Machine Learning Assignment: Fake News Classifier

Group Members:

Philani Mpofu (1848751)
Chloë Smith (1877342)
Matthew Kruger (1669326)
Jesse Bristow (1875955)

(1) The chosen dataset for this assignment was taken from kaggle.com (Fake News | Kaggle, 2020). The dataset, which consists of 20 800 news articles from various different sites and authors, has the following attributes: *ID, Title, Author, Text, and Label*. In this case, every individual article is a single datapoint. The attribute 'Label' refers to the binary target class that takes on values 0 or 1. Articles labelled as 0 are confirmed reliable sources ('real news'), and articles labelled as 1 are confirmed unreliable sources ('fake' news). Some articles may be incomplete. These attributes make up the standard format for any news article.

The main aim throughout this assignment was to determine if an article's reliability has some corporeal aspect to it that is somehow embedded in the basal word choice; the assumption being that 'fake' and 'real' news must have some specific word choice for their titles and texts that make them lean to one extreme side on the reliability spectrum. Also note that an author's name is not considered as relevant in this assignment, as judging by an author's name would be superficial and not very meaningful. See below a few examples of the articles (datapoints) in the chosen dataset, all of which follows the above-mentioned format *[ID, Title, Author, Text, and Label]*:

(i)

1
'FLYNN: Hillary Clinton, Big Woman on Campus - Breitbart'
'Daniel J. Flynn'
'Ever get the feeling your life circles the roundabout rather than heads in a straight line toward the intended destination? [Hillary Clinton remains the big woman on campus in leafy, liberal Wellesley, Massachusetts. Everywhere else votes her most likely to don her inauguration dress for the remainder of her days the way Miss Havisham forever wore that wedding dress. Speaking of Great Expectations, Hillary Rodham overflowed with them 48 years ago when she first addressed a Wellesley graduating class. The president of the college informed those gathered in 1969 that the students needed "no debate so far as I could ascertain as to who their spokesman was to be" (kind of the like the Democratic primaries in 2016 minus the terms unknown then even at a Seven Sisters school). "I am very glad that Miss Adams made it clear that what I am speaking for today is all of us — the 400 of us," Miss Rodham told her classmates. After appointing herself Edger Bergen to the Charlie McCarthys and Mortimer Snerds in attendance, the bespectacled in granny glasses (awarding her matronly wisdom — or at least John Lennon wisdom) took issue with the previous speaker. Despite becoming the first to win election to a seat in the U. S. Senate since Reconstruction, Edward Brooke came in for criticism for calling for "empathy" for the goals of protestors as he criticized tactics. Though Clinton in her senior thesis on Saul Alinsky lamented "Black Power demagogues" and "elitist arrogance and repressive intolerance" within the New Left,

similar words coming out of a Republican necessitated a brief rebuttal. “Trust,” Rodham ironically observed in 1969, “this is one word that when I asked the class at our rehearsal what it was they wanted me to say for them, everyone came up to me and said ‘Talk about trust, talk about the lack of trust both for us and the way we feel about others. Talk about the trust bust.’ What can you say about it? What can you say about a feeling that permeates a generation and that perhaps is not even understood by those who are distrusted?” The “trust bust” certainly busted Clinton’s 2016 plans. She certainly did not even understand that people distrusted her. After Whitewater, Travelgate, the vast conspiracy, Benghazi, and the missing emails, Clinton found herself the distrusted voice on Friday. There was a load of compromising on the road to the broadening of her political horizons. And distrust from the American people — Trump edged her 48 percent to 38 percent on the question immediately prior to November’s election — stood as a major reason for the closing of those horizons. Clinton described her vanquisher and his supporters as embracing a “lie,” a “con,” “alternative facts,” and “a assault on truth and reason. ” She failed to explain why the American people chose his lies over her truth. “As the history majors among you here today know all too well, when people in power invent their own facts and attack those who question them, it can mark the beginning of the end of a free society,” she offered. “That is not hyperbole. ” Like so many people to emerge from the 1960s, Hillary Clinton embarked upon a long, strange trip. From high school Goldwater Girl and Wellesley College Republican president to Democratic politician, Clinton drank in the times and the place that gave her a degree. More significantly, she went from idealist to cynic, as a comparison of her two Wellesley commencement addresses show. Way back when, she lamented that “for too long our leaders have viewed politics as the art of the possible, and the challenge now is to practice politics as the art of making what appears to be impossible possible. ” Now, as the big woman on campus but the odd woman out of the White House, she wonders how her current station is even possible. “Why aren’t I 50 points ahead?” she asked in September. In May she asks why she isn’t president. The woman famously dubbed a “congenital liar” by Bill Safire concludes that lies did her in — theirs, mind you, not hers. Getting stood up on Election Day, like finding yourself the jilted bride on your wedding day, inspires dangerous delusions.'
0

(ii)

7

'Benoît Hamon Wins French Socialist Party's Presidential Nomination - The New York Times'
'Alissa J. Rubin'

