REPORT

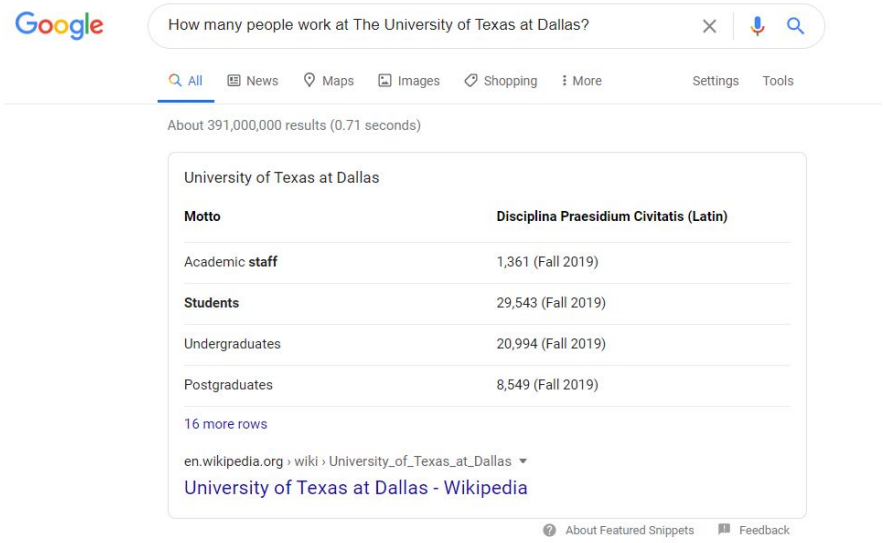
Closed Domain Question Answering System

MOTIVATION:

Language Understanding Capabilities of computers is crucial to many modern applications such as voice assistants,autocomplete suggestions for emails and answering questions by users in search engines.Out of them Reading Comprehension is an important and challenging problem. Reading Comprehension consists of a pipeline designed to answer the question in natural language. One of the challenging tasks of Reading Comprehension is Closed Domain Question Answering. Solving such tasks requires a semantic and syntactic understanding of the question as well as context paragraph.

APPLICATION:

Search Engines, Knowledge Powered Virtual Assistant and Virtual Question Answering



PROBLEM STATEMENT:

In our project ,we implemented Machine Learning and Deep Learning approaches for predicting answers to questions for corresponding paragraphs.

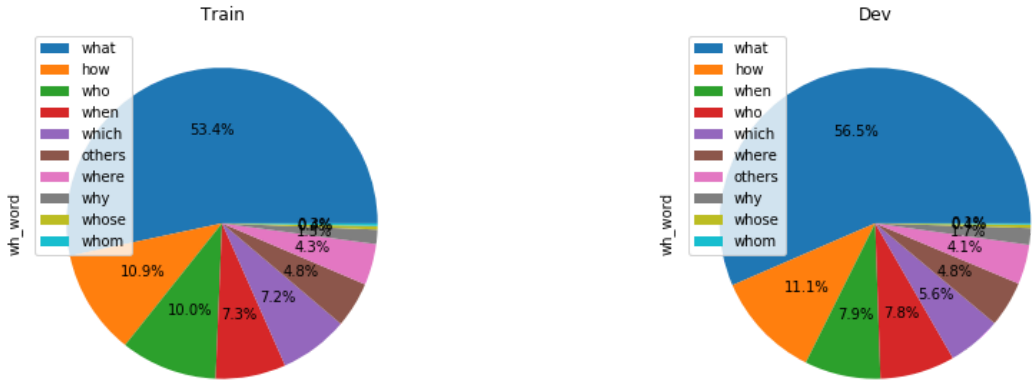
The dataset contains a question, paragraph(context) and an answer for each sample annotated by human labour.

- Deep Learning: Predicts the exact span(start and end point) of the answer in a paragraph.
- Machine Learning: Predict the sentence which contains the span answer.

DATASET INFORMATION

SQuAD 2.0	**Train**	**Development**	**Test**
Total examples	130319	11873	8,862
Negative Examples	43498	5945	4332
Positive Examples	86821	5928	4,530
Total Articles	442	35	28
Articles with negatives	285	35	28

The Stanford Question Answering Dataset (SQuAD 2.0) permits specialists to plan ML/AI models for reading comprehension tasks. Below is a distribution of Wh-words(ex.what) in train and dev sets Which shows the variety question and pose great challenge to learning algorithms



PROBLEM FORMULATION:

Deep Learning:

Paragraph consists of *l* no. of tokens(*p1,p2,..pl*) and question consists of *k* no. of token (*q1,q2..qk*). Our objective is to predict a span (start_index, end_index) such that 1<start_index<end_index<*l.* So the corresponding string is P_{start_index},P_{end_index} is the answer to the question.

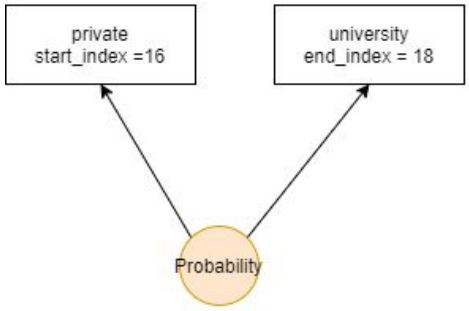
	question	paragraph	answer_span	start	end	q_count	p_count	start_token	end_token
0	When did Beyonce start becoming popular?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	in the late 1990s	269	286	6	109	56.0	59.0
1	What areas did Beyonce compete in when she was...	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	singing and dancing	207	226	11	109	44.0	46.0
2	When did Beyonce leave Destiny's Child and bec...	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	2003	526	530	11	109	112.0	112.0
3	In what city and state did Beyonce grow up?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	Houston, Texas	166	180	9	109	36.0	38.0
4	In which decade did Beyonce become famous?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	late 1990s	276	286	7	109	58.0	59.0

Machine Learning:

Question Answering is formulated as supervised learning problem and training examples are in the form of (paragraph,question,answers).We slightly modified the task for machine learning algorithms. Instead of predicting two probabilities for span, we are predicting the sentence index containing answers to simplify the problem to make it suitable for our algorithm.
We are basically learning function *f(paragraph,question)-> sentence index*

	question	paragraph	answer_span	start	end	q_len	p_len	answer_span_len	sentences	sentences_len	target
0	When did Beyonce start becoming popular?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	in the late 1990s	269	286	6	109	4	[Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ ...	4	2
1	What areas did Beyonce compete in when she was...	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	singing and dancing	207	226	11	109	3	[Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ ...	4	2
2	When did Beyonce leave Destiny's Child and bec...	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	2003	526	530	11	109	1	[Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ ...	4	4
3	In what city and state did Beyonce grow up?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	Houston, Texas	166	180	9	109	2	[Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ ...	4	2
4	In which decade did Beyonce become famous?	Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ b...	late 1990s	276	286	7	109	2	[Beyoncé Giselle Knowles-Carter (/biːˈjɒnseɪ/ ...	4	2

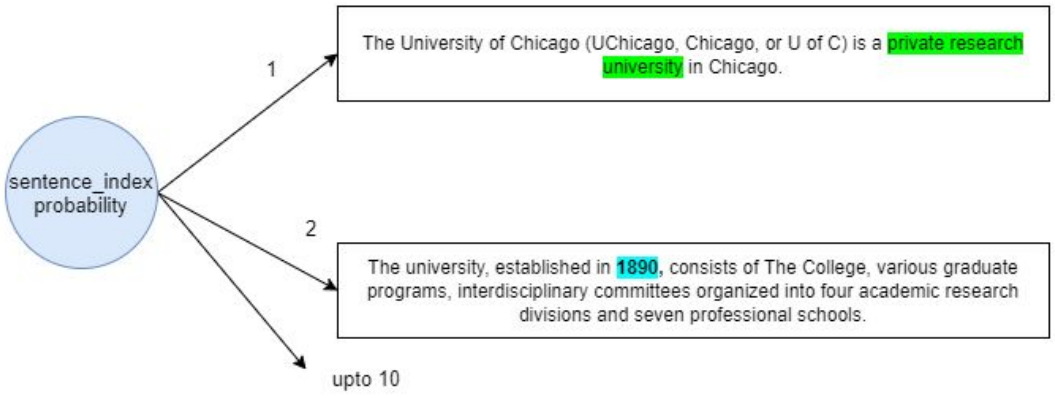
The University of Chicago (UChicago, Chicago, or U of C) is a private research university in Chicago. The university, established in 1890, consists of The College, various graduate programs, interdisciplinary committees organized into four academic research divisions and seven professional schools.



