A simple batch machine learning tool that would ease users in development to handle data with graphical support and statistical metrics.

Batch Machine Learning

STAT995 – Research Project (AUT Coursework)

Created By: Minnie Tan

StudentID: 17996397, NetworkID: xrx5385

Date: 10 June 2022

Adviser: Victor Miranda

Contents

[I. Acknowledgements 1](#_Toc105604337)

[II. Overview of the Project 1](#_Toc105604338)

[III. Abstract / Executive Summary 1](#_Toc105604339)

[IV. Objectives 2](#_Toc105604340)

[V. Research Questions 2](#_Toc105604341)

[VI. Literature Review 2](#_Toc105604342)

[VII. Scope & Limitation 3](#_Toc105604343)

[VIII. Approach 3](#_Toc105604344)

[IX. Plan of Study 3](#_Toc105604345)

[X. Timetable for Completion (including key milestones) 5](#_Toc105604346)

[XI. Design Level 8](#_Toc105604347)

[A. High Level 8](#_Toc105604348)

[B. Detailed Design 8](#_Toc105604349)

[XII. Process Functionality 9](#_Toc105604350)

[A. Pre-processor 9](#_Toc105604351)

[2. Modelling Technique 13](#_Toc105604352)

[1. Regressor 14](#_Toc105604353)

[2. Classifier 15](#_Toc105604354)

[3. Cluster 20](#_Toc105604355)

[4. Rules 23](#_Toc105604356)

[5. Sampling 24](#_Toc105604357)

[6. Natural Language Processing 26](#_Toc105604358)

[7. Neural Network 27](#_Toc105604359)

[8. Others (Supervised or Unsupervised) 38](#_Toc105604360)

[XIII. Software Use 41](#_Toc105604361)

[XIV. Conclusion, Recommendations and Future Use 46](#_Toc105604362)

[A. Conclusion 46](#_Toc105604363)

[B. Recommendations 46](#_Toc105604364)

[C. Future Use 46](#_Toc105604365)

[XV. Appendix 47](#_Toc105604366)

[A. Appendix: Disclaimer 47](#_Toc105604367)

[B. Feature Selection 47](#_Toc105604368)

[C. Model Selection 49](#_Toc105604369)

[D. Metrices Used 50](#_Toc105604370)

[XVI. References 52](#_Toc105604371)

# Acknowledgements

Firstly, I like to acknowledge my adviser Victor Miranda for giving me opportunity to do this project, his support and guidance, to be available whenever I need his advises. At the same time, providing me freedom to explore on things and guiding me when I need, to set on limitations and how to improve further.

Secondly, I’m grateful to all my professors at AUT, onsite and offsite, offline and online learnings and other organizations: Udemy, Datacamp, PyImageSearch for without those incremental learnings, this project of me creating a batch machine learning tool from scratch won’t be feasible using Object-oriented approach by interfaces.

Lastly, I like to thank God above for the provision of knowledge, finances, grace, mercies, and everything I need to make all possible.

# Overview of the Project

Everyone consumes data each day. The data growth according to statistics is that by end of 2020, 44 zettabytes make up the data in the digital universe. (Bulao, 2022) ”AI is forecast to reach $62.5 Billion in 2022 at a Compound Annual Growth Rate (CAGR) of 21.3% during 2021-2022", with demand for skills of programming, libraries and frameworks, mathematics and statistics, machine learning and deep learning, natural language processing and computer vision, data science and data analysis. (Peleton Technology, 2022) For 2022, the demand for Alexa skills is about 66k in US.(G., n.d.) By 2025, 200+ zettabytes would be in the cloud space (Bulao, 2022) with the generated data each day is predicted to reach 463 exabytes globally (seedscientific, 2021) having 75 billion IoT devices in the world.(Statista Research Department, 2016) By 2025, the global AI market is expected to be around $60B (G., n.d.), while the unmet clinical demand of AI industry outlook is $$390B (ResearchAndMarket, 2020). By 2030, global GDP will grow $15.7T due to AI (G., n.d.), 9 out of 10 people aged 6 and above would be digitally active. (seedscientific, 2021) Therefore, its pertinent to make those data into useful information through batch machine learning.

Machine learning is used to leverage the raw data to an intelligent data usable for many various applications like manufacturing, research, banking, finance, marketing, medical and other technological innovations that would drive strategic and operational decisions or find knowledge discovery or validate discovery or re-discovery existing knowledge. The batch machine learning is about creating a usable tool in processing the raw data in an automated way, minimizing user interaction, thereby, optimizing efficiency and effectiveness by having the recorded results.

# Abstract / Executive Summary

The trend and demand for Artificial Intelligence, Machine Learning grows exponentially over time including jobs and skills related to it. No one can keep up with the increasing data produced and consumed by humans to make then useful and intelligent information. As Gray Scott said, ‘There is no reason and no way that a human mind can keep up with an artificial intelligence machine by 2035.’ (Marr, n.d.) To understand the data, make an intelligent usable information out of them can be time-consuming and expensive, to find an expert to analyse it can be quite challenging and to get the information right. Eliezer Yudkowsky coined it nicely, “By far, the greatest danger of Artificial Intelligence is that people conclude too early that they understand it.” (Marr, n.d.)

To have a tool like a batch machine learning that groups together similar items to make them generic that can be setup thru input files instead of development, customize further, with metrices and graphs organized in an output folder by name, date, time can help minimize a lot of messy small scripts, development in organized ways. With the result being traceable by names, date, time, one can backtrack the historical details. This ease a lot of development routinary processes that can be automated, so more time can be allocated for more important things. Because one thing, data can vary, but it can have the same or similar format just different parameters to build the model. To have it configurable by dynamic parameters and customizable in certain ways gives power.

# Objectives

The aim of making it into batch machine learning:

* To minimize user interaction and/or development by creating a usable tool for user to add or update only input files to execute instead of scripts that produced graphs and excel metrices with minimal customization or development required
* To have saveable retrievable results organized and archived by name, date, time
* To have the statistical metrics in selection of different modelling techniques used
* To use hyper-tuning using GridSearchCV for some scikit-learn samples whenever applicable, and/or elbow method to get the optimal cluster for the clustering. Also, tensorflow don’t have automated hyper-tuning
* To provide software tool usage with some sample input files, so user can regenerate the results generated in the document via the developed tool.
* To do some deep learnings like ANN, CNN, RNN to handle image, text processing and other techniques like SOM, RBM.

# Research Questions

What can batch machine learning do to mimic an expert or professional or any person who models data to train, predict, and analyse based on the graphs and metrices to gain efficiency and effectiveness in understanding, verifying, confirming new and existing domain knowledge to become useful information?

# Literature Review

Machine learning is a branch of artificial intelligence that focuses on data usage and algorithms to imitate the way human learn while gradually improving accuracy. (IBM Cloud Education, 2020) This provides system to automatically learn and improve experience without being explicitly programmed with the aim of the computers to learn automatically without human intervention. (Selig, 2022) The learning process can be supervised and unsupervised, which can be categorized into supervised, unsupervised, semi-supervised and reinforcement algorithms. (Selig, 2022) As the data began to evolve in big data, the evolution of data analytics engine has also evolved accordingly that also impacts the speed of training and improving the analytics of machine learning. The need of grouping the machine learning into batch process also arises with the rise of data to minimize the human intervention for efficient and effective handling of the data into usable information. This is implemented in the batch machine learning to have supervised, unsupervised and reinforcement learnings.

Table 1: Evolution of data analytics engine (Rengarajan & Menon, n.d.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Generation** | **1st Generation** | **2nd Generation** | **3rd Generation** | **4th Generation** |
| Processing Types | Batch | Batch Interactive | Batch Interactive  Near-real-time time streaming  Iterative processing | Hybrid (Streaming + Batch)  Interactive  Real-time streaming  Native-iterative processing |
| Technology | APACHE HADOOP | APACHE TEZ | APACHE SPARK | APACHE FLINK |
| Transformation | MapReduce | **D**irect **A**cyclic **G**raphs (DAG) Data flows | RDD: **R**esilient **D**istributed **D**atasets | Cyclic data flows |

# Scope & Limitation

* Development and testing done are only for specified models and examples due to the variety and vastness of all the models and functionality
* Limited testing is done for image processing and limited testing in associate rule mining and natural language processing, i.e., Porter Stemmer are done for text processing using natural language toolkit (*nltk*) in Python, and CONV1D using tensorflow in CNN.
* Some functionalities are learned as its implemented
* Some models have no commonality to make it generic as there are some functions available only to some models.

# Approach

* Incremental development based on the lessons learned from AUT coursework, Datacamp, Udemy (*Complete Python Course by Telcado*, *Machine Learning A-Z: Hands-On Python & R In Data Science, Deep Learning: Convolutional Neural Networks in Python, Deep Learning A-Z™ Hands-On Artificial Neural Networks, Deep Learning: Recurrent Neural Networks in Python*) from scripts converted into tool as part of the study to create OOP Object-Oriented Programming from scripts of batch machine learning development usable tool, using same data files with addon though not similar data processing with development, customization, additional updates. The baseline from the lessons learned is used to meet the target of having a batch machine learning tool in which the development from scratch with some adaptation from lessons learnt, since there are vast features and functionalities of models in Python machine learning.
* Additional functionality is added as needed like reading of different input files, catering of many different feature selections, hyper-tuning, metrices, graphs.

# Plan of Study

The design and the implementation of the batch machine learning is to consider the usage of abstraction layered architecture using layering pattern for a top-down approach of interfaces or MVC model-view-controller to have zero-coupling if possible and composition of each item instead of decomposition of each item by its interactions, though the decomposition of items would be by its layers. (Spray, 2022) To minimize the tight coupling, interfaces are implemented with sharable modules. Instead of the webpage and database design as initially plan if time permits, more deep learning modules and its examples are considered as part of the project.

Figure 1: Flowchart Design Implementation Plan

Application Layer

Domain Abstraction Layer

Programming Paradigm Layer

Programming Language Layer (Libraries)

main

input.txt

config.json

 preprocessor

-scale, feature, hyper tuning

-

model

regression

classification

cluster

rules

neural

helper

constant

sampling

nonsupervised

Table 1: Item Functionality

|  |  |
| --- | --- |
| **Process** | **Description** |
| Main | * Process per record to have logging capability. * Add the ArgsParse to cater for pointing of different input files, commented it out to minimize typing instead of making a multi-thread application. * If time permits process per record group of constant number to execute per group in thread |
| config.json | * global configuration json file |
| input.txt | * JSON format of each item with pipe delimeted: model|file| |preprocessor|model|graphs * Input file can have mime and non-mime type that uses the dataframe.dataset |
| Preprocessor | * to create a generic parser to solve multi-parameter by using JSON dict to dataframe to make it pluggable * use pipeline to make the code maintainable only in missing values and polynomial * handing of TSV, for text processing using nltk, natural language toolkit, after it will go to the corresponding regression and classification models * included feature selection and hyper tuning whenever applicable |
| Model | * the modelling technique use with the abstract method of train, predict, graph, also pre-process the preprocessor and global metrics handling |
| Scale | * include as a functionality in the preprocessor |
| Feature | * several feature selections are used as indicated in the appendix except for the red ink |
| Helper | * utility helper function |
| Metric | * for metrics purposes mapped according to regression, classification, deep learning and others in the config.json in the *model* module |
| Regression  (combined in regression\_  classification) | * regression modelling (test udemy input files first) in *regression\_classification* with output files: i.e. graph and excel * to apply some or all regression model listed: LinearRegression, MultiLinearRegression, PolynomialRegression, SVR, DecisionTreeRegressor, RandomForestRegressor, MultiLinearRegression, LogisticRegression. |
| Classification  (combined in regression\_  classification) | * selected classification techniques in *regression\_classification* with output files: i.e. graph and excel – (y\_true vs y\_pred in dataset and/or user input, confusion matrix, classification report by target classes) * to apply several classifier models listed: KNeighborsClassifier, SVC, GaussianNB, DecisionTreeClassifier, RandomForestClassifier |
| Cluster | * clustering technique in selection * to apply some or all cluster model listed: KMeans, AgglomerativeClustering, DBSCAN |
| Rules | * rules in selection * apply rules like apriori, eclat |
| Sampling | * applied the reinforcement learning of sampling for UCB (upper confidence bound), Thompson Samling |
| Natural Language Processing | * Porter Stemmer algorithm is used in the natural language processing, thereafter, any modelling technique is applicable |
| Neural network | * neural network * ANN, CNN, RNN |
| Nonsupervised | * SOM, RBM |

The batch processing maybe made limited into certain specific modelling techniques, feature selection. To include the hypertuning in the pre-processing of the modelling whenever applicable to the model samples. Option to use pydoc (\_\_DOC\_\_) to document the process. Testing is initially done to udemy input files to have a skeleton baseline, and extend some features to make it more robust on per need basis.

