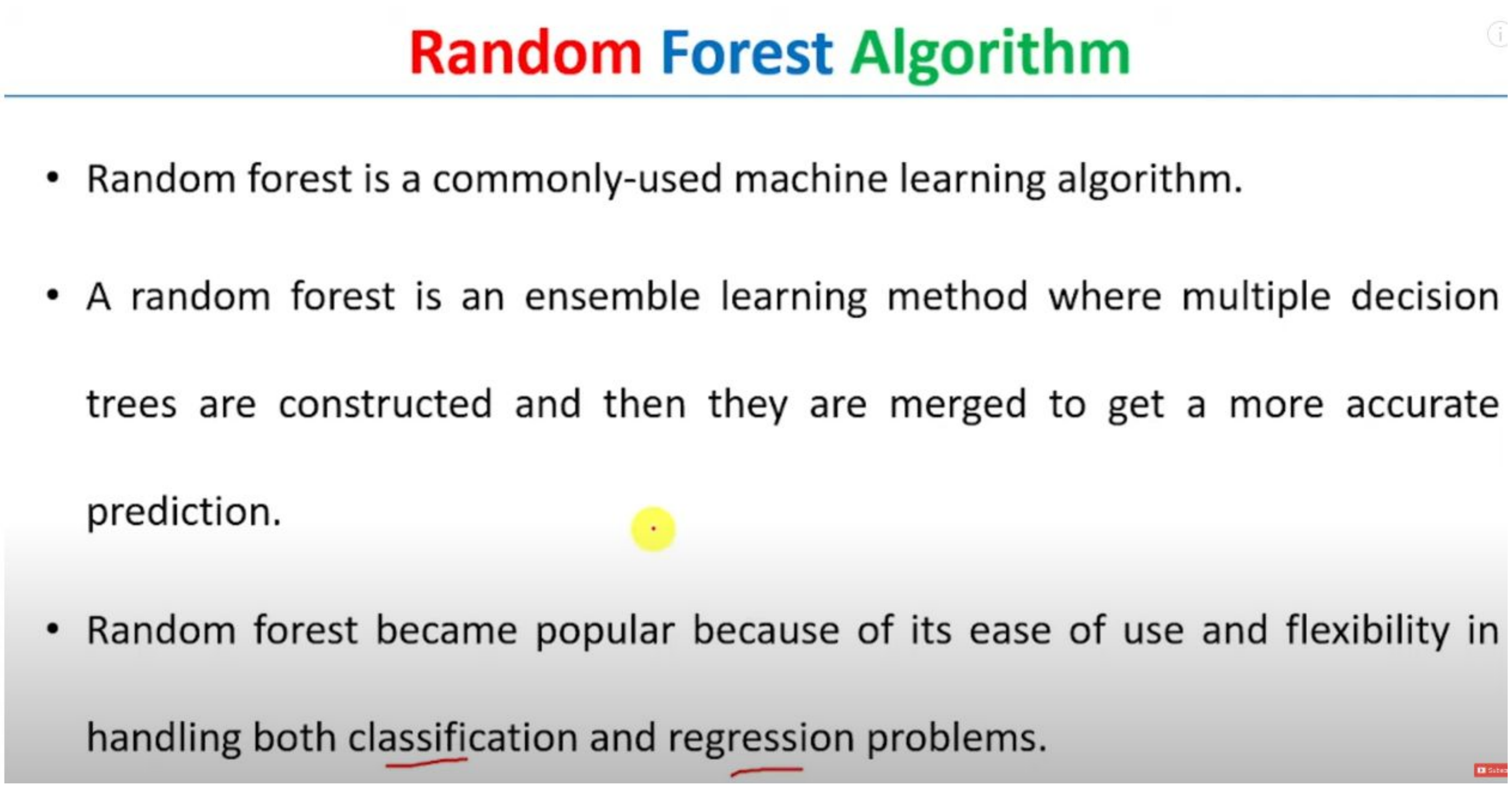
