rus-tweets-nlp-text-classification

April 19, 2023

[379]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/covid-19-nlp-text-classification/Corona_NLP_test.csv
/kaggle/input/covid-19-nlp-text-classification/Corona_NLP_train.csv
```

0.0.1 Dataset Link:

https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification

1 Data Loading

[380]:
```
train = pd.read_csv("/kaggle/input/covid-19-nlp-text-classification/
 ↪Corona_NLP_train.csv", encoding= 'latin-1')
```

[381]:
```
test=pd.read_csv('/kaggle/input/covid-19-nlp-text-classification/
 ↪Corona_NLP_test.csv',  encoding= 'latin-1')
```

[382]:
```
train.head()
```

[382]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i…	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha…	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde…	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp…	Positive
4	3803	48755	NaN	16-03-2020	Me, ready to go at supermarket during the #COV…	Extremely Negative

[383]:
```
test.head()
```

[383]:

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	1	44953	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermar…	Extremely Negative
1	2	44954	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Me…	Positive
2	3	44955	NaN	02-03-2020	Find out how you can protect yourself and love…	Extremely Positive
3	4	44956	Chicagoland	02-03-2020	#Panic buying hits #NewYork City as anxious sh…	Negative
4	5	44957	Melbourne, Victoria	03-03-2020	#toiletpaper #dunnypaper #coronavirus #coronav…	Neutral

2 Data Exploration:

2.1 a. Shape of Data

[384]:
```
train.shape
```

[384]: (41157, 6)

[385]:
```
test.shape
```

[385]: (3798, 6)

2.2 b. Size of Data

[386]:
```
train.size
```

[386]: 246942

[387]:
```
test.size
```

[387]: 22788

2.3 c. Attributes

[388]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41157 entries, 0 to 41156
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   UserName       41157 non-null  int64 
 1   ScreenName     41157 non-null  int64 
 2   Location       32567 non-null  object
 3   TweetAt        41157 non-null  object
 4   OriginalTweet  41157 non-null  object
 5   Sentiment      41157 non-null  object
dtypes: int64(2), object(4)
memory usage: 1.9+ MB
```

[389]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3798 entries, 0 to 3797
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   UserName       3798 non-null   int64 
 1   ScreenName     3798 non-null   int64 
 2   Location       2964 non-null   object
 3   TweetAt        3798 non-null   object
 4   OriginalTweet  3798 non-null   object
 5   Sentiment      3798 non-null   object
dtypes: int64(2), object(4)
memory usage: 178.2+ KB
```

2.4 d. Properties

[390]:
```
train.describe()
```

[390]:
```
          UserName    ScreenName
count  41157.000000  41157.000000
mean   24377.000000  69329.000000
std    11881.146851  11881.146851
min     3799.000000  48751.000000
25%    14088.000000  59040.000000
50%    24377.000000  69329.000000
75%    34666.000000  79618.000000
max    44955.000000  89907.000000
```

[391]:
```
test.describe()
```

[391]:
```
          UserName    ScreenName
count  3798.000000   3798.000000
mean   1899.500000  46851.500000
std    1096.532489   1096.532489
min       1.000000  44953.000000
25%     950.250000  45902.250000
50%    1899.500000  46851.500000
75%    2848.750000  47800.750000
max    3798.000000  48750.000000
```

[392]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41157 entries, 0 to 41156
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   UserName       41157 non-null  int64 
 1   ScreenName     41157 non-null  int64 
 2   Location       32567 non-null  object
 3   TweetAt        41157 non-null  object
 4   OriginalTweet  41157 non-null  object
 5   Sentiment      41157 non-null  object
dtypes: int64(2), object(4)
memory usage: 1.9+ MB
```

[393]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3798 entries, 0 to 3797
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
```