Question: What kind of university is the University of Chicago?

Ground Truth Answers: private research university

Deep Learning approach



Machine Learning Approach

Example of Problem Formulation

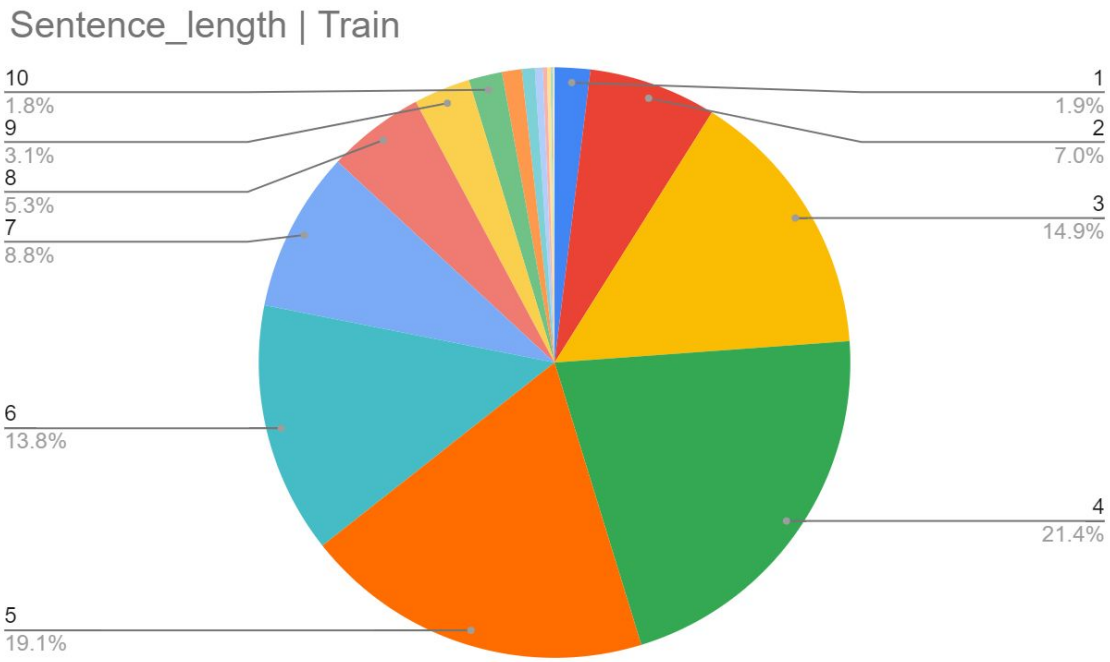
DATA PREPROCESSING

“Machine Learning problems are ill posed.”
Understanding a Machine Learning problem depends mainly on understanding the dataset. To echo this fact, we have studied our dataset thoroughly and preprocessed it to extract several valuable features for our goal.

One of the most challenging parts of a project is pre-processing data for feeding it into algorithms. We tried to make the dataset consistent for both approaches to allow us comparison between deep learning and machine learning algorithms.

ML= Machine Learning
DL= Deep Learning
Some preprocessing are specific to **ML** and **DL** and some are common.

1. Although unanswered questions can be handled with help of thresholding or other techniques. For the purpose of this project,we removed such examples with no answer so as to simplify the problem of predicting span or sentence index.[**DL,ML**]
2. Due to formulation of our problem setting in ML, We only kept sentences length varying from 3 to 10 which makes approximately 90% of the dataset. Sentences with length one do not require prediction as it is not a task of prediction if there is only one sentence.We have taken upper bound as 10 and lower bound as 3 so as to reduce sparsity of features due missing values.[**ML**]
3. SQUAD 2.0 dataset contains answer spans in character offset format. For the training of our deep learning based model, we have converted character offset to token offset with the help of spacy tokenizer instead of simple space tokenizer. Choice of tokenizer is crucial and should be consistent across multiple preprocessing cycles.Moreover, we removed examples containing inconsistent with respect to our mapping between character offset and token offset [**DL**]
4. We also removed paragraphs containing more than 156 tokens(spacy requirement) as it is 80% of the training dataset where the maximum number of tokens are 623. It reduces training parameters for deep learning approach and reduces the sparsity of input due to padding.[**DL**]
5. We removed examples containing answers span more than 15 tokens to control the maximum length of the answers.[**DL**]



Distribution of sentence length in paragraphs

FEATURE EXTRACTION

We extracted a total of 5 features to train algorithms.

ML= Machine Learning
DL= Deep Learning

Sentence tokenization : We tokenize each context to sentences and perform further feature extraction on them.**[ML]**

Word tokenization : We tokenize sentences to words and perform further feature extraction on them.**[ML,DL]**

Target : We have extracted the target sentence within a paragraph which contains the answer span for the question[**ML**].

Word Embedding : Word embedding is the representation of words in vector space. It is capable of capturing word's semantic and syntactic similarity, relation with other words, etc. We experimented with multiple embedding dimensions(100d,300d) . We calculate word vectors for a sentence and then perform a max pool operation to get a single feature representing the sentence. Out of several embeddings that we analyzed, 'GLOVE' embedding performed best for

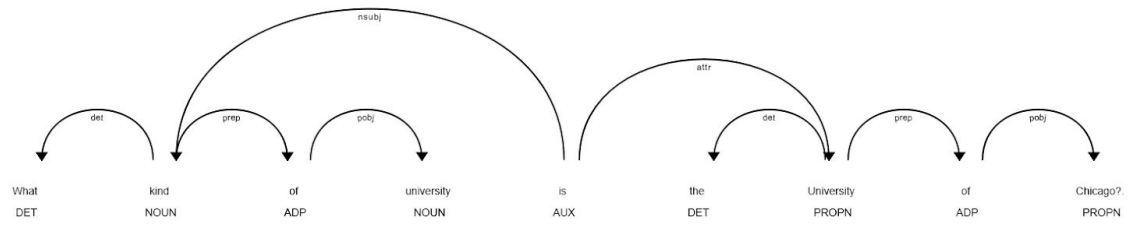
us. Idea behind using word embedding is to use semantic knowledge of language required for answering a question.[**DL,ML**]

TF-IDF : TF-IDF stands for **"Term Frequency — Inverse Document Frequency"**. We calculated the TF IDF score treating each question and sentence of the relevant paragraph as a set of documents. TF-IDF can be seen as local features in a sense that during inference time we don't need external information(pre-trained word embedding) to use this feature.[**ML**]

Euclidean Distance : This is the distance between two vectors in a vector space. For the task of prediction of sentences containing the answer, euclidean distance is calculated between vectors of question and each sentence in the paragraph. This is calculated for both word embedding and tf-idf vector representation of the texts and questions.[**ML**]

Cosine Similarity : As the name suggests, cosine similarity is a measure of similarity of two vectors in a vector space. This represents similarity better than euclidean distance as it considers the alignment / inclination of vectors. Two vectors could be close to each other but be 90 degrees apart in alignment. This is also calculated for both word embedding and tf-idf vector representation of the texts and questions.[**ML**]

Dependency Parsing : We use dependency parsing of spacy to parse a question and sentences from relevant paragraphs. We compare root node word lemma from question w.r.t non leaf node word lemma of each sentence. We used this to match words in question and sentences and check the presence of common words.[**ML**]



Replacing missing feature values : About 90% of the data contains sentence length varying between 3 to 10. But we have fixed our sentence length(categories) as 10 for feature extraction and model training. Samples having less than 10 sentences contain features less than 10 values for each feature. So to maintain uniformity we have filled features with appropriate values such that they don't interfere with our model training.

