Mapping functionality scripts to documents by multi class classification

Prince kumar

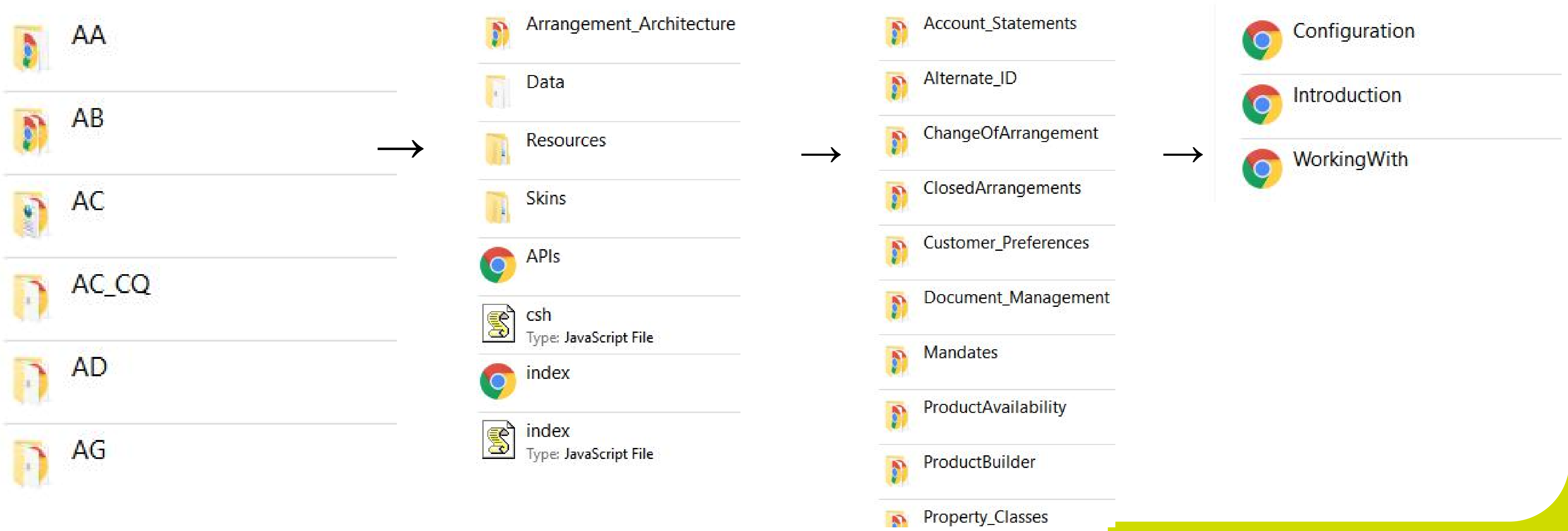
CONTENT

Introduction

1. We have been provided with corpus of functionality scripts and documents.
2. Our approach is to understand documents text and script data and try to find out most relevent features to map them together.
3. Documents text are in html format and functionality scripts are in key value pair (json) format.
4. Preprocessing of raw text and then convert the text into vectors .
5. Multi label classification using machine learning algorithms.

Extraction of text from document step by step

- As we can see ,this process of extraction of text is multi level .
- We chose the folders name as labels for the classsification of our scripts because html text are present in its subfolders .



AA
AB
AC
AC_CQ
AD
AG

→

Arrangement_Architecture
Data
Resources
Skins
APIs
csh
Type: JavaScript File
index
index
Type: JavaScript File

→

Account_Statements
Alternate_ID
ChangeOfArrangement
ClosedArrangements
Customer_Preferences
Document_Management
Mandates
ProductAvailability
ProductBuilder
Property_Classes

