# Fake News Detection

Byte-Sized brains

# Data preprocessing

- At first, we had data of 44919 rows and 6 columns (ID, title, text, subject, date, and class).
- It contained some duplicated, null data that won't help in training the model.
- We processed with various techniques to preprocess the data like drop duplicates and null in addition to casting the data type of the columns.

In [3]:
```
df = pd.read_csv(file_path)
df
```

Out[3]:

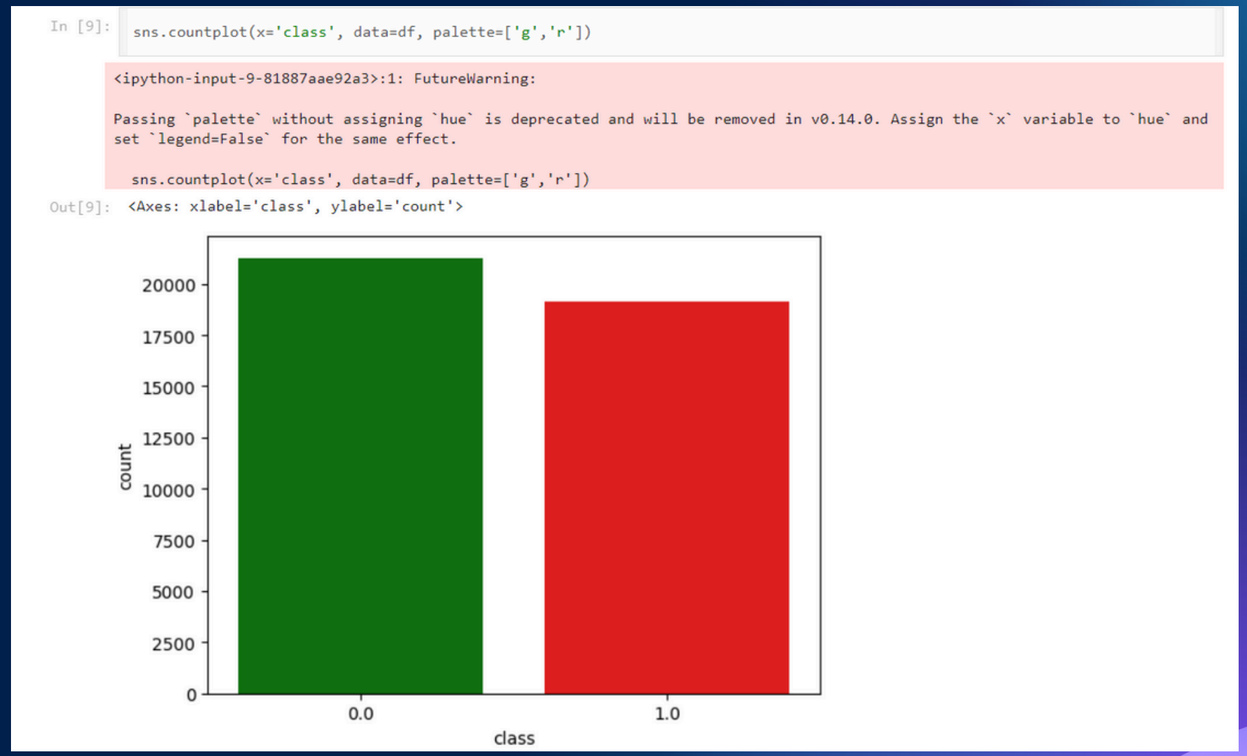
| | ID | title | text | subject | date | class |
|---|---|---|---|---|---|---|
| **0** | 0.0 | #AfterTrumpImplodes Hashtag Hilariously Imagi... | What will the world be like post-Donald Trump?... | News | 5-Aug-16 | 0.0 |
| **1** | 1.0 | #BlackLivesMatter Leader To Run For Mayor Of ... | The police shooting of black teen Michael Brow... | News | 4-Feb-16 | 0.0 |
| **2** | 2.0 | #BringBackObama Hashtag Blows Up On Twitter A... | The six months since President Donald Trump wa... | News | 13-Jul-17 | 0.0 |
| **3** | 3.0 | #FreeChrisChristie: Twitter Reacts To The 'Ho... | Last Friday, New Jersey Governor Chris Christi... | News | 2-Mar-16 | 0.0 |
| **4** | 4.0 | #MakeAmericaBrannigan: Futurama Voice Actor R... | The incredibly talented voice actor, Billy Wes... | News | 13-Aug-16 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **44914** | NaN | NaN | NaN | NaN | NaN | NaN |
| **44915** | NaN | NaN | NaN | NaN | NaN | NaN |
| **44916** | NaN | NaN | NaN | NaN | NaN | NaN |
| **44917** | NaN | NaN | NaN | NaN | NaN | NaN |
| **44918** | NaN | NaN | NaN | NaN | NaN | NaN |

