Honours Project - Fake News Detection

April 7, 2020

1 Honours Project - Fake News Detection

[1]:
```
"""
Code to add/ commit/ push repository to github from cmd
git code:
git add .
git commit -m "First commit"
git push origin master
"""
```

[1]:
```
'\nCode to add/ commit/ push repository to github from cmd\ngit code:\ngit add
.\ngit commit -m "First commit"\ngit push origin master\n'
```

This project will be using the approach of CRISP-DM (Cross-industry standard process for data mining), which is a widely used process for knowledge discovery in data sets. The process encompasses several phases:

```
1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment
```

1.1 Step 1 - Business Understanding

Goals/ Objectives: 1. sucessfully create a piece of software that is able to identify fake news 2. model's accuracy to be around the same or better than others 3. model to help answering the author's research questions

Success Criteria: 1. Is the software able to identify fake news with a high accuracy, precision, recall? 2. Is the software more effective in identifying fake news than humans? 3. No errors, bugs etc in the code

1.2 Step 2 - Data Understanding

1.2.1 2 a) - Importing Libraries

[2]:
```
#PREREQUISITES
#   -Anaconda (version Python 3.7) -> for jupyter notebook
#    see https://www.anaconda.com/distribution/
########################################################

import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to C:\Users\Oliver
[nltk_data]     Vinzelberg\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to C:\Users\Oliver
[nltk_data]     Vinzelberg\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to C:\Users\Oliver
[nltk_data]     Vinzelberg\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

[2]: True

[3]:
```
import pandas as pd
import numpy as np
import nltk
import sklearn
import string
import time
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
D:\Anaconda\lib\site-packages\statsmodels\tools\_testing.py:19: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at
pandas.testing instead.
  import pandas.util.testing as tm
```

1.2.2 2 b) - Loading the Data Set

[4]:
```
df = pd.read_csv("data/FakeNews-(balanced)/fake_or_real_news.csv",␣
 ↪encoding="utf-8")
df.head()
```

[4]:
```
   Unnamed: 0                                              title  \
0        8476                       You Can Smell Hillarys Fear
1       10294  Watch The Exact Moment Paul Ryan Committed Pol...
2        3608        Kerry to go to Paris in gesture of sympathy
3       10142  Bernie supporters on Twitter erupt in anger ag...
4         875   The Battle of New York: Why This Primary Matters

                                                text label
0  Daniel Greenfield, a Shillman Journalism Fello...  FAKE
1  Google Pinterest Digg Linkedin Reddit Stumbleu...  FAKE
2  U.S. Secretary of State John F. Kerry said Mon...  REAL
3   Kaydee King (@KaydeeKing) November 9, 2016 T...  FAKE
4  It's primary day in New York and front-runners...  REAL
```

[5]:
```python
#title of the article below
df.iloc[16,1]
```

[5]:
```
'Shocking! Michele Obama & Hillary Caught Glamorizing Date Rape Promoters'
```

[6]:
```python
#preview of a FAKE article
df.iloc[16,2]
```

[6]:
```
'Shocking! Michele Obama & Hillary Caught Glamorizing Date Rape Promoters First
lady claims moral high ground while befriending rape-glorifying rappers
Infowars.com - October 27, 2016 Comments \nAlex Jones breaks down the complete
hypocrisy of Michele Obama and Hillary Clinton attacking Trump for comments he
made over a decade ago while The White House is hosting and promoting rappers
who boast about date raping women and selling drugs in their music. \nRappers
who have been welcomed to the White House by the Obamas include Rick Ross,
who promotes drugging and raping woman in his song U.O.N.E.O. \nWhile
attacking Trump as a sexual predator, Michelle and Hillary have further
mainstreamed the degradation of women through their support of so-called
musicians who attempt to normalize rape. NEWSLETTER SIGN UP Get the latest
breaking news & specials from Alex Jones and the Infowars Crew. Related
Articles'
```

[7]:
```python
#title of the article below
df.iloc[8,1]
```

[7]:
```
"Fact check: Trump and Clinton at the 'commander-in-chief' forum"
```

[8]:
```python
#preview of a REAL article
df.iloc[8,2]
```

[8]:
```
'Hillary Clinton and Donald Trump made some inaccurate claims during an NBC
commander-in-chief forum on military and veterans issues:\n\n Clinton wrongly
claimed Trump supported the war in Iraq after it started, while Trump was wrong,
once again, in saying he was against the war before it started.\n\n\xa0Trump
said that President Obama set a certain date for withdrawing troops from Iraq,
when that date was set before Obama was sworn in.\n\n\xa0Trump said that
Obamas visits to China, Saudi Arabia and Cuba were the first time in the
```

history, the storied history of Air Force One when high officials of a host country did not appear to greet the president. Not true.\n\n\xa0Clinton said that Trump supports privatizing the Veterans Health Administration. Thats false. Trump said he supports allowing veterans to seek care at either public or private hospitals.\n\n\xa0Trump said Clinton made a terrible mistake on Libya when she was secretary of State. But, at the time, Trump also supported U.S. action that led to the removal of Moammar Gadhafi from power.\n\n\xa0Trump cherry-picked Clintons words when he claimed Clinton said vets are being treated, essentially, just fine. Clinton had said the problems in the Department of Veterans Affairs were not as widespread as some Republicans claimed, but she went on to acknowledge problems, including the issue of wait times for doctors.\n\nThe forum, sponsored by NBC News and the Iraq and Afghanistan Veterans of America, was held Sept. 7 at the Intrepid Sea, Air & Space Museum in New York City. Today\xa0show host Matt Lauer, and members of the military and veterans in the audience, questioned the candidates separately.\n\nTrump said he was totally against the war in Iraq, while Clinton claimed that he supported the Iraq War before and after it started. The facts dont support either candidates strong assertions.\n\nOur review of Trumps statements before and after the Iraq War started found no evidence that Trump opposed the war before it started. In fact, he expressed mild support for invading Iraq when asked about it on the Howard Stern radio show on Sept. 11, 2002 about six months before the war started.\n\nStern asked Trump if he supported a war with Iraq, and Trump responded, Yeah, I guess so.\n\nIn the NBC commander in chief forum, Trump cited an Esquire article that appeared in August 2004 to show his opposition to the war. But that article appeared 17 months after the war started.\n\nAs for Clinton, who as a senator voted in October 2002 to authorize the war in Iraq, the Democratic nominee claimed that Trump supported it before it happened, he supported it as it was happening and he is on record as supporting it after it happened.\n\nBut just as there is no evidence that Trump opposed the Iraq War before it started, the Clinton campaign offered no evidence that Trump supported the war after it happened.\n\nThe Clinton campaign cited Trumps interview on March 21, 2003, with Neil Cavuto of Fox Business just two days after the war started.\n\nCavuto asked Trump about the impact of the war on the stock market. Trump said the war looks like a tremendous success from a military standpoint, and he predicted the market will go up like a rocket after the war. But Cavuto does not ask Trump whether the U.S. should have gone to war with Iraq or whether he supports the war, and Trump doesnt offer an opinion.\n\nAs early as July 2003, Trump expressed concern on Hardball with Chris Matthews\xa0about money being spent in Iraq rather than in the U.S. Two months later, Trump told MSNBCs Joe Scarborough, I guess maybe if I had to do it, I would have fought terrorism but not necessarily Iraq.\n\nClinton invited her audience to read Trumps comments on the Iraq War. They can read our timeline, Donald Trump and the Iraq War.\n\nTrump said President Obama set a certain date for withdrawing troops from Iraq, but that date was actually set by President George W. Bush.\n\nNBCs Matt Lauer asked Trump about his tendency to respond, when pushed for details on his military proposals, that hes not going to give details because he wants to be unpredictable. Trump responded, Absolutely, and went on to criticize Obama for revealing the withdrawal date.\n\nAs we said then, Republicans and Democrats disagree on whether Obama or Bush is to blame for withdrawing all combat troops from Iraq at the end of 2011. But that date was set when Bush signed the Status of Forces Agreement on Dec. 14, 2008. It said: All the United States Forces shall withdraw from all Iraqi territory no later than December 31, 2011.\n\nIn the NBC forum, Trump also called the withdrawal of troops a terrible decision. As weve explained before, Condoleezza Rice, Bushs secretary of State, later wrote that Bush wanted an agreement for a residual force to remain, but Iraqi Prime Minister Nouri al-Maliki objected.\n\nOnce Obama took office in January 2009, he had three years to renegotiate the deal, which his administration tried to do, to leave a residual American troop force. But Maliki still didnt agree. Negotiations broke down in October 2011 over the issue of whether U.S. troops would be shielded from criminal prosecution by Iraqi authorities. Whether Obama did enough is a matter of opinion: His then defense secretary, Leon Panetta, later wrote that the president didnt press hard enough for a deal. But some experts say Iraq was more closely aligned at the time with Iran and there wasnt a deal to be made with Maliki.\n\nSo, both presidents had a role in the withdrawal of troops. But Trump wrongly said that Obama was the one who set a certain date for withdrawal and let U.S. enemies know about it, when that date was set before Obama was sworn in.\n\nIts worth noting that Trump said in a March 16, 2007, interview on CNN that the troops should be withdrawn quickly from Iraq.\n\nTrump said that Obamas visits to China, Saudi Arabia and Cuba were the first time in the history, the storied history of Air Force One when high officials of a host country did not appear to greet the president.\n\nThats not true. Other presidents have encountered similar low-key greetings on foreign trips aboard the presidential aircraft.\n\nTrump referred to the fact that Cubas president, Raul Castro, did not greet Obama at the airport on his historic visit to Cuba in March, that Saudi Arabias King Salman did not meet Air Force One at the start of Obamas trip to Riyadh in April, and he referred to Chinas handling of the presidents arrival in Hangzhou last Saturday for a Group of 20 meeting.\n\nWhether or not those arrivals constituted snubs of a U.S. president as Trump claims is a matter of debate. But Trump is wrong on the facts when he claims it has not happened before. It has.\n\nIn 1984, for example, Ronald Reagan landed in Beijing and was received by Chinas foreign minister rather than the president, whom he met only later. Similarly, on a 1985 trip to West Germany, Reagan was met by the foreign minister and not Chancellor Helmut Kohl.\n\nThese and other examples were dug up by our friend Glenn Kessler, the Washington Posts Fact Checker, who researched a Trump claim in April that Cubas and Saudi Arabias handling of Obamas visits were without precedent. Kessler said of Trump, once again hes wrong, wrong, wrong.\n\nKessler also noted that during Richard Nixons historic 1972 visit to China he was greeted at the airport by the countrys number two man, Premier Zhou Enlai. His boss, Chairman Mao, didnt even agree to meet with Nixon until after he had arrived at a guest house.\n\nClinton said that her plan to overhaul the Veterans Health Administration would not include privatization, which she said Trump supports.\n\nBut Trump refuted that statement when it was his turn to discuss his plan to help veterans. I would not do that, Trump said, referring to Clintons claim that he supports privatization.\n\nTrumps campaign published The Goals Of Donald J. Trumps Veterans Plan on its website last October. It doesnt call for the VA to be completely privatized.\n\nOne of the biggest changes that plan would make to the current VA health care system is allowing veterans to get care at any non-VA medical center that accepts Medicare.\n\nUnder a Trump Administration, all veterans eligible for VA health care can bring their veterans ID card to any doctor or care facility that accepts Medicare to get the care they need immediately, the plan states.\n\nThe power to choose will stop the wait time backlogs and force the VA to improve and compete if the department wants to keep receiving veterans healthcare dollars, the plan says.\n\nTrumps proposal would seemingly go further than the Non-VA Medical Care Program, which allows eligible veterans to access care outside of the VA under certain circumstances, such as when VA medical centers cannot provide services. The program requires pre-approval for veterans to receive care at a non-VA facility in non-emergency situations.\n\nTrumps proposal would also go further than the bipartisan Veterans Choice Act of 2014 that President Obama signed into law, creating a temporary program, separate from the Non-VA Medical Care Program, that allows eligible veterans to receive health care at a non-VA facility if they would have to wait more than 30 days for an appointment at a VA medical center, or if they live more than 40 miles from the nearest VA hospital.\n\nTrump stuck to the idea of allowing veterans to choose between public and private hospitals when he released his most recent Ten Point Plan To Reform The VA in July.\n\nPoint 10 of the plan says: Mr. Trump will ensure every veteran has the choice to seek care at the VA or at a private service provider of their own choice. Under a Trump Administration, no veteran will die waiting for service.\n\nTrump reinforced that part of his plan during the NBC News forum as well.\n\nTo be clear, Trump supports giving veterans a choice between VA hospitals and private ones. Thats not the same thing as supporting the complete privatization of the system that provides care to veterans.\n\nTrump criticized Clinton for making a terrible mistake on Libya when she was secretary of State. But, at the time, Trump also supported U.S. action that led to the removal of Moammar Gadhafi from power.\n\nTrump made his claim in response to a question posed by Lauer on whether Trump will be prepared on Day One, if elected president, to tackle complex national security issues.\n\nThis isnt the first time Trump has ignored his past support for the U.S. intervention in Libya.\n\nDuring the 10th GOP debate, Trump said he had never discussed that subject when Sen. Ted Cruz called him out on supporting U.S. action in the country. But, as we wrote, Trump said in 2011 that the U.S. should go into Libya on a humanitarian basis and knock [Gadhafi] out very quickly, very surgically, very effectively and save the lives.\n\nTrump made that comment in a video posted to his YouTube channel in February 2011:\n\nEven though Trump now says Clintons support for intervention in Libya was a terrible mistake, it doesnt change the fact that five years ago he supported Gadhafis removal.\n\nTrump twisted Clintons words when he claimed Clinton said vets are being treated, essentially, just fine. Clinton said the problems in the Department of Veterans Affairs were not as

