LTI CAMPUS INTERN

TOOL STATE CLASSIFICATION USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS

Mentors:
Ajesh Rajan, Rahmatullah Zacki

Performed By:
Nikhil Kumar

Outline

INTRODUCTION

What is Machine Learning?
Application of AI which provides systems an ability to automatically learn and improve from experience without being explicitly programmed.

Tool condition monitoring has become an essential industry phenomenon to achieve high-quality machine operation and cost management.
This method monitors the performance and condition of equipment during normal operation to reduce the likelihood of failures

OBJECTIVE

The objective of this use-case is to develop an algorithm that will classify if the tool will wear out or not given raw machine inputs.

We use the concept of machine learning and deep learning to classify whether the tool will wear out or not using the raw machine data. Using raw data as input to predict the tool state eases out any statistical calculation or filtering for the machine operators.

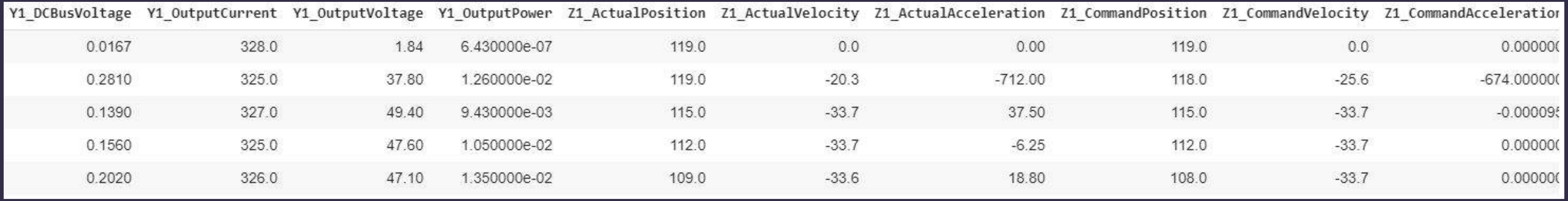
Here we present machine learning approaches for classifying the Tool state and integrate the most successful approaches into the experimental workflow.

DATASET AND DESCRIPTION

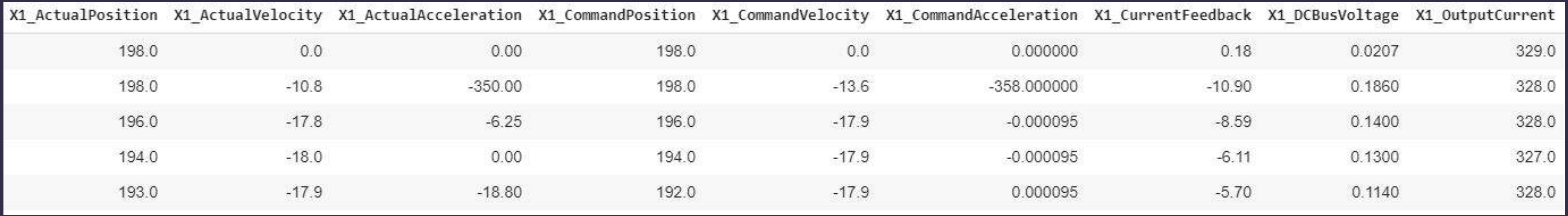
A series of 18 experiments were conducted whose time series data was collected. The Experiments were conducted on a CNC machine to create a wax structure. This Experiments were done at the smart lab of University of Michigan.

The dataset consisted of 25,286 instances of 51 features.
The features were the measurement of velocity, acceleration, position, voltage, current, power, position taken by sensors attached to X,Y,Z and spindle axes and also features like feed rate, pressure, material, process step and tool condition.

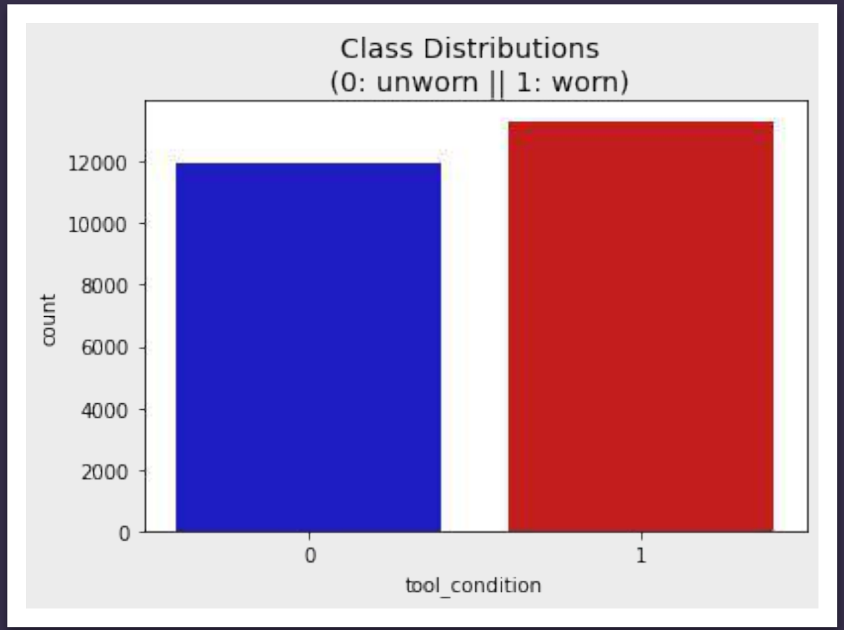
DATASET AND DESCRIPTION



Y1_DCBusVoltage	Y1_OutputCurrent	Y1_OutputVoltage	Y1_OutputPower	Z1_ActualPosition	Z1_ActualVelocity	Z1_ActualAcceleration	Z1_CommandPosition	Z1_CommandVelocity	Z1_CommandAcceleratio
0.0167	328.0	1.84	6.430000e-07	119.0	0.0	0.00	119.0	0.0	0.00000
0.2810	325.0	37.80	1.260000e-02	119.0	-20.3	-712.00	118.0	-25.6	-674.00000
0.1390	327.0	49.40	9.430000e-03	115.0	-33.7	37.50	115.0	-33.7	-0.00009
0.1560	325.0	47.60	1.050000e-02	112.0	-33.7	-6.25	112.0	-33.7	0.00000
0.2020	326.0	47.10	1.350000e-02	109.0	-33.6	18.80	108.0	-33.7	0.00000



X1_ActualPosition	X1_ActualVelocity	X1_ActualAcceleration	X1_CommandPosition	X1_CommandVelocity	X1_CommandAcceleration	X1_CurrentFeedback	X1_DCBusVoltage	X1_OutputCurrent
198.0	0.0	0.00	198.0	0.0	0.000000	0.18	0.0207	329.0
198.0	-10.8	-350.00	198.0	-13.6	-358.000000	-10.90	0.1860	328.0
196.0	-17.8	-6.25	196.0	-17.9	-0.000095	-8.59	0.1400	328.0
194.0	-18.0	0.00	194.0	-17.9	-0.000095	-6.11	0.1300	327.0
193.0	-17.9	-18.80	192.0	-17.9	0.000095	-5.70	0.1140	328.0



A balanced Dataset

DATA PREPROCESSING

In this step the data gets transformed, or Encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.

- Replacing Null values
- Categorical data to Numerical data
- Dimensionality Reduction using PCA

DATA PRE-PROCESSING

Why is Data Pre-processing important?

Features which have less null values are imputed using interpolation or mean, median values to create a consistent model. Features are dropped if high number of values are missing.

Categorical Data cannot be understood by computer so it is necessary to convert it to numerical form.

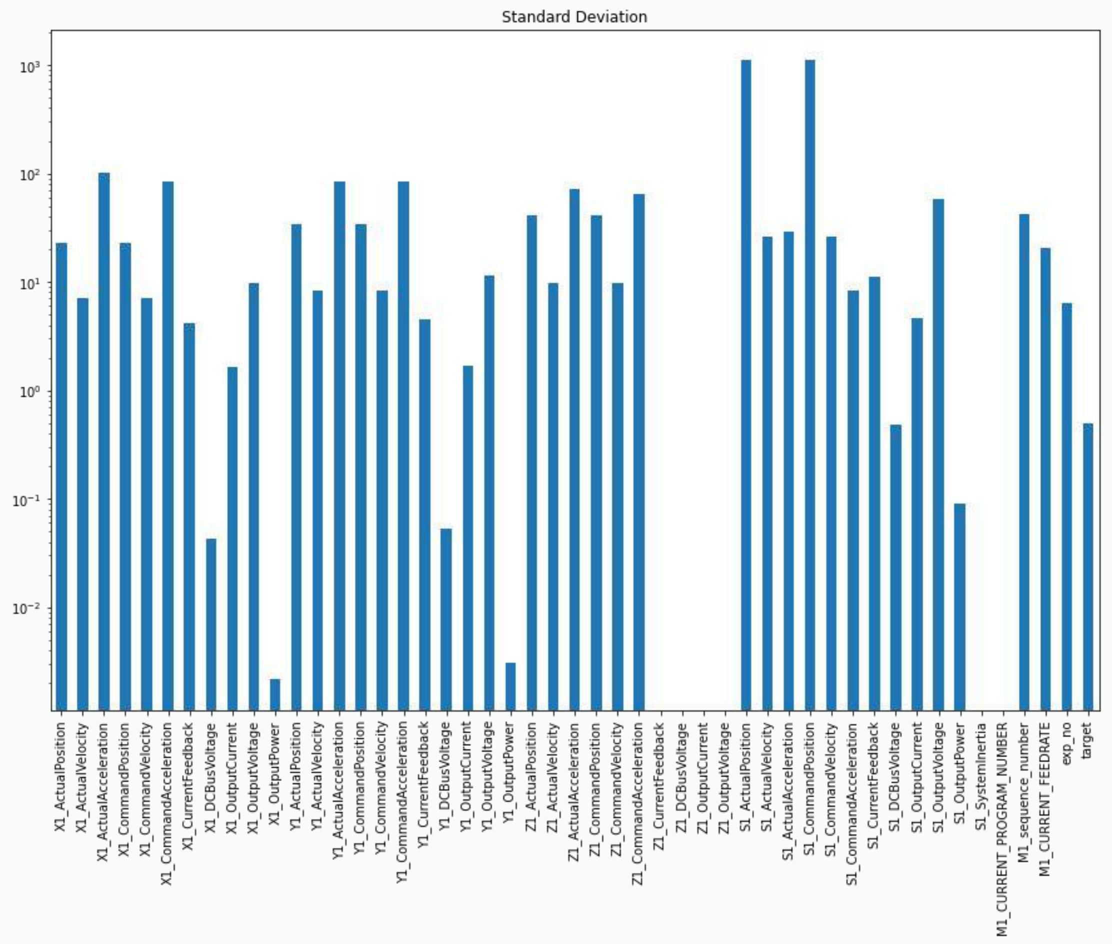
Data analysis tasks become significantly harder as the dimensionality of the data increases. As the dimensionality increases, the number planes occupied by the data increases thus the data becomes difficult to model and visualize.

