Handwritten Signature Verification using Local Binary Pattern features and KNN

Tejas Jadhav - B002

Under the guidance of Mentor
Prof. Abhay Kolhe

Contents

Problem definition

- Signature Verification is the procedure of determining to whether a particular signature is genuine or forged.

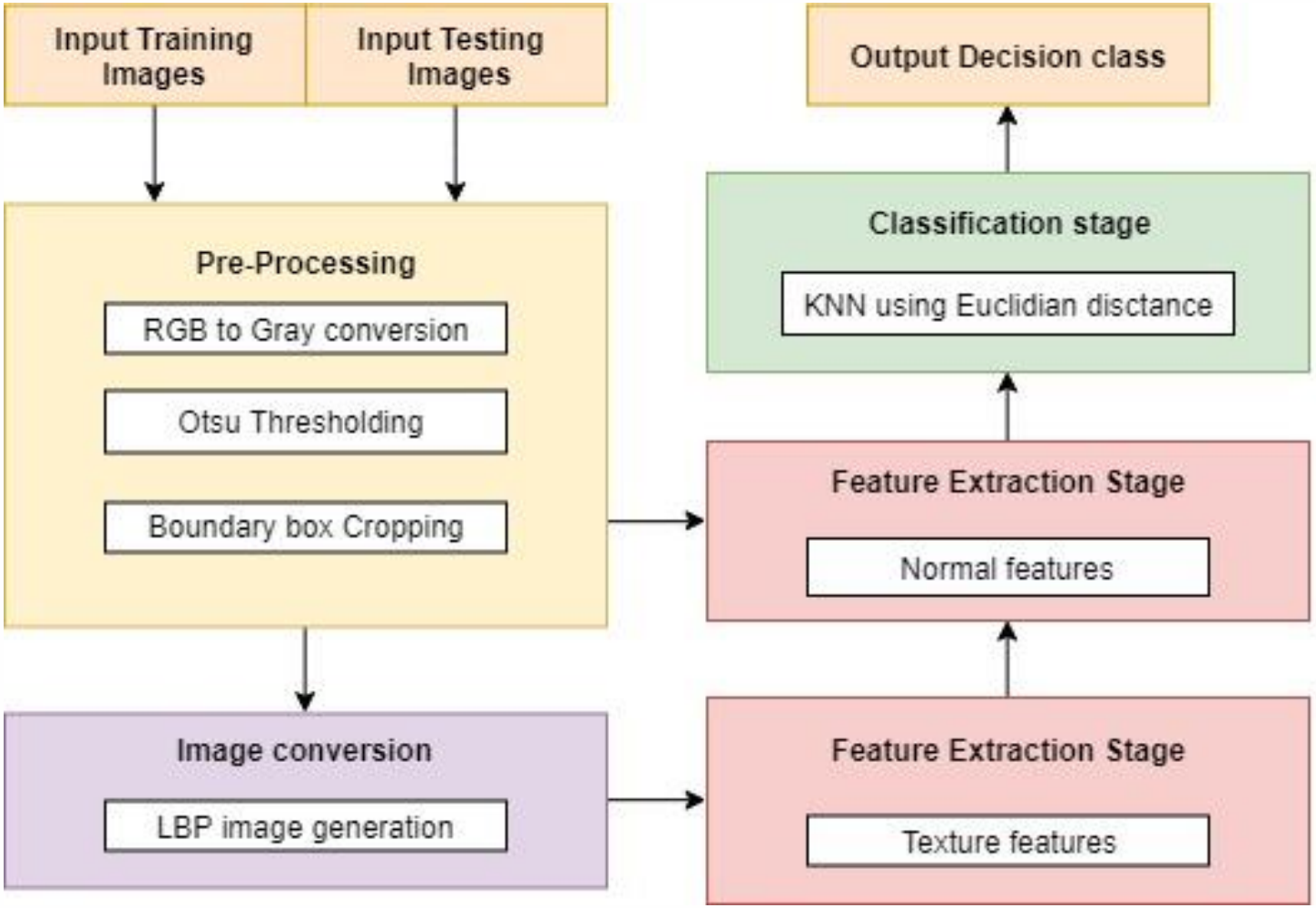
- System would take as input author ID and signature images and tell us, If the signature ,ust be 'Accepted' or 'Rejected'

Literature Review

- An excel sheet of all the Literature Review has been prepared and is been attached with this slide.
- The literature review contains total of 40 research papers based on the topic Signature Recognition
- Most of the papers make use of 3 stages:
  - Preprocessing stage
  - Feature extraction stage
  - Classification stage

Microsoft Excel Worksheet

Proposed Work



Proposed Work

- Local Binary Pattern

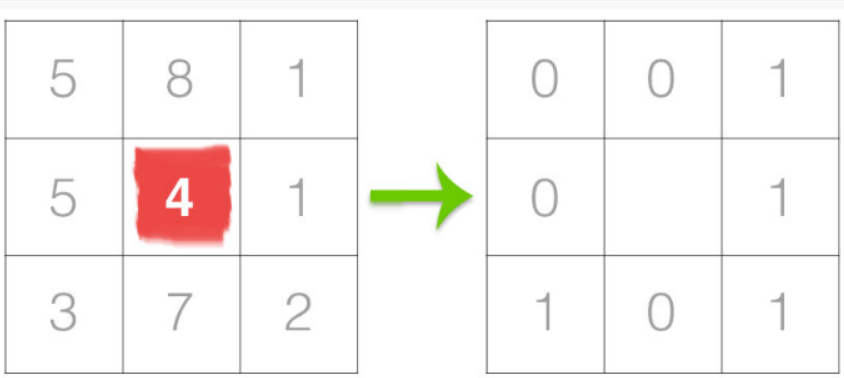


Figure 1: The first step in constructing a LBP is to take the 8 pixel neighborhood surrounding a center pixel and threshold it to construct a set of 8 binary digits.

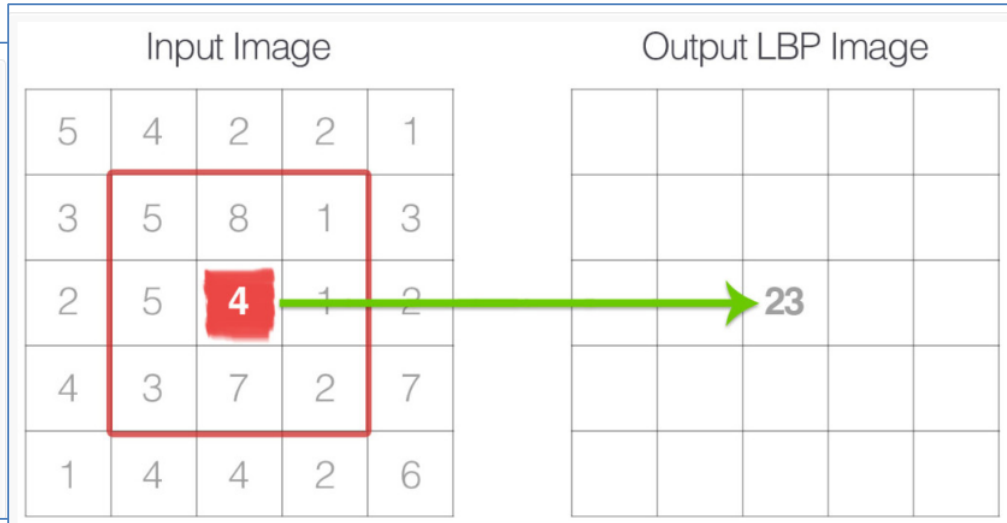


Figure 3: The calculated LBP value is then stored in an output array with the same width and height as the original image

