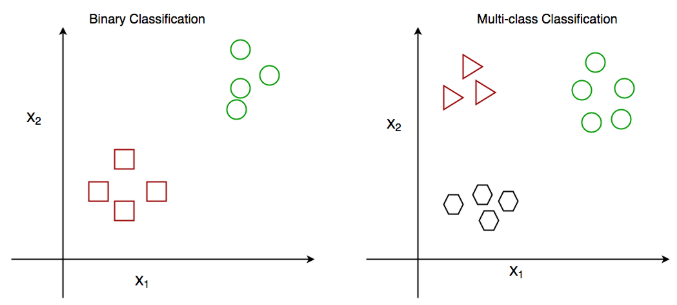
## 2.5.2. Classification and Regression

Classification and Regression are a couple of main prediction difficulties which are normally dealt with Data mining and machine learning. Classification is the process of finding or exploring a model or function which helps to divide the data into several categorical groups, i.e. discrete values. In classification, data is classified according to some parameters given in input under various labels and then the labels for the data are expected. The derived mapping function could be seen in the "IF-THEN" rules format. The labelling process tackles problems where the data can be broken down into binary or multiple discrete labels.

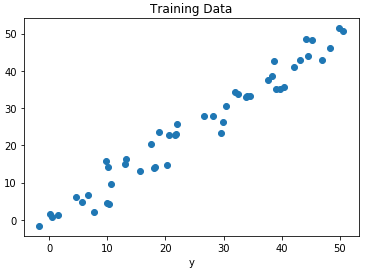
The below image shows the possibility of the winning of the match by Team A based on some parameters recorded earlier. Then there would be two labels Yes and No.



*Classification and Multiclass Classification*

Figure 9: Classification and Multiclass Classification

Regression means the process of seeking a model or function to differentiate the data from classes or isolated values into permanent real values. Based on historical records, it may also classify the distribution process. As a predictive regression model predicts an item, the competence of the model must be identified as a mistake in such predictions. The below image shows the possibility of rain in some regions with the help of some parameters recorded earlier. Then there is a probability associated with the rain.



*Regression of Day vs Rainfall (in mm)*

Figure 10: Regression of Day vs Rainfall

## 2.5.3. Classification

### 2.5.3.1 Gradient Boosting Classifier

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting. Gradient boosting classifiers are a group of machines learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets and have recently been used to win many data science competitions.

Gradient boosting classifier algorithms are used to build a model to predict diabetes. It is applied to train a dataset. In conclusion, this application is used to figure out a meal plan and predict diabetes status. The predict diabetes status accuracy of the model is 79.2%.

### 2.5.3.2 Random forest classifier

Random forests are a supervised learning algorithm. It can be used as a classification as well as regression. An algorithm is also the most robust and easy to use. A forest consists of trees. The more trees he has, the stronger the forest. Random forests build randomly selected data sample decision trees, extract a prediction from each tree, and choose the best solution by voting. This also provides a very good indicator of the value of the function. Random forests are used in a variety of applications, including recommendation engines, image classification, and feature selection. This is ideal for the classification of trustworthy lenders, the detection of illegal behavior, and disease prediction. It is based on the algorithm Boruta, which selects important characteristics in a dataset (Navlani, 2018).



Figure 11: Prediction

Random forest classifier algorithms are used to build a model to predict diabetes. It is applied to train a dataset. In conclusion, this application is used to figure out a meal plan and predict diabetes. The predict diabetes status accuracy of the model is 78.6%.

### 4.5.3.3 Decision Tree Classifier

A Decision Tree is a simple representation for classifying examples. It is a Supervised Machine Learning where the data is continuously split according to a certain parameter. It is a hierarchical structure that allows data to be determined. There is a client, for example, that gets an application for a new job. This will then choose whether to accept or reject this position on the grounds of pay, time the goal variable is represented in the decision tree structure by leaves and a branch function combination. Dependent on the goal variable, we will break the option book into two substrates. When the target variable responds consistently, the regression tree is renamed. It can be considered a classification tree because it takes categorical tests. of service, or other advantages. You will use a decision tree to do so. Next, start the cycle from the root node and answer each node's query and choose the branch corresponding to the corresponding answer.

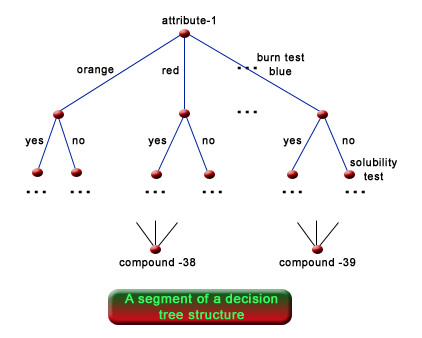


Figure 12: Decision Tree Classifier

Decision tree classifier algorithms use to build a model to predict diabetes. It is applied to train a dataset. In conclusion, this application is used to figure out predict meal plan and predict diabetes. The predict diabetes accuracy of the model is 71.4%.