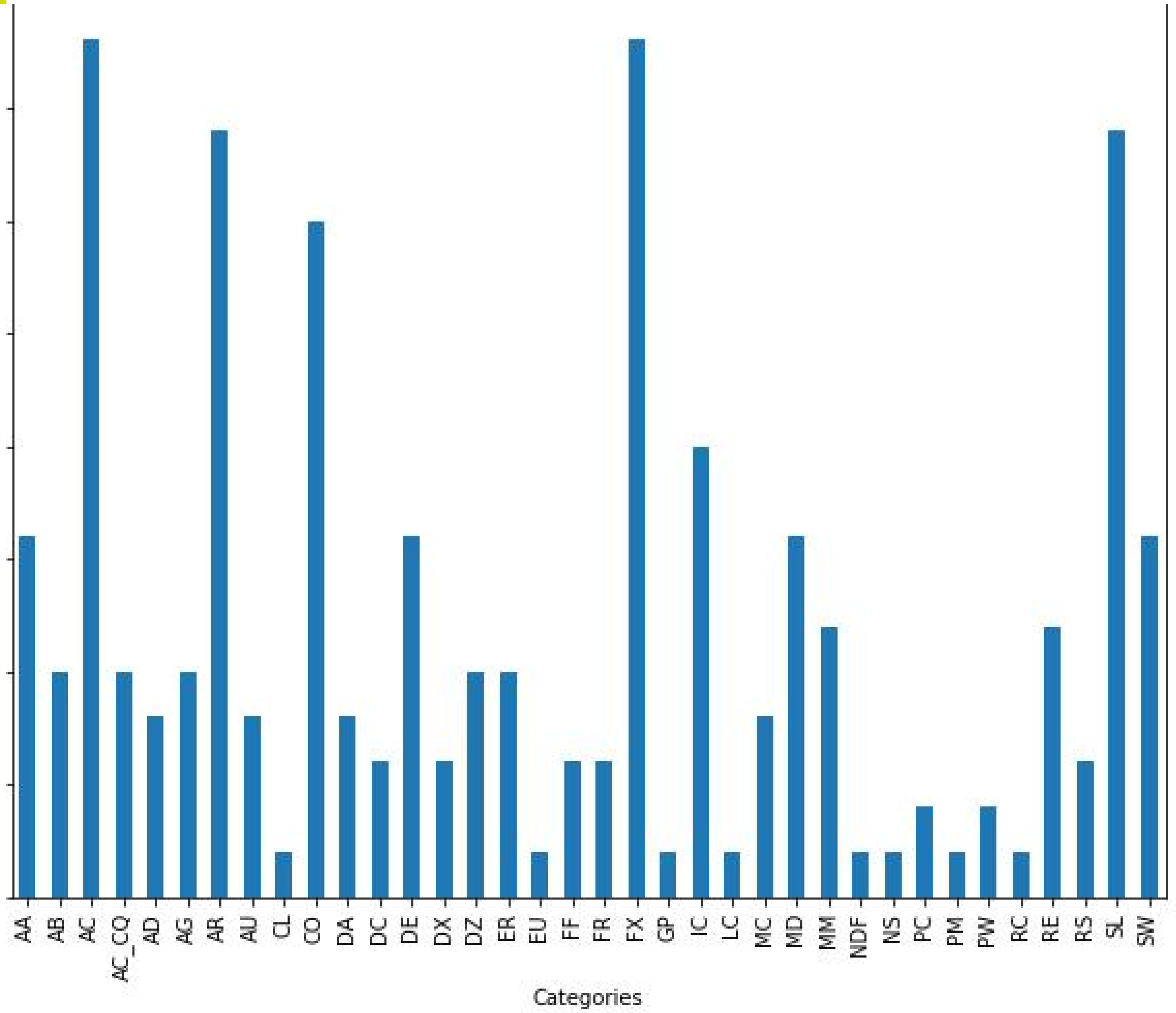
→

Configuration
Introduction
WorkingWith

Mapping Implementation

- Our training data obtained from html text from the documents.
- Our testing data obtained from the functionality scripts.
- Preprocessing is one of the main steps,from the documents,the text being converted from html to plain text using ”Beautifulsoup” and the functionality scripts are in key value pair(json format).
- Prediction of labels for test data obtained from the functionlity script.

