# TACKLING "NON DEEP" "CLASSIFICATION PROBLEMS" FROM TEXT AND/OR NUMERICAL DATA

Coding track discussion 02

Bhaskarjyoti Das

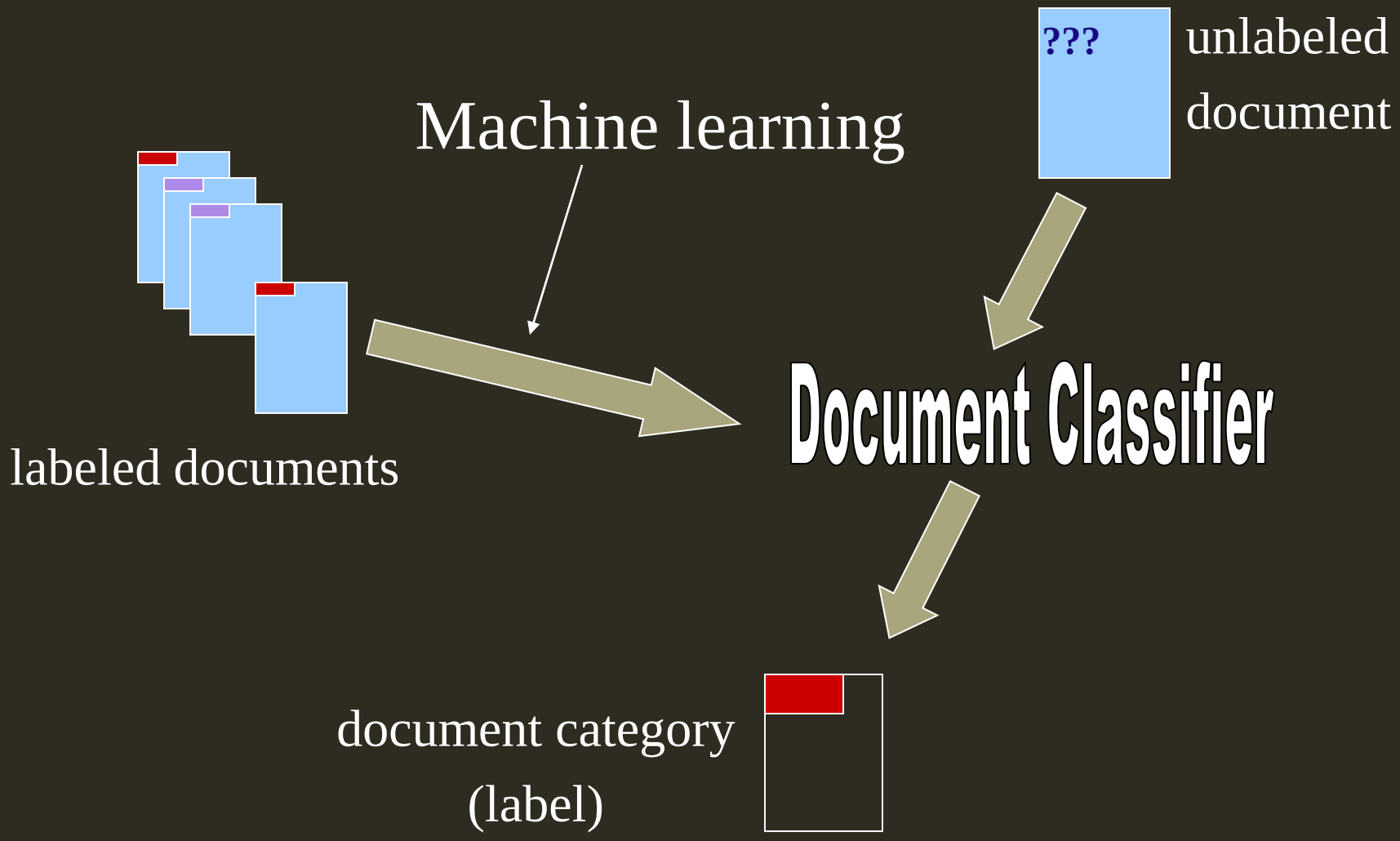
# WHAT ARE WE PLANNING TO COVER ?