'PARIS — France chose an idealistic, traditional candidate in Sunday's primary to represent the Socialist and parties in the presidential election this spring. The candidate, Benoît Hamon, 49, who ran on the slogan that he would "make France's heart beat," bested Manuel Valls, the former prime minister, whose campaign has promoted more policies and who has a strong background. Mr. Hamon appeared to have won by a wide margin, with incomplete returns showing him with an estimated 58 percent of the vote to Mr. Valls's 41 percent. "Tonight the left holds its head up high again it is looking to the future," Mr. Hamon said, addressing his supporters. "Our country needs the left, but a modern, innovative left," he said. Mr. Hamon's victory was the clearest sign yet that voters on the left want a break with the policies of President François Hollande, who in December announced that he would not seek . However, Mr. Hamon's strong showing is unlikely to change widespread assessments that candidates have little chance of making it into the second round of voting in the general election. The first round of the general election is set for April 23 and the runoff for May 7. The Socialist Party is deeply divided, and one measure of its lack of popular enthusiasm was the relatively low number of people voting. About two million people voted in the second round of the primary on Sunday, in contrast with about 2. 9 million in the second round of the last presidential primary on the left, in 2011. However, much of the conventional wisdom over how the elections will go has been thrown into question over the past week, because the leading candidate, François Fillon, who represents the main party, the Republicans, was accused of paying his wife large sums of money to work as his parliamentary aide. While nepotism is legal in the French political system, it is not clear that she actually did any work. Prosecutors who specialize in financial malfeasance are reviewing the case. France's electoral system allows multiple candidates to run for president in the first round of voting, but only the top two go on to a second round. Mr. Hamon is entering a race that is already crowded on the left, with candidates who include Mélenchon on the far left, and Emmanuel Macron, an independent who served as economy minister in Mr. Hollande's government and who embraces more policies. Unless he decides to withdraw, Mr. Fillon, the mainstream right candidate, will also run, as will the extreme right candidate Marine Le Pen. The two have been expected to go to the runoff. Mr. Hamon's victory can be attributed at least in part to his image as an idealist and traditional leftist candidate who appeals to union voters as well as more environmentally concerned and socially liberal young people. Unlike Mr. Valls, he also clearly distanced himself from some of Mr. Hollande's more unpopular policies, especially the economic ones. Thomas Kekenbosch, 22, a student and one of the leaders of the group the

Youth With Benoît Hamon, said Mr. Hamon embodied a new hope for those on the left. “We have a perspective we have something to do, to build,” Mr. Kekenbosch said. Mr. Hollande had disappointed many young people because under him the party abandoned ideals, such as support for workers, that many voters believe in, according to Mr. Kekenbosch. Mr. Hollande’s government, under pressure from the European Union to meet budget restraints, struggled to pass labor code reforms to make the market more attractive to foreign investors and also to encourage French businesses to expand in France. The measures ultimately passed after weeks of strikes, but they were watered down and generated little concrete progress in improving France’s roughly 10 percent unemployment rate and its nearly 25 percent youth joblessness rate. Mr. Hamon strongly endorses a stimulus approach to improving the economy and has promised to phase in a universal income, which would especially help young people looking for work, but would also supplement the livelihood of French workers. The end goal would be to have everyone receive 750 euros per month (about $840). “We have someone that trusts us,” Mr. Kekenbosch said, “who says: ‘I give you enough to pay for your studies. You can have a scholarship which spares you from working at McDonald’s on provisional contracts for 4 years. ” Mr. Hamon advocates phasing out diesel fuel and encouraging drivers to replace vehicles that use petroleum products with electrical ones. His leftist pedigree began early. His father worked at an arsenal in Brest, a city in the far west of Brittany, and his mother worked off and on as a secretary. He was an early member of the Movement of Young Socialists, and he has continued to work closely with them through his political life. He also worked for Martine Aubry, now the mayor of Lille and a former Socialist Party leader.'

0

(iii)

143

'U.N. Secretary General Complains That The ‘Masses Have Rejected Globalism’ In Favor Of Nationalism'

'Geoffrey Grider'

'U.N. Secretary General Complains That The ‘Masses Have Rejected Globalism’ In Favor Of Nationalism Antonio Guterres, elected in October to take over as U.N. secretary general next year, told a conference in his native Lisbon that this trend had undermined the willingness to receive refugees in Europe this year. He said the world must re-establish international protection for refugees coming from war zones such as Syria, but it would not be easy as developed countries were turning to nationalist agendas. 22, 2016 The incoming head of the United Nations warned on Tuesday that ‘losers of globalization’ in rich countries have felt ignored by establishment politicians, prompting them to turn to nationalist agendas, as in the U.S. election and Brexit referendum. “Therefore wait ye upon me, saith the LORD, until the

