Handwriting Based Gender Classification Project

Team 12

Name	Sec	B.N	Student Code
Robert Mounier	1	32	9190737
Tarek Yasser	1	38	9191298
Mostafa Wael	2	29	9190068
Yahia Zakaria	2	38	9191209

May 31th, 2022

Introduction to the problem

Handwriting is an important piece of evidence that is examined in the questioned document to ensure its authenticity. Handwriting is the written speech of an individual who has characteristics that set him apart from others. It is a learned skill and a complex perceptual-motor task, also known as a neuromuscular task. Visual observation, outline conception, central nervous system pathways, and the anatomy and physiology of the bones and muscles of the hand and arm all work together to produce the desired output. Writing is a function of the conscious and subconscious mind, as well as of the body's motor, muscular, and nerve movements.

Handwriting has distinct characteristics that are unique to each individual and can be used for personal identification. In an extended handwriting sample, no two people write exactly alike, according to the basic principle of handwriting identification. The possibility of distinguishing the gender of the writer from his/her handwriting has been the pursuit of many investigations. Handwriting is said to be brainwriting, so the thinking of a male differs greatly from that of a female due to hormonal involvement and differences in their physiology, which may not only alter their neuromotor functioning, as a result of which writing work is done but also their way of thinking differs and thus characteristic, so handwriting which is observed during the examination may be proven to be showing discriminatory features in male and female handwriting.

Literature Review

1. Improving handwriting based gender classification using ensemble classifiers:

   This paper presents a system to predict gender using ensemble classifiers. The technique relies on extracting a set of textural features from handwriting samples of male and female writers and training multiple classifiers to learn to discriminate between the two gender classes. The features include local binary patterns (LBP), histogram of oriented gradients (HOG), and (GLCM). For classification, it employs artificial neural networks (ANN), support vector machine (SVM), the nearest neighbor classifier (NN), decision trees (DT), and random forests (RF). Classifiers are then combined using bagging, voting, and stacking techniques to enhance the overall system performance.

2. Handwriting Based Gender Classification Using COLD and Hinge Features:

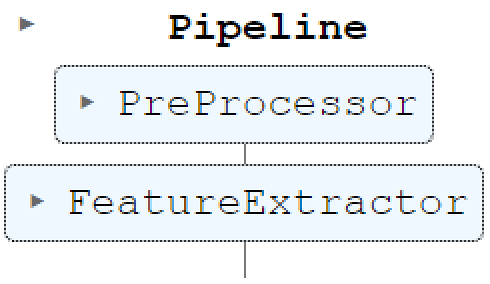
   This paper presents a new method based on the Cloud of Line Distribution (COLD) and Hinge feature for distinguishing the gender from handwriting. The SVM classifier combination decides the assigned class based on the maximum of the two decision values resulting from COLD and Hinge features.

3. Gender classification by means of online uppercase handwriting: A text-dependent allographic approach

   This paper introduces a gender classification scheme based on online handwriting. It classifies the writer as male or female based on samples obtained with a digital tablet that captures the dynamics of the writing. The proposed method is allographic, with strokes serving as the structural units of handwriting.
   Pen-up (in-air) strokes and strokes performed while the writing device is not exerting any pressure on the writing surface are also considered. The method is also text-dependent, which means that training and testing are done with the same text. Text-dependency enables classification to be performed with very little text. Experimenting with samples from the BiosecurID database yields results that are in line with the classification averages expected from human judges. The average rate of well-classified writers is 68% with four repetitions of a single uppercase word; with sixteen words, the rate rises to 72.6%.
   According to statistical analysis, the aforementioned rates are highly significant. These are also considered in order to investigate the classification potential of pen-up strokes. Although the results, in this case, are inconclusive, when information from pen-up strokes is combined with information from pen-down strokes, an outstanding average of 74% of well-classified writers is obtained

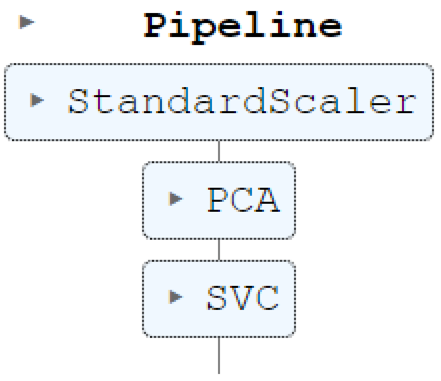
Project Pipeline

Our system is implemented using two sub-pipelines: the data pipeline and the inference (model) pipeline. This separation of work between how we preprocess the data and how we build the model was essential for fast development and experiment, it also helped to make the model ready for deployment.