1. Pre processing is a common step for all NLP projects i.e. pure NLP, machine learning or Deep Learning on text. Pre processing non-text data is also important as we mostly get mixed data set.
2. Many steps are common between text and non-text !
3. Will look at usual steps for typical ML problems (that may involve text)
4. Will use Scikit Learn by default and Keras in some cases like deep learning. Have not yet looked at tensorflow in details. Generally the blue print or template is same

# NORMALIZING TEXT DATA

Bhaskarjyoti Das

# DOCUMENT CATEGORIZATION



# NORMALIZING TEXT – USUAL STEPS

1. Clean out text (this is really specific to the data)
   - Split by white space after loading the text into string (but it preserves punctuation as separate token)
   - Split by selecting for string of alphanumeric  using regex ( it may split the contractions though)
   - Split by white space and then use regex to replace all punctuation by space ( seems to fix issues)
   - Filter out all non-printable characters
   - Normalize to lower case
2. Using NLTK like library to tokenize into sentences
3. Using NLTK like library to tokenize sentences into words
4. Remove all words that are not alphabetic  (suppose there are numeric stuff)
5. Remove all stop words ( you can enhance this list)
6. Do stemming or lemmatization

# NORMALIZING TEXT : FEW POSSIBLE ADDITIONAL STEPS

7. Expand contractions i.e. can't -> cannot
8. Handle domain specific words
9. Remove HTML/pdf  fragments /markups
10. Handle amount, phone numbers, dates that have specific formats
11. Handle misspelt words and typos
12. Handle Unicode characters ( if any) into normalized forms such as UTF 8
13. Correcting repeating characters ( typically typos)

***No "one size" fits all ! At the end, we are in a position to create vocabulary.***

# CREATE A VOCABULARY THAT WE WANT TO WORK WITH

1. It is important to work with a smaller and more meaningful vocabulary.
2. Ngrams are typically sparse. If we consider all n-grams in the text data we have, there will be many ngrams having very less frequency (Zipf's law).
3. If we are working with very large vocabulary, we may like to set a minimum frequency and decide to work with smaller vocabulary to save on memory and CPU requirements.
4. Vocabulary hyper parameter = min_frequency
5. Frequency need NOT be the only strategy. It can be selecting words carrying emotion for sentiment analysis job !

# UNDERSTANDING TEXT SYNTAX (OPTIONAL)

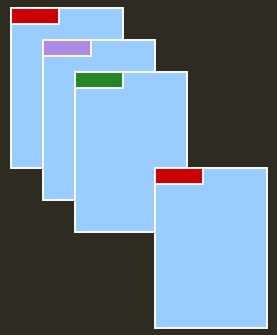
1. This is really an optional step.
2. Shallow parsing or chunking is essentially useful to extract meaningful chunks out of the sentences
3. Dependency and constituency parsing essentially provides deep parsing of the sentences, provides a structure showing syntactic as well as semantic dependencies
4. STOP WORDS elimination  before or after chunking ?
   - These parsers typically will require POS Tagged input and POS Tagging needs full sentences preferably. So, for cases like chunking, we should avoid stop words elimination before POS Tagging !

# VECTORIZING TEXT

1. We have to convert text into numbers that software can process i.e. words have to be converted into numbers
2. Typically a Bag Of Words (BOW) model will be used
   - CountVectorizer (An encoded vector is returned with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document)
   - TfIdfVectorizer ( a normalized vector for tfidf count)
   - HashingVectorizer (returns a long integer but it is one way i.e. from hash, you cannot reconstruct inputs unlike the other two)

# ARCHITECTURE

Feature construction



labeled documents

vectors of n-grams

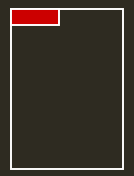
Subproblem definition

Feature selection

Classifier construction


unlabeled document

Document Classifier



document category (label)

