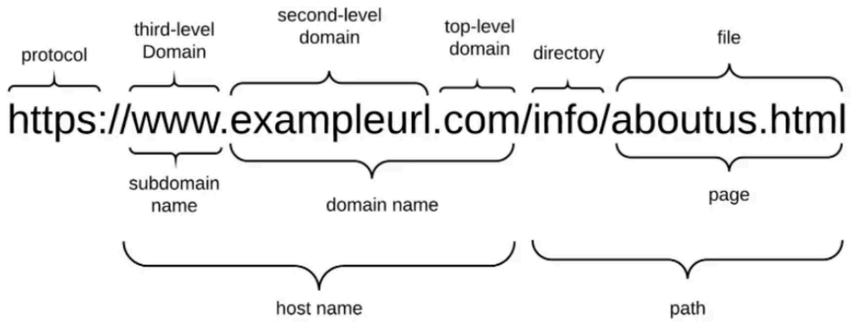
IIT Guwahati

PHISHING URL DETECTION

CS361-MACHINE LEARNING



Team Members (SARS):

Aditya Gupta - 210101010
Riya Mittal - 210101089
Shally Kandoi - 210101096
Shivam Agrawal - 210101119

TARGET PROBLEM

Enhancing Online Security Through Machine Learning Phishing Detection

Phishing Threat:

- Global menace causing financial losses and data breaches.
- Existing anti-phishing methods, including human-verified URL blacklists, often struggle to adapt to new and evolving phishing attempts.

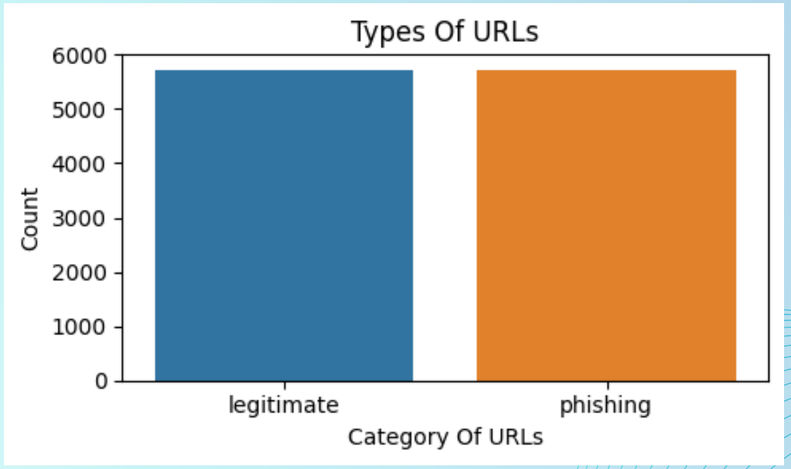
Project Objective:

- Develop effective phishing detection models.
- Utilize machine learning: SVM, Decision Trees, kNN.
- **Goal**: compare and identify the most accurate and efficient approach for detecting phishing websites.
- Enhance online security and mitigate cyber threats.

DATASET OVERVIEW

Description of Dataset:

- Utilized a comprehensive Kaggle dataset comprising both legitimate and phishing URLs.
- Originally contained approximately 11430 rows and 89 columns.
- Streamlined to 25 columns for analysis purposes.



Data Preprocessing:

- Refers to techniques and procedures used to clean, transform, and prepare raw data for analysis.
- Included elimination of inconsequential features with only one value across the dataset or redundant features.
- Aimed to ensure data quality and relevance for subsequent analysis.

FEATURES IN THE DATASET:

URL-Based Features:

- Characteristics extracted directly from URLs.
- Include features like embedded domain, IP address usage, URL length, and presence of sensitive words.

Page-Based Features:

- Attributes obtained from webpage properties.
- Examples include global PageRank and Google Index, indicating webpage reliability.

Domain-Based Features:

- Attributes derived from domain properties.
- Consist of features such as domain registration length, domain age, and web traffic, reflecting the legitimacy of the domain.

