REPORT FOR
DECISION TREE AND NAIVE BAYES

Data mining is used to extract useful data from large datasets, and it is very easy to display in visualizations.

Decision Tree is the popular tool for classification and prediction. It is commonly used classification systems based on multiple covariates or for developing prediction algorithms for a target variable. Most common usages of decision tree models are Variable selection, Assessing the relative importance of variables, Handling of missing values, Prediction, Data manipulation. The main components of a decision tree model are nodes and branches and the most important steps in building a model are splitting, stopping, and pruning. Decision tree may neglect some key values in training data, which may lead to the loss in accuracy this can be solved to some extent in naïve bayes classification. In this we need to transverse through every node to decide which is very time consuming, so we will do processing of data to remove unwanted data.

Naïve Bayes classifier is a probabilistic model, and it is based on Bayes theorem. The core assumption of naïve bayes is that all features are independent of each other. Most applications for naive Bayes algorithms include sentiment analysis, spam filtering, recommendation systems, etc. Although they are quick and simple to use, their major drawback is the need for independent predictors. The classifier performs worse when the predictors are dependent, which occurs in most real-world situations.

1. **Understanding dataset:**

   The given dataset contains 20050 entries with 26 columns. The target variable is gender. Unique values of gender are male, female, brand, other and unknown. So in this we took rows which are either male or female and also the data with gender:confidence>=0.8 and profile:confidence >=0.8. The dataset for the Gender_classifier are shown in the below picture.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20050 entries, 0 to 20049
Data columns (total 26 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   _unit_id               20050 non-null  int64
 1   _golden                20050 non-null  bool
 2   _unit_state            20050 non-null  object
 3   _trusted_judgments     20050 non-null  int64
 4   _last_judgment_at      20000 non-null  object
 5   gender                 20050 non-null  object
 6   gender:confidence      20024 non-null  float64
 7   profile_yn             20050 non-null  object
 8   profile_yn:confidence  20050 non-null  float64
 9   created                20050 non-null  object
 10  description            16306 non-null  object
 11  fav_number             20050 non-null  int64
 12  gender_gold            50 non-null     object
 13  link_color             20050 non-null  object
 14  name                   20050 non-null  object
 15  profile_yn_gold        50 non-null     object
 16  profileimage           20050 non-null  object
 17  retweet_count          20050 non-null  int64
 18  sidebar_color          20050 non-null  object
 19  text                   20050 non-null  object
 20  tweet_coord            159 non-null    object
 21  tweet_count            20050 non-null  int64
 22  tweet_created          20050 non-null  object
 23  tweet_id               20050 non-null  float64
 24  tweet_location         12566 non-null  object
 25  user_timezone          12252 non-null  object
dtypes: bool(1), float64(3), int64(5), object(17)
memory usage: 3.8+ MB
```

Data pre processing:

The gender_gold, profile_yn_gold and tweet_cord have around 20000 null values in a dataset of size of 20050 and thus can be removed from the dataset.The two columns gender:confidence and profile_yn:confidence give the confidence levels of values in columns gender and profile_yn respectively. So we have filtered/removed the rows of the dataset where gender:confidence and profile_yn:confidence is less than 0.8 or 80%. And then removed those confidence columns from the dataset.

On visualizing the dataset we can see that the following fields are not correlated to the target variable:

1. Tweet_location
2. User_timezone
3. _unit_id
4. _last_judgment_at
5. Created
6. Name
7. Profileimage
8. Tweet_created
9. _trusted_judgments
10. _golden

11. Link_color
12. Sidebar_color

So, we drop these columns as well.

After that when we print the unique values of all remaining fields, we can see that profile_yn and tweet_id have only one unique value. Thus, there values won't be useful in building the model. Thus, we drop those columns as well.

We have added two additional columns in the dataframe.

1. Desc_len: This field contains the length of description field.
2. Text_len: This field contains the length of text field.

As string fields cannot be directly used to create the model we have converted the categorical values of _unit_state ('finalized': 1, 'golden': 0) and gender ('male': 1, 'female': 2, 'brand': 3, 'unknown': 4) to numerical data.

