Construction of a Recommendation System Guided by Machine Learning Intended to Accurately Match Students to Professors Based on the Research Interests of Both Groups*

Brenda Izquierdo
Florida International University
Miami, Florida

Alekhya Pulipaka
Florida International University
Miami, Florida

Rafay Khan
Florida International University
Miami, Florida

KEYWORDS

Text Classification, Multinomial Naive Bayes, Recommendation System

ACM Reference Format:
Brenda Izquierdo, Alekhya Pulipaka, and Rafay Khan. 2018. Construction of a Recommendation System Guided by Machine Learning Intended to Accurately Match Students to Professors Based on the Research Interests of Both Groups. In *Proceedings of ML Project (CAP 5610'18).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PROJECT DEFINITION

1.1 Project Overview

The aim of this project is to build a recommendation system that will provide a solid platform to correctly match students to their potential faculty mentors. In order to build our model we relied on research articles and conference papers published by the leading faculty from the Computing & Information Sciences Department at Florida International University (FIU). This decision was made in order to match the research interests of the student with those of faculty members. Supervised learning was utilized to train our model.

Recommendation systems have been widely adopted everywhere in our life. News websites suggest articles related to topics that we searched before. Streaming websites such as Netflix recommend new movies and TV series based on our viewing history. Online shopping sites like Amazon recommend articles related to previous purchases and what we are currently viewing. Recommendation systems are one of the most successful and widespread applications of machine learning technologies in business. However, the field remains quite open for additional recommendation systems that would serve different applications or goals.

The vast majority of available recommendation systems cannot produce the results we seek given our problem statement. There has been a similar recommendation system built by students at Stanford, and we hope to replicate their model. Our problem is also unique in that it carries a significant amount of real world impact and future implications.

The recommendation system we aim to develop will match a student with a potential professor who is the best fit for them by analyzing the students' interests and attempting to match them with those of the professor. This will be done via collecting research papers/articles of FIU CIS professors and extracting the abstract from these to use as text input for our classifier. The input of the student will also be of type text, and thus, this avoids the need to build a user profile. Both inputs will simultaneously be dealt with.

The text input will have to be vectorized, as 'text' type cannot be utilized as input to ML models. We aim to achieve this vectorization through constructing a transformer. This will be our tool for feature extraction. To train the model, we plan on implementing four Machine Learning algorithms, specifically; Multinomial Naive Bayes (MNB), Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT).

1.1.1 Collaborative Filtering. Collaborative filtering methods [11] are a usually common approach to building recommendation systems and are based on collecting and compiling a significant amount of the records of users behaviors, choices and preferences, and then trying to predict what the end users preference on specific items is based on the choice or preferences of other users that are similar to the end user, i.e. the user has made a similar choice or has similar preferences on other items.

The advantage here is that no prior knowledge is required of the item that the recommender is recommending. The downside is that a strong assumption is required in this scenario, that is, that a user will make a similar choice in the future solely based on their previous selection. This can prove too troublesome in many cases. For instance, in our scenario of matching the research interests of students to professors, when people further their study into one area, their focus and interests within that area are prone to adaptation and change.

A second issue with this model is commonly referred to as the 'cold start problem'. This means that in order for the recommender to provide useful output, it needs an enormous amount of other users' input data that must already be present and accessible. A news, video, or e-commerce website have the opportunity to gather user input for a long period of time before rolling out their recommender system. However, this fact does not translate over to gathering any user input for people searching for professors.

*Produces the permission block, and copyright information

CAP 5610'18, December 2018, Miami, FL USA

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Other problems of sparsity and scalability also apply to various extents. A professor will usually have a limited number of students and it will be difficult to make a correlation based on the small number of samples.

1.1.2 Content-Based Recommender. Content-Based Recommendation systems [13] are another example of a typical recommend-er. These systems rely on keywords to describe items. However, there is a big drawback in this approach, in that multiple items can be labeled identically while in fact there are subtleties present between them. For instance, various professors might be attributed to the same labels, i.e. supervised learning, deep-learning, etc., but they might be focusing on a different sub-domain, where there might be no appropriate model.

