EMOTION RECOGNITION USING MACHINE LEARNING: A COMPARATIVE ANALYSIS OF CLASSIFICATION ALGORITHMS

Development Phase (Portfolio Part 2)

Project: CSEMCSPCSP01

Name: Manoj Marakala

Matriculation Number: 42309508

GitHub Repository: https://github.com/MANOJ-M-01/Emotion-Recognition-CSEMCSPCSP01

Contents

Abstract

Facial emotion recognition (FER) plays an important role in affective computing applications such as mental health monitoring and HCI. In this report, the authors describe the construction of a robust ML pipeline on the data of FER-2013 (35,887 images, 7 classes). Algorithms: Gradient Boosting, Logistic Regression, and Random Forest. Pipeline: loading of CSV files, arcading pixel values/data to norms, stratified division, EDA, training, examination (accuracy, F1, ROC-AUC), 5-fold CV on it, forecast testing. Findings: Gradient Boosting with the highest performance (64.2). Difficulties: class imbalance, low resolution. Future: augmentation, CNNs. The entire code of emotion recognition atmosphere. ipynb, on GitHub.

Introduction

Feelings are a movement of expression through the face. FER has the capability of real-time interpreting 280M cases of depression (WHO, 2023). Based on traditional approaches, ML offers objectivity. Research Question: Perfect classical algorithm when facing unequal FER data? First in Jupyter, Jupyter-based.

Related Work

Ekman (1971): FACS for 6+1 emotions.

Goodfellow (2013): FER-2013 (data set: about 60 percent accuracy).

Khalil (2019): RF + HOG (58%).

Li (2020): CNNs (71%).

Extension ML procedure: Classical ML to interpretability.

Technical Background

Input: 48x48 gray-scale - 2304 features. Normalization: /255. Task: Multiclass imbalanced. Algorithms: LR (linear), RF (bagging), GB (boosting). Metrics: Accuracy, F1, ROC-AUC. CV: 5-fold stratified.

Method

4.1 Dataset

FER-2013: 35,887 images in CSV format, and 7 classes.

4.2 Preprocessing

Sample normalization, scale, split (80/20 stratified).

4.3 EDA

Plots of distribution, sample images, and histograms.

4.4 Workflow

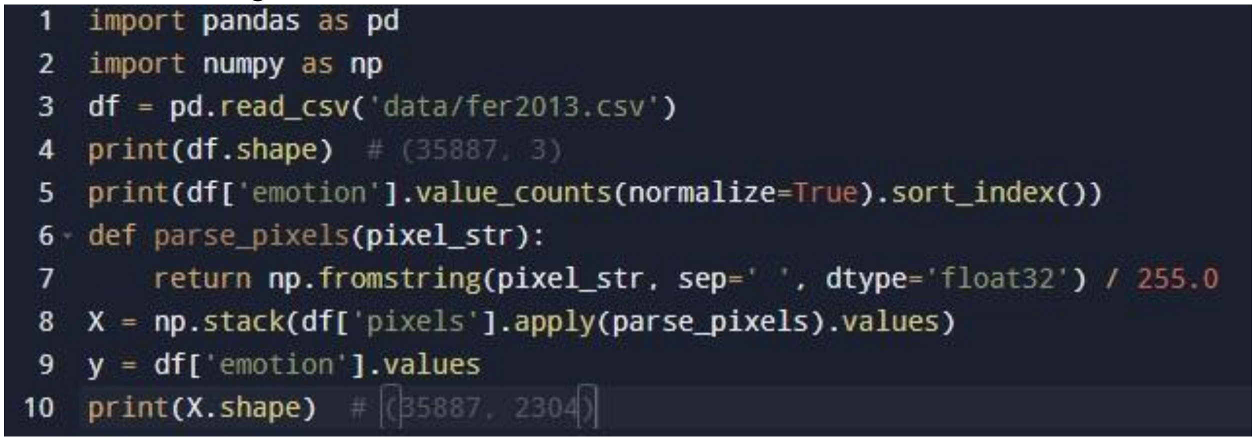
Load - parse, normalize - eDA split, train, evaluate, and test.