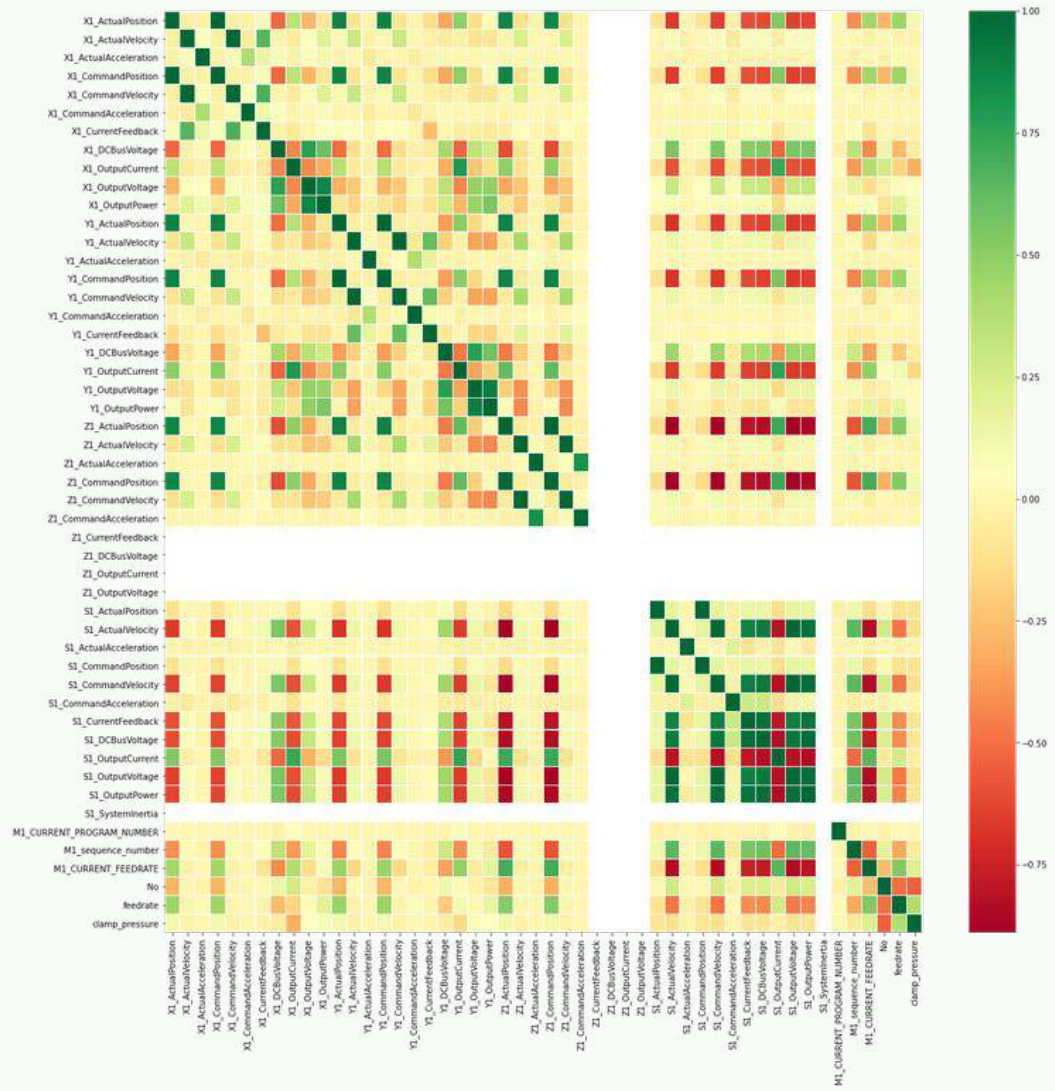
FEATURE SELECTION

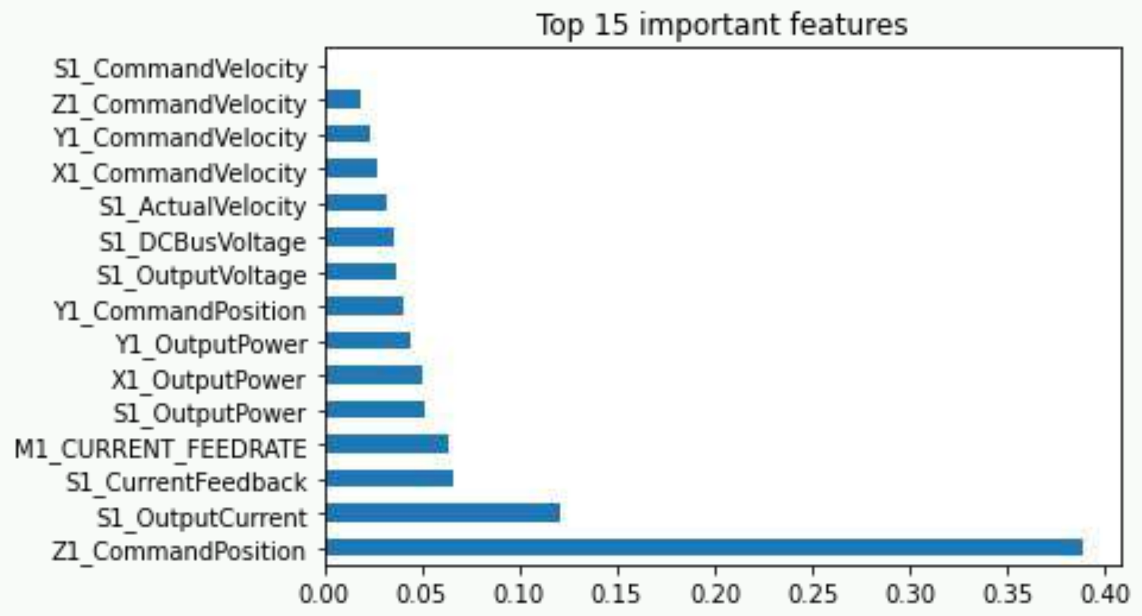
In feature selection we select those features which contribute most to the target variable and remove those features which affects negatively to the model performance.

- Selection on the basis of standard deviation
- Selecting values on the basis of Correlation matrix values
- Selection using library functions

FEATURE IMPORTANCE DISTRIBUTIONS







SD

Based on the standard deviation we choose features. The features who are constant or nearly constant are ignored

CORRELATION

It tells us how features are related to the target. Correlation values are both positive and negative

FEATURE IMP.

Feature importance gives us a score for each feature of your data

FEATURE SELECTION

Why is Data Feature selection important?

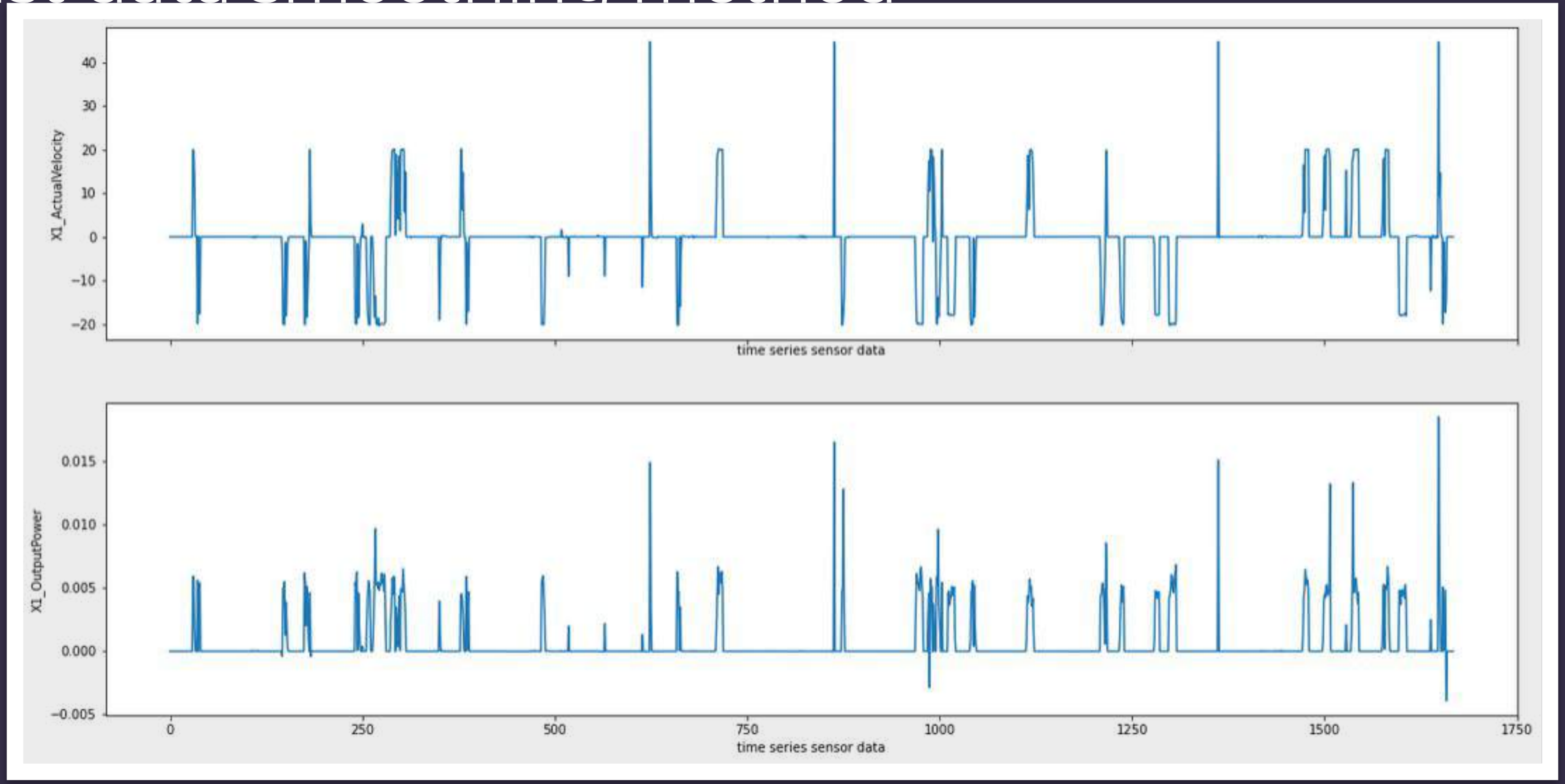
Feature selection hugely impacts the performance of the model. So the irrelevant features should be removed so that the performance of model is least affected.

- It helps in improving accuracy.
- It reduces overfitting because the model is not being trained on noisy data.
- It reduces model training time.

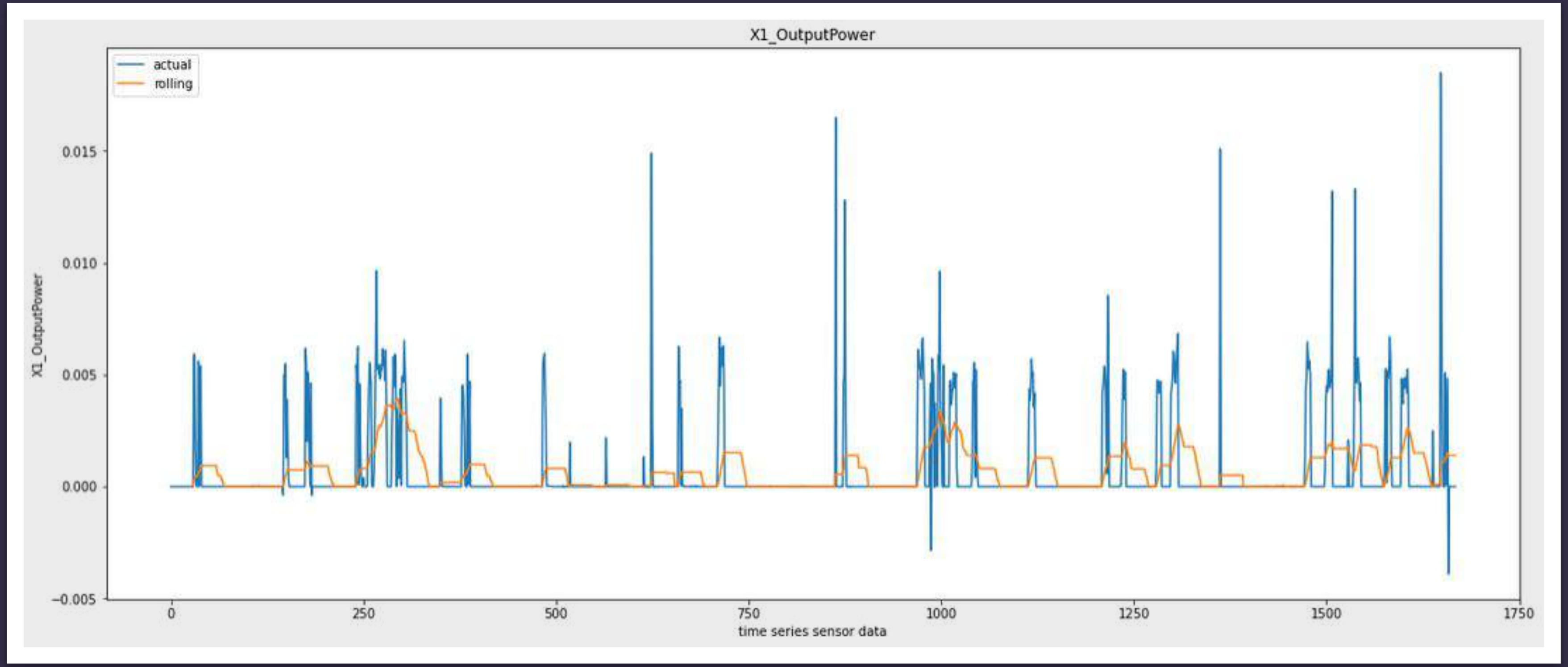
FEATURE CREATION AND SCALING

Feature creation is a step which is performed for a dataset or certain series if the features contain high amount of noise. Noisy data are inaccurate and are caused mainly due to human errors.
We used two methods to remove noise and smooth the data.