ALGORITHMS

Deep Learning :

As stated earlier, we are predicting start and end index probability for answer span. We used word embeddings(Glove 100D,300D) of each word and fed it into the standard Bi-direction LSTM network.LSTM output along with self attention is utilized to prepare sentence level representation in case of question. Self attention encodes the importance of certain words into it as against the same weight to all question words. Main idea behind using pre trained word embedding to use existing similarity between words and also to generalize tasks beyond training and validation dataset.

$$b_j = \frac{\exp(\mathbf{w}^{q\intercal}\mathbf{q}_j)}{\sum_{j'}\exp(\mathbf{w}^{q\intercal}\mathbf{q}_{j'})}$$
$$\mathbf{q} = \sum_j b_j \mathbf{q}_j$$

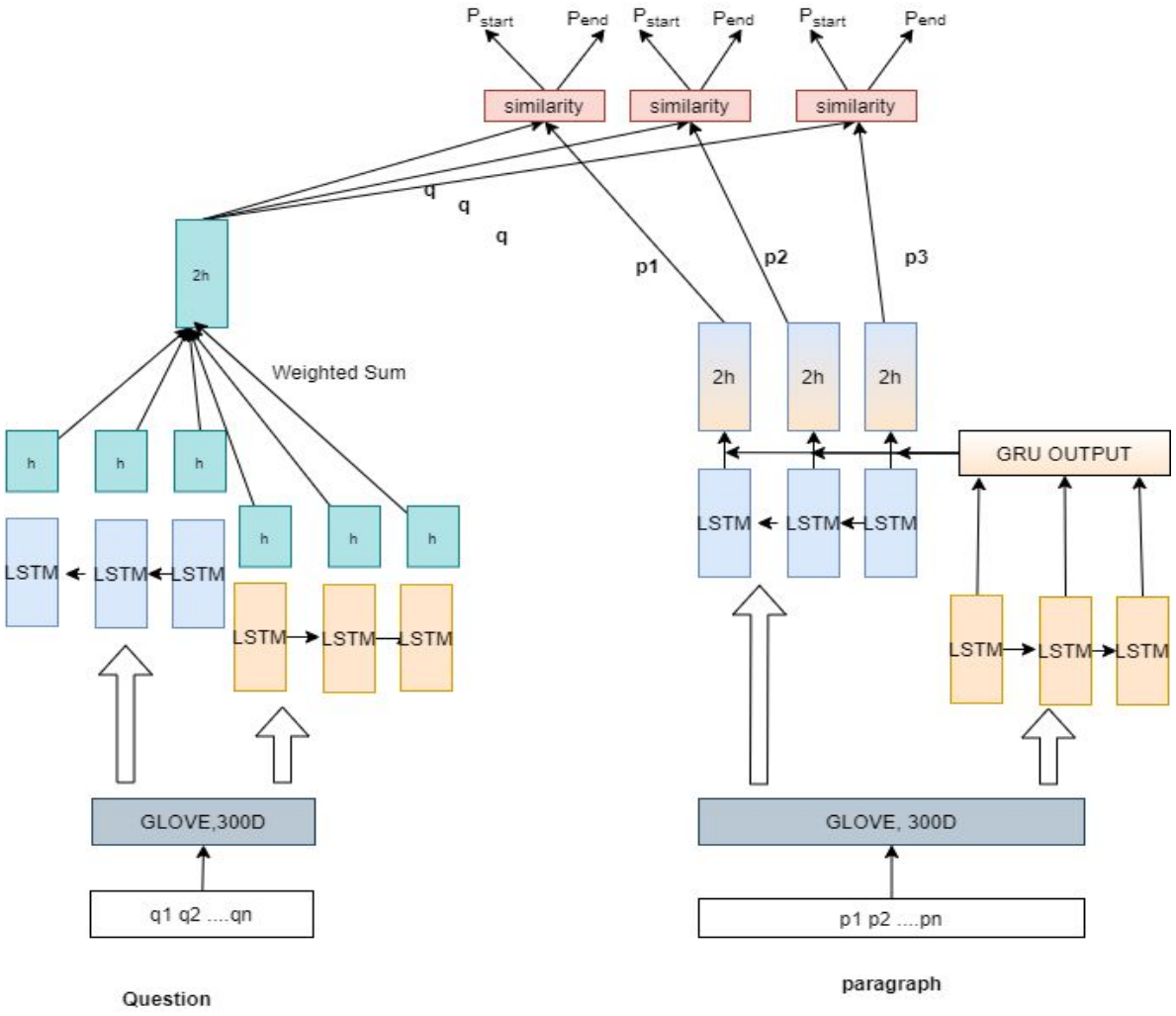
self-attention for question representation

For paragraph, we used word embedding and fed it into Bi-direction LSTM. We employed the idea of attention mechanism trained two separate classifiers using output of LSTM output, one is to predict the start position of the span while the other is to predict the end position.This is slightly different from the formulation of attention as we don't need to take the weighted sum of all the vector representations. Instead, we use the normalized weights to make direct predictions.

$$P^{(\text{start})}(i) = \frac{\exp\left(\mathbf{p}_i \mathbf{W}^{(\text{start})}\mathbf{q}\right)}{\sum_{i'}\exp\left(\mathbf{p}_{i'}\mathbf{W}^{(\text{start})}\mathbf{q}\right)}$$
$$P^{(\text{end})}(i) = \frac{\exp\left(\mathbf{p}_i \mathbf{W}^{(\text{end})}\mathbf{q}\right)}{\sum_{i'}\exp\left(\mathbf{p}_{i'}\mathbf{W}^{(\text{end})}\mathbf{q}\right)},$$

Separate Classifier for start and end index**(similarity)**

Model Architecture



Model Parameters:

Word Embedding Size(GLOVE): 100d,300d
Vocabulary Size: 10000
Hidden LSTM size: 56,128,256
Num of LSTM layer: 1
Optimizer: adadelta
Learning rate :0.5
Padding:True for example more than max len
Epochs: 30
Max_len: 100
Batch_Size:64

Model Training:

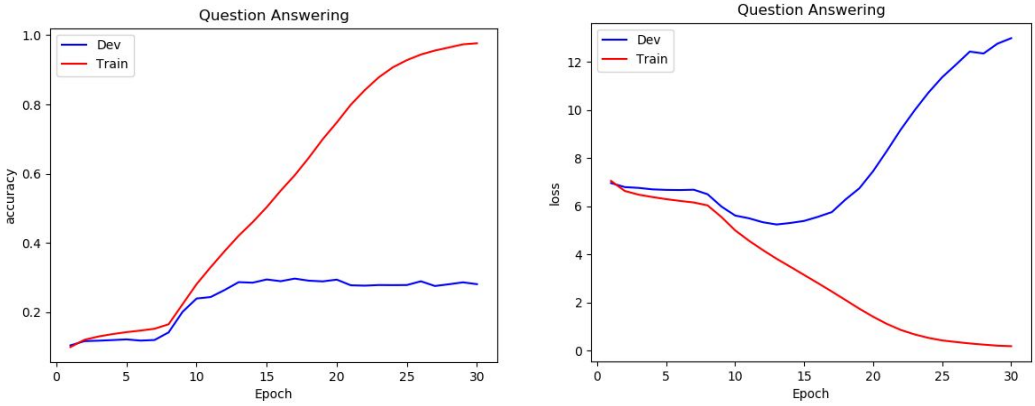
We experiment with embedding layers in 3 settings in embedding layers.

