**ML Assignment**

**Questions:**

1. We have chosen the ‘News Headlines for Sarcasm Detection’ dataset from Kaggle. This dataset consists of 436 sarcastic headlines from The Onion and 524 serious/non-sarcastic headlines from the HuffPost. We have chosen this dataset, because not many people can recognize sarcasm – especially on social media, thus we would like to know what words in a sentence makes it sarcastic or serious.

Sample: 2 of 960 records

"dataset\_sarcasm": [

    {"is\_sarcastic": 1, "headline": "thirtysomething scientists unveil doomsday clock of hair loss", "article\_link": "https://www.theonion.com/thirtysomething-scientists-unveil-doomsday-clock-of-hai-1819586205"},

    {"is\_sarcastic": 0, "headline": "dem rep. totally nails why congress is falling short on gender, racial equality", "article\_link": "https://www.huffingtonpost.com/entry/donna-edwards-inequality\_us\_57455f7fe4b055bb1170b207"},

For our purposes we only used “is\_sarcastic” as our label and made a Bag of Words Model using the “headline” such that every word in the dataset is our features.

* “is\_sarcastic”: indicates whether the record is sarcastic (1) or serious (0).
* “headline”: indicates the news headline of article.
* “article\_link”: indicates the link of the full article online.

1. In order to have a concise dataset, we assigned the “dataset\_sarcasm” array to a dictionary in Python. We looped through the dictionary and added only the necessary data to a numpy 2D array which consisted of the “is\_sarcastic” value (label) and “headline” (features).

Sample:

[["is\_sarcastic": 1, "headline": "thirtysomething scientists unveil doomsday clock of hair loss”],

["is\_sarcastic": 0, "headline": "dem rep. totally nails why congress is falling short on gender, racial equality”]]

Once we have our concise dataset, we can turn this raw datapoints into meaningful values that cam be used by our algorithms. We convert our labels into real-numbered values and convert our headlines into a Bag of Words vector:

All the Bag of Words vectors go into a feature matrix that is Number of datapoints x Length of Vector (each element in vector is the feature for specific headline with length being the vocabulary list).

* 1: word from vocabulary list is present in datapoint(headline)
* 0: word from vocabulary list is NOT present in datapoint(headline)

Fortunately, our dataset did not contain y missing values or strings. In case of missing values in the dataset we would:

* Replace the missing or corrupt with the median, mode, mean or unique value.
* Delete record with corrupt/missing value.

We split the feature matrix as well as the corresponding labels in 80:20, where 80% is the training data and 20% is the test data. We also further split the training data into 80:20 where 80% is the training and 20% is the validation data. We split the data to minimize the data discrepancies as well as expose the model to real and random effects. We used validation data to optimize the best model to detect sarcasm.

1. I chose regularised logistic regression as it the linearly discriminative model in which we can determine an unbiased prediction value based on the categorical dataset. Logistic regression is also one of the best models to use with a Bag of Words model as it is easy to implement as well as compatible with the format of word vectors. I used regularisation to prevent the model from over-fitting as well as improve the reliability and accuracy of model.

**NOTE**: Hit = how many predictions are correct Miss = how many predictions are wrong

λ value = 0.02 α = 0.1:

training data

Hit: 508

Miss: 92

84.66666666666667

validation data

Hit: 119

Miss: 49

70.83333333333334

test data

Hit: 121

Miss: 71

63.020833333333336

λ value = 0.2 α = 0.1:

training data

Hit: 369

Miss: 231

61.5

validation data

Hit: 101

Miss: 67

60.11904761904761

test data

Hit: 113

Miss: 79

58.854166666666664

λ value = 0.002 α = 0.1:

training data

Hit: 594

Miss: 6

99.0

validation data

Hit: 125

Miss: 43

74.40476190476191

test data

Hit: 124

Miss: 68

64.58333333333334

The basis function for the model:

Z = W.X = w0 + w1x1 + … + wnxn

* W = [w0, w1, w2, …, wn] //weights are randomly generated between -0.5 and 0.5
* X = [x0, x1, x2, …, xn]
* n = number of features in datapoint (headline) which are the words present in vocabulary (length of word vector).

The gradient descent of model:

α = 0.1 λ value = 0.002:

training data

Hit: 508

Miss: 92

84.66666666666667

validation data

Hit: 119

Miss: 49

70.83333333333334

test data

Hit: 122

Miss: 70

63.541666666666664

α = 0.01 λ value = 0.002:

training data

Hit: 528

Miss: 72

88.0

validation data

Hit: 123

Miss: 45

73.21428571428571

test data

Hit: 108

Miss: 84

56.25

**Hyperparameters for this algorithm are weights, learning rate and regularised lambda rate. The weights are randomly initialized between -0.5 and 0.5 so there is not any bias. Therefore, I chose the learning rate as 0.1 as it indicates a more accurate predictions when I update the weights. I chose the regularised lambda value as 0.02 as it gives more accurate label predictions for the test data.**

TEST DATA Confusion Matrix for 192 records (λ value = 0.002 & α = 0.1):

|  |  |  |  |
| --- | --- | --- | --- |
| Predict Sarcasm? | | | |
| Really Sarcastic? |  | Sarcastic | Serious |
| Sarcastic | 57 | 32 |
| Serious | 38 | 65 |

* Classification Error: 70/192 = 36.46%
* Accuracy: 122/195 = 63.54%
* False Sarcastic Rate = 38/103 = 36.89%
* False Negative Rate = 32/89 = 35.96%
* True Positive Rate = 57/89 = 64.04%
* Precision = 57/95 = 60%

Logistic Regression Model:

α = 0.1 λ value = 0.002:

training data

Hit: 508

Miss: 92

84.66666666666667

validation data

Hit: 119

Miss: 49

70.83333333333334

test data

Hit: 122

Miss: 70

63.541666666666664

Random Guess:

guess:

Hit: 95

Miss: 97

49.47916666666667

The Model performs better at predicting sarcasm than the random guess.

Time taken for model:

training data

Hit: 508

Miss: 92

84.66666666666667

validation data

Hit: 119

Miss: 49

70.83333333333334

test data

Hit: 121

Miss: 71

63.020833333333336

guess:

Hit: 98

Miss: 94

51.041666666666664

Runtime of the program is 442.0488212108612

Resources:

* <https://stackoverflow.com/questions/13610074/is-there-a-rule-of-thumb-for-how-to-divide-a-dataset-into-training-and-validation>
* <https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html>
* <https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971>
* <https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/>