Distribution of dataframe

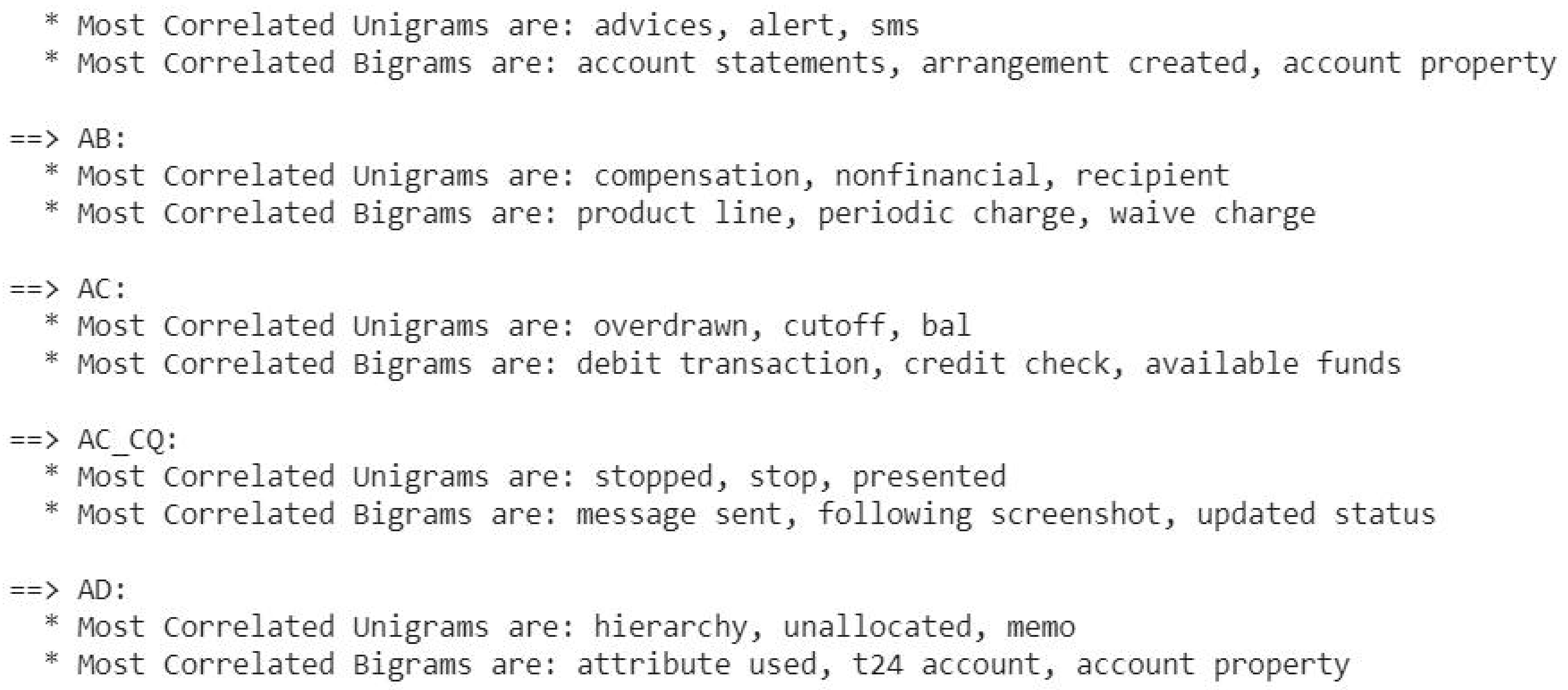


"Categories" column consist the labels of document's text

Catching context using n-gram

- We tried to analyse 2-grams(or bigram) in our dataframe to find out most correlated words.
- You can think of an N-gram as the sequence of N words, by that notion, a 2-gram (or bigram) is a two-word sequence of words .

