19CSE305 MACHINE LEARNING

Churn Prediction

Team 14

Shankara Narayanan V : CB.EN.U4CSE20656
Shivesh S. : CB.EN.U4CSE20657
Sneha Varsha M : CB.EN.U4CSE20659
Syed Ashfaq Ahmed : CB.EN.U4CSE20665
Tarun Ramaswamy : CB.EN.U4CSE20666

Content

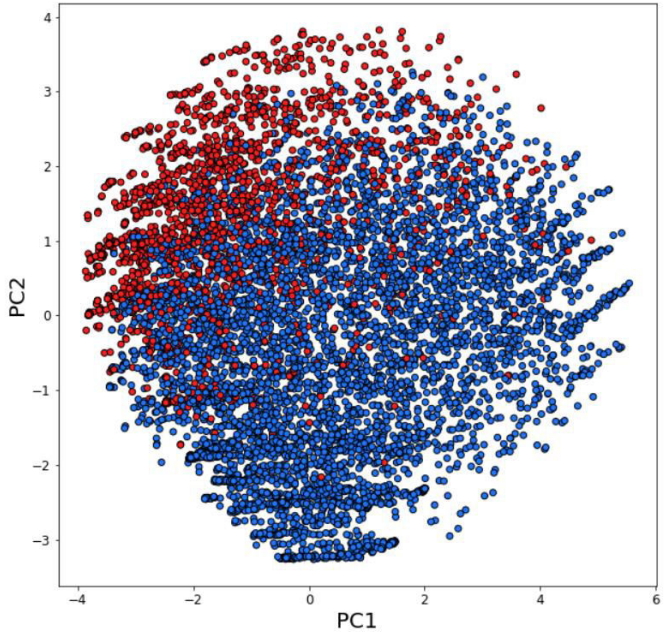
- Principal Component Analysis
- K Means Clustering
- Setting up the Dataset
- Simple Candidate Models
  - Decision Tree
  - Support Vector Machines
  - Gaussian Naive Bayes
  - Multilayer Perceptron
  - Logistic Regression
  - Extra Trees
- Ensembling Techniques
  - Max Voting
  - Bagging
  - Boosting
- GridSearchCV
  - Logistic Regression using GridSearchCV
  - Random Forest using GridSearchCV
- AutoKeras
- Artificial Neural Networks

Principal Component Analysis

Principal component analysis (PCA) is a statistical technique that is used to identify patterns in data. It does this by finding a new set of variables, called principal components, that are linear combinations of the original variables and are ordered so that the first principal component has the highest possible variance, the second principal component has the second highest variance, and so on.

PCA is often used to reduce the dimensionality of large data sets, by identifying the variables that account for most of the variance in the data and keeping only those variables. This can be useful when the data contains a large number of variables that are correlated with each other, as it can help to eliminate redundancy and improve the interpretability of the data.

PCA is an unsupervised learning technique, which means that it does not require a dependent variable or label in order to perform the analysis. Instead, it relies on the relationships between the variables in the data to identify patterns and trends.

