Student Declaration of Authorship

HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Course code and name:	F20ML Statistical Machine Learning
Type of assessment:	**Individual**
Coursework Title:	Coursework
Student Name:	Lih Woei Siow
Student ID Number:	H00345942

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature *(type your name):* *Lih Woei Siow*

Date: *14/11/2023*

Copy this page and insert it into your coursework file in front of your title page.
For group assessment each group member must sign a separate form and all forms must be included with the group submission.

Your work will not be marked if a signed copy of this form is not included with your submission.

F20ML/F21ML - Coursework – 2023/24

Student: Siow Lih Woei **HW-ID:** H00345942

1.0 Introduction and Brief Description of the Dataset

This project aims to develop many machine-learning algorithms for the purpose of detecting and classifying a dataset containing phishing websites. The programming language utilized for conveying my findings and outcomes will be Python.

The dataset included is the UCI phishing websites dataset, which was developed to classify websites into phishing and non-phishing categories. Phishing websites are categorized as malicious online platforms that aim to illegally collect sensitive data from individuals, sometimes by masquerading as legitimate websites.

The data is stored in the Attribute-Relation File Format, or ARFF. The 'scipy' module was used to import the ARFF file into a Pandas DataFrame for data manipulation and analysis in the Python programming language. Following the import, the data was prepped for machine learning algorithms by applying typical pre-processing techniques.

2.0 Exploratory Data Analysis (EDA) Summary

The dataset contains data as 'objects,' which are represented as strings rather than numerical values. Consequently, the computation of basic statistics for numerical features could be more feasible due to their unsuitability in format. Consequently, to modify the data types of numerical features to 'int64', the code .astype(int) must be utilized. Descriptions of the data set and analytical figures are provided below:

Size of Data Set: 11055

Mean / Standard Deviation of Numerical Features:

	having IP Address	URL Length	Shortening Service	having At Symbol	double slash redirecting
Mean	0.314	-0.633	0.739	0.701	0.741
Standard Deviation	0.950	0.766	0.674	0.714	0.671

	Prefix Suffix	having Sub Domain	SSLfinal State	Domain registration length	Favicon
Mean	-0.735	0.064	0.251	-0.337	0.629
Standard Deviation	0.678	0.818	0.912	0.942	0.778

	Port	HTTPS token	Request URL	URL of Anchor	Links in tags
Mean	0.728	0.675	0.187	-0.077	-0.118
Standard Deviation	0.685	0.738	0.982	0.715	0.764

	SFH	Submitting to email	Abnormal URL	Redirect	On mouseover
Mean	-0.596	0.636	0.705	0.116	0.762
Standard Deviation	0.759	0.772	0.709	0.320	0.647

	Right Click	PopUp Window	Iframe	Age of Domain	DNS Record
Mean	0.914	0.613	0.817	0.061	0.377
Standard Deviation	0.406	0.790	0.577	0.998	0.926

	Web Traffic	Page Rank	Google Index	Links pointing to page	Statistical Report
Mean	0.287	-0.484	0.722	0.344	0.720
Standard Deviation	0.828	0.875	0.692	0.570	0.694

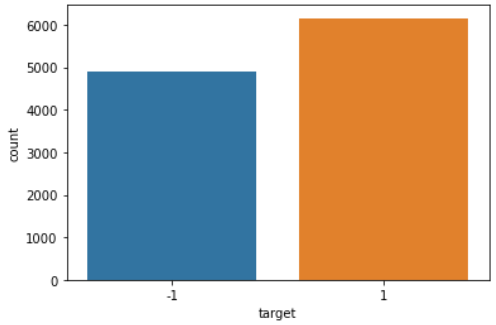


Figure 1: Histogram for Target (y)

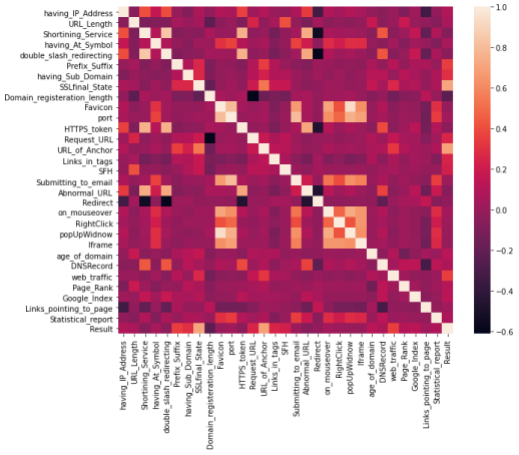


Figure 2: Heatmap of the Correlation matrix of features

3.0 Perceptron Implementation

Perceptron is a supervised machine-learning approach for binary classifiers. Perceptron identifies patterns or calculations in incoming data. Thus, create a perceptron algorithm from scratch to classify dataset phishing websites. The following functions are implemented in the perceptron:

__init__: The constructor for the Perceptron class, responsible for initializing the Perceptron object and storing its hyperparameters

predict: Utilized the trained Perceptron to generate predictions on the test dataset by applying the function, $\left(input_{feature} \cdot w_d\right) + bias$

fit: Trained the Perceptron using the training data from the dataset and adjusts weights and bias using the perceptron learning rule, $w_d \pm learning\ rate \times (target - prediction) \times input\_feature$, to alter decision boundaries if the prediction is incorrect

***get_params*:** Retrieve perceptron parameters

***set_params*:** Sets perceptron parameters from input dictionary 'params'

When trained with the default hyperparameters (namely, a learning rate of 0.0001 and a maximum iteration of 1000), the Perceptron model achieved an accuracy of 88.74% on the given dataset. The accuracy metric represents the proportion of accurately categorized instances within the sample.

4.0 Finetune Perceptron 1

To optimize the perceptron's performance, it is necessary to fine-tune the algorithm's hyperparameters. Furthermore, splitting the dataset into several subsets, namely the training, validation, and test sets, is possible. This division allows for evaluating and comparing various hyperparameter combinations in terms of their performance. The objective was to identify the most effective combination of learning rate (learning_rate) and a maximum number of iterations (max_iter) to maximize accuracy on a validation dataset.

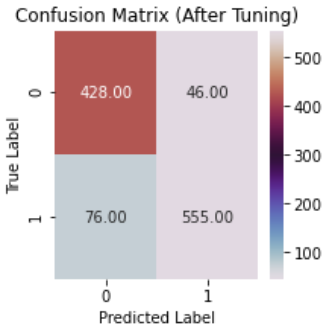
The dataset has been split into three sets: training (80%), validation (10%), and test (10%), implementing the custom function ***train_val_test_split***. Nestled loops are used to go over various combinations and fine-tune the hyperparameters. The final Perceptron model is trained to utilize the ideal hyperparameters (learning_rate = 0.0001 and max_iter = 100), generating a classification accuracy of 90.86% on the test dataset. This statement provides a concise overview of how the selected hyperparameters contribute to optimizing the prediction capabilities of the model.

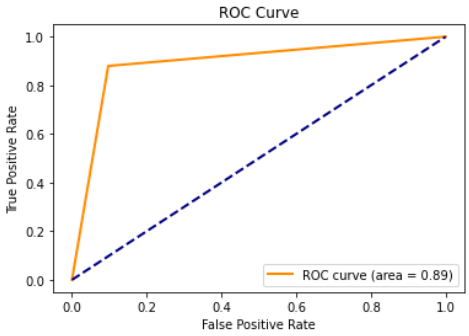
5.0 Finetune Perceptron 2

Instead of using the training, validation, and test split technique, we use 10-fold cross-validation to fine-tune the perceptron algorithms. Cross-validation is essential for accurately testing a model's performance on a dataset other than the training dataset. It was possible to do a full grid search with cross-validation by using the ***GridSearchCV*** method. The parameter grid from the previous section was used to test all hyperparameter combinations. Using the maximum iteration of 150 and the ideal learning rate of 0.001 with its best result of 91.18%, the learning Perceptron's performance was assessed utilizing evaluation metrics.

A confusion matrix including true positive, false positive, true negative, and false negative data was created from the best findings after each fold. The model's accuracy shows that the specified hyperparameters improve prediction. The F-score, a test accuracy statistic, was calculated to balance precision and recall. The Receiver Operating Characteristic (ROC) curve shows the model's ability to distinguish positive and negative classes, revealing its prediction performance. Evaluation metrics are shown below. Following hyperparameter tuning, we can see that the accuracy is the same as the initial perceptron. Optimizing the Perceptron model for binary classification prepares it for reliable real-world predictions, but it might not work for this algorithm.

