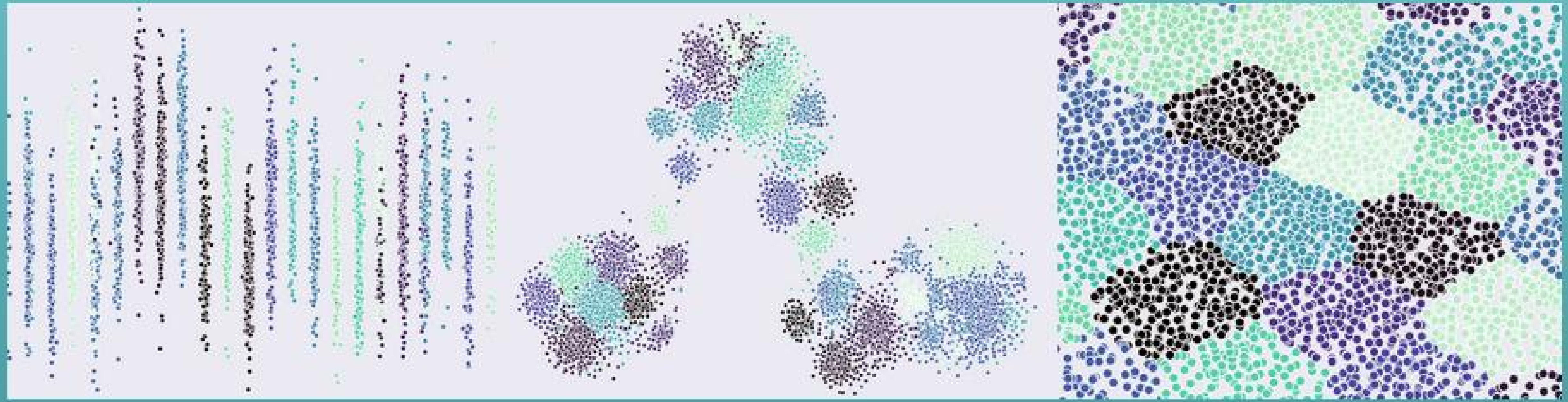
Math Lectures

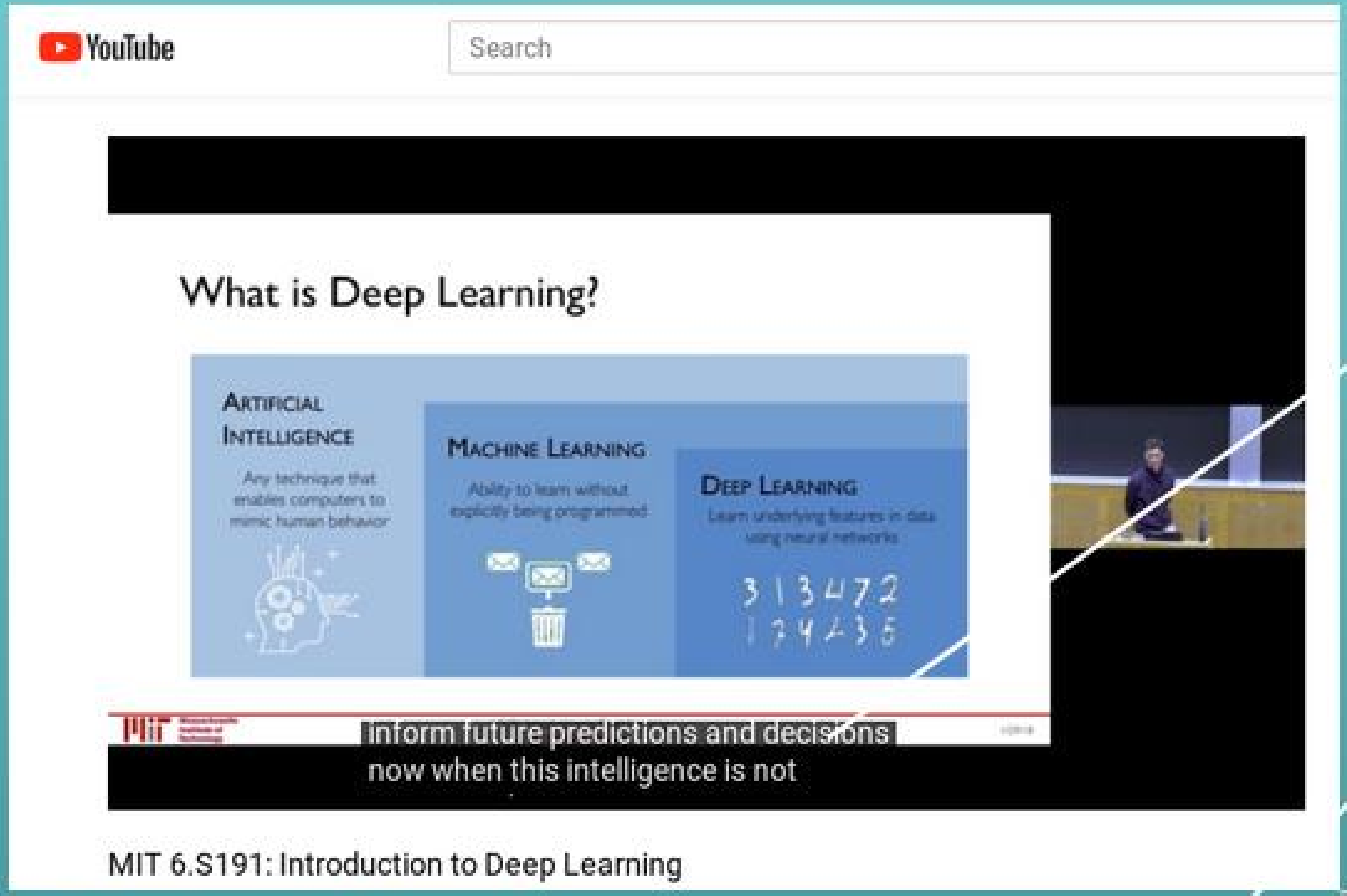


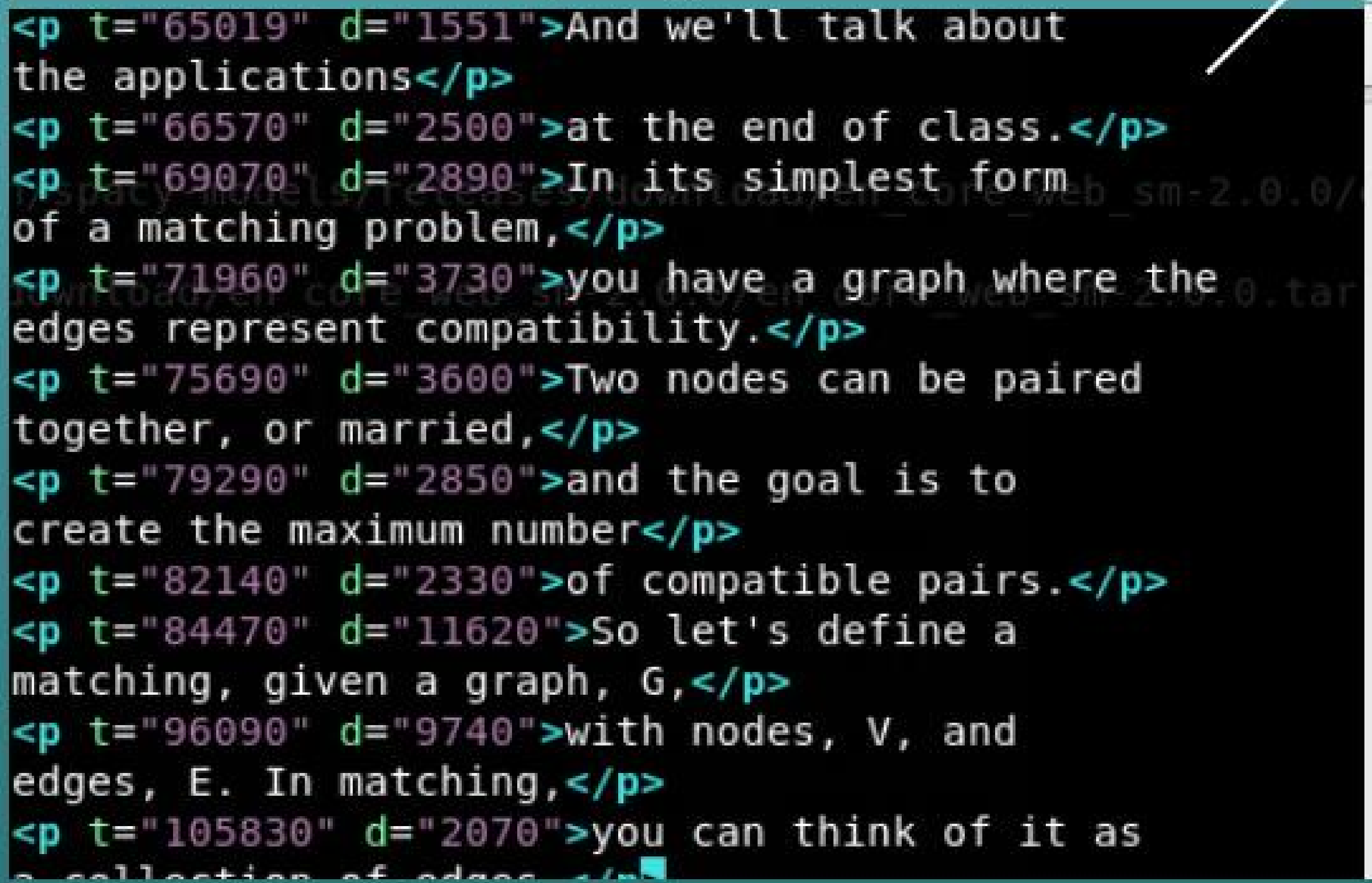
Objective:

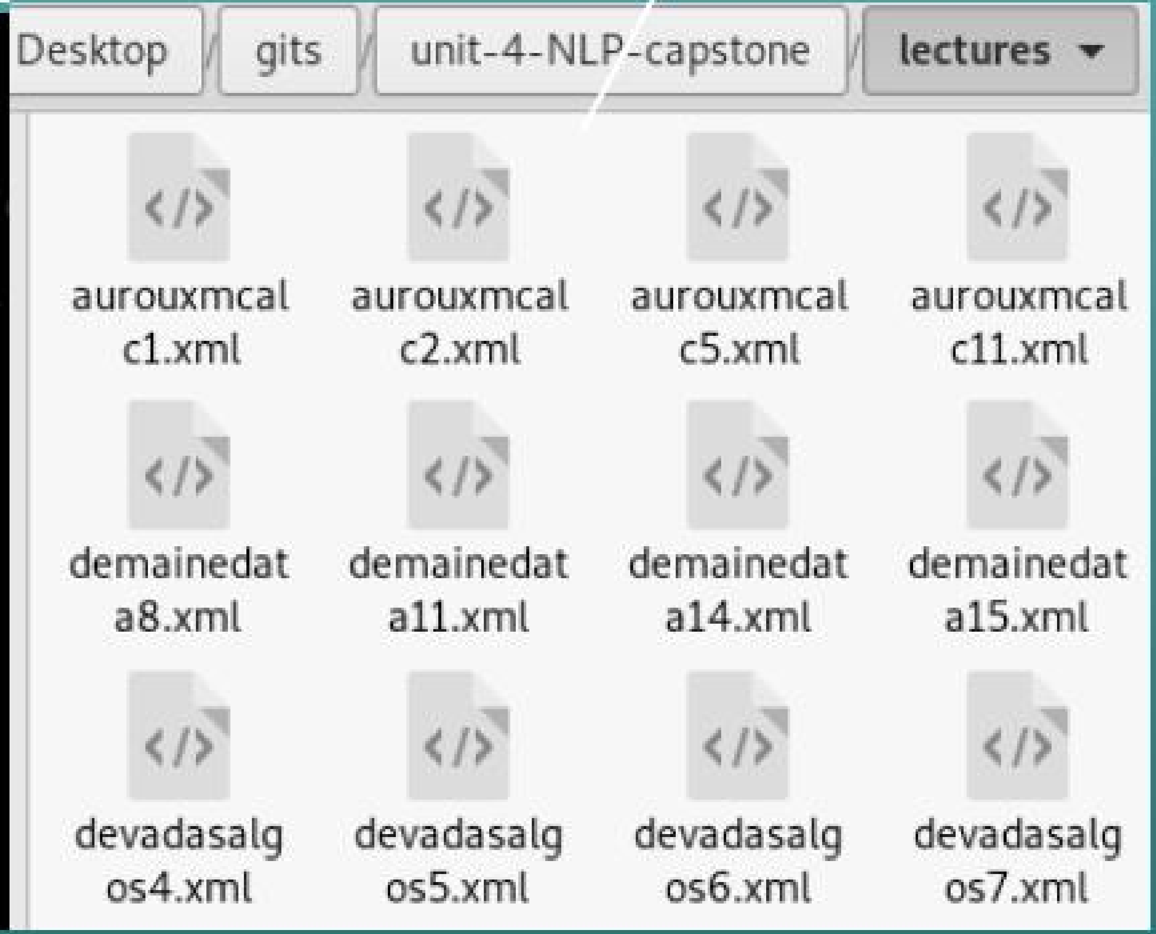
Combine NLP with supervised and unsupervised learning to classify math lectures

Use closed captioning from 92 lectures

Save XML file in a directory







Objective: Use supervised and unsupervised learning techniques to best classify math lectures.

