importing modules

```
import pandas  as pd
import requests as req
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import ParameterGrid
from sklearn.preprocessing import LabelEncoder
import re
```

*****************************PROBLEM 1 – Reading the data*************************

urls to the datasets

```
positive_baits="https://raw.githubusercontent.com/pfrcks/clickbait-
detection/master/clickbait"
negative_baits="https://raw.githubusercontent.com/pfrcks/clickbait-
detection/master/not-clickbait"
```

A function to fetch the data from a URL

```

def my_read_function(url,label):
    response=req.get(url)
    text = response.text
    lines = text.split('\n')
    df=pd.DataFrame({'baits': lines})
    df.index.name = 'Index'
    df['Label'] = label
    #df.to_csv('baits_data.csv', index=False)
    return df
```

Read the positive and negative datasets

```
positive_dataset = my_read_function(positive_baits,'clickbait')
negative_dataset = my_read_function(negative_baits,'not-clickbait')


#reading from  the positive_dataset/click_baits_dataset
df=pd.DataFrame(positive_dataset)
df


                                                   baits      Label
Index
0      Man repairs fence to contain dog, hilarity ens...  clickbait
1      Long-Term Marijuana Use Has One Crazy Side Eff...  clickbait
2      The water from his ear trickles into the bucke...  clickbait
3      You'll Never Guess What Nick Jonas Does in the...  clickbait
4      How Cruise Liners Fill All Their Unsold Cruise...  clickbait
...                                                  ...        ...
810    OITNB's Taylor Schilling and Carrie Brownstein...  clickbait
811    Researchers have discovered the average penis ...  clickbait
812    Why it may be smart to wait to put on sunscree...  clickbait
813    What state has highest rate of rape in the cou...  clickbait
814                                                       clickbait

[815 rows x 2 columns]


#reading from  the positive_dataset/click_baits_dataset
df=pd.DataFrame(negative_dataset)
df

                                                   baits
Label
Index

0      Congress Slips CISA Into a Budget Bill That's ...  not-
clickbait
1                              DUI Arrest Sparks Controversy  not-
clickbait
2      It’s unconstitutional to ban the homeless from...  not-
clickbait
3      A Government Error Just Revealed Snowden Was t...  not-
clickbait
4      A toddler got meningitis. His anti-vac parents...  not-
clickbait
...                                                  ...            ..
.
1570   Loophole means ecstasy and loads of other drug...  not-
clickbait
```

```
1571        Astronomers Watch a Supernova and See Reruns  not-
clickbait
1572   In Indian Rapists’ Neighborhood, Smoldering An...  not-
clickbait
1573                       Strong earthquake jolts Islamabad  not-
clickbait
1574                                                          not-
clickbait

[1575 rows x 2 columns]
```

Combining the datasets

```
combined_dataset = pd.concat([positive_dataset, negative_dataset], 
ignore_index=True)
df=pd.DataFrame(combined_dataset)

df.to_csv("combined.csv",index=False) #




#reading the first 8 rows of the combined dataset
df.head(8)
```

```
                                                baits      Label
0  Man repairs fence to contain dog, hilarity ens...  clickbait
1  Long-Term Marijuana Use Has One Crazy Side Eff...  clickbait
2  The water from his ear trickles into the bucke...  clickbait
3  You'll Never Guess What Nick Jonas Does in the...  clickbait
4  How Cruise Liners Fill All Their Unsold Cruise...  clickbait
5                 Could Queen Elizabeth Veto Brexit?  clickbait
6      This Is the Worst Color to Paint Your Kitchen  clickbait
7                     The Shocking Truth About Sugar  clickbait
```