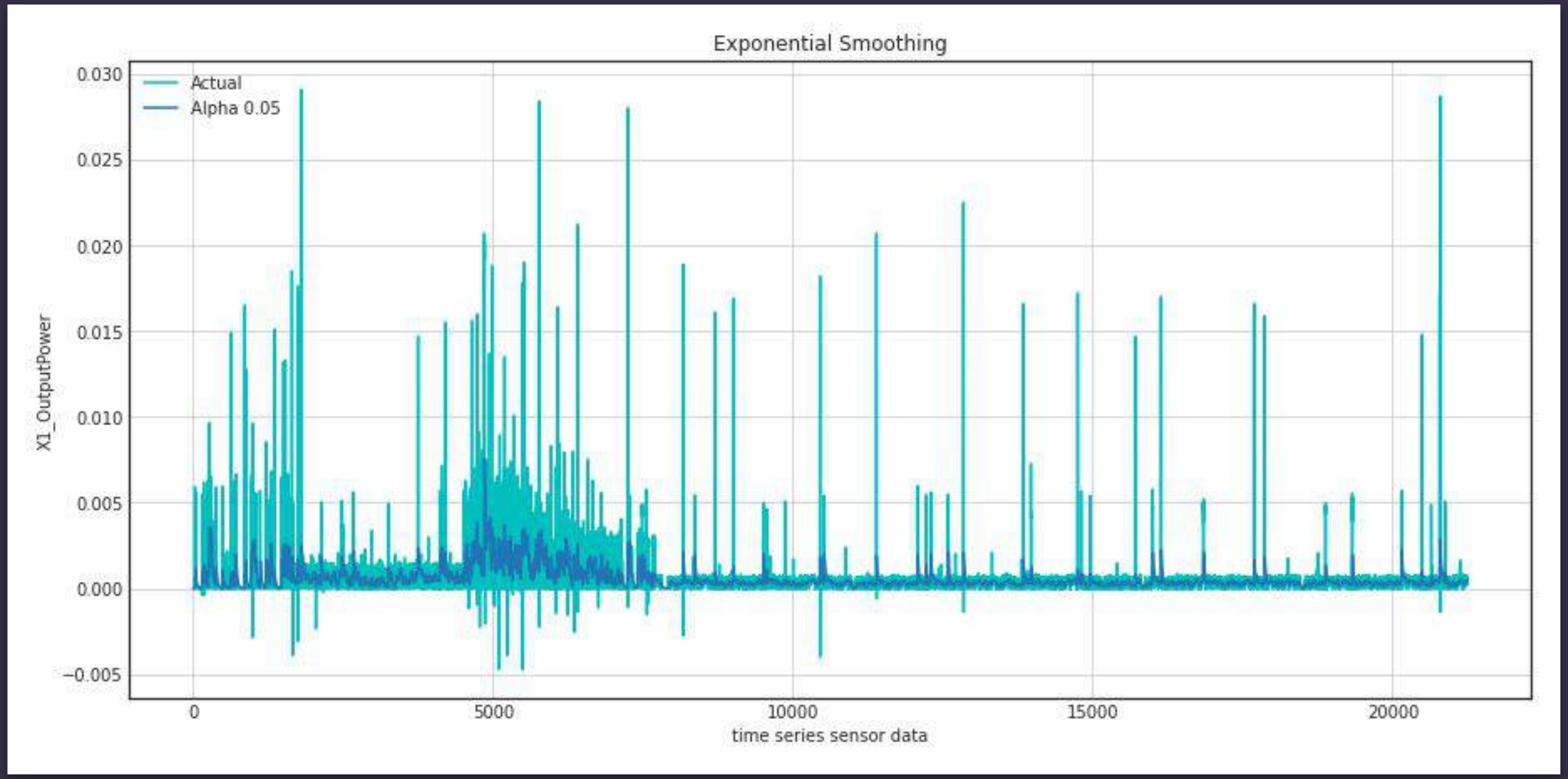
- Sliding window average mean
- Exponential forecast data smoothing method



FEATURE CREATION AND SCALING



Running Mean helps to determine the underlying trend in a feature of the data. A running mean is an average for a window of data, where the window is a series of sequential values from the data.



Exponential smoothing is a technique for smoothing data using the exponential window function. Here the exponential smoothing function acts as a low pass filt er.

FEATURE CREATION AND SCALING

Feature Scaling is a normalization technique to convert the data such that it lies within a range using min-max scaling method or deviating the data assuming it is centered at 0 by standard scaler method.

Why are feature creation and scaling important?

Feature creation removes the noise which if left will let the model to learn the noise as a pattern and thereby decrease model accuracy.
Feature scaling speeds up the algorithm calculations.

DATA DIVISION

We split the data as a training set and testing set. We used two methods to split the dataset.

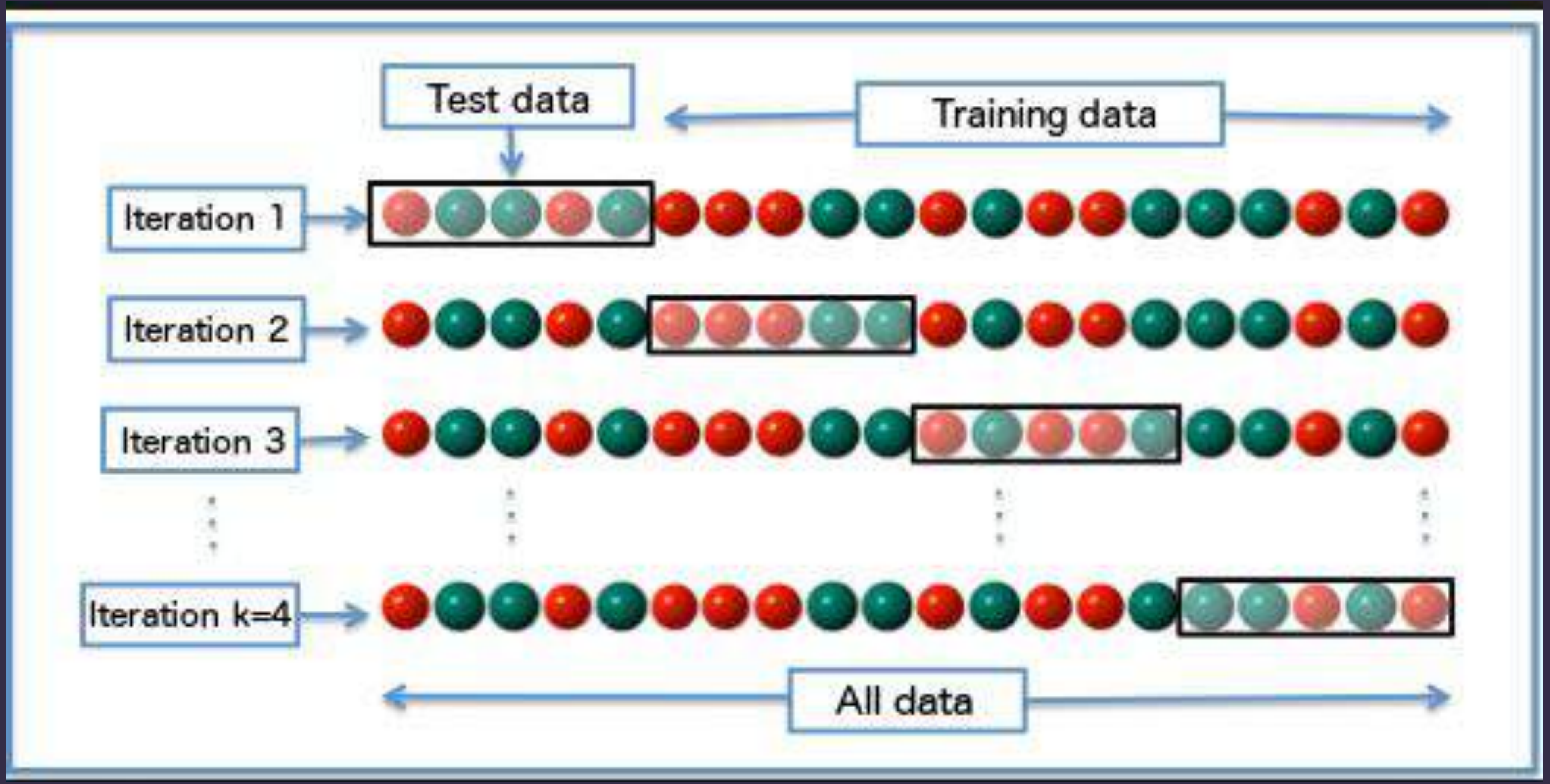
- Clubbing all experiments and use 20% data for testing and other for training purpose.
- Creating all possible combination of balanced data based on the experiments. Here the time series feature of the data is exploited to see if the data shows any trend with respect to time.

CROSS VALIDATION

We test the model on unseen data to see if the model overfits or underfits. Cross validation helps to train and test the model on entire data by splitting the data in k folds.

Why cross validation?

Cross-Validation helps us to test the effectiveness of our machine learning model. By finding the mean of all K folds we get the best accuracy for our model.



HYPERPARAMETER TUNING

Hyperparameter tuning is a method to choose the best parameters for any model for which the performance of the model is best.

The parameters like n_neighbours, max_depth, learning rate, optimizers, epochs, hidden layers etc. govern the training process of any model and produce a significant impact on the accuracy of the model.

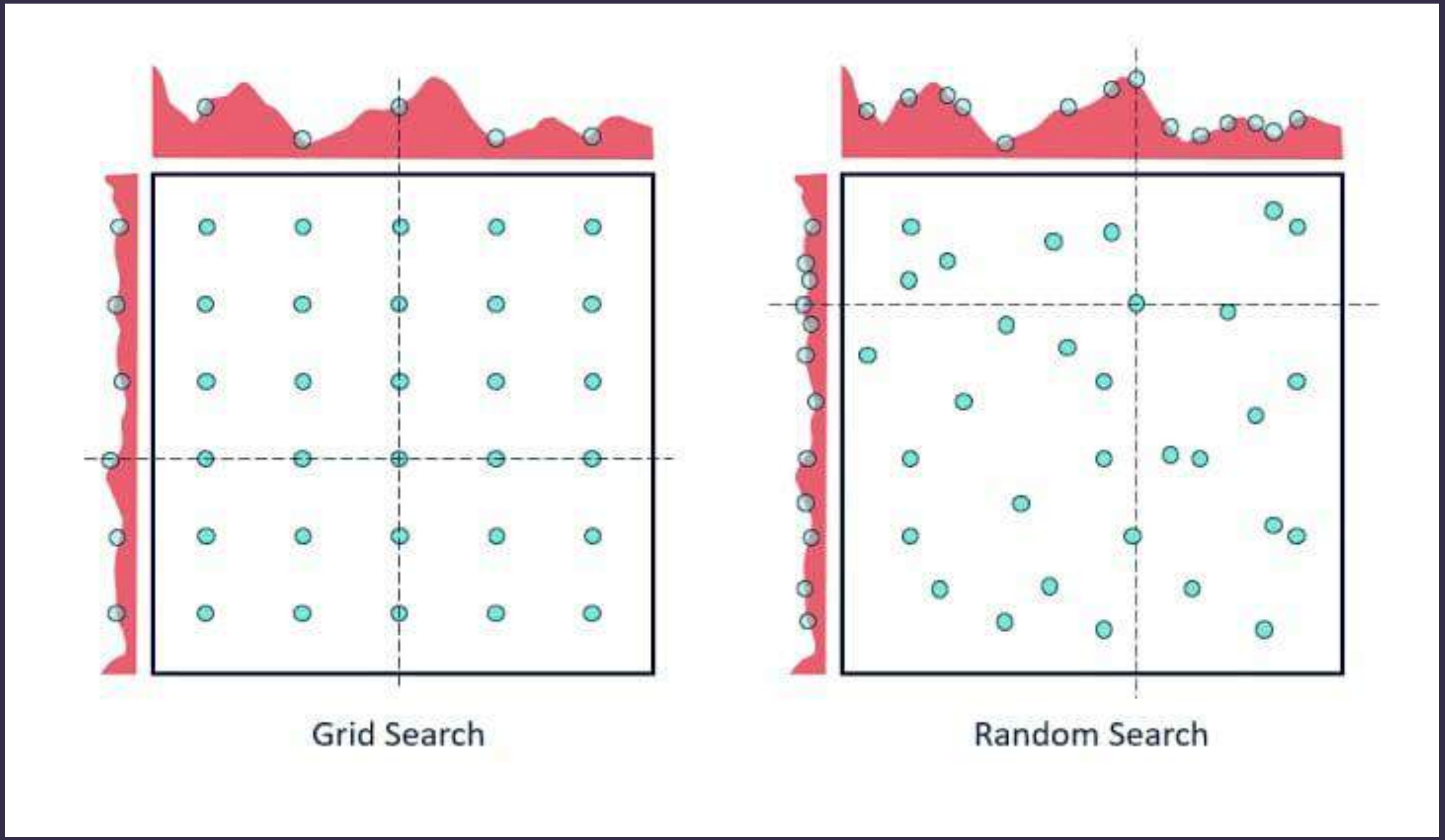
To tune the hyperparameters we used:

- Grid Search method
- Randomized grid search method

Grid Search method is a brute force method which trains, tests and validates the model for every combination of parameters which are passed and stores the parameters of best performing model.

Randomized search is similar to grid search method but instead of training for every combination it tests for random n combination and it helps to reduce tuning time.

HYPERPARAMETER TUNING



Why Hyperparameter Tuning?

Hyperparameter Tuning is done because a good parameter gives most accurate model and reduces loss.

