# ML MAJOR PROJECT SUMMARY

*NAME:-* *ASTHA VERMA*
*Email ID:-* *astha.18bit1192@abes.ac.in*
*astharocky390@gmail.com*
*GOOGLE CLASS ROOM:-* *ML063B1*

## Dataset:-

**Name:-** "Information.csv"

**Source:-** Email enclosed with dataset from Event@verzeo.in

**Description:-** Accurate prediction of demographic attributes from social media and other informal online content is valuable for marketing, personalization, and legal investigation. This Dataset consists of user's gender classification in social media, with an application to Twitter. Twitter does not collect users' self-reported gender as do other social media sites (e.g., Facebook and Google), but such information could be useful for targeting a specific audience for advertising, for personalizing content, and for legal investigation. "In this Dataset We describe the construction of dataset labeled with gender and utilizes the machine learning approach for determining the gender of Twitter users. We test the accuracy of our approach by varying various tokenizer options , algorithms and find the best fit."

## Classification Algorithms Chosen:-

1. Naive Bayes Classification Algorithm

2. Logistic Regression

3. Random Forest

## Questions Chosen:-

**1 .) Most common words used by males and females in their tweets ?**

**A:-** For finding this we used ***FreqDist( )***- frequency distribution function to count the occurance of particular word & ***intersection*( )**-function to find common words between male and female and can be easily visualized from the **count Vs word** plot

**Common words are:**

***{get,like,one,day,time,u,love,new,go,amp,people,make,know,best,see,got,year,good}***

**2.) Which day attracted the new users the most ?**

**A:-** For getting the answer we used ***Groupby( )*** function to split the data into groups based on day & ***nunique( )*** function to get a count of unique days. Which can be easily visualized from **piechart.**

***"Wednesday" attracted most number of users***

**3.) Twitter users with the highest tweet_count and the highest re-tweet count?**

**A:-** For getting the answer we used max( ) function to find :

User with highest tweet count is " ***Mr. Gabriel dsmqs" count = 2680199***

User with highest re-tweet count is " ***MarcusButler"*** ***count= 153***

## EDA and Working of Algorithms:-

- Firstly, we performed info( ) function on data set to obtain the information about the DataFrame including the index type and column dtypes and especially to point out the Null values in the data set.

- We then found null values in columns :- { gender, gender:confidence, description, gender_gold, profile_yn_gold, tweet_coord, tweet_location, user_timezone }.

- Then we dropped columns with which aren't necessary for our Objective and also dropped null values in necessary columns for better results.

- We then check for duplicate entries in this dataset and but didn't found any.

- Lastly , we removed the values of gender which are unknown and also eliminated the users that were not 100% confident about their gender.

## Data Visualisation:-

- We plotted all relevant columns in various types of pie-charts and bar graphs and plots to get a nice visual representation of our data set.

## Feature Engineering:-

- After careful analysis of our data, keeping in mind all the questions we wanted to answer using the data , we came to conclusion that most our data is not clean and featurized.

- So, we changed columns named :- 'description' & 'text' into string type & then removed special characters from the data.

- For removing Special characters we used Regular Expressions as they are powerful tools for extracting and filtering data. Especially they are used by string-searching algorithms for "find" or "find and replace" operations on strings

- Finally , we used count vectorizer to transform Text and description in to features used to predict the accuracy.

**Regular expressions:-** **command-** ***import re***

| | |
|---|---|
| **^** | Matches the beginning of a line |
| **$** | Matches the end of the line |
| **.** | Matches any character |
| **\s** | Matches whitespace |
| **\S** | Matches any non-whitespace character |

|  |  |
|---|---|
| ***** | Repeats a character zero or more times |
| ***?** | Repeats a character zero or more times (non-greedy) |
| **+** | Repeats a character one or more times |
| **+?** | Repeats a character one or more times (non-greedy) |
| **[aeiou]** | Matches a single character in the listed set |
| **[^XYZ]** | Matches a single character *not* in the listed set |
| **[a-z0-9]** | The set of characters can include a range |
| **(** | Indicates where string extraction is to start |
| **)** | Indicates where string extraction is to end |

- Later we removed stop words from the data to improve the performance of the algorithm & to increase classification accuracy.
- Lastly, we used ***Lematization*** to extract the ***Rootword.***

## Assigning Variables and Setting Train/Test /Split:-

- Keeping in mind the questions we choose to answer with this dataset,we assigned a set of variables as independent variables and put them in a list and put the dependent variable in another list.

- **Dependent variable:-** ***" Gender"***
- **Independent variable:-** ***"Everything else that remained after EDA"***

- After this ,we set the data into Train/Test /Split and proceeded with our choosen models.

## Ensemble Machine Learning Modelling:-

### 1. Naive Bayes Classification Algorithm:

- For this model , we simply imported the concerned module from the sklearn package and passed our data set to the concerned function from the imported module.

