Ensemble Models & Random Forests

Contents

Contents

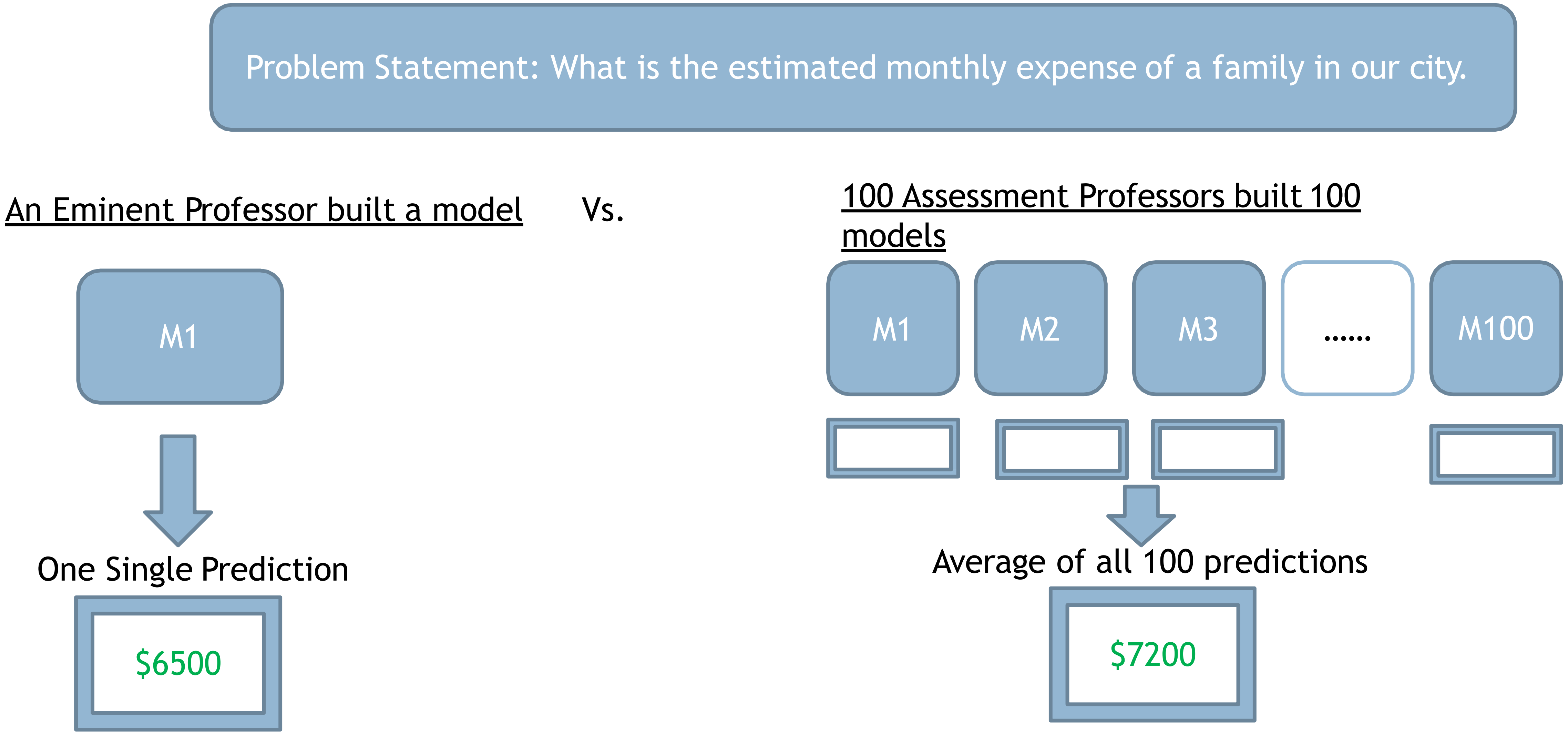
The Wisdom of Crowds

The wisdom of crowds

- “One should not expend energy trying to identify an expert within a group but instead rely on the group’s collective wisdom, however make sure that opinions must be independent and some knowledge of the truth must reside with some group members” – Surowiecki

- So instead of trying to build one great model, its better to build some independent moderate models and take their average as final prediction

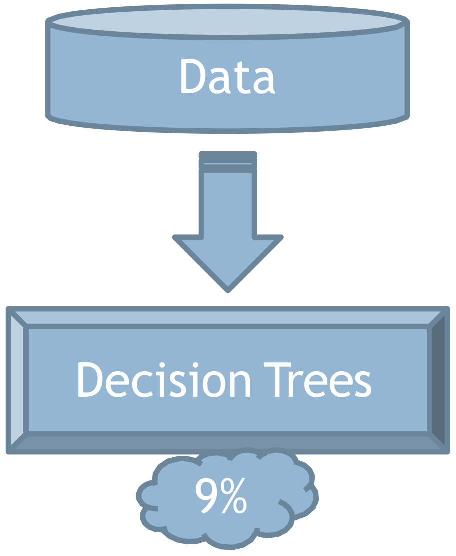
The wisdom of crowds

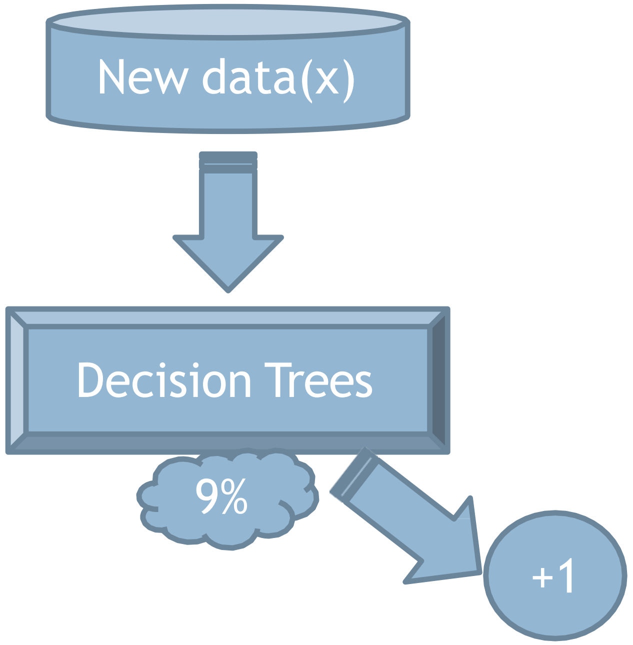

Problem Statement: What is the estimated monthly expense of a family in our city.
An Eminent Professor built a model
Vs.
100 Assessment Professors built 100 models
M1
One Single Prediction
$6500
M1
M2
M3
......
M100
Average of all 100 predictions
$7200

What is Ensemble Learning

What is Ensemble Learning

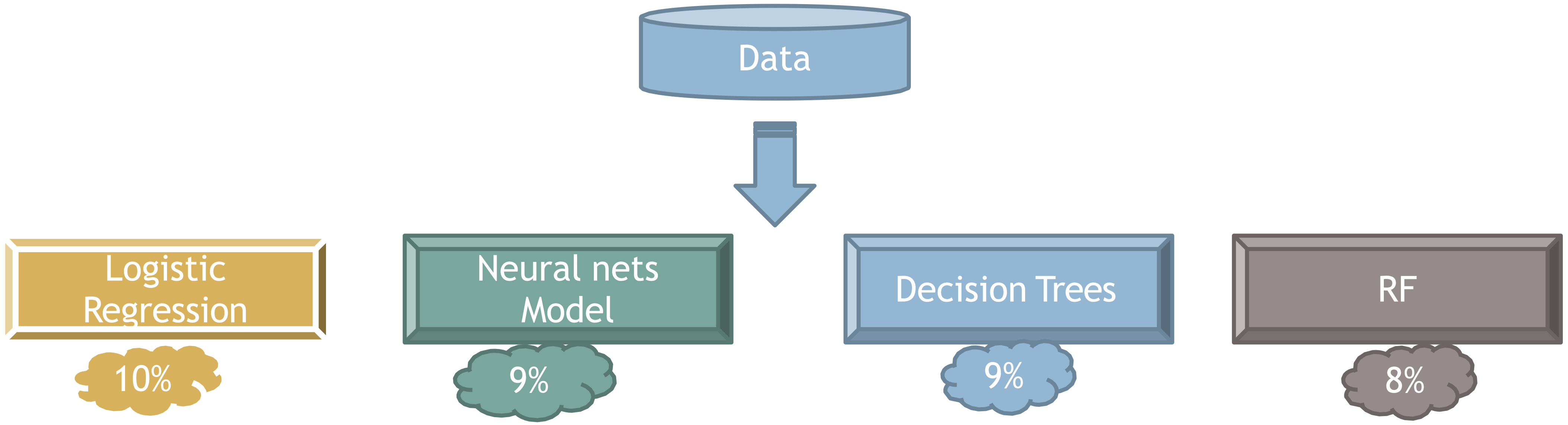
- Imagine a classifier problem, there are two classes +1 & -1 in the target
- Imagine that we built a best possible decision tree, it has 91% accuracy
- Let x be the new data point and our decision tree predicts it to be +1. Is there a way we can do better than 91% by using the same data
- Lets build 3 more models on the same data. And see we can improve the performance