# Timetable for Completion (including key milestones)

For each milestones continues reading, online research for different theories and implementation are done whenever applicable. At times, verification is done if the implementation can be workable for the project. I’m doing part-time work or resting or doing other staffs, so I usually only do it 2-3 or few days in a week. Due to the time constraint with selected models only are tested.

|  |  |  |  |
| --- | --- | --- | --- |
| **Week** | **From** | **To** | **List** |
| Week 1 | 11 July 21 | 17 July 21 | Updated the version 2 the base version created on the prior dates from the version1 development done – created the static pipeline’, debugging |
| Week 2 | 18 July 21 | 24 July 21 | Proposal documentation, develop, testing, debugging, updated to have the dynamic parser with different implementation’, debugging |
| Week 3 | 25 July 21 | 31 July 21 | Add/Update document, develop, testing, debugging, updated the preprocessor module for the missing data to be dynamic execution’, debugging |
| Week 4 | 1 Aug 21 | 7 Aug 21 | Add/Update document, develop, testing, debugging, updated the preprocessor execution to have configuration for high ordinality values via GLMMEncoder(), scaling, added static feature selection and hypertuning, added Feature Type graph’, debugging |
| Week 5 | 8 Aug 21 | 14 Aug 21 | Add/Update document, develop, testing, debugging preprocessor: updated feature selection PCA, SelectKBest to have the scores in excel and savable graph of the scores, updated the hypertuning to be dynamic, combine the regression and classification’, debugging |
| Week 6 | 15 Aug 21 | 21 Aug 21 | Add/Update document develop, testing, debugging preprocessor: hypertuning of dynamic scores, spool in excel, reading of non-MIME datasets of fetch (real-world) and load (sample), use measure\_performance as a decorator’, debugging |
| Week 7 | 22 Aug 21 | 28 Aug 21 | Add/Update document, develop, testing, debugging updated model interface to have metrices, added regressor – LinearRegressor, updated main() to have classification types between Regressor and Classifier’, debugging |
| Week 8 | 29 Aug 21 | 4 Sep 21 | Add/Update document, develop, testing, debugging added regressor to have PolynomialFeatures, SVR, DecisionTreeRegressor, RandomForestRegressor, smoothing for the graph applied only for SVR, DecisionTreeRegressor, RandomForestRegressor, batch testing, separated metrics of regression and classification’, debugging |
| Week 9 | 5 Sep 21 | 11 Sep 21 | Add/Update document, develop, testing, debugging added classifier to have LogisticRegression, KNeighborsClassifier, plot for contour and scatter, plot report for confusion matrix, classification report, prediction error, roc\_auc’, debugging |
| Week 10 | 12 Sep 21 | 18 Sep 21 | Add/Update document, develop, testing, debugging added classifier to have SVC, GaussianNB, DecisionTreeClassifier, RandomForestClassifier, plot for kde with scatter’, debugging |
| Week 11 | 19 Sep 21 | 25 Sep 21 | Add/Update document, develop, testing, debugging added cluster: KMeans, placed in the title ‘scatter plot’, debugging |
| Week 12 | 26 Sep 21 | 2 Oct 21 | Add/Update document, develop, testing, debugging cluster: Agglomerative, debugging, enhance to point to different input files, ppt presentation, added plot\_dendograph() |
| Week 13 | 3 Oct 21 | 9 Oct 21 | Add/Update document, develop, testing, debugging cluster: MiniBatchKMeans, DBSCAN |
| Week 14 | 10 Oct 21 | 16 Oct 21 | - |
| Week 15 | 17 Oct 21 | 23 Oct 21 | Add/Update document, develop, testing, debugging rules: APRIORI, ECLAT |
| Week 16 | 24 Oct 21 | 30 Oct 21 | Add/Update document, develop, testing, debugging sampling:  UCB. Thompson Sampling |
| Week 17 | 1 Nov 21 | 7 Nov 21 | Add/Update document, develop, testing, debugging natural language processing: Porter Stemmer |
| Week Summer | 8 Nov 21 | 14 Nov 21 | Add/Update document, develop, testing, debugging neural network: ANN |
| Week Summer/  Week 18 | 22 Jan 22  /  27 Feb 22 | 11 Feb 22  /  5 Mar 22 | Add/Update document, develop, testing, debugging neural network: ANN neural network – update graph to have validate\_data,confusion matrix for discrete, plot3d for continuous number, added CNN (study and implement as I do)  Updated other files: helper, constant, graphics |
| Week 19 | 6 Mar 22 | 12 Mar 22 | Add/Update document, develop, testing, debugging neural network: loading with language encoding, confusion matrix, CNN for natural language processing, and loading to local directory, retest sample tests, change config.json to have base directory |
| Week 20 | 13 Mar 22 | 19 Mar 22 | Add/Update document, develop, testing, debugging neural network: to include y-test and y-prediction in XLSX tab if applicable, updated CNN for one sample based on filename if no y in the XLSX, study RNN – LSTM, added load\_files(), preprocess\_target() |
| Week 21 | 20 Mar 22 | 26 Mar 22 | Add/Update document, develop, testing, debugging neural network: to update RNN – LSTM: updated preprocessor.py: preprocess\_target(), added to\_dataframe(), neural.py: added plot\_target() and LSTM handling |
| Week 22 | 27 Mar 22 | 2 Apr 22 | Backup, Reset / format PC for 2 days as reset twice still getting high memory usage , Add/Update document, develop, testing, debugging regression\_classification.py as the ax.plot in subplot becomes unavailable so use seaborn instead in my PC by switch server; and debugging issues in regression (regression\_classification.py) for 2 days; review SOM, added nonsupervised.py for SOM handling today (1 day) |
| Week 23 | 3 Apr 22 | 9 Apr 22 | Add/Update document, develop, test for SOM to include summary\_stats, plot of confusion matrix, scatter |
| Week 24 | 10 Apr 22 | 16 Apr 22 | Study RBM, Auto-encoder, Add/Update document, develop test for RBM in the preprocessor, adding input\_file, added the initial development for RBM using Gibb Sampling in nonsupervised.py – added some functions convert(), evaluate(), rate\_data() |
| Week 25 | 17 Apr 22 | 23 Apr 22 | Add/Update document, develop, test for RBM – nosupervised.py - Added plot\_rbm(), updated evaluate() - add history of loss and accuracy score to xls, updated rate\_data() to make the scoring setup generic; model.py - updated plot\_cm() to check for if cm\_df exists to prevent errors |
| Week 26 | 24 Apr 22 | 30 Apr 22 | Add/Update document, develop, test further for RNN – updated model.py, preprocessor.py - load\_auto\_generated\_by(), constant.py, neural.py - multistep\_forecast() |
| Week 27 | 1 May 22 | 7 May 22 | Add/Update document, develop, test further for RNN – updated model.py, preprocessor.py, neural.py - to handle to RNN, GRU, LSTM; created powerpoint, pipenv for project virtual environment |
| Week 28 | 8 May 22 | 14 May 22 | * Checking on video making, having some software issues, i.e. speaker, microphone in my laptop; requested computer shop to revert to windows 10 |
| Week 29 | 15 May 22 | 21 May 22 | Updated the filename to ANN if the built model has Dense only, updated multistep\_forecast() for ANN |
| Week 30 | 22 May 22 | 29 May 22 | Update plot\_cm() to put label text for specific example, Updated plot\_confusion\_example() to add the index in the title, updated powerpoint, updated samples to add xlsx, updated document, added ads\_selected\_count() |
| Week 31 | 30 May 22 | 5 June 22 | Updated project document – abstract, acknowledgment, result analysis, com; input file of sample; created the reflective report document |
| Week 32 | 6 June 22 | 12 June 22 | Updated the reflective report document, cleanup unused codes |

# Design Level

## High Level

Figure 1: High Level Design

Input Files

Preprocessor

Modelling Processing

Output Files

1. Input files consist of config.json, input.txt and files to be processed. File config.json includes the global setup of input and output directory, attributes like *test\_size* to indicate the data split of train and test sets when per record account is not provided, flags like *save* to spool the excel files in local drive, *graph* to indicate when the graphics are displayed and file\_stats the file of the statistical metrics. The batch input files to be processed are indicated in the input.txt having the modifier, file, dictionary format of preprocessor parameter, model parameter and graph parameters delimited by pipe. The dictionary format allows flexibility in the design for the data processing of the estimators and the attributes.
2. Preprocessor handles the missing value for both numeric and categorical values in any ordinality. Also, feature scaling, feature selection and classification hyper-tuning maybe done at this level.
3. Classification processing are the handling of various classifiers for different modelling techniques, which includes its training, testing, graphic visualization, statistical measures.
4. Output files are the timestamp graphics output, excel files of the y, y predicted values, confusion matrix and statistical results. The statistical results can serve as a record per run or as a comparison for other classifications executed on the same file. The timestamp in the file provides traceback capability and as a reference to verify the data visualization.

## Detailed Design

Figure 2: Flowchart Detailed Design Implementation Plan

Application Layer

Domain Abstraction Layer

Programming Paradigm Layer

Programming Language Layer (Libraries)

main

input.txt

config.json

 preprocessor

model

regression

classification

cluster

rules

neural

helper

log

constant

sampling

nonsupervised

# Process Functionality

## Pre-processor

Input

Missing Data

Train Test Split

Data

Scale

Graph /

Excel

Select Features

Scale

Hyper

tuning

Scale

Figure: Pre-processor Detailed Design

* *Missing Data* - It handles the missing value of the numeric and categorical variables
* *Train Test Split* - Split the data from training and test sets when needed
* *Data Scale* - Scale the features *(X)* according and perform data transformation *(X, y)*
* *Select Features* - selected the features based on the record configuration
* *Hyper-tuning* – get the hyper parameter of the model
* Graph / Excel - Graph the count of the data features, selected feature and spool to file selected features with the scores, best estimator with the configurable score and the configuration matrix

The batch input file contents are defined in dictionary that can be loaded as a data frame. The pre-processor parameter can be segregated by values or method. The method is also in dictionary format with key ID, predefined values of *method* and *params* to distinguish its method and parameters like a function pointer. The graphic files are placed in folder output director, which is configured in the config.json, the folder of the input file name without the extension. An AssertError is thrown whenever the function name or the attribute is wrong as it uses the underlying software to evaluate. The error is spooled into the log file with date stamp for quick reference.

config.json

The config.json has *max\_cardinality* to indicate the setup of high cardinal values to know which encoder to use between OneHotEncoder and GLMMEncoder for low and high cardinal values accordingly. If the *max\_cardinal=10*, which indicates the unique cardinal values of column < 10 use OneHotEncoder, else GLMMEncoder.

1. Missing Data, Train Test Split, Scale

|  |  |
| --- | --- |
| ***config.json*** | max\_cardinal=10 |

Batch Input File (input.txt):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table A.1: Preprocessor Execution | | | | |
| ***Modifier*** | | SVC | | |
| ***File*** | | Data.csv | | |
| ***Pre-processor Paramater*** | | {"test\_size":0.2,"scaler":{"method":"StandardScaler","params":{}},"imputer":{"method":"SimpleImputer","params":{"strategy":"mean"}}} | | |
| ***Modifier Paramater*** | |  | | |
| ***Graph Parameter*** | |  | | |
| Input File | Missing Data | | Train Test Split | Data Scale |
| Table  Description automatically generated | X: [[1.00000000e+00 0.00000000e+00 0.00000000e+00 4.40000000e+01 7.20000000e+04]  [0.00000000e+00 0.00000000e+00 1.00000000e+00 2.70000000e+01 4.80000000e+04]  [0.00000000e+00 1.00000000e+00 0.00000000e+00 3.00000000e+01 5.40000000e+04]  [0.00000000e+00 0.00000000e+00 1.00000000e+00 3.80000000e+01 6.10000000e+04]  [0.00000000e+00 1.00000000e+00 0.00000000e+00 4.00000000e+01 6.37777778e+04]  [1.00000000e+00 0.00000000e+00 0.00000000e+00 3.50000000e+01 5.80000000e+04]  [0.00000000e+00 0.00000000e+00 1.00000000e+00 3.87777778e+01 5.20000000e+04]  [1.00000000e+00 0.00000000e+00 0.00000000e+00 4.80000000e+01 7.90000000e+04]  [0.00000000e+00 1.00000000e+00 0.00000000e+00 5.00000000e+01 8.30000000e+04]  [1.00000000e+00 0.00000000e+00 0.00000000e+00 3.70000000e+01 6.70000000e+04]] | | X\_train: [[1.00000000e+00 0.00000000e+00 0.00000000e+00 4.40000000e+01 7.20000000e+04]  [0.00000000e+00 0.00000000e+00 1.00000000e+00 2.70000000e+01 4.80000000e+04]  [0.00000000e+00 1.00000000e+00 0.00000000e+00 3.00000000e+01 5.40000000e+04]  [0.00000000e+00 0.00000000e+00 1.00000000e+00 3.80000000e+01 6.10000000e+04]  [0.00000000e+00 1.00000000e+00 0.00000000e+00 4.00000000e+01 6.37777778e+04]  [1.00000000e+00 0.00000000e+00 0.00000000e+00 3.50000000e+01 5.80000000e+04]  [0.00000000e+00 0.00000000e+00 1.00000000e+00 3.87777778e+01 5.20000000e+04]  [1.00000000e+00 0.00000000e+00 0.00000000e+00 4.80000000e+01 7.90000000e+04]]  X\_test: [[0.0e+00 1.0e+00 0.0e+00 5.0e+01 8.3e+04]  [1.0e+00 0.0e+00 0.0e+00 3.7e+01 6.7e+04]] | X\_train: [[ 1.29099445 -0.57735027 -0.77459667 0.9919531 1.13089809]  [-0.77459667 -0.57735027 1.29099445 -1.64177922 -1.33030078]  [-0.77459667 1.73205081 -0.77459667 -1.17700292 -0.71500106]  [-0.77459667 -0.57735027 1.29099445 0.06240052 0.00284861]  [-0.77459667 1.73205081 -0.77459667 0.37225138 0.28770959]  [ 1.29099445 -0.57735027 -0.77459667 -0.40237577 -0.30480125]  [-0.77459667 -0.57735027 1.29099445 0.18289808 -0.92010097]  [ 1.29099445 -0.57735027 -0.77459667 1.61165483 1.84874777]]  X\_test: [[-0.77459667 1.73205081 -0.77459667 1.92150569 2.25894758]  [ 1.29099445 -0.57735027 -0.77459667 -0.09252491 0.61814833]]  y: [0 1 0 0 1 1 0 1 0 1] |
| Graph | | | Info | |
| Chart, bar chart  Description automatically generated | | | <class 'pandas.core.frame.DataFrame'>  RangeIndex: 10 entries, 0 to 9  Data columns (total 4 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Country 10 non-null object  1 Age 9 non-null float64  2 Salary 9 non-null float64  3 Purchased 10 non-null object  dtypes: float64(2), object(2)  memory usage: 448.0+ bytes | |

1. Feature Selection

The feature selection is categorized into matrix decomposition (*sklearn.decomposition* (scikit-learn, n.d.-k)*)*, discriminant analysis (*sklearn.discriminant\_analysis* (scikit-learn, n.d.-i)) and normal feature selection (*sklearn.feature\_selection* (scikit-learn, n.d.-h)). For reference, the [appendix](#_Feature_Selection) provides a quick summary of the usage, while the [scikit learn](#_Feature_Selection) website is for more detailed reference. The tool spools the result into the excel and graph of the top features and scores. Matrix decomposition records the score variance score of each of the principal components with the higher score holding the most information and graph the relationship of its features in the heatmap format. The excel has top features of the method and its score.

|  |  |
| --- | --- |
| ***config.json*** | max\_cardinal=2 |

Batch Input File (input.txt):

|  |  |
| --- | --- |
| ***Modifier*** | SVC |
| ***File*** | Data.csv |
| ***Pre-processor Paramater*** | {"test\_size":0.2,"scaler":{"method":"StandardScaler","params":{}},"feature":{"method":"KernelPCA","params":{"n\_components":3}},"imputer":{"method":"SimpleImputer","params":{"strategy":"mean"}}} |
| ***Modifier Paramater*** | {"C": [0.25, 0.5, 0.75, 1], "kernel": ["rbf","linear"],"gamma": [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]} |
| ***Graph Parameter*** |  |

1. Matrix Decomposition: sklearn.decomposition (scikit-learn, n.d.-k)

The common attribute in matrix decomposition is *n\_components* to reduce dataset dimensionality.