```
widespread as some Republican supporters of privatization of the VA claim, but
she went on to acknowledge problems in the VA system  including the issue of
wait times for doctors  and what she would do to address them.\n\nTrump
highlighted the issue of wait times to see a doctor as one of the big problems
in the VA, and then suggested Clinton doesnt think the VA has
problems.\n\nLauer interrupted, noting that Clinton went on after that and laid
out a litany of problems within the VA.\n\nTrump insisted his version was
accurate, adding, Im telling you  she said she was satisfied with what was
going on in the Veterans Administration.\n\nThats not accurate. The comments
in question from Clinton came during an interview with MSNBCs Rachel Maddow on
Oct. 23, 2015. Maddow asked about talk among some Republicans of abolishing the
VA and privatizing it. The reason they are able to propose something that
radical is because the problems at the VA seem so intractable, Maddow
said.\n\nMaddow asked if Clinton had any new ideas for trying to fix the VA.
Here was Clintons response, with the part Trump is referring to in
bold.\n\nClinton accused Republicans of underfunding the VA because they want
it to fail so they can privatize it.\n\nClinton added, But we have to be more
creative about trying to fix the problems that are the legitimate concern, so
that we can try to stymie the Republican assault.\n\nIndeed, the Clinton
campaign website states that Clinton wants to fundamentally reform veterans
health care to ensure access to timely and high quality care. The campaign says
Clinton was outraged by the recent scandals at the VA, and as president, she
will demand accountability and performance from VA leadership. The site
specifically mentions Clintons dissatisfaction that [m]any veterans have to
wait an unacceptably long time to see a doctor or to process disability claims
and appeals and promises she will [b]uild a 21st-century Department of
Veterans Affairs to deliver world-class care.\n\nTrump cherry-picked the part
of Clintons response that said problems in the VA have not been as widespread
as it has been made out to be, to make the blanket claim that Clinton is
satisfied with what was going on in the Veterans Administration and that vets
are being treated, essentially, just fine. But Trump is leaving out the parts
of Clintons answer that acknowledged problems in the VA  including the wait
time issue Trump highlighted as one of his biggest concerns.'
```

[9]:
```
df.info
```

[9]:
```
<bound method DataFrame.info of       Unnamed: 0
title  \
0           8476                       You Can Smell Hillarys Fear
1          10294  Watch The Exact Moment Paul Ryan Committed Pol...
2           3608        Kerry to go to Paris in gesture of sympathy
3          10142  Bernie supporters on Twitter erupt in anger ag...
4            875   The Battle of New York: Why This Primary Matters
...          ...                                                ...
6330        4490  State Department says it can't find emails fro...
6331        8062  The P in PBS Should Stand for Plutocratic ...
6332        8622  Anti-Trump Protesters Are Tools of the Oligarc...
6333        4021  In Ethiopia, Obama seeks progress on peace, se...
```

```
6334        4330  Jeb Bush Is Suddenly Attacking Trump. Here's W...

                                                   text label
0     Daniel Greenfield, a Shillman Journalism Fello...  FAKE
1     Google Pinterest Digg Linkedin Reddit Stumbleu...  FAKE
2     U.S. Secretary of State John F. Kerry said Mon...  REAL
3      Kaydee King (@KaydeeKing) November 9, 2016 T...  FAKE
4     It's primary day in New York and front-runners...  REAL
...                                                 ...   ...
6330  The State Department told the Republican Natio...  REAL
6331  The P in PBS Should Stand for Plutocratic ...  FAKE
6332   Anti-Trump Protesters Are Tools of the Oligar...  FAKE
6333  ADDIS ABABA, Ethiopia President Obama convene...  REAL
6334  Jeb Bush Is Suddenly Attacking Trump. Here's W...  REAL

[6335 rows x 4 columns]>
```

[10]:
```
columns = df.columns.tolist()
```

[11]:
```
print(columns)
```

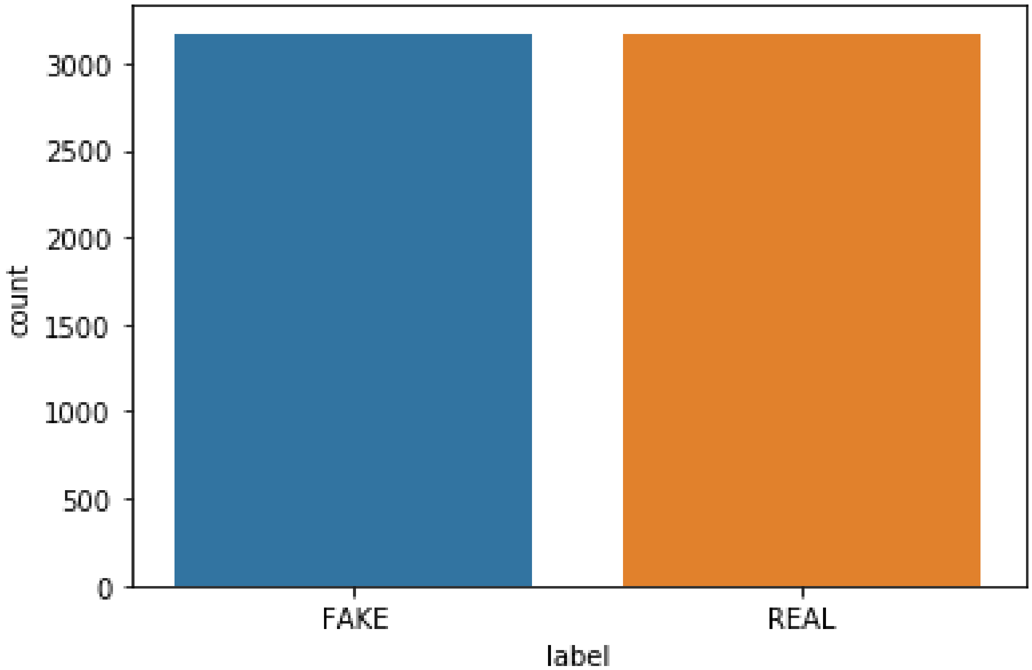
```
['Unnamed: 0', 'title', 'text', 'label']
```

[12]:
```
df["label"].value_counts()
```

[12]:
```
REAL    3171
FAKE    3164
Name: label, dtype: int64
```

[13]:
```
sns.countplot(df['label'])
```

[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2afd42bc128>
```



The data set seems to be well balanced.

[14]:
```
type(df["title"])
```

[14]:
```
pandas.core.series.Series
```

[15]:
```
type(df["text"])
```

[15]:
```
pandas.core.series.Series
```

[16]:
```
df.dtypes
```

[16]:
```
Unnamed: 0     int64
title         object
text          object
label         object
dtype: object
```

It appears that "Unnamed: 0" is the index since it only contains numbers. The column will be checked for duplicates to check.

[17]:
```
print(any(df["Unnamed: 0"].duplicated()))
```

```
False
```

[18]:
```
df.isnull().values.any()
```

[18]:
```
False
```

1.3 Step 3 - Data Preparation

1.3.1 3 a) - General Polishing

[19]:
```
#rename "Unnamed: 0" and make it the index of the data frame
df.columns = ["index", "title", "text", "label"]
df.set_index("index", inplace=True)
```

[20]:
```
df.head()
```

[20]:
```
                                                    title  \
index
8476                         You Can Smell Hillarys Fear
10294  Watch The Exact Moment Paul Ryan Committed Pol...
3608         Kerry to go to Paris in gesture of sympathy
10142  Bernie supporters on Twitter erupt in anger ag...
875     The Battle of New York: Why This Primary Matters

                                                     text label
index
8476   Daniel Greenfield, a Shillman Journalism Fello...  FAKE
10294  Google Pinterest Digg Linkedin Reddit Stumbleu...  FAKE
3608   U.S. Secretary of State John F. Kerry said Mon...  REAL
10142   Kaydee King (@KaydeeKing) November 9, 2016 T...  FAKE
875    It's primary day in New York and front-runners...  REAL
```

[21]:
```
#order by index
df.sort_index(inplace=True)
```

[22]:
```
df.head()
```

[22]:
```
                                                    title  \
index
2      Study: women had to drive 4 times farther afte...
3            Trump, Clinton clash in dueling DC speeches
5      As Reproductive Rights Hang In The Balance, De...
6      Despite Constant Debate, Americans' Abortion O...
7      Obama Argues Against Goverment Shutdown Over P...

                                                     text label
index
2      Ever since Texas laws closed about half of the...  REAL
3      Donald Trump and Hillary Clinton, now at the s...  REAL
5      WASHINGTON -- Forty-three years after the Supr...  REAL
6      It's been a big week for abortion news.\n\nCar...  REAL
7      President Barack ObamaăsaidăSaturday night tha...  REAL
```

[23]:
```
df.index
```

[23]:
```
Int64Index([    2,     3,     5,     6,     7,     9,    10,    12,    14,
               16,
```

```
       ...
       10543, 10545, 10546, 10547, 10548, 10549, 10551, 10553, 10555,
       10557],
      dtype='int64', name='index', length=6335)
```

The index seems to skip some numbers, for example 8 and 11. The index will be properly assigned.

[24]:
```
df['index'] = df.reset_index().index
```

[25]:
```
df.set_index("index", inplace=True)
df.head()
```

[25]:
```
                                                   title  \
index
0      Study: women had to drive 4 times farther afte...
1             Trump, Clinton clash in dueling DC speeches
2      As Reproductive Rights Hang In The Balance, De...
3      Despite Constant Debate, Americans' Abortion O...
4      Obama Argues Against Goverment Shutdown Over P...

                                                    text label
index
0      Ever since Texas laws closed about half of the...   REAL
1      Donald Trump and Hillary Clinton, now at the s...   REAL
2      WASHINGTON -- Forty-three years after the Supr...   REAL
3      It's been a big week for abortion news.\n\nCar...   REAL
4      President Barack ObamaăsaidăSaturday night tha...   REAL
```

It appears that the data contains several characters like or İThey will be removed from the data set.

1.3.2 3 b) - Normalising The Data

[26]:
```
# the function strip() will be used to remove those characters
# Example:
s = "\n \a abc \n \n"
print(s.strip())
```

```
abc
```

[27]:
```
s = "\n\nCar"
print(s.strip())
```

```
Car
```

[28]:
```
df["text"] = df["text"].apply(lambda x: x.strip())
```

[29]:
```
# since some characters are part of the string, they have to be removed with␣
↪the replace function
```

```
df["text"] = df["text"].apply(lambda x: x.replace("\n", ""))
df["text"] = df["text"].apply(lambda x: x.replace("\t", ""))
#df["text"] = df["text"].apply(lambda x: x.replace("\x", ""))
df["text"] = df["text"].apply(lambda x: x.replace("\xa0", ""))

df["title"] = df["title"].apply(lambda x: x.replace("\n", ""))
df["title"] = df["title"].apply(lambda x: x.replace("\t", ""))
#df["title"] = df["title"].apply(lambda x: x.replace("\x", ""))
df["title"] = df["title"].apply(lambda x: x.replace("\xa0", ""))
```

[30]:
```
df.head()
```

[30]:
```
                                                   title  \
index
0      Study: women had to drive 4 times farther afte...
1            Trump, Clinton clash in dueling DC speeches
2      As Reproductive Rights Hang In The Balance, De...
3      Despite Constant Debate, Americans' Abortion O...
4      Obama Argues Against Goverment Shutdown Over P...

                                                    text label
index
0      Ever since Texas laws closed about half of the...  REAL
1      Donald Trump and Hillary Clinton, now at the s...  REAL
2      WASHINGTON -- Forty-three years after the Supr...  REAL
3      It's been a big week for abortion news.Carly F...  REAL
4      President Barack ObamasaidSaturday night that ...  REAL
```

The next step is to remove punctuation.