HTML-Based Features:

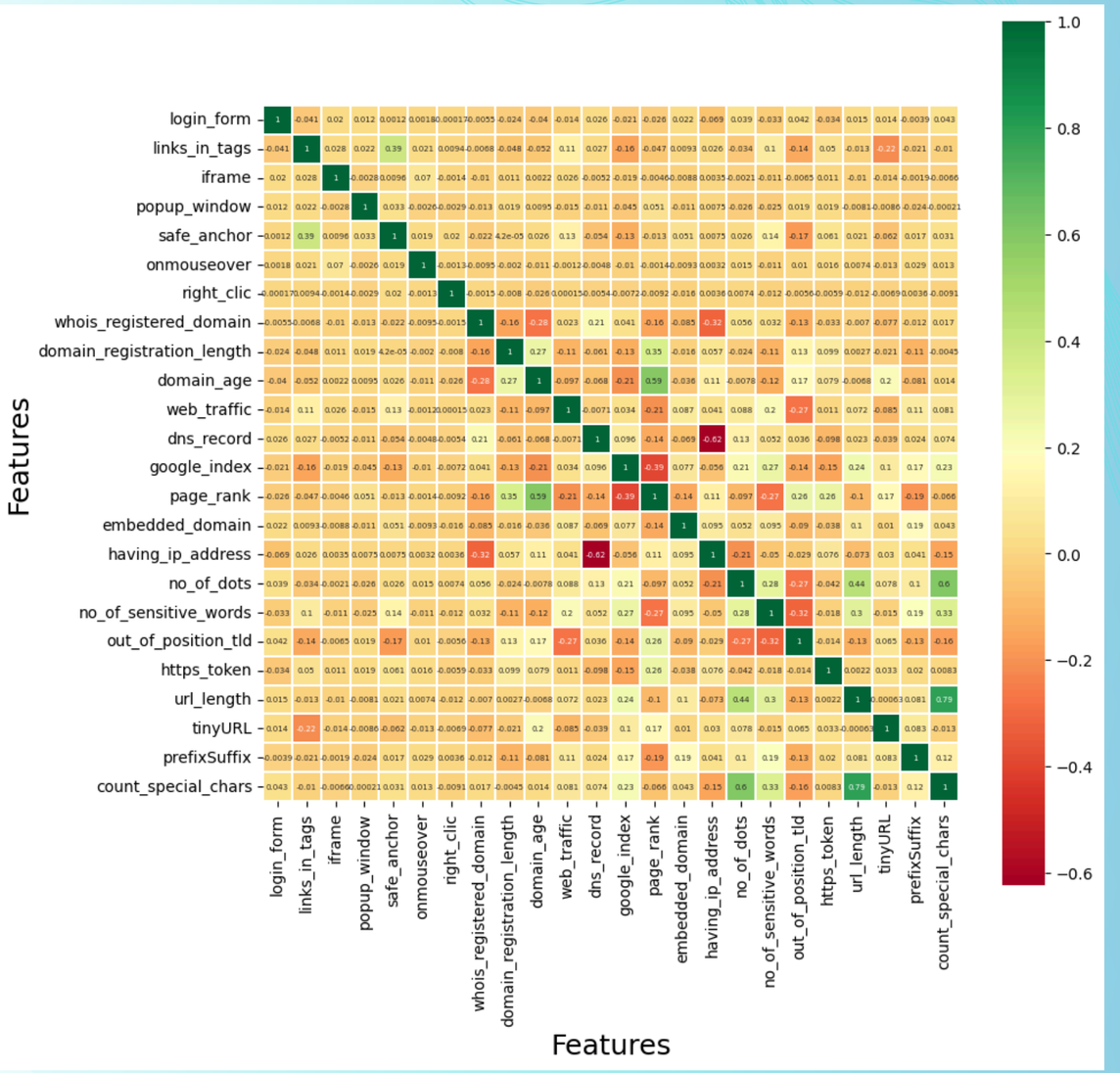
- Attributes extracted from the content of HTML pages.
- Include features like presence of login forms, links in tags, and usage of iframes, indicative of potential phishing attempts.

EXPLORATORY DATA ANALYSIS:

- EDA involves analyzing and visualizing data to understand its characteristics, patterns, and relationships.
- Features categorized as categorical and continuous for analysis.
- Utilized histograms, box plots, count plots, and heatmaps to examine features.

Insights from Heatmap:

- Heatmap analysis revealed weak correlations among features.
- Suggests feature independence and provides insights for model development.



SUPPORT VECTOR MACHINE (SVM)

We used C-support vector classification for the classification of phishing and benign URLs. It is ideal for our case as we only have 2 classes. Given training vectors xi ∈ R n, i = 1, . . . , l, in two classes, and an indicator vector y ∈ R l such that yi ∈ {1, −1}, C-SVC solves the following primal optimization problem

$$\begin{aligned} &\min_{w,b,\xi} \tfrac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ &\text{subject to} \quad y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ &\xi_i \geq 0, \quad i = 1, \ldots, l \end{aligned}$$

where ϕ(xi) maps xi into a higher-dimensional space and C > 0 is the regularization parameter. Due to the possible high dimensionality of the vector variable w, usually we solve the dual problem.

