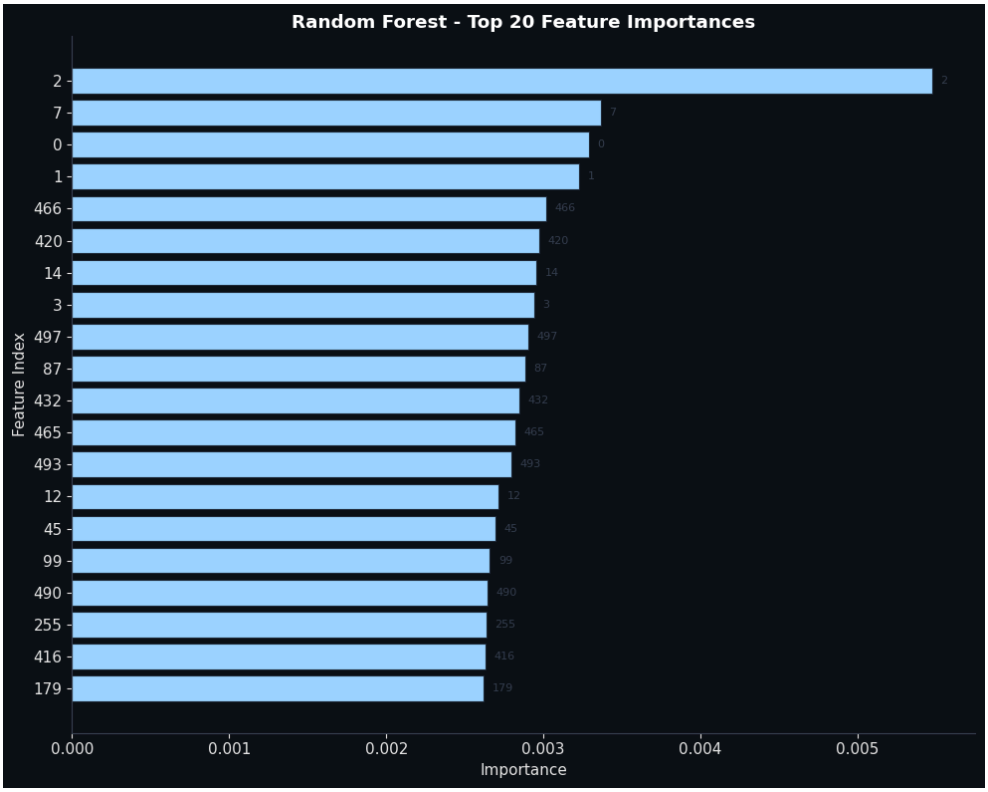


Figure 20: SVM Calibrated and Accuracy -Dataset B

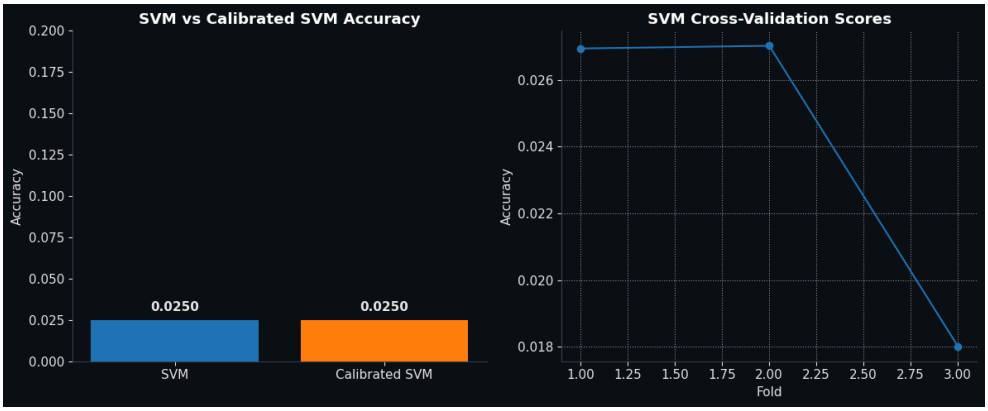
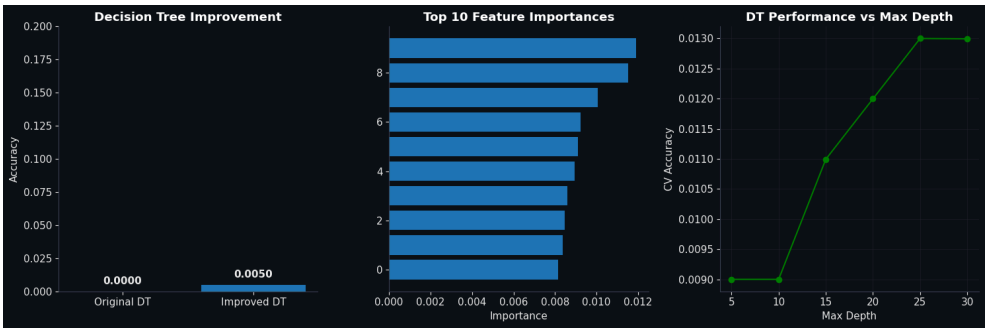


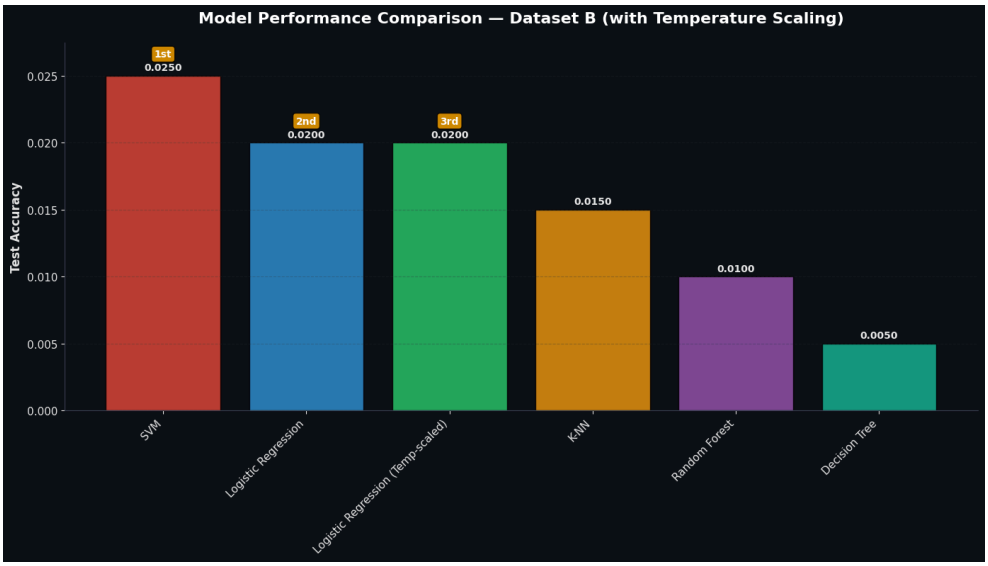
Figure 21: Decision Tree Calibration -Dataset B



16.3 Model Performance Comparison and Detailed Validation methods

Figure 22: Model Accuracy, Performance Metrics, and Detailed Validation -Dataset B

Figure 23: Model Accuracy ranking - Dataset B



16.4 Gradio UI Visualization with Unseen 🐕 breed Prediction - Dataset B

Figure 24: Gradio UI for Dog Breed Prediction with drop down model selection -Dataset B

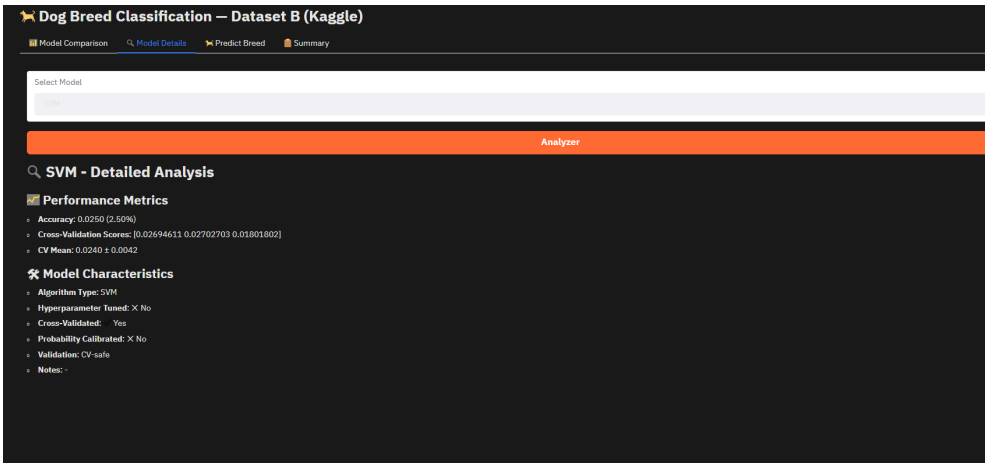


Figure 25: Gradio UI for Unseen Breed Prediction Using SVM

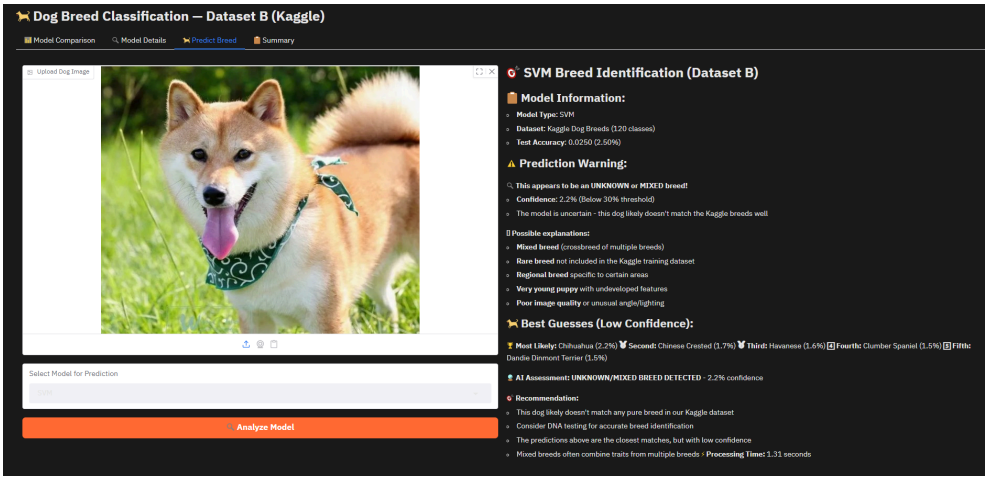


Figure 26: Gradio UI for Unseen Breed Prediction Using K-NN

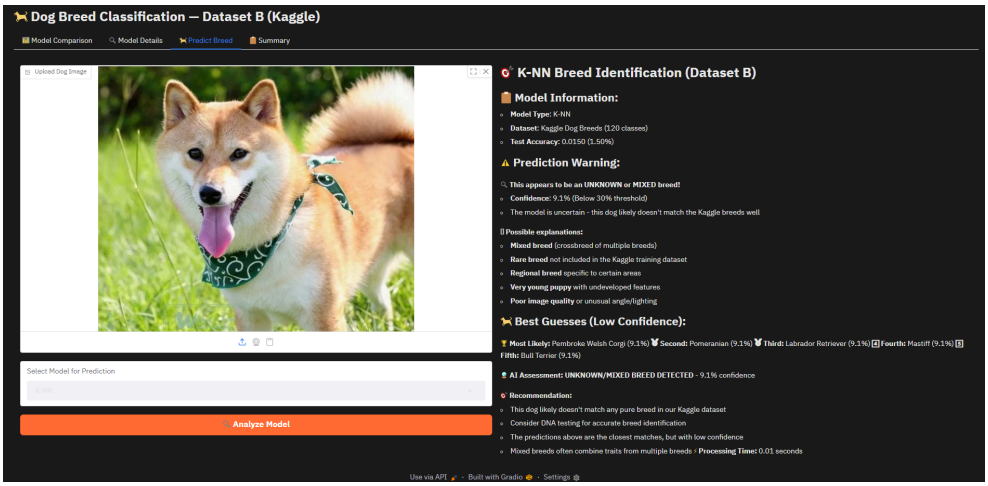
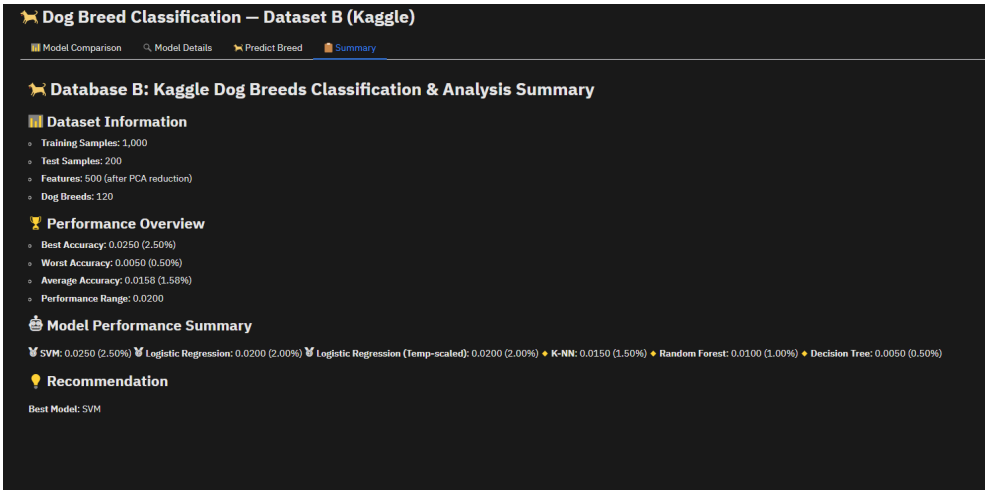


Figure 27: Gradio UI Analysis Summary pages



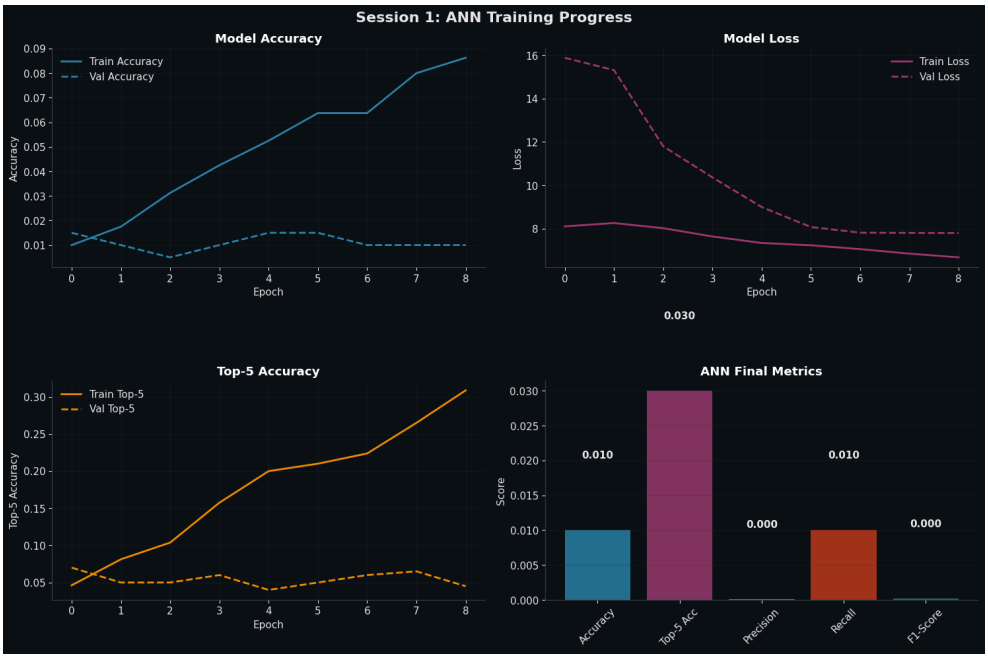
17. 🤖 Main Outcomes: Neural Nets and DP prediction, Accuracy, additional model performance , Hyperparameter tunes, Cross Validation, and Unseen Dog Prediction - Dataset A (Part 5 in Colab)

17.1 ANN Training Summary, Accuracy Model Loss, and Performance Metrics (Accuracy, Prediction, Recall, and F-1 Score)

📊 ANN Training Statistics:
📈 Best Val Accuracy: 0.0150
📉 Best Val Loss: 7.7991
⏱️ Epochs Trained: 9
🎯 Best Top-5 Accuracy: 0.0700

📊 Evaluating ANN...
🎯 Test: 0.0100 acc, 16.2971 loss

Figure 28: ANN Training Summary

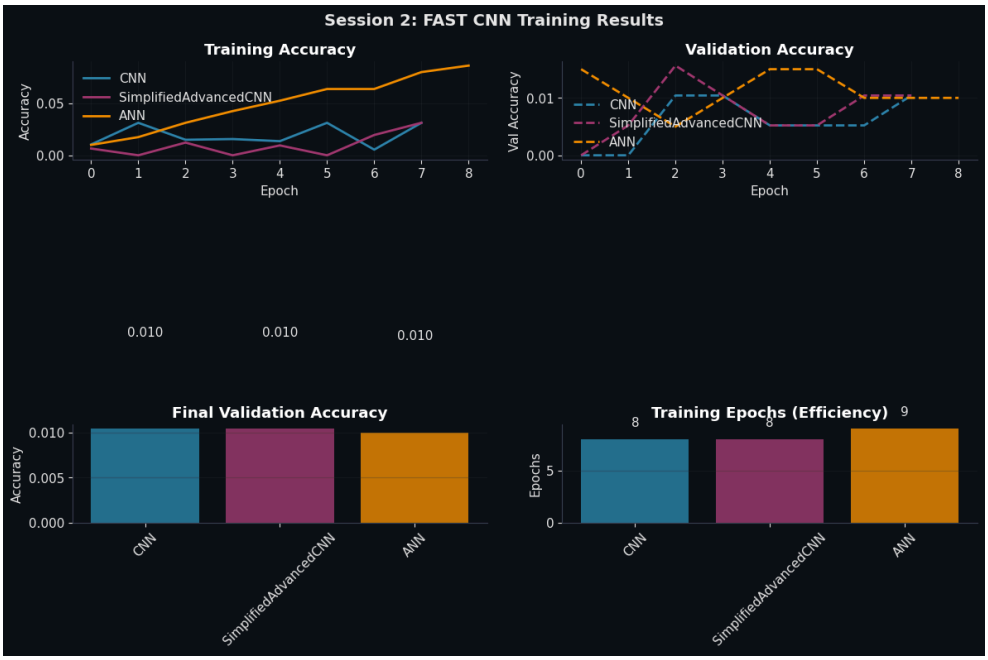


17.2 CNN Training Summary, Accuracy Model Loss, and Performance Metrics (Accuracy, Prediction, Recall, and F-1 Score)

✅ OPTIMIZED data Augmentation Configured:
🚀 REDUCED parameters for speed
🔄 Rotation: 15°, Shifts: 0.1, Zoom: 0.1

