Article Classification

BY: Vincent Welsh

Outline

- EDA
  - Scrub Methodology
  - EDA Methodology
  - Train Test Split Methodology
- Modeling Process
- Results
- Additional Data
- About Me
- Project Technical Details

EDA

- Scrub Methodology
  - To start, it was necessary to remove many unwanted characters e.g. <>/\
  - The data was formatted for older programming languages so there are markers indicating the start and end of a certain text.
  - Once I removed unhelpful characters, I removed rows of data where the topics column was empty.

EDA

- EDA Methodology
  - In order to understand the nature of the text I visually inspected the following characteristics within the text & title columns:
    - Length of words
    - Number of words
    - Word Length
      - The characteristics above are all important for preparing the data for a neural network. I.e. They provide insight into how much data the model will be trained on and at what frequency the data will be inputted.
    - Stop Words
    - Most Common Non Stop Words
      - These data I inspected in case my models were performing poorly later in the process. I did not remove stop words because I knew that I would be using the GloVe model that includes stop words.

EDA

- Train Test Split Methodology
  - For this project I decided to train test split three twice to include a validation set. To split the data I imported Train_Test_Split from sklearn.
  - I decided to make a validation set because I knew I would be working with Keras. I use the test set as the val_data parameter when fitting my network. Then, I use the validation set to validate the model and analyze performance.
  - The test and validation sets are each 15% of the training data, leaving 70% of the data for training.
  - Additionally I set the stratify parameter to True.

Modeling

- The Models I created were trained on a binary decision between the two most frequent topics within the data (earn & acq).
- The reason I decided to only use the top two classes was to create a highly accurate model.
  - As data becomes more imbalanced accuracy loses value and extra model bias is added.
  - When class imbalance is minimal it is easy to mitigate by adding penalizing weights to the model's loss function.
  - Considering this and the class instances below I decided to only teach the model the top two classes.
    - Class one has 3746 instances
    - Class two has 2147
    - The next class has 361 instances.

Modeling

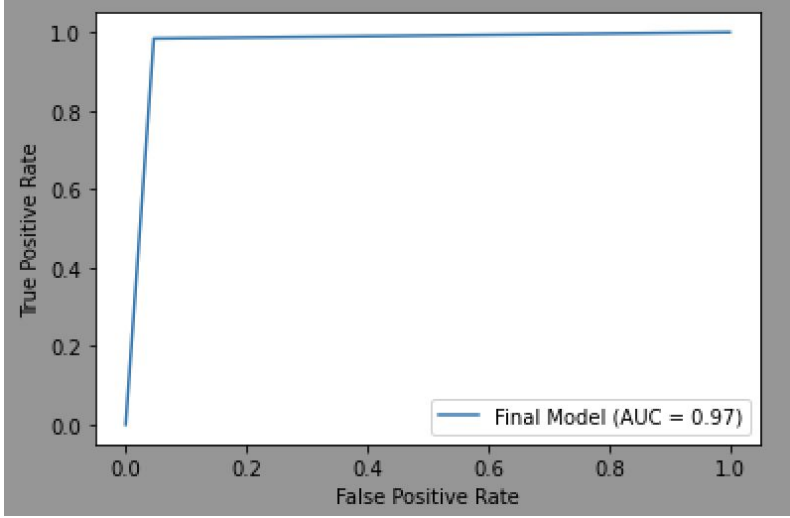
- To determine the best possible model I prefer to follow the diagram on the right ⇨⇨⇨
- I started with a baseline model and evaluated it. Then I made several other models, each one building off the past models.
    - e.g If the past model overfit I added drop out layers
- Once I determined the best model, I created several more models but with different batch sizes.

