Individual Assignment

Statistical Machine Learning

Vinay Rajagopalan

Contents

Algorithms

Logistic Regression

Logistic Regress is mostly used in classification problems with two outcomes. It's an extension of the linear regression model for classification problems. A sigmoid function is utilized instead of a straight line or hyperplane to fit the data.

Objective Function

The objective Function for logistic Regression as follows:

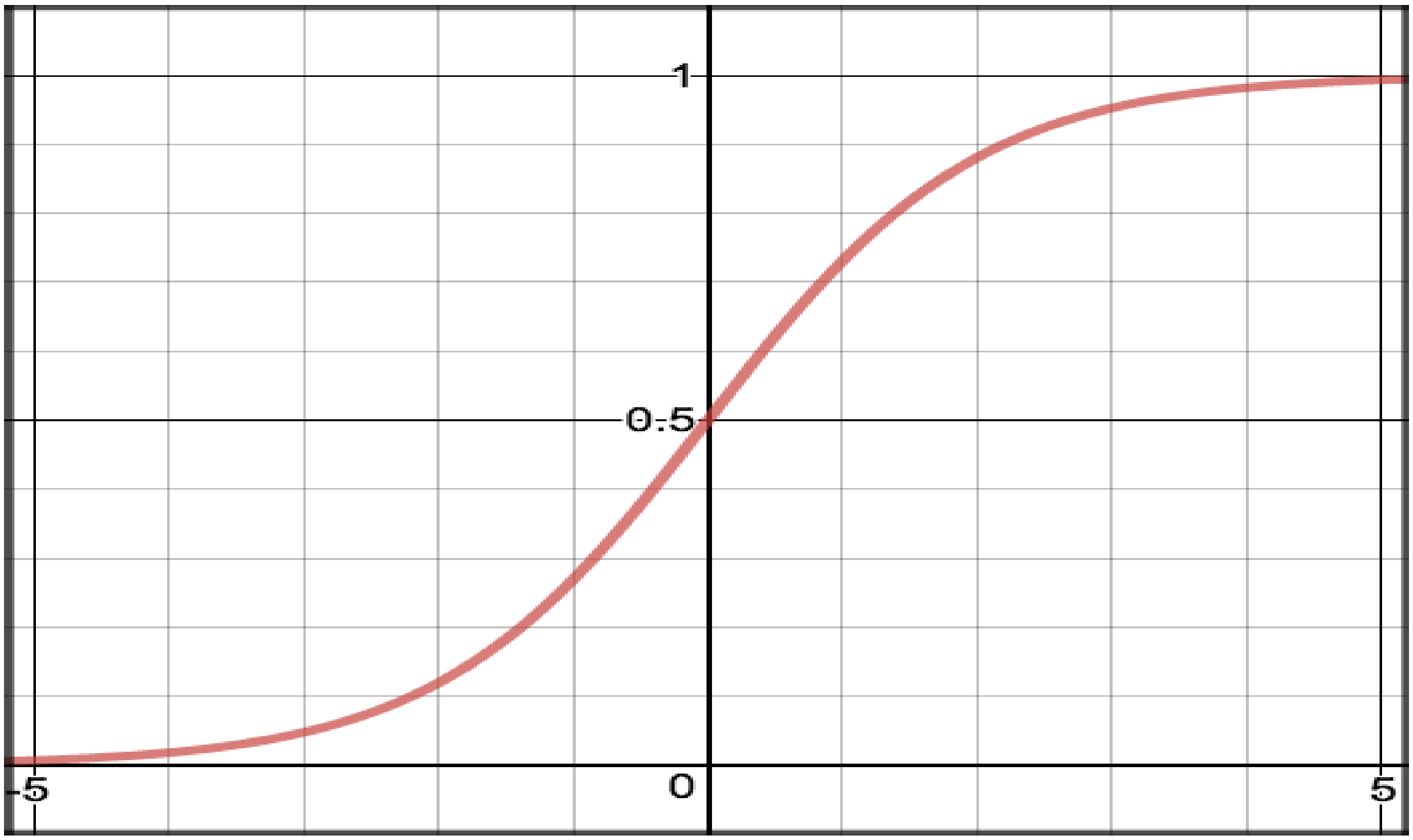
$$\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

$e \approx 2.71828$: Euler's number (math constant).