Shuffling the combined dataset using numpy

```
# Create an array of indices from 0 to the length of the combined 
dataset
shuffled_indices = np.arange(len(combined_dataset))
# Shuffle the array of indices randomly using numpy's random.shuffle
```

```
function
np.random.shuffle(shuffled_indices)
# Use the shuffled indices to rearrange the rows of the combined
dataset, creating the shuffled dataset
shuffled_dataset = combined_dataset.iloc[shuffled_indices]
df2=pd.DataFrame(shuffled_dataset)
df2.to_csv("shuffled.csv",index=False)




#reading the first ten rows of the shuffled_dataset
df=pd.DataFrame(shuffled_dataset)
df.head(10)


                                                  baits          Label
356   You Won’t Believe What Material These Sunglass...      clickbait
1671   Lenovo caught installing adware on new computers  not-clickbait
60    How to tell if someone is a narcissist with on...      clickbait
1339  Disney Donates $1 Million to Orlando Shooting ...  not-clickbait
1568  Who's minding the marijuana? Banned pesticide ...  not-clickbait
570   Why did Croatia fans disrupt their Euro 2016 m...      clickbait
1214  Don't laugh: Google's Parsey McParseface is a...  not-clickbait
309   Paul McCartney Reveals What Really Split Up th...      clickbait
2113  Gay Chinese man sues mental hospital for tryin...  not-clickbait
538   Abercrombie & Fitch is facing a terrifying rea...      clickbait
```

Split the shuffled dataset into train, validation, and test sets

```


train_data_percentage = 0.72  # 72% for training
validation_data_percentage = 0.08  # 8% for validation
test_data_percentage = 0.20  # 20% for testing




# Calculate the number of samples for each split
total_samples = len(shuffled_dataset) #number of rows in the combined
dataset
train_data_samples = int(train_data_percentage * total_samples)
#number of rows in the training dataset
validation_data_samples = int(validation_data_percentage *
total_samples) #number of rows in the validation dataset
```

```
test_data_samples = total_samples - train_data_samples -
validation_data_samples #number of rows in the testing dataset




#printing the number of rows/ samples in each dataset
print("********************* OUTPUT ********************\n")
print(f"total sample or the number of rows ={total_samples}
samples/rows")
print(f"validation samples or the number of rows
={validation_data_samples} samples/rows")
print(f"Training samples or the number of rows ={train_data_samples}
samples/rows")
print(f"test samples or the number of rows ={test_data_samples}
samples/rows")


********************* OUTPUT ********************

total sample or the number of rows =2390 samples/rows
validation samples or the number of rows =191 samples/rows
Training samples or the number of rows =1720 samples/rows
test samples or the number of rows =479 samples/rows



# Spliting the dataset into training and the remaining data
(remaining_data)
training_data, remaining_data = train_test_split(shuffled_dataset,
test_size=(1 - train_data_percentage))

# Split the remaining data into validation and test sets
validation_data, test_data = train_test_split(remaining_data,
test_size=test_data_percentage / (test_data_percentage +
validation_data_percentage))
#saving the datasets
train_dataset=pd.DataFrame(training_data) #training dataset
train_dataset.to_csv("traning_data.csv",index=False)

validating_dataset=pd.DataFrame(validation_data) #validating  set
validating_dataset.to_csv("validating_data.csv",index=False)

testing_dataset=pd.DataFrame(test_data) #testing set
testing_dataset.to_csv('testing_dataset.csv',index=False)





#reading from each and every dataset after split
```

```
train_dataset #reading from the training dataset
```

```
                                                  baits          Label
1187  It is rare for a new animal species to emerge ...  not-clickbait
1701  Ontario parents who object to vaccines could b...  not-clickbait
1673  Panama Papers: British Banker Funded North Kor...  not-clickbait
1781           Burnt: 220 hives containing 250,000 bees  not-clickbait
2014  Artificial intelligence: ‘Homo sapiens will be...  not-clickbait
...                                                 ...            ...
436                      The ‘Right’ Age to Get Married      clickbait
2386       Astronomers Watch a Supernova and See Reruns  not-clickbait
2324  German spy agency says ISIS sending fighters d...  not-clickbait
992   The disappeared: Chicago police detain America...  not-clickbait
1702  Pregnant women warned not to travel to Rio Oly...  not-clickbait

[1720 rows x 2 columns]
```

```
#reading from the testing dataset
testing_dataset
```

```
                                                  baits          Label
1873         Only 3 northern white rhinos left on Earth  not-clickbait
701   Trump's campaign cycles $6 million into Trump ...      clickbait
2366  Top UN Official Says 'Global War on Terror' Is...  not-clickbait
1698  Kerry calls for democracy in Cuba as U.S. flag...  not-clickbait
1032  Knife turned into police allegedly found on O....  not-clickbait
...                                                 ...            ...
1456  Proposed California Ballot Initiative That Wou...  not-clickbait
22     Here's what those floaty things in your eyes are      clickbait
2168  FIFA scandal: Sepp Blatter wins another term a...  not-clickbait
655                 A Waterfall in the Middle of a Lake      clickbait
2131  Ikea vows to be net exporter of renewable ener...  not-clickbait

[479 rows x 2 columns]
```

```
#reading from the validation  dataset
validating_dataset
```

```
                                                  baits          Label
382   Why Has Mark Zuckerberg Put Tape Over His Camera?      clickbait
628   Dad And Son Are Seconds From Assassination By ...      clickbait
1125  Martin Shkreli fired as CEO of KaloBios Pharma...  not-clickbait
1935  Former Brazilian soccer star: Don't come to th...  not-clickbait
363   Guess how much Google paid the guy who briefly...      clickbait
```

```
...                                                       ...            ...
2248  Stop refugees or we'll stop aid, Germany tells...  not-clickbait
574   Guess Who’s Complaining About Obama’s New Over...      clickbait
2042  Saudi Arabia insists UN keeps LGBT rights out ...  not-clickbait
1321  Napolitano Says ‘We Don’t Have To Listen To Th...  not-clickbait
1123  Vladimir Putin says the Panama Papers are part...  not-clickbait

[191 rows x 2 columns]
```