unigrams and bigrams in our datframe in each labels



```
  * Most Correlated Unigrams are: advices, alert, sms
  * Most Correlated Bigrams are: account statements, arrangement created, account property

==> AB:
  * Most Correlated Unigrams are: compensation, nonfinancial, recipient
  * Most Correlated Bigrams are: product line, periodic charge, waive charge

==> AC:
  * Most Correlated Unigrams are: overdrawn, cutoff, bal
  * Most Correlated Bigrams are: debit transaction, credit check, available funds

==> AC_CQ:
  * Most Correlated Unigrams are: stopped, stop, presented
  * Most Correlated Bigrams are: message sent, following screenshot, updated status

==> AD:
  * Most Correlated Unigrams are: hierarchy, unallocated, memo
  * Most Correlated Bigrams are: attribute used, t24 account, account property
```

TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)

Terminology

- t — term (word)
- d — document (set of words)
- N — count of corpus
- corpus — the total document set

- Text-processing:- we have to convert the text to vectorize it after lowering and removing stop words . TF_IDF will give us the most important words in our text

- The tf-idf model is the model of tf and idf. TF:- term frequency ,it tells us the frequency of words in a given document IDF:- Inverse-Document-Frequency,it tells us ,how rare or commom a word in the document

➢ text ➢ Preprocessing ➢ Tf-idf ➢ Vectorize

CountVectorizer

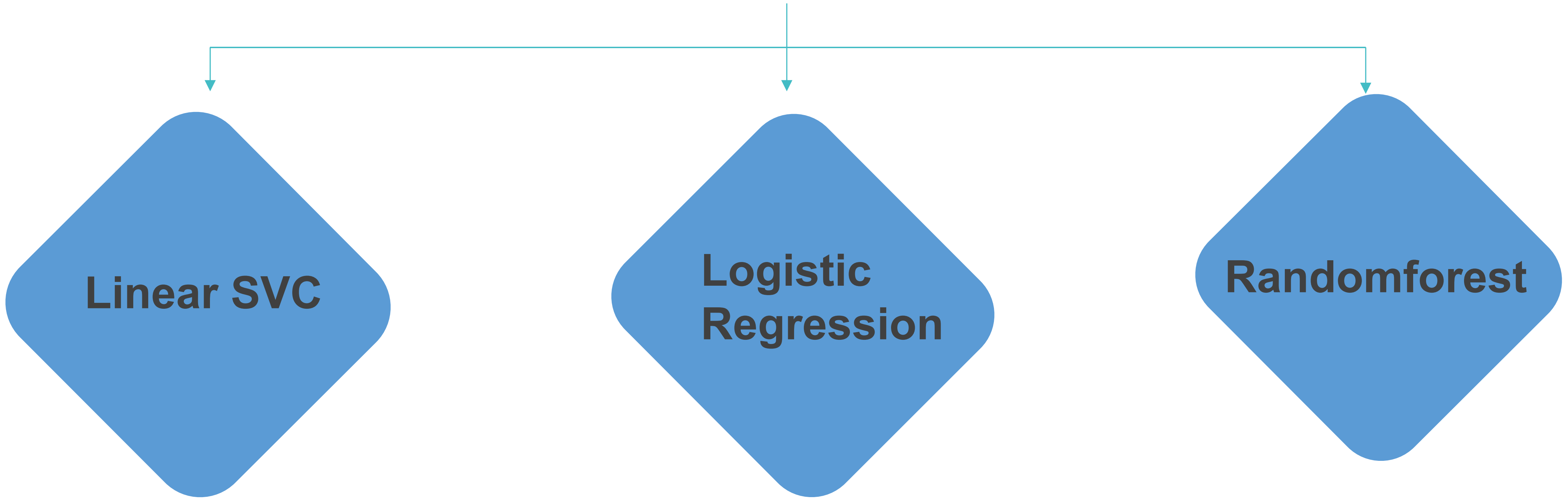
- It provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

  You can use it as follows:
- Create an instance of the CountVectorizer class.
- Call the fit() function in order to learn a vocabulary from one or more documents.
- Call the transform() function on one or more documents as needed to encode each as a vector.
- An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

One vs Rest

- This strategy, also known as one-vs-all, is implemented in OneVsRestClassifier.
- The strategy consists in fitting one classifier per class.
- For each classifier, the class is fitted against all the other classes.
- Advantage of this approach is its interpretability. Since each class is represented by one and only one classifier, it is possible to gain knowledge about the class by inspecting its corresponding classifier.
- This is the most commonly used strategy and is a fair default choice. So we shall go by this method