Also, these systems necessitate the creation of a user profile. And the system still needs two types of information, the model of users' preferences and users' interaction with the recommend-er system. The system creates a content-based profile of a user based on a weighted vector of the items' features extracted from the item.

The inherent properties of a content-based filtering system result in a significant issue with the model, that is, its ability to deal with cross domain content. Because of its nature, it is usually only suitable to the same type of content. This is not appropriate for the problem we are tackling.

1.1.3 Hybrid Recommender. This system operates as its name implies. It is a hybrid recommender that combines the properties of content-based and collaborative filtering systems [9]. However, due to the reasons described above of why those methods are not practical for our use, the combination of them produces no difference in their capabilities to make them suitable for our utilization. Hence, we chose to also forgo this approach.

1.1.4 Match Your Research Interest with Stanford AI Professor Recommendation System. This is a novel recommendation system that was built by two students, Weiqing Li and Haochong Shenat, at Stanford University in 2017 [12]. One of the main drawbacks of the methods described previously was the need to create user profiles. Li and Shenat proposed to use only one main building block as the recommender.

This new approach would allow both the user input and the item content to go through the same input block. Hence, the need to build a user profile is evaded and user input can directly be fed into the recommender.

The recommender system is made up of two functional blocks, the transformer and the classifier. The transformer transforms the input text into feature vectors that are operated on by the classifier. The classifier directly outputs the recommendation as a scoring vector and the top scored item will be used as the output.

1.2 Problem Statement

The vast majority of available recommendation systems cannot produce the results we seek given our problem statement. There has been a similar recommendation system built by students at Stanford, and we hope to replicate their model. Our problem is also unique in that it carries a significant amount of real world impact and future implications.

The recommendation system we aim to develop will match a student with a potential professor who is the best fit for them by analyzing the students' interests and attempting to match them with those of the professor. This will be done via collecting research papers/articles of FIU CIS professors and extracting the abstract from these to use as text input for our classifier. The input of the student will also be of type text, and thus, this avoids the need to build a user profile. Both inputs will simultaneously be dealt with.

The text input will have to be vectorized, as 'text' type cannot be utilized as input to ML models. We aim to achieve this vectorization through constructing a transformer. This will be our tool for feature extraction. To train the model, we plan on implementing four Machine Learning algorithms, specifically; Multinomial Naive Bayes (MNB), Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT).

1.3 Metrics

We will utilize the cross-validation metric to measure the performance of our model. This metric allows a drastic increase in the number of samples that can be used for learning the model and the output of the model can be dependent on a particular random choice for the pair of (train, validation) sets.

In the backend of the cross-validation or CV for short metric, the training set is split into k smaller sets and the following two rules or steps are abided by each of the k-folds: A model is trained using k-1 of the folds as training data and the resulting model is validated on the remaining part of the data. We then take the average of the values computed in the loop and present them as the performance measure of the k-fold CV. This is suitable for our problem because this approach keeps most of the data intact, instead of losing it, and offers a welcoming advantage where the number of samples are fairly small [1].

We calculated the CV metric using the cross_val_score helper function on the estimator and the dataset [5]. We then calculate the mean score and the 95% confidence interval of the score and report that as the accuracy measure of our model.

2 ANALYSIS

2.1 Data Exploration and Visualization

We obtained our dataset from the Scopus database. We selected the 22 faculty members who have the most published papers at Florida International University from the Computer Science & Information Technology Department and selected their research articles and conference papers ranging from the years of 2004 through 2018. The range of articles per professor was from 20 to 50, allowing us to have a dataset containing a total of 912 entries. The table to the right displays the names of the professors along with one of the

courses they have taught or are currently teaching.