Data Pipeline

This pipeline is used for transforming input images to the required shape before using it in the model. It passes images through a preprocessor which extracts contours from the image and it passes these contours to the feature extractor to generate feature vectors

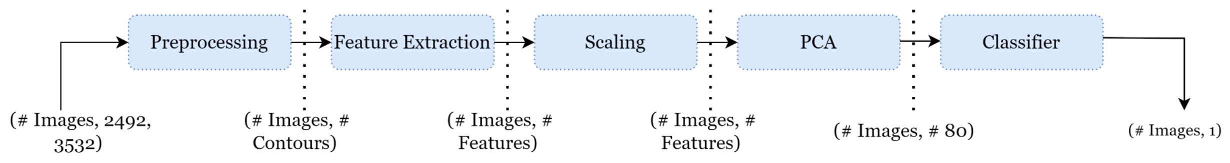


Model Pipeline

This pipeline is responsible for generating the predictions from the given feature vectors. The pipeline starts by normalizing the data to reduce the numerical difference between the features then it reduces the number of features using PCA then uses a classification algorithm such as Random Forest or SVM to get the prediction

Data Flow

Another way to represent the system's general architecture is by representing the output and input data shape between each module. This representation is very useful for debugging as many errors happen because of wrong input shapes

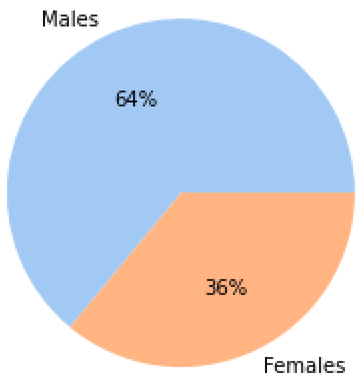


System Modules

Preprocessing

The main task for the preprocessing module is to clear the data and fix any issues in it. After we explored the data and do some analysis on it we found some issues and we tried to solve them as follows:

Data Unbalancing

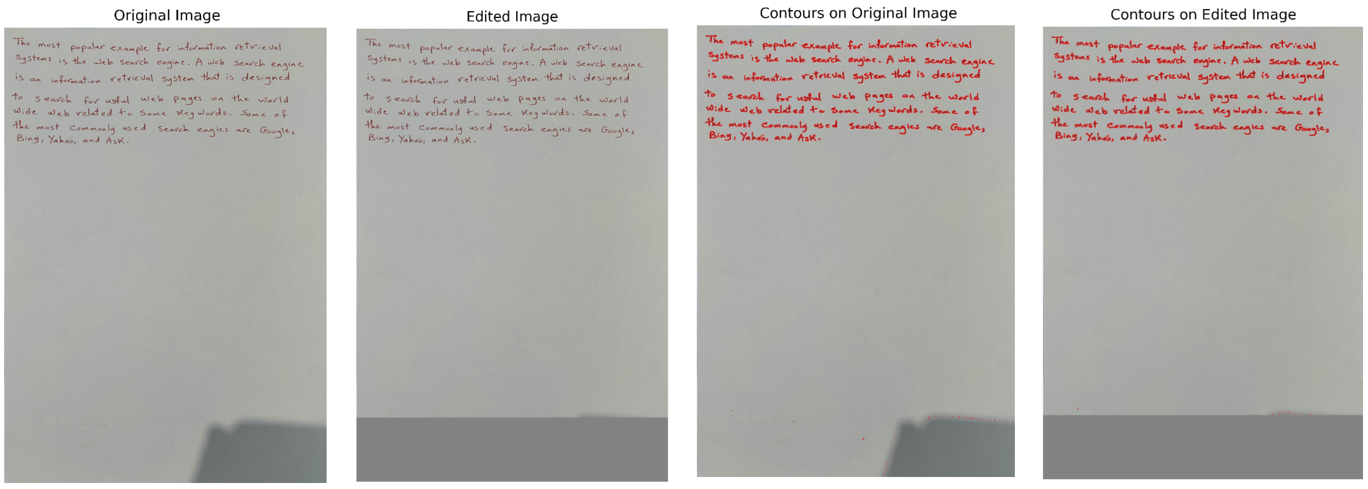


The data unbalancing problem happens when the samples given for a class are much more than the sample for the second the other class. This can damage the generalization performance for the model as it will tend to classify all points to the first class.
To mitigate this problem we did the following:

1. Created the dev set with an equal number of labels for each class so that we can rely on it for measuring generalization performance
2. The number of males in the train set is slightly higher than the number of females by about 30 examples. This threshold was found empirically as we wanted to use as much data as we can but were constrained to the performance of the females class

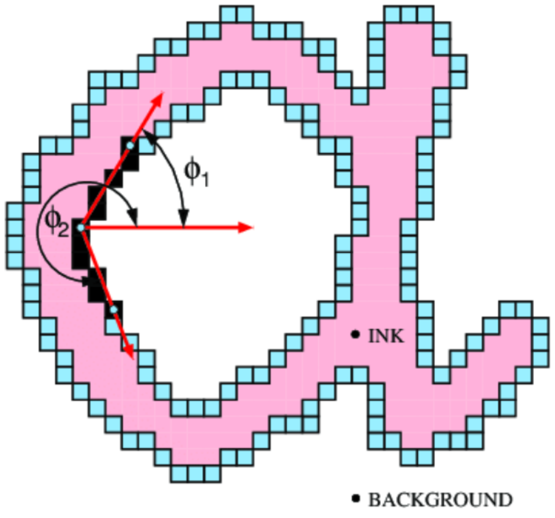
Shadows on Images

The bottom third or so of the image had hand shadows due to poor imaging conditions, no useful information was in that area, but it proved to be an issue during preprocessing due to thresholding transforming the shadow into a black blob. This was overcome by unifying the color of the bottom third or so of the image to a gray color.



Feature extraction

Hinge Features



Hinge features capture information about the curvature and slant / angle of line at the point of intersection between two lines. This is done by computing the joint probability distribution of the orientations of the two legs. This extractor has 2 parameters. The length of each leg, and the number of angle bins. In our implementation we obtain legs by finding all contours in the image, then we filter out any contours shorter than 25 pixels. We then compute the angles between each two neighboring contours and construct a histogram using the angles ϕ_1, ϕ_2 (demonstrated in the above figure).

COLD Features

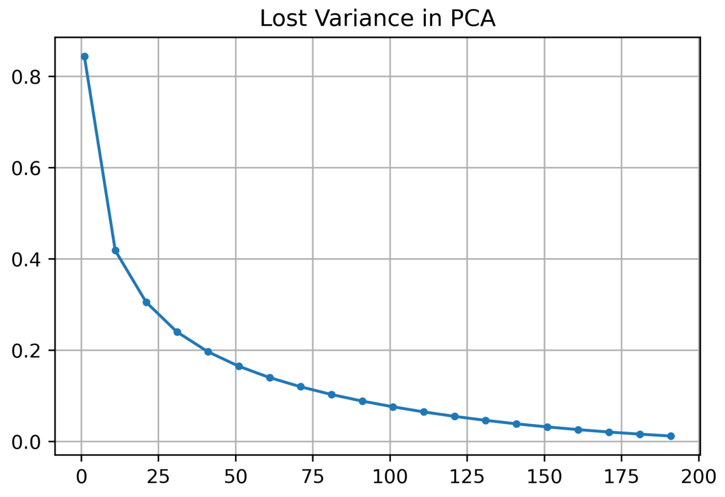
COLD is based on the shape context descriptor. In a nutshell, we pick points from the contours of the shape, we then construct a set of vectors between the point p_i and all other $n - 1$ points, we then build a histogram based on the relative coordinates between p_i and the $n - 1$ other points. COLD further uses dominant points; such as straight, angle-oriented features and curvature over contours of handwritten text components.

Standard Scaler

This module was fairly simple. It normalized our data by subtracting the mean and dividing by the standard deviation.