$p(X) = Pr(Y = 1/X)$: probability that observation X is in class 1, return value in [0, 1].

β_0 , β_1 : the coefficient of Logistic Regression model.

And it looks like this:



We need to fit the model to a dataset to estimate the parameters β_0 , β_1. Pick β_0 , β_1 to maximize the likelihood of the observed data. Maximum likelihood will provide values of β_0 , β_1 which maximizes the probability of the outcome within the dataset

The log likelihood function for Logistic Regression is as follows:

$$\sum_{i=1}^{n}\{y_i \ \beta^T x_i - \log(1 + e^{\beta^T x_i})\}$$

y_i = [0:1]

β = { β_0 , β_1 }

x_i = vector input, including constant term 1 for the intercept β_0

Gradient Descent or Iterative computing could be used to obtain the optimum value of β_0 , β_1 which would maximize the probability.

Pros

1. Logistic regression is one of the most straightforward supervised machine learning algorithms. This is because, in comparison to other approaches, the algorithm does not have a high compute power, making it appropriate for classification in the machine learning sector.
2. Unlike other approaches, the logistic algorithm allows users to simply update models to get/reflect new data. The main method for updating data in logistic regression is stochastic gradient descent.
3. The probability generated by this method are well-calibrated. This distinguishes it from other models or methods that merely provide the final categorization as a result.
4. Less prone to over-fitting- logistic regression is less prone to over-fitting in low-dimensional datasets.
5. You will get more accurate results than any other approach for many simple datasets. However, it works well if the dataset has linearly separable features.
6. Logistic regression can be easily extended to multiple classes and natural probabilities.

Cons

1. High-dimensional datasets lead to overfitting of the model, resulting in inaccurate test set results. Regularization techniques are used to contain overfitting defects. However, very high regularization can lead to model incompatibility and lead to inaccurate results.

2. Not all problems can be solved with this approach Non-linear problems cannot be solved with logistic regression techniques. Therefore, converting these non-linear problems into linear problems can be difficult and time consuming.
3. Logistic regression is not as powerful as other algorithms such as neural networks, so problems can occur when capturing complex relationships.
4. Many observations are required. This technique is typically used when the number of observations is greater than the feature used. Otherwise, overfitting can occur if the number of observations is low.
5. Advanced data maintenance with logistic regression. Data maintenance is high because data preparation is cumbersome. This is achieved by scaling and normalizing the data

K-nearest neighbors (KNN)

K-nearest neighbors (KNN) is a type of supervised learning algorithm. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is close to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of ‘K’ training data and class holds the highest probability will be selected. The K nearest Neighbors (KNN) algorithm uses "feature similarity" to predict the value of a new data point. This means that new data points will be assigned a value based on the degree of matching with the points in the training set.