Accuracy	0.89
F1 Score	0.89
ROC AUC	0.89

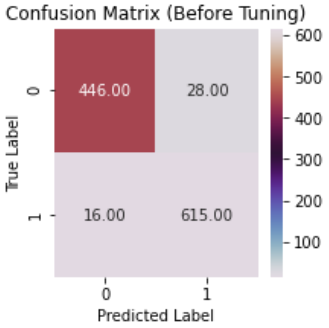


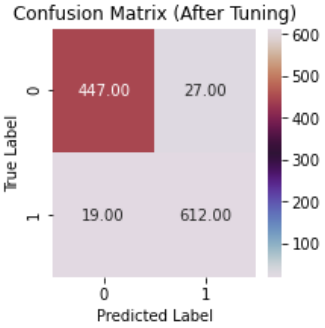


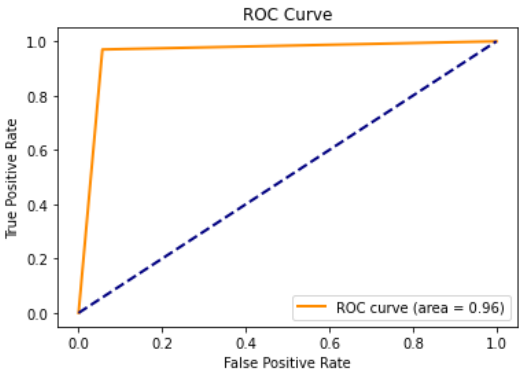
6.0 Decision Tree

Decision Trees (DT) are a non-parametric supervised learning method that may be used for classification and regression applications. The goal is to create a prediction model that uses information to infer basic decision rules that determine the target variable value. The first root node represents the complete collection. After that, internal nodes represent traits and decision rules. Finally, the leaf node shows class labels.

A Decision Tree Classifier object (DT) was developed to train the dataset and forecast using input characteristics and labels. We will tune hyperparameters to maximize classifier performance. We will focus on two hyperparameters: the tree's maximum depth (max_depth) and each leaf's minimum sample count. To do this, we will employ GridSearchCV, a technique that combines grid search with cross-validation. In our case, we will execute 10-fold cross-validation to evaluate the different combinations of hyperparameters. After fine-tuning hyperparameters to the best (max_depth = 26, min_samples_leaf = 1), our performance results are below.







	Before	After
Accuracy	0.96	0.96
F1 Score	0.96	0.96
ROC AUC	0.96	0.96

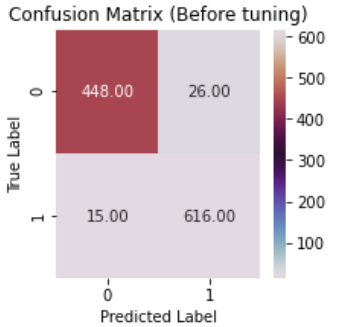
The confusion matrix and evaluation metrics demonstrate that the model does well, with high true negatives and true positives and 96% performance across all numbers. However, finetuning hyperparameters may not be a valid or useful way to make the Decision Tree algorithm work better because the result from evaluation metrics is the same after finetuning as it was before.

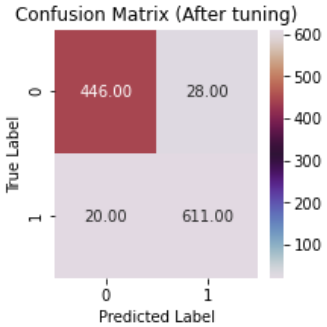
7.0 Neural Network

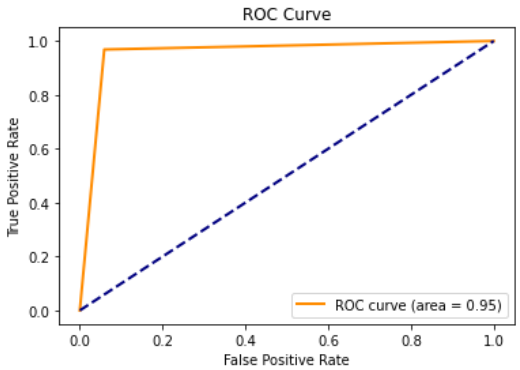
A type of machine learning model called Neural Network is made up of a node layer with an input layer, one or more hidden layers, and an output layer. Using training data, neural networks learn and get better over time.