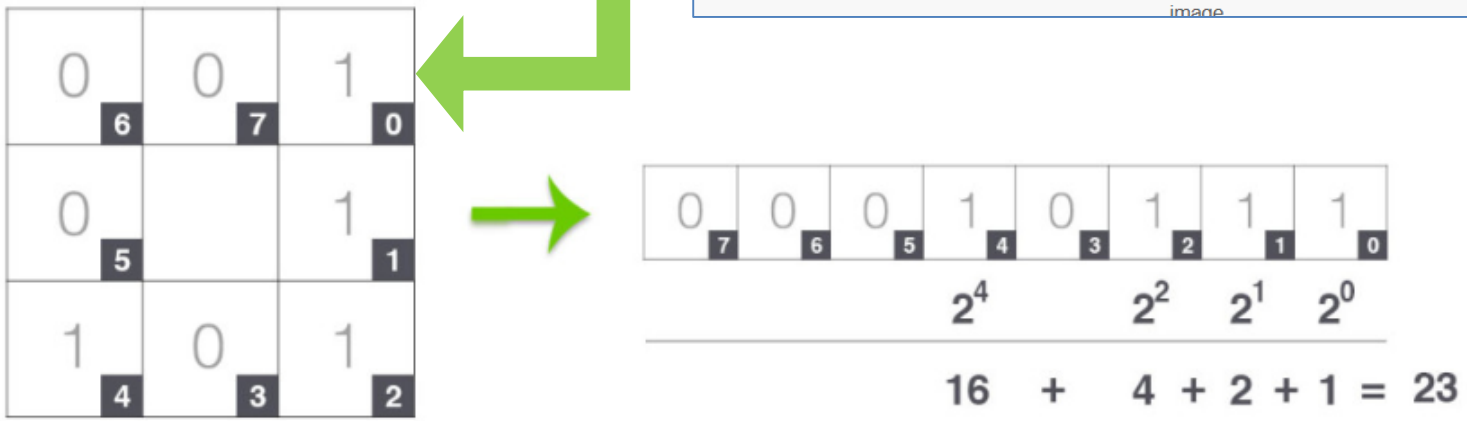


Figure 2: Taking the 8-bit binary neighborhood of the center pixel and converting it into a decimal representation. (Thanks to Bikramjot of Hanzra Tech for the inspiration on this visualization!)

Implementation tool & setup

Python using PyCharm

- Python is a popular programming language used in web & software development, mathematics, system scripting
- PyCharm is a python editor and compiler allows intelligent code completion, on-the-fly error checking and quick-fixes, easy project navigation, and much more.

SQL using MySQL

- SQL is a standard language for storing, manipulating and retrieving data in databases.
- MySQL is an open source relational database management system, very easy to establish, use and manage

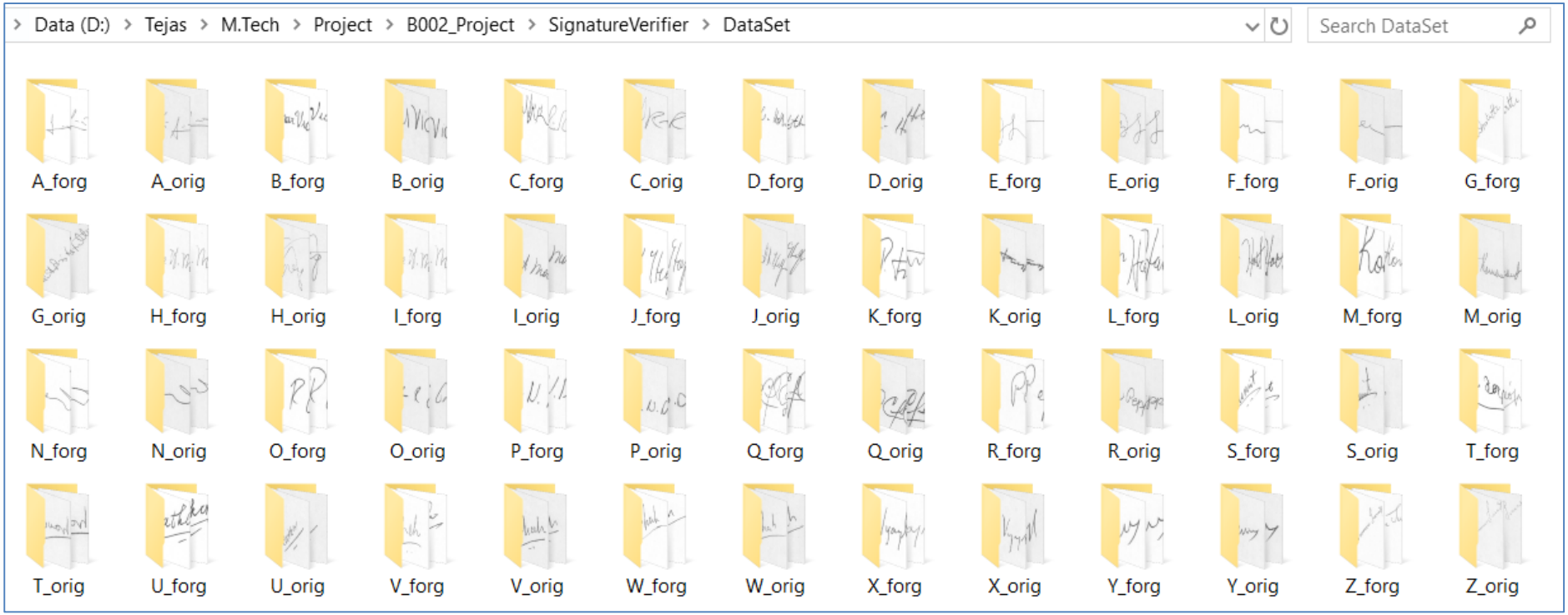
Implementation tool & setup

❑ Python libraries

Using number of libraries, which are easy to install & import..

Package	Description
OpenCV	Computer vision and machine learning software library.
NumPy	Scientific computing & array-processing
Imutils	Functions to make basic image processing functions easier
Math	Provides access to the mathematical functions
MatplotLib	Python 2D plotting library
Pymysql	A simple database interface for Python
OS	allows easy file handling
Scipy	Provides many user-friendly and efficient numerical routines
PySimpleGUI	User interface renderer

