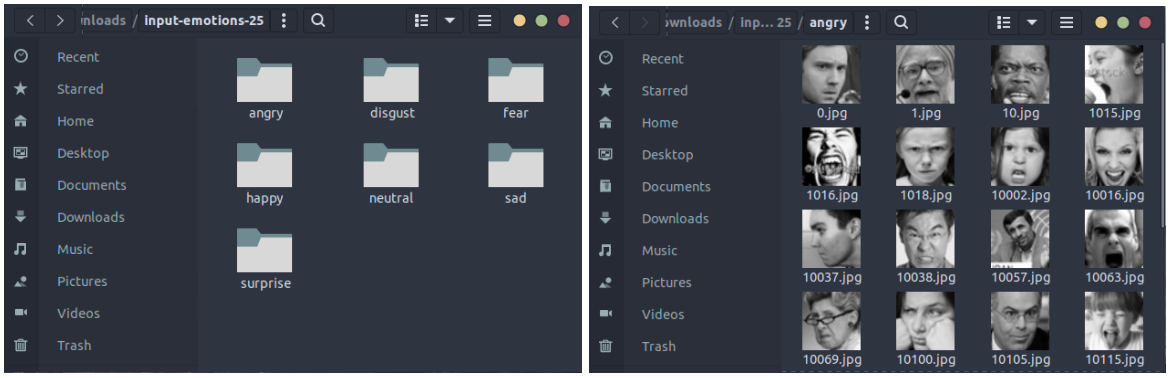
Facial Expression Emotion Classification

Firstly, we download the needed dataset from the website and place them in their corresponding folders according to their class. For this case, we downloaded 25 images per emotion class.



Then we include the needed imports for the project

```
import warnings
warnings.filterwarnings('ignore')

# General
import os
import time
from random import shuffle

# Math
import numpy as np

# Data management
import pandas as pd

#project
import dlib
from imutils import face_utils

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
```

Then we write first the method for extracting the feature, for this project we used an external tool to help up generate the landmarks on the face.

```
path = r"../input/shape-predictor/shape_predictor_68_face_landmarks.dat"
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(path)
landmarks = []

def feature_extraction(file):
    for picture in pictures:
        image = picture
        rects = detector(image, 1)

    for (i, rect) in enumerate(rects):
        shape = predictor(image, rect)
        shape = face_utils.shape_to_np(shape)

    landmarks.append(shape)

    return shape
```

Then initialize the path of the dataset and the arrays to be used.

```
x,y=[],[]
directory = '../input/facialexpressionmini/input-emotions-25'
extracted_features=[]
```

Then extract the features of the image files per folder whilst saving their labels as well.

```
for folder in os.listdir(directory):
    subfile = os.path.join(directory, folder)

    if os.path.isfile(subfile):
        null
    else:
        for imgfile in os.listdir(subfile):
            imgfile_loc = os.path.join(subfile, imgfile)
            # feature extraction from file
            feature=feature_extraction(imgfile_loc)
            x.append(feature)
            y.append(folder)
            extracted_features.append([imgfile,feature,folder])
```

To prepare the data for analysis, we convert the arrays to a numpy array.

```
numpy_landmarks = numpy.array(x)
numpy_labels = numpy.array(y)
```

Reshape the arrays, encode the labels to numbers, and scale the values for the classifiers. We also split the test set into 70-30 and employed randomness, shuffling, and stratification of the data.

```
scaler  =  StandardScaler()
label_encoder = LabelEncoder()

pic, points, pos = numpy_landmarks.shape
x = numpy_landmarks.reshape((pic, points*pos))
x = (x - numpy.mean(x)) / numpy.std(x)
x = scaler.fit_transform(x)

y = label_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(
    x, y, test_size=0.3, random_state=42, shuffle = True, stratify = y)
```

For the first part, we use the decision tree classifier.

```
decision_tree = tree.DecisionTreeClassifier(max_depth=14)
decision_tree = decision_tree.fit(X_train, Y_train)
tree_prediction = decision_tree.predict(X_test)

print(classification_report(Y_test, tree_prediction))
print(confusion_matrix(Y_test, tree_prediction))
```

Then the K-nearest neighbors

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, Y_train)
knn_predictions = knn.predict(X_test)

print(classification_report(Y_test, knn_predictions))
print(confusion_matrix(Y_test, knn_predictions))
print(accuracy_score(Y_test, predictions)*100)
```

Then the naive bayes classifier