Professor	Teaching	Course Name
Alex Afanasyev	TCN 6430	Networks Management and Ctrl Stndrds
Bogdan Carbunar	CIS 5372	Foundations of Computer Security
Christine Lisetti	CAP 5602	Introduction to Artificial Intelligence
Deng Pan	EEL 6787	Network Security
Fahad Saeed	ECE 2510	Introduction to Microprocessors
Geoffrey S. Smith	COT 5428	Formal Foundations for Cybersecurity
Giri Narasimhan	CAP 5510	Introduction to Bioinformatics
Jainendra K Navlakha	COP 4710	Database Management
Jason Liu	COP 4520	Introduction to Parallel Computing
Leonardo Bobadilla	CDA 4625	Introduction to Mobile Robotics
Mark Finlayson	CAP 5640	Natural Language Processing
Mark Weiss	COP 4338	Programming III
Naphtali Rishe	COP 5725	Principles of DBMS
Niki Pissinou	TCN 6450	Wireless Information Systems
Ning Xie	COT 6446	Randomized Algorithms
Peter J Clarke	CEN 5076	Software Testing
Raju Rangaswami	COP 6611	Advanced Operating Systems
Shu-Ching Chen	CIS 5027	Computer Systems Fundamentals
S. Masoud Sadjadi	CEN 5082	Grid Enablement of Scientific Apps
S. S. Iyengar	N/A	N/A
Xudong He	CEN 6075	Software Specification
Wei Zeng	COT 6405	Analysis of Algorithms

Below is a chart of the professors and their paper distribution

Afanasyev	38	**Navlakha**	20
Bobadilla	33	**Pan**	43
Carbunar	50	**Pissinou**	47
Chen	49	**Rangaswami**	40
Clarke	49	**Rishe**	48
Finlayson	19	**Sadjadi**	50
He	49	**Saeed**	41
Iyengar	49	**Smith**	48
Lisetti	48	**Weiss**	20
Liu	50	**Xie**	23
Narasimhan	49	**Zeng**	49

2.2 Algorithms and Techniques

2.2.1 Multinomial Naive Bayes. Multinomial Naive Bayes is the most popular classifier to perform text analysis. Derived from the Naive Bayes algorithm, it applies Bayesian inference to an unordered set of words. Bayesian inference is the application of Bayes' rule in order to determine the probability of a particular element belonging to a class.

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

The Scikit-learn Python library already provides a model with the Multinomial Naive Bayes Classifier method that we put in use. There are three ways to create a Naive Bayes classifier with Scikit Learn libraries. The first is the GaussianNB function which implements the Gaussian Naive Bayes algorithm. The Gaussian approach is best for continuous variables that can be fit to a Gaussian distribution. The second approach is BernoulliNB which address discrete data but makes classifying decisions based on binary or boolean features. The last approach is MultinomialNB which implements the Naive Bayes algorithm on discrete data and is often used for text classification. We chose this approach because of its ability to handle more than two classes in a dataset and because it is the baseline approach for text classification using Machine Learning. In order to execute it, we relied on the SKlearn package, from which we also used the function of CountVectorizer to split the abstract text in words while also implementing the library built-in function of stop words for the English language [2]. To create the test/train split we use K-Fold Cross-Validation from the SKlearn package, which will give us the X_train, X_test, y_train, and y_test datasets. The text counts or classification dictionary are generated by vectorizing X_train, then the classifier is fitted with this vectorized X_train and y_train datasets. To gauge the accuracy, the cross validation score is computed by running the classifier against vectorized X_test and comparing the result with y_test.

2.2.2 Support Vector Machine. Support Vector Machines are supervised learning models that analyze data and is used for classification and regression analysis or other tasks like outliers detection.Given a set of training examples, each marked as belonging to one or the other of two categories. An SVM model is a representation of the

examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

A support vector machine constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space.[6] A good separation is achieved by the hyper plane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Finding perfect class for large amount of training data set takes lot of time. Regularization parameter and gamma are tuning parameters in Support Vector Machine classifier. Varying those we can achieve considerable non linear classification line with more accuracy in reasonable amount of time. One more parameter is kernel. It defines whether we want a linear of linear separation. For linear kernel the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows: $f(x)=B(0)+\sum (ai*(x,xi))$[7].