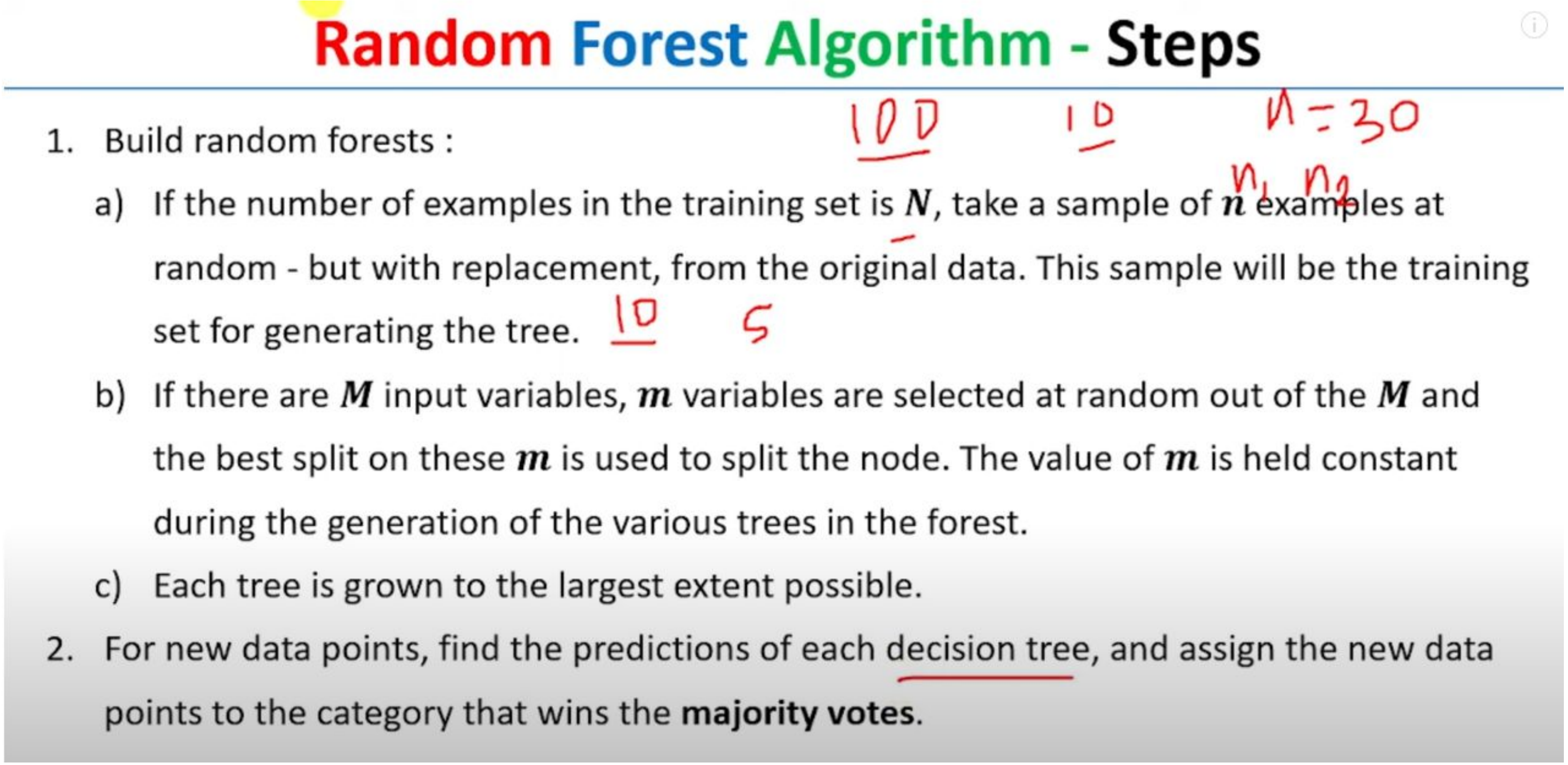
Lab program no 9a and 9b