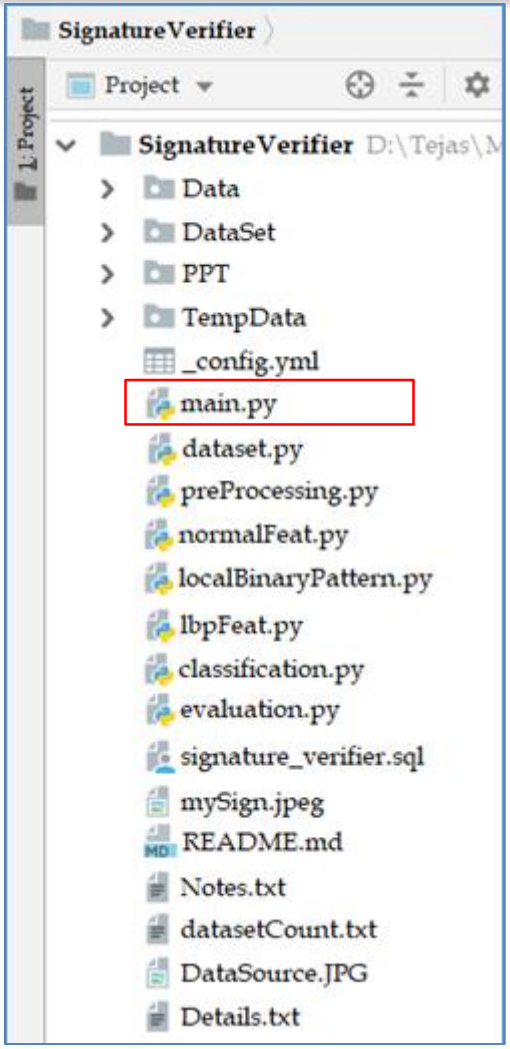
Dataset



26 Authors, 1272 Images
1007 Training images (79.17%)
265 Testing images (20.83%)

Implementation Work done

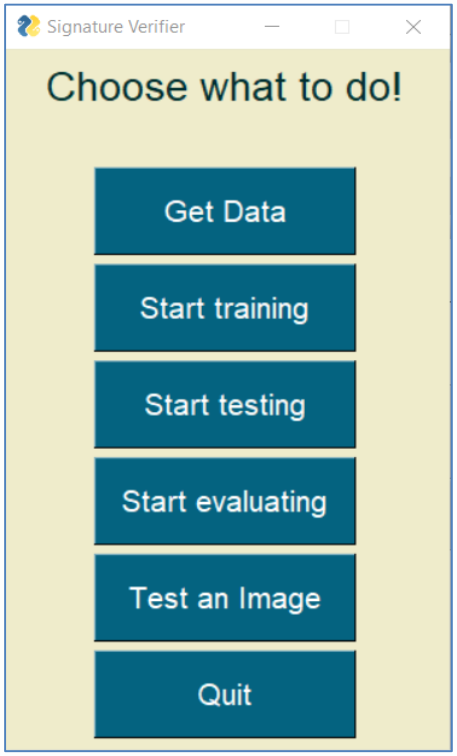
Implementation



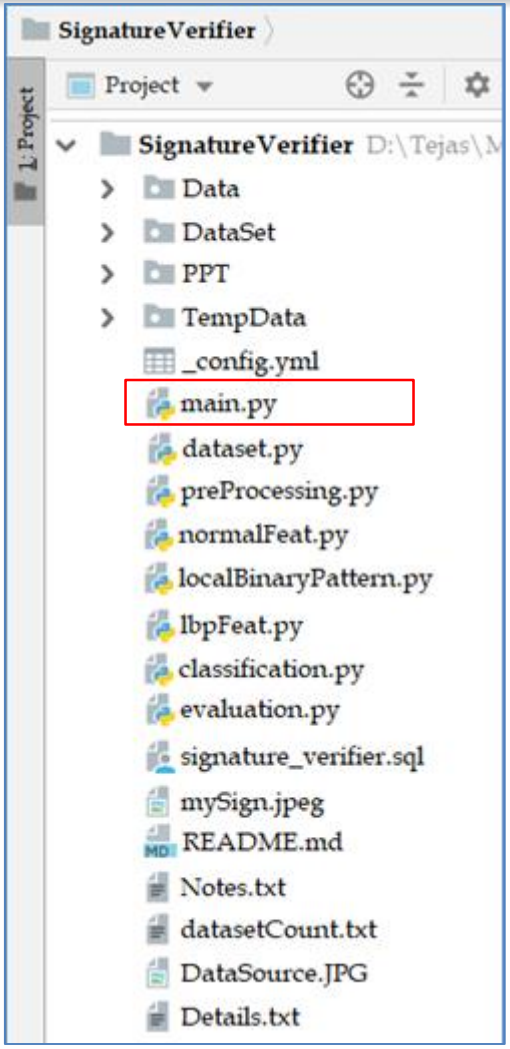
❑ Main.py

- This is the main python file where the system working starts, calls the other functions, gives the appropriate results and ends.

```python
while(str(button) != "Quit"):
    window = sg.Window('Signature Verifier', de
    button, values = window.Read()

    if(str(button) == "Get Data"):
        window.Close()
        ds.getData()
    elif(str(button) == "Start training"):
        window.Close()
        train()
    elif(str(button) == "Start testing"):
        window.Close()
        test()
    elif(str(button) == "Start evaluating"):
        window.Close()
        ev.evaluate()
    elif(str(button) == "Test an Image"):
        window.Close()
        testOne()
    else:
        window.Close()
```

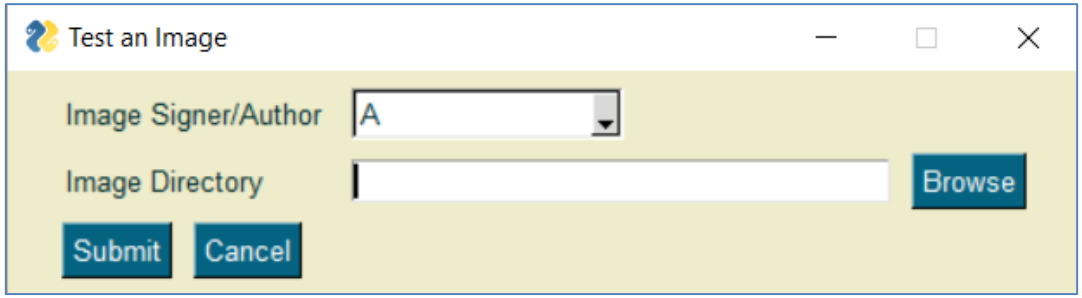


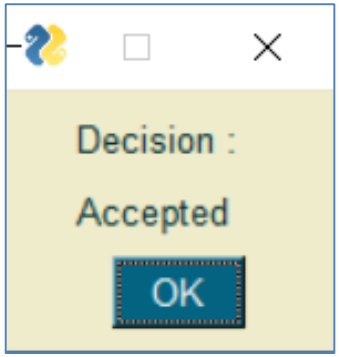
Implementation

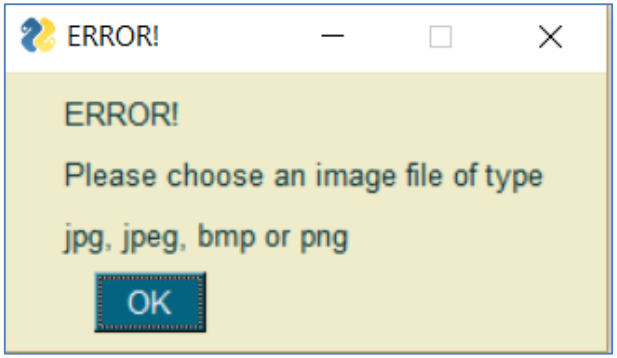


❑ Main.py

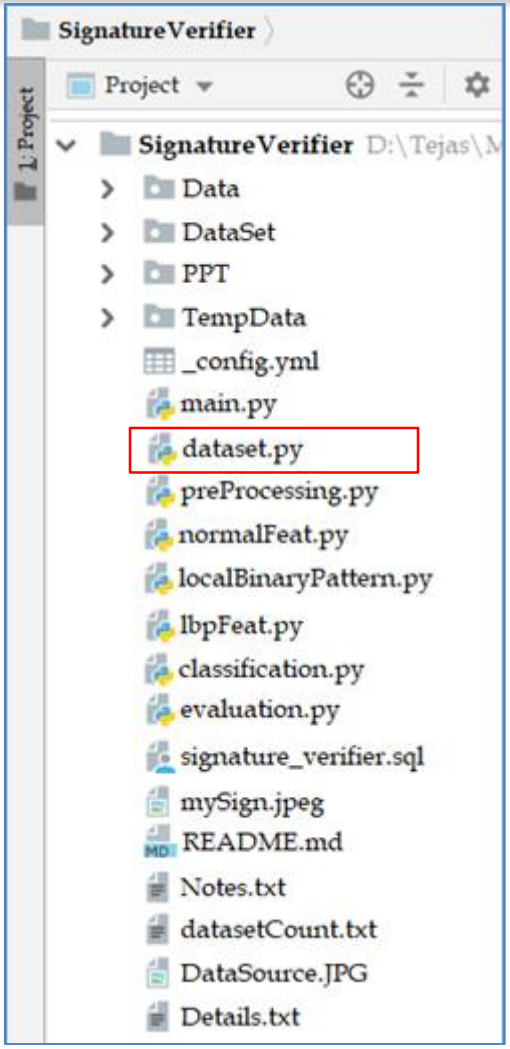
- Test an Image: Clicking on this button will display a popup form and allow user to browse through his computer and perform testing over a single image.