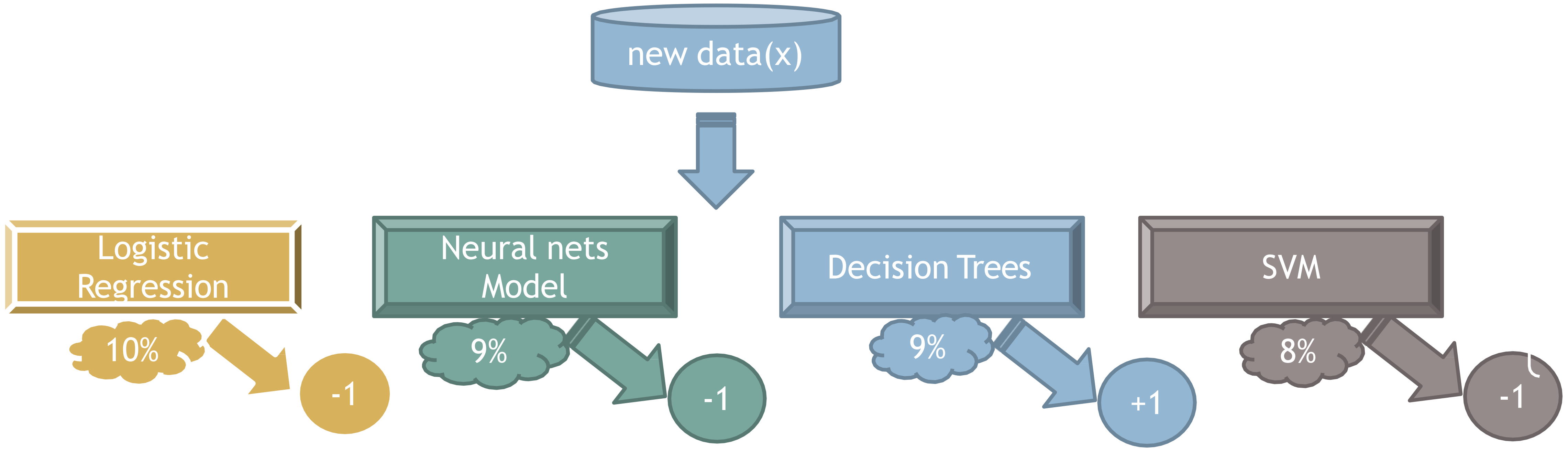
What is Ensemble Learning

- We have four models on the same dataset, Each of them have different accuracy. But unfortunately there seem to be no real improvement in the accuracy.



What is Ensemble Learning

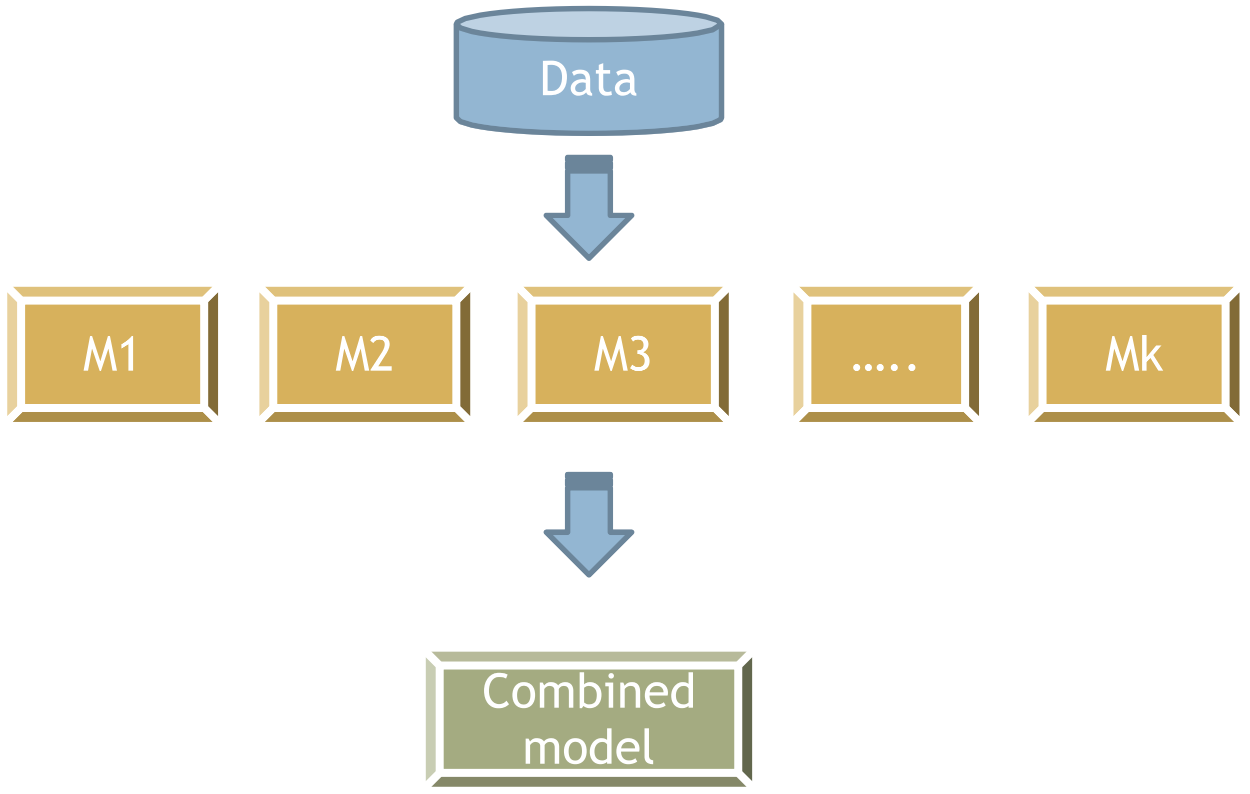
- What about prediction of the data point x?
- Except the decision tree, the rest all algorithms are predicting the class of x as -1
- Intuitively we would like to believe that the class of x is -1
- The combined voting model seem to be having less error than each of the individual models.
- This is the actual philosophy of ensemble learning



Ensemble Models

Ensemble Models

- Obtaining a better predictions using multiple models on the same dataset
- Not every time it is possible to find single best fit model for our data, ensemble model combines multiple models to come up with one consolidated model
- Ensemble models work on the principle that multiple moderately accurate models can give us a highly accurate model
- Understandably, the Building and Evaluating the ensemble models is computationally expensive
- Build one really good model is the usual statistical approach. Build many models and average the results is the philosophy of Ensemble learning



Why Ensemble technique works?

- Imagine three models
  - M1 with an error rate of 10%
  - M2 with an error rate of 10%
  - M3 with an error rate of 10%
- The three models have to be independent, we can’t build the same model three times and expect the error to reduce. Any changes to the modeling technique in model -1 should not impact model-2
- In this scenario, the worst ensemble model will have 10% error rate

Types of Ensemble Models

- The above example is a very primitive type of ensemble model. There are better and statistically stronger ensemble methods that will yield better results
- Two most popular ensemble methodologies are
  - Bagging
  - Boosting

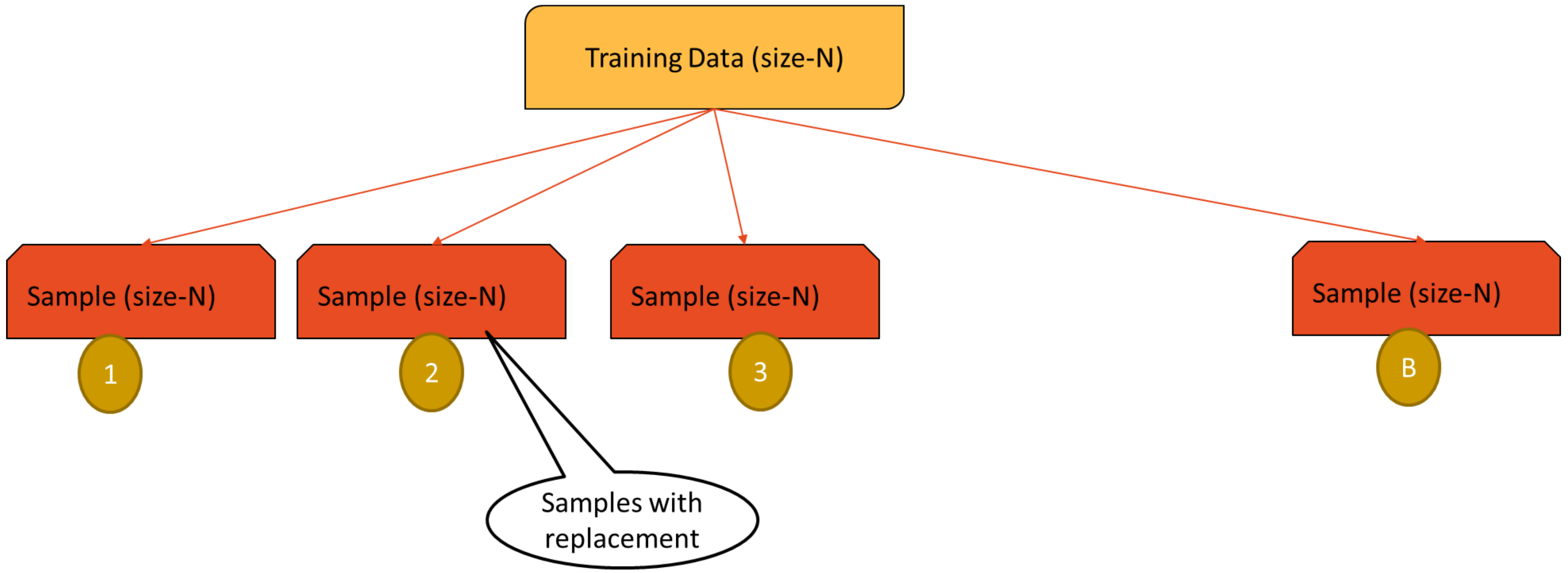
Bagging

Bagging

- Take multiple boot strap samples from the population and build classifiers on each of the samples. For prediction take mean or mode of all the individual model predictions.
- Bagging has two major parts 1) Boot strap sampling 2) Aggregation of learners
- **Bagging** = **B**ootstrap **Agg**regat**ing**
- In Bagging we combine many unstable models to produce a stable model. Hence the predictors will be very reliable(less variance in the final model).