PCA – Principal Component Analysis caters for linear dimensionality reduction to project the data into lower space. The multidimensional datasets correlate the independent variables via Singular Value Decomposition (SVD) using the variance score on which variables holds the most information. ((datacamp, n.d.), (scikit-learn, n.d.-m))

Old testing data

|  |  |
| --- | --- |
| Table A.2a: Preprocessor: PCA Feature Execution | |
| Excel | Graph |
| Table  Description automatically generated | Chart, bar chart  Description automatically generated |

|  |  |
| --- | --- |
| Table A.2a: Preprocessor: PCA Feature Execution | |
| Excel | Graph |
| Table  Description automatically generated | A picture containing chart  Description automatically generated |

KernelPCA – Non-linearity dimension thru using the kernel. (scikit-learn, n.d.-j)

The method of the *feature* is change to KernelPCA.

|  |  |
| --- | --- |
| Table A.2a: Preprocessor: KernelPCA Feature Execution | |
| **Excel** | **Graph** |
| Graphical user interface, text, table  Description automatically generated | Chart, bar chart  Description automatically generated |

1. sklearn.feature\_selection (scikit-learn, n.d.-h)

GenericUnivariateSelect - This feature selection is the generic module that handles any of the feature selection SelectPercentile, SelectKBest, SelectFpr, SelectFdr, SelectFwe.

|  |  |  |  |
| --- | --- | --- | --- |
| Table A.2b1: Preprocessor: GenericUnivariate Feature Execution | | | |
| ***Modifier*** | SVC | **Excel** | **Graph** |
| ***File*** | Data.csv | Table  Description automatically generated | Chart, bar chart  Description automatically generated |
| ***Pre-processor Paramater*** | {"test\_size":0.2,"scaler":{"method":"StandardScaler","params":{}},"feature":{"method":"GenericUnivariateSelect","params":{"score\_func":"f\_classif", "mode":"k\_best", "param":"all"}},"imputer":{"method":"SimpleImputer","params":{"strategy":"mean"}}} |

SelectKBest – select the features according to the k highest scores (scikit-learn, n.d.-n)

|  |  |  |  |
| --- | --- | --- | --- |
| Table A.2b2: Preprocessor: SelectKBest Feature Execution | | | |
| ***Modifier*** | SVC | **Excel** | **Graph** |
| ***File*** | Data.csv | Table  Description automatically generated | Chart, bar chart  Description automatically generated |
| ***Pre-processor Paramater*** | {"test\_size":0.2,"scaler":{"method":"StandardScaler","params":{}},"feature":{"method":"SelectKBest","params":{"k":3}},"imputer":{"method":"SimpleImputer","params":{"strategy":"mean"}}} |

1. Hyper tuning

Hyper tuning of classifiers and/or parameters can be done in several ways using the hyperparameter optimizers in model selection (scikit-learn, n.d.-o), boosting algorithms like XGBoost using XGBClassifier. Currently, the tool focuses only in GridSearchCV.

GridSearchCV technique performs exhaustive search over specified parameter (hyper parameter) values for an estimator with the output complying to the bias-variance trade-off via cross-validation on the train and test sets. (Kumar, 2020) The tool would spool the best score, parameter and estimator used, which would be used for the actual classification of the modelling technique. It also provides a binary confusion matrix to derive the best score for the hyper parameter, which is configurable in the hypertune() by the setup of any statistical measures like *accuracy, precision, recall, f1-score, mcc* with the default sets to *f1-score*

Batch Input File (input.txt):

|  |  |  |
| --- | --- | --- |
| Table A.3: Preprocessor: Hypertuning Execution | | |
| ***Modifier*** | RandomForestClassifier | ***Excel*** |
| ***File*** | load\_breast\_cancer | Graphical user interface, table  Description automatically generated |
| ***Pre-processor Paramater*** | {"test\_size":0.2,"scaler":{"method":"StandardScaler","params":{}},"feature":{"method":"GenericUnivariateSelect","params":{"score\_func":"f\_classif", "mode":"k\_best", "param":"all"}},"imputer":{"method":"SimpleImputer","params":{"strategy":"mean"}}} |
| ***Modifier Paramater*** | {"max\_depth":[2, 3, 4], "max\_features":[2, 3, 4, 5, 6]} |
| ***Graph Parameter*** |  |

## Modelling Technique

Preprocessor

(Text (TSV) – Natural Language)

Modelling Processing

Graph / Excel

Regression & Classification

Figure: Model Detailed Design

Clustering

Rules

Sample

Neural

Network

Non-supervised

Modelling techniques can be categorized into regression, classification, clustering, rules, and neural network. Some modelling techniques can intersect with one another like there are some classification techniques that can have both regression and classification. (Refer to Appendix C for details) Due to the vastness of the models only the specified classifiers are done in a limited time frame and parameter combination. The best parameter(s) and other variables used in the preprocessor are passed to the model. When there’s no parameter given, it uses the default parameter bypassing the hyper-tuning functionality. Then, the model implements each modelling techniques accordingly. The statistical measures are placed in the model in a matrix format (dataframe) to spool the summary result in Excel.

### Regressor

This adheres for various regression techniques. A flag *classification\_type = ‘regressor’* would identify the difference between classification and regression.

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_regression.txt |

1. LinearRegressor – ordinal least of squares linear regression that minimize the residual sum of squares between the targets and the predicted.(scikit-learn, n.d.-t)

|  |  |  |
| --- | --- | --- |
| Table B.1a: LinearRegressor Execution: Graph Ouput | | |
| ***Modifier*** | LinearRegression | Chart, line chart  Description automatically generated |
| ***File*** | Salary\_Data.csv |
| ***Pre-processor Paramater*** | {"test\_size":0.333,"random\_state":0} |
| ***Modifier Paramater*** |  |
| ***Graph Parameter*** | {"title":"Salary vs Experience","xlabel":"Years of Experience","ylabel":"Salary"}  *Note: If no title is given the default one is set to the classifier name* |

1. PolynomialRegressor – combination of PolynomialFeatures and LinearRegressor, though other linear regressor modelling technique like Ridge for nonlinear dimensionality can be used for linear, nonlinear, which is used to generate polynomial and interaction features. (scikit-learn, n.d.-u)

|  |  |  |
| --- | --- | --- |
| Table B.1b: PolynomialRegressor Execution: Graph Ouput | | |
| ***Modifier*** | PolynomialFeatures | Chart, line chart  Description automatically generated |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** |  |
| ***Modifier Paramater*** | {"x":"Level"} |
| ***Graph Parameter*** | {"degree":3} |

1. SVR – Epsilon-Support Vector Regression as it uses epsilon and C library (libsvm) to be the based module having more than quadratic functions made it hard to scale, so it uses Nystroem transformer.(scikit-learn, n.d.-w)

|  |  |  |
| --- | --- | --- |
| Table B.1c: SVR Execution: Graph Ouput | | |
| ***Modifier*** | SVR | Chart, line chart  Description automatically generated |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"x":"Level","predict\_test":5.5,"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** | {"kernel":["rbf"]} |
| ***Graph Parameter*** |  |

1. DecisionTreeRegressor – Decision Tree Regresor

|  |  |  |
| --- | --- | --- |
| Table B.1d: DecisionTreeRegressor Execution: Graph Ouput | | |
| ***Modifier*** | DecisionTreeRegressor | Chart, line chart  Description automatically generated |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"x":"Level"} |
| ***Modifier Paramater*** | {"random\_state":0} |
| ***Graph Parameter*** |  |

1. RandomForestRegressor- is a meta estimator that classifies based on various sub-samples and uses averaging to improve accuracy and control overfitting using *max\_samples* and *boostrap=True*. (scikit-learn, n.d.-v)

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.1e: RandomForestRegressor Execution: Graph Ouput | | |  |
| ***Modifier*** | RandomForestRegressor | Chart, line chart  Description automatically generated |  |
| ***File*** | Position\_Salaries.csv |  |  |
| ***Pre-processor Paramater*** | {"x":"Level"} |  |  |
| ***Modifier Paramater*** | {"n\_estimators":10,"random\_state":0} |  |  |
| ***Graph Parameter*** |  |  |  |

Batch processing of the regressor for the various regression modelling given the regression metrices.

|  |
| --- |
| Batch Executon: Excel Output |
|  |

|  |  |
| --- | --- |
| Table 1: Linear Regression Batch Run | |
| Result Analysis | is used to measure for the linear regression as 1 indicates it fits to a line, any value closer to 1 indicates the close correlation to a line, with 0 means no correlation. So, from the file Position\_Salaries.csv, the best fits to line is (1) DecisionTreeRegressor, (2) Polynomial Linear Regression, (3) RandomForestRegressor. |

### Classifier

This adheres for various classification techniques. A flag *classification\_type = ‘classifier’* would identify the difference between classification and regression.

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_classifier.txt |

1. LogisticRegression – in multiclass case, the training algorithm uses the one-over-rest (OvR) scheme when *multiclass=’ovr’* and uses the cross-entropy when *multiclass=’multinominal’.* (scikit-learn, n.d.-b)

|  |  |  |
| --- | --- | --- |
| Table B.2a: LogisticRegression Execution: Excel Output | | |
| ***Modifier*** | LogisticRegression |  |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"test\_size": 0.25,"random\_state":0,"predict\_test":[30,87000],"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** | {"random\_state":0} |
| ***Graph Parameter*** | {"smooth":[10,1000]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.2a: LogisticRegression Execution: Graph / Excel Output | | | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated |
| Chart, treemap chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Chart, bar chart  Description automatically generated | Chart, line chart  Description automatically generated |

1. KNeighborsClassifier – implements nearest neighbours algorithm

|  |  |  |
| --- | --- | --- |
| Table B.2b: KNeighnorsClassifier Execution: Excel Output | | |
| ***Modifier*** | KNeighborsClassifier |  |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"test\_size": 0.25,"random\_state":0,"predict\_test":[30,87000],"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** | {"n\_neighbors":5,"metric":"minkowski","p":2} |
| ***Graph Parameter*** | {"smooth":[10,1000]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.2b: KNeighnorsClassifier Execution: Graph / Excel Output | | | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated |
| Chart, treemap chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Chart, bar chart  Description automatically generated | Chart, scatter chart  Description automatically generated |

1. SVC – Support Vector Classification

Test1 *– kernel=linear*

|  |  |  |
| --- | --- | --- |
| Table B.2c1: SVC Execution: Excel Output | | |
| ***Modifier*** | SVC |  |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"test\_size": 0.25,"random\_state":0,"predict\_test":[30,87000],"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** | {"kernel":"linear","random\_state":0} |
| ***Graph Parameter*** | {"smooth":[10,1000]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.2c1: SVC Execution: Graph / Excel Output | | | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated |
| Chart, treemap chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Chart, bar chart  Description automatically generated |  |

Test2 – *kernel=rbf*

|  |  |  |
| --- | --- | --- |
| Table B.2c2: SVC Execution: Excel Output | | |
| ***Modifier*** | SVC |  |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"test\_size": 0.25,"random\_state":0,"predict\_test":[30,87000],"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** | {"kernel":"rbf","random\_state":0} |
| ***Graph Parameter*** | {"smooth":[10,1000]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.2c2: SVC Execution: : Graph / Excel Output | | | |
| Chart, scatter chart  Description automatically generated | Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated |
| Chart, treemap chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Chart, bar chart  Description automatically generated |  |

1. GaussianNB – Gaussian Naïve Bayes

|  |  |  |
| --- | --- | --- |
| Table B.2d: GaussianNB Execution: Excel Output | | |
| ***Modifier*** | GaussianNB |  |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"test\_size": 0.25,"random\_state":0,"predict\_test":[30,87000],"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** |  |
| ***Graph Parameter*** | {"smooth":[10,1000]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.2d: GaussianNB Execution: Graph / Excel Output | | | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated | Chart, scatter chart  Description automatically generated |
| Chart, treemap chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Chart, bar chart  Description automatically generated | Chart, line chart  Description automatically generated |

1. DecisionTreeClassifier

|  |  |  |
| --- | --- | --- |
| Table B.2e: DecisionTreeClassifier Execution: Excel Output | | |
| ***Modifier*** | DecisionTreeClassifier |  |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"test\_size": 0.25,"random\_state":0,"predict\_test":[30,87000],"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** | {"criterion":"entropy","random\_state":0} |
| ***Graph Parameter*** | {"smooth":[10,1000]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.2e: DecisionTreeClassifier Execution: Graph / Excel Output | | | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated |
| Chart, treemap chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Chart, bar chart  Description automatically generated | Chart, line chart  Description automatically generated |

1. RandomForestClassifier – it fits the estimator on various sub-samples and uses averaging to improve the predictive accuracy and control over-fitting.(scikit-learn, n.d.-c)

|  |  |  |
| --- | --- | --- |
| Table B.2f: RandomForestClassifier Execution: Excel Output | | |
| ***Modifier*** | RandomForestClassifier |  |
| ***File*** | Position\_Salaries.csv |
| ***Pre-processor Paramater*** | {"test\_size": 0.25,"random\_state":0,"predict\_test":[30,87000],"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** | {"criterion":"entropy","n\_estimators":10,"random\_state":0} |
| ***Graph Parameter*** | {"smooth":[10,1000]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.2f: RandomForestClassifier Execution: Graph / Excel Output | | | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated | Chart  Description automatically generated |
| Chart, treemap chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Chart, bar chart  Description automatically generated | Chart, line chart  Description automatically generated |

Batch processing of the regressor for the various classifier modelling given the classification metrices. From the sample metrics result of the batch run, SVC with *kernel=rbf* and KNeighbors performs better than the other classifiers.

