Helwan University

كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

FACULTY OF COMPUTERS AND ARTIFICIAL INTELLIGENCE

COMPUTER SCIENCE DEPARTMENT

2021/2022

CS 361 ARTIFICIAL INTELLIGENCE (AI)

	ID	NAME	LEVEL	DEPARTMENT
1.	201900022	Ahmed Hatem Fathy Khedr	3	CS
2.	201900174	Omnia Sayed Hamed Abdelrahman	3	CS
3.	201900309	Rewaa Ragab Mahmoud Mogazy	3	CS
4.	201900274	Hala TagElser Rabia AbdElaziz	3	CS
5.	201900245	George Elamir Mkram Sleem	3	CS
6.	201900863	Mo'men Emad Elden Abdel Fattah Abdel Ghaffar	3	CS

INTRODUCTION

WHAT IS OCR?

OCR stands for Optical Character Recognition. It is a widespread technology to recognize text inside images, such as scanned documents and photos. OCR technology is used to convert virtually any kind of image containing written text (typed, handwritten, or printed) into machine-readable text data.

HOW DOES IT WORK?

A text classification is a well-formed process using various measurable properties and computerized logical procedure to fetch a pattern from different classes. There are Two basic steps of OCR:

Step 1: Image Pre-Processing in OCR

OCR software often pre-processes images to improve the chances of successful recognition. The aim of image pre-processing is an improvement of the actual image data. In this way, unwanted distortions are suppressed and specific image features are enhanced. These two processes are important for the following steps.

Step 2: Character Recognition in OCR

For the actual character recognition, it is important to understand what "feature extraction" is. When the input data is too large to be processed, only a reduced set of features is selected. The features selected are expected to be the important ones while those that are suspected to be redundant are ignored. By using the reduced set of data instead of the initial large one, the performance is increased. For the process of OCR, this is important as the algorithm must detect specific portions or shapes of a digitized image or video stream. Different algorithms were used and evaluated in this investigation such as k-Nearest Neighbor (k-NN), Support-Vector machine (SVM), Naïve Bayes, Decision Tree, Random Forest, and Convolution Neural Network (CNN). We will focus on explaining OCR Handwritten English Letters classifiers using Random Forests and Decision Tree.

RANDOM FOREST ALGORITHM

The Random Forest algorithm is an ensemble learning method that can be used for both regression and classification machine learning problems. It works by building a multitude of decision trees during training, and then using a combined output of these trees to determine the output during a prediction. The Random Forest algorithm has number of advantages such as immunity to noisy datasets, unimportant features and mislabeled input data.

DECISION TREES (DTS)

Are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

MAIN FUNCTIONALITY OF OCR:

Forms containing characters images can be scanned through scanner and then recognition engine of the OCR system interpret the images and turn images of handwritten or printed characters into ASCII data. In addition, OCR uses a modular architecture that is open, scalable and workflow controlled. It includes forms definition, scanning, image pre-processing, and recognition capabilities.

APPLICATIONS USING OCR IN THE MARKET

Handwriting recognition has been around for decades, starting with the PalmPilot and the Newton MessagePad from the 1990s. These popular PDAs recognized character input with a stylus. You had to write legibly for the Newton to recognize lettering, and you had to learn the Graffiti language for the Palm to do the same.

Many mobile apps let you draw letters, strokes and shapes onscreen with iOS and Android devices, but only a few recognize, translate or digitize that input.

The following apps can automatically recognize and digitize your handwriting. Some of the apps are free, some operate with integrated keyboards, while others have in-app purchases or fees, or rely on the MyScript AI handwriting recognition and digital ink management engine.