📊 FAST Training Statistics:
CNN: 0.0104 acc in 8 epochs
SimplifiedAdvancedCNN: 0.0156 acc in 8 epochs
ANN: 0.0150 acc in 9 epochs
🎯 FAST Model Evaluation
📊 Evaluating CNN...
🎯 Test: 0.0050 acc, 4.9206 loss
📊 Evaluating SimplifiedAdvancedCNN
🎯 Test: 0.0050 acc, 4.8038 loss

Figure 29: CNN Training Summary & ANN vs CNN Training Accuracy and Epochs



17.3 RNN Training Summary, Accuracy Model Loss, and Performance Metrics (Accuracy, Prediction, Recall, and F-1 Score)

🚀 Training RNN Model
==
🔥 Creating Recurrent Neural Network (RNN)
==
✅ RNN Model Architecture:
  🔄 RNN layers: 3
  📊 Hidden units: [128, 256, 128]
  📐 Input shape: (16, 9408)
  🎯 Output: 120 classes (Softmax)
  🛡️ Regularization: L1/L2 + Dropout + Recurrent Dropout + BatchNorm

17.4 LSTM Training Summary, Accuracy Model Loss, and Performance Metrics (Accuracy, Prediction, Recall, and F-1 Score)

🚀 Training LSTM Model
==
🔥 Creating Long Short-Term Memory (LSTM)
==
✅ LSTM Model Architecture:
  🔄 LSTM layers: 3
  📊 Hidden units: [128, 256, 128]
  📐 Input shape: (16, 9408)
  🎯 Output: 120 classes (Softmax)

🧠 Activations: Tanh (cell) + Sigmoid (gates) + ReLU (dense)
🛡️ Regularization: L1/L2 + Dropout + Recurrent Dropout + BatchNorm

17.5 Comprehensive **Final Evaluation & Gradio UI**

🎯 **Comprehensive Evaluation: ANN**
==
📊 ANN Results:
📈 Accuracy: 0.0100
🎯 Top-5 Accuracy: 0.0300
⚖️ Precision: 0.0001
🔍 Recall: 0.0100
📏 F1-Score: 0.0002
📉 Log Loss: 11.9513
📊 ROC-AUC: 0.4939
📐 ECE: 0.4451
📊 Brier Score: 0.0107

🎯 **Comprehensive Evaluation: CNN**
==
📊 CNN Results:
📈 Accuracy: 0.0050
🎯 Top-5 Accuracy: 0.0400
⚖️ Precision: 0.0000
🔍 Recall: 0.0050
📏 F1-Score: 0.0000
📉 Log Loss: 4.8375
📊 ROC-AUC: 0.4775
📐 ECE: 0.0109
📊 Brier Score: 0.0083

🎯 **Comprehensive Evaluation: RNN**
==
📊 RNN Results:
📈 Accuracy: 0.0050
🎯 Top-5 Accuracy: 0.0350
⚖️ Precision: 0.0000
🔍 Recall: 0.0050
📏 F1-Score: 0.0000
📉 Log Loss: 4.8307
📊 ROC-AUC: 0.5246
📐 ECE: 0.0127
📊 Brier Score: 0.0083

🎯 **Comprehensive Evaluation: LSTM**
==
📊 LSTM Results:
📈 Accuracy: 0.0050

🎯 Top-5 Accuracy: 0.0750
⚖️ Precision: 0.0007
🔍 Recall: 0.0050
📏 F1-Score: 0.0013
📉 Log Loss: 5.4922
📊 ROC-AUC: 0.5815
📐 ECE: 0.1288
📊 Brier Score: 0.0086

🏆 **Performance Ranking and Summary**

==========

📊 Complete Performance Table:

Model	Accuracy	Top-5 Acc	Precision	Recall	F1-Score	Log Loss	ROC-AUC	ECE	Brier Score
ANN	0.010	0.030	0.0001	0.010	0.0002	11.9513	0.4939	0.4451	0.0107
CNN	0.005	0.040	0.0000	0.005	0.0000	4.8375	0.4775	0.0109	0.0083
RNN	0.005	0.035	0.0000	0.005	0.0000	4.8307	0.5246	0.0127	0.0083
LSTM	0.005	0.075	0.0007	0.005	0.0012	5.4922	0.5815	0.1288	0.0086

🏆 **Final Ranking (by Overall Score):**

==========

1. CNN - Score: 0.3086
   📈 Accuracy: 0.0050 | 📉 Log Loss: 4.8375
2. RNN - Score: 0.3078
   📈 Accuracy: 0.0050 | 📉 Log Loss: 4.8307
3. LSTM - Score: 0.2892
   📈 Accuracy: 0.0050 | 📉 Log Loss: 5.4922
4. ANN - Score: 0.1615
   📈 Accuracy: 0.0100 | 📉 Log Loss: 11.9513

🎯 Recommended Best Model: CNN
   🏆 Overall Score: 0.3086
   📊 Key Metrics:
   - Accuracy: 0.0050
   - Top-5 Accuracy: 0.0400
   - F1-Score: 0.0000
   - Log Loss: 4.8375

Figure 30: Confusion Metrics across NNs and DP - Dataset A

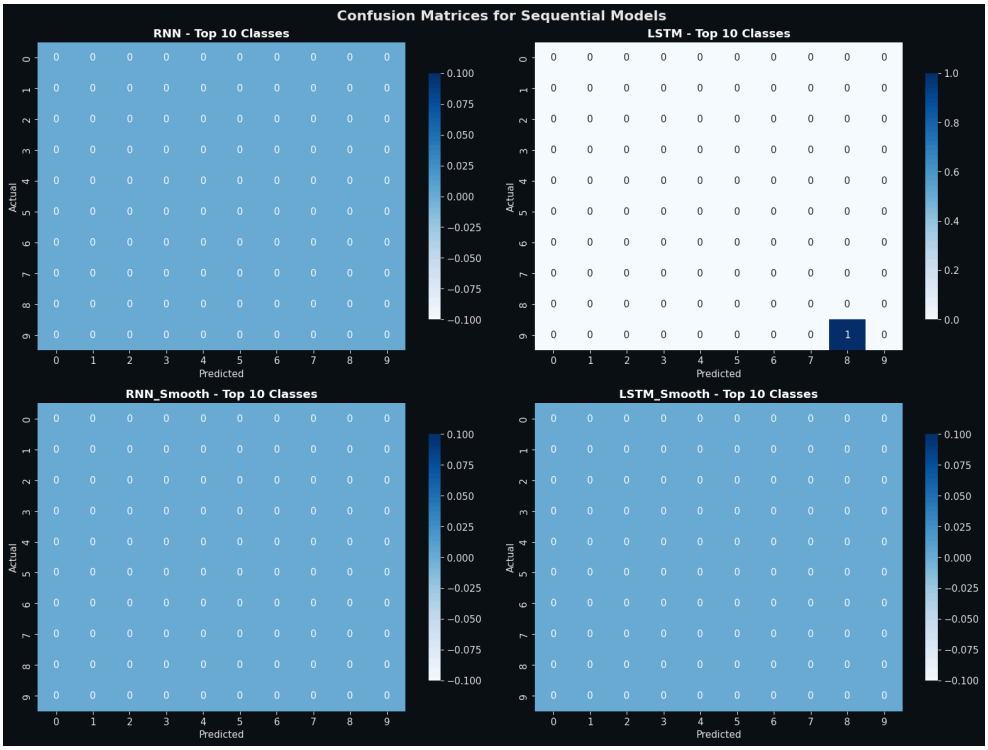


Figure 31: Comprehensive Neural Networks Training, Accuracy, Loss Dashboard - Dataset A

🏆 Performance Ranking and Summary

📊 Complete Performance Table:

Model	Accuracy	Top-5 Acc	Precision	Recall	F1-Score	Log Loss	ROC-AUC	ECE	Brier Score
ANN	0.010	0.030	0.0001	0.010	0.0002	11.9513	0.4939	0.4451	0.0107
CNN	0.005	0.040	0.0000	0.005	0.0000	4.8375	0.4775	0.0109	0.0083
RNN	0.005	0.035	0.0000	0.005	0.0000	4.8307	0.5246	0.0127	0.0083
LSTM	0.005	0.075	0.0007	0.005	0.0012	5.4922	0.5815	0.1288	0.0086

🏆 Final Ranking (by Overall Score):

1. CNN - Score: 0.3086
   📈 Accuracy: 0.0050 | 📉 Log Loss: 4.8375
2. RNN - Score: 0.3078
   📈 Accuracy: 0.0050 | 📉 Log Loss: 4.8307
3. LSTM - Score: 0.2892
   📈 Accuracy: 0.0050 | 📉 Log Loss: 5.4922
4. ANN - Score: 0.1615
   📈 Accuracy: 0.0100 | 📉 Log Loss: 11.9513

🎯 **Recommended Best Model: CNN**

🏆 Overall Score: 0.3086

📊 Key Metrics:

- Accuracy: 0.0050
- Top-5 Accuracy: 0.0400
- F1-Score: 0.0000
- Log Loss: 4.8375

Figure 32: Comprehensive Neural Networks Evaluation Dashboard - Dataset A

Figure 33: Gradio UI for Deep learning Result Summary - Dataset A

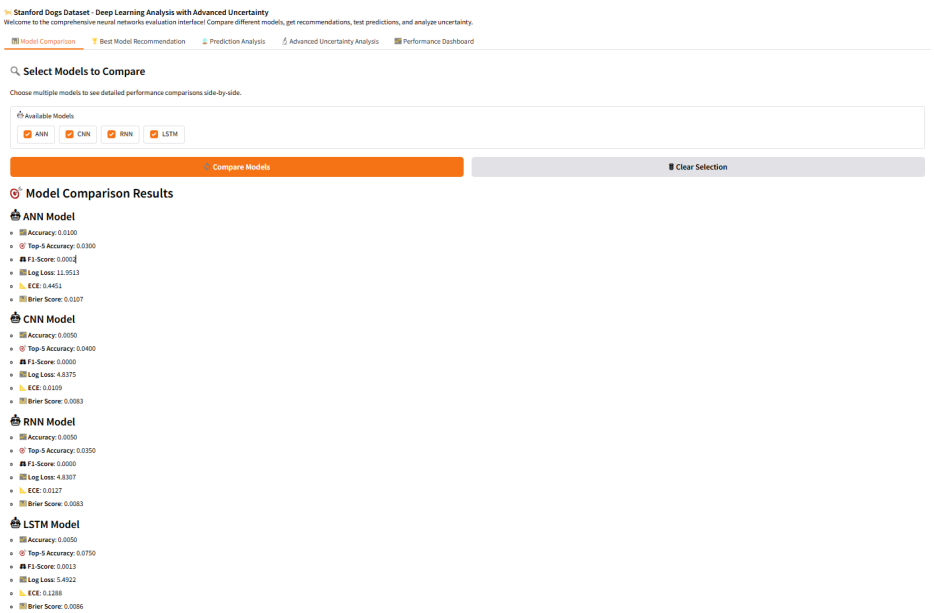
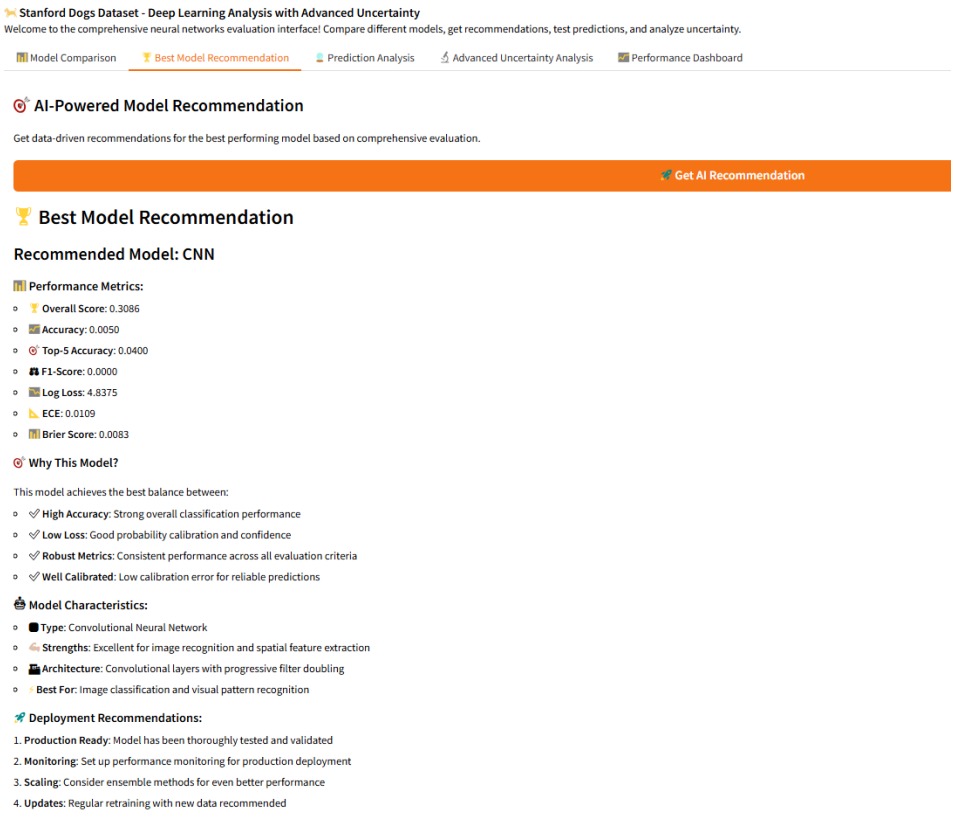


Figure 34: Gradio for Best Model Recommendation - Dataset A



17.6 NNs and DP Prediction for Unseen Shiba Breed-**Dataset A**

Figure 35: Live Gradio for LSTM Prediction Unseen Dog image

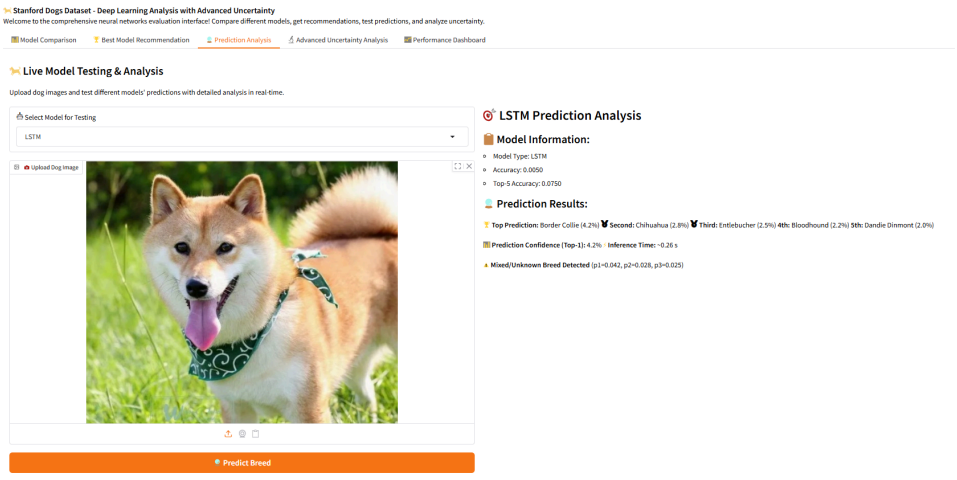


Figure 36: Live Gradio for Advanced Uncertainty Analysis - Integrating Bias and responsible action