|  |  |
| --- | --- |
| Batch RunExcel | Snapshot |
|  |  |

|  |  |
| --- | --- |
| Table 2: Classifier Batch Run | |
| Result Analysis | Using the Social\_Networks\_ads..csv `based on one’s ages, estimated salary who are likely to purchase social network ads having the correct predictions between the y\_true value and the forecasted value, using the following metrices: accuracy\_score, precision\_score, recall\_score,f1\_score, matthews\_corrcoef, the best classifiers are SVC with rbf, which is in the 3rd index compared to SVC with linear in the 2nd index and others, equally the KNeighborsClassifier top the list |

### Cluster

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_cluster.txt |

1. KMeans – one of the fastest clustering algorithms, but it falls in local minima. (scikit-learn, n.d.-f)

* No y, no parameter in model – using elbow method to find the clusters in the range of *(2,11)*.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table B.3a: KMeans Execution: Excel Output | | | | | |
| ***Modifier*** | KMeans | | |  | |
| ***File*** | Mall\_Customers.csv | | |
| ***Pre-processor Paramater*** | {"has\_y": false,"x":["Annual Income","Spending Score"]} | | |
| ***Modifier Paramater*** |  | | |
| ***Graph Parameter*** |  | | |
| Table B.3a: KMeans Execution: Graph / Excel Output | | | | | |
| Chart, line chart  Description automatically generated | | Chart, scatter chart  Description automatically generated |  | | Chart, bar chart  Description automatically generated |

* With y, with parameter in the model – using GridSearchCV to find the optimal parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.3a2: KMeans Execution: Graph / Excel Output | | | |
| ***Modifier*** | KMeans |  | Chart, bar chart  Description automatically generated |
| ***File*** | load\_breast\_cancer |
| ***Pre-processor Paramater*** | {"test\_size":0.2,"scaler":{"method":"StandardScaler","params":{}},  "feature":{"method":"PCA","params":{"n\_components":5}},  "imputer":{"method":"SimpleImputer","params":{"strategy":"mean"}}} |
| ***Modifier Paramater*** | {"n\_clusters":[2,3,4,5,6,7,8,9,10]} |
| ***Graph Parameter*** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table B.3a2: KMeans Execution: Graph / Excel Output | | | | | |
| Scatter chart  Description automatically generated | | Scatter chart  Description automatically generated | | Chart, scatter chart  Description automatically generated | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | | Chart  Description automatically generated | | Chart  Description automatically generated |

1. AgglomerativeClustering – do recursively pairing of clusters of the dataset using linkage distance. (scikit-learn, n.d.-d)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table B.3b: AgglomerativeClustering Execution: Graph / Excel Output | | | | | |
| ***Modifier*** | AgglomerativeClustering | | |  | |
| ***File*** | Mall\_Customers.csv | | |
| ***Pre-processor Paramater*** | {"has\_y": false,"x":["Annual Income","Spending Score"]} | | |
| ***Modifier Paramater*** |  | | |
| ***Graph Parameter*** |  | | |
| Chart, line chart  Description automatically generated | | Chart, scatter chart  Description automatically generated | Chart, scatter chart  Description automatically generated | | Chart, line chart  Description automatically generated |

1. MiniBatchKMeans – Mini batch KMeans clustering.

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.3c: MiniBatchKMeans Execution: Graph / Excel Output | | | |
| ***Modifier*** | MiniBatchKMeans |  | Chart, bar chart  Description automatically generated |
| ***File*** | load\_breast\_cancer |
| ***Pre-processor Paramater*** | {"test\_size":0.2,"scaler":{"method":"StandardScaler"  ,"params":{}},  "feature":{"method":"PCA","params":{"n\_components":5}},  "imputer":{"method":"SimpleImputer","params":{"strategy":"mean"}}} |
| ***Modifier Paramater*** | {"n\_clusters":[2,3,4,5,6,7,8,9,10]} |
| ***Graph Parameter*** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table B.3c: MiniBatchKMeans Execution: : Graph / Excel Output | | | | | |
| Scatter chart  Description automatically generated | | Scatter chart  Description automatically generated | | Chart, scatter chart  Description automatically generated | |
| Chart, scatter chart  Description automatically generated | Chart  Description automatically generated | | Chart  Description automatically generated | | Chart  Description automatically generated |

1. DBSCAN – Density Based Spatial Clustering of Application with Noise. (scikit-learn, n.d.-e) It finds core samples of high density and expands clusters from them, which is useful for cluster with similar density.(scikit-learn, n.d.-e) It can increase memory complexity as its bulk computations using the nearest neighbours depending on the algorithm. (scikit-learn, n.d.-e)

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.3d: DBSCAN Execution: Graph / Excel Output | | | |
| ***Modifier*** | DBSCAN |  |  |
| ***File*** | load\_breast\_cancer |
| ***Pre-processor Paramater*** | {"test\_size":0.2,"scaler":{"method":"StandardScaler","params":{}},  "feature":{"method":"PCA","params":{"n\_components":5}},  "imputer":{"method":"SimpleImputer","params":{"strategy":"mean"}}} |
| ***Modifier Paramater*** | {"eps":[0.1,0.2,0.3,0.4,0.5,0.6],"min\_samples":[2,3,4,5,6,7,8,9]} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.3d: DBSCAN Execution: Graph / Excel Output | | | |
|  |  |  |  |

|  |  |
| --- | --- |
| Batch RunExcel | Snapshot |
|  |  |

|  |  |
| --- | --- |
| Table 3: Cluster Batch Run | |
| Result Analysis | The Mall\_Customers.csv dataset is the customer spending habit in the mall based on customer, sex, age, income, spending score using 2 clustering methods: KMeans, AgglomerativeClustering using calinski\_harabasz\_score, davies\_bouldin\_score, silhouette\_score that KMeans perform better according to the metrices used as davies\_bouldin\_score the smaller the better, silhouette\_score the higher the better with the similar uniformity in the optimal cluster used.  load\_breast\_cancer is a scikit-learn dataset from  UCI ML Breast Cancer Wisconsin (Diagnostic) that checks if the breast cancer is 'malignant', 'benign' using DBSCAN, KMeans, MiniBatch using various metrices like calinski\_harabasz\_score, davies\_bouldin\_score, silhouette\_score, completeness\_score, homogeneity\_score, v\_measure\_score, KMeans wins as for the overall metrices used. |

### Rules

* Associate rule mining is a data mining technique based in the frequency of patterns and associations to predict the next important item in the dataset according to the customer purchases. (Dannina, 2020)

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_rules.txt |

1. Apriori – text-based processing that based the scores on lift, confidence, support by using association rule mining to build correlations for a series of frequent items in the dataset. (Dannina, 2020)

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.4a: APRIORI Execution: Graph Output | | |  |
| ***Modifier*** | Apriori |  |  |
| ***File*** | Market\_Basket\_Optimisation.csv |
| ***Pre-processor Paramater*** |  |
| ***Modifier Paramater*** | {"min\_support":0.003,"min\_confidence":0.2,"min\_lift":3,"min\_length":2} |
| ***Graph Parameter*** |  |

1. Eclat – Equivalence Class Clustering and bottom-up Lattice Traversal that is more efficient and scalable than apriori, which uses horizontal approach of Breadth-First search ­, while this algorithm uses vertical approach of Depth-First Search text-based processing to have the support score.(Gupta, 2019)

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.4b: ECLAT Execution: Graph Output | | |  |
| ***Modifier*** | Eclat |  |  |
| ***File*** | Market\_Basket\_Optimisation.csv |
| ***Pre-processor Paramater*** |  |
| ***Modifier Paramater*** |  |
| ***Graph Parameter*** |  |

There’s no global summary statistics as the rules are associative by the given items.

|  |  |
| --- | --- |
| **Rules Batch Run** | |
| **Apriori** | **Eclat** |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Table 4: Rules Batch Run | |
| Result Analysis | Market\_Basket\_Optimisation.csv is the grocery shopping cart of the customer when buying products. As the dataset is applied to association rules, hence if a customer bought a certain product , the other product would be bought is likely to be a complimentary or supporting product of that certain product . Two association rules: Apriori, Eclat using the measuring criterion of support, which is the proportion of those products existing in the other transactions too. Comparing the two rules using support value, Eclat wins over Apriori with higher value having the top3 product combination of (1) herb & pepper+ground beef, (2) whole wheat pasta+olive oil, (3) pasta+escalope, which can be used for cooking ground beef and pasta. |

### Sampling

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_sample.txt |

### 

* 1. UCB – Upper confidence bound is using the Multi-armed Bandit problem to solve the maximum UCB for each selection is the accumulative value of the average rewards , and the confidence interval . (Eremenko & de Ponteves, n.d.-c)

(Eremenko & de Ponteves, n.d.-c)

(Eremenko & de Ponteves, n.d.-c)

(Eremenko & de Ponteves, n.d.-c)

Where:

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.5a: UCB Execution: Graph Output | | |  |
| ***Modifier*** | UCB |  |  |
| ***File*** | Ads\_CTR\_Optimisation.csv |  |  |
| ***Pre-processor Paramater*** |  |  |  |
| ***Modifier Paramater*** |  |  |  |
| ***Graph Parameter*** | {"xlabel":"Ads"} |  |  |

* 1. Thompson Sampling – the algorithm would compute for the maximum beta variate based on the number of rewards using Bayesian algorithm by selecting the highest . (Eremenko & de Ponteves, n.d.-b)

(Eremenko & de Ponteves, n.d.-b)

Where:

– the number of times the item got reward 1 until round n.

– the number of times the item got reward 0 until round n.

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.5a: Thompson Sampling Execution: Graph Output | | |  |
| ***Modifier*** | ThompsonSampling |  |  |
| ***File*** | Ads\_CTR\_Optimisation.csv |
| ***Pre-processor Paramater*** |  |
| ***Modifier Paramater*** |  |
| ***Graph Parameter*** | {"xlabel":"Ads"} |

In the reinforcement learning of sampling, the points are based on rewards, so the record metrices are used for each model.

|  |  |
| --- | --- |
| Table 5: Sample Batch Run | |
| Result Analysis | Ads\_CTR\_Optimisation dataset is using 10 Ads () with 10,000 records by reinforcement learning of UCB and Thompson Sampling, which ad transactions are likely to be clicked by the customer, ad4 () wins for both the sampling method. |

### Natural Language Processing

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_text.txt |

### 

1. Porter Stemmer - Stemming is process of reducing the words to its word stem that appends to suffixes and prefixes or to the root words called lemma. (Singh, n.d.) It’s important for it provides a simple approach to conflation that works for a range of languages as well as spurred interest like research. (Singh, n.d.) The Porter Stemmer in the batch machine learning parses a TSV file that uses a bag of words approach to count the frequency of the words used in sequence based on the word stopper defined like ‘English’ attaching to the header of prefix and suffix’. After a selected modelling technique is used to classify the model.

Figure: Porter Stemmer Algorithm (Eremenko & de Ponteves, n.d.-a)

Seq2Seq



Bag of Words

=>

Porter Stemmer Algorithm

Figure: Porter Stemmer Implementation

TSV Input Files

Preprocessor

Modelling Processing

Output Files

|  |  |  |
| --- | --- | --- |
| Table B.6a: NLP Execution | | |
| ***Modifier*** | GaussianNB |  |
| ***File*** | Restaurant\_Reviews.tsv |
| ***Pre-processor Paramater*** | {"test\_size": 0.2,"random\_state":0,"stop\_words":"english","text":  {"method":"CountVectorizer","params":{"max\_features":1500}}} |
| ***Modifier Paramater*** |  |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.6a: NLP: : Graph / Excel Output | | | |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| Table B.6b: NLP Execution | | |
| ***Modifier*** | RandomForestClassifier |  |
| ***File*** | Restaurant\_Reviews.tsv |
| ***Pre-processor Paramater*** | {"test\_size": 0.2,"random\_state":0,"stop\_words":"english","text":  {"method":"CountVectorizer","params":{"max\_features":1500}}} |
| ***Modifier Paramater*** |  |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.6b: NLP: : Graph / Excel Output | | | |
|  |  |  |  |

|  |  |
| --- | --- |
| Batch Run Excel | Snapshot |
|  |  |

|  |  |
| --- | --- |
| Table 6: Natural Language Processing Batch Run | |
| Result Analysis | Restaurant\_Reviews.tsv is a dataset of a restaurant review, whether it’s like or not. that initially preprocessed using natural language before doing deep learning using scikit-learn models: GaussianNB, RandomForestClassifier using the classifier metrices () with RandomForestClassifier performing better than GaussianNB. |

### Neural Network

* It simulates the function of a network like a brain composed of dendrite, axons, synapse having input layer, hidden or intermediate layer, output layer based on the activation functions, which can be different from each layer.

*f(x)*

Input Layer

Hidden Layer

Output Layer

Figure: Neural Network Architecture

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_neural\_network.txt |

### 

* 1. ANN – Artificial Neural Network. It involves many processors operating in parallel and arranged in layers, with the first layer as input, succeeding optional layer as hidden, last layer as output that is adaptive to modify themselves as they train in subsequent way to provide information of the real world. (Burns & Burke, n.d.)