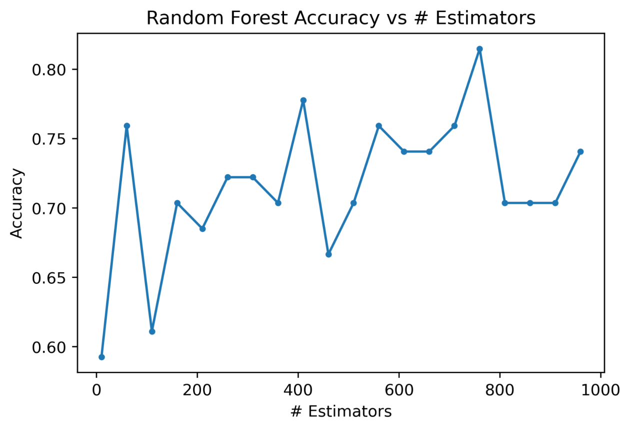
PCA

This module helped us speed up our training by analyzing our components/features and sorting them descendingly by variance. This helped us to focus on more discriminatory features and thus train faster with fewer features and lower resource usage, allowing faster experimentation. For choosing the number of components we used the elbow method as illustrated in this graph and we chose a value between 80 to 90 components.



Model selection

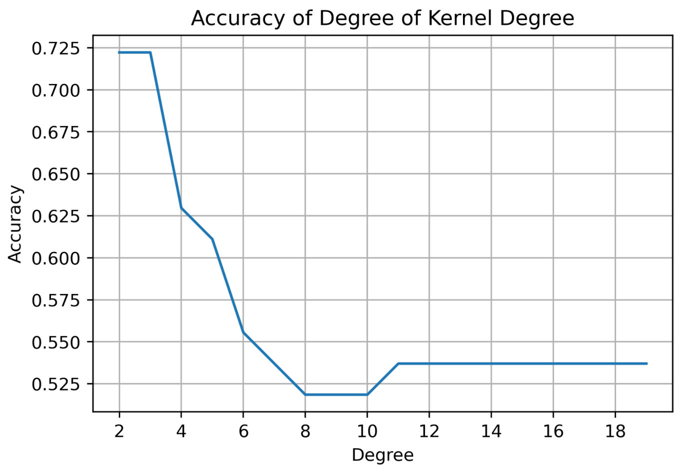
1. Random forest
   We decided to use the random forest as our first model for various reasons:
   - They are based on trees, so the scaling of the variables doesn't matter. Any monotonic transformation of a single variable is implicitly captured by a tree.
   - They use the random subspace method and feature bagging to prevent overfitting by decreasing the correlation between decision trees considerably. Hence, increasing the mean accuracy of predictions automated feature selection is built-in



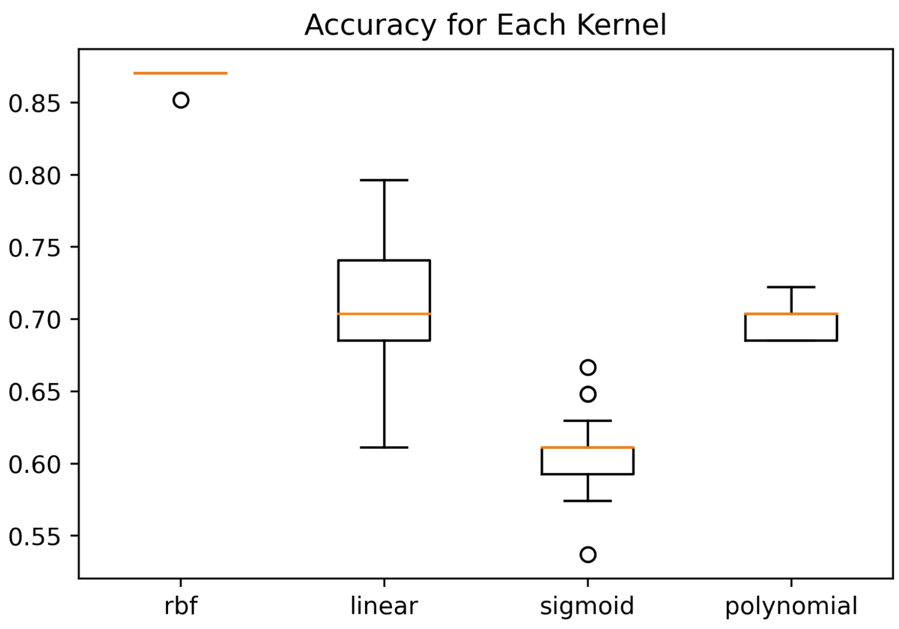
.For Hypertuning the number of estimators parameter we used a grid search from 10 to 1000 with a step size equal to 50 and get the following results

2. SVM

SVM was our final model. The reason is that SVM is one of the most robust and accurate among the other classification algorithms, as It can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. Furthermore, SVM is effective in cases where the number of dimensions is greater than the number of samples, which is the case in our problem.



