Data Mining (CSE 5334)

Assignment 2 (DT_NB)

Student details:

- Kuldip Rameshbhai Savaliya – 1001832000
- Meghaben Ghanshyambhai Patel – 1002006777
- Shivani Manojkumar Panchiwala – 1001982478

1) Description of the Decision tree methods, and naïve bayes classifier:

Decision Tree: -

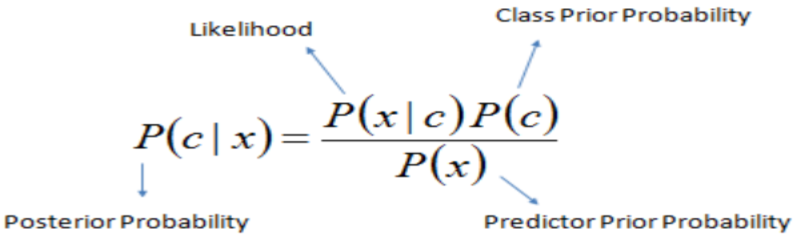
Classification and prediction are most effectively done using a Decision Tree. Decision tree looks like a tree structure. Where internal node represents a test on an attribute, branch represents an output of the test, and leaf node contains class label. Node splitting the process of dividing a node into multiple sub-nodes to create relatively pure nodes**.** Splitting of node in decision tree using Gini index, entropy (Information gain) and misclassification error. Decision tree is easy to construct and extremely fast at classify unknown records.

Decision Tree Methods: -

- **Select input variables:**
  In decision tree selection of input variables based on the Gini-index, information gain(entropy), and chi-square for categorical variables.
- **Pruning:**
  A decision tree consists of a root node, several branch nodes, and several leaf nodes. Pruning means to change the model by deleting the child nodes of a branch node. The pruned node is regarded as a leaf node. Leaf nodes cannot be pruned.
- **Splitting:**
  When creating the model, first determine the most relevant input variables, and then divide records at the root node and subsequent internal nodes into two or more categories or 'bins' based on the status of these variables.

Naive Bayes

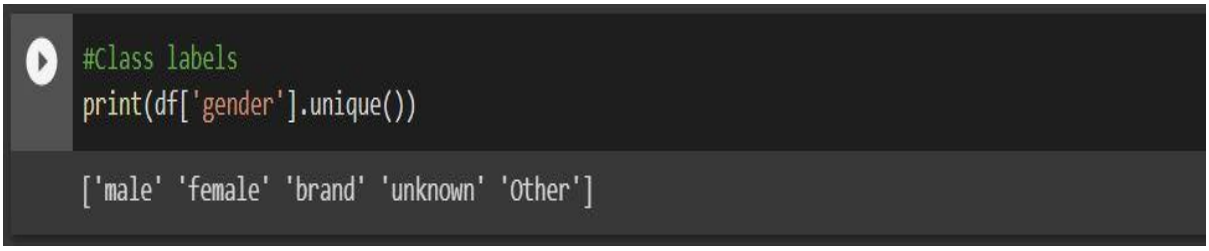
The naive Bayes classifier is a classification technique founded on the independence of predictors and the Bayes theorem. The naive Bayes classifier assumes that the existence of one feature in a class is unrelated to the existence of other characteristics. Naive Bayes employs the Bayes theorem but presupposes that every variable in the model is unrelated to every other variable. As a result, you can multiply everything without having to figure out the more intricate conditional probabilities that probability theory requires.


Likelihood
Class Prior Probability
$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

2) description about given dataset

The given dataset is about the tweets done by people and the output class is “Gender” and gender contains five unique value such as male, female, brand, unknown and other. The purpose of the data is to be used to identify the gender of the person based on their tweets and given the rest of the data.