#### 4.5.3.3.1 Naïve Bayes Classifier

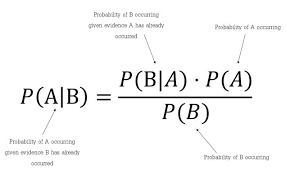
Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of [feature](https://en.wikipedia.org/wiki/Feature_vector) values, where the class labels are drawn from some finite set. There is not a single [algorithm](https://en.wikipedia.org/wiki/Algorithm) for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is [independent](https://en.wikipedia.org/wiki/Independence_(probability_theory)) of the value of any other feature, given the class variable.

Figure 13: Naïve Bayes Classifier Algorithm

#### 4.5.3.3.2 Gaussian Classification

The **Gaussian Processes Classifier** is a classification machine learning algorithm.

Gaussian Processes are a generalization of the Gaussian probability distribution and can be used as the basis for sophisticated non-parametric machine learning algorithms for classification and regression. Gaussian processes are a type of kernel method, like SVMs, although they can predict highly calibrated probabilities, unlike SVMs.

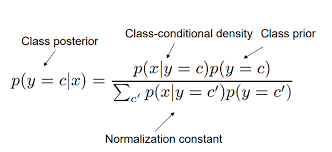


Figure 14: Gaussian Classifier Algorithm

Classification accuracy measured by the SVM with Gaussian kernel. Feature selection with information gain, random forest and GA was evaluated using incrementally smaller sets of most important features from the 1000-gene data set.

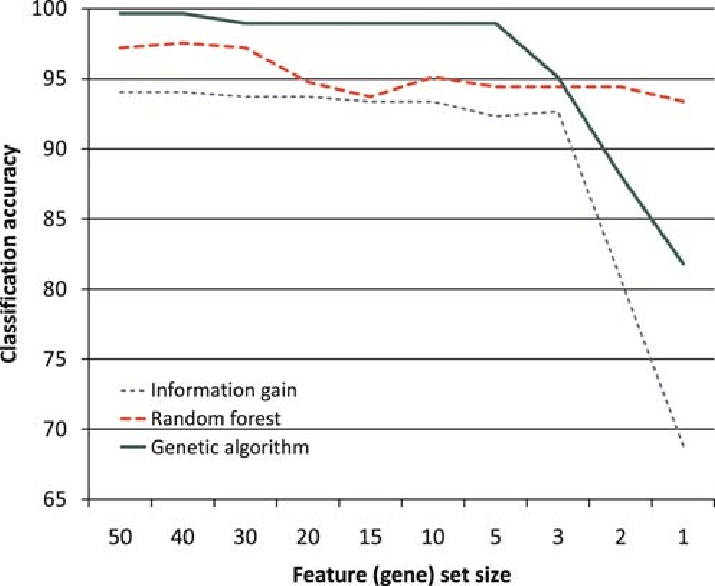


Figure 15: Classification Accuracy

#### 4.5.3.3.3 XG-Boost

XGBoost is an algorithm that has recently been dominating applied machine learning and Kaggle competitions for structured or tabular data. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.



Feature separation



Text feature



Numerical

feature



Keyword extraction



Outlier p

rocessing



RFE feature selection



Training Model



Data processing



Raw

data

Table 4: Diabetes prediction model trained based on XGBoost algorithm

By the iterated training and adjustment parameters of the training set, the comparison results are showed between XGBoost and the traditional algorithm models below, (Li and Fu, 2020)

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Algorithm | Accuracy | Operation hours |
| 1 | SVM | 0.655 | 1.269 |
| 2 | KNN | 0.719 | 1.253 |
| 3 | NB | 0.767 | 1.024 |
| 4 | DT | 0.703 | 1.254 |
| 5 | LR | 0.767 | 1.121 |
| 6 | XGBoost | 0.812 | 0.203 |

