Facial Expression Recognition via Logistic Regression and K-Nearest Neighbors

---

1. Introduction

In this project, we'll classify facial expressions into five categories using two machine learning algorithms:

• Logistic Regression

• K-Nearest Neighbors (KNN) Classifier

We will use a publicly available dataset, preprocess the images, extract relevant features, and evaluate the models based on various classification metrics.

---

2. Dataset Description

General Information

- DataSet Name: Facial Expression Recognition Challenge (FER 2013)
- Source: Kaggle Competition - Challenges in Representation Learning: Facial Expression Recognition Challenge

Data Summary

- Total Number of Samples in Dataset: 28,709 images
- Classes and Their Labels:

Label	Emotion
0	Angry
1	Disgust
2	Fear
3	Happy
4	Sad
5	Surprise
6	Neutral

Selected Classes for This Project

We removed the classes labeled as 0 for Angry and 1 for Disgust to fulfill the project requirement of using only up to 5 classes.

- Classes Used:

Label	Emotion
2	Fear
3	Happy
4	Sad
5	Surprise
6	Neutral

Image Specifications

- Image Size: Original images are 48x48 pixels in grayscale.
- Processed Image Size: Resized to 64x64 pixels for feature extraction.

Data Split

- Total Processed Images: 22,063
- Training Set Size: 17,650 images
- Testing Set Size: 4,413 images
- Validation Set: Not used in this project.

---

3. Data Preprocessing

We performed several preprocessing steps to prepare the data for modeling.

Filtering Classes

We removed the classes with labels 0 and 1.

```
# Exclude classes 0 and 1
df = df[df['emotion']!= 0]
df = df[df['emotion']!= 1]
# Check unique emotion labels
df["emotion"].unique()
Output:
array([2, 4, 6, 3, 5])
```

No missing values were found in the dataset.

---

4. Exploratory Data Analysis

Visualizing Sample Images

We visualized a subset of images to get an understanding of the data.

```
emotion_dict = {2: 'Fear', 3: 'Happy', 4: 'Sad', 5: 'Surprise', 6: 'Neutral'}
fig = plt.figure(figsize=(8, 8))
rows = 3
columns = 3
 pixels = np.fromstring(subset_df['pixels'].iloc[i], sep=' ')
plt.show()
```

---

5. Feature Extraction

Preparing Data for Feature Extraction

```
pixels = df['pixels']
labels = df['emotion']
```

5.2 HOG Feature Extraction

We used Histogram of Oriented Gradients (HOG) to extract features from the images.

```
# HOG Feature Extraction Function
def extract_hog_features(image):
        pixels_per_cell=(8, 8),
        cells_per_block=(2, 2),
        orientations=9,
        visualize=False)
  return features
```

5.3 Processing Images and Extracting Features

To manage computational resources and balance the dataset, we limited the number of images per emotion to 5,000.

1Processed 22063 images.

---

6. Model Implementation

6.1 Splitting the Data

6.2 Logistic Regression

6.2.1 Training the Model

Output:

Logistic Regression - Train accuracy: 0.6107648725212464

Logistic Regression - Test accuracy: 0.48810333106730114

6.3 K-Nearest Neighbors

6.3.1 Training the Model

6.3.2 Predictions

Output:

KNN - Train accuracy: 0.5741643059490085

KNN - Test accuracy: 0.489689553591661

---

7. Model Evaluation

7.1 Logistic Regression Evaluation

7.1.1 Confusion Matrix

7.1.2 Classification Report

Output (Training Data):

Classification Report - Logistic Regression (Training Data):

	precision	recall	f1-score	support
2	0.56	0.46	0.50	3277
3	0.66	0.74	0.70	4001
4	0.54	0.55	0.54	3863
5	0.70	0.70	0.70	2534
6	0.59	0.62	0.60	3975
accuracy			0.61	17650
macro avg	0.61	0.61	0.61	17650
weighted avg	0.61	0.61	0.61	17650

Output (Test Data):

Classification Report - Logistic Regression (Test Data):

```
              precision    recall  f1-score   support

           2       0.37      0.31      0.34       820
           3       0.58      0.64      0.61       999
           4       0.41      0.42      0.41       967
           5       0.59      0.57      0.58       637
           6       0.48      0.50      0.49       990

    accuracy                           0.49      4413
   macro avg       0.49      0.49      0.49      4413
weighted avg       0.48      0.49      0.49      4413
```

7.1.3 ROC and AUC

7.2 K-Nearest Neighbors Evaluation

7.2.1 Confusion Matrix

7.2.2 Classification Report

Output (Training Data):

```
Classification Report - KNN (Training Data):
              precision    recall  f1-score   support

           2       0.57      0.36      0.44      3277
           3       0.56      0.84      0.67      4001
           4       0.60      0.40      0.48      3863
           5       0.67      0.63      0.65      2534
           6       0.53      0.62      0.57      3975

    accuracy                           0.57     17650
   macro avg       0.59      0.57      0.56     17650
weighted avg       0.58      0.57      0.56     17650
```

Test Data:

Classification Report - KNN (Test Data):

	precision	recall	f1-score	support
2	0.44	0.27	0.34	820
3	0.52	0.80	0.63	999
4	0.49	0.29	0.36	967
5	0.57	0.55	0.56	637
6	0.43	0.51	0.47	990
accuracy			0.49	4413
macro avg	0.49	0.49	0.47	4413
weighted avg	0.49	0.49	0.47	4413

7.2.3 ROC and AUC

---

8. Results and Discussion

8.1 Comparison of Models

Metric	Logistic Regression	K-Nearest Neighbors
Training Accuracy	0.6108	0.5742
Testing Accuracy	0.4881	0.4897

8.2 Observations

- Accuracy:
- Both models have testing accuracy of ~49%.
- Training accuracy is higher than testing accuracy, which means there is a possibility of overfitting.
- Confusion Matrices:
- The confusion matrices show that some classes are better predicted than others.
- Class 3 (Happy) has higher recall and precision across both models.
- Classification Reports:
- Precision and recall vary across classes.
- Class 3 consistently shows better performance.
- ROC Curves and AUC:
- ROC curves provide a graphical representation of the models' performance across different thresholds.

• Because of the multiclass nature and possible warnings, interpretation might be complex.

8.3 Discussion

• Model Performance:

• Logistic Regression performed slightly better on training data.

• KNN had comparable performance on test data.

• The models might be affected by class imbalance or limited feature representation.

• Feature Extraction:

• HOG features capture important structural information but may not be sufficient alone.

• Considering other feature extraction methods or deep learning approaches might improve performance.

• Class Imbalance:

• Though we capped the number of images per class, there may still be some imbalance.

• Data augmentation or class weighting techniques could be useful.

---

9. Conclusion

In this project, we developed and implemented Logistic Regression and K-Nearest Neighbors classifiers for recognizing facial expressions in grayscale images. We preprocessed the data, extracted HOG features, and evaluated the models using different metrics for classification.

Key Observations:

• Both models had an accuracy of approximately 49% for the test data.

• Class 3, or Happy, was most predictable while the other classes showed low performance.

• There is much scope for improving feature extraction and the choice of models.

Future Scope:

• Deep learning models like Convolutional Neural Networks (CNNs) could be explored since these perform well in image classification problems.

• Data augmentation will be done to further diversify the dataset.

• Perform hyperparameter tuning. More feature extraction techniques may be tried.