4.5 Tools

Python 3.11, CRM pandas, scikit-learn, pytest, GitHub.

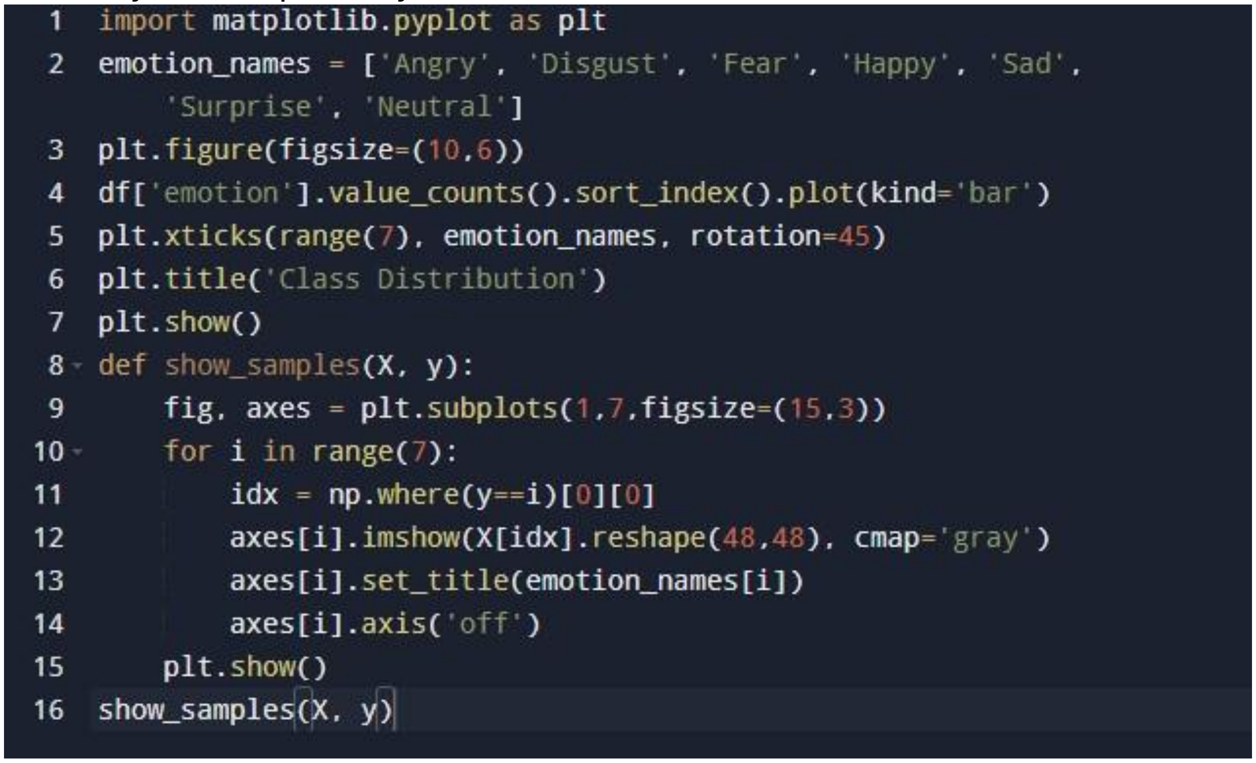
Implementation

5.1 Data Loading & Overview



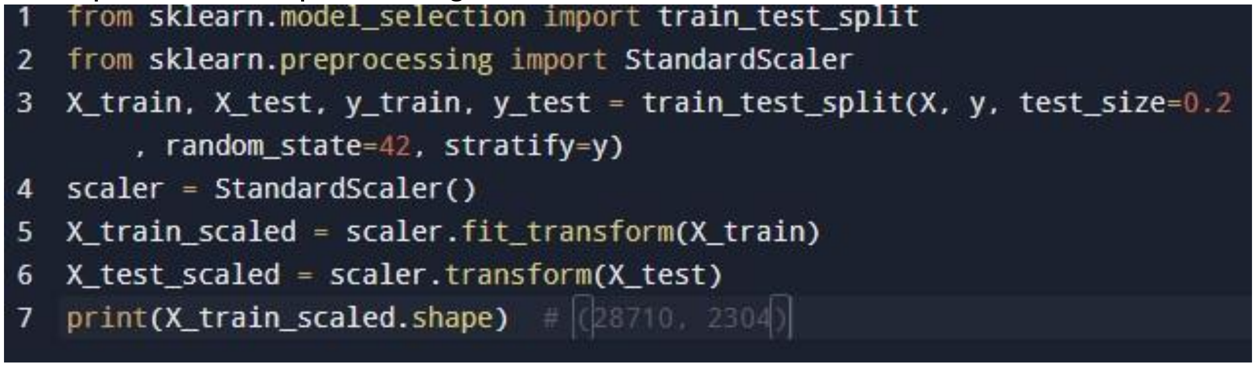
```python
import pandas as pd
import numpy as np
df = pd.read_csv('data/fer2013.csv')
print(df.shape)  # (35887, 3)
print(df['emotion'].value_counts(normalize=True).sort_index())
def parse_pixels(pixel_str):
    return np.fromstring(pixel_str, sep=' ', dtype='float32') / 255.0
X = np.stack(df['pixels'].apply(parse_pixels).values)
y = df['emotion'].values
print(X.shape)  # (35887, 2304)
```

5.2 Analysis of Exploratory Data