# CAN WE DO TEXT CLEANING AND VECTORIZATION IN KERAS ?

1. Should investigate just in case you are doing deep learning with keras. Keras gives some APIs
2. There are several convenience functions.
   - Splitting text into words with text_to_word_sequence (takes care of lower case, punctuation and split by space). You can then estimate the size of vocabulary.
   - Function one_hot() will do the above plus encode the document with the hashing trick using the size of vocabulary
3. There is a tokenizer API that offers standard bag of words encoding schemes (binary, count, tfidf etc.) once we fit into the text data
   - The function text_to_matrix() will let you create the matrix that you will work with

# NORMALIZING NON-TEXT NUMERICAL DATA

Bhaskarjyoti Das

# UNDERSTAND THE DATA BEFORE YOU GET CRACKING

1. Load into Pandas that makes it easy for subsequent processing
2. Look at the shape() : too many rows means long time to train and too many columns means possible curse of dimensionality ( do you want to work with so many ? )
3. Get the data type : which attribute requires converting to float etc. ?
4. Describe () : mean, median, mode, $25^{th}$, $50^{th}$, $75^{th}$ percentile, min and max value, standard deviation etc. for each attribute . Any insight ?
5. Missing value ( NA) strategy ? Example - Row having more than k NA, delete, replace all NA in a column with a particular average ( mean, median etc) etc.

# UNDERSTAND THE DATA BEFORE YOU GET CRACKING (2)

7. How imbalanced my dataset is : count the number and frequency of classes in the target variable. If the dataset is imbalanced, it will influence classification adversely.
   - Clamp as lowest minimum count
   - Synthetically generate data for minority classes ( should have the similar statistical distribution of existing data)
8. Which pairs of attributes are correlated ? Calculate correlation matrix .
   - Useful insight in choosing regression parameters and features in classification
9. Find the distribution of each attribute
   - Univariate visualization : Histogram, Density plot ( same as histogram) and Box–whisker plot ( it shows where most of the values are and whether it is skewed)
   - Bivariate visualization : correlation matrix plot and scatter plot
10. If the algorithm being used assume Gaussian for attribute but the distribution of the attribute is shifted ( skewed), you should address skew. How do you get to know the distribution of the attributes ?

# PREPARE THE NON TEXT DATA FOR MACHINE LEARNING WORK

1. Why ?
   - To best expose the structure of your problem to the modelling algorithms.
   - Different algorithms make different assumptions about your data and may require different transforms.
2. Rescaling the attributes so that all have the same MIN-MAX scale
   - MinMaxScaler() in scikitlearn
   - Useful for all optimization algorithms like Gradient Descent
   - Useful for all algorithms that use weights ( KNN, ANN, Regression)

# PREPARE THE NON TEXT DATA FOR MACHINE LEARNING WORK (2)

3. When the attributes follow Gaussian, transform them to Standard Gaussian ( mean 0 and SD 1) by StandardScaler()
   - Works well for algorithms that assume a Gaussian Distribution of input variables ( linear regression, LDA, Logistic Regression )
4. Binarize using a threshold ( 0s and 1s)
5. Normalize: Normalization refers to rescaling real valued numeric attributes into the range 0 and 1.
6. Which is the best ?
   - Just try out many options .
   - Sometimes algorithms deliver better value without these !

# ONE HOT ENCODING FOR NOMINAL OR ORDINAL NUMERIC DATA

1. Attribute can be numeric but in a nominal ( different values for different countries) or ordinal scale ( happiness in a scale of 0 to 5)
2. Unless handled, these can be interpreted differently
3. One hot encoding converts them into multiple attributes ( scale of 0 to 5 means 5 separate attribute having 0 or 1 as value)

# MACHINE LEARNING FROM NORMALIZED DATA

Bhaskarjyoti Das

# ADDRESS IMBALANCED DATASET IF IT IS A CLASSIFICATION PROBLEM

1. A dataset will have bias when there is imbalance in classes of the training samples.
2. Example – we are considering movie review dataset where number of negative reviews is 10 times that of positive reviews. So classifier will tend to predict negative.
3. One simple way of addressing the balance is  to make sure that the least common class defines  the maximum number of examples of any other candidate class.
4. If the least common class is too low, we need to see how new dataset can be generated preferably of similar statistical property.