[31]:
```
#df["text"] = df["text"].apply(lambda x: x.replace(string.punctuation, ""))
#df["title"] = df["title"].apply(lambda x: x.replace(string.punctuation, ""))

df["title"] = df["title"].str.replace("[{}]".format(string.punctuation), "")
df["text"] = df["text"].str.replace("[{}]".format(string.punctuation), "")
```

[32]:
```
#convert every word to lower case - normalising case
df["title"] = df["title"].str.lower()
df["text"] = df["text"].str.lower()
df["label"] = df["label"].str.lower()
```

[33]:
```
df.head()
```

[33]:
```
                                                   title  \
index
0      study women had to drive 4 times farther after...
1             trump clinton clash in dueling dc speeches
2      as reproductive rights hang in the balance deb...
3      despite constant debate americans abortion opi...
4      obama argues against goverment shutdown over p...
```

```
                                                     text label
index
0      ever since texas laws closed about half of the...   real
1      donald trump and hillary clinton now at the st...   real
2      washington  fortythree years after the supreme...   real
3      its been a big week for abortion newscarly fio...   real
4      president barack obamasaidsaturday night that ...   real
```

Additionally the labels will be converted to binary values: 0 and 1.

[34]:
```
#df["label"] = df["label"].apply(lambda x: x.replace("real", 0))
#df["label"] = df["label"].apply(lambda x: x.replace("fake", 1))

df["label"] = df["label"].replace(to_replace=["real", "fake"], value=[0, 1])
```

[35]:
```
df.head()
```

[35]:
```
                                                    title  \
index
0      study women had to drive 4 times farther after...
1             trump clinton clash in dueling dc speeches
2      as reproductive rights hang in the balance deb...
3      despite constant debate americans abortion opi...
4      obama argues against goverment shutdown over p...

                                                     text  label
index
0      ever since texas laws closed about half of the...      0
1      donald trump and hillary clinton now at the st...      0
2      washington  fortythree years after the supreme...      0
3      its been a big week for abortion newscarly fio...      0
4      president barack obamasaidsaturday night that ...      0
```

In the next step stopwords such as "the" or "a" will be removed since they do not contribute to a deeper meaning of a sentence.

[36]:
```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
print(stop_words)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
"you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's",
'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what',
'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is',
'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above',
'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',
'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why',
```

```
'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some',
'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn',
"couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn',
"hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
"shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```

[37]:
```
from nltk.tokenize import word_tokenize
import string

#function that tokenises words and removes stop words, punctuation and non␣
↪alphanumerical characters in a sentence
def func_normalise(sentence):
    tokens = word_tokenize(sentence)
    #print(tokens)
    stop_words = set(stopwords.words("english"))
    stop_words.add("n't")
    stop_words.add("nt")
    stop_words.add("u")

    table = str.maketrans("", "", string.punctuation)
    stripped = [w.translate(table) for w in tokens]

    words = [word for word in stripped if word.isalpha()]

    new_sentence = [w for w in words if not w in stop_words]

    new_sentence_str = " ".join(new_sentence)

    return new_sentence_str
```

[38]:
```
#testing the function
func_normalise("ever ? since hasn't the / texa*s laws closed about half of the␣
↪where didn't")
```

[38]:
```
'ever since texas laws closed half'
```

Now the above function will be applied to the data in order to normalise it.

[39]:
```
df["title"] = df["title"].apply(func_normalise)
```

[40]:
```
df["text"] = df["text"].apply(func_normalise)
```

[41]:
```
df.head()
```

[41]:
```
                                                     title  \
index
0      study women drive times farther texas laws clo...
```

```
1                trump clinton clash dueling dc speeches
2      reproductive rights hang balance debate modera...
3      despite constant debate americans abortion opi...
4      obama argues goverment shutdown planned parent...

                                                  text  label
index
0      ever since texas laws closed half states abort...      0
1      donald trump hillary clinton starting line gen...      0
2      washington fortythree years supreme court esta...      0
3      big week abortion newscarly fiorinas passionat...      0
4      president barack obamasaidsaturday night congr...      0
```

As seen in the above example the text data has been (successfully) normalised.

1.3.3 3 c) - Stemming

Stemming is the process of reducing words to their root. For example, "playing" and "played" reduce to the stem "play". Therefore stemming helps with reducing the vocabulary and allows to focus on the sense of a sentence.

[42]:
```
#from nltk.stem.porter import PorterStemmer
#according to the nltk website the snowballstemmer is better than the␣
 ↪"original" porter stemmer
#https://www.nltk.org/howto/stem.html

from nltk.stem.snowball import SnowballStemmer

#function that stems words in a sentence
def func_stem(sentence):
    tokens = word_tokenize(sentence)
    snowball_stemmer = SnowballStemmer("english")
    stemmed_sentence = [snowball_stemmer.stem(word) for word in tokens]
    stemmed_sentence_str = " ".join(stemmed_sentence)
    return stemmed_sentence_str
```

[43]:
```
#test
func_stem("playing player play played plays")
```

[43]:
```
'play player play play play'
```

[44]:
```
df["title"] = df["title"].apply(func_stem)
```

[45]:
```
df["text"] = df["text"].apply(func_stem)
```

[46]:
```
df.head()
```

[46]:
```
                                                 title  \
index
0      studi women drive time farther texa law close ...
1                     trump clinton clash duel dc speech
2      reproduct right hang balanc debat moder drop ball
```

```
3      despit constant debat american abort opinion r...
4              obama argu gover shutdown plan parenthood

                                                    text  label
index
0      ever sinc texa law close half state abort clin...      0
1      donald trump hillari clinton start line genera...      0
2      washington fortythre year suprem court establi...      0
3      big week abort newscar fiorina passion inaccur...      0
4      presid barack obamasaidsaturday night congress...      0
```

1.3.4 3 d) - Lemmatising

According to literature lemmatising and stemming words is similar. However, stemming tries to cut off endings of words whereas lemmatising compares them to other words. To test whether lemmatising makes a difference in accuracy it will be implemented.

[47]:
```
"""
import nltk
nltk.download('wordnet')
"""

def func_lemmatise(sentence):
    tokens = word_tokenize(sentence)
    lemmatiser = nltk.WordNetLemmatizer()
    lemmatised_sentence = [lemmatiser.lemmatize(word) for word in tokens]
    lemmatised_sentence_str = " ".join(lemmatised_sentence)
    return lemmatised_sentence_str
```

[48]:
```
#test
func_lemmatise("playing player play played plays")
```

[48]:
```
'playing player play played play'
```

[49]:
```
df["title"] = df["title"].apply(func_lemmatise)
```

[50]:
```
df["text"] = df["text"].apply(func_lemmatise)
```

[51]:
```
df.head()
```

[51]:
```
                                                   title  \
index
0      studi woman drive time farther texa law close ...
1                        trump clinton clash duel dc speech
2      reproduct right hang balanc debat moder drop ball
3      despit constant debat american abort opinion r...
4              obama argu gover shutdown plan parenthood

                                                    text  label
index
```

```
0      ever sinc texa law close half state abort clin...      0
1      donald trump hillari clinton start line genera...      0
2      washington fortythre year suprem court establi...      0
3      big week abort newscar fiorina passion inaccur...      0
4      presid barack obamasaidsaturday night congress...      0
```

Six articles are going to be removed since they are part of the survey and because the model should make a prediction on these without being previously biased.

[52]:
```
# 5099         smell hillari fear         daniel greenfield shillman journal␣
↪fellow free...          1
# 4988         poll find american support polic highest near ...         past␣
↪year american seen polic offic ambush ass...          1
# 534          battl new york primari matter         primari day new york␣
↪frontrunn hillari clinton...          0
# 4777         uk announc new troop deploy near russia border         militari␣
↪british defens secretari michael fall...          1
# 2734         russia join franc strike isi stronghold syria         russian␣
↪militari might join french warplan tue...          0
# 2058         shortag lethal inject drug put death penalti s...         suprem␣
↪court mondaydecid oklahoma may continu ...          0

# find a row
#df[df['title'].str.contains("lethal inj")]

# del a row
df.drop(index=5099, inplace=True)
df.drop(index=4988, inplace=True)
df.drop(index=534, inplace=True)
df.drop(index=4777, inplace=True)
df.drop(index=2734, inplace=True)
df.drop(index=2058, inplace=True)
```

[53]:
```
#assign index anew
df['index'] = df.reset_index().index
df.set_index("index", inplace=True)
df.head()
```

[53]:
```
                                                   title  \
index
0      studi woman drive time farther texa law close ...
1                          trump clinton clash duel dc speech
2      reproduct right hang balanc debat moder drop ball
3      despit constant debat american abort opinion r...
4                obama argu gover shutdown plan parenthood

                                                    text  label
index
0      ever sinc texa law close half state abort clin...      0
```

```
1      donald trump hillari clinton start line genera...      0
2      washington fortythre year suprem court establi...      0
3      big week abort newscar fiorina passion inaccur...      0
4      presid barack obamasaidsaturday night congress...      0
```

1.3.5 3 e) - TF-IDF

Since ML algorithms require numerical data as input instead of text, the text will be vectorised using the TF-IDF method, which stands for term frequency - inverse document frequency. TF-IDF is measure of orginiality of a word by comparing the number of times a word appears in a doc with the number of docs the words appears in.

[54]:
```python
df["title"] = df["title"].replace("[^a-zA-Z0-9 ]", "", regex=True)
```

[55]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_df=0.7)
feature_matrix = tfidf.fit_transform(df["title"])
```

