**Big Data Analytics - Assignment**

Twitter profiles have been chosen as the baseline of our data collection. When a user registers himself/herself to the Twitter platform, has to fill a profile form, consisting of about 30 fields containing biographical and other information, such as personal interests and hobbies.

However, many of those fields are optional, and therefore a substantial set of Twitter users leave blank many (or all) of those optional fields. Moreover, Twitter’s profile form does not include a specific “gender” field, which complicates gender identification for Twitter users. Additionally, in Twitter there are profiles not related to a single user: for example companies profiles, fan pages, press profiles and so on.

In these cases, it is useless to assign a specific “gender” to these profiles, since people that write on behalf of such profiles could be of both sex (there isn’t just one person writing).

1. **Data Processing Steps:**
2. **Analysis of the dataset for finding out which columns are required for solving this problem.**

For our analysis I have identified in the Twitter user profile 3 groups of fields that can help in devising an algorithm to classify a generic profile into one of the following categories:

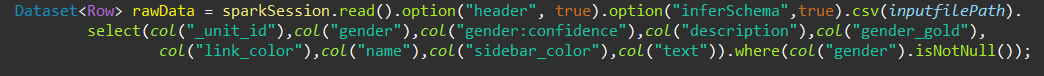
● **Male**

● **Female**

● **Brand**

The 3 groups of fields are :

1. Name , Description
2. Text (informal content)
3. User Profile colors

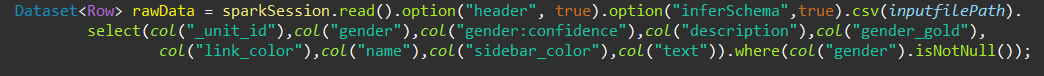


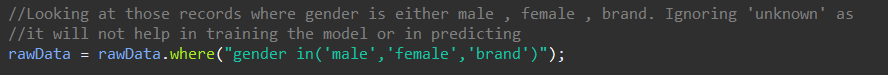
**Brand** refers to all the profiles which are not related to a single user(fan pages, official pages, advertising pages and so on).

1. **In the gender column, there are more than 3 categories – Male , Female , Brand , Unknown and nulls.**

The **unknown** category is related to the profiles that cannot be recognized from the algorithm because of missing parameters (missing description and name not recognizable) or not known from the algorithm

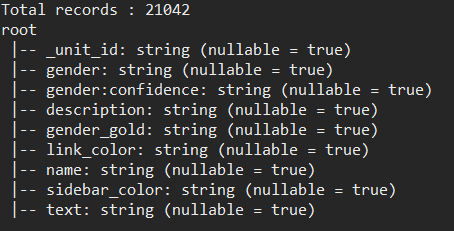
Since, we need to train our model only on – Male , Female , Brand, this is how we get rid of Nulls and Unknowns in gender column :





1. **Analysis of schema to understand what data types different columns have.**

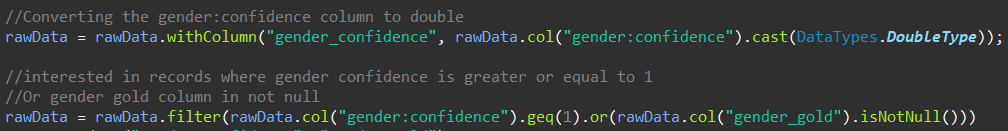




1. **Gaining confidence on different gender values present in “gender” column**

First of all , gender:confidence is converted to Double type.

