Facial Emotion Recognition

Nitesh Agrawal

Dataset

https://www.kaggle.com/datasets/tapakah68/facial-emotion-recognition

Motivation for choosing this dataset

I wanted to learn about face landmarks, and how these are related to facial expressions.

Data preparation and preprocessing

Dataset had 152 images with images classified with the emotions:

Anger, Contempt, Disgust, Fear, Happy, Neutral, Sad, Surprised

- used Google's **mediapipe** library to do landmarking for each image
- generated 52 features for each image, and assigned a number [0, 1] to each feature
- created csv file with 52 features and the target (emotion)
- added more images from the internet, final sample size 148

Example Features

- "browInnerUp" # upward movement of the inner portion of both eyebrows
- "cheekPuff" # outward movement of both cheeks
- "eyeBlinkLeft" # closure of the eyelids over the left eye
- "eyeLookUpLeft" # movement of the left eyelids consistent with an upward gaze
- "eyeSquintLeft" # contraction of the face around the left eye
- "jawOpen" # an opening of the lower jaw
- "mouthDimpleLeft" # backward movement of the left corner of the mouth
- "mouthStretchLeft" # leftward movement of the left corner of the mouth
- "noseSneerLeft" # raising of the left side of the nose around the nostril

Model Selection

- multicollinearity
- not normally distributed
- overlap between classes

➔ k-NN

➔ LDA

➔ Random Forest

Feature Selection

- **n** is not too big compared to **p** (148 vs 52)
- Feature Selection
    - Minimum-redundancy-maximum-relevance (mRMR)

      pymrmr.mRMR

    - The Mutual Information (MI) classifier

      from sklearn.feature_selection import mutual_info_classif

Feature Selection

- Forward selection and backward elimination gave errors, probably because of high multicollinearity
- pymrmr.mRMR yielded poor results
  - for top 20 features the accuracy was half of the accuracy without feature selection
- used MI classifier for kNN and LDA
- for Random Forest feature selection is inbuilt

Cross Validation

- Leave one out cross validation
- k-fold cross validation (k=5)
  - Smaller test error variance than LOOCV

```
for i in range(0, y.shape[0], k):
    indices_to_test = [j for j in range(i,i+k) if j < y.shape[0]]
    X_train, X_test = X.drop(indices_to_test), X.iloc[indices_to_test]
    y_train, y_test = y.drop(indices_to_test), y.iloc[indices_to_test]

    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy_list.append(accuracy_score(y_test, y_pred))
```

k-NN

Mean Accuracy = 0.60

```
model  = KNeighborsClassifier(n_neighbors = 4  )
mi_scores = mutual_info_classif(X, y)

# Create a DataFrame to display the MI scores
mi_df = pd.DataFrame({
    'Feature': feature_names,
    'MI Score': mi_scores
}).sort_values(by='MI Score', ascending=False)

# Select top 45  features based on MI scores
top_features = mi_df['Feature'].head(45).values
```

Linear Discriminant Analysis

LDA gave the best accuracy

Number of features	Mean Accuracy
37	0.64
38	0.64

Random Forest

Hyperparameter Tuning using
Bayesian optimization

- from skopt import BayesSearchCV

'n_estimators' = Number of trees in the forest

'max_features' = Fraction of features to consider when looking for the best split

```
param_space = {
    'n_estimators': Integer(100, 500),
    'max_features': Real(0.1, 1.0)
}

# Initialize Bayesian Search
opt = BayesSearchCV(
    estimator=rf,
    search_spaces=param_space,
    n_iter=4,  # Number of parameter settings sampled
    cv=cv,
    n_jobs=-1, # The number of jobs to run in parallel
    scoring='accuracy',
    random_state=random_state
)
opt.fit(X_train, y_train)
model = opt.best_estimator_
```

Random Forest

Hyperparameter tuning varied greatly based on the split (random_state)

random_state	max_features	n_estimators	Mean Accuracy
83	0.24	109	0.59
92	0.22	347	0.62
38	0.20	297	0.62
76	0.90	116	0.57
80	0.77	491	0.59
48	0.34	273	0.60
33	0.50	146	0.61

Learnings

- Initially I assumed that the classification would be easy, but human faces are very unique and expression of an emotion is not very standardized.
- This can be overcome by a large dataset, given that expressions for different emotions can be quite similar. For example, anger vs fear vs surprised, and sad vs neutral.
- Random Forest hyperparameter tuning turned out to be more variable than expected. And without a better accuracy.
- LDA seemed to perform the best. And most cost effective (faster).