Paper Title* (use style: paper title)

*Note: Sub-titles are not captured in Xplore and should not be used

line 1: 1st Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 2nd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 3rd Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 4th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 5th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

line 1: 6th Given Name Surname
line 2: *dept. name of organization (of Affiliation)*
line 3: *name of organization (of Affiliation)*
line 4: City, Country
line 5: email address or ORCID

***Abstract*—To be done**

Keywords—to be done

I. INTRODUCTION (*HEADING 1*)

To be done

II. DATA PREPROCESSING

The majority of the datasets available are called Raw Datasets. The reason they are labelled raw datasets is that they contain many anomalies which further implies that, the dataset is not suitable to train our Machine Learning and Deep Learning Models. The dataset needs to be nurtured and structured via a series of steps commonly called the Data Preprocessing Steps, to make it suitable for the training purpose of the proposed models.

The anomalies associated with the Raw Dataset are listed as follows –

1) *Missing Values*
2) *Skewness*
3) *Kurtosis*
4) *Outliers*
5) *Feature Selection or Data Reduction*

Once the dataset is Preprocessed, it improves the proposed models' overall accuracy, precision, recall and F1-Score.

A. Missing Values

In the dataset, some values might be missing from some or all of the attributes. These missing values are represented with a 'Blank', 'null' and 'NaN' (Not a Number). The standard procedures for the effective removal/replacement of the missing values are listed below –

1) *Drop down the entire row column containing at least one missing value*
2) *Replace the missing value(s) with the median of that attribute if applicable*
3) *Replace the missing value(s) with the mean of that attribute if applicable*
4) *Replace the missing value(s) with the Forward Fill technique*
5) *Replace the missing value(s) with the Backward Fill technique*

The forward Fill technique puts in the last known value before the first missing cell of that particular attribute, and each missing cell of that particular attribute is then replaced by the same last known value. Think of it as, surfing through each row from top to bottom, identifying the last known cell value before the missing cell and then replacing all the missing cells with that value.

The Backward Fill technique puts in the last known value before the first missing cell of that particular attribute while going backwards, and each missing cell of that particular attribute is then replaced by the same last known value. Think of it as, surfing through each row from bottom to top, identifying the last known cell value before the missing cell and then replacing all the missing cells with that value.

B. Skewness

Before knowing what skewness is, the concept of Normal Distribution is important. Normal Distribution or commonly termed a Gaussian Distribution is a type of continuous probability distribution, with a bell-shaped curved. The probability density function can be defined as follows:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2}$$

where, f(x) = probability density function,
σ = standard deviation, and
μ = mean

A typical symmetric Gaussian Distribution looks like follows:

Now, skewness is a measure of how much the Gaussian Distribution is skewed or shifted from its normal position towards left or right and accordingly called Right Skewed or Left Skewed respectively. The Right Skewed value is represented using a positive value and for Left Skewed, it is represented using a negative value. The skewness can be calculated using the following formula:

$$\tilde{\mu}_3 = \frac{\sum_i^N (X_i - \bar{X})^3}{(N-1) * \sigma^3}$$

The industrial accepted skewness value ranges from -1.5 to +1.5. Anything greater than +1.5 or smaller than -1.5 needs to be reduced to fit within the range. For this we can apply the following methods:

1) *Box-Cox Transformation – It can be calculated using the following function*

$$y(\lambda) = \begin{cases} \frac{(y+\lambda_2)^{\lambda_1}-1}{\lambda_1}, & \text{if } \lambda_1 \\ \log(y+\lambda_2), & \text{if } \lambda_1 \end{cases}$$

2) *Cube root Transformation - It can be calculated using the following function*

$$f(x) = \sqrt[3]{x}$$

3) *Log Transformation - It can be calculated using the following function*

$$f(x) = \log(x)$$

A normally distributed attribute helps to increase the overall accuracy, precision, recall and F1-Score of the proposed models.

C. Kurtosis

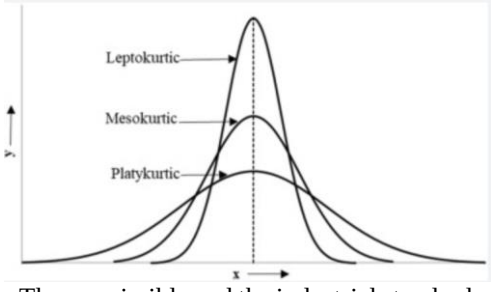
It is a measure of how much the data clusters near the tails or the head of the curve. The Kurtosis value can be obtained using the stated equation:

$$\text{Kurtosis} = \frac{\sum(x_i - \bar{x})^4}{nS^4}$$

$\bar{x}$ = mean of the given data
S = standard deviation of the data
n = total number of observations