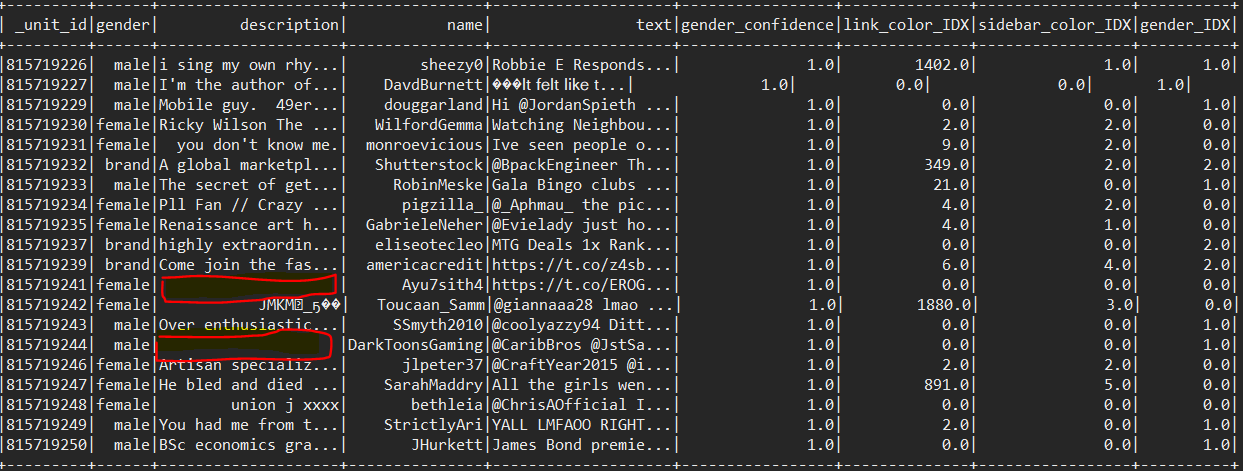
And then : take those records where gender confidence is greater than or equal to 1 OR gender\_gold is given.



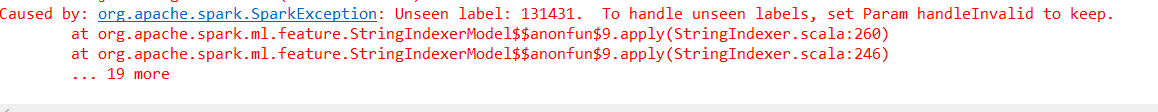
1. **Handling missing values:**

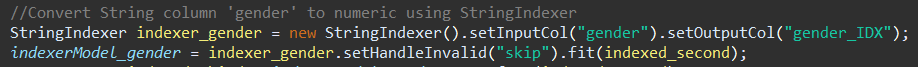
* **Handling nulls** -> Could not use **na().drop()**  as that would return extremely small number of rows in dataset. All the rows had one or more null values and hence this would not work.
* **Replacing nulls with other values** -> Used **.na().fill("")** to replace null values in String columns with blank value like – “”

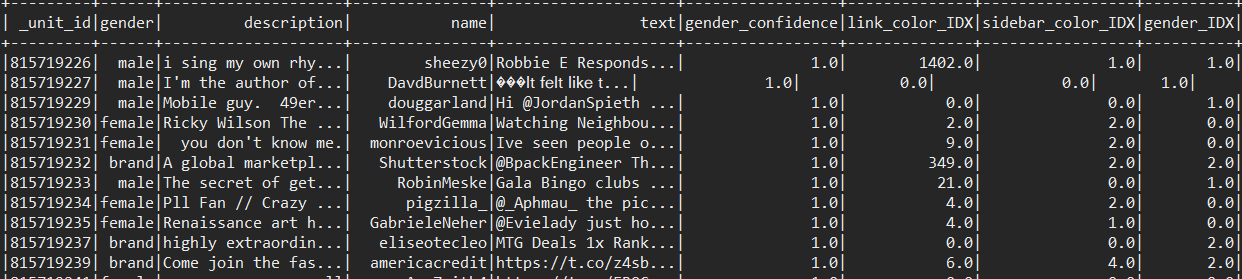




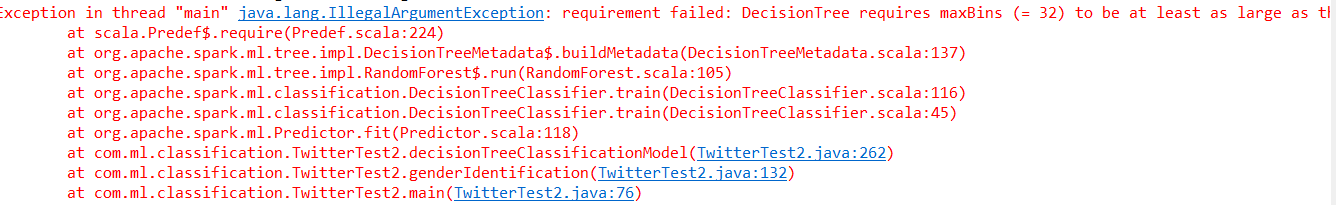
* **Handling unseen labels** -> Used **.setHandleInvalid("skip")**