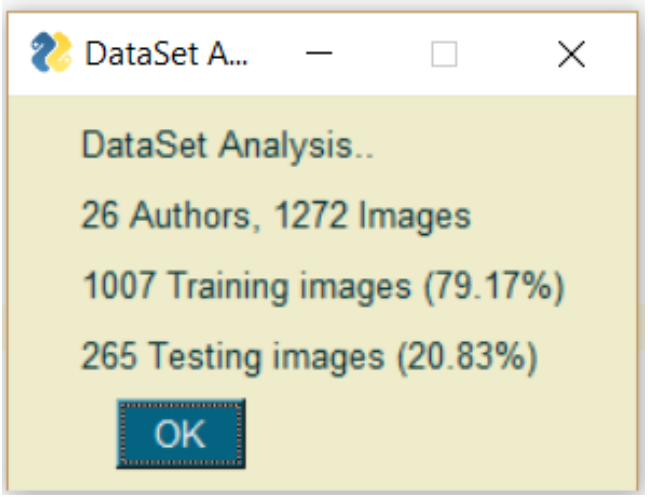
Implementation



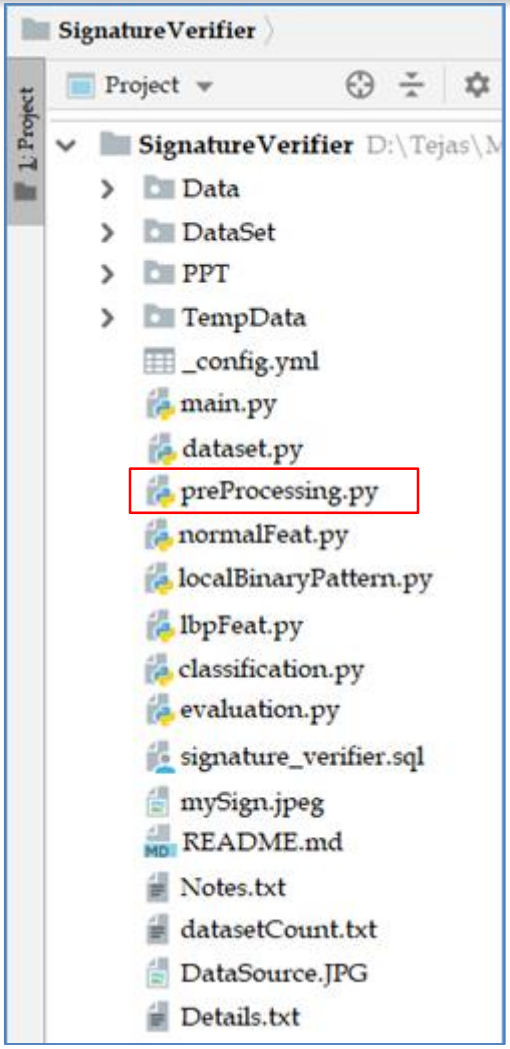
❑ Dataset.py

- Our dataset is read, renamed, copied and organized in the correct naming convention to a different folder, from where our system will use
- xyz.png → A_orig_17.png
- Also gives an analysis of the count of dataset

```
images+=1
exists = os.path
if not exists:
    os.rename("D

dataFolder = tra
dataexists = os.
if not dataexist
    shutil.copy(
```

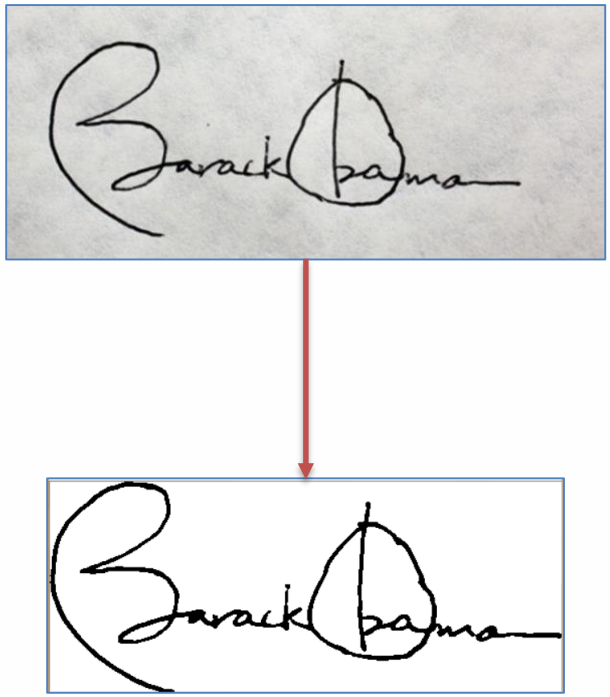


Implementation

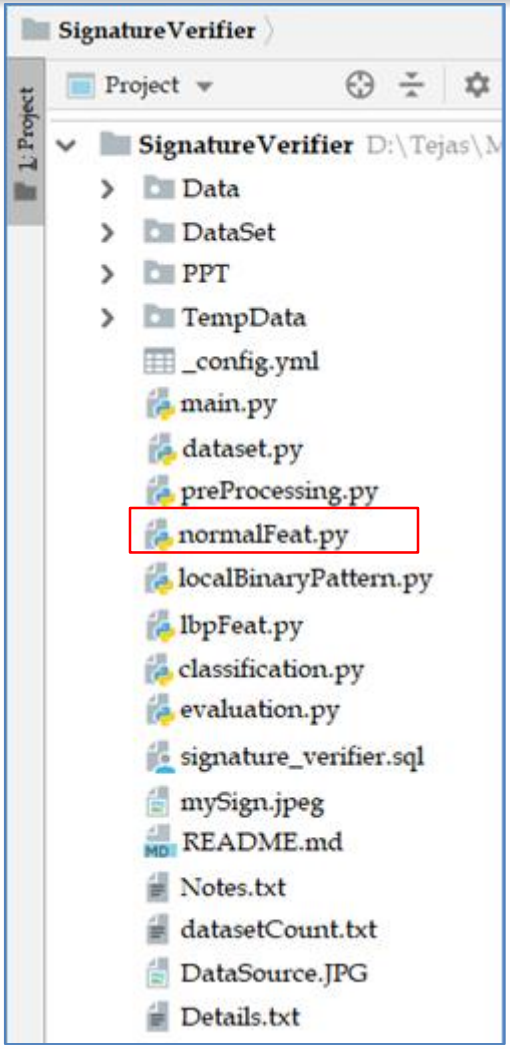


- **Preprocessing.py**
  - This function takes the image in the raw format and converts into a pre-processed format.

1. Resize ,
2. RGB to Grey ,
3. Otsu thresholding,
4. Boundary Box cropping

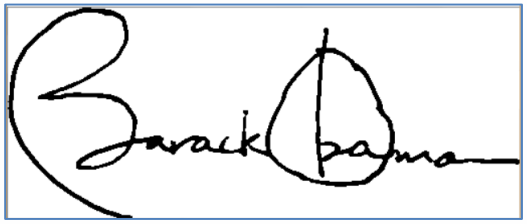


Implementation



❑ NormalFeat.py

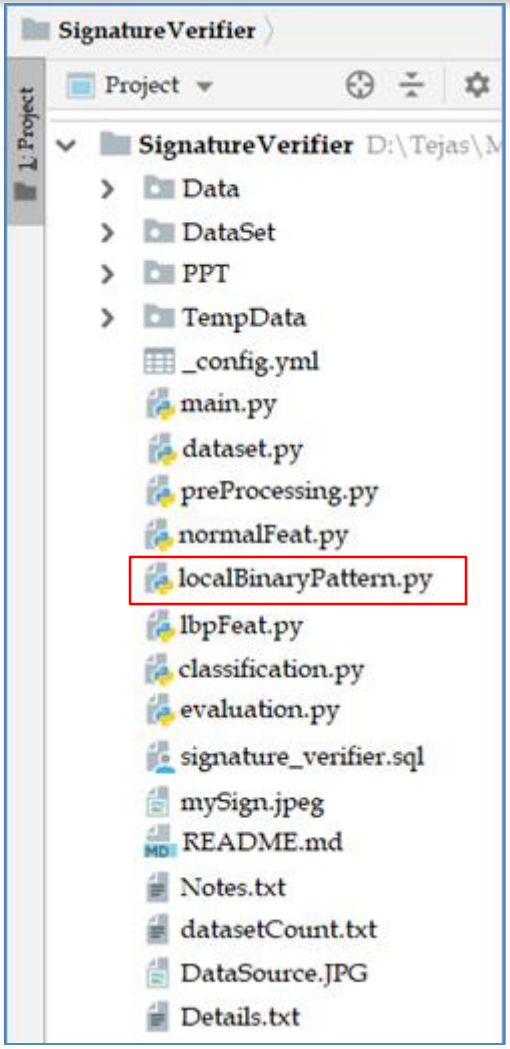
- This function extracts features of the signature from the pre-processed images.