$$\begin{aligned} &\max_{\alpha} L_{\text{dual}} = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ &\text{subject to } 0 \leq \alpha_i \leq C, \text{ and } \textstyle\sum_{i=1}^{n} \alpha_i y_i = 0 \\ &\text{where } K(x_i, x_j) \text{ is the kernel function.} \end{aligned}$$

KERNEL FUNCTIONS

- **Linear Kernel**
- **Polynomial Kernel**
- **Sigmoid Kernel**
- **Gaussian Kernel**
- **Laplace Kernel**

$$K_{\text{linear}}(x, z) = x^T z$$

$$K_{\text{poly}}(x, z) = (c + x^T z)^q, c = 75, q = 2$$

$$K_{\text{sigmoid}}(x, z) = \tanh(\alpha x^T z + c), c = 75, \alpha = 0.1$$

$$K_{\text{rbf}}(x, z) = \exp\left\{ -\frac{||x - z||^2}{2\sigma^2} \right\}, \sigma = 1$$

$$K_{\text{laplace}}(x, z) = \exp\left(-\frac{\|x - z\|}{\sigma} \right), \sigma = 1$$

OPTIMIZATION ALGORITHMS

1 Gradient Descent

- Well-suited for convex optimization problems and is ideal for maximizing the quadratic objective function of the dual SVM formulation.
- Does not work with large datasets in SVM due to the large memory requirements.
- To address this challenge, we limit its usage to a training size of 0.4 with our dataset.

2 Mini-Batch Gradient Descent

- Computes the gradient using a smaller subset of the data called a mini-batch.
- This reduces the memory requirement and allows training SVM with a train size of 0.2 leading to higher accuracy.

3 PEGASOS with Stochastic Gradient Descent

- Primal Estimated sub-GrAdient SOlver for SVM is an effective iterative algorithm for solving the optimization problem of linear SVM.
- It updates the weights on the basis of a single training example in each epoch.
- It's rapid convergence allowed us to train linear SVM for a large number of iterations which allowed us to improve the accuracy of linear SVM.
- Cannot be used for non-linear SVMs.

SMO : COORDINATE ASCENT ON TWO COORDINATES

SMO or Sequential Minimal Optimization chooses to solve the smallest possible optimization problem at every step. For the standard SVM QP problem, the smallest possible optimization problem involves two Lagrange multipliers, because the Lagrange multipliers must obey a linear equality constraint.

$$\eta = K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2)$$

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta}$$

$$E_i = u_i - y_i$$

$$\alpha_2^{new,clipped} = min(R, max(L, \alpha_2^{new}))$$

$$\alpha_1^{new} = \alpha_1 + y_1 y_2(\alpha_2 - \alpha_2^{new,clipped})$$

- L and R are the lower and upper bounds on the value of the second Lagrange multiplier.
- We can use different heuristics for choosing the two Lagrange multipliers to optimize in each step.
- We used an optimized SMO algorithm which achieves faster convergence.
- The outer loop keeps alternating between single passes over the entire training set and multiple passes over the non-bound subset until the entire training set obeys the KKT conditions.
- The second Lagrange multiplier is chosen to maximize the size of the step taken during joint optimization.

RESULTS

Kernel function used	Accuracy	LIBSVM Accuracy
Linear	0.87	0.88
Polynomial	0.73	0.91
Gaussian	0.89	0.92
Sigmoid	0.78	0.83
Laplace	0.72	-

Table 1. Comparison of SVM Accuracy with library implementation

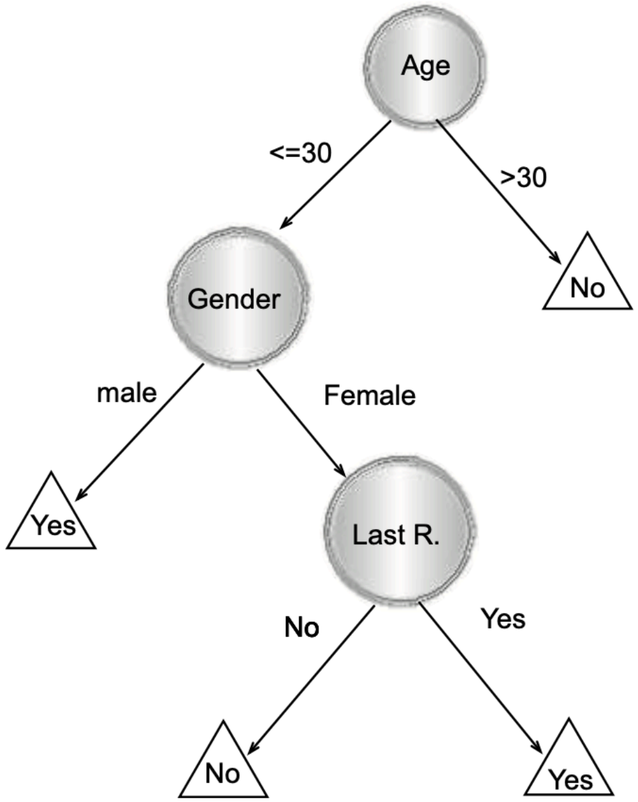
RBF kernel is giving the best results because it works well with non-linear and high-dimensionality datasets similar to ours. The RBF kernel implicitly maps the input features into a higher-dimensional space, where the data becomes more separable. The RBF kernel's smoothness and local nature make it inherently robust to noise in the data. This robustness helps prevent overfitting and allows the model to generalize well to unseen examples.