```
---  ------          --------------  -----
 0   UserName        3798 non-null   int64
 1   ScreenName      3798 non-null   int64
 2   Location        2964 non-null   object
 3   TweetAt         3798 non-null   object
 4   OriginalTweet   3798 non-null   object
 5   Sentiment       3798 non-null   object
dtypes: int64(2), object(4)
memory usage: 178.2+ KB
```

[394]:
```
train.dtypes
```

[394]:
```
UserName          int64
ScreenName        int64
Location         object
TweetAt          object
OriginalTweet    object
Sentiment        object
dtype: object
```

[395]:
```
test.dtypes
```

[395]:
```
UserName          int64
ScreenName        int64
Location         object
TweetAt          object
OriginalTweet    object
Sentiment        object
dtype: object
```

2.5 e. EDA

[396]:
```
train.columns
```

[396]:
```
Index(['UserName', 'ScreenName', 'Location', 'TweetAt', 'OriginalTweet',
       'Sentiment'],
      dtype='object')
```

[397]:
```
len(train)
```

[397]:
```
41157
```

[398]:
```
test.columns
```

[398]:
```
Index(['UserName', 'ScreenName', 'Location', 'TweetAt', 'OriginalTweet',
       'Sentiment'],
      dtype='object')
```

[399]:
```
len(train)
```

[399]:
```
41157
```

2.6 f. Null Values

[400]:
```
train.isnull().sum()
```

[400]:
```
UserName            0
ScreenName          0
Location         8590
TweetAt             0
OriginalTweet       0
Sentiment           0
dtype: int64
```

[401]:
```
train.isnull().sum()/len(train)*100
```

[401]:
```
UserName          0.000000
ScreenName        0.000000
Location         20.871298
TweetAt           0.000000
OriginalTweet     0.000000
Sentiment         0.000000
dtype: float64
```

[402]:
```
test.isnull().sum()
```

[402]:
```
UserName            0
ScreenName          0
Location          834
TweetAt             0
OriginalTweet       0
Sentiment           0
dtype: int64
```

[403]:
```
test.isnull().sum()/len(test)*100
```

[403]:
```
UserName          0.000000
ScreenName        0.000000
Location         21.958926
TweetAt           0.000000
OriginalTweet     0.000000
Sentiment         0.000000
dtype: float64
```

2.7 g. Unique

[404]:
```
train.nunique()
```

[404]:
```
UserName         41157
ScreenName       41157
Location         12220
TweetAt             30
OriginalTweet    41157
Sentiment            5
dtype: int64
```

[405]:
```
train.nunique().sum()
```

[405]:
```
135726
```

[406]:
```
test.nunique().sum()
```

[406]:
```
13131
```

3 3. Data Pre-processing

3.1 WordCloud

[407]:
```
import matplotlib.pyplot as plt
```

[408]:
```
from wordcloud import WordCloud
```

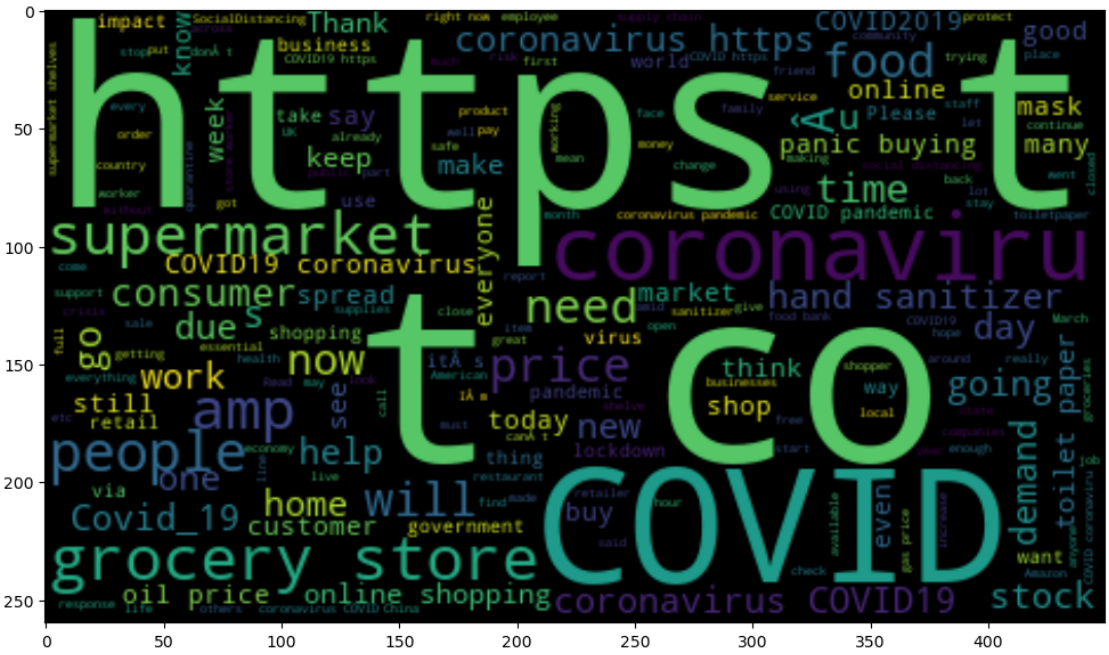
[409]:
```
allWords = ' '.join([twts for twts in train['OriginalTweet']])
```

[410]:
```
allWords[:500]
```

[410]:
```
'@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/iFz9FAn2Pa and
https://t.co/xX6ghGFzCC and https://t.co/I2NlzdxNo8 advice Talk to your
neighbours family to exchange phone numbers create contact list with phone
numbers of neighbours schools employer chemist GP set up online shopping
accounts if poss adequate supplies of regular meds but not over order
Coronavirus Australia: Woolworths to give elderly, disabled dedicated shopping
hours amid COVID-19 outbreak https://t.co/bInCA9Vp8P My food stock is n'
```

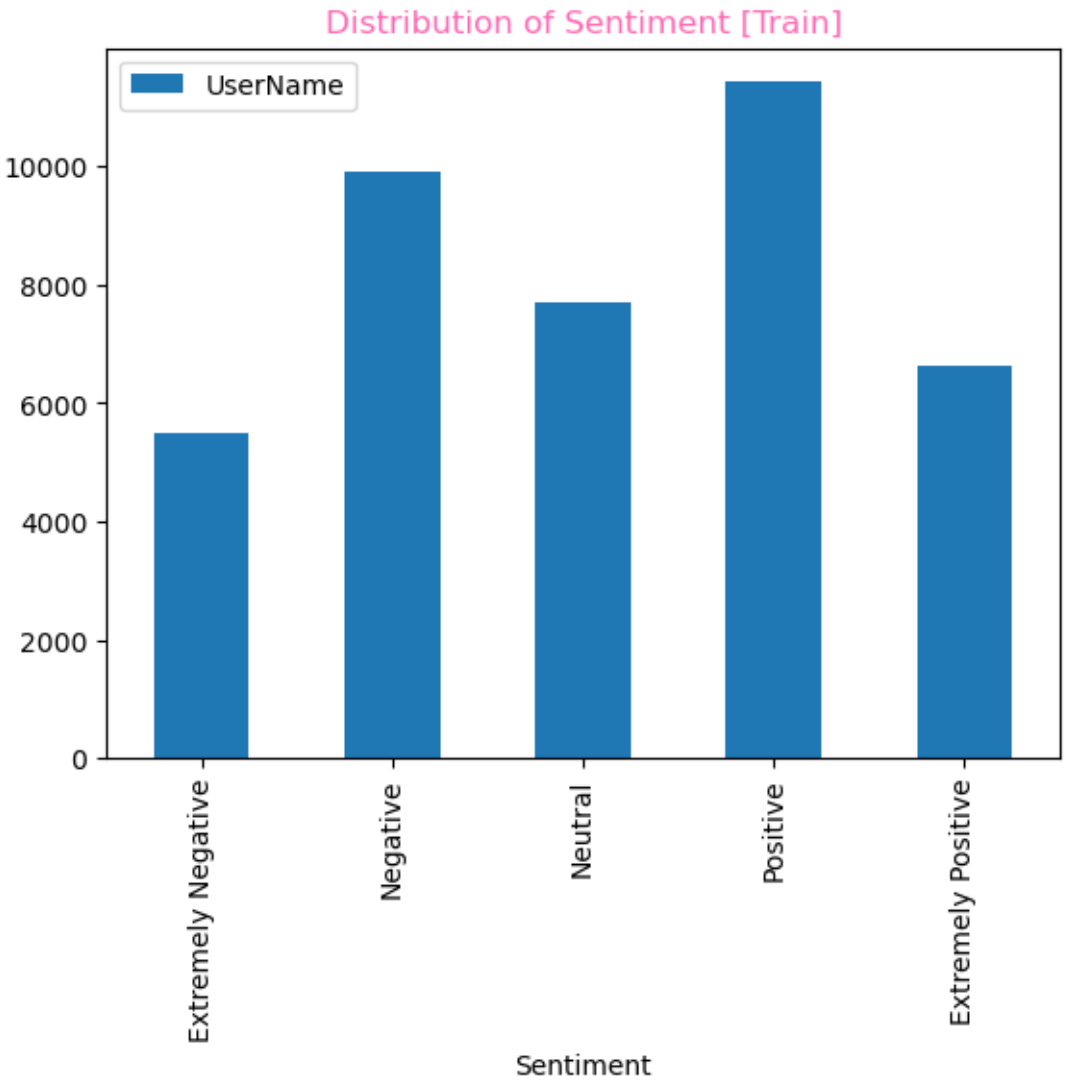
[411]:
```
wordcloud = WordCloud(width=450, height=260, random_state=42,␣
↪max_font_size=105).generate(allWords)
```

[412]:
```
plt.figure(figsize=(12,8))
plt.imshow(wordcloud, interpolation='bilinear', cmap='viridis')
plt.show()
```



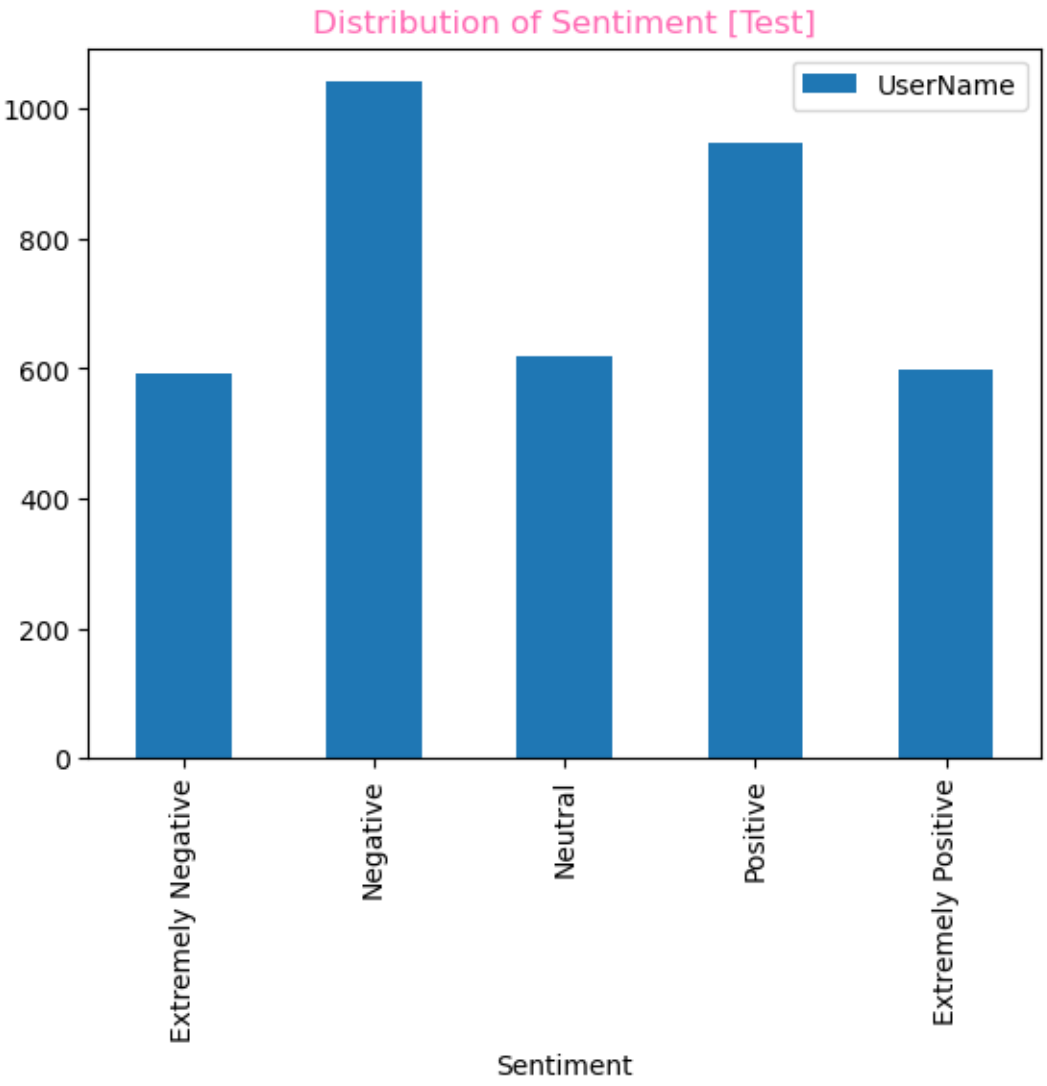
3.1.1 Tweets Sentiment Distribution –> Train

[413]:
```
train.groupby(['Sentiment']) \
    .count()[['UserName']] \
    .reindex(['Extremely Negative',
              'Negative',
              'Neutral',
              'Positive',
              'Extremely Positive']) \
    .plot(kind="bar")
plt.title("Distribution of Sentiment [Train]",color='hotpink')
plt.show()
```



3.1.2 Tweets Sentiment Distribution –> Test

[414]:
```
test.groupby(['Sentiment']) \
    .count()[['UserName']] \
    .reindex(['Extremely Negative',
              'Negative',
              'Neutral',
              'Positive',
              'Extremely Positive']) \
    .plot(kind="bar")
plt.title("Distribution of Sentiment [Test]",color='hotpink')
plt.show()
```



3.2 a. NULL Values

[415]:
```
total_null_train = train.isnull().sum().sort_values(ascending = False)
percentage_null_train=((train.isnull().sum()/train.isnull().count())*100).
  ↪sort_values(ascending = False)