The above equation involves calculating the inner products of a new input vector (x) with all support vectors in training data.

The coefficients B0 and ai for each input must be estimated from the training data by the learning algorithm.

The polynomial kernel can be written as, $K(x,xi) = 1 + \sum(x * xi)\hat{d}$ and
exponential as, $K(x,xi) = \exp(-gamma * \sum((x-xi\hat{2}))$.
The Regularization parameter is how much we want to avoid misclassification of each training example. For large values of regularization parameter, the optimization will choose a smaller-margin hyper plane if that hyper plane does a better job of getting all the training points classified correctly. Conversely, a very small value of regularization parameter will cause the optimizer to look for a larger-margin separating hyper plane, even if that hyper plane misclassifies more points.

The gamma parameter defines how far the influence of a single training example reaches. With low gamma, points far away from plausible seperation line are considered in calculation for the seperation line. Where as high gamma means the points close to plausible line are considered in calculation.

2.2.3 Random Forest classifier. Random Forest Classifier is an ensemble algorithm [10]. Ensembled algorithms are those which combines more than one algorithms of same or different kind for classifying objects. For example, running prediction over Naive Bayes, SVM and Decision Tree and then taking vote for final consideration of class for test object. Alternatively, the random forest can apply weight concept for considering the impact of result from any decision tree. Tree with high error rate are given low weight value and vise versa. This would increase the decision impact of trees with low error rate.

Basic parameters to Random Forest Classifier can be total number of trees to be generated and decision tree related parameters like split criteria, features, estimators etc.

2.2.4 Decision Tree Classifier. The classification technique is a systematic approach to build classification models from an input data set. Each technique adopts a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data. Therefore, a key objective of this learning algorithm is to build predictive model that accurately predict the class labels of previously unknown records[3].

Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time it receive an answer, a follow-up question is asked until a conclusion about the class label of the record is reached.

Construction of Decision Tree : A tree can be learned by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.[3]

The decision tree induction algorithm works by recursively selecting the best attribute to split the data and expanding the leaf nodes of the tree until the stopping criterion is met. The choice of best split test condition is determined by comparing the impurity of child nodes and also depends on which impurity measurement is used. After building the decision tree, a tree-pruning step can be performed to reduce the size of decision tree. Decision trees that are too large are susceptible to a phenomenon known as over fitting. Pruning helps by trimming the branches of the initial tree in a way that improves the generalization capability of the decision tree.

A tree is build by choosing the best dividing criteria for the given data set. It is called with list of rows and then loops through every column (except the last one, which has the result in it), finds every possible value for that column, and divides the dataset into two new subsets. It calculates the weighted average entropy for every pair of new subsets by multiplying each setâĂŹs entropy by the fraction of the items that ended up in each set, and remembers which pair has the lowest entropy. If the best pair of subsets doesnâĂŹt have a low weighted-average entropy than the current set, that branch ends and the counts of the possible outcomes are stored. Otherwise, build tree is called on each set and they are added to the tree. The results of the calls on each subset are attached to the True and False branches of the nodes, eventually constructing an entire tree.

Pruning involves checking pairs of nodes that have a common parent to see if merging them would increase the entropy by less than a specified threshold. If so, the leaves are merged into a single node with all the possible outcomes. This helps avoid over fitting and stops the tree from making predictions that are more confident than what can really be gleaned from the data. When prune function is called on the root node, it will traverse all the way down the tree to the nodes that only have leaf nodes as children. It will create a combined list of results from both of the leaves and will test the entropy. If the change in entropy is less than the minimum gain parameter, the leaves will be deleted and all their results moved to their parent node. The combined node then becomes a possible candidate for deletion and merging with another node [3].