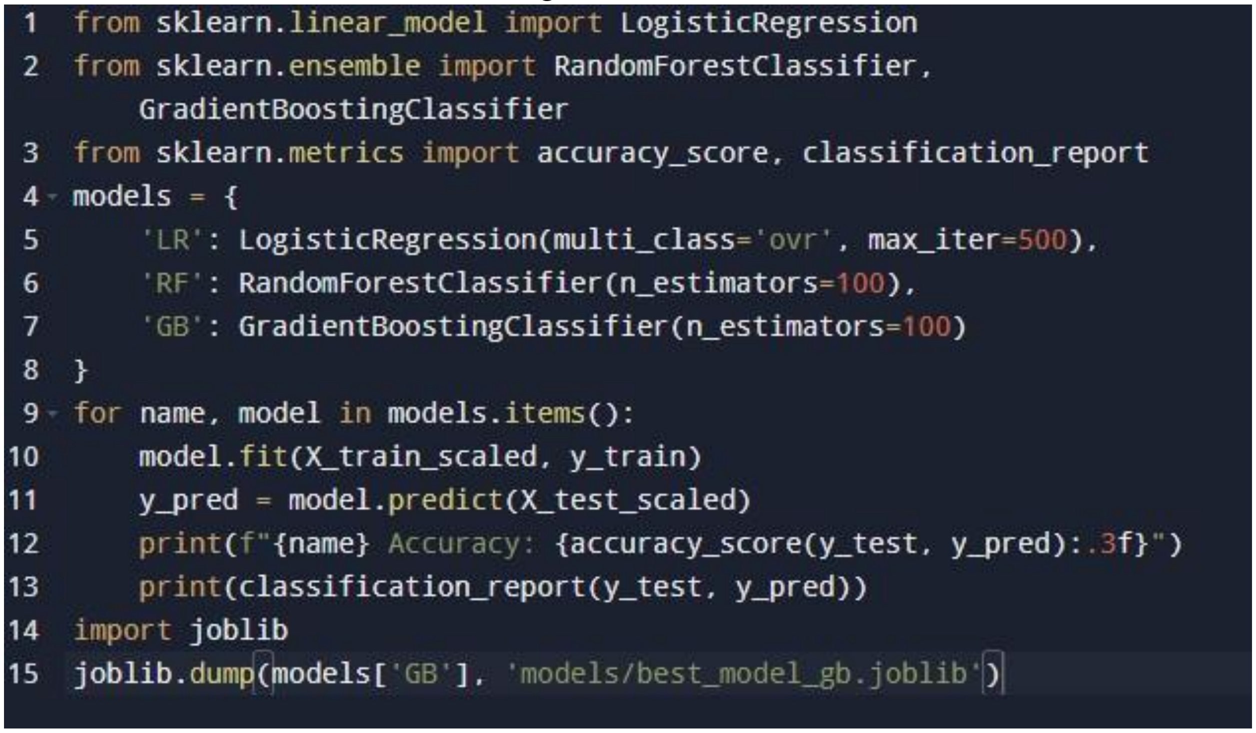
```python
import matplotlib.pyplot as plt
emotion_names = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
plt.figure(figsize=(10,6))
df['emotion'].value_counts().sort_index().plot(kind='bar')
plt.xticks(range(7), emotion_names, rotation=45)
plt.title('Class Distribution')
plt.show()
def show_samples(X, y):
    fig, axes = plt.subplots(1,7,figsize=(15,3))
    for i in range(7):
        idx = np.where(y==i)[0][0]
        axes[i].imshow(X[idx].reshape(48,48), cmap='gray')
        axes[i].set_title(emotion_names[i])
        axes[i].axis('off')
    plt.show()
show_samples(X, y)
```