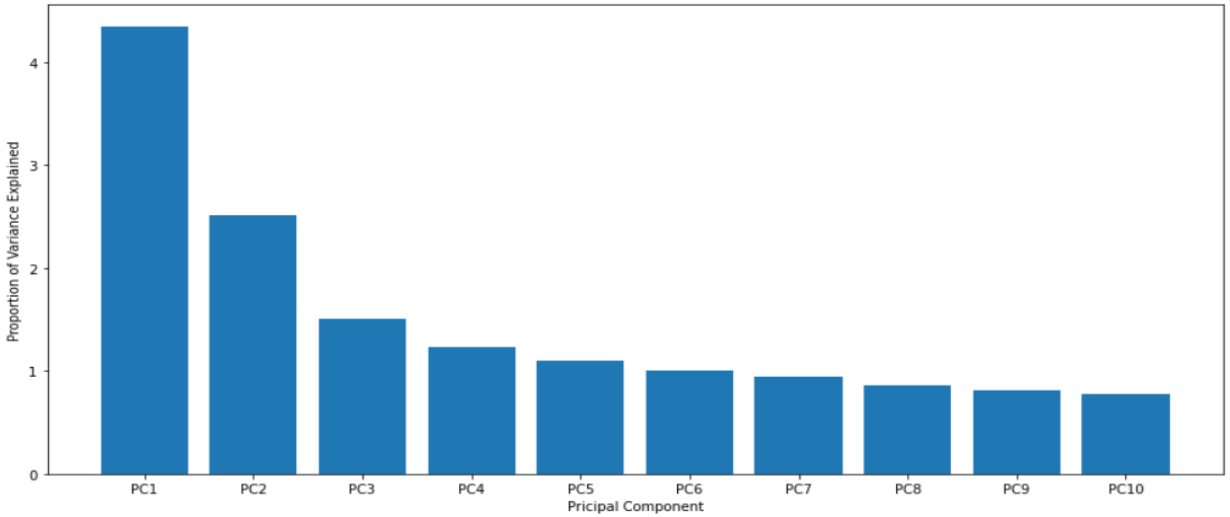


Finding the Principal Component Analysis for the features and taking the two highest values and plotting it using a scatter plot.

As we can see the visualization does not provide a clear and distinct classes.

This could mean that the contribution of PC1 and PC2 maybe very less.

To check this we can check the percentage of variance each Principal Component provides.

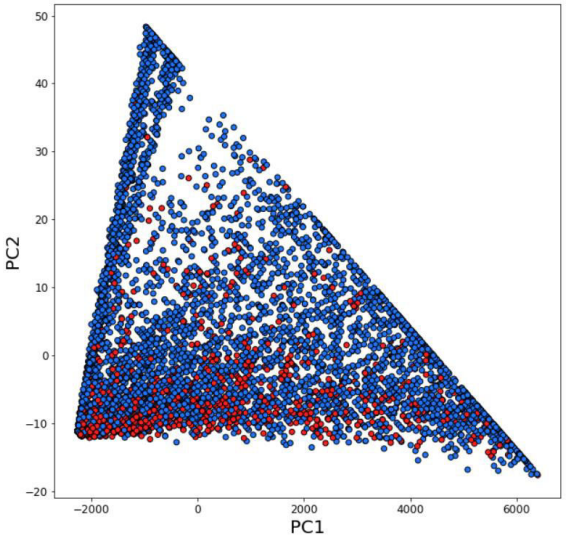


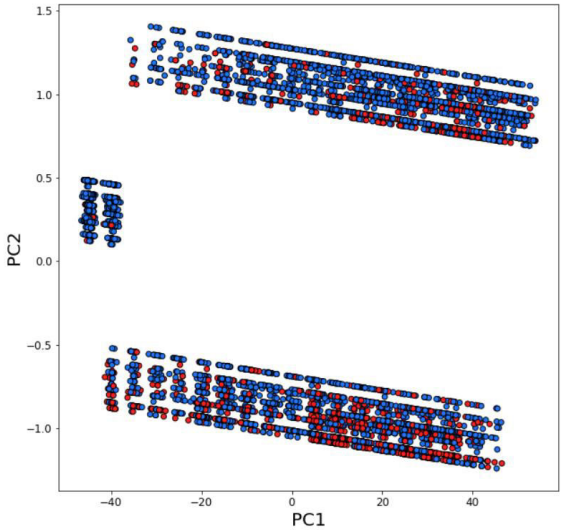
As visible from this graph the proportion of variance obtained from Principal Component 1 and Principal Component 2 is together is 4.5 and 2.5 approximately.
Together their variance contribution is less than 10% hence they do not form a good indication of the variance shown in the dataset.

An Alternative Workaround

Correlation between the features and the churn result is calculated and sorted to give us the features that have the most amount of correlation with Churn predictions.

Principal Component Analysis is applied on the two highest correlating features and plotted to give the following results.





K Means Clustering

K-means clustering is an unsupervised learning algorithm that divides a set of n samples into k distinct clusters based on the mean distance between the points and the center of the cluster.

The algorithm iteratively assigns each sample to the nearest cluster and updates the mean of the cluster until convergence.

The convergence is reached when the assignments do not change or the maximum number of iterations is reached.

The main steps of the K-means algorithm are:

1. Initialize the cluster centers (also known as centroids) randomly or with a predetermined value.
2. Assign each sample to the nearest cluster by calculating the distance between the sample and the centroids.
3. Calculate the new mean of each cluster based on the samples belonging to the cluster.
4. Assign each sample to the new nearest cluster.
5. Repeat steps 3 and 4 until convergence or the maximum number of iterations is reached.