2.3 Benchmark

We will be using a set of small statements as threshold for comparing the performances of our models. The input will be entered by the user and the algorithms are expected to match the keywords with the respective professors. These are our ten entries and their expected results

Input: "I am very interested in information security, especially how fraudulent android applications operate, email phishing, and also I would like to learn more about bitcoins"
Expected Output: Bogdan Carbunar
Input: "I would like to do research in areas of natural language processing, deep learning, image recognition and convolutional neural networks"
Expected Output: Shu-Ching Chen
Input: "Research about multimodal spoken dialogue system"
Expected Output: Naphtali Rishe
Input: "I am very interested in software engineering and the application of software validation and verification techniques"
Expected Output: Peter Clarke
Input: "Work on optimization for geometric computational algorithms"
Expected Output: Wei Zeng
Input: "I enjoy mathematics (especially algebra) and computing; I would like to do research about probabilistic process algebra and Markov decision process"
Expected Output: Xudong He
Input: "Mobile focused learning and mobile computing techniques"
Expected Output: Niki Pissinou
Input: "I am interested in Natural Language Processing and would like to learn more about it during my PhD, as well computational linguistics, and sentiment analysis on text classification"
Expected Output: Mark Finlayson
Input: "I would like to know more about affective computing, and do research in areas of emotional agents using supervised learning and human-computer interaction theories"
Expected Output: Christine Lisetti
Input: "Research on Gene Onotology, SS score and semantic similarity of genes"
Expected Output: Giri Narasimhan

3 METHODOLOGY

3.1 Data Preprocessing

To preprocess our input, we extracted the Author Name, Author Names, and Abstract from all of the papers as three columns into a comma separated values (csv) file. We then removed any potential troublesome symbols, such as the copyright symbol, from the abstract. The Author Name here refers to the professor affiliated with FIU, who is part of our research in this project, while Author Names includes the FIU professor as well as all other co-authors. The figure below displays a sample of our input data

	Label	Authors	Abstract
0	Afanasyev	Zhang H., Li Y., Zhang Z., Afanasyev A., Zhang L.	As a proposed Internet architecture, Named Dat...
1	Afanasyev	Gibson C., Bermell-Garcia P., Chan K., Ko B., ...	The fundamental aim of this paper is to positi...
2	Afanasyev	Flittner M., Mahfoudi M.N., Saucez D., Hlisch ...	Reproducibility of artifacts is a cornerstone ...
3	Afanasyev	Zhang Y., Wang L., Afanasyev A., Zhang L.	This papersummarizes our comparative study on ...
4	Afanasyev	Chan K., Ko B., Mastorakis S., Afanasyev A., Z...	In the current Named Data Networking implement...
5	Afanasyev	Shang W., Afanasyev A., Zhang L.	Distributed dataset synchronization (sync for ...
6	Afanasyev	Zhang Z., Afanasyev A., Zhang L.	The Named Data Networking (NDN) architecture b...
7	Afanasyev	Zhang Z., Yu Y., Afanasyev A., Burke J., Zhang L.	As a proposed Internet architecture, Named Dat...

We chose to select the abstract as a representation of the paper since it is a concise detailed summary that is very likely to contain the information we are seeking, the keywords associated with a particular professor that would point to their particular field of interest and/or research. Also, this allowed us to keep our data size as concise as possible without compromising the quality.

3.2 Implementation and Refinement

Our input, X, was represented by the abstract of the paper, and our label, y, was represented by the professor's name. As text input cannot be directly fed into ML models, we performed some necessary transformations on our data. We mapped each professor's last name to a numerical label, ranging from 0 to 22, representing the 21 professors.

To deal with the input, X, we constructed a vectorizer to transform the abstract text [4]. There was an argument selected for stop_words to insure removal of uninformative information that might be mistaken as signal for prediction [2]. The transformer was then fit on the training and test input. This was used as the input to the classifier.