# SELECTING FEATURES

1. Challenge :
   - Typically too many possible features
   - Good features help
   - Bad features adversely impact
   - Is there an automatic strategy  to select features ?
2. ScikitLearn offers at least 4 options
   - Statistically select the K best features based on correlation with class variable. The function can use $Chi^2$ function for non-negative features.
   - Recursive feature elimination ( RFE) : the library itself builds a classification model, tries out by removing attributes one by one and gives the n best possible ones !
   - Dimensionality reduction ( PCA) : does not select features but **extracts features** that capture maximum amount of variances !
   - The bagged decision tree models like ExtraTreesClassifier and RandomForest can also provide feature_importances_ that we can use

# CREATE TRAIN TEST SPLIT AND DO K FOLD CROSS VALIDATION

1. We should do a train test split of the dataset
2. It is common to use 67% of the data for training and the remaining 33% for testing
3. But this is only one such possible split
4. K fold  Cross-validation is an approach that we can  use to estimate the performance of a machine learning algorithm to achieve **less variance than a single train-test set split**.
5. This library will return mean and standard deviation of the performance measures such as accuracy for all k folds.
6. There are other variations of K fold i.e. repeated random test train split and leave one out .
7. But K fold is the gold standard!

# CHOOSE THE PERFORMANCE METRICS OF THE ALGORITHM

1. Classification accuracy
   - Classification accuracy is the number of correct predictions made as a ratio of all predictions made. Use it only when there are an equal number of observations in each class(which is rarely the case).
2. Logarithmic loss function
   - Logarithmic loss (or logloss) is a performance metric for evaluating the predictions of probabilities of membership to a given class.
3. ROC curve : Area under ROC Curve (or AUC for short) is a performance metric for binary classification problems. The AUC represents a model's ability to discriminate between positive and negative classes. An area of 1.0 represents a model that made all predictions perfectly.
4. Confusion matrix

yijyij is a binary indicator of whether or not label j is the correct classification for instance i

# LOGARITHMIC LOSS FUNCTION

1. It is a classification loss function. Most competitions like Kaggle use this. Why ?
2. Logloss $= -1/N \sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log p_{ij}$ where N is the number of instances, M is the number of classes/labels, p is the model probability of assigning label j to instance i. $y_{ij}$ is the binary indicator of whether or not label j is the correct classification for instance I
3. Note that for each instance only the term for the correct class actually contributes to the sum.
4. Log Loss quantifies the accuracy of a classifier by penalising false classifications. Minimising the Log Loss is basically equivalent to maximising the accuracy of the classifier
5. In order to calculate Log Loss the classifier must assign a probability to each class rather than simply yielding the most likely class.

# METRICS FOR REGRESSION

1. **The Mean Absolute Error (or MAE) :** the sum of the absolute differences between predictions and actual values. It gives an idea of how wrong the predictions were.
2. **Root Mean Squared Error (or RMSE) :** Taking the square root of the mean squared error converts the units back to the original units of the output variable and can be meaningful for description and presentation.
3. **The $R^2$ or coefficient of determination** : provides an indication of the goodness of t of a set of predictions to the actual values.
4. **Adjusted $R^2$** with some regularization involved with just **$R^2$**

# TRY OUT DIFFERENT LEARNING ALGORITHMS OF DIFFERENT TYPES

1. We can now try out different ML algorithms on the prepared data sets
2. For example, both linear and non-linear classifiers can be attempted
3. Which class does better depends on whether the classes are linearly or non-linearly separable !
4. Linear classifiers : Logistic regression, SVM, LDA etc.
5. Non linear classifiers : naïve Bayes, K NN , ANN etc.