a. Freezing embedding layer parameter with glove
b. Without Freezing embedding layer with glove

Exp Id	Embedding	LSTM	Dev Accuracy	Train Acc	Freeze Embedding
1	100	128	27.74	84.27	False
2	300	128	24.17	94.18	False
3	100	56	26.56	57.63	False
4	300	256	28.07	96.67	False
5	300	256	**27.01**	97.05	True

Accuracy & Loss Curves:

We have plotted train and validation accuracy with respect to no. of epochs for all experiments. Below given curves are for Exp Id 4



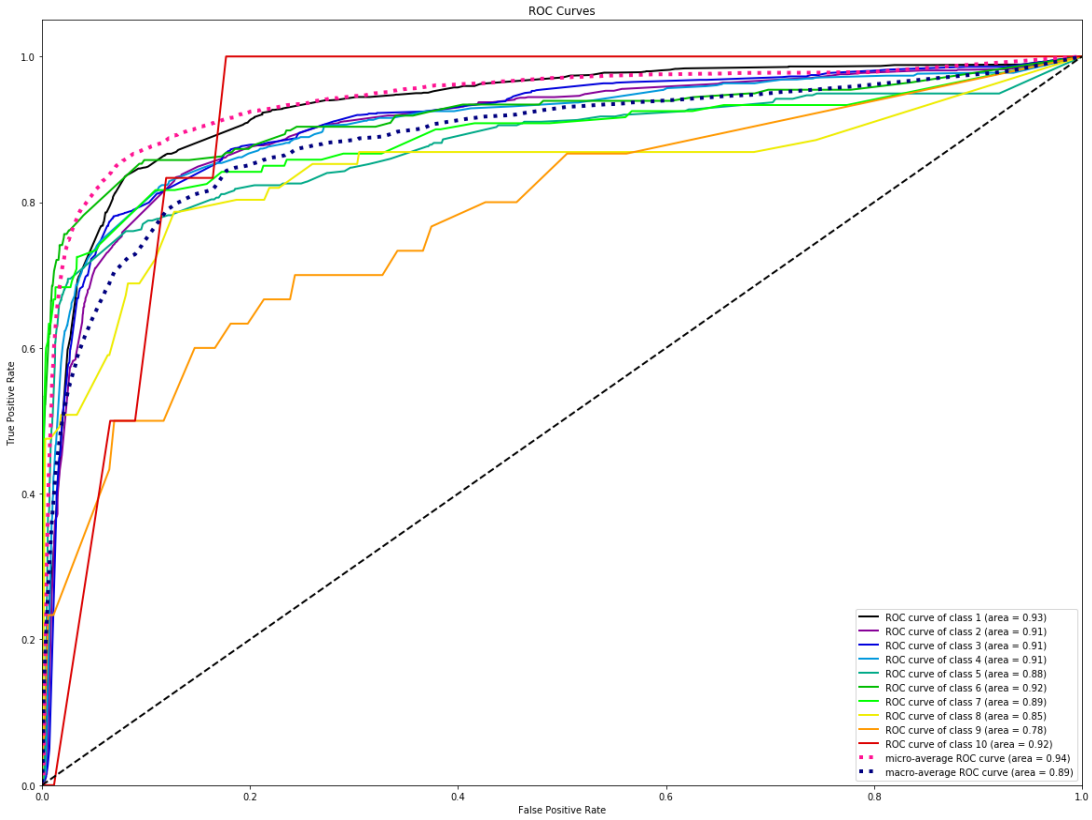
Machine Learning :

Decision Tree:

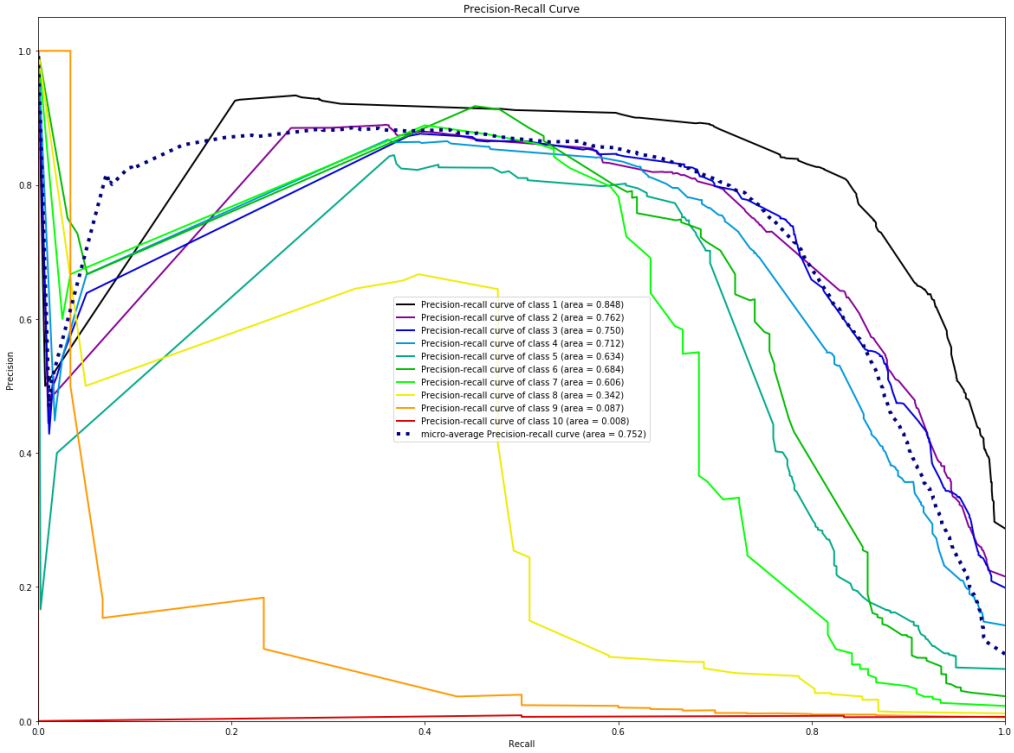
Parameters:
max depth = 10

Training Accuracy : 75.48%
Test Accuracy : 75.42%

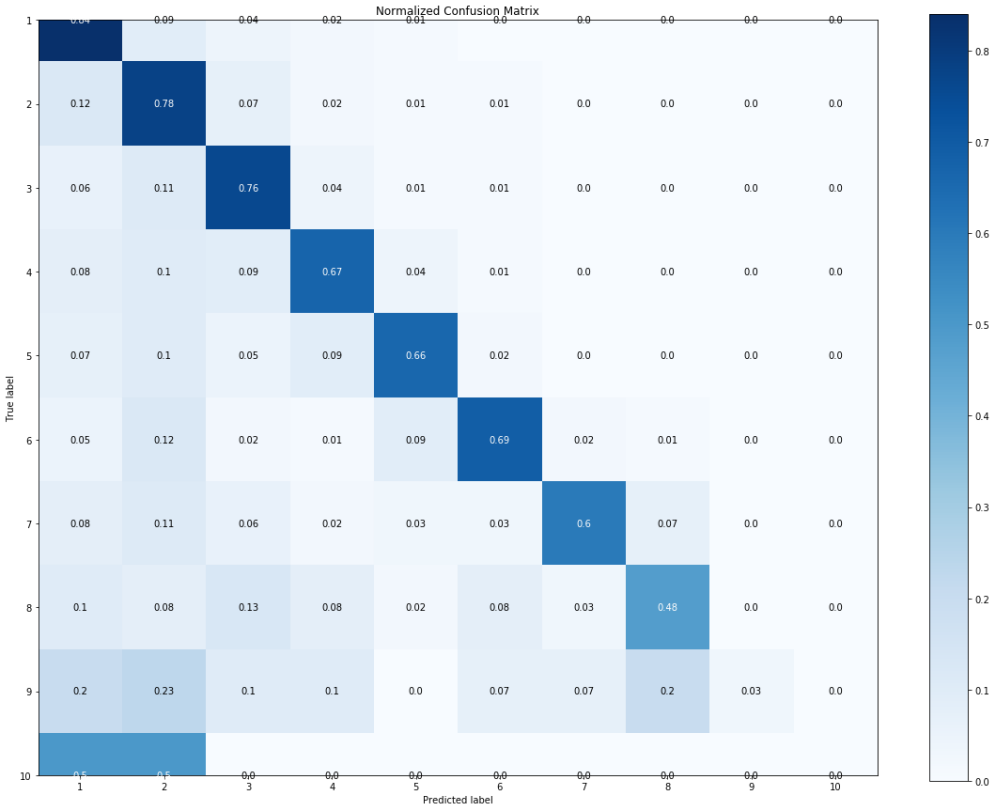
ROC Curve



Precision-Recall Curve



Confusion Matrix



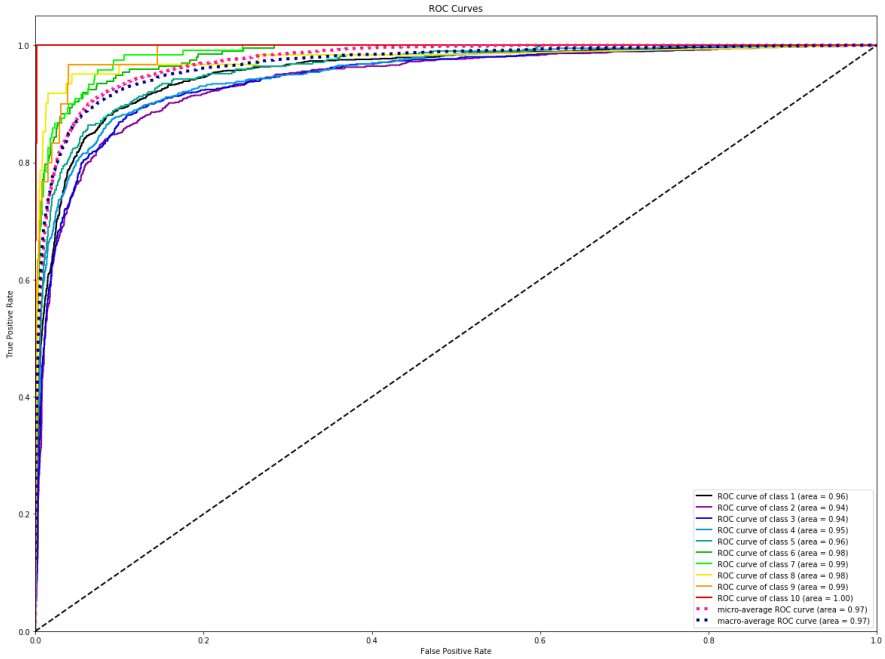
Classification Report

	precision	recall	f1-score
1	0.80	0.84	0.82
2	0.68	0.79	0.73
3	0.76	0.76	0.76
4	0.78	0.67	0.72
5	0.75	0.66	0.71
6	0.73	0.69	0.70
7	0.77	0.60	0.68
8	0.64	0.48	0.55
9	1.00	0.03	0.06
10	0.00	0.00	0.00
accuracy			0.75
macro avg	0.69	0.55	0.57
weighted avg	0.76	0.75	0.75

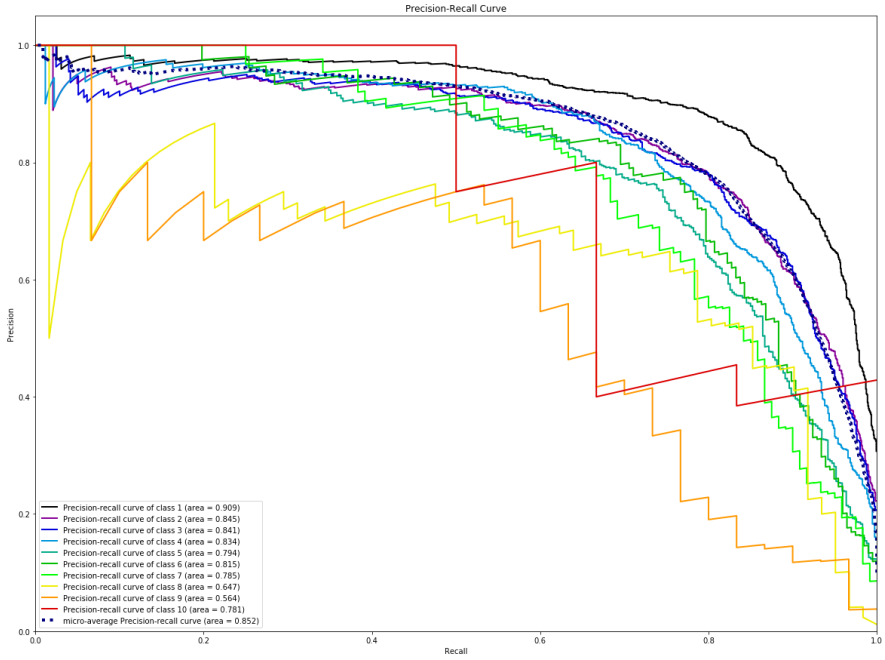
Random Forest Classifier:

Parameters:
min_sample_leaf :8
N_estimators :60
Training Accuracy : 83.18%
Test Accuracy : 78.43%