Accuracy obtained from basic clustering and classification: **79.194%**

Accuracy obtained from K Means clustering and classification: **79.004%**

Reasons:

When the clusters are **highly overlapping**:

If the clusters have a high degree of overlap, K-means may not be able to differentiate between them and may give a poor score.

When the data is **noisy**:

K-means is sensitive to noise and may be affected by outlier points. If the data has a lot of noise, the algorithm may give a poor score.

Accuracy of K-Means clustering for different values of K

K - Values	Accuracy
2	46.055
3	37.740
4	16.560
5	12.367
6	23.241
7	19.261
8	8.102
9	19.403

Setting Up the Dataset

Class Imbalances present in the dataset are being removed from the random samples being generated.

The ‘Total Charges’ feature is removed from the dataset due to its high correlation with ‘Monthly Charges’ and ‘Tenure’.

Oversampling:

Random oversampling is a technique used in machine learning to balance the class distribution in a dataset by duplicating samples from the minority class. It is a simple and straightforward method that can be useful when the minority class is underrepresented in the training data.

To perform random oversampling, the algorithm randomly selects samples from the minority class and duplicates them to increase the number of samples. The duplicated samples are added to the training set, which increases the weight of the minority class and helps the model to learn more about it.

Simple Candidate Models

Decision Tree

Decision trees are a type of machine learning algorithm that can be used for both classification and regression tasks.

They work by creating a tree-like model of decisions based on different features, with the goal of predicting the outcome of a new data point.

The tree is made up of nodes that represent a feature to test, and branches that represent the possible outcomes of the test.

Accuracy obtained from Decision Tree Model: **85.915%**

Different Criterion and their respective accuracies

Criterion	Accuracy
gini	85.334
entropy	86.060
log_loss	86.157

K-Nearest Neighbours

K nearest neighbors (KNN) is a non-parametric, lazy learning algorithm.

It works by storing all available cases and classifying new cases based on a similarity measure (e.g. distance functions).

An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

KNN can be used for both classification and regression tasks.

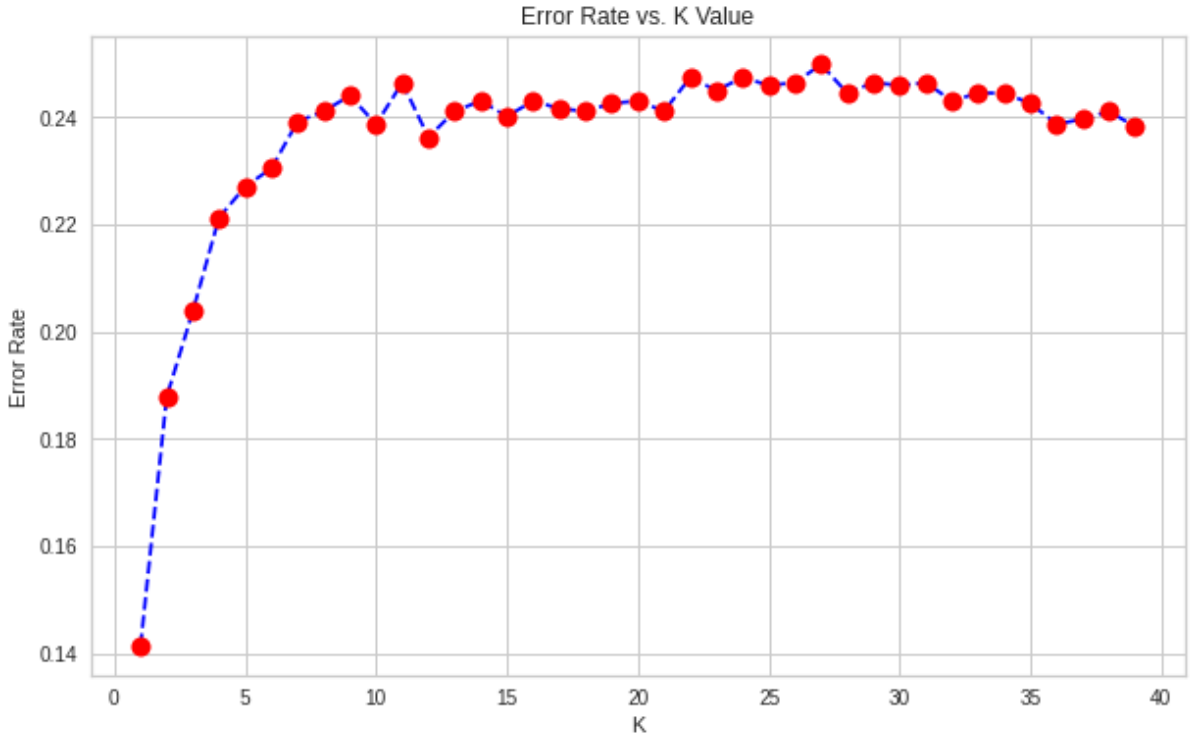
Accuracy obtained from K-Nearest Neighbours for K value 3 : **79.622%**