# COMPARING PERFORMANCES OF DIFFERENT MODELS

1. Most ML libraries will make it easy
2. For scikitlearn, we can get the result of each fold of K–fold cross validations for each model.
3. The above can be used to make a box and whisker plot of mean and standard deviation of accuracy for all models

***At the end of this stage, few models should look more promising !***

# DATA LEAKAGE AND PIPELINE

1. **Data leakage during data preparation step :** preparing your data using normalization or standardization on the entire training dataset before learning would not be a valid test because the training dataset would have been influenced by the scale of the data in the test set.
   - pipeline( ) will have estimators[] containing standardization functions as well as models
   - This pipeline itself, once constructed, will be executed under the control of k-fold cross validation function thus avoiding data leakage
2. **Data leakage during feature extraction :** The features may be extracted in different ways ( say PCA gives top 3 and statistically selecting 3). There is a feature_union () that allows combination of these features and then passing the feature_union as parameter to estimator, estimator as parameter to pipeline and pipeline as parameter to k fold cross validation

# YOU MAY CONSIDER PUTTING EVERYTHING IN A PIPELINE

1. **Pipelines automate ML workflows.** Pipelines allow a linear sequence of data transforms ( often very standard) to be chained together culminating in a modelling process that can be evaluated.
2. **Pipelines also help in avoiding data leakage** trap in machine learning exercises

# ACHIEVING BETTER PERFORMANCE

Bhaskarjyoti Das

# ENSEMBLE STRATEGY TO IMPROVE PERFORMANCE MORE

1. **Bagging :** The final output prediction is averaged across the predictions of all of the sub-models ( Decision tree, Random Forest, KNN etc)
2. **Boosting** (Stochastic Gradient Boosting, Adaboost)
3. **Voting**
4. You have all these classifiers in scikitlearn and in other libraries.

# FINDING THE BEST HYPER PARAMETERS TO IMPROVE PERFORMANCE EVEN MORE

1. **Algorithm tuning** is also called hyper parameters tuning
2. **Grid search :** methodically build and evaluate a model for all parameters option specified in some kind of grid. The key is : you do not specify all parameter options
3. **Random Search :** sample algorithm parameters from a random distribution (i.e. uniform) for a fixed number of iterations. A model is constructed and evaluated for each combination of parameters chosen.
4. The function outputs the best performance and the parameter values leading to that.
5. Once you have discovered the **best hyper parameters, redo your pipeline.**

# SAVE YOUR MODEL AND LOAD IT

1. You need to save your model once finalized and later on load the model to do classification on new test data
2. Python pickle() is the standard serialization library
3. Sometimes for lazy learning models like KNN, you store lot of data as well. Consider scipy joblib() as a more efficient option
4. Watch out as these things always create problem
   - Python version (major and minor)
   - Python libraries the were used while creating the model ( version )
   - Specially numpy and scipy versions