(Clean the data)

Part 1
Clustering and Similarity

- Train Doc2Vec Model
- Dimensionality Reduction for Visualization
- Clustering the data
- KMeans Silhouette Scores
- 9, 10, and 11 clusters
- Topic Extraction (NMF and LDA)

Part 2
Modeling

- Tf-idf vectorization
- Initial model
- Parts of Speech
- Parameter Search
- Final Model

Clean and tokenize the text

	lecture id	str	label	label	Spacy Doc
	filename	**raw_text**	**Professor**	**Subject**	**sdoc**
0	aurouxmcalc1	So let us start right away with stuff that we ...	Auroux	Calculus	(So, let, us, start, right, away, with, stuff,...
1	aurouxmcalc11	to So far we have learned about partial...	Auroux	Calculus	(, to, , So, far, we, have, learned, ab...
2	aurouxmcalc2	So , So, yesterday we learned about the questi...	Auroux	Calculus	(So, ,, So, ,, yesterday, we, learned, about, ...

- extract lemmas, remove punctuation and stop words

```python
#create a new data frame for the professor,subject and the spacy doc
sentences = raw_data[['filename','Professor','Subject','sdoc']].copy()

#create a list of lists of tokens (remove stop words and punct)
sentences['sents'] = [ [ [token.lemma_.lower() for token in sent if not token.is_stop
        and not token.is_punct] for sent in doc.sents] for doc in sentences.sdoc]

#convert lecture lists of sentences to lecture string
sentences['text'] = [' '.join([str( ' '.join(i)) for i in j]) for j in sentences.sents]
```

	filename	Professor	Subject	sdoc	sents	text
0	aurouxmcalc1	Auroux	Calculus	(So, let, us, start, right, away, with, stuff,...	[[so, let, start, right, away, stuff, need, ad...	so let start right away stuff need advanced th...
1	aurouxmcalc11	Auroux	Calculus	(, to, , So, far, we, have, learned, ab...	[[,], [so, far, learn, partial, deriva...	so far learn partial derivative use f...
2	aurouxmcalc2	Auroux	Calculus	(So, ,, So, ,, yesterday, we, learned, about, ...	[[so, so, yesterday, learn, question, plane, t...	so so yesterday learn question plane think 3x3...

Use Doc2Vec to vectorize each lecture

- Converts each lecture into a 65 dimensional vector

```
X = np.array(sentences['text'])
y = np.array(sentences[['filename','Professor','Subject']]) #keep both labels

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.25, random_state=43)
X_train.shape
```

```
(69,)
```

```
#tag the data
tagged_data = [TaggedDocument(words=word_tokenize(_d.lower()), tags=[str(i)]) for i, _d in enumerate(X_train)]
```

Lecture $_i$

[x_i1, x_i2 … x_i64, x_i65]

```
#Train the model

#max training epochs
max_epochs = 100

model = Doc2Vec(vector_size=65, # lower dimensions than observations
                alpha=.025, #initial learning rate
                min_alpha=0.00025, #learning rate drops linearly to this
                min_count=7, #ignores all words with total frequency lower than this.
                dm =1) #algorith 1=distributed memory

#Build vocabulary from a sequence of sentences (can be a once-only generator stream).
model.build_vocab(tagged_data)

#train 100 epochs and save the model
t1 = time.time()
for epoch in range(max_epochs):
    print('iteration {0}'.format(epoch))
    model.train(tagged_data,
                total_examples=model.corpus_count,
                epochs=model.iter)
    # decrease the learning rate
    model.alpha -= 0.0002
    # fix the learning rate, no decay
    model.min_alpha = model.alpha
t2 = time.time()
model.save("full_lects.model")
print("Model Saved")
print("Time: {}".format(t2-t1))
```

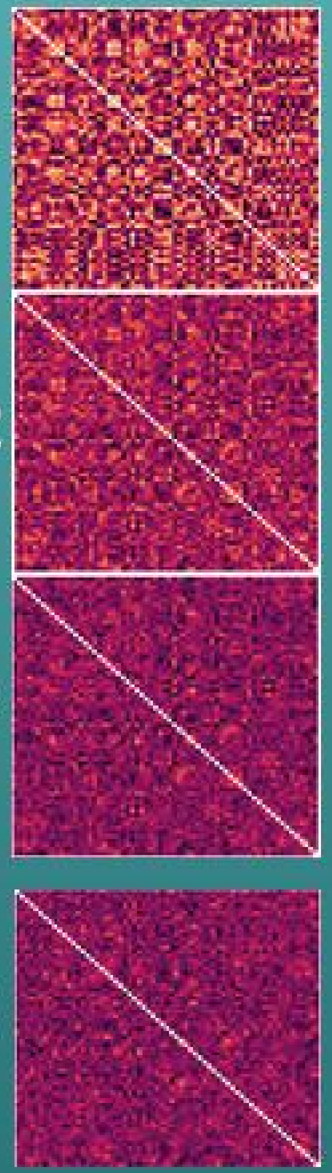
Epoch 1

Epoch 2

Epoch 3

…

Epoch 100



Extract the numerical representation for each lecture

```
len(tagged_data)
```

***tagged_data[0]=1 full lecture**

```
69
```

```
#view some of the text and its vector representation
print(tagged_data[0][0][:25])
```

```
['all', 'right', 'hello', 'everybody', 'welcome', 'lecture', 'seven', 'maybe', 'definitely', '
today', 'beginning', 'talk', 'model', 'matter', 'practice', 'will', 'talk', 'today', 'simple',
'recurrent', 'neural', 'network', 'model', 'think', 'but']
```

```
#extract the vectors from the model
vecs = pd.DataFrame([model.docvecs[str(i)] for i in range(len(tagged_data))])
```

```
vecs.head()
```

	0	1	2	3	4	5	6	7	8	9	...	55
0	2.662631	1.020528	6.387894	-5.270965	-3.704841	-0.420956	2.778944	5.019346	-4.486396	-2.107513	...	3.396559
1	-2.337144	-0.877562	-2.179196	3.026798	2.296631	2.526698	-3.398579	-2.706168	-0.723500	-3.683723	...	0.562401
2	-1.693762	1.025025	-6.552120	0.428101	-0.465139	5.433910	0.244512	-3.862596	0.788520	-8.810690	...	-1.666339
3	2.106681	2.410404	1.423349	-0.458352	-4.432656	-1.351960	-4.553487	-4.567388	0.558583	0.871899	...	-1.446398
4	2.490935	-7.870018	3.407579	5.935986	1.284999	4.579460	-3.233096	-3.997610	-1.596189	-1.476011	...	2.666141

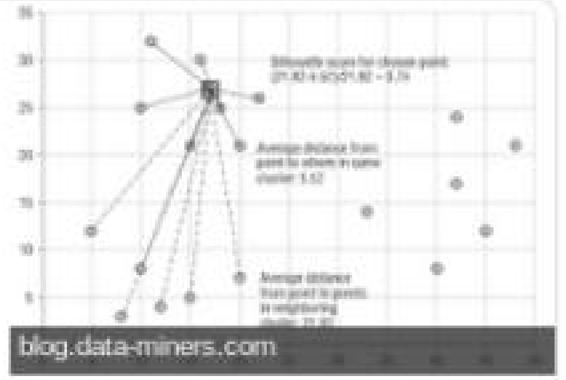
Calculate cosine similarity of lectures

```
lecture = y_train[:,0][0]
d2v_fullsim[[lecture, 'Original_Sentence', 'Professor','Subject',
            'filenames','mean_similarity']].sort_values(by=[lecture],ascending=False)[:5]
```

	manningnlp8	Original_Sentence	Professor	Subject	filenames	mean_similarity
manningnlp8	1.000000	all right hello everybody welcome lecture se...	Manning	NLP	manningnlp8	0.080441
sochernlp5	0.680272	> > about propagation algorithm and promise...	Socher	NLP	sochernlp5	0.073998
sochernlp3	0.554571	alright hello everybody welcome lecture ric...	Socher	NLP	sochernlp3	0.086428
manningnlp10	0.342864	okay cs224n. so let so term go to today me...	Manning	NLP	manningnlp10	0.105305
manningnlp2	0.304359	okay let go welcome second class cs224n /l 2...	Manning	NLP	manningnlp2	0.109334

Metrics for clustering and determining similarity

The **silhouette** value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The **silhouette** ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

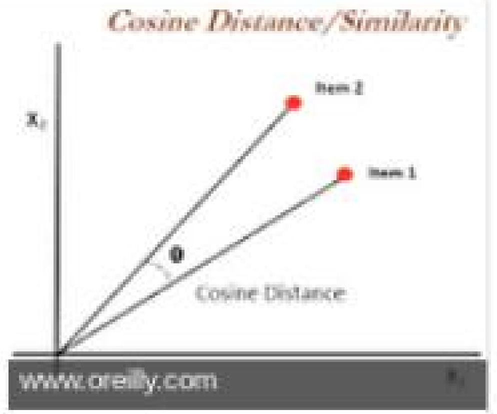


Silhouette (clustering) - Wikipedia

https://en.wikipedia.org/wiki/Silhouette_(clustering)

About this result Feedback

Cosine similarity is a measure of **similarity** between two non-zero vectors of an inner product space that measures the **cosine** of the angle between them. The **cosine** of 0° is 1, and it is less than 1 for any angle in the interval (0,π] radians.