Table 5: XGBoost and the traditional algorithm models

### 4.5.3.5 Model Comparison (Classifiers and Boosters)

|  |  |  |
| --- | --- | --- |
| Models | Advantages | Disadvantages |
| Gradient Boosting Classifier | * Often provides predictive accuracy that cannot be beat. * Lots of flexibility - can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible. * No data pre-processing required - often works great with categorical and numerical values as is. * Handles missing data - imputation not required. (Gradient Boosting Machines · UC Business Analytics R Programming Guide, 2020) | * GBMs will continue improving to minimize all errors. This can overemphasize outliers and cause overfitting. Must use cross-validation to neutralize. * Computationally expensive - GBMs often require many trees (>1000) which can be time and memory exhaustive. * The high flexibility results in many parameters that interact and influence heavily the behavior of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning. * Less interpretable although this is easily addressed with various tools (variable importance, partial dependence plots, LIME, etc.). (Gradient Boosting Machines · UC Business Analytics R Programming Guide, 2020) |
| Random forest Classifier | * Random Forest is based on the bagging algorithm and uses Ensemble Learning technique. It creates as many trees as possible on the subset of the data and combines the output of all the trees. In this way it reduces overfitting problem in decision trees and reduces the variance and therefore improves the accuracy. * Random Forest can be used to solve both classification as well as regression problems. * Random Forest works well with both categorical and continuous variables. * Random Forest can automatically handle missing values. * No feature scaling required: No feature scaling (standardization and normalization) required in case of Random Forest as it uses rule-based approach instead of distance calculation. * Handles non-linear parameters efficiently: Non-linear parameters don't affect the performance of a Random Forest unlike curve-based algorithms. So, if there is high non-linearity between the independent variables, Random Forest may outperform as compared to other curve-based algorithms. * Random Forest can automatically handle missing values. * Random Forest is usually robust to outliers and can handle them automatically. * Random Forest algorithm is very stable. Even if a new data point is introduced in the dataset, the overall algorithm is not affected. much since the new data may impact one tree, but it is very hard for it to impact all the trees. * Random Forest is comparatively less impacted by noise. (Kumar, 2019) | * **Complexity**: Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources.   On the other hand, decision tree is simple and does not require so much computational resources.   * **Longer Training Period:** Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes. (Kumar, 2019) |
| Decision Tree Classifier | * Compared to other algorithms decision trees requires less effort for data preparation during pre-processing. * A decision tree does not require normalization of data. * A decision tree does not require scaling of data as well. * Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent. * A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders. (K, 2019) | * A small change in the data can cause a large change in the structure of the decision tree causing instability. * For a Decision tree sometimes, calculation can go far more complex compared to other algorithms. * Decision tree often involves higher time to train the model. * Decision tree training is relatively expensive as the complexity and time have taken are more. * The Decision Tree algorithm is inadequate for applying regression and predicting continuous values. (K, 2019) |
| Naïve Bayes Classifier | * When assumption of independent predictors holds true, a Naive Bayes classifier performs better as compared to other models. * Naive Bayes requires a small amount of training data to estimate the test data. So, the training period is less. * Naive Bayes is also easy to implement. (Kumar, 2019) | * Main imitation of Naive Bayes is the assumption of independent predictors. Naive Bayes implicitly assumes that all the attributes are mutually independent. In real life, it is almost impossible that we get a set of predictors which are completely independent. * If categorical variable has a category in test data set, which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as Zero Frequency. To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation. (Kumar, 2019) |
| Gaussian Classification | * Gaussian process (GP) directly captures the model uncertainty. As an example, in regression, GP directly gives you a distribution for the prediction value, rather than just one value as the prediction. This uncertainty is not directly captured in neural networks (see for example [https://arxiv.org/abs/1506.02142](https://www.researchgate.net/deref/https%3A%2F%2Farxiv.org%2Fabs%2F1506.02142)) * When using GP, you can add prior knowledge and specifications about the shape of the model by selecting different kernel functions. (Daee, 2017) | * The main limitation is the assumption of independent predictor features. * If a categorical variable has a category in the test dataset, which was not observed in training dataset, then the model will assign a 0 (zero) probability and will be unable to make a prediction. |
| XG-Boost | * **Regularization:** XGBoost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting. That is why, XGBoost is also called regularized form of GBM (Gradient Boosting Machine).  While using Scikit Learn library, we pass two hyper-parameters (**alpha**and **lambda**) to XGBoost related to regularization. **alpha**is used for L1 regularization and **lambda**is used for L2 regularization. * **Parallel Processing:** XGBoost utilizes the power of parallel processing and that is why it is much faster than GBM. It uses multiple CPU cores to execute the model.  While using Scikit Learn library, **nthread**hyper-parameter is used for parallel processing. **nthread**represents number of CPU cores to be used. If you want to use all the available cores, don't mention any value for **nthread**and the algorithm will detect automatically. * **Handling Missing Values:** XGBoost has an in-built capability to handle missing values. When XGBoost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data. * **Cross Validation:** XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run. This is unlike GBM where we must run a grid-search and only a limited value can be tested. * **Effective Tree Pruning:** A GBM would stop splitting a node when it encounters a negative loss in the split. Thus, it is more of a greedy algorithm. XGBoost on the other hand make splits up to the **max\_depth**specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain. (Kumar, 2019) | * In Xgboost, you must manually create dummy variable/ label encoding for categorical features before feeding them into the models. Catboost/Lightgbm can do it own their own, you just need to define categorical features names or indexes. * Training time is high for larger dataset, if you compare against catboost/lightgbm. (Mehta, 2018) |

Table 6: Model Comparison