[56]:
```python
feature_matrix.toarray()
```

[56]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

[57]:
```python
tfidf.get_feature_names()
```

[57]:
```
['aap',
 'abandon',
 'abbi',
 'abc',
 'abcwapo',
 'abduct',
 'abdullah',
 'abedin',
 'abil',
 'abl',
 'abnorm',
 'aboard',
 'abolish',
 'abort',
 'abortionrevers',
 'abridg',
 'abroad',
 'abrog',
 'absenc',
```

```
'absente',
'absolut',
'abstain',
'absurd',
'abu',
'abus',
'abyss',
'aca',
'accept',
'access',
'accid',
'accident',
'accomplish',
'accord',
'account',
'accur',
'accus',
'acela',
'acheron',
'achiev',
'ackbar',
'acknowledg',
'aclu',
'acquir',
'acquisit',
'acquit',
'acquitt',
'acr',
'across',
'act',
'action',
'activ',
'activist',
'actor',
'actual',
'acupunctur',
'ad',
'adam',
'adapt',
'add',
'adderal',
'addict',
'addictionher',
'address',
'adelson',
'adequ',
'adhd',
```

```
'adhm',
'adjust',
'admin',
'administr',
'admir',
'admit',
'adopt',
'ador',
'adpr',
'adpresnet',
'adult',
'advanc',
'advantag',
'advert',
'advertis',
'advic',
'advis',
'advisor',
'advoc',
'aerodynam',
'afar',
'affair',
'affect',
'affili',
'affirm',
'afford',
'affront',
'afghan',
'afghanistan',
'aficionado',
'afraid',
'africa',
'african',
'africanamerican',
'aftermath',
'aftershock',
'ag',
'age',
'agenc',
'agenda',
'agent',
'aggress',
'aggro',
'agit',
'agn',
'ago',
'agoni',
```

```
'agre',
'agreement',
'agricultur',
'agropoison',
'ahead',
'ai',
'aid',
'aig',
'ail',
'aim',
'air',
'airbnb',
'aircraft',
'airman',
'airplan',
'airport',
'airstrik',
'akbar',
'al',
'alabadi',
'alabama',
'alarm',
'alarmist',
'alasdair',
'alaska',
'alassad',
'albert',
'alberto',
'albright',
'album',
'alcohol',
'aleppo',
'alert',
'alex',
'alfatah',
'ali',
'alicia',
'alien',
'align',
'alist',
'alito',
'aliv',
'allah',
'allahu',
'allamerican',
'allay',
'alleg',
```

```
'allegi',
'allen',
'alli',
'allianc',
'allin',
'alloc',
'allout',
'allow',
'allpay',
'alltim',
'alltoofamiliar',
'almost',
'alnusra',
'alon',
'along',
'aloof',
'alphabet',
'alqaeda',
'alreadi',
'alreadyflag',
'also',
'alsoran',
'alt',
'alter',
'altern',
'altmarket',
'altright',
'altruist',
'alway',
'amaz',
'amazinglook',
'ambassador',
'ambit',
'ambival',
'ambul',
'ambush',
'ambushstyl',
'ame',
'amend',
'america',
'american',
'americana',
'ami',
'amid',
'amish',
'ammo',
'ammon',
```

```
'ammosexu',
'amnesti',
'among',
'amount',
'amp',
'amphibi',
'amtrak',
'amur',
'amurexit',
'amus',
'anaheim',
'analysi',
'analyst',
'anarchi',
'anarchist',
'anastasia',
'anatomi',
'anbar',
'anchor',
'ancient',
'anderson',
'andes',
'andrew',
'anew',
'angel',
'angela',
'angelina',
'anger',
'anglin',
'angri',
'angriest',
'anim',
'animos',
'ankl',
'ann',
'anna',
'annihil',
'anniversari',
'annot',
'announc',
'annual',
'annul',
'anonym',
'anot',
'anoth',
'answer',
'ant',
```

```
'antarct',
'antarctica',
'anthem',
'anthoni',
'anthropecen',
'anti',
'antiaircraft',
'antibiot',
'antic',
'anticip',
'anticlinton',
'anticommunist',
'anticorrupt',
'antidepress',
'antiestablish',
'antifamin',
'antigovern',
'antigovt',
'antihillari',
'antiiran',
'antiiraq',
'antiisi',
'antiislam',
'antiisrael',
'antijihad',
'antilgbt',
'antimalaria',
'antimissil',
'antimuhammad',
'antimuslim',
'antiparticl',
'antiplan',
'antipoverti',
'antiqu',
'antirussia',
'antirussian',
'antisemit',
'antisex',
'antitank',
'antiterror',
'antithet',
'antitrump',
'antiunion',
'antivax',
'antiwar',
'antonin',
'antonio',
```

```
'anxieti',
'anxious',
'anybodi',
'anymor',
'anyon',
'anyth',
'anyway',
'anywher',
'ao',
'ap',
'apart',
'apartheid',
'api',
'apocalyps',
'apocalypt',
'apolog',
'apologis',
'apologistinchief',
'apost',
'app',
'appal',
'appar',
'appeal',
'appear',
'appeas',
'appl',
'applebe',
'appli',
'applic',
'appoint',
'appreci',
'apprentic',
'approach',
'appropri',
'approv',
'april',
'aqutru',
'ar',
'arab',
'arabia',
'arabian',
'arc',
'archil',
'architect',
'arctic',
'arcturian',
'area',
```

```
'arent',
'aretha',
'argu',
'argument',
'ari',
'aria',
'arianna',
'ariel',
'aris',
'aristocraci',
'arizona',
'ark',
'arkansa',
'arm',
'armenia',
'armenian',
'armi',
'armstrong',
'arn',
'arnab',
'around',
'arquett',
'arrang',
'arrest',
'arriv',
'arrog',
'arsenal',
'arson',
'art',
'arteri',
'articl',
'artifact',
'artifici',
'artist',
'aryan',
'as',
'ascend',
'asda',
'ashton',
'ashutosh',
'asia',
'asid',
'ask',
'aspartam',
'ass',
'assad',
'assang',
```

```
'assangepilg',
'assasin',
'assassin',
'assault',
'assembl',
'assert',
'asset',
'assist',
'assistedsuicid',
'associ',
'assum',
'assumpt',
'assur',
'asthm',
'astonish',
'astronaut',
'astronom',
'asylum',
'atf',
'atlanta',
'atlanti',
'atm',
'atroc',
'att',
'attach',
'attack',
'attempt',
'attend',
'attende',
'attent',
'attir',
'attitud',
'attorney',
'attract',
'atttim',
'audaci',
'audienc',
'audio',
'auditor',
'aumf',
'aussi',
'australia',
'australian',
'austria',
'austrian',
'austyn',
'authent',
```

```
'author',
'authoritarian',
'autism',
'autist',
'auto',
'autom',
'automat',
'autonom',
'avail',
'aveng',
'averag',
'avert',
'aviv',
'avocado',
'avoid',
'aw',
'await',
'awaken',
'award',
'away',
'awe',
'awesom',
'awkward',
'awol',
'axelrod',
'axi',
'ayahuasca',
'ayatollah',
'ayott',
'az',
'baba',
'babi',
'babylon',
'back',
'backbon',
'backer',
'backfir',
'background',
'backlash',
'backlog',
'backout',
'backseat',
'backstori',
'backup',
'backward',
'bacon',
'bad',
```

```
'badass',
'badasseri',
'bader',
'badg',
'badham',
'badtemp',
'baffl',
'bag',
'baghdad',
'bahaha',
'bahraini',
'baier',
'bail',
'bailout',
'bake',
'bakeri',
'balanc',
'baldfac',
'baldwin',
'balfour',
'balkan',
'ball',
'ballist',
'balloo',
'ballot',
'balochistan',
'baltic',
'baltimor',
'ban',
'banana',
'bancroft',
'bang',
'banish',
'bank',
'banker',
'bankrupt',
'banner',
'bannon',
'bar',
'barack',
'barag',
'barbar',
'barbara',
'barbarian',
'bare',
'bargain',
'barista',
```

```
'barney',
'barnstorm',
'baron',
'barrag',
'barrel',
'barrier',
'barter',
'base',
'basebal',
'basement',
'bash',
'bashar',
'basi',
'basic',
'basil',
'basket',
'batch',
'bathroom',
'baton',
'batteri',
'battl',
'battleground',
'battleship',
'bauer',
'bay',
'bayh',
'baylor',
'bbc',
'beammeupscotti',
'bean',
'bear',
'beast',
'beat',
'beaten',
'beau',
'beaulieu',
'beauti',
'becam',
'beck',
'beckel',
'becom',
'bee',
'beef',
'beeley',
'beer',
'beg',
'began',
```

```
'begin',
'begun',
'behavior',
'behaviour',
'behead',
'behind',
'bela',
'belarus',
'belgian',
'belgium',
'belief',
'believ',
'beliz',
'bella',
'belliger',
'belong',
'belt',
'beltway',
'ben',
'bend',
'beneath',
'benefit',
'bengaluru',
'benghazi',
'benjamin',
'benni',
'bergdahl',
'berkeley',
'berkley',
'berlin',
'bern',
'bernardino',
'berni',
'berniac',
'bernienom',
'berserk',
'besid',
'best',
'bet',
'betoota',
'betray',
'better',
'bev',
'bevin',
'bewar',
'beyonc',
'beyond',
```

```
'bezo',
'bias',
'bibi',
'bibl',
'biblic',
'bid',
'biden',
'big',
'bigger',
'biggest',
'bigmoney',
'bigot',
'bigotri',
'bike',
'biker',
'bikini',
'bill',
'billari',
'billclinton',
'billion',
'billionair',
'bimbo',
'bin',
'bind',
'biolog',
'biometr',
'bionic',
'bipartisan',
'bird',
'birkenfeld',
'birth',
'birthday',
'birther',
'birthplac',
'birthright',
'bishop',
'bison',
'bit',
'bitch',
'bitcoin',
'bite',
'bitter',
'biz',
'bizarr',
'black',
'blackberri',
'blackhead',
```

```
'blackheart',
'blacklivesmatt',
'blackmail',
'blackout',
'blackwhit',
'blair',
'blake',
'blame',
'blanc',
'blanket',
'blasio',
'blasphem',
'blasphemi',
'blast',
'blatant',
'bleed',
'bless',
'blew',
'blind',
'blindsid',
'blitz',
'blizzard',
'blm',
'blob',
'bloc',
'block',
'blockbust',
'blog',
'blogopen',
'blood',
'bloodbath',
'bloodi',
'bloomberg',
'bloombergback',
'blot',
'bloviat',
'blow',
'blowback',
'blowhard',
'blown',
'blowout',
'blu',
'blue',
'bluebeam',
'bluff',
'blunder',
'blunt',
```

```
'board',
'boast',
'boat',
'boati',
'bob',
'bobbi',
'bodden',
'bode',
'bodi',
'bodyguard',
'boehner',
'boil',
'boko',
'bold',
'bolivian',
'bollywood',
'bolster',
'bomb',
'bombard',
'bomber',
'bombshel',
'bond',
'bone',
'bonfir',
'bonni',
'bonus',
'boo',
'boobi',
'book',
'booker',
'boom',
'boost',
'booth',
'booti',
'booz',
'border',
'bore',
'bori',
'borja',
'born',
'borromeo',
'bos',
'bosanski',
'boston',
'bot',
'botch',
'botox',
```

```
'bottom',
'bottomfish',
'bought',
'boulder',
'boulevard',
'bounc',
'boundless',
'bourgeoi',
'bow',
'bowl',
'box',
'boxer',
'boy',
'boycott',
'boyfriend',
'boyl',
'brace',
'bradley',
'brag',
'brain',
'brainwash',
'braless',
'branch',
'brand',
'brandt',
'brave',
'braverman',
'brawl',
'brazen',
'brazil',
'breach',
'break',
'breakdown',
'breakfast',
'breakingexclus',
'breakneck',
'breakout',
'breakthrough',
'breast',
'breath',
'breathtak',
'breitbart',
'breitbartgravi',
'bret',
'brew',
'breweri',
'brexit',
```

```
'brexittyp',
'brian',
'bribe',
'briberi',
'bric',
'brick',
'brickbat',
'bridg',
'brief',
'briefli',
'brigad',
'bright',
'brighten',
'brilliant',
'brimelow',
'bring',
'brink',
'brinkmanship',
'bristl',
'britain',
'british',
'briton',
'brittl',
'broad',
'broadcast',
'broaden',
'broadest',
'broccoli',
'broil',
'broke',
'broken',
'broker',
'bronx',
'brothel',
'brother',
'brotherhood',
'brotherjohnf',
'brought',
'broward',
'brown',
'bruce',
'bruis',
'brush',
'brussel',
'brutal',
'bryce',
'bubbl',
```

```
'bubl',
'buchanan',
'buck',
'buckley',
'budg',
'budget',
'budgetari',
'budweis',
'buffalo',
'buffett',
'bug',
'build',
'buildup',
'built',
'bulg',
'bull',
'bulldog',
'bullet',
'bulli',
'bullion',
'bum',
'bumbl',
'bumperstick',
'bundi',
'bungl',
'bunk',
'bunni',
'bure',
'bureau',
'burger',
'burglar',
'buri',
'burial',
'burka',
'burn',
'burnout',
'burnt',
'burst',
'bus',
'buse',
'bush',
'busi',
'bust',
'buster',
'butthurt',
'button',
'buy',
```

```
'buzz',
'buzzfe',
'bypass',
'ca',
'cabal',
'cabin',
'cabinet',
'cabl',
'cach',
'cadet',
'cafe',
'cahil',
'cahoot',
'cair',
'caitlyn',
'cake',
'calai',
'calam',
'calcium',
'calcul',
'calendar',
'calif',
'california',
'call',
'calm',
'cam',
'camden',
'came',
'camera',
'cameron',
'camilla',
'camp',
'campaign',
'camper',
'campus',
'canada',
'canadian',
'cancel',
'cancer',
'cancerlink',
'candac',
'candid',
'candidaci',
'cannabi',
'cannib',
'cano',
'canspam',
```

```
 'cant',
 'canter',
 'cap',
 'capabl',
 'capit',
 'capitan',
 'capitol',
 'capston',
 'captiv',
 'captur',
 'car',
 'carbon',
 'carcinogen',
 'card',
 'cardin',
 'care',
 'career',
 'carey',
 'cargo',
 'caricatur',
 'carl',
 'carlo',
 'carlzimm',
 'carmel',
 'carnag',
 'carney',
 'carol',
 'carolina',
 'carri',
 'carrier',
 'carrot',
 'carson',
 'cart',
 'carter',
 'cartoon',
 'carv',
 'case',
 'cash',
 'cashin',
 'cashstrap',
 'cast',
 ...]
```

[58]:
```
df_title_tfidf = pd.DataFrame(feature_matrix.toarray(), columns=tfidf.
 ↪get_feature_names())
```

[59]:
```
df_title_tfidf.head()
```

[59]:
```
   aap  abandon  abbi  abc  abcwapo  abduct  abdullah  abedin  abil  abl  ...  \
0  0.0      0.0   0.0  0.0      0.0     0.0       0.0     0.0   0.0  0.0  ...
1  0.0      0.0   0.0  0.0      0.0     0.0       0.0     0.0   0.0  0.0  ...
2  0.0      0.0   0.0  0.0      0.0     0.0       0.0     0.0   0.0  0.0  ...
3  0.0      0.0   0.0  0.0      0.0     0.0       0.0     0.0   0.0  0.0  ...
4  0.0      0.0   0.0  0.0      0.0     0.0       0.0     0.0   0.0  0.0  ...

   zika  zimbabw  zion  zionist  zip  zone  ztech  zuckerberg  zuess  zulu
0   0.0      0.0   0.0      0.0  0.0   0.0    0.0         0.0    0.0   0.0
1   0.0      0.0   0.0      0.0  0.0   0.0    0.0         0.0    0.0   0.0
2   0.0      0.0   0.0      0.0  0.0   0.0    0.0         0.0    0.0   0.0
3   0.0      0.0   0.0      0.0  0.0   0.0    0.0         0.0    0.0   0.0
4   0.0      0.0   0.0      0.0  0.0   0.0    0.0         0.0    0.0   0.0

[5 rows x 7174 columns]
```