print("Total records = ", train.shape[0])

missing_data = pd.concat([total_null_train, percentage_null_train.round(2)],␣
  ↪axis=1, keys=['Total Missing', 'In Percent'])
missing_data.head(12)
```

```
Total records =  41157
```

[415]:
	Total Missing	In Percent
Location	8590	20.87
UserName	0	0.00
ScreenName	0	0.00
TweetAt	0	0.00
OriginalTweet	0	0.00
Sentiment	0	0.00

[416]:
```
total_null_test = test.isnull().sum().sort_values(ascending = False)
percentage_null_test=((test.isnull().sum()/test.isnull().count())*100).
 ↪sort_values(ascending = False)

print("Total records = ", test.shape[0])

missing_data = pd.concat([total_null_test, percentage_null_test.round(2)],␣
 ↪axis=1, keys=['Total Missing', 'In Percent'])
missing_data.head(12)
```

```
Total records =  3798
```

[416]:
	Total Missing	In Percent
Location	834	21.96
UserName	0	0.00
ScreenName	0	0.00
TweetAt	0	0.00
OriginalTweet	0	0.00
Sentiment	0	0.00

[417]:
```
# train=train.drop(columns=['Location'],axis=1,inplace=True)
```

[418]:
```
# test=test.drop(columns=['Location'],axis=1,inplace=True)
```

3.3 b. Reduction of Data

[419]:
```
train['Sentiment'].nunique()
```

[419]:
```
5
```

[420]:
```
train.Sentiment.value_counts()
```

[420]:
```
Positive              11422
Negative               9917
Neutral                7713
Extremely Positive     6624
Extremely Negative     5481
Name: Sentiment, dtype: int64
```

[421]:
```
test['Sentiment'].nunique()
```

[421]:
```
5
```

[422]:
```
test.Sentiment.value_counts()
```

[422]:
```
Negative              1041
Positive               947
Neutral                619
Extremely Positive     599
Extremely Negative     592
Name: Sentiment, dtype: int64
```

Extract and separate the data based on their labels

[423]:
```
train0=train[train['Sentiment']=='Negative']
train1=train[train['Sentiment']=='Positive']
train2=train[train['Sentiment']=='Neutral']
train3=train[train['Sentiment']=='Extremely Positive']
train4=train[train['Sentiment']=='Extremely Negative']
```

[424]:
```
train0.shape, train1.shape, train2.shape, train3.shape, train4.shape
```

[424]:
```
((9917, 6), (11422, 6), (7713, 6), (6624, 6), (5481, 6))
```

Reducing size of each label by 1/5

[425]:
```
train0=train0[:int(train0.shape[0]/5)]
train1=train1[:int(train1.shape[0]/5)]
train2=train2[:int(train2.shape[0]/5)]
train3=train3[:int(train3.shape[0]/5)]
train4=train4[:int(train4.shape[0]/5)]
```

[426]:
```
train0.shape, train1.shape, train2.shape, train3.shape, train4.shape
```

[426]:
```
((1983, 6), (2284, 6), (1542, 6), (1324, 6), (1096, 6))
```

[427]:
```
train=pd.concat([train0,train1,train2,train3,train4],axis=0)
```

[428]:
```
train.shape
```

[428]:
```
(8229, 6)
```

[429]:
```
train.head()
```

[429]:
```
     UserName  ScreenName          Location     TweetAt  \
9        3808       48760  BHAVNAGAR,GUJRAT  16-03-2020
24       3823       48775  Downstage centre  16-03-2020
```

```
26      3825       48777    Ketchum, Idaho  16-03-2020
28      3827       48779      New York, NY  16-03-2020
30      3829       48781               NaN  16-03-2020
```