Calculating the "target rate" for each dataset (training,validation and test)

```
train_data_target_rate = (train_dataset['Label'] ==
'clickbait').mean()
validation_data_target_rate = (validating_dataset['Label'] ==
'clickbait').mean()
test_data_target_rate = (testing_dataset['Label'] ==
'clickbait').mean()
```

what % of the three datasets is t is labeled as clickbait?

```
print("********************* OUTPUT ********************\n")
print(f"{train_data_target_rate.round(4)*100}% of the training data is
labeled as clickbait")
print(f"{validation_data_target_rate.round(4)*100}% of the validating
data is labeled as clickbait")
print(f"{test_data_target_rate.round(4)*100}% of the testing data is
labeled as clickbait")
```

```
********************* OUTPUT ********************

34.36% of the training data is labeled as clickbait
35.08% of the validating data is labeled as clickbait
32.78% of the testing data is labeled as clickbait
```

********************** PROBLEM 3 – Training a single Bag-of-Words (BOW) Text Classifier ***************

```
# Loading the training and validation datasets
validating_dataset  #the dataset containing the validating data
train_dataset #the dataset containing the traingin data


# Map labels to binary values (1 for clickbait, 0 for non-clickbait)
train_dataset['Label'] = (train_dataset['Label'] ==
'clickbait').astype(int)
validating_dataset['Label'] = (validating_dataset['Label'] ==
'clickbait').astype(int)



# Creating  a Pipeline with CountVectorizer and MultinomialNB
pipeline = Pipeline([
    ('vectorizer', CountVectorizer(ngram_range=(1, 2))),
    ('classifier', MultinomialNB())
])


# Fiting  the classifier on the training dataset
pipeline.fit(train_dataset['baits'], train_dataset['Label'])


# Predict on the training and validation datasets
train_predictions = pipeline.predict(train_dataset['baits'])
validation_predictions = pipeline.predict(validating_dataset['baits'])



# Computing  precision, recall, and F1-score for training and
validation datasets with zero_division='warn'
train_precision = precision_score(train_dataset['Label'],
train_predictions, pos_label=1, zero_division='warn')
train_recall = recall_score(train_dataset['Label'], train_predictions,
pos_label=1, zero_division='warn')
train_f1 = f1_score(train_dataset['Label'], train_predictions,
pos_label=1, zero_division='warn')

validation_precision = precision_score(validating_dataset['Label'],
validation_predictions, pos_label=1, zero_division='warn')
validation_recall = recall_score(validating_dataset['Label'],
validation_predictions, pos_label=1, zero_division='warn')
validation_f1 = f1_score(validating_dataset['Label'],
validation_predictions, pos_label=1, zero_division='warn')




# Print the results
print("********************* OUTPUT ********************\n")
print(f"Training Precision is  {train_precision}  or
{train_precision.round(4)*100}%")
print(f"Training Recall is {train_recall}  or
{train_recall.round(4)*100}%")
print(f"Training F1-score is  {train_f1} or
{train_recall.round(4)*100}%")
print("\n")
print(f"Validation Precision is {validation_precision} or
{validation_precision.round(4)*100}% ")
print(f"Validation Recall is  {validation_recall} or
{validation_precision.round(4)*100}%")
print(f"Validation F1-score is { validation_f1} or
{validation_f1.round(4)*100}%")
```

```
********************* OUTPUT ********************

Training Precision is  0.9915966386554622  or  99.16%
Training Recall is 0.9983079526226735  or 99.83%
Training F1-score is  0.9949409780775718 or 99.83%


Validation Precision is 0.8615384615384616 or 86.15%
Validation Recall is  0.835820895522388 or 86.15%
Validation F1-score is 0.8484848484848485 or 84.85000000000001%
```