Test 1 – using excel file

|  |  |
| --- | --- |
| ***Modifier*** | Ann |
| ***File*** | Churn\_Modelling.csv |
| ***Pre-processor Paramater*** | {"test\_size":0.2,"random\_state":0,"scaler":{"method":"StandardScaler","params":{}}} |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"binary\_crossentropy","metrics": ["accuracy"],  "add1":{"method":"Dense","params":{"units":6,"activation":"relu"}},  "add2":{"method":"Dense","params":{"units":6,"activation":"relu"}},  "add3":{"method":"Dense","params":{"units":1,"activation":"sigmoid"}},  "fit":{"method":"fit","params":{"batch\_size":32,"epochs":100}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7a1: Artificial Neural Network Execution: Graph / Excel Output (added validate\_data) | | | |
| Chart, bar chart  Description automatically generated | Chart, line chart  Description automatically generated | Chart, treemap chart  Description automatically generated |  |

Test 2 – using keras dataset

|  |  |
| --- | --- |
| ***Modifier*** | Ann |
| ***File*** | tf.keras.datasets.mnist |
| ***Pre-processor Paramater*** |  |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"sparse\_categorical\_crossentropy","metrics": ["accuracy"],  "add1":{"method":"Flatten","params":{"input\_shape":[28,28]}},  "add2":{"method":"Dense","params":{"units":128,"activation":"relu"}},  "add3":{"method":"Dropout","params":{"rate":0.2}},  "add4":{"method":"Dense","params":{"units":10,"activation":"softmax"}},  "fit":{"method":"fit","params":{"epochs":10}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7a2: Artificial Neural Network Execution: Graph / Excel Output | | | |
| Chart, line chart  Description automatically generated | Chart  Description automatically generated |  |  |

Test 3 – using auto-generated continuous numbers from -3 to 3

|  |  |
| --- | --- |
| ***Modifier*** | Ann |
| ***File*** | auto\_generated\_-3to3 |
| ***Pre-processor Paramater*** |  |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"mse","metrics": ["accuracy"],  "add1":{"method":"Dense","params":{"units":128,"activation":"relu"}},  "add2":{"method":"Dense","params":{"units":1,"activation":"softmax"}},  "fit":{"method":"fit","params":{"epochs":10}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7a3: Artificial Neural Network Execution: Graph / Excel Output | | | |
| Chart, scatter chart  Description automatically generated | Chart, line chart  Description automatically generated |  |  |

* 1. CNN – Convolutional Neural Network are distinguished from other neural networks by its superior performance with image, speech, or audio signal inputs composed of 3 main layers: convolutional layer, pooling layer, fully connected layer. (IBM Cloud Education, 2022)

(SuperDataScience Team, 2018a), or

which denotes i = height, j = width, c = color; H\*W\*C

Summary of CNN Architecture

Strided Conv

Strided Conv

Dense

(FC)

Conv

Pool

Conv

Pool

Conv

Pool

Dense

(FC)

Dense

(FC)

Strided Conv

Dense

(FC)

Test 1 – Tensorflow Keras Dataset

|  |  |
| --- | --- |
| ***Modifier*** | Cnn |
| ***File*** | tf.keras.datasets.fashion\_mnist |
| ***Pre-processor Paramater*** |  |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"sparse\_categorical\_crossentropy","metrics": ["accuracy"],  "add1":{"method":"Conv2D","params":{"filters":32,"kernel\_size":[3,3],"strides":2,"activation":"relu"}},  "add2":{"method":"Conv2D","params":{"filters":64,"kernel\_size":[3,3],"strides":2,"activation":"relu"}},  "add3":{"method":"Flatten","params":{}},  "add4":{"method":"Dropout","params":{"rate":0.2}},  "add5":{"method":"Dense","params":{"units":512,"activation":"relu"}},  "add6":{"method":"Dropout","params":{"rate":0.2}},  "add7":{"method":"Dense","params":{"units":0,"activation":"softmax"}},  "fit":{"method":"fit","params":{"epochs":3}}} |
| ***Graph Parameter*** | {"ylabel":["T-shirt/top","Trouser","Pullover","Dress","Coat","Sandal","Shirt","Sneaker","Bag","Ankle boot"]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7b1: Convolutional Neural Network Execution: Graph / Excel Output | | | |
| Chart, line chart  Description automatically generated | Graphical user interface  Description automatically generated | A picture containing histogram  Description automatically generated |  |

Test 2 – Tensorflow Using 1D for Natural Language Processing

|  |  |
| --- | --- |
| ***Modifier*** | Cnn |
| ***File*** | spam.csv |
| ***Pre-processor Paramater*** | {"encoding":"ISO-8859-1","columns":["labels","data"],"x":"data","y":"labels","test\_size":0.33} |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"binary\_crossentropy","metrics":["accuracy"],  "add1":{"method":"Conv1D","params":{"filters":32,"kernel\_size":3,"activation":"relu"}},  "add2":{"method":"MaxPooling1D","params":{"pool\_size":3}},  "add3":{"method":"Conv1D","params":{"filters":64,"kernel\_size":3,"activation":"relu"}},  "add4":{"method":"MaxPooling1D","params":{"pool\_size":3}},  "add5":{"method":"Conv1D","params":{"filters":128,"kernel\_size":3,"activation":"relu"}},  "add6":{"method":"GlobalMaxPooling1D","params":{}},  "add7":{"method":"Dense","params":{"units":1,"activation":"sigmoid"}},  "fit":{"method":"fit","params":{"epochs":5}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7b2: Convolutional Neural Network Execution: Graph / Excel Output | | | |
| Chart, line chart  Description automatically generated | Chart, treemap chart  Description automatically generated |  |  |

Test 3 – Tensorflow Using Local Directory with encoding enhancement. The sample has no defined y so no confusion matrix or misclassified sample generated, as it’s not ideal to define the y one by one. The directory uses the base in the config.json

|  |  |
| --- | --- |
| ***Modifier*** | Cnn |
| ***File*** | {"filename":"cats\_n\_dogs","train\_file":"dataset\\training\_set","test\_file":"dataset\\test\_set",  "predict\_file":"dataset\\single\_prediction\\cat\_or\_dog\_1.jpg"} |
| ***Pre-processor Paramater*** | {"has\_y":false,"test\_size":1,"target\_size":[64,64],"batch\_size":32,"class\_mode":"binary",  "image":{"method":"ImageDataGenerator","params":{"rescale":0,"shear\_range":0.2,"zoom\_range":0.2,"horizontal\_flip":true}}} |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"binary\_crossentropy","metrics":["accuracy"],  "add1":{"method":"Conv2D","params":{"filters":32,"kernel\_size":3},"activation":"relu","input\_shape":[64,64,3]},  "add2":{"method":"MaxPool2D","params":{"pool\_size":2,"strides":2}},  "add3":{"method":"Conv2D","params":{"filters":32,"kernel\_size":3,"activation":"relu"}},  "add4":{"method":"MaxPool2D","params":{"pool\_size":2,"strides":2}},  "add5":{"method":"Flatten","params":{}},  "add6":{"method":"Dense","params":{"units":128,"activation":"relu"}},  "add7":{"method":"Dense","params":{"units":1,"activation":"sigmoid"}},  "fit":{"method":"fit","params":{"epochs":3}}} |
| ***Graph Parameter*** | {“ylabel":["cat","dog"]} |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7b3: Convolutional Neural Network Execution: Graph / Excel Output | | | |
| Chart, line chart  Description automatically generated |  |  |  |

* 1. Long Short-Term Memory – LSTM (=1) is a variant of Recurrent Neural Network that is efficient for it allows block processing of neurons usually given time-series dataset to solve vanishing gradient (<1) and exploding gradient (>1) problems thru sigmoid (logistic) and tanh activation functions, ranging from (0,1) and (-1,1) respectively.(SuperDataScience Team, 2018b) [34]

Summary of LSTM Architecture [34]

A

tanh

A

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | {"filename":"Google\_Stock\_Price","train\_file":"Google\_Stock\_Price\_Train.csv","test\_file":"Google\_Stock\_Price\_Test.csv"} |
| ***Pre-processor Paramater*** | {"timesteps":60,"interval":20,"x":["Open"],"scaler":{"method":"MinMaxScaler","params":{"feature\_range":[0,1]}}} |
| ***Modifier Paramater*** | "optimizer":"adam","loss":"mean\_squared\_error","metrics":["accuracy"],  "add1":{"method":"LSTM","params":{"units":50,"return\_sequences":true,"input\_shape":[60,1]}},  "add2":{"method":"Dropout","params":{"rate":0.2}},  "add3":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add4":{"method":"Dropout","params":{"rate":0.2}},  "add5":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add6":{"method":"Dropout","params":{"rate":0.2}},  "add7":{"method":"LSTM","params":{"units":50}},  "add8":{"method":"Dropout","params":{"rate":0.2}},  "add9":{"method":"Dense","params":{"units":1}},  "fit":{"method":"fit","params":{"epochs":100,"batch\_size":32}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7c1:Recurrent Neural Network (LSTM) Execution: Graph / Excel Output | | | |
| Calendar  Description automatically generated with low confidence | Chart, line chart  Description automatically generated |  |  |

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | {"filename":"Google\_Stock\_Price1",  "train\_file":"Google\_Stock\_Price\_Train.csv","test\_file":"Google\_Stock\_Price\_Test.csv"} |
| ***Pre-processor Paramater*** | {"timesteps":60,"interval":20,"x":["High"],  "scaler":{"method":"MinMaxScaler","params":{"feature\_range":[0,1]}}} |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"mean\_squared\_error","metrics":["accuracy"],  "add1":{"method":"LSTM","params":{"units":50,"return\_sequences":true,"input\_shape":[60,1]}},  "add2":{"method":"Dropout","params":{"rate":0.2}},  "add3":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add4":{"method":"Dropout","params":{"rate":0.2}},  "add5":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add6":{"method":"Dropout","params":{"rate":0.2}},  "add7":{"method":"LSTM","params":{"units":50}},  "add8":{"method":"Dropout","params":{"rate":0.2}},  "add9":{"method":"Dense","params":{"units":1}},  "fit":{"method":"fit","params":{"epochs":100,"batch\_size":32}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7c2:Recurrent Neural Network (LSTM) Execution: Graph / Excel Output | | | |
|  |  |  |  |

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | {"filename":"Google\_Stock\_Price2",  "train\_file":"Google\_Stock\_Price\_Train.csv","test\_file":"Google\_Stock\_Price\_Test.csv"} |
| ***Pre-processor Paramater*** | {"timesteps":120,"interval":20,"x":["Open"],"scaler":{"method":"MinMaxScaler","params":{"feature\_range":[0,1]}}} |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"mean\_squared\_error","metrics":["accuracy"],  "add1":{"method":"LSTM","params":{"units":50,"return\_sequences":true,"input\_shape":[120,1]}},  "add2":{"method":"Dropout","params":{"rate":0.2}},  "add3":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add4":{"method":"Dropout","params":{"rate":0.2}},  "add5":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add6":{"method":"Dropout","params":{"rate":0.2}},  "add7":{"method":"LSTM","params":{"units":50}},  "add8":{"method":"Dropout","params":{"rate":0.2}},  "add9":{"method":"Dense","params":{"units":1}},  "fit":{"method":"fit","params":{"epochs":100,"batch\_size":32}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7c3:Recurrent Neural Network (LSTM) Execution: Graph / Excel Output | | | |
|  |  |  |  |

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | {"filename":"Google\_Stock\_Price3",  "train\_file":"Google\_Stock\_Price\_Train.csv","test\_file":"Google\_Stock\_Price\_Test.csv"} |
| ***Pre-processor Paramater*** | {"timesteps":60,"interval":20,"x":["Open"],"scaler":{"method":"MinMaxScaler","params":{"feature\_range":[0,1]}}} |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"mean\_squared\_error","metrics":["accuracy"],  "add1":{"method":"LSTM","params":{"units":50,"return\_sequences":true,"input\_shape":[60,1]}},  "add2":{"method":"Dropout","params":{"rate":0.2}},  "add3":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add4":{"method":"Dropout","params":{"rate":0.2}},  "add5":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add6":{"method":"Dropout","params":{"rate":0.2}},  "add7":{"method":"LSTM","params":{"units":50,"return\_sequences":true}},  "add8":{"method":"Dropout","params":{"rate":0.2}},  "add9":{"method":"LSTM","params":{"units":50}},  "add10":{"method":"Dropout","params":{"rate":0.2}},  "add11":{"method":"Dense","params":{"units":1}},  "fit":{"method":"fit","params":{"epochs":100,"batch\_size":32}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7c4:Recurrent Neural Network (LSTM) Execution: Graph / Excel Output | | | |
|  |  |  |  |

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | {"filename":"Google\_Stock\_Price4",  "train\_file":"Google\_Stock\_Price\_Train.csv","test\_file":"Google\_Stock\_Price\_Test.csv"} |
| ***Pre-processor Paramater*** | {"timesteps":60,"interval":20,"x":["Open"],"scaler":{"method":"MinMaxScaler","params":{"feature\_range":[0,1]}}} |
| ***Modifier Paramater*** | {"optimizer":"adam","loss":"mean\_squared\_error","metrics":["accuracy"],  "add1":{"method":"LSTM","params":{"units":100,"return\_sequences":true,"input\_shape":[60,1]}},  "add2":{"method":"Dropout","params":{"rate":0.2}},  "add3":{"method":"LSTM","params":{"units":100,"return\_sequences":true}},  "add4":{"method":"Dropout","params":{"rate":0.2}},  "add5":{"method":"LSTM","params":{"units":100,"return\_sequences":true}},  "add6":{"method":"Dropout","params":{"rate":0.2}},  "add7":{"method":"LSTM","params":{"units":100}},  "add8":{"method":"Dropout","params":{"rate":0.2}},  "add9":{"method":"Dense","params":{"units":1}},  "fit":{"method":"fit","params":{"epochs":100,"batch\_size":32}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7c5:Recurrent Neural Network (LSTM) Execution: Graph / Excel Output | | | |
|  |  |  |  |

* 1. RNN – Recurrent Neural Network is one of the most advanced algorithms in the supervised deep learning as it mimics the function of the brain of frontal lobe for short-term memory, parietal lobe for sensation, perception, constructing a spatial coordination system in a real world behaving in a squash network, on top of one another, unrolling in a temporal loop with a whole layer of neurons instead of one neuron. (SuperDataScience Team, 2018c)

Architecture of RNN(udemy, n.d.)