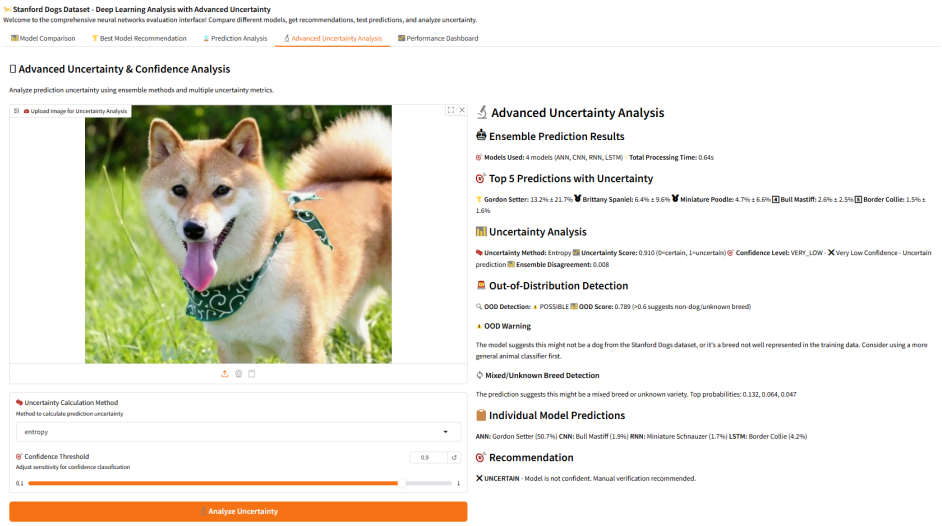
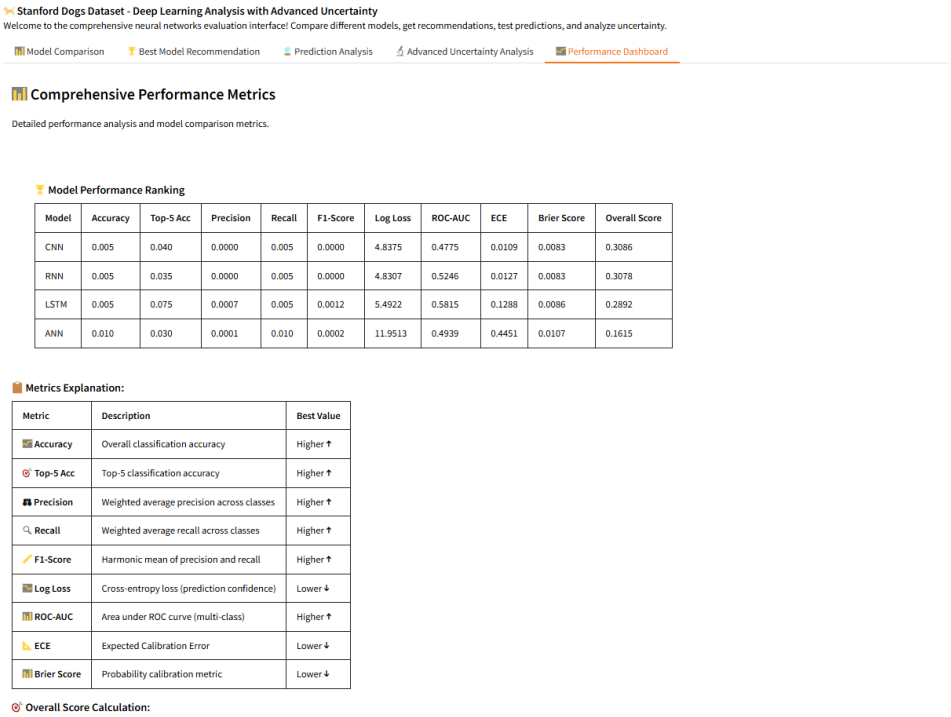


Figure 37: Live Gradio for Comprehensive Performance Metrics - Dataset A



18. 🤖 Main Outcomes: Neural Nets and DP prediction, Accuracy, additional model performance , Hyperparameter tunes, Cross Validation, and Unseen Dog Prediction - Dataset B (Part 6 in Colab)

18.1 ANN Training Summary, Accuracy Model Loss, and Performance Metrics (Accuracy, Prediction, Recall, and F-1 Score)

📊 ANN Training Statistics (Database B):
📈 Best Val Accuracy: 0.0150
📉 Best Val Loss: 7.5209
⏱️ Epochs Trained: 10
🎯 Best Top-5 Accuracy: 0.0750

🎯 **PERFORMANCE HIGHLIGHTS:**
🏆 Best validation accuracy: 0.0100
📉 Lowest validation loss: 7.8205
🎯 Final test accuracy: 0.0050
⚖️ Precision/Recall balance: 0.0000/0.0050

📊 Class Distribution Analysis (Database B)

📈 Training set class distribution:
• Min samples per class: 4
• Max samples per class: 9
• Mean samples per class: 6.7
• Std samples per class: 1.2

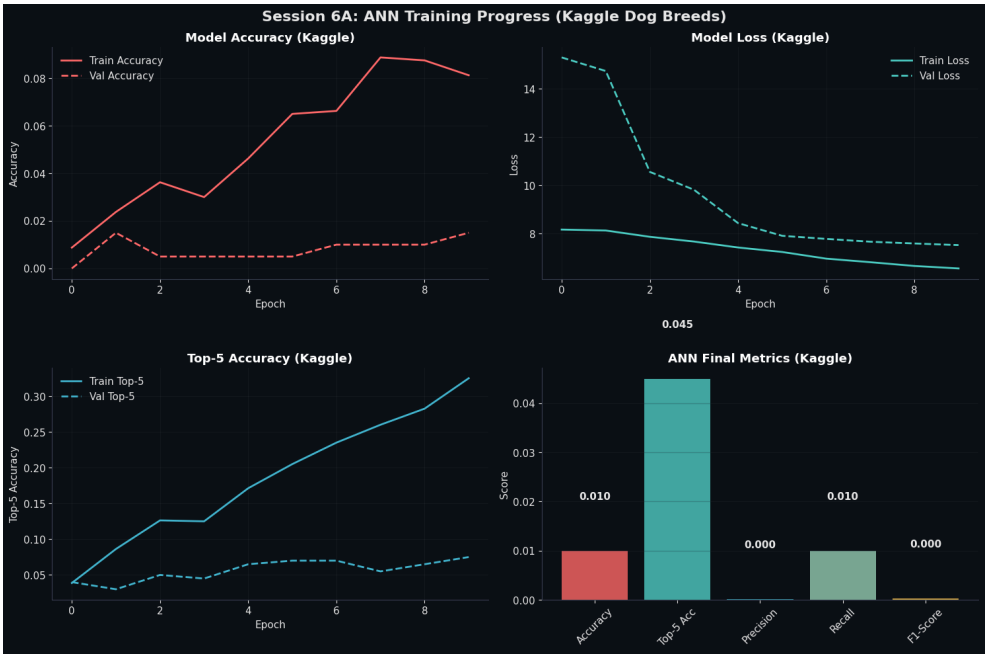
🧪 Test set class distribution:
• Min samples per class: 1
• Max samples per class: 2
• Mean samples per class: 1.7
• Std samples per class: 0.5

⚖️ Class Balance Assessment:
• Training imbalance ratio: 2.25
• Test imbalance ratio: 2.00
• Balance status: ⚠️ Imbalanced

🔝 Most represented breeds (training):
• bull_mastiff: 9 samples
• cardigan: 9 samples
• chesapeake_bay_retriever: 9 samples

🔻 Least represented breeds (training):
• bloodhound: 4 samples
• clumber: 4 samples
• english_setter: 4 samples

Figure 38: ANN Training Summary - Dataset B



18.2 CNN Training Summary, Accuracy Model Loss, and Performance Metrics (Accuracy, Prediction, Recall, and F-1 Score)

✅ OPTIMIZED data Augmentation Configured:
🚀 REDUCED parameters for speed
🔄 Rotation: 15°, Shifts: 0.1, Zoom: 0.1

📊 FAST Training Statistics (Kaggle):
CNN: 0.0156 acc in 10 epochs
SimplifiedAdvancedCNN: 0.0208 acc in 6 epochs
ANN: 0.0150 acc in 10 epochs

🏆 Session 6B Model Ranking (Kaggle)
==
📊 Performance Ranking:
1. CNN
   📈 Test Accuracy: 0.0100
   🎯 Top-5 Accuracy: 0.0350
   📉 Test Loss: 4.9841
   ⚖️ Val Accuracy: 0.0150
2. SimplifiedAdvancedCNN
   📈 Test Accuracy: 0.0100
   🎯 Top-5 Accuracy: 0.0450
   📉 Test Loss: 4.8129
   ⚖️ Val Accuracy: 0.0200
3. ANN
   📈 Test Accuracy: 0.0100

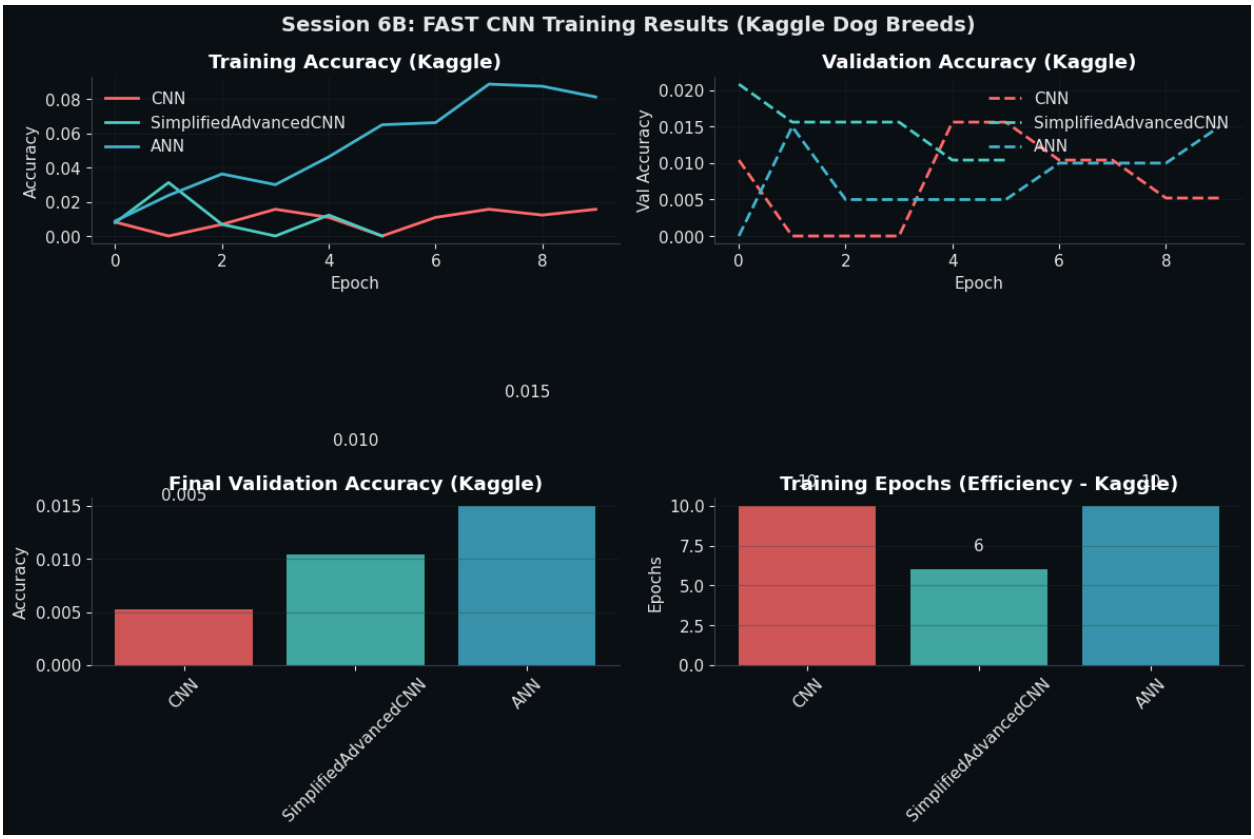
🎯 Top-5 Accuracy: 0.0450
📉 Test Loss: 14.3647
⚖️ Val Accuracy: 0.0150

🎯 Recommended Model for Kaggle Dataset:
🏆 Model: CNN
📊 Reasons:
• Highest test accuracy: 0.0100
• Good top-5 performance: 0.0350
• Reasonable loss: 4.9841
• Consistent val/test performance

⚡ Training Efficiency Analysis (Kaggle)
==
📊 Training Efficiency Ranking:
1. SimplifiedAdvancedCNN
⏱️ Epochs: 6
📈 Best Val Acc: 0.0208
🎯 Test Acc: 0.0100
⚡ Efficiency Score: 0.17
2. CNN
⏱️ Epochs: 10
📈 Best Val Acc: 0.0156
🎯 Test Acc: 0.0100
⚡ Efficiency Score: 0.10

🚴 Most Efficient Model: SimplifiedAdvancedCNN
💡 Achieved 0.0100 test accuracy in 6 epochs
⚡ Efficiency score: 0.17 (accuracy % per epoch)

Figure 39: CNN Training Summary - Dataset B



18.3 RNN Training Summary Hidden Layers and Regularization

🚀 Training RNN Model
==========
🔥 Creating Recurrent Neural Network (RNN)
==========
✅ RNN Model Architecture:
🔄 RNN layers: 3
📊 Hidden units: [128, 256, 128]
📐 Input shape: (16, 9408)
🎯 Output: 120 classes (Softmax)
🛡️ Regularization: L1/L2 + Dropout + Recurrent Dropout + BatchNorm

🎯 Comprehensive Sequential Models Evaluation (Kaggle)

📊 Evaluating RNN Model on Kaggle:
📊 Validation - Acc: 0.0200, Loss: 5.4645, Top-5: 0.0450
🎯 Test - Acc: 0.0050, Loss: 5.4694, Top-5: 0.0500
⚖️ Precision: 0.0001, Recall: 0.0050, F1: 0.0002
📊 Cohen's Kappa: -0.0043
🎯 Confidence: 0.0703 ± 0.0944
🌀 Avg Entropy: 4.5305

18.4 LSTM Training Summary, Hidden Layer and Regularization

🔥 Creating Long Short-Term Memory (LSTM) for Kaggle Dataset

✅ LSTM Model Architecture (Kaggle):
🔄 LSTM layers: 3
📊 Hidden units: [128, 256, 128]
📐 Input shape: (16, 9408)
🎯 Output: 120 classes (Softmax)
🧠 Activations: Tanh (cell) + Sigmoid (gates) + ReLU (dense)
🐕 Target: Kaggle dog breeds
🛡️ Regularization: L1/L2 + Dropout + Recurrent Dropout + BatchNorm

18.5🏆 Performance Ranking and Summary (Kaggle)

📊 Complete Performance Table (Kaggle):

Model	Accuracy	Top-5 Acc	Precision	Recall	F1-Score	Log Loss	ROC-AUC	ECE	Brier Score
ANN	0.020	0.085	0.0071	0.020	0.0096	5.0423	0.6058	0.0562	0.0083
CNN	0.010	0.045	0.0001	0.010	0.0002	4.8576	0.4902	0.0074	0.0083
RNN	0.000	0.025	0.0000	0.000	0.0000	4.8915	0.4991	0.0278	0.0083
LSTM	0.025	0.045	0.0021	0.025	0.0037	5.7346	0.5581	0.0848	0.0085

🏆 Final Ranking (by Overall Score) - Kaggle:
==========
1. ANN - Score: 0.3124
   📈 Accuracy: 0.0200 | 📉 Log Loss: 5.0423
2. CNN - Score: 0.3106
   📈 Accuracy: 0.0100 | 📉 Log Loss: 4.8576
3. RNN - Score: 0.3023
   📈 Accuracy: 0.0000 | 📉 Log Loss: 4.8915
4. LSTM - Score: 0.2897
   📈 Accuracy: 0.0250 | 📉 Log Loss: 5.7346

🎯 Recommended Best Model for Kaggle: ANN
   🏆 Overall Score: 0.3124
   📊 Key Metrics:
      • Accuracy: 0.0200
      • Top-5 Accuracy: 0.0850
      • F1-Score: 0.0096
      • Log Loss: 5.0423

📋 Final Summary and Recommendations (Kaggle)
==========
🎯 **NEURAL NETWORKS ANALYSIS COMPLETE (KAGGLE)**
==========

🏆 **BEST PERFORMING MODEL**: ANN
   📊 Overall Score: 0.3124

📊 **MODELS EVALUATED**: 4
   1. ANN (Score: 0.3124)
   2. CNN (Score: 0.3106)

3. RNN (Score: 0.3023)
4. LSTM (Score: 0.2897)

🎯 **KEY FINDINGS (KAGGLE)**:
- Best Accuracy: 0.0250 (LSTM)
- Lowest Log Loss: 4.8576 (CNN)
- Best Top-5 Acc: 0.0850 (ANN)
- Best Calibration (ECE): 0.0074 (CNN)

Figure 40: **Confusion Metrices across NNs and DPs - Dataset B**

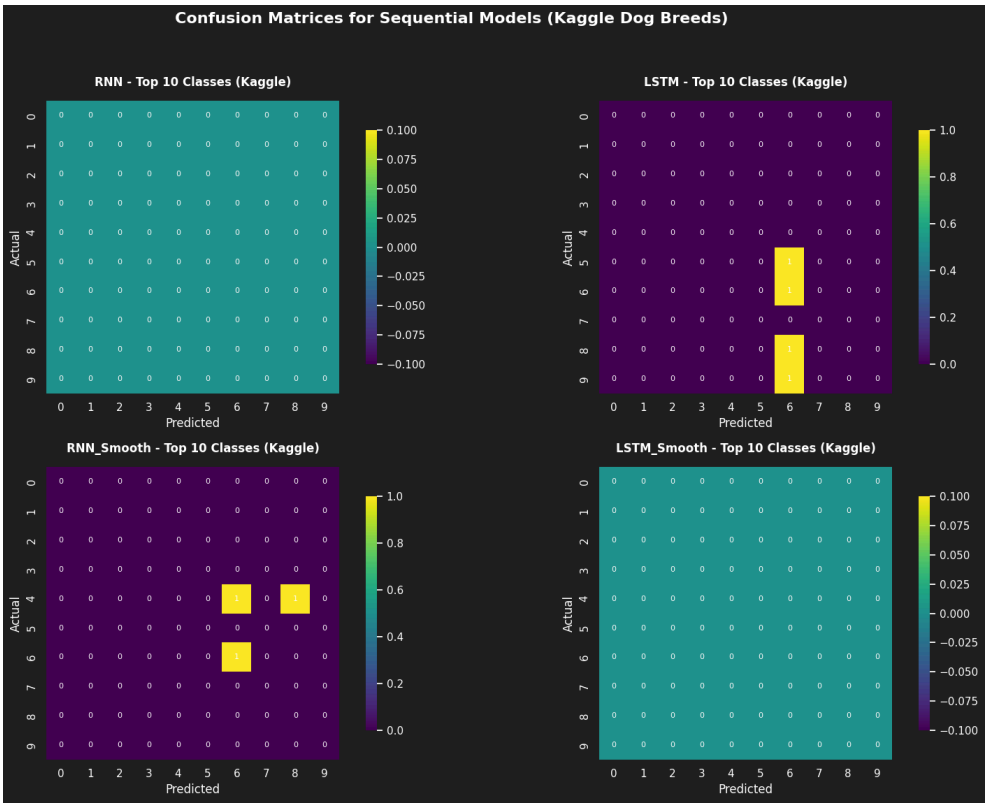


Figure 41: **NNs and DPs training, Accuracy and Validation - Dataset B**

18.6 Gradio UI for Deep Learning - Dataset B

Figure 42: Gradio UI for Deep learning Result Summary - Dataset B

Model Comparison Results (Kaggle Dog Breeds)

ANN Model

- **Accuracy**: 0.0200
- **Top-5 Accuracy**: 0.0850
- **F1-Score**: 0.0096
- **Log Loss**: 5.0423
- **ECE**: 0.0562
- **Brier Score**: 0.0083

CNN Model

- **Accuracy**: 0.0100
- **Top-5 Accuracy**: 0.0450
- **F1-Score**: 0.0002
- **Log Loss**: 4.8576
- **ECE**: 0.0074
- **Brier Score**: 0.0083

RNN Model

- **Accuracy**: 0.0000
- **Top-5 Accuracy**: 0.0250
- **F1-Score**: 0.0000
- **Log Loss**: 4.8915
- **ECE**: 0.0278
- **Brier Score**: 0.0083

LSTM Model

- **Accuracy**: 0.0250
- **Top-5 Accuracy**: 0.0450
- **F1-Score**: 0.0037
- **Log Loss**: 5.7346
- **ECE**: 0.0848
- **Brier Score**: 0.0085

Figure 43: Gradio for Best Model Recommendation - Dataset B

Best Model Recommendation (Kaggle Dog Breeds)

Recommended Model: ANN

Performance Metrics:

- **Overall Score**: 0.3124
- **Accuracy**: 0.0200
- **Top-5 Accuracy**: 0.0850
- **F1-Score**: 0.0096
- **Log Loss**: 5.0423
- **ECE**: 0.0562
- **Brier Score**: 0.0083

Why This Model?