Based on this, there are three types of kurtosis:

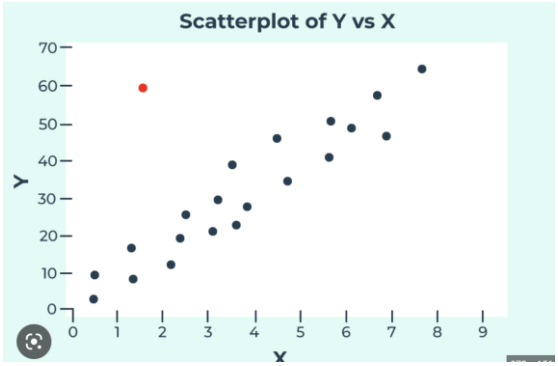
1) *Mesokurtic – The data here is moderately spread, with a medium peak.*
2) *Leptokurtic – The majority of the data is concentrated near the tails, leading to a pointed peak.*
3) *Platykurtic – The data is evenly spread-out, implying a flat line almost with a little bump.*



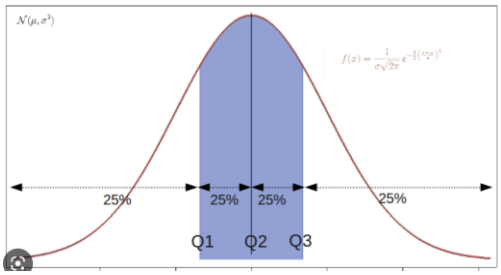
The permissible and the industrial standard accepted value is 3. Anything less than 3 or greater than 3 is not permissible.

D. Outliers

To put it in simple words, Outlier means the odd one out. From Fig.XX it is evident that the marked points have too much deviated from the rest of the pack. These points are labelled as Outliers. The best method/practice to find out the outlier of the attribute is the visual representation. From the distribution plot, the outliers can be identified easily.



It is essential to remove or replace those outliers because they can interfere with the efficiency, accuracy, precision, recall and F1-Score of the proposed models. The industry-accepted and effective method to remove or replace those points is called Interquartile Range Function (IQR).



IQR is a statistical dispersion that represents the range from the first quartile (Q_1) to the third quartile (Q_3), implying covering 50% of the distribution area. The IQR is defined as:

$$IQR = Q_3 - Q_1$$

The values lying within the IQR are considered to be in the safe zone. Any points not within the range are either dropped, if the dataset is very large, or the points are replaced by the median, which is a safer option and works well for large and relatively small datasets.

E. Feature Selection or Data Reduction

A dataset comprises several attributes which are responsible for obtaining vital information to learn and understand the pattern and relationship between the input and output values. However, not all attributes contribute the same amount of information. Dropping down those specific attributes does not harm the goal of seeking the relationship between input and output variables and also reduces the training time of our proposed ML and DL models.

There are various techniques for feature selection, including:

1) *Filter methods involve selecting features based on statistical tests such as correlation, chi-square, or mutual information.*
2) *Wrapper methods: This involves selecting features based on the performance of a machine learning model using different subsets of features.*
3) *Embedded methods: This involves selecting features during the training process of a machine learning model, such as regularization techniques.*
4) *Dimensionality reduction: This involves transforming the original dataset into a lower-dimensional space, such as principal component analysis (PCA) or linear discriminant analysis (LDA).*

When selecting features, it is important to consider the trade-off between the accuracy and complexity of the model, as well as the interpretability of the features. Additionally, it is important to use a validation set or cross-validation to ensure that the selected features generalize well to new data.

Principal Component Analysis or PCA in short is a widely used statistical technique used to reduce the dimensionality of a dataset. Dimension Reduction is a very important problem in machine learning as for smaller datasets the number of attributes can be higher than the number of instances which can lead to overfitting. PCA can be used to solve this problem by transforming the dataset into a different coordinate system where the dataset can be represented in a lower dimensional space.

PCA works by representing the original dataset in a new coordinate system where each dimension is a linear combination of the original attributes. This new coordinate system is selected in such a way that the first component captures the maximum variance in the dataset, the second component captures the maximum variance that is orthogonal to the first component and so on.

To perform PCA we first standardize the data points by subtracting the mean and divide by the standard deviation. We then compute the covariance matrix of the dataset which measures the relative variance of the attributes of the dataset.

Next we compute eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the directions of the original feature space along which the variance of the data points are the highest, the eigenvalues represent how much the data points vary.

Finally we select the highest N eigenvalues and the corresponding eigenvectors and use them as the basis of the new coordinate system. We then project the data along the new coordinate system to obtain the lower dimensional representation of the dataset.

