BIOSOC PROJECTS

BIOBYTES

MIDTERM EVALUATION
GROUP - 1

MENTORS
NISCHAY PATEL AND VAMSI

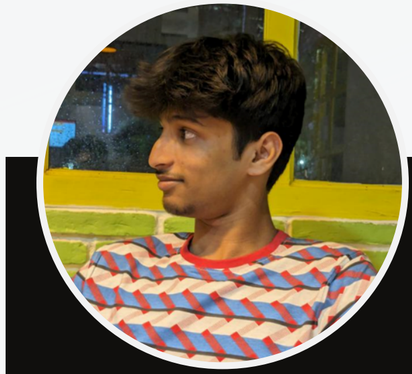
GROUP MEMBERS
JAIVEER SABHARWAL
HIMANSHU DUGAR
ENOCH EMMANUEL

CONTENT

GROUP

Jaiveer Sabharwal

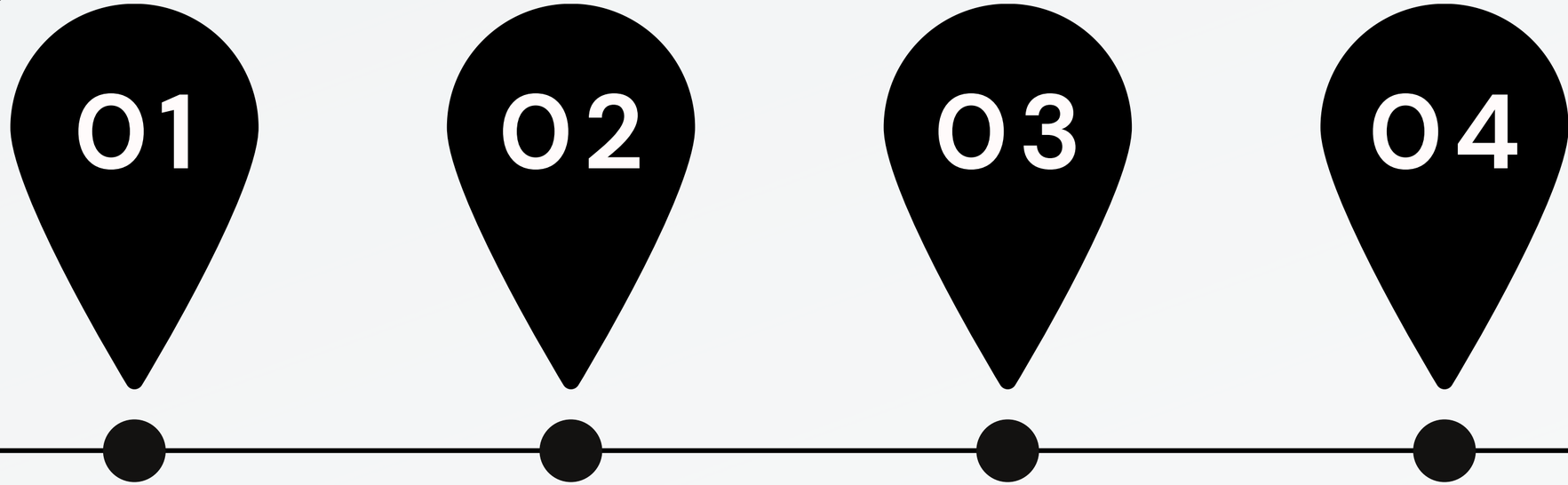
Himanshu Dugar



Enoch Emmanuel

TOPICS COVERED

Our progression has been organic. Starting with the basics of Git/GitHub and Python, we moved to data analysis using Matplotlib and Seaborn. Then we learnt about what ML is and various algorithms associated with it with a focus on the applicative side of Machine Learning. Working with many kinds of datasets from Kaggle has helped develop a holistic understanding of available datasets.