```
                                        OriginalTweet Sentiment
9   For corona prevention,we should stop to buy th…  Negative
24  @10DowningStreet @grantshapps what is being do…  Negative
26  In preparation for higher demand and a potenti…  Negative
28  Do you see malicious price increases in NYC? T…  Negative
30  There Is of in the Country  The more empty she…  Negative
```

Dropping all the columns OriginalTweet rating and Sentiment

[430]:
```
train=train.drop(['UserName','ScreenName','Location','TweetAt'],axis=1)
```

[431]:
```
train.head()
```

[431]:
```
                                        OriginalTweet Sentiment
9   For corona prevention,we should stop to buy th…  Negative
24  @10DowningStreet @grantshapps what is being do…  Negative
26  In preparation for higher demand and a potenti…  Negative
28  Do you see malicious price increases in NYC? T…  Negative
30  There Is of in the Country  The more empty she…  Negative
```

[432]:
```
train.Sentiment.value_counts()
```

[432]:
```
Positive              2284
Negative              1983
Neutral               1542
Extremely Positive    1324
Extremely Negative    1096
Name: Sentiment, dtype: int64
```

[433]:
```
test=test.drop(['UserName','ScreenName','Location','TweetAt'],axis=1)
```

[434]:
```
test.head()
```

[434]:
```
                                        OriginalTweet           Sentiment
0  TRENDING: New Yorkers encounter empty supermar…  Extremely Negative
1  When I couldn't find hand sanitizer at Fred Me…            Positive
2  Find out how you can protect yourself and love…  Extremely Positive
3  #Panic buying hits #NewYork City as anxious sh…            Negative
4  #toiletpaper #dunnypaper #coronavirus #coronav…             Neutral
```

[435]:
```
test.Sentiment.value_counts()
```

[435]:
```
Negative              1041
Positive               947
Neutral                619
Extremely Positive     599
Extremely Negative     592
Name: Sentiment, dtype: int64
```

3.4 c. Data Cleaning

3.4.1 i. Hashtag Removal

[436]:
```
import re
```

[437]:
```
def hashtags_removal(text):
    hashtags = "#[\S]+"
    text = re.sub(hashtags,"",text)
    return text
```

[438]:
```
#Remove Hashtags train
train['OriginalTweet'] = train['OriginalTweet'].apply(lambda x:␣
 ↪hashtags_removal(x))
```

[439]:
```
#Remove Hashtags test
test['OriginalTweet'] = test['OriginalTweet'].apply(lambda x:␣
 ↪hashtags_removal(x))
```

3.4.2 ii. Mentions Removal

[440]:
```
def mentions_removal(text):
    mentions = "@[\S]+"
    text = re.sub(mentions,"",text)
    return text
```

[441]:
```
#Remove Mention train
train['OriginalTweet'] = train['OriginalTweet'].apply(lambda x:␣
 ↪mentions_removal(x))
```

[442]:
```
#Remove Mention test
test['OriginalTweet'] = test['OriginalTweet'].apply(lambda x:␣
 ↪mentions_removal(x))
```

3.4.3 iii. URL Removal

[443]:
```
def url_removal(text):
    url = "https?://[A-z0-9_%/\-\.]+[A-z0-9_\.\-\?&=%]+"
    text = re.sub(url,"",text)
    return text
```

[444]:
```
#Remove URL train
train['OriginalTweet'] = train['OriginalTweet'].apply(lambda x: url_removal(x))
```

[445]:
```
#Remove URL test
test['OriginalTweet'] = test['OriginalTweet'].apply(lambda x: url_removal(x))
```

3.4.4 iv. Stopwords Removal

[446]:
```
# Import stopwords with nltk.
from nltk.corpus import stopwords
stop = stopwords.words('english')
```

[447]:
```
train['OriginalTweet'] = train['OriginalTweet'].apply(lambda x: ' '.join([word␣
 ↪for word in x.split() if word not in (stop)]))
```

[448]:
```
test['OriginalTweet'] = test['OriginalTweet'].apply(lambda x: ' '.join([word␣
 ↪for word in x.split() if word not in (stop)]))
```

3.4.5 v. LowerCase

[449]:
```
train['OriginalTweet']=train['OriginalTweet'].str.lower()
```

[450]:
```
test['OriginalTweet']=test['OriginalTweet'].str.lower()
```

3.4.6 vi. Stemming

[451]:
```
import nltk
```

[452]:
```
from nltk.stem import PorterStemmer
```

[453]:
```
stemmer = PorterStemmer()
```

[454]:
```
def stem_sentence(sentence):
    words = nltk.word_tokenize(sentence.lower())
    stemmed_words = [stemmer.stem(word) for word in words if word not in stop]
    stemmed_sentence = " ".join(stemmed_words)
    return stemmed_sentence
```

[455]:
```
train["OriginalTweet"] = train["OriginalTweet"].apply(stem_sentence)
```

3.4.7 vi. Removing Punctuations

[456]:
```
def punctuations_removal(text):
    puntuations = "[\.\?!,;:]+"
    text = re.sub(puntuations,"",text)
    return text
```

[457]:
```
#Remove Punctuations train
train['OriginalTweet'] = train['OriginalTweet'].apply(lambda x:␣
 ↪punctuations_removal(x))
```

[458]:
```
#Remove Punctuations train
test['OriginalTweet'] = test['OriginalTweet'].apply(lambda x:␣
 ↪punctuations_removal(x))
```

3.5 d. Randomization

[459]:
```
train_array = train.to_numpy()
np.random.shuffle(train_array)
train = pd.DataFrame(train_array, columns=train.columns)
```

[460]:
```
test_array = test.to_numpy()
np.random.shuffle(test_array)
test = pd.DataFrame(test_array, columns=test.columns)
```

[461]:
```
train.head()
```

[461]:

	OriginalTweet	Sentiment
0	work groceri store front line shit nobodi wa…	Extremely Positive
1	hello brother 's sister 's let 's connect hand…	Positive
2	still believ bare work despit full deliveri us…	Neutral
3	system place ensur run food	Positive
4	corona prevent stop buy thing cash use onlin …	Negative

[462]:
```
test.head()
```

[462]:

	OriginalTweet	Sentiment
0	no toilet paper local supermarket bought kitch…	Negative
1	continues effect including in-store experiences	Neutral
2	having extra income stock non-perishables meds…	Negative
3	state suburban grocery store time thanksgiving…	Positive
4	irish people italian experience anything go by…	Neutral

4 4. EDA for final Dataset

[463]:
```
train
```

[463]:

	OriginalTweet	Sentiment
0	work groceri store front line shit nobodi wa…	Extremely Positive
1	hello brother 's sister 's let 's connect hand…	Positive
2	still believ bare work despit full deliveri us…	Neutral
3	system place ensur run food	Positive
4	corona prevent stop buy thing cash use onlin …	Negative

```
…                                                   …                …
8224  peopl selfish  stop stock pile food  peopl lik…           Negative
8225  turner join groceri chain repres updat public …            Neutral
8226  australia wors itali valid advic cmo brendan m…           Negative
8227  pleas confirm whether dis-chemâ  statement ( p…  Extremely Positive
8228  went groceri store morn deliveri servic slot s…  Extremely Negative