The algorithm for computing PCA can be represented as:

1) *Standardize the data by subtracting mean and dividing by standard deviation.*

$$X_{STD} = \frac{(X - X_{Mean})}{Standard\ Deviation\ (X)}$$

$$X = Attributes$$

2) *Compute the covariance matrix of the standardized dataset.*
3) *Compute the eigenvalues and eigenvectors of the covariance matrix.*
4) *Sort the eigenvalues and eigenvectors according the eigenvalues in descending order*
5) *Select N largest eigenvalues and corresponding eigenvectors and use them as the basis for new coordinate system to represent the dataset in a lower dimensional feature space.*

III. Machine Learning Models

A. K-Nearest Neighbors Classifier

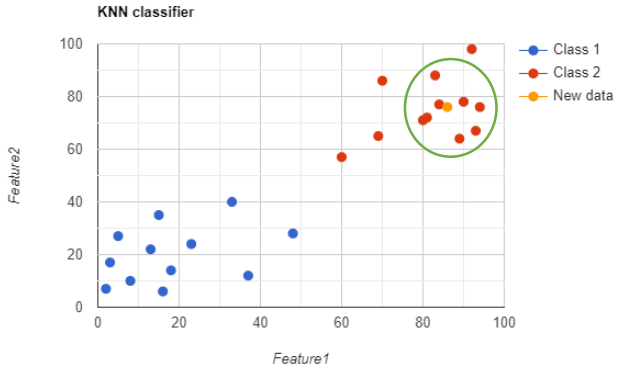
K-nearest neighbours classifier is a supervised learning technique which is non-parametric i.e. it makes no assumptions about the underlying distributions of the data. The working principle of the algorithm is the prediction of the classes based on some similarity measures (generally Euclidian distance). The new data point is classified by observing the k nearest data points around it and the most common class among the k neighbours is assigned the class to the new data point.

Euclidian distance is the measure of distance between two points in a Cartesian coordinate system. For instance, let us assume two data points M and N having coordinates (x_m, y_m) and (x_n, y_n) respectively. Then the Euclidian distance between the points can be calculated as:

$$D(M, N) = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}$$

For classification using the knn algorithm, the dataset is first split into the training and testing part. Then for each new data point, the k nearest neighbours are considered and assigned the most common class among the neighbours.

The number of neighbours i.e. K is the hyperparameter that needs to be adjusted carefully to avoid problems like overfitting or underfitting.



B. Support Vector Classifier

Support vector classifier (SVC) is a classification technique used for supervised learning problems. SVC finds out the optimal separating hyperplane also called the decision boundary that separates the data points belonging to different classes. The goal of SVC is to maximise the distance between data points close to the margin and the separating hyperplane. The data points closest to the margin are called support vectors and hence the name.

SVC has a few important parameters which need to be tuned carefully for proper results:

1) *c: Regularization parameter with default value 1.0, type float.*

2) *kernels: Out of several choices for the kernels the appropriate one must be chosen (more on kernels later).*

Popular choices for kernels:

1) *Linear: It calculates the dot product of input vectors.*

$$L(x, x') = x.x'$$

2) *Radial Basis Function (RBF): It uses a Gaussian function to map the input vectors to an infinite dimensional feature space.*

$$R(x, x') = e^{-\Upsilon\|x-x'\|^2}$$

3) *Polynomial: It is a non-linear kernel that calculates the similarity between two vectors using the following relation:*

$$P(x, x') = (\Upsilon * (x.x') + coef_0)^d$$

4) *Sigmoid: It uses a tan hyperbolic function to compute the similarity between the inputs. It is a non-linear kernel.*

$$S(x, x') = \tanh(\Upsilon * (x.x') + coef_0)$$

Where x,x' are the input vectors, Υ is the hyperparameter controlling the width of the Gaussian kernel, d is the hyperparameter degree and $coef_0$ is the coefficient hyperparameter.

The Support Vector machine uses hinge loss as a loss function. The goal of SVM is to minimise hinge loss. The loss function is given as:

$$l = \max(0, 1 - y^i(x^i - b))$$

Where y^i and x^i are the i^{th} instance in the training set and b is the bias.

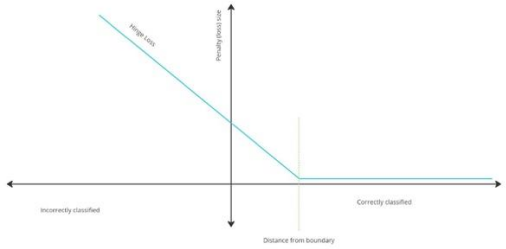
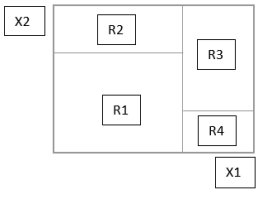


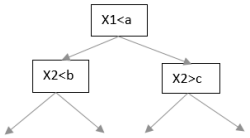
Figure showing the graph of hinge loss.