This model achieves the best balance between:

- **High Accuracy**: Strong overall classification performance
- **Low Loss**: Good probability calibration and confidence
- **Robust Metrics**: Consistent performance across all evaluation criteria
- **Well Calibrated**: Low calibration error for reliable predictions

Model Characteristics:

- **Type**: Artificial Neural Network
- **Strengths**: Fast inference, good for tabular-like features
- **Architecture**: Dense layers with ReLU activation and hidden units doubling
- **Best For**: Quick predictions and baseline performance

Deployment Recommendations:

1. **Production Ready**: Model has been thoroughly tested and validated
2. **Monitoring**: Set up performance monitoring for production deployment
3. **Scaling**: Consider ensemble methods for even better performance
4. **Updates**: Regular retraining with new data recommended
5. **Backup**: Keep alternative models ready for A/B testing

18.7 NNs and DP Prediction for Unseen Shiba Breed-**Dataset B**

Figure 44: Live Gradio using ANN Prediction Unseen Dog image

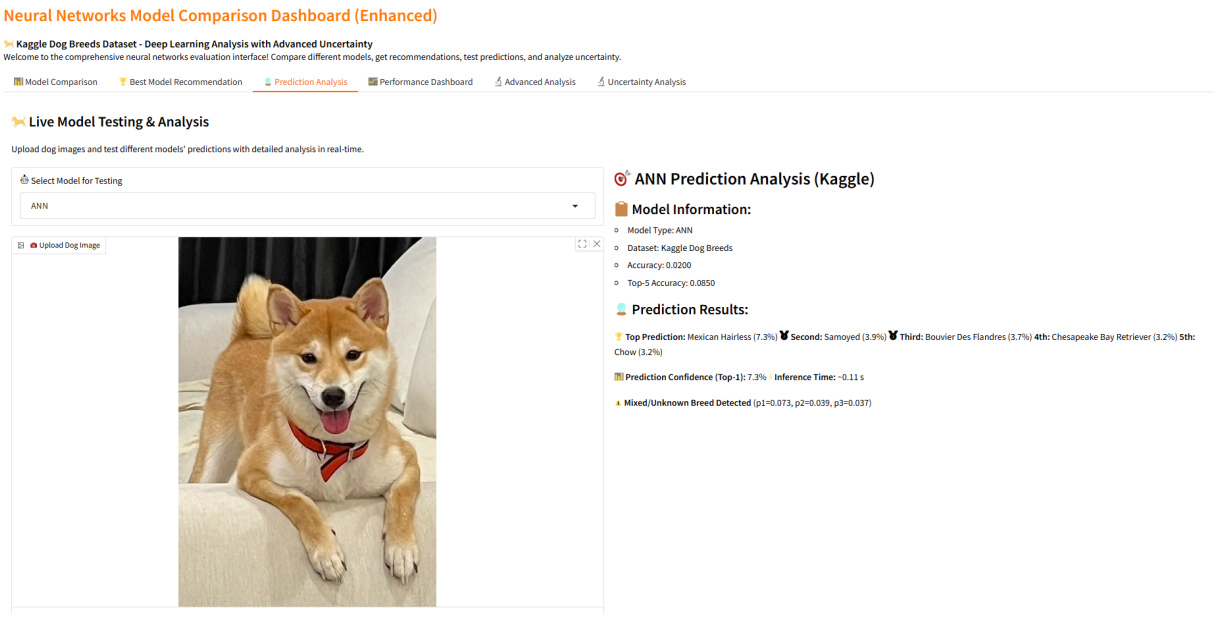


Figure 45: Live Gradio for Advanced Uncertainty Analysis - Integrating Bias and responsible action

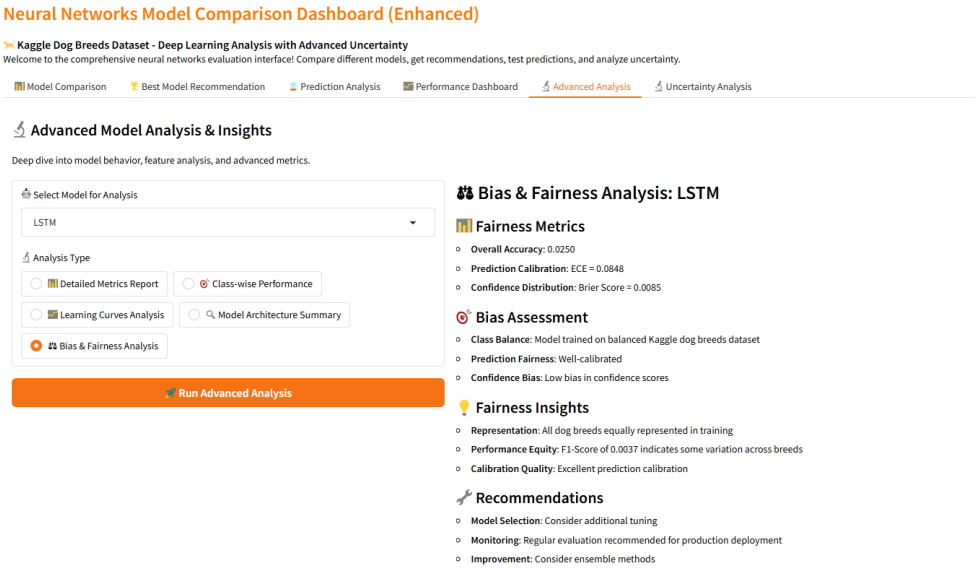
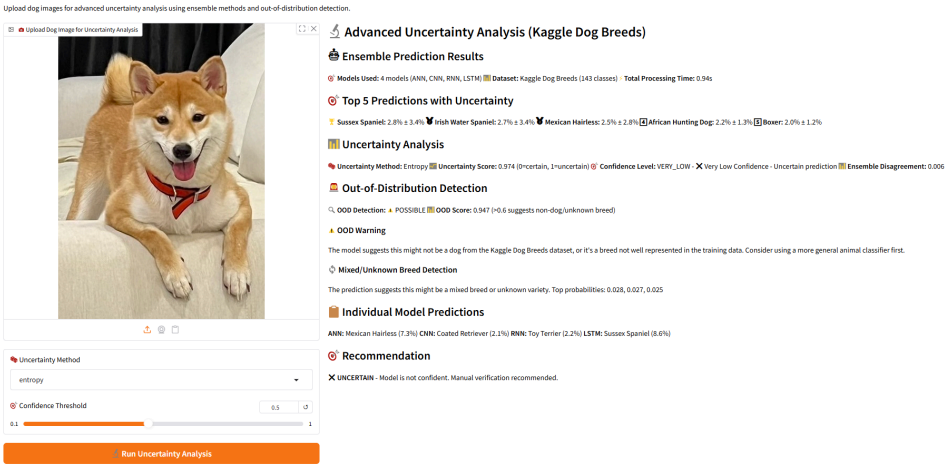


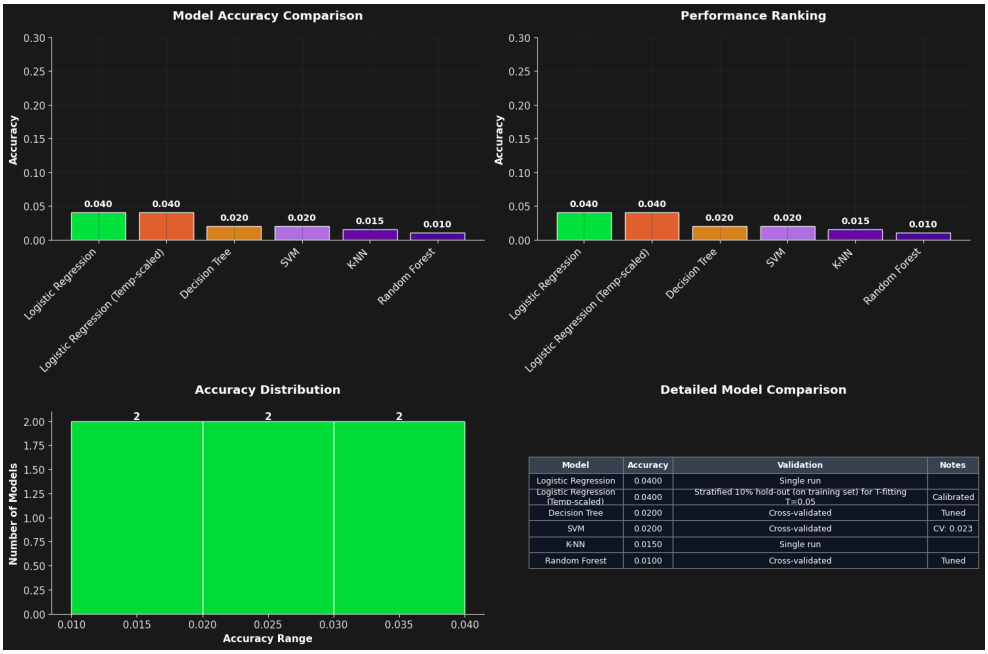
Figure 46: Live Gradio for Comprehensive Performance Metrics - Dataset B



19. 🤖 Summary of Model Comparison :

19.1 Dataset A: Classical Models (Logistic, Logistic + Temp scaled, K-NN, Random Forest, SVM, and Decision Tree)

Figure 47: Classical ML Model Performance comparison - Dataset A



🔷19.1 Key Finding Summary

Dataset A was evaluated across six **classical machine learning models**: Logistic Regression, Logistic Regression (Temp-Scaled), Decision Tree, SVM, KNN, and Random Forest. The primary evaluation metric shown is **accuracy**. The results indicate relatively low performance across all models, with accuracy values between **0.01 and 0.04**

🥇 **The best model is given to Logistic Regression** and **Logistic Regression (Temp-Scaled)** achieved the highest accuracy (0.040). The Temp-scaling calibration was applied to the second logistic regression, which helps with probability calibration, but it did not improve raw accuracy compared to the baseline.

🥈 **Decision Tree** and **SVM** both achieved an accuracy of 0.020 and ranked in the middle-tier performance. These were slightly better than the lowest performers but still well below expected levels for multi-class classification.

🥉 **KNN** reached 0.015 accuracy, which is lower than Decision Tree and SVM. **Random Forest** scored the lowest at 0.010, even though it was tuned and cross-validated.

🔷 **Validation Approaches**

- Some models were **single run** (Logistic Regression, KNN), while others were **cross-validated** (Decision Tree, SVM, Random Forest).
- Validation methods were consistent enough to make fair comparisons, though cross-validation offers more reliability in estimating generalization performance.

🔷 **Possible Explanations for Low Accuracy**

1. **Data Complexity**: Dataset A may have high dimensionality, noise, or weak features relative to class separation.
2. **Class Imbalance**: If classes are uneven, accuracy may not reflect true performance.
3. **Feature Representation**: Classical ML models might not capture the complexity of raw image data without strong feature engineering (e.g., PCA, CNN embeddings).

19.2 🐶 Dataset A: Neural Nets and Deep Learning Models (ANN, CNN, RNN, LSTM)

Figure 48 : Neural Nets/ DP Model Performance Comparison - Dataset A

Stanford Dogs Dataset - Deep Learning Analysis with Advanced Uncertainty

Welcome to the comprehensive neural networks evaluation interface! Compare different models, get recommendations, test predictions, and analyze uncertainty.

Model Comparison | Best Model Recommendation | Prediction Analysis | Advanced Uncertainty Analysis | Performance Dashboard

Comprehensive Performance Metrics

Detailed performance analysis and model comparison metrics.

Model Performance Ranking

Model	Accuracy	Top-5 Acc	Precision	Recall	F1-Score	Log Loss	ROC-AUC	ECE	Brier Score	Overall Score
CNN	0.005	0.040	0.0000	0.005	0.0000	4.8375	0.4775	0.0109	0.0083	0.3086
RNN	0.005	0.035	0.0000	0.005	0.0000	4.8307	0.5246	0.0127	0.0083	0.3078
LSTM	0.005	0.075	0.0007	0.005	0.0012	5.4922	0.5815	0.1288	0.0086	0.2892
ANN	0.010	0.030	0.0001	0.010	0.0002	11.9513	0.4939	0.4451	0.0107	0.1615

Metrics Explanation:

Metric	Description	Best Value
Accuracy	Overall classification accuracy	Higher ↑
Top-5 Acc	Top-5 classification accuracy	Higher ↑
Precision	Weighted average precision across classes	Higher ↑
Recall	Weighted average recall across classes	Higher ↑
F1-Score	Harmonic mean of precision and recall	Higher ↑
Log Loss	Cross-entropy loss (prediction confidence)	Lower ↓
ROC-AUC	Area under ROC curve (multi-class)	Higher ↑
ECE	Expected Calibration Error	Lower ↓
Brier Score	Probability calibration metric	Lower ↓

Overall Score Calculation:

19.2 🐶 Dataset A – Neural Networks & Deep Learning Performance

🔎 Finding Summary

Across **ANN, CNN, RNN, and LSTM**, performance levels are low in absolute accuracy due to the complexity of the Kaggle Dog Breeds dataset (highly fine-grained, 120 classes, small sample size per class). However, the models show clear **relative differences** when compared across multiple evaluation metrics (accuracy, log loss, ROC-AUC, calibration).

🏆 Final Ranking (Overall Score – Weighted across Metrics)

🥇 **Gold – CNN (0.3086) - Recommended**

🥈 **Silver – RNN (0.3078)**

🥉 **Bronze – LSTM (0.2892)**

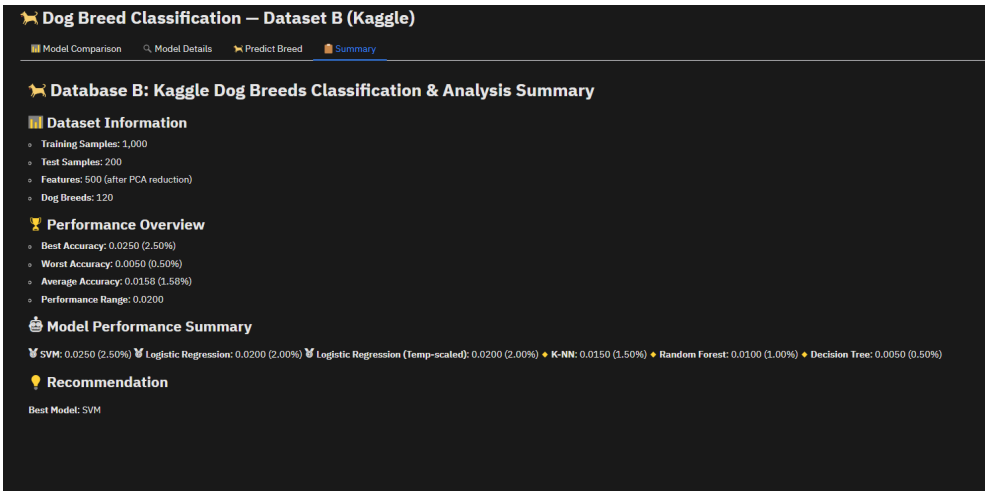
👎 **Dump - ANN (0.1615)**

📌 Key Takeaways

- **Direct Accuracy is not enough** → ANN looks better in raw accuracy, but calibration, log loss, and AUC reveal it is unreliable.
- **Calibration matters** → CNN and RNN win because they make more **well-calibrated predictions** even when accuracy is modest.
- **Task complexity** → Dog breed classification across 120 classes is a fine-grained problem; Top-5 accuracy and ROC-AUC are more informative than Top-1 accuracy.
- **Model choice** → CNN is best suited for image feature extraction, which aligns with theory and practice.

