FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

INTRODUCTION TO DATA SCIENCE

BITI 2513

MINI PROJECT

TASK 3: MODELING AND EVALUATION

GROUP : HYPER

NAME	MATRIC NUMBER
THITTHIMA A/P WAT	B031810361
ELLE ALIZ BT. AMINUDDIN	B031810295
MUHAMMAD HARRAZ B. HARUN	B031810426

CLASS : 2 BITI S1G2

LECTURER NAME : AP DR SHARIFAH SAKINAH SYED AHMAD

Table of Contents

Algorithms Used

1. Support Vector Machine (SVM) Classifier

Support Vector Machine (SVM) is one of the supervised machine learning algorithms. It can be used for both classification and regression challenges. SVM mostly used in classification problems. In SVM, we will plot each data item as a point in n-dimensional space where n is number of features that we have with the value of each feature being that value of coordinate. It is a discriminative classifier formally defined by a separating hyperplane. After that we will perform classification by finding the hyper-plane that differentiates that two classes very well. By using a given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples. In 2-dimensional space the hyperplane is a line dividing a plane in two parts where in each class lay in either side. Hyperplane can be more than one dimensions.

Strength of SVM:

i. The training process is relatively easy

ii. It does not have local optimal

iii. It scales relatively well to high dimensional data and trade-off between classifier complexity

iv. Error can be controlled explicitly

v. Really well in complicated domains where it have a clear margin of separation.

Weakness of SVM:

i. SVM algorithm has several key parameters that need to be set correctly to achieve the best classification results for any given problem. Parameters that may result in an excellent classification accuracy for problem A, may result in a poor classification accuracy for problem B.

ii. It needs a good kernel function.

iii. Not perform so well in a very large data sets (because the training time happens to be cubic in the size of the data sets).

iv. SVM might be prone to over fitting to some of the noise in big data set.

2. Naïve Bayes Classifier

A Naive Bayes classifier is a probabilistic machine learning model that is used for classification task. Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle and every pair of features being classified is independent of each other.

Strength of Naïve Bayes Classifier

i. The advantage of using Naïve Bayes Classifier is that it works well with high-dimensional data such as text classification which is suitable for our fake news detection.
ii. Naïve Bayes Classifier is also very fast compared to other complicated algorithms.

Weakness of Naïve Bayes Classifier

i. the assumption that all features are independent is not usually the case in real life, so it makes it less accurate than complicated algorithms.

3. Stochastic Gradient Descent (SGD)

Stochastic gradient descent is an iterative method for optimizing an objective function with suitable smoothness properties. It is a popular optimization technique in Machine Learning and Deep Learning, and it can be used with most, if not all, of the learning algorithms. A gradient is the slope of a function. It measures the degree of change of a variable in response to the changes of another variable. In a typical gradient descent, the whole dataset is taken as a batch (the total number of samples from a dataset used to calculate the gradient for each iteration) which is problematic when the dataset is significantly large. It becomes computationally expensive to perform. Stochastic gradient descent solves this problem by using a single sample to perform each iteration.

Strength of SGD

i. It is suited for highly non-convex loss functions, such as those entailed in training deep networks for classification.
ii. It does the calculations faster than gradient descent and batch gradient descent.
iii. Stochastic gradient descent performs updates more frequently and therefore can converge faster on huge datasets.

Weakness of SGD

i. SGD requires several hyperparameters and a few iterations.
ii. It is also sensitive to feature scaling.
iii. Error function is not well minimized.
iv. There is a common learning rate for all parameters.

4. Passive Aggressive Algorithm

The passive-aggressive algorithms are a family of algorithms for large-scale learning and a family of online learning algorithms (for both classification and regression) proposed by Crammer at al. The Passive Aggressive (PA) algorithm is perfect for classifying massive streams of data.

Advantages of PA Algorithm

i. Very fast
ii. Easy to implement

Disadvantages of PA Algorithm

i. does not provide global guarantees like the support-vector machine (SVM).