C. Decision Tree Classifier

The decision tree classifier is a supervised learning technique used for the classification of input. It works by continuously separating the input space into several segments and making decisions based on certain criteria. This algorithm can be easily visualised as a tree of decisions which has consequences at each decision node.



Segment of input database.



Tree like structure of decisions

After building the tree, sometimes it tends to take a very complex form which has very high computational complexity. In order to reduce the computational load the tree is pruned which also avoids overfitting.

The decision tree algorithm uses various metrics to select the feature for the best split. Some of the commonly used metrics are:

1) *Information Gain:*

$$Gain(D, A) = Info(D) - Info(D|A)$$

$$Info(D) = -\sum_{i=1}^{c}(p_i * log_2 p_i)$$

$$Info(D|A) = \sum_{i=1}^{c}\frac{|D_i|}{|D|} * Info(D_i)$$

2) *Gini Index:*

$$Gini(D) = 1 - \sum_{i=1}^{c} {p_i}^2$$

Where D is the instance of the dataset, c is the number of possible classes and p_i is the probability that D_i belongs to class i.

D. Random Forest Classifier

Random forest classifier is a bit different classification algorithm. It is an ensemble learning technique which uses multiple decision trees to generate its output (a group of trees is called a forest).

Bagging of these individual trees is done to reduce the correlation between trees. This algorithm creates multiple decision trees where each tree constructed uses a different subset of features, selected with replacement also called Bootstrapping.

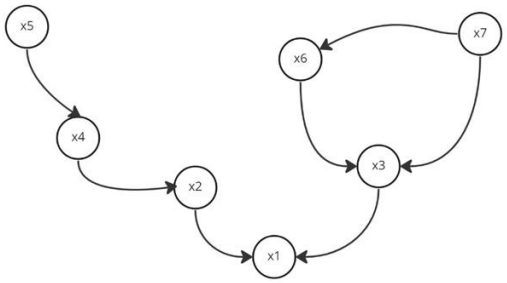
After all, the trees are constructed and output generated, the final output is obtained by taking the majority voting or the mean of all predictions. A random forest works similarly to the decision tree and hence is mathematically identical. However, prediction is generated as follows:

$$\text{Final prediction} = \text{mode}(\text{Prediction of individual trees})$$

E. Naïve Bayes Classifier

Naïve Bayes classifier is a probabilistic machine learning algorithm used for classification. The backbone for this algorithm is the Bayesian network, also known as the Bayes network or Belief network.

A Bayesian network is a directed probability graph which depicts the conditional dependency of the random variables on one another. The Bayesian network has to be an acyclic graph.



Let us consider the above Bayesian network. This network shows that x_5 is an independent variable and x_4 depends on x_5 which is shown by an arrow from a random variable to the dependent variable. The joint probability distribution of the network can be factorized by using the

chain rule. The factorization of the above network can be observed to be:

$$P = p(x_1|x_2, x_3).\, p(x_3|x_6, x_7).\, p(x_2|x_4).\, p(x_4|x_5).\, p(x_5).\, p(x_6|x_7).\, p(x_7)$$

From this graph we can also say, knowing x_2 (say) separates x_1 and x_4 i.e. x_1 and x_4 are conditionally independent if x_2 is known.

The Naïve Bayes classification algorithm uses the principle of Bayes theorem assuming that all the features are independent of each other and hence the name. The model calculates the probability of an instance belonging to a class based on the features associated with it. This is done by calculating the conditional probability of each feature given the class and then calculating the total probability using the Bayes theorem.

The instance having the highest probability of belonging to a class is classified to be a point in that class.

Conditional probability is calculated using Bayes theorem as follows:

$$P(C|X_1, X_2, X_3, \dots, X_n) = \frac{P(C)P(X_1|C)P(X_2|C)P(X_3|C) \dots P(X_n|C)}{P(X_1, X_2, X_3, \dots, X_n)}$$

Naive Bayes has different variations and can be used for both binary and multi-class classification.

F. Logisitic Regression

Logistic regression is a classification algorithm for binary class problems using a statistical approach. This algorithm works very well where the prediction of the probability of events is involved. The logistic regression algorithm uses a logit transformation of the probability as follows:

$$\log\left(\frac{p(x)}{1 - p(x)}\right) = \beta_0 + \beta x$$

Solving which gives us:

$$p(x) = \frac{1}{1 - e^{-(\beta_0 + \beta x)}}$$

Which is nothing but a sigmoid.