[8229 rows x 2 columns]
```

[464]:
```
test
```

[464]:
```
                                          OriginalTweet           Sentiment
0     no toilet paper local supermarket bought kitch…           Negative
1      continues effect  including in-store experiences              Neutral
2     having extra income stock non-perishables meds…           Negative
3     state suburban grocery store time thanksgiving…           Positive
4     irish people italian experience anything go by…            Neutral
…                                                   …                …
3793          shoppers stockpiling food covid-19 fears            Negative
3794     after best friend conquer local grocery store  Extremely Positive
3795  grocery store customer bought cart full grocer…           Negative
3796  panic buying swamped supermarkets tonight new …           Negative
3797  the intensifying it's placing added stress liv…           Negative

[3798 rows x 2 columns]
```

4.0.1 one hot encoding

[465]:
```
one_hot_encoding = {
    'Positive': 0,
    'Negative': 1,
    'Neutral': 2,
    'Extremely Positive': 3,
    "Extremely Negative": 4
}
```

[466]:
```
train['Sentiment']=train['Sentiment'].map(one_hot_encoding)
```

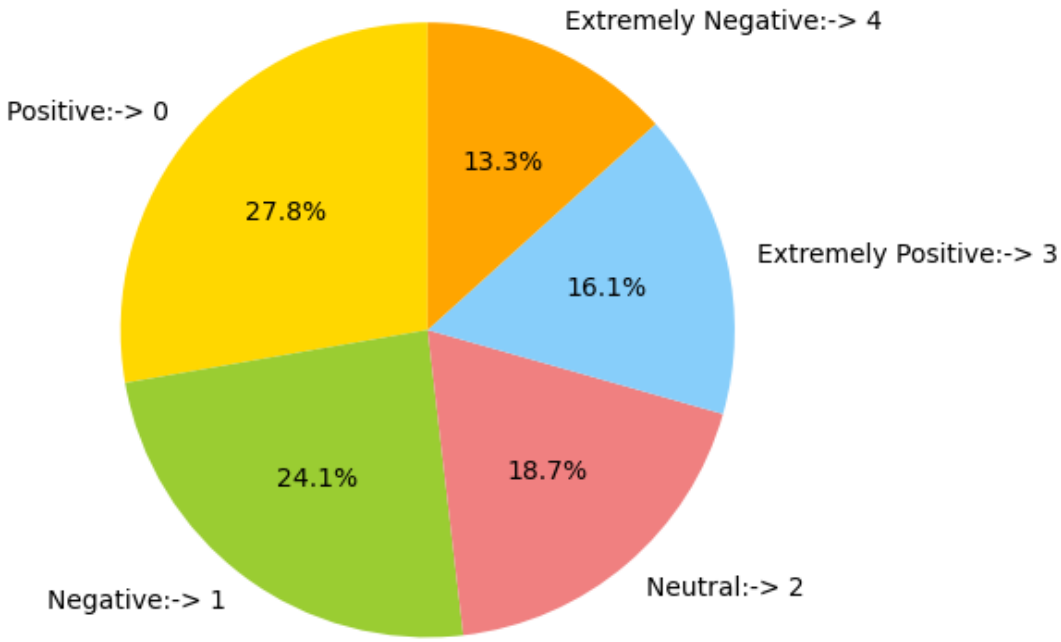
[467]:
```
train['Sentiment'].value_counts()
```

[467]:
```
0    2284
1    1983
2    1542
3    1324
4    1096
Name: Sentiment, dtype: int64
```

4.0.2 Pie Chart Distribution of sample train tweets

[468]:
```
sentiment_counts = train['Sentiment'].value_counts()
labels = ['Positive:-> 0', 'Negative:-> 1', 'Neutral:-> 2', 'Extremely Positive:
 ↪-> 3', 'Extremely Negative:-> 4']
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'orange']

plt.pie(sentiment_counts, labels=labels, colors=colors, autopct='%1.1f%%',␣
 ↪startangle=90)
plt.axis('equal')
plt.show()
```



[469]:
```
test['Sentiment']=test['Sentiment'].map(one_hot_encoding)
```

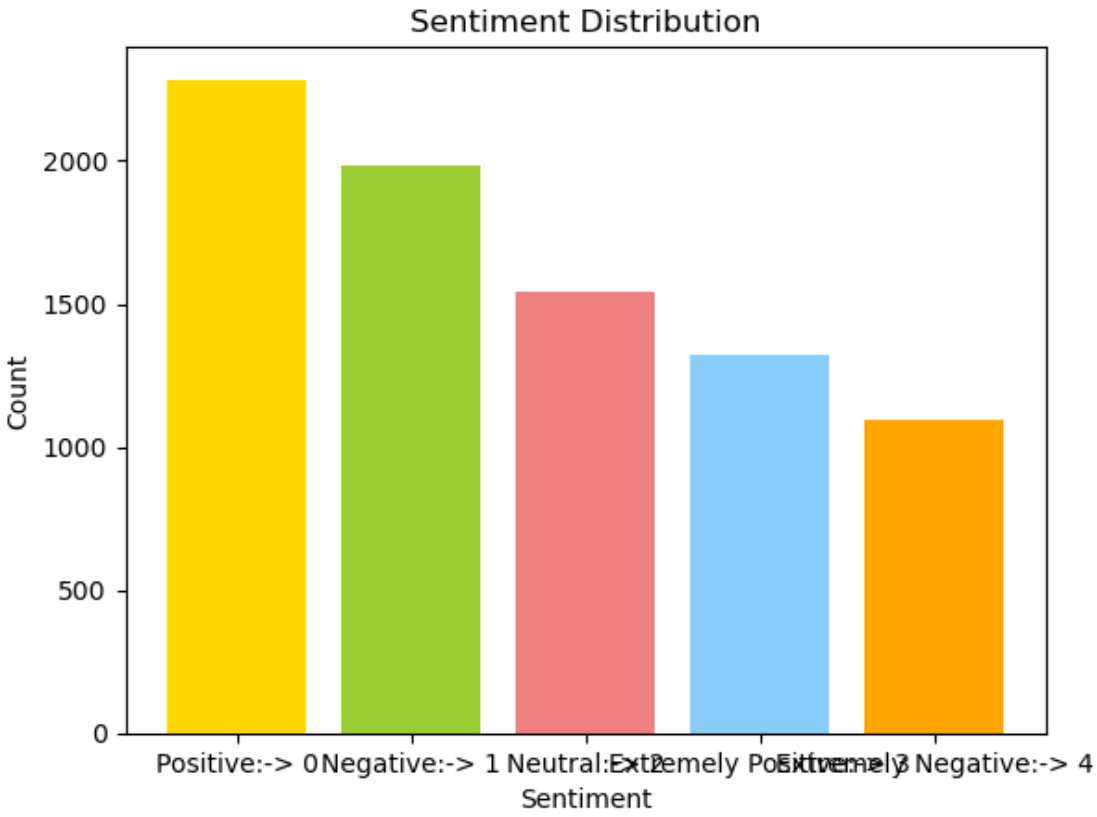
[470]:
```
test['Sentiment'].value_counts()
```

[470]:
```
1    1041
0     947
2     619
3     599
4     592
Name: Sentiment, dtype: int64
```

4.1 Bar plot for tweets sentiment distribution on test data

[471]:
```
sentiment_counts = train['Sentiment'].value_counts()
labels = ['Positive:-> 0', 'Negative:-> 1', 'Neutral:-> 2', 'Extremely Positive:
 ↪-> 3', 'Extremely Negative:-> 4']
colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'orange']

plt.bar(labels, sentiment_counts, color=colors)
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Sentiment Distribution')
plt.show()
```



[472]:
```
x = train["OriginalTweet"].copy()
y = train["Sentiment"].copy()
```

[473]:
```
x.shape
```

[473]: (8229,)

[474]:
```
x.head()
```

[474]:
```
0    work groceri store  front line shit  nobodi wa…
1    hello brother 's sister 's let 's connect hand…
2    still believ bare work despit full deliveri us…
3                          system place ensur run food
4    corona prevent  stop buy thing cash use onlin …
Name: OriginalTweet, dtype: object
```

[475]:
```
y.shape
```

[475]:
```
(8229,)
```

[476]:
```
y.head()
```

[476]:
```
0    3
1    0
2    2
3    0
4    1
Name: Sentiment, dtype: int64
```

[477]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8229 entries, 0 to 8228
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   OriginalTweet  8229 non-null   object
 1   Sentiment      8229 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 128.7+ KB
```