We compare the accuracy of the RBF and Polynomial kernel by using different optimization algorithms. A significant difference between the accuracies using different optimization algorithms is seen in case of polynomial kernel. The optimized variant of SMO implemented by us achieves the highest accuracy.

Optimization technique	RBF	Polynomial
Gradient Descent	0.89	0.73
Mini-Batch Gradient Descent	0.89	0.86
SMO	0.90	0.88

Table 2. Comparison of Optimization Algorithms using different kernels

DECISION TREES



- A decision tree is a non-parametric supervised learning algorithm utilised for classification and regression tasks.
- It has a hierarchical tree structure consisting of a root node, branches, internal nodes and leaf nodes.
- It is a classifier expressed as a recursive partition of the instance space. In a decision tree, each internal node splits the instance space into two or more sub-spaces according to a specific discrete function of the input attribute values.

Advantages of DT :-

- Decision trees are self-explanatory
- A decision tree does not require normalization and scaling of data.

SPLITTING CRITERIA

Information Gain

1

$$InformationGain(a_i, S) = Entropy(y, S) - \sum_{v_{i,j} \in dom(a_i)} \frac{|\sigma_{a_i = v_{i,j}} S|}{|S|} \cdot Entropy(y, \sigma_{a_i = v_{i,j}} S)$$

where:

$$Entropy(y, S) = \sum_{c_j \in dom(y)} -\frac{|\sigma_{y = c_j} S|}{|S|} \cdot \log_2 \frac{|\sigma_{y = c_j} S|}{|S|}$$

Gini Impurity

2

$$Gini(y, S) = 1 - \sum_{c_j \in dom(y)} \left(\frac{|\sigma_{y = c_j} S|}{|S|} \right)^2$$

Consequently the evaluation criterion for selecting the attribute a_i is defined as:

$$GiniGain(a_i, S) = Gini(y, S) - \sum_{v_{i,j} \in dom(a_i)} \frac{|\sigma_{a_i = v_{i,j}} S|}{|S|} \cdot Gini(y, \sigma_{a_i = v_{i,j}} S)$$

Gini Impurity vs Infomation Gain

Both metrics have their strengths and weaknesses, and which one is better to use depends on the specific dataset and problem at hand. In general, Information Gain tends to work well when the target variable has many possible values, whereas Gini Impurity tends to work well when the target variable has only a few possible values. Additionally, Gini Impurity can be more efficient to calculate than Information Gain, especially when dealing with large datasets.

Information Gain Ratio

3

$$GainRatio(a_i, S) = \frac{InformationGain(a_i, S)}{Entropy(a_i, S)}$$

DECISION TREES VARIANTS

- **ID3(Iterative Dichotomiser 3)**

1. First Decision Tree Algorithm
2. Splitting Criteria:- Information gain
3. Does not handle numeric attributes (does not work on our dataset)
4. It splits the data into multiple on the basis of different categories