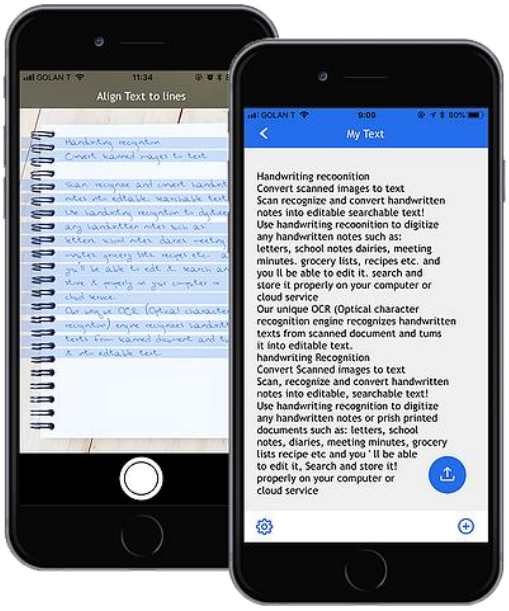
HOW DO HANDWRITING RECOGNITION APPS WORK?

Thanks to the high-power computational power that today's mobile devices and desktop computers harness, handwriting recognition technology can help digitize a person's unique handwriting style. Born from an older type of technology known as optical character recognition, handwriting recognition tech converts the written word into digital approximations.

After scanning a document, or once someone writes a note on a touchscreen, the device in question attempts to separate each letter to compare them to a database of letters that the handwriting may likely be. Some of the more advanced handwriting recognition software utilizes complex algorithms and compares written letters to an extensive database to identify characters before churning out a digital approximation. Over time, some handwriting recognition tech will also get better at deciphering even some of the hardest to read chicken scratch.

1. PEN TO PRINT (FREE)

Credit: Serendi LTD



In a variation on the handwriting recognition concept, Pen to Print reads scanned handwritten documents and converts them into editable, searchable digital text that can be stored on your device or within a cloud service.

The app's handwriting OCR (optical character recognition) engine extracts text from paper documents, like letters, school notes, meeting notes, and grocery lists, allowing those who prefer to write in longhand the freedom to continue.

The handwriting recognition system works with block letters, cursive and script. Thierry Tremblay, CEO and founder of Kohezion, said he highly recommends the application.

"It is very simple, straightforward, user-friendly, fast, and pretty accurate," Tremblay said of Pen to Print. "It's working great on both my iPad and over my smartphones. One of my co-workers is also practicing it on her Android tablet and also very gratified."

A premium monthly and yearly subscription plans let you save your text to a file, copy, email, add to Notes, or share on Message, WhatsApp, Hangout, WeChat, Messenger, and Telegram. You can transfer the text to word processors like Microsoft Word or Google Docs, or export to Evernote, OneNote or Google Keep. The app works with iOS 9 or later and Android 4.4 and later in English, Portuguese, and Spanish.

Available on both Google Play & App Store

APPLICATION FUNCTIONALITIES / FEATURES:

- Recognize and Convert handwritten documents into digital text that can be edited, searched and stored on any device or cloud service.
- Handwritten notes, such as letters, school notes, diaries, meeting minutes, grocery lists, recipes etc can now be scanned and converted, by our handwriting recognition engine, from image to text, available for use in any digital platform.
- You can edit your text, save it to files, copy it, email it, add to Notes, or share on messaging apps available on your device. Then use your text in any word processor like Microsoft Word, Google Docs and similar, or export it to a note organizing app like Evernote, OneNote, Google Keep or similar.

2. MAZEC ($12.99)

Credit: Mazec

Mazec is a keyboard app that provides handwriting conversion to text in a variety of apps like email, notes and social posts. Semantic databases, combined with the MyScript engine, let you search, browse the web and complete online forms. You can choose the font size, auto-scroll area width, word spacing and more. Updates improve the built-in dictionaries and streamline Apple Pencil usability.

As you begin to write, Mazec displays predictive suggestions and phrases to choose from so you usually don't have to write out an entire word before the app completes it. Mazec intelligently detects your choices, learns specific phrases and even offers emojis – if you write "emoji" or a recognized emoji category name. Mazec supports 12 languages, but you must buy a language pack if you want to use any other than the one you signed in with.