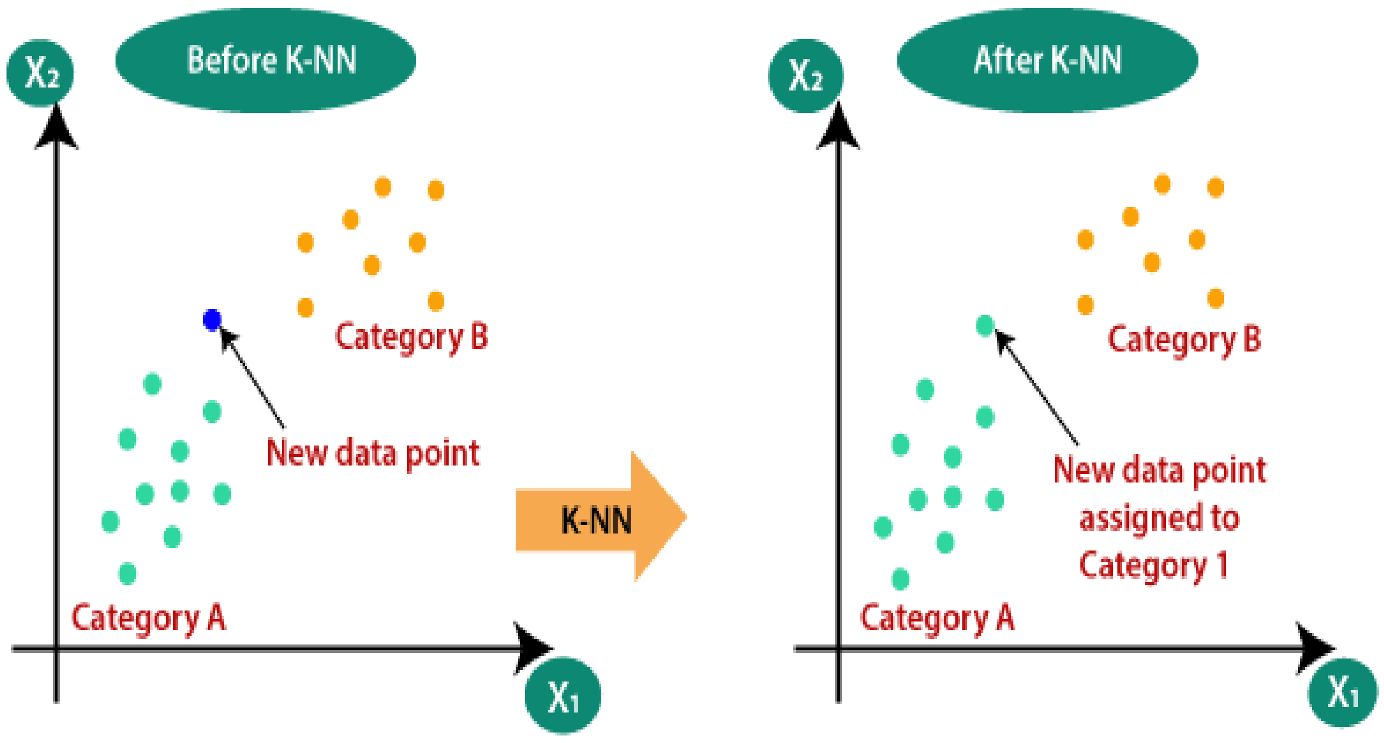
Algorithm

The first step in KNN is to load both training and test data.
Next, you need to select a value for K. K can be any integer.
For each point in the test data, do the following: -

1. Calculate the distance between each set of test and training data using either the Euclidean, Manhattan, or Hamming distance method. The most used method for calculating distance is the Euclidean method.
2. Sort in ascending order based on distance value.
3. Next, the top K rows are selected from the sorted array.
4. The test points are now assigned classes based on the most frequent classes in these rows.

Euclidean Distance is given by

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$



Pros

1. It is extremely easy to implement
2. As said earlier, it is lazy learning algorithm and therefore requires no training prior to making real time predictions. This makes the KNN algorithm much faster than other algorithms that require training e.g., SVM.
3. Since the algorithm requires no training before making predictions, new data can be added seamlessly.
4. There are only two parameters required to implement KNN i.e., the value of K and the distance function (e.g., Euclidean or Manhattan etc.)

Cons

1. The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate distance in each dimension.
2. The KNN algorithm has a high prediction cost for large datasets. This is because in large datasets the cost of calculating distance between new point and each existing point becomes higher.
3. Finally, the KNN algorithm doesn't work well with categorical features since it is difficult to find the distance between dimensions with categorical features.

Support Vector Machine

A support vector machine is a binary classifier which allows to distinguish whether an event or object belongs to either of two classes. It constructs a surface in an m-dimensional space, which separates two regions corresponding to the two classes we aim to distinguish.

Objective Function

The objective function is as follows:

$$f(x) = \beta_0 + \sum_{j=1}^{p} \alpha_i K(x, x_i)$$

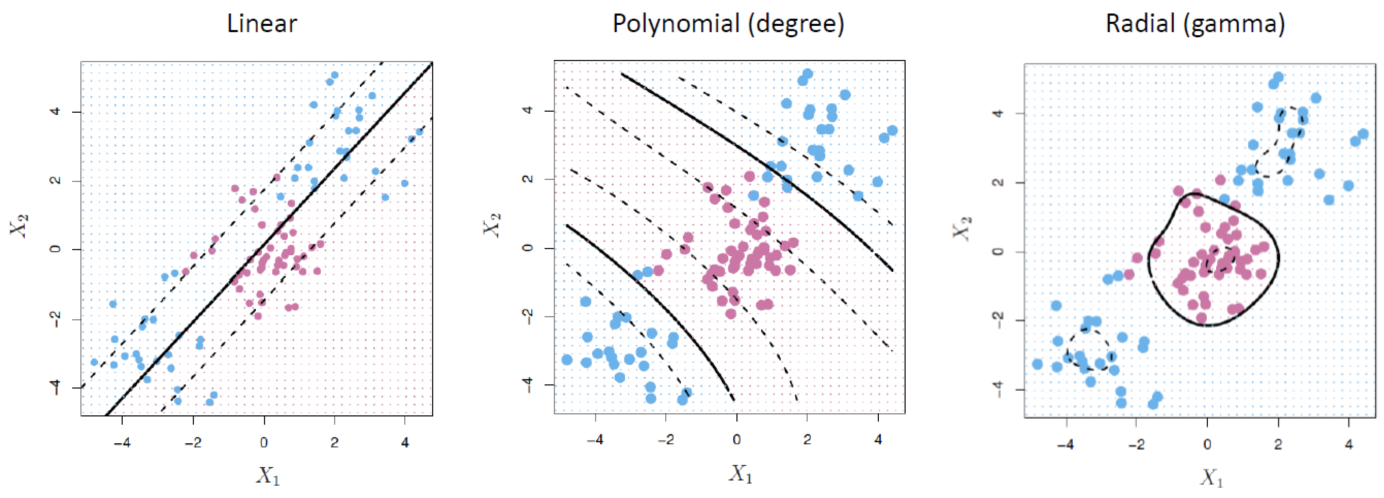
Where:

β_0 and α_i: parameter of the model

$K(x, x_i)$: the kernel function quantifies the similarity of two observations

S: the collection of indices data points where $\alpha i \neq 0$, i.e. the support vectors

We could use various kernel function in support vector machines which it a very powerful algorithm. There are three types of kernels in Support Vector Machine, linear, polynomial and radial kernel. An example of each can be seen below.



Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated.

Linear Kernel:

$$K(x_i, x_{i'}) = \sum_{i=1}^{n} x_{ij} x_{i'j}$$

Polynomial Kernel:

$$K(x_i, x_{i'}) = \left(1 + \sum_{i=1}^{n} x_{ij} x_{i'j}\right)^{d}$$

Radial Kernel:

$$K(x_i, x_{i'}) = \exp\left(-\Upsilon \sum_{i=1}^{n} (x_{ij} - x_{i'j})^2\right)$$

Maximal Margin Classifier (Hard margin)
Maximize M, β_0, $\beta_1, \ldots, \beta_p$

subject to: $\sum_{j=1}^{p} \beta_j^2 = 1$

$$y_i(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip}) \geq M \; \forall \; i = 1, \ldots, n$$

Maximal Margin Classifier (Soft margin)
Maximize M, β_0, $\beta_1, \ldots, \beta_p$, $€_1, \ldots, €_n$

subject to: $\sum_{j=1}^{p} \beta_j^2 = 1$

$$y_i(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip}) \geq M(1 - €_i)$$