- **C4.5**

1. Advancement of ID3
2. Splitting Criteria:- Information gain ratio
3. Can handle numeric attributes

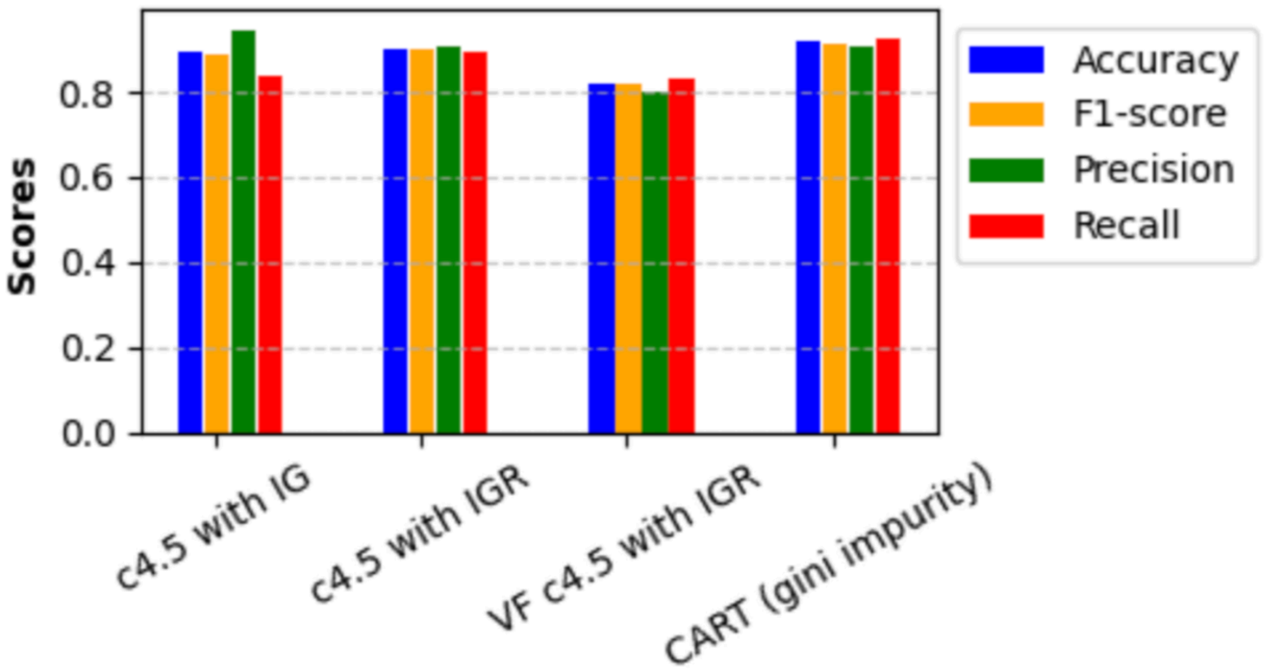
- **Very Fast C4.5**

1. Variant of C4.5
2. Splitting on continuous features happens at the mean or median
3. Avoids sorting process
4. Little Less Accuracy, but model training is very fast

- **CART(Classification and Regression Tree**

1. It constructs binary trees, where each node has exactly two outgoing edge
2. Splitting Criteria:- Gini Impurity
3. Can handle numeric attributes

RESULTS



- CART outperformed the rest of the variants
- As can be seen, using the VF C4.5 algorithm gives a lower accuracy, but training was very fast in this case.
- The results produced by our implementation are almost the same as that of sci-kit models.

C4.5, employing Information Gain Ratio (IGR), outperforms traditional Information Gain due to its ability to mitigate bias towards attributes with numerous distinct values. However, directly applying IGR can lead to skewed results favouring attributes with small denominators, resulting in lower accuracy (0.60). To address this, I calculate information gain for all attributes and only consider those with gains at least as high as the average when selecting the best attribute based on gain ratio

Model	Accuracy	Accuracy
	My Model	sklearn
C4.5		0.908 *(criterion='entropy')*
Inf Gain	0.89	
IG Ratio	0.902	
VF IGR	0.817	
CART	0.917	0.92*(criterion='gini')*
RF	0.94	0.945
Bagging	0.913	0.925