1.3.6 3 f) - Most Frequent Words

Most frequent words in "real" news within the data set:

[60]:
```python
df_real = df.loc[df["label"] == 0]
```

Most frequent words in titles

[61]:
```python
from collections import Counter
Counter(" ".join(df_real["title"]).split()).most_common(10)
```

[61]:
```
[('trump', 634),
 ('clinton', 399),
 ('obama', 293),
 ('gop', 243),
 ('donald', 185),
 ('hillari', 184),
 ('debat', 167),
 ('republican', 163),
 ('new', 140),
 ('say', 138)]
```

Most frequent words in texts

[62]:
```python
Counter(" ".join(df_real["text"]).split()).most_common(10)
```

[62]:
```
[('said', 15046),
 ('trump', 13899),
 ('clinton', 9701),
 ('state', 9368),
 ('would', 7742),
 ('republican', 7679),
 ('presid', 6376),
 ('say', 6336),
 ('one', 6198),
```

```
 ('peopl', 6044)]
```

Most frequent words in “fake” news within the data set:

[63]:
```
df_fake = df.loc[df["label"] == 1]
```

Most frequent words in titles

[64]:
```
Counter(" ".join(df_fake["title"]).split()).most_common(10)
```

[64]:
```
[('trump', 451),
 ('hillari', 397),
 ('clinton', 336),
 ('elect', 211),
 ('u', 200),
 ('new', 138),
 ('russia', 124),
 ('fbi', 122),
 ('video', 122),
 ('america', 115)]
```

Most frequent words in texts

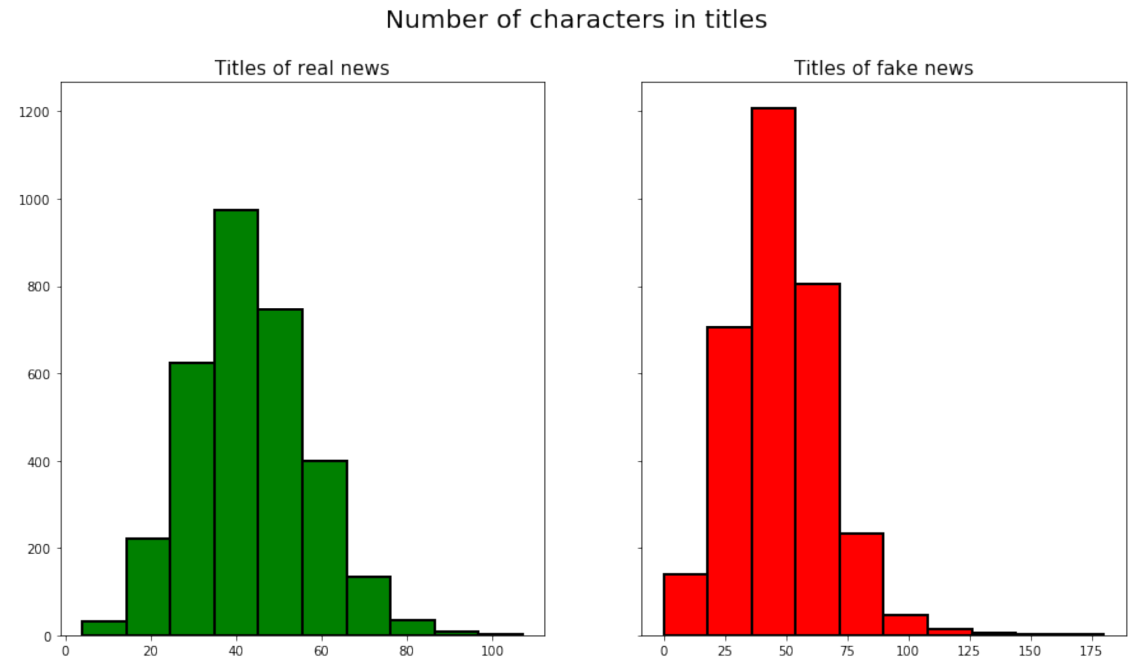
[65]:
```
Counter(" ".join(df_fake["text"]).split()).most_common(10)
```

[65]:
```
[('clinton', 6921),
 ('u', 6850),
 ('trump', 6511),
 ('peopl', 5472),
 ('state', 5401),
 ('one', 5198),
 ('would', 4891),
 ('hillari', 4498),
 ('like', 4098),
 ('elect', 4008)]
```

A plot showing the number of characters in titles of real and fake news will be created.

[66]:
```
plot, (ax1, ax2) = plt.subplots(1, 2, figsize = (15,8), sharey=True)
plot.suptitle('Number of characters in titles', fontsize=20)
length = df[df['label']==0]['title'].str.len()
ax1.hist(length, color = 'green', linewidth = 2, edgecolor = 'black')
ax1.set_title('Titles of real news', fontsize = 15)
length = df[df['label']==1]['title'].str.len()
ax2.hist(length,linewidth = 2, edgecolor = 'black', color = 'red')
ax2.set_title('Titles of fake news', fontsize=15)
```

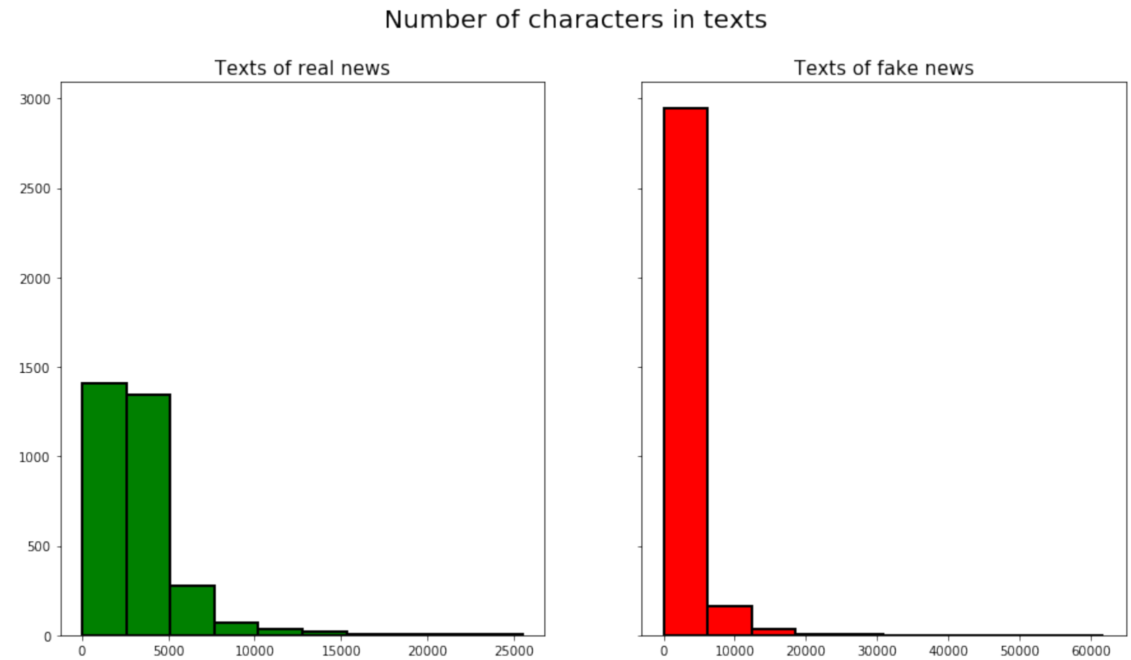
[66]:
```
Text(0.5, 1.0, 'Titles of fake news')
```



A plot showing the number of characters in texts of real and fake news will be created.

[67]:
```
plot, (ax1, ax2) = plt.subplots(1, 2, figsize = (15,8), sharey=True)
plot.suptitle('Number of characters in texts', fontsize=20)
length = df[df['label']==0]['text'].str.len()
ax1.hist(length, color = 'green', linewidth = 2, edgecolor = 'black')
ax1.set_title('Texts of real news', fontsize = 15)
length = df[df['label']==1]['text'].str.len()
ax2.hist(length,linewidth = 2, edgecolor = 'black', color = 'red')
ax2.set_title('Texts of fake news', fontsize=15)
```

[67]:
```
Text(0.5, 1.0, 'Texts of fake news')
```



1.4 Step 4 - Machine Learning / Modeling

[68]:
```
#assign the labels to y to compare them with the predictions made by model
y = df["label"]
```

[69]:
```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
from sklearn.metrics import roc_auc_score

#function that compares different test_size splits and its result on multiple␣
 ↪metric scores
def func_classifier_metrics(classifier):
    start = time.time()
    print("-start-")
    for x in range(1, 10):
        print(x)
        X_train, X_test, y_train, y_test = train_test_split(df_title_tfidf, y,␣
 ↪test_size=x/10, random_state=42, shuffle=True)
        classifier.fit(X_train ,y_train)

        pred_on_test_data = classifier.predict(X_test)
        acc_score = accuracy_score(y_test, pred_on_test_data)
```

```
        prec = precision_score(y_test.values, pred_on_test_data, pos_label=1)
        recall = recall_score(y_test.values, pred_on_test_data)
        f1 = f1_score(y_test.values, pred_on_test_data, average="binary")
        l_loss = log_loss(y_test.values, pred_on_test_data)
        roc_auc = roc_auc_score(y_test.values, pred_on_test_data)

        print("Test size: ", x/10, "| Accuracy: ", "{0:.4f}".format(acc_score),␣
 ↪"| Precision: ", "{0:.4f}".format(prec), "| Recall: ", "{0:.4f}".
 ↪format(recall), "| F1: ", "{0:.4f}".format(f1), "| ROC-AUC: ", "{0:.4f}".
 ↪format(roc_auc), "| Log. Loss: ", "{0:.4f}".format(l_loss))
        x = x + 1
    print("end of loop")
    print("Time: {} mins".format(round((time.time() - start) / 60, 2)))
```

[70]:
```
#function that builds a model with a specific test set size
def func_classifier_metrics_ts(classifier, ts):
    start = time.time()
    print("-start-")
    X_train, X_test, y_train, y_test = train_test_split(df_title_tfidf, y,␣
 ↪test_size=ts, random_state=42, shuffle=True)
    classifier.fit(X_train ,y_train)

    pred_on_test_data = classifier.predict(X_test)
    acc_score = accuracy_score(y_test, pred_on_test_data)
    prec = precision_score(y_test.values, pred_on_test_data, pos_label=1)
    recall = recall_score(y_test.values, pred_on_test_data)
    f1 = f1_score(y_test.values, pred_on_test_data, average="binary")
    l_loss = log_loss(y_test.values, pred_on_test_data)
    roc_auc = roc_auc_score(y_test.values, pred_on_test_data)

    print("Test size: ", ts, "| Accuracy: ", "{0:.4f}".format(acc_score), "|␣
 ↪Precision: ", "{0:.4f}".format(prec), "| Recall: ", "{0:.4f}".
 ↪format(recall), "| F1: ", "{0:.4f}".format(f1), "| ROC-AUC: ", "{0:.4f}".
 ↪format(roc_auc), "| Log. Loss: ", "{0:.4f}".format(l_loss))
    print("Time: {} mins".format(round((time.time() - start) / 60, 2)))
```

1.4.1 4 a) - Naive Bayes Classifier

[71]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB
```

[72]:
```
#Multinomial Naive Bayes
#https://stats.stackexchange.com/questions/33185/
 ↪difference-between-naive-bayes-multinomial-naive-bayes

mNB = MultinomialNB()
```

```
func_classifier_metrics(mNB)
```

```
-start-
1
Test size:  0.1 | Accuracy:  0.8104 | Precision:  0.8419 | Recall:  0.7484 | F1:
0.7924 | ROC-AUC:  0.8084 | Log. Loss:  6.5477
2
Test size:  0.2 | Accuracy:  0.8152 | Precision:  0.8516 | Recall:  0.7524 | F1:
0.7990 | ROC-AUC:  0.8137 | Log. Loss:  6.3840
3
Test size:  0.3 | Accuracy:  0.8110 | Precision:  0.8557 | Recall:  0.7415 | F1:
0.7945 | ROC-AUC:  0.8100 | Log. Loss:  6.5295
4
Test size:  0.4 | Accuracy:  0.7990 | Precision:  0.8303 | Recall:  0.7413 | F1:
0.7833 | ROC-AUC:  0.7979 | Log. Loss:  6.9433
5
Test size:  0.5 | Accuracy:  0.7801 | Precision:  0.8222 | Recall:  0.7114 | F1:
0.7628 | ROC-AUC:  0.7797 | Log. Loss:  7.5953
6
Test size:  0.6 | Accuracy:  0.7722 | Precision:  0.8104 | Recall:  0.7069 | F1:
0.7552 | ROC-AUC:  0.7718 | Log. Loss:  7.8663
7
Test size:  0.7 | Accuracy:  0.7608 | Precision:  0.8095 | Recall:  0.6797 | F1:
0.7389 | ROC-AUC:  0.7605 | Log. Loss:  8.2626
8
Test size:  0.8 | Accuracy:  0.7494 | Precision:  0.7608 | Recall:  0.7217 | F1:
0.7408 | ROC-AUC:  0.7492 | Log. Loss:  8.6552
9
Test size:  0.9 | Accuracy:  0.7267 | Precision:  0.7243 | Recall:  0.7276 | F1:
0.7259 | ROC-AUC:  0.7267 | Log. Loss:  9.4396
end of loop
Time: 0.24 mins
```

[73]:
```
bNB = BernoulliNB()
func_classifier_metrics(bNB)
```

