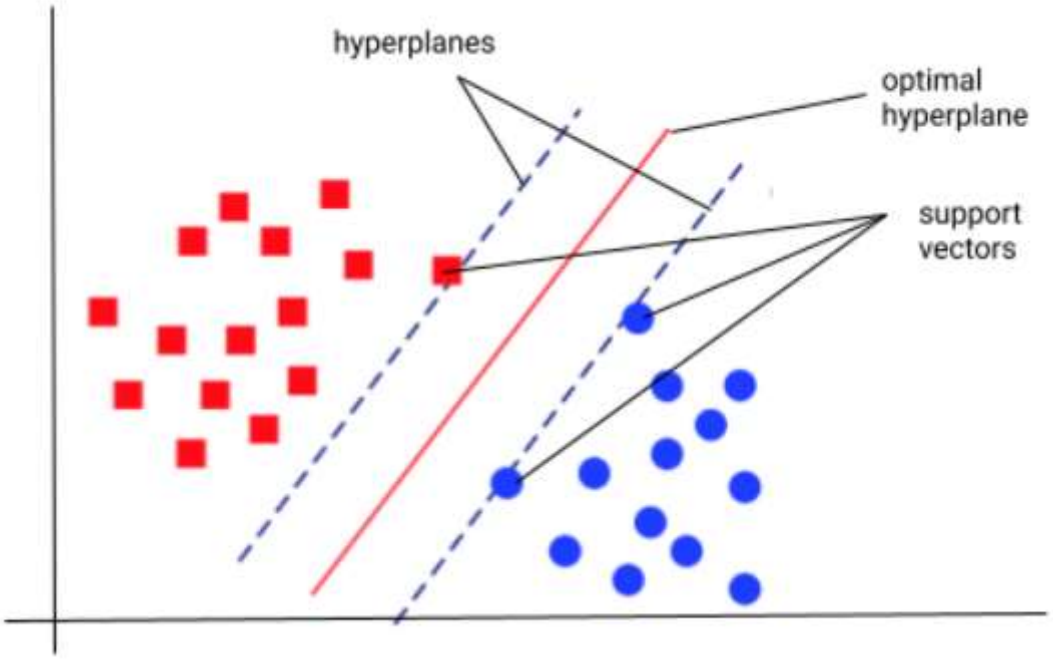
Tree-Based Methods and The Support Vector Machine

Linear and non-linear Support Vector Machine

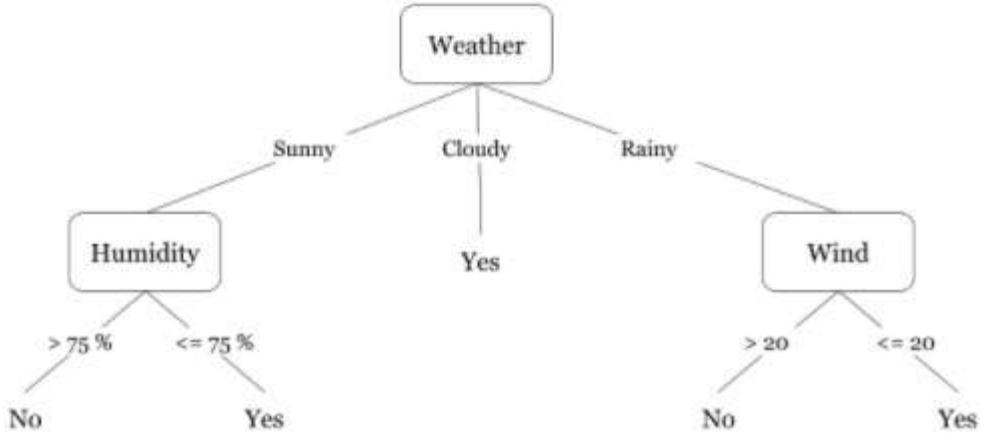
Support Vector Machine (SVM) is a supervised machine learning algorithm that is used to solve both classification and regression tasks. In classification, the algorithm uses a line or hyperplane to separate classes by using data points close to the boundary (support vector) for each class and a hyperplane that maximizes the distance between the classes. For clarity, a hyperplane is a line that linearly separates data points. Although there can be several hyperplanes between classes, the optimal hyperplane which has the maximum distance or margin between itself and the support vectors is chosen.



As we know, data is not always linearly separable such that a straight line might not be able to adequately segregate classes. Although SVM is a linear classifier, it can be used to classify a non-linear dataset by transforming the dataset to a higher dimensional feature space where it can be linearly separable. This is done using the kernel trick such that a kernel function is applied on each data point to map to a higher dimensional space.

Decision Trees and CART algorithm

The decision tree is a widely used non-parametric supervised machine learning approach that splits instances in a dataset based on different decision rules inferred from the features in the dataset. It is a tree-based algorithm with nodes that represent a specific attribute or decision rule such that for an instance, a question is asked at a node and possible answers to the question found on both edges. This is a sequential process that involves recursive partitioning of nodes for several features until the leaves for the tree provides the final output or class for that instance. Decision trees can also be used to solve regression problems.



ID3 - Iterative Dichotomiser 3, CART - Classification and Regression Trees and C4.5 are some examples of decision tree algorithms. In this section, we only discuss the CART algorithm. The CART predictive model generates decision rules that have a binary tree representation such that each non-terminal node has two child nodes as opposed to some other tree-based methods that have more child nodes. It supports numerical target variables. At every node, the best split is chosen such that the splitting criterion is maximised. Gini impurity index is used as the splitting criterion in CART.

Gini Impurity: this is a measure of the chance that a randomly selected instance will be wrongly classified when selected. For different classes in a dataset, with p(i) as the probability that the chosen instance belongs to class i, the gini impurity index for all classes G, can be calculated such that:

$$G = 1 - \sum_{i} p(i)^2$$

Gini impurity index values range between 0 and 1 such that 0 translates to a pure classification where all instances belong to the same class while 1 means that there is a random distribution of the instances across different classes. To select the best split, the gini gain is calculated by taking a weighted sum of the gini impurity index then subtracting from the original impurity. Higher gini gain leads to better splits simply put, the lower the gini impurity, the better the split.

Overfitting in Decision Trees, Early Stopping and Pruning

The recursive partitioning of nodes until the final subsets are obtained in decision trees makes it prone to overfitting. The deeper the tree, the higher the chances of the overfitting. This can be prevented using a stopping criterion such as early stopping and pruning. Early stopping or pre-pruning involves stopping the tree-building process before the tree becomes too complex and the training data is perfectly classified. An early stopping condition like the maximum depth can be set to avoid deep trees such that the tree stops growing after reaching the set maximum depth for the tree. Another early stopping criterion that can be used is the classification error. At every splitting stage, the error is checked. If there is no significant decrease in the error, there is no need to make the tree more complex. When there are fewer data points than a set threshold value, early stopping can also take place. Early stopping may also produce underfit models if it stops too early. Post-pruning, on the other hand, allows the tree to be fully built before simplifying by removing sections of the tree at different levels by calculating the error rate.


from sklearn.tree import DecisionTreeClassifier
dec_tree = DecisionTreeClassifier()
dec_tree.fit(normalised_train_df, y_balanced)