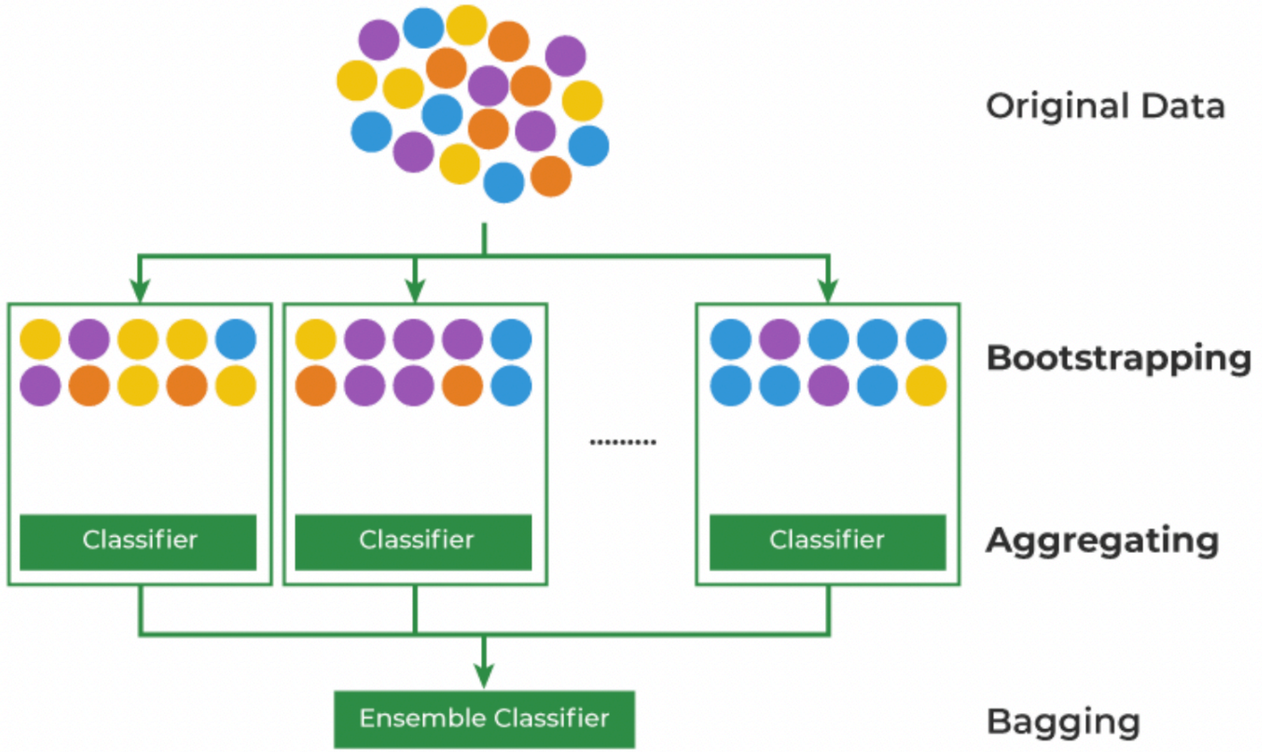
My results vs Scikit reults

BAGGING CLASSIFIER

is a type of ensemble learning

What is Ensemble learing?

Ensemble learning, inspired by the "wisdom of crowds," combines multiple models to improve predictions together. Individual weak models (also known as base learners) , which may suffer from high variance or bias, are strengthened when aggregated, resulting in a better model, reducing bias and variance



1 Bootstrapping

Bootstrap Sampling randomly selects 'n' subsets of the original training data with replacement, promoting diversity among the base models by potentially including repeated samples and omitting others. This mitigates overfitting risks and enhances model accuracy.

2 Parallel training

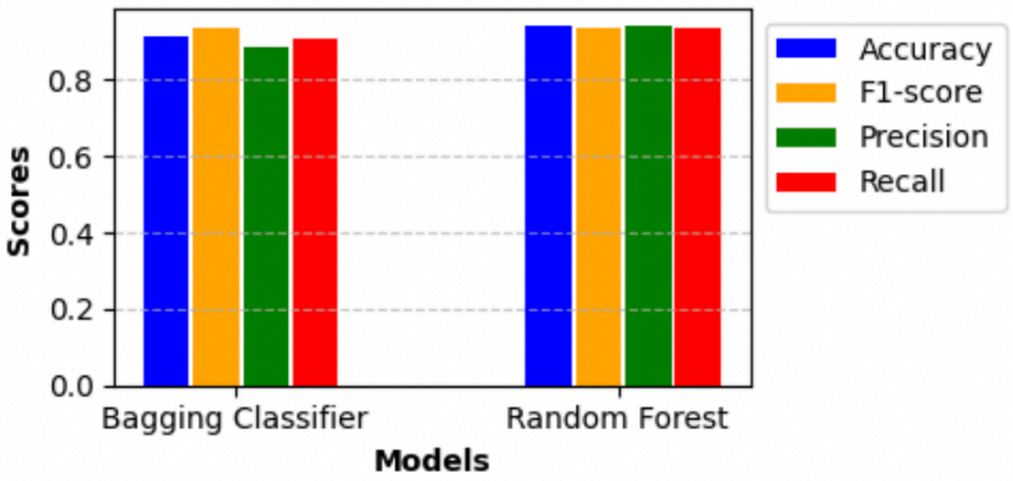
Base models are independently trained using a specific learning algorithm.

3 Aggregation

Predictions on unseen data are made by aggregating the majority votes of all base models trained on different subsets, determining the predicted class label.