Available on both Google Play & App Store

FEATURES:

- One of a kind handwriting input technology
- Rich Semantic Dictionaries of Popular Phrases
- Auto and Manual Learning Dictionary
- Emoji graphics can be added on the fly
- Cross application handwriting recognition
- There are 3 modes of operation in mazec:

3. GOOGLE HANDWRITING INPUT (FREE)

Credit: Google

Google Handwriting Input, an Android-only app, translates your scribbles directly onscreen as you write. Upon installing the app, you get a few setup panes where you can choose your language and an optional keyboard, which lets you use the utility with other text input apps. In addition to supporting over 100 languages, it allows easy input of ideographic lettering and voice, and recognizes emoji-style drawings.

This app understands the sloppiest scrawl quite well and offers predictive text at the top of the window to let you tweak its interpretation – or you can correct spelling directly on the text output. One proponent of Google Handwriting Input is Emily Deaton, a financial journalist at Let Me Bank.

"Like with any handwriting app, it will take time to learn how you write, but the AI works very well - even against my very messy handwriting," Deaton said. "It translates as you write, perfect for when you are on the go and need something written out quickly."

An online feature sends information to Google to decipher your handwriting to improve the recognition engine, but you can opt out of this in favor of more private local device translation. While there is no specific iOS version, you can enable Google Handwrite in mobile Safari, Chrome, or the Gboard to search by writing with your finger or stylus.

Available on Google Play

FEATURES:

- It supports multiple languages.
- It is easy to use and has a clean user-interface.
- It is a free tool that can be used both online and offline.
- You can easily integrate it with the Chrome browser to use online.
- It allows us to type text through handwriting by drawing characters with our fingers. This feature is currently only available in its Chrome extension.
- Convert text sound/phonetics to the best match text languages, i.e., "namaste" will convert to "नमस्ते" in Hindi.

USEFUL PAPERS USED

1. HANDWRITTEN DIGIT CLASSIFICATION USING MACHINE LEARNING MODELS

New challenges in pattern recognition and machine learning have arisen because of the rapid increase of new documents and multimedia news. Handwritten digit recognition has long been an unsolved problem in the field of pattern recognition. Because everyone in the world has their own writing style, handwriting recognition is one of the most fascinating research projects currently underway. It is the computer's ability to automatically recognise and understand handwritten digits or characters. The primary goal of this paper is to compare various classification models in order to provide efficient and reliable techniques for recognising handwritten numerals. The MNIST dataset, which contains 70000 handwritten digits, is widely used in this recognition process. The emphasis of this research is feature extraction and classification. The quality of the features can affect a classifier's performance just as much as the classifier itself. The accuracy and performance measures of the algorithms Support Vector Classification model (SVC), Logistic Regression model, and Random Forest Classification model are examined in this study.

RESEARCH METHODOLOGY

1. ***DESCRIPTION OF THE DATASET***
   The MNIST dataset consists of 70,000 handwritten digits, divided into 60,000 training examples and 10,000 testing samples. The images in the MNIST dataset are present in form of an array consisting of 28x28 values representing an image along with their labels.
2. ***DATA PRE-PROCESSING***
   Images at the binary level are required for the variables described in this paper to make handwritten digit categorization. The binarization method implies that images have two types of pixels: foreground (or white pixels with highest intensity, i.e., 1,) and background (or black pixels with minimum intensity, i.e., 0). The method's purpose is to identify all pixels with values greater than or equal to the set threshold as white and all others as black.
3. ***IMPLEMENTATION***
   1. ***Logistic Regression Model***
      On test data, Logistic Regression yielded an accuracy of 88.97%.
   2. ***Random Forest Classifier***
      The Test Data Set obtained accuracy of 96.3% on MNIST data set.
   3. ***Support Vector Machine (SVM) Classifier***
      The Test Data Set obtained accuracy of 98.3% on MNIST data set.[1]