$$€_i > 0, \sum_{i=1}^{n} €_i \leq C$$

Pros

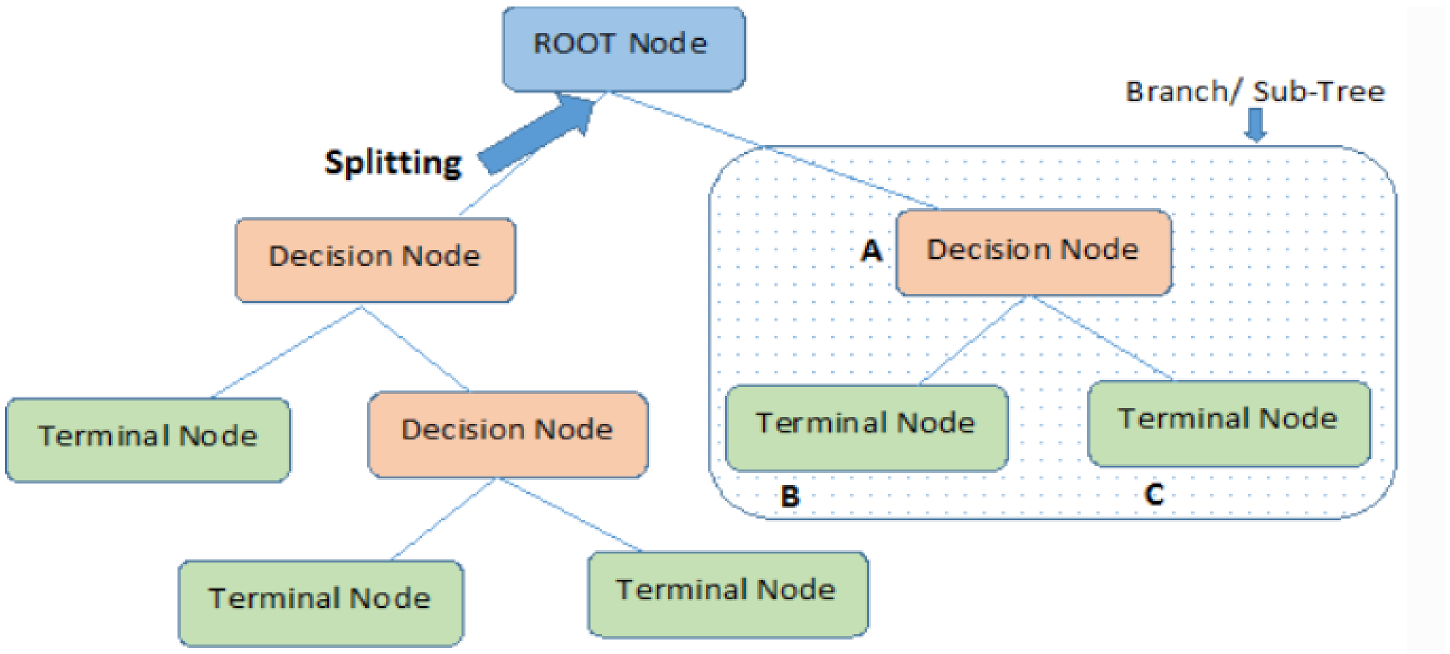
1. It works really well with a clear clearance margin.
2. Effective in high-dimensional space.
3. This is useful when the number of dimensions is greater than the number of samples.
4. It is also memory efficient because the decision function (called a support vector) uses a subset of training points.

Cons

1. If the dataset is large, it will not work well as it will require more training time.
2. It also doesn't work well if the dataset is noisy, that is, if the target classes are duplicated.
3. SVM does not provide direct probability estimation. These are calculated using the expensive 5-fold cross-validation test.

Decision Tree

Decision Trees are nonparametric supervised learning methods used for classification and regression. The deeper the tree, the more complex the decision rules and the better the model. The decision tree builds a classification or regression model in the form of a tree structure. Divide the dataset into smaller subsets while developing the relevant decision trees in stages. The result is a tree with decision nodes and leaf nodes. The decision node has more than one branch. Leaf nodes represent classifications or decisions. The top-level decision node in the tree that corresponds to the best predictor, called the root node. The decision tree can handle both categorical and numeric data.



The main challenge in implementing a decision tree is to identify the root node and the attributes that are considered at each level. This process is called attribute selection. There are several approaches to choosing an attribute to identify what can be considered a root node at each level.

There are two common attribute selection scales. They are:
Information gain

Using the information gain as a reference, the attribute assumes that the attribute is categorical. Information gain is the decrease in entropy. Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values.
Entropy is represented by the following formula

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

Where, c is the number of classes
and pi is the probability associated with the ith class.

Gini Index

Gini Index assumes that the attribute is continuous. According to the Gini Index, if you randomly select two items from the population, they must be in the same class, and if the population is pure, the probability is 1. Works with the target variable "Success" or "Failure" in the category. Performs only binary splits. The higher the value of Gini, the higher the homogeneity.
Gini Index is represented by the following formula

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

Where, c is the number of classes
and pi is the probability associated with the ith class.

Steps to Calculate Gini for a split

1. Calculate Gini for sub-nodes, using formula sum of the square of probability for success and failure.

2. Calculate Gini for split using weighted Gini score of each node of that split.

Pros

1. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
2. A decision tree does not require normalization of data.
3. A decision tree does not require scaling of data as well.
4. Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.
5. A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.