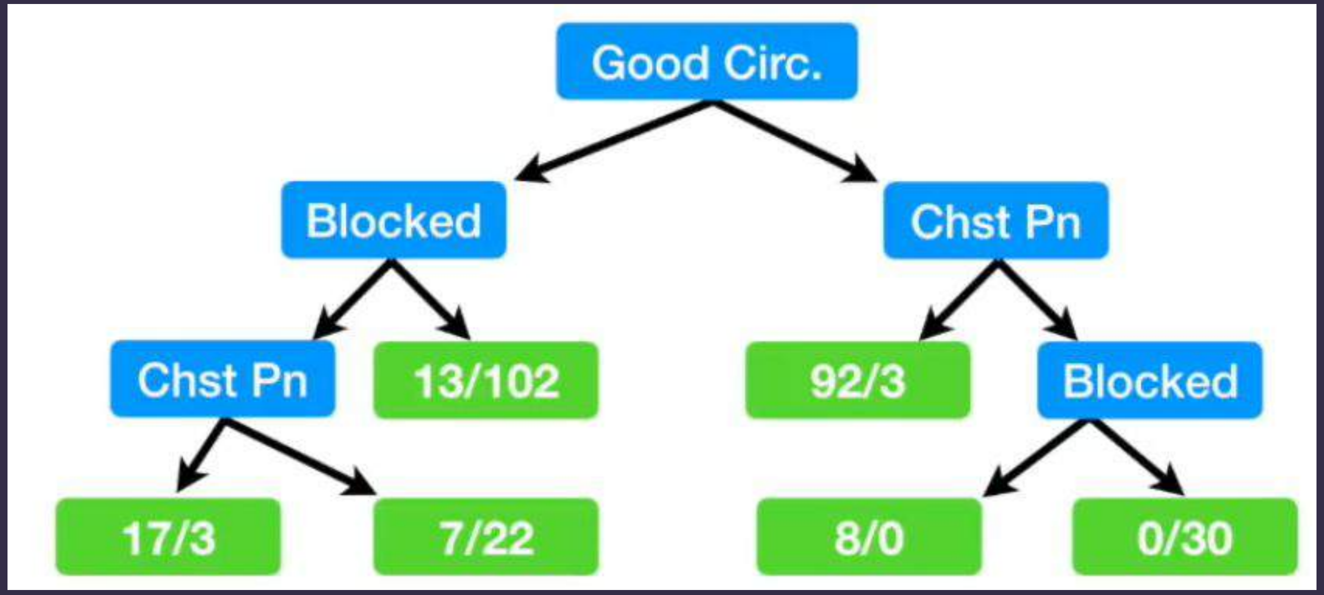
LEARNING MODELS

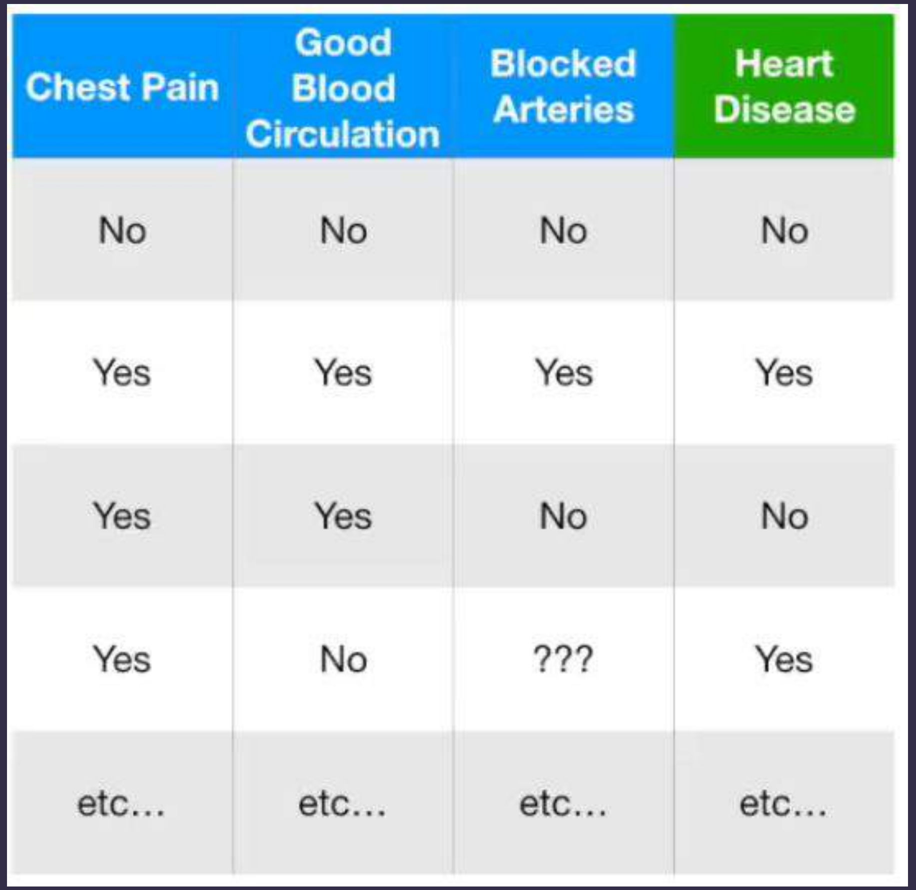
We trained and tested our dataset on different machine learning and deep learning models and the models on which our data showed good accuracy are:

- Decision Tree Classifier
- Random Forest Classifier
- XG Boost Classifier
- K Nearest Neighbours
- Artificial Neural Networks

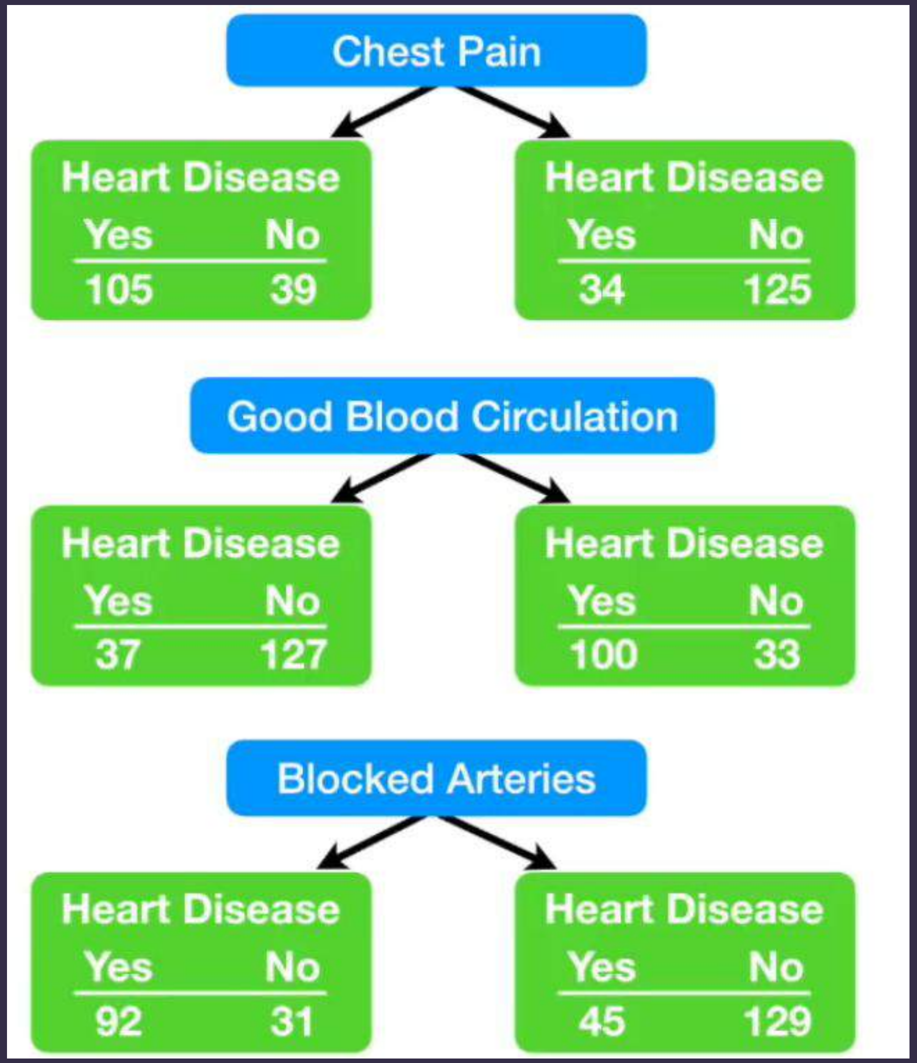
DECISION TREE CLASSIFIER

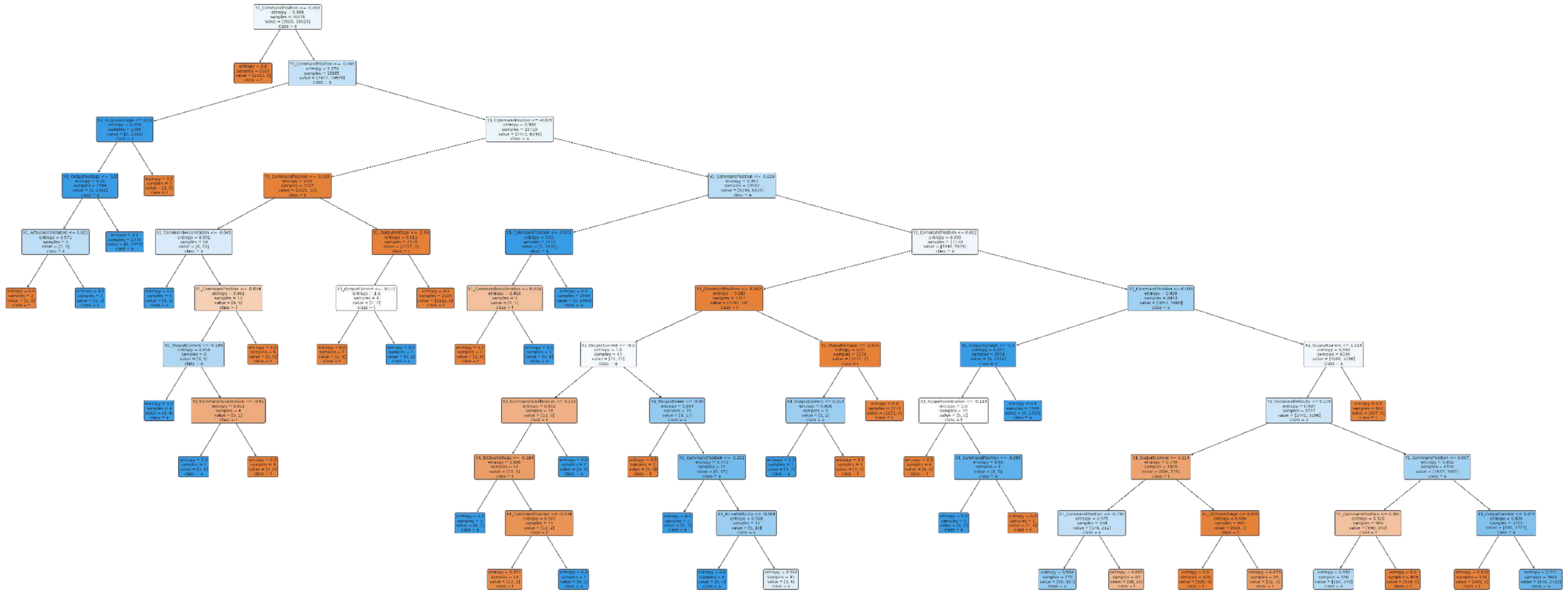
Decision Tree is like a flowchart tree structure where every node has a decision function and the leaves are the outcomes. Decision Tree takes very less computation time and is faster than Neural Networks. For noisy and unbalanced data the algorithm does not work well so we need to remove noises to improve the model in such cases. The hierarchy of the tree is decided on the basis of entropy.