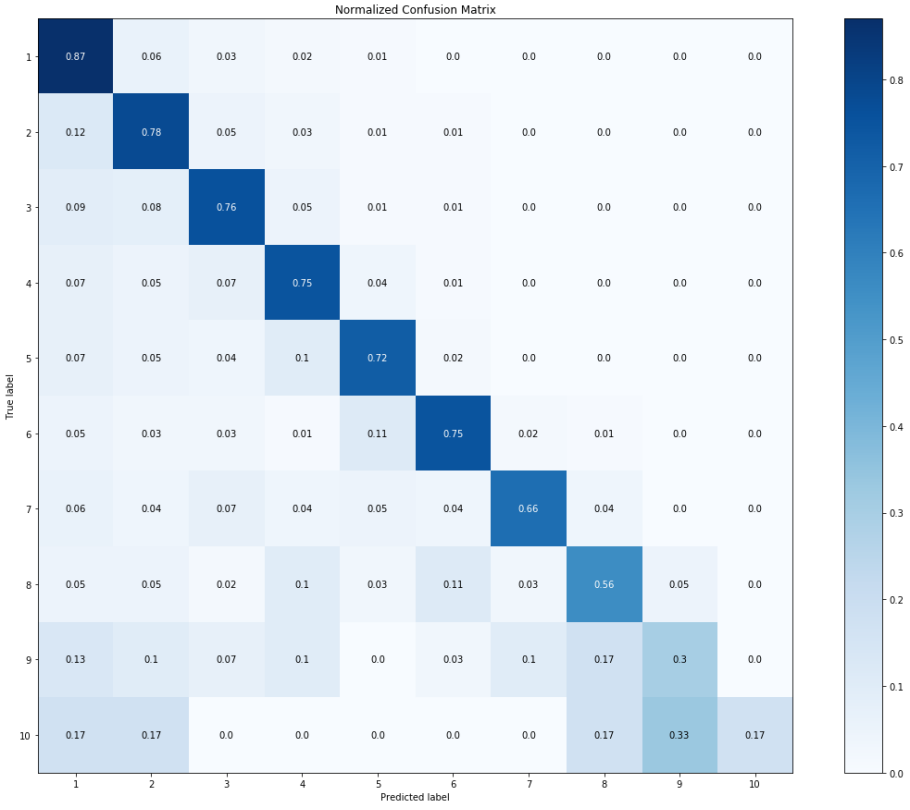
ROC Curve



Precision Recall Curve



Confusion Matrix



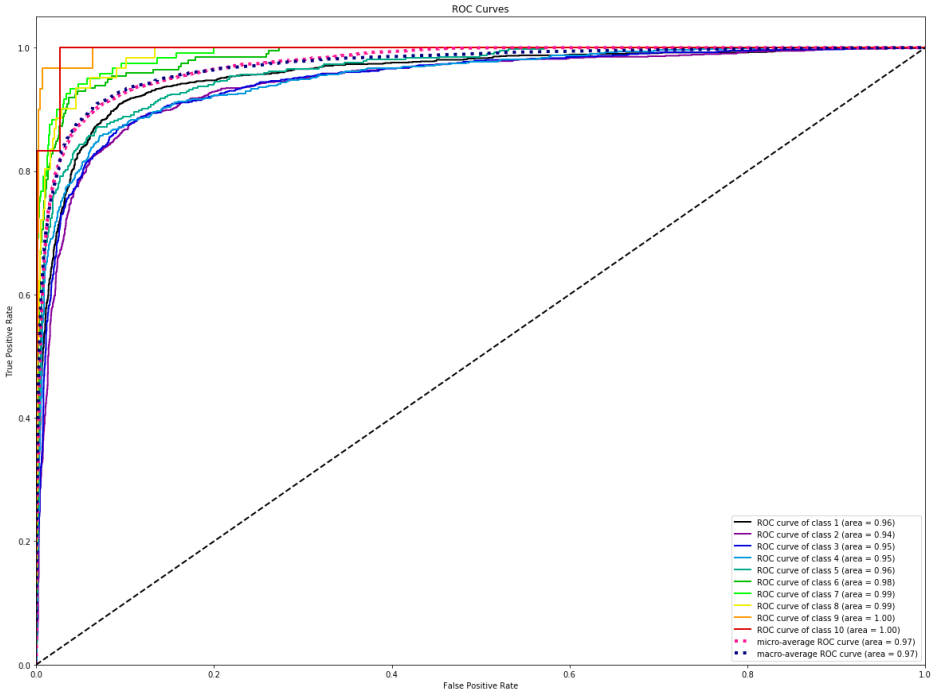
Classification Report

	precision	recall	f1-score
1	0.80	0.87	0.83
2	0.78	0.78	0.78
3	0.81	0.76	0.78
4	0.77	0.75	0.76
5	0.76	0.72	0.74
6	0.74	0.75	0.75
7	0.80	0.66	0.72
8	0.68	0.56	0.61
9	0.64	0.30	0.41
10	1.00	0.17	0.29
accuracy			0.78
macro avg	0.78	0.63	0.67
weighted avg	0.78	0.78	0.78

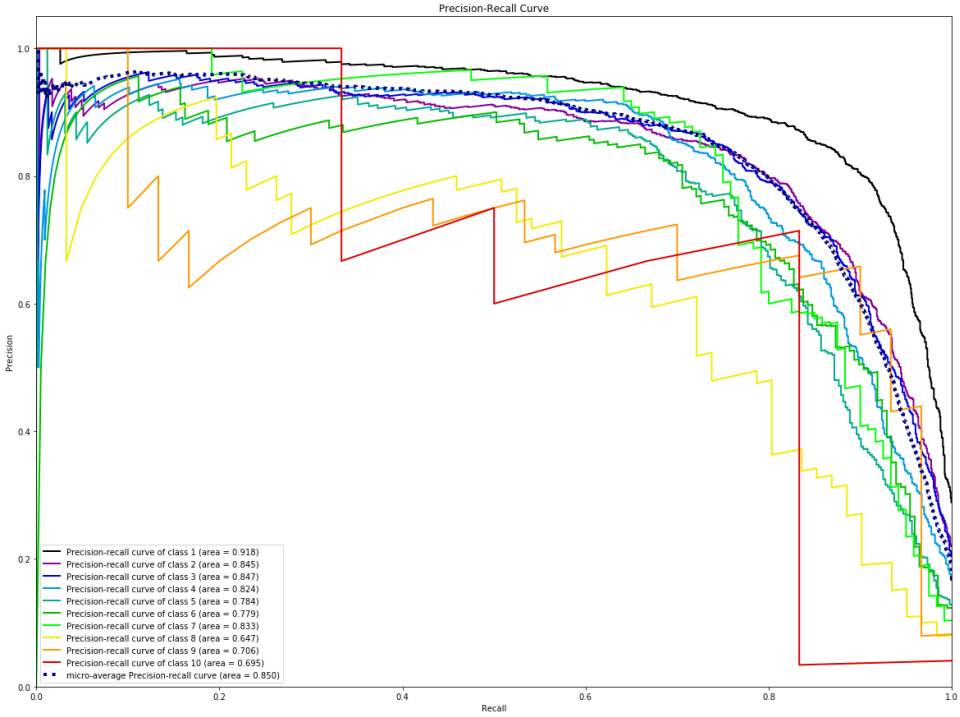
Logistic Regression :