day that I rise up to the prey: for my determination is to gather the nations, that I may assemble the kingdoms, to pour upon them mine indignation, even all my fierce anger: for all the earth shall be devoured with the fire of my jealousy." Zephaniah 3:8 (KJV) EDITOR'S NOTE: The Bible clearly says that God's desire is to "gather all the nations of the world together", in order to "pour His fury" upon them. The United Nations, something unique in world history, has been created by the will of God, and things are going to end exactly how the Bible says they will end. All Muslims will be driven out of Israel as prophesied in Zechariah 14:21 (KJV), Israel will expand its borders to cover the size of the original land grant to Abraham , and Jesus will rule the world from Jerusalem . And that will be the "new" world order. Antonio Guterres, elected in October to take over as U.N. secretary general next year , told a conference in his native Lisbon that this trend had undermined the willingness to receive refugees in Europe this year. He said the world must re-establish international protection for refugees coming from war zones such as Syria, but it would not be easy as developed countries were turning to nationalist agendas. Antonio Guterres formally elected as UN chief: Europe has struggled to handle a huge influx of refugees, many of whom displaced by the war in Syria. The United States has accepted only a very small number of refugees and may take in even fewer next year. "In 2016, we have witnessed a dramatic deterioration of that international protection regime (for refugees)," Guterres said. "This example started in the developed world, it started essentially in Europe, it is spreading now like a virus into other parts of the world." Guterres, who was U.N. High Commissioner for Refugees until last year, linked the growing resistance to accepting refugees to wider concerns about globalism. "I don't think we can look strictly at the refugee issue, I think the problem is a broader problem," he told the conference on Europe's refugee crisis. There was a consensus in the mid-1990s that globalization would benefit all, he said. "But a lot of people were left behind … In the developed world, (there are) those who have been losers in globalization," he said. "The recent analysis of the rust belt in the United States, I think, is a clear demonstration of that, when we speak about the elections." Donald Trump won this month's election in the United States in part thanks to support from voters who have seen their jobs lost to countries with cheaper labor. "So globalization has not been as successful as we had hoped and lots of people became not only angry with it, but feeling that political establishments and international organizations are not paying attention, were not taking care (of them)," he said. This led to what he called "a kind of evolution" in which anti-establishment parties now tended to win elections and referendums tended to attract majorities against whatever was put to a vote. source SHARE THIS ARTICLE'

(iv)

1056
'Turmeric is More Powerful Than 14 Artificial Drugs'
'Eddy Lavine'
'posted by Eddie A lot of research has been done on turmeric. And the benefits of its compound curcumin have been confirmed by numerous studies. This herb has been proven to have medicinal and therapeutic properties more powerful than artificial drugs. Here are 14 drugs that can be replaced with turmeric/ curcumin. 1. Prozac and Imipramine (Antidepressants) This study found that turmeric was as effective as Prozac and Imipramine at reducing depression in animals. 2. Aspirin (blood thinner) One study found that curcumin has the same anti-platelet effects as aspirin. And it can also help fight arthritis. 3. Metformin (diabetes drug) Turmeric can be helpful to pre-diabetics and diabetics. In fact, this study found that curcumin was up to 100,000 times more potent than metformin in increasing glucose uptake and suppressing glucose production in the liver. 4. Corticosteroids (steroids) Studies show that curcumin is as potent as corticosteroids which are used to treat lupus and asthma. You're better off using turmeric because corticosteroids have side effects like increasing risk of infections and high blood sugar. 5. Lipitor (for cholesterol) Turmeric showed great results when compared to Lipitor, according to research . It can lower cholesterol, fight inflammation and reduce oxidative stress. 6. Oxaliplatin (chemotherapy medication) A study published by Journal of Cancer found that curcumin compared well with oxaliplatin as an antiproliferative agenet in colorectal cell lines. 7. Anti-inflammatory drugs Research shows that curcumin is more potent than drugs that fight inflammation including: Sulindac'
1

(v)

2014
'He Loses His Pulse For 45 Minutes, Wakes Up And Reveals THIS Afterlife Chilling Vision'
'Dikran Arakelian (noreply@blogger.com)'
"Share on Facebook This man loses pulse for 45 minutes, but then wakes up with an incredible vision of afterlife! A trucker in Ohio shocked hospital staff after coming back to life nearly an hour after he lost his pulse following a massive heart attack — but it's what he claims to have seen during those tense moments that has him sure there's an afterlife. Brian Miller, 41, was opening the lid of a container when he knew something felt wrong — he immediately called 911 and told the operator, "I'm a truck driver and I think I'm having a heart attack." Sure enough, his main artery was completely blocked — causing what's known as a "widow-maker" heart attack, Fox 8 Cleveland reported. He was rushed to a local hospital where doctors managed to revive him and clear the blockage, but after regaining consciousness and

feeling the pain dissipate, he developed ventricular fibrillation, when the heart starts quivering wildly and is unable to pump blood. “He had no heart rate, he had no blood pressure, he had no pulse,” said ICU nurse Emily Bishop. “I mean think about that.” Doctor's performed “strong, hard, fast CPR” and shocked Miller four times to try to revive him, but had no luck. It was during that time that Miller said he slipped away into a celestial world, “The only thing I remember I started seeing the light, and started walking toward the light.” He described walking down a flower-lined path into white light — until he came upon his step mother, who had died recently, “She was the most beautiful thing when I seen her, it was like the first day I met her, (she) looked so happy.” Miller remembered, “She grabbed ahold of my arm and told me, ‘It's not your time, you don't need to be here, we've got to take you back you've got things to go and do.’” After 45 minutes, his pulse returned “out of nowhere,” Bishop said. ”His brain had no oxygen for 45 minutes, the fact that he's up walking, talking, laughing, everything is amazing.” Glad to be back amongst the living, Miller now says there is one thing he is sure of, “There is an afterlife and people need to believe in it, big time.” Related:"