RANDOM FOREST

- Random forest is an extension of the bagging classifier with a minor tweak. The main difference between bagging and random forest is how they introduce randomness in the dataset.
- Bagging introduces randomness by sampling with replacement (tree-level sampling), while random forest introduces randomness by using a subset of features (node-level sampling).
- Random forest outperforms the bagging classifier by reducing the problem of correlated trees by using a subset of features at each node. This results in a diverse set of trees that can improve the accuracy of a model by reducing overfitting and increasing the diversity of the model.



Conclusion :-

- Utilizing ensemble techniques results in superior performance compared to individual decision trees, with a maximum accuracy of 0.917 for single decision trees and an improved accuracy of 0.94 achieved with random forests.
- Among ensemble methods, random forests surpass bagging, with the bagging classifier achieving an accuracy of 0.92, which outperforms single decision trees

K NEAREST NEIGHBORS (KNN)

In our implementation of the k Nearest Neighbors (KNN) algorithm, we aimed to classify query points based on the majority class among their k nearest neighbors in the feature space.

Classic kNN

The class label of a query point x_q is determined by the majority class among its k nearest neighbors. The class prediction y_q for x_q is calculated as follows:

$$\hat{y}_q = \arg\max_y \sum_{i=1}^{k} I(y = y_i)$$

where y_i represents the class labels of the k nearest neighbors of x_q, and I() is the indicator function.

VARIANTS OF KNN

We implemented and experimented with several variants of the classic KNN algorithm to explore their performance and capabilities. The various variants are listed below:

Fuzzy KNN

Provides soft classifications by considering the degree of membership to multiple classes.

Weight Adjusted KNN

Enhances KNN by assigning weights to neighbors based on kernel functions, adjusting their influence.

Hassanat KNN

Adapts k based on local data density, utilizing Hassanat distance for potential robustness to outliers.

Ensemble Approach KNN

Combines predictions from multiple KNN classifiers, offering adaptability to varying neighborhood sizes and dynamic parameter selection.

FUZZY KNN

- Fuzzy kNN considers the degree of membership of each data point to multiple classes
- The class membership is defined using a fuzzy membership function.
- The class prediction ŷ_q for x_q is computed as a weighted sum of class memberships:

$$\hat{y}_q = \frac{\sum_{i=1}^{k} \mu_{iq} \cdot y_i}{\sum_{i=1}^{k} \mu_{iq}}$$

Key Features:

1. Offers soft classifications, assigning membership values to each class unlike classic KNN.
2. Employs distance-based weighting, giving higher weights to closer neighbors and lower weights to farther ones when computing class counts.

WEIGHT ADJUSTED KNN

- Weight Adjusted KNN enhances classic K-nearest neighbors by incorporating attribute weighting.
- It assigns weights to neighbors using kernel functions like inverse distance, Gaussian, or Epanechnikov.
- These weights adjust each neighbor's influence in classification.
- Class prediction ŷ_q for x_q is computed as a weighted sum of the class labels of nearest neighbors.

$$\hat{y}_q = \frac{\sum_{i=1}^{k} w_i \cdot y_i}{\sum_{i=1}^{k} w_i}$$

Key Features:

1. Choice of kernel function provides flexibility in adjusting the influence of neighbors based on distance.
2. Attribute weighting in Weight Adjusted KNN boosts robustness to noise.

HASSANAT KNN

- Hassanat KNN adjusts k based on local data density for dynamic neighborhood selection.
- The class prediction is determined by adaptive k-nearest neighbor search within a radius determined by local data density.

Hassanat Distance Calculation:

The Hassanat distance (d_H) is computed as:

$$d_H = || \max(x_1, x_2) - \min(x_1, x_2) ||$$

Here, x_1 and x_2 are the vectors representing the two samples, and || * || denotes the Euclidean norm.

Key Features:

1. Hassanat distance metric offers more informative measurement than Euclidean distance.
2. Potentially more robust to outliers due to consideration of extreme points.

ENSEMBLE APPROACH KNN

- Ensemble Approach KNN combines multiple KNN classifiers using bagging or boosting techniques.
- Each base KNN classifier is trained on a subset of the data or assigned different weights based on performance.
- Final prediction $\hat{y}_q$ is obtained by aggregating individual KNN classifiers' predictions.