2. USING RANDOM FORESTS FOR HANDWRITTEN DIGIT RECOGNITION

In recent years, Multiple Classifier Systems, particularly Bagging, Boosting, and Random Subspaces, have sparked increased interest in the Pattern Recognition discipline. These strategies try to produce diversity at several levels in order to induce an ensemble of classifiers. Breiman created a new family of approaches dubbed Random Forest in 2001 based on this premise. Our research intends to investigate those methods using a pragmatic approach in order to give practitioners with parameter setting guidelines. On the MNIST handwritten digits database, we tested the Forest-RI algorithm, which is regarded the Random Forest reference method. In this paper, we go through the basics of Random Forests and look at some of the methods that have been offered in the literature. Following that, we will present our experimental approach and outcomes. Finally, based on their parameter adjustment, we draw some inferences about Random Forest global behaviour.[2]

3. ARABIC FONT RECOGNITION USING DECISION TREES BUILT FROM COMMON WORDS

We introduce an a priori Arabic optical Font Recognition AFR method. The fundamental aim is to be able to detect the typefaces of some often-used Arabic words. Because these components of a literary material almost all share the same font, they can be extrapolated to lines, paragraphs, or nearby non-common words once their fonts are recognised. To distinguish Arabic fonts, we used a decision tree. The tree is learned using a collection of 48 characteristics. Horizontal projections, Walsh coefficients, invariant moments, and geometrical properties are among these aspects. A total of 36 fonts are examined. The success rate is 90.8 percent overall. Some fonts have a 100% success rate. The average amount of time it takes to recognise the word font is 0.30 seconds.[3]

4.IMPROVEMENT OF RANDOM FOREST CLASSIFIER THROUGH LOCALIZATION OF PERSIAN HANDWRITTEN OCR

The random forest (RF) classifier is a decision tree-based ensemble classifier. However, the random forest is a very strong classifier, with accuracy rates comparable to most commonly used classifiers, because to the concurrent operations of multiple classifiers and the use of randomness in sample and feature selection. RF is used in this research to recognise Persian handwritten characters, despite the fact that the use of random forest on handwritten digits has been considered previously. In this research, a random forest classifier is utilised to distinguish 33 unique Persian letters written by hand. The random forest (RF) classifier is a reliable and quick classifier that can handle a high number of feature variables. It is also suggested that the RF's efficiency be improved using a localization procedure that transforms decision trees into multi-class trees.

EXPERIMENTAL RESULTS

- The database consists of 2838 binary image samples of 33 Persian single characters written by 86 different writers. A complete set of 1650 samples was used to train the classifier and the rest 1180 were used as the test set to verify it‘s efficiency.

- The highest accuracy was reached using 120 trees and 7 random splitting features with 87% accuracy. The total time needed for training and testing a 120-tree forest with 2838 feature vectors of size 81, is just about 5s.

- The implementation of the localization idea raises the accuracy rate to 94%.[4]