44919 rows × 6 columns

# Data preprocessing

- First thing was done is to drop the duplicates. Then dropping the null values in ‘class’ column.
- However, there were 21 null values in ‘ID’ column. Since we are not using this column in training the model we dropped it.
- We made a count plot to estimate how much there are fake and real news.

```
In [9]: sns.countplot(x='class', data=df, palette=['g','r'])

<ipython-input-9-81887aae92a3>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and
set `legend=False` for the same effect.

  sns.countplot(x='class', data=df, palette=['g','r'])

Out[9]: <Axes: xlabel='class', ylabel='count'>
```



# Data preprocessing

- Taking a look into ‘subject’ unique values, we found similar values like politicsNews and politics. In addition to some values that have different names but equal values.
- left-news were a subset from politics. In addition to having polticsNews and politics which were mainly focusing on the same category of news.
- Therefore, we applied lambda function to change the subject into politics for both.

In [12]:

```
politics_news = df[df['subject'] == 'politics']
left_news = df[df['subject'] == 'left-news']

# Display sample titles and text for comparison
print(politics_news[['title', 'text']].head())
print(left_news[['title', 'text']].head())
```

```
                                                  title  \
9051  "It would be huge": U.S. border town confronts...
9052  "Make Republicans Whole Again!" A divided part...
9057  #AnyoneButHillary: NEW POLL Shows Bernie Suppo...
9059  #Austin: Fights Break Out Between Police and S...
9061  #Berkeley CRAZY! RIOTERS CHASE And Beat People...

                                                  text
9051  NOGALES, Arizona (Reuters) - For up to 16 hour...
9052  CLEVELAND (Reuters) - Ted Cruz was recounting ...
9057  Hillary may find out she needs more than black...
9059  Commies carrying flags fought with the Austin ...
9061  Is anyone else thinking what we re thinking ab...
                                                  title  \
9058  #AnyoneButHillary: NEW POLL Shows Bernie Suppo...
9060  #Austin: Fights Break Out Between Police and S...
9062  #Berkeley CRAZY! RIOTERS CHASE And Beat People...
9064  #Berkeley IRONY ALERT! ANARCHISTS LOOT STARBUC...
9067  #BlackLivesMatter Supporters Say No Connection...

                                                  text
9058  Hillary may find out she needs more than black...
9060  Commies carrying flags fought with the Austin ...
9062  Is anyone else thinking what we re thinking ab...
9064   Smashing windows How progressive!Protests aga...
9067  THE COMMON THREAD IN ALL THIS IS HATE   THE HA...
```

# Data preprocessing

- We tried to make date ranges classify the news according to their date.
- After removing the duplicates from ‘text’ and ‘title’ columns we had 34763 rows.
- We first get the range of the data (1066) then classified the ranges into 16 classes (2^n > 34763/1066 --> n = 16)
- Hence some dates were not valid, we made a function to clean the date values.

In [20]:
```
min_date = df['date'].min()
max_date = df['date'].max()

print(min_date)
print(max_date)
```

```
2015-03-31 00:00:00
2018-02-19 00:00:00
```

In [21]:
```
date_range = max_date - min_date
date_range
```

Out[21]: `Timedelta('1056 days 00:00:00')`

We can classify the date to 16 time class and each class has 66 values.

In [22]:
```
date_ranges = []
curr_start_date = min_date
range_duration = pd.Timedelta(days=66)

while curr_start_date < max_date:
    curr_end_date = curr_start_date + range_duration
    if curr_end_date > max_date:
      curr_end_date = max_date
    date_ranges.append((curr_start_date, curr_end_date))
    curr_start_date = curr_end_date

date_ranges
```

```
def clean_date(date):
    try:
        return pd.to_datetime(date, format='%d-%b-%y')
    except (ValueError, TypeError):
        return np.nan
```

# Data preprocessing

- Last step, we tokenized the text data by downloading stopwords from nltk library.
- The model won’t understand the text data then we had to tokenize it to be able to understand the data.