Random Forest Algorithm

- Random forest is a commonly-used machine learning algorithm.
- A random forest is an ensemble learning method where multiple decision trees are constructed and then they are merged to get a more accurate prediction.
- Random forest became popular because of its ease of use and flexibility in handling both classification and regression problems.



Random Forest Algorithm - Steps

1. Build random forests : 100 10 $n=30$

   n_1 n_2

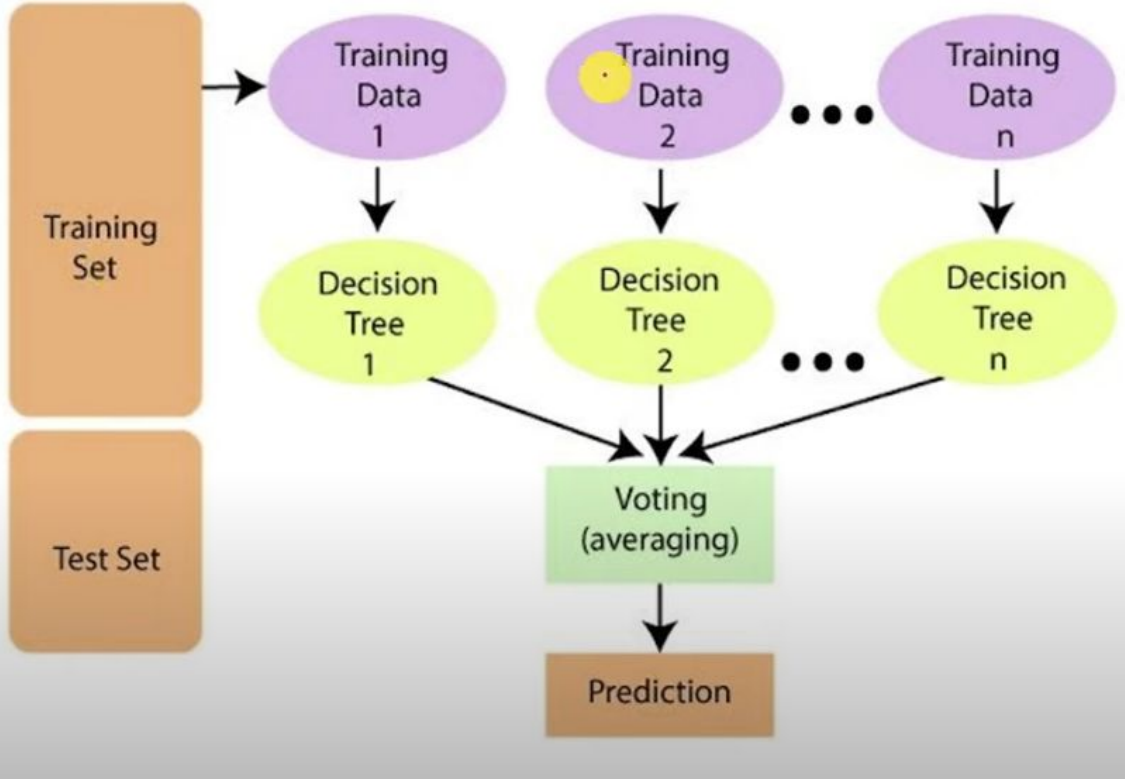
   a) If the number of examples in the training set is $\boldsymbol{N}$, take a sample of $\boldsymbol{n}$ examples at random - but with replacement, from the original data. This sample will be the training set for generating the tree. 10 5

   b) If there are $\boldsymbol{M}$ input variables, $\boldsymbol{m}$ variables are selected at random out of the $\boldsymbol{M}$ and the best split on these $\boldsymbol{m}$ is used to split the node. The value of $\boldsymbol{m}$ is held constant during the generation of the various trees in the forest.

