Project Title: Animal Image Classifier using Traditional Machine Learning and CNN

Internship Project Documentation

---

1. Objective

The objective of this project is to **classify images into multiple categories**—specifically cats vs. dogs—using **traditional machine learning algorithms** rather than deep learning or transfer learning. The project explores feature extraction techniques, model training, and evaluation of performance using algorithms like **KNN, Gaussian Naive Bayes, and Decision Tree**, with additional experimentation using **CNN (TensorFlow)** to compare model performance.

---

2. Dataset

- **Source**: KaggleHub
- **Dataset Name**: `salader/dogs-vs-cats`
- **Description**: The dataset contains thousands of images of dogs and cats in `.jpg` format.
- **Download Code**:

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

Dataset Composition:

- Images of cats and dogs
- Format: JPEG, JPG, PNG
- Varying image sizes and lighting conditions

---

3. Folder Structure

```
AnimalPlant_Classifier/
|
|-- data/
|   |-- animals/
|       |-- cat/
|       |-- dog/
```

```
|
|-- models/
|   |-- knn.pkl
|   |-- naive_bayes.pkl
|   |-- decision_tree.pkl
|   |-- cnn_model.h5
|
|-- predict.py
|-- app.py
|-- README.md
|-- documentation.pdf / doc.txt
```

---

4. Problem Statement

Due to RAM limitations (12GB) in Google Colab, applying computationally heavy models like SVM, Random Forest, XGBoost led to session crashes. Therefore, simpler models were explored, aligning with the objective of using traditional ML algorithms.

Step 1: Loading the dataset

We used here Kaggle dataset

Step 2: Feature Extraction

To apply traditional ML, handcrafted features were extracted using:

- **Image resizing**
- **Conversion to grayscale**
- **Histogram of Oriented Gradients (HOG)**
- **Flattening pixel arrays**

Step 3: Splitting into training and test sets

- **Step 4**: Training models:
    - K-Nearest Neighbors (KNN)
    - Naive Bayes (GaussianNB)
    - Decision Tree
    - (SVM, Logistic Regression, XGBoost were tried but caused memory issues in Google Colab)

- **Step 5**: Evaluate performance using accuracy, confusion matrix, and classification report

4.2 Deep Learning Model - CNN (TensorFlow)

- Input layer: 128x128x3 images
- Convolutional layers + MaxPooling
- Dense layers + Dropout
- Output layer with Softmax
- Trained on GPU locally or on Colab (when feasible)
- Achieved **90% Accuracy**

---

5. Results

Model	Accuracy
Naive Bayes	59%
KNN	55%
Decision Tree	53%
CNN (TensorFlow)	**90%**

---

6. Challenges Faced

- Google Colab session crashed while using heavy ML models (SVM, RF, XGBoost)
- Limited to using lightweight models like Naive Bayes, KNN, etc.
- CNN was successfully implemented with good performance

---

7. Tools & Libraries Used

- Python
- scikit-learn
- OpenCV
- TensorFlow
- Matplotlib / Seaborn
- Streamlit (optional for frontend)

---

8. Deployment

- A Streamlit UI was created for local execution using `app.py`
- Used `predict.py` to make predictions directly without UI

---

9. Conclusion

- Traditional ML with handcrafted features gives decent results
- CNN clearly outperforms traditional models in terms of accuracy
- Resource limitations like RAM and GPU affect model selection

---

10. Future Enhancements

- Try advanced feature extraction like SIFT or SURF
- Implement transfer learning models like ResNet or VGG if memory allows
- Use cloud-based solutions for large-scale training

Author

Vaishnavi Badjate Data Science Intern