```
from google.colab import drive
drive.mount('/content/drive')

Output: Mounted at /content/drive

import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
regr = linear_model.LinearRegression()
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
from sklearn.model_selection import cross_val_score
from sklearn.metrics import r2_score
#we have to upload cyberbullying_tweets file in sample_data
file=pd.read_csv("/content/sample_data/cyberbullying_tweets.csv")
data=file

Output: File will be uploaded in the sample_data from the drive

data.head(10)

Output:
```

	tweet_text	cyberbullying_type
0	In other words #katandandre, your food was cra...	not_cyberbullying
1	Why is #aussietv so white? #MKR #theblock #ImA...	not_cyberbullying
2	@XochitlSuckkks a classy whore? Or more red ve...	not_cyberbullying
3	@Jason_Gio meh. :P thanks for the heads up, b...	not_cyberbullying
4	@RudhoeEnglish This is an ISIS account pretend...	not_cyberbullying
5	@Raja5aab @Quickieleaks Yes, the test of god i...	not_cyberbullying
6	Itu sekolah ya bukan tempat bully! Ga jauh kay...	not_cyberbullying
7	Karma. I hope it bites Kat on the butt. She is...	not_cyberbullying
8	@stockputout everything but mostly my priest	not_cyberbullying
9	Rebecca Black Drops Out of School Due to Bully...	not_cyberbullying

```
data.info()
```

```
Output: <class 'pandas.core.frame.DataFrame'>

RangeIndex: 15289 entries, 0 to 15288
```

```
Data columns (total 2 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   tweet_text          15289 non-null  object
 1   cyberbullying_type  15289 non-null  object
dtypes: object(2)
memory usage: 239.0+ KB
```

```
data.describe()

Output:
```

tweet_text	cyberbullying_type	
count	15289	15289
unique	15267	3
top	But you all respect him....Pete hasn't read tw...	not_cyberbullying
freq	2	7945

```
data.isnull()

Output:
```

tweet_text	cyberbullying_type	
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
15284	False	False
15285	False	False
15286	False	False
15287	False	False
15288	False	False

15289 rows × 2 columns

```
result=data.dropna()
```

```
print(result)

Output:                                                         tweet_text
cyberbullying_type

0      In other words #katandandre, your food was cra...
not_cyberbullying
1      Why is #aussietv so white? #MKR #theblock #ImA...
not_cyberbullying
2      @XochitlSuckkks a classy whore? Or more red ve...
not_cyberbullying
3      @Jason_Gio meh. :P  thanks for the heads up, b...
not_cyberbullying
4      @RudhoeEnglish This is an ISIS account pretend...
not_cyberbullying
...                                                  ...
...
15284  Black ppl aren't expected to do anything, depe...
ethnicity
15285  Turner did not withhold his disappointment. Tu...
ethnicity
15286  I swear to God. This dumb nigger bitch. I have...
ethnicity
15287  Yea fuck you RT @therealexel: IF YOURE A NIGGE...
ethnicity
15288  Bro. U gotta chill RT @CHILLShrammy: Dog FUCK ...
ethnicity

[15289 rows x 2 columns]

count=data.isna().sum()

print(count)
Output:
tweet_text           0
cyberbullying_type   0
dtype: int64

data.nunique()

Output:
tweet_text       15267
cyberbullying_type   3
dtype: int64

X = data.drop('cyberbullying_type',axis=1)
Y = data['cyberbullying_type']

X.shape , y.shape

Output: ((15289, 1), (15289,))

data.corr()
```

```
stop_words = stopwords.words("english")
stemmer = SnowballStemmer("english")

def preprocess(text, stem=False):
    # Remove link,user and special characters
    text = re.sub(TEXT_CLEANING_RE, ' ', str(text).lower()).strip()
    tokens = []
    for token in text.split():
        if token not in stop_words:
            if stem:
                tokens.append(stemmer.stem(token))
            else:
                tokens.append(token)
    return " ".join(tokens)

%%time
df.text = df.text.apply(lambda x: preprocess(x))
```

```
 Output : CPU times: user 58.4 s, sys: 172 ms, total: 58.5 s
Wall time: 58.6 s
```

```
from sklearn.model_selection import train_test_split
X_train , X_test , Y_train , Y_test =
train_test_split(X,y,random_state=101,test_size=0.2)
X_train.shape , X_test.shape , Y_train.shape , Y_test.shape

Output: ((12231, 1), (3058, 1), (12231,), (3058,))

x = data['tweet_text'].values
y = data['cyberbullying_type'].values

x_train, x_test, y_train, y_test = train_test_split(x,y,train_size =
0.8, test_size=0.2, random_state=21)
x_train = x_train.reshape(-1,1)
x_test = x_test.reshape(-1,1)
x_train

#transformation of words using count vectorizer

from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(stop_words = 'english', lowercase = True)

train_data = cv.fit_transform(X_train)

 test_data = cv.transform(X_test)

#tuning hyper parameters
```

```
from sklearn.metrics import f1_score, accuracy_score, precision_score,
recall_score, make_scorer
from sklearn.model_selection import train_test_split, GridSearchCV
def param_tuning(clf, param_dict, X_train, y_train, X_test, y_test):

 # make scorer object
    scorer = make_scorer(f1_score)

    # perform Grid Search for Parameters
    grid_obj = GridSearchCV(estimator = clf,
                            param_grid = param_dict,
                            scoring = scorer,
                            cv = 5)

    grid_fit = grid_obj.fit(X_train, y_train)

    # Get the estimator
    best_clf = grid_fit.best_estimator_

    # Make predictions using the unoptimized and model
    predictions = (clf.fit(X_train, y_train)).predict(X_test)
    best_predictions = best_clf.predict(X_test)

    # Report the before-and-afterscores
    print(clf.__class__.__name__)
    print("\nOptimized Model\n------")
    print("Best Parameters: {}".format(grid_fit.best_params_))
    print("Accuracy:{:.4f}".format(accuracy_score(y_test,
best_predictions)))
    print("F1-score: {:.4f}".format(f1_score(y_test,
best_predictions)))
    print("Precision: {:.4f}".format(precision_score(y_test,
best_predictions)))
    print("Recall: {:.4f}".format(recall_score(y_test,
best_predictions)))


#Logistic Regression Classifier
```

```
# Dict for parameters
param_grid = {
    'C': [1, 1.2, 1.3, 1.4]
}

clf_lr = LogisticRegression()

param_tuning(clf_lr, param_grid, train_data, y_train, test_data, y_test)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432
: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
```

```
  FutureWarning)

Output:
LogisticRegression

Optimized Model
------
Best Parameters: {'C': 1.2}
Accuracy: 0.9270
F1-score: 0.9433
Precision: 0.9638
Recall: 0.9236

#Decision Tree Classifier

param_grid = {
    'min_samples_split': [2, 5, 8],
    'min_samples_leaf': [1, 2, 5, 8]
}

clf_dt = DecisionTreeClassifier()

param_tuning(clf_dt, param_grid, training_data, y_train, testing_data,
y_test)

Output:

DecisionTreeClassifier

Optimized Model
------
Best Parameters: {'min_samples_leaf': 1, 'min_samples_split': 5}
Accuracy: 0.9244
F1-score: 0.9417
Precision: 0.9554
Recall: 0.9284

#Random Forest Classifier

param_grid = {
    'n_estimators': [50,150],
    'min_samples_leaf': [1, 5],
    'min_samples_split': [2, 5]
}

clf_rf = RandomForestClassifier()

param_tuning(clf_rf, param_grid, training_data, y_train, testing_data,
y_test)
/opt/conda/lib/python3.6/site-packages/sklearn/ensemble/forest.py:245: Futu
reWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

Output:

RandomForestClassifier
```

```
Optimized Model
------
Best Parameters: {'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimat
ors': 150}
Accuracy: 0.9157
F1-score: 0.9351
Precision: 0.9462
Recall: 0.9243
```