The Multi-layer Perceptron (MLP) Classifier object sorts things into groups based on a Neural Network that is running underneath. In the same way, we tested and trained our model and then used 10-fold cross-validation to finetune three hyperparameters: the strength of the L2 regularisation term (alpha), the schedule for weight updates (learning_rate_init), and the activation function for the hidden layer. This improved the model's ability to learn this classification. After using GridSearchCV to do 10-fold cross-validation and finetune with the best hyperparameters (learning_rate_init = 0.001, alpha = 0.0001, activation = "tanh"), the results are shown below.

The model doesn't predict well after finetuning, with its confusion matrix showing a true negative and true positive decrease and no evaluation metrics increase. There is no improvement in learning ability for neural networks through hyperparameter tuning, which means that hyperparameter tuning doesn't work as well as expected to increase the accuracy of the algorithm prediction.







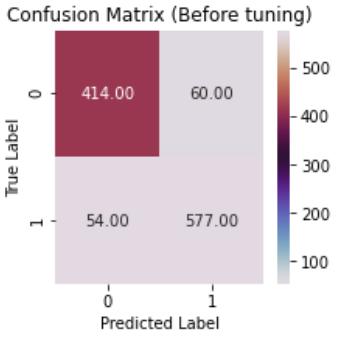
	Before	After
Accuracy	0.96	0.96
F1 Score	0.96	0.96
ROC AUC	0.96	0.96

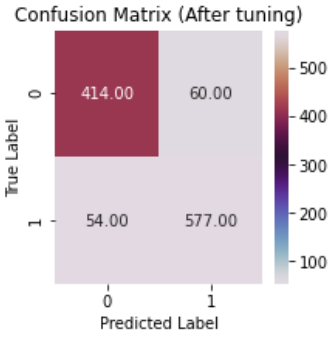
8.0 Naïve Bayes

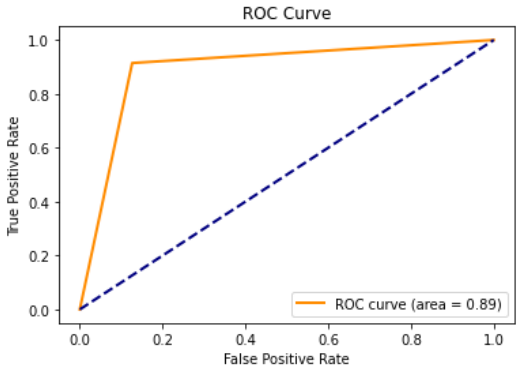
Naïve Bayes is a classification technique based on the Bayes theorem:

$$P(C|A) = \frac{P(A \cap C)}{P(A)} = \frac{P(A|C)P(C)}{P(A)}$$

Bernoulli Naïve Bayes (BernuolliNB) will be used for the classification job because the features and target names are made up of discrete values. The same method we used for past algorithms was used to test and train the model. Then, use 10-fold cross-validation and hyperparameter tuning with GridSearchCV to enhance learning stronger. The smoothing parameter, which considers features and stops null probabilities from showing up in the probability estimate (alpha), is the best hyperparameter to finetune. With the best hyperparameters (alpha = 0.0001), the best BernuolliNB classifier was trained. The findings are presented in the next section.







	Before	After
Accuracy	0.90	0.90
F1 Score	0.90	0.90
ROC AUC	0.89	0.89

The model does a good job, as shown by the confusion matrix, which shows most of the true negatives and true positives. We try to finetune Bernoulli Naïve Bayes' hyperparameter alpha and get the best result, which is 0.0001. The model did not perform better as our results were the same after finetuning as they were before. This could mean that changing hyperparameters is not a true or useful way to make the Naïve Bayes classifier work better.

9.0 K-Nearest Neighbours

The K-Nearest Neighbours (KNN) method is a widely utilized machine learning technique employed for both classification and regression problems. Its underlying

principle is based on the idea that data points that are similar in nature are likely to possess comparable labels or values. To determine the nearest neighbours to a given input data point based on their distances, it is necessary to implement a K-Nearest Neighbours (KNN) classifier from scratch. Details of KNN implementation functions:

__init__: The constructor method initializes with the parameter 'k,' which represents the number of neighbours to consider during classification

fit: Alignment or compatibility between different elements or components. Using the training data from the dataset and memorizing for future predictions