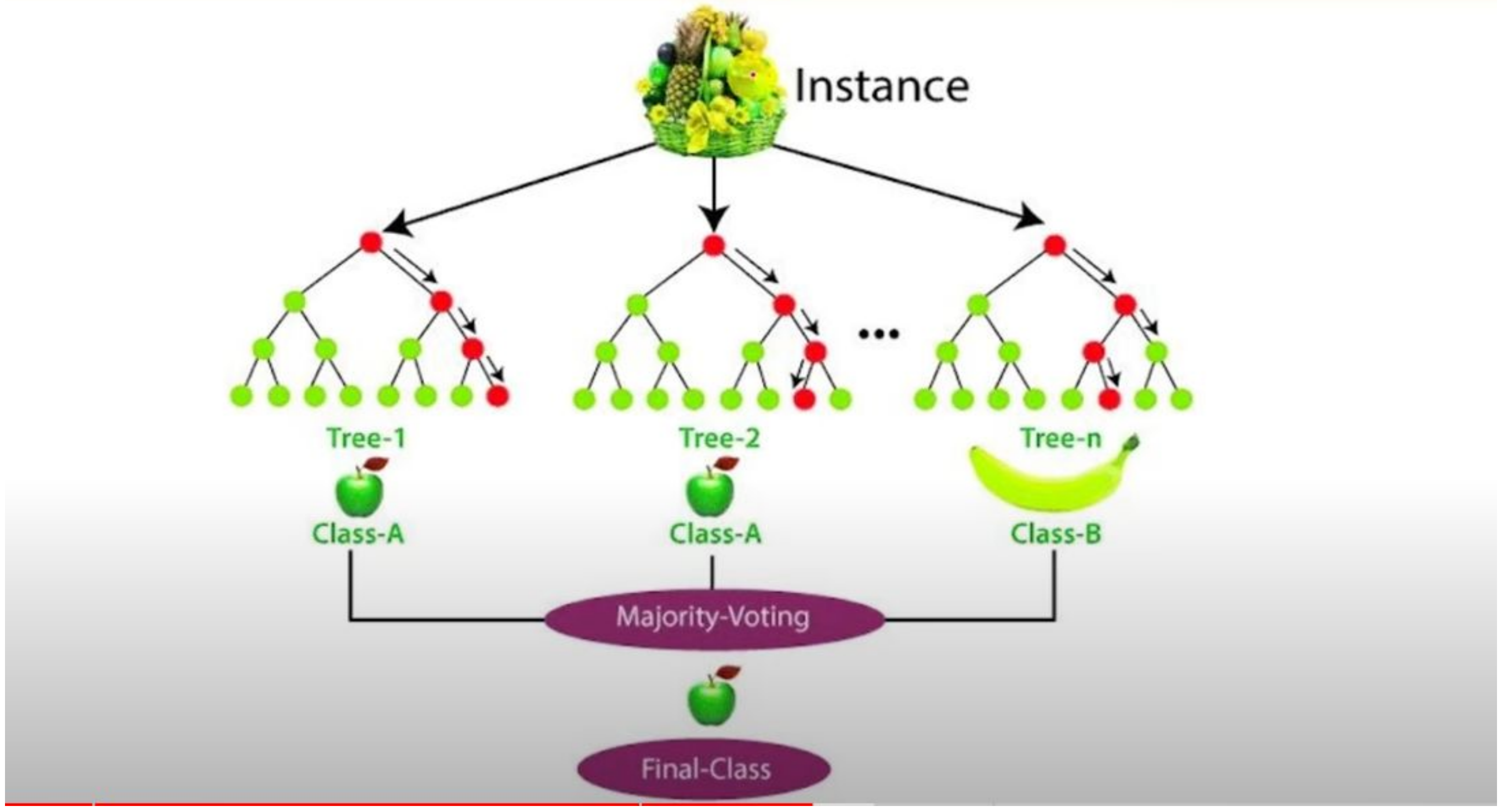
   c) Each tree is grown to the largest extent possible.