Classifiers



Linear SVC

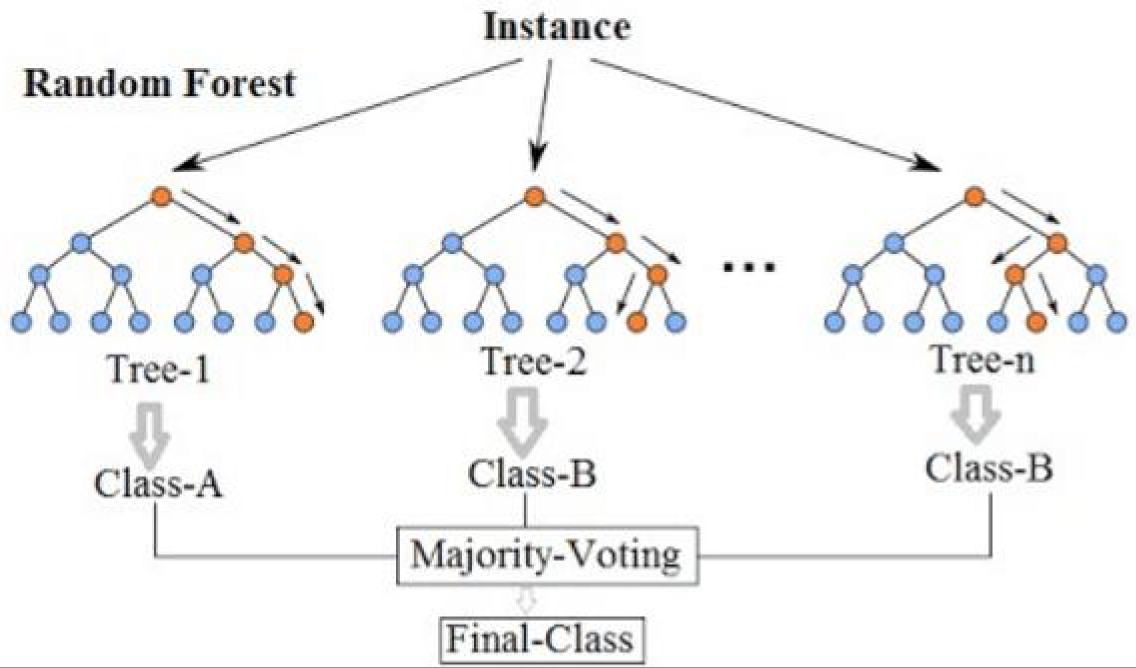
- The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data.
- From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.
- This makes this specific algorithm rather suitable for our use.

Logistic Classifier

A contradiction appears when we declare a classifier whose name contains the term ‘Regression’ is being used for classification, but this is why Logistic Regression is magical: using a linear regression equation to produce discrete binary outputs . And yes, it is also categorized in ‘Discriminative Models’ subgroups of ML methods like Support Vector Machines and Perceptron where all use linear equations as a building block and attempts to maximize the quality of output on a training set.

Random Forest Classifier

- It is an ensemble tree-based learning algorithm.
- The Random Forest Classifier is a set of decision trees from randomly selected subset of training set.
- It aggregates the votes from different decision trees to decide the final class of the test object.



Model prediction on test script

```
 X_train, X_test, y_train, y_test,indices_train,indices_test = train_test_split(features,
labels,df.index, test_size=0.25,  random_state=1)

model = OneVsRestClassifier(LinearSVC())
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('accuracy %s' % accuracy_score(y_test, y_pred))
```

```
accuracy:  0.7358490566037735
label   : ['CO']
```

Conclusion

- Using various algorithms like linear SVC, randomforest classifier.
- we trained our model on documents text.
- Testing data is the text from Script
- Our classifier predict the labels .
- Thus using this multi class classification model we can map our scripts to documents.

THANK YOU