WEEK 1

Configuring GIt
Setting up GitHub
Command Line Interface
Google Collab
Kaggle
Numpy operations
Assignment on NumPy

WEEK 2

Pandas library
Data visualization
Matplotlib
Seaborn
Application on house pricing dataset
Introduction to ML
Assignment on Data analysis

WEEK 3

KNN classifier
Decision Trees
Random Forest
Gradient Boosting
Logisitc Regression
Linear Regression
Validation Techniques
Application on iris dataset

WEEK 4

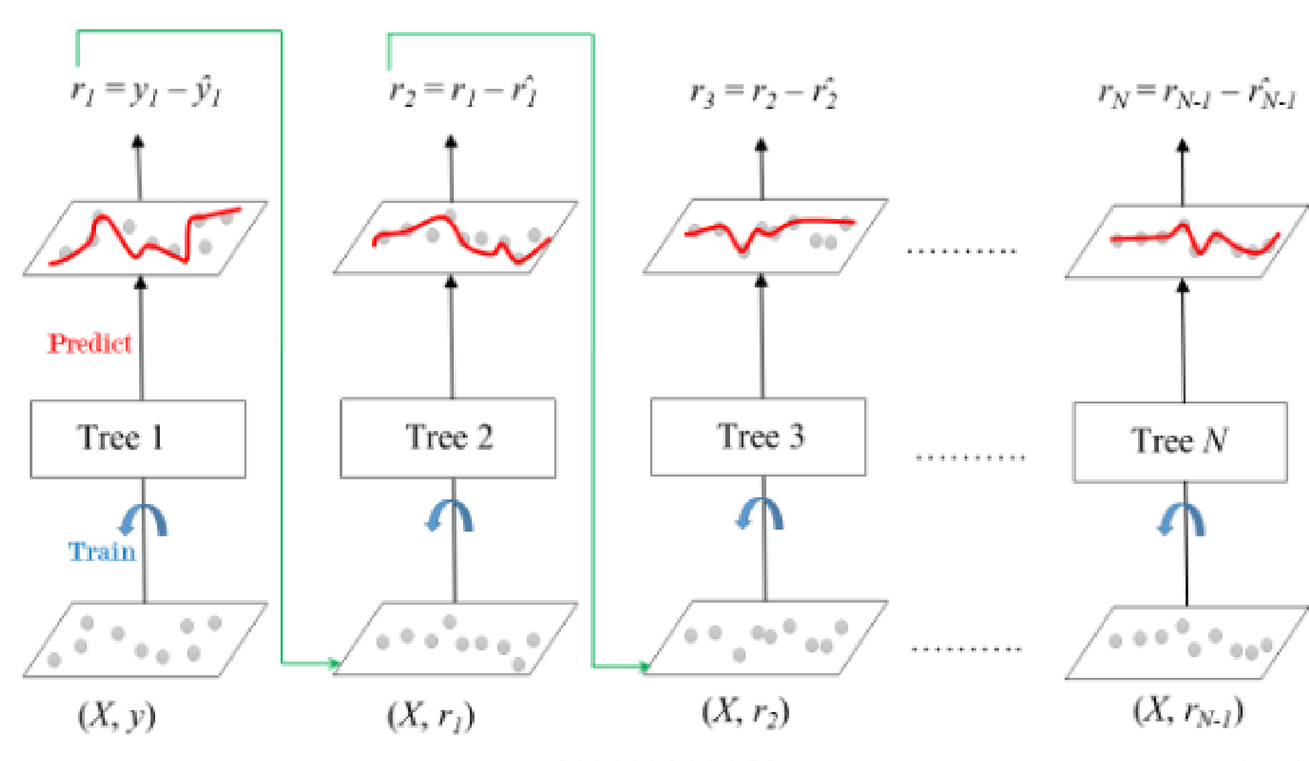
(ongoing)
Confusion Matrix
Naive Bayes Classifier
Entropy in Decision Trees

GRADIENT BOOSTING

Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent.