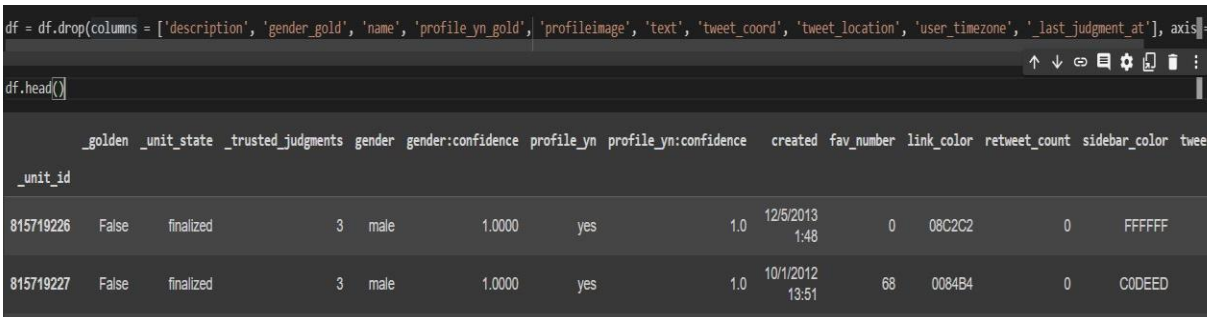


```
#Class labels
print(df['gender'].unique())
```

```
['male' 'female' 'brand' 'unknown' 'Other']
```

Pre-processing:

First check the how many columns contains null value. After analysing, I drop some columns which are not useful for prediction of the gender class.

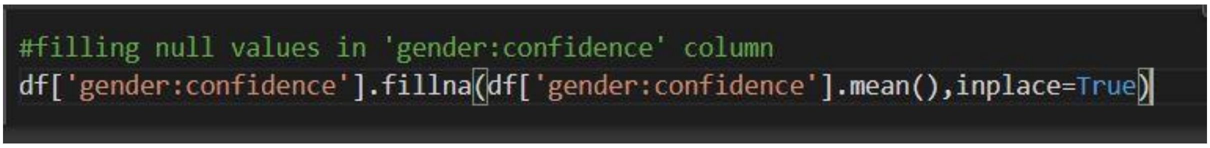


```
df = df.drop(columns = ['description', 'gender_gold', 'name', 'profile_yn_gold', 'profileimage', 'text', 'tweet_coord', 'tweet_location', 'user_timezone', '_last_judgment_at'], axis=
```

```
df.head()
```

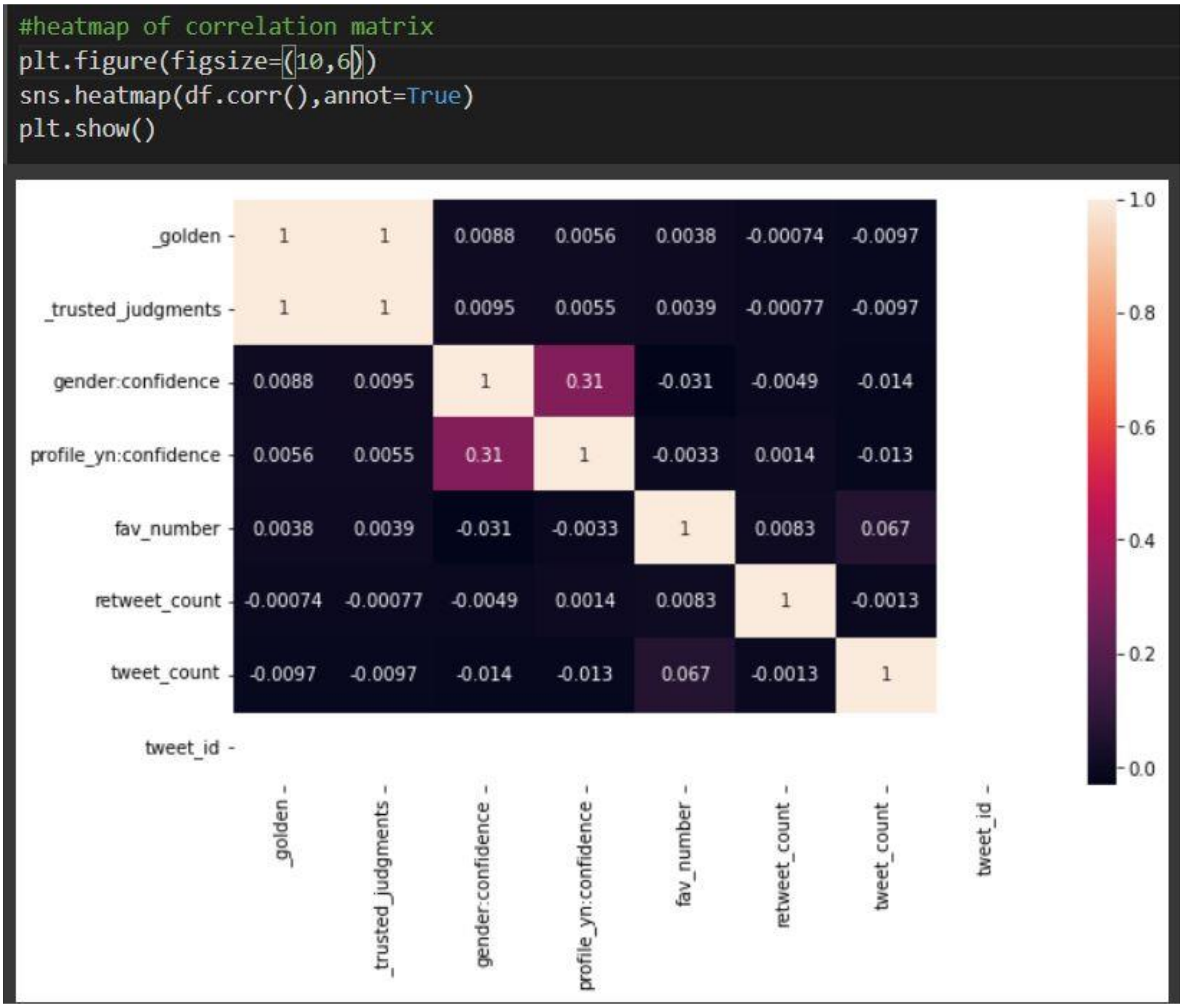
_unit_id	_golden	_unit_state	_trusted_judgments	gender	gender:confidence	profile_yn	profile_yn:confidence	created	fav_number	link_color	retweet_count	sidebar_color	twee
815719226	False	finalized	3	male	1.0000	yes	1.0	12/5/2013 1:48	0	08C2C2	0	FFFFFF	
815719227	False	finalized	3	male	1.0000	yes	1.0	10/1/2012 13:51	68	0084B4	0	C0DEED	