Boot strapping

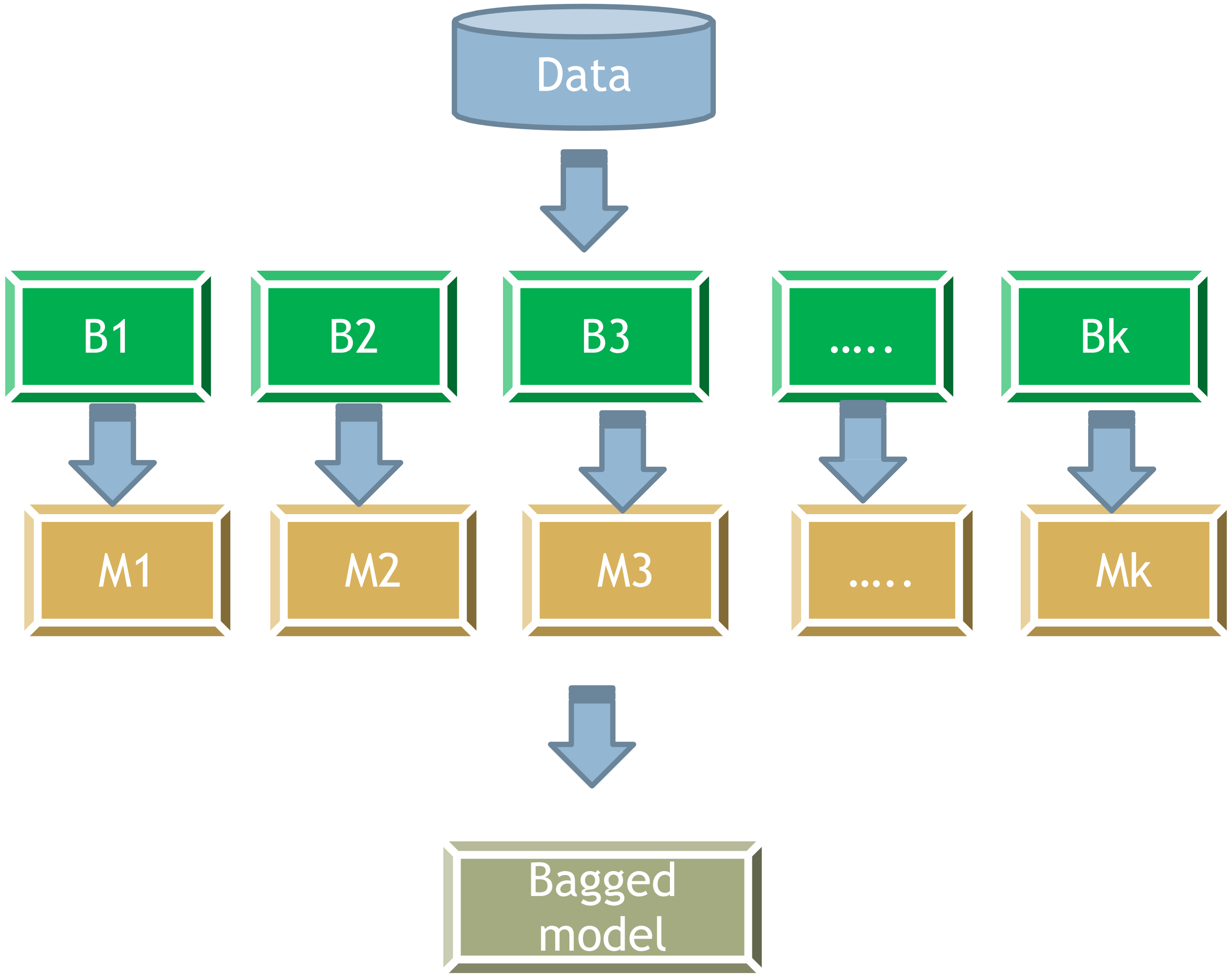
- We have a training data is of size N
- Draw random sample with replacement of size N – This gives a new dataset, it might have repeated observations, some observations might not have even appeared once.
- We are selecting records one-at–a-time, returning each selected record back in the population, giving it a chance to be selected again
- Create B such new datasets. These are called boot strap datasets



The Bagging Algorithm

The Bagging Algorithm

- The training dataset D
- Draw k boot strap sample sets from dataset D
- For each boot strap sample i
  - Build a classifier model M_i
  - We will have total of k classifiers M_1 , M_2 ,….. M_k
  - Vote over for the final classifier output and take the average for regression output



Why Bagging works

- We are selecting records one-at-a-time, returning each selected record back in the population, giving it a chance to be selected again
- Note that the variance in the consolidated prediction is reduced, if we have independent samples. That way we can reduce the unavoidable errors made by the single model.
- In a given boot strap sample, some observations have chance to select multiple times and some observations might not have selected at all.
- There a proven theory that boot strap samples have only 63% of overall population and rest 37% is not present.
- So the data used in each of these models is not exactly same, This makes our learning models independent. This helps our predictors have the uncorrelated errors.
- Finally the errors from the individual models cancel out and give us a better ensemble model with higher accuracy
- Bagging is really useful when there is lot of variance in our data

Random Forest

Random Forest

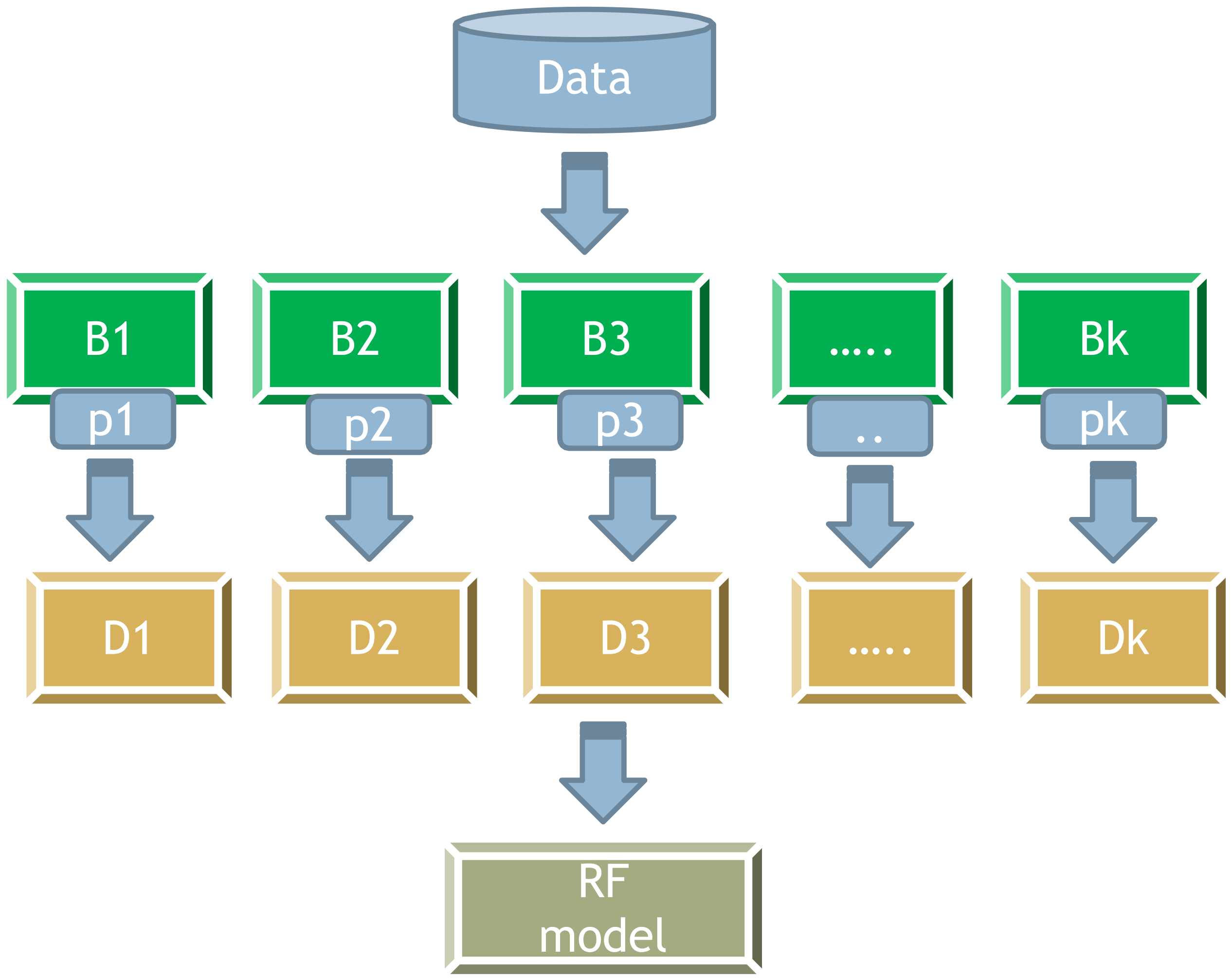
- Random forest is a specific case of bagging methodology. Bagging on decision trees is random forest
- Like many trees form a forest, many decision tree model together form a Random Forest model

