I. Definition

Project Overview

The project's domain background is in Media bias. Media bias exists and shapes the views of their readers subconsciously. Major news outlets are known to be aligned with certain political ideologies – for example, Media Bias Ratings of online news coverage by AllSides rated The Washington Times as Right-leaning while The Guardian is rated as Left-Leaning [5]. However, it is not sufficient to rely only on an article's origin to determine if a news article is biased because media outlets' ideologies are not hard and fast rules. In addition, as most people have personal preferences, individual reporters may exhibit bias in their writings. Hence, to analyze actual article text to determine biases is important.

Problem Statement

Currently, there is not enough research on the intersection of computer science and social sciences in identifying bias in the media [1] . This project will build and deploy a classification model that can suggest if an article text is leaning towards the Democrats or the Republicans. The development of AI-based bias detector tools is becoming popular in recent years [2]. There are different approaches to identifying potential bias in text. For example, a past project [3] made use of averaged feature values of 141 features used in another project to detect fake news, while another [4] made use of news outlet classification. This project will utilize NLP techniques on actual article text.

Metrics

Accuracy score will be used as a measurement of the model's performance. It is used by other studies in measuring media detection models , such as this Support Vector Machine Model [3] to identify bias in articles based on Article Body feature with averaged F1 and Accuracy score: 36.63 and 41.74

II. Analysis

Data Exploration

The dataset is obtained from https://deepblue.lib.umich.edu/data/concern/data_sets/8w32r569d?locale=en and consists of 21004 rows of Article URL links and their corresponding positive, neutral, or negative ratings. Of the 21004 articles, the following ratings exist which show that there are rating overlap:

```
newsLabels["democrat.vote"].value_counts()
```

```
Neutral             13626
SomewhatNegative     3507
SomewhatPositive     2242
Negative             1238
Positive              391
Name: democrat.vote, dtype: int64
```

```
newsLabels["republican.vote"].value_counts()
```

```
Neutral             15192
SomewhatNegative     3114
SomewhatPositive     1401
Negative             1073
Positive              224
Name: republican.vote, dtype: int64
```

Figure 1

Articles with rating overlap (i.e Negative/Positive for both Democrat and Republican) will be removed in data preprocessing. The features used in this project should be articles that are biased against only one political party (either Republican or Democrat) or Neutral for both political parties.

Statistics of article text (figure 2) show that the mean length of articles in the finalized dataset of 1000 articles is 3656 and there is a standard deviation of 2284, showing that article lengths vary widely. However, between Bias group and Neutral group, the difference is reduced – around 200 words difference in mean article length. This project assumes that the article length difference between groups will not be a strong confounding variable.

```
balanceddfarticles['article_length'].describe()
```

```
count     1000.000000
mean      3656.247000
std       2284.405154
min         57.000000
25%       2114.750000
50%       3364.500000
75%       4894.500000
max      18836.000000
Name: article_length, dtype: float64
```

```
balanceddfarticles.groupby(['newlabel']).describe()
```

	article_length							
	count	mean	std	min	25%	50%	75%	max
newlabel								
Bias	500.0	3778.750	2240.194109	57.0	2254.75	3506.5	4942.5	11650.0
Neutral	500.0	3533.744	2323.559334	82.0	2004.50	3279.5	4872.5	18836.0

Figure 2

Algorithms and Techniques

Newspaper3k Library was used to extract article text from each article URL. Python functions were used to clean the acquired article texts of html formatting and special characters. XGBoost algorithm, a gradient boosted trees algorithm, was used to train the model in predicting article bias. It is selected because it is powerful in dealing with tabular data- the format the dataset of this project is in. Initially, five classes of outcomes were included in the study – Neutral, Democrat-Positive, Democrat-Negative, Republian-Positive, Republican-Negative, and a multi-softmax objective was used. However, as the multi-class model returned low accuracy score (0.133 on average) for the classification, a binary model was used instead where the four bias-containing classes were merged into one class –Bias. A binary classification objective is used in this new model.

XGboost Gradient boosting library is used to train the model. The finalized dataset is uploaded to AWS S3, and retrieved in SageMaker. The dataset is label-encoded then split into training and testing sets and transformed into arrays and and csv files for training and testing. XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs). . XGBoost trains iteratively; it adds new trees that predict the residuals or errors of previous trees, altogether combined in the end to make the final prediction. The model uses a gradient descent algorithm to minimize the loss when adding new models [6].

TfidfVectorizer is used to convert article texts into numerical representation. It transforms text to tf-idf vectors. It is a technique which includes stop-words filtering, and weighs words by its importance.

Benchmark

An accuracy score above 0.5 would indicate that the model is better than chance at predicting if an article is biased. The benchmark is set at 0.5 as it would be better performing than a Support Vector Machine Model [3] to identify bias in articles based on Article Body feature with Accuracy score of 41.74.

III. Methodology

Data Preprocessing

Articles text were cleaned such that only content text remain.