For fields such as description we have used label encoder provided by scikit for pre-processing which transforms and assigns labels to string values.

After that we select the values that are required i.e.

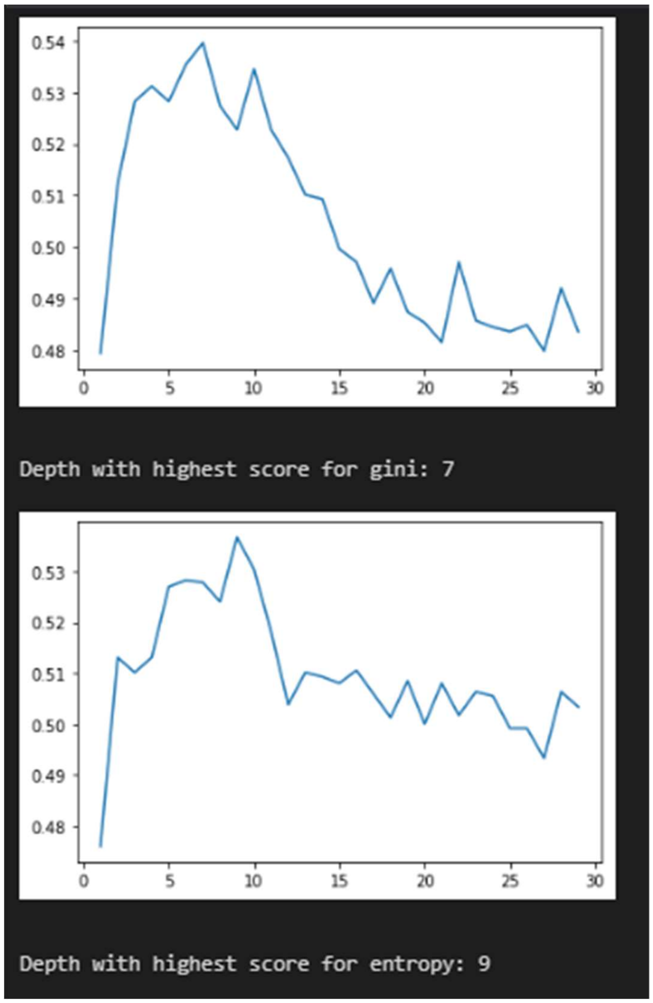
#	Column	Non-Null Count	Dtype
---	------	--------------	-----
0	description	13939 non-null	int32
1	desc_len	11858 non-null	float64
2	tweet_count	13939 non-null	int64
3	text_len	13939 non-null	int64
4	fav_number	13939 non-null	int64
5	gender	13939 non-null	int64

We also drop the rows if there are any NA or missing values in the dataset.

We then split the dataset into train (60%), test (20%) and validation (20%) sets using the train_test_split function provided in scikit-learn.

Building the decision tree:

For building the decision tree, we first found the depth of decision tree where the score is maximum using gini and entropy as criterion. For decision tree using gini we found that the score is maximum at depth of 7 and that for entropy is 9.



Then using these depth we have built 2 decision tree i.e. decision tree with max depth = 7 and criterion as gini and another with depth = 9 and criterion as entropy.

Printing the confusion matrix and classification report and Interpreting the results using DT:

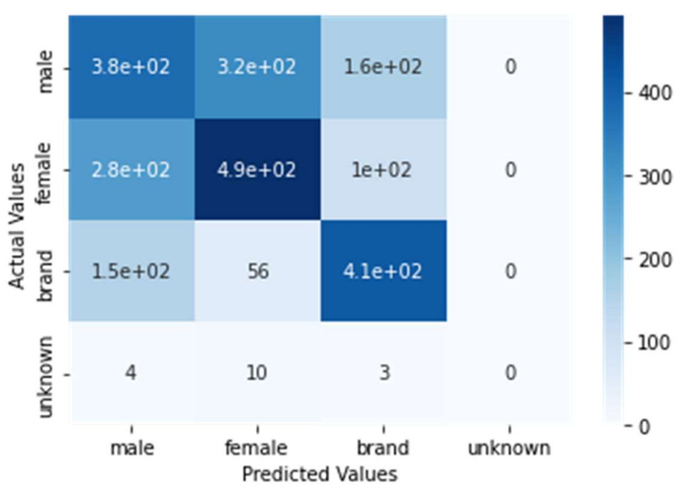
For printing the classification report we have used the ‘classification_report’ function present in scikit learn metrics. And for calculating the confusion matrix we used ‘confusion_matrix’ function. Below is the snapshot of confusion matrix and classification report. We have also verified the values of classification report by doing manual calculation of classification report using confusion matrix.