(2) **Inputs, Targets, Preprocessing & Normalizing**

The inputs were initially stored in a 2d matrix, where the rows stored individual data points and the columns stored the feature values for each respective data point. Then the data was cleaned. The cleaning included removing data points that had null values for some of their features, removing punctuation (except '.','?','!') and removing new lines. Once the data had been cleaned it was then used to train an embedding model. The embedding model analyzes the context in which each of the words were used, in order to assign each word a vector value in such a way that these vectors can convey meaning. For example, this is the embedding for the word 'community':

```
print(testEmbeddingModel['community'])
```

```
[ 0.74551004 -1.925625   -0.35264933 -0.16347292 -0.43206435 -2.4527886
 -1.7963977  -1.2403412  -0.79590076 -1.2238048   0.93789625  2.1455448
  1.9138812   1.2433157   0.8998263  -0.7207123   0.8172904   0.37591842
  3.0307348   1.454316   -2.1107852   1.625682   -0.16205058 -0.28203258
 -3.0170453  -4.1984468   0.1264153  -0.25504237  1.887289   -0.7993006
 -1.2719527   0.8130429   1.832432   -0.3656026   2.778661   -3.5208344
 -3.2012172  -1.7571218   2.6617284  -2.638855   -1.0610956   2.8663352
  1.9410295  -2.0745687   1.1829054   0.74189866  0.23189653  0.65073484
  0.15135823 -0.8838222   0.587269   -0.5725262  -0.04280417 -2.168724
  0.5997227  -1.1877364   0.53662646  2.8958428   1.6146225   0.27319577
 -0.33549374 -0.49973658 -0.2728241  -1.5008385  -2.3417366   0.63011074
  1.2928436  -1.882446   -0.41590756  1.4887761  -0.2717651   1.9350119
 -2.0292227   0.7006474   0.5744112   1.589373   -2.0576668   1.4088539
 -0.97648156 -1.1124293   2.6030974   0.51089495 -0.9466587  -0.20501074
  2.6823754  -2.1917672  -0.9929996   1.4972256  -1.1082795   1.4138021
  0.7992863  -1.7513208  -0.37373403 -0.29823056  0.58770174  2.0304072
  0.53187764 -0.27203295  1.1258776   2.6240299 ]
```

This embedding model allowed us to then convert the data points into a vector that can be used to train the fake news predictor model. This conversion of the data points included taking the average embedding representation for each word in the title, then appending the average embedding representation for each word in the articles themselves and lastly appending a value representing how likely the article is to be fake based on the author and a percentage of how often they post fake articles. Finally, the data was split into training/validation/test data in a 0.7:0.2:0.1 ratio respectively.

Training, Validation and Testing Data

We used a sample random that would generate 10% of the indices to be test indices, 20% to be validation indices and 70% to be training indices. These indices were recorded and then used to create Pandas dataframes for testing, validation and training. This ensured that one would always get a random set of data to train, test and validate with.

Additional data splitting

With regards to how the data was split, we wanted to ensure that debugging was easy, so we created sets of 10%, 20%, 40%, 80%, and 100% of the data. This was due to the fact that the run times got significantly longer as you got closer to the actual dataset. For example, 40% of the data could take close to thirty minutes (depending on the machine you're using) to preprocess and normalize.

(3) The following algorithms were implemented on the chosen dataset:

(a) Logistic Regression
(b) Support Vector Classifier
(c) Naïve Bayes

(a) One classification algorithm that was implemented in this assignment was that of logistic regression, which was used to classify whether an article was either fake news or it was not, i.e. 'real' news. Logistic regression is an appropriate algorithm to implement on the chosen dataset. Since the target variable of the dataset is binary, logistic regression can be used to predict the probabilities of each data point x, and then categorise each data point into 0 or 1, depending on its probability (categorise it 0 if $p(x) < 0.5$ or 1 if $p(x) \geq 0.5$).

In an attempt to determine the best implementation of the algorithm it was implemented using two methods: gradient descent (with loss minimization) and gradient ascent (with the maximum likelihood estimation), with both implementations yielding very interesting and accurate results. By continually adjusting the hyperparameters, such as the learning rate, it was determined that the loss minimization implementation is the more accurate implementation with an accuracy of just over 95%. However, by adjusting the learning rate (alpha) on the gradient ascent algorithm, it was discovered that we could slightly increase the algorithm's accuracy, but it would take significantly longer for that algorithm to complete running. See the discussion on hyperparameters and their effect on accuracy and time in sections i and ii.

Both implementations of the algorithm use the same set of initially randomised weights (Please see the output in the 'LogisiticRegression.ipynb' for the entire set, only the first 8 elements will be selected for this document for readability purposes):

Θ = [0.002547, 0.455505, 0.833198, 0.32285, 0.471227, 0.691664, 0.20063, 0.438592]

i. Gradient Descent with Loss Minimisation (Logistic Regression)

Running this algorithm on the dataset yields following set of weights:

Θ = [4.526196, 4.979154, 5.356846, -0,5.5113, -3.170662, -6.099912, -0.800181, -7.758719]

As one can see, the weights that are eventually learnt do not grow/decrease to incredibly large amounts when compared to the initially randomised values for the weights.