4 RESULTS

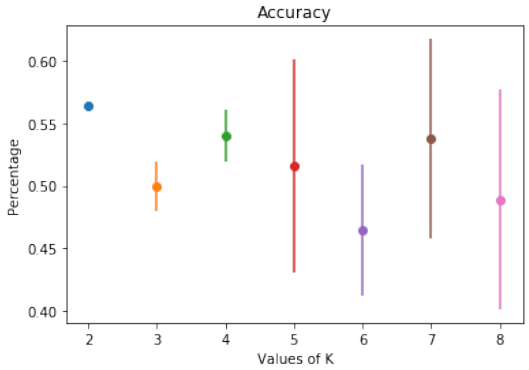
In this section we will go over the results we obtained after we ran our four algorithms with the dataset.

4.1 Model Evaluation and Validation

To build our recommender model, we decided to first start by implementing four algorithms. For each of these algorithms, the train/test split was accomplished using K-Fold Cross-Validation, a resampling procedure used to evaluate machine learning models by separating the dataset into a number of k groups. For example, if K = 5, the total data is divided into 5 folds of equally represented class labels. One of those folds is set apart as the testing set and the other 4 are used as the training set [8]. After building our four classifiers, these were the results obtained:

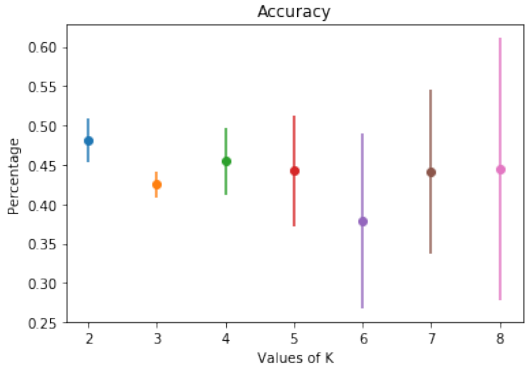
Multinomial Naive Bayes Classifier

Parameters:
alpha = 1.0 (Laplace smoothing)
fit_prior = False (Uniform prior is used, the accuracy was marginally lower when set to True)
class_prior = None (Priors are adjusted according to the data)
Results:
K = 2, Accuracy: 0.57 (+/- 0.01)
K = 3, Accuracy: 0.51 (+/- 0.04)
K = 4, Accuracy: 0.54 (+/- 0.04)
K = 5, Accuracy: 0.54 (+/- 0.18)
K = 6, Accuracy: 0.48 (+/- 0.10)
K = 7, Accuracy: 0.53 (+/- 0.18)
K = 8, Accuracy: 0.50 (+/- 0.19)



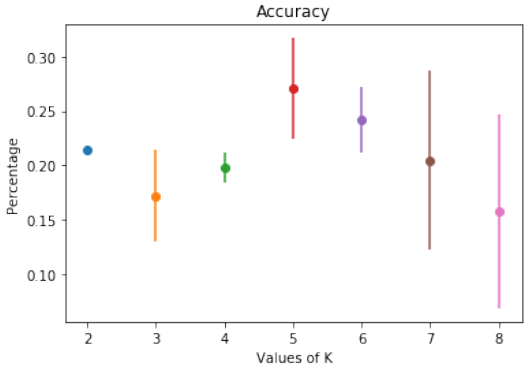
Support Vector Machine Classifier

Parameters:
kernel = 'linear' (Linear kernel is often recommended for text classification with SVM. Text data is linearly separable)
random_state = 0 (Pseudo random number generator used when shuffling the data for probability estimates)
Note: Other parameters not specified are set to default
Results:
K = 2, Accuracy: 0.48 (+/- 0.06)
K = 3, Accuracy: 0.43 (+/- 0.03)
K = 4, Accuracy: 0.45 (+/- 0.09)
K = 5, Accuracy: 0.44 (+/- 0.14)
K = 6, Accuracy: 0.38 (+/- 0.22)
K = 7, Accuracy: 0.44 (+/- 0.21)
K = 8, Accuracy: 0.44 (+/- 0.33)