In [26]:
```
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'<.*?>', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stopwords.words('english')]
    ps = PorterStemmer()
    tokens = [ps.stem(word) for word in tokens]
    return ' '.join(tokens)
```

In [27]:
```
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[27]: True

In [28]:
```
df['title'] = df['title'].apply(preprocess_text)
df['text'] = df['text'].apply(preprocess_text)
```

# Data preprocessing

```
preprocessor = ColumnTransformer(
    transformers=[
        ('text', TfidfVectorizer(max_features=5000, ngram_range=(1, 2)), 'combined_title_and_text'),
        ('others', OneHotEncoder(handle_unknown='ignore'), ['subject', 'date_range'])
    ],
    remainder='passthrough'
)
```

- To Convert a collection of raw documents (text) into a matrix of TF-IDF features
- Converts categorical variables into a form that could be provided to ML algorithms to do a better job in prediction.

# Failed attempts

- First failed attempt was to classify the data according to title to continents. We tried 3 different attempts but each decreased the accuracy of the model.

```python
import spacy    Import "spacy" could not be resolved
import pycountry    Import "pycountry" could not be resolved

nlp = spacy.load("en_core_web_sm")

continents = {
    "Africa": ['DZ', 'AO', 'BJ', 'BW', 'BF', 'BI', 'CM', 'CV', 'CF', 'TD', 'KM', 'CD', 'CG', 'CI', 'DJ', 'EG', 'GQ', 'ER', 'SZ', 'ET', 'GA', 'GM', 'GH', 'GN', 'GW', 'KE', 'LS', 'LR'
    "Asia": ['AF', 'AM', 'AZ', 'BH', 'BD', 'BT', 'BN', 'KH', 'CN', 'CY', 'GE', 'IN', 'ID', 'IR', 'IQ', 'IL', 'JP', 'JO', 'KZ', 'KW', 'KG', 'LA', 'LB', 'MY', 'MV', 'MN', 'MM', 'NP',
    "Europe": ['AL', 'AD', 'AM', 'AT', 'AZ', 'BY', 'BE', 'BA', 'BG', 'HR', 'CY', 'CZ', 'DK', 'EE', 'FI', 'FR', 'GE', 'DE', 'GR', 'HU', 'IS', 'IE', 'IT', 'KZ', 'XK', 'LV', 'LI', 'LT'
    "North America": ['AG', 'BS', 'BB', 'BZ', 'CA', 'CR', 'CU', 'DM', 'DO', 'SV', 'GD', 'GT', 'HT', 'HN', 'JM', 'MX', 'NI', 'PA', 'KN', 'LC', 'VC', 'TT', 'US'],
    "Oceania": ['AU', 'FJ', 'KI', 'MH', 'FM', 'NR', 'NZ', 'PW', 'PG', 'WS', 'SB', 'TO', 'TV', 'VU'],
    "South America": ['AR', 'BO', 'BR', 'CL', 'CO', 'EC', 'GY', 'PY', 'PE', 'SR', 'UY', 'VE']
}

country_to_continent = {country: continent for continent, countries in continents.items() for country in countries}

def get_country_code(country_name):
    try:
        return pycountry.countries.lookup(country_name).alpha_2
    except LookupError:
        return None

def assign_continent(title):
    doc = nlp(title)
    for ent in doc.ents:
        if ent.label_ == "GPE":
            country_code = get_country_code(ent.text)
            if country_code:
                continent = country_to_continent.get(country_code, 'WorldNews')
                return continent
    return 'WorldNews'
```

Python

- Second one was to use the title length and text length as features but they didn’t effectively work. They reduced the model’s accuracy too.

```python
df['title_length'] = df['title'].apply(lambda x: len(x.split()))
df['text_length'] = df['text'].apply(lambda x: len(x.split()))
```

# Best Accuracy Overview

## Random Forest Classifier

```
Accuracy:  0.998274374460742
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1610
         1.0       1.00      1.00      1.00      1867

    accuracy                           1.00      3477
   macro avg       1.00      1.00      1.00      3477
weighted avg       1.00      1.00      1.00      3477

Cross-Validation Scores: [0.99808245 0.99824197 0.99664376 0.99648394 0.99808215]
Mean CV Score: 0.9975068536853298
```

**99.82%**

## XGBClassifier

```
Accuracy:  0.997411561691113
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1610
         1.0       1.00      1.00      1.00      1867

    accuracy                           1.00      3477
   macro avg       1.00      1.00      1.00      3477
weighted avg       1.00      1.00      1.00      3477

Cross-Validation Scores: [0.99808245 0.99744286 0.99680358 0.99680358 0.99808215]
Mean CV Score: 0.997442925285138
```

**99.74%**

**99.62%**