See below the error on the test set taken directly from the Jupyter Notebook output in the 'LogisticRegression.ipynb' file (illustrated with the assistance of the scikit-learn library):

- **Confusion Matrix:** [1944 104]
  [77 1897]

- **Accuracy Score:** 0.9549975136747887

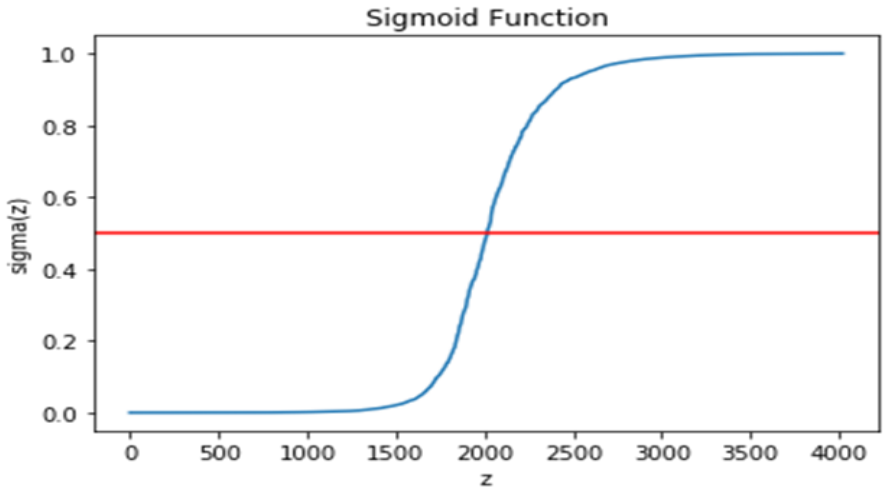
- **Report:**

	precision	recall	f1-score	support
0.0	0.96	0.95	0.96	2048
1.0	0.95	0.96	0.95	1974
accuracy			0.95	4022
macro avg	0.95	0.96	0.95	4022
weighted avg	0.96	0.95	0.96	4022

For this implementation of the gradient descent algorithm, we chose alpha = 0.8. As you will see in the table below there is an almost positive, linear correlation between alpha and the accuracy of the model. Increasing alpha by 0.1 slightly increases the accuracy of the model, but it also so mewhat increases the time in which it takes for the gradient descent function to complete. After running and timing the duration of the gradient descent algorithm on a test set (which was about 20% of the data) we got the following results:

Alpha	Accuracy	Time (in seconds)
0.95	0.950522	117.660653
0.9	0.950273	100.889569
0.8	0.950273	96.5892434
0.7	0.949528	98.431086
0.6	0.949279	96.678892
0.5	0.949030	85.790105
0.4	0.948284	99.308077
0.3	0.94729	100.289893
0.2	0.946544	110.43432
0.1	0.94182	98.144306

As you can see, if we were to pick an alpha in the interval (0.8, 1) we could possibly find an alpha that would yield a higher accuracy, but that would make the gradient descent take a slightly longer amount of time to run. In addition, slightly increasing after alpha = 0.9 increases the accuracy by 0.000249%, which is almost insignificant if there's already an accuracy of 95%. Clearly, alpha = 0.8 seemed to be the most optimal value to pick when scaled up to a much larger dataset. Below is the Sigmoid function produced by the gradient descent function implemented on the test set:



i. Maximum Likelihood Estimation with gradient descent (Logistic Regression)

Running this algorithm on the dataset yields following set of weights:

Θ = [7046.150991, 7046.150991, -1113.332136, -4830.573828, -9784.339699, -635.85943, -9935.953705, 4516.882078]

As one can see, the weights grow/decrease to incredibly large values when compared to those after running the loss minimization implementation of gradient descent, and even those of the randomly initialised set. Perhaps implementing this algorithm with regularization would be necessary to ensure that the values do not become too high.

See below the error on the test set taken directly from the Jupyter Notebook output in the 'LogisticRegression.ipynb' file (illustrated with the assistance of the scikit-learn library):

- **Confusion Matrix:** [2015 33]
  [272 1702]

- **Accuracy Score:** 0.9241670810542019

- **Report:**

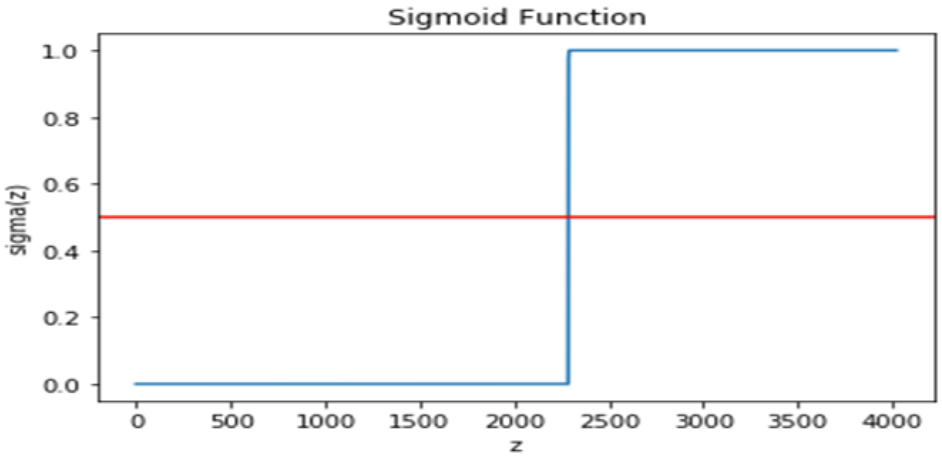
	precision	recall	f1-score	support
0.0	0.88	0.98	0.93	2048
1.0	0.98	0.86	0.92	1974
accuracy			0.92	4022
macro avg	0.93	0.92	0.92	4022
weighted avg	0.93	0.92	0.92	4022