********************** PROBLEM 4 – Hyperparameter Tuning**********************

```
# Mapping  labels to binary values (1 for clickbait, 0 for non-
clickbait)
```

```
train_dataset['Label'] = (train_dataset['Label'] ==
'clickbait').astype(int)
validating_dataset['Label'] = (validating_dataset['Label'] ==
'clickbait').astype(int)


# Defining a grid of hyperparameters to search
parameter_grid = {
    'vectorizer__max_df': [0.5, 0.75, 1.0],  # Vary max_df for
CountVectorizer
    'classifier__alpha': [1.0, 0.5, 0.1],     # Vary smoothing for
MultinomialNB
    'vectorizer__ngram_range': [(1, 1), (1, 2)]  # Include or exclude
bigrams in CountVectorizer
}


# Initializing  an empty  list to store the results
results = []


# Iterating  over the parameter grid
for params in ParameterGrid(parameter_grid):
    # Creating  a Pipeline with CountVectorizer and MultinomialNB with
the current parameters
    pipeline = Pipeline([
        ('vectorizer',
CountVectorizer(max_df=params['vectorizer__max_df'],
ngram_range=params['vectorizer__ngram_range'])),
        ('classifier',
MultinomialNB(alpha=params['classifier__alpha']))
    ])


    # Fiting  the classifier on the training dataset
    pipeline.fit(train_dataset['baits'], train_dataset['Label'])


    # Predicting  on the validation dataset
    validation_predictions =
pipeline.predict(validating_dataset['baits'])


    # Computing  precision, recall, and F1-score for validation
dataset
    validation_precision =
precision_score(validating_dataset['Label'],
validation_predictions,zero_division=1)
    validation_recall = recall_score(validating_dataset['Label'],
validation_predictions,zero_division=1)
```

```
    validation_f1 = f1_score(validating_dataset['Label'],
validation_predictions,zero_division=1)


    # Storing  the results
    results.append({
        'params': params,
        'validation_precision': validation_precision,
        'validation_recall': validation_recall,
        'validation_f1': validation_f1
    })

# Converting  results to a DataFrame for analysis
results_dataframe = pd.DataFrame(results)

# Sorting  the results by F1-score in descending order
results_dataframe = results_dataframe.sort_values(by='validation_f1',
ascending=False)

# Displaying  the top and bottom results
print("********************* OUTPUT ********************\n")
print("Top Results")
print(results_dataframe.head())
print("Bottom Results")
print(results_dataframe.tail())
```

```
********************* OUTPUT ********************

Top Results
                                               params
validation_precision  \
0   {'classifier__alpha': 1.0, 'vectorizer__max_df...
1.0
1   {'classifier__alpha': 1.0, 'vectorizer__max_df...
1.0
16  {'classifier__alpha': 0.1, 'vectorizer__max_df...
1.0
15  {'classifier__alpha': 0.1, 'vectorizer__max_df...
1.0
14  {'classifier__alpha': 0.1, 'vectorizer__max_df...
1.0

    validation_recall  validation_f1
0                 1.0            1.0
1                 1.0            1.0
16                1.0            1.0
```

```
15                1.0            1.0
14                1.0            1.0
Bottom Results
                                               params
validation_precision  \
5   {'classifier__alpha': 1.0, 'vectorizer__max_df...
1.0
4   {'classifier__alpha': 1.0, 'vectorizer__max_df...
1.0
3   {'classifier__alpha': 1.0, 'vectorizer__max_df...
1.0
2   {'classifier__alpha': 1.0, 'vectorizer__max_df...
1.0
17  {'classifier__alpha': 0.1, 'vectorizer__max_df...
1.0

    validation_recall  validation_f1
5                 1.0            1.0
4                 1.0            1.0
3                 1.0            1.0
2                 1.0            1.0
17                1.0            1.0
```