Random Forest

- In random forest we induce two types of randomness
  - Firstly, we take the boot strap samples of the population and build decision trees on each of the sample.
  - While building the individual trees on boot strap samples, we take a subset of the features randomly
- Random forests are very stable they are as good as NN and SVMs sometimes better

Random Forest algorithm

- The training dataset D with t number of features
- Draw k boot strap sample sets from dataset D
- For each boot strap sample i
  - Build a decision tree model M_i using only p number of features (where p<<t)
  - Each tree has maximal strength they are fully grown and not pruned.
  - We will have total of k decision treed M_1 , M_2 ,….. M_k; Each of these trees are built on reactively different training data and different set of features
  - Vote over for the final classifier output and take the average for regression output



The Random Factors in Random Forest

- We need to note the most important aspect of random forest, i.e inducing randomness into the bagging of trees. There are two major sources of randomness
  - Randomness in data: Boot strapping, this will make sure that any two samples data is somewhat different
  - Randomness in features: While building the decision trees on boot strapped samples we consider only a random subset of features.

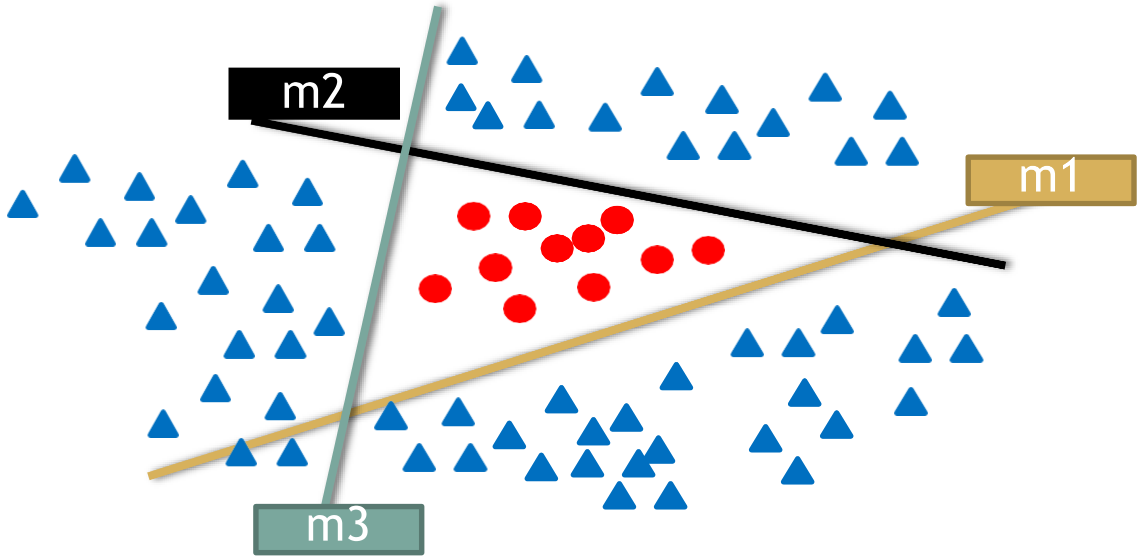
Why to induce the randomness?

- The major trick of ensemble models is the independence of models.
- If we take the same data and build same model for 100 times, we will not see any improvement
- To make all our decision trees independent, we take independent samples set and independent features set
- As a rule of thumb we can consider square root of the number features, if ‘t’ is very large else p=t/3

Why Random ForestWorks

- For a training data with 20 features we are building 100 decision trees with 5 features each, instated of single great decision.
- The individual trees may be weak classifiers.
- Its like building weak classifiers on subsets of data. The grouping of large sets of random trees generally produces accurate models.

Why Random Forest Works



- In this example we have three simple classifiers.
  - m1 classifies anything above the line as +1 and below as -1
  - m2 classifies all the points above the line as -1 and below as +1
  - m3 classifies everything on the left as -1 and right as +1
- Each of these models have fair amount of misclassification error.
- All these three weak models together make a strong model.

LAB: Random Forest

LAB: Random Forest

- Dataset: /Car Accidents IOT/Train.csv
- Build a decision tree model to predict the fatality of accident
- Build a decision tree model on the training data.
- On the test data, calculate the classification error and accuracy.
- Build a random forest model on the training data.
- On the test data, calculate the classification error and accuracy.
- What is the improvement of the Random Forest model when compared with the single tree?

Code: Random Forest

```
import pandas as pd
import sklearn as sk
import numpy as np
import scipy as sp

#Importing dataset
car_train=pd.read_csv("D:\\Google Drive\\Training\\Datasets\\Car Accidents IOT\\train.csv")
car_test=pd.read_csv("D:\\Google Drive\\Training\\Datasets\\Car Accidents IOT\\test.csv")

from sklearn import tree

var=list(car_train.columns[1:22])
c=car_train[var]
d=car_train['Fatal']

###building Decision tree on the training data ####
clf =tree.DecisionTreeClassifier()
clf.fit(c,d)
```

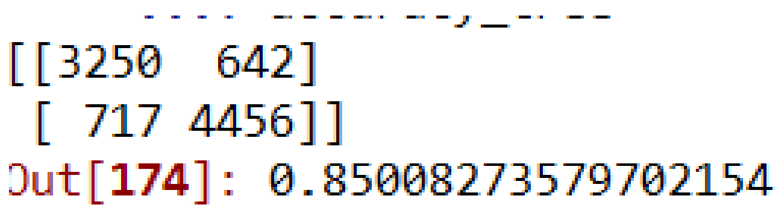
Import data and build a decision tree

Code: Random Forest

```
#####predicting on test data ####
tree_predict=clf.predict(car_test[var])

from sklearn.metrics import confusion_matrix###for using confusion matrix###
cm1=confusion_matrix(car_test[['Fatal']],tree_predict)
print(cm1)

#####from confusion matrix calculate accuracy
total1=sum(sum(cm1))
accuracy_tree=(cm1[0,0]+cm1[1,1])/total1
accuracy_tree
```



```
[[3250  642]
 [ 717 4456]]
Out[174]: 0.85008273579702154
```

Accuracy of the tree

Code: Random Forest

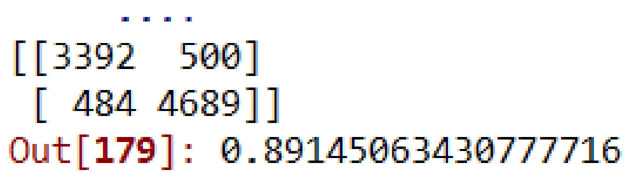
```
####Building RandomForest Model
from sklearn.ensemble import RandomForestClassifier
forest=RandomForestClassifier(n_estimators=10,    min_samples_split=2,  min_samples_leaf=1)

forest.fit(c,d)

forestpredict_test=forest.predict(car_test[var])
e=car_test['Fatal']

###check the accuracy on test data
from sklearn.metrics import confusion_matrix###for using confusion matrix###
cm2=confusion_matrix(car_test[['Fatal']],forestpredict_test)
print(cm2)
total2=sum(sum(cm2))
#####from confusion matrix calculate accuracy
accuracy_forest=(cm2[0,0]+cm2[1,1])/total2
accuracy_forest
```

Random forest model and its accuracy



```
      ....
[[3392  500]
 [ 484 4689]]
Out[179]: 0.8914506343077716
```

