Summary of Major Project

(by group ML09B5)

Libraries used: pandas, numpy, spellchecker, nltk, sklearn.

EDA and Cleaning:

- At first, only the rows with **'gender:confidence' > 0.99** were kept. Then, only the **'male'** and **'female'** genders were kept and the rest were dropped.
- Then, after analyzing the dataset, those columns which contained mostly same values or null values (for example, 'gender_gold' column contained only 50 non-null values, and every entry in the 'tweet_created' column is identical) were dropped.
- Dropped also were the columns containing values that were unimportant to the ML algorithm (such as '_unit_id').
- Finally, all rows containing any null values anywhere were removed, and thus a fully cleaned and modified dataframe **'df_filtered'** was obtained.

Questions asked on the dataset after EDA and cleaning:

Q1) What is the average number of typos made by males and females in their respective tweets?

Ans) For males: 0.829219, For females: 0.928594

Explanation: We first created a function **remove_mention_url**, which would remove all hashtags (i.e., all words beginning with '#'), mentions (i.e., all words beginning with '@') and URLs (i.e., all words beginning with 'http' or 'www'), since those aren't considered as spelling errors, and stored the cleaned text in an array. With the help of this function, as well as the functions **SpellChecker** and **RegexpTokenizer**, we counted the number of typos in each cleaned text tweet. These values were then stored in a new column **'typos'**.

Then, we separated the typos for male entries in a new dataframe **'df_male'** and calculated the average of all the values stored in it. The same process was repeated for female entries with the new dataframe **'df_female'**.

Q2) What is the most common sidebar colour for male and female twitter accounts?

Ans) For males: C0DEED, For females: FFFFFF

Explanation: We separated the male and female entries into 2 separate dataframes **'df_male'** and **'df_female'** respectively, and then ran the **value_counts()** function on the **'sidebar_color'** column of 'df_male' **and 'df_female'** to find out the most common sidebar colour for male and female twitter accounts.

Algorithms used: Naive Bayes Classifier, Logistic Regression, Random Forest Classifier

Dependent variable: 'gender'

Independent variables: 'all_text_features', 'typos', 'tweet_count', 'link_color_e', 'sidebar_color_e', 'user_timezone_e'

Steps undertaken for Naive Bayes Classifier model:

We wanted to create a ML algorithm that would take as input the description, text, hashtags and mentions (minus the URLs and typos) of an account as input and then try to predict the gender of that account based on the supplied information. For this, we created a function **remove_url** that would remove the URLS (words starting with 'http' or 'www') in the input text and store the output in a clean array. We also created a function **data_prep_for_nb** which, with the help of the previously created **remove_url** function, as well as the function **RegexpTokenizer**, eliminated the URLs and typos in each entry of the columns **'text'** and **'decription'** and stored the outputs in the new columns **'text_normalized'** and **'description_normalized'** respectively. Each entry of these two columns were then concatenated and stored in a new column **'all_text_features'**. We passed the **'df_filtered'** array through the **data_prep_for_nb** function, and obtained the corresponding **'all_text_features'** column.

We then transformed the column **'all_text_features'** into a sparse matrix using the **CountVectorizer** function, and this sparse matrix **'x'** became the independent variable. The dependent variable was **'gender'** column transformed via **LabelEncoder** function, which we named **'y'**.

Finally, we ran the **train_test_split** function on x and y, and passed the **x_train** and **y_train** through the **MultinomialNB** function.

Steps undertaken for Logistic Regression and Random Forest Classifier models:

First we transformed the columns **'link_color'** and **'sidebar_color'** and **'user_timezone'** via **LabelEncoder** function and stored the converted values in the new columns **link_color_e'**, **'sidebar_color_e'**, and **'user_timezone_e'** respectively.

Then, the columns **'typos'**, **'tweet_count'**, **'link_color_e'**, **'sidebar_color_e'**, and **'user_timezone_e'**, which were the independent variables, were passed through **'X'** and the column **'gender'**, which was the dependent variable, was passed through **'Y'**. Finally, we ran the **train_test_split** function on X and Y, and passed the **X_train** and **Y_train** through **LogisticRegression** and **RandomForestClassifier** functions.

Accuracy of Naive Bayes Classifier model: 0.679109364767518.

Accuracy of Logistic Regression model: 0.6779802974374961

Accuracy of Random Forest Classifier model: 0.610347085789129

Therefore, the most accurate model is the **Naive Bayes Classifier.**

Prediction of a random input by majority vote:

We created an array **'test_row'** which contained some random entries, which we then converted into a dataframe **'test_df'** using **pd.DataFrame** function. Then we passed this dataframe through the previously created **data_prep_for_nb** function and obtained the respective **'all_text_features'** array, which we then converted into a sparse matrix via **CountVectorizer** function and passed through the **MultinomialNB.predict** function. The obtained **'y_predicted'** output was then fed into **LabelEncoder.inverse_transform** function which then predicted the gender as **'male'**. So, the prediction according to the **Naive Bayes Classifier** model is **male**.

Then, we transformed the **'link_color'**, **'sidebar_color'** and **'user_timezone'** columns of **'test_df'** using **LabelEncoder** function into **'link_color_e'**, **'sidebar_color_e'** and **'user_timezone_e'** respectively. Then, the columns **'typos'**, **'tweet_count'**, **'link_color_e'**, **'sidebar_color_e'**, and **'user_timezone_e'** of **'test_df'** were stored in a new datframe **'df_with_feateng'**. This **'df_with_feateng'** array was then passed through the functions **LogisticRegression.predict** and **RandomForestClassifier.predict**, which predicted the gender as **'male'** and **'female'** respetively. So, the prediction according to the **Logistic Regression** model is **male**, and the prediction according to the **Random Forest CLassifier** model is **female**.

Thus, by majority vote, the predicted gender of the random input is **male**.