5. THE PERFORMANCE EVALUATION OF DEEP LEARNING CLASSIFIER TO RECOGNIZE DEVANAGARI HANDWRITTEN CHARACTERS AND NUMERICAL

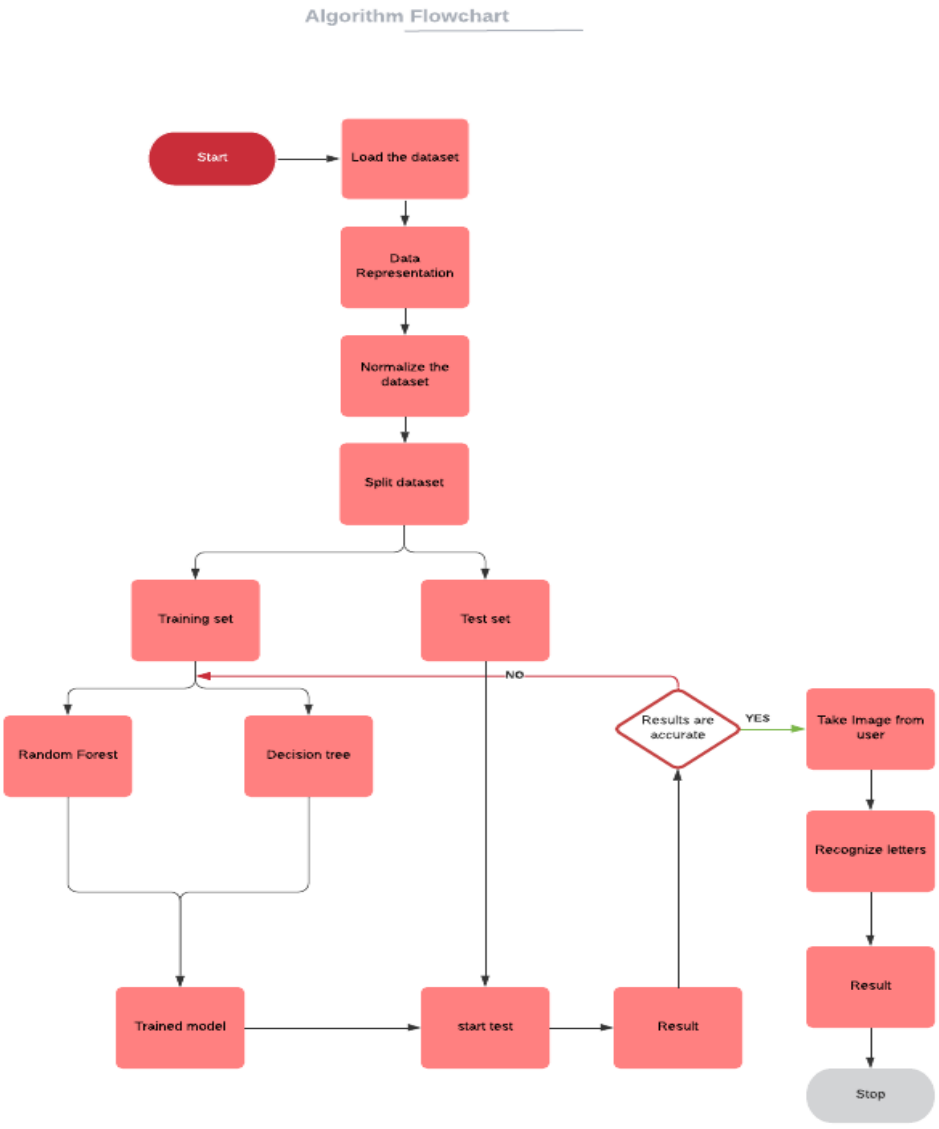
The purpose of this work is to give a result-based comparative research of various classifiers, as well as the best recognition of results calculation using Devanagari Handwritten characters and numerical values. k-Nearest Neighbor (k-NN), Support-Vector Machine (SVM), Nave Bayes, Decision Tree, Random Forest, and Convolution Neural Network were among the classifiers utilised and assessed in this study (CNN). This paper used an unbiased dataset with 123 samples consisting of 123 characters and 123 numerical values to achieve the experiment's goal. The performance of the classifiers was evaluated using Python 3.0 and the sciket learn machine learning open-source environment module. The accuracy rate, True/False acceptance rate, True/False rejection rate, and the area covered by the receiver operating characteristic curve were all used to assess the classifiers' performance.

EXPERIMENTAL RESULTS

- A dataset of 92,000 examples has been considered for search results (72,000 images in consonant/character datasets and 20,000 numeral value dataset) for search results
- The accuracy of recognition is 87.9% , 82.5% , 75.4% , 74.7% , 70.7% , *and* 66.3% with Random Forest, SVM, CNN, K-NN, Decision Tree, and Naïve Bayes respectively have been completed separately with the classifier.[5]

PROPOSED SOLUTION

Our proposed solution is a software that scans an image containing a Handwritten English Letters that can be scanned through scanner and then recognition engine of the OCR system interprets the images and turn images of handwritten or printed characters into ASCII data. Moreover, our software uses a modular architecture that is open, scalable and workflow controlled. It includes forms definition, scanning, image pre-processing, and recognition capabilities.



DATASET USED

The EMNIST Letters train Dataset (88,800)

The Dataset merges a balanced set of the uppercase and lowercase letters into a single 26-class task.