Boosting

Contents

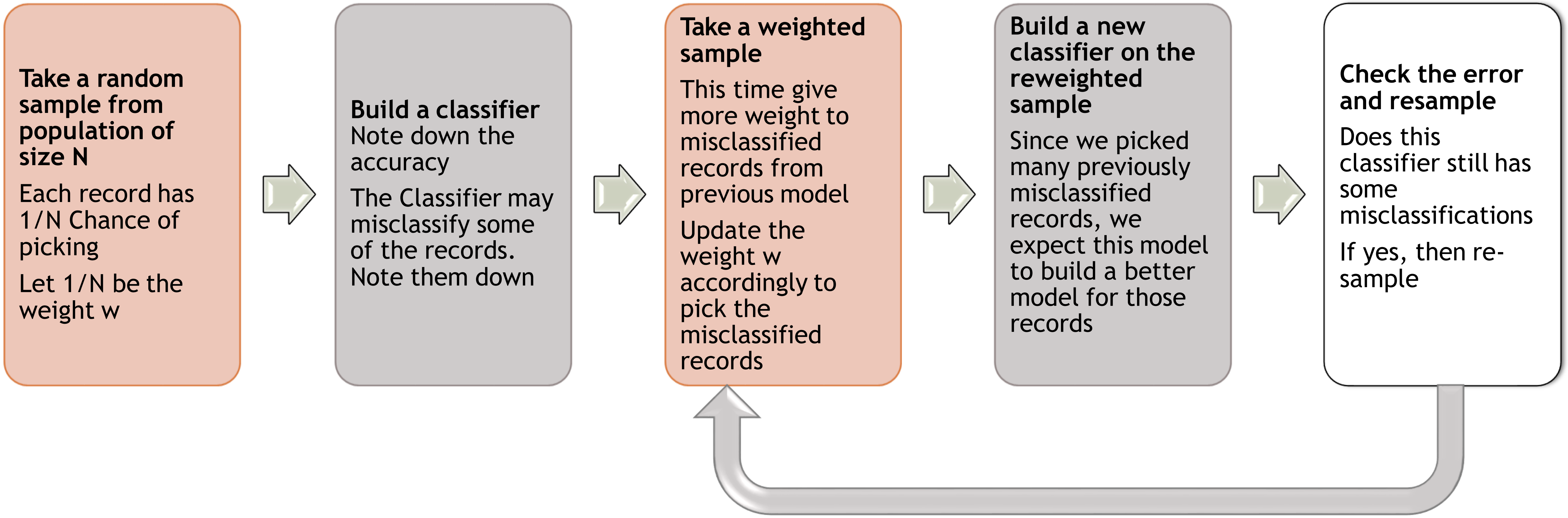
Boosting

- Boosting is one more famous ensemble method
- Boosting uses a slightly different techniques to that of bagging.
- Boosting is a well proven theory that works really well on many of the machine learning problems like speech recognition
- If bagging is wisdom of crowds then boosting is wisdom of crowds where each individual is given some weight based on their expertise

Boosting

- Boosting in general decreases the bias error and builds strong predictive models.
- Boosting is an iterative technique. We adjust the weight of the observation based on the previous classification.
- If an observation was classified incorrectly, it tries to increase the weight of this observation and vice versa.

Boosting Main idea



Take a random sample from population of size N

Each record has 1/N Chance of picking

Let 1/N be the weight w

Build a classifier

Note down the accuracy

The Classifier may misclassify some of the records. Note them down

Take a weighted sample

This time give more weight to misclassified records from previous model

Update the weight w accordingly to pick the misclassified records

Build a new classifier on the reweighted sample

Since we picked many previously misclassified records, we expect this model to build a better model for those records

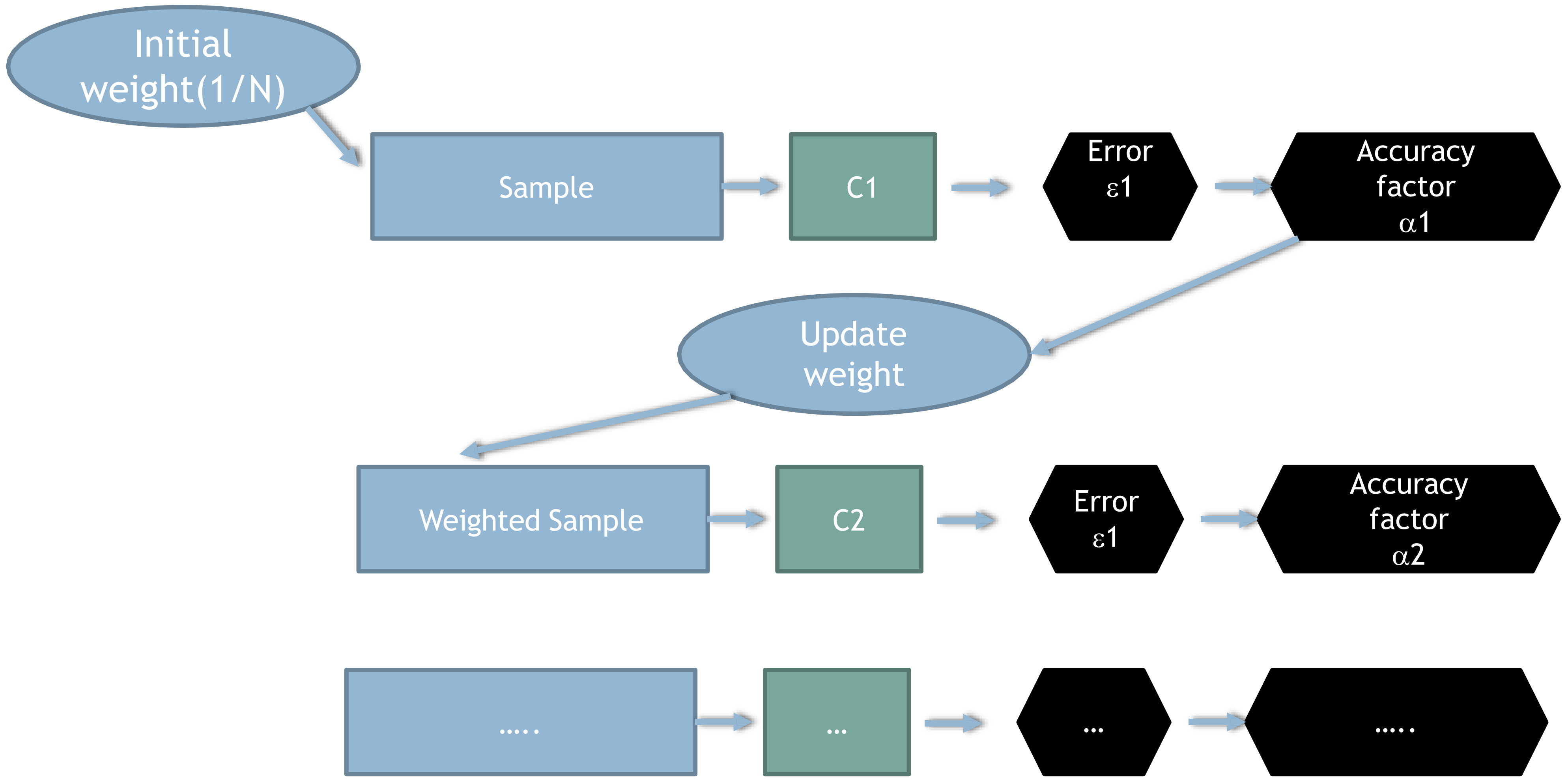
Check the error and resample

Does this classifier still has some misclassifications

If yes, then re-sample

Final Weighted Classifier $C= \sum \alpha_i c_i$

Boosting Main idea



How weighted samples aretaken

Data	1	2	3	4	5	6	7	8	9	10
Class	-	-	+	+	-	+	-	-	+	+
Predicted Class M1	-	-	-	-	-	-	-	-	+	+
M1 Result	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓

Weighted Sample1	1	2	3	4	5	6	7	4	3	6
Class	-	-	+	+	-	+	-	+	+	+
Predicted Class M2	-	-	+	+	+	+	+	+	+	+
M2 Result	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓

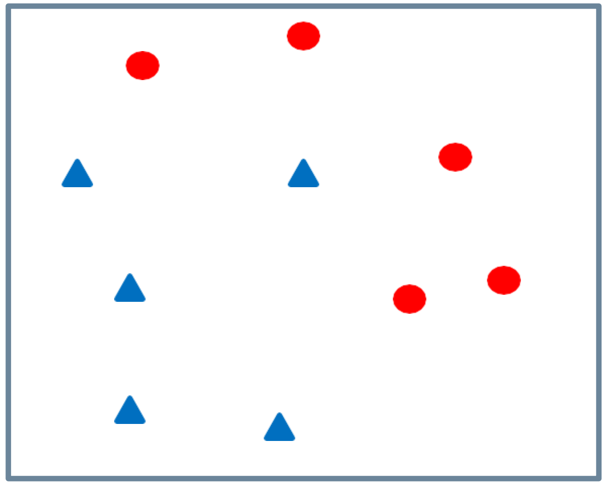
Weighted Sample2	6	5	3	4	5	6	7	7	5	7
Class	+	-	+	+	-	+	-	-	-	-
Predicted Class M3	+	-	+	+	-	+	-	-	-	-
M3 Result	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