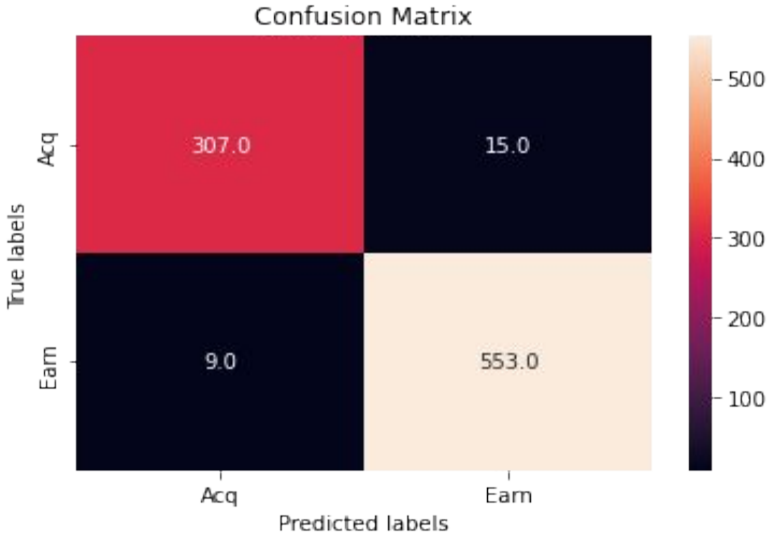
Results

- Predicting:
  - When provided with the validation data, the model predicts a probability between 0-1.
    - If the probability is greater than .5 the model predicts 1 (majority class, earn)
    - If the probability is less than .5 the model predicts 0 (minority class, acq)
- Evaluating:
  - To determine the best model, before adjusting batch size and learning rate, I measure model performance based on:
    - Model Training History Graph
      - Line graph denoting loss and a accuracy of training and validation sets over epochs.
    - Classification Report Metrics
      - Precision, Recall, F1 of each class
      - Overall Accuracy

Results

- Final Model Evaluation:
  - To understand the final model performance I used an ROC curve visualization.
  - This is a preferred evaluation method because it is a strong visual representation of several metrics like Precision, Recall, and Accuracy.
  - Additionally, I looked at a confusion matrix.
  - A confusion matrix allows me to know where the model is making the most mistakes.
    - E.g. In this case it is predicting Earn 15 times when the true label is Acq





Additional Data

- Provided a working model:
  - I would classify the unlabeled data.
  - Determine subcategories within the provided categories.
  - Separate other features from the data to be used as the label e.g.
    - Location
    - Author
    - Multi Topic Text
  - Add sentiment analysis
  - The end result would be a model that could
    - Predict Topic(s)
    - Predict other attributes like author
    - Use sentiment to aid in prediction

A Bit About Me

- Professional:
- 3 years programming experience have given me a strong intuition for software best practices e.g.
  - Accomplish tasks in the most computationally efficient way
  - PEP-8
  - Effective commenting
- I've worked in high paced team environments for 10 consecutive years, food service industry & NSA, which has allowed me to develop a strong ability to adapt to new team environment. Plus, I have a high stress tolerance in the workplace.
- Very eager to learn/grow as a machine learning engineer

- Personal:
- STEM Nerd, Vegan, Deep Thinker, Perfectionist, Family Oriented
- Hobbies:
- Dog walks, live music, comedy, reading, podcasts, youtube drama, working out, and skincare

Project Technical Details

Time Spent : ~25 hours

- In order to access the reuters data in the notebook you must double click on the Interos_MLApplication_Welsh folder that I shared and select Add Shortcut To Drive. Otherwise the data will not be loaded.

Validation Data Method:

- There are two methods to predict on unseen similar data:
  - process_file_data
    - This method has one input parameter file, and it should be the file pathname. This method will return a list of 1s and 0s where 1 == earn and 0 == acq.
  - process_raw_text
    - This method also has one input parameter text_, and it should be a string. This method will return a string of ‘earn’ or ‘acq’
- Both methods can only be executed once the notebook has been run. I left two example cells at the bottom of the notebook showing their usage.