Accuracy score for different K-values

K - value	Accuracy
1	85.866
2	81.220
3	79.622
4	77.880
5	77.299
6	76.960
7	76.089
8	75.895
9	75.605
10	76.137

K - value	Accuracy
11	75.363
12	76.379
13	75.895
14	75.702
15	75.992
16	75.702
17	75.847
18	75.895
19	75.750
20	75.702

K - value	Accuracy
21	75.895
22	75.266
23	75.508
24	75.266
25	75.411
26	75.363
27	75.024
28	75.557
29	75.363
30	75.411



This graph represents the error rate for different values of k in the K Nearest Neighbour Model

Support Vector Machines

Support Vector Machines (SVMs) are a type of supervised learning algorithm that can be used for both classification and regression tasks.

They work by finding the hyperplane in a high-dimensional space that maximally separates data points of different classes.

SVMs are known for their robustness and ability to handle high-dimensional data and are often used in text classification and image classification tasks.

Accuracy Obtained from Support Vector Machine for RBF kernel and Auto Gamma : **81.22%**

Accuracies obtained by using various kernels

Kernels	Accuracy
linear	75.121
poly	71.491
rbf	74.443
sigmoid	61.907

Gaussian Naive Bayes

Gaussian Naive Bayes is a classification algorithm based on the Bayes Theorem, which states that the probability of a hypothesis (in this case, a class label) given some data is equal to the prior probability of the hypothesis multiplied by the likelihood of the data given the hypothesis.

In the case of Gaussian Naive Bayes, it is assumed that the likelihood of the data is a Gaussian distribution, hence the name.

It is a fast and simple algorithm that is often used as a baseline for classification tasks.

Accuracy obtained from Gaussian Naive Bayes: **74.298%**

Parameters and their corresponding values

Parameters	Values
priors	None
var_smoothing	1e-09

Multi Layer Perceptron

Multi-layer Perceptron (MLP) is a class of feedforward artificial neural network.

It consists of multiple layers of interconnected nodes, where the output of one layer becomes the input of the next layer.

MLP can be used for both classification and regression tasks and is often used for predictive modeling and feature extraction.

MLP requires a large amount of labeled data for training and can be sensitive to the choice of hyperparameters.

Accuracy obtained from Multi Layer Perceptron with 100 hidden layer neurons, Adam optimizer and ReLU activation : **75.992%**

Accuracies obtained for various parameters

Hidden Layer size	Activations	Solvers	Accuracy
100	relu	adam	75.992
200	identity	lbfgs	75.266
256	logistic	sgd	75.799
512	tanh	adam	80.106
1024	relu	sgd	75.944

Logistic Regression

Logistic regression is a supervised learning algorithm used for classification tasks.

It works by predicting the probability of an event occurring based on the values of one or more input features. The predicted probability is then transformed into a binary output, indicating the presence or absence of the event.

Logistic regression is a simple and fast algorithm that is often used as a baseline for classification tasks.

Accuracy obtained from Logistic Regression Model: **75.411%**

Accuracies for different parameters

Solver	Penalty	Accuracy
lbfgs	l2	75.411
liblinear	l1	75.315
newton-cg	l2	75.169
newton-cholesky	None	75.218

Extra Trees

Extra Trees (Extremely Randomized Trees) is a machine learning algorithm that belongs to the family of decision trees. It is an ensemble method that builds multiple decision trees and combines their predictions to make a final decision.

The main difference between Extra Trees and other decision tree algorithms is that it uses randomness to split the features when building the trees.

This makes the trees more diverse and helps to reduce the variance of the model. It also makes the model more robust to overfitting, as the trees are less likely to overfit the training data.