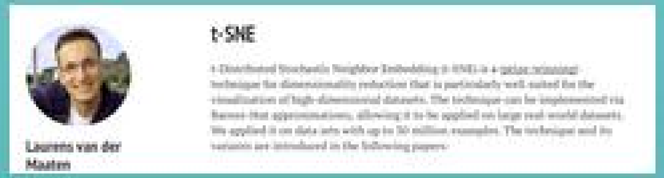
Cosine similarity - Wikipedia

https://en.wikipedia.org/wiki/Cosine_similarity

About this result Feedback

Reducing the Dimensionality

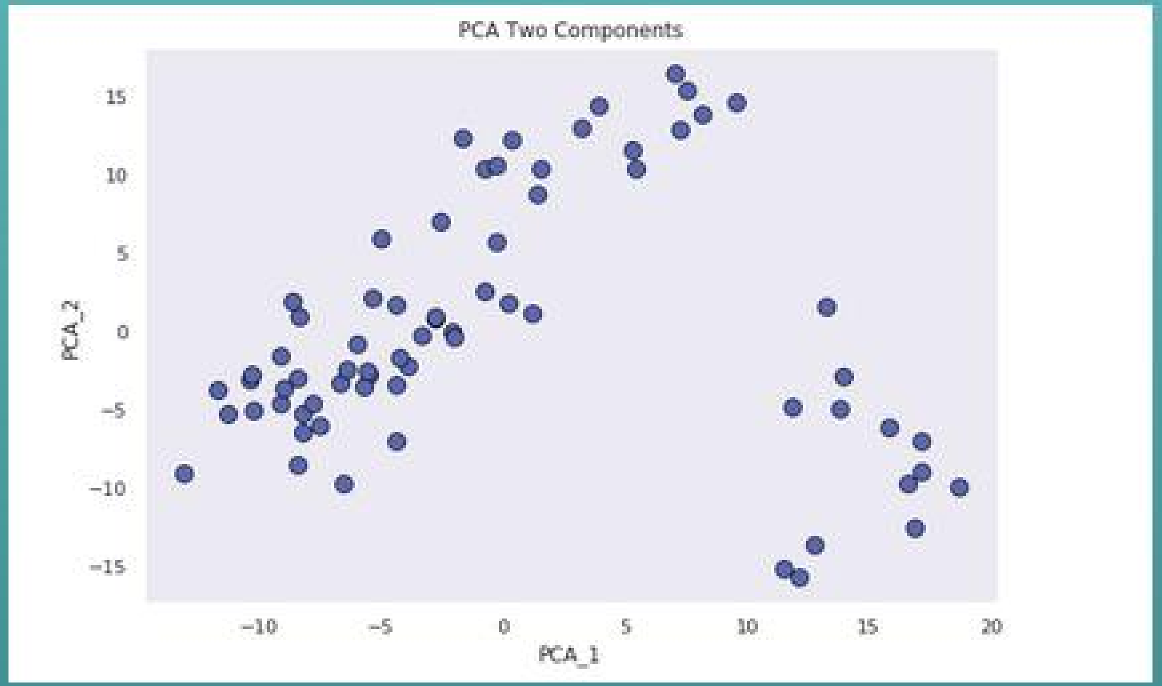
Click to Learn About t-SNE



for Visualization

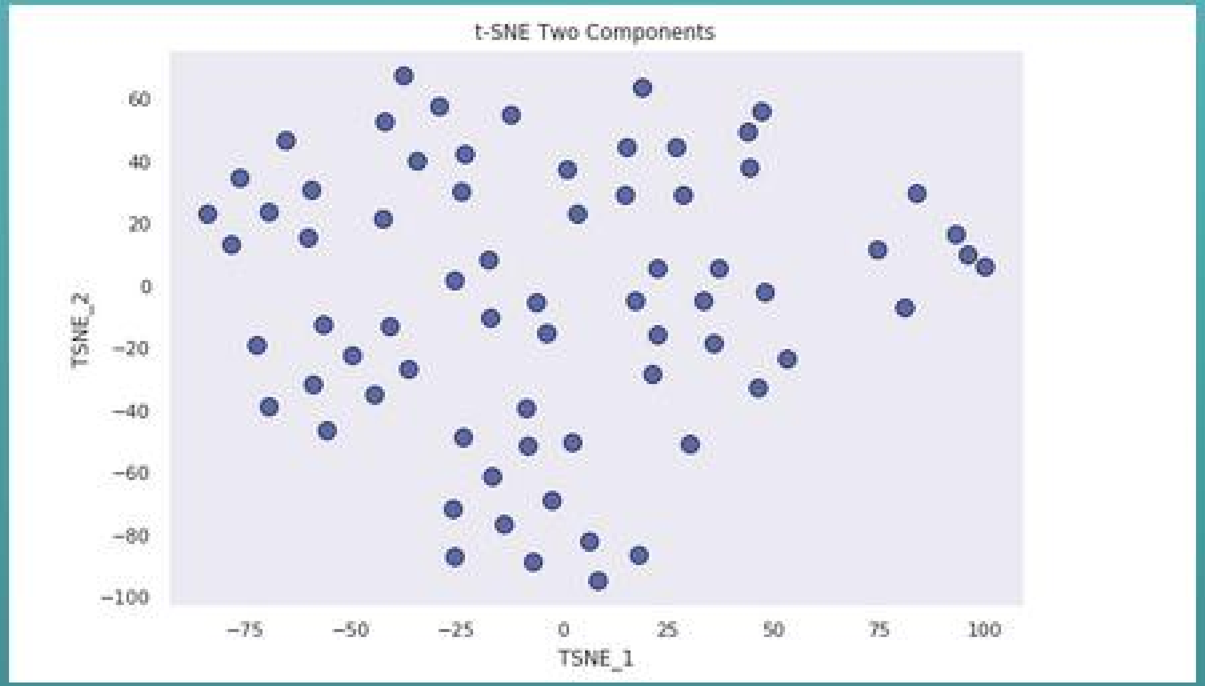
Using PCA

Y=pca.fit_transform(vecs)



Using t-SNE

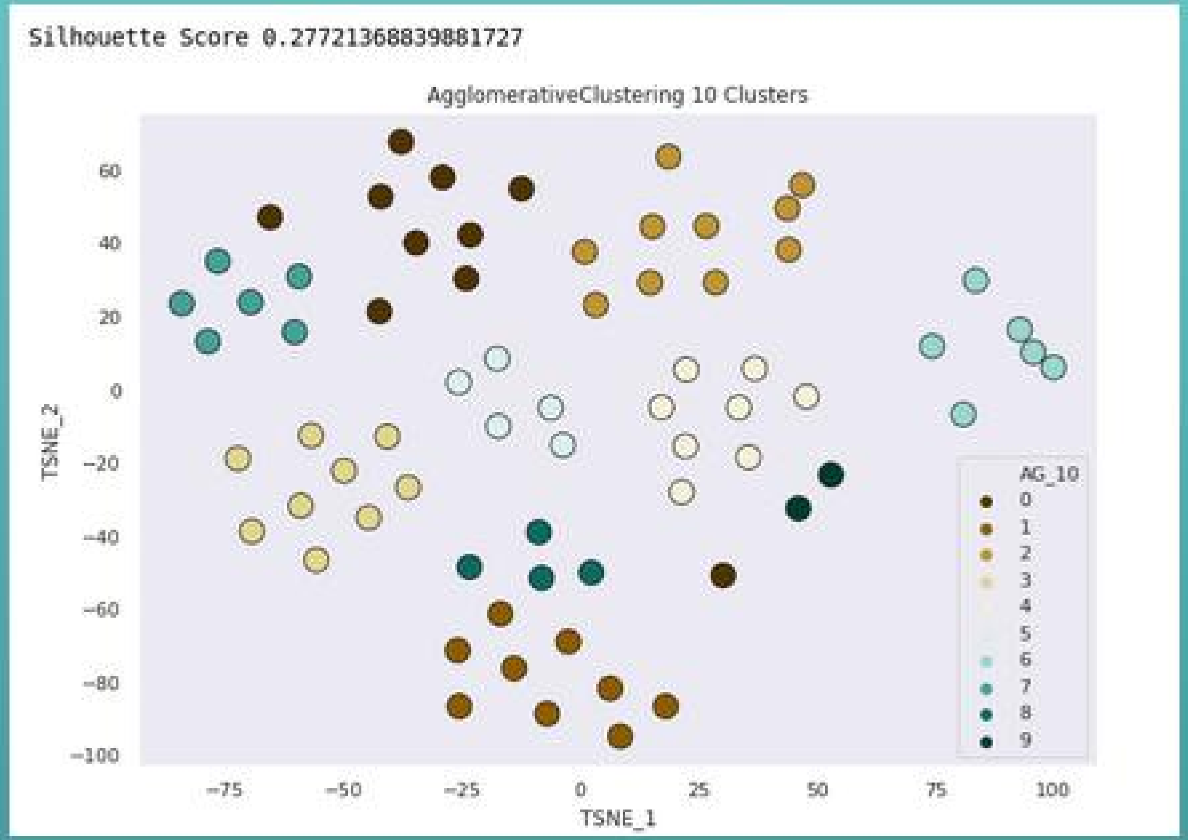
Yt=tsne.fit_transform(vecs)



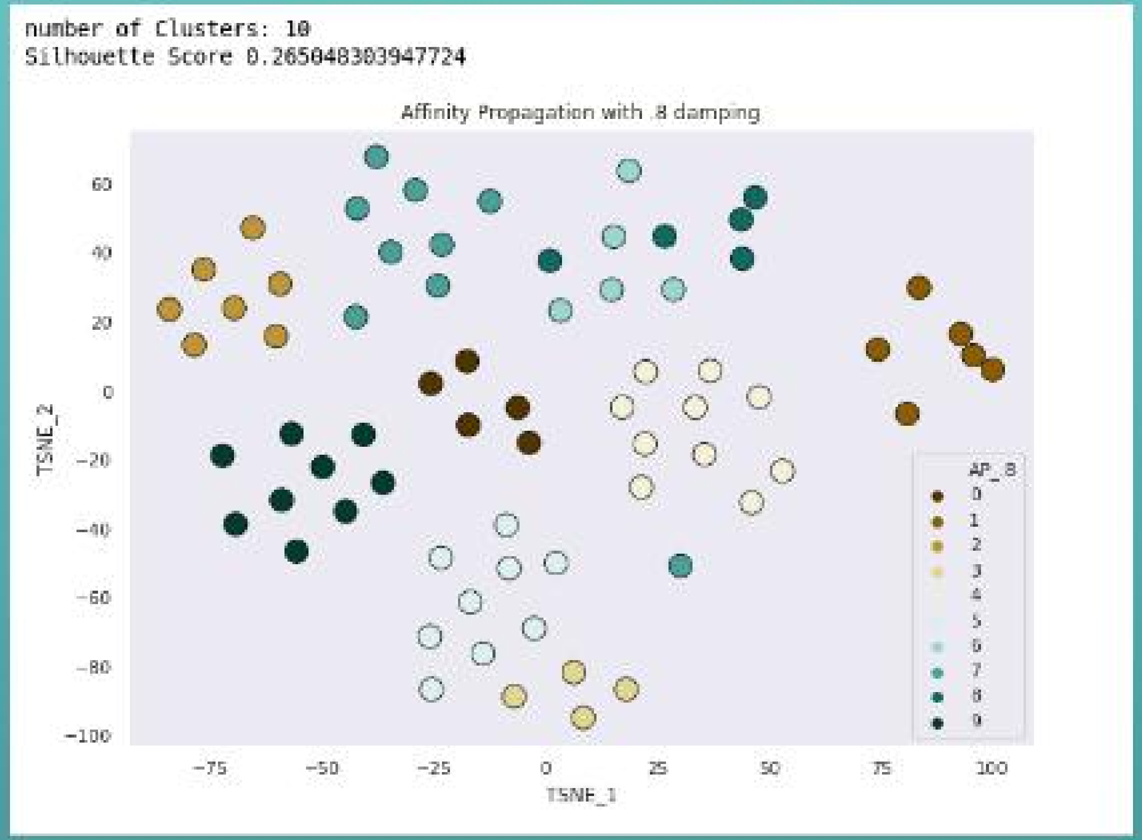
The scatter plot of the t-SNE components is much easier to interpret. For this reason t-SNE was chosen as the prefered method for reducing the dimensionality

Clustering the Data

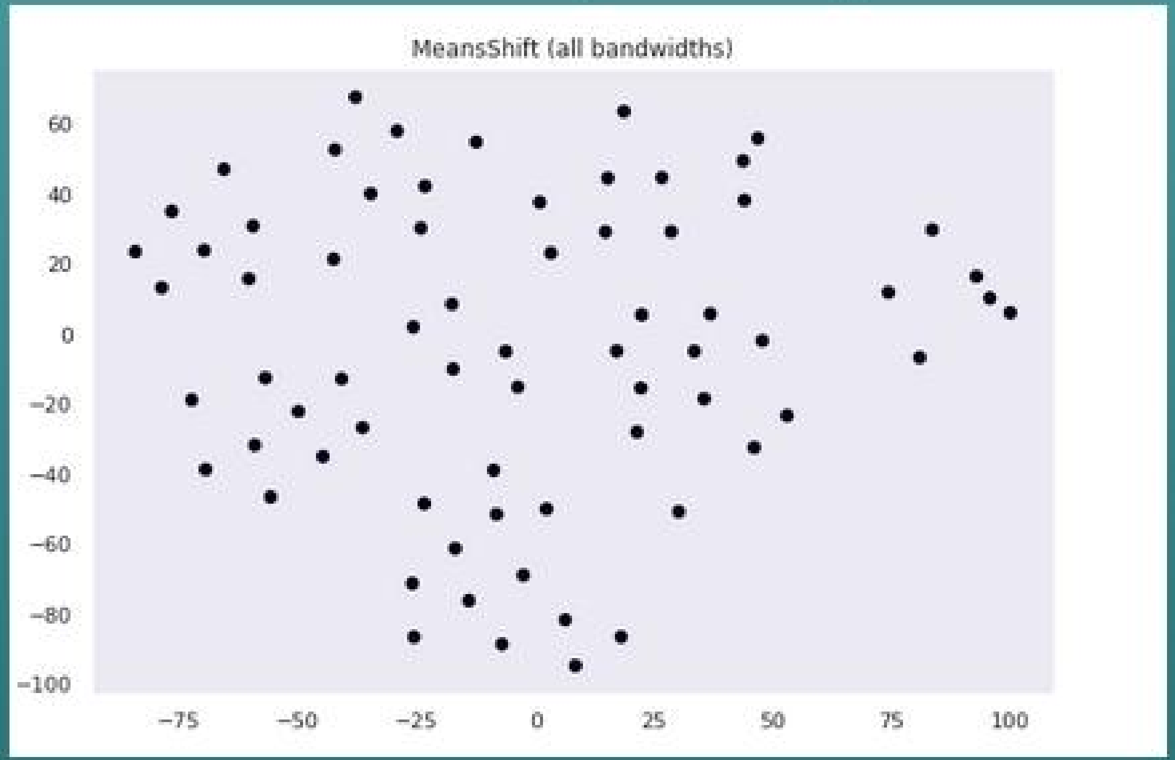
Agglomerative Clustering - 10 Clusters



Spectral Clustering (damping .8)