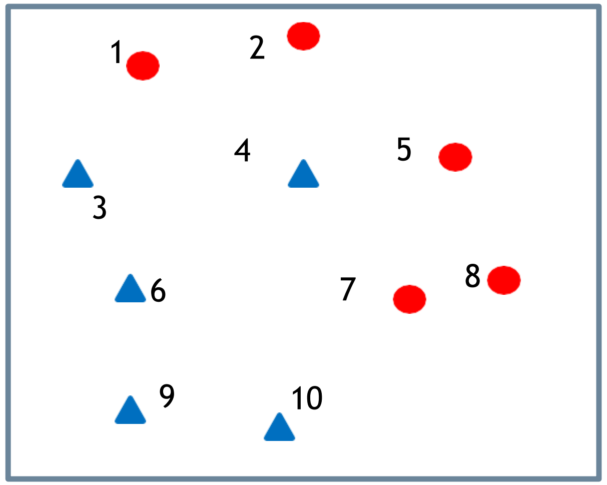
Boosting illustration

Boosting illustration

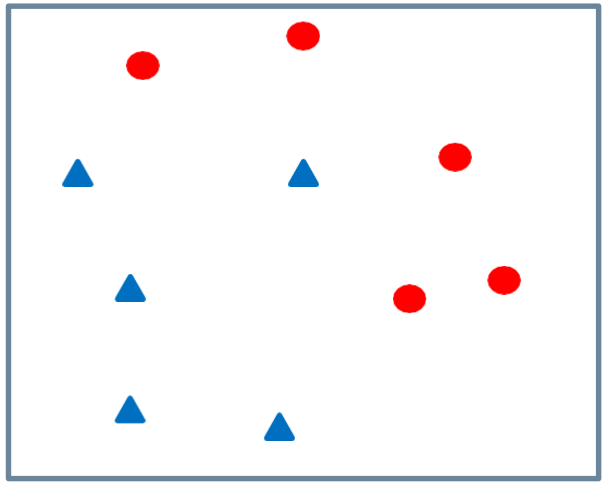
Below is the training data and their class
We need to take a note of record numbers, they will help us in weighted sampling later

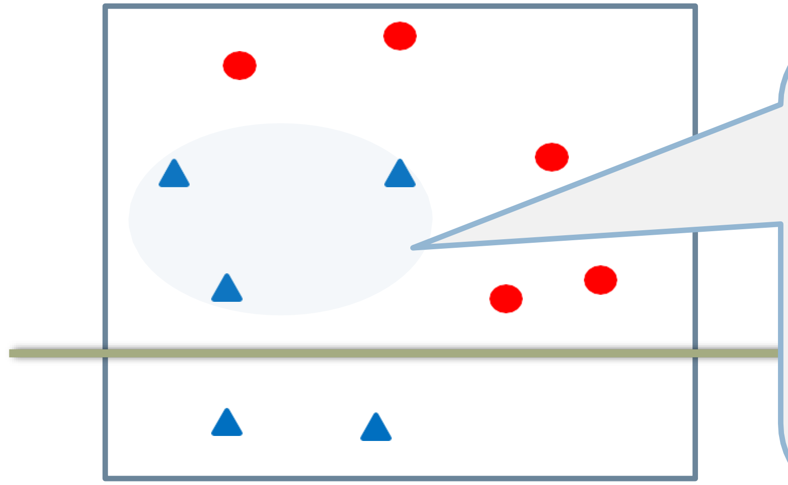
Data Points	1	2	3	4	5	6	7	8	9	10
Class	-	-	+	+	-	+	-	-	+	+





Boosting illustration

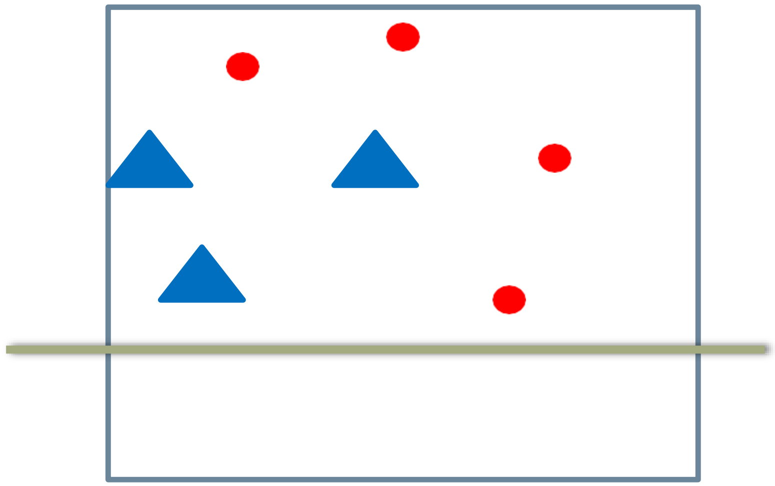


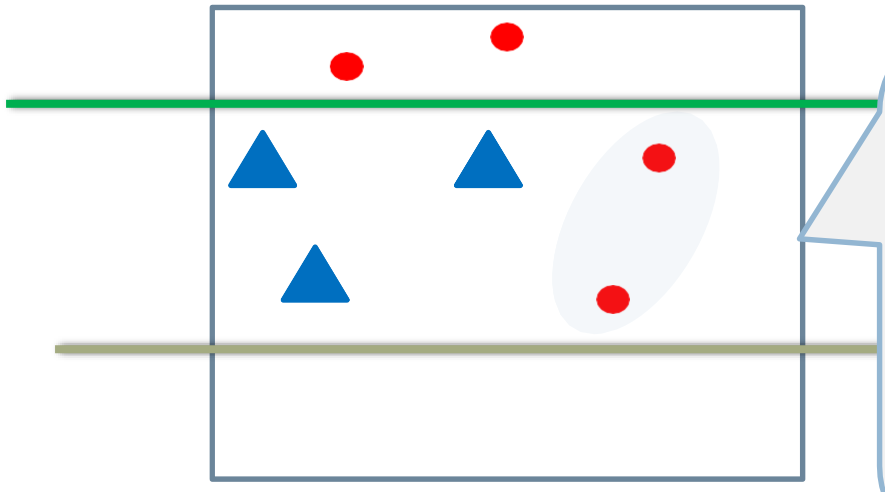


- Model M1 is built, anything above the line is – and below the line is +
- 3 out of 10 are misclassified by the model M1
- These data points will be given more weight in the re-sampling step
- We may miss out on some of the correctly classified records

Data	1	2	3	4	5	6	7	8	9	10
Class	-	-	+	+	-	+	-	-	+	+
Predicted Class M1	-	-	-	-	-	-	-	-	+	+
M1 Result	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓

Boosting illustration

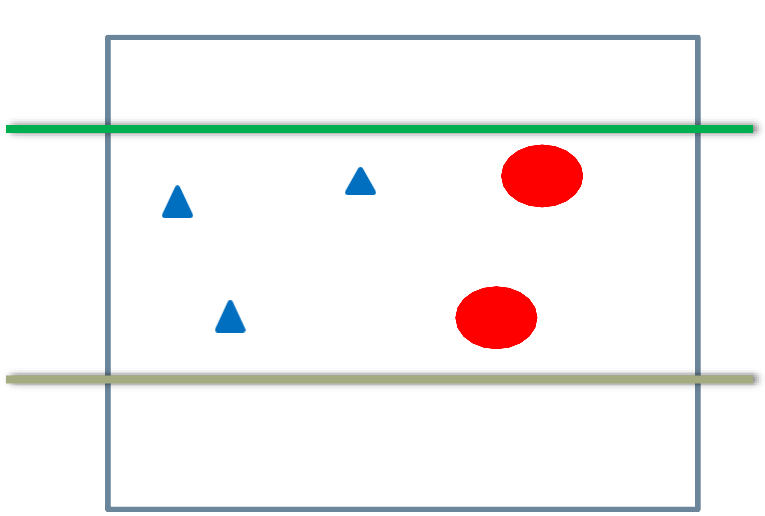


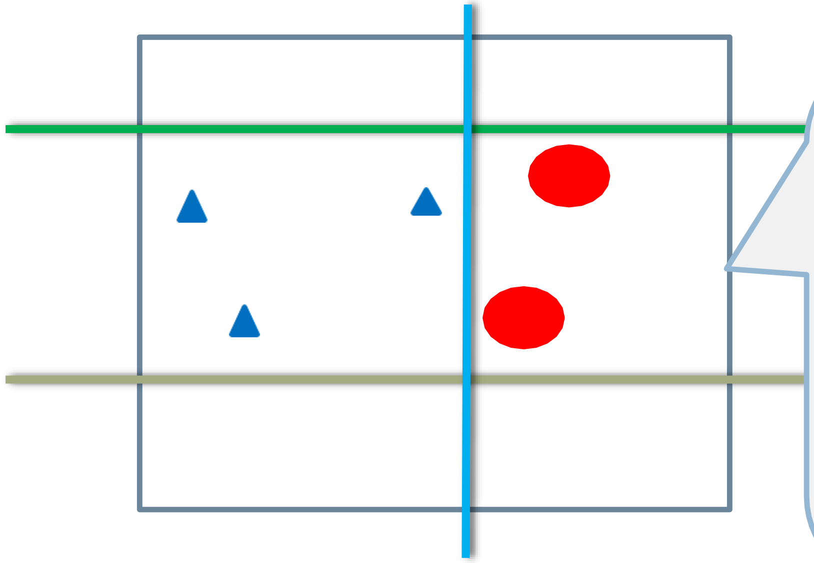


- The misclassified points 3,4,& 6 have appeared more often than others in this weighted sample.
- The sample points 9,10 didn't appear
- M2 is built on this data. Anything above the line is – and below the line is +
- M2 is classifying the points 5 & 7 incorrectly.
- They will be given more weight in the next sample