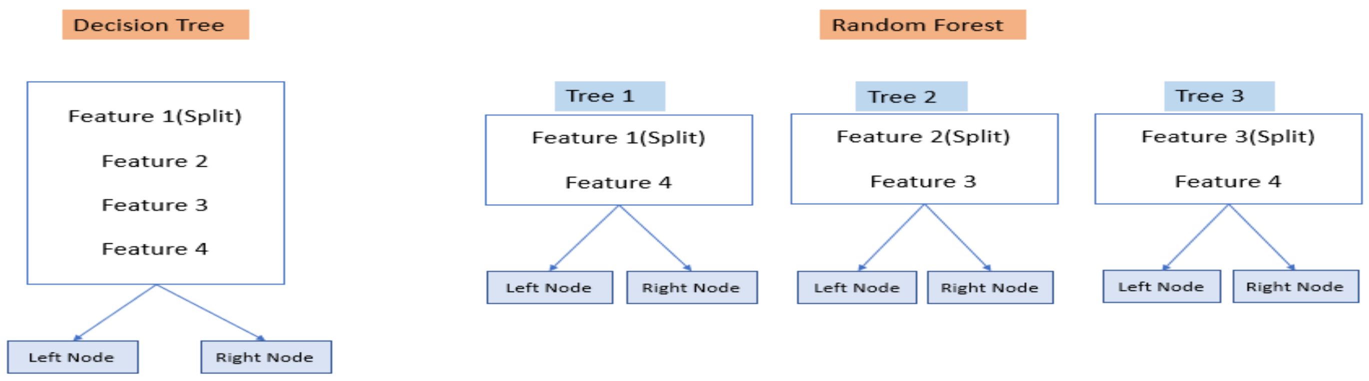
Cons

1. High variance
2. Single tree is not good as other models

Random Forrest

Random forest is a supervised learning algorithm. The forest it builds, is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. For classification tasks, the output of the random forest is the class selected by most trees. Random decision forests correct for decision tree's habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

Random forests are frequently used as Blackbox models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.



Bagging
Bagging involves sampling multiple training sets from a data set and fitting each of them to a different classifier. It is usually applied to decision tree methods. The method of sample selection is called Bootstrapping which is sampling with replacement. So, it generates multiple decision trees with each trained on a different training data set. Each node in the tree searches over all the features to find the best feature to split a node. The splitting criteria can be Gini Index or Information Gain as discussed in decision tree. The best feature is the one which results in the largest Information Gain or in other words which splits a set into more homogeneous subsets.
Bagging is going to generate many decision trees. Then all the trees are combined to create a single predictive model. It can be used both in Classification and Regression. Since averaging a set of observations reduces variance bagging will yield a low variance learning model.

Entropy is represented by the following formula

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

Gini Index is represented by the following formula

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

Algorithm

1. Grow a forest of many trees.
2. Grow each tree on an independent bootstrap sample from the training data.
3. At each node:
    1. Select m variables at random out of all M possible variables (independently for each node).
    2. Find the best split on the selected m variables.
4. Grow the trees to maximum depth (classification).
5. Vote the trees to get predictions for new data.

A case in the training data is not in the bootstrap sample for about one third of the trees (we say the case is “out of bag” or “oob”). Vote the predictions of these trees to give the RF predictor. The oob error rate is the error rate of the RF predictor. The oob confusion matrix is obtained from the RF predictor. For new cases, vote all the trees to get the RF predictor

Pros

1. It can perform both regression and classification tasks.
2. A random forest produces good predictions that can be understood easily.
3. It can handle large datasets efficiently.
4. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.

Cons

1. When using a random forest, more resources are required for computation.
2. It consumes more time compared to a decision tree algorithm.

Data Preprocessing

Column	*NA handling*	*Preprocessing*
age	All NAs were filled with age.mean	Were split into category and new columns were created for age10-20, age20-30 and so on. Finally, all the age categories were dummy encoded using pd.dummies
job	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
marital	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
education	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
default	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
housing	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
loan	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
contact	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
month	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
day_of_week	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
poutcome	All NAs were filled with unknown	Were categorical variables. New columns were added which were dummy encoded using pd.dummies
campaign	All NAs were filled with -9999	An extra column was added to keep track of where NA columns were imputed
pdays	All NAs were filled with -9999	An extra column was added to keep track of where NA columns were imputed
previous	All NAs were filled with -9999	An extra column was added to keep track of where NA columns were imputed
emp.var.rate	All NAs were filled with -9999	An extra column was added to keep track of where NA columns were imputed

cons.price.idx	All NAs were filled with -9999	An extra column was added to keep track of where NA columns were imputed
cons.conf.idx	All NAs were filled with -9999	An extra column was added to keep track of where NA columns were imputed
euribor3m	All NAs were filled with -9999	An extra column was added to keep track of where NA columns were imputed
nr.employed	All NAs were filled with -9999	An extra column was added to keep track of where NA columns were imputed

Our target variable is the subscribe column and the distribution is 0.11355 which would act as our threshold for the model to predict whether a client will subscribe or not.