```
bayes = CategoricalNB()
bayes.fit(X_train, Y_train)
bayes_prediction = bayes.predict(X_test)

print(classification_report(Y_test, bayes_prediction))
print(confusion_matrix(Y_test, bayes_prediction))
```

The results are as follows:

Decision tree

	precision	recall	f1-score	support
0	0.60	0.43	0.50	7
1	0.00	0.00	0.00	7
2	0.83	0.62	0.71	8
3	0.40	0.25	0.31	8
4	0.44	0.50	0.47	8
5	0.64	1.00	0.78	7
6	0.50	1.00	0.67	8
accuracy			0.55	53
macro avg	0.49	0.54	0.49	53
weighted avg	0.49	0.55	0.49	53

```
[[3 0 0 0 3 0 1]
 [1 0 0 2 0 0 4]
 [0 0 5 1 0 0 2]
 [0 1 0 2 2 2 1]
 [1 0 1 0 4 2 0]
 [0 0 0 0 0 7 0]
 [0 0 0 0 0 0 8]]
```

Knn

	precision	recall	f1-score	support
0	0.36	0.57	0.44	7
1	0.17	0.14	0.15	7
2	0.33	0.25	0.29	8
3	0.33	0.12	0.18	8
4	0.25	0.38	0.30	8
5	0.50	0.57	0.53	7
6	0.43	0.38	0.40	8
accuracy			0.34	53
macro avg	0.34	0.34	0.33	53
weighted avg	0.34	0.34	0.33	53

```
[[4 0 0 0 3 0 0]
 [0 1 0 2 2 0 2]
 [3 0 2 0 1 1 1]
 [1 1 2 1 2 1 0]
 [1 1 0 0 3 2 1]
 [1 2 0 0 0 4 0]
 [1 1 2 0 1 0 3]]
```

Naive bayes

	precision	recall	f1-score	support
0	0.50	0.43	0.46	7
1	0.00	0.00	0.00	7
2	0.60	0.38	0.46	8
3	0.25	0.12	0.17	8
4	0.27	0.38	0.32	8
5	0.60	0.43	0.50	7
6	0.25	0.62	0.36	8
accuracy			0.34	53
macro avg	0.35	0.34	0.32	53
weighted avg	0.35	0.34	0.32	53

```
[[3 1 0 0 0 0 3]
 [0 0 0 1 1 0 5]
 [0 0 3 1 1 0 3]
 [0 1 1 1 2 0 3]
 [2 0 0 0 3 2 1]
 [1 0 0 1 2 3 0]
 [0 0 1 0 2 0 5]]
```

Analysis

According to the decision tree, the accuracy is 0.55 or 55%. In the result, anger, disgust, fear, happiness, and sadness are categorized into numbers: 0 is for angry, 1 is for disgust, 2 is for fear, 3 is for happy, 4 is for neutral, 5 is for sad, and 6 is for surprise. Every emotion appears to have a different score in the precision performance. Angry has 60%, disgust has 0%, fear has 83%, happy has 40%, neutral has 44%, sad has 64% and surprise has 50%. It appears that fear has the highest precision score which is 83%. Although it only has a 49% precision, the recall performed better with 55%.

As for the accuracy of the second classifier, the KNN classifier, is 0.34 or 34%. Every emotion appears to have a different score in the precision performance. Angry has 36%, disgust has 17%, fear has 33%, happy has 33%, neutral has 25%, sad has 50% and surprise has 43%. With a score of 50%, sad appears to have the highest score. There is no difference in the scores between fear and happy emotions. For this classifier, the average recall and precision is similar.

The accuracy of the third classifier, Naive Bayes, is 0.34 or 34%. In the result, anger, disgust, fear, happiness, and sadness are categorized into numbers as mentioned earlier: 0 is for angry, 1 is for disgust, 2 is for fear, 3 is for happy, 4 is for neutral, 5 is for sad, and 6 is for surprise. Angry has 50%, disgust has 0%, fear has 60%, happy has 25%, neutral has 27%, sad has 60% and surprise has 25%. With a score of 60%, fear appears to have the highest score. There was not quite a positive result to the disgust emotion, as it scored 0%. There is no difference in the scores between happy and surprised emotions. Also for this classifier, the average recall and precision is similar.

We can conclude from our results that for emotion classification of facial features, among the three classifiers used, the Decision Tree classifier is the most accurate with a 55% accuracy, whereas Naive Bayes and KNN classifiers had the lowest accuracy of 34%. We can also infer that the decision tree recall performs differently and better than its precision unlike the two other classifiers.