Key Features:

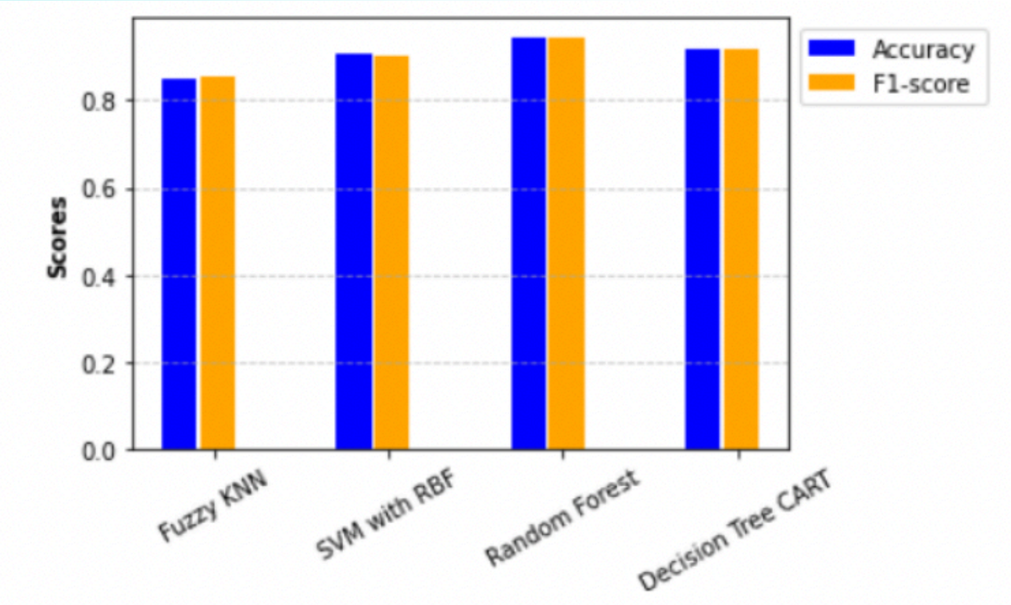
- Combines predictions from various k values via weighted summation, adapting to different local neighborhood sizes.
- Avoids manual selection of a fixed k by considering a range, enhancing adaptability across datasets and tasks.

RESULTS

After conducting a comprehensive analysis of these algorithms, we've obtained performance metrics across various key indicators. Below, you'll find a summary of our findings showcasing the F1 score, recall, precision, and accuracy for each kNN variant.

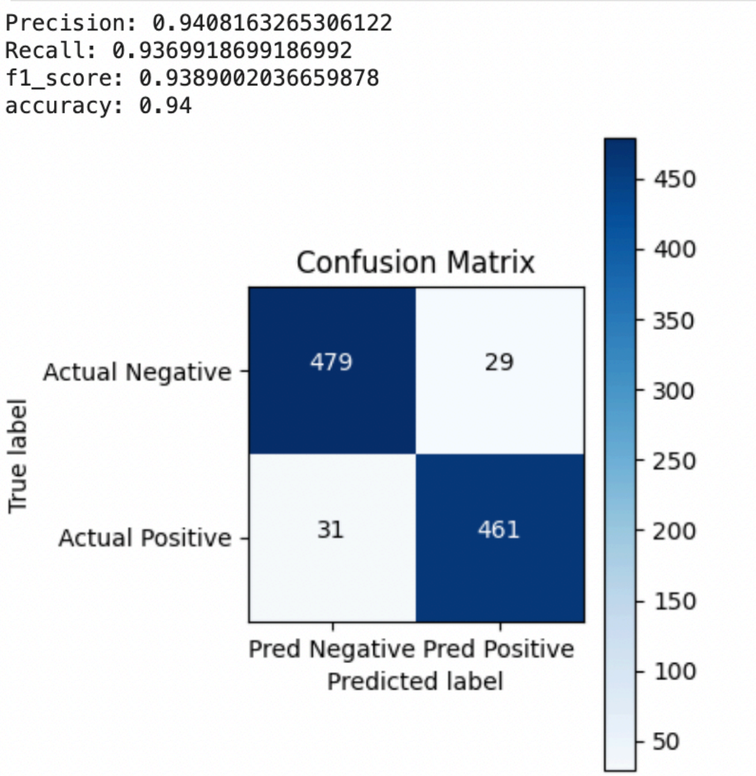
Model	Accuracy	Precision	Recall	F1 Score
Classic	0.812	0.777	0.876	0.824
Standard	0.813	0.849	0.763	0.804
Fuzzy	0.839	0.826	0.860	0.843
Weight Adjusted	0.831	0.817	0.855	0.836
Hasanat	0.812	0.778	0.876	0.823
Ensemble	0.826	0.828	0.824	0.826

COMPARISON OF DIFFERENT MODELS



We can see that Random Forest achieves the highest accuracy and F1 score. Random forests are an ensemble learning method that combines multiple decision trees to make predictions.

By averaging the predictions of multiple trees and introducing randomness in the training process, random forests are able to reduce overfitting and improve generalization performance.



Result of Random Forest