MAJOR PROJECT (DS-MAJOR-NOV) INTERNSHIP

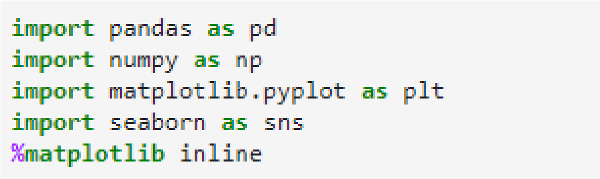
TEAM MEMBERS:

1. Sowmya Chowdary
2. Evanka D'souza
3. Keerthana k
4. Sai Karran mk
5. Gautam MG
6. Kruthika P
7. Kanishk Soman
8. Pal Varma
9. Yashasvi verma
10. Shakeeb Ilmi Noor

DATASET USED:

Information.csv (Twitter dataset)

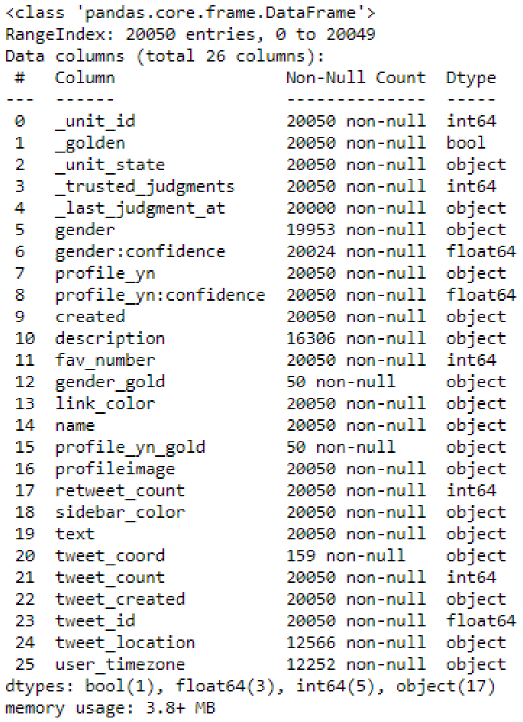
Libraries used:



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Libraries used are pandas, numpy, matplotlilb and seaborn.

Information for the given dataset:



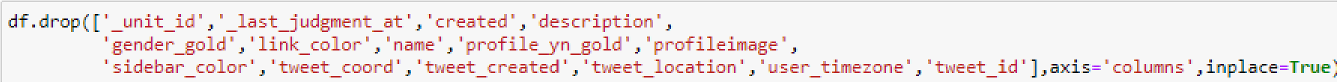
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20050 entries, 0 to 20049
Data columns (total 26 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   _unit_id               20050 non-null  int64  
 1   _golden                20050 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                 19953 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
```

The total entries are 20050 and 25 columns

1 boolean, 3 float, 5 int and 17 object values

Exploratory data analysis and Data Cleaning

Dropping unnecessary values:

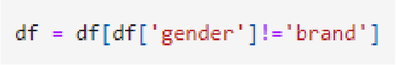


```
df.drop(['_unit_id','_last_judgment_at','created','description',
         'gender_gold','link_color','name','profile_yn_gold','profileimage',
         'sidebar_color','tweet_coord','tweet_created','tweet_location','user_timezone','tweet_id'],axis='columns',inplace=True)
```

Columns name 'gender_gold', 'profile_yn_gold', 'tweet_coord'**, have least non-null values so we will remove them.**

Columns name '_unit_id', '_last_judgment_at', 'created', 'description', 'link_color', 'name', 'tweet_id', 'profileimage', 'sidebar_color', 'tweet_created', 'tweet_location', 'user_timezone' **are not to be used for the analysis and do not need to be encoded, so we will also remove them.**

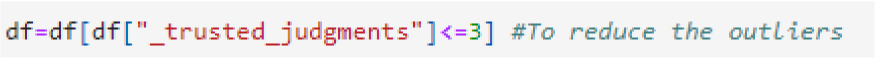
Removing 'Brand' out of gender:



```
df = df[df['gender']!='brand']
```

Gender column has 4 values, out of which 'brand' is unnecessary and has to be removed.

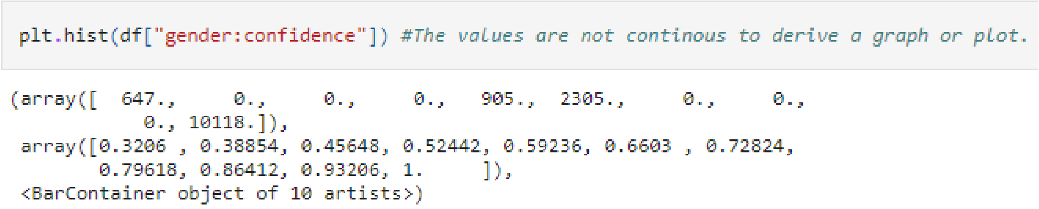
Reducing the outliers:



```
df=df[df["_trusted_judgments"]<=3] #To reduce the outliers
```

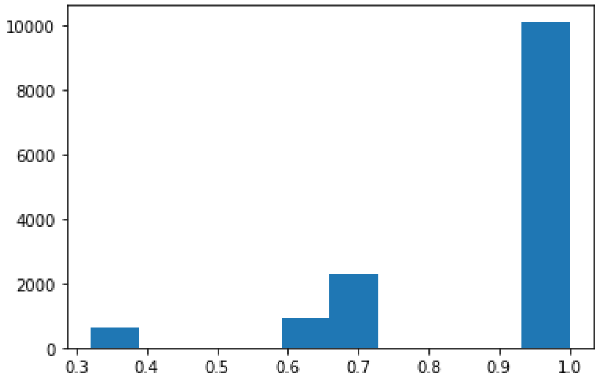
The column name ‘trusted_judgements’ should be greater than or equal to 3 then only it will be considered important

Plotting histogram for ‘gender:confidence’:



```
plt.hist(df["gender:confidence"]) #The values are not continous to derive a graph or plot.
```

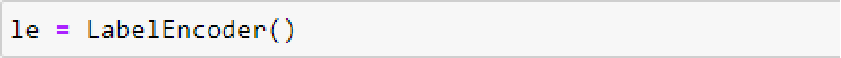
```
(array([  647.,     0.,     0.,     0.,   905.,  2305.,     0.,     0.,
          0., 10118.]),
 array([0.3206 , 0.38854, 0.45648, 0.52442, 0.59236, 0.6603 , 0.72824,
        0.79618, 0.86412, 0.93206, 1.     ]),
 <BarContainer object of 10 artists>)
```



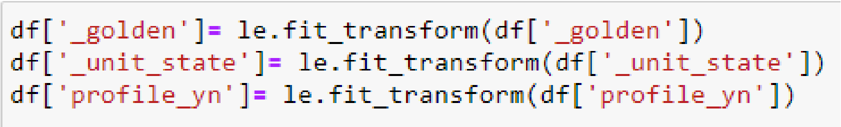
Histogram shows how many people have judgements whose ‘gender_confidence’ is shown.

Label Encoding (to convert string data into int data)