The client_id column is the primary key and will be dropped from the table while training all our models.

Benchmark

Feature Selection

The forward and backward feature selection method was used. Top 15 features were extracted from total 82 features that were used to train the basic models.
The model used for feature selection was Ridge Regression.

In forward feature selection the selected features were:
'job_student', 'marital_single', 'default_no' , 'contact_telephone', 'month_apr', 'month_dec', 'month_jun', 'month_mar' , 'month_oct' , 'month_sep', 'day_of_week_mon', 'poutcome_success', 'age-group_60-70' , 'age-group_70-80', 'age-group_80-90'

In backward feature selection the selected features were:
'pdays', 'cons.conf.idx', 'nr.employed', 'job_blue-collar','contact_telephone', 'month_jul', 'month_jun', 'month_mar', 'month_may', 'day_of_week_mon', 'poutcome_nonexistent', 'poutcome_success', 'pdays_missing', 'cons.conf.idx_missing', 'nr.employed_missing'

Logistic Regression

Cross Validation was performed on entire dataset with 5 Folds and the average test score were 0.8964.
Results:

Parameters	Features Selected	AUC_Train	AUC_Test	Acc_Train	Acc_Test
random_state=0, max_iter=1000	All	75.64	75.31	78.85	78.66
random_state=0, max_iter=1000	Forward Selected Features	75.10	75.42	85.007	84.48
random_state=0, max_iter=100000, penalty='l2'	Backward Selected Features	58.09	58.45	70.73	71.65

In the case of model running with all features we see that there is a marginal overfitting in terms of AUC while when model is run with forward selected features we see that results are better and there is no overfitting in terms of AUC and lastly for the model with backward selected features we see that the AUC is not overfitting but the accuracy seems to be very low as model is predicting a lot of false positives and false negatives. Thus, in this scenario model with forward selected features seems to be performing better than the rest of the models although there is marginal overfitting in terms of accuracy. In addition, model was tuned with various parameters to see which values gives the best performance, the one listed above seems to perform the best.

K-nearest neighbors (KNN)

Cross Validation was performed on entire dataset with 5 Folds to determine the value for optimum number of neighbors. On completing the cross validation for various values for neighbors we see that we get an optimum test score when value of neighbors is 221.

Results:

Parameters	Features Selected	AUC_Train	AUC_Test	Acc_Train	Acc_Test
n_neighbors=221	All	78.93	78.74	79.55	78.53
n_neighbors=221	Forward Selected Features	74.06	73.12	76.24	76.1
n_neighbors=221	Backward Selected Features	78.55	78.29	81	79.93

The model seems to be overfitting in all three cases within marginal boundaries in terms of AUC and accuracy also follows the same pattern in all three cases. The model seems to perform better in terms of accuracy in comparison to Logistic Regression with all and backward selected features, but logistic regression has less overfitting compared to the KNN model. In addition, model was tuned with various parameters to see which values gives the best performance, the one listed above seems to perform the best.

Support Vector Machine

Parameters	Features Selected	AUC_Train	AUC_Test	Acc_Train	Acc_Test
n_neighbors=221	All	78.93	78.74	79.55	78.53

n_neighbors=221	Forward Selected Features	74.06	73.12	76.24	76.1
n_neighbors=221	Backward Selected Features	78.55	78.29	81	79.93

Decision Tree

Cross Validation was performed on entire dataset with 5 Folds and the average test score were 0.8967.

Results:

Parameters	Features Selected	AUC_Train	AUC_Test	Acc_Train	Acc_Test
max_leaf_nodes=15, random_state=42	All	77.42	77.17	83.21	82.43
max_leaf_nodes=15, random_state=42	Forward Selected Features	72.99	72.64	88.40	87.56
max_leaf_nodes=15, random_state=42	Backward Selected Features	80.28	79.12	84.62	83.36

We see that we get best accuracy score compared to logistic and KNN from decision tree with forward selected features. The same is true for AUC as well we see the best result with decision tree with backward selected features. But all the models seem to be marginally overfitting in terms of both AUC and accuracy. In addition, model was tuned with various parameters to see which values gives the best performance, the one listed above seems to perform the best.