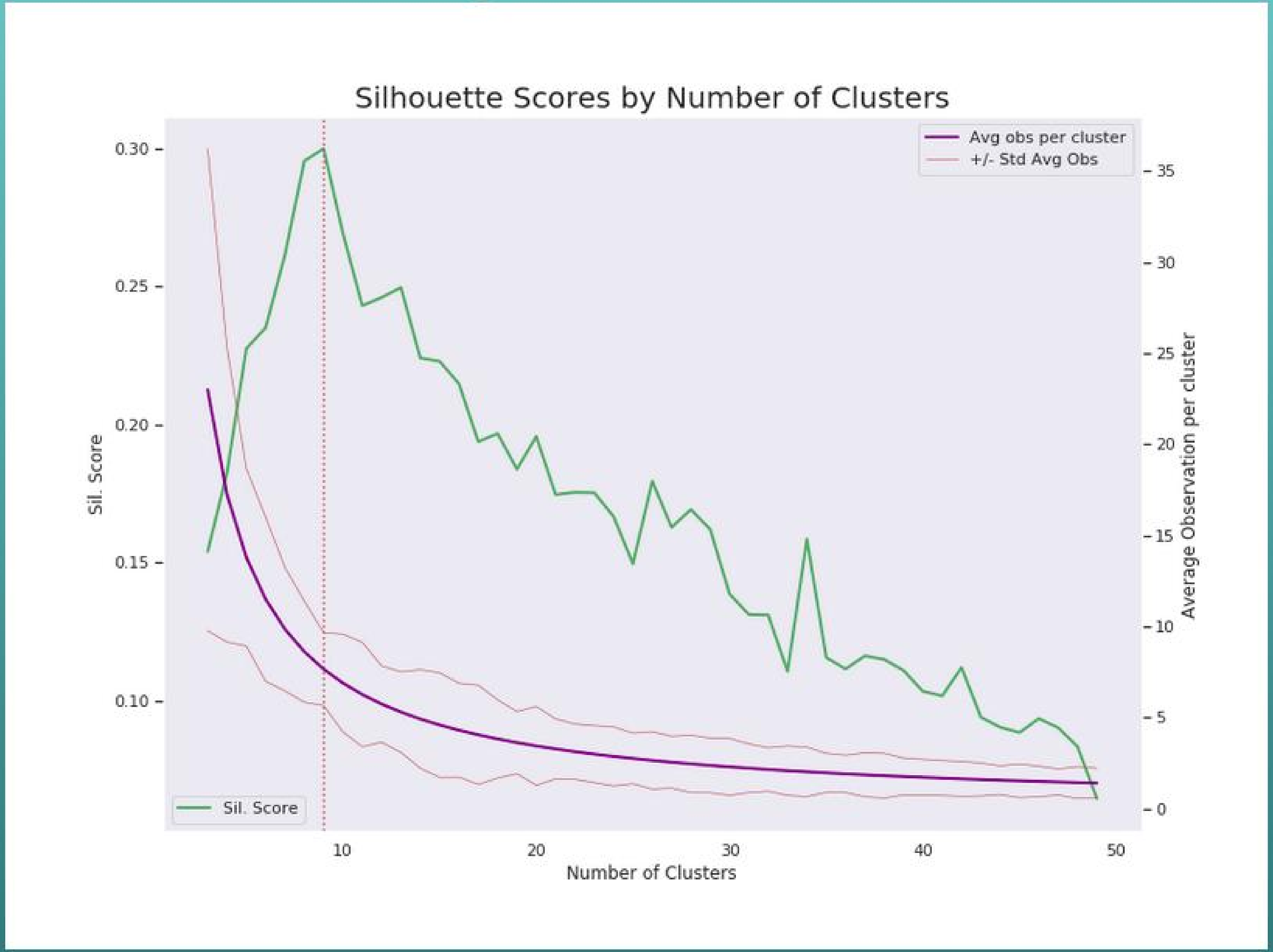


Mean Shift (all bandwidth)