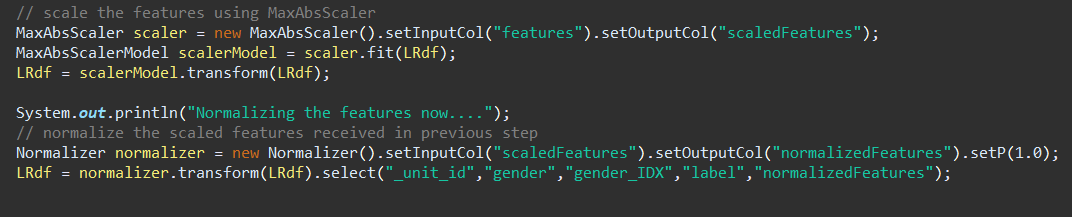


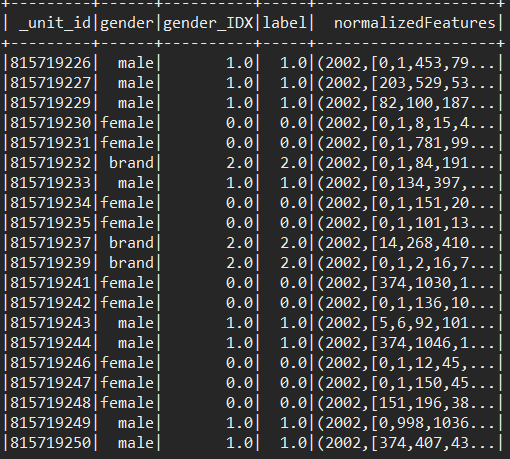




* **Handling categorical values with large number of unique values** -> Used **Scaling and Normalization** to resolve the issue