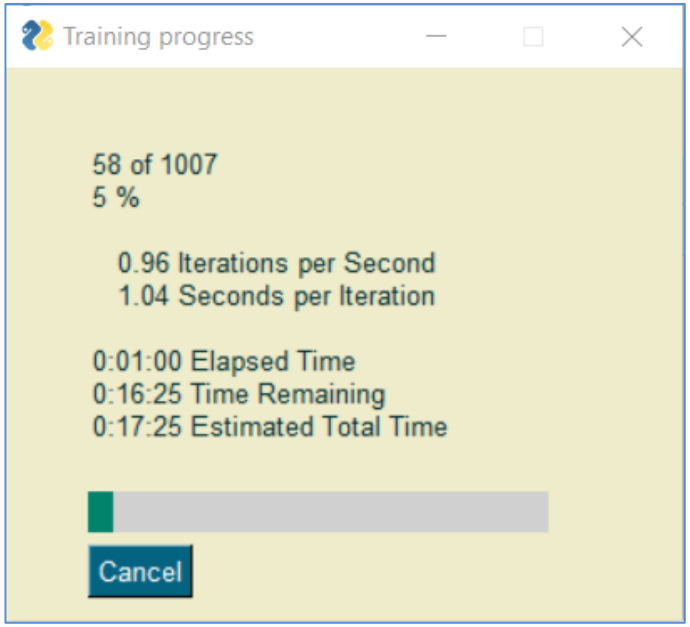
```
~~~ Normal features ~~~
Aspect Ratio :  2.402135231316726
X_COG: 338.5539421701799
Y_COG: 140.53258886252678
Baseline shift:  0.7266805826126017
Energy:  0.958966395847317
Dissimilarity:  1.9347761808716222
Haralick:  950.8804485905165
Kurtosis:  28.81512693799091
```

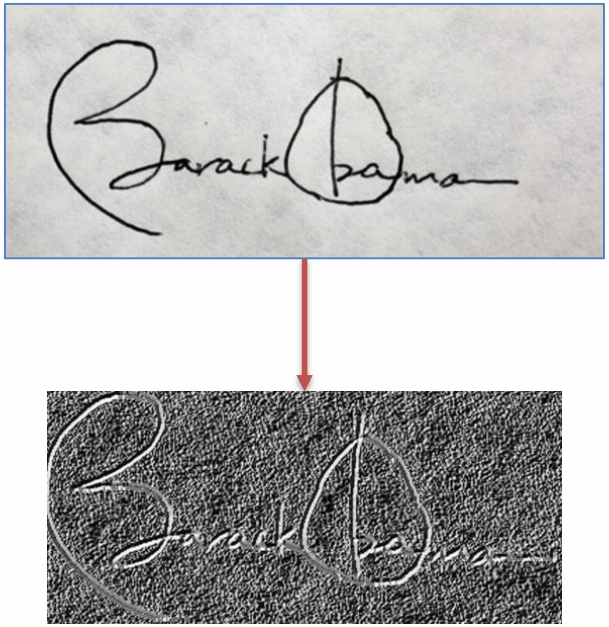
1. Aspect Ratio
2. Center of Gravity - X
3. Center of Gravity - Y
4. Baseline Shift
5. Energy: local difference in brightness, or square of brightness
6. Dissimilarity: the weights with which instensities move linearly away from the diagonal
7. Haralick: quantify an image based on texture
8. Kurtosis: Kurtosis is a measure of the combined weight of a distribution's tails relative to the center of the distribution

Implementation

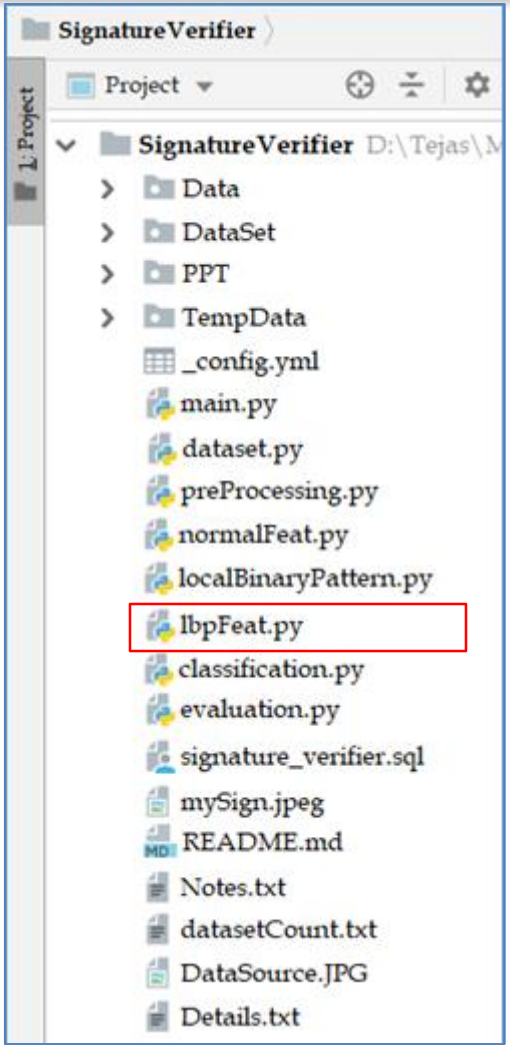


- **LocalBinaryPattern.py**
  - This python file contains function that converts the image into LBP image
  - LBP image is darker and shows off the textures of the image to help us extract them





Implementation



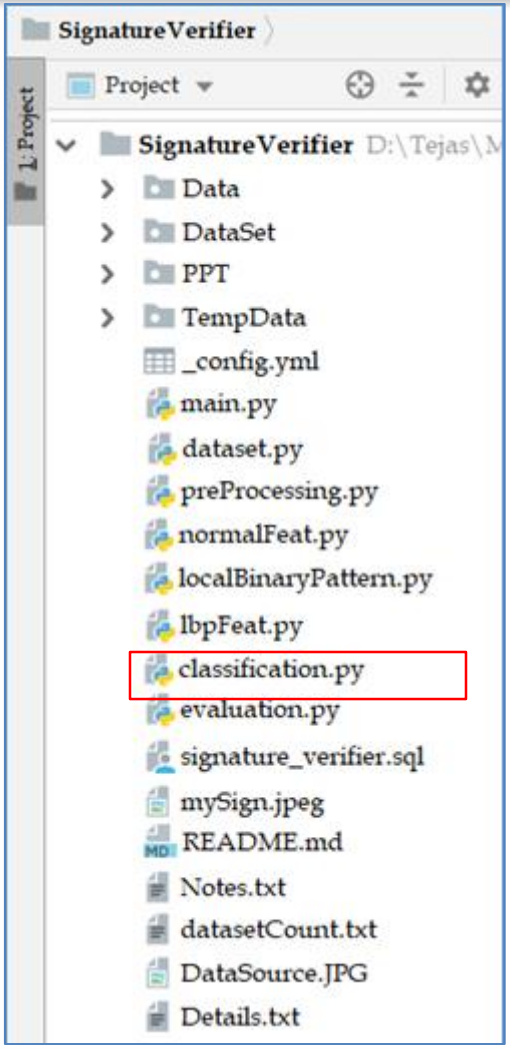
❑ LbpFeat.py

- This function extracts texture based features of the signature from the LBP images.

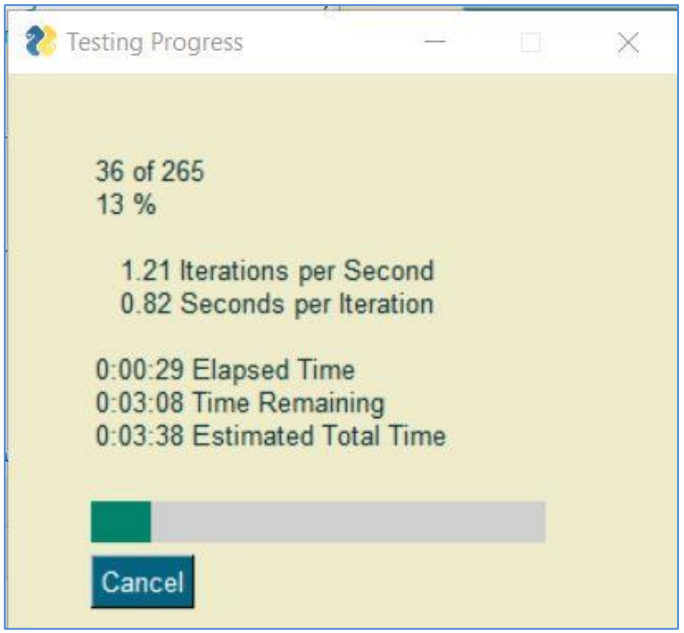
```
~~~ LBP features ~~~
Contrast:  17.679818216493544
Normalized area:  4.2712653288740245
Homogeneity:  0.8302425919185281
Energy:  0.6608064821892133
Dissimilarity:  1.0182713821221416
Haralick:  157.50156224641563
Skewness:  -5.43953922460237
Kurtosis:  33.18020199220615
```