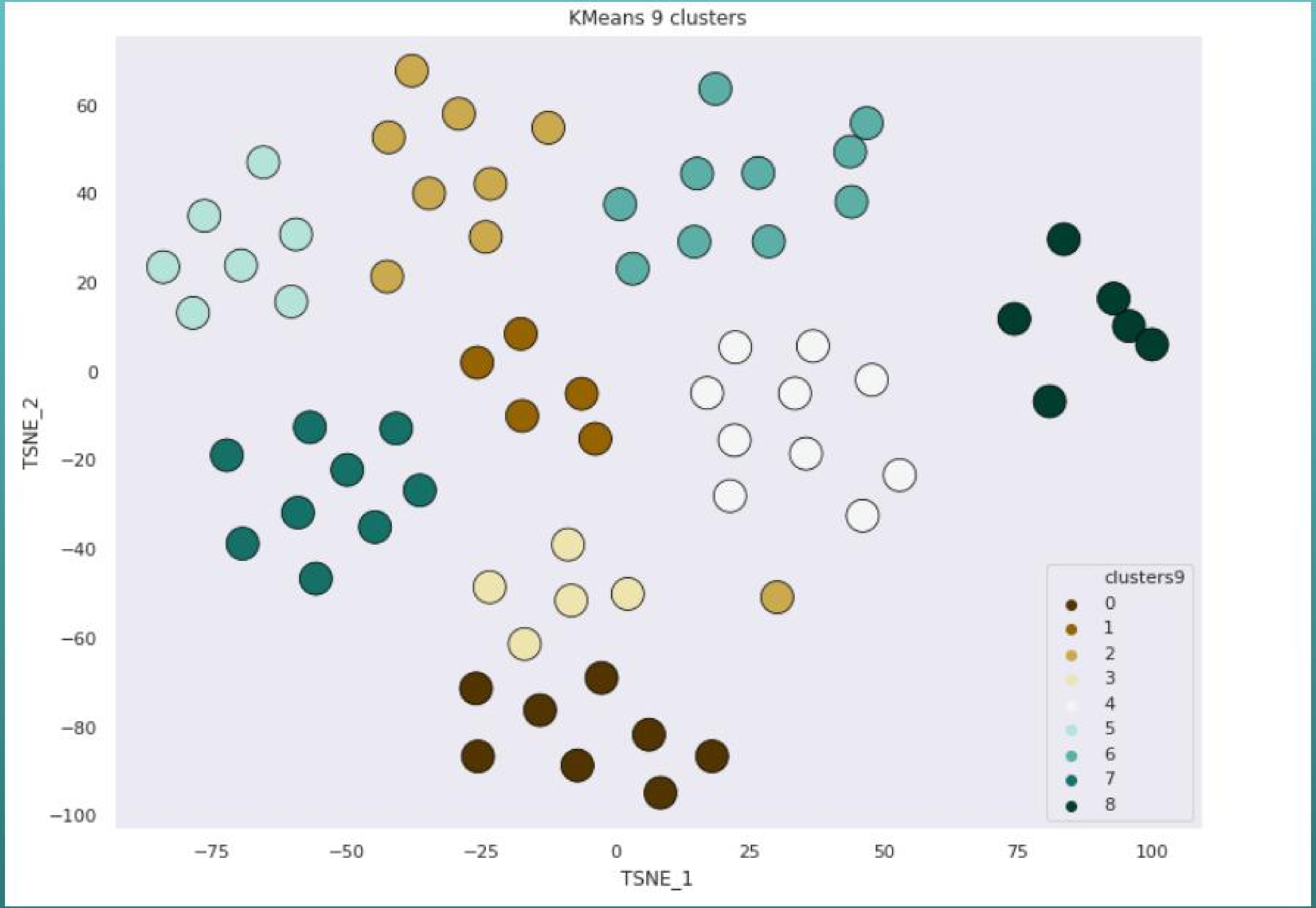


KMeans - 10 Clusters

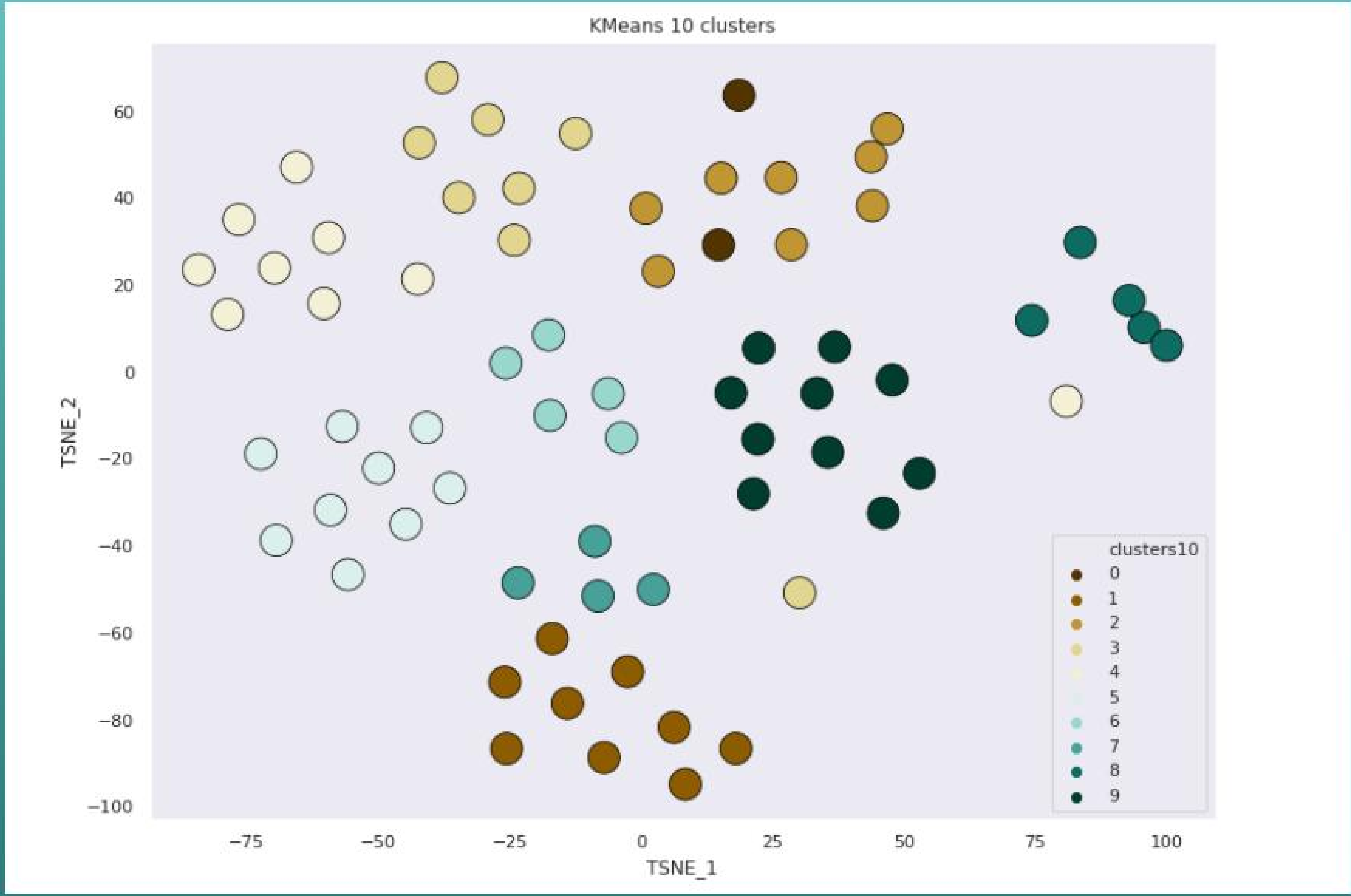
Clustering the Data with KMeans



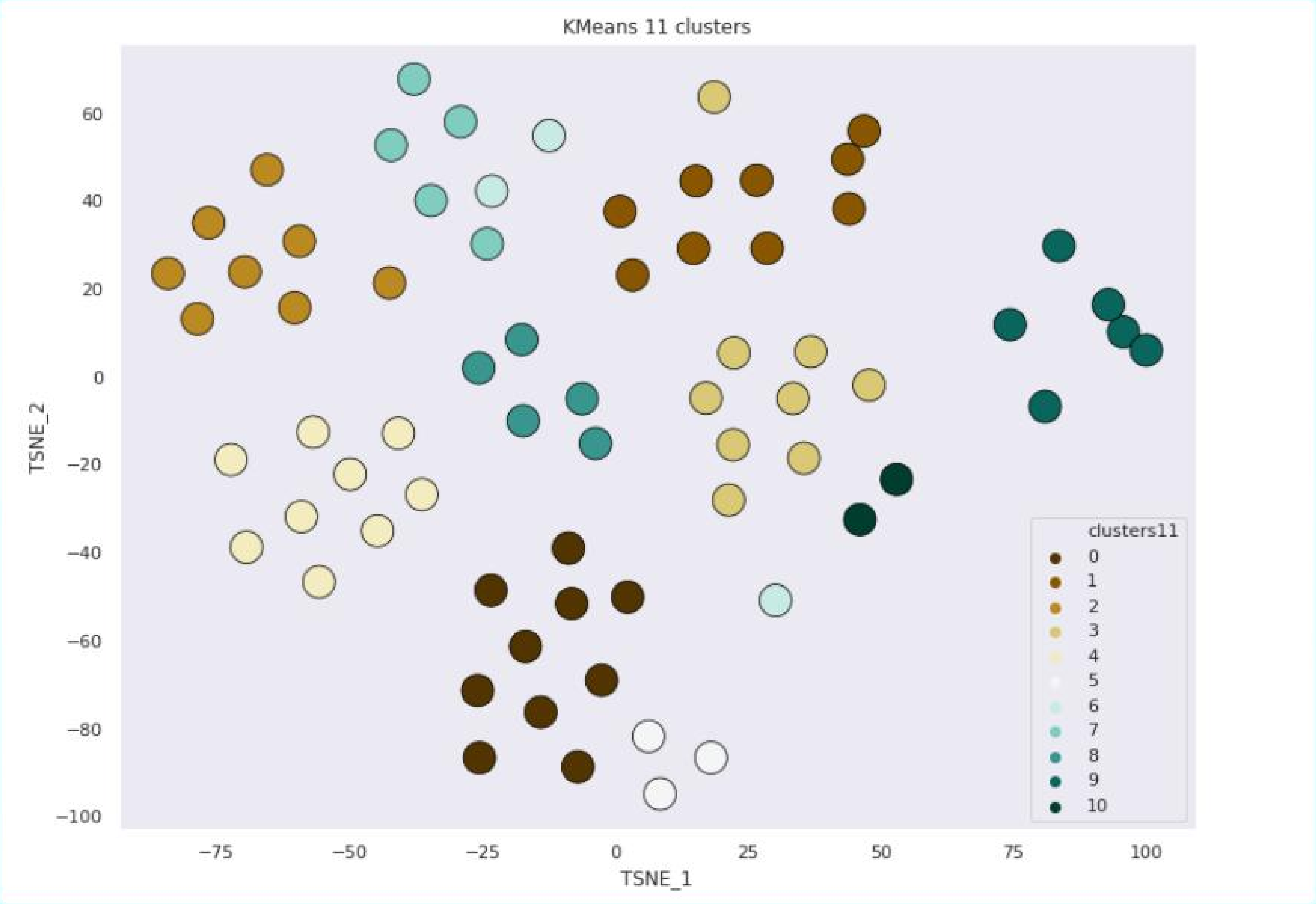
KMeans 9 Clusters



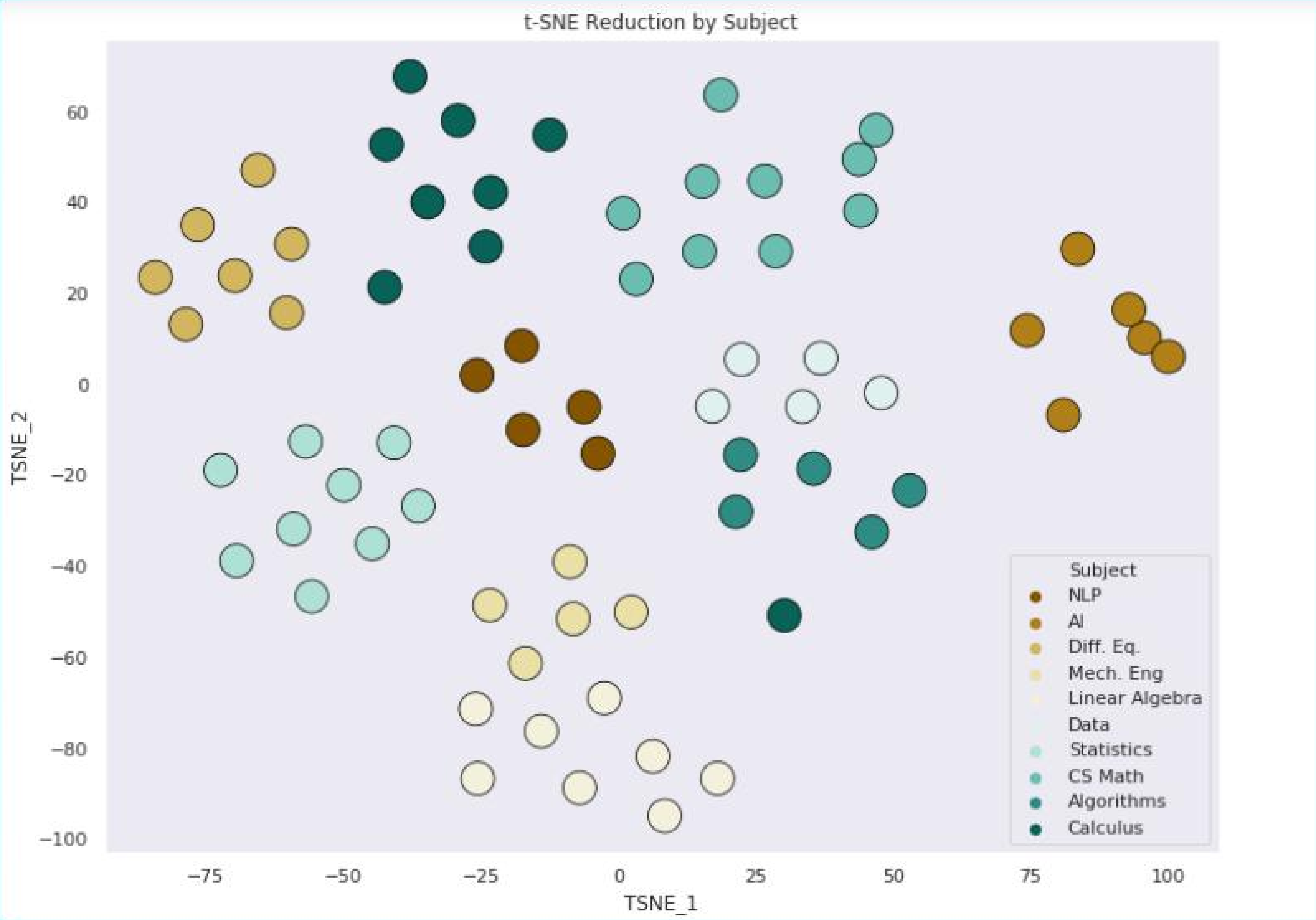
KMeans 10 clusters

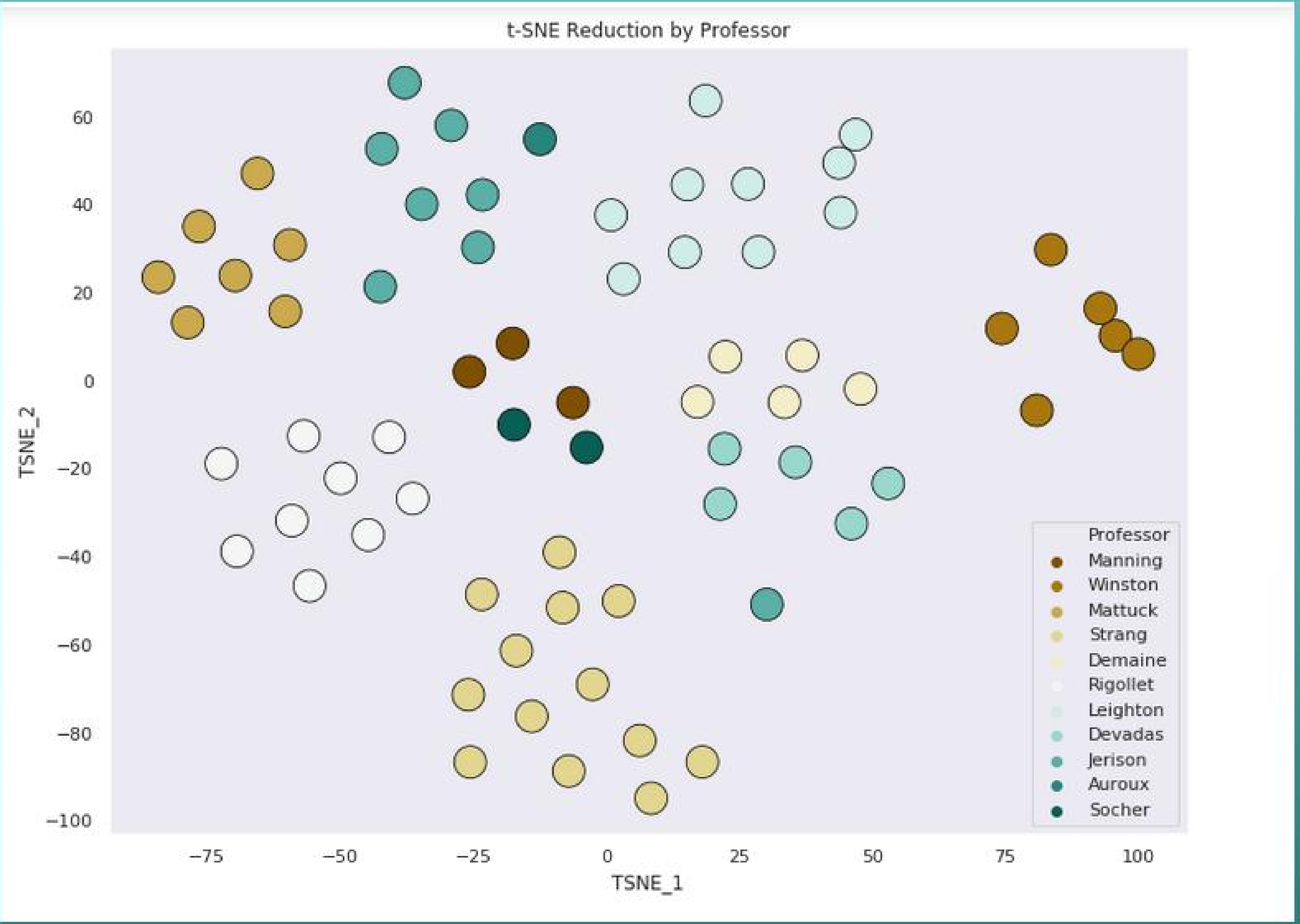


KMeans 11 Clusters

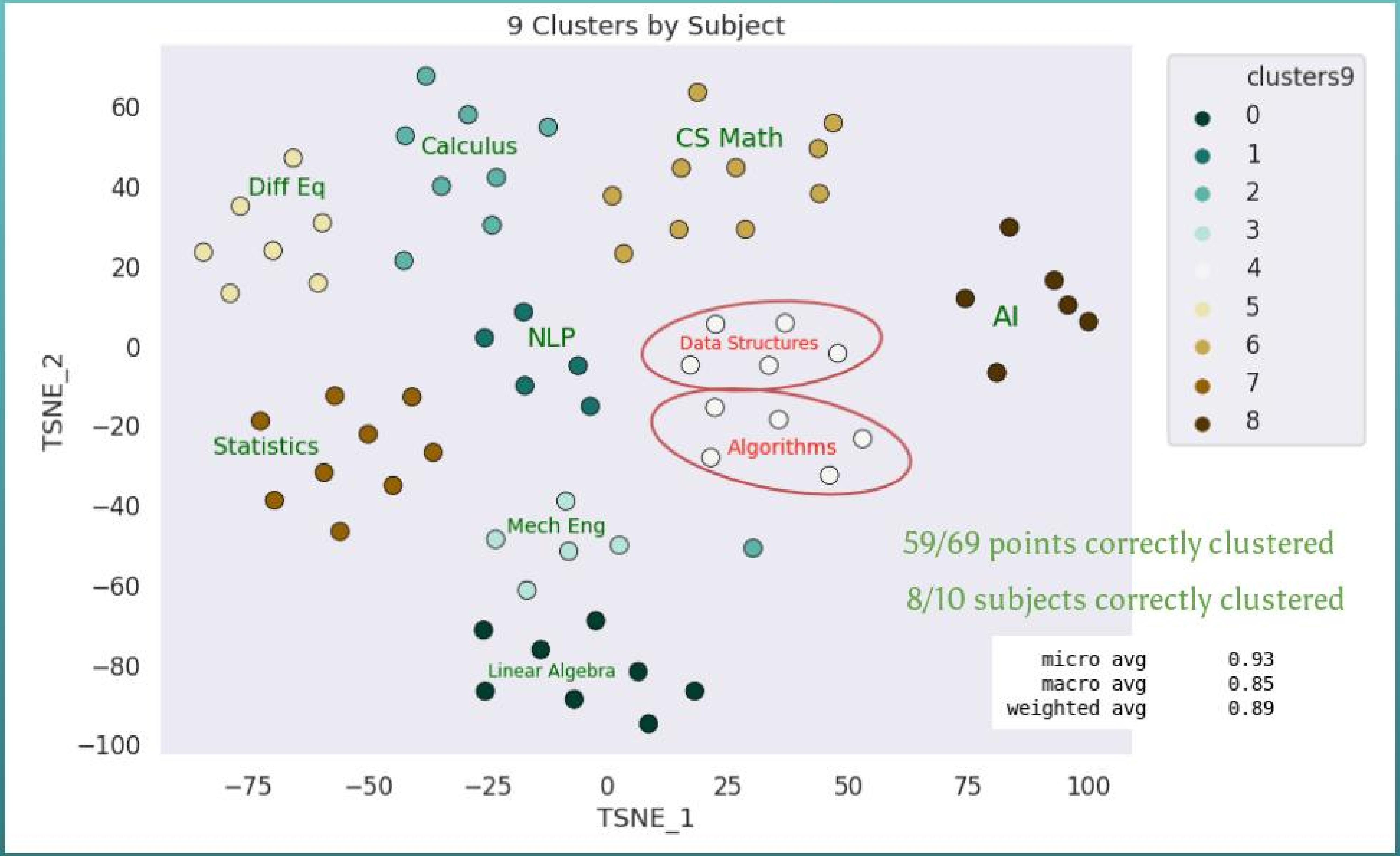


Actual labels

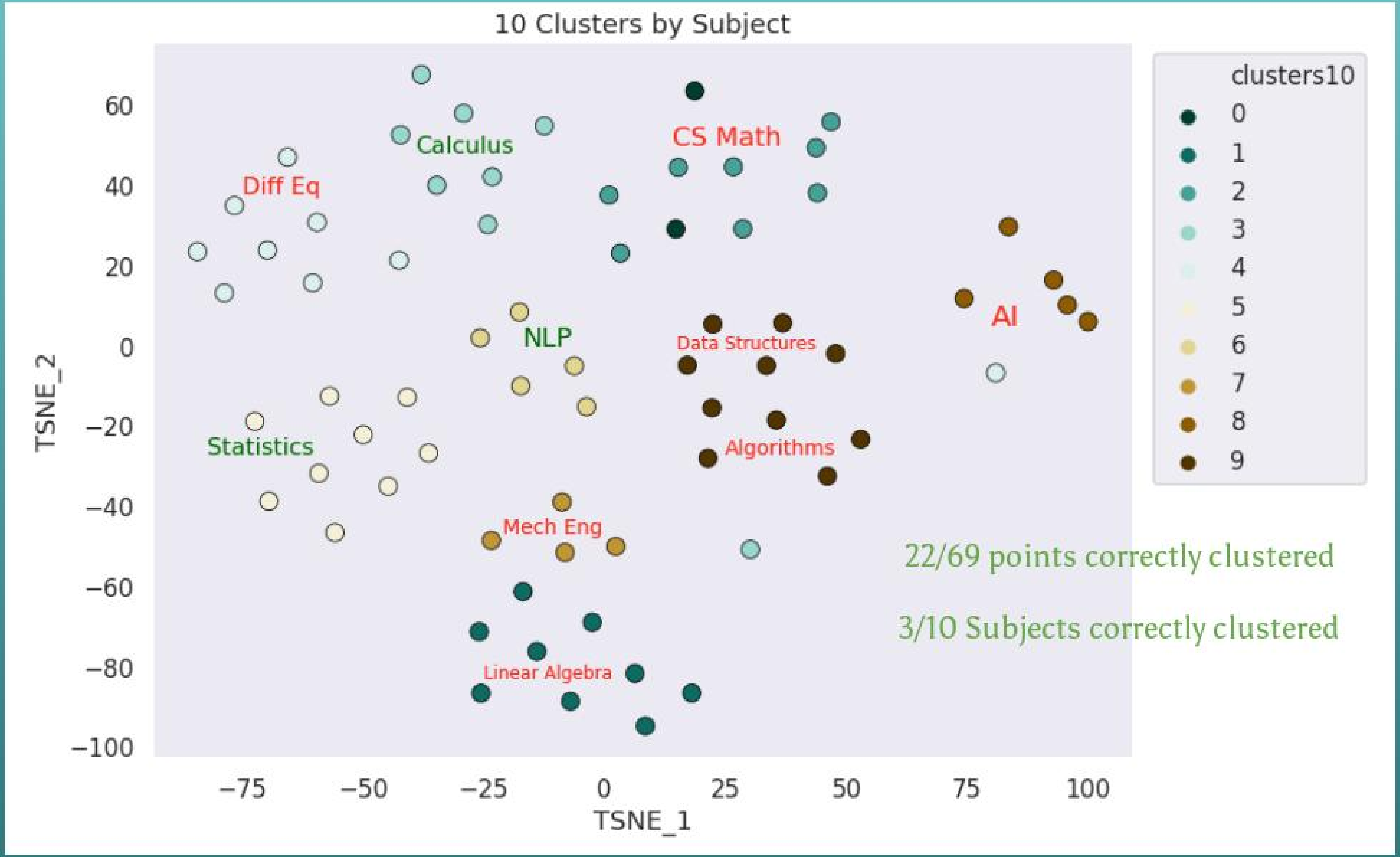




9 clusters by Subject



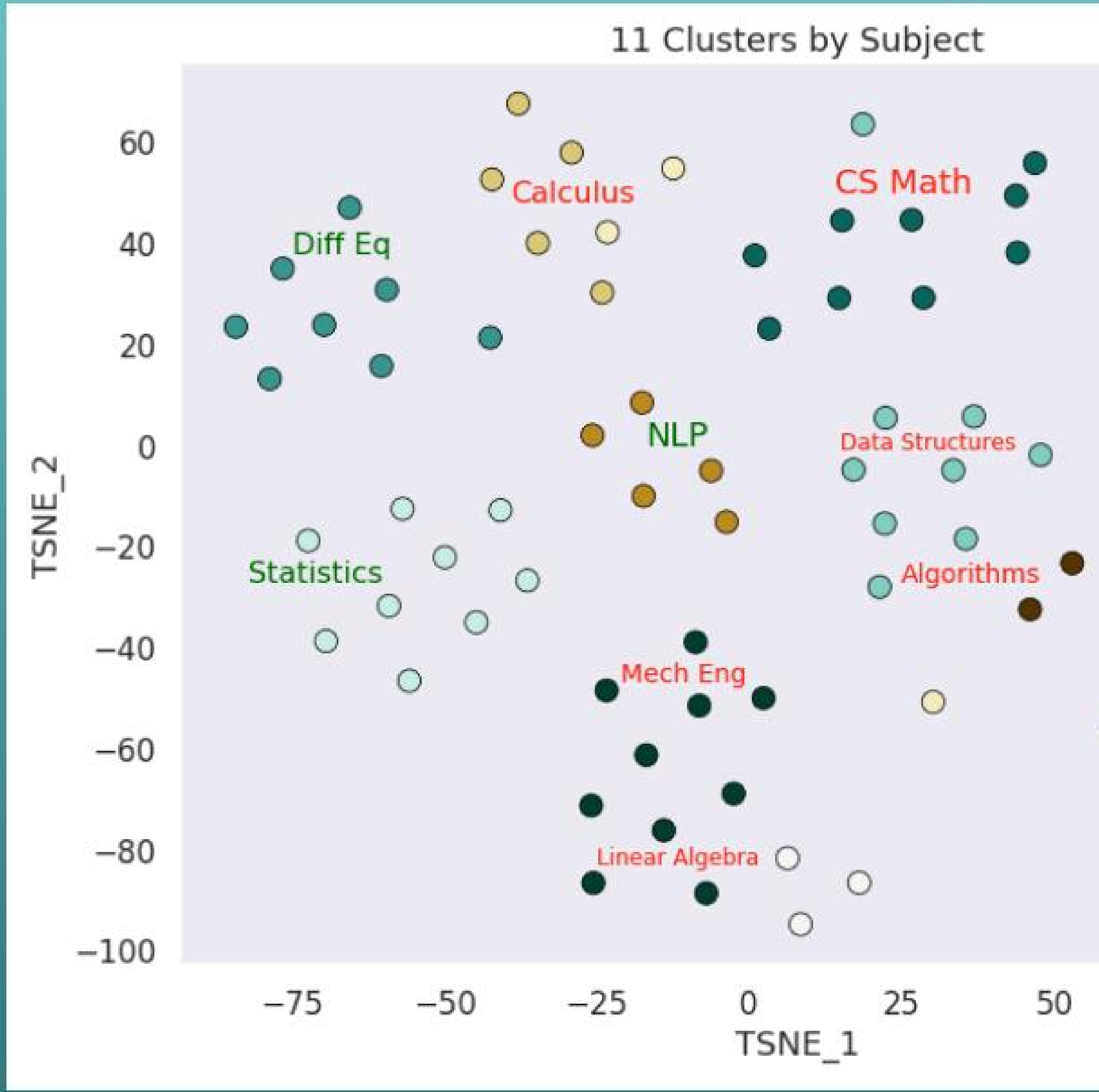
10 Clusters by Subject



22/69 points correctly clustered

3/10 Subjects correctly clustered

11 Clusters by Subject



27/69 points correctly clustered

4/10 Subjects correctly clustered