#Multinomial Naïve Bayes Classification

```
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split, GridSearchCVtext_clf =
Pipeline([('vect', CountVectorizer()),
                     ('tfidf', TfidfTransformer()),
                     ('clf', MultinomialNB())])tuned_parameters = {
    'vect__ngram_range': [(1, 1), (1, 2), (2, 2)],
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'clf__alpha': [1, 1e-1, 1e-2]
}
```

```
from sklearn.metrics import classification_reportclf = GridSearchCV(text_clf,
tuned_parameters, cv=10, scoring=score)
clf.fit(x_train, y_train)

print(classification_report(y_test, clf.predict(x_test), digits=4))
```

Output:

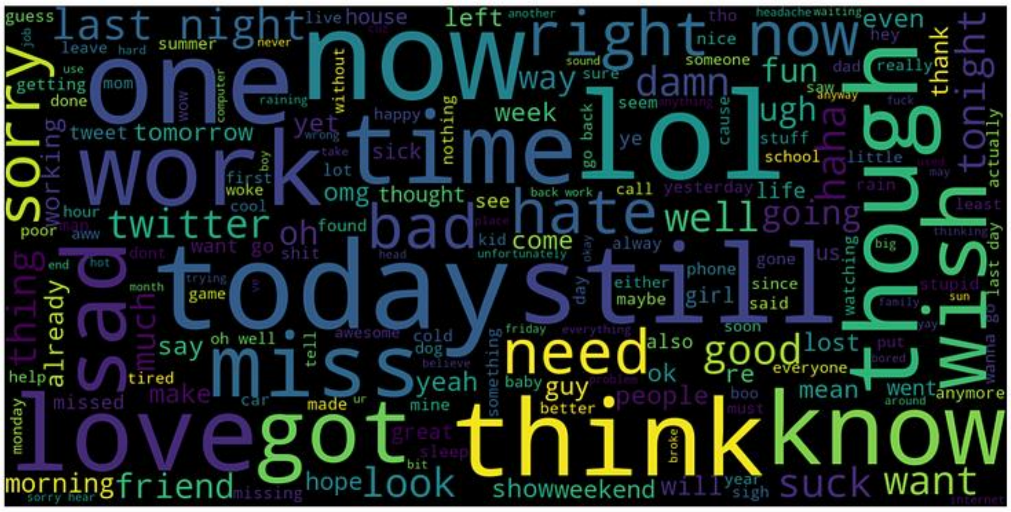
```
Multinomial Naïve Bayes Classification

Optimized Model
------
Accuracy: 0.7941
F1-score: 0.7560
Precision: 0.7577
Recall: 0.7759
```

```
neg_tweets = my_df[my_df.target == 0]
neg_string = []
for t in neg_tweets.text:
    neg_string.append(t)
neg_string = pd.Series(neg_string).str.cat(sep=' ')from
wordcloud import WordCloud

wordcloud = WordCloud(width=1600,
height=800,max_font_size=200).generate(neg_string)
plt.figure(figsize=(12,10))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()

Output:
```



```
#conclusion
Accuracy of Logistic Regression: 0.9270
Accuracy of Random Forest Classification: 0.9244
Accuracy of Decision Tree Classification: 0.9157
Accuracy of Multinomial Naïve Bayes Classification: 0.7941

So, Logistic regression is the most accurate algorithm of determining
cyberbullying status for the given dataset.
```

PREPARED BY:
EGA SHAIVI