1. Contrast:
2. Normalized Area
3. Energy
4. Dissimilarity
5. Haralick
6. Kurtosis
7. Skewness: asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right
8. Homogeneity: value that calculates the tightness of distribution of the elements in the GLCM to the GLCM diagonal

Implementation

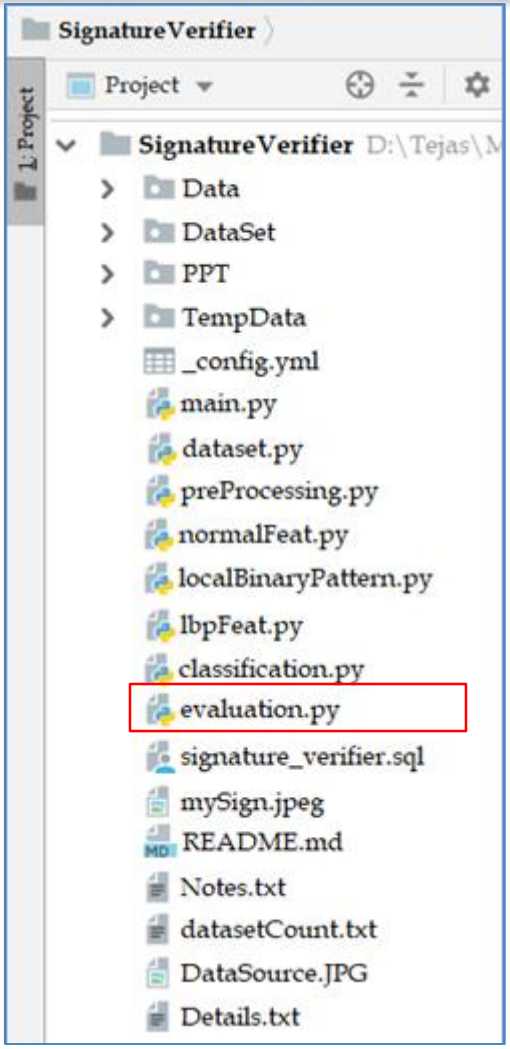


- **Classification.py**
  - This python file contains KNN classification function which classifies the given test data to a class.
  - KNN algorithm works on the Euclidian distance based approach where the classes of the K nearest neighbours is given to the test data vector.



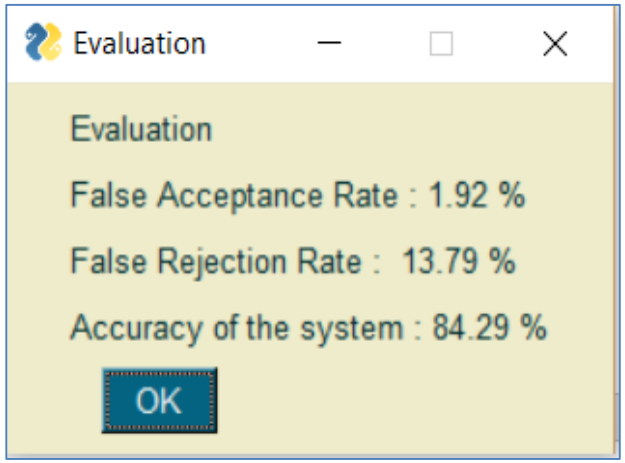

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Image file :  A_forged_21.png
                ■
                ■
                ■

Actual Class:  A_forged
Decision:  Rejected
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

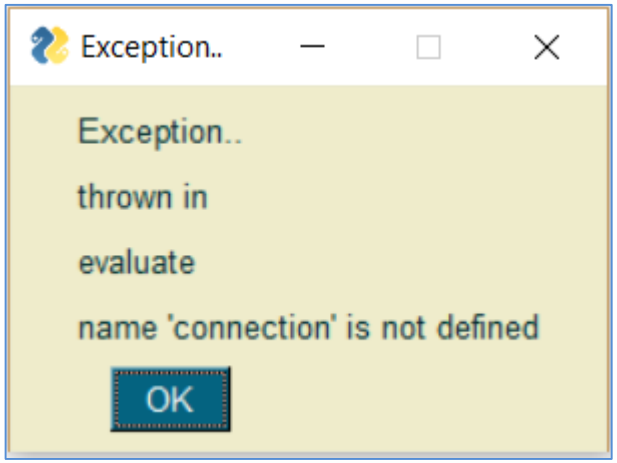
Implementation



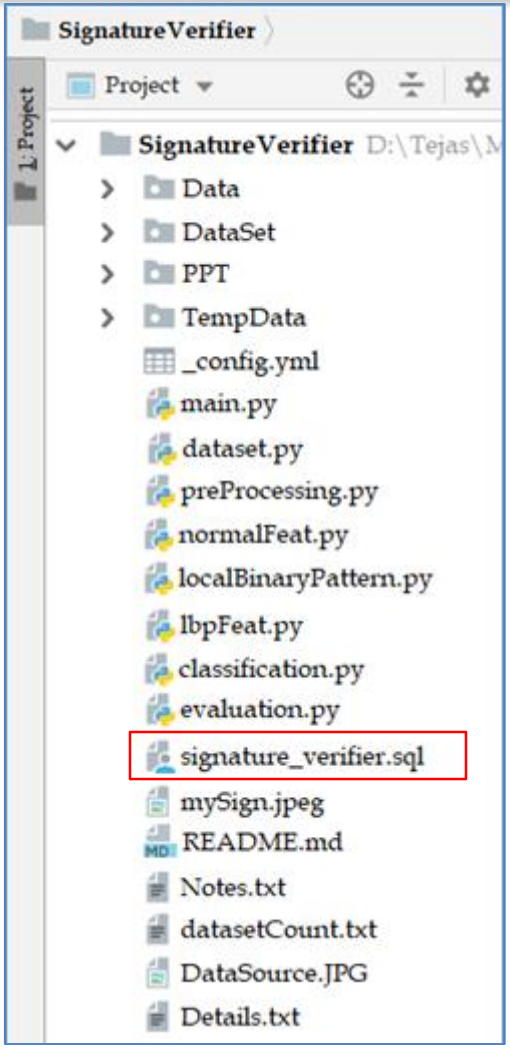
❑ Evaluation.py

- The evaluation parameter used for measuring accuracy of the system is recognition rate. We find the FAR and FRR which stand for False Acceptance Rate and False Rejection Rate. The number of falsely accepted images over the total images is FAR and the number of falsely rejected images over the total images is FRR
- In our experimentation, we find our accuracy to be highest with 84.29% at K = 22 in our KNN classifier. The table below shows the different recognition rates for different values for K.