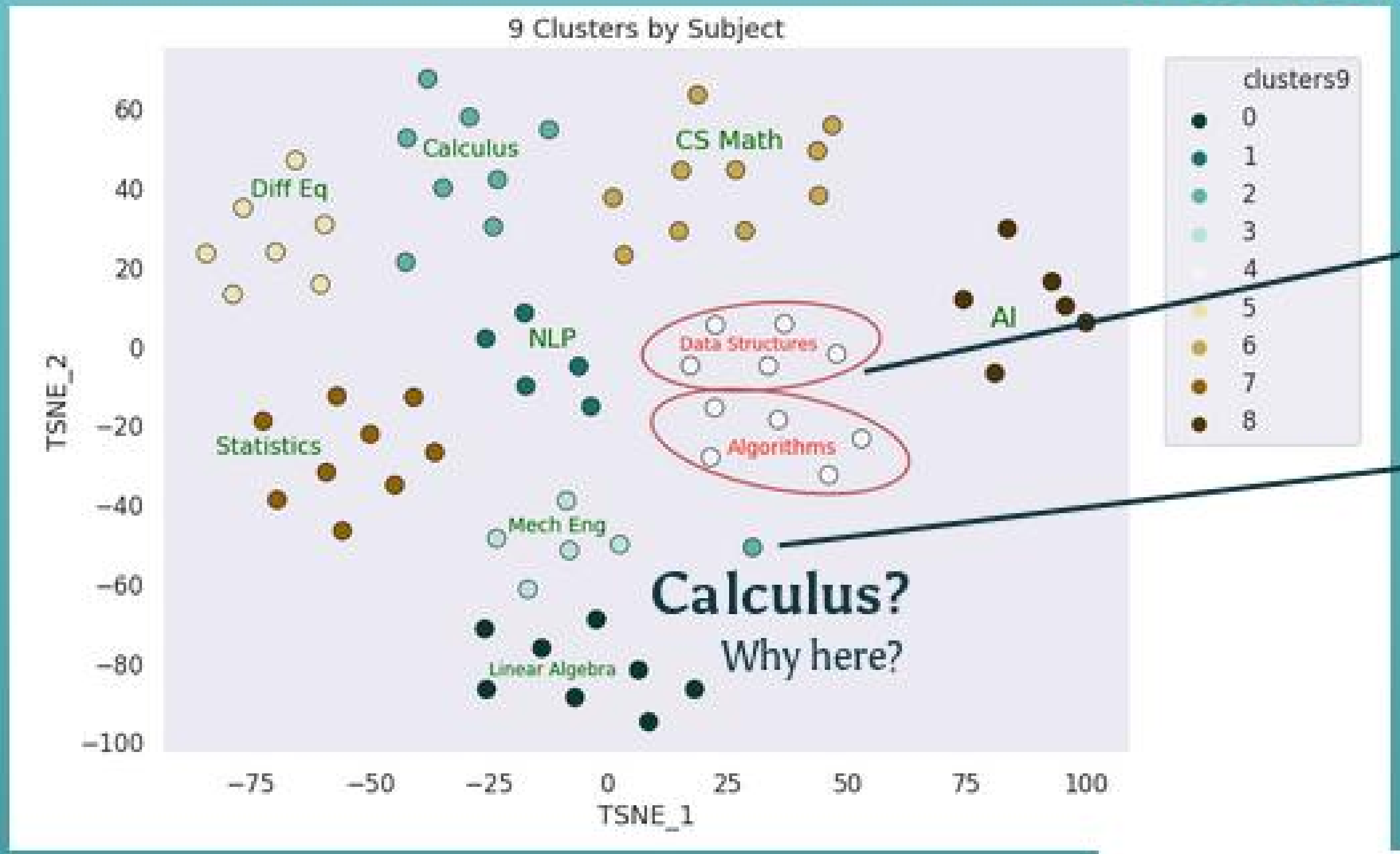
Results of Clustering

	9 Clusters	10 Clusters	11 Clusters
By Subject	85.5%	31.88%	39.13%
By Professor	46.37%	10.14%	33%

	precision	recall	f1-score	support
AI	1.00	1.00	1.00	6
Algorithms	0.00	0.00	0.00	5
CS Math	1.00	1.00	1.00	10
Calculus	1.00	1.00	1.00	9
Data	0.50	1.00	0.67	5
Diff. Eq.	1.00	1.00	1.00	7
Linear Algebra	1.00	1.00	1.00	8
Mech. Eng	1.00	1.00	1.00	5
NLP	1.00	1.00	1.00	5
Statistics	1.00	1.00	1.00	9
micro avg	0.93	0.93	0.93	69
macro avg	0.85	0.90	0.87	69
weighted avg	0.89	0.93	0.90	69

Score is based on cluster completeness. (Only lectures perfectly in their true label group are scored)

But wait...