Evaluation Measure

To determine which algorithms is the best we use accuracy, precision, recall and f1-score. In measurement of a set, accuracy is closeness of the measurements to a specific value, while precision is the closeness of the measurements to each other. Accuracy is a description of systematic errors, a measure of statistical bias, low accuracy causes a difference between a result and a "true" value ISO calls this trueness. Moreover, the definition of precision shows that lowering the threshold of a classifier may increase the denominator, by increasing the number of results returned. If the threshold was previously set too high, the new results may all be true positives, which will increase precision. If the previous threshold was about right or too low, further lowering the threshold will introduce false positives, decreasing precision. Next, Recall is defined as formula does not depend on the classifier threshold. This means that lowering the classifier threshold may increase recall, by increasing the number of true positive results. It is also possible that lowering the threshold may leave recall unchanged, while the precision fluctuates. Furthermore, F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

Accuracy

Accuracy is defined as the ability of the classifier to select all the cases that need to be selected and reject all cases that need to be rejected.

$$Accuracy = \frac{TP + TN}{(TP + FP + TN + FN)}$$

Precision

Precision is defined as the proportion of cases found that were actually relevant.

$$\Pr ecision = \frac{TP}{(TP + FP)}$$

Recall

Recall is defined as the proportion of the relevant cases that we actually found among all the relative cases.

$$\mathrm{Re}call = \frac{TP}{(TP + FN)}$$

F1-score

F1-score also called as the F score or the F measure. It conveys the balance between the precision and the recall.

$$F1 = 2 \times \frac{(\mathrm{Pr}ecision \times \mathrm{Re}call)}{(\mathrm{Pr}ecision + \mathrm{Re}call)}$$

Modelling Preparation

1. Import library

```
In [1]: import pandas as pd
```

```
In [2]: from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
        from sklearn.linear_model import PassiveAggressiveClassifier, SGDClassifier
        from sklearn.svm import LinearSVC
        from sklearn.naive_bayes import MultinomialNB
        from sklearn import metrics
        import matplotlib.pyplot as plt
        from sklearn.utils import shuffle
        from sklearn.metrics import classification_report
        import seaborn as sns
```

Figure 1: Import Library

Firstly, we import all the library needed in this project.

2. Import Training Data Set.

```
In [3]: train = pd.read_csv(r'C:/Users/User/Desktop/UTeM/Data Science/Project/news.csv')
        train.head()
```

Out[3]:

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

Figure 2: Import Training Data Set

Secondly, we imported the data set that we will use for training process and we displayed the data just for the conformation either the data we imported is true or not.

3. Change the data into Data Frame and classify only REAL and FAKE

```
In [11]: train_dict = {'Title':train['title'],
                       'Text':train["text"],
                       'Label':train['label']}
         train_df = pd.DataFrame(train_dict)

In [12]: train_df.count()

Out[12]: Title    6335
         Text     6335
         Label    6335
         dtype: int64

In [15]: fake_set = train_df.loc[train_df['Label'] == 'FAKE']
         real_set = train_df.loc[train_df['Label'] == 'REAL']
         print(fake_set.count())
         print(real_set.count())

         Title    3164
         Text     3164
         Label    3164
         dtype: int64
         Title    3171
         Text     3171
         Label    3171
         dtype: int64

In [16]: real_train_data = pd.concat([fake_set,real_set])

In [17]: real_train_data = shuffle(real_train_data)

In [18]: real_train_data.columns

Out[18]: Index(['Title', 'Text', 'Label'], dtype='object')
```

Figure 3: Change Data Frame

We renamed the column name to make it standardized and then we change the data into DataFrame using pandas. After that we count the train data to ensure all there are no missing values. Next, we set the label as FAKE and REAL only and we count the data to see either the data is bias or not. Based on our output, we can see that there are 3164 fake news and 3171 real news. We can say that this data sets have low bias since the number of fake news data and real news data are almost the same. Lastly, we are concatenating the REAL and FAKE labeled data.