Where β_0 and β are the weights and $p(x)$ is the probability of instance x of the dataset.

Logistic regression uses the maximum likelihood estimation method to estimate the parameters of the linear combination and maximises the likelihood of the observed data given the model. An iterative algorithm such as gradient descent is used to estimate the parameters.

G. Extreme Gradient Boosting Classifier

Extreme Gradient boosting classifier, popularly known as XGBoost classifier, works using an ensemble of decision trees. This algorithm also uses multiple decision trees but in a sequential manner which is not the case in Random Forest. Here every subsequent tree tries to correct the errors of the previous tree.

The gradient-boosting algorithm is used to optimize an objective function:

$$Obj = loss(actual, predicted) + \Omega(predicted)$$

Where loss is the loss function of the algorithm and Ω is the regularization function used for penalization. Actual and predicted are the vectors of actual labels and predicted labels respectively.

The loss function used can be any of the popular choices such as Mean Squared Error (MSE), log loss, hinge loss etc. The mean squared error can be calculated as:

$$SSE = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

$$MSE = \frac{1}{N} * (SSE)$$

Where y_i is the actual label and $f(x_i)$ is the predicted label of the i^{th} instance of the dataset having N data points.

The regularization parameter can use various methods like L1 or L2 norms. Here Ω using L2 norm is represented as:

$$\Omega(predicted) = \gamma \sum w^2$$

Where 'Y is the strength of regularization and w is the weight of the leaves.

XGBoost also uses pruning to decrease computational complexity and prevent overfitting.

H. Gradient Boosting Classifier

The Gradient Boosting algorithm uses the idea of combining various weak classifiers to make a strong classifier altogether. It iteratively keeps on adding new trees to the ensemble and updates the weights of the training examples based on the errors of the previous trees.

The algorithm first initialises the ensemble with the mean of the target variables (say). Then for each iteration i, fits a regression tree to the negative gradient of the loss function. The negative gradient can be calculated as:

$$r_{im} = -\left[\frac{\partial Loss(y_i, F_i)}{\partial F_i}\right]_{m-1}$$

Where Loss() is the loss function used, y_n is the actual label and F_n is the predicted label for the n^{th} index of the training samples for the m^{th} iteration.

Then the optimal leaf weight is calculated for the new tree minimises the loss function.

$$\gamma_{jm} = \text{argmin}_\gamma \sum_i (Loss(y_i, F_i + \gamma_{jm} h_{jm}(x_i)))$$

Where h_{jm} is the prediction of the current tree for the i^{th} instance given that it falls in the j^{th} leaf.

The weights are then updated with the new tree and the optimal weight.

$$F_{mi} = F_{mi-1} + \eta \gamma_{jm} h_{jm}(x_i)$$

These steps are repeated until the desired result is achieved.

IV. Neural Networks

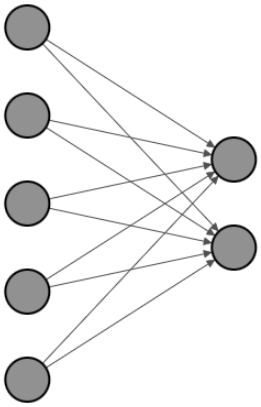
A. Multilayer Perceptron Model

This is based on the concept of the Perceptron Learning Algorithm (PLA), used for binary classification with the help of a single separating hyperplane. Note that, PLA is only able to classify linearly separable data. For non-linearly separable data, it needs to be shifted to a higher dimension to make it linearly separable on that particular plane.

Multilayer Perceptron Classifier (MLP Classifier) can be thought of as a network of elementary processors arranged in a very specific manner to mimic the neural structure of a human brain, capable of learning the hidden patterns and connections between the input and output values to classify them accordingly.

The classical Perceptron (single layer) can find the separating hyperplane for linearly separable data. The first layer is called the Input Layer, comprising d number of nodes or processing units signifying d attributes present in the dataset. The final layer is called the Output Layer, comprising

two nodes being a binary classifier. Each node of the Input Layer is connected to each node of the output layer, leading to a fully connected network. Refer to Fig.XX.



Input Layer $\in \mathbb{R}^5$ Output Layer $\in \mathbb{R}^2$

Each Input Layer node takes in the value from each attribute. All the edges connected from the Input Layer to the Output Layer are assigned random weights. So, the total input of each node of the Output Layer is defined by

$$u_{nj} = \sum_{i=0}^{d} x_{n_i} w_{ji}$$

Where j = 1, 2, ..., c implying c number of classes, u_{nj} is the total input from the input layer to the j[th] node of the output layer, x_{n_i} is the i[th] feature of the input layer, w_{ji} is the weight associated between the j[th] node of the output layer and i[th] node of the input layer for the n[th] instance of the dataset.