SimpleRNN

X(t)

h(t-1)

h(t)

Where:

Test 1: RNN compared with ANN

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | auto\_noise\_200by0.1 |
| ***Pre-processor Paramater*** | {"timesteps":10,"model\_add":true,"x\_shape":3} |
| ***Modifier Paramater*** | {"optimizer":"Adam(lr=0.001)","loss":"mse","metrics": ["accuracy"],  "add1":{"method":"SimpleRNN","params":{"units":15,"activation":"None"}},  "add2":{"method":"Dense","params":{"units":1}},"fit":{"method":"fit","params":{"epochs":80}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7d1:Recurrent Neural Network (SimpleRNN) Execution: Graph / Excel Output | | | |
|  |  |  |  |

Test 1a: Compared with ANN without noise

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | auto\_generated\_200by0.1 |
| ***Pre-processor Paramater*** | {"timesteps":10,"model\_add":true} |
| ***Modifier Paramater*** | {"optimizer":"Adam(lr=0.1)","loss":"mse","metrics": ["accuracy"],"add1":{"method":"Dense","params":{"units":1}},"fit":{"method":"fit","params":{"epochs":80}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7d1a:Recurrent Neural Network (SimpleRNN) Execution: Graph / Excel Output | | | |
|  |  |  |  |

Test 1b: Compared with ANN with noise

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | auto\_noise`\_200by0.1 |
| ***Pre-processor Paramater*** | {"timesteps":10,"model\_add":true} |
| ***Modifier Paramater*** | {"optimizer":"Adam(lr=0.1)","loss":"mse","metrics": ["accuracy"],"add1":{"method":"Dense","params":{"units":1}},"fit":{"method":"fit","params":{"epochs":80}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7d1b:Recurrent Neural Network (SimpleRNN) Execution: Graph / Excel Output | | | |
|  |  |  |  |

Test 2: RNN (b) compared with ANN (a), GRU (c), LSTM (d)

1. ANN

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | auto\_generated\_400by0.1raised2 |
| ***Pre-processor Paramater*** | {"timesteps":10,"model\_add":true} |
| ***Modifier Paramater*** | "add1":{"method":"Dense","params":{"units":1}},"fit":{"method":"fit","params":{"epochs":80}}} |
| ***Graph Parameter*** |  |

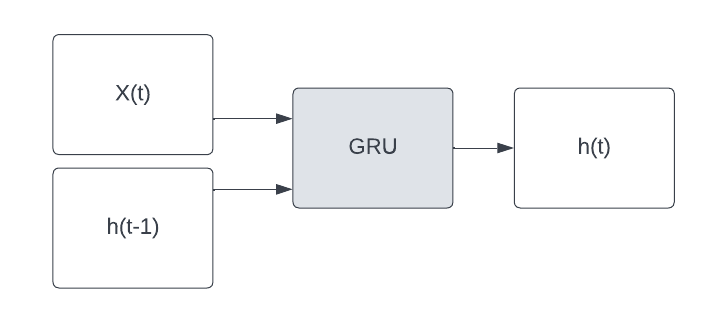
|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7d2a:Recurrent Neural Network (ANN) Execution: Graph / Excel Output | | | |
|  |  |  |  |

1. RNN

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | auto\_generated\_400by0.1raised2 |
| ***Pre-processor Paramater*** | {"timesteps":10,"model\_add":true} |
| ***Modifier Paramater*** | {"optimizer":"Adam(lr=0.05)","loss":"mse","metrics":["accuracy"],  "add1":{"method":"SimpleRNN","params":{"units":10}},  "add2":{"method":"Dense","params":{"units":1}},  "fit":{"method":"fit","params":{"epochs":200,"batch\_size":32}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7d2b:Recurrent Neural Network (SimpleRNN) Execution: Graph / Excel Output | | | |
|  |  |  |  |

1. GRU – Gated Recurrent Neural Network Unit

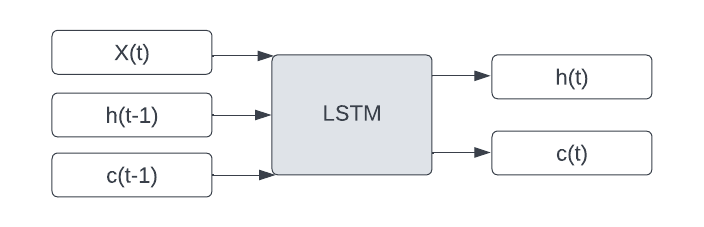


Where:

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | auto\_generated\_400by0.1raised2 |
| ***Pre-processor Paramater*** | {"timesteps":10,"model\_add":true} |
| ***Modifier Paramater*** | {"optimizer":"Adam(lr=0.05)","loss":"mse","metrics":["accuracy"],  "add1":{"method":"GRU","params":{"units":10}},  "add2":{"method":"Dense","params":{"units":1}},  "fit":{"method":"fit","params":{"epochs":200,"batch\_size":32}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7d2c:Recurrent Neural Network (GRU) Execution: Graph / Excel Output | | | |
|  |  |  |  |

1. LSTM



Where:

()

|  |  |
| --- | --- |
| ***Modifier*** | Rnn |
| ***File*** | auto\_generated\_400by0.1raised2 |
| ***Pre-processor Paramater*** | {"timesteps":10,"model\_add":true} |
| ***Modifier Paramater*** | {"optimizer":"Adam(lr=0.05)","loss":"mse","metrics":["accuracy"],  "add1":{"method":"LSTM","params":{"units":10}},  "add2":{"method":"Dense","params":{"units":1}},  "fit":{"method":"fit","params":{"epochs":200,"batch\_size":32}}} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.7d2d:Recurrent Neural Network (LSTM) Execution: Graph / Excel Output | | | |
|  |  |  |  |

|  |  |
| --- | --- |
| Batch Run Excel | Snapshot |
|  |  |

|  |  |
| --- | --- |
| ***Table 7: Neural Network Batch Run*** | |
| Result Analysis | : ANN is executed using initially 3 dataset: (1) *Churn\_Modelling.csv* – to check customer 0-not exited, 1-exited the account based on credit score to know fraudulent customer, (2) *tf.keras.datasets.mnist* – numbers 0-9, (3) *auto\_generated\_-3to3* – auto-generated real numbers from -3 to 3 in rows. has , 0.853875, has  0.070294, 0.977517, has  1.992985, 0, with the loss score decreases and accuracy score increases over each epoch, except for the when accuracy is constant to zero with the continuous target.  In customer who didn't close the account about 1527 is more than who closed 137 as the fraudulent customer can be determined who closed their account early with high credit score. In 1 has the most correctly classified items, 2-7 and 7-9 most confused numbers.  : CNN is executed using 2 dataset: (1) *tf.keras.datasets.fashion\_mnist* that contains images of 0-T-shirt/top, 1-Trouser, 2-Pullover, 3-Dress, 4-Coat, 5-Sandal, 6-Shirt, 7-Sneaker, 8-Bag, 9-Ankle boot, (2) *spam.csv* – for text processing with 2 variables has  0.257122, 0.9036, has  0.070294, 0.977517 with the loss score decreases and accuracy score increases over each epoch of the datasets. one of the most misclassified is T-shirt/top with and shirt with having misclassified examples of 180, one of the most correctly classified is the trouser with has value of 981. One misclassified exampled generated is the -T-shirt/top, -Pullover.  : RNN is executed using 3 dataset: (1) *Google\_Stock\_Price* – google stock price with 5 examples of LSTM combined with Dropout: (a) 50 neurons using ‘*Open*’ column of 3 layers with 60 timesteps, (b) 50 neurons using ‘*High*’ column of 3 layers with 60 timesteps, (c) 50 neurons using ‘*Open*’ column of 3 layers with 120 timesteps, (d) 50 neurons using ‘*Open*’ column of 4 layers with 60 timesteps, (e) 100 neurons using ‘*Open*’ column of 3 layers with 60 timesteps, (2) *auto\_noise\_200by0.1* – random generated number of sin(200 \* 0.1) added the noise using SimpleRNN, ANN, (3) *auto\_generated\_400by0.1raised2* – random generated number of sin() using ANN, SimpleRNN, GRU, LSTM. For 100 neurons perform the best having  0.001709. For ANN performs better than the SimpleRNN  having 0.015916 compared to  0.017226, which says one needs to test different neural network, as usually RNN is the compressed neural network horizontally, vertically, so efficiency-wise, it’s effective, but performance can also depends on the dataset type. For with high and low frequencies, ANN performs poorly as it didn’t capture different waves, SimpleRNN is able to capture high frequency waves but not the low frequency, GRU performs better but only captured a bit of the low frequencies, LSTM performs best among the rest as identified by the graph as well as the 0.003288 |

### Others (Supervised or Unsupervised)

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_unsupervised.txt |

1. SOM – Self Organizing Map, pioneered by Finnish researcher Dr. Tuevo Kohenen is an unsupervised learning model intended for applications that maintains topology between input and output spaces is of importance via feature maps using a distance method, to give a pragmatic visualization of high-dimensional data into lower dimensional spaces, e.g. 2D. (Eklavya, 2019)

Test 1 – without test\_size

|  |  |
| --- | --- |
| ***Modifier*** | SOM |
| ***File*** | Credit\_Card\_Applications.csv |
| ***Pre-processor Paramater*** |  |
| ***Modifier Paramater*** |  |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.8a1:SOM Execution: Graph / Excel Output | | | |
| Chart, bar chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Calendar  Description automatically generated |  |

Test 2 – with test\_size

|  |  |
| --- | --- |
| ***Modifier*** | SOM |
| ***File*** | Credit\_Card\_Applications.csv |
| ***Pre-processor Paramater*** | {"test\_size":0.2} |
| ***Modifier Paramater*** |  |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.8a2:SOM Execution: Graph / Excel Output | | | |
| Chart, bar chart  Description automatically generated | Chart, treemap chart  Description automatically generated | Calendar  Description automatically generated |  |

|  |  |
| --- | --- |
| Batch RunExcel | Snapshot |
|  |  |

|  |  |
| --- | --- |
| Table 8: Unsupervised Batch Run | |
| Result Analysis | *Credit\_Card\_Applications.csv* – dataset is the credit card application with customer number who will get accepted or not without and with train-test split of 0.2, using SOM with 1 nearest neighbor, the one with train\_test\_split performs better than the one without split, having the 0.528986, = 0.5, , = 0.48, = 0.05093. |

1. RBM – Restricted Boltzmann Machine

The modelling is derived from Boltzmann Machine wherein the hidden modes and the visible nodes are connected to one another. The only difference is that there’s no connection between the same nodes, e.g. visible to visible, hidden to hidden, hence, the name of *restricted* Boltzmann machine.

Architecture of Restricted Boltzmann Machine

v – visible; h -hidden

Test 1

|  |  |
| --- | --- |
| ***Execute*** | python -W ignore main.py -i input\_unsupervised1.txt |

|  |  |
| --- | --- |
| ***Modifier*** | RBM |
| ***File*** | {"filename":"movies","train\_file":"ml-100k/u1.base","test\_file":"ml-100k/u1.test"} |
| ***Pre-processor Paramater*** | {"delimiter":"\t","xlabel":["user","movie","rating","date"]} |
| ***Modifier Paramater*** | {"hidden":100,"batch\_size":100,"epoch":10} |
| ***Graph Parameter*** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Table B.8a1:SOM Execution: Graph / Excel Output | | | |
|  |  |  |  |

# Software Use

Execute Program

Input Files

Figure: Software Use Flowchart

Output Files

Two files need to be configured: config.json and input.txt

1. Config.json – global configuration

|  |  |  |
| --- | --- | --- |
| ***Folder Name*** | ***Data Type*** | ***Definition*** |
| Base |  | base directory |
| Input | str | input directory of the csv file |
| Output | str | output directory where the results are spooled |
| Log | str | log file location |
| ***Name*** | ***Data Type*** | ***Definition*** |
| test\_size | float | global train test split 0, means no split, which can be overridden by the per record setup |
| Save | bool | true, yes save the output XLS files in the specified output directory, else false |
| file\_stats | str | Prefix of the stats file name appended by the date |
| Graph | bool | true, yes save the graphic PNG files in the specified output directory, else false |
| max\_cardinality | int | number of feature threshold to use for OneHotEncoder, else use GLMM |
| regression\_scores | list | scores for the regression model specified in the API sklearn.metrics(scikit-learn, n.d.-s) |
| classification\_scores | list | scores for the classification model specified in the API sklearn.metrics(scikit-learn, n.d.-q), also used by non-supervised learnings like SOM |
| cluster\_scores | list | scores for the cluster model specified in the API sklearn.metrics(scikit-learn, n.d.-r) |
| neural\_scores | list | Scores for the neural network using tensorflow to default only for accuracy, loss scores, else for some examples the other scores are default in the input file when building the model like mean square error (mse). |

1. Input.txt
2. File – the file the dataset is referring
3. Value

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| File | str | The file in the input directory, or can be the IOT based on the scikit-learn using the prefix itself, i.e. fetch, open, or tensorflow using the prefix of tf.keras.datasets |

File can be loaded automatically using the online dataset

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| Load\_<dataset>,  Fetch\_<dataset | str | For scikit-learn predefined dataset |
| tf.keras.datasets.<dataset> | str | For tensorflow keras dataset |

File can be auto-generated

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| auto\_generated\_<lower\_bound>to<upper\_bound> | str | For auto-generated dataset with lower and upper bounds, where X is the random number of those bounds, y = cos2x1+cos3x2 |
| auto\_generated\_<number>by<multiplier> | str | For auto-generated where X= np.sin(multiplier\*the sequence of the number excluding that number), y = the X from the timestep indicated |
| auto\_noise\_<number>by<multiplier> | str | For auto-generated numbers, similar with auto\_generated\_<number>by<multiplier> adding noise of a random <number> indicated multiplied by the multiplier |
| auto\_generated\_<number>by<multiplier>raised<exponent> / auto\_noise\_<number>by<multiplier>raised<exponent> | str | For auto-generated numbers:  np.sin((random<number>\*<multiplier>)\*\*<exponent>) + [noise] |

For the local directory handling of separate train and test directory of the image processing dictionary processing can be used via:

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| Filename | Str | Specify the filename will result in the output directory of the dataset tested, specifically used only for CNN |
| train\_dir | Str | The train files dataset directory of the CNN images |
| test\_dir | Str | The test files dataset directory of the CNN images |
| predict file | Str | The predict test file dataset of the specific CNN image |

For the local files handling of separate train and test files processing that used the config.json base input directory as its file location:

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| Filename | Str | Specify the filename will result in the output files of the dataset tested, specifically used only for RNN |
| train\_file | Str | The train files dataset |
| test\_file | Str | The test files dataset |

1. Preprocessor – the pre-processor is divided into values and function setup. Values are for predefined in the module, while *function=(imputer,scaler,feature)* is a JSON format
2. Value

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| has\_y | bool | true / false – true when dataset has predefined y else false |
| X | str / int | X column name or column index for user specified options |
| Y | str / int | y column name or column index for user specified options |
| test\_size | float | 0 < values < 1 |
| predict\_test | list, any primary data type | user defined test for the prediction of the estimator, spool the result in the estimator XLS |
| random\_state | Int | train test split to set the random state parameter |
| stop\_words | Str | For natural language processing to indicate the stop\_words used |
| Encoding | Str | Language encoding specific for reading the csv file |
| target\_size | list / set | Target size of the image specifically for local directory file loading |
| batch\_size | Int | Batch size for the image processing specifically for local directory file loading |
| class\_mode | Str | Type mode of target specifically for local directory file loading |
| Timesteps | Int | The timestep block of a dataset, e.g. 60 means the row0 is from start time 0 to 59, row1 from start time 60 to 119, and so forth, used for LSTM |
| Interval | Int | Interval is the until to the time of testing dataset starting from the timesteps, used for LSTM |
| Model\_add | bool | Adding tensorflow function the primary way, not by sequential add |
| X\_shape | int | Dimension shape of the X dataset, e.g. 2D is x\_shape =2, 3D is x\_shape =3 |

1. Function – json format is <function>:{“method”:<method\_name>,”params”:<parameters>}

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| Function | Constant | Constant values can be *“imputer”*, “scaler”, “feature”, “text” to indicate the respective function of the imputer, scaler algorithm, feature selections, text for natural language processing, image for CNN of local directory file handling |
| method\_name | Constant | Corresponding method name for the constant function defined |
| Parameters | Dict | Actual function parameters |

* Feature – feature selections options are specified in the Appendix
* Imputer – can be SimpleImputer, IterativeImputer, MissingIndicator, KNNImputer (scikit-learn, n.d.-p)
* Scaler (Pandey, n.d.)
  1. MinMaxScaler – avoid data distortion using *min()* and *max()*
  2. Standard Scaler -standarize the dataset using *mean()* and *sd()*
  3. Robust Scaler – scales feature robust to outliers via quartiles
  4. MaxAbsScaler – scales each feature by its maximum absolute value
* Text – natural language processing that can use Porter Stemmer
* Image – CNN image preprocessing of local files that use ImageDataGenerator() of the keras

1. Model – dictionary format that takes in the default estimator arguments as is, except for the neural network wherein it can have the combination of values and functions. For the neural network needs to setup that is done by using the functions like add, fit and the values would be the values passed on the model estimator, i.e. compile().