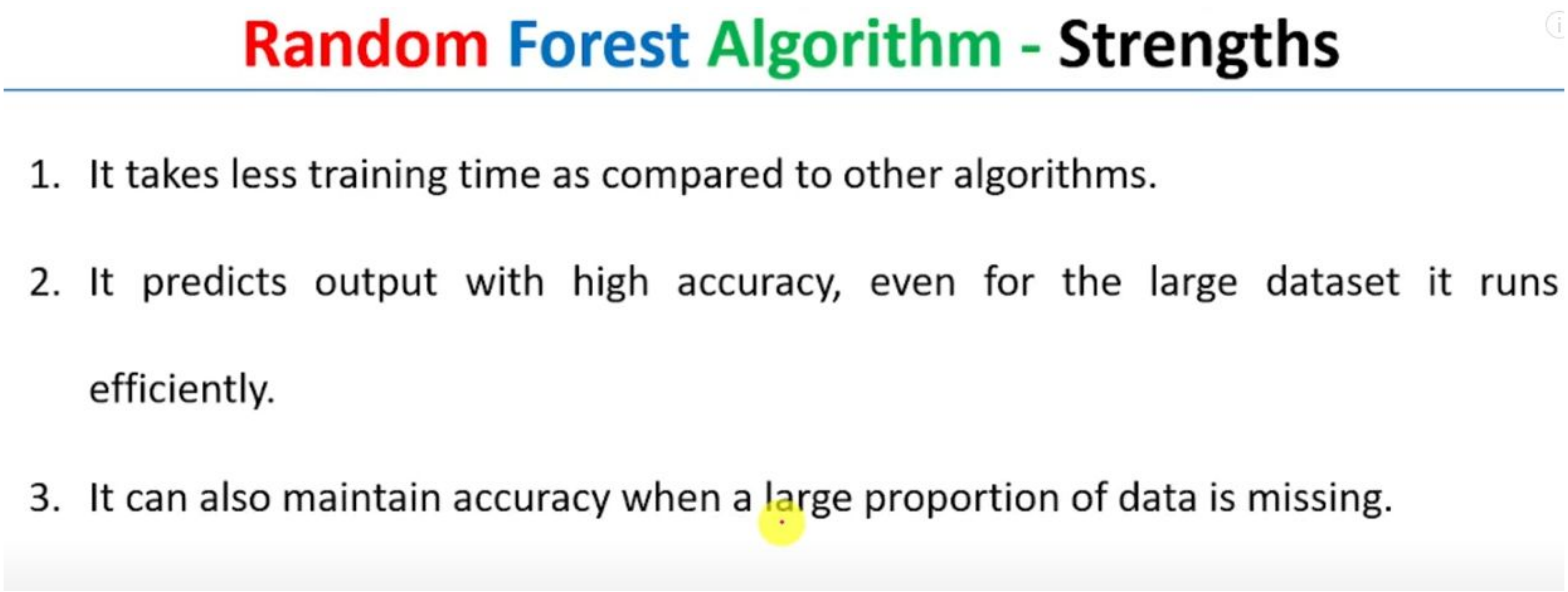
2. For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the **majority votes**.

Random Forest Algorithm - Steps



Random Forest Algorithm - Steps





Random Forest Algorithm - Strengths

1. It takes less training time as compared to other algorithms.
2. It predicts output with high accuracy, even for the large dataset it runs efficiently.
3. It can also maintain accuracy when a large proportion of data is missing.