```
-start-
1
Test size:  0.1 | Accuracy:  0.8073 | Precision:  0.8433 | Recall:  0.7386 | F1:
0.7875 | ROC-AUC:  0.8051 | Log. Loss:  6.6568
2
Test size:  0.2 | Accuracy:  0.8049 | Precision:  0.8404 | Recall:  0.7411 | F1:
0.7876 | ROC-AUC:  0.8034 | Log. Loss:  6.7387
3
Test size:  0.3 | Accuracy:  0.8015 | Precision:  0.8516 | Recall:  0.7233 | F1:
0.7822 | ROC-AUC:  0.8004 | Log. Loss:  6.8569
4
```

```
Test size:  0.4 | Accuracy:  0.8049 | Precision:  0.8474 | Recall:  0.7341 | F1:
0.7867 | ROC-AUC:  0.8035 | Log. Loss:  6.7387
5
Test size:  0.5 | Accuracy:  0.7836 | Precision:  0.8343 | Recall:  0.7044 | F1:
0.7639 | ROC-AUC:  0.7831 | Log. Loss:  7.4753
6
Test size:  0.6 | Accuracy:  0.7678 | Precision:  0.8199 | Recall:  0.6826 | F1:
0.7449 | ROC-AUC:  0.7672 | Log. Loss:  8.0209
7
Test size:  0.7 | Accuracy:  0.7585 | Precision:  0.8188 | Recall:  0.6615 | F1:
0.7318 | ROC-AUC:  0.7581 | Log. Loss:  8.3405
8
Test size:  0.8 | Accuracy:  0.7510 | Precision:  0.7667 | Recall:  0.7158 | F1:
0.7404 | ROC-AUC:  0.7507 | Log. Loss:  8.6007
9
Test size:  0.9 | Accuracy:  0.7311 | Precision:  0.6890 | Recall:  0.8373 | F1:
0.7560 | ROC-AUC:  0.7316 | Log. Loss:  9.2881
end of loop
Time: 0.15 mins
```

[74]:
```
gNB = GaussianNB()
func_classifier_metrics(gNB)
```

```
-start-
1
Test size:  0.1 | Accuracy:  0.6872 | Precision:  0.7571 | Recall:  0.5196 | F1:
0.6163 | ROC-AUC:  0.6818 | Log. Loss:  10.8037
2
Test size:  0.2 | Accuracy:  0.6730 | Precision:  0.7383 | Recall:  0.5113 | F1:
0.6042 | ROC-AUC:  0.6692 | Log. Loss:  11.2947
3
Test size:  0.3 | Accuracy:  0.6761 | Precision:  0.7450 | Recall:  0.5214 | F1:
0.6135 | ROC-AUC:  0.6740 | Log. Loss:  11.1856
4
Test size:  0.4 | Accuracy:  0.6817 | Precision:  0.7346 | Recall:  0.5488 | F1:
0.6282 | ROC-AUC:  0.6791 | Log. Loss:  10.9946
5
Test size:  0.5 | Accuracy:  0.6689 | Precision:  0.7193 | Recall:  0.5474 | F1:
0.6217 | ROC-AUC:  0.6682 | Log. Loss:  11.4366
6
Test size:  0.6 | Accuracy:  0.6701 | Precision:  0.7072 | Recall:  0.5734 | F1:
0.6333 | ROC-AUC:  0.6695 | Log. Loss:  11.3948
7
Test size:  0.7 | Accuracy:  0.6660 | Precision:  0.7009 | Recall:  0.5745 | F1:
0.6315 | ROC-AUC:  0.6656 | Log. Loss:  11.5364
8
Test size:  0.8 | Accuracy:  0.6598 | Precision:  0.6820 | Recall:  0.5884 | F1:
```

```
0.6318 | ROC-AUC:  0.6592 | Log. Loss:  11.7518
9
Test size:  0.9 | Accuracy:  0.6405 | Precision:  0.6493 | Recall:  0.6030 | F1:
0.6253 | ROC-AUC:  0.6403 | Log. Loss:  12.4164
end of loop
Time: 0.23 mins
```

Multinomial Naive Bayes seems to provide the best accuracy score with a test size of 0.1 Accuracy: 0.8215

[75]:
```python
#save the model
mNB = MultinomialNB()

X_train, X_test, y_train, y_test = train_test_split(df_title_tfidf, y,␣
 ↪test_size=0.1, random_state=2, shuffle=True)
mNB.fit(X_train ,y_train)
pred_on_test_data = mNB.predict(X_test)
mNB_acc_score = accuracy_score(pred_on_test_data, y_test)
mNB_prec = precision_score(y_test.values, pred_on_test_data, pos_label=1)
mNB_recall = recall_score(y_test.values, pred_on_test_data)
```

[76]:
```python
#save the other models as well!
bNB = BernoulliNB()
X_train, X_test, y_train, y_test = train_test_split(df_title_tfidf, y,␣
 ↪test_size=0.1, random_state=2, shuffle=True)
bNB.fit(X_train ,y_train)
pred_on_test_data = bNB.predict(X_test)
bNB_acc_score = accuracy_score(pred_on_test_data, y_test)
bNB_prec = precision_score(y_test.values, pred_on_test_data, pos_label=1)
bNB_recall = recall_score(y_test.values, pred_on_test_data)
```

[77]:
```python
gNB = GaussianNB()
X_train, X_test, y_train, y_test = train_test_split(df_title_tfidf, y,␣
 ↪test_size=0.1, random_state=2, shuffle=True)
gNB.fit(X_train ,y_train)
pred_on_test_data = gNB.predict(X_test)
gNB_acc_score = accuracy_score(pred_on_test_data, y_test)
gNB_prec = precision_score(y_test.values, pred_on_test_data, pos_label=1)
gNB_recall = recall_score(y_test.values, pred_on_test_data)
```

1.4.2 4 b) - Support Vector Machines

[78]:
```python
from sklearn.svm import SVC
#svc = SVC(gamma='auto', random_state=0)
#svc = SVC(gamma="auto")
svc = SVC(gamma="scale")
```

Google sheet for comparing the different settings and their results
https://docs.google.com/spreadsheets/d/1eSNWea1PujxDQeiwCEZZRcOWYB3lpEsqjekS0HRSUBw/edit#

[79]:
```
#svc = SVC(gamma='scale')
#func_classifier_metrics(svc)
```

[80]:
```
#svc_test = SVC(gamma="scale", C=1.5, kernel="poly", degree=2, coef0=0.001)
```

SVM with gamma=scale, C=1.5, kernel="poly", degree=2, coef0=0.001 and a test size of 0.3 yielded in the highest accuracy: 0.8380

[81]:
```
#save the model
svc = SVC(gamma="scale", C=1.5, kernel="poly", degree=2, coef0=0.001)

X_train, X_test, y_train, y_test = train_test_split(df_title_tfidf, y,
 ↪test_size=0.3, random_state=2, shuffle=True)
svc.fit(X_train ,y_train)
pred_on_test_data = svc.predict(X_test)
svc_acc_score = accuracy_score(pred_on_test_data, y_test)
svc_prec = precision_score(y_test.values, pred_on_test_data, pos_label=1)
svc_recall = recall_score(y_test.values, pred_on_test_data)
```

1.4.3 4 c) - Multi Layer Perceptron

[82]:
```
from sklearn.neural_network import MLPClassifier
```

Google sheet for comparing the different settings and their results
https://docs.google.com/spreadsheets/d/1BBmq1wlc3AzKkBj5wE9EmoaM–XYjovsaVtyVbbHxes/edit#gid=0

[83]:
```
mlp = MLPClassifier(alpha=0.6, learning_rate="invscaling")
func_classifier_metrics(mlp)
```

```
-start-
1
Test size:  0.1 | Accuracy:  0.8215 | Precision:  0.8025 | Recall:  0.8366 | F1:
0.8192 | ROC-AUC:  0.8220 | Log. Loss:  6.1658
2
Test size:  0.2 | Accuracy:  0.8152 | Precision:  0.7909 | Recall:  0.8447 | F1:
0.8169 | ROC-AUC:  0.8158 | Log. Loss:  6.3840
3
Test size:  0.3 | Accuracy:  0.8131 | Precision:  0.7891 | Recall:  0.8472 | F1:
0.8171 | ROC-AUC:  0.8135 | Log. Loss:  6.4568
4
Test size:  0.4 | Accuracy:  0.8085 | Precision:  0.7877 | Recall:  0.8340 | F1:
0.8102 | ROC-AUC:  0.8089 | Log. Loss:  6.6159
5
Test size:  0.5 | Accuracy:  0.7949 | Precision:  0.7780 | Recall:  0.8220 | F1:
0.7994 | ROC-AUC:  0.7951 | Log. Loss:  7.0825
6
Test size:  0.6 | Accuracy:  0.7830 | Precision:  0.7596 | Recall:  0.8241 | F1:
0.7905 | ROC-AUC:  0.7833 | Log. Loss:  7.4935
```

```
7
Test size:  0.7 | Accuracy:  0.7709 | Precision:  0.7589 | Recall:  0.7916 | F1:
0.7749 | ROC-AUC:  0.7710 | Log. Loss:  7.9118
8

D:\Anaconda\lib\site-
packages\sklearn\neural_network\multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Test size:  0.8 | Accuracy:  0.7617 | Precision:  0.7355 | Recall:  0.8113 | F1:
0.7715 | ROC-AUC:  0.7620 | Log. Loss:  8.2324
9

D:\Anaconda\lib\site-
packages\sklearn\neural_network\multilayer_perceptron.py:566:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)

Test size:  0.9 | Accuracy:  0.7299 | Precision:  0.6832 | Recall:  0.8522 | F1:
0.7584 | ROC-AUC:  0.7305 | Log. Loss:  9.3305
end of loop
Time: 22.41 mins
```

MLP with alpha=0.6, learning_rate="invscaling" and a test size of 0.25 yielded in the highest accuracy: 0.8258

[84]:
```
#save the model
mlp = MLPClassifier(alpha=0.6, learning_rate="invscaling")

X_train, X_test, y_train, y_test = train_test_split(df_title_tfidf, y,␣
 ↪test_size=0.25, random_state=2, shuffle=True)
mlp.fit(X_train ,y_train)
pred_on_test_data = mlp.predict(X_test)
mlp_acc_score = accuracy_score(pred_on_test_data, y_test)
mlp_prec = precision_score(y_test.values, pred_on_test_data, pos_label=1)
mlp_recall = recall_score(y_test.values, pred_on_test_data)
```

1.4.4 4 c) - Random Forest

[85]:
```
from sklearn.ensemble import RandomForestClassifier
```

Google sheet for comparing the different settings and their results
https://docs.google.com/spreadsheets/d/1Etjo4qA_P7Ko148z329JFzbxBC1ITmxGhsnD9OzfYBE/edit#gid=

[86]:
```
rfc = RandomForestClassifier()
func_classifier_metrics(rfc)
```

```
-start-
1

D:\Anaconda\lib\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: 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)