[478]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3798 entries, 0 to 3797
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   OriginalTweet  3798 non-null   object
 1   Sentiment      3798 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 59.5+ KB
```

[479]:
```
train.describe()
```

[479]:
```
          Sentiment
count  8229.000000
mean      1.631182
std       1.381683
min       0.000000
25%       0.000000
50%       1.000000
75%       3.000000
max       4.000000
```

[480]:
```
test.describe()
```

[480]:
```
          Sentiment
count  3798.000000
mean      1.696682
std       1.400419
min       0.000000
25%       1.000000
50%       1.000000
75%       3.000000
max       4.000000
```

[481]:
```
x.isnull().sum()
```

[481]:
```
0
```

[482]:
```
y.isnull().sum()
```

[482]:
```
0
```

[483]:
```
x.dtypes
```

[483]:
```
dtype('O')
```

[484]:
```
y.dtypes
```

[484]:
```
dtype('int64')
```

5 5. Vectorization

5.1 a. TF-IDF

[485]:
```
# TfidfVectorizer from sklearn.feature_extraction.text module
from sklearn.feature_extraction.text import TfidfVectorizer
```

[486]:
```
# Creating a word corpus for vectorization
corpus = []
```

```
for i in range(x.shape[0]):
    corpus.append(x.iloc[i])

vectorizer1 = TfidfVectorizer(max_features=1000)
X1 = vectorizer1.fit_transform(x)
feature_names1 = vectorizer1.get_feature_names()
denselist1 = X1.todense().tolist()
train = pd.DataFrame(denselist1, columns=feature_names1)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/deprecation.py:87:
FutureWarning: Function get_feature_names is deprecated; get_feature_names is
deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out
instead.
  warnings.warn(msg, category=FutureWarning)
```

5.2 b. BoW

[487]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

[488]:
```
corpus = []
for i in range(x.shape[0]):
    corpus.append(x.iloc[i])

vectorizer = CountVectorizer(max_features=1000)
X = vectorizer.fit_transform(corpus)
feature_names = vectorizer.get_feature_names()
denselist = X.todense().tolist()
train = pd.DataFrame(denselist, columns=feature_names)
```

6 6. Model Application

[489]:
```
x
```

[489]:
```
0       work groceri store  front line shit  nobodi wa…
1       hello brother 's sister 's let 's connect hand…
2       still believ bare work despit full deliveri us…
3                              system place ensur run food
4       corona prevent  stop buy thing cash use onlin …
                               …
8224    peopl selfish  stop stock pile food  peopl lik…
8225    turner join groceri chain repres updat public …
8226    australia wors itali valid advic cmo brendan m…
8227    pleas confirm whether dis-chemâ  statement ( p…
8228    went groceri store morn deliveri servic slot s…
Name: OriginalTweet, Length: 8229, dtype: object
```

[490]:
```
y
```

[490]:
```
0       3
1       0
2       2
3       0
4       1
       ..
8224    1
8225    2
8226    1
8227    3
8228    4
Name: Sentiment, Length: 8229, dtype: int64
```

6.0.1 Train Test Split

[491]:
```
from sklearn.model_selection import train_test_split
```

[492]:
```
x_train,x_test,y_train,y_test=train_test_split(train,y,train_size=0.
↪8,random_state=0)
```

[493]:
```
x_train.shape, x_test.shape,y_train.shape,y_test.shape
```

[493]:
```
((6583, 1000), (1646, 1000), (6583,), (1646,))
```

6.1 a. Linear Regression

[494]:
```
from sklearn.linear_model import LinearRegression
```

[495]:
```
lin_reg=LinearRegression()
```

[496]:
```
lin_reg.fit(x_train,y_train)
```

[496]:
```
LinearRegression()
```

[497]:
```
lin_reg_ypred=lin_reg.predict(x_test)
```

[498]:
```
lin_reg_acc=lin_reg.score(x_test,y_test)
```

[499]:
```
lin_reg_acc
```

[499]:
```
-0.04374663510971
```

7 b. Logistic Regression

[500]:
```
from sklearn.linear_model import LogisticRegression
```

[501]:
```
log_reg=LogisticRegression(C=1.0,penalty='l2',solver='newton-cg')
```

[502]:
```
log_reg.fit(x_train,y_train)
```

[502]:
```
LogisticRegression(solver='newton-cg')
```

[503]:
```
log_reg_ypred=log_reg.predict(x_test)
```

[504]:
```
log_reg_acc=log_reg.score(x_test,y_test)
```

[505]:
```
log_reg_acc
```

[505]:
```
0.5103280680437424
```

7.1 c. Decision Tree

[506]:
```
from sklearn.tree import DecisionTreeClassifier
```

[507]:
```
dt_lcf=DecisionTreeClassifier(criterion='gini',splitter='best',max_depth=2,min_samples_split=2
```

[508]:
```
dt_lcf.fit(x_train,y_train)
```

[508]:
```
DecisionTreeClassifier(max_depth=2, max_leaf_nodes=3)
```

[509]:
```
dt_ypred=dt_lcf.predict(x_test)
```

[510]:
```
dt_acc=dt_lcf.score(x_test,y_test)
```

[511]:
```
dt_acc
```

[511]:
```
0.3201701093560146
```

7.2 d. Random Forest

[512]:
```
from sklearn.ensemble import RandomForestClassifier
```

[513]:
```
rf_clf=RandomForestClassifier(n_estimators=100, criterion="gini", max_depth=4,␣
↪min_samples_split=2, min_samples_leaf=1,random_state=0)
```

[514]:
```
rf_clf.fit(x_train,y_train)
```

[514]:
```
RandomForestClassifier(max_depth=4, random_state=0)
```

[515]:
```
rf_ypred=rf_clf.predict(x_test)
```

[516]:
```
rf_acc=rf_clf.score(x_test,y_test)
```

[517]:
```
rf_acc
```

[517]: 0.31713244228432563

7.3 e. KNN

[518]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[519]:
```
knn=KNeighborsClassifier(metric='manhattan',n_neighbors=5,weights='distance')
```

[520]:
```
knn.fit(x_train,y_train)
```

[520]: KNeighborsClassifier(metric='manhattan', weights='distance')

[521]:
```
knn_ypred=knn.predict(x_test)
```

[522]:
```
knn_acc=knn.score(x_test,y_test)
```

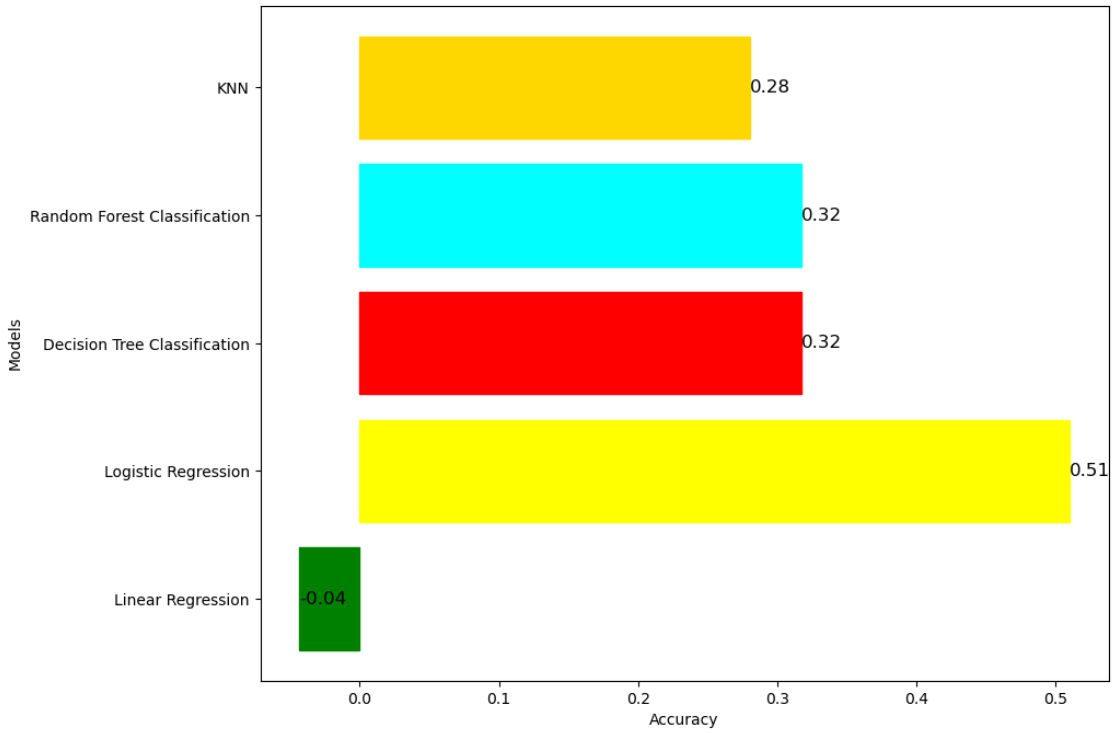
[523]:
```
knn_acc
```

[523]: 0.2800729040097205

8 7. Drawing Plots to presents Results

[524]:
```
names=['Linear Regression','Logistic Regression', 'Decision Tree␣
 ↪Classification' ,'Random Forest Classification','KNN']
acc=[lin_reg_acc,log_reg_acc,rf_acc,rf_acc,knn_acc]
```

[525]:
```
plt.figure(figsize=(10, 8))
graph = plt.barh(names, acc)
plt.xlabel('Accuracy')
plt.ylabel('Models')
graph[0].set_color('green')
graph[1].set_color('yellow')
graph[2].set_color('red')
graph[3].set_color('cyan')
graph[4].set_color('gold')
for i, v in enumerate(acc):
    plt.text(v, i, str(round(v, 2)), color='black', fontsize=12, va='center')
plt.show()
```



9 8. HyperParameter Tuning

[526]:
```
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score,␣
↪confusion_matrix
```

9.1 i. Linear Regression

[527]:
```
from sklearn.model_selection import GridSearchCV
```

[528]:
```
lr = LinearRegression()
```

[529]:
```
param_grid = {
    'fit_intercept': [True, False],
    'normalize': [True, False]
}
```

[530]:
```
grid_search = GridSearchCV(estimator=lr, param_grid=param_grid, cv=5, n_jobs=-1)
```

[531]:
```
grid_search.fit(x_train, y_train)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_base.py:155:
FutureWarning: 'normalize' was deprecated in version 1.0 and will be removed in
```