Random Forest Algorithm - Weaknesses

1. A weakness of random forest algorithms is that when used for regression they cannot predict beyond the range in the training data, and that they may over-fit data sets that are particularly noisy.

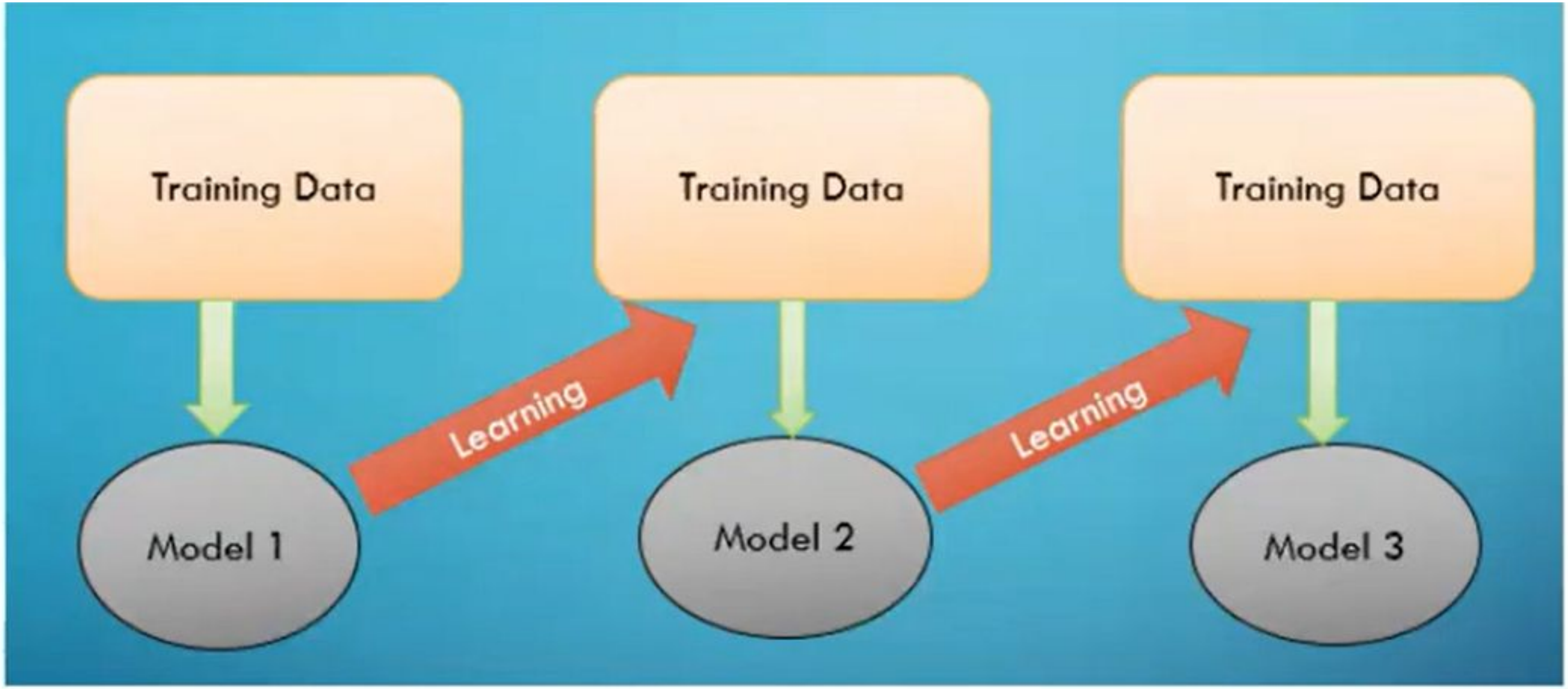
2. The sizes of the models created by random forests may be very large. It may take hundreds of megabytes of memory and may be slow to evaluate.