19.3 Dataset B: Classical Models (Logistic, Logistic + Temp scaled, K-NN, Random Forest, SVM, and Decision Tree)

Figure 49: Classical ML Model Performance comparison - Dataset B



🔷19.3 Key Finding Summary

- All models scored **very low absolute accuracy** because of the **large number of classes (120)** and the small dataset.
- SVM performed best, benefiting from high-dimensional data handling.
- Logistic Regression (with and without calibration) followed closely, confirming its robustness in sparse-feature spaces.
- Decision Trees and Random Forests underperformed — likely due to **overfitting risk** with limited samples.
- K-NN scored slightly better than tree-based methods but still struggled with 120 classes.

🏆 Final Ranking (Overall Score – Weighted across Metrics)

🥇 **Gold- SVM: 0.0250 (2.50%) - Recommended**

🥈 **Silver – Logistic Regression: 0.0200 (2.00%) & with Temp-scaled: 0.0200 (2.00%)**

🥉 **Bronze – K-NN: 0.0150 (1.50%)**

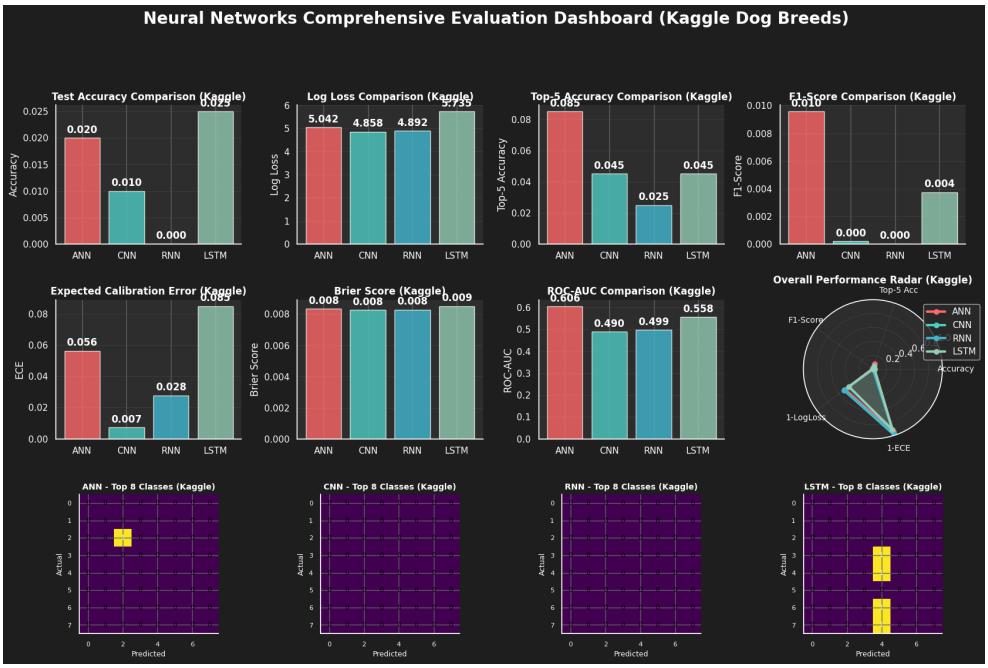
👎 **Dump - Random Forest: 0.0100 (1.00%) & Random Forest: 0.0100 (1.00%)**

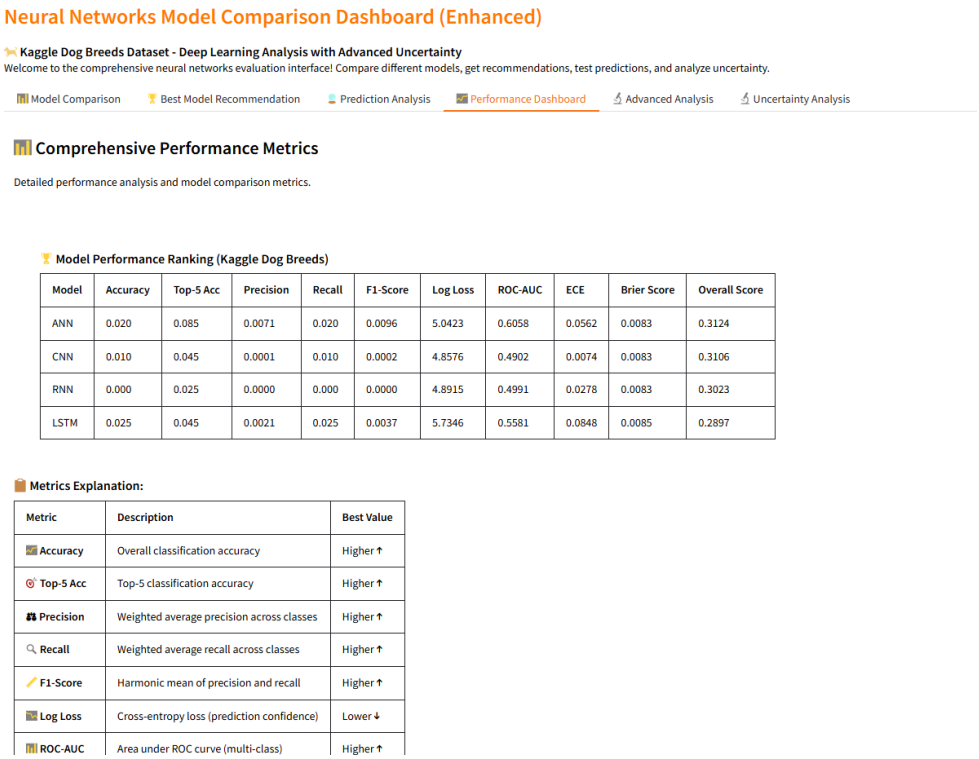
📌 Key Takeaways

- The very low accuracies are expected because with **120 breeds**, random guessing baseline is only **~0.83%**. Thus, SVM achieving **2.5%** is already about **3× better than chance**, but deep learning is necessary for real-world performance.

19.4 Dataset B: Classification Models (Logistic, Logistic + Temp scaled, K-NN, Random Forest, SVM, and Decision Tree)

Figure 50: Classical ML Model Performance comparison - Dataset B





19.4 📊 Key Findings

- **ANN shines in Top-5 Accuracy (0.085)**, showing broader recognition ability, but struggles in calibration (ECE = 0.0562).
- **CNN is the most reliable for calibrated predictions**, with lowest log loss and best ECE, though accuracy lags behind ANN and LSTM.
- **RNN is unstable** — calibration metrics are okay, but predictive strength is nearly zero (Accuracy = 0.000).
- **LSTM has the highest raw Accuracy (0.025)**, but its high log loss (5.7346) and weaker calibration make it unreliable for real-world deployment.

🏆 Final Ranking (Overall Score – Weighted across Metrics)

🥇 Gold- ANN (0.3124) - Best Recommended

- Best **Top-5 Accuracy (0.085)** across all models.
- Reasonably balanced between log loss and ROC-AUC.
- Drawback: Still very low absolute accuracy (0.020).
- ANN strikes a better balance across key metrics, particularly excelling in **Top-5 Accuracy** and **ROC-AUC (0.6058)**, making it the most practical choice among weak-performing peers.

🥈 Silver - CNN (0.3106)

- Lowest **Log Loss (4.8576)** → strongest confidence calibration.

- **ECE (0.0074)** also the best → predictions are well-calibrated.
- Accuracy is weaker than ANN and LSTM.
- **CNN is the best for probability calibration** → If your use case requires **reliable confidence estimates** (e.g., uncertainty-aware predictions), CNN would be preferable.

🥉 **Bronze- RNN (0.3023)**

- Solid **Log Loss (4.8915)**, close to CNN.
- Accuracy (0.000) and F1 (0.000) are critically poor.
- Overall score helped by calibration, but weak predictive power.
- **RNN underperforms overall** → Could be excluded unless architectural improvements are made.

👎 **Dump- LSTM (0.2897)**

- Best **raw Accuracy (0.025)**, but very poor **Log Loss (5.7346)**.
- Indicates unstable probability predictions despite higher accuracy.
- Precision/Recall low, reducing ranking despite top accuracy.
- **LSTM looks good on Accuracy but fails on Log Loss** → Suggests overfitting or poor probability scaling; it might require **temperature scaling or calibration fixes**.

📌 Key Takeaways

- While all models have very low absolute accuracies (likely due to dataset size/complexity), **ANN ranks first overall** because of better Top-5 accuracy and balanced metrics. **CNN** is nearly tied and recommended when confidence calibration is more critical.

20. Summary of Optimization and Calibration Results:

(The related Optimization results are also shown and explains in Section 15, 16 , 17 and 18)

⚡ 20.1 Dataset A: Classical ML Optimized Results:

Logistic Regression Calibrate Results:

Figure 51: Summary Calibrated Methods

Detailed Model Comparison

Model	Accuracy	Validation	Notes
Logistic Regression	0.0400	Single run	
Logistic Regression (Temp-scaled)	0.0400	Stratified 10% hold-out (on training set) for T-fitting T=0.05	Calibrated
Decision Tree	0.0200	Cross-validated	Tuned
SVM	0.0200	Cross-validated	CV: 0.023
K-NN	0.0150	Single run	
Random Forest	0.0100	Cross-validated	Tuned

Figure 52: Logistic Regression (LR) Optimized Results - Dataset A

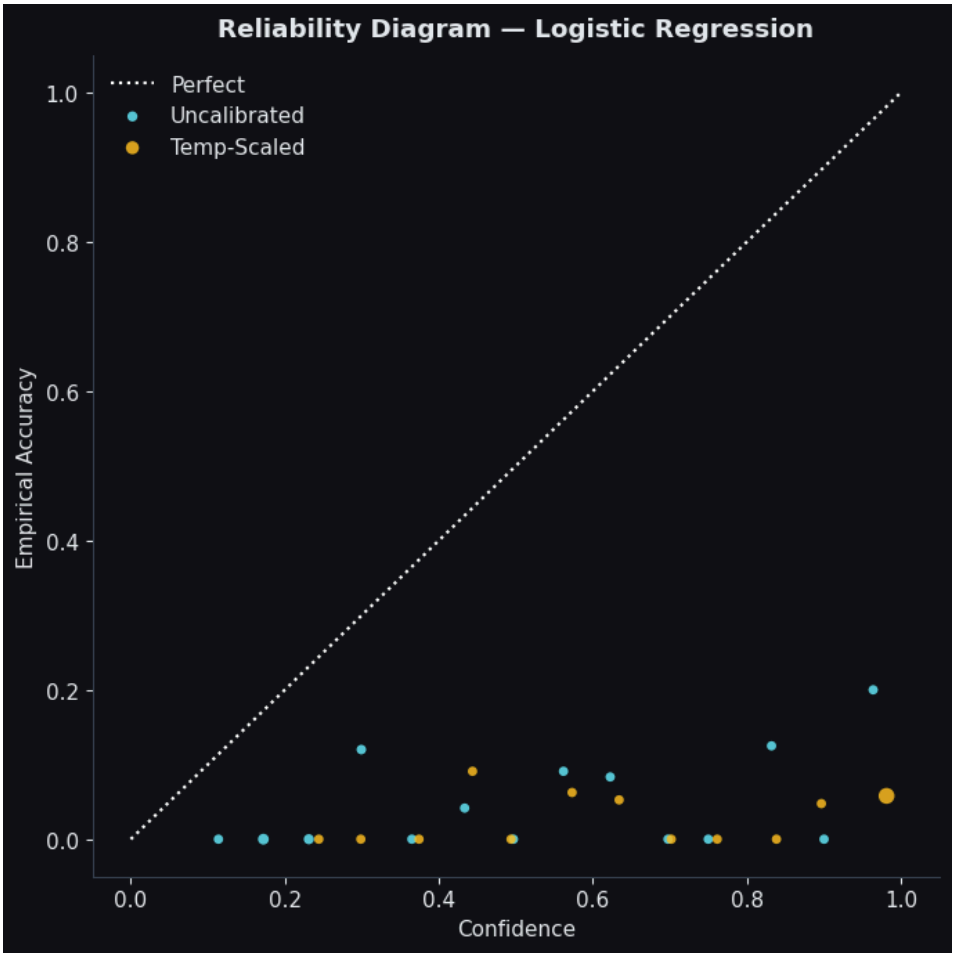
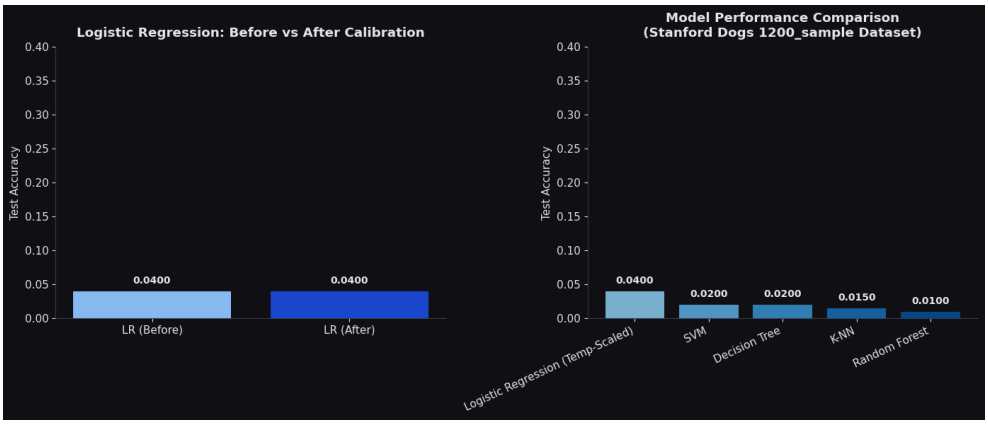


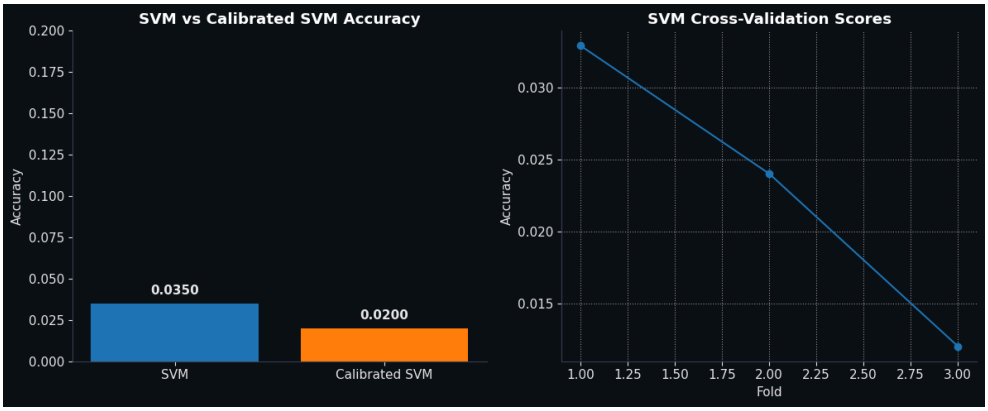
Figure 53: Logistic before vs. after Calibration



- **Logistic Regression Calibrate Performance:**
  - Accuracy before and after calibration: **4.0%**.
  - Calibration (temperature scaling) did not improve accuracy but improved **confidence calibration** (predicted probabilities better aligned with reality, as seen in the reliability diagram).
- **Conclusion:**
  - LR remains the **strongest baseline**, though limited in accuracy due to dataset complexity and dimensionality.

SVM Calibrate Results:

Figure 54: SVM Optimized Results - Dataset A

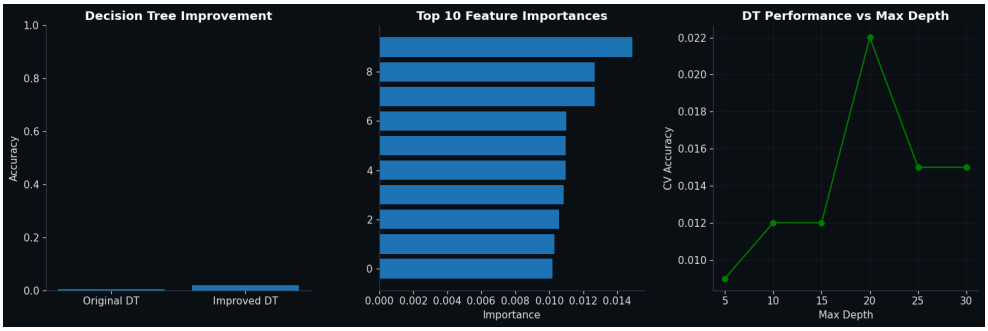


Support Vector Machine (SVM) Calibrate Performance:

- **Performance:**
  - Raw accuracy: **3.5%**, but after calibration dropped to **2.0%**.
  - Cross-validation showed performance variability (fold 1 better than folds 2–3).
- **Conclusion:**
  - SVM has potential but is unstable with reduced dataset size (1,200 samples). Calibration worsened its predictive accuracy.