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| Function | Constant | For neural network, constant values can be *“add<n>”*, “*fit*” to indicate the respective function of the ‘*add<n>*’ – adding of the neural network with the value of <n> to indicate the number of the input, hidden, output layers added, ‘*fit*’- is the additional optional functional parameter that can be added when training or fitting the training sets to the test sets. |
| method\_name | Constant | Corresponding method name for the constant function defined |
| Parameters | Dict | Actual function parameters accordingly |

For RBM, the implementation is done using torch, the parameter used:

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| Hidden | Int | The number of hidden nodes |
| batch\_size | Int | The number of batch\_size |
| Epoch | int | The number of epoch |

1. Graph – is a dictionary value setup

|  |  |  |
| --- | --- | --- |
| ***Name*** | ***Type*** | ***Definition*** |
| Title | Str | Title to be use in the estimator graph, by default is the classifier name with training/test option |
| Xlabel | str | Label in the x-axis of the estimator graph, by default is the X column name |
| Ylabel | Str | Label in the y-axis of the estimator graph, by default is the y column name |
| Smooth | number, numpy of number | Sets the jitter values for the contour of the estimator like the regression, classification |

To execute the tool use the below command with *-i* option to change the input file *<input\_\*>.txt*:

*python -W ignore main.py -i input\_cluster.txt*

Alternatively, the tool can be installed as a project with the predefined modules after the project files have been extracted.

|  |  |  |
| --- | --- | --- |
| ***No*** | ***Command*** | ***Description*** |
| 1 | cd <directory>\my\_practise\_v2 | Unzip the project file and go to the project directory |
| 2 | pip install pipenv | Install the pipenv in the command console terminal of any environment |
| 3 | pipenv install | Install the library modules for the project |
| 4 | pipenv run python -W ignore main.py -I <input\_\*>.txt | Execute the python according to the input files |

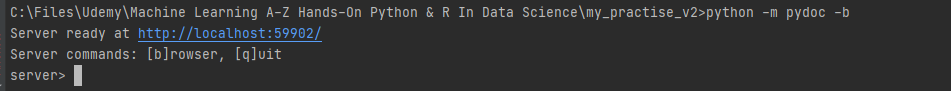
1. Python Documentation

The application utilizes the default Python documentation to have an API web references to generate the function modules used.

To execute use:

*python -m pydoc -b*

After executing the command, it will give a local web server for the API modules used in the batch machine learning.



Timeline

Description automatically generated

Graphical user interface, application

Description automatically generated

# Conclusion, Recommendations and Future Use

## Conclusion

The tool is a simple input driven file to generate the batch process in a sequential format to help ease the use of users and help train, predict the model by providing graphs, metrices. The sample batch is executing 45 models and/or examples from regression, classification, clustering, rules, sampling – reinforcement learnings, text – natural language processing to deep learning for about 30 minutes to 1-hour-quarter in 2 to 4 Gig free memory in group mode by classification of regression, classifier, cluster, rules, sample, text, neural network, unsupervised learnings, which can be more efficient and effective than developing multiple scripts as it can be through setup with a bit of customization. It may also run faster for a better memory with good CPU. It also organizes the results per filename, data and time stamp. The tool can be upgraded, made into a reference for some functionalities.

### Recommendations

The tool can be improved further by optimizing the object-oriented functionality of Python like (inheritance, repr, start), and optimizing further the pipeline. Additional features and functionality can be added. Though, at the moment, it’s usable tool to have the setup template of the various classifications from regression, classifier, cluster, neural network, etc., but, it’s a text-based input file, so to have a GUI to setup the input files can eliminate the need for the user to be familiar with Python machine learning. But, the input-text based template can be reusable for other datasets and/or models. which is one model of file processing with one line of record in the input files that parameters from value to functions are dynamically setup. So, the implications of having multiple scripts of various modelling techniques that one saves the output in folders, it can be a 1 input file or few input files organized in name resulting to output of name, date, time, more on input file setup instead of Python AI development. As an ad hoc, experts can add layering pf modelling or create their own models like LeNet, GoogleNet due to dynamic parameters.

### Future Use

The tool is usable for simple batch processing of one record at a time with some constraints that can be extended for future use. The text in paragraph form and image processing maybe included in the tool in limited capacity like PorterStemmer with some neural networks only.

1. Feature Selection
2. sklearn.feature\_selection [9]

The feature selection not handled are SelectFromModel, RFE, RFECV, SequentialFeatureSelector, VarianceThreshold.

1. Matrix Decomposition: sklearn.decomposition [7]

The image and text processing are tested in limited capacity. So, there can be several decompositions not fully tested: LatentDirichletAllocation, NMF, SparseCoder, TruncatedSVD among others.

The testing of the modelling techniques, functions and attributes are limited only to the specified ones with time constraint and other factors. The development (from scratch) is a simple batch according to the applied learnings from the different sources and previous practices. The baseline of the testing is done in comparison only to the Udemy to have a foundation using its input files with an extension of the things learn during the Machine Learning coursework and other online learnings.

# Appendix

## Appendix: Disclaimer

Auckland University of Technology

Master of Analytics

Research Project

**Disclaimer:**

**Clients should note the general basis upon which the Auckland University of Technology undertakes its student projects on behalf of external sponsors:**

While all due care and diligence will be expected to be taken by the students, (acting in data analytics, statistics, research or other professional capacities), and the Auckland University of Technology, and student efforts will be supervised by experienced AUT lecturers, it must be recognised that these projects are undertaken in the course of student instruction. There is therefore no guarantee that students will succeed in their efforts. This inherently means that the client assumes a degree of risk. This is part of an arrangement, which is intended to be of mutual benefit. On completion of the project it is hoped that the client will receive a professionally documented project report, while the students are exposed to live external environments and problems, in a realistic project and customer context. In consequence of the above, the students, acting in their assigned professional capacities and the Auckland University of Technology, disclaim responsibility and offer no warranty in respect of outcomes of the project both in relation to their use and results from their use.

## Feature Selection

1. sklearn.feature\_selection [9]

|  |  |  |
| --- | --- | --- |
| **No** | **Feature** | **Arguments** |
| 1 | GenericUnivariateSelect | score\_func=chi2, f\_classif, f­\_regression, mutual\_info\_classif, mutual\_info\_regression  **mode**{‘percentile’, ‘k\_best’, ‘fpr’, ‘fdr’, ‘fwe’}, default=’percentile’  param: float or int depending on the feature selection, default=1e-5 |
| 2 | SelectPercentile  - percentile of the highest score¶ | score\_func:callable, default=f\_classif  percentile:int, default=10 |
| 3 | SelectKBest  - k highest score | score\_func:callable, default=f\_classif  k:int or “all”, default=10 |
| 4 | SelectFpr  - FPR rate¶ | score\_func:callable, default=f\_classif  alpha:float, default=5e-2  - highest p-value to be kept |
| 5 | SelectFdr  - false discovery rate | score\_func:callable, default=f\_classif  alpha:float, default=5e-2  - highest uncorrected p-value to be kept |
| 6 | SelectFwe  - family wise error rate | score\_func:callable, default=f\_classif  alpha:float, default=5e-2  - highest uncorrected p-value to be kept |

|  |  |  |
| --- | --- | --- |
| **No** | **Function (score\_func)** | **Definition** |
| 1 | chi2 | Chi-squared stats of non-negative features for classification tasks. |
| 2 | f\_classif | ANOVA F-value between label/feature for classification tasks. |
| 3 | f\_regression | F-value between label/feature for regression tasks. |
| 4 | mutual\_info\_classif | Mutual information for a discrete target. |
| 5 | mutual\_info\_regression | Mutual information for a continuous target. |

The attributes returned are scores\_ and pvalues\_.

1. Matrix Decomposition: sklearn.decomposition [7]

The common attribute in matrix decomposition is *n\_components* to reduce dataset dimensionality. The scikit learn website is a solid reference of the other attributes.

|  |  |  |
| --- | --- | --- |
| **No** | **Feature** | **Definition** (scikit-learn, n.d.-l) |
| 1 | DictionaryLearning | Dictionary learning  This is for optimization problem to update the sparse code as a solution to multiple Lasso problem for image processing |
| 2 | FactorAnalysis | Factor analysis  It can model the variance in all direction independently (heteroscedastic noise) |
| 3 | FastICA | a fast algorithm for Independent Component Analysis  This separates the multivariate signals into maximum independent components. Typically, it is not use for dimensionality reduction, but, it can be used with some sparsity. It has no noise handling, which can be corrected if whitening is applied. |
| 4 | IncrementalPCA | Incremental principal components analysis  Faster then PCA to process in mini-batch in memory incrementally that supports large datasets |
| 5 | KernelPCA | Kernel Principal component analysis  Non-linear dimensionality reduction that supports both *transform* and *inverse\_transform* |
| 6 | LatentDirichletAllocation | Latent Dirichlet Allocation with online variational Bayes algorithm  A probabilistic model of discrete dataset such as text |
| 7 | MiniBatchDictionaryLearning | Mini-batch dictionary learning  It implements faster but less accurate version of dictionary learning algorithm suited for large datasets |
| 8 | MiniBatchSparsePCA | Mini-batch Sparse Principal Components Analysis |
| 9 | NMF | Non-Negative Matrix Factorization  Decomposition approach that assumes non-negative components in additive way, which is efficient to image and text processing that uses Frobenius norm, an extension of Euclidean distance |
| 10 | PCA | Principal component analysis  Data is not scale before applying SVD. Attribute *whiten=True* projects the data while scaling the component per unit variance, which is useful in Support Vector Machine with RBF kernel and K-Means algorithm  PCA with randomized SVD using *svd\_solver=’randomized’* is not the exact *inverse\_transform* even with the default *whiten=False* |
| 11 | SparsePCA | Sparse Principal Components Analysis  The degree of penalization, thus sparsity inducing to prevent noise for few training sets that enables regularized factorization via *alpha* hyperparatemer. |
| 12 | SparseCoder | Sparse coding  It transforms signals into sparse linear space that implements *transform\_method* not *fit* method |
| 13 | TruncatedSVD | Dimensionality reduction using truncated SVD (Singular Value Decomposition)  This is applied to term-document matrices to transform the text as latent “semantic” analysis (LSA) |
| 14 | dict\_learning | Solves a dictionary learning matrix factorization problem. |
| 15 | Fastica | Perform Fast Independent Component Analysis. |
| 16 | non\_negative\_factorization | non\_negative\_factorization |
| 17 | sparse\_encode | Sparse coding |

|  |  |
| --- | --- |
| ***Note:*** | For Linear Combination |

## Model Selection

The list is mainly for Regression, Classification having both usage.

|  |  |
| --- | --- |
| **Library** | **Model** |
| sklearn.linear\_model | ARDRegression, BayesianRidge, ElasticNet, ElasticNetCV, GammaRegressor, Hinge, Huber, HuberRegressor, Lars, LarsCV, Lasso, LassoCV, LassoLars, LassoLarsCV, LassoLarsIC, LinearRegression, Log, LogisticRegression, LogisticRegressionCV, ModifiedHuber, MultiTaskElasticNet, MultiTaskElasticNetCV, MultiTaskLasso, MultiTaskLassoCV, OrthogonalMatchingPursuit, OrthogonalMatchingPursuitCV, PassiveAggressiveClassifier, PassiveAggressiveRegressor, Perceptron, PoissonRegressor, RANSACRegressor, Ridge, RidgeCV, RidgeClassifier, RidgeClassifierCV, SGDClassifier, SGDRegressor, SquaredLoss, TheilSenRegressor, TweedieRegressor |
| sklearn.kernel\_ridge | KernelRidge |
| sklearn.svm | LinearSVC,LinearSVR,NuSVC,NuSVR,OneClassSVM,SVC,SVR |
| sklearn.neural\_network | BernoulliRBM, MLPClassifier,MLPRegressor |
| sklearn.ensemble | AdaBoostClassifier, AdaBoostRegressor, BaggingClassifier, BaggingRegressor, BaseEnsemble, ExtraTreesClassifier, ExtraTreesRegressor, GradientBoostingClassifier, GradientBoostingRegressor, IsolationForest, RandomForestClassifier, RandomForestRegressor, RandomTreesEmbedding, StackingClassifier, StackingRegressor, VotingClassifier, VotingRegressor |

## Metrices Used

1. Regression
   1. *R-Squared ()* – in regression model, the statistical measure represents the portion of the difference of the variance for a dependent variable (), which can be explained by the independent variables ().(Katara & Vaidya, 2022)

(Katara & Vaidya, 2022)

1. Classifier, Unsupervised Learnings
2. *Accuracy Score* – the proportion of the total number of predictions that were correct. (Sadawi, n.d.)
3. *Precision Score* - measure the model performance in measuring the count of true positives in the correct manner out of all positive predictions made. (Kumar, 2022)

(Kumar, 2022)

1. *Recall Score* - measure the model performance in terms of measuring the count of true positives in a correct manner out of all the actual positive value (Kumar, 2022)

(Kumar, 2022)

1. *F1-Score* - harmonic mean of precision and recall score to choose between precision or recall score can result in compromise in terms of model giving high false positives and false negatives respectively. (Kumar, 2022)

(Kumar, 2022)

1. *Matthew Correlation Coefficient (*MCC*) -* its advantage over F1 score and accuracy in binary classification evaluation is that it’s more informative than F1 score and accuracy in evaluating binary classification problems as it balances the ratios of the four confusion matrix categories. (wikipedia, n.d.) It’s defined to the phi coefficient, or mean square contingency coefficient, also closely related to chi-square for a binary classification. (wikipedia, n.d.)