For choosing the best hyperparameters we started by using a polynomial kernel and searched for the best degree but as we chose from the graph the results were decreasing as the model was overfitting so we tried the RBF, liner, and sigmoid kernel. To model was the uncertainty between each run we created each experiment about 100 times and then plotted the results using a boxplot graph as shown



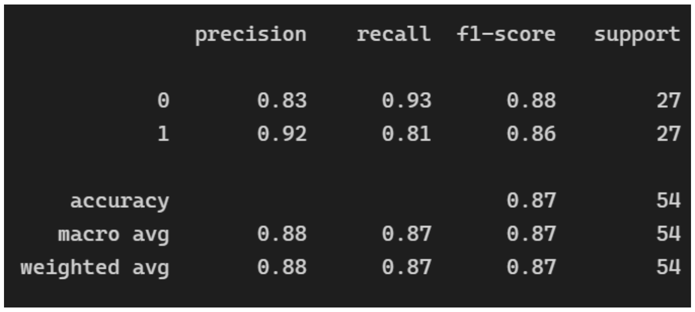
Performance Analysis

Experiments Setup

CPU	Core i5 9th Gen
RAM	8 Gigs
GPU Acceleration	No
Number of workers	1

Accuracy

SVMs outperformed Random Forests, averaging 85%-87% accuracy on different runs. For different data splits the SVM can reach 90% accuracy. This is the results we obtained on our splitted dev set and it shows that the model treats booth class equally.



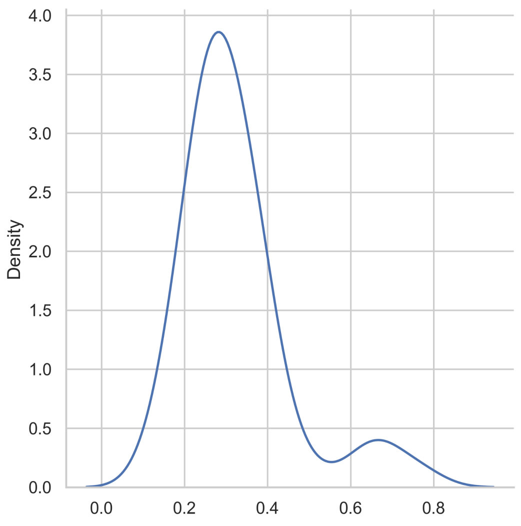
```
              precision    recall  f1-score   support

           0       0.83      0.93      0.88        27
           1       0.92      0.81      0.86        27

    accuracy                           0.87        54
   macro avg       0.88      0.87      0.87        54
weighted avg       0.88      0.87      0.87        54
```

EnfranceTime

0.323 sec ± 0.129 sce with about 99% of this time in feature extraction



Comparison between the modules and the approach

The first issue we encountered was data imbalance; we had 363 images, with only 36% of them being for females and the rest being for males. This unwanted bias had a significant impact on our accuracy and made our model extremely sensitive to randomness in splitting. To address this issue, we sacrificed some of the male data in order to achieve unbiased and evenly distributed data.

Another issue was with preprocessing; we used a variety of preprocessing techniques such as contrast enhancement, dilation, erosion, and so on. Unfortunately, our accuracy suffered as a result of this preprocessing. We discovered that most of our preprocessing techniques had a significant impact on our features and reduced their impact. Dilation and erosion, for example, had an effect on the curvatures of the letters. As a result, we extracted bad Hinge features and so on. As a result, we eliminated the majority of the preprocessing steps, leaving only those necessary for feature extraction. For example, removing shadows to make contour extraction easier.

