Animal Image Classifier using Traditional ML

Internship Project Documentation

---

1. Objective

The goal of this project is to classify images of animals (primarily dogs and cats) using **traditional machine learning** techniques like SVM,Random Forest ,and other models .The emphasis is on handcrafted feature extraction rather than transfer learning or deep learning, aligning with real-world constraints like limited RAM and compute resources.

---

2. Dataset

- **Source**: Kaggle
- **Dataset Name:** `salader/dogs-vs-cats`
- **Description:** Contains thousands of images in JPG format representing cats and dogs in various lighting conditions and sizes.

Download Code

```
import kagglehub
path = kagglehub.dataset_download("salader/dogs-vs-cats")
print("Path to dataset files:", path)
```

Dataset Composition

- Cats and Dogs in `.jpg`, `.jpeg`, and `.png` formats
- Varying sizes and image quality

---

3. Folder Structure

```
AnimalPlant_Classifier/
|
|-- data/
|   |-- animals/
|       |-- cat/
|       |-- dog/
|
|-- models/                        # Contains .pkl files
|   |-- knn.pkl
|   |-- naive_bayes.pkl
|   |-- decision_tree.pkl
|   |-- random_forest.pkl
|   |-- cnn_model.h5
|
|-- test_images/                   # Images for evaluation
|-- streamlit_app.py               # Streamlit app
|-- main_classifier.py             # ML model training
|-- advanced_models.py             # SVM & Logistic Regression
|-- predict.py                     # Prediction from test images
|-- README.md
|-- requirements.txt
|-- streamlit_preview1.png
|-- streamlit_preview2.png
```

---

4. ❓ Problem Statement

Due to **RAM limitations** in Google Colab (12GB), training compute-heavy models like SVM, Random Forest, or XGBoost often led to session crashes. Thus, we explored simpler ML algorithms and extracted handcrafted features for classification.

Traditional ML Workflow

1. **Loading Dataset**
2. **Preprocessing & Feature Extraction**:
   - Image resizing
   - Grayscale conversion
   - Histogram of Oriented Gradients (HOG)
   - Pixel flattening
3. **Model Training**:
   - K-Nearest Neighbors (KNN)
   - Gaussian Naive Bayes
   - Decision Tree
   - Logistic Regression
   - Support Vector Machine (SVM)
   - Random Forest
4. **Model Evaluation**
   - Confusion Matrix
   - Accuracy Score
   - Classification Report

CNN Architecture (TensorFlow)

- Input: 128x128x3 images
- Conv2D + MaxPooling

- Dense Layers + Dropout
- Softmax output
- Achieved **90% accuracy**

---

5. 🔢 Model Performance

Model	Accuracy
Logistic Regression	51.50%
SVM	53.50%
K-Nearest Neighbors	57.00%
Decision Tree	54.00%
Gaussian Naive Bayes	56.00%
Random Forest	**62.00%** ✅
CNN (TensorFlow)	90.00%

---

6. 📈 Streamlit UI

A Streamlit web interface was developed for real-time image classification.

Features:

- Upload image
- Run model (Random Forest)
- Output: `It is a cat/dog`

Why Random Forest?

Although all models performed fairly well, **Random Forest** achieved the highest accuracy among traditional models.

Screenshot:

---

7. 🚧 Challenges Faced

- Google Colab RAM limitations (12GB)
- Session crashes with XGBoost, SVM, RF
- Needed to switch to lightweight models
- Managed to run CNN locally/GPU with better performance

---

8. 🪜 Tools & Libraries Used

- Python 3.x
- scikit-learn
- OpenCV
- Streamlit
- Seaborn, Matplotlib

---

9. 🚀 Future Work

- Explore feature extraction using SIFT/SURF
- Use cloud GPU platforms (AWS/GCP/Azure)
- Train Transfer Learning models like ResNet, VGG, etc.
- Deploy on HuggingFace/Gradio for interactive demos

---

10. 📁 Additional Scripts

- `main_classifier.py`: Trains KNN, RF, GNB, DT
- `advanced_models.py`: Trains SVM and Logistic Regression
- `predict.py`: Predicts image class from `.pkl` models

👤 Author

Vaishnavi Badjate
Data Science Intern