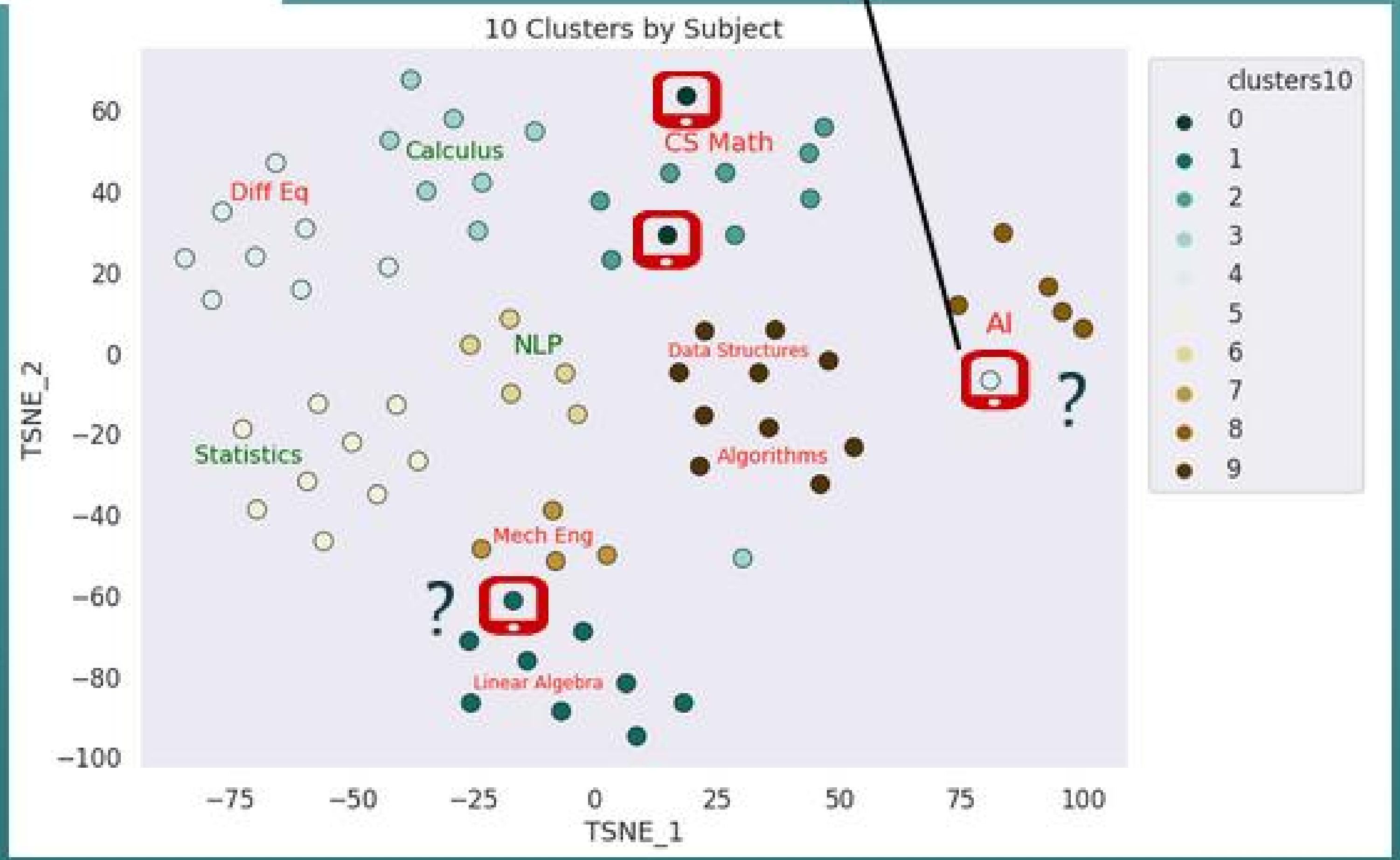
Why can't the KMeans discern between data structures and algorithms?

This calculus lecture ended up far from its cluster during the t-SNE decomposition

Why does this AI lecture get clustered to Differential Equations?

Calculus is *very* close to Differential Equations.

Math for Computer Science, Artificial Intelligence, Algorithms, Data Structures in this context are more closely related to one another than the others. This relationship is captured in coordinates of the lectures



Topic Extraction using Non negative matrix factorization

Data Structures

```
Topic #0:
log divide epsilon square sort query word size factor subtree base s
pace bind afford time achieve overall fit update pay

Topic #1:
time constant number access touch key build linear compute operation
know algorithm update trie need sequence travel change order cost

Topic #2:
tree search binary know way model fast build good problem optimal ba
lanced basically actually obvious nice start question turn black

Topic #3:
node pointer store touch array want root subtree visit path version
new tree ancestor particular let old know leaf rotation

Topic #4:
emde van boas thing size sort root word use think algorithm author w
ay kind number ram happen square basically cache

Topic #5:
item insert want right word interval array list order delete guy sto
re size promote buffer sort small cluster half shift

Topic #6:
like point look set add level rectangle picture kind right mean gues
s want past thing path guy ok access equal
```

Algorithms

```
Topic #0:
tree binary search lg height insert structure balance actually avl d
elete need leaf thing lecture happen check sorted let good

Topic #1:
heap max build heapify property min run invariant structure array no
de maintain child trivial extract root unordered different violate b
ig

Topic #2:
minus divide plus square equal xi sub root epsilon compute great lik
e raise lg newton let method comma function xn

Topic #3:
algorithm complexity problem shall class python talk different set g
ood efficient version correspond comparison input peak write correct
term analyze

Topic #4:
time constant order word item operation case lg linear spend array w
ork run bad sum et cetera landing think bunch

Topic #5:
sort insertion merge like way array use look thing count place theta
example run auxiliary turn structure space kind particular

Topic #6:
log theta base raise write step swap alpha way equal compare complex
ity bound squared mean insertion end cost time prove
```

Do these look *that* different?

Topic Extraction using LDA
(Latent Dirichlet Allocation)

AI

ic #0:
e question answer goal behavior program know build kind forward a
way work backward probability shall leave 80 base particular

ic #1:
us plus alpha equal square sub negative sum integral sample fourt
utter function dx time close oh different situation likewise

ic #2:
tle bit talk want course day subject tell start shall solution le
intelligence model type maybe example artificial turn look

ic #3:
rch path depth want use breadth queue extend goal node good beam
tead quiz order close heuristic pretty british museum

ic #4:
blem solve kind need work transformation way order talk final met
idea table test slagle algorithm program think today huffman

ic #5:
e junction right boundary way label object arrow draw constraint
ange possibility street form face fork discover try possible worl

ic #6:
tor sample partial dot function respect product time street decis
p2 depend great discover want performance value derivative width
nitude

Winston AI 10

Topic #0:
talk stuff thing learning near like pattern neighbor word recognit
ion computer straight today base magazine lot town country traject
ory position

Topic #1:
say feature vector come day way worth invent compare recognition h
appen easy good library value guy decision control world robot

Topic #2:
area total cover want like concert hole electrical guy custodian a
ttempt come sort include shall measure maximum idea knowledge let

Topic #3:
stuff velocity acceleration guy want know ball think sleep speed p
articular trajectory need associate movement value position look t
alk arm

Topic #4:
little good variance particular piece movement try 100 record asso
ciate want shall just stuff think table easy prime right time

Topic #5:
perpendicular divide bisector human space article area maximum sim
ple instead use construct boundary line decision thing talk comput
er want equal

Topic #6:
10 pitch sleep need want memory try 25 know day original 20 record
hour likely run simple time worth guess

Differential Equations

Topic #0:
value plus negative minus form want equal answer prime way zero w
ite real general number positive time standard law course

Topic #1:
point y1 calculate mean word factor curve minute zero omega angle
rt slope cosine integral theorem times sort y2 room

Topic #2:
number theta complex exponential cosine unit vector angle know si
e high word involve case product formula hand euler expression la

Topic #3:
solution curve initial word constant equation condition start for
half equal salt steady concentration long particular differential
geometric state term

Topic #4:
let little constant temperature good euler method example concent
ation want use formula bit work model step equation external size
try

Topic #5:
function want equal word number way talk spring method use think
requency respect slope mass good draw like complex okay

Topic #6:
equation hand solve differential linear right solution left term
omogeneous kind talk form prime like method function time bernoul
i think

Modeling with TF-IDF vectorization

```
#Split the data into train and test set.
X = np.array(sentences['text'])
y = np.array(sentences[['Professor','Subject','filename']]) #keep all labels


#As we are modeling, vectorize all of the lectures, before splitting the data

#Instantiate tf-idf vectorizer
vectorizer = TfidfVectorizer(max_df=0.50, # drop words that occur in more 50% of the sentences
                             min_df=25, # only use words that appear at least 25
                             stop_words='english',
                             lowercase=True,
                             use_idf=True,
                             norm=u'l2',
                             smooth_idf=True)
```

```
Xt = vectorizer.fit_transform(X)
tfidf_vecs = pd.DataFrame(Xt.todense())
print(tfidf_vecs.shape)
tfidf_vecs.head()
```

```
(92, 334)
```

	0	1	2	3	4	5	6	7	8	9	...	324	325	326
0	0.014227	0.0	0.014004	0.0	0.00000	0.011054	0.000000	0.000000	0.013378	0.00000	...	0.00000	0.000000	0.000000
1	0.000000	0.0	0.015423	0.0	0.00000	0.048699	0.000000	0.000000	0.000000	0.00000	...	0.00000	0.000000	0.000000
2	0.000000	0.0	0.000000	0.0	0.00000	0.016340	0.000000	0.000000	0.000000	0.00000	...	0.00000	0.000000	0.000000
3	0.000000	0.0	0.000000	0.0	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	...	0.00000	0.020213	0.000000
4	0.000000	0.0	0.000000	0.0	0.00785	0.015006	0.003477	0.003925	0.000000	0.00338	...	0.00798	0.000000	0.007374

Initial Results with TF-IDF vectores

Logistic Regression

	precision	recall	f1-score	support
AI	0.00	0.00	0.00	2
Algorithms	0.50	0.25	0.33	4
CS Math	0.25	1.00	0.40	1
Calculus	0.67	0.50	0.57	4
Data	1.00	0.50	0.67	2
Diff. Eq.	0.67	1.00	0.80	2
Linear Algebra	0.67	1.00	0.80	2
Mech. Eng	1.00	0.33	0.50	3
NLP	1.00	1.00	1.00	2
Statistics	0.33	1.00	0.50	1
micro avg	0.57	0.57	0.57	23
macro avg	0.61	0.66	0.56	23
weighted avg	0.65	0.57	0.55	23

0	0	2	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0
0	0	0	2	0	1	0	0	0	1
0	1	0	0	1	0	0	0	0	0
0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	1	0	0	1	1	0	0
0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	1

Logistic Regression

61%

Random Forest

	precision	recall	f1-score	support
AI	1.00	0.50	0.67	2
Algorithms	1.00	0.50	0.67	4
CS Math	0.50	1.00	0.67	1
Calculus	0.80	1.00	0.89	4
Data	0.67	1.00	0.80	2
Diff. Eq.	0.50	0.50	0.50	2
Linear Algebra	0.50	1.00	0.67	2
Mech. Eng	1.00	0.33	0.50	3
NLP	1.00	1.00	1.00	2
Statistics	1.00	1.00	1.00	1
micro avg	0.74	0.74	0.74	23
macro avg	0.80	0.78	0.74	23
weighted avg	0.83	0.74	0.72	23

1	0	0	0	0	1	0	0	0	0
0	2	1	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	2	1	0	0
0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	1

Random Forest

80%

Multinomial NB

	precision	recall	f1-score	support
AI	0.00	0.00	0.00	2
Algorithms	0.75	0.75	0.75	4
CS Math	0.00	0.00	0.00	1
Calculus	0.67	0.50	0.57	4
Data	1.00	0.50	0.67	2
Diff. Eq.	0.67	1.00	0.80	2
Linear Algebra	0.50	1.00	0.67	2
Mech. Eng	1.00	0.33	0.50	3
NLP	0.67	1.00	0.80	2
Statistics	0.50	1.00	0.67	1
micro avg	0.61	0.61	0.61	23
macro avg	0.57	0.61	0.54	23
weighted avg	0.64	0.61	0.58	23

0	0	1	0	0	0	0	0	1	0
0	3	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	2	0	1	0	0	0	1
0	1	0	0	1	0	0	0	0	0
0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	2	1	0	0
0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	1

Multinomial NB

57%

K Neighbors

	precision	recall	f1-score	support
AI	0.00	0.00	0.00	2
Algorithms	0.67	0.50	0.57	4
CS Math	0.00	0.00	0.00	1
Calculus	1.00	0.50	0.67	4
Data	0.50	0.50	0.50	2
Diff. Eq.	0.50	1.00	0.67	2
Linear Algebra	1.00	1.00	1.00	2
Mech. Eng	1.00	1.00	1.00	3
NLP	0.67	1.00	0.80	2
Statistics	1.00	1.00	1.00	1
micro avg	0.65	0.65	0.65	23
macro avg	0.63	0.65	0.62	23
weighted avg	0.70	0.65	0.65	23