```
Printing the classification report:
              precision    recall  f1-score   support

           1       0.46      0.44      0.45       857
           2       0.56      0.56      0.56       880
           3       0.60      0.67      0.63       618
           4       0.00      0.00      0.00        17

    accuracy                           0.54      2372
   macro avg       0.41      0.42      0.41      2372
weighted avg       0.53      0.54      0.54      2372
```

Confusion matrix of tree with criterion = gini and depth = 7



Calculation:

1. Precision:

male = 376/(376+284+151+4) = 0.46

female = 491/(318+491+56+10) = 0.56

brand = 411/(163+105+411+3) = 0.60

unknown = 0/(0+0+0+0) = 0

2. Recall:

male = 376/(376+318+163+0) = 0.43

female = 491/(284+491+105+0) = 0.56

brand = 411/(151+56+411+0) = 0.67

unknown = 0/(4+10+3+0) = 0

3. F1 score:

male = (2*Precision*Recall)/(Precision+Recall) = (2 * 0.46 * 0.43)/(0.46 + 0.43) = 0.45

female = (2*Precision*Recall)/(Precision+Recall) = (2 * 0.56 * 0.56)/(0.56 + 0.56) = 0.56

brand = (2*Precision*Recall)/(Precision+Recall) = (2 * 0.60 * 0.67)/(0.60 + 0.67) = 0.63

unknown = (2*Precision*Recall)/(Precision+Recall) = (2 * 0 * 0)/(0 + 0) = 0

4. Support:

male = 377+317+163+0 = 857

female = 284+491+105+0 = 880

brand = 151+56+411+0 = 618

unknown = 4+10+3+0 = 17

5. Accuracy = (377 + 491 + 411 + 0)/(2372) = 0.54

We have done the same calculations and printing for the decision tree using entropy as criterion. The calculations are present in the Jupyter notebook as well.

For both the trees using Gini and Entropy, we have obtained accuracy of 0.54 for both and similar values for classification report parameters as well as we have built both the trees with depth that gives maximum score.