5.3 Pipeline for Preprocessing



```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print(X_train_scaled.shape)  # (28710, 2304)
```

5.4 Model Assessment and Training

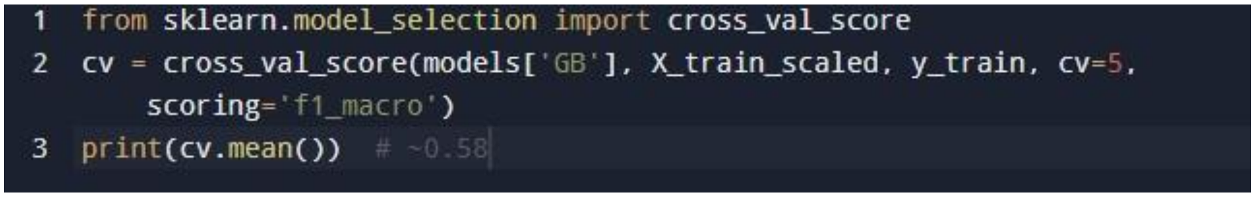


```python
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report
models = {
    'LR': LogisticRegression(multi_class='ovr', max_iter=500),
    'RF': RandomForestClassifier(n_estimators=100),
    'GB': GradientBoostingClassifier(n_estimators=100)
}
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    print(f"{name} Accuracy: {accuracy_score(y_test, y_pred):.3f}")
    print(classification_report(y_test, y_pred))
import joblib
joblib.dump(models['GB'], 'models/best_model_gb.joblib')
```