4. Import Testing Data Set

```
In [27]: test = pd.read_excel(r'C:/Users/User/Desktop/UTeM/Data Science/Project/Test_data.xlsx')
         test.head()
```

Out[27]:

	title	text	subject	label
0	Donald Trump Sends Out Embarrassing New Year□...	Donald Trump just couldn t wish all Americans ...	News	FAKE
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	FAKE
2	BARBRA STREISAND Gives Up On Dream Of Impeachi...	Barbra Streisand was an Obama sycophant and on...	left-news	FAKE
3	WATCH: SENATOR LINDSEY GRAHAM DROPS BOMBSHELL□ ...	Everyone suspected the sketchy Steele Dossier ...	left-news	FAKE
4	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	FAKE

```
In [29]: test_dict = {'Title':test['title'],
                     'Text':test['text'],
                     'Label':test['label']}
         test_df = pd.DataFrame(test_dict)
```

```
In [32]: test_fake_set = test_df.loc[test_df['Label'] == 'FAKE']
         test_real_set = test_df.loc[test_df['Label'] == 'REAL']
         print(test_fake_set.count())
         print(test_real_set.count())
```

```
Title    104
Text     104
Label    104
dtype: int64
Title    174
Text     174
Label    174
dtype: int64
```

Figure 4: Import Testing Data Set

We imported the data that we want to use as testing data. Then, we standardize the column name same as training data. After that we count the FAKE and REAL data. Here in the test data, we must separate the true and false labels.

5. Concatenate the testing data

```
In [33]: ## concatinating true and false labeled datasets
         real_test_data = pd.concat([test_fake_set,test_real_set])
         ## shuffling real_train_data
         real_test_data = shuffle(real_test_data)
```

```
In [42]: ## CountVectorizer tokenizes the collection of text documents and build a vocabulary of known words it returns ints
         count_vectorizer = CountVectorizer(stop_words = 'english')

         ## function of fit_transform is fit and transform the function for feature extraction
         count_train = count_vectorizer.fit_transform(real_train_data['Title'])

         ## transforms documents to document-type matrix
         count_test = count_vectorizer.transform(real_test_data['Title'])
```

```
In [43]: ## TfidVectorizer is same as CountVectorizer but it returns float  values. In the below I'd compare the both values it false
         tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)

         ## function of fit_transform is fit and transform the function for feature extraction
         tfidf_train = tfidf_vectorizer.fit_transform(train_df['Title'])

         ## transforms documents to document-type matrix
         tfidf_test = tfidf_vectorizer.transform(test_df['Title'])
```

Figure 5: Concatenate Testing Data Set.

We concatenate the testing data like what we have done in training data.

Modelling Implementation

a. Naive Bayes Classifier

In [75]:
```
## ## Naive Bayes classifier

## Instantiate a Naive Bayes classifier: nb_classifier
nb_classifier = MultinomialNB()

## Fit the classifier to the training data
nb_classifier.fit(count_train, real_train_data['Label'])

## Create the predicted tags: pred
mnb_pred = nb_classifier.predict(count_test)

## Create the predicted tags: pred
mnb_score = metrics.accuracy_score(real_test_data['Label'], mnb_pred)

## Calculate the confusion matrix: mnb_cm
mnb_cm = metrics.confusion_matrix(real_test_data['Label'], mnb_pred, labels=['REAL', 'FAKE'])
print('Confusion Matrix')
print(mnb_cm)
print("Multinomial Naive Bayes accuracy:   %0.3f" % mnb_score)
```

In [76]:
```
mnb_cm = metrics.confusion_matrix(real_test_data['Label'],mnb_pred)
plt.show()
plt.figure(figsize=(5,5))
sns.heatmap(mnb_cm, annot=True, linewidth=.5, square = True, cmap = 'Blues_r',fmt='f');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
plt.title('Confusion Matrix for MultinomialNB', size = 10);

report = classification_report(real_test_data['Label'],mnb_pred)
print(report)
```

Figure 6: Naïve Bayes Classifier Implementation

Firstly, we initialize a multinomial Naive Bayes classifier. Next, we fit the classifier to the training data and predicted tags will be created. Then we apply the test model in the algorithms. After that the confusion matrix and the accuracy of this model are calculated. Lastly the precision, recall, f1-score and support for REAL and FAKE news are calculated.