Type:multiclass
Training Accuracy : 78.06%
Test Accuracy : 79.60%

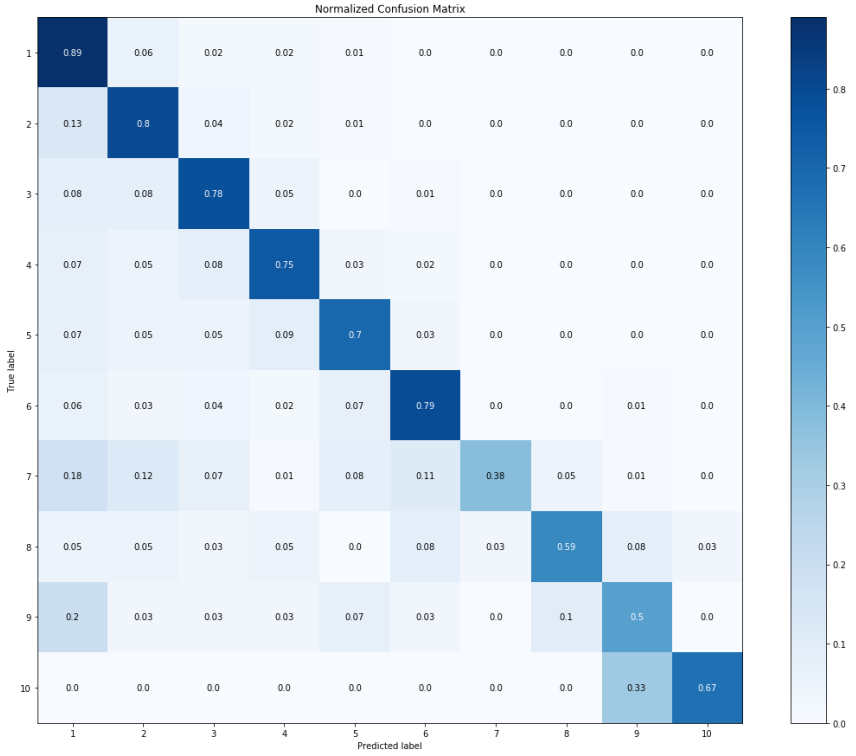
ROC Curve



Precision-Recall Curve



Confusion Matrix

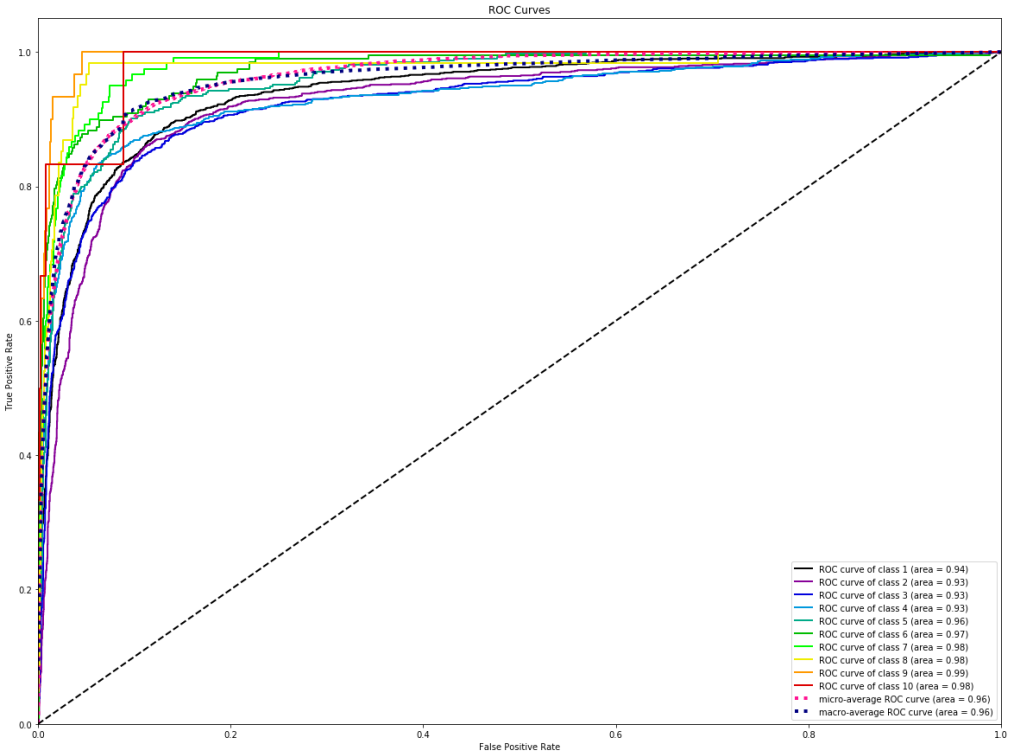


Classification Report

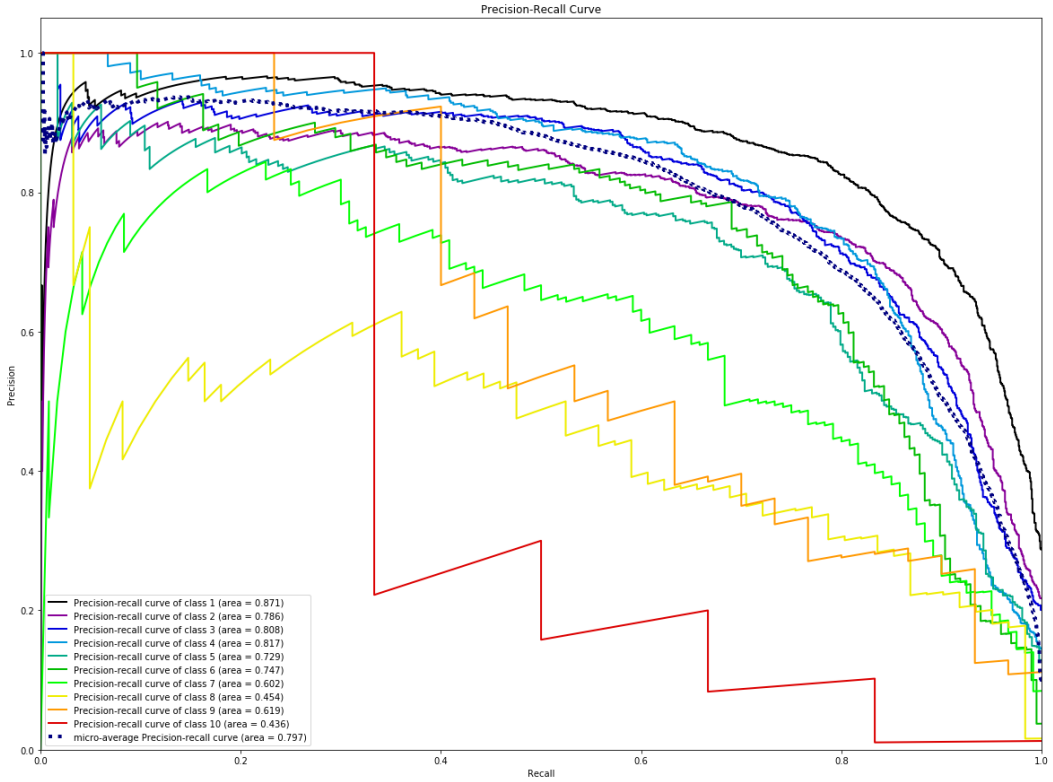
	precision	recall	f1-score
1	0.80	0.89	0.84
2	0.78	0.80	0.79
3	0.82	0.78	0.80
4	0.80	0.75	0.78
5	0.81	0.70	0.75
6	0.71	0.79	0.75
7	0.96	0.38	0.54
8	0.72	0.59	0.65
9	0.62	0.50	0.56
10	0.67	0.67	0.67
accuracy			0.80
macro avg	0.77	0.69	0.71
weighted avg	0.80	0.80	0.79

Support Vector Machine(SVM) :

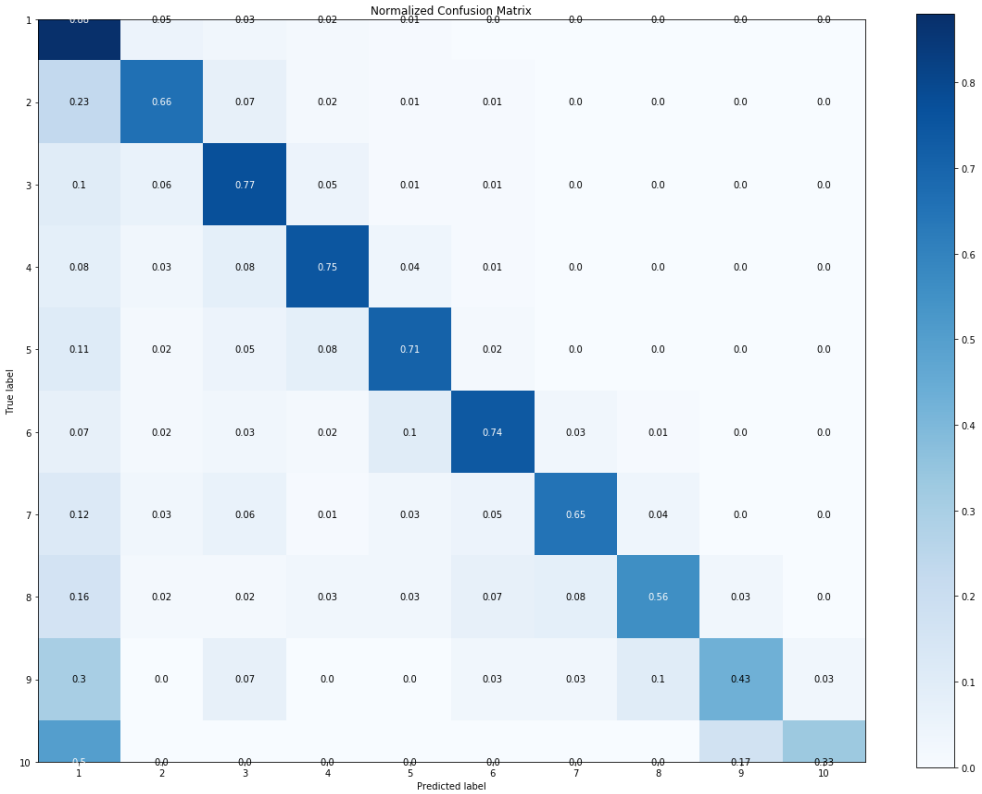
Parameters:
Kernel Type:polynomial of degree 10
Training Accuracy : 77.23%
Test Accuracy : 76.17%
ROC Curve