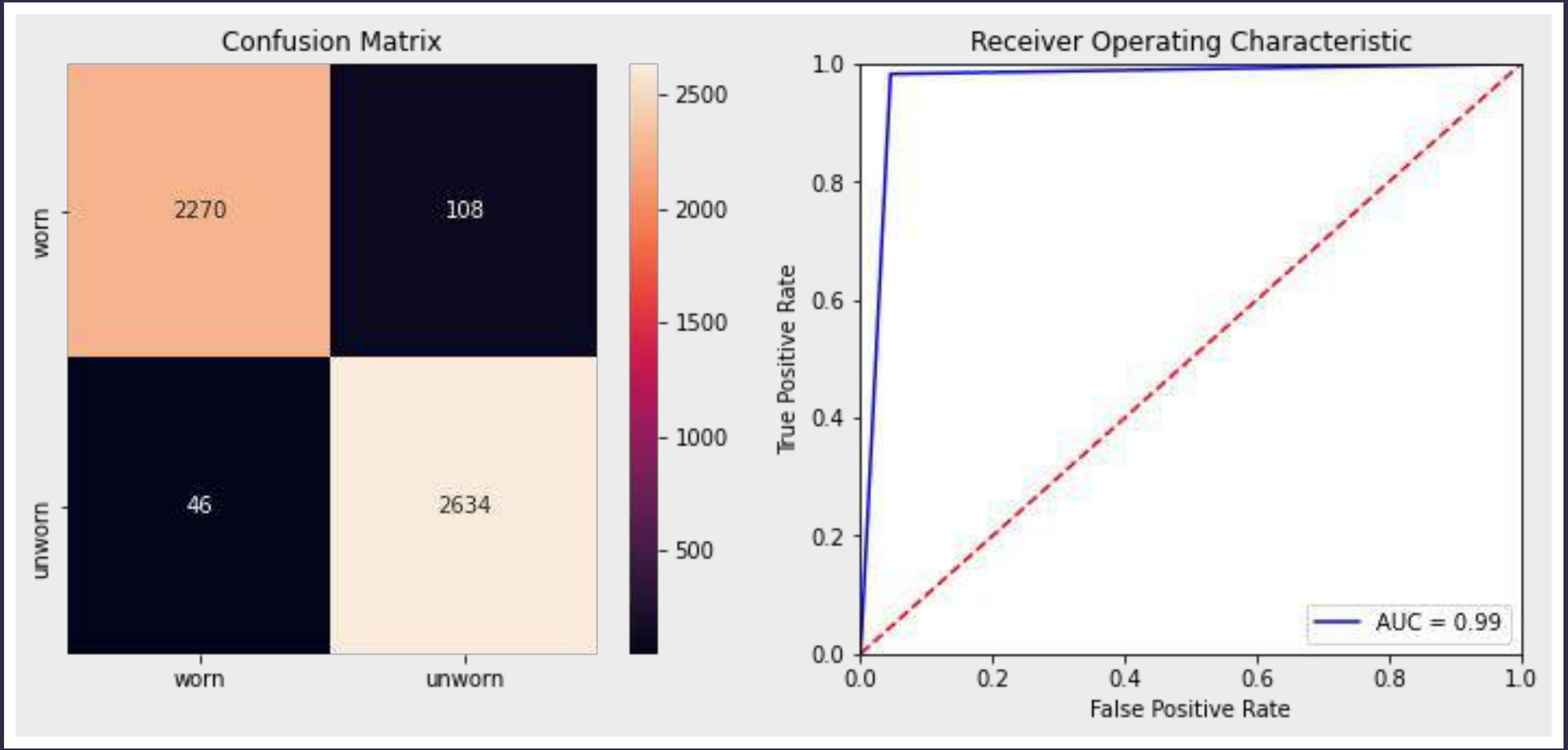


Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc...	etc...	etc...	etc...




Y1_CommandPosition <= -0.069
entropy = 0.998
samples = 20278
value = [9600, 10628]
class = a
entropy = 0.0
samples = 2123
value = [2123, 0]
class = t
Y1_CommandPosition <= -0.065
entropy = 0.978
samples = 18105
value = [7477, 10628]
class = a
Y1_CommandPosition <= -0.035
entropy = 0.998
samples = 15719
value = [7473, 8246]
class = a

DECISION TREE CLASSIFIER



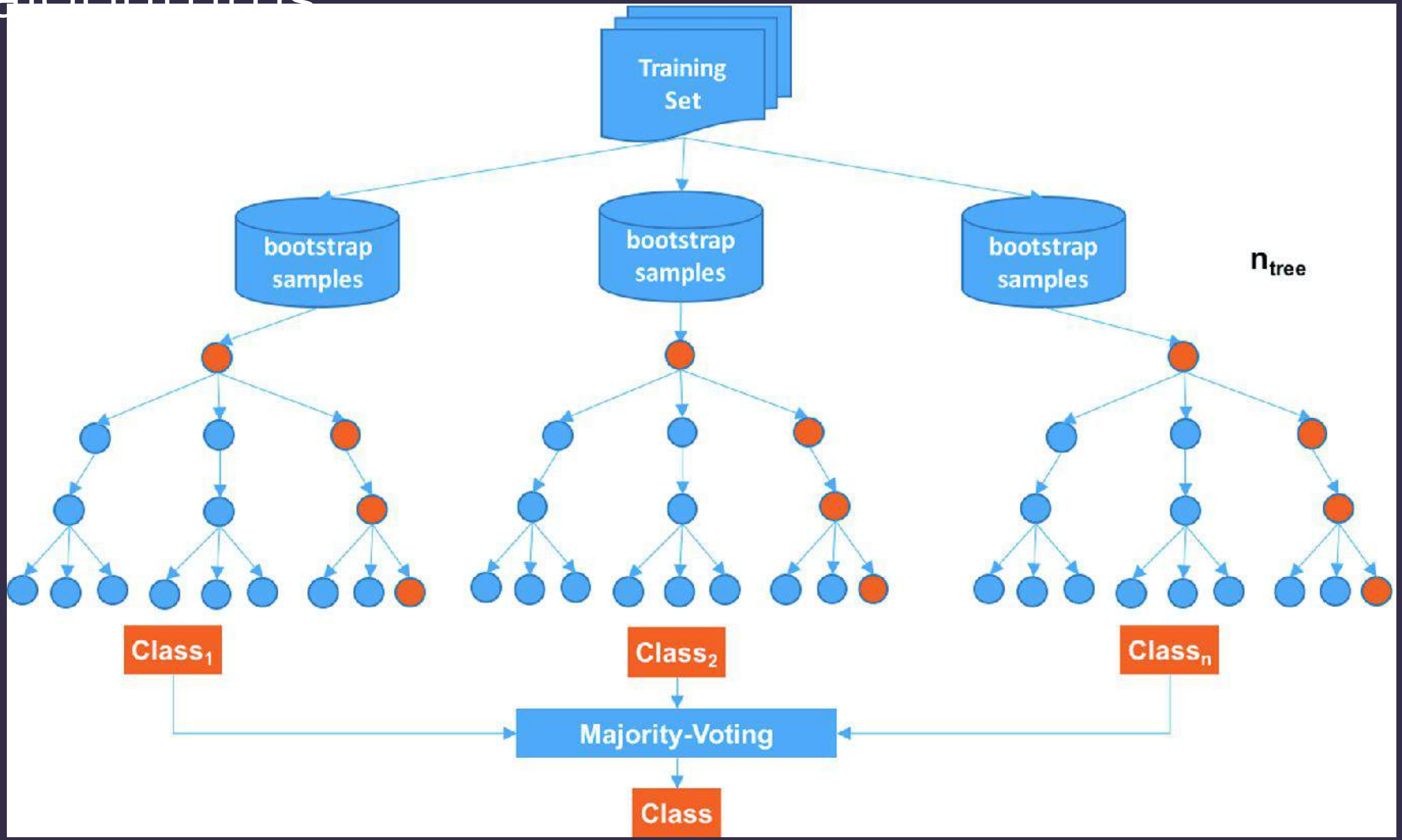
Training Accuracy: 0.986

Testing Accuracy: 0.969

ROC AUC SCORE: 0.985

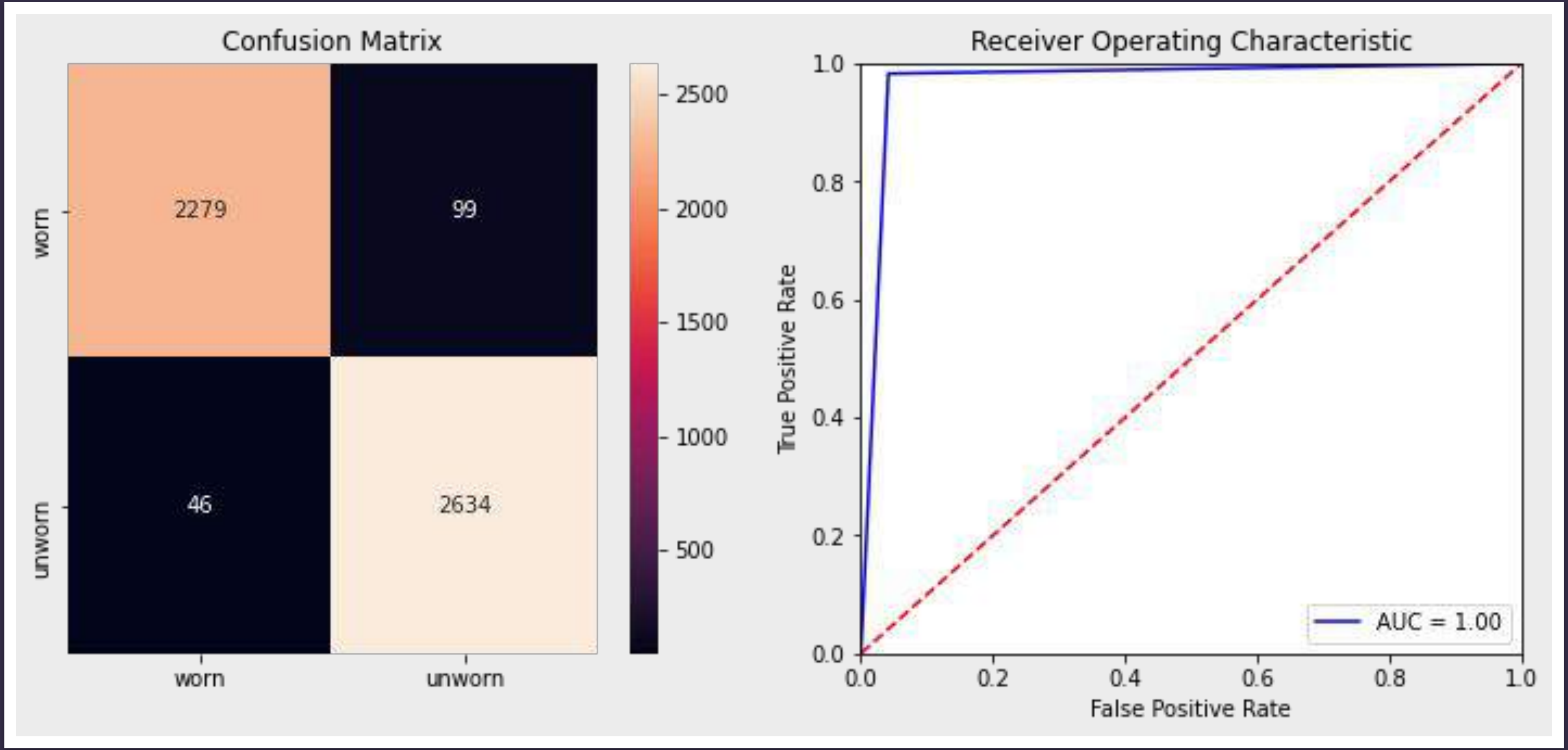
RANDOM FOREST CLASSIFIER

It is an ensemble tree based algorithm. Random Forest classifier is a set of decision trees from randomly selected subset of training set. The trees are voted and the majority voting class decides the final class for the test set. Random forest provides the best accuracy but it takes longer computation time than decision tree, Artificial neural networks and gradient boosting algorithms.



- Bootstrap sample of same shape from training set.
- Build a decision tree on the sampled data.
- Repeat the steps to create n trees.
- Check the accuracy on the out of bag data.

RANDOM FOREST CLASSIFIER



Training Accuracy: 1.0

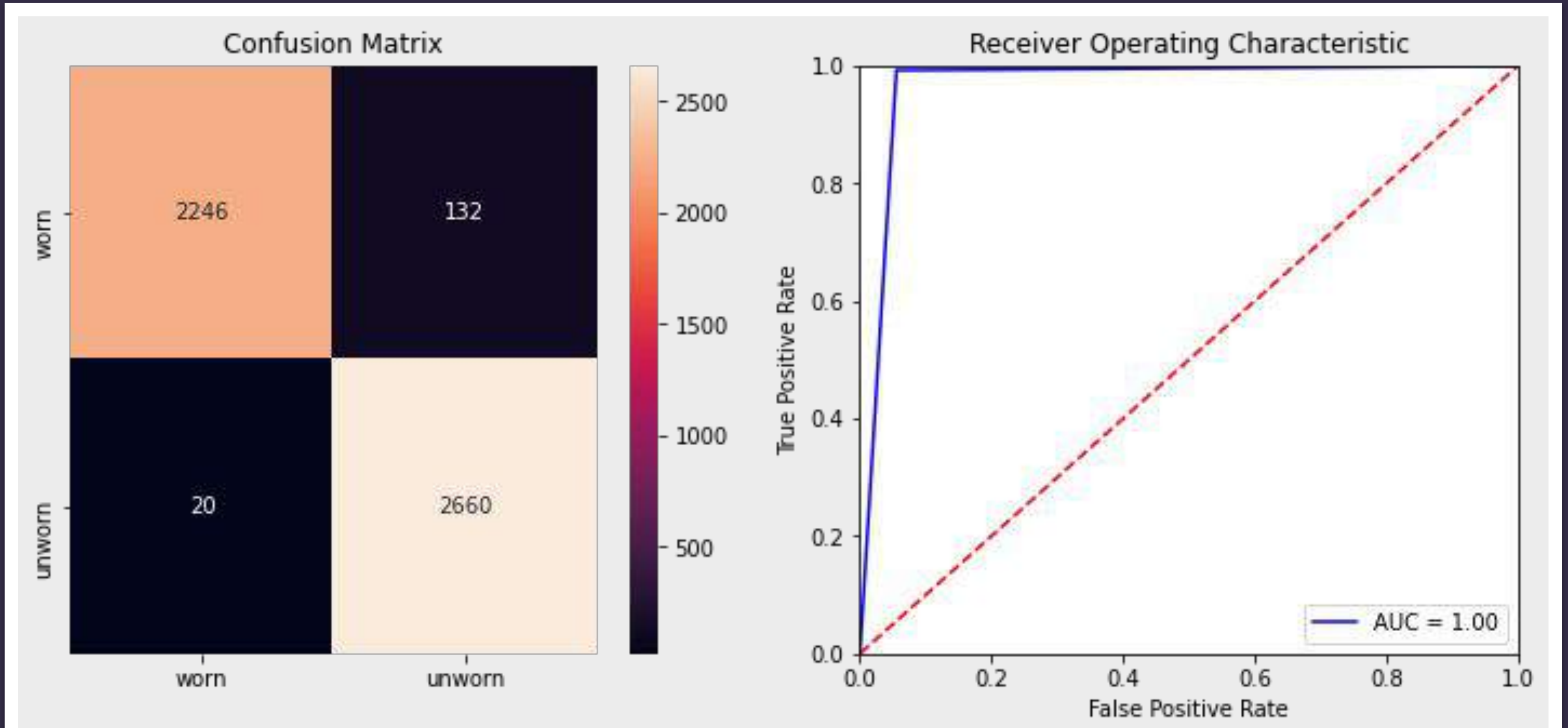
Testing Accuracy: 0.97

ROC AUC SCORE: 0.99

XG BOOST CLASSIFIER

XG Boost Classifiers use optimized gradient boosting algorithm through parallel processing, tree pruning, handling missing values and regularization to avoid overfitting. It works best for small to medium structured data however it is outperformed by neural nets when large datasets are used.