LINK: https://www.kaggle.com/crawford/emnist?select=emnist-letters-train.csv

DEVELOPMENT PLATFORM

1. **Using python 3.0**
2. **Google colab**
3. **Libraries:**
   - Glob
   - Pandas
   - NumPy
   - Matplotlib.pyplot
   - Google.colab
   - Sklearn.preprocessing
   - Sklearn.model_selection
   - Sklearn.ensemble
   - Sklearn.tree
   - Sklearn.metrics
   - Seaborn
   - Time
   - CV2

DETAILS OF THE ALGORITHM

1. Import the libraries.
2. Read the dataset from csv file using pandas.
3. Rename the column with position 0 to be the label of the data.
4. We observed from the description of the dataset that the minimum value is equal to 1 that points to the A Letter and maximum value is equal to 26 that points to the Z letter.

5. Our dataset information is:
   Range Index: *372450 entries, 0 to 372449*
   Columns: *785 entries, label to 0.648*
   Dtypes: *int64(785)*
   Memory usage: *531.8 MB*
6. We defined x variable to contain all the dataset without the label column and variable y to contain the label.
7. Cross Validation is used in our model as it results in More accurate estimate of out-of-sample accuracy and more "efficient" use of data as every observation is used for both training and testing. From sklearn.model_selection we imported KFold to split our dataset into train and test by using **160** KFold splits.
8. From sklearn.preprocessing we import StandardScaler to transform our data such that its distribution will have a mean value 0 and standard deviation of 1.
9. From sklearn.ensemble we import RandomForestClassifier to build our Random Forest model and use 24 tree with:

   Number of estimators*: 150*
   Criterion: *entropy*
   Random state: *42*
   Min samples split*: 6*
   Min sample leaf*: 1*
   Max features*: auto*
   Max depth*: 90*
   Bootstrap: *false*

10. From sklearn.tree we import DecisionTreeClassifier to build our Decision Tree model with:

    criterion: entropy
    random state: 42
    splitter: best

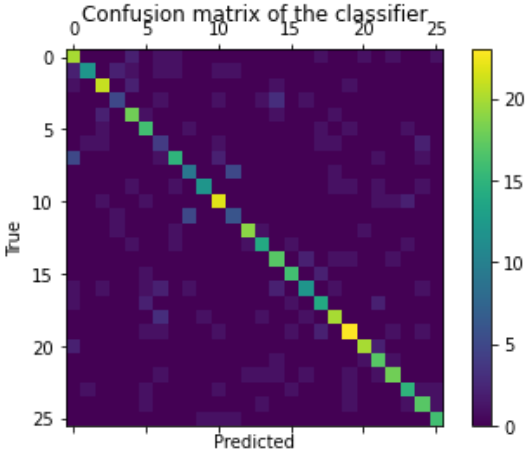
11. We trained and fitted our 2 models.
12. From sklearn.metrics we imported accuracy_score, confusion_matrix, classification_report and we used them to define a function that print the accuracy score, confusion matrix and classification report for both train and test data, then we call the function for both algorithms to get the result of them.
13. We compared between the actual results and predicted one.
14. We read an external image from the drive to test our model and apply some pre-processing methods to make it suitable.
15. We declared a dictionary that have the mapping of all the letters so we can convert the number to its corresponding letter.
16. The user can now see the tested result of the image.