Test size:  0.1 | Accuracy:  0.8009 | Precision:  0.7922 | Recall:  0.7974 | F1:
0.7948 | ROC-AUC:  0.8008 | Log. Loss:  6.8751
2
Test size:  0.2 | Accuracy:  0.7899 | Precision:  0.7848 | Recall:  0.7848 | F1:
0.7848 | ROC-AUC:  0.7898 | Log. Loss:  7.2570
3
Test size:  0.3 | Accuracy:  0.7783 | Precision:  0.7820 | Recall:  0.7628 | F1:
0.7723 | ROC-AUC:  0.7781 | Log. Loss:  7.6572
4
Test size:  0.4 | Accuracy:  0.7867 | Precision:  0.7732 | Recall:  0.7994 | F1:
0.7861 | ROC-AUC:  0.7870 | Log. Loss:  7.3662
5
Test size:  0.5 | Accuracy:  0.7757 | Precision:  0.7636 | Recall:  0.7947 | F1:
0.7788 | ROC-AUC:  0.7758 | Log. Loss:  7.7481
6
Test size:  0.6 | Accuracy:  0.7562 | Precision:  0.7318 | Recall:  0.8039 | F1:
0.7662 | ROC-AUC:  0.7565 | Log. Loss:  8.4211
7
Test size:  0.7 | Accuracy:  0.7556 | Precision:  0.7412 | Recall:  0.7825 | F1:
0.7613 | ROC-AUC:  0.7557 | Log. Loss:  8.4419
8
Test size:  0.8 | Accuracy:  0.7336 | Precision:  0.7100 | Recall:  0.7826 | F1:
0.7446 | ROC-AUC:  0.7340 | Log. Loss:  9.2009
9
Test size:  0.9 | Accuracy:  0.6981 | Precision:  0.6628 | Recall:  0.8003 | F1:
0.7251 | ROC-AUC:  0.6986 | Log. Loss:  10.4279
end of loop
Time: 0.53 mins
```

[87]:
```
rfc = RandomForestClassifier(n_estimators=500)
func_classifier_metrics_ts(rfc, 0.25)
```

```
-start-
Test size:  0.25 | Accuracy:  0.8136 | Precision:  0.7860 | Recall:  0.8518 |
F1:  0.8176 | ROC-AUC:  0.8144 | Log. Loss:  6.4366
Time: 3.66 mins
```

RFC with n_estimators=500 and a test size of 0.25 yielded in the highest accuracy: 0.8112.

[88]:
```
#save the model
rfc = RandomForestClassifier(n_estimators=500)

X_train, X_test, y_train, y_test = train_test_split(df_title_tfidf, y,␣
 ↪test_size=0.25, random_state=2, shuffle=True)
rfc.fit(X_train ,y_train)
pred_on_test_data = rfc.predict(X_test)
rfc_acc_score = accuracy_score(pred_on_test_data, y_test)
rfc_prec = precision_score(y_test.values, pred_on_test_data, pos_label=1)
rfc_recall = recall_score(y_test.values, pred_on_test_data)
```

[89]:
```
#from sklearn.linear_model import LogisticRegression
#lr = LogisticRegression()
#func_classifier_metrics_ts(lr, 0.33)
```

1.5 5 - Prediction

Now the previously deleted articles will be added to a new dataframe. After that the different models which have been built will make predictions on whether they are real or fake news.

[90]:
```
pred_data = {'index':  [5099, 4988, 534, 4777, 2734, 2058],
             'title': ['smell hillari fear', 'poll find american support polic␣
 ↪highest near year','battl new york primari matter', 'uk announc new troop␣
 ↪deploy near russia border', 'russia join franc strike isi stronghold syria',␣
 ↪'shortag lethal inject drug put death penalti suprem court'],
```

```
             'label': [1, 1, 0, 1, 0, 0],
            }
```

[91]:
```
pred_tfidf_df = tfidf.transform(pred_data["title"])
#prediction = mNB.predict(pred_tfidf_df)
```

1.5.1 5 a) Multinomial Naive Bayes

[92]:
```
mNB = MultinomialNB()
func_classifier_metrics_ts(mNB, 0.2)
```

```
-start-
Test size:  0.2 | Accuracy:  0.8152 | Precision:  0.8516 | Recall:  0.7524 | F1:
0.7990 | ROC-AUC:  0.8137 | Log. Loss:  6.3840
Time: 0.04 mins
```

[93]:
```
#mNB.predict_proba(pred_tfidf_df)
```

[94]:
```
mNB_pred = mNB.predict(pred_tfidf_df)

mNB_pred_acc = accuracy_score(pred_data["label"], mNB_pred)
mNB_pred_prec = precision_score(pred_data["label"], mNB_pred)
mNB_pred_recall = recall_score(pred_data["label"], mNB_pred)
mNB_pred_f1 = f1_score(pred_data["label"], mNB_pred, average="binary")
mNB_pred_l_loss = log_loss(pred_data["label"], mNB_pred)
mNB_pred_roc_auc = roc_auc_score(pred_data["label"], mNB_pred)

print("Accuracy: ", "{0:.4f}".format(mNB_pred_acc), "| Precision: ", "{0:.4f}".
 ↪format(mNB_pred_prec), "| Recall: ", "{0:.4f}".format(mNB_pred_recall), "|␣
 ↪F1: ", "{0:.4f}".format(mNB_pred_f1), "| ROC-AUC: ", "{0:.4f}".
 ↪format(mNB_pred_roc_auc), "| Log. Loss: ", "{0:.4f}".format(mNB_pred_l_loss))
```

```
Accuracy:  0.6667 | Precision:  0.6667 | Recall:  0.6667 | F1:  0.6667 | ROC-
AUC:  0.6667 | Log. Loss:  11.5131
```

[95]:
```
mNB_pred
```

[95]:
```
array([0, 1, 0, 1, 1, 0], dtype=int64)
```

1.5.2 5 b) Gaussian Naive Bayes

[96]:
```
gNB = GaussianNB()
func_classifier_metrics_ts(gNB, 0.33)
```

```
-start-
Test size:  0.33 | Accuracy:  0.6812 | Precision:  0.7452 | Recall:  0.5331 |
F1:  0.6216 | ROC-AUC:  0.6786 | Log. Loss:  11.0115
Time: 0.05 mins
```

[97]:
```
gNB_pred = gNB.predict(pred_tfidf_df.toarray())

gNB_pred_acc = accuracy_score(pred_data["label"], gNB_pred)
gNB_pred_prec = precision_score(pred_data["label"], gNB_pred)
gNB_pred_recall = recall_score(pred_data["label"], gNB_pred)
gNB_pred_f1 = f1_score(pred_data["label"], gNB_pred, average="binary")
gNB_pred_l_loss = log_loss(pred_data["label"], gNB_pred)
gNB_pred_roc_auc = roc_auc_score(pred_data["label"], gNB_pred)

print("Accuracy: ", "{0:.4f}".format(gNB_pred_acc), "| Precision: ", "{0:.4f}".
 ↪format(gNB_pred_prec), "| Recall: ", "{0:.4f}".format(gNB_pred_recall), "|␣
 ↪F1: ", "{0:.4f}".format(gNB_pred_f1), "| ROC-AUC: ", "{0:.4f}".
 ↪format(gNB_pred_roc_auc), "| Log. Loss: ", "{0:.4f}".format(gNB_pred_l_loss))
```

```
Accuracy:  0.8333 | Precision:  1.0000 | Recall:  0.6667 | F1:  0.8000 | ROC-
AUC:  0.8333 | Log. Loss:  5.7565
```

[98]:
```
gNB_pred
```

[98]:
```
array([1, 1, 0, 0, 0, 0], dtype=int64)
```

1.5.3 5 c) Bernoulli Naive Bayes

[99]:
```
bNB = BernoulliNB()
func_classifier_metrics_ts(bNB, 0.33)
```

```
-start-
Test size:  0.33 | Accuracy:  0.8013 | Precision:  0.8452 | Recall:  0.7290 |
F1:  0.7828 | ROC-AUC:  0.8001 | Log. Loss:  6.8615
Time: 0.02 mins
```

[100]:
```
bNB_pred = bNB.predict(pred_tfidf_df.toarray())

bNB_pred_acc = accuracy_score(pred_data["label"], bNB_pred)
bNB_pred_prec = precision_score(pred_data["label"], bNB_pred)
bNB_pred_recall = recall_score(pred_data["label"], bNB_pred)
bNB_pred_f1 = f1_score(pred_data["label"], bNB_pred, average="binary")
bNB_pred_l_loss = log_loss(pred_data["label"], bNB_pred)
bNB_pred_roc_auc = roc_auc_score(pred_data["label"], bNB_pred)

print("Accuracy: ", "{0:.4f}".format(bNB_pred_acc), "| Precision: ", "{0:.4f}".
 ↪format(bNB_pred_prec), "| Recall: ", "{0:.4f}".format(bNB_pred_recall), "|␣
 ↪F1: ", "{0:.4f}".format(bNB_pred_f1), "| ROC-AUC: ", "{0:.4f}".
 ↪format(bNB_pred_roc_auc), "| Log. Loss: ", "{0:.4f}".format(bNB_pred_l_loss))
```

```
Accuracy:  0.8333 | Precision:  0.7500 | Recall:  1.0000 | F1:  0.8571 | ROC-
AUC:  0.8333 | Log. Loss:  5.7566
```

[101]:
```
bNB_pred
```

[101]:
```
array([1, 1, 0, 1, 1, 0], dtype=int64)
```

1.5.4 5 d) Support Vector Machine

[102]:
```
svc_pred = SVC(gamma="scale", C=1.5, kernel="poly", degree=2, coef0=0.001)
func_classifier_metrics_ts(svc_pred, 0.25)
```

```
-start-
Test size:  0.25 | Accuracy:  0.8484 | Precision:  0.8213 | Recall:  0.8827 |
F1:  0.8509 | ROC-AUC:  0.8490 | Log. Loss:  5.2365
Time: 4.39 mins
```

[103]:
```
svc_pred = svc_pred.predict(pred_tfidf_df.toarray())

svc_pred_acc = accuracy_score(pred_data["label"], svc_pred)
svc_pred_prec = precision_score(pred_data["label"], svc_pred)
svc_pred_recall = recall_score(pred_data["label"], svc_pred)
svc_pred_f1 = f1_score(pred_data["label"], svc_pred, average="binary")
svc_pred_l_loss = log_loss(pred_data["label"], svc_pred)
svc_pred_roc_auc = roc_auc_score(pred_data["label"], svc_pred)

print("Accuracy: ", "{0:.4f}".format(svc_pred_acc), "| Precision: ", "{0:.4f}".
 ↪format(svc_pred_prec), "| Recall: ", "{0:.4f}".format(svc_pred_recall), "|␣
 ↪F1: ", "{0:.4f}".format(svc_pred_f1), "| ROC-AUC: ", "{0:.4f}".
 ↪format(svc_pred_roc_auc), "| Log. Loss: ", "{0:.4f}".format(svc_pred_l_loss))
```

```
Accuracy:  0.6667 | Precision:  0.6667 | Recall:  0.6667 | F1:  0.6667 | ROC-
AUC:  0.6667 | Log. Loss:  11.5131
```

[104]:
```
svc_pred
```

[104]:
```
array([1, 0, 0, 1, 1, 0], dtype=int64)
```

1.5.5 5 e) Multilayer Perceptron

[105]:
```
mlp_pred = MLPClassifier(alpha=0.6, learning_rate="invscaling")
func_classifier_metrics_ts(mlp_pred, 0.2)
```

```
-start-
Test size:  0.2 | Accuracy:  0.8191 | Precision:  0.8092 | Recall:  0.8236 | F1:
0.8164 | ROC-AUC:  0.8192 | Log. Loss:  6.2476
Time: 3.79 mins
```

[106]:
```
mlp_pred = mlp_pred.predict(pred_tfidf_df)

mlp_pred_acc = accuracy_score(pred_data["label"], mlp_pred)
mlp_pred_prec = precision_score(pred_data["label"], mlp_pred)
mlp_pred_recall = recall_score(pred_data["label"], mlp_pred)
mlp_pred_f1 = f1_score(pred_data["label"], mlp_pred, average="binary")
mlp_pred_l_loss = log_loss(pred_data["label"], mlp_pred)
mlp_pred_roc_auc = roc_auc_score(pred_data["label"], mlp_pred)

print("Accuracy: ", "{0:.4f}".format(mlp_pred_acc), "| Precision: ", "{0:.4f}".
 ↪format(mlp_pred_prec), "| Recall: ", "{0:.4f}".format(mlp_pred_recall), "|␣
 ↪F1: ", "{0:.4f}".format(mlp_pred_f1), "| ROC-AUC: ", "{0:.4f}".
 ↪format(mlp_pred_roc_auc), "| Log. Loss: ", "{0:.4f}".format(mlp_pred_l_loss))
```

```
Accuracy:  0.6667 | Precision:  0.6667 | Recall:  0.6667 | F1:  0.6667 | ROC-
AUC:  0.6667 | Log. Loss:  11.5131
```

[107]:
```
mlp_pred
```

[107]:
```
array([1, 0, 0, 1, 1, 0], dtype=int64)
```

1.5.6 5 f) Random Forest

[108]:
```
rfc_pred = RandomForestClassifier(n_estimators=500)
func_classifier_metrics_ts(rfc_pred, 0.25)
```