Accuracy obtained from Extra Trees Model with 100 trees and ‘gini’ criterion is **85.237%**

Accuracies obtained for different parameters

N_estimators	Criterions	Accuracy
100	gini	85.237
120	entropy	85.382
140	log_loss	85.382
160	gini	85.237
200	entropy	85.382

Accuracies of Models so far

Model	Accuracy
Decision Tree	85.914
K- Nearest Neighbours	79.622
Support Vector Machine	81.219
Gaussian Naïve Bayes	74.298
Multi Level Perceptron	75.992
Logistic Regression	75.411
Extra Trees Classifier	85.237

Ensembling Techniques

Max Voting

Max voting is a simple ensemble method that is used to combine the predictions of multiple models to make a final prediction. It is typically used in classification problems.

To use max voting, multiple models are trained on the same dataset. When making a prediction, each model will output a probability or a class label. The final prediction is made by selecting the class label or probability that has the highest value across all of the models.

Max voting can be useful in improving the overall performance of the model because it can reduce the impact of any individual model that may be overfitting or underfitting the data.

Accuracy obtained from Max Voting Model: **84.269%**

Stacking

Stacking is an ensemble method that involves training a second model, called a meta-model, to make predictions based on the predictions of several base models.

The base models are trained on the same dataset and then used to make predictions on an additional holdout dataset.

The predictions of the base models on the holdout dataset are then used as inputs to the meta-model, which makes the final prediction.

List of Meta Learners used:

- KNN
- DT
- SVM
- GNB
- ADABOOST
- BAGGING
- MLP
- Logistic Regression

Maximum Accuracy achieved from Stacking model is **88.888%** obtained from the Logistic Regression Meta Learner.

Meta Learners and their Accuracies

Model	Accuracy
KNN	85.024155
DT	87.922705
ETC	87.922705
SVM	87.922705
GNB	83.816425
ADABOOST	88.647343
BAGGING	87.922705
MLP	88.405797
LOG_REG	88.888889

Bagging

Bagging, or bootstrapped aggregation, is a technique in machine learning that involves training several models on different subsets of the training data and then aggregating their predictions.

This can be done with decision trees, artificial neural networks, and other types of models.

Bagging is used to reduce the variance of a model, meaning that it can produce more stable and accurate predictions. It is often used in ensemble methods, where several models are combined to create a more powerful model.

Bagging using Random Forest

Random Forest is an ensemble learning method that uses bagging as a building block. It is a type of decision tree algorithm that is trained on bootstrapped samples of the training data.

During training, each tree in the ensemble is built using a random sample of the training data, and a random subset of the features.

The final prediction is made by averaging the predictions of all the trees in the ensemble. Random Forests are used for both classification and regression tasks, and are often considered a strong baseline model in machine learning.

Accuracy obtained from Random Forest Model with 100 n_estimators and gini criterion is **88.383%**

Bagging Classifier

A bagging classifier is a type of ensemble classifier that combines the predictions of multiple classifiers trained on different subsets of the training data to improve the overall accuracy of the model.

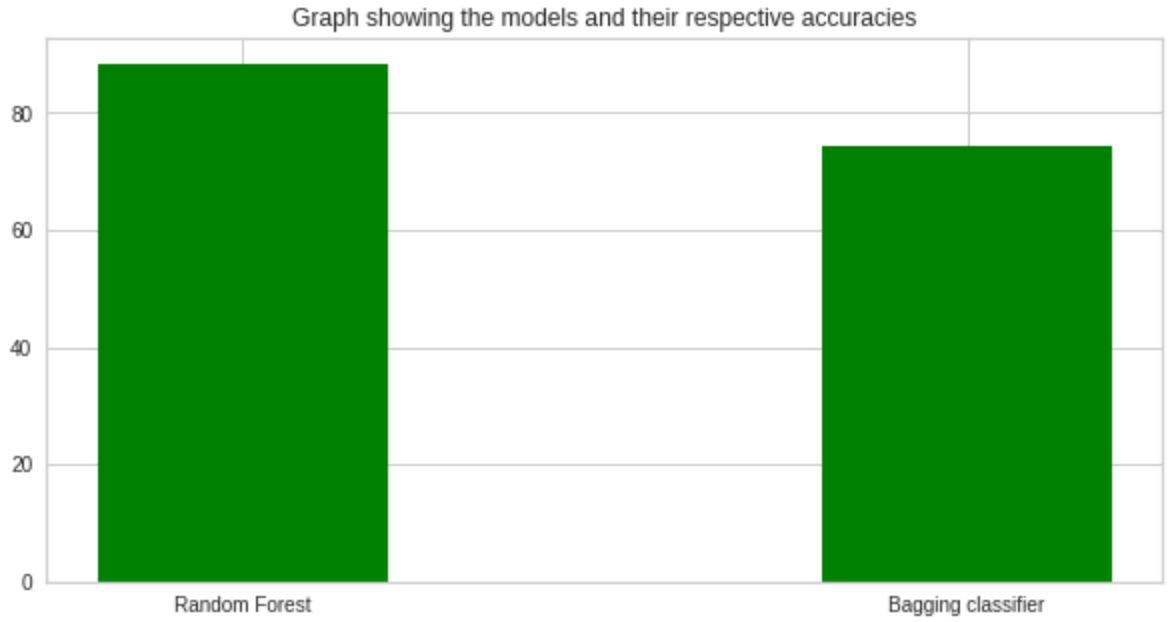
To use a bagging classifier, the training data is first divided into a number of subsets, and a separate classifier is trained on each subset.