Next, articles with similar Negative/Positive ratings for both Democrat and Republican were removed so that now the only remaining articles are articles that are Neutral to both Republican and Democrat, or biased towards one side. Those articles are identified with functions (Figure 2) and identified articles were dropped (Figure 3).

```
def findDouble(df):
    if (df['democrat.vote'] == "SomewhatNegative" )or (df['democrat.vote'] == "Negative"):
        if (df['republican.vote'] == "SomewhatNegative" )or (df['republican.vote'] == "Negative"):
            return "Y"
    if (df['democrat.vote'] == "SomewhatPositive" )or (df['democrat.vote'] == "Positive"):
        if (df['republican.vote'] == "SomewhatPositive" )or (df['republican.vote'] == "Positive"):
            return "Y"
    else:
        return "N"
```

Figure 3

```
doubletodelete=NewsLabelsWithNoNa[NewsLabelsWithNoNa['double_label']=="Y"].index
```

```
NewsLabelsWithNoNa.drop(doubletodelete,inplace=True)
```

Figure 4

Article text were cleaned of non-alphanumeric characters (Figure 4).

```
def clean_articles(txt):
    txt=list(txt)
    txt=[character for character in txt if character.isalnum() or character.isspace()]
    txt="".join(txt)

    return txt
```

```
NewsLabelsWithNoNa["article_text"] = NewsLabelsWithNoNa["article_text"].apply(clean_articles)
```

Figure 5

Implementation

For the XGBoost model, maximum depth of the tree is set to 5. Higher levels may increase the chance of overfitting. Learning rate is set to 0.2. Gamma, the minimum loss reduction required to make further partition on leaf node is set to 4. The minimum sum of instance weight is set to 6. The subsample ratio of the training instances is set to 0.8, early stopping rounds is set to 10, and number of rounds for boosting is set to 300. The Objective of the model is binary logistic.

Refinement

The first attempt to build a XGBoost Model (figure 3) with the 5 categories returned a low accuracy score for each category. Hence, the 4 categories indicating bias are combined into 1 category – Bias. The model the project aims to build is now a Binary Classifier.

```
xgb = sagemaker.estimator.Estimator(container, # The location of the container we wish to use
                                    role,                                    # What is our current IAM Role
                                    train_instance_count=1,                  # How many compute instances
                                    train_instance_type='ml.m4.xlarge',      # What kind of compute instances
                                    output_path='s3://{}/{}/output'.format(sagemaker_session.default_bucket(), prefix),
                                    sagemaker_session=sagemaker_session)

# And then set the algorithm specific parameters.
xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='multi:softprob',
                        num_class=5,
                        early_stopping_rounds=10,
                        num_round=500)
```

```
s3_input_train = sagemaker.s3_input(s3_data=train_location, content_type='csv')
```

```
xgb.fit({'train': s3_input_train})
```

```
2020-04-23 01:51:50 Starting - Starting the training job...
2020-04-23 01:51:52 Starting - Launching requested ML instances......
2020-04-23 01:52:51 Starting - Preparing the instances for training......
2020-04-23 01:53:59 Downloading - Downloading input data...
2020-04-23 01:54:40 Training - Downloading the training image..Arguments: train
[2020-04-23:01:55:00:INFO] Running standalone xgboost training.
[2020-04-23:01:55:00:INFO] Path /opt/ml/input/data/validation does not exist!
[2020-04-23:01:55:00:INFO] File size need to be processed in the node: 777.48mb. Available memory size in the node: 8508.44mb
[2020-04-23:01:55:00:INFO] Determined delimiter of CSV input is ','
```

```
: from sklearn.metrics import accuracy_score
  accuracy_score(y_test, transformerjob)
```

```
: 0.133
```

```
: print(metrics.classification_report(y_test,transformerjob))
```

	precision	recall	f1-score	support
0	0.06	0.49	0.11	71
1	0.24	0.17	0.20	282
2	0.25	0.11	0.15	274
3	0.19	0.09	0.12	230
4	0.00	0.00	0.00	143
avg / total	0.19	0.13	0.13	1000

An attempt was made to use Glove Vectorizer (https://nlp.stanford.edu/projects/glove/) to build a vocabulary from the training articles.

```
import numpy as np
class GloveVectorizer:
    def __init__(self):
        # Load in pre-trained word vectors
        print('Loading word vectors...')
        word2vec = {}
        embedding = []
        idx2word = []
        with open('glove.6B.50d.txt',encoding='utf-8') as f:
      # is just a space-separated text file in the format:
      # word vec[0] vec[1] vec[2] ...
            for line in f:
                values = line.split()
                word = values[0]
                vec = np.asarray(values[1:], dtype='float32')
                word2vec[word] = vec
                embedding.append(vec)
                idx2word.append(word)
        print('Found %s word vectors.' % len(word2vec))
        self.word2vec = word2vec
        self.embedding = np.array(embedding)
        self.word2idx = {v:k for k,v in enumerate(idx2word)}
        self.V, self.D = self.embedding.shape
    def fit(self, data):
        pass

    def transform(self, data):
        X = np.zeros((len(data), self.D))
        n = 0
        emptycount = 0
        for sentence in data:
            tokens = sentence.lower().split()
            vecs = []
            for word in tokens:
                if word in self.word2vec:
                    vec = self.word2vec[word]
                    vecs.append(vec)
            if len(vecs) > 0:
                vecs = np.array(vecs)
                X[n] = vecs.mean(axis=0)
            else:
                emptycount += 1
            n += 1
        print("Numer of samples with no words found: %s / %s" % (emptycount, len(data)))
        return X

    def fit_transform(self, data):
        self.fit(data)
        return self.transform(data)
```

```
vectorizer = GloveVectorizer()
X_trainVectorized=vectorizer.fit_transform(X_train)
```

```
Loading word vectors...
Found 400000 word vectors.
Numer of samples with no words found: 0 / 6295
```