In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient. The predictions of the new model are then added to the ensemble, and the process is repeated until a stopping criterion is met.

- The ensemble consists of M trees. Tree1 is trained using the feature matrix X and the labels y. The predictions labeled y1(hat) are used to determine the training set residual errors r1.
- Tree2 is then trained using the feature matrix X and the residual errors r1 of Tree1 as labels. The predicted results r1(hat) are then used to determine the residual r2.
- The process is repeated until all the M trees forming the ensemble are trained.



- There is an important parameter used in this technique known as Shrinkage. Shrinkage refers to the fact that the prediction of each tree in the ensemble is shrunk after it is multiplied by the learning rate (eta) which ranges between 0 to 1.

- Shrinkage essentially determines how much weight each tree carries.

- A shrinkage parameter of 1 would mean we are simply adding the residual to the mean value, leading to the original value. This causes overfitting.

- A low shrinkage parameter would mean that the trees are not effective enough.

- The final predicted output is given as

```
y(pred) = y1 + (eta *  r1) + (eta * r2) + ..... + (eta * rN)
```

Mathematics of Gradient Boosting

Input:

- **Data:** $\{(x_i, y_i)\}_{i=1}^{n}$ - a dataset of n samples where x_i are the features and y_i are the target values.
- **Loss Function:** $L(y_i, F(x))$ - a differentiable loss function that measures the difference between the true values y_i and the model's predictions $F(x)$.

Step 1: Initialize the Model

- **Initial Model:**

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

Start by finding a constant γ that minimizes the loss function for th entire dataset. This step essentially initializes the model $F_0(x)$ with simple prediction (e.g., the mean of y_i in case of mean squared error).

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Step 3: Output $F_M(x)$

Mathematics of Gradient Boosting

Step 2: Iterative Boosting Process

For $m = 1$ to M (where M is the total number of boosting iterations):

Step 2(A): Compute Residuals

- **Residuals:**

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n$$

Compute the residuals r_{im}, which represent the negative gradient of the loss function with respect to the current model $F_{m-1}(x)$. The residuals indicate how much the current model's predictions need to be adjusted.

Step 2(B): Fit Regression Tree

- **Fit Tree:** Fit a regression tree to the residuals r_{im} to create terminal regions R_{jm} for $j = 1, \ldots, J_m$. A regression tree is fitted to the residuals, partitioning the data into J_m terminal regions R_{jm}.

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for m = 1 to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,\ldots,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Step 3: Output $F_M(x)$

Mathematics of Gradient Boosting

Step 2(C): Compute Region Predictions

- **Region Predictions:** For $j = 1, \ldots, J_m$, compute

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

For each terminal region R_{jm}, compute a value γ_{jm} that minimizes the loss function for the data points in that region. This value γ_{jm} is the adjustment to be made in that region.

Step 2(D): Update the Model

- **Model Update:**

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$

Update the model by adding the adjustments γ_{jm} scaled by a learning rate ν. The indicator function $I(x \in R_{jm})$ is 1 if x is in region R_{jm} and 0 otherwise.

Mathematics of Gradient Boosting

Step 3: Output the Final Model

- **Final Model:** $F_M(x)$ After M iterations, the final model $F_M(x)$ is obtained, which is the sum of the initial model and the adjustments made in each iteration.

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable **Loss Function** $L(y_i, F(x))$

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: for $m = 1$ to M:

(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$ for $i = 1,...,n$

(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

Step 3: Output $F_M(x)$

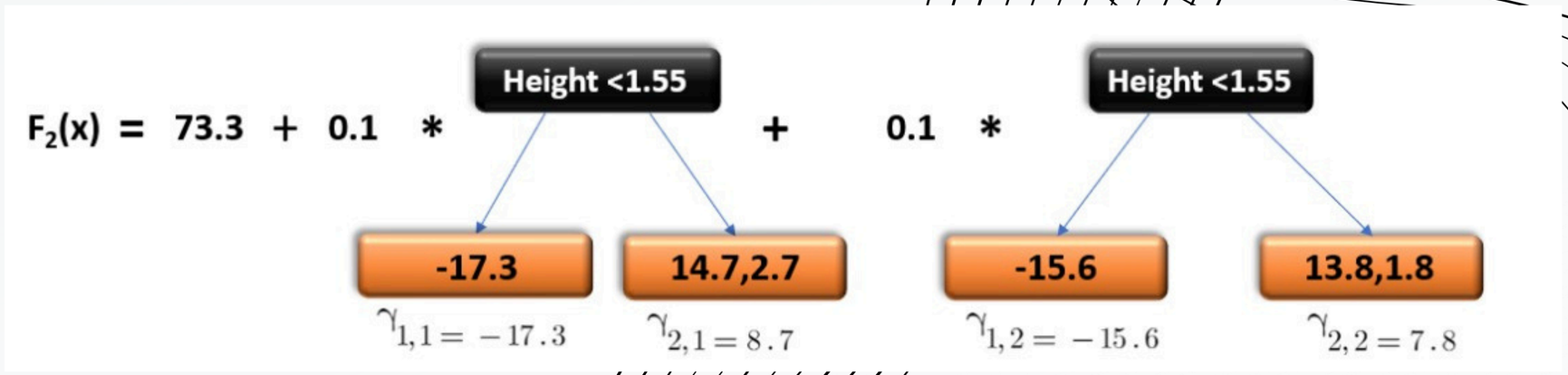
Code for Gradient Boosting

```
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier(n_estimators=300, learning_rate=0.05, random_state=100,max_features=5)
gbc.fit(train_X, train_y)
pred_y = gbc.predict(test_X)
```

Visualization using an example

Row No.	Cylinder Number	Car Height	Engine Location	Price	Prediction 1	Residual 1
1	Four	48.8	Front	12000	14500	-2500
2	Six	48.8	Back	16500	14500	2000
3	Five	52.4	Back	15500	14500	1000
4	Four	54.3	Front	14000	14500	-500