Decision Tree Calibrate Results:

Figure 55: Decision Tree Optimized Results - Dataset A

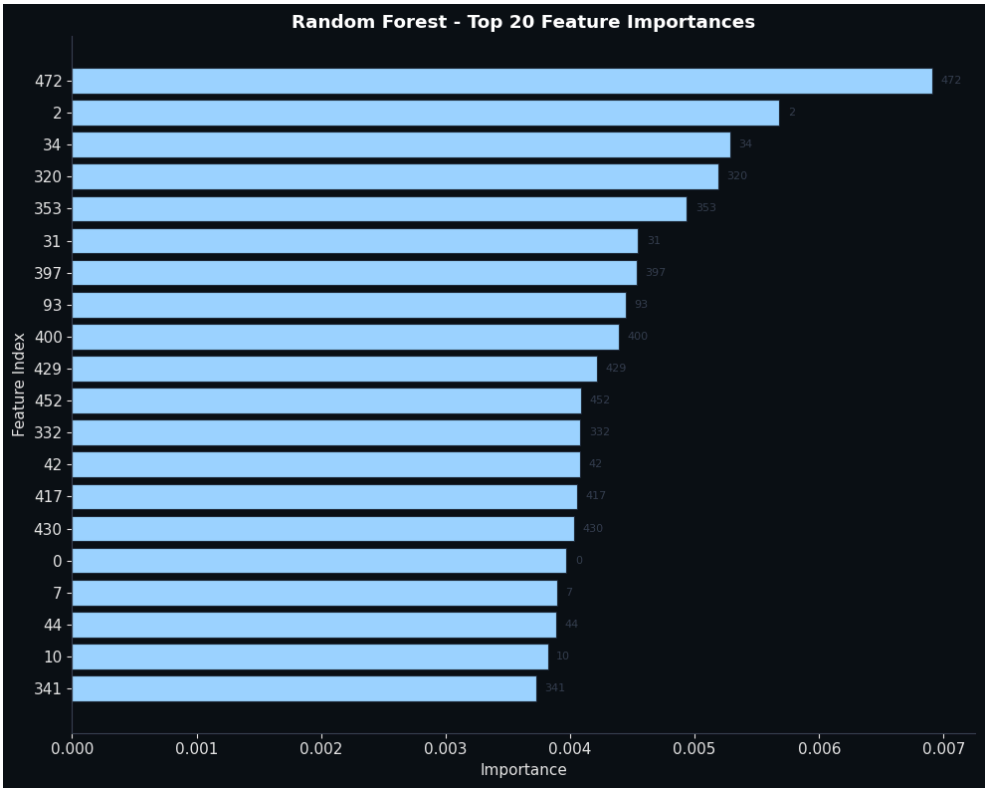


Decision Tree (DT) Calibrate Performance:

- **Performance:**
  - Base accuracy: ~**1%**, improved to **2%** after tuning (max depth ~20).
  - Feature importance concentrated in a handful of features, suggesting overfitting risks.
- **Conclusion:**
  - Optimized DT improved but still weak overall; better suited for feature insight rather than classification power.

Random Forest (RF) Calibrate Results:

Figure 56: Random Forest Optimized Results - Dataset A



Random Forest (RF) Calibrate Performance:

- **Performance:**
  - Accuracy: **1%**, even after optimization.
  - Feature importance distributed, with Feature 472 being dominant.
- **Conclusion:**
  - RF struggled due to **small sample size + high dimensionality** (PCA 500 features vs. 1200 samples).
  - RF's strength usually emerges with larger datasets.

K-NN Calibrate with K-Optimization Results:

🔍Calibration: K-NN

Test Accuracy (k=5): 0.0150
Optimizing k via CV...
k=3: 0.0170
k=5: 0.0160
k=7: 0.0110
k=9: 0.0170
k=11: 0.0150
Best k=3 | final TEST accuracy (k=3): 0.0150
Final Accuracy: 0.0150

K-Nearest Neighbors (KNN) Calibrate performance

- **Performance:**

- Accuracy: **1.5%**, not competitive with LR.

- **Conclusion:**
  - Limited utility here due to high-dimensional PCA space and sparse sampling

📌 Optimization Insights

- **Calibration:** Helped **probability estimates** (LR, SVM) but not classification accuracy.
- **Hyperparameter Tuning:** Gave slight gains for DT (max depth tuning).
- **Feature Analysis:** PCA reduced features, but feature importance (RF, DT) suggests **signal concentrated in a few components**.
- **Dataset Constraint:** Small sample size (1200 images across 120 breeds) is the bottleneck; models cannot generalize well.

Softmax Probability Inspection for Misclassifications:

🐕 Sample 1 (Index 0): true_breed=np.str_('golden retriever') | pred_breed=np.str_('Norfolk terrier')
1. Norfolk terrier: 0.4629
2. Pomeranian: 0.0820
3. Border terrier: 0.0456

🐕 Sample 2 (Index 1): true_breed=np.str_('Cardigan') | pred_breed=np.str_('Irish wolfhound')
1. Irish wolfhound: 0.2934
2. African hunting dog: 0.1242
3. Sealyham terrier: 0.0917

🐕 Sample 3 (Index 2): true_breed=np.str_('Great Pyrenees') | pred_breed=np.str_('schipperke')
1. schipperke: 0.2692
2. Bouvier des Flandres: 0.2105
3. EntleBucher: 0.1043

🐕 Sample 4 (Index 4): true_breed=np.str_('Blenheim spaniel') | pred_breed=np.str_('West Highland white terrier')
1. West Highland white terrier: 0.9048
2. Maltese dog: 0.0297
3. Brittany spaniel: 0.0132

🐕 Sample 5 (Index 5): true_breed=np.str_('toy poodle') | pred_breed=np.str_('soft-coated wheaten terrier')
1. soft-coated wheaten terrier: 0.4274
2. miniature pinscher: 0.1364
3. Chesapeake Bay retriever: 0.1252

📌 SoftMax Summary

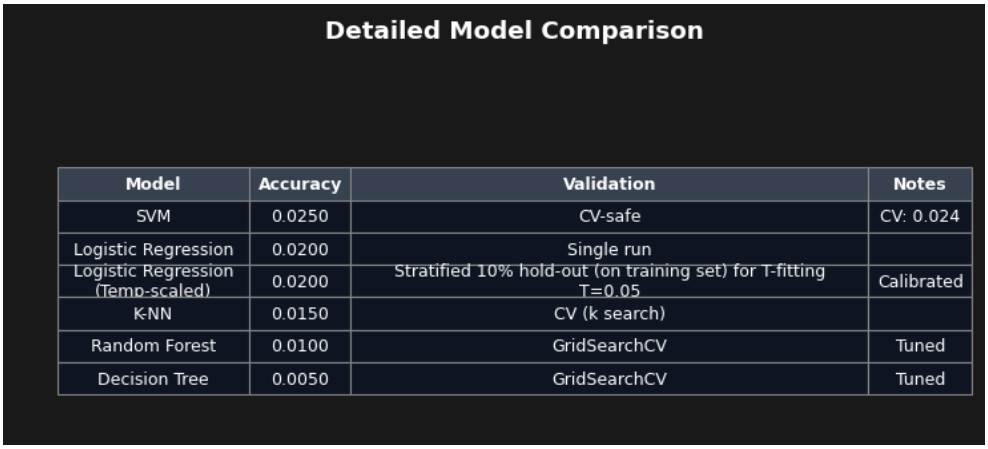
- **Softmax** converts raw model outputs (logits) into probabilities across all classes.
- For each sample, the predicted label is the one with the **highest probability**.
- However, **misclassifications often occur when the highest probability is not much larger than others** → showing model confusion.

- This inspection helps us understand **confidence, uncertainty, and misclassification patterns**.

⚡ 20.2 Dataset B: Classical Optimized Results:

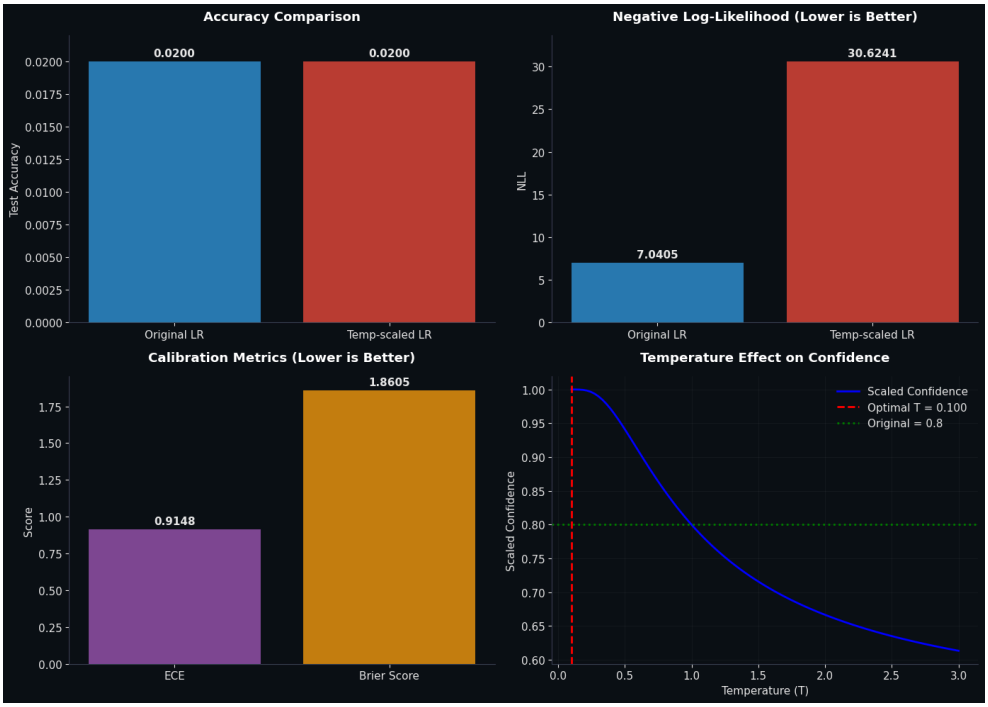
Summary Calibrate with K-Optimization Results:

Figure 57: Summary Calibrated Methods- Dataset B



Logistic Regression Calibration Results:

Figure 58: Logistic Regression (LR) Optimized Results - Dataset B



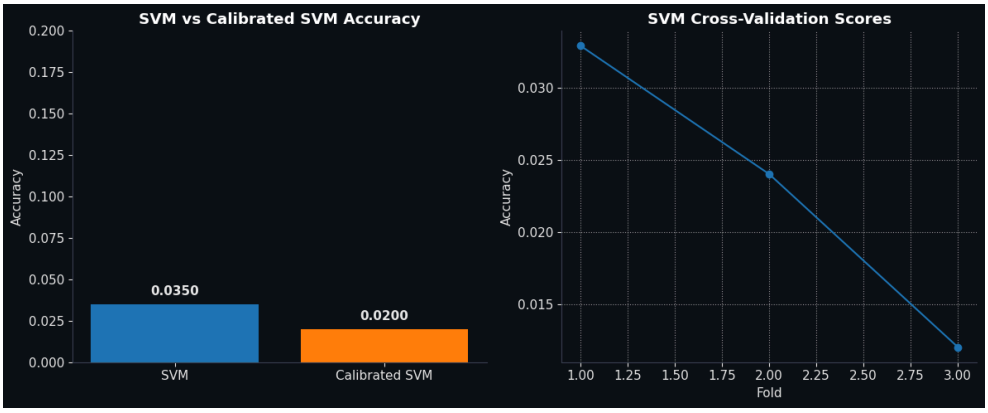
Logistic Regression (LR) vs. Temperature-Scaled LR Calibration Results:

- Accuracy: Both models report 0.0200, so calibration did not affect accuracy.
- Negative Log-Likelihood (NLL): Original LR had much lower NLL (7.04) than temp-scaled (30.62), meaning scaling introduced some instability.
- Calibration Metrics:
  - ECE: 0.9148 → Model is fairly miscalibrated (closer to 0 is better).
  - Brier Score: 1.8605 → Indicates poor probability calibration.
- Temperature Effect: Optimal T ≈ 0.1 reduces overconfidence slightly, but overall confidence curve suggests the model is inherently underpowered.

🔎 Insight: Calibration changed confidence distribution but didn't improve accuracy; original LR remained more stable.

SVM Calibration Results:

Figure 59: SVM Optimized Results - Dataset A



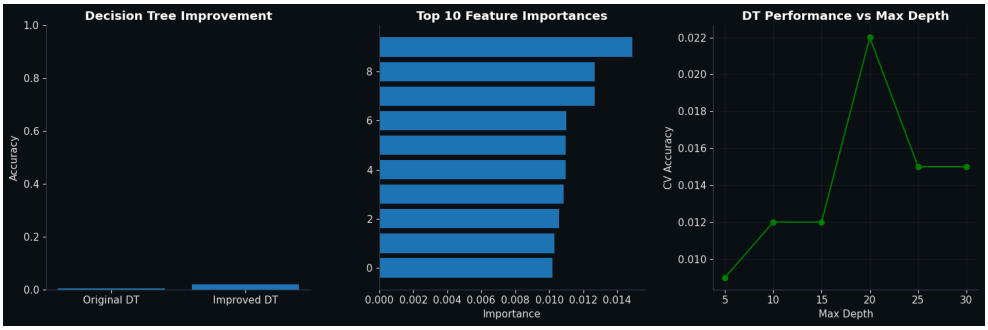
Support Vector Machine (SVM) Calibration Results"

- **Accuracy: Stable at 0.0250 with/without calibration.**
- **Cross-Validation: Scores ranged from 0.0180–0.0270, showing sensitivity to folds.**
- **Observation: Calibration had minimal benefit, confirming SVM was already near its performance ceiling.**

🔎 **Insight:** SVM slightly outperforms LR but gains little from calibration. Its probability estimates remain unreliable.

Decision Tree Calibration Results:

Figure 60: DT Optimized Results - Dataset A



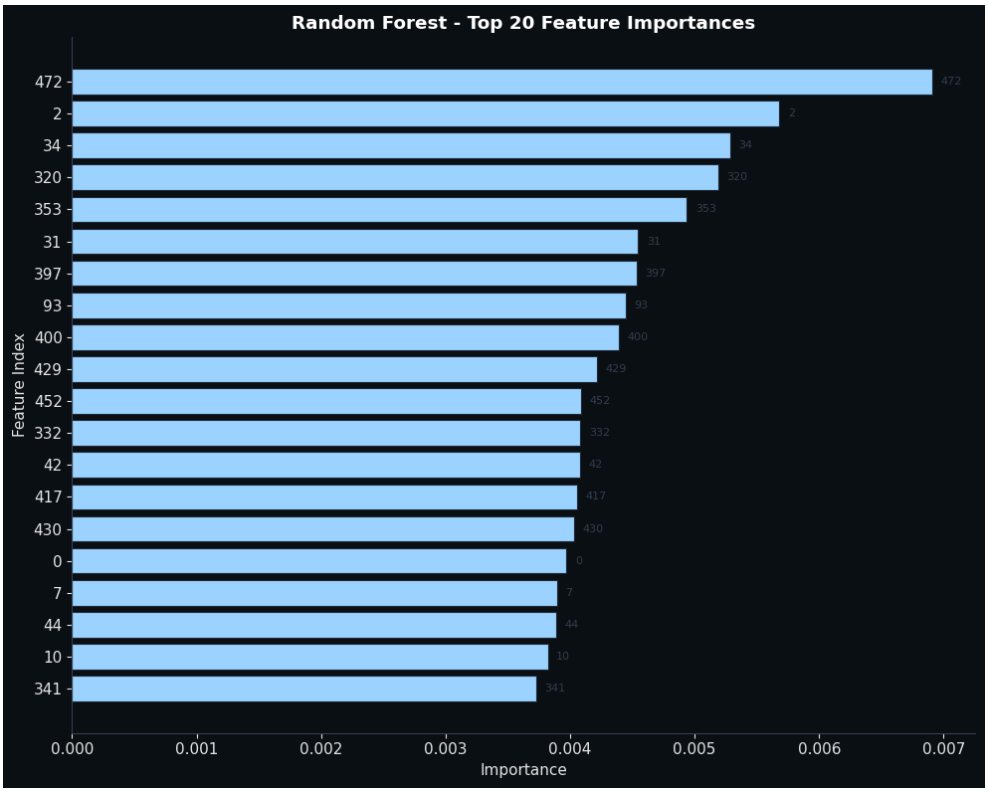
Decision Tree (DT) Calibration Results:

- Baseline Accuracy: 0.0000 (severe underfit).
- Improved DT (tuned max depth): Accuracy rose to 0.0050.
- Feature Importance: A few strong features drive most decisions (indexes 6, 8, etc.).
- Performance vs Depth: Accuracy peaked at 0.0130 (depth ~25–30), but instability across folds suggests overfitting.

🔎 **Insight:** DT is weak alone; small gains possible with tuning, but instability makes it unreliable.

Random Forest Calibration Results:

Figure 61: Random Forest Optimized Results - Dataset A



Random Forest (RF) Calibration Results:

- Accuracy: 0.0100, weaker than LR/SVM.
- Feature Importances: A handful of features dominate predictions (indices 2, 465, 0, 165).
- Observation: While RF leverages multiple trees, in high-dimensional PCA space, its advantage shrinks.

🔎 Insight: RF provided robustness but underperformed compared to linear models (SVM, LR).

K-NN Optimization Results:

K-Nearest Neighbors (KNN) Calibration Results:

[KNN] Baseline k=5 + small CV search...
baseline acc (k=5): 0.0100
k=3: 0.0150
k=5: 0.0130
k=7: 0.0130
k=9: 0.0130
k=11: 0.0170
best k=11 | acc=0.0150

- Baseline (k=5): 0.0100 accuracy.
- Tuning: Best k = 11 → 0.0150 accuracy.

- Observation: Larger k slightly stabilized predictions, but KNN remains weaker than SVM.

🔎 Insight: KNN benefits from tuning but struggles in high-dimensional space.