The classifiers can be trained in parallel, which makes the method suitable for large datasets. The final prediction is made by aggregating the predictions of all the individual classifiers using a voting or averaging method.

Accuracy obtained from the Bagging Classifier Model is **74.347%**

Accuracies with different n_estimators

N_estimators	Accuracy
10	74.347
12	74.298
14	74.347
16	74.347
20	74.395


Graph showing the models and their respective accuracies
80
60
40
20
0
Random Forest
Bagging classifier

Bagging Meta-Estimator

A bagging meta-estimator is an ensemble machine learning method that combines the predictions of multiple "base" estimators to improve the overall performance of the model. It is called a "meta-estimator" because it is a higher-level estimator that is built on top of other estimators.

To use a bagging meta-estimator, the training data is first divided into a number of subsets, and a separate base estimator is trained on each subset.

The base estimators can be trained in parallel, which makes the method suitable for large datasets. The final prediction is made by aggregating the predictions of all the individual base estimators using a voting or averaging method.

Accuracy obtained from Bagging Meta-estimator is **89.593%**

Boosting

Boosting is an ensemble method used in machine learning to improve the accuracy of a model by training it on a sequence of sub-models. Each sub-model is trained to correct the mistakes made by the previous sub-model.

Boosting is effective in reducing bias and improving the model's ability to generalize to new data. The final model is a weighted sum of the sub-models, where the weights are determined by the performance of each sub-model.

Boosting algorithms can be used for both classification and regression tasks. Some popular boosting algorithms include AdaBoost, Gradient Boosting, and XGBoost.

CatBoost

CatBoost is a gradient boosting algorithm for machine learning developed by Yandex.

It is designed to handle categorical data automatically, which is one of the main reasons it is named "CatBoost." It is implemented in Python, R, and C++ and can be used for classification and regression tasks.

CatBoost is known for its high performance and speed, making it a popular choice in machine learning competitions.

One unique feature of CatBoost is its ability to handle missing values and categorical features without the need for preprocessing.

Accuracy obtained from the CatBoost Model is **82.139%**

Light Gradient Boost Machine

Light Gradient Boosting Machine (LightGBM) is a gradient boosting framework developed by Microsoft that is used for machine learning tasks such as classification and regression.

It is designed to be faster and more efficient than traditional gradient boosting algorithms by using a unique technique called "Gradient-based One-Side Sampling" (GOSS).

LightGBM also uses a histogram-based algorithm to choose the best split points for decision trees, which reduces the time complexity of training. LightGBM has gained popularity in recent years due to its high performance and efficiency in handling large-scale data.

Accuracy obtained from the Light Gradient Boost Machine is **81.51%**

AdaBoost

AdaBoost (Adaptive Boosting) is a boosting algorithm that is used to improve the performance of a model by training it on a sequence of sub-models.

It is an iterative process, where each sub-model is trained to correct the mistakes made by the previous sub-model.

AdaBoost is a popular boosting algorithm because it is simple to implement and can be used for both classification and regression tasks.

The final model is a weighted sum of the sub-models, where the weights are determined by the performance of each sub-model. AdaBoost is sensitive to noisy data and outliers, so it is important to preprocess the data before training a model using AdaBoost.