The accuracy and results of this specific iteration of this algorithm slightly differs from those of the discussions of alpha values that are to follow. This could be because of the initial weights that are randomly initialised, so the algorithm is always working on a different set of thetas.

For this implementation of the gradient ascent algorithm, we chose alpha = 0.5. As you will see in the table below there is an almost positive, linear correlation between alpha and the accuracy of the model (though the correlation is not as strong as in the loss minimization implementation of the gradient descent algorithm). Increasing alpha by 0.1 slightly increases the accuracy of the classifying model, but it also somewhat increases the time in which it takes for the gradient ascent function to finish running. We ran and timed the duration of the gradient ascent on a test set, and got the following results:

Alpha	Accuracy	Time (in seconds)
0.95	0.954251	165.928035
0.9	0.94903	145.716267
0.8	0.948036	146.313839
0.7	0.944306	148.20403
0.6	0.950025	144.38948
0.5	0.948036	102.46435
0.4	0.935107	154.129854
0.3	0.948533	143.555976
0.2	0.935853	152.79072
0.1	0.867976	137.240029

Similar to the loss minimization implementation of gradient descent, picking alpha in the interval (0.5, 1) could possibly yield a higher accuracy, but that would result in longer computation of the gradient ascent. Clearly, alpha = 0.5 would be the most optimal value to pick when scaled up to a much larger dataset. Below is the Sigmoid function produced by the gradient descent function implemented on the test set:



(b) The support vector classifier model was implemented to try to predict fake news as well. The data points used were all vectors and therefore made it viable to use this model since it would find separating hyperplanes between the data. These hyperplanes are not restricted to linear hyperplanes either, and the model can easily be adjusted to find more flexible separating hyperplanes. This model was also suitable because of how efficiently it can classify data once it has been trained, which could be very beneficial depending on where this model is used.

The model was implemented using gradient descent on the cost function. This is because it is a lot simpler yet still very effective compared to the numerical method approach of training the model and can easily handle different training data with any number of dimensions. Regularisation was also used in order to improve performance. The regularisation provided the capability to control how much of an effect the outliers have on training the model, thus allowing us to minimize the effect the outliers will have when training the model. In order to ensure the model would not train forever, there were two terminating conditions implemented. The first being a maximum number of epochs that can run, which was set to 1000 in order to allow the model to train for a viable period of time while still not taking too long. Secondly the program would terminate if the absolute cost function difference was small enough after a specific epoch iteration. To also allow for less biased training, the dataset was randomly shuffled at the beginning of each epoch.

In order to choose good hyper-parameters, an initial set of hyper-parameters were tested and then they were tweaked and changed from there in order to find better performing hyper-parameters. These were all tested using the validation data. The regularisation constant was chosen to be 10000 initially because if it were too small then the algorithm would treat many more data points as outliers while if it were too big, then it would treat too few data points as outliers which would both give skewed results. The learning rate was initialised at 0.0000001 because it needed to be small in order to actually start converging and improving and because the regularisation constant was set so high, while not too small so that it still learned at a viable rate. The maximum number of epochs was chosen to be 1000 and the termination constant was chosen to be 500.

The results of the different hyper-parameter settings using the validation data are shown below:

Current model hyperparameters:

Regularization strength: 10000

Learning rate: 1e-07

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 94.110337972167

Confusion Matrix:

[[1966   96]

[141 1821]]

Current model hyperparameters:

Regularization strength: 1000

Learning rate: 1e-07

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 93.61332007952286

Confusion Matrix:

[[1937   87]

[170 1830]]

Current model hyperparameters:

Regularization strength: 100000

Learning rate: 1e-07

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 94.06063618290258

Confusion Matrix:

[[1966   98]

[141 1819]]

Current model hyperparameters:

Regularization strength: 10000

Learning rate: 1e-06

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 93.98608349900596

Confusion Matrix:

[[1960   95]

[147 1822]]

Current model hyperparameters:

Regularization strength: 10000

Learning rate: 1e-08

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 93.88667992047714

Confusion Matrix:

[[1953   92]

[154 1825]]

Current model hyperparameters:

Regularization strength: 100000

Learning rate: 1e-08

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 94.110337972167

Confusion Matrix:

[[1959   89]

[148 1828]]

Current model hyperparameters:

Regularization strength: 1000

Learning rate: 1e-06

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 93.58846918489066

Confusion Matrix:

[[1928   79]

[179 1838]]

Current model hyperparameters:

Regularization strength: 100000

Learning rate: 1e-06

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 92.51988071570577

Confusion Matrix:

[[1878   72]