3. Random forest models are black boxes that are very hard to interpret.

Steps for implementation

- From sklearn.ensemble import randomforestclassifier
- Data preparation(csv file)//read csv file
- Separating column like independent variable and target column
  - Train test split
- Model training and evaluation
  - Create randomforestclassifier()
  - Fit
  - Predict
  - Accuracy

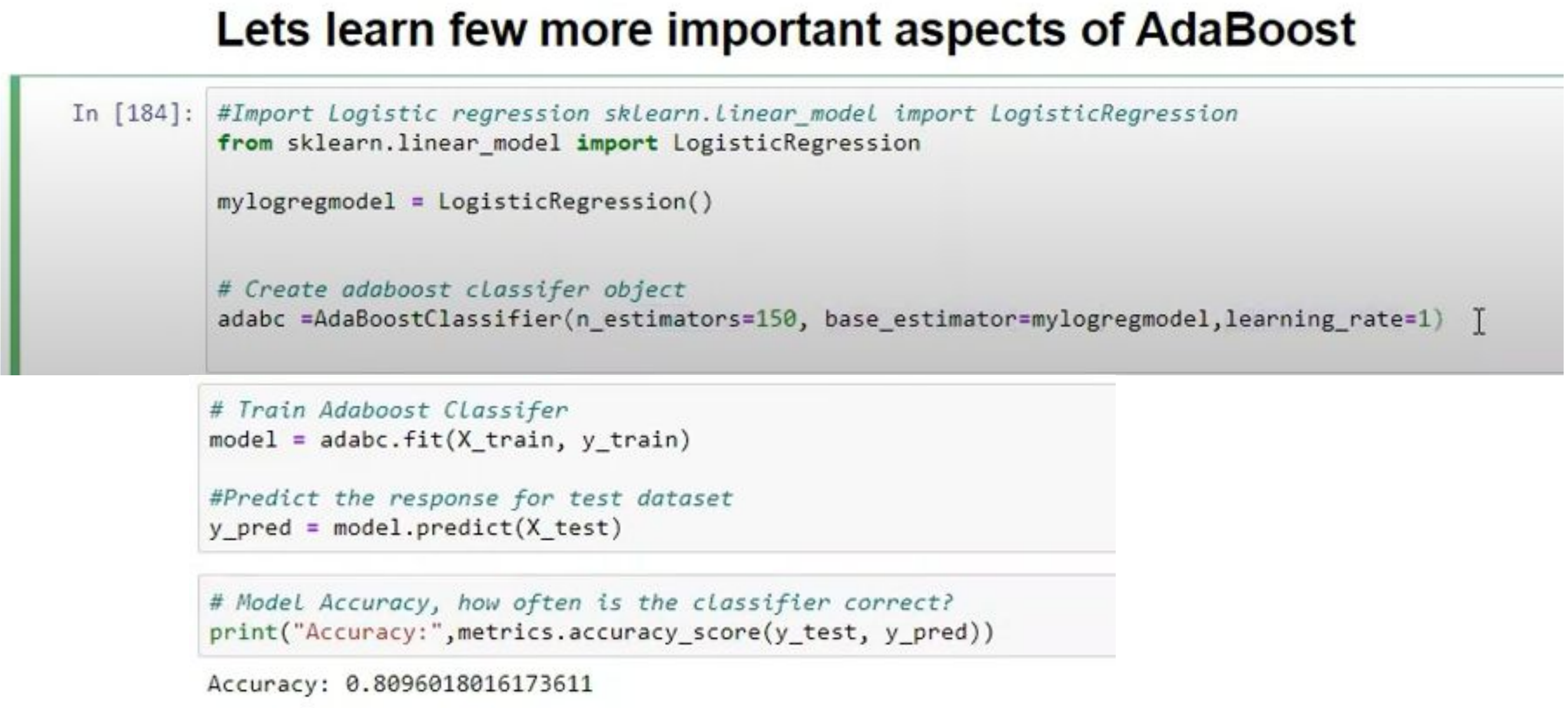
Lab program no 9b: AdaBoost



Steps for implementation

- Import
- Data preparation(csv file)
- Separating column like independent variable and target column
    - Train test split
- Model training and evaluation
    - Create adaboostclassifier(estimater,learning rate)
    - Fit
    - Predict
    - Accuracy

Comparison



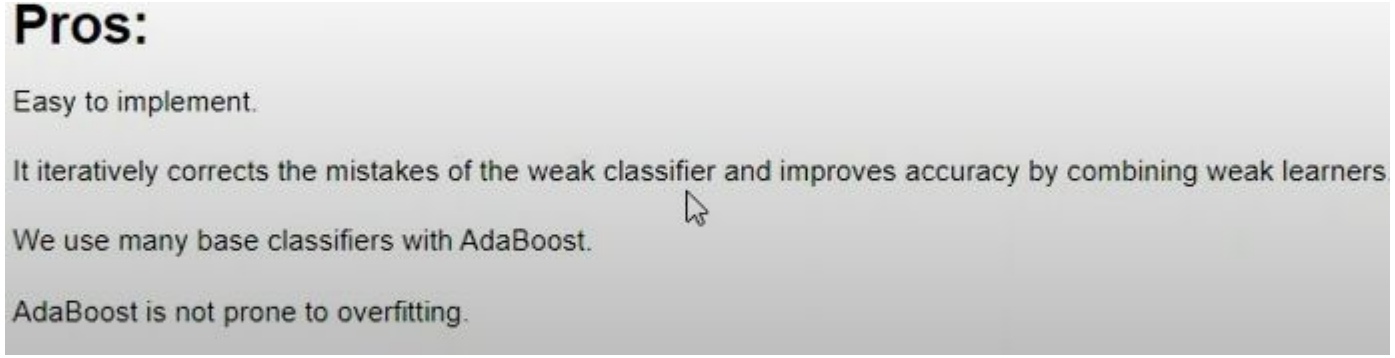
Lets learn few more important aspects of AdaBoost

```
In [184]: #Import Logistic regression sklearn.linear_model import LogisticRegression
          from sklearn.linear_model import LogisticRegression

          mylogregmodel = LogisticRegression()


          # Create adaboost classifer object
          adabc =AdaBoostClassifier(n_estimators=150, base_estimator=mylogregmodel,learning_rate=1)
```

```
# Train Adaboost Classifer
model = adabc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8096018016173611
```



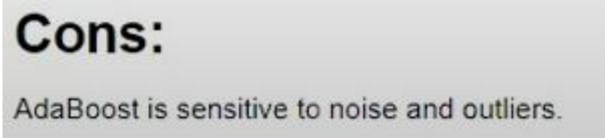
Pros:

Easy to implement.

It iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners.

We use many base classifiers with AdaBoost.

AdaBoost is not prone to overfitting.



Cons:

AdaBoost is sensitive to noise and outliers.