*********************************PROBLEM 5 – Model selection ************************

```
#To select the best model from the results of the hyperparameter
tuning,
#i choose the one that achieved the highest F1-score on the validation
set. how do choose.


# Finding  the best model based on validation F1-score values
best_model = max(results, key=lambda x: x['validation_f1'])

# Displaying  the best model's parameters and validation F1-score
value
print("********************* OUTPUT ********************\n")
print("Best Model - Parameters ")
print(best_model['params'])
print("\n")
print(" Validation F1-Score value  of  the Best Model is ",
best_model['validation_f1'])
```

```
********************* OUTPUT ********************

Best Model - Parameters
{'classifier__alpha': 1.0, 'vectorizer__max_df': 0.5,
'vectorizer__ngram_range': (1, 1)}


 Validation F1-Score value  of  the Best Model is  1.0



#applying the  model to my  test set and computing the  precision,
recall, and F1-score values

   # Creating  a pipeline with the best model's parameters
best_model_pipeline = Pipeline([
    ('vectorizer', CountVectorizer(max_df=best_model['params']
['vectorizer__max_df'], ngram_range=best_model['params']
['vectorizer__ngram_range'])),
    ('classifier', MultinomialNB(alpha=best_model['params']
['classifier__alpha']))
])

# Fiting my  best model on the training dataset
best_model_pipeline.fit(train_dataset['baits'],
train_dataset['Label'])

# Predicting  on the test dataset
test_predictions =
best_model_pipeline.predict(testing_dataset['baits'])

#removing posible NAN values
clean_testing_data = testing_dataset.dropna()

# Convert true labels to numeric format
clean_testing_data['Label'] =
clean_testing_data['Label'].map({'clickbait': 1, 'not-clickbait': 0})

# Computing the  precision, recall, and F1-score values  for the test
dataset
test_precision = precision_score(clean_testing_data['Label'],
test_predictions,zero_division=1)
test_recall = recall_score(clean_testing_data['Label'],
test_predictions)
test_f1 = f1_score(clean_testing_data['Label'], test_predictions)




print("********************* OUTPUT ********************\n")
```

```
# Display the test set results
print("These are Test Set Metrics for Best Model ")
print("Precision ", test_precision)
print("Recall ", test_recall)
print("F1-Score " , test_f1)




********************* OUTPUT ********************

These are Test Set Metrics for Best Model
Precision  1.0
Recall  0.0
F1-Score  0.0
```