Weighted Sample1	1	2	3	4	5	6	7	4	3	6
Class	-	-	+	+	-	+	-	+	+	+
Predicted Class M2	-	-	+	+	+	+	+	+	+	+
M2 Result	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓

Boosting illustration

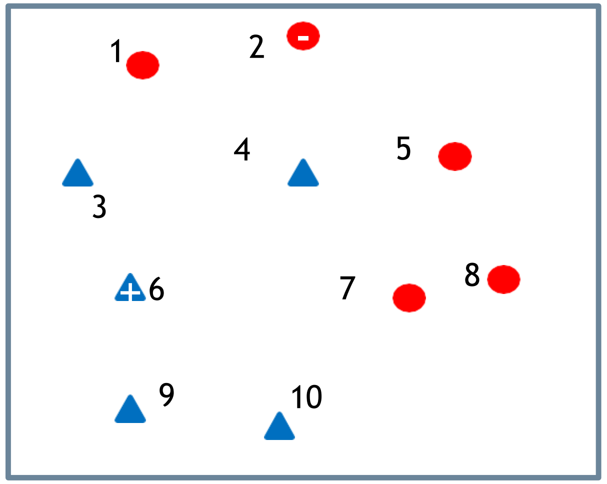


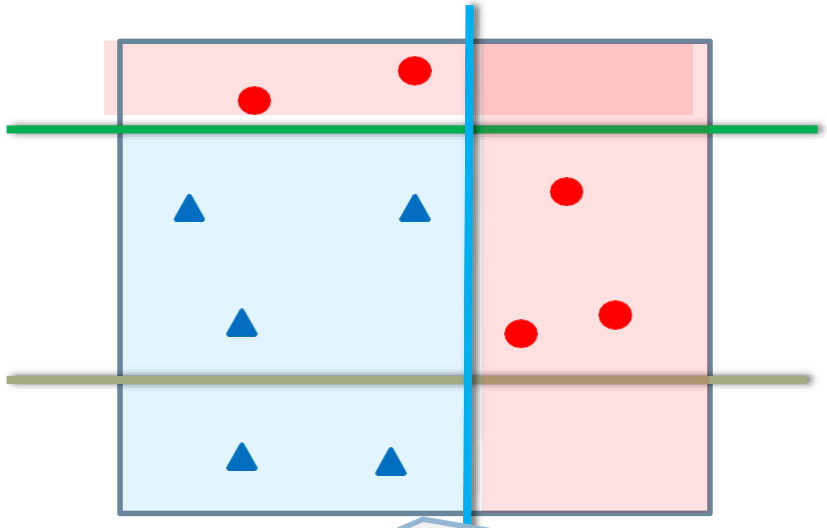


- The misclassified points 5 & 7 have appeared more often than others in this weighted sample.
- M3 is built on this data. Anything above the line is – and below the line is +
- M3 is now classifying everything correctly

Weighted Sample2	6	5	3	4	5	6	7	7	5	7
Class	+	-	+	+	-	+	-	-	-	-
Predicted Class M3	+	-	+	+	-	+	-	-	-	-
M3 Result	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Boosting illustration





- The final model now will be picked on weighted Votes.
- For a given data point more than 2 models seam to be indicating the right class.
- For example take point 6, it is classified as – by M1, + by M2 and + by M3, final result will be +
- Similarly take a point 2, it will be classified as –by M1, -by M2 and + by M3, final result will be -
- So the final weighted combination of three models predictions will yield in accurate perdition.

Theory behind Boosting Algorithm

Theory behind Boosting Algorithm

- Take the dataset Build a classifier C_m and find the error
- Calculate error rate of the classifier
  - Error rate of ε_m
    - $= \sum w_i I(y_i \neq C_m x_i) / (\sum w_i)$
    - =Sum of misclassification weight / sum of sample weights
- Calculate an intermediate factor called α. It analogous to accuracy rate of the model. It will be later used in weight updating. It is derived from error
  - $\alpha_m = \log((1-\varepsilon_m)/\varepsilon_m)$

Theory behind Boosting Algorithm..contd

- Update weights of each record in the sample using the α factor. The indicator function will make sure that the misclassifications are given more weight
  - For i=1,2….N
    - $w_{i+1} = w_i e^{\alpha_m I(y_i \neq C_m x)}$ ( )
    - Renormalize so that sum of weights is 1
- Repeat this model building and weight update process until we have no misclassification
- Final collation is done by voting from all the models. While taking the votes, each model is weighted by the accuracy factor α
  - C= $sign(\sum \alpha_i C_i(x))$

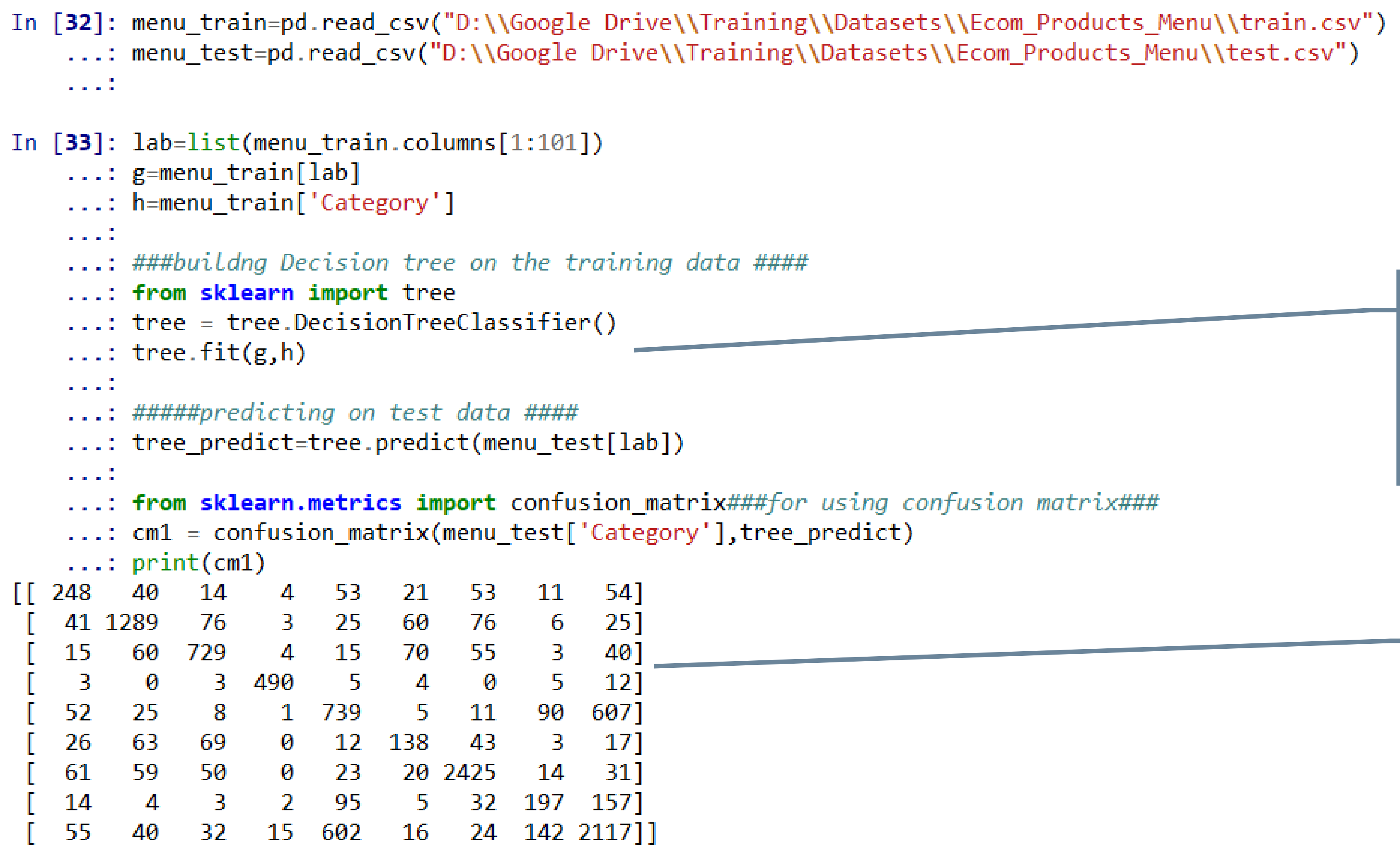
Gradient Boosting

- **Ada boosting**
  - Adaptive Boosting
  - Till now we discussed Ada boosting technique. Here we give high weight to misclassified records.
- **Gradient Boosting**
  - Similar to Ada boosting algorithm.
  - The approach is same but there are slight modifications during re-weighted sampling.
  - We update the weights based on misclassification rate and gradient
  - Gradient boosting serves better for some class of problems like regression.