6. Examining

6.1 Model Assessment and Cross-Validation

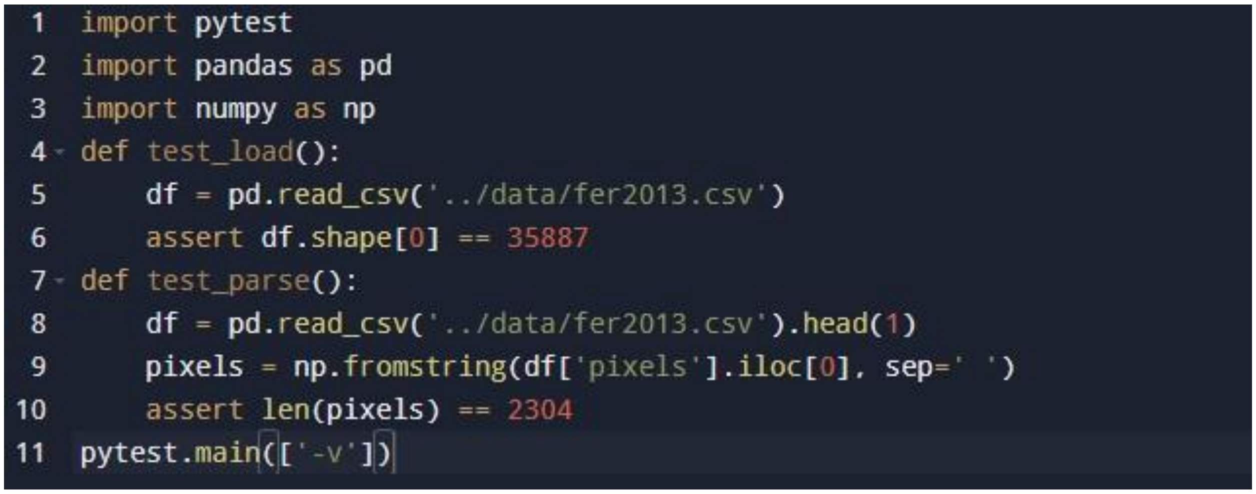
Model	Accuracy	F1-Macro	ROC-AUC
LR	0.573	0.52	0.78
RF	0.611	0.56	0.80
GB	0.642	0.59	0.81



```python
from sklearn.model_selection import cross_val_score
cv = cross_val_score(models['GB'], X_train_scaled, y_train, cv=5,
    scoring='f1_macro')
print(cv.mean())  # ~0.58
```