```
from sklearn.preprocessing import LabelEncoder
```



```
le = LabelEncoder()
```



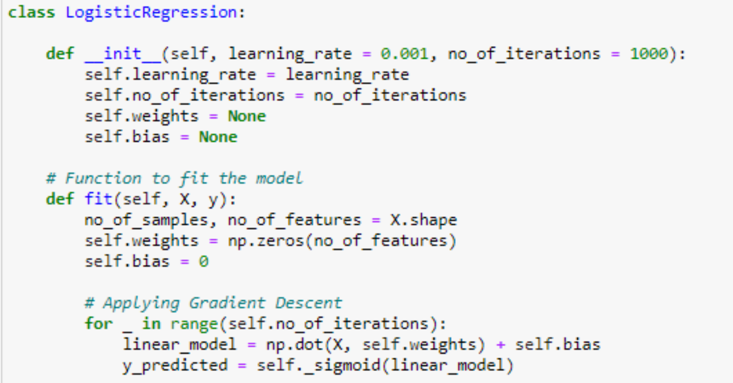
```
df['_golden']= le.fit_transform(df['_golden'])
df['_unit_state']= le.fit_transform(df['_unit_state'])
df['profile_yn']= le.fit_transform(df['profile_yn'])
```

Label encoder convers string values to int values that can be used later for the training so ‘_golden’, ‘_unit_state’ and ‘profile_yn’ **are converted.**

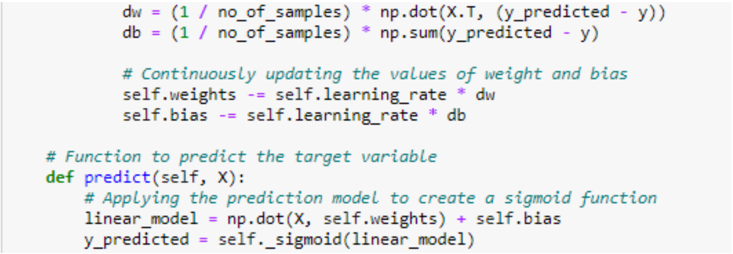
Implementing Logistic Regression Model From Scratch

Creating a Class for Performing Logistic Regression

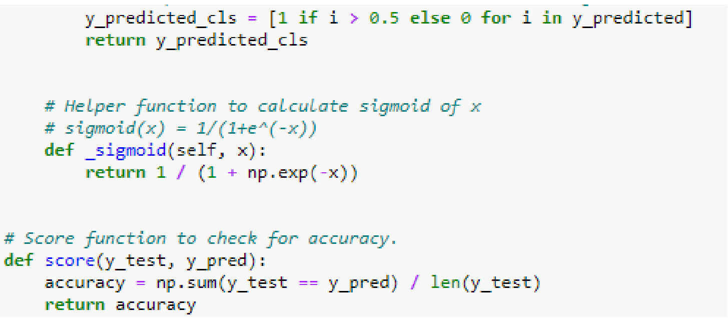
STEPS FOR CREATING CLASS



```
class LogisticRegression:

    def __init__(self, learning_rate = 0.001, no_of_iterations = 1000):
        self.learning_rate = learning_rate
        self.no_of_iterations = no_of_iterations
        self.weights = None
        self.bias = None

    # Function to fit the model
    def fit(self, X, y):
        no_of_samples, no_of_features = X.shape
        self.weights = np.zeros(no_of_features)
        self.bias = 0

        # Applying Gradient Descent
        for _ in range(self.no_of_iterations):
            linear_model = np.dot(X, self.weights) + self.bias
            y_predicted = self._sigmoid(linear_model)
```



```
            dw = (1 / no_of_samples) * np.dot(X.T, (y_predicted - y))
            db = (1 / no_of_samples) * np.sum(y_predicted - y)

            # Continuously updating the values of weight and bias
            self.weights -= self.learning_rate * dw
            self.bias -= self.learning_rate * db

    # Function to predict the target variable
    def predict(self, X):
        # Applying the prediction model to create a sigmoid function
        linear_model = np.dot(X, self.weights) + self.bias
        y_predicted = self._sigmoid(linear_model)
```

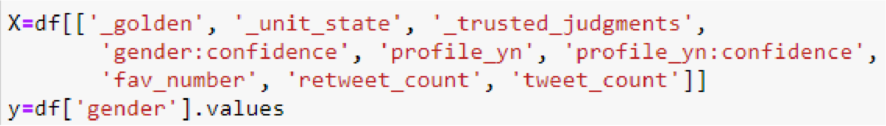


```
        y_predicted_cls = [1 if i > 0.5 else 0 for i in y_predicted]
        return y_predicted_cls


    # Helper function to calculate sigmoid of x
    # sigmoid(x) = 1/(1+e^(-x))
    def _sigmoid(self, x):
        return 1 / (1 + np.exp(-x))


# Score function to check for accuracy.
def score(y_test, y_pred):
    accuracy = np.sum(y_test == y_pred) / len(y_test)
    return accuracy
```

Ensemble Machine learning Modelling

Classification Algorithms

Assigning values for train and test (‘X’ and ‘Y’):