The value of $u_{nj} \in (-\infty, \infty)$, but the actual output has a value of 1 or 0, implying Class A and Class B respectively. Thus, a mapping is required to map the value of u_{nj} to {1,0} using a function. This function is called the Activation Function which maps the output value of the output node to {-1,1}. The most effective and commonly used Activation function is the sigmoid function.

$$v_{ji} = \emptyset(u_{ji})$$

Where, j = 1, 2, …, c, i = 0, 1, 2, …, d, v_{ji} is the output of the j[th] node of the output layer and $\emptyset()$ is the activation function defined by

$$\emptyset(x) = \frac{1}{1 + e^{-ax}}$$

Where a is constant.

The learning technique associated with MLP is called the backpropagation algorithm which utilizes the concept of the Gradient Descent technique proposed by Newton. The aim is to come up with a corrected set of weights for each edge by minimizing the error using the backpropagation technique. The pointwise error is calculated as below:

$$e_n = \frac{1}{2} \sum_{j=0}^{c} (y_{nj} - v_{nj})^2$$

Where e_n is the squared error for the n[th] data point of the dataset, y_{nj} is the actual output of the n[th] data point of the dataset and j[th] node, v_{nj} is the calculated output of the n[th] data point of the dataset and j[th] node of the output layer and c is the total number of classes.

Hence, the total error or the overall error (E) is calculated as

$$E = \frac{1}{N} \sum_{n=1}^{N} e_n$$

$$or, \quad E = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{2} \sum_{j=0}^{c} (y_{nj} - v_{nj})^2$$

$$or, \quad E = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j=1}^{c} (y_{nj} - v_{nj})^2$$

Where N is the number of entries in the dataset.

The aim is to reduce the total error (E) using the backpropagation algorithm. The only variable associated with E is the weights. So, the target is to optimize the weights to minimize the error. Thus, the Error(E) can be expressed as a function of weights.

$$E = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j=1}^{c} (y_{nj} - v_{nj})^2 = \psi(w_{ji})$$

Where w_{ji} is the weight between the j[th] node of the output layer and i[th] node of the input layer.

The backpropagation algorithm is stated as

Step 1 : Randomly assign weights (w_{ji}) for i = 0, 1, 2, …, d and j = 1, 2, …, c

Step 2 : Repeat loop starts

Step 3 : for loops starts for n = 1 to N

Step 4 : computer u_{nj}, v_{nj} and e_n

Step 5 : $w_{ji} = w_{ji} - \eta \frac{\partial e_n}{\partial w_{ji}}$

Step 6: for loop ends

Step 7: Repeat loop ends if the error has reduced to the permissible range or the iterations have reached their limit.

Since the error for each point is calculated and used for adjusting the weights, it is an extension of Gradient Descent known as Stochastic Gradient Descent (SGD). SGD takes care of the small local minima and ensures the algorithm is not stuck at those minima.

The MLP with a single layer can create a hyperplane to separate linearly separable data. For non-linearly separable data, it needs more than one layer called the Hidden Layer.

For MLP with one or more hidden layers, the concept remains the same. However, for each hidden layer, the weights need to be adjusted.

B. Artificial Neural Network

Artificial Neural Networks (ANNs) are a type of machine learning model or to be precise a Deep Learning Model that mimics the structure and function of the human brain. ANNs consist of a large number of interconnected processing nodes known as neurons, which are arranged in a very specific format. Each neuron receives input from all the neurons of the previous layer and processes that input before passing it on to the next layer.

The working of an ANN involves following stages

1) *Initialization: The network is structured by setting up the architecture which includes the total number of layers, the total number of neurons in each layer and the activation function to be used. The weights associated with each node are randomly assigned initially.*
2) *Forward Propagation: The data is first processed by the input layer and passed on to the first hidden layer. The neurons take in all the inputs, pass the value through an activation function and yield an output which is passed onto the next hidden layer. This process is continued till the output layer.*
3) *Activation Function: At each neuron, an activation function is applied to the input before it is passed on to the next layer. The activation function introduces non-linearity into the model, allowing the network to learn complex patterns in the data. There are many types of Activation Functions such as relu, softmax, sigmoid etc. among which for the intermediate layers, relu (Recti linear unit) is preferred and for the output layer sigmoid is preferred.*
4) *Loss Function: The output produced by the network is compared with the true output, and the difference between them is calculated. Usually, the Mean Squared Error (MSE) is used for Error Calculation. This can also be labelled as the loss function. The loss function is further used for the adjustment of the weights to make the predictions more accurate.*
5) *Backward Propagation: The Backpropagation method is used to consider the feedback of the loss function. It uses the concept of the Gradient Descent Algorithm proposed by Newton to update the weights of each edge. However, the implementation of Gradient Descent, has some disadvantages, like getting stuck in local minima and not being able to find the global minima. So, the Genetic Algorithm can also be used to overcome the shortages of Backpropagation.*
6) *Repeat: The entire process discussed above is repeated until the error is within an acceptable range or the epochs have reached the limit.*

The working of the ANN is very similar to that of the MLP Model and it is safe to conclude, MLP Model is a sub-class of ANN.