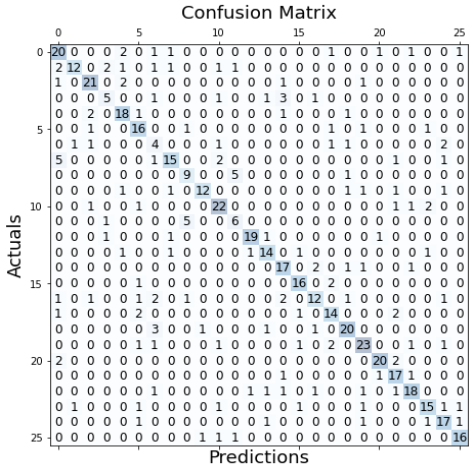
EXPERIMENTS & RESULTS

DECISION TREE

Train Accuracy Score: **100%**

Test Accuracy Score: **71.84%.**

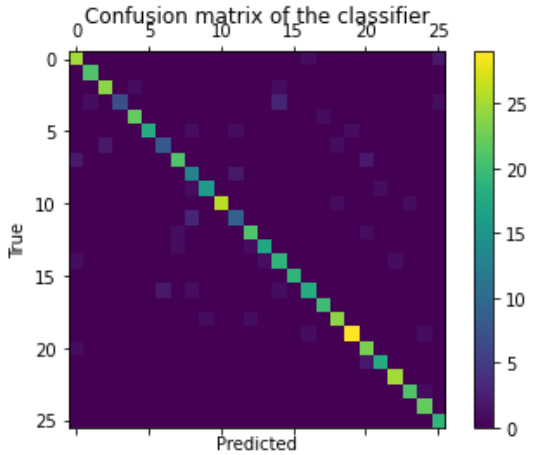


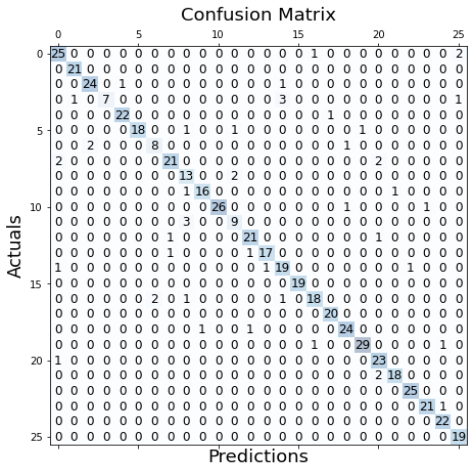


RANDOM FOREST

Train Accuracy Score: **100%**

Test Accuracy Score: **91.16%**





ANALYSIS, DISCUSSION, AND FUTURE WORK

ANALYSIS OF THE RESULTS, WHAT ARE THE INSIGHTS?

Random Forest Algorithm achieves a better accuracy than Decision Tree Algorithm as Random Forest can generalize over the data in a better way. The randomized feature selection of it makes it much more accurate than a decision tree. Random Forest is way better than Decision Tree as it can predict the output at a higher percentage rate.

WHAT ARE THE ADVANTAGES / DISADVANTAGES?

The advantages of our model is that it is based on the bagging algorithm and uses Ensemble Learning technique. It creates as many trees on the subset of the data and combines the output of all the trees. In this way Random Forest reduces overfitting problem in decision trees and also reduces the variance and therefore improves the accuracy. Moreover, Random Forest is comparatively less impacted by noise.

The main disadvantage of our model is that it does not predict the tested samples 100% accurate. Obviously it is not the preferable algorithm to be used in such cases. Furthermore, the Complexity is not good as it creates a lot of trees and combines their outputs. By default, Random Forest creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources. Not only that but also, it takes Longer Training Period as it requires much more time to train.

WHY DID THE ALGORITHM BEHAVE IN SUCH A WAY?

The main limitation of random forest is that a large number of trees can make the algorithm too slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained.

WHAT MIGHT BE THE FUTURE MODIFICATIONS YOU'D LIKE TO TRY WHEN SOLVING THIS PROBLEM?

We will go through Deep Learning since it is good for solving complex problems that require discovering hidden patterns in the data, the ability to deliver high-quality results and elimination of the need for data labelling.

PAPERS REFERENCES

1. https://www.irjet.net/archives/V6/i11/IRJET-V6I11328.pdf
2. https://hal.archives-ouvertes.fr/hal-00436372/document
3. https://pdfs.semanticscholar.org/a1a1/5e64875398813b8d215601a412ff8f093e2a.pdf?_ga=2.27433020.1453436148.1640307279-2081501805.1640017447
4. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.671.9682&rep=rep1&type=pdf
5. https://ejmcm.com/article_9794_989c199fb7f32961a17205c5e2a65c40.pdf