LAB: Boosting

- Ecom products classification. Rightly categorizing the items based on their detailed feature specifications. More than 100 specifications have been collected.
- Data: Ecom_Products_Menu/train.csv
- Build a decision tree model and check the training and testing accuracy
- Build a boosted decision tree.
- Is there any improvement from the earlier decision tree

Code: Boosting



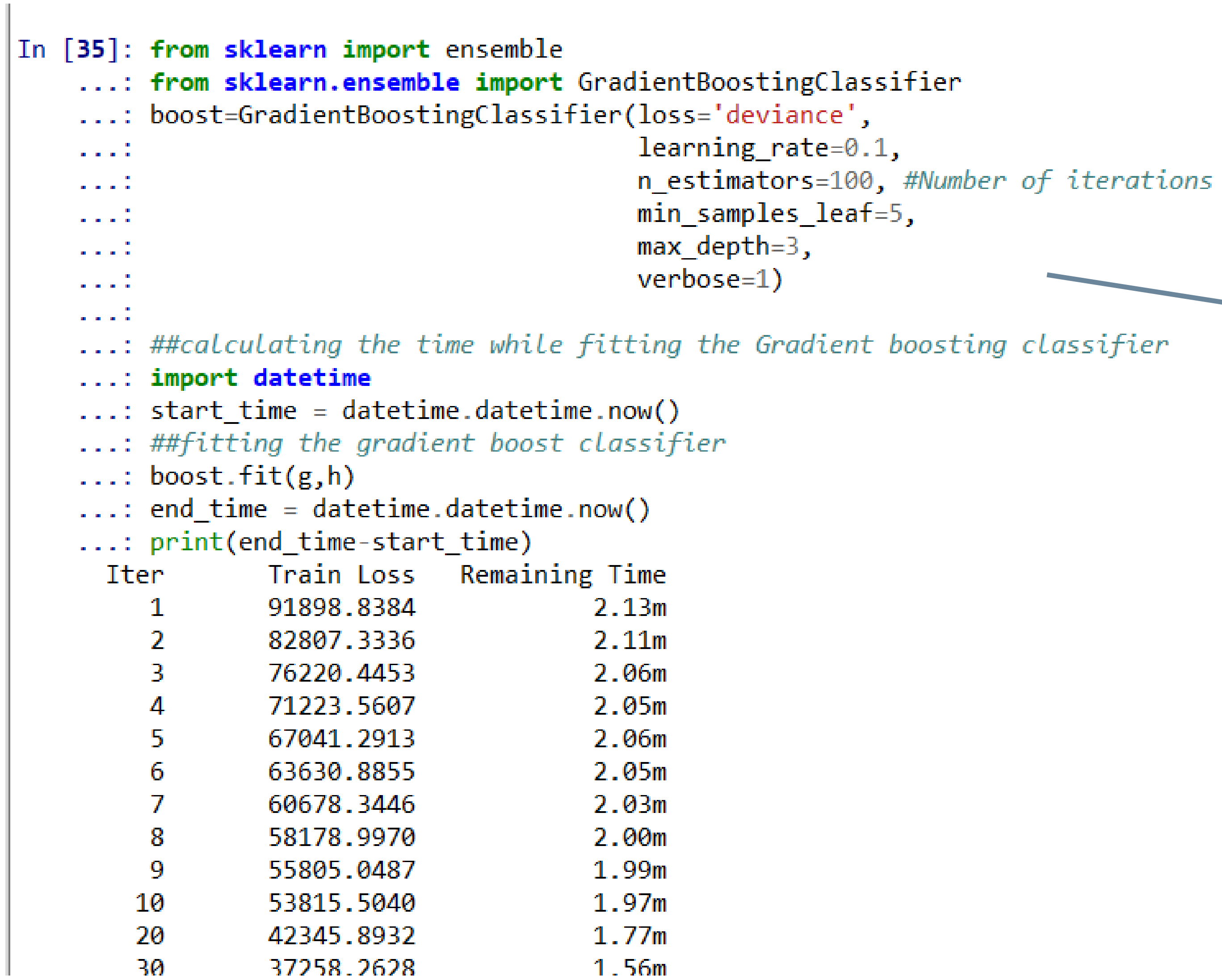
```
In [32]: menu_train=pd.read_csv("D:\\Google Drive\\Training\\Datasets\\Ecom_Products_Menu\\train.csv")
    ...: menu_test=pd.read_csv("D:\\Google Drive\\Training\\Datasets\\Ecom_Products_Menu\\test.csv")
    ...:

In [33]: lab=list(menu_train.columns[1:101])
    ...: g=menu_train[lab]
    ...: h=menu_train['Category']
    ...:
    ...: ###buildng Decision tree on the training data ####
    ...: from sklearn import tree
    ...: tree = tree.DecisionTreeClassifier()
    ...: tree.fit(g,h)
    ...:
    ...: #####predicting on test data ####
    ...: tree_predict=tree.predict(menu_test[lab])
    ...:
    ...: from sklearn.metrics import confusion_matrix###for using confusion matrix###
    ...: cm1 = confusion_matrix(menu_test['Category'],tree_predict)
    ...: print(cm1)
[[ 248   40   14    4   53   21   53   11   54]
 [  41 1289   76    3   25   60   76    6   25]
 [  15   60  729    4   15   70   55    3   40]
 [   3    0    3  490    5    4    0    5   12]
 [  52   25    8    1  739    5   11   90  607]
 [  26   63   69    0   12  138   43    3   17]
 [  61   59   50    0   23   20 2425   14   31]
 [  14    4    3    2   95    5   32  197  157]
 [  55   40   32   15  602   16   24  142 2117]]
```

Import data and build basic decision tree model

Accuracy

Code: Boosting



```
In [35]: from sklearn import ensemble
    ...: from sklearn.ensemble import GradientBoostingClassifier
    ...: boost=GradientBoostingClassifier(loss='deviance',
    ...:                                 learning_rate=0.1,
    ...:                                 n_estimators=100, #Number of iterations
    ...:                                 min_samples_leaf=5,
    ...:                                 max_depth=3,
    ...:                                 verbose=1)
    ...:
    ...: ##calculating the time while fitting the Gradient boosting classifier
    ...: import datetime
    ...: start_time = datetime.datetime.now()
    ...: ##fitting the gradient boost classifier
    ...: boost.fit(g,h)
    ...: end_time = datetime.datetime.now()
    ...: print(end_time-start_time)
      Iter       Train Loss   Remaining Time
         1       91898.8384            2.13m
         2       82807.3336            2.11m
         3       76220.4453            2.06m
         4       71223.5607            2.05m
         5       67041.2913            2.06m
         6       63630.8855            2.05m
         7       60678.3446            2.03m
         8       58178.9970            2.00m
         9       55805.0487            1.99m
        10       53815.5040            1.97m
        20       42345.8932            1.77m
        30       37258.2628            1.56m
```

GBM Model and iterations

Code: Boosting

```
In [39]: boost_predict=boost.predict(menu_test[lab])
    ...: from sklearn.metrics import confusion_matrix###for using confusion matrix###
    ...: cm2 = confusion_matrix(menu_test['Category'],boost_predict)
    ...: print(cm2)
    ...:
[[ 304   30    0    1   28    7   46    6   76]
 [  15 1460   31    1    3   19   44    0   28]
 [   3   40  851    2    0   27   38    0   30]
 [   0    0    0  498    1    0    0    1   22]
 [  22    6    5    0  665    0    3   20  817]
 [   7   71   82    0    4  148   29    0   30]
 [  38   36   33    1    3   10 2517    3   42]
 [   7    1    0    3   71    0   21  168  238]
 [  19    8    8    9  330    0    8   26 2635]]
```

```
boost_predict=boost.predict(menu_test[lab])
from sklearn.metrics import f1_score
f1_score(menu_test['Category'], boost_predict, average='micro')



0.78649200408302145
```

Boosted tree accuracy

When Ensemble doesn’t work?

When Ensemble doesn’twork?

- The models have to be independent, we can’t build the same model multiple times and expect the error to reduce.
- We may have to bring in the independence by choosing subsets of data, or subset of features while building the individual models
- Ensemble may backfire if we use dependent models that are already less accurate. The final ensemble might turn out to be even worse model.

When Ensemble doesn’twork?

- Yes, there is a small disclaimer in “Wisdom of Crowd” theory. We need good independent individuals.
- If we collate any dependent individuals with poor knowledge, then we might end up with an even worse ensemble.
- For example, we built three models, model-1 , model-2 are bad, model-3 is good. Most of the times ensemble will result the combined output of model-1 and model-2, based on voting

Conclusion

Conclusion

- Ensemble methods are most widely used methods these days. With advanced machines, its not really a huge task to build multiple models.
- Random forests are relatively fast, since we are building many small trees, it doesn’t put lot of pressure on the computing machine
- Random forest can also give the variable importance. We need to be careful with categorical features, random forests trend to give higher importance to variables with higher number of levels.
- Ensemble models are the final effort of a data scientist, while building the most suitable predictive model for the data