Accuracy obtained from the AdaBoost Model is **76.089%**

Accuracies obtained for different n_estimators

N_estimators	Accuracy
100	76.089
120	76.283
140	76.476
160	76.621
180	76.767
200	76.718
220	77.009
240	76.621
260	76.670
280	76.670
300	76.525

Gradient Boost Machine

Gradient Boost is an ensemble learning method that combines the predictions of multiple weaker models to form a more powerful model.

It involves training weak models sequentially, with each model trying to correct the mistakes of the previous model. The weak models are typically decision trees.

The predictions of the weak models are combined using a weighted average, where the weights are learned through optimization.

The objective function being optimized is typically the mean squared error between the predicted and actual values.

Accuracy obtained from Gradient Boost Machine is **75.218%**

Accuracies obtained by varying parameters

N_estimators	Loss	Criterion	Accuracy
100	log_loss	friedman_mse	75.218
120	deviance	squared_error	75.073
140	exponential	friedman_mse	75.024
160	log_loss	squared_error	76.041
180	deviance	friedman_mse	76.089
200	exponential	squared_error	76.331
220	log_loss	friedman_mse	77.202
240	deviance	squared_error	76.912
260	exponential	friedman_mse	76.525
280	log_loss	squared_error	76.960
300	deviance	friedman_mse	76.960

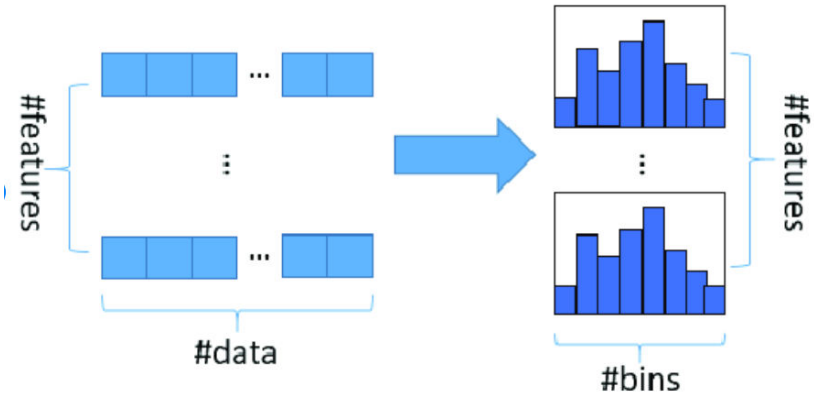
Extreme Gradient Boost Machine

- In this algorithm, decision trees are created in sequential form.
- Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results.
- The weight of variables predicted wrong by the tree is increased and the variables are then fed to the second decision tree.
- These individual classifiers/predictors then ensemble to give a strong and more precise model.
- It can work on regression, classification, ranking, and user-defined prediction problems.
- Gradient Boost method of Boosting gives an accuracy of **77.444**%

Histogram based Gradient Boost Machine

- A histogram is used to count or illustrate the frequency of data (number of occurrences) over discrete periods called bins.
- Each bin represents the frequency of the associated pixel value, and the histogram algorithm is conceptually quite simple.

- Accuracy of Histogram based Gradient boost Machine is **81.849%**



Accuracies obtained by varying parameters

Loss	Accuracy
log_loss	81.849
auto	81.849
binary_crossentropy	81.849

Improvements using GridSearchCV

GridSearchCV

GridSearchCV is a method that can be used to tune the hyperparameters of a model in an automated fashion.

It works by defining a grid of hyperparameter values and evaluating the model performance for each combination of these values, using cross-validation.

It can be used to find the combination of hyperparameter values that performs the best on the training data.

For example, if you are training a decision tree model, you may want to find the optimal value for the maximum depth of the tree. You could specify a list of possible maximum depths, and GridSearchCV will train a decision tree model for each value of the maximum depth and evaluate its performance using cross-validation. The maximum depth value that results in the best performance will be selected as the optimal value.

Logistic Regression using GridSearchCV

In logistic regression, the predicted probability that an example belongs to a certain class is calculated using a logistic function, which takes the weighted sum of the input features and applies a sigmoid function to it. The weights (also known as coefficients) and the intercept term of the model are learned from the training data using an optimization algorithm, such as stochastic gradient descent.

GridSearchCV can be used to tune the hyperparameters of a logistic regression model in an automated fashion.