```
Accuracy:  0.9962611446649411
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1610
         1.0       1.00      1.00      1.00      1867

    accuracy                           1.00      3477
   macro avg       1.00      1.00      1.00      3477
weighted avg       1.00      1.00      1.00      3477

Cross-Validation Scores: [0.99744327 0.99760268 0.99600447 0.99632412 0.99680358]
Mean CV Score: 0.9968356259142525
```

## GradientBoostingClassifier

# Hyperparameters

```
model_rf = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=1000, class_weight='balanced'))
])

model_gb = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', GradientBoostingClassifier(n_estimators=1000))
])

model_xgb = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', xgb.XGBClassifier())
])
```

Estimators:

- **Sets the number of decision trees in the forest to 1000.**
- **Sets the number of boosting stages (decision trees) to 1000.**

Class Weight:

- **Adjusts weights inversely proportional to class frequencies to handle class imbalance.**

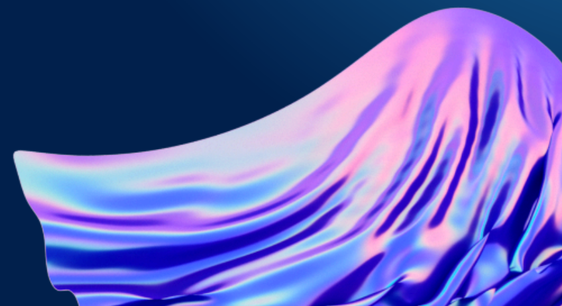
# Test with Best Algorithms

```
Random Forest:
Class 0 count: 2200
Class 1 count: 2296

Gradient Boosting:
Class 0 count: 2212
Class 1 count: 2284

XGBoost:
Class 0 count: 2162
Class 1 count: 2334
```

**99.82%**

**99.62%**

**99.74%**



# We tried , but we got less accuracy

## Logistic Regression

```
Accuracy:  0.9910842680471671
Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99      1610
         1.0       0.99      0.99      0.99      1867

    accuracy                           0.99      3477
   macro avg       0.99      0.99      0.99      3477
weighted avg       0.99      0.99      0.99      3477

Cross-Validation Scores: [0.99025248 0.99120984 0.99168931 0.99152949 0.99216877]
Mean CV Score: 0.99136997754589
```

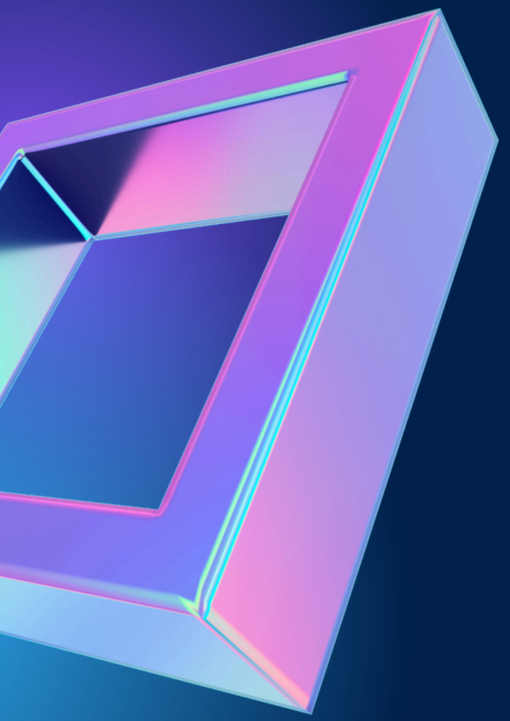
## SVC

```
Accuracy:  0.995973540408398
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      0.99      1.00      1610
         1.0       1.00      1.00      1.00      1867

    accuracy                           1.00      3477
   macro avg       1.00      1.00      1.00      3477
weighted avg       1.00      1.00      1.00      3477

Cross-Validation Scores: [0.99568552 0.99472591 0.99536519 0.99536519 0.99664376]
Mean CV Score: 0.995557114095492
```

# Evaluation Metrics:

```
acc=accuracy_score(y_test, y_pred)
CR=classification_report(y_test, y_pred)
cv_scores = cross_val_score(model, X_train, y_train, cv=5)
MeanCV=np.mean(cv_scores)
```

- **Accuracy Score: Provides a simple measure of how often the model predictions match the actual outcomes.**
- **Classification Report: Gives a detailed report including precision, recall, F1-score, and support for each class.**
- **Cross-Validation: Estimates the model's accuracy and generalizability by training and evaluating it multiple times on different subsets of the data.**
- **Mean Cross-Validation Score: Aggregates the cross-validation scores to provide a single measure of model performance across different subsets of the data.**



# Thank You