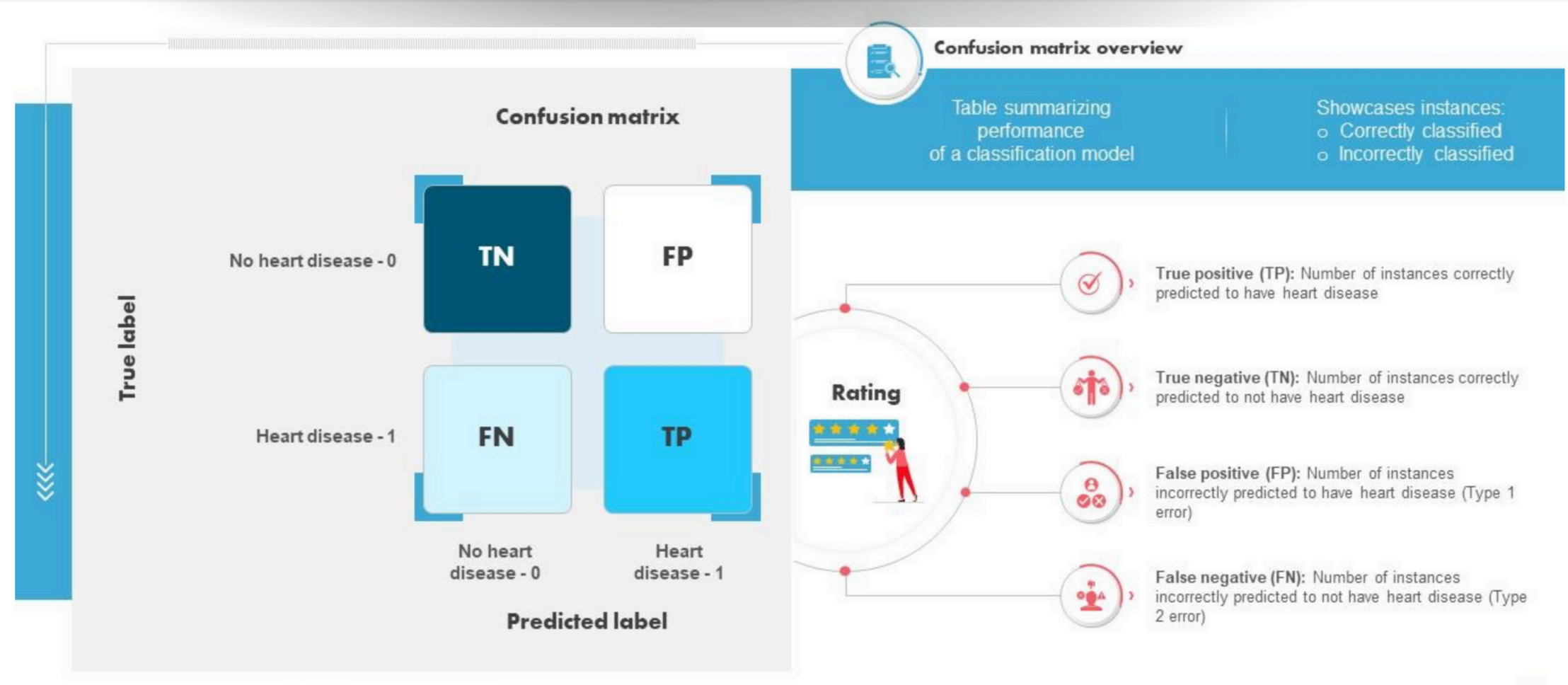
CONFUSION MATRIX

A confusion matrix is a matrix that summarizes the performance of a machine learning model on a set of test data. It is a means of displaying the number of accurate and inaccurate instances based on the model's predictions.

It offers a thorough analysis of true positive, true negative, false positive, and false negative predictions, facilitating a more profound comprehension of a model's recall, accuracy, precision, and overall effectiveness in class distinction. When there is an uneven class distribution in a dataset, this matrix is especially helpful in evaluating a model's performance beyond basic accuracy metrics..

The matrix displays the number of instances produced by the model on the test data.

- True positives (TP): occur when the model accurately predicts a positive data point.
- True negatives (TN): occur when the model accurately predicts a negative data point.
- False positives (FP): occur when the model predicts a positive data point incorrectly.
- False negatives (FN): occur when the model mispredicts a negative data point.



CONFUSION MATRIX :METRICS

Accuracy

It is the ratio of Total correct instances to the total instances.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

Precision

Out of all the positive predicted, what percentage is truly positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall

Out of the total positive, what percentage are predicted positive.

$$Recall = \frac{TP}{TP + FN}$$

F 1 – Score

It is the harmonic mean of precision and recall. It takes both false positive and false negatives into account. Therefore, it performs well on an imbalanced dataset.

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

$$= \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

PROSPECTS OF ML

STROKE DIAGNOSIS

Machine learning uses pattern recognition to help diagnose, treat, and predict complications in various neurological diseases. Acute Ischemic Stroke (AIS) treatment has experienced significant advancement. Machine learning algorithms are now being used to predict motor deficits in stroke patients. The most commonly used methods are Support Vector Machines (SVM), and 3D Conventional Neural Network (CNN).

DRUG DICOVERY

Machine learning is widely used in the early stages of drug discovery processes. Some of the research and development technologies used include precision medicine and next-generation sequencing. They have proven to be helpful in finding alternative options for multifactorial disease therapy.

ANTI-CANCER DRUG EFFICACY PREDICTION

This enhances the efficacy of treatment and reduces the suffering and misery experienced by non-responders. Thus, there is an immediate need to find reliable biomarkers (i.e., genes or proteins) that can precisely predict which patients respond best to which medications. s.

THANK YOU