```
X=df[['_golden', '_unit_state', '_trusted_judgments',
        'gender:confidence', 'profile_yn', 'profile_yn:confidence',
        'fav_number', 'retweet_count', 'tweet_count']]
y=df['gender'].values
```

X values are the values that we are providing as an input to our data, that will be trained. Y will be given as the output.

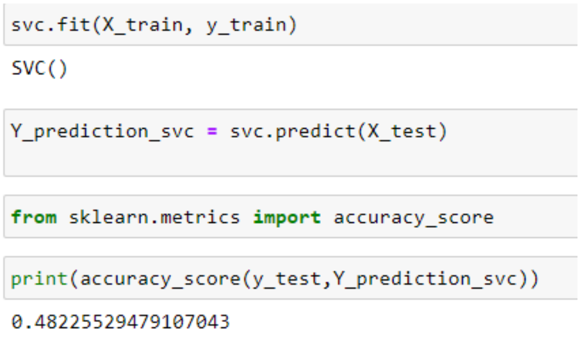
X_train is the given input

Y_train is the given output

X_test is the testing input

Y_train is the testing output.

1) SVM Algorithm



```
svc.fit(X_train, y_train)
```

```
SVC()
```

```
Y_prediction_svc = svc.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
```

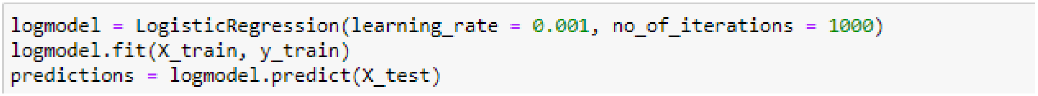
```
print(accuracy_score(y_test,Y_prediction_svc))
```

```
0.48225529479107043
```

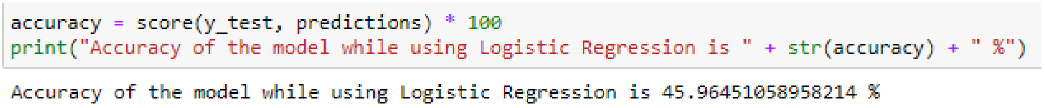
The SVM algorithm is trained and using accuracy score we have got the output as 0.48, so he accuracy for SVM algorithm is 48%

2) Logistic Regression

LOGISTIC ALGORITHM IS MADE FROM SCRATCH AND NO EXTERNAL LIBRARIES ARE USED FOR THIS



```
logmodel = LogisticRegression(learning_rate = 0.001, no_of_iterations = 1000)
logmodel.fit(X_train, y_train)
predictions = logmodel.predict(X_test)
```

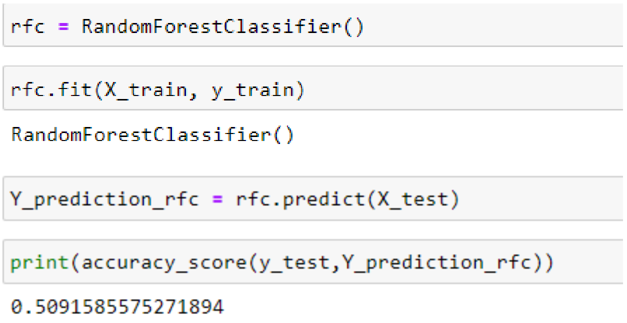


```
accuracy = score(y_test, predictions) * 100
print("Accuracy of the model while using Logistic Regression is " + str(accuracy) + " %")
```

```
Accuracy of the model while using Logistic Regression is 45.96451058958214 %
```

The Linear Regression algorithm is trained and using accuracy score we have got the output as 45.96 so the accuracy for Logistic Regression algorithm is 46%

3) Random Forest Classifier



```
rfc = RandomForestClassifier()
```

```
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
Y_prediction_rfc = rfc.predict(X_test)
```

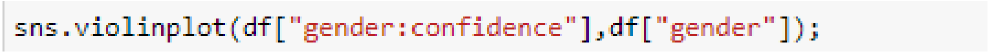
```
print(accuracy_score(y_test,Y_prediction_rfc))
```