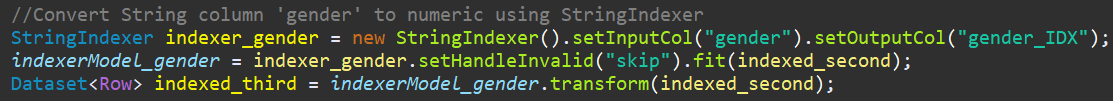




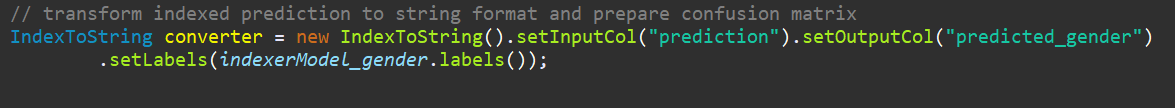


1. **Extracting, transforming String and Categorical features:**

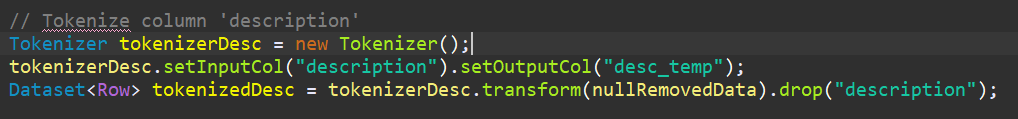
**String Indexer**



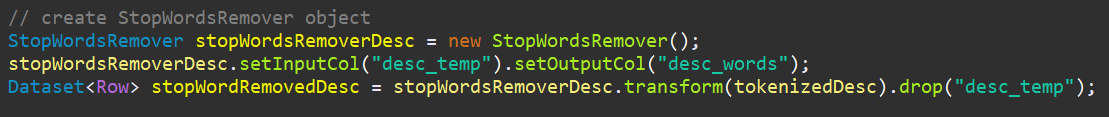
**Index to String**



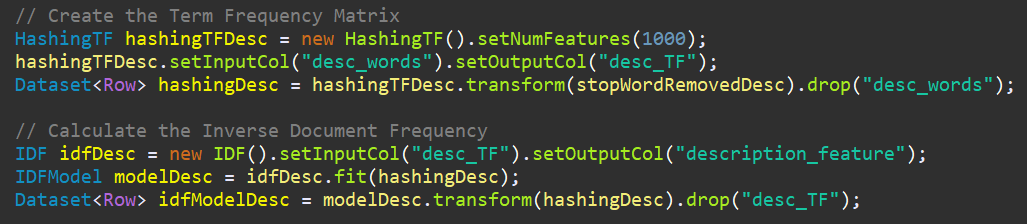
**Tokenizer**



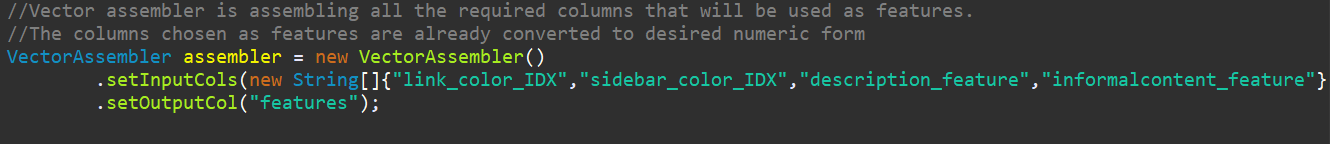
**Stop Words Remover**



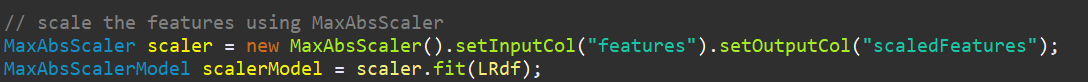
**TF-IDF**



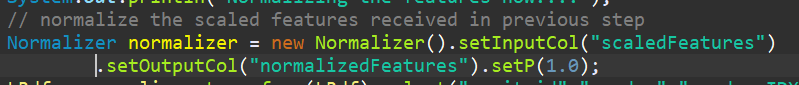
**Vector Assembler**



**MaxAbsScaler**



**Normalizer**



1. **Model Building:**

The problem that we are solving here is of type – **Classification**.

The two models used by me for this problem are:

1. ***Decision Tree Algorithm***
2. ***Random Forest Algorithm***

Identified below set of columns in following groups:

1. ***Description***
2. ***Link\_color , Sidebar\_color***
3. ***Text***

When each of these groups was used individually, the **accuracy** of the algorithm seemed like:

|  |  |  |
| --- | --- | --- |
|  | Decision Tree | Random Forest |
| **Description** | 47% | 49% |
| **LinkColor, SideBar color** | 39% | 40% |
| **Text** | 48% | 45% |

When grouped together, **accuracy** of the algorithm seemed like:

|  |  |  |
| --- | --- | --- |
|  | Decision Tree | Random Forest |
| **LinkColor, SideBar color, Decription** | 45% | 47% |
| **Text , Description** | 52% | 52% |

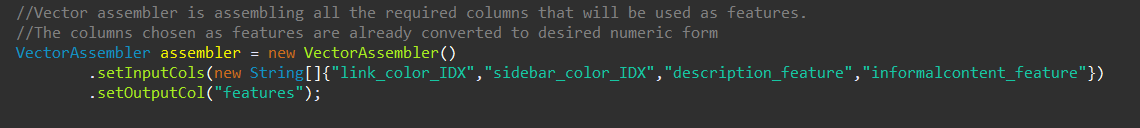
When grouped further together, the **accuracy** improved like:

|  |  |  |
| --- | --- | --- |
|  | Decision Tree | Random Forest |
| **LinkColor, SideBar color, Decription, Text** | 54% | 55% |

Hence, the final choice of columns for training both algorithms is :

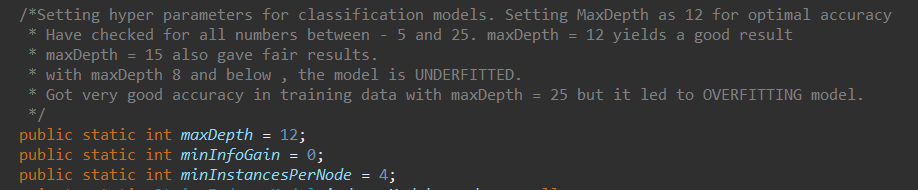
1. ***Description***
2. ***Link\_color , Sidebar\_color***
3. ***Text***

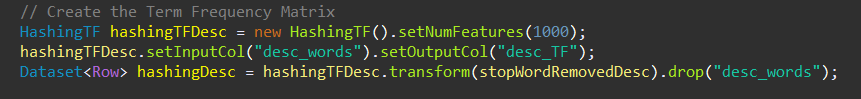
**All of these combined together as Feature Vector:**