Then filling the null values in gender: confidence column using the mean of the colums.



```
#filling null values in 'gender:confidence' column
df['gender:confidence'].fillna(df['gender:confidence'].mean(),inplace=True)
```

Heatmap of correlation between the features to find the best features.



```
#heatmap of correlation matrix
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True)
plt.show()
```

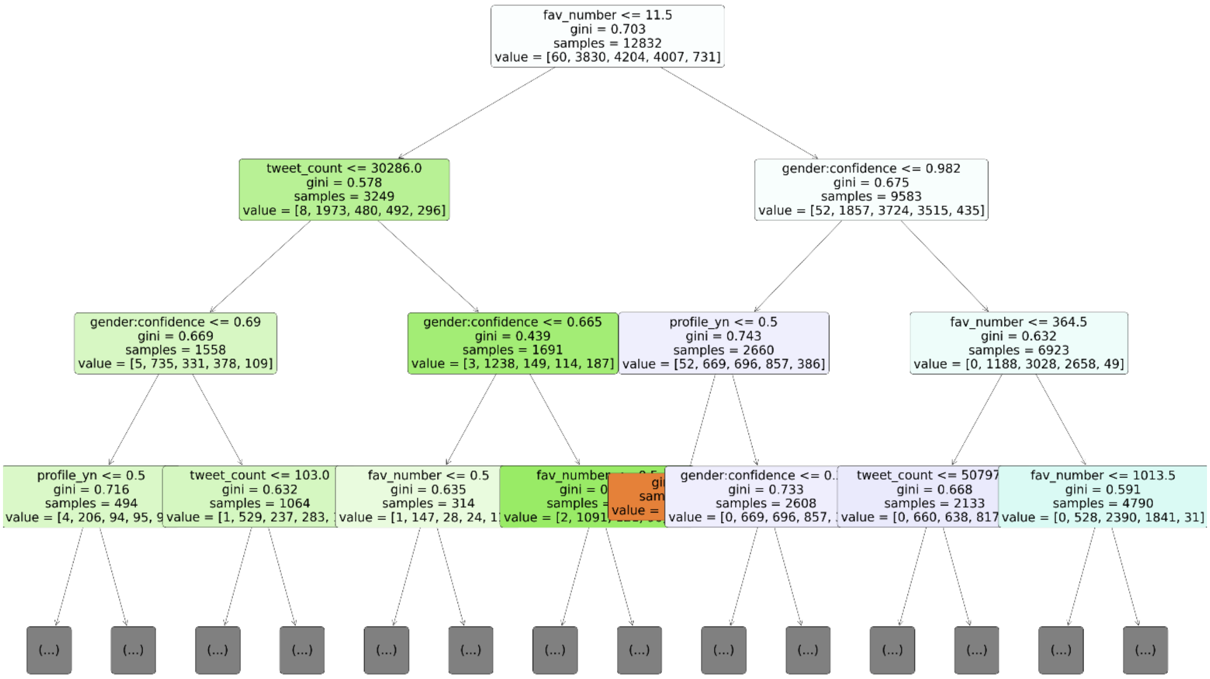
_golden
_trusted_judgments
gender:confidence
profile_yn:confidence
fav_number
retweet_count
tweet_count
tweet_id
_golden: 1 1 0.0088 0.0056 0.0038 -0.00074 -0.0097
_trusted_judgments: 1 1 0.0095 0.0055 0.0039 -0.00077 -0.0097
gender:confidence: 0.0088 0.0095 1 0.31 -0.031 -0.0049 -0.014
profile_yn:confidence: 0.0056 0.0055 0.31 1 -0.0033 0.0014 -0.013
fav_number: 0.0038 0.0039 -0.031 -0.0033 1 0.0083 0.067
retweet_count: -0.00074 -0.00077 -0.0049 0.0014 0.0083 1 -0.0013
tweet_count: -0.0097 -0.0097 -0.014 -0.013 0.067 -0.0013 1
1.0
0.8
0.6
0.4
0.2
0.0