For example, you could specify a list of different values for the regularization strength (e.g. L1, L2) and the inverse of the regularization strength (C), and GridSearchCV will evaluate the performance of the logistic regression model for each combination of these values using cross-validation. The combination of hyperparameters that results in the best performance on the training data will be selected as the optimal set of hyperparameters.

Accuracy obtained from the Logistic Regression Model using GridSearchCV is **76.38%**

Random Forest Using GridSearchCV

To use GridSearchCV with a Random Forest model, you will need to specify the hyperparameters that you want to tune and the values that you want to try for each hyperparameter.

For example, you might want to tune the number of trees in the forest and the maximum depth of each tree. Then, you would pass these hyperparameters and their corresponding values to GridSearchCV, along with the training data and the model that you want to use.

GridSearchCV will then train and evaluate a Random Forest model for each combination of hyperparameter values, and return the set of hyperparameters that resulted in the best performance.

Accuracy obtained from Random Forest using GridSearchCV model is **87.481%**

AutoKeras

AutoKeras is an open-source library for automated machine learning (AutoML). It is built on top of the popular deep learning library Keras and is designed to allow users to easily search for the best neural network architecture for their data without the need for expert knowledge.

AutoKeras uses a search algorithm to try out different neural network architectures and hyperparameter configurations, and it automatically selects the best model based on the performance on a validation set. It also provides a number of preprocessing and data augmentation functions to help users get the most out of their data.

To use AutoKeras, users simply need to provide their training data and the library will handle the rest. It is a simple and easy-to-use solution for those who want to leverage the power of deep learning without having to spend a lot of time on model selection and hyperparameter tuning.

Accuracy obtained from AutoKeras Model is **77.182%**

Possible reasons for a lower accuracy compared to other models:

1. When the data is small: Neural networks require a large amount of data to train, and may not perform well on small datasets. In these cases, decision trees or other simpler models may be more suitable.
2. When the data is highly imbalanced: Decision trees are generally more robust to imbalanced datasets, as they can handle missing or rare classes better than neural networks.
3. When the data is noisy: Decision trees are more robust to noise and can handle outliers better than neural networks.
4. When the data has a simple structure: If the data has a simple linear or additive structure, decision trees may be able to model it more accurately than a neural network.

Artificial Neural Networks

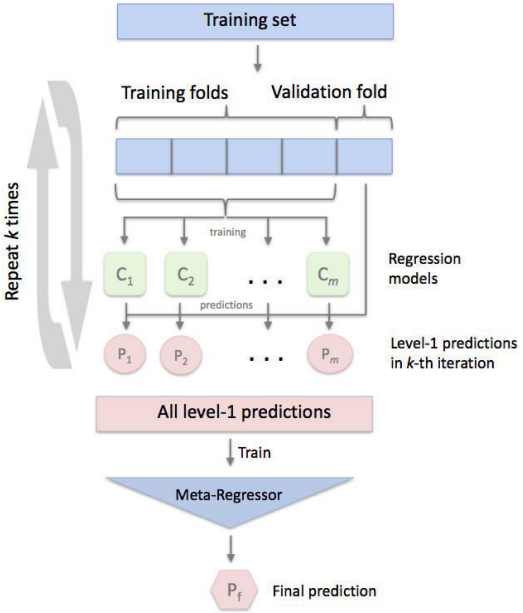
Artificial neural networks (ANNs) are a type of machine learning model inspired by the biological neural networks of the human brain. They are composed of interconnected units called "neurons" that communicate with each other to process data and make decisions.

Each neuron receives input from other neurons or external sources, processes the input using a simple mathematical operation, and produces an output that is passed on to other neurons or external outputs. The process of passing the inputs through the neural network and producing an output is known as "forward propagation."

ANNs can be trained to perform a variety of tasks, such as classification, regression, and prediction, by adjusting the weights and biases of the connections between the neurons. The training process involves presenting the neural network with a set of input-output pairs and adjusting the weights and biases to minimize the difference between the predicted output and the true output. This process is known as "backpropagation."

Accuracy obtained from Artificial Neural Networks Model is **75.411%**

MultiLevel Stacking



The process of stacking will be repeated multiple times and the output after each level for the corresponding model will be compiled and fed to the next iteration.

After all the iterations/cycles the original dataframe will be updated and trained test split will be done and will finally be passed to the model.

Accuracy : **99.751%**

THANK YOU