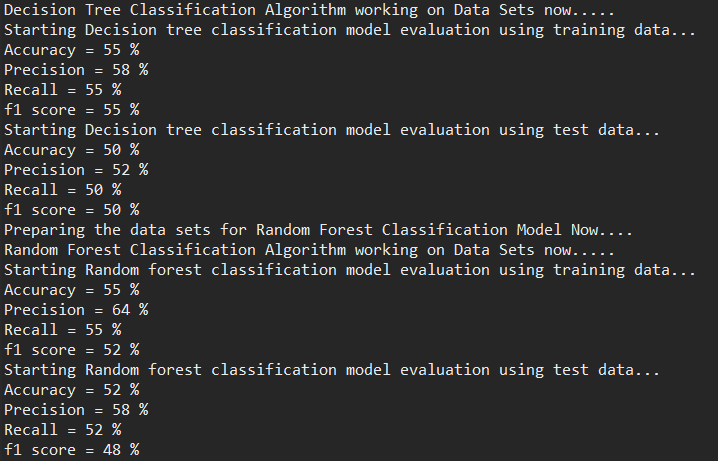
Hyperparameters used while building the model:

1. ***maxDepth***
2. ***minInfoGain***
3. ***minInstancesPerNode***
4. ***setNumFeatures***





Result Obtained:



***maxDepth*** value was checked for numbers between 5 and 25.

After 12, I found that model started OVERFITTING

For <=10 , both Models were Underfitted

For > 12, both Models started Overfitting

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| maxDepth | Decision Tree Training(%) | Decision Tree Test(%) | Random Forest Training (%) | Random Forest Test(%) | Results |
| 7 | 52 | 49 | 49 | 46 | Accuracy can still be improved |
| 10 | 53 | 50 | 53 | 49 | Looks Optimal. Can be improved |
| 12 | 55 | 50 | 55 | 52 | Giving good result |
| 15 | 56 | 50 | 55 | 49 | Scope for fitting. Model getting Overfitted Slowly |
| 20 | 58 | 51 | 60 | 51 | Model quite overfitted |
| 25 | 60 | 50 | 63 | 52 | Getting too Overfitted |

***minInstancesPerNode*** value was checked between 1 and 5.

For a node to be split further, each of its children must receive at least this number of training instances. This is commonly used with RandomForest since those are often trained deeper than individual trees.

Better results were yielded with minInstancesPerNode = 4

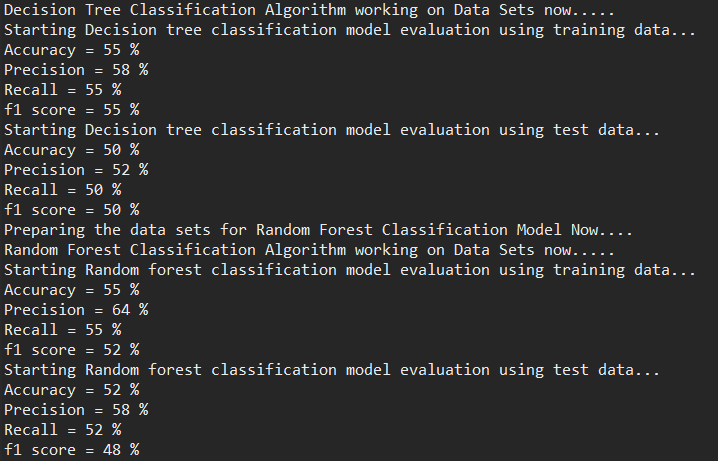
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| minInstancesPerNode | Decision Tree Training(%) | Decision Tree Test(%) | Random Forest Training (%) | Random Forest Test(%) | Results |
| 1 | 55 | 50 | 58 | 51 | Accuracy is great but Radom Forest is Overfitted |
| 2 | 56 | 49 | 57 | 51 | Accuracy is still good and it is becoming better fitted |
| 3 | 55 | 51 | 55 | 50 | Giving good result |
| 4 | 55 | 50 | 55 | 52 | Seems optimal here. Both models perform good here. |
| 5 | 55 | 49 | 55 | 51 | Both Models seem doing okay here |

***minInfoGain*** value was tried with 0 and 1. **Finally kept it constant at 0** because with minInfoGain = 1 , program did not give good performance or accuracy.

***setNumFeatures*** was also explored for better feature extraction. **Lower values (<1000)** , did not give good results and **higher values (>1000)** led to increased computation time. Hence, sticking to 1000 here.