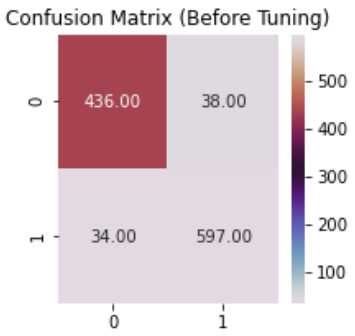
predict: Utilized the trained KNN algorithm to make predictions on the test dataset. It returns the predicted most common class among these neighbour instances, 'x'

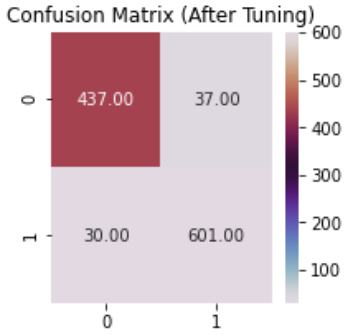
***get_params*:** Get the parameters of the perceptron

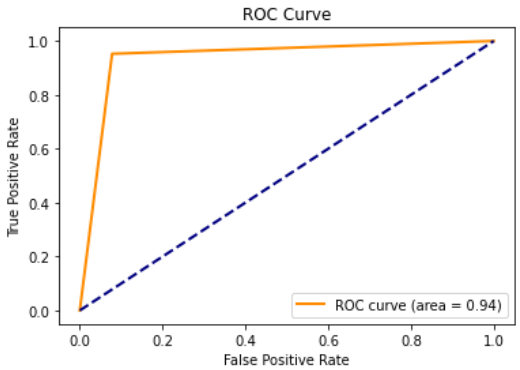
***set_params*:** Sets the parameters of the perceptron from the input dictionary 'params'.

We used a default hyperparameter to test and train the KNN Classifier. We will finetune the only hyperparameter in the Classifier above, k, to make it better at learning. We use GridSearchCV for 10-fold cross-validation to find the most accurate hyperparameters. The evaluation measures' outcome is shown below after the hyperparameter was fine-tuned (k=3).

The general performance of a classification model has gotten better, as shown by the confusion matrix and evaluation measures, which all show a 1% rise. Since True Positive (TP) and True Negative (TN) are going up, the model shows that it makes fewer mistakes in both groups.







	Before	After
Accuracy	0.93	0.94
F1 Score	0.93	0.94
ROC AUC	0.93	0.94

10.0 Comparisons on the performance of the algorithms

After evaluating the algorithms individually, we will be comparing all the algorithms.

	Perceptron	Decision Tree	Neural Network	Naïve Bayes	KNN
Accuracy	0.89	0.96	0.96	0.90	0.94
F1 Score	0.89	0.96	0.96	0.90	0.94
ROC AUC	0.89	0.96	0.96	0.89	0.94

The confusion matrix's highest True Negatives (TN) among all algorithms suggests that the model accurately detects a sizable percentage of negative instances, which is consistent with both the model's predictions and the real data.

The provided table illustrates that all the algorithms achieve accuracy, F1 score, and ROC values over 80%. However, the perceptron algorithm presents the lowest results compared to the other algorithms, with Naïve Bayes following behind. Neural networks have higher accuracy relative to other models. Nevertheless, the effectiveness of all the algorithms does not exhibit significant improvement even after undergoing hyperparameter adjustment using 10-fold cross-validation. This observation suggests that the algorithms mentioned earlier demonstrate effective solutions to the multiclass problem without necessitating the need for feature scaling.

The receiver operating characteristic (ROC) curve illustrates the classification model's performance across various categorization criteria. In general, all the Area Under the Curve (AUC) values exceed 0.8. Moreover, the receiver operating characteristic (ROC) curve exhibits a displacement towards the (0.0, 1.0) coordinate, representing the ideal learning curve. This observation demonstrates that the algorithms exhibit a consistently high level of accuracy in their predictive capabilities, regardless of the chosen classification threshold. The Receiver Operating Characteristic (ROC) curve analysis of the Perceptron algorithm suggests that its predictive performance may be poorer than that of other algorithms.

11.0 Conclusion

In this study, a variety of machine learning methods were employed to categorize a dataset consisting of phishing websites. In our simulation of multiclass classification, the Neural Network demonstrates the highest level of effectiveness. Nevertheless, following the refinement of hyperparameters using a 10-fold cross-validation process for all machine learning algorithms, the differences that were observed between the pre-finetuning and post-finetuning stages are minimal. This could be because the optimal hyperparameters have already been selected or because the data was initially overfit or underfit, rendering the hyperparameter tuning insufficient to resolve the underlying issue. Therefore, in substitution of the hyperparameter approach, alternative strategies for enhancing model performance might be explored.