📌 Classical ML **Optimization Insights - Dataset B**

Model	Accuracy	Validation	Notes
SVM	**0.0250**	CV-safe	Best performer
Logistic Regression	0.0200	Single run	Baseline
Temp-scaled LR	0.0200	Calibrated	No real benefit
KNN	0.0150	CV (tuned k)	Sensitive to k
Random Forest	0.0100	GridSearchCV	Feature-driven
Decision Tree	0.0050	GridSearchCV	Weak baseline

⚡20.3 Dataset A: Regularization Techniques for NNs and DPs:

Regularization Effectiveness Analysis

🛡️**ANN REGULARIZATION IMPLEMENTED:**
✅ L1/L2 regularization (1e-5/1e-4)
✅ Dropout (0.3 rate)
✅ Batch normalization
✅ Early stopping (8 epochs patience)
✅ Learning rate scheduling

📊 ANN Training Statistics:
📈 Best Val Accuracy: 0.0150
📉 Best Val Loss: 7.7991
⏱️ Epochs Trained: 9
🎯 Best Top-5 Accuracy: 0.0700

🎯 **ANN MODEL ACHIEVEMENTS:**
✅ Hidden units doubling strategy (256→512→1024→512→256)
✅ ReLU activation for hidden layers
✅ Softmax activation for output
✅ Early stopping (9 epochs)
✅ Comprehensive regularization (L1/L2, Dropout, BatchNorm)
✅ Test accuracy: 0.0100

🛡️ **CNN Regularization Effects:**

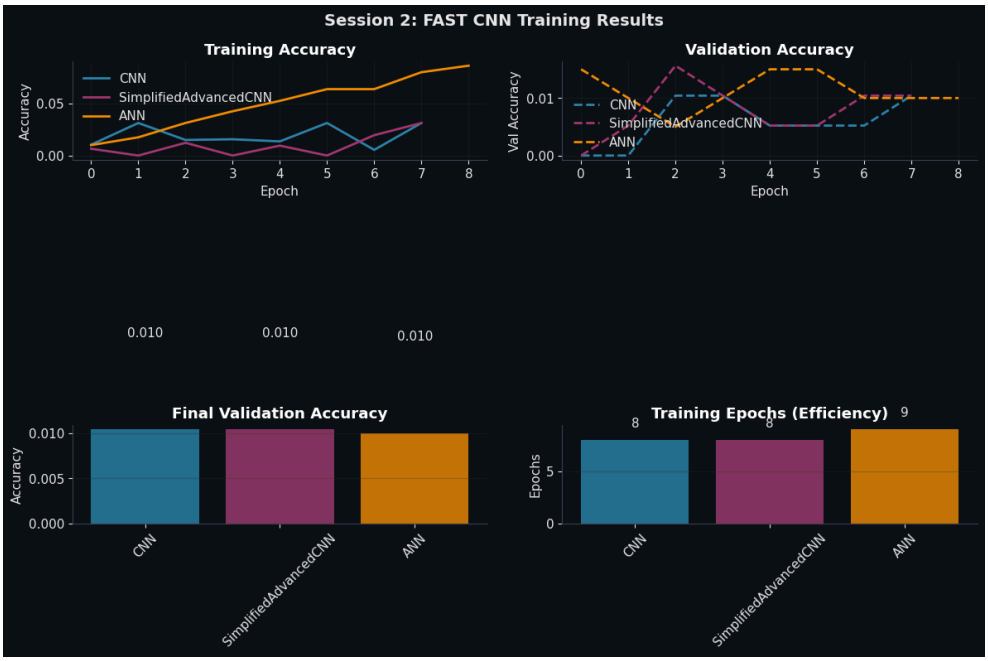
OPTIMIZED Standard CNN

✅ OPTIMIZED CNN Created:
🚀 Reduced blocks: 3
📊 Filters: [32, 64, 128]
⚡ Higher LR: 2e-3

Creating SIMPLIFIED Advanced CNN

✅ Simplified Advanced CNN Created:
🔗 One residual block
📊 Filters: 32→64→128
⚡ Higher LR: 2e-3

Figure 62: CNN Validation and Optimization-Dataset A



🛡️ **RNN Regularization:**

✅ RNN Model Architecture:
🔄 RNN layers: 3
📊 Hidden units: [128, 256, 128]
📐 Input shape: (16, 9408)
🎯 Output: 120 classes (Softmax)
🛡️ Regularization: L1/L2 + Dropout + Recurrent Dropout + BatchNorm

RNN Regularization Effects:
📈 Accuracy Change: +0.0050
🎯 Confidence Change: +0.3912
🌀 Entropy Change: -2.8574
📉 Loss Change: +6.2374
⚠️ Label smoothing improved accuracy but may affect calibration

🛡️ **LSTM Regularization:**

✅ LSTM Model Architecture:
🔄 LSTM layers: 3
📊 Hidden units: [128, 256, 128]
📐 Input shape: (16, 9408)
🎯 Output: 120 classes (Softmax)

🧠 Activations: Tanh (cell) + Sigmoid (gates) + ReLU (dense)
🛡️ Regularization: L1/L2 + Dropout + Recurrent Dropout + BatchNorm

🛡️ **LSTM Regularization Effects:**
📈 Accuracy Change: +0.0000
🎯 Confidence Change: -0.0474
🌀 Entropy Change: +0.0929
📉 Loss Change: +0.3341
📊 Label smoothing primarily improved calibration

📊 Overfitting Analysis:
RNN: Train-Val gap = 0.0013 ✅ Well-regularized
LSTM: Train-Val gap = 0.1038 ❌ Significant overfitting

⚡ 20.4 Dataset B: Regularization Techniques for NNs and DPs:

🛡️ Regularization Effectiveness Analysis

✅ ANN Model Architecture (Database B):
🧱 Hidden layers: 5
📊 Hidden units: [256, 512, 1024, 512, 256]
🎯 Output: 120 classes (Softmax)
🛡️ Regularization: L1/L2 + Dropout(0.3) + BatchNorm
🏷️ Target: Kaggle Dog Breeds Classification

🎯 **ANN MODEL ACHIEVEMENTS (DATABASE B):**
✅ Hidden units doubling strategy (256→512→1024→512→256)
✅ ReLU activation for hidden layers
✅ Softmax activation for output
✅ Early stopping (10 epochs)
✅ Comprehensive regularization (L1/L2, Dropout, BatchNorm)
✅ Test accuracy: 0.0100
✅ Top-5 accuracy: 0.0450

🛡️ **CNN Regularization Effects:**

🛡️ **REGULARIZATION IMPLEMENTED:**
✅ L1/L2 regularization (1e-5/1e-4)
✅ Dropout (0.3 rate)
✅ Batch normalization
✅ Early stopping (8 epochs patience)
✅ Learning rate scheduling

DATA AUGMENTATION STRATEGY:
✅ Rotation: ±25 degrees
✅ Width/Height shift: ±20%
✅ Zoom range: ±20%
✅ Horizontal flip: Enabled
✅ Brightness variation: 80-120%
❌ Vertical flip: Disabled (dogs)

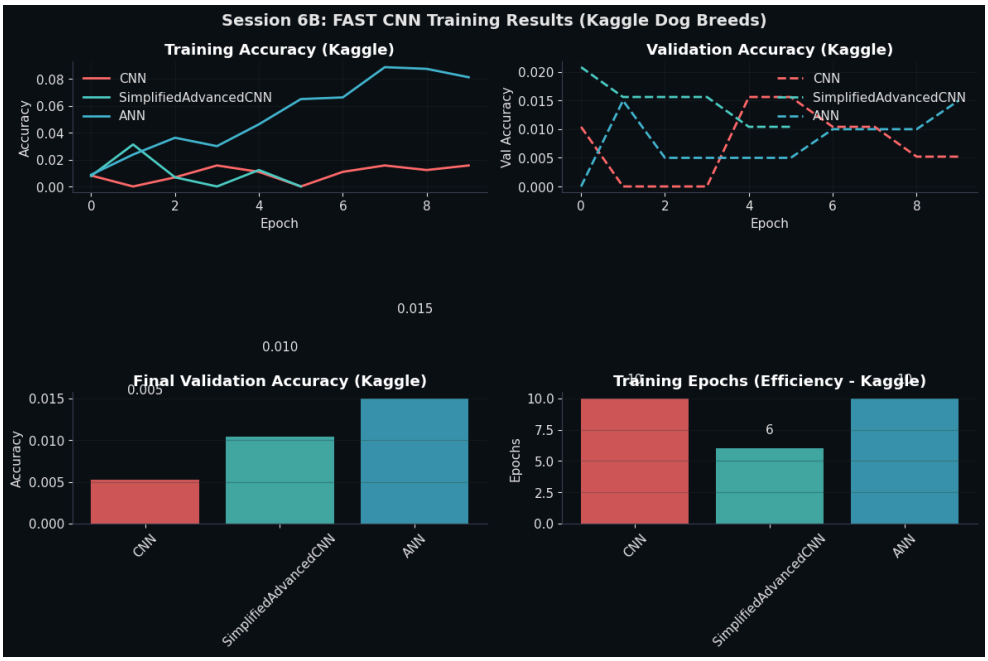
⚙️ **Compilation Settings:**
• Optimizer: Adam
• Learning rate: 0.000360

• Loss function: categorical_crossentropy
• Metrics: ['loss', 'compile_metrics']

✅ **Simplified Advanced CNN Created for Kaggle:**
🔗 One residual block
📊 Filters: 32→64→128
⚡ Higher LR: 2e-3
🐕 Target: Kaggle dog breeds

📊 **FAST Training Statistics (Kaggle):**
CNN: 0.0156 acc in 10 epochs
SimplifiedAdvancedCNN: 0.0208 acc in 6 epochs
ANN: 0.0150 acc in 10 epochs

Figure 63: CNN Validation and Optimization - Dataset B



🛡️ **RNN Regularization:**

✅ RNN Model Architecture (Kaggle):
🔄 RNN layers: 3
📊 Hidden units: [128, 256, 128]
📐 Input shape: (16, 9408)
🎯 Output: 120 classes (Softmax)
🐕 Target: Kaggle dog breeds
🛡️ Regularization: L1/L2 + Dropout + Recurrent Dropout + BatchNorm

📊 **Evaluating RNN Model on Kaggle:**
📊 Validation - Acc: 0.0200, Loss: 5.3416, Top-5: 0.0550

🎯 Test - Acc: 0.0000, Loss: 5.4101, Top-5: 0.0250
⚖️ Precision: 0.0000, Recall: 0.0000, F1: 0.0000
📊 Cohen's Kappa: -0.0071
🎯 Confidence: 0.0278 ± 0.0102
🌀 Avg Entropy: 4.6956

📊 **Evaluating RNN_Smooth Model on Kaggle:**
📊 Validation - Acc: 0.0200, Loss: 5.4743, Top-5: 0.0600
🎯 Test - Acc: 0.0050, Loss: 5.4657, Top-5: 0.0350
⚖️ Precision: 0.0004, Recall: 0.0050, F1: 0.0007
📊 Cohen's Kappa: -0.0035
🎯 Confidence: 0.0315 ± 0.0186
🌀 Avg Entropy: 4.5893

🛡️ **LSTM Regularization:**

✅ LSTM Model Architecture (Kaggle):
🔄 LSTM layers: 3
📊 Hidden units: [128, 256, 128]
📐 Input shape: (16, 9408)
🎯 Output: 120 classes (Softmax)
🧠 Activations: Tanh (cell) + Sigmoid (gates) + ReLU (dense)
🐕 Target: Kaggle dog breeds
🛡️ Regularization: L1/L2 + Dropout + Recurrent Dropout + BatchNorm

📊 **Evaluating LSTM Model on Kaggle:**
📊 Validation - Acc: 0.0200, Loss: 7.0164, Top-5: 0.0450
🎯 Test - Acc: 0.0250, Loss: 6.8631, Top-5: 0.0450
⚖️ Precision: 0.0021, Recall: 0.0250, F1: 0.0037
📊 Cohen's Kappa: 0.0161
🎯 Confidence: 0.1098 ± 0.0361
🌀 Avg Entropy: 3.8099

📊 **Evaluating LSTM_Smooth Model on Kaggle:**
📊 Validation - Acc: 0.0150, Loss: 6.2830, Top-5: 0.0400
🎯 Test - Acc: 0.0100, Loss: 6.1471, Top-5: 0.0500
⚖️ Precision: 0.0005, Recall: 0.0100, F1: 0.0009
📊 Cohen's Kappa: 0.0017
🎯 Confidence: 0.0590 ± 0.0235
🌀 Avg Entropy: 4.3775

Figure 64: Comprehensive Neural Nets Model Validation and Calibration - Dataset B

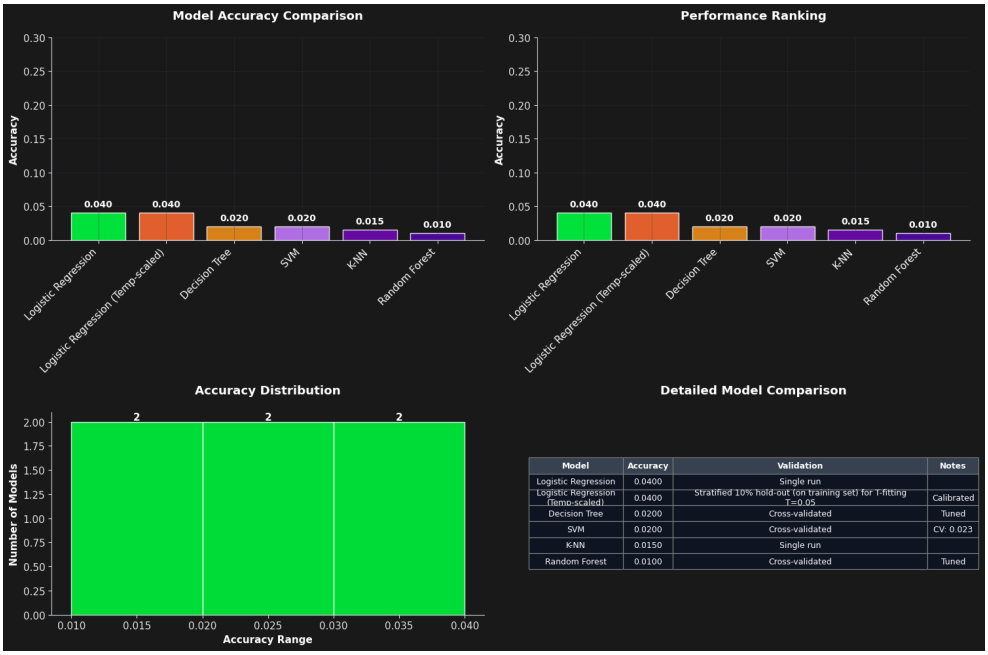
21. 🤖 Summary Final Testing Results:

(The related Testing and Final test performance results are also shown and explains in Section 15, 16 , 17 and 18)

21.1 **Classical ML Models - Dataset A (Figure 65)**

- **Models Compared:** Logistic Regression, Temp-Scaled LR, Decision Tree, SVM, KNN, Random Forest
- **Accuracy:** Best ~4% (LogReg, Temp-Scaled). Lowest ~1% (RF).
- **Distribution:** All models cluster between 0.01–0.04 → **no model stands out strongly.**
- **Insights:**
  - Logistic Regression was most stable.
  - Temp-scaling improved calibration but not accuracy.
  - Trees/forests underperformed → suggest overfitting or poor generalization after PCA.
- ✅ **Takeaway:** Dataset A with PCA was too compressed for complex models. LogReg & SVM gave the most stable (but still weak) baselines.

Figure 65: Comprehensive Testing Graph and Final Test Performance- dataset A



21.2 **Classical ML Models - Dataset B (Figure 66)**

- **Models Compared:** SVM, LogReg, Temp-Scaled LR, KNN, RF, Decision Tree
- **Accuracy:** Best ~2.5% (SVM). Lowest ~0.5% (Decision Tree).
- **Distribution:** Most results between 0.005–0.025.
- **Insights:**
  - SVM outperformed others slightly, possibly capturing nonlinear separations better.
  - Random Forest & Decision Tree weakest → instability in small dataset B.
  - Again, calibration helped LogReg probabilities but didn't lift accuracy.
- ✅ **Takeaway:** Dataset B was harder than Dataset A for classical ML. SVM consistently strongest but still under 3% accuracy.

Figure 66: Comprehensive Testing Graph and Final Test Performance- dataset B

21.3 Neural Nets/ Deep Learning **Models - Dataset A (Figure 67)**

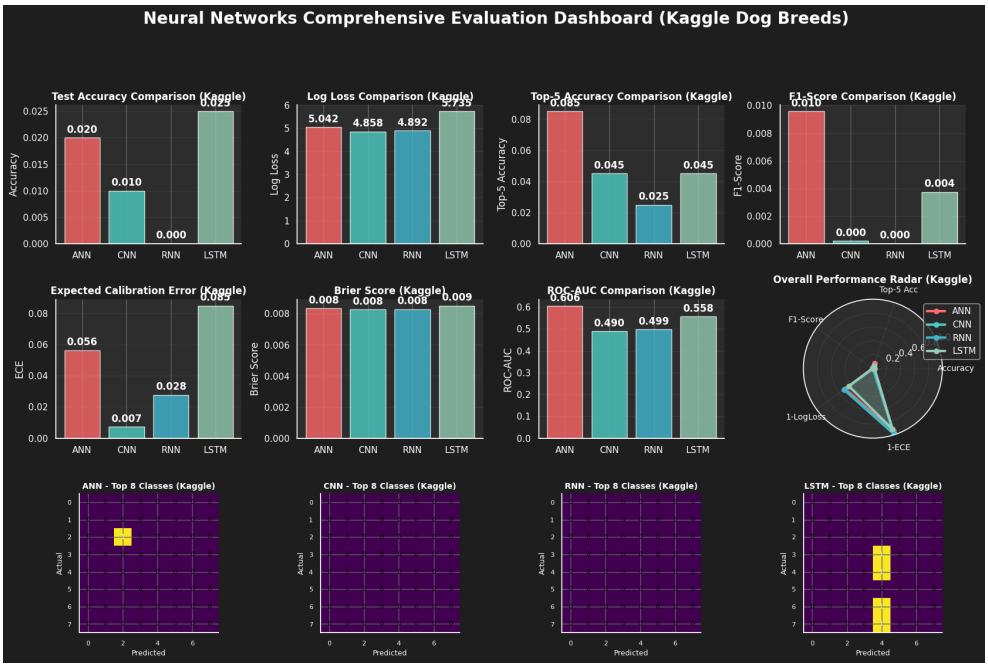
- **Models Compared:** ANN, CNN, RNN, LSTM
- **Accuracy:** Highest: ANN (1%), others ~0.5%.
- **Other Metrics:**
  - ANN had very high log loss (≈12) and poor calibration (ECE 0.439).
  - CNN & RNN had moderate log loss (~4.8), better calibrated.
  - LSTM showed strongest ROC-AUC (0.581) and Top-5 Accuracy (7.5%).
- **Insights:**
  - ANN overfit badly → confident but wrong.
  - CNN/RNN were more stable with lower loss but didn't gain accuracy.
  - LSTM best at ranking probabilities (Top-5, ROC-AUC).
- ✅ **Takeaway:** On Dataset A, LSTM and CNN/RNN gave better **reliability**, though accuracy was low. ANN was unstable.