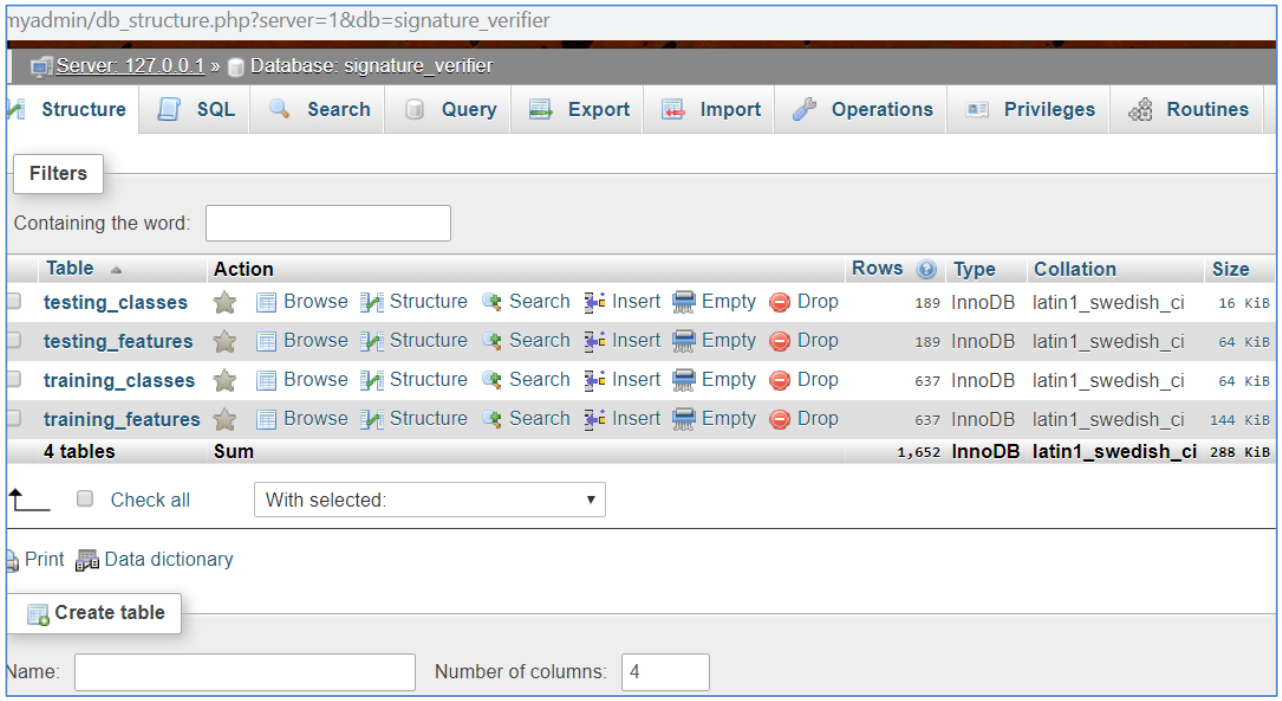




Implementation



- **Signature_verifier.sql**
  - This has SQL queries for creating database, creating tables, inserting entries into the tables.
  - Main.py also inserts and reads data into the database one row at a time



Comparative study

Aftaab99 / OfflineSignatureVerification

Watch 0 | Unstar 3 | Fork 0

Code | Issues 1 | Pull requests 0 | Projects 0 | Wiki | Insights

Writer independent offline signature verification using convolutional siamese networks

pytorch | siamese-cnn | siamese-network | siamese-neural-network | signature-verification | python3 | machine-learning | contrastive-loss

7 commits | 1 branch | 0 releases | 1 contributor | MIT

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

Aftaab99 Create LICENSE.md — Latest commit 0b28c26 on Feb 6

static/images	bug fiex	4 months ago
templates	bug fiex	4 months ago
.gitignore	Added script to download data. Reformated code	2 months ago
Dataloaders.py	Added script to download data. Reformated code	2 months ago
DownloadData.py	Added script to download data. Reformated code	2 months ago
LICENSE.md	Create LICENSE.md	2 months ago
Model.py	Added script to download data. Reformated code	2 months ago
Preprocessing.py	bug fiex	4 months ago

Comparative study

❑ Preprocessing

- Grayscale, invertion and scaled down to 0 or up to Images were grouped in pairs of genuine and forged images, where the label was 1 if both were genuine and of the same writer and 0 otherwise.

❑ Model used

- Convolutional Siamese network. Siamese neural network is a class of neural network architectures that contain two or more *identical* subnetworks along with Constrastive loss function.
- This network uses Euclidian distance as the distance metric for comparing the output feature vectors, similar to how we used it in our K nearest neighbor classification algorithm.

❑ Evaluation

- The model acheived an accuracy of 73.34% on the CEDAR signature dataset

```
Run:  Train  Test
Max accuracy for batch 246 = 0.6875 at d = 0.26373077356815355
Max accuracy for batch 247 = 0.7142857142857143 at d = 0.22994920825958265
Max accuracy for batch 248 = 0.8571428571428572 at d = 0.27605368983745593
Max accuracy for batch 249 = 0.9375 at d = 0.16995522955060016
Max accuracy for batch 250 = 0.7857142857142857 at d = 0.2559534368515016
Max accuracy for batch 251 = 0.6833333333333333 at d = 0.1791844154894353
Max accuracy for batch 252 = 0.8461538461538461 at d = 0.2619353567361833
D:/Tejas/M.Tech/Project/GITHUB-OfflineSignatureVerification-master/OfflineSign
Max accuracy for batch 253 = 0 at d = 0
  tpr = float(np.sum(labels[idx1] == 1)) / nsame
```

https://github.com/TeeeJaey/SignatureVerifier

README.md

SignatureVerifier - Python project

- SignatureVerifier is a project that runs on Python 3.X and MySQL
- Performs Handwritten signature Verification
- Uses Image Processing and Machine Learning

Requirements

Python	Package
cv2	pypi package 4.0.0.21
numpy	pypi package 1.16.2
imutils	pypi package 0.5.2
pymysql	pypi package 0.9.3
os	pypi package 4.2.0
scipy	pypi package 1.2.1
mahotas	pypi package 1.4.5
matplotlib	pypi package 3.0.3
PySimpleGUI	pypi package 3.25.0

References

[1] S. F. A. Zaidi and S. Mohammed, "Biometric Handwritten Signature Recognition," 2007.

[2] D. Morocho, A. Morales, J. Fierrez, and R. Vera-Rodriguez, "Towards human-assisted signature recognition: Improving biometric systems through attribute-based recognition," *ISBA 2016 - IEEE Int. Conf. Identity, Secur. Behav. Anal.*, 2016.

[3] S. A. Angadi, S. Gour, and G. Bhajantri, "Offline Signature Recognition System Using Radon Transform," *2014 Fifth Int. Conf. Signal Image Process.*, pp. 56–61, 2014.

[4] S. Hangai, S. Yamanaka, and T. Hammamoto, "ON-LINE SIGNATURE VERIFICATION BASED ON ALTITUDE AND DIRECTION OF PEN MOVEMENT," *Proc. 15th Int. Conf. Pattern Recognition. ICPR-2000*, vol. 3, no. l, pp. 479–482, 2000.

[5] I. V Anikin and E. S. Anisimova, "Handwritten signature recognition method based on fuzzy logic," *2016 Dyn. Syst. Mech. Mach.*, 2016.