```
1.2. Please leave the normalize parameter to its default value to silence this
warning. The default behavior of this estimator is to not do any normalization.
If normalization is needed please use sklearn.preprocessing.StandardScaler
instead.
  FutureWarning,
```

[531]:
```
GridSearchCV(cv=5, estimator=LinearRegression(), n_jobs=-1,
             param_grid={'fit_intercept': [True, False],
                         'normalize': [True, False]})
```

[532]:
```
print("Best hyperparameters: ", grid_search.best_params_)
print("Best accuracy: ", grid_search.best_score_)
```

```
Best hyperparameters:  {'fit_intercept': True, 'normalize': False}
Best accuracy:  -0.08663145349073673
```

[533]:
```
y_pred_lin_reg_ht = grid_search.predict(x_test)
```

[534]:
```
print("Mean squared error: ", mean_squared_error(y_test, y_pred_lin_reg_ht))
print("R^2 score: ", r2_score(y_test, y_pred_lin_reg_ht))
```

```
Mean squared error:  2.0424428842512534
R^2 score:  -0.04374663510971
```

[535]:
```
sorted(grid_search.cv_results_.keys())
```

[535]:
```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_fit_intercept',
 'param_normalize',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'split3_test_score',
 'split4_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

9.2 ii. Logistic Regression

[536]:
```
param_grid = {
    'C': [0.1, 1, 10],
    'penalty': ['l2']
```

```
}
```

[537]:
```
lr = LogisticRegression(random_state=42, solver='liblinear')
grid_search = GridSearchCV(estimator=lr, param_grid=param_grid, cv=5, n_jobs=-1)
```

[538]:
```
grid_search.fit(x_train, y_train)
```

[538]:
```
GridSearchCV(cv=5,
             estimator=LogisticRegression(random_state=42, solver='liblinear'),
             n_jobs=-1, param_grid={'C': [0.1, 1, 10], 'penalty': ['l2']})
```

[539]:
```
print("Best hyperparameters: ", grid_search.best_params_)
print("Best accuracy: ", grid_search.best_score_)
```

```
Best hyperparameters:  {'C': 1, 'penalty': 'l2'}
Best accuracy:  0.48078563466291857
```

[540]:
```
y_pred_log_ht = grid_search.predict(x_test)
```

[541]:
```
sorted(grid_search.cv_results_.keys())
```

[541]:
```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_C',
 'param_penalty',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'split3_test_score',
 'split4_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

9.3 iii. Decision Tree

[542]:
```
param_grid = {
    'max_depth': [2, 4],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1]
}
```

[543]:
```
dt = DecisionTreeClassifier(random_state=42)
grid_search = GridSearchCV(estimator=dt, param_grid=param_grid, cv=5, n_jobs=-1)
```

[544]:
```
grid_search.fit(x_train, y_train)
```

[544]:
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(random_state=42), n_jobs=-1,
             param_grid={'max_depth': [2, 4], 'min_samples_leaf': [1],
                         'min_samples_split': [2, 5]})
```

[545]:
```
print("Best hyperparameters: ", grid_search.best_params_)
print("Best accuracy: ", grid_search.best_score_)
```

```
Best hyperparameters:  {'max_depth': 4, 'min_samples_leaf': 1,
'min_samples_split': 2}
Best accuracy:  0.3141402007417614
```