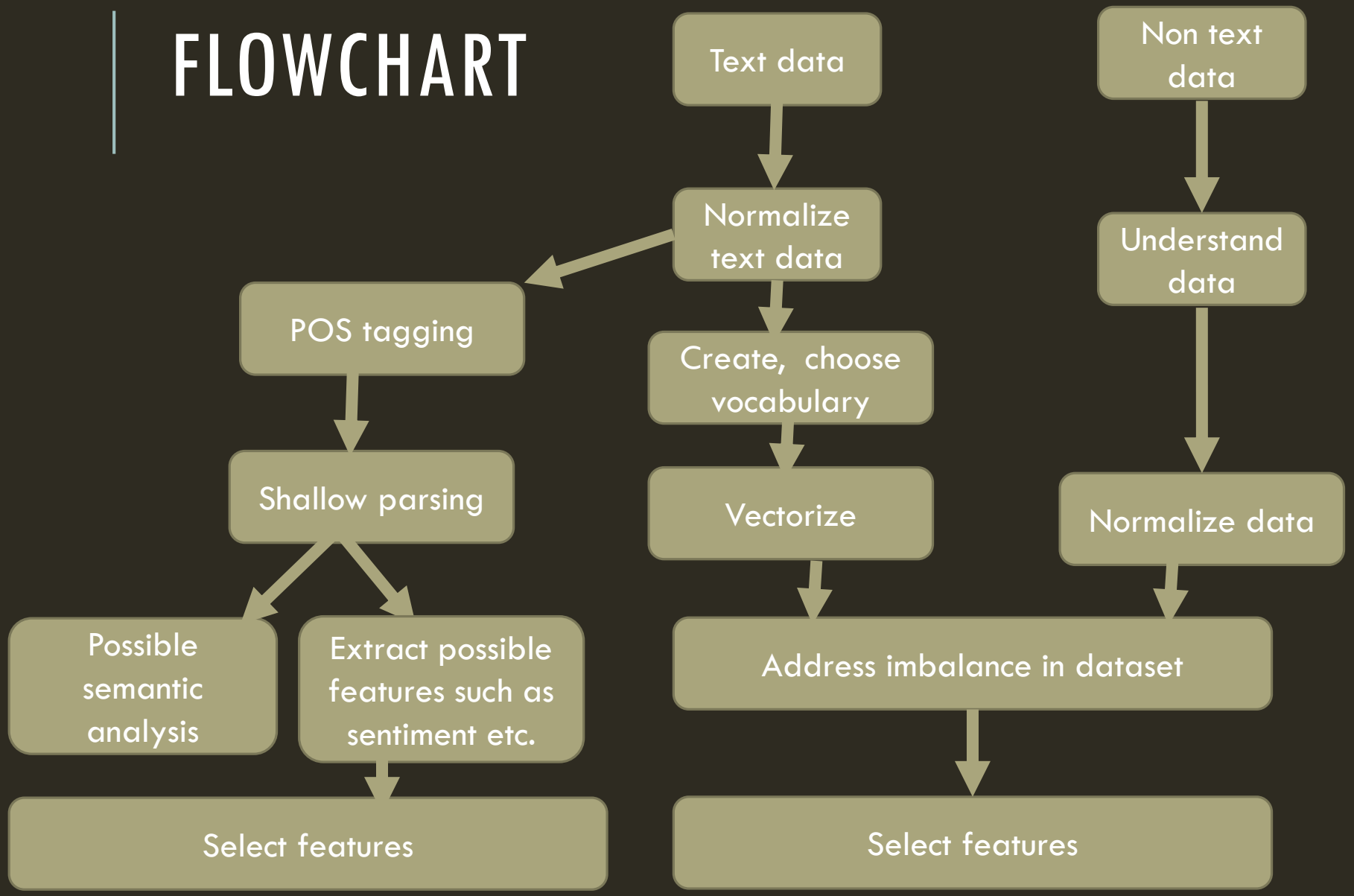
# SUMMING UP

Bhaskarjyoti Das

# HOW DO YOU POSSIBLY WORK WITH BOTH TEXT AND NON TEXT DATA ?

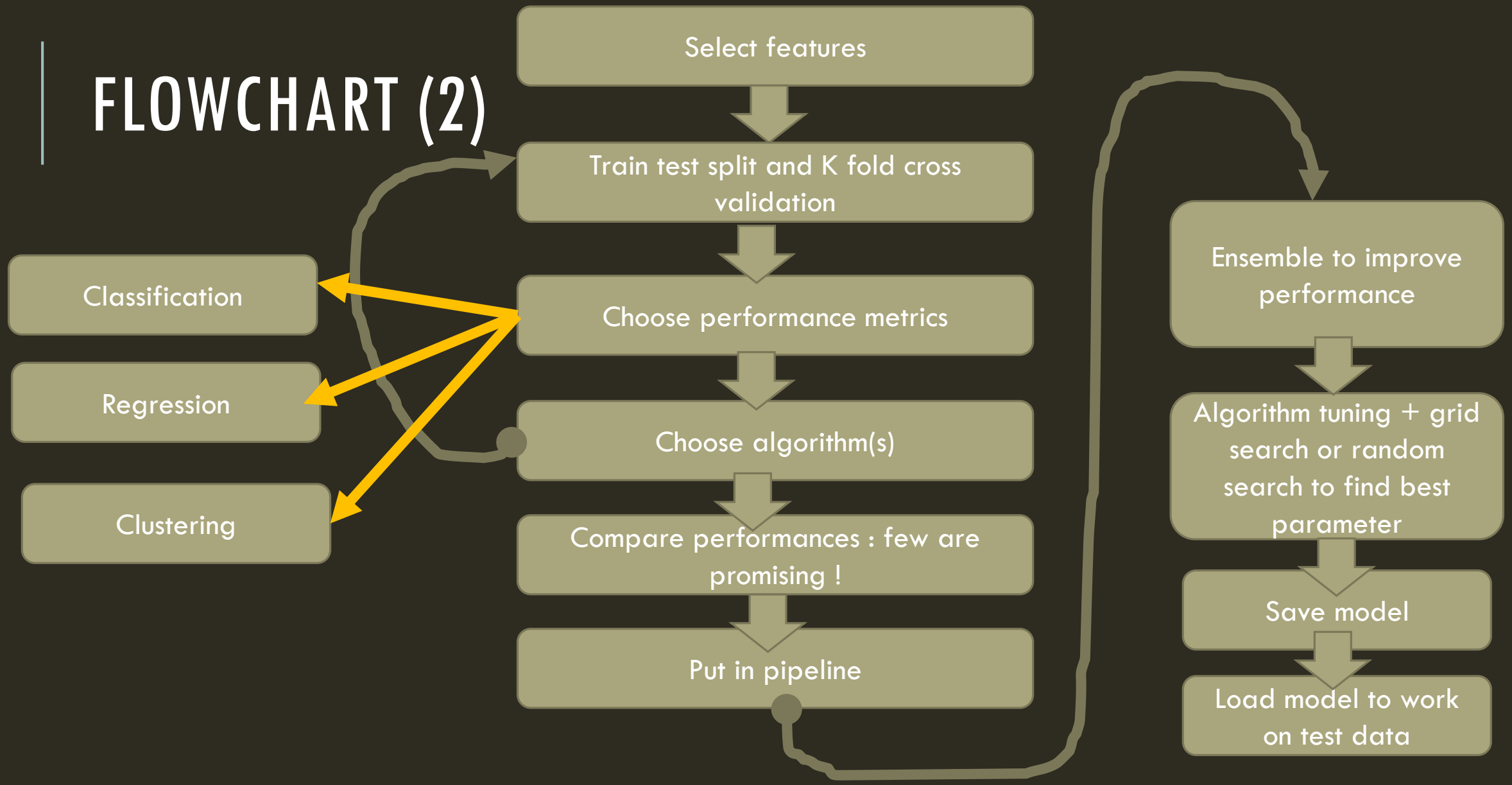
1. Text data will lead to sparse features and non text data will lead to dense features. So, these cannot be generally combined
2. If combined, ensure to normalize but it may not still work due to sparsity issue
3. Possibly you can get a model based on text features alone, get the prediction as an attribute to be incorporated into the other model using non-text features ( make sure to normalize any way)
4. Yet another option is to extract features instead of using n-gram as feature. Example – sentiment polarity, emotion class etc. Use them as attributes in the final model that uses other non text features

# FLOWCHART




Next page ..

# FLOWCHART (2)


Select features
Train test split and K fold cross validation
Choose performance metrics
Choose algorithm(s)
Compare performances : few are promising !
Put in pipeline
Classification
Regression
Clustering
Ensemble to improve performance
Algorithm tuning + grid search or random search to find best parameter
Save model
Load model to work on test data

# SUMMING UP

1. We have introduced steps in a typical machine learning problem when the data is text and/or numerical
2. We have provided a high level conceptual flow chart
3. For any typical machine learning assignment, this flow will be common (not only scikit learn and python).
4. For doing the similar assignment on R, tensorflow, Matlab, Java, IBM Watson or Microsoft Azure etc, relevant libraries with similar functionalities should be used.