[229 1845]]

Current model hyperparameters:

Regularization strength: 1000

Learning rate: 1e-08

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 92.71868787276343

Confusion Matrix:

[[1914 100]

[193 1817]]

Current model hyperparameters:

Regularization strength: 50000

Learning rate: 1e-08

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 93.96123260437376

Confusion Matrix:

[[1963   99]

[144 1818]]

Current model hyperparameters:

Regularization strength: 5000

Learning rate: 1e-06

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 93.78727634194831

Confusion Matrix:

[[1951   94]

[156 1823]]

Current model hyperparameters:

Regularization strength: 12500

Learning rate: 1e-07

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 94.06063618290258

Confusion Matrix:

[[1960   92]

[147 1825]]

Current model hyperparameters:

Regularization strength: 7500

Learning rate: 1e-07

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 94.110337972167

Confusion Matrix:

[[1962   92]

[145 1825]]

Current model hyperparameters:

Regularization strength: 10000

Learning rate: 3e-07

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 94.01093439363817

Confusion Matrix:

[[1957   91]

[150 1826]]

Current model hyperparameters:

Regularization strength: 10000

Learning rate: 7e-08

Termination constant: 500

Max number of iterations for training: 1000

Accuracy: 94.110337972167

Confusion Matrix:

[[1965   95]

[142 1822]]

As can be seen the best performing hyper-parameters with an accuracy of 94.11% are as follows:

Regularization strength: 10000
Learning rate: 1e-07
Termination constant: 500

Max number of iterations for training: 1000

Therefore, these settings were chosen and then tested using the testing data. The results are shown below:

```
Current model hyperparameters:
Regularization strength: 10000
Learning rate: 1e-07
Termination constant: 500
Max number of iterations for training: 1000
Accuracy:  92.94234592445328
Confusion Matrix:
[[928  61]
 [81 942]]
```

(c) The final algorithm we used to classify our dataset was Naïve-Bayes. We chose this algorithm because it is particularly well-suited to our type of dataset. Applying Naïve-Bayes to classify text data is simple and intuitive, so the only complications arise when you have a large dataset. As our dataset is very large (~20000 articles in the training set), we had to make a few adjustments to the basic algorithm to suit our needs. We trained the Naïve-Bayes model using 80 percent of the dataset, 10 percent for testing and 10 percent for validation. 80 percent seemed suitable as this had ~18000 articles for training, and ~2000 articles each for testing and validation. Training the model took an extremely long time, at just under 13 hours. This would be a result of having to vectorise the entire set of seen words, as well as then count the occurrences of each of these words in every article.

The algorithm was initially implemented as is, with no smoothing or normalisation, but this proved insufficient. This code was run on one article at a time, which it in general correctly classified (choosing random articles from the testing set), but running it on the entire testing set revealed what would later be identified as an underflow problem in particular.

We were not sure what the issue was at first, and so implemented Laplacian smoothing in the predict function as an attempt to improve performance. This did improve results slightly, however, the algorithm was still classifying large blocks of articles as fake news, following no pattern. Further investigation revealed that this was the result of underflow, as while predicting the class for an article, our model was multiplying a great many very small probability values, resulting in the probability rounding to zero, and logic errors to follow.

We used normalisation to solve this problem, particularly the log-sum-exp “trick”. This “trick” involves using the logarithmic identities to our advantage, changing the

multiplication/division operation in each iteration of the loop to an addition/subtraction operation, and then normalising or “shifting” the result by the maximum value at the end.

These are the results of our investigation:

Testing the model on a small subset (100 articles) of the testing set:

```
Correctly identified: 68
Incorrectly identified: 32
True Positives: 27
True Negatives: 41
False Positives: 13
False Negatives: 19
```

This seemed promising, so the model was then run on the full testing set of ~2000 articles, resulting in the strange output of the entire set classified as fake news. This oddly had an accuracy of around 55%, but that’s not much better than guessing.

With Laplacian smoothing implemented, these results pointed us to normalisation, which had much better results on the full testing dataset:

```
Correctly identified: 1393
Incorrectly identified: 629
True Positives: 657
True Negatives: 736
False Positives: 314
False Negatives: 315

Error: 0.311078
Accuracy: 0.688922
Precision: 0.676622    Recall: 0.675926
Miss: 0.324074    False Alarm: 0.299048
Recall: 0.675926    False Alarm: 0.299048

Confusion Matrix:
[[657 315]
 [314 736]]
```

With these results, we were confident to move onto validation. There are very few hyperparameters to tweak with Naïve-Bayes, so this was apparently as good as it would get on this dataset.

These were our results from validation, the final results for our Naïve-Bayes model:

```
Correctly identified: 1368
Incorrectly identified: 641
True Positives: 660
True Negatives: 708
False Positives: 344
False Negatives: 297
```

```
Error: 0.319064
Accuracy: 0.680936
Precision: 0.657371    Recall: 0.689655
Miss: 0.310345    False Alarm: 0.326996
Recall: 0.689655    False Alarm: 0.326996

Confusion Matrix:
[[660 297]
 [344 708]]
```

The final iteration of our model had about the same accuracy on the dataset as the as-is algorithm did on a small subset of the data. The accuracy of the final model on the testing set and the validation set were almost equal.