For model selection, in addition to the random forest and SVM algorithms shown in the previous point, we used XGboost, which is one of the most widely used algorithms in machine learning, whether the problem is a classification or a regression problem. It is well-known for outperforming all other machine learning algorithms. This is due to various properties it possesses, such as:

- Regularization is regarded as a key component of the algorithm.
- Has an inbuilt Cross-Validation function.
- It is designed in such a way that it can handle missing values. It identifies and apprehends trends in missing values.

XGboost carries out the gradient boosting decision tree algorithm. Boosting is nothing more than an ensemble technique that resolves previous model errors in new models. Like the AdaBoost algorithm, these models are added sequentially until no other improvement is observed. Gradient boosting is a method that creates new models that compute the error in the previous model and then adds leftovers to make the final prediction.
So, why did we leave it over? Although XGboost is a powerful algorithm, it needs many hyperparameters to be tuned and that seemed like learning how to fly an Airplane only to go to the end of the street!

Another problem was determining how to test and evaluate our accuracy. We did not use the traditional method of calculating accuracy, despite the fact that it is simple and interpretable, it does not take into account how the data is distributed. This could be critical and lead to an incorrect conclusion. So far, we've used the Precision, Recall, and F1 score models.

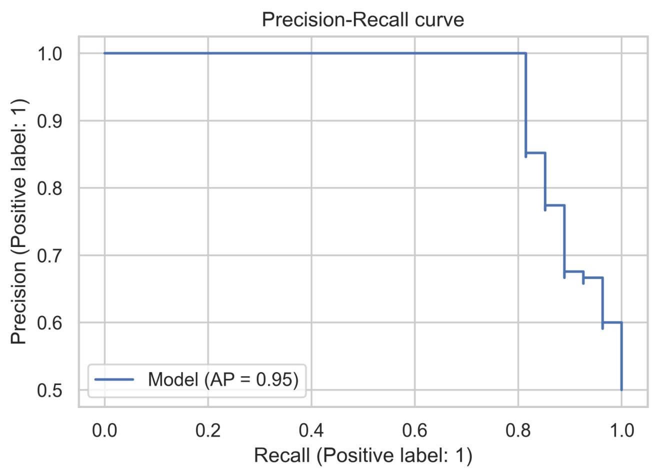
		Predicted class	
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

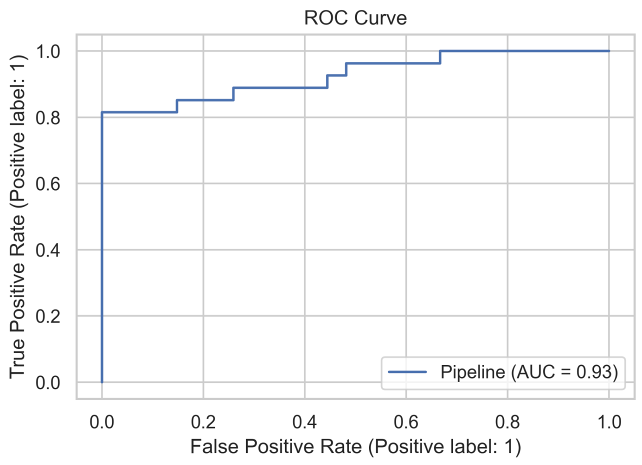
Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. (Precision = TP/TP+FP)

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to all observations in the actual class. (Recall = TP/TP+FN)

F1 Score is calculated as the weighted average of Precision and Recall. As a result, this score considers both false positives and false negatives. It is not as intuitive as accuracy, but F1 is usually more useful than accuracy, especially if the class distribution is uneven. Accuracy works best when the costs of false positives and false negatives are comparable. If the cost of false positives and false negatives differ significantly, it is preferable to consider both Precision and Recall. (F1 Score = 2*(Recall * Precision) / (Recall + Precision))

Our Classifier Results





Enhancement & Future Work

Despite achieving an average of 80% or more on our small, handmade dataset, our work could be improved in several aspects:

- Larger datasets.
- Better balanced data.
- Better imaging conditions.
- Using more advanced classifiers such as deep neural networks and CNNs.

Workload dist.

- Yahia: Pipeline optimization and model development
- Robert: Features extraction and image processing
- Mostafa: Model development
- Tarek: Features extraction model development.