```
-start-
Test size:  0.25 | Accuracy:  0.8168 | Precision:  0.7879 | Recall:  0.8570 |
F1:  0.8210 | ROC-AUC:  0.8176 | Log. Loss:  6.3275
Time: 3.75 mins
```

[109]:
```
rfc_pred = rfc_pred.predict(pred_tfidf_df)

rfc_pred_acc = accuracy_score(pred_data["label"], rfc_pred)
rfc_pred_prec = precision_score(pred_data["label"], rfc_pred)
rfc_pred_recall = recall_score(pred_data["label"], rfc_pred)
rfc_pred_f1 = f1_score(pred_data["label"], rfc_pred, average="binary")
rfc_pred_l_loss = log_loss(pred_data["label"], rfc_pred)
rfc_pred_roc_auc = roc_auc_score(pred_data["label"], rfc_pred)

print("Accuracy: ", "{0:.4f}".format(rfc_pred_acc), "| Precision: ", "{0:.4f}".
 ↪format(rfc_pred_prec), "| Recall: ", "{0:.4f}".format(rfc_pred_recall), "|␣
 ↪F1: ", "{0:.4f}".format(rfc_pred_f1), "| ROC-AUC: ", "{0:.4f}".
 ↪format(rfc_pred_roc_auc), "| Log. Loss: ", "{0:.4f}".format(rfc_pred_l_loss))
```

```
Accuracy:  0.8333 | Precision:  1.0000 | Recall:  0.6667 | F1:  0.8000 | ROC-
AUC:  0.8333 | Log. Loss:  5.7565
```

[110]:
```
rfc_pred
```

[110]:
```
array([1, 0, 0, 1, 0, 0], dtype=int64)
```

1.5.7 5 g) Calculating KPIs of Classification Performance by Survey Participants

[111]:
```
# 0.3780487805         0.8536585366         0.4329268293         0.
↪1829268293         0.3536585366         0.3719512195
humans_pred = np.array([0, 1, 0, 0, 0, 0])
humans_pred
```

[111]:
```
array([0, 1, 0, 0, 0, 0])
```

[112]:
```
humans_pred_acc = accuracy_score(pred_data["label"], humans_pred)
humans_pred_prec = precision_score(pred_data["label"], humans_pred)
humans_pred_recall = recall_score(pred_data["label"], humans_pred)
humans_pred_f1 = f1_score(pred_data["label"], humans_pred, average="binary")
humans_pred_l_loss = log_loss(pred_data["label"], humans_pred)
humans_pred_roc_auc = roc_auc_score(pred_data["label"], humans_pred)

print("Accuracy: ", "{0:.4f}".format(humans_pred_acc), "| Precision: ", "{0:.
↪4f}".format(humans_pred_prec), "| Recall: ", "{0:.4f}".
↪format(humans_pred_recall), "| F1: ", "{0:.4f}".format(humans_pred_f1), "|␣
↪ROC-AUC: ", "{0:.4f}".format(humans_pred_roc_auc), "| Log. Loss: ", "{0:.
↪4f}".format(humans_pred_l_loss))
```

```
Accuracy:  0.6667 | Precision:  1.0000 | Recall:  0.3333 | F1:  0.5000 | ROC-
AUC:  0.6667 | Log. Loss:  11.5129
```

1.5.8 5 h) Graphs Comparing ML Models' Predictions

[113]:
```
acc_labels={'Multinomial Naive Bayes':mNB_pred_acc,
            'Gaussian Naive Bayes':gNB_pred_acc,
            'Bernoulli Naive Bayes':bNB_pred_acc,
            'Support Vector Machines':svc_pred_acc,
            'Multilayer Perceptron':mlp_pred_acc,
            'Random Forest Classifier':rfc_pred_acc
            }
```

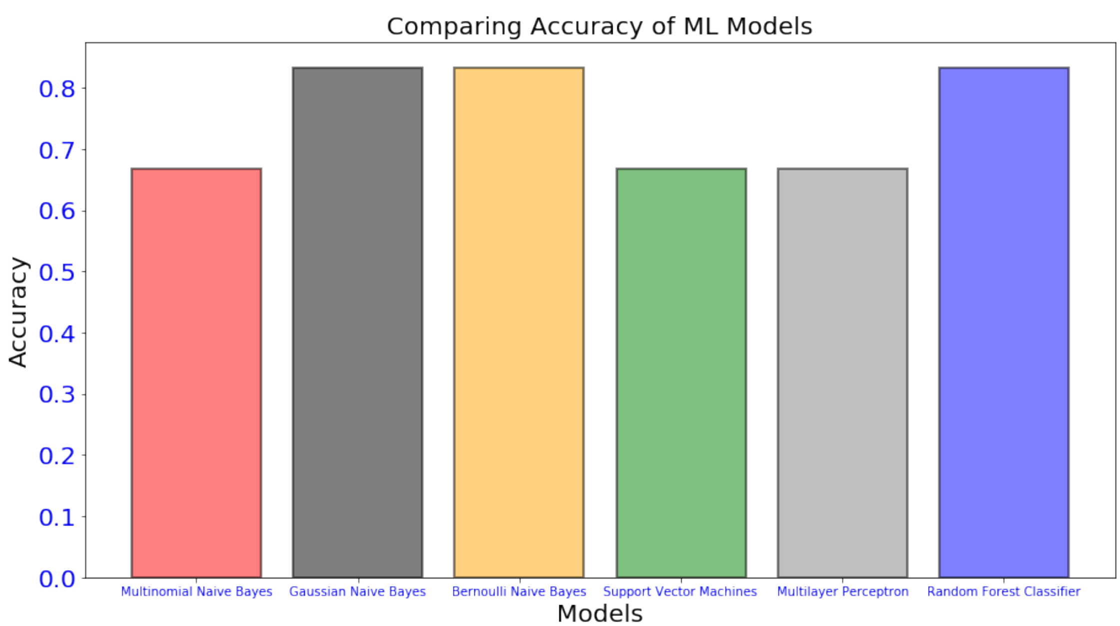
[114]:
```
precision_labels={'Multinomial Naive Bayes':mNB_pred_prec,
                  'Gaussian Naive Bayes':gNB_pred_prec,
                  'Bernoulli Naive Bayes':bNB_pred_prec,
                  'Support Vector Machines':svc_pred_prec,
                  'Multilayer Perceptron':mlp_pred_prec,
                  'Random Forest Classifier':rfc_pred_prec
                  }
```

[115]:
```
recall_labels={'Multinomial Naive Bayes':mNB_pred_recall,
               'Gaussian Naive Bayes':gNB_pred_recall,
```

```
                'Bernoulli Naive Bayes':bNB_pred_recall,
                'Support Vector Machines':svc_pred_recall,
                'Multilayer Perceptron':mlp_pred_recall,
                'Random Forest Classifier':rfc_pred_recall
                 }
```

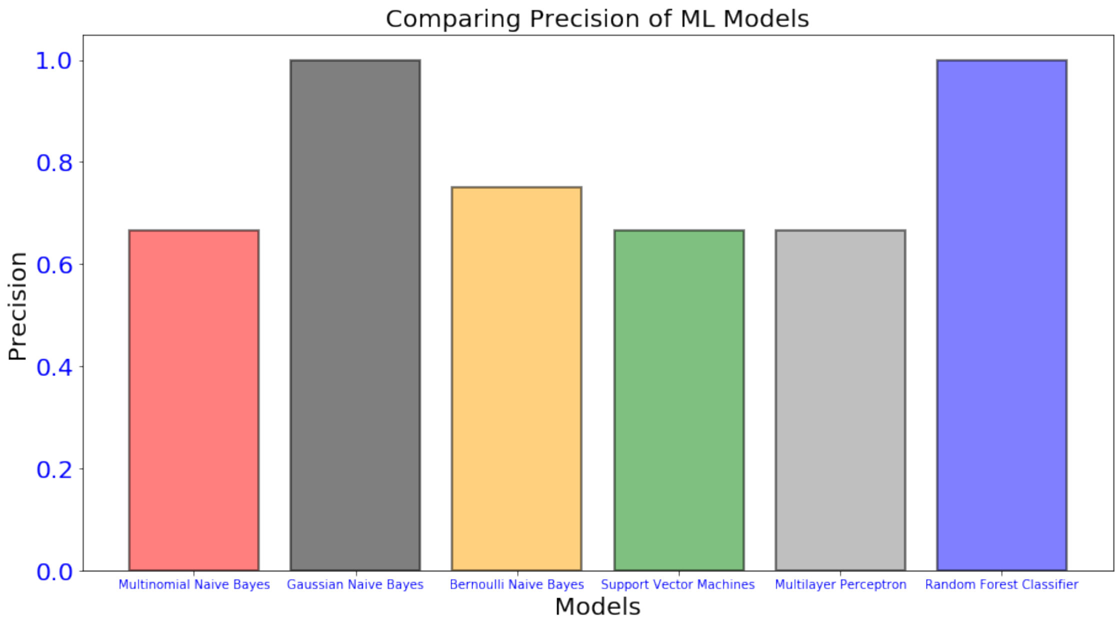
[116]:
```
plt.figure(figsize=(15,8))
plt.title('Comparing Accuracy of ML Models',fontsize=20)
colors=['red','black','orange','green','grey','blue']
plt.xticks(fontsize=10,color='blue')
plt.yticks(fontsize=20,color='blue')
plt.ylabel('Accuracy',fontsize=20)
plt.xlabel('Models',fontsize=20)
plt.bar(acc_labels.keys(),acc_labels.values(), edgecolor='black', color=colors,␣
 ↪linewidth=2,alpha=0.5)
```

[116]: <BarContainer object of 6 artists>



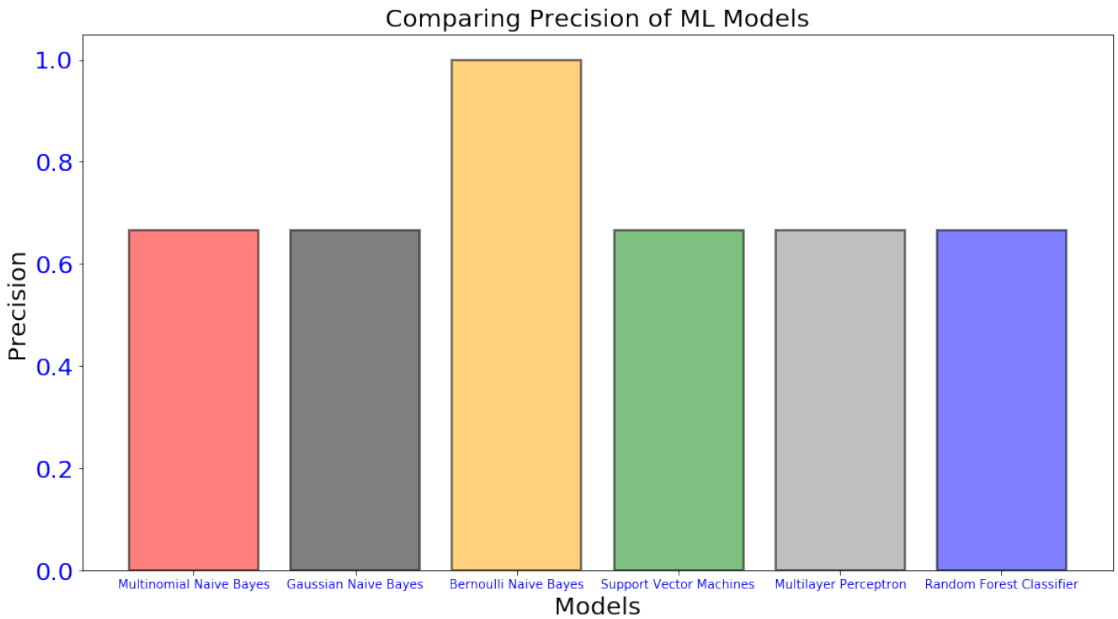
[117]:
```
plt.figure(figsize=(15,8))
plt.title('Comparing Precision of ML Models',fontsize=20)
colors=['red','black','orange','green','grey','blue']
plt.xticks(fontsize=10,color='blue')
plt.yticks(fontsize=20,color='blue')
plt.ylabel('Precision',fontsize=20)
plt.xlabel('Models',fontsize=20)
plt.bar(precision_labels.keys(),precision_labels.values(), edgecolor='black',␣
 ↪color=colors, linewidth=2,alpha=0.5)
```

[117]: <BarContainer object of 6 artists>



[118]:
```
plt.figure(figsize=(15,8))
plt.title('Comparing Precision of ML Models',fontsize=20)
colors=['red','black','orange','green','grey','blue']
plt.xticks(fontsize=10,color='blue')
plt.yticks(fontsize=20,color='blue')
plt.ylabel('Precision',fontsize=20)
plt.xlabel('Models',fontsize=20)
plt.bar(recall_labels.keys(),recall_labels.values(), edgecolor='black',␣
↪color=colors, linewidth=2,alpha=0.5)
```

[118]: <BarContainer object of 6 artists>



Since this data set is well balanced, accuracy can be perceived as a reliable metric.