Naïve-Bayes assumes feature independence, but in general has a surprisingly high accuracy even with this assumption. For this particular dataset, we would recommend a more complex model, but the advantage of this model is that is simplifies the thought process and intuition behind implementing this technique, with a trade-off in performance.

(4) By looking at the results it can be seen that both logistic regression and support vector classifier models performed very well. Both models have accuracies, recalls, precisions and F-scores greater than 0.9. These performed better than the Naïve-Bayes model because they were better suited for the problem at hand. This is because both of them simply try to separate the data as distinctively as possible, while the Naïve-Bayes model tries to assign each classification a score, and then choose the best one. The problem with assigning each classification a score is that certain assumptions must be made, and also these scores may be calculated with some irrelevant or even destructive information which can easily skew the results of the model. Or maybe these scores just don't capture the distinctive qualities of each class at all. By just trying to separate the data, no assumptions need to be made and the model is less susceptible to suffer skewed results due to irrelevant information. Although, these models will still be susceptible to outliers skewing the results.

One of the biggest advantages of both the logistic regression and support vector classifier models is their ability to efficiently classify data points once the model has been trained, unlike the Naïve-Bayes model. This is due to the fact that both models essentially just need to compute a single dot product in order to provide a prediction. This can easily be verified above where the time for prediction for models is seen to be < 1s while for the Naïve-Bayes

model it is between 2s and 4s. This benefit could be extremely useful depending on the context in which the model is implemented.

None of the above models are optimised to train on very big datasets since the training time will scale badly as the data increases in size. Although they were sufficient to train using our chosen dataset, as can be seen from the values in the time to train category. Another disadvantage of both the logistic regression and support vector classifier models is that they would both need to be completely re-trained if one acquired more data to train the models on. Luckily for the Naïve-Bayes model, the entire model does not need to be re-trained but can instead just be updated using the new data.

There are also other advantages and disadvantages of each model that one may want to consider when deciding on a model to implement:

- An advantage of using either a logistic regression or Naïve-Bayes model is that you get probabilistic outputs/predictions unlike the support vector classifier model. This allows one to view how confident the model is in its prediction, which is very valuable information.

- Both the logistic regression and support vector classifier models are relatively memory efficient.

- The kernel 'trick' can be exploited to find more complex separating hyperplanes when using the support vector classifier.

- Data needs to be pre-processed more often than not when using either the logistic regression or support vector classifier models.

- The Naïve-Bayes model assumes that the words used are all independent of each other.

- The Naïve-Bayes model does not contain many hyper-parameters that allow for effective fine tuning.

- Naïve-Bayes is greatly disadvantaged by the presence of a large variety of subjects, considering that the model is built on the probability of certain words occurring. Having words occurring once or twice in the entire dataset both skews results as well as increases the training time and the time taken to predict the class of unseen data.

There will always be a way to improve a model's performance. Some of the solutions that could have been implemented to improve performance are as follows:

- Take advantage of the kernel ‘trick’ in order to find the most suitable separating hyperplane for the support vector classifier.

- Clean the data more effectively. For example, deal with words in foreign languages, articles that end short, emojis etc.

- Could add extra features such as average word length, number of spelling errors, lengths of paragraphs, lengths of articles.

- Try to narrow the subject matters of the articles in the dataset. For example, only train on political articles in order to make better predictions on political articles.

- Try to algorithmically fine tune the hyper-parameters instead of tuning them manually.

Please see below summaries of the performance of the three algorithms on the chosen dataset:

Logistic Regression:

Accuracy: 0.9549975136747887
Confusion Matrix:
[[1944 104]
[77 1897]]
Time to train: Around 7 mins to train the model with the big dataset

Time to predict: <1s
Recall: 0.96
Precision: 0.96
F-Score: 0.96

Support Vector Classifier:

Accuracy: 92.94234592445328Confusion Matrix: [[928 61] [81 942]]
Time to train: Around 7 mins to train the model with the big dataset
Time to predict: <1s
Recall: 0.94
Precision: 0.92
F-Score: 0.93

Naïve-Bayes Classifier:

Accuracy: 68.0936%
Confusion Matrix:
[[660 297]
[344 708]]
Time to train: Just under 13 hours on ~18000 articles
Time to predict: 2 – 4 seconds per article
Recall: 0.689655
Precision: 0.657351
F-Score: 0.6731156415

References

1. Kaggle.com. 2020. *Fake News | Kaggle*. [online] Available at: <https://www.kaggle.com/c/fake-news/data> [Accessed 24 May 2020].

2. Kaggle.com. 2020. *Logistic Regression From Scratch - Python*. [online] Available at: <https://www.kaggle.com/jeppbautista/logistic-regression-from-scratch-python> [Accessed 17 April 2020].

3. Bird, R., 2020. *Logistic Regression*. [online] Medium. Available at: <https://medium.com/greyatom/logistic-regression-89e496433063> [Accessed 18 April 2020].

4. Harris, M., 2020. The PR Profession Doesn't Seem To Care About Fake News. [online] Prweek.com. Available at: <https://www.prweek.com/article/1591975/pr-profession-doesnt-seem-care-fake-news> [Accessed 20 April 2020].