1. **Evaluation Metrics:**

The final results on console looked like this:



**Decision Tree Model Evaluation: Random Forest Model Evaluation:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Training Data** | **Test Data** |  |  | **Training Data** | **Test Data** |
| Accuracy | 55% | 50% |  | Accuracy | 55% | 52% |
| Precision | 58% | 52% |  | Precision | 64% | 58% |
| Recall | 55% | 50% |  | Recall | 55% | 52% |
| F1Score | 55% | 50% |  | F1Score | 52% | 48% |

**Please Note: In my program, Confusion matrix is being written in .csv format to Output directory.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Decision Tree Model Random Forest Model Confusion Matrix : Confusion Matrix :** | | | | | | |
| **gender** | **predicted\_gender** | **count** |  | **gender** | **predicted\_gender** | **count** |
| brand | brand | 450 |  | brand | brand | 345 |
| brand | female | 210 |  | brand | female | 535 |
| brand | male | 375 |  | brand | male | 153 |
| female | brand | 108 |  | female | brand | 21 |
| female | female | 691 |  | female | female | 1428 |
| female | male | 732 |  | female | male | 142 |
| male | brand | 131 |  | male | brand | 43 |
| male | female | 407 |  | male | female | 986 |
| male | male | 784 |  | male | male | 270 |

In both the Models, the accuracy difference is only **3% - 5%** between Training Data and Test Data, which is a fairly fitted model.

Checked it on basis of evaluation metrics:



* For **Decision Tree**, difference in accuracy in Training and Test Dataset is : **5%**
* For **Random Forest**, difference in accuracy in Training and Test Dataset is : **3%**

**Please Note:** The cases for **UNDERFITTING** and **OVERFITTING** have been avoided by changing the hyperparameters accordingly, as described in section – 2. Model Building

1. **Inferences & Suggestions:**
2. **Advantages and Disadvantages**

**About Decision Tree Algorithm:**

|  |  |  |
| --- | --- | --- |
|  | **Advantages** | **Disadvantages** |
| 1 | Decision Trees are easy to interpret. | Tree might get too large even after some pruning leading to instability. It can be difficult to control the size of the tree. |
| 2 | Decision Trees are not complex. Data classification without much calculations. Can Implicitly perform feature selection. | The high classification error rate while training set is small in comparison with the number of classes |
| 3 | Decision trees are applicable for both continuous and categorical inputs | In some complex cases, splitting data into classes might not be helpful. |
| 4 | Decision trees are good at dealing with noisy or incomplete data |  |
| 5 | Universal for solving both classification and regression problems |  |

**About Random Forest Algorithm**:

* All advantages and Disadvantages of Decision Tree algorithm are applicable to Random Forest algorithm too.
* The idea in Random Forest Classification is to combine a group of average performing classifiers to form a good classifier. In the algorithm, you construct multiple decision trees by considering random subsets of data each time and finally take a cumulative measure when you predict the results.
* **Added Advantages**:
* Powerful and accurate
* It's fairly good for training even small samples and can be easily parallelized in R/Python/other software
* Random forests in general perform well than a single decision tree as they manage to reduce both bias and variance
* **Disadvantage**
  + it fails when there are rare outcomes or rare predictors.

**Comparison:** I would choose Random Forest Algorithm based on the output received from my program solution. Reasons:

* Faster
* Less Prone to Overfitting
* More Accurate

1. **Improvisation Techniques**

* **Pruning**: It is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances.

Pruning reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting.

* **Variable preselection:** Different tests can be done like multicollinearity test, VIF calculation, IV calculation on variables to

select only a few top variables. This will lead in improved performance as it would strictly cut out the undesired variables.

* **K-Fold cross validation**: Cross validation in the training data itself can improve the performance of the model a bit.
* **Hybrid Model:** Use a hybrid model, i.e. use logistic regression after using decision trees to improve performance.

1. **Choosing one among the two models:**

Between the two model used - Decision Tree Algorithm and Random Forest algorithm, I would go with **Random Forest Classifier .**

* With a relatively small dataset used for this assignment, I could get better accuracy and less overfitting %age with Random Forest Classifier as illustrated in section - 3. Evaluation Metrics.
* It is an Ensemble Learning technique which uses multiple trees (random forests) to predict the outcomes. Random forests in general ***perform well*** than a single decision tree as they manage to ***reduce both bias and variance***. They are ***less prone to overfitting*** as well.