```
0.5091585575271894
```

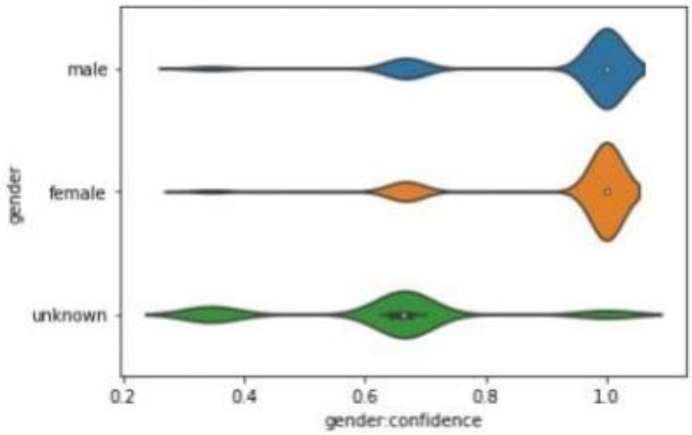
The RFC algorithm is trained and using accuracy score we have got the output as 0.50 so the accuracy for Random Forest Classifier algorithm is 51%

Questions asked on dataset and answers for the same with brief explanation

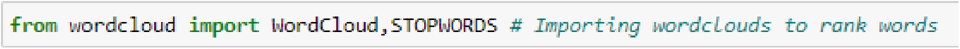
Q1) Is there any outlier after data cleaning?



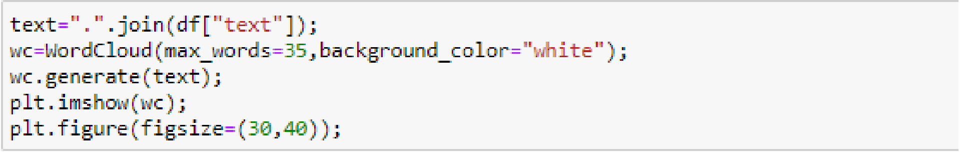
```
sns.violinplot(df["gender:confidence"],df["gender"]);
```



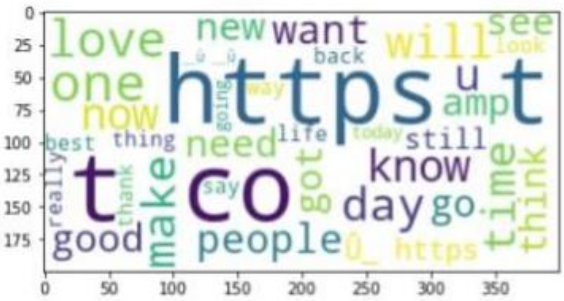
Q2) How many keywords are frequently repeated in column 'text'?

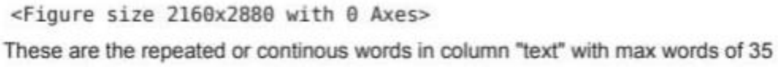


```
from wordcloud import WordCloud,STOPWORDS # Importing wordclouds to rank words
```



```
text=".".join(df["text"]);
wc=WordCloud(max_words=35,background_color="white");
wc.generate(text);
plt.imshow(wc);
plt.figure(figsize=(30,40));
```

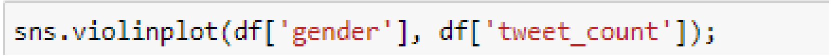




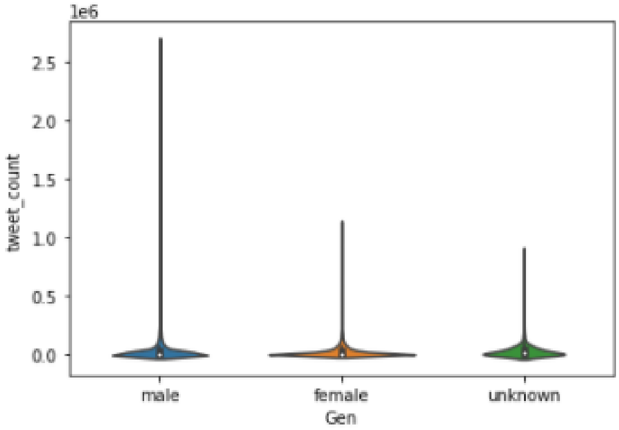
```
<Figure size 2160x2880 with 0 Axes>
```

These are the repeated or continous words in column "text" with max words of 35

Q3) What is the relation between gender and tweet counts?



```
sns.violinplot(df['gender'], df['tweet_count']);
```


1e6
2.5
2.0
1.5
1.0
0.5
0.0
tweet_count
male
female
unknown
Gen