Printing the classification report:

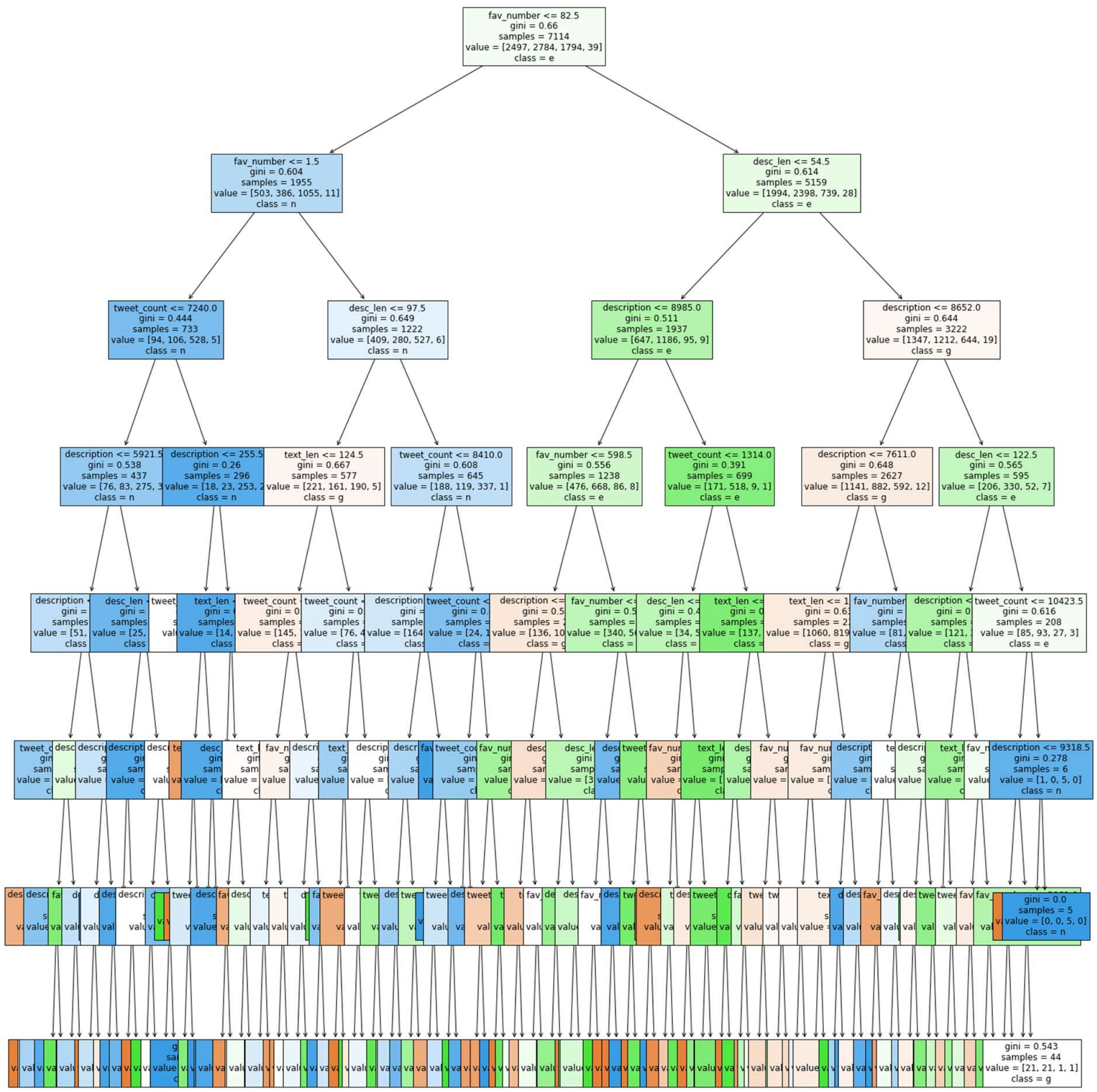
	precision	recall	f1-score	support
1	0.46	0.44	0.45	857
2	0.56	0.56	0.56	880
3	0.60	0.67	0.63	618
4	0.00	0.00	0.00	17
accuracy			0.54	2372
macro avg	0.41	0.42	0.41	2372
weighted avg	0.53	0.54	0.53	2372

Printing the classification report:

	precision	recall	f1-score	support
1	0.45	0.46	0.46	857
2	0.56	0.57	0.57	880
3	0.63	0.60	0.61	618
4	0.00	0.00	0.00	17
accuracy			0.54	2372
macro avg	0.41	0.41	0.41	2372
weighted avg	0.53	0.54	0.53	2372

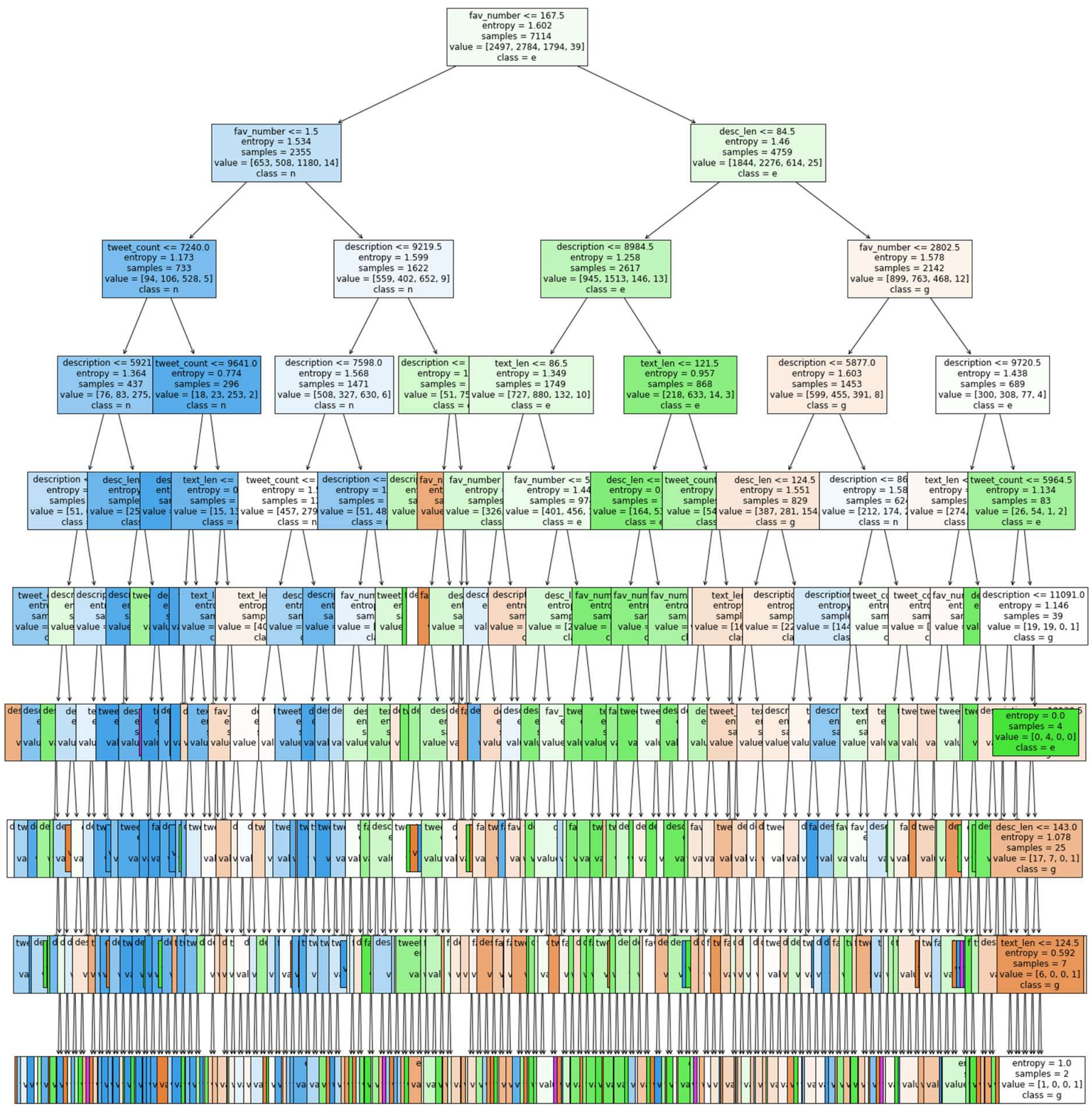
Visualization For Gini:

Then we printed the decision trees built using scikit-tree’s plot_tree function. They are as shown below.



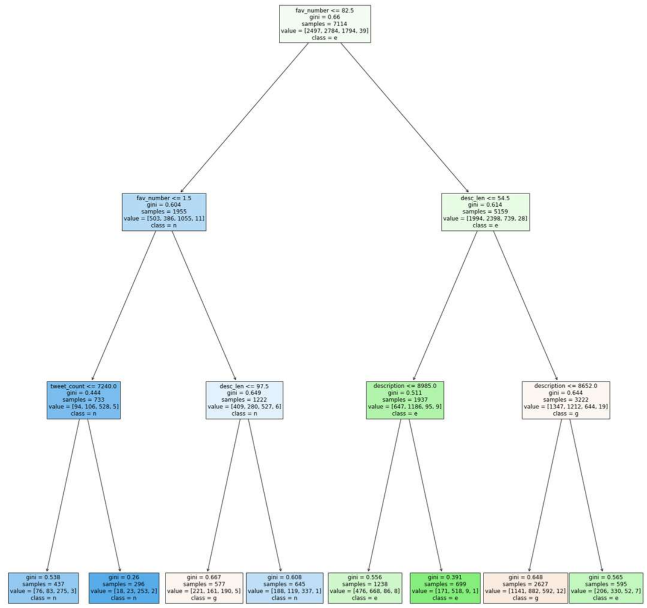
Decision tree with criterion = Gini and depth = 7

Visualization for Entropy:

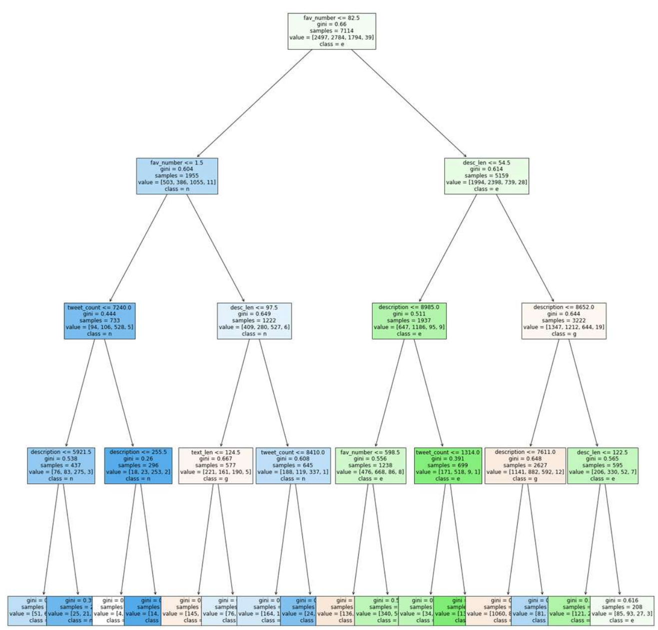


Decision tree with criterion = entropy and depth = 9

We have also built and printed two different trees with different depths and criterion. They are as shown below.

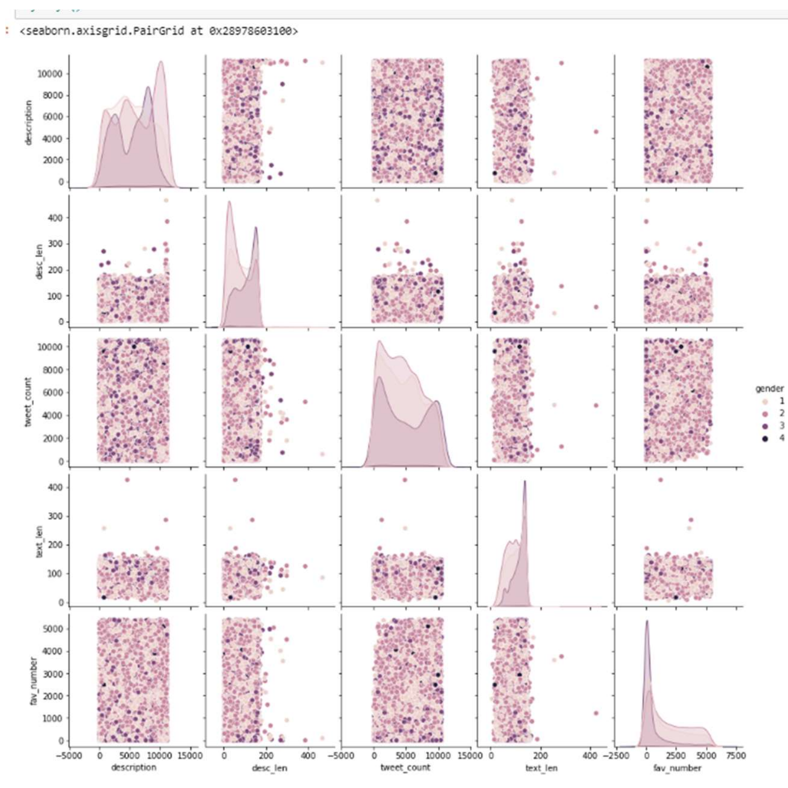


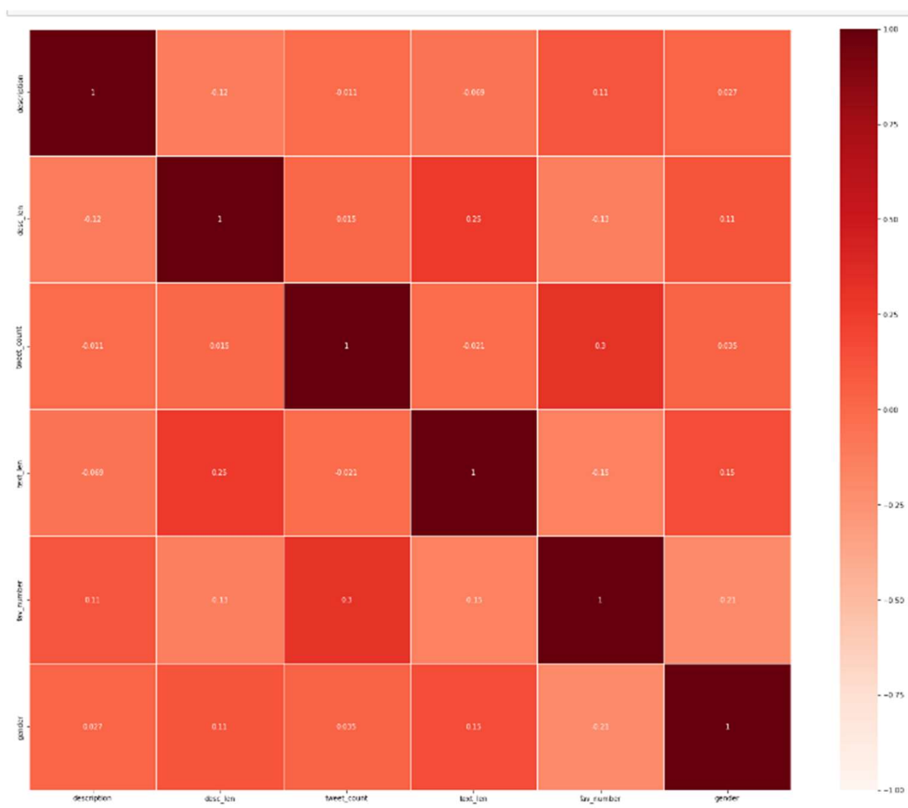
Decision tree with depth = 3 and criterion as entropy



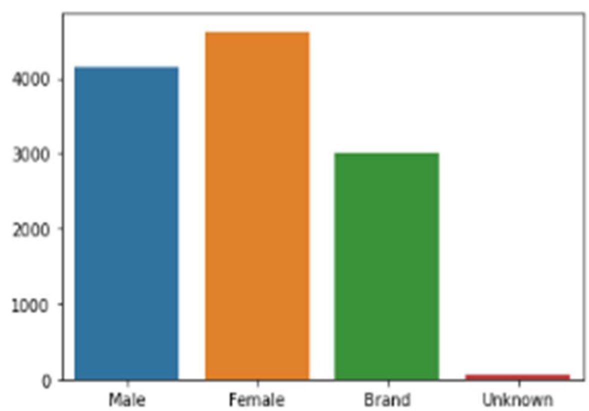
Decision tree with depth = 4 and criterion as gini

Graphs for the data set:





From the two visualizations above we can see that the two fields tweet_count and fav_number are linearly correlated to the target field i.e. gender.



From the above graph we can see the count of twitter account for each gender in the given dataset.

Naive Bayes:

We have implemented 5 different types of Naive Bayes classifier on the given dataset that are available in scikit-learn. Following are the names of those classifiers:

a. Gaussian Naive Bayes
b. Multinomial Naive Bayes
c. Complement Naive Bayes
d. Bernoulli Naive Bayes
e. Categorical Naive Bayes

We did not get any error while building and predicting the values for test set for any Naive Bayes classifier.

The Gaussian Naive Bayes classifier gives the highest accuracy i.e. of 0.5 as compared to all other types of Naive Bayes classifiers.

References:

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/#:~:text=Decision%20tree%20methodology%20is%20a,algorithms%20for%20a%20target%20variable.
- https://mljar.com/blog/visualize-decision-tree/
- https://scikit-learn.org/stable/modules/naive_bayes.html
- https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c#:~:text=A%20Naive%20Bayes%20classifier%20is,based%20on%20the%20Bayes%20theorem.
- https://www.geeksforgeeks.org/naive-bayes-classifiers/