Selecting important columns for x and target column to y.

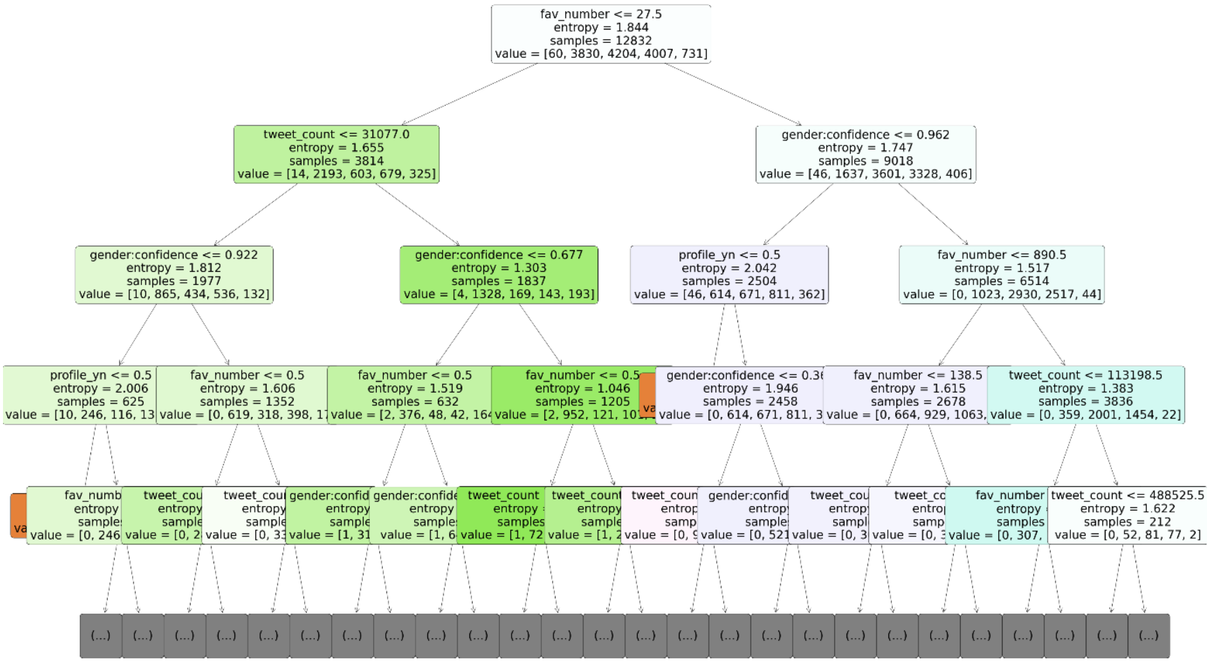
Last is the divide dataset in train, test, and validation.