- For calculating the Accuracy we imported the module **accuracy_score** from ***sklearn.metrics***

- Finally, we calculated the accuracy of the model in predicting the data.

**Concerned module:-** ***from sklearn.naive_bayes import MultinomialNB***

**Concerned function:-** ***nb = MultinomialNB()***

***Advantages of Naive Bayes:***

- The **Naive Bayes** algorithm affords fast, highly scalable model building and scoring. It scales linearly with the number of predictors and rows. The build process for **Naive Bayes** is parallelized.

***Disadvantages of Naive Bayes:***

- Main limitation of Naive Bayes is the **assumption of independent predictors**. Naive Bayes implicitly assumes that all the attributes are mutually independent. In real life, it is almost impossible that we get a set of predictors which are completely independent.

### 2. Logistic Regression:

- For this model , we imported the concerned module from the sklearn package and passed our data set to the concerned function from the imported module.

- For calculating the Accuracy we imported the module **accuracy_score** from ***sklearn.metrics***

- Finally, we calculated the accuracy of the model in predicting the data.

**Concerned module:-** ***from sklearn.linear_model import LogisticRegression***

**Concerned function:-** ***logReg= LogisticRegression()***

***<u>Advantages of Logistic Regression:</u>***

1. Logistic Regression performs well when the **dataset is linearly separable**.
2. Logistic Regression not only gives a measure of how relevant a predictor (coefficient size) is, but also its direction of association (+ve or -ve).
3. Logistic regression is easier to implement, interpret and very efficient to train.

***<u>Disadvantages of Logistic Regression:</u>***

1. Main limitation of Logistic Regression is the **assumption of linearity** between the dependent variable and the independent variables. In the real world, the data is rarely linearly separable. Most of the time data would be a jumbled mess.

2. If the number of observations are lesser than the number of features, Logistic Regression should not be used, otherwise it may lead to overfit.

**3.** Logistic Regression can only be **used to predict discrete functions**. Therefore, the dependent variable of Logistic Regression is restricted to the discrete number set. This restriction itself is problematic, as it is prohibitive to the prediction of continuous data.

## 3. Random Forest :

- For this model , we imported the concerned module from the sklearn package and passed our data set to the concerned function from the imported module.

- For calculating the Accuracy we imported the module **accuracy_score** from ***sklearn.metrics***

- Finally, we calculated the accuracy of the model in predicting the data.

**Concerned module:-** ***from sklearn.ensemble import RandomForestClassifier***
**Concerned function:-** ***clf= RandomForestClassifier ()***

***Advantages of Random Forest:***

1. Random Forest is based on the **bagging** algorithm and uses **Ensemble Learning** technique. It creates as many trees on the subset of the data and combines the output of all the trees. In this way it **reduces overfitting** problem in decision trees and also **reduces the variance** and therefore **improves the accuracy**.

2. Random Forest can be used to **solve both classification as well as regression problems**.

3. Random Forest works well with both **categorical and continuous variables**.

***Disdvantages of Random Forest:***

**1. Complexity:** Random Forest creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources. On the other hand decision tree is simple and does not require so much computational resources.

2. **Longer Training Period:** Random Forest require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes.

## Comparing Accuracies:-

### Considering Text:-

- **1.Naive Bayes Algorithm:-** ***52.44%***
- **2.Logistic Regression:-** ***51.09%***
- **3.Random Forest:-** ***49.01%***

### Considering Deccription:-

- **1.Naive Bayes Algorithm:-** ***67.08%***
- **2.Logistic Regression:-** ***65.31%***
- **3.Random Forest:-** **63.04%**

**Considering Text:-**
**Accuracy order:-** ***Random forest< Logistic Regression <Naive Bayes***

**Considering Description:-**
**Accuracy order :-** ***Random forest< Logistic Regression < Naive Bayes***

## Conclusion:-

- We can clearly say that ***Naive Bayes Algorithm*** wins in terms of accuracy, followed by *Logistic regression & Random Forest*
- Ensemble methods helps improve machine **learning** results by combining multiple models. Using ensemble methods allows to produce better predictions compared to a single model. Therefore, the ensemble methods placed first in many prestigious machine **learning** competitions, such as Netflix Competition, KDD 2009, and Kaggle
- So, here we used Naive bayes , Logistic Regression & Random Forest for getting best accuracy in prediciting the ***TARGET VARIABLE.***
- Our target variable is ***Gender***
- Generally ***accuracy-score*** is nothing but ratio of the number of correct predictions to the total number of predictions.

**I.e.**

Accuracy = $\frac{Total\ Positive\ Prediction}{Total\ Number\ of\ Prediction}$

(OR)

Accuracy = $\frac{(TP+TN)}{(TP+FP+FN+TN)}$

**Where,**

***TP=***True Positives

***TN=***True Negatives

***FP=*** False Positives

***FN=*** False Negatives