(wikipedia, n.d.)

(wikipedia, n.d.)

1. Cluster
2. Calinski and Harabasz score - also known as the Variance Ratio Criterion, the score is defined as ratio of the sum of between-cluster dispersion and of within-cluster dispersion (scikit-learn, n.d.-a) having the higher the score the better the performance. (scikit-learn, n.d.-a)

(Zuccarelli, 2021)

Where = trace of the between group of dispersion matrix

= trace of the within-cluster dispersion matrix defined by:

(Zuccarelli, 2021)

(Zuccarelli, 2021)

1. Davies-Bouldin Index - the average similarity measure of each cluster with its most similar cluster, with the minimum score is zero, the lower values the better clustering performance. (Zuccarelli, 2021)

(HandWiki, n.d.),

where the measure of distance between cluster and , is the centroid of , is the centroid of , using a distance method, i.e., Euclidean distance, with at least 2 clusters , with the movement () of the points in cluster (, ) to the mean as:

(HandWiki, n.d.)

1. Silhouette Score and Silhouette Plot are used to measure the separation distance between clusters. (Zuccarelli, 2021) Silhouette Score is calculated using the mean intra-cluster distance ) and the mean nearest-cluster distance ) for each sample having the silhouette plot with proportional size/thickness of the silhouettes to the number of samples inside that cluster. (Zuccarelli, 2021)

(Zuccarelli, 2021)

Where = distance between each sample and the nearest cluster that the sample is not a part of cluster

= the mean distance within each cluster

1. Completeness Score - A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster with the performance metrics the higher the score the better.(scikit-learn, n.d.-g)

(oreilly, n.d.-a)

1. Homogeneity Score - this score checks the clustering algorithm that a specific cluster should contain only samples belonging to a single class with higher value indicates higher homogeneity. (oreilly, n.d.-b)

(oreilly, n.d.-b)

1. V-measure Score - harmonic mean between homogeneity and completeness, the higher the score the better the performance. (scikit-learn, n.d.-x)

(scikit-learn, n.d.-x)

1. Rules
2. Support – determines how popular an itemset is, as measured by the proportion of transactions in which an itemset appears. (Ng, n.d.)
3. Confidence - how likely item Y is purchased when item X is purchased, expressed as {X -> Y} that measures the proportion of transactions with item X, in which item Y also appears. (Ng, n.d.)

(Ng, n.d.)

1. Lift - determines how likely item Y is purchased when item X is purchased, while controlling for how popular item Y is. (Ng, n.d.)

(Ng, n.d.)

1. Neural Network
2. *Accuracy Score* – same as classifier
3. *Loss Score* - measures how good a neural network model is in performing a certain task, which in most cases is regression or classification that minimize the value of the loss function during the backpropagation step in order to make the neural network better. (Oppermann, n.d.)

(Oppermann, n.d.)

(Oppermann, n.d.)

# References

Bulao, J. (2022, March 14). *How Much Data Is Created Every Day in 2022?* https://techjury.net/blog/how-much-data-is-created-every-day/#gref

Burns, E., & Burke, J. (n.d.). Neural: ANN. *What Is a Neural Network? Explanation and Examples*. https://searchenterpriseai.techtarget.com/definition/neural-network

Dannina, H. (2020, January 29). Rule: Apriori. *Underrated Machine Learning Algorithms — APRIORI*. https://towardsdatascience.com/underrated-machine-learning-algorithms-apriori-1b1d7a8b7bc

datacamp. (n.d.). *Feature Selection: PCA - Variance Score*. https://www.datacamp.com/community/tutorials/principal-component-analysis-in-python

Eklavya. (2019, October 29). Nonsupervised:SOM. *Kohonen Self-Organizing Maps:A Special Type of Artificial Neural Network*. https://towardsdatascience.com/kohonen-self-organizing-maps-a29040d688da

Eremenko, K., & de Ponteves, H. (n.d.-a). NLP: Porter Stemmer, Bag of Words. *Udemy:Part 7 - Natural Language Processing, Porter Stemmer - Bag of Words*. https://www.udemy.com/course/machinelearning/learn/lecture/20091386#overview

Eremenko, K., & de Ponteves, H. (n.d.-b). *Sampling: Thompson Sampling*. https://www.udemy.com/course/machinelearning/learn/lecture/6456840#overview

Eremenko, K., & de Ponteves, H. (n.d.-c). *Sampling: Upper Confidence Bound (UCB)*. https://www.udemy.com/course/machinelearning/learn/lecture/6456832#overview

G., N. (n.d.). *101 Artificial Intelligence Statistics [Updated for 2022]*. https://techjury.net/blog/ai-statistics/

Gupta, A. (2019, June 12). Rule: ECLAT. *ML | ECLAT Algorithm*. https://www.geeksforgeeks.org/ml-eclat-algorithm/

HandWiki. (n.d.). *Davies–Bouldin index*. https://handwiki.org/wiki/Davies%E2%80%93Bouldin\_index#:~:text=The%20Davies%E2%80%93Bouldin%20index%20%28DBI%29%2C%20introduced%20by%20David%20L.,using%20quantities%20and%20features%20inherent%20to%20the%20dataset.

IBM Cloud Education. (2020, July 15). What is Machine Learning? *Machine Learning*. https://www.ibm.com/cloud/learn/machine-learning

IBM Cloud Education. (2022, October 20). Convolutional Neural Network. *Convolutional Neural Network*. https://www.ibm.com/cloud/learn/convolutional-neural-networks

Katara, H., & Vaidya, D. (2022, May 28). *What is R Squared (R2) in Regression?* https://www.wallstreetmojo.com/r-squared-formula/

Kumar, A. (2020, August 29). *Grid Search Explained – Python Sklearn Examples*. https://vitalflux.com/grid-search-explained-python-sklearn-examples/

Kumar, A. (2022, April 8). *Accuracy, Precision, Recall & F1-Score – Python Examples*. https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/#:~:text=Accuracy%20score%20is%20used%20to%20measure%20the%20model,true%20negatives%20out%20of%20all%20the%20predictions%20made.

Marr, B. (n.d.). *28 Best Quotes About Artificial Intelligence*. https://www.forbes.com/sites/bernardmarr/2017/07/25/28-best-quotes-about-artificial-intelligence/?sh=5864c0be4a6f

Ng, A. (n.d.). *Association Rules and the Apriori Algorithm: A Tutorial*. https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html

Oppermann, A. (n.d.). *Think You Don’t Need Loss Functions in Deep Learning? Think Again.* https://builtin.com/machine-learning/loss-functions

oreilly. (n.d.-a).

*Completeness score*. https://www.oreilly.com/library/view/mastering-machine-learning/9781788621113/f4a7b48b-0021-4917-ad0d-e89cef3b4df6.xhtml#:~:text=Completeness%20score.%20This%20score%20is%20complementary%20to%20the,the%20same%20true%20label%20to%20the%20same%20cluster.

oreilly. (n.d.-b). *Homogeneity Score*. https://www.oreilly.com/library/view/mastering-machine-learning/9781788621113/2830f738-c6a5-460a-b518-23ecd3745c2d.xhtml#:~:text=Homogeneity%20score.%20This%20score%20is%20useful%20to%20check,1%2C%20with%20low%20values%20indicating%20a%20low%20homogeneity.

Pandey, P. (n.d.). *Feature Scaling: MinMax, Standard and Robust Scaler*. https://machinelearninggeek.com/feature-scaling-minmax-standard-and-robust-scaler

Peleton Technology. (2022, March 6). *What are the Most In-demand Skills in Artificial Intelligence?* https://peletontechnology.com/what-are-the-most-in-demand-skills-in-artificial-intelligence/

Rengarajan, K., & Menon, V. K. (n.d.). *Generalizing Streaming Pipeline Design for Big Data*. 149–160. https://doi.org/10.1007/978-981-15-1366-4\_12

ResearchAndMarket. (2020, February 10). *$390+ Billion Artificial Intelligence (AI) Industry Outlook, 2025—Big Data Fueling AI & Machine Learning Profoundly, AI to Counter Unmet Clinical Demand*. https://www.prnewswire.com/news-releases/390-billion-artificial-intelligence-ai-industry-outlook-2025---big-data-fueling-ai--machine-learning-profoundly-ai-to-counter-unmet-clinical-demand-301001823.html

Sadawi, N. (n.d.). Model Evaluation Classification. *Evaluating Classifiers: Understaning the Confusion Matrix*. https://www.youtube.com/watch?v=zrr4eu29uww

scikit-learn. (n.d.-a). *Calinski and Harabasz score*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski\_harabasz\_score.html?highlight=calinski\_harabasz\_score

scikit-learn. (n.d.-b). *Classifier: LogisticRegression*. https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html

scikit-learn. (n.d.-c). *Classifier: RandomForestClassifier*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforestclassifier#sklearn.ensemble.RandomForestClassifier

scikit-learn. (n.d.-d). *Cluster: Agglomerative*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html?highlight=agglomerativeclustering#sklearn.cluster.AgglomerativeClustering

scikit-learn. (n.d.-e). *Cluster: DBSCAN*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html?highlight=dbscan#sklearn.cluster.DBSCAN

scikit-learn. (n.d.-f). *Cluster: KMeans*. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html?highlight=kmeans#sklearn.cluster.KMeans

scikit-learn. (n.d.-g). *Completeness Score*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness\_score.html

scikit-learn. (n.d.-h). *Feature Selection*. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature\_selection

scikit-learn. (n.d.-i). *Feature Selection: Discriminant Analysis*. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.discriminant\_analysis

scikit-learn. (n.d.-j). *Feature Selection: KernelPCA - Definition*. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html#sklearn.decomposition.KernelPCA

scikit-learn. (n.d.-k). *Feature Selection: Matrix Decomposition*. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

scikit-learn. (n.d.-l). Feature Selection: Matrix Decomposition Usage. *Feature Selection: Matrix Decomposition Usage*. https://scikit-learn.org/stable/modules/decomposition.html#fa

scikit-learn. (n.d.-m). *Feature Selection: PCA - Definition*. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA

scikit-learn. (n.d.-n). *Feature Selection: SelectKBest*. https://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.SelectKBest.html?highlight=selectkbest#sklearn.feature\_selection.SelectKBest

scikit-learn. (n.d.-o). *Hyper-parameter Optimizers*. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model\_selection

scikit-learn. (n.d.-p). Imputer Types. *Imputer Types*. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.impute

scikit-learn. (n.d.-q). *Metrics: Classification*. https://scikit-learn.org/stable/modules/model\_evaluation.html#classification-metrics

scikit-learn. (n.d.-r). *Metrics: Cluster*. https://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation

scikit-learn. (n.d.-s). *Metrics: Regression*. https://scikit-learn.org/stable/modules/model\_evaluation.html#regression-metrics

scikit-learn. (n.d.-t). *Regression: Linear Regression*. https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html?highlight=linear#sklearn.linear\_model.LinearRegression

scikit-learn. (n.d.-u). *Regression: PolynomialFeatures*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html?highlight=polynomialfeatures

scikit-learn. (n.d.-v). *Regression RandomForestRegressor*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=randomforestregressor#sklearn.ensemble.RandomForestRegressor

scikit-learn. (n.d.-w). *Regression: SVR*. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html?highlight=svr#sklearn.svm.SVR

scikit-learn. (n.d.-x). *V-measure Score*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v\_measure\_score.html#:~:text=The%20V-measure%20is%20the%20harmonic%20mean%20between%20homogeneity,won%E2%80%99t%20change%20the%20score%20value%20in%20any%20way.

seedscientific. (2021, October 28). How Much Data Is Created Every Day? [27 Staggering Stats]. *How Much Data Is Created Every Day? [27 Staggering Stats]*. https://seedscientific.com/how-much-data-is-created-every-day/

Selig, J. (2022, March 14). What Is Machine Learning? A Definition. *What Is Machine Learning? A Definition.* https://www.expert.ai/blog/machine-learning-definition/

Singh, S. P. (n.d.). NLP: Porter Stemmer Algorithm. *Porter Stemmer Algorithm*. https://iq.opengenus.org/porter-stemmer/

Spray, J. (2022, March 30). *Abstraction Layered Architecture*. http://abstractionlayeredarchitecture.com/#truelayers-instead-of-hierarchical-decomposition

Statista Research Department. (2016, November 27). *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025*. https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

SuperDataScience Team. (2018a, 28). The Ultimate Guide to Convolutional Neural Networks (CNN). *The Ultimate Guide to Convolutional Neural Networks (CNN)*. https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn

SuperDataScience Team. (2018b, August 30). The Ultimate Guide to Recurrent Neural Networks (RNN). *The Ultimate Guide to Recurrent Neural Networks (RNN)*. https://www.superdatascience.com/blogs/the-ultimate-guide-to-recurrent-neural-networks-rnn

SuperDataScience Team. (2018c, August 30). *The Ultimate Guide to Recurrent Neural Networks (RNN)*. https://www.superdatascience.com/blogs/the-ultimate-guide-to-recurrent-neural-networks-rnn

udemy. (n.d.). *Deep Learning: Recurrent Neural Networks in Python*. https://www.udemy.com/course/deep-learning-recurrent-neural-networks-in-python/

wikipedia. (n.d.). Phi coefficient. *MCC*. https://en.wikipedia.org/wiki/Phi\_coefficient

Zuccarelli, E. (2021, February 1). *Performance Metrics in Machine Learning—Part 3: Clustering*. https://towardsdatascience.com/performance-metrics-in-machine-learning-part-3-clustering-d69550662dc6