Training Accuracy: 0.972
Testing Accuracy: 0.969
ROC AUC SCORE: 0.997

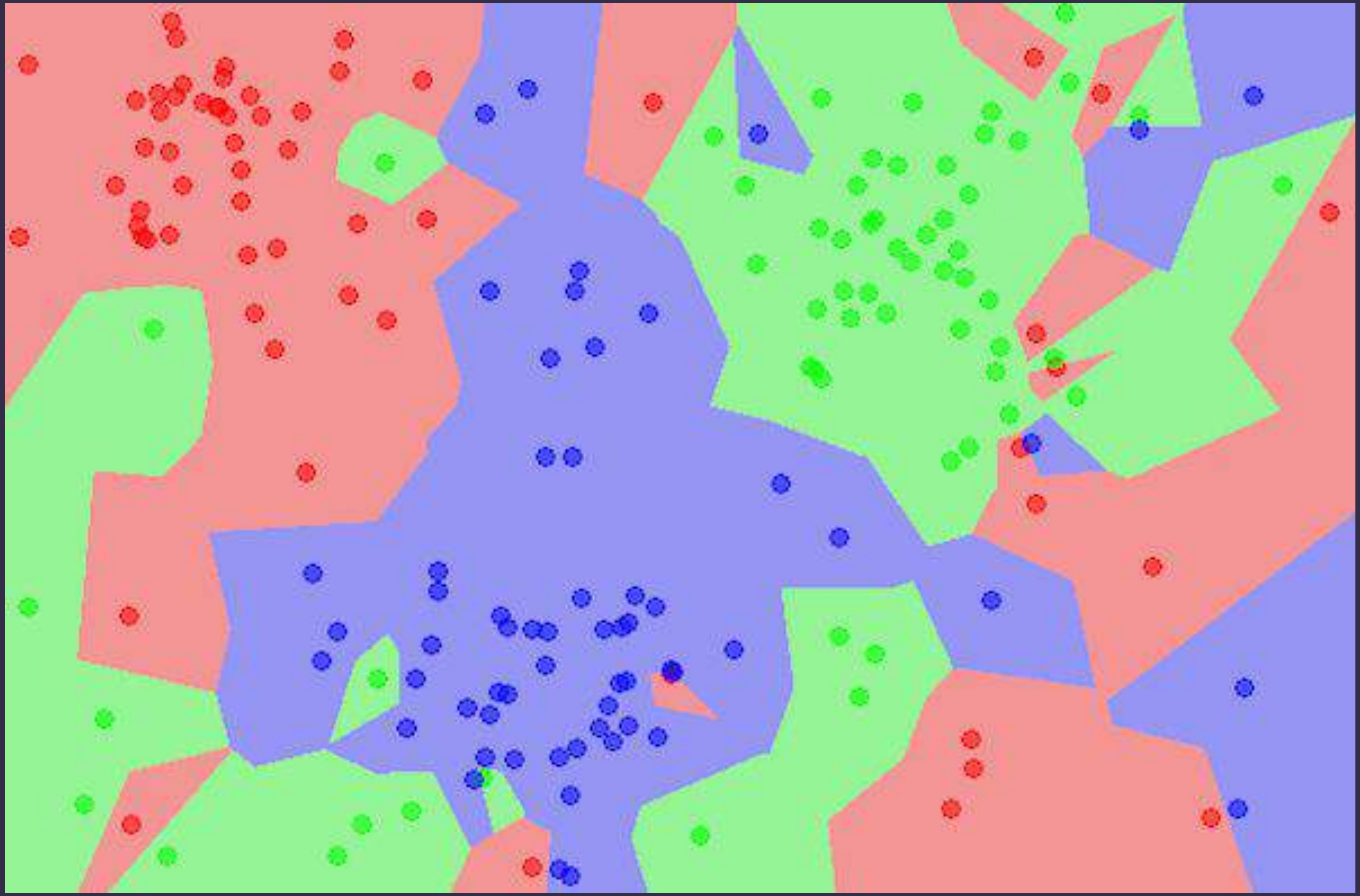


K NEAREST NEIGHBOURS

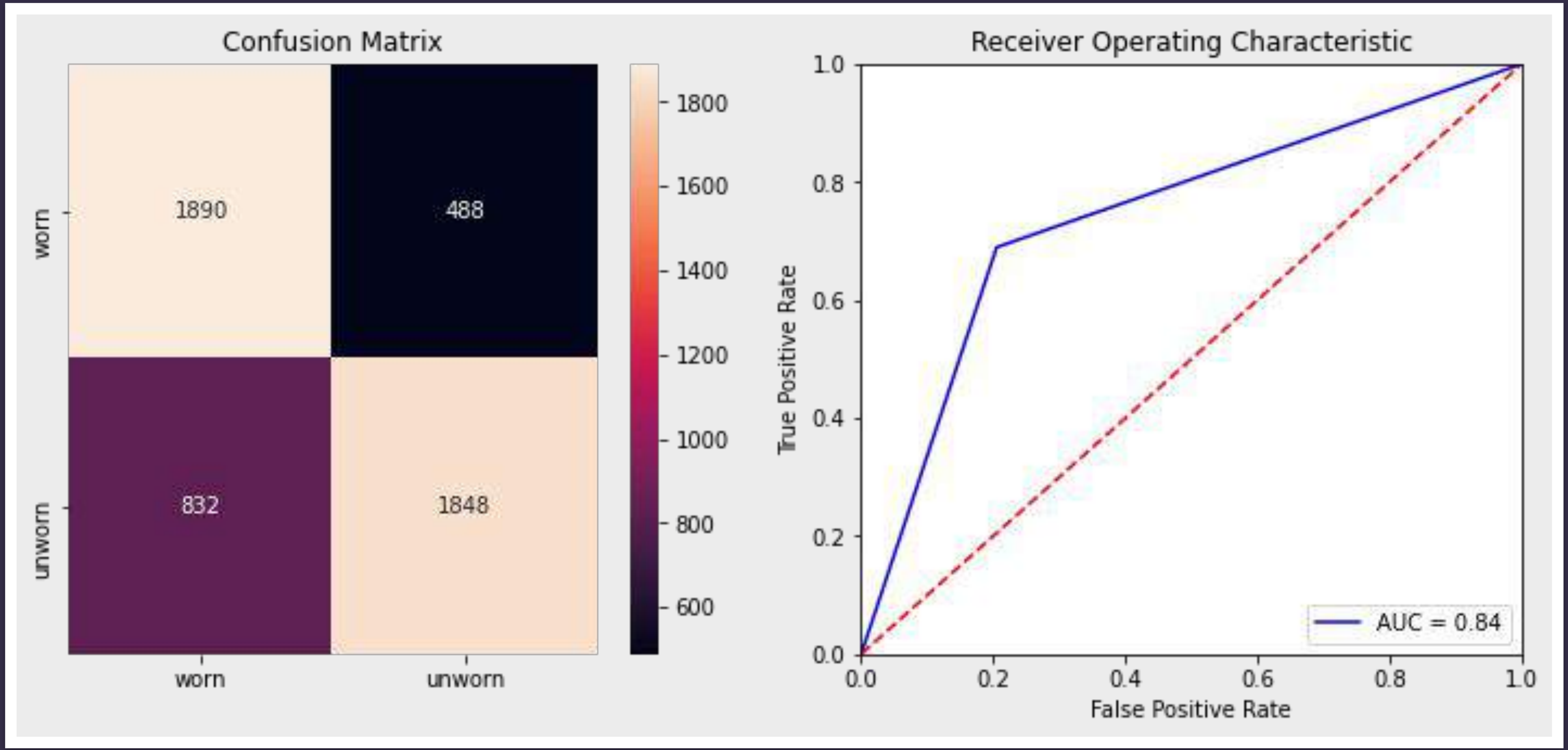
KNN is a non-parametric and lazy learning algorithm i.e; model structure is determined by the dataset and KNN does not needs to be trained for model generation so it makes testing phase slower and consumes high amount of memory. KNN model performs poor at high dimensional dataset as is the case. We normalize the dataset in this case and also tune this algorithm for different hyper-parameters to achieve best estimator.

- Randomly select K data values.
- Measure the distance of every data from those randomly selected data.
- Now based on distance value sort them in ascending order.
- Using that order we create the clusters.

K NEAREST NEIGHBOURS



K NEAREST NEIGHBOURS



Training Accuracy: 0.807

Testing Accuracy: 0.735

ROC AUC SCORE: 0.831

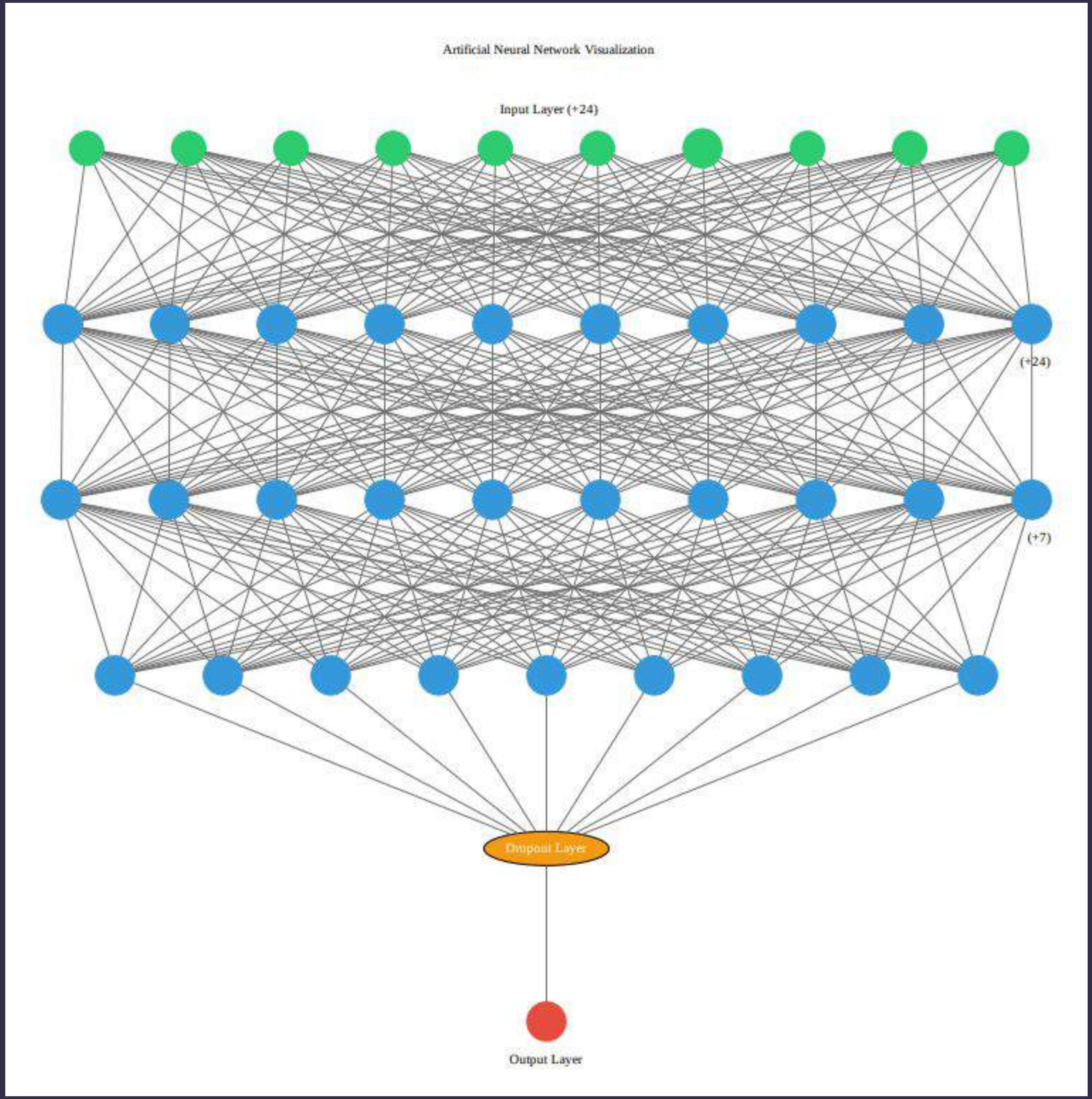
ARTIFICIAL NEURAL NETWORKS

Artificial Neural Network consists of an artificial network of functions, called parameters, which allows the computer to learn, and to fine tune itself by analyzing new data. For this model the number of input parameters (34) were more than that of ML models (16).