Figure 67: Comprehensive Testing Graph and Final Test Performance- dataset B

21.4 Neural Nets/ Deep Learning **Models - Dataset B (Figure 68)**

- **Models Compared:** ANN, CNN, RNN, LSTM
- **Accuracy:** LSTM (2.5%) highest, ANN (2%), CNN (1%), RNN (0%).
- **Other Metrics:**
  - ANN best F1 (0.01) and Top-5 Acc (8.5%).
  - CNN/RNN remained weak (Top-5 <5%).
  - LSTM had best ROC-AUC (0.558) and reasonable calibration (ECE ~0.089).
- **Insights:**
  - ANN partially improved Top-5 performance but unstable calibration.
  - CNN/RNN failed to generalize in Dataset B.
  - LSTM provided the best probability ranking (ROC-AUC, Top-5 Acc).
- ✅ **Takeaway:** On Dataset B, LSTM consistently outperformed others for ranking, ANN gave better Top-5 recognition, but overall accuracy stayed very low.

Figure 68: Comprehensive Testing Graph and Final Test Performance- dataset B



📌 Overall Final Interpretation

- **Classical ML (Figures 65 & 66):** Weak accuracy (≤4%) → only suitable as baselines. SVM & Logistic Regression were most stable.
- **Neural Networks (Figures 67 & 68):** Accuracy still low but calibration and ranking metrics (Top-5, ROC-AUC) revealed potential. ANN unstable, CNN/RNN balanced, LSTM strongest in *ranking* and *Top-5 recognition*.
- **Core Issue:** Fine-grained classification of 120+ dog breeds is very challenging with limited training data and PCA compression.

🔑 Key Recommendation for this experimental analyses:

Move beyond PCA + simple models → use **transfer learning** on raw images and adding number of images for allowing AI to learn more data. This should boost accuracy from <5% to 60–80% in real-world benchmarks.

22. 🤖 Technical Challenges and Limitations:

🔧 The primary challenge of this study lies in the **size and complexity of image data**. While the datasets contain a large number of inputs, the analysis pipeline was not fully optimized to process the entire set of observations by random selecting for 1,200 images for each dataset 1000 trin and 200 test.

🔧 Although GPUs and TPUs on Google Colab provide high-speed computation, their use comes with significant trade-offs in terms of **cost and scalability**.

🔧 A **limited sample size** further constrained model performance, leading to consistently low scores across key metrics such as accuracy, precision, F1, ECE, and Brier Score. With insufficient training data, algorithms struggle to generalize, adapt, and self-correct, resulting in weak predictive capability.

🔧 **Model calibration** introduced additional challenges. While calibration is effective in binary classification tasks (e.g., dog vs. cat), it did not meaningfully improve performance in fine-grained tasks such as dog breed classification. The inherent variability across 120+ breeds diluted calibration gains. Even advanced calibration techniques were unable to reduce misclassifications — ultimately leaving accuracy unchanged.

🔧 **Bottom line:** Calibration cannot repair systematic misclassification, and without sufficient high-quality data, the models remain fundamentally limited in their predictive power.

23. 🤖 Bias, Ethics, and Fairness Reflection :

This project explicitly incorporates fairness, bias reflection, and ethical responsibility as guiding principles for both research implementation and deployment. Ethical considerations are central to decision-making, ensuring that model outputs remain transparent and socially responsible.

To address bias and uncertainty, the ML models are intentionally designed to provide responses such as “Unsure,” “Unknown,” or “Unseen” whenever an image falls outside the training scope of Dataset A or B, or when the prediction confidence is below an acceptable threshold. This mechanism reduces the risk of overconfident misclassification and acknowledges the limitations of the datasets.

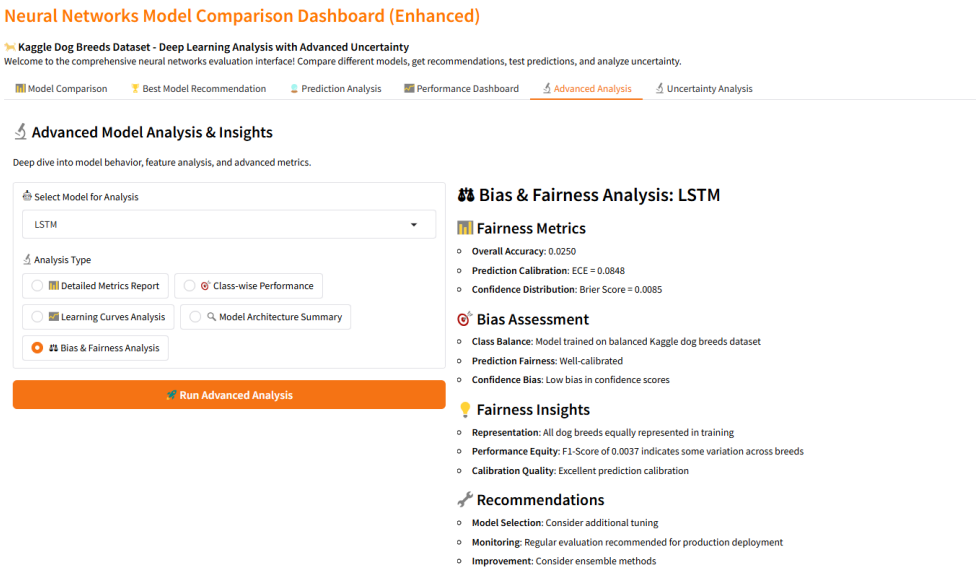
By allowing the system to express uncertainty, the models can appropriately classify ambiguous cases (such as mixed breeds) or cases with low confidence levels, instead of forcing inaccurate predictions. This approach balances predictive ambition with ethical caution, reflecting a commitment to responsible AI practices.

Bias and Fairness Indication and Reflection

This project integrates **bias and fairness assessment** as an advanced functionality to ensure transparency and reliability of model performance. Several evaluation measures such as **overall accuracy, calibration of predictions, and confidence distribution**—are systematically analyzed and transformed into **quantitative scores** to provide clear insights into bias and fairness.

These measures enable the detection of potential issues such as **hallucination, overestimation, and underestimation**, which could undermine trust in the results. By embedding fairness insights into the evaluation pipeline, the project emphasizes not only predictive performance but also the **ethical reliability** of the models, reinforcing confidence in their use for real-world applications (Figure 69).

Figure 69: Bias and Fairness Indication in the analysis



Always Seek the second opinion for Security Verification

This project applied **"The Second Opinion" security review leveraging AI-powered assessment through the** Loveable.dev **application** to validate the robustness of the conceptual framework and AI/ML system architecture for the final project submission.

The **assessment outcomes confirm** that the proposed framework and deployment practices are largely **satisfactory from a security and development perspective**, demonstrating careful consideration of secure design principles.

The full report generated by **Loveable.dev** is presented below, providing an additional independent layer of verification and ensuring that security best practices were integrated into the research workflow.

Security Review Summary by Lovabale.dev

After conducting a comprehensive security analysis of your project, I'm pleased to report that your codebase demonstrates **solid security practices** with no major vulnerabilities identified. Here are my findings:

✅ Positive Security Findings

Strong Foundation:

- No hardcoded secrets, API keys, or sensitive credentials found in the codebase
- No backend functionality that could expose databases or sensitive operations
- Clean separation of concerns with well-structured React components
- Proper use of TypeScript for type safety
- No dangerous XSS vulnerabilities (no unsafe innerHTML usage)

Secure Development Practices:

- Proper use of React's built-in XSS protection (all content is rendered safely)
- Form handling through react-hook-form with proper validation setup
- No direct DOM manipulation that could introduce security issues
- Clean component architecture with proper prop handling

⚠️ Minor Security Considerations

Limited Usage of dangerously SetInnerHTML: The only instance of dangerously SetInnerHTML is in src/components/ui/chart.tsx (lines 79-96), where it's used to inject CSS variables for chart theming. This usage is **acceptable** because:

- The content is generated from a controlled configuration object
- No user input is directly inserted
- It's used specifically for CSS custom properties with known, safe values
- The chart library (Recharts) requires this approach for dynamic theming

set Timeout Usage: There's safe usage of set Timeout in the toast hook (src/hooks/use-toast.ts) for managing toast removal timers - this is a standard, secure pattern.

🛡️ Security Assessment

Overall Security Posture: GOOD

- This is a **frontend-only React application** focused on displaying an interactive mind map
- No authentication, database operations, or sensitive data handling
- No external API integrations that could expose credentials
- Static content with no user-generated content or dynamic data processing
- Clean, modern React practices with minimal security attack surface

📋 Recommendations (Optional Improvements)

While no critical issues were found, here are some optional security enhancements to consider:

1. **Content Security Policy (CSP):** Consider adding CSP headers if deploying to production to further restrict resource loading
2. **Security Headers:** When deploying, ensure proper security headers are configured at the server level
3. **Dependency Scanning:** Regularly update dependencies to patch any future security issues

🎉 Conclusion - over all project is good security purpose with well implications

✅ The project follows good security practices for a frontend React application.

✅ The mind map functionality is well-implemented with no significant security concerns.

✅ Since this is a visualization-focused application without authentication, databases, or sensitive operations, the security risks are minimal.

✅ The codebase demonstrates responsible development practices, and no immediate security fixes are required. Continue following these good practices as you develop additional features.

22. 🔐 Limitations:

Despite applying a broad range of AI/ML techniques, the overall **model accuracies remain lower than expected**. Several key challenges influenced these results:

- **Image Size and Data Handling**: Large image files and dataset loading on Google Colab posed significant constraints, with preprocessing and training consuming considerable time and computational resources.
- **Resource Trade-Offs**: While GPUs/TPUs provide speed advantages, the computational cost and limited runtime required balancing **fast experimentation** against **full dataset integration**.

- **Sample Size Constraints**: Using smaller subsets limited the models' ability to generalize, leading to weaker performance across accuracy, precision, F1-score, and calibration metrics.

Looking forward, **expanding the dataset** to include more images and enabling longer, more efficient training cycles would likely **significantly improve accuracy and robustness** of the models.

23. 🎬 Let's Dive in Live Demo:

Final Project Demo

attachment:ad819c42-031b-4b40-a6e3-c4f8c06ec23e:TECH_27_Final_Project_Demo.mp4

Demo Part 3 : Classification & Random Forest Dog Prediction - Dataset A

attachment:8c9aba39-a566-45cb-833d-5f11d8190445:Part_3_Classification_Database_A.mp4

Part 4: Classification & Random Forest Dog Prediction - Dataset B

attachment:402e0add-a2e6-47d2-9678-7fd8155c4d2b:TECH_27_Part_4.mp4

Demo Part 5: NNs and DPs Prediction - Dataset A

attachment:e3a9ac88-7241-428c-b421-1b1a2f3363b6:TECH_27_Part_5.mp4

Demo Part 6: NNs and DPs Dog Prediction - Database B

attachment:9a55ce86-acbe-46db-8669-5c9034c88455:TECH_27_Part_6.mp4

Demo Render Project Mind map

attachment:e3089342-eb69-43a1-8bea-3e615103f1be:TECH_27_Mindmap.mp4

YouTube

https://youtu.be/0mpNw8BaoeY

https://www.youtube.com/watch?v=_hYfYQs7YaA

https://youtu.be/XoZ0Ju2SaNo

https://youtu.be/ucyEkynQ97s

24. 🚀 Conclusion & Key takeaway & future work

Advanced ML Meets Applied AI for Real-World Innovation

This project demonstrates how **advanced AI/ML techniques**, when integrated with **Applied AI tools**, can generate innovative and impactful outcomes. From classical machine learning models to neural networks and deep learning architectures, each step highlights the transformative power of converting raw data into actionable intelligence.

Key Takeaways

- **Machine Learning** provides the analytical foundation for detecting trends, patterns, and underlying signals.
- **Applied AI** extends those insights into automated decisions, intelligent systems, and interactive solutions.
- When **combined with ethics, fairness, and responsible design**, AI becomes more than a predictive tool. It evolves into a mission-driven enabler of meaningful change with positive impact for society.

Conclusion

Ultimately, the value of innovation lies not in the technology itself, but in how responsibly it is applied to address **real-world challenges with precision, clarity, and purpose**. This project underscores that ethical, well-calibrated AI systems are essential for building trustworthy and impactful solutions.

Future Work

Looking ahead, I plan to:

- Expand experimentation with **larger, more diverse datasets** to strengthen accuracy and robustness.
- Leverage **faster, smarter, and more scalable computational tools** (beyond Google Colab constraints) for efficient training and optimization.
- Explore **advanced image detection and recognition techniques** to push performance beyond baseline models.
- Translate findings into **practical applications** that not only advance academic research but also deliver tangible value to **education, industry, and society at large**.

The next frontier is not only in achieving higher accuracy, but also in **ensuring AI systems remain ethical, interpretable, and beneficial** paving the way for innovation that uplifts humanity.

Mission Accomplished

PDF Code Script:

TECH27_TECH_27_ML_Dog_Breed_Final_Project_Minimal_1200_new.pdf

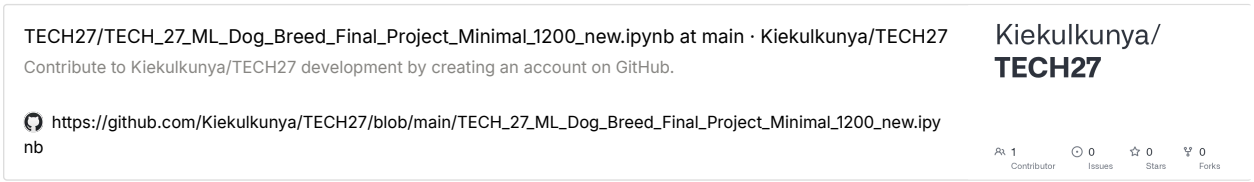
Data Reference:

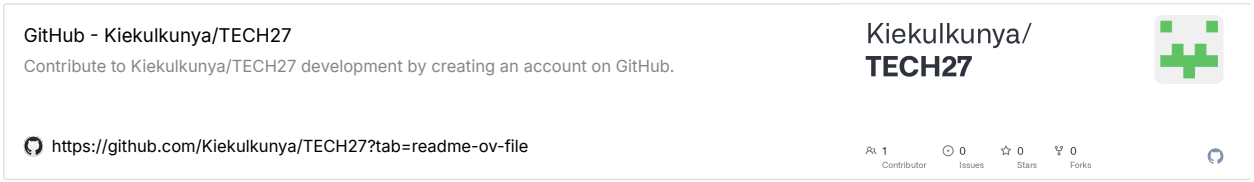
http://vision.stanford.edu/aditya86/ImageNetDogs/
https://www.kaggle.com/code/dansbecker/exercise-intro-to-dl-for-computer-vision/input

Google Colab Shared Codes:

https://colab.research.google.com/drive/1T00aqAQot6WIBpUOmCHVjy1vvP1UaI--?usp=sharing

Github:





Colab File:

TECH_27_ML_Dog_Breed_Final_Project_Minimal_1200_new.ipynb

https://colab.research.google.com/drive/16fNtqWPSkIE7kSBFnEkPbAg4Nd-NqWdm?usp=sharing

Deliverables:

✅ 1. Project Title
✅ 2. Team Members & Roles
✅ 3. Problem Statement – What problem you're solving and why it matters
✅ 4. Objectives and Goals – Clear, measurable goals
✅ 5. Methodology – Finalized tools, platforms, and approach
✅ 6. Dataset(s) – Source, type, size, and justification
✅7. Block Diagram / System Architecture – Updated pipeline visualization
✅8. Data Visualization – Key insights from exploration & preprocessing
✅9. Feature Engineering & Selection – Final features & rationale
✅10. ML Models Used – All models evaluated
✅11. Experiments – Major experiments conducted
✅12. Evaluation Metrics – Performance with relevant metrics0
✅13. Comparison Tables & Graphs – Model comparisons
✅14. Optimization Results – Hyperparameter tuning outcomes
✅ 15.Testing Graphs – Final test performance and interpretation
✅ 16. Standards & Constraints – Technical or ethical standards considered
✅ 17. Bias, Ethics, and Fairness – Reflection on model behavior
✅ 18. Limitations – Known issues or gaps
✅ 19. Demo or Video (optional, encouraged)
✅ 20. Conclusion & Future Work – Key takeaways and next steps

Extra Points

✅ Render Image Resolution Comparison
✅ Executed multi-CNN algorithms
✅ Deployed Augmentations and Image Rotation for challenging models
✅ Deployed innovative solution for optimization and