4)Visualization of the decision tree for Gini and Entropy.

Gini-index (Depth = 3)



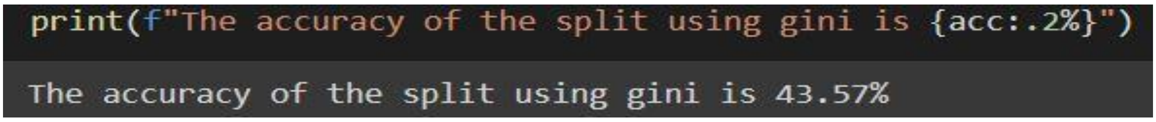
Entropy (Depth = 4)



5. Interpret your results, compare Gini and Entropy.

Using the gini to measure the quality of split give the accuracy of split is 43.57% and using the entropy to measure the quality of split give the accuracy of split is 46.61%. so, we can see that entropy split gives better accuracy compared to gini split for the given dataset using important features of the dataset.

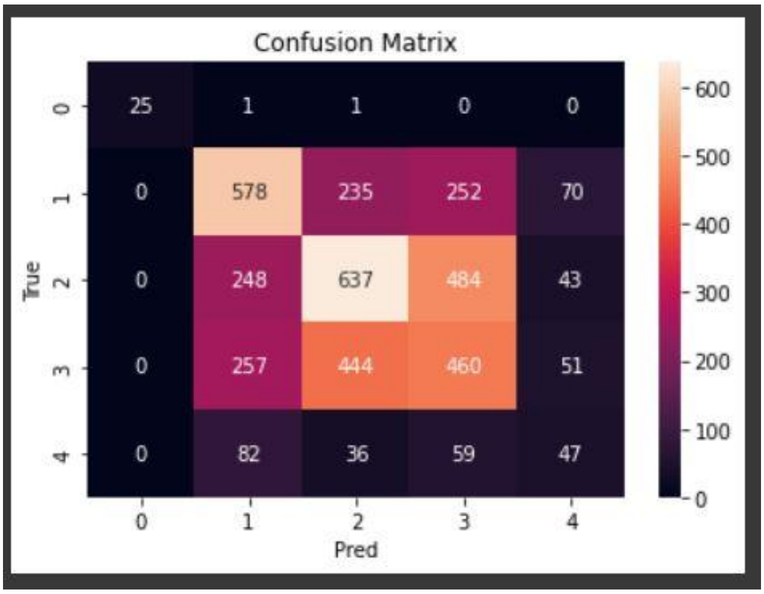
Using gini:



```
print(f"The accuracy of the split using gini is {acc:.2%}")
```

```
The accuracy of the split using gini is 43.57%
```

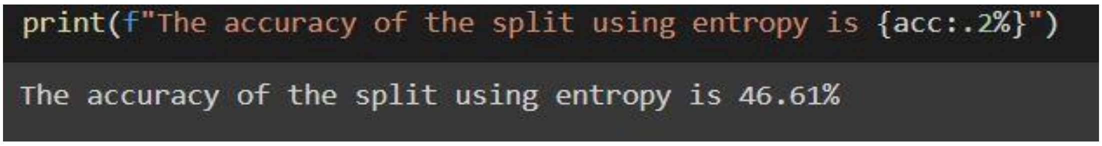
Confusion matrix for gini:



Confusion Matrix

True \ Pred	0	1	2	3	4
0	25	1	1	0	0
1	0	578	235	252	70
2	0	248	637	484	43
3	0	257	444	460	51
4	0	82	36	59	47