0	0	1	0	0	0	0	0	1	0
0	2	2	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	2	0	2	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	1

KNN

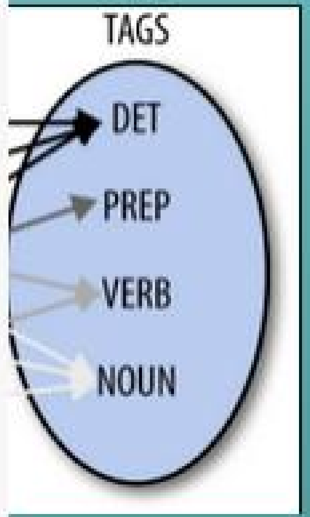
63%

Other feature generation

sdoc
(So, let, us, start, right, away, with, stuff,...
(, to, , So, far, we, have, learned, ab...

```
pos_count = get_pos(sentences.sdoc, True)
```


```
#iterate over each lecture extracting lists of POS for each sentence
def get_pos (doc_list, norm):
    #start timer, creat lists
    t1 = time.time()
    pos_list = [] #list of all POS
    poss_list = []#list of sentences as POS

    #iterate over list of spacy docs
    for lecture in doc_list:
        pss = []
        #Extract POS
        for token in lecture:
            pss.append(token.pos_)
            pos_list.append(token.pos_)
        poss_list.append(pss)

    #Set up up a DataFrame to count occurance of POS per lecture
    pos_df = pd.DataFrame(columns=set(pos_list))
    pos_df['pos_sent'] = poss_list
    pos_df.loc[:, pos_list] = 0

    for i, sentence in enumerate(pos_df['pos_sent']):

        # Convert the sentence words to POS
        words = pos_df.pos_sent[i]

        # Populate the row with word counts.
        for word in words:
            pos_df.loc[i, word] += 1

    # get total pos count in the lecture
    pos_df['length'] = pos_df.drop(['pos_sent'],1).sum(axis=1)

    if norm == True:
        #if True, divids POS count by length (total POS count)
        for col in pos_df.drop(['pos_sent','length'],1).columns:
            pos_df[col] = pos_df[col]/pos_df.length


    pos_df.drop(['pos_sent'],1,inplace=True)

    print("time: {} minutes".format((time.time()-t1)/60))
    return pos_df
```



norm=

False

True

- Extract parts of speech (POS)

```
['PRON', 'ADV', 'ADJ'],
['ADV', 'VERB', 'NOUN', 'NOUN', 'NOUN', 'NOUN', 'NOUN'],
['ADV', 'ADV', 'ADV', 'NOUN', 'VERB', 'NOUN', 'NOUN', 'ADP', 'INTJ'],
['CCONJ', 'NOUN', 'VERB', 'NOUN'],
['CCONJ', 'NOUN', 'VERB', 'NOUN', 'NOUN', 'ADV', 'VERB', 'NOUN', 'NOUN'],
```

- Count occurrence of POS by lecture

	ADP	PART	PUNCT	PRON	NOUN	NUM	VERB	SPACE
0	524	104	1001	467	891	137	964	0
1	677	151	783	573	1071	51	1144	2
2	629	104	1113	476	941	169	1139	0
3	632	116	1152	599	898	127	1143	2
4	1128	322	1935	1125	2300	140	2769	2

- Divide each POS by lecture length

CCONJ	X	ADV	PROPN	DET	SYM	length	lecture
0.0310872	0.000521014	0.0793678	0.0109413	0.096214	0.00277874	5758.0	aurouxmcalc1
0.040658	0.011018	0.0856611	0.00574178	0.0853507	0.00481068	6444.0	aurouxmcalc11
0.0305839	0	0.0875811	0.0114303	0.0946864	0.0021625	6474.0	aurouxmcalc2
0.0359928	0.00479904	0.0911818	0.00419916	0.0920816	0.00524895	6668.0	aurouxmcalc5
0.0323816	0.00111421	0.0983287	0.00473538	0.103273	0.000139276	14360.0	demainedata1

Logistic Regression

	ADP	PART	PUNCT	PRON	NOUN
0	524	104	1001	467	891

using POS only

Random Forest

micro avg	0.87	0.87	0.87	23
macro avg	0.90	0.90	0.87	23
weighted avg	0.93	0.87	0.88	23

90%

2	0	0	0	0	0	0	0	0	0
1	3	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
1	0	0	3	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	1

micro avg	0.74	0.74	0.74	23
macro avg	0.74	0.72	0.68	23
weighted avg	0.79	0.74	0.73	23

74%

1	1	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	2	0	0	2	0	0	0
0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	1	0	0	0	2	0	0
0	0	0	0	0	0	0	0	2	0
0	0	1	0	0	0	0	0	0	0

PROPN	DET	SYM	length
0.0109413	0.096214	0.00277874	5758.0

using POS / len(total_lecture_pos)

micro avg	0.30	0.30	0.30	23
macro avg	0.24	0.35	0.27	23
weighted avg	0.23	0.30	0.25	23

24%

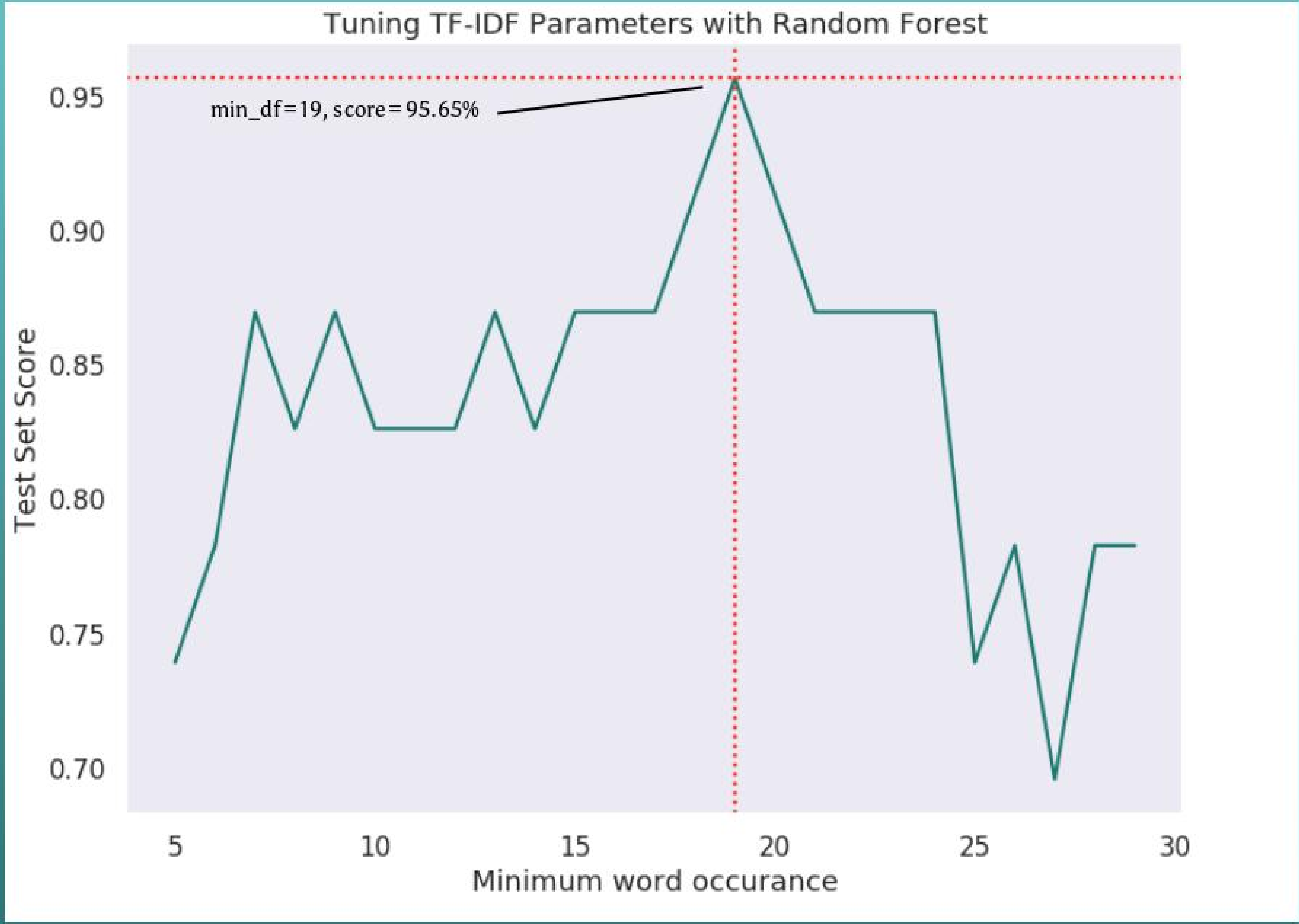
2	0	0	0	0	0	0	0	0	0
0	2	0	2	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	3	0	0	0
0	0	0	0	0	0	0	0	0	2
0	1	0	1	0	0	0	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	1	0	2	0	0	0	0
0	0	0	0	0	0	0	0	0	2
0	0	1	0	0	0	0	0	0	0

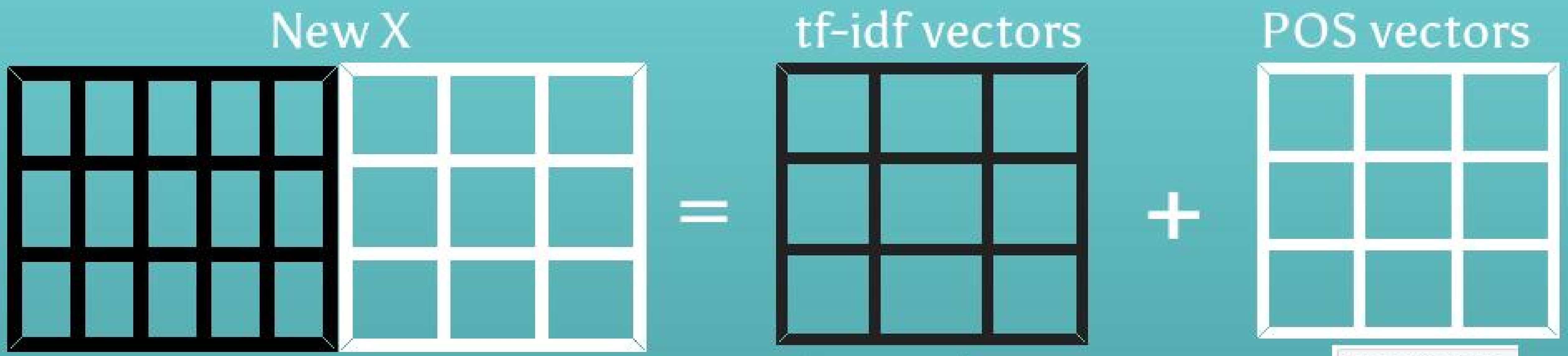
micro avg	0.87	0.87	0.87	23
macro avg	0.92	0.90	0.89	23
weighted avg	0.90	0.87	0.87	23

92%

1	0	0	1	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	2	0	2	0	0	0	0
0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	1

Parameter Search






```
model_vecs.shape
(92, 351)
```

```
tfidf_vecs.shape
(92, 334)
```

```
pos_count.shape
(92, 17)
```

Cross Validation 5 folds `[0.95, 0.86, 0.95, 0.94, 0.93]`

mean `0.93`

tf-idf df_min = 25

Random Forest (n_estimators=200, max_depth=4, min_samples_leaf=4, random_state=43, class_weight='balanced')

	precision	recall	f1-score	support
AI	1.00	1.00	1.00	2
Algorithms	1.00	1.00	1.00	4
CS Math	1.00	1.00	1.00	1
Calculus	1.00	1.00	1.00	4
Data	1.00	1.00	1.00	2
Diff. Eq.	1.00	1.00	1.00	2
Linear Algebra	1.00	1.00	1.00	2
Mech. Eng	1.00	1.00	1.00	3
NLP	1.00	1.00	1.00	2
Statistics	1.00	1.00	1.00	1
micro avg	1.00	1.00	1.00	23
macro avg	1.00	1.00	1.00	23
weighted avg	1.00	1.00	1.00	23

2	0	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	0	1

Clustering:

+ Able to identify similar subjects.
- Unable to decipher closely related subjects
- Unable to decipher professors

Modeling:

+ Very accurate
+ able to decipher professors

For further study

- Scale the data collection
  - Programmatically obtain lecture subtitles from youtube's API

- Generate an overall rating of each lecture
  - Youtube ratings, comments, view counts
  - Create new features based on sentiment analysis of comments

- Predict the quality of newly posted content
  - match most relevant new content for a given user