Reference List

1. *Using Pandas, CSV and ARFF files — pymfe 0.4.2 documentation* (no date). Available at: https://pymfe.readthedocs.io/en/latest/auto_examples/03_miscellaneous_examples/plot_using_pandas_csv_arff.html (Accessed: November 11, 2023).
2. Mahadevan, M. (2023) *Step-by-Step Exploratory Data Analysis (EDA) using Python*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2022/07/step-by-step-exploratory-data-analysis-eda-using-python/ (Accessed: November 11, 2023).
3. *Perceptron in Machine Learning - Javatpoint* (no date) *www.javatpoint.com*. Available at: https://www.javatpoint.com/perceptron-in-machine-learning (Accessed: November 11, 2023).
4. Galarnyk, M. (2022) *Understanding Train Test Split*, *Built In*. Available at: https://builtin.com/data-science/train-test-split (Accessed: November 11, 2023).
5. *How to split data into training and testing in Python without sklearn* (2023) *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/how-to-split-data-into-training-and-testing-in-python-without-sklearn/ (Accessed: November 11, 2023).
6. *An Introduction to GridSearchCV | What is Grid Search | Great Learning* (2023) *Great Learning Blog: Free Resources what Matters to shape your Career!* Available at: https://www.mygreatlearning.com/blog/gridsearchcv/ (Accessed: November 11, 2023).
7. Chouinard, J.-C. (2023) *Confusion Matrix in Python (Scikit-learn Example)*, *JC Chouinard*. Available at: https://www.jcchouinard.com/confusion-matrix-in-scikit-learn/ (Accessed: November 11, 2023).
8. Blog, D. C. (2023) *Understanding F1 Score, Accuracy, ROC-AUC, and PR-AUC Metrics for Models*, *Deepchecks*. Available at: https://deepchecks.com/f1-score-accuracy-roc-auc-and-pr-auc-metrics-for-models/ (Accessed: November 11, 2023).
9. Rojas, M. G., Olivera, A. C. and Vidal, P. J. (2022) *Optimising Multilayer Perceptron weights and biases through a Cellular Genetic Algorithm for medical data classification*, *Array*. Elsevier BV. doi: 10.1016/j.array.2022.100173.
10. *Decision Trees* (no date) *scikit-learn*. Available at: https://scikit-learn.org/stable/modules/tree.html (Accessed: November 11, 2023).

11. *sklearn.tree.DecisionTreeClassifier* (no date) *scikit-learn*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html (Accessed: November 11, 2023).
12. *What are Neural Networks?* | *IBM* (no date). Available at: https://www.ibm.com/topics/neural-networks (Accessed: November 11, 2023).
13. *sklearn.neural_network.MLPClassifier* (no date) *scikit-learn*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (Accessed: November 11, 2023).
14. Akshaysharma (2021) *Naive Bayes with Hyperpameter Tuning*, *Kaggle*. Available at: https://www.kaggle.com/code/akshaysharma001/naive-bayes-with-hyperpameter-tuning (Accessed: November 11, 2023).
15. *K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint* (no date) *www.javatpoint.com*. Available at: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning (Accessed: November 11, 2023).
16. *Python Machine Learning - K-nearest neighbors (KNN)* (no date). Available at: https://www.w3schools.com/python/python_ml_knn.asp (Accessed: November 11, 2023).
17. Devos, A. *et al.* (2007) *Classification of Brain Tumours by Pattern Recognition of Magnetic Resonance Imaging and Spectroscopic Data*, *Elsevier eBooks*. doi: 10.1016/b978-044452855-1/50013-1. (Accessed: November 11, 2023).
18. *How to explain the ROC AUC score and ROC curve?* (no date). Available at: https://www.evidentlyai.com/classification-metrics/explain-roc-curve#:~:text=ROC%20AUC%20score%20shows%20how,have%20an%20AUC%20of%200.5. (Accessed: November 11, 2023).
19. *Fine-tuning Models: Hyperparameter Optimization* (no date) *Encord*. Available at: https://encord.com/blog/fine-tuning-models-hyperparameter-optimization/ (Accessed: November 11, 2023).