We created feed forward sequential model using keras library. The network consisted of three dense layers. It took a 2D input of dimension (1, 34) and passed it to next dense layer of 17 nodes through a Relu activation function. The second dense layer passes its input through a function and Relu activation function to next layer of nine nodes which then passes it to the last layer of single node which uses sigmoid function to determine the truth value of the input.

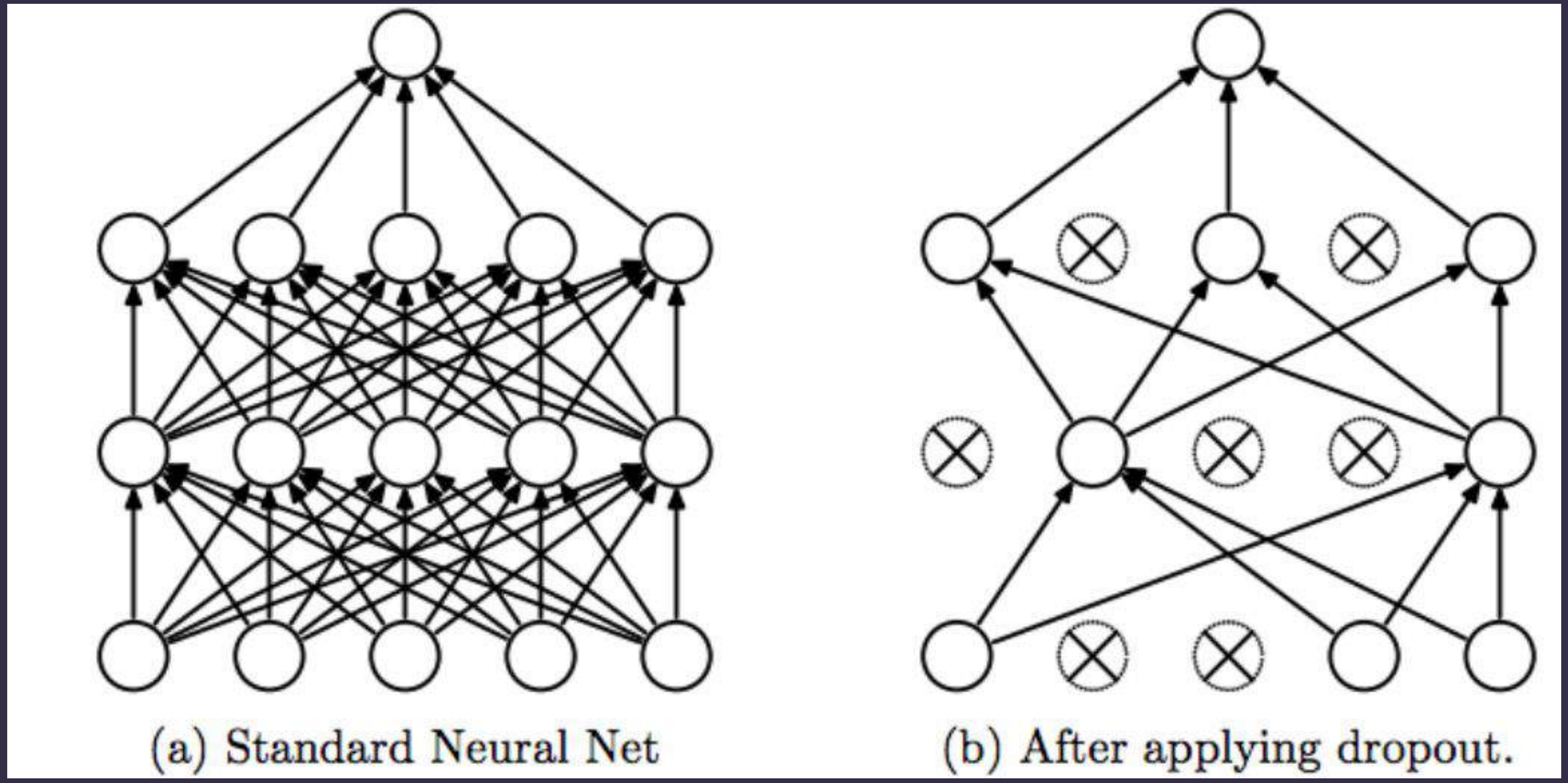
The train and test sets were scaled using standard scaler function so that input feature values implicitly weights all features equally in their

ARTIFICIAL NEURAL NETWORKS



- Input vector value
- Forward Propagation
- Calculate the loss value from cost function
- Back-propagation
- Nudge the weights and biases according to the loss value
- Repeat it according to the epochs