[546]:
```
y_pred_dt_ht = grid_search.predict(x_test)
```

[547]:
```
sorted(grid_search.cv_results_.keys())
```

[547]:
```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_max_depth',
 'param_min_samples_leaf',
 'param_min_samples_split',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'split3_test_score',
 'split4_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

9.4 iv. Random Forest

[548]:
```
param_grid = {
    'n_estimators': [50],
    'max_depth': [2, 4],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1]
}
```

[549]:
```
rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1)
```

[550]:
```
grid_search.fit(x_train, y_train)
```

[550]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42), n_jobs=-1,
             param_grid={'max_depth': [2, 4], 'min_samples_leaf': [1],
                         'min_samples_split': [2, 5], 'n_estimators': [50]})
```

[551]:
```
print("Best hyperparameters: ", grid_search.best_params_)
print("Best accuracy: ", grid_search.best_score_)
```

```
Best hyperparameters:  {'max_depth': 4, 'min_samples_leaf': 1,
'min_samples_split': 2, 'n_estimators': 50}
Best accuracy:  0.3045725409826607
```

[552]:
```
y_pred_rf_ht = grid_search.predict(x_test)
```

[553]:
```
sorted(grid_search.cv_results_.keys())
```

[553]:
```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_max_depth',
 'param_min_samples_leaf',
 'param_min_samples_split',
 'param_n_estimators',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'split3_test_score',
 'split4_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

9.5 v. KNN

[554]:
```
param_grid = {
    'n_neighbors': [3, 5],
    'weights': ['uniform', 'distance']
}
```

[555]:
```
knn = KNeighborsClassifier()
grid_search = GridSearchCV(estimator=knn, param_grid=param_grid, cv=5,␣
 ↪n_jobs=-1)
```

[556]:
```
grid_search.fit(x_train, y_train)
```

[556]:
```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(), n_jobs=-1,
             param_grid={'n_neighbors': [3, 5],
                         'weights': ['uniform', 'distance']})
```

[557]:
```
print("Best hyperparameters: ", grid_search.best_params_)
print("Best accuracy: ", grid_search.best_score_)
```

```
Best hyperparameters:  {'n_neighbors': 3, 'weights': 'distance'}
Best accuracy:  0.29621907115970025
```

[558]:
```
y_pred_knn_ht = grid_search.predict(x_test)
```

[566]:
```
print("Confusion matrix: ", confusion_matrix(y_test, y_pred_knn_ht))
```

```
Confusion matrix:  [[105  51 299  11   5]
 [ 39 100 235   3  19]
 [ 19  28 242   3   1]
 [ 50  29 153  19   5]
 [ 15  77 107   0  31]]
```

[559]:
```
sorted(grid_search.cv_results_.keys())
```

[559]:
```
['mean_fit_time',
 'mean_score_time',
 'mean_test_score',
 'param_n_neighbors',
 'param_weights',
 'params',
 'rank_test_score',
 'split0_test_score',
 'split1_test_score',
 'split2_test_score',
 'split3_test_score',
 'split4_test_score',
 'std_fit_time',
 'std_score_time',
 'std_test_score']
```

[560]:
```
log_reg_acc_ht=accuracy_score(y_test,y_pred_log_ht)
log_reg_acc_ht
```

[560]:
```
0.4957472660996355
```

[561]:
```
dt_acc_ht=accuracy_score(y_test,y_pred_dt_ht)
dt_acc_ht
```

[561]:
```
0.31652490886998785
```

[562]:
```
rf_acc_ht=accuracy_score(y_test,y_pred_rf_ht)
rf_acc_ht
```

[562]:
```
0.3219927095990279
```

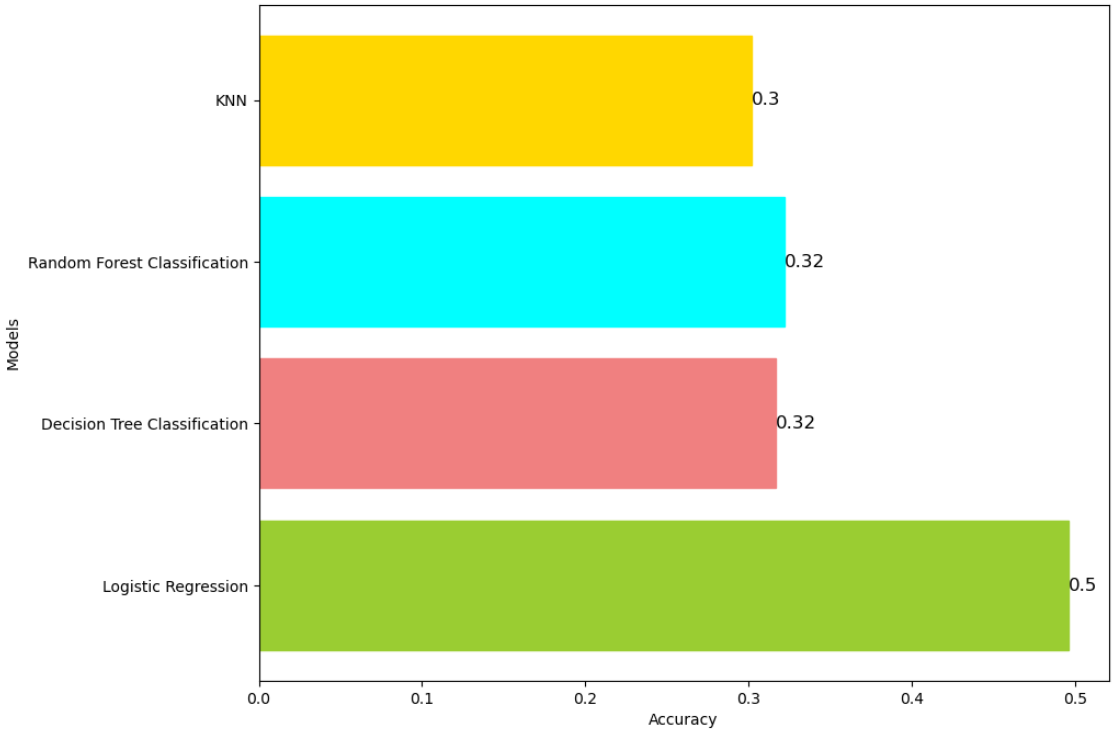
[563]:
```
knn_acc_ht=accuracy_score(y_test,y_pred_knn_ht)
knn_acc_ht
```

[563]:
```
0.3019441069258809
```

9.6 Drawing Plots to presents Results after hyper parameter tuning

[564]:
```
names=['Logistic Regression', 'Decision Tree Classification' ,'Random Forest␣
 ↪Classification','KNN']
acc=[log_reg_acc_ht,dt_acc_ht,rf_acc_ht,knn_acc_ht]
```

[565]:
```
plt.figure(figsize=(10, 8))
graph = plt.barh(names, acc)
plt.xlabel('Accuracy')
plt.ylabel('Models')
graph[0].set_color('yellowgreen')
graph[1].set_color('lightcoral')
graph[2].set_color('cyan')
graph[3].set_color('gold')
for i, v in enumerate(acc):
    plt.text(v, i, str(round(v, 2)), color='black', fontsize=12, va='center')
plt.show()
```


KNN
Random Forest Classification
Decision Tree Classification
Logistic Regression
Models
0.3
0.32
0.32
0.5
0.0
0.1
0.2
0.3
0.4
0.5
Accuracy

Logistic Regression is performing better among the applied classification algorithms