C. Residual Neural Network

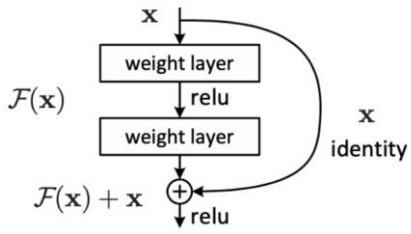
Residual Neural Network (ResNet) is an extension of the Convolution Neural Network (CNN). ResNet and CNN are primarily deployed for image processing as an application of Computer Vision.

The CNN was able to handle a particular number of hidden layers. For updating the weights, the Backpropagation method is used. After going back, a certain number of layers, there is a shift down of loss function. Over a significant number of layers, the gradient "vanishes" leading to the vanishing gradient problem, leading to a saturation in the performance of the CNN model. To overcome this issue, the ResNet model is deployed.

ResNet developed a unique solution, known as "skip connections". It stacks multiple identity mappings (convolutional layers that do nothing at first), ignores those particular sets of layers and again reuses the activation functions of the previous layer. Skipping or ignoring speeds up initial training by compressing the network into compact layers.

While the network is again trained, all the layers are uncompressed and the remaining parts of the network, known as the residual parts, are allowed to elaborate more on the feature space of the input image.

The majority of the ResNet models skip two or three layers at a time with nonlinearity and batch normalization in between. More advanced ResNet models, known as HighwayNets, can learn to "skip weights", which dynamically determine the number of layers to skip.



This shows the working of the Residual Block, which is responsible for skip connections in ResNet.

V. Methodology

Data PreProcessing is a very important step in machine learning. The goal of data preprocessing is to prepare and clean the raw dataset so that efficiency and accuracy of the machine learning algorithms can be maximized.

Similarly before processing our dataset through the different machine learning algorithms we had to prepare our dataset for optimal use.

The Processes we had to go through to prepare our dataset were:

1) *Null Value Replacement: Real World datasets can have missing data which needs to be replaced by the mean value of the missing attribute. We checked our dataset for missing values and as the dataset did not have any missing values we did not have to do anything.*

2) *Skewness Reduction: Skewness is the measure of the asymmetry of the probability distribution of an attribute. Excessive skewness can lead to bias in the final model. For our dataset, we first check each attribute for its skewness and for any attribute*

which has an absolute skewness value of greater than 1 we append that attribute column to a list named skewedCols. Below is the list of 5 attributes each which have the highest and lowest skewness before skewness reduction:

Attribute Name	Skew Value
tqwt_TKEO_mean_dec_32	26.48258509147365
tqwt_TKEO_std_dec_32	26.0620838755717
tqwt_TKEO_mean_dec_33	24.9442663610479
tqwt_TKEO_std_dec_33	24.283816068288253
det_TKEO_mean_3_coef	20.874872200923843
det_LT_entropy_shannon_7_coef	-21.4150979727166
tqwt_medianValue_dec_29	-21.623643949849406
tqwt_skewnessValue_dec_24	-22.684339987002808
tqwt_entropy_shannon_dec_33	-25.061352277117013
tqwt_entropy_shannon_dec_32	-25.672811274750888

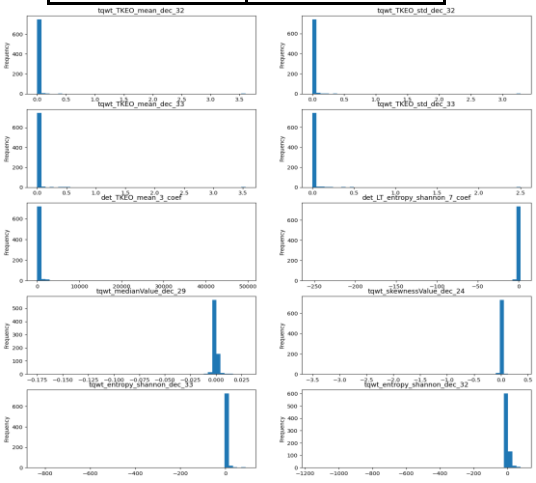


Fig. Attributes with the highest skewness before processing

Then we further classify the skewedCols based on whether that attribute contains any positive, zero or negative values into three separate lists named skewedCols_PositiveVals, skewedCols_ZeroVals, and SkewedCols_NegativeVals respectively.