b. Passive Aggressive Algorithm Classifier

```
In [77]: ## Instantiating a Passive Aggressive Classifier classifier: pa_tfidf_clf
pa_tfidf_clf = PassiveAggressiveClassifier()

## Fit the classifier to the training data
pa_tfidf_clf.fit(count_train, real_train_data['Label'])

## Create the predicted tags: pac_pred
pac_pred = pa_tfidf_clf.predict(count_test)
## Calculate the accuracy score: pac_score
pac_score = metrics.accuracy_score(real_test_data['Label'], pac_pred)

## Calculate the confusion matrix: pac_cm
pac_cm = metrics.confusion_matrix(real_test_data['Label'], pac_pred, labels=['REAL', 'FAKE'])
print('Confusion Matrix --- PassiveAggressiveClassifier')
print(pac_cm)
print("accuracy:   %0.3f" % pac_score)
```

```
In [98]: pac_cm = metrics.confusion_matrix(real_test_data['Label'],pac_pred)
plt.show()
plt.figure(figsize=(5,5))
sns.heatmap(pac_cm, annot=True, linewidth=.5, square = True, cmap = 'Blues_r',fmt='f');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
plt.title('Confusion Matrix', size = 10);

report = classification_report(real_test_data['Label'],pac_pred)
print(report)
```

Figure 7: Passive Aggressive Algorithm Classifier Implementation

In passive aggressive classifier, firstly the passive aggressive classifier is initialized. After that, the classifier is fitted to the training data and the tags are created. The algorithm is applied to the testing data and accuracy for this model is calculated. Lastly, the precision, recall, f1-score and support for REAL and FAKE news are calculated.

c. Support Vector Machine Classifier

In [88]:
```
## Support Vector Classifier
## Instantiate a Support Vector classifier: svc_tfidf_clf
svc_tfidf_clf = LinearSVC()

## Fit the classifier to the training data
svc_tfidf_clf.fit(count_train, real_train_data['Label'])

## Create the predicted tags: svc_pred
svc_pred = svc_tfidf_clf.predict(count_test)

## Calculate the accuracy score: svc_score
svc_score = metrics.accuracy_score(real_test_data['Label'], svc_pred)

## Calculate the confusion matrix: cm
svc_cm = metrics.confusion_matrix(real_test_data['Label'], svc_pred, labels=['REAL', 'FAKE'])
print('Confusion Matrix --- LinearSVC')
print(svc_cm)
print("accuracy:   %0.3f" % svc_score)
```

In [100]:
```
svc_cm = metrics.confusion_matrix(real_test_data['Label'],svc_pred)
plt.show()
plt.figure(figsize=(5,5))
sns.heatmap(svc_cm, annot=True, linewidth=.5, square = True, cmap = 'Blues_r',fmt='f');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
plt.title('Confusion Matrix', size = 10);

report = classification_report(real_test_data['Label'],svc_pred)
print(report)
```

Figure 8: Support Vector Machine Classifier Implementation

SVM algorithm is applied to the training and testing data. After that, the accuracy, precision, recall, f1 score and support are calculated.

d. Stochastic Gradient Descent Classifier

```
In [83]: ## Stochastic Gradient Descent Classifier
         ## Instantiate a Multinomial Naive Bayes classifier: sgd_tfidf_clf
         sgd_tfidf_clf = SGDClassifier()

         ## Fit the classifier to the training data
         sgd_tfidf_clf.fit(count_train, real_train_data['Label'])

         ## Create the predicted tags: sgd_pred
         sgd_pred = sgd_tfidf_clf.predict(count_test)

         ## Calculate the accuracy score: score
         sgd_score = metrics.accuracy_score(real_test_data['Label'], sgd_pred)

         ## Calculate the confusion matrix: cm
         sgd_cm = metrics.confusion_matrix(real_test_data['Label'], sgd_pred, labels=['REAL', 'FAKE'])
         print('Confusion Matrix --- SGD Classifier')
         print(sgd_cm)

         print("accuracy:   %0.3f" % sgd_score)
```

```
In [102]: sgd_cm = metrics.confusion_matrix(real_test_data['Label'],sgd_pred)
          plt.show()
          plt.figure(figsize=(5,5))
          sns.heatmap(sgd_cm, annot=True, linewidth=.5, square = True, cmap = 'Blues_r',fmt='f');
          plt.ylabel('Actual label');
          plt.xlabel('Predicted label');
          plt.title('Confusion Matrix', size = 10);

          report = classification_report(real_test_data['Label'],sgd_pred)
          print(report)
```

Figure 9: Stochastic Gradient Descent Classifier Implementation

For Stochastic Gradient Descent Classifier, the training and testing data are applied in the model. After that, the accuracy, precision, recall, f1 score and support are calculated.

Results

a. Naive Bayes Classifier

Confusion Matrix:

```
Confusion Matrix
[[148  26]
 [ 32  72]]
Multinomial Naive Bayes accuracy:    0.791
```

Figure 10: Naïve Bayes Confusion Matrix

Precision, Recall, f1-score and support:

	precision	recall	f1-score	support
FAKE	0.73	0.69	0.71	104
REAL	0.82	0.85	0.84	174
accuracy			0.79	278
macro avg	0.78	0.77	0.77	278
weighted avg	0.79	0.79	0.79	278

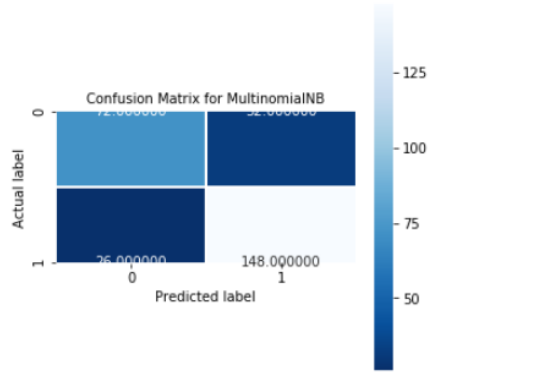

Figure 11: Naïve Bayes Precision, Recall, f1-score & Support

b. Passive Aggressive Algorithm Classifier

Confusion Matrix:

```
Confusion Matrix --- PassiveAggressiveClassifier
[[132  42]
 [ 27  77]]
accuracy:   0.752
```

Figure 12: Passive Aggressive Algorithm Classifier Confusion Matrix

Precision, Recall, f1-score and support:

	precision	recall	f1-score	support
FAKE	0.65	0.74	0.69	104
REAL	0.83	0.76	0.79	174
accuracy			0.75	278
macro avg	0.74	0.75	0.74	278
weighted avg	0.76	0.75	0.75	278

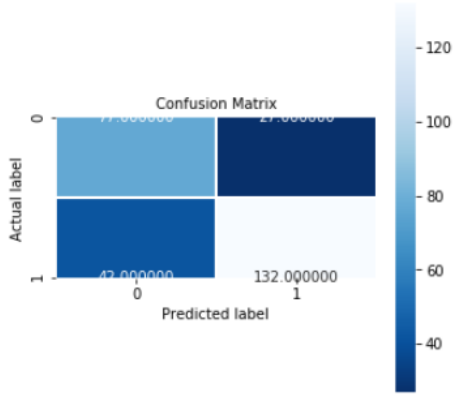

Figure 13: Passive Aggressive Algorithm Classifier Precision, Recall, f1-score, and Support

c. Support Vector Machine Classifier

Confusion Matrix:

```
Confusion Matrix --- LinearSVC
[[138  36]
 [ 29  75]]
accuracy:   0.766
```

Figure 14: Support Vector Machine Classifier Confusion Matrix

Precision, Recall, f1-score, and support:

	precision	recall	f1-score	support
FAKE	0.68	0.72	0.70	104
REAL	0.83	0.79	0.81	174
accuracy			0.77	278
macro avg	0.75	0.76	0.75	278
weighted avg	0.77	0.77	0.77	278

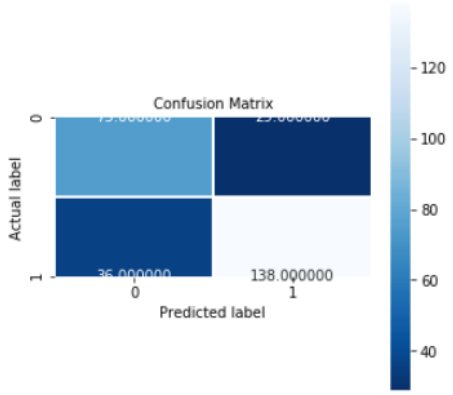

Figure 15: Support Vector Machine Classifier Precision, Recall, f1-score, and Support

d. Stochastic Gradient Descent Classifier

Confusion Matrix:

```
Confusion Matrix --- SGD Classifier
[[137  37]
 [ 27  77]]
accuracy:   0.770
```

Figure 16: Stochastic Gradient Descent Classifier Confusion Matrix

Precision, Recall, f1-score and support:

	precision	recall	f1-score	support
FAKE	0.68	0.74	0.71	104
REAL	0.84	0.79	0.81	174
accuracy			0.77	278
macro avg	0.76	0.76	0.76	278
weighted avg	0.78	0.77	0.77	278

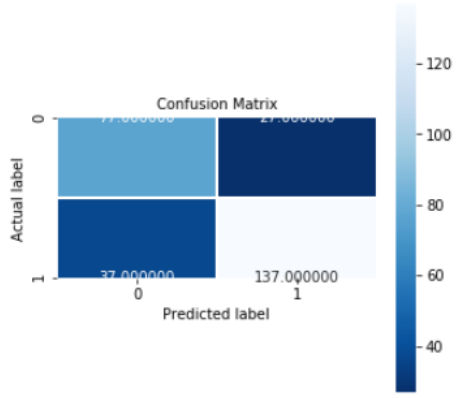

Figure 17: Stochastic Gradient Descent Classifier Precision, Recall, f1-score, and Support

Conclusion

Classifier	*Naive Bayes*	*Passive Aggressive*	*SVM*	*Stochastic Gradient Descent*
Accuracy	**0.791**	0.745	0.766	0.770
Precision (Macro Avg)	**0.78**	0.73	0.75	0.76
Precision (Weighted Avg)	**0.79**	0.75	0.77	0.78
Recall (Macro Avg)	**0.77**	0.73	0.76	**0.77**
Recall (Weighted Avg)	**0.79**	0.74	0.77	0.77
F1-Score (Macro Avg)	**0.77**	0.73	0.75	0.76
F1-Score (Weighted Avg)	**0.79**	0.75	0.77	0.77

Table 1: Modelling Evaluation and Analysis

Best Algorithm: **Naive Bayes**

Conclusion

In conclusion, we can observe that our fake news classifier model is good but not the best since the accuracy of prediction is just about 79.1%. Naive Bayes algorithm has the highest score for all the evaluation result followed by Stochastic Gradient Descent algorithm, SVM and lastly Passive Aggressive algorithms. All the algorithms get the accuracy more than 70%.

References

1. Naïve Bayes Classified- Explained: https://towardsdatascience.com/naive-bayes-classifier-explained-50f9723571ed
2. Understanding Support Vector Machine: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
3. Optimization Algorithms for Machine Learning Models: https://www.globallogic.com/paper/optimization-algorithms-for-machine-learning-models/
4. Stochastic Gradient Descent: https://www.globallogic.com/paper/optimization-algorithms-for-machine-learning-models/
5. Passive Aggressive Algorithms: https://www.bonaccorso.eu/2017/10/06/ml-algorithms-addendum-passive-aggressive-algorithms/
6. Detecting Fake News with Scikit-Learn by Ktharine Jarmul on August 24th, 2017. https://www.datacamp.com/community/tutorials/scikit-learn-fake-news
7. Classification Accuracy is Not Enough: More Performance Measures You Can Use by Jason Brownlee on March 21, 2014 in Machine Learning Process. https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/#:~:text=F1%20Score,the%20precision%20and%20the%20recall
8. Full Pipeline Project: Python AI for detecting fake news by Johnny Wales on March 6, 2019 https://towardsdatascience.com/full-pipeline-project-python-ai-for-detecting-fake-news-with-nlp-bbb1eec4936d
9. Fake News Detection on Social Media: A Data Mining Perspective on September 2017 https://dl.acm.org/doi/abs/10.1145/3137597.3137600