************************PROBLEM 6 – Key Indicators*********************

```
#Convert the text to lowercase to ensure consistency.
train_dataset['baits'] = train_dataset['baits'].str.lower()



# Defining  a dictionary to map labels to binary values
label_mapping = {'clickbait': 1, 'not-clickbait': 0} # Mapping  labels
to binary values (1 for clickbait, 0 for non-clickbait)



# Using  the map method to update my  'Label' column
train_dataset['Label'] = train_dataset['Label'].map(label_mapping)



# Creating  a CountVectorizer to extract key words
vectorizer = CountVectorizer(ngram_range=(1, 2))
```

```python
# Transforming  the training data into a feature matrix
X_train = vectorizer.fit_transform(train_dataset['baits'])


# Initializing  and training  a Multinomial Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train, train_dataset['Label'])


# Geting  the feature log probabilities for the "clickbait" class
clickbait_log_probs = clf.feature_log_prob_[1]


# Geting the corresponding vocabulary words
feature_names = np.array(vectorizer.get_feature_names_out())


# Creating  a DataFrame to analyze the results
clickbait_dataframe = pd.DataFrame({'Word': feature_names, 'Log
Probability': clickbait_log_probs})


# Sorting  the DataFrame by log probability in descending order
clickbait_dataframe = clickbait_dataframe.sort_values(by='Log
Probability', ascending=False)








# Display the top 5 words with the highest log probability
top_clickbait_words = clickbait_dataframe.head(6)
print("********************* OUTPUT ********************\n")
print("Top 5 Clickbait Indicator Words")
print(top_clickbait_words)
```

```
********************* OUTPUT ********************

Top 5 Clickbait Indicator Words
       Word  Log Probability
16995   the        -4.851182
20060   you        -5.373704
17625    to        -5.462499
17409  this        -5.521340
```

```
9028     is        -5.567860
11905    of        -5.789021
```

************** PROBLEM 7 – Regular expression *************************************

```
# A list of words that are recognized as clickbaits in the previous
Question
top_keywords = top_clickbait_words['Word'].unique()

# Constructing  a regular expression pattern to match any of the top
keywords with word boundaries
pattern = r'\b(?:' + '|'.join(re.escape(keyword) for keyword in
top_keywords) + r')\b'

#A testing message
text = "you will be a billionaire within two days if you do the
following. Follow this steps ."
print("********************* OUTPUT ********************\n")
# Using  re.search function to find matches in the text
if re.search(pattern, text):
    print("Keyword found in the text.")
else:
    print("Keyword not found in the text.")
```

```
********************* OUTPUT ********************

Keyword found in the text.
```

```
#removing any nans

testing_dataset=testing_dataset.dropna()



# Defining  a dictionary to map labels to binary values
label_mapping = {'clickbait': 1, 'not-clickbait': 0} # Mapping  labels
to binary values (1 for clickbait, 0 for non-clickbait)


# Using  the map method to update my  'Label' column
testing_dataset['Label'] = testing_dataset['Label'].map(label_mapping)



# Creating a function  to check if any top keywords are present in the
text from the test data_set
def keyword_classifier(text):
    pattern = r'\b(?:' + '|'.join(re.escape(keyword) for keyword in
top_keywords) + r')\b'
    return bool(re.search(pattern, text))




# Applying  the function to  test_dataset

testing_dataset['Predicted'] =
testing_dataset['baits'].apply(keyword_classifier)



# Calculating the  precision and recall values

precision = precision_score(testing_dataset['Label'],
testing_dataset['Predicted'], zero_division='warn')

recall = recall_score(testing_dataset['Label'],
testing_dataset['Predicted'], zero_division='warn')

f1=f1_score(testing_dataset['Label'], testing_dataset['Predicted'],
zero_division='warn')
```

```
#printing the results
print("********************* OUTPUT ********************\n")
print(f"The Precision of this classifier is {precision} or
{precision.round(4)*100}%")
print(f"The Recall of this classifier is  {recall} or
{recall.round(4)*100}%")
print(f"The Recall of this classifier is  {f1} or
{f1.round(4)*100}%")
```

```
********************* OUTPUT ********************

The Precision of this classifier is 0.32589285714285715 or  32.59%
The Recall of this classifier is  0.46496815286624205 or  46.5%
The Recall of this classifier is  0.3832020997375329 or  38.32%
```