ARTIFICIAL NEURAL NETWORKS



Without Dropout

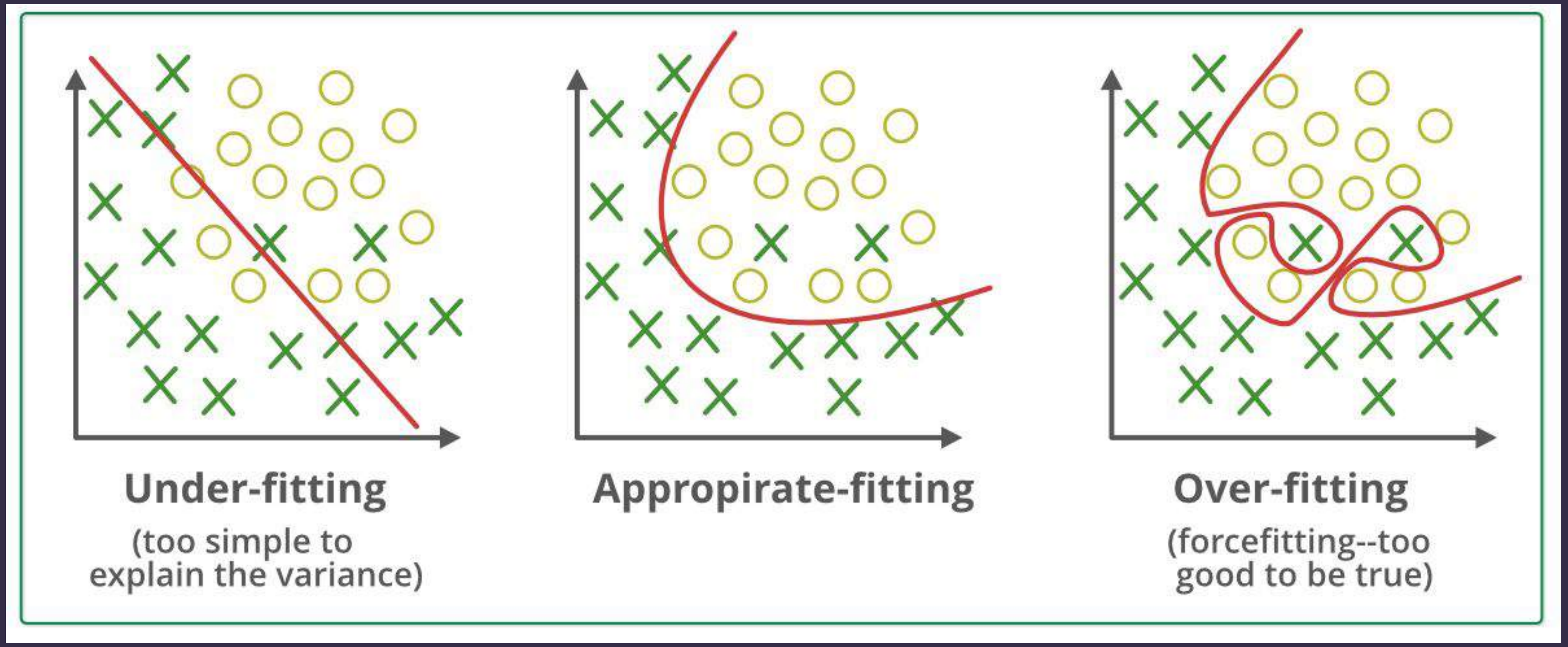
Training Accuracy: 0.98

Testing Accuracy: 0.92

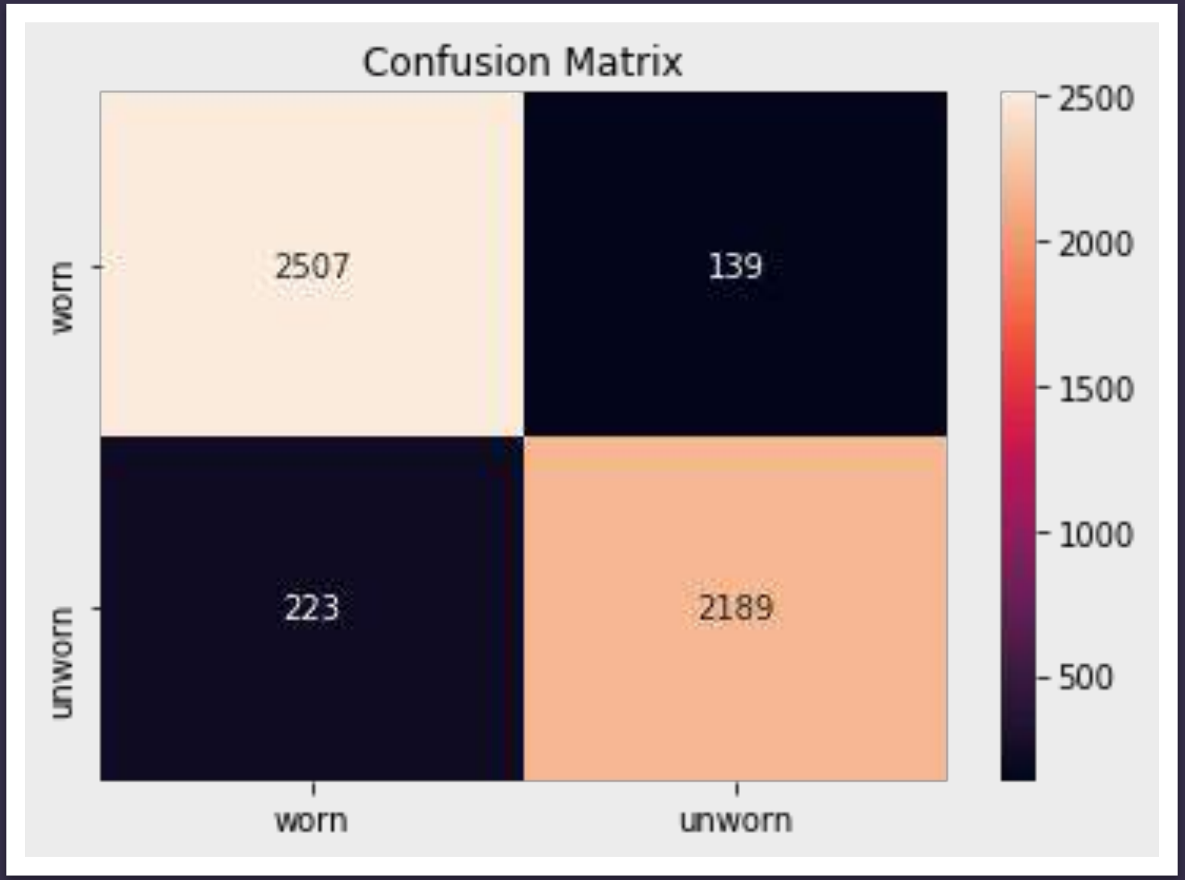
With Dropout

Training Accuracy: 0.94

Testing Accuracy: 0.93

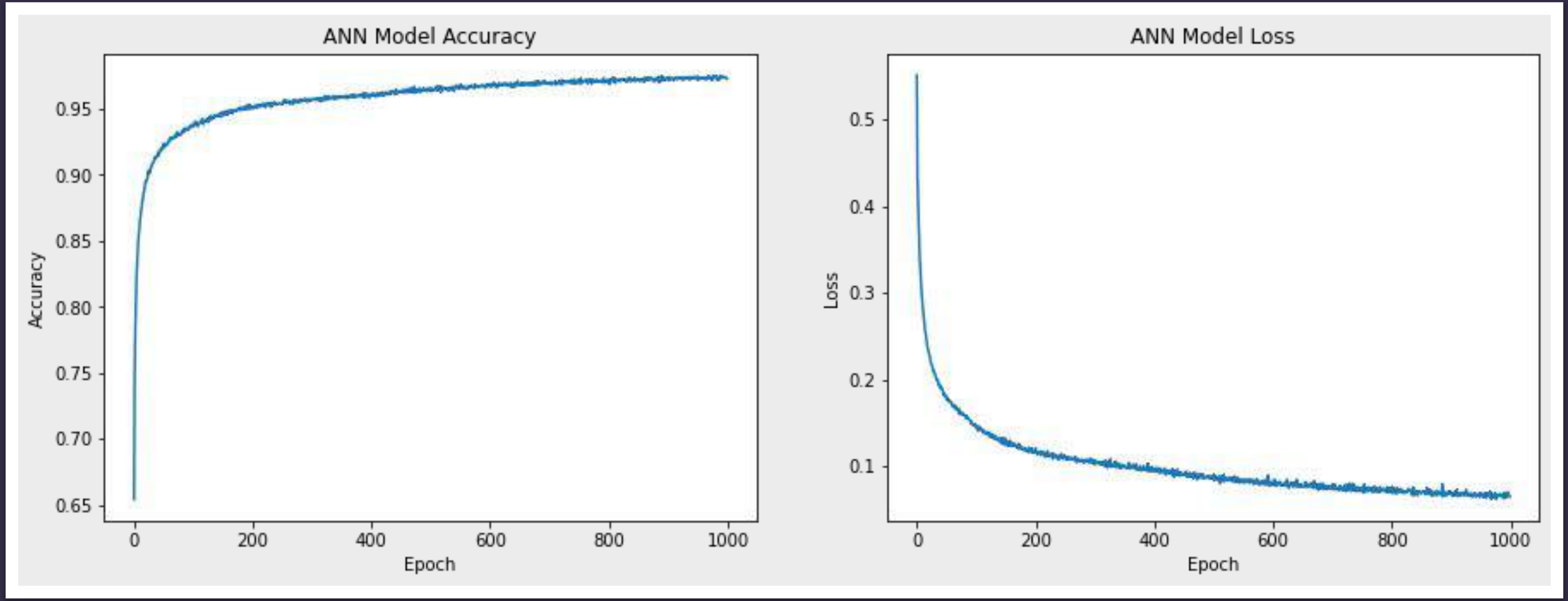

Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

ARTIFICIAL NEURAL NETWORKS



Training Accuracy: 0.94

Testing Accuracy: 0.93



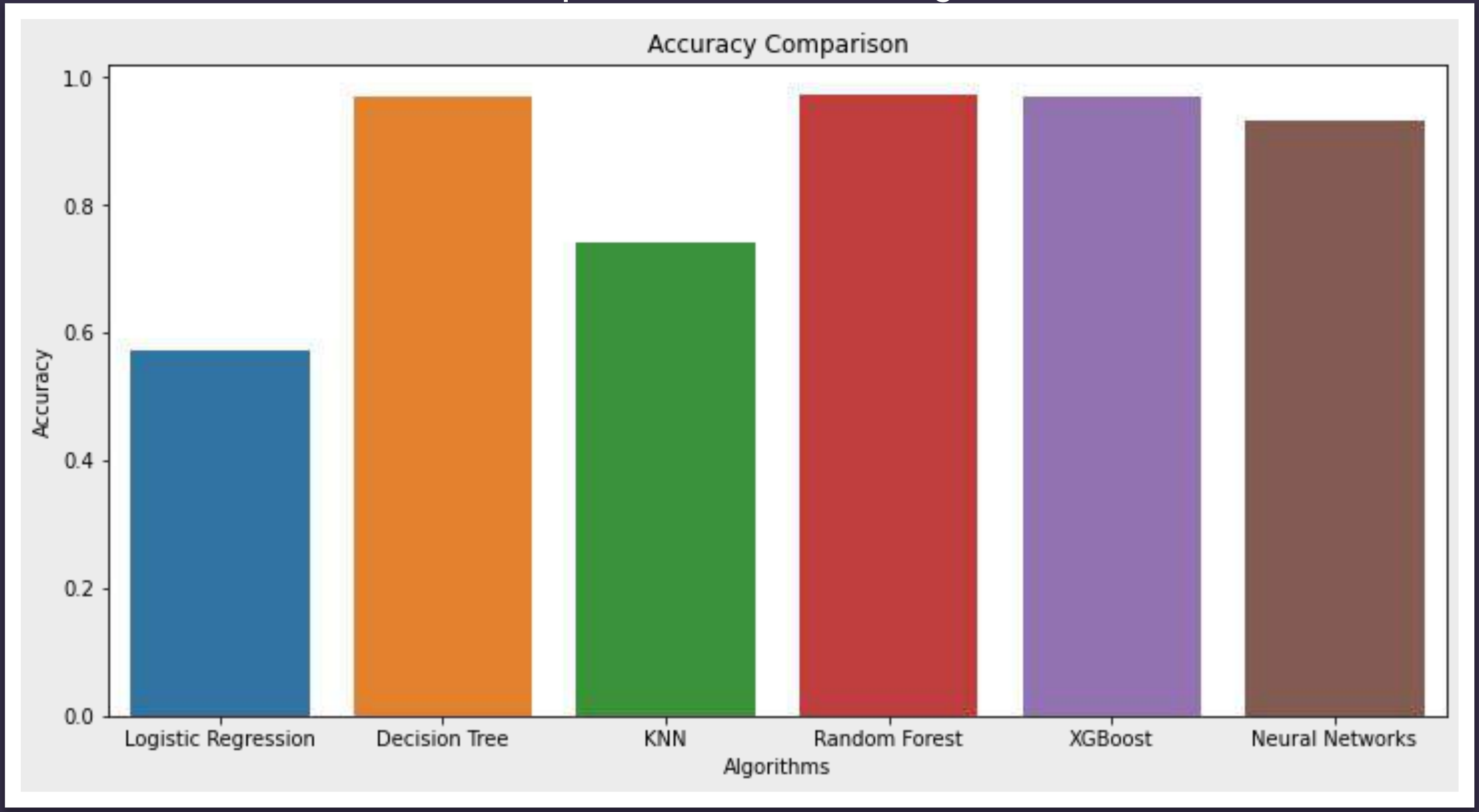
EVALUATION

KNN model under performs as compared to other algorithms because it uses euclidean distances to cluster the data and in the case of high dimension data clustering becomes difficult since most of the vector data become equidistant to the search query vector.

Decision tree performs good in because it grows exponentially for each level and it is easily able to handle medium to high dimensional data.

Similarly random forest and XG boost are uses decision tree as on of the component so they perform good for high dimensional data.

Neural Networks algorithm gives good accuracy but less than tree algorithms because it requires high amount of data to train and choose best weights for it's neurons.



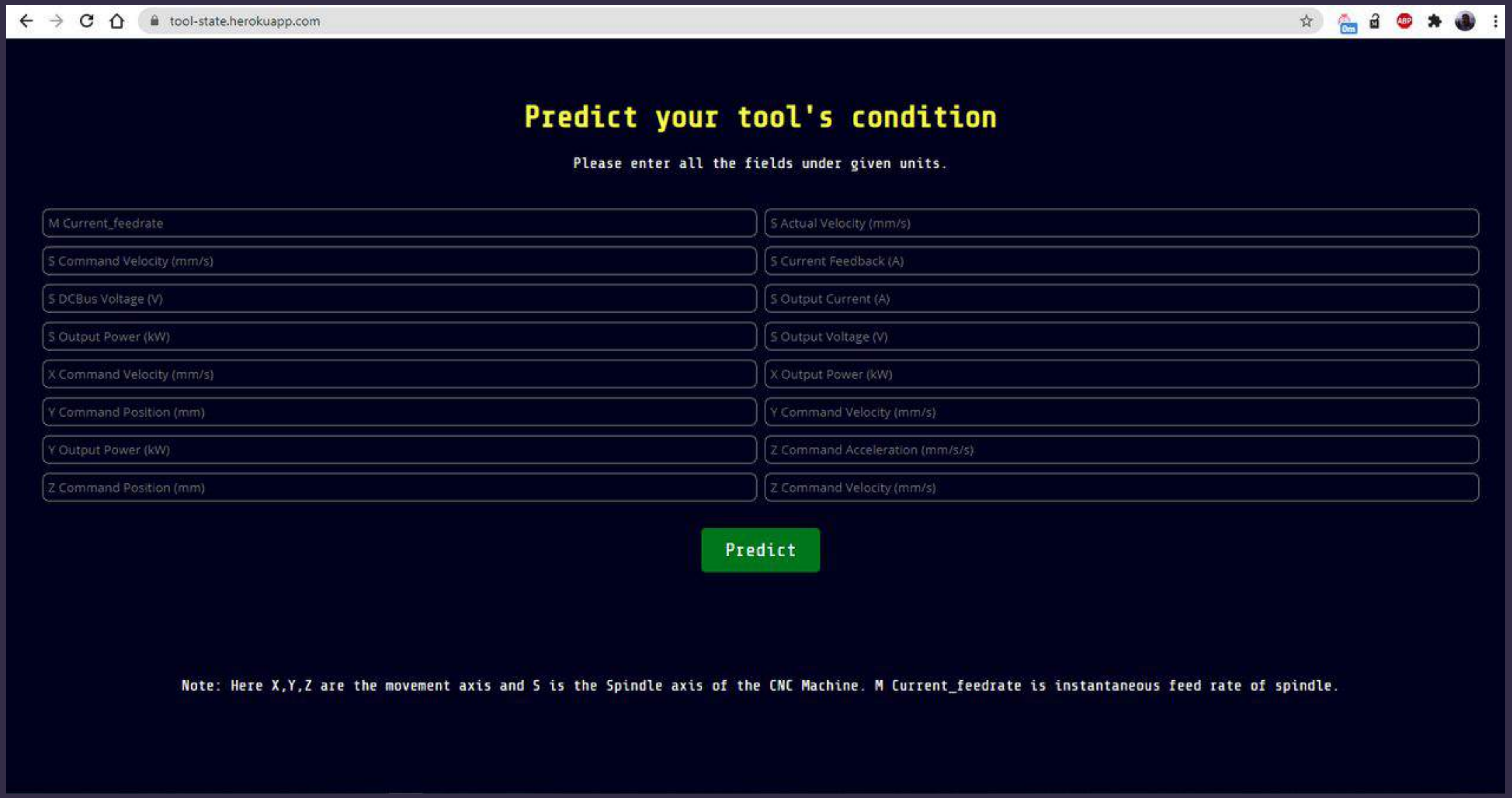
BUILDING API AND WEB-APPLICATION DEPLOYMENT

We choose Random Forest classifier as the prediction model and build an API using Flask. We created a Web-app using HTML and CSS and deployed the website using Heroku web services. The Web app takes 16 raw numerical machine features and predicts the tool state. The Web-App is live at

https://tool-state.herokuapp.com/

Why Random Forest?

Random forest was used as the prediction model because it gave the most accurate result although it is bit slower than other algorithms.


tool-state.herokuapp.com

Predict your tool's condition

Please enter all the fields under given units.

M Current_feedrate	S Actual Velocity (mm/s)
S Command Velocity (mm/s)	S Current Feedback (A)
S DCBus Voltage (V)	S Output Current (A)
S Output Power (kW)	S Output Voltage (V)
X Command Velocity (mm/s)	X Output Power (kW)
Y Command Position (mm)	Y Command Velocity (mm/s)
Y Output Power (kW)	Z Command Acceleration (mm/s/s)
Z Command Position (mm)	Z Command Velocity (mm/s)

Note: Here X,Y,Z are the movement axis and S is the Spindle axis of the CNC Machine. M Current_feedrate is instantaneous feed rate of spindle.

TECHNOLOGY AND FRAMEWORK

- I used python language it is a powerful high-level language and I am well versed with it.
- I used scikit learn library for the development of Machine Learning models as scikit learn provides many useful libraries and has very good documentation of its tools.
- I used Keras library on TensorFlow as Keras is easy to learn and is a very powerful library used for deep learning purposes.
- For the data handling, computation, and visualization purposes Pandas, numpy, seaborn, and matplotlib was used.
- For the development of API, we used the Flask library and wrote the code in python. Flask is a well-known technology and helps in the creation of robust applications very easily. It also provides good documentation.
- The website was written using HTML and styled with CSS. The website is hosted on Heroku because it provides a sufficient amount of memory for free and easy hosting.
- I used Google colaboratory as our platform so that we can use it's fast and free TPU accelerators which

CONCLUSION

From the above use-case, we determined the important features which affect the tool state. The models were trained on a sufficient amount of data and so the results are reliable and can be used during CNC experiments to protect the cutting tool. The cutting tool is made of high-speed steel or carbides and is therefore very costly. We also determined the accuracy of algorithms and which of them will be most efficient for Web-app creation.