Precision-Recall Curve



Confusion Matrix



Classification Report

	precision	recall	f1-score
1	0.72	0.88	0.79
2	0.81	0.66	0.73
3	0.78	0.77	0.77
4	0.80	0.75	0.77
5	0.75	0.71	0.73
6	0.75	0.74	0.74
7	0.75	0.65	0.70
8	0.65	0.56	0.60
9	0.81	0.43	0.57
10	0.50	0.33	0.40
accuracy			0.76
macro avg	0.73	0.65	0.68
weighted avg	0.77	0.76	0.76

Gradient Boosting :

Parameters:

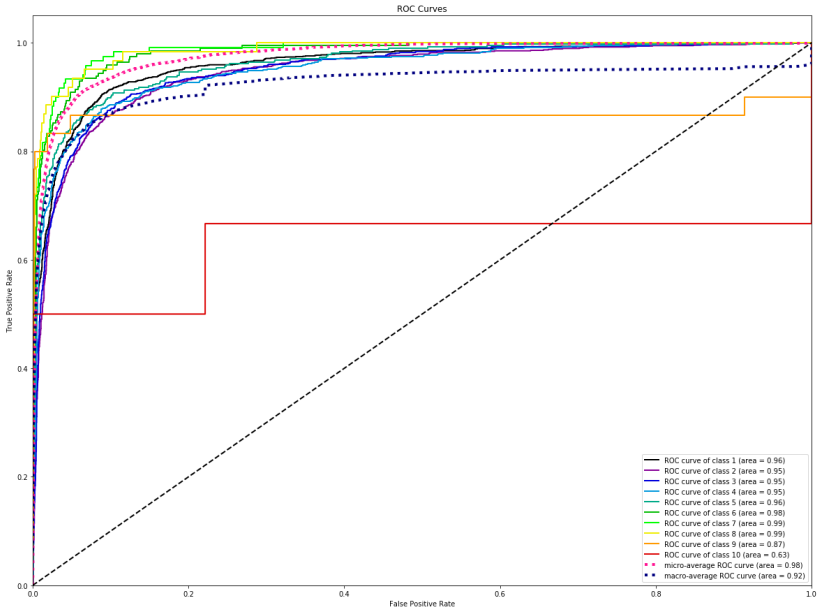
Min_leaves:8

Num. of estimators:60

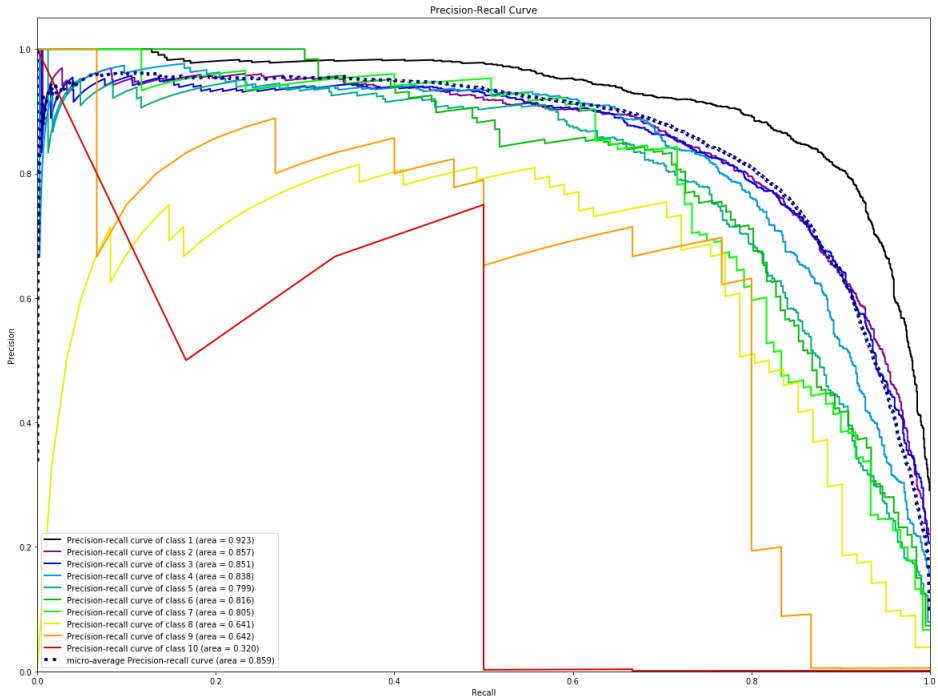
Training Accuracy : 79.77%

Test Accuracy : 78.79%

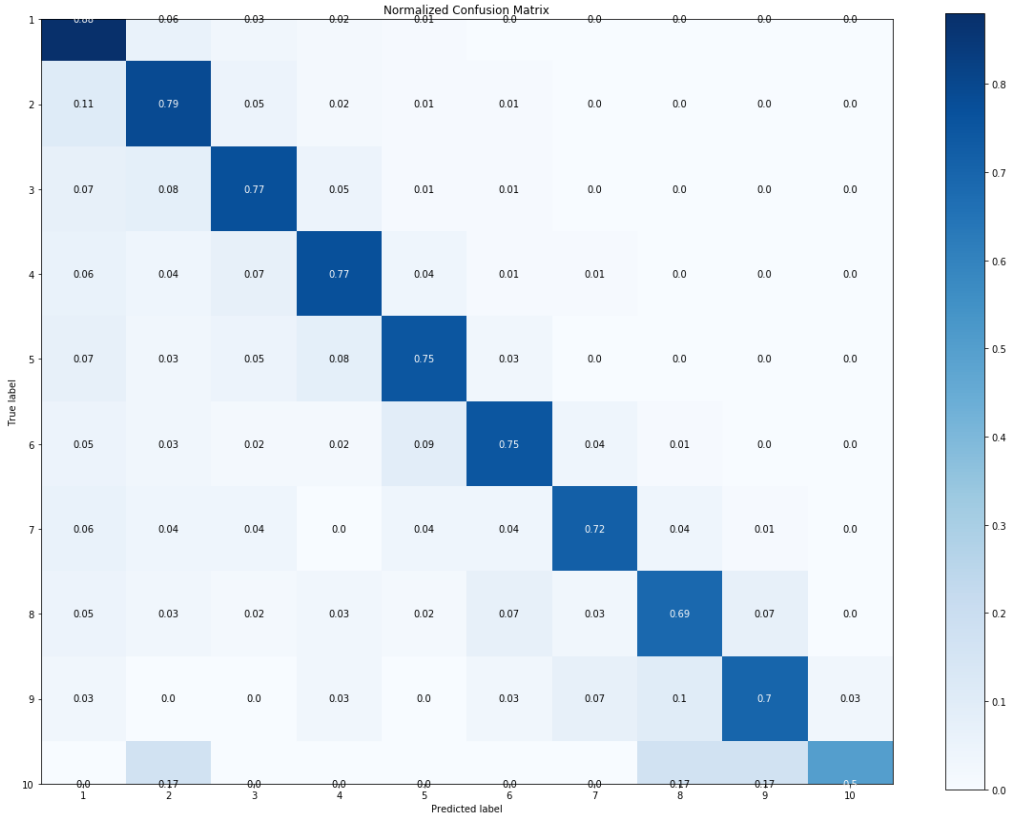
ROC Curve



Precision-Recall Curve



Confusion Matrix

