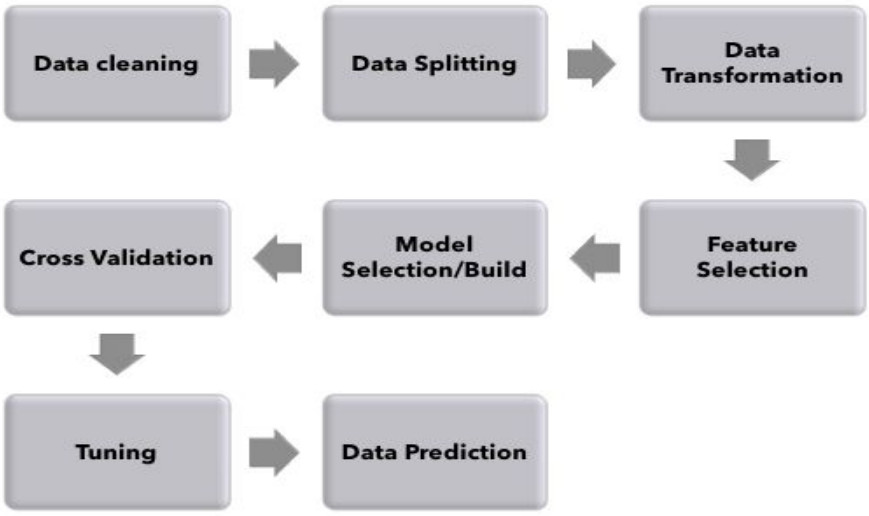
I. Vision

For the final project, I aim to implement a sentiment-analysis on examining how immigration/refugee issues (government policy focused) is portrayed by several prominent newspapers such as The Washington Post, The Wall Street Journal, The Times, The Economist, The New York Times, MSNBC, Fox, NPR, etc.; selections would be made to cover entire spectrum of the partisanships evenly. From this analysis, I aim to find the link between general partisanship of newspaper to their portrayal on immigration/refugee issues.

II. Background

Creating the model for this specific sentiment-analysis, I will take the strategy of building upon the general sentiment-analysis model exists in public to predict sentiment on news data. Considering the high level of difficulty of finding news data (tagged data) itself, training the model with prepped public dataset came up to be the most efficient and effective way to operate this project. The basic algorithm outline of the model I will reference is “to combine the understanding of semantics and symbolic representations of language” and to train the model “from labeled data and predict the label of new/unseen data points” (Bhandari 2018). The training dataset for the model will be one with the news titles, tagged ones (supervised learning) from Kaggle with 1 for positive, 0 for neutral, and -1 for negative responses (2018). Tools would be as following: language - Python, scraping library - beautifulSoup, machine learning library - scikit-learn, data wrangling - Pandas and Numpy, plotting - matplotlib (2018). The main plan for this project will follow training, testing, and cross-validation process with graphing ‘bias’ and ‘variance’ to depicts the fluctuations and sensitivity of the model. In order to measure the performance of the model, I will plot the learning curve to figure out the spot with the minimized bias and variance.

<Model Development>



III. Implementation

- Data Loading and Splitting
    - 80/20 divide for training and testing data set.

- Feature Extraction
    - This process is needed to turn text documents into a numerical representation by vectorization with the process of tokenization (converting text into a list of words by removing stop words and punctuations), frequency counting (counting the occurence of individual token into each of the document), and normalizing (reducing the importance of token having frequent occurrence in a majority of documents by giving it lower weight)
    - The dataset will be represented by the occurrences of words instead of their relative position.
- Model testing
    - Term frequency-Inverse document frequency (TfiDF) evaluates how important a word is to a document in a collection or corpus

- Model Selection
    - Support vector machine (SVM)

- Cross validation
    - It accesses the predictive performance of the model.
    - Using cross validation will allow us to observe how the model will behavior in new data in terms of accuracy.
    - Instead of using a subset of training dataset for validation, it is running sequence of fits on the model using subsets which act as both training and validation set. Cross_val_score will be used to achieve this.
        - Cross_val_score: uses StratifiedKFold to create k folds of data then compute accuracy in individual run. For each of the k folds:
            - Model raining is done in k-1 of the folds as training data
            - The trained model is validated against remaining part of the data
    - There will be two types of cross validation, for model selection and parameters tuning

- Data Prediction
    - By the best estimator given by GridSearchCV, the model predicts test data.
        - Here, the test data will be the dataset of my own with the article titles from different sources. Dataset I create will be tested on the model trained with the dataset from the Kaggle.
        - This process will give the score according to the tagged values (-1, 0, 1) for the titles from each source, and predicts the type of sentiment it gives away.