[6] E. Ozgunduz, T. Senturk, and M. E. Karsligil, "Off-line signature verification and recognition by support vector machine," *13th Eur. Signal Process. Conf.*, no. 90, pp. 1–4, 2005.

[7] S. A. Angadi and S. Gour, "Euclidean Distance Based Offline Signature Recognition System Using Global and Local Wavelet Features," *2014 Fifth Int. Conf. Signal Image Process.*, pp. 87–91, 2014.

[8] Ruangroj Sa-Ardship and K. Woraratpanya, "Offline Handwritten Signature Recognition Using Adaptive Variance Reduction," *7th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE), Chiang Mai, Thail. Offline*, pp. 258–262, 2015.

[9] M. A. Djoudjai, Y. Chibani, and N. Abbas, "Offline signature identification using the histogram of symbolic representation," *5th Int. Conf. Electr. Eng. - Boumerdes*, pp. 1–5, 2017.

[10] A. B. Jagtap and R. S. Hegadi, "Offline Handwritten Signature Recognition Based on Upper and Lower Envelope Using Eigen Values," *Proc. - 2nd World Congr. Comput. Commun. Technol. WCCCT 2017*, pp. 223–226, 2017.

References

[11] T. Marušić, Ž. Marušić, and Ž. Šeremet, "Identification of authors of documents based on offline signature recognition," *MIPRO*, no. May, pp. 25–29, 2015.

[12] S. L. Karanjkar and P. N. Vasambekar, "Signature Recognition on Bank cheques using ANN," *IEEE Int. WIE Conf. Electr. Comput. Eng.*, no. December, 2016.

[13] G. S. Prakash and S. Sharma, "Computer Vision & Fuzzy Logic based Offline Signature Verification and Forgery Detection," *IEEE Int. Conf. Comput. Intell. Comput. Res.*, 2014.

[14] M. S. Shirdhonkar and M. B. Kokare, "Document image retrieval using signature as query," *2011 2nd Int. Conf. Comput. Commun. Technol. ICCCT-2011*, pp. 66–70, 2011.

[15] R. Sa-Ardship and K. Woraratpanya, "Offline Handwritten Signature Recognition Using Polar-Scale Normalization," *8th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE), Yogyakarta, Indones.*, pp. 3–7, 2016.

[16] A. Piyush Shanker and A. N. Rajagopalan, "Off-line signature verification using DTW," *Pattern Recognit. Lett.*, vol. 28, no. 12, pp. 1407–1414, 2007.

[17] Nancy and P. G. Goyal, "Signature Processing in Handwritten Bank Cheque Images," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 2, no. 5, pp. 1239–1243, 2014.

[18] A. T. Nasser and N. Dogru, "Signature recognition by using SIFT and SURF with SVM basic on RBF for voting online," *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018–Janua, pp. 1–5, 2017.

[19] A. Kumar and K. Bhatia, "A Robust Offline Handwritten Signature Verification System Using Writer Independent Approach," *3rd Int. Conf. Adv. Comput. Autom.*, 2017.

[20] H. Anand, "Enhanced Signature Verification and RECOGNITION USING MATLAB," *Int. J. Innov. Res. Adv. Eng.*, vol. 1, no. 4, pp. 88–94, 2014.

[21] B. H. Shekar and R. K. Bharathi, "Eigen-signature: A robust and an efficient offline signature verification algorithm," *Int. Conf. Recent Trends Inf. Technol. ICRTIT 2011*, pp. 134–138, 2011.

[22] A. R. Rahardika, "Global Features Selection for Dynamic Signature Verification," *Int. Conf. Inf. Commun. Technol. Glob.*, pp. 348–354, 2018.

[23] T. Handhayani, A. R. Yohannis, and Lely Hiryanto, "Hand Signature and Handwriting Recognition as Identification of the Writer using Gray Level Co- Occurrence Matrix and Bootstrap," *Intell. Syst. Conf.*, no. September, pp. 1103–1110, 2017.

References

[24] A. Beresneva, A. Epishkina, and D. Shingalova, "Handwritten Signature Attributes for its Verification," pp. 1477–1480, 2018.

[25] M. P. Patil, Bryan Almeida, Niketa Chettiar, and Joyal Babu, "Offline Signature Recognition System using Histogram of Oriented Gradients," *Int. Conf. Adv. Comput. Commun. Control*, 2017.

[26] M. M. Kumar and N. B. Puhan, "Offline signature verification using the trace transform," *2014 IEEE Int. Adv. Comput. Conf.*, pp. 1066–1070, 2014.

[27] O. Miguel-Hurtado, L. Mengibar-Pozo, M. G. Lorenz, and J. Liu-Jimenez, "On-Line Signature Verification by Dynamic Time Warping and Gaussian Mixture Models," *2007 41st Annu. IEEE Int. Carnahan Conf. Secur. Technol.*, pp. 23–29, 2007.

[28] M. Ferrer and J. Vargas, "Robustness of offline signature verification based on gray level features," *IEEE Trans. Inf. FORENSICS Secur.*, vol. 7, no. 3, pp. 966–977, 2012.

[29] B. Erkmen, N. Kahraman, R. A. Vural, and T. Yildirim, "Conic section function neural network circuitry for offline signature recognition," *IEEE Trans. Neural Networks*, vol. 21, no. 4, pp. 667–672, 2010.

[30] M. A. Ferrer, A. Morales, and J. F. Vargas, "Off-line signature verification using local patterns," *2nd Natl. Conf. Telecommun.*, pp. 1–6, 2011.

[31] M. Tahir and M. U. Akram, "Online Signature Verification using Hybrid Features," *Conf. Inf. Commun. Technol. Soc. Online*, pp. 11–16, 2018.

[32] M. Pal, S. Bhattacharyya, and T. Sarkar, "Euler number based feature extraction technique for Gender Discrimination from offline Hindi signature using SVM & BPNN classifier," *2018 Emerg. Trends Electron. Devices Comput. Tech.*, pp. 1–6, 2004.

[33] W. Pan and G. Chen, "A Method of Off-line Signature Verification for Digital Forensics," *12th Int. Conf. Nat. Comput. Fuzzy Syst. Knowl. Discov.*, pp. 488–493, 2016.

References

[34] M. A. Ferrer, J. B. Alonso, and C. M. Travieso, "Offline geometric parameters for automatic signature verification using fixed-point arithmetic," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 993–997, 2005.

[35] S. A. Farimani and M. V. Jahan, "An HMM for Online Signature Verification Based on Velocity and Hand Movement Directions," *Iran. Jt. Congr. Fuzzy Intell. Syst.*, pp. 205–209, 2018.

[36] G. Pirlo and D. Impedovo, "Verification of static signatures by optical flow analysis," *IEEE Trans. Human-Machine Syst.*, vol. 43, no. 5, pp. 499–505, 2013.

[37] A. Alaei, S. Pal, U. Pal, and M. Blumenstein, "An Efficient Signature Verification Method Based on an Interval Symbolic Representation and a Fuzzy Similarity Measure," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 10, pp. 2360–2372, 2017.

[38] D. S. Guru and H. N. Prakash, "Online Signature Verification and Recognition: An Approach based on Symbolic Representation.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1059–73, 2009.

[39] A. Sharma and S. Sundaram, "On the Exploration of Information from the DTW Cost Matrix for Online Signature Verification," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 611–624, 2018.

[40] G. Pirlo, V. Cuccovillo, M. Diaz-Cabrera, D. Impedovo, and P. Mignone, "Multidomain Verification of Dynamic Signatures Using Local Stability Analysis," *IEEE Trans. Human-Machine Syst.*, vol. 45, no. 6, pp. 805–810, 2015.

[41] Python : https://www.python.org/

[42] Dataset source :- http://www.iapr-tc11.org/mediawiki/index.php?title=Datasets_List#Handwritten%20Documents

[45] Github's existing system used for comparison https://github.com/Aftaab99/OfflineSignatureVerification

Thank You ☺

Any Questions?