Random Forrest

Cross Validation was performed on entire dataset with 5 Folds and the average test score were 0.89765.

Results:

Parameters	Features Selected	AUC_Train	AUC_Test	Acc_Train	Acc_Test
kernel='linear', probability=True	All Features were scaled using StandardScalar	55.23	54.62	89.97	89.28
kernel='linear', probability=True	Forward Selected Features	62.21	60.03	89.92	89.26
kernel='sigmoid', probability=True	Forward Selected Features	61.43	60.21	56.50	56.21
kernel='sigmoid', C = 100, probability=True	Normalized the features	52.16	46.78	72.83	71.1

In terms of SVM all models seem to have a very low AUC score compared to other models. Various kernels were tried and different values of cost functions were used but AUC score did not seem to improve. In addition, model was tuned with various parameters to see which values gives the best performance.

Decision Tree

Cross Validation was performed on entire dataset with 5 Folds and the average test score were 0.8967.

Results:

Parameters	Features Selected	AUC_Train	AUC_Test	Acc_Train	Acc_Test
max_leaf_nodes=15, random_state=42	All	77.42	77.17	83.21	82.43
max_leaf_nodes=15, random_state=42	Forward Selected Features	72.99	72.64	88.40	87.56
max_leaf_nodes=15, random_state=42	Backward Selected Features	80.28	79.12	84.62	83.36

We see that we get best accuracy score compared to logistic and KNN from decision tree with forward selected features. The same is true for AUC as well we see the best result with decision tree with backward selected features. But all the models seem to be marginally overfitting in terms of both AUC and accuracy. In addition, model was tuned with various parameters to see which values gives the best performance, the one listed above seems to perform the best.

Random Forrest

Cross Validation was performed on entire dataset with 5 Folds and the average test score were 0.89765.

Results:

Parameters	Features Selected	AUC_Train	AUC_Test	Acc_Train	Acc_Test
n_estimators=2000, max_depth=5, n_jobs = 5000, max_samples = 3000, random_state=42	All	80.16	79.14	82.26	81.61
n_estimators=2000, max_depth=5, n_jobs = 5000, max_samples = 3000, random_state=42	Forward Selected Features	75.72	76.01	84.72	84.16
n_estimators=2000, max_depth=5, n_jobs = 5000, max_samples = 3000, random_state=42	Backward Selected Features	78.37	78.67	83.5	82.43

In the case of model running with all features we see that there is a marginal overfitting in terms of AUC while when model is run with forward selected features we see that results are better and there is no overfitting in terms of AUC and lastly for the model with backward selected features we see that the AUC is not. The accuracy for all three models seem to be marginally overfitting. Thus, in this scenario model with forward selected features seems to be performing better than the rest of the models although there is marginal overfitting in terms of accuracy. In addition, model was tuned with various parameters to see which values gives the best performance, the one listed above seems to perform the best.

References

https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4#:~:text=K%2Dnearest%20neighbors%20(KNN),closet%20to%20the%20test%20data

https://prosancons.com/education/pros-and-cons-of-logistic-regression/
https://medium.com/@luigi.fiori.lf0303/distance-metrics-and-k-nearest-neighbor-knn-1b840969c0f4#:~:text=The%20formula%20to%20calculate%20Euclidean,total%20is%20our%20Euclidean%20distance

https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/
https://wiki.ecdc.europa.eu/fem/Pages/Fitting%20logistic%20regression%20models.aspx
https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm
https://towardsdatascience.com/k-nearest-neighbors-algorithm-with-examples-in-r-simply-explained-knn-1f2c88da405c
https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
https://chirag-sehra.medium.com/decision-trees-explained-easily-28f23241248
https://www.kaggle.com/code/prashant111/decision-tree-classifier-tutorial/notebook
https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a
https://builtin.com/data-science/random-forest-algorithm
https://en.wikipedia.org/wiki/Random_forest
https://medium.com/all-about-ml/bagging-random-forests-and-boosting-8c728e91a85d
https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/
https://www.usu.edu/math/adele/RandomForests/Ovronnaz.pdf