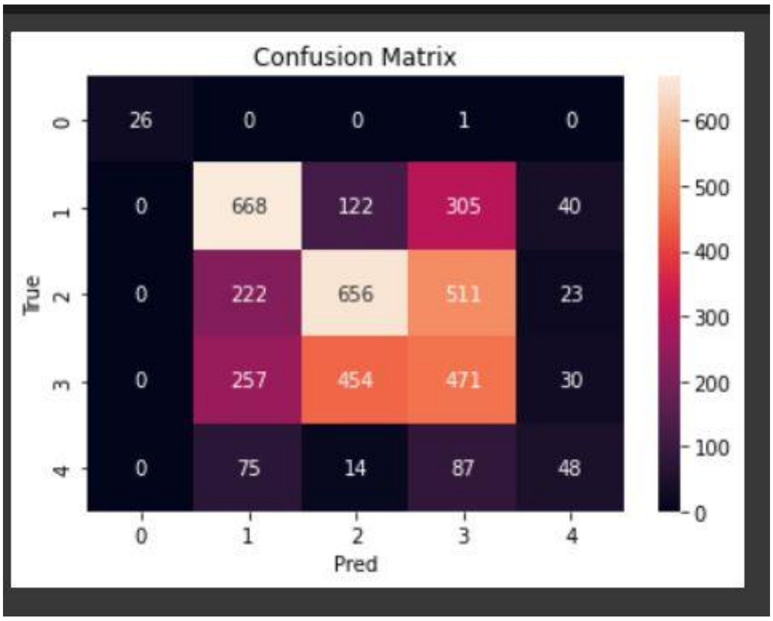
Using entropy:



```
print(f"The accuracy of the split using entropy is {acc:.2%}")
```

```
The accuracy of the split using entropy is 46.61%
```

Confusion matrix for entropy:

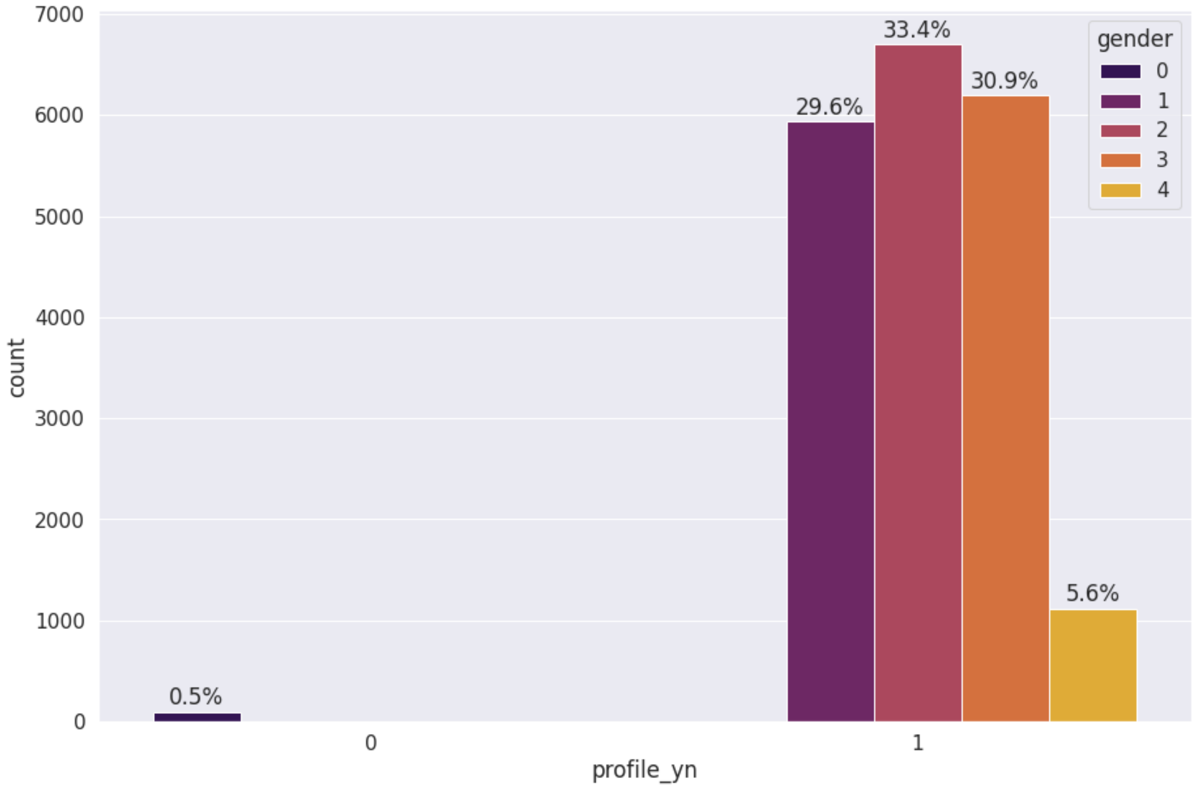


True \ Pred	0	1	2	3	4
0	26	0	0	1	0
1	0	668	122	305	40
2	0	222	656	511	23
3	0	257	454	471	30
4	0	75	14	87	48

I construct the decision tree using gini and entropy and we can see that both tree looks like similar but only values of gini and entropy is slightly different.