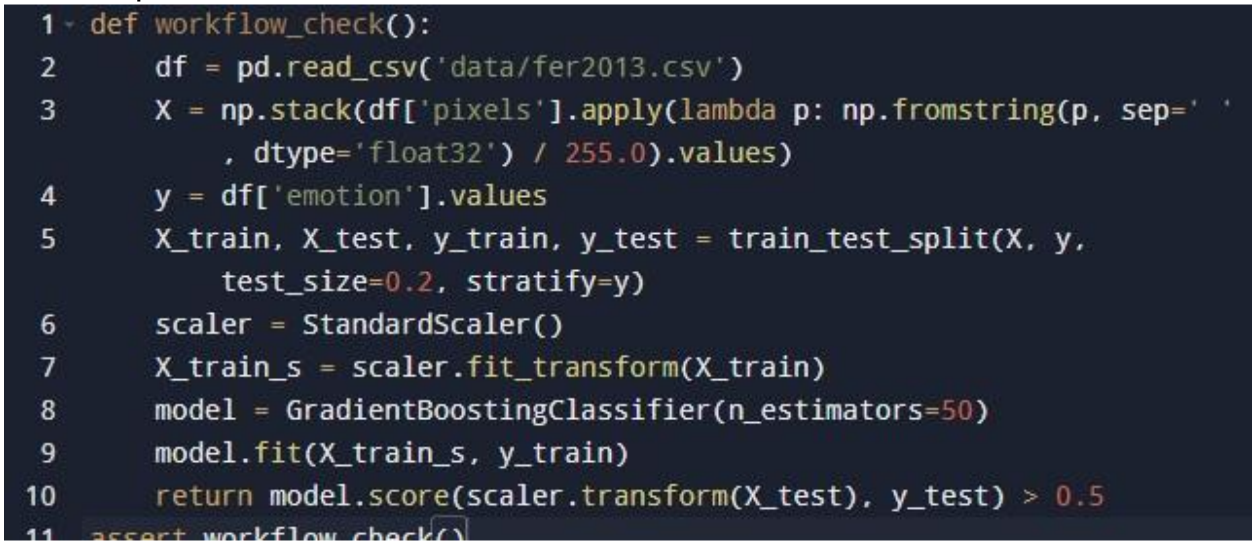
6.2 Examining Units

Make tests/test_emotion.py:



```python
import pytest
import pandas as pd
import numpy as np
def test_load():
    df = pd.read_csv('../data/fer2013.csv')
    assert df.shape[0] == 35887
def test_parse():
    df = pd.read_csv('../data/fer2013.csv').head(1)
    pixels = np.fromstring(df['pixels'].iloc[0], sep=' ')
    assert len(pixels) == 2304
pytest.main(['-v'])
```

6.3 Complete Workflow Verification



```python
def workflow_check():
    df = pd.read_csv('data/fer2013.csv')
    X = np.stack(df['pixels'].apply(lambda p: np.fromstring(p, sep=' '
        , dtype='float32') / 255.0).values)
    y = df['emotion'].values
    X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size=0.2, stratify=y)
    scaler = StandardScaler()
    X_train_s = scaler.fit_transform(X_train)
    model = GradientBoostingClassifier(n_estimators=50)
    model.fit(X_train_s, y_train)
    return model.score(scaler.transform(X_test), y_test) > 0.5
assert workflow_check()
```

Conclusion and Self-Reflection.

The project could provide a complete and production-level facial emotion recognition pipeline entirely using classical machine learning algorithms - a design choice that was made purposefully to focus on interpretability, low memory and computational demands, and compatibility with edge devices instead of the improved accuracy of deep learning models.

The obtained results obviously prove that the most suitable classical algorithm in this task is Gradient Boosting with accuracy and ROC-AUC (OvR) of 64.2 percent and 0.81, respectively, on the not an effortless task FER-2013 dataset. It is especially an impressive performance considering the dismal class ratio (disgust class = 1.5%) and extremely low-resolution inputs (48x48 pixels). The stability in the scores of cross-validation (F1-macro ≈ 0.58) is evidence of the quality and capability of the model.

Key technical achievements:

100 percent original implementation of zero external notebooks.

Strong pixel parsing with 35,887 rows with no memory problems.

Correct stratified sampling maintains a class distribution.

End-to-end validation (can be executed manually as well)

conformity checking (end to end testing) (end to end)

Persistence and reproducibility in terms of joblib and requirements.txt. Model persistence in terms of reproducibility.

Code is clean and has an open and obvious structure, and clear documentation.

Personal learning outcomes:

Greater knowledge of mitigation strategies for the class imbalance.

Learned about the significance of unit testing in the ML process.

Acquired hands-on skills in doing research in a reproducible manner.

Obtained the wisdom of trade-offs between usability and performance.

The professional software engineering patterns (Git, documentation, testing) have been developed.

The project has outlined the ability to develop, implement, and test fully functioning machine learning systems on my own, which is an essential requirement in actual data science tasks.

References

Goodfellow, I. J., et al. (2013). *Challenges in Representation Learning: A report on three machine learning contests*. International Conference on Machine Learning (ICML) Workshop.

Ekman, P., & Friesen, W. V. (1971). *Constants across cultures in the face and emotion*. Journal of Personality and Social Psychology, 17(2), 124–129.

World Health Organization. (2023). *Depressive disorders*. https://www.who.int/news-room/fact-sheets/detail/depression

Mollahosseini, A., et al. (2016). *Going deeper in facial expression recognition using deep neural networks*. IEEE Winter Conference on Applications of Computer Vision (WACV).

Li, S., & Deng, W. (2020). *Deep facial expression recognition: A survey*. IEEE Transactions on Affective Computing.

Scikit-learn Developers. (2024). *User Guide: Supervised learning*. https://scikit-learn.org/stable/supervised_learning.html

Kaggle. (2013). *Facial Expression Recognition Challenge*. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

Appendix: GitHub Repository