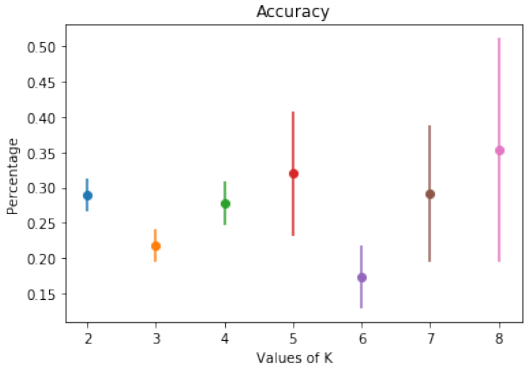
Decision Tree Classifier

Parameters:
class_weight = None (In this problem weights are not associated with classes)
criterion = 'entropy' (Function to measure the quality of a split. Entropy was chosen for information gain)
random_state = 0 (Seed used by the random number generator)
Note: Other parameters not specified are set to default
Results:
K = 2, Accuracy: 0.21 (+/- 0.01)
K = 3, Accuracy: 0.17 (+/- 0.09)
K = 4, Accuracy: 0.20 (+/- 0.03)
K = 5, Accuracy: 0.27 (+/- 0.09)
K = 6, Accuracy: 0.24 (+/- 0.06)
K = 7, Accuracy: 0.20 (+/- 0.17)
K = 8, Accuracy: 0.16 (+/- 0.18)

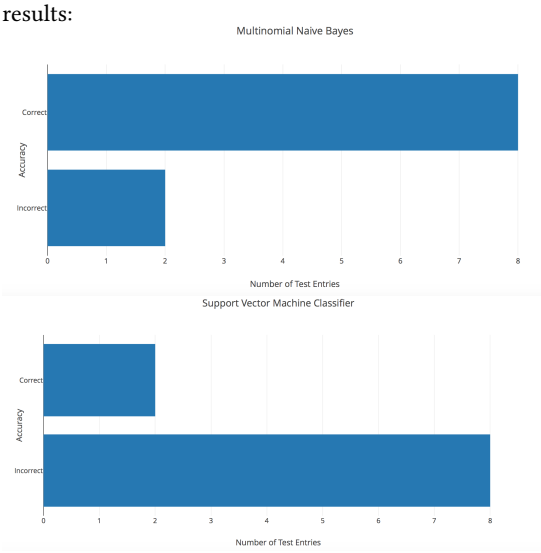


Random Forests Classifier

Parameters:
n_estimators = 15 (The number of trees in the forest.)
criterion = 'entropy' (Function to measure the quality of a split. Entropy was chosen for information gain)
random_state = 0 (Seed used by the random number generator)
Note: Other parameters not specified are set to default
Results:
K = 2, Accuracy: 0.29 (+/- 0.05)
K = 3, Accuracy: 0.22 (+/- 0.05)
K = 4, Accuracy: 0.28 (+/- 0.06)
K = 5, Accuracy: 0.32 (+/- 0.18)
K = 6, Accuracy: 0.17 (+/- 0.09)
K = 7, Accuracy: 0.29 (+/- 0.19)
K = 8, Accuracy: 0.35 (+/- 0.32)



Since Multinomial Naive Bayes and Support Vector Machine were the classifiers with higher accuracy, we chose them as our two models. To validate them, we designed an interface that receives a small statement of purpose, paragraph of description of their interest, or even a small abstract and the model will match it with one of the professors based on their research and recommend this faculty member to the user. In order to achieve this, we prepared 10 short statements (See Benchmark subsection under Analysis) and matched them with a professor based on their research area in order to validate the accuracy of the classifier. This were the results:



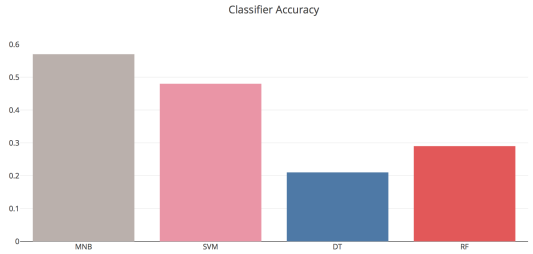
4.2 Justification

Multinomial Naive Bayes gave the most accurate classification out of all the algorithms tried. Ironically the interface testing accuracy was higher (0.8) than the model's at its best split (0.57) at K = 2. Linear SVM is also a very good method for text analysis and sometimes outperforms the MNB classifier, especially when trained with large documents. In this experiment we train the classifier only with the abstracts of the papers, and our recommendation input in similarly small. This proves that MNB is better on snippets than its SVM counterpart [14].

5 CONCLUSIONS

5.1 Visualization

To conclude this project, we used 4 algorithms to find the best classifier for our recommendation system. This is a comparison of their accuracy at K = 2:



While both the MNB and SVM algorithms had an accuracy of around 50%, the MNB was far superior when it came to providing an accurate recommendation for the user. We implemented the 10 testing entries specified in the Benchmark section and obtained a 80% of prediction accuracy for the Multinomial Naive Bayes model, thus corroborating what we found in our research about its efficacy for text classification.

5.2 Reflections

We observed better accuracy for the model by using train_test_split method from SKlearn to randomly split data samples (without taking into account the class balance) than the K-Fold method. Also, when running the MNB against a balanced dataset with entries for only ten professors the accuracy was much higher, 0.64 (+/- 0.18) for K = 3. The SVM algorithm was also at its best at K = 3 with 0.57 (+/- 0.12).

5.3 Improvements

Increasing the number of training samples, data prepossessing to the point and by using more balanced dataset would have resulted in a more accurate model.

Also applying techniques like feature weight computation and tree selection method to random forest classifier would have resulted in more accurate random forest classifier. Similarly, adding more information about paper instead of just the Abstract could have resulted in a better accuracy for the SVM classifier.

REFERENCES

[1] [n. d.]. 3.1. Cross-validation: evaluating estimator performance. https://scikit-learn.org/stable/modules/cross_validation.html
[2] [n. d.]. 4.2. Feature extraction. https://scikit-learn.org/stable/modules/feature_extraction.html#stop-words
[3] [n. d.]. Decision Tree Classifier. http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/decisionTree.html
[4] [n. d.]. sklearn.feature_extraction.text.CountVectorizer. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[5] [n. d.]. sklearn.model_selection.cross_val_score. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html
[6] [n. d.]. Support Vector Machine. https://en.wikipedia.org/wiki/Support_vector_machine
[7] [n. d.]. Understanding Support Vector Machine. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
[8] [n. d.]. Vectorization, Multinomial Naive Bayes Classifier and Evaluation. https://www.ritchieng.com/machine-learning-multinomial-naive-bayes-vectorization/
[9] 2012. Hybrid Recommender Systems. http://recommender-systems.org/hybrid-recommender-systems/
[10] Yunming Ye Jiefeng Cheng Baoxun Xu, Xiufeng Guo. 2012. An Improved Random Forest Classifier for Text Categorization.
[11] Prince Grover. 2017. Various Implementations of Collaborative Filtering Towards Data Science. https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

[12] Li and Shen. 2017. A New Approach to Recommender Systems.
[13] Carlos Pinela. 2017. Content-Based Recommender Systems âĂŞ Carlos Pinela âĂŞ Medium. https://medium.com/@cfpinela/content-based-recommender-systems-a68c2aee2235
[14] Sida Wang and Christopher D. Manning. 2012. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2 (ACL '12)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 90–94. http://dl.acm.org/citation.cfm?id=2390665.2390688