Then for the attributes present in the list skewedCols_PositiveVals we first use Box-Cox Transformation and to reduce the skewness of the attributes. Box-Cox transformation works by applying a power function to the dependent variable which allows it to be transformed into a normal distribution and reduce its skewness.

We also use cube root transformation to reduce skewness of the attributes which are present in the lists skewedCols_ZeroVals, SkewedCols_NegativeVals. Cube Root transformation works by taking the cube root of each value of the attribute and making the attribute more closely resemble a normal distribution and reduce its skewness. Below is the final skewness of those attributes:

Attribute Name	Skew Value
tqwt_TKEO_mean_dec_32	0.19846778387365632
tqwt_TKEO_std_dec_32	0.01987384966738003
tqwt_TKEO_mean_dec_33	0.3104628329189008
tqwt_TKEO_std_dec_33	0.07507677689667193
det_TKEO_mean_3_coef	1.4434955464066024
det_LT_entropy_shannon_7_coef	-4.780452134423335
tqwt_medianValue_dec_29	-0.31596805061181005
tqwt_skewnessValue_dec_24	-1.690083153162617
tqwt_entropy_shannon_dec_33	-2.023628408215805
tqwt_entropy_shannon_dec_32	-2.120342043665047

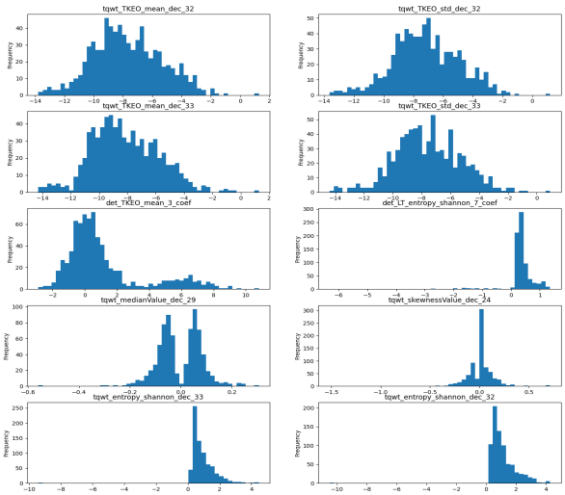


Fig. Attributes with the highest skewness after skewness Reduction

3) *Kurtosis Reduction: Kurtosis of an attribute is the measure of the peakedness/flatness of the probability distribution when measured relative to the normal distribution. The kurtosis of the normal distribution is considered zero. High kurtosis can lead to biases in the final model and hence we need to reduce kurtosis before processing our data. For our dataset, we check for attributes which have a*

kurtosis value greater than 3. For our dataset, we did not have any such attribute and hence we did not need to do any kurtosis reduction.

4) *Outlier Detection: Outliers are data points in an attribute which are significantly different from the rest of the data. Outliers are bad for machine learning as they can bias the results of the final model. For our model, We check each attribute and look for data points which are lower than the 25 percentile or higher than the 75 percentile of that attribute any data point which falls outside of this criteria is replaced by the mean of that attribute.*

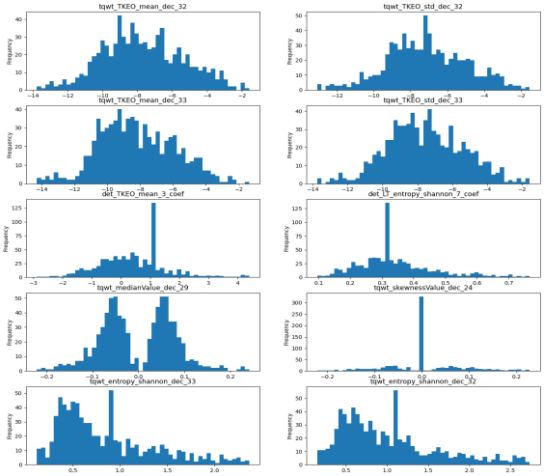


Fig. Attributes with the highest skewness after outlier Removal

5) *PCA: For our dataset we loop through different number of principal components and finally got 148 principal components which was giving us the optimal results.*

REFERENCES

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2].

We suggest that you use a text box to insert a graphic (which is ideally a 300 dpi TIFF or EPS file, with all fonts embedded) because, in an MSW document, this method is somewhat more stable than directly inserting a picture.

To have non-visible rules on your frame, use the MSWord "Format" pull-down menu, select Text Box > Colors and Lines to choose No Fill and No Line.

Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.