6) Visualize the dataset for the target variable.

1st visualization:



From the above graph it is inferred that where profile_yn = 1, the distribution of gender is as follows:

{'Other': 0, 'brand': 1, 'female': 2, 'male': 3, 'unknown': 4}

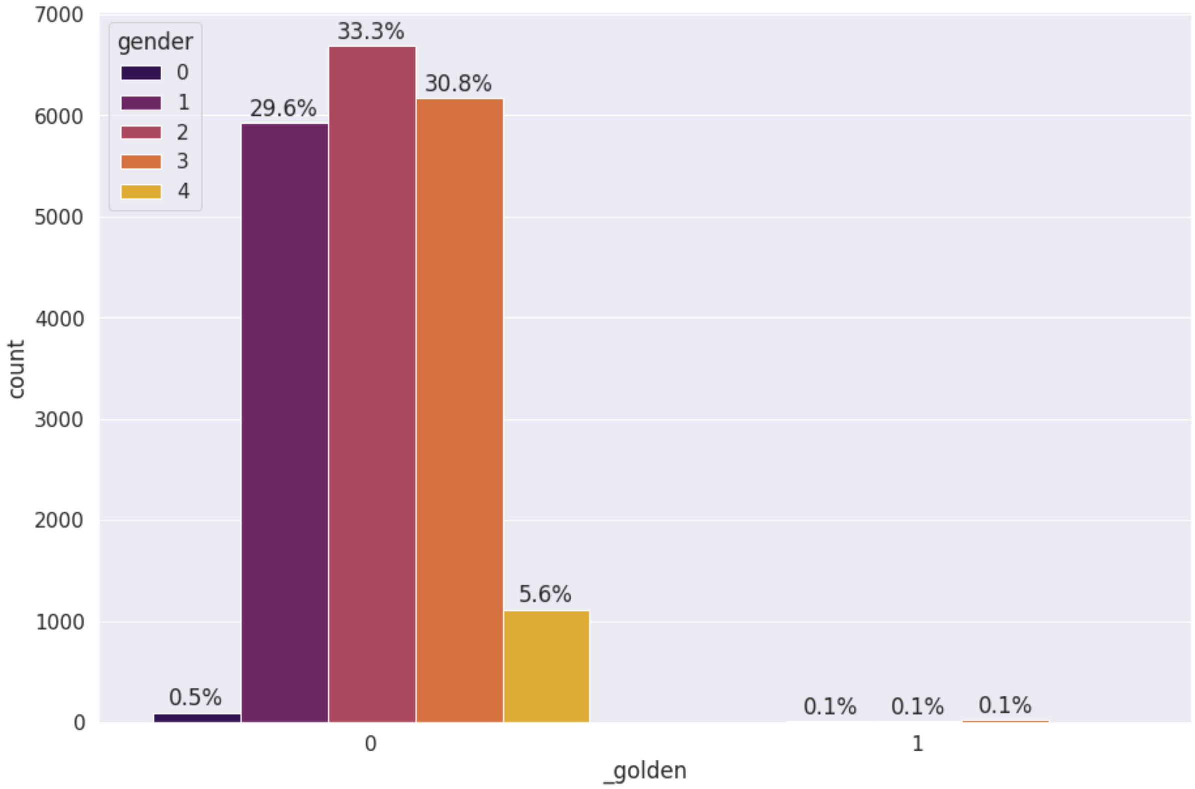
brand - 29.6%

female - 33.4%

male - 30.9%

unknown - 5.6%

2nd visualization:



From the above graph it is inferred that where _golden = 0, the distribution of gender is as follows:

{'Other': 0, 'brand': 1, 'female': 2, 'male': 3, 'unknown': 4}

Other - 0.5%

brand - 29.6%

female - 33.3%

male - 30.8%

unknown - 5.6%

References:

For error solving: - https://stackoverflow.com/questions/48067514/utf-8-codec-cant-decode-byte-0xa0-in-position-4276-invalid-start-byte

For pre-processing: -

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

For visualization: - https://seaborn.pydata.org/generated/seaborn.heatmap.html

https://scikit-learn.org/stable/modules/tree.html

https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/