Figure 5

However, accuracy scores returned from using Glove Vectorizer are not better than accuracy scores returned from training and testing sets vectorized with TfidfVectorizer from sklearn.feature_extraction.text. As a larger memory space is required to perform Glove Vectorization than TfidfVectorization, TfidfVectorizer is used to vectorize the article texts in this project.

The dataset containing articles labelled with “Bias” or “Neutral” were randomly shuffled and selected to generate a smaller dataset of 1000 articles. This is to minimize AWS cost – training a model with over 6000 article data led to 3123 billable seconds (figure 6).

```
[02:44:46] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 26 pruned nodes, max_depth=0
[02:44:49] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 24 pruned nodes, max_depth=0

[02:44:52] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 14 pruned nodes, max_depth=0
[194]#011train-merror:0.1605
[02:44:56] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 14 pruned nodes, max_depth=0
[02:44:58] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 18 pruned nodes, max_depth=0
[02:45:01] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 12 pruned nodes, max_depth=0
[02:45:04] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 22 pruned nodes, max_depth=0
[02:45:07] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 18 pruned nodes, max_depth=0
[195]#011train-merror:0.16075
[02:45:11] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 16 pruned nodes, max_depth=0
[02:45:14] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 18 pruned nodes, max_depth=0
[02:45:17] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10 extra nodes, 24 pruned nodes, max_depth=5
[02:45:20] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 16 pruned nodes, max_depth=0
[02:45:22] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 28 pruned nodes, max_depth=3
[196]#011train-merror:0.161
[02:45:26] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 16 pruned nodes, max_depth=0
[02:45:29] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 8 extra nodes, 10 pruned nodes, max_depth=4
[02:45:32] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 22 pruned nodes, max_depth=0
[02:45:35] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 30 pruned nodes, max_depth=0
[02:45:38] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 14 pruned nodes, max_depth=3
[197]#011train-merror:0.1605
[02:45:42] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 20 pruned nodes, max_depth=0
[02:45:44] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 16 pruned nodes, max_depth=0

2020-04-23 02:46:02 Uploading - Uploading generated training model
2020-04-23 02:46:02 Completed - Training job completed
[02:45:47] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 16 pruned nodes, max_depth=0
[02:45:50] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 22 pruned nodes, max_depth=0
[02:45:53] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 20 pruned nodes, max_depth=0
[198]#011train-merror:0.1605
Stopping. Best iteration:
[188]#011train-merror:0.16

Training seconds: 3123
Billable seconds: 3123
```

IV. Results

Model Evaluation and Validation

Testing the model show an accuracy score, recall score, and f1-score of 0.66. The model would be able to predict if an article text exhibit bias at an accuracy rate of 66%, but would not be able to pinpoint the exact direction of the bias.

Justification

This model has an accuracy score that is lower than a deep learning model from a media bias study [4] where the author obtained a 0.9 accuracy score for a model that predict Left or Right leaning articles. However, the model in this project predicts Neutral stance or Biased Stance, and performs better than another Support Vector Machine Model [3] to identify bias in articles based on Article Body feature with its averaged F1 and Accuracy score of 36.63 and 41.74.

V. Conclusion

Reflection

The model could pinpoint if an article text is potentially biased or is neutral. Further tunings on the dataset or the model will improve its function to predict more nuances in terms of biases.

Improvement

Article text length could be padded to enforce standardization in training and testing dataset.

[1] Hamborg, F., Donnay, K. & Gipp, B. Automated identification of media bias in news articles: an interdisciplinary literature review. Int J Digit Libr 20, 391–415 (2019). https://doi.org/10.1007/s00799-018-0261-y

[2] https://www.forbes.com/sites/simonchandler/2020/03/17/this-website-is-using-ai-tocombat-political-bias/#6ef847446f4c

[3] Baly, Ramy & Karadzhov, Georgi & Alexandrov, Dimitar & Glass, James & Nakov, Preslav. (2018). Predicting Factuality of Reporting and Bias of News Media Sources. 10.18653/v1/D18-1389.

[4] https://towardsdatascience.com/media-bias-detection-using-deep-learning-libraries-inpython-44efef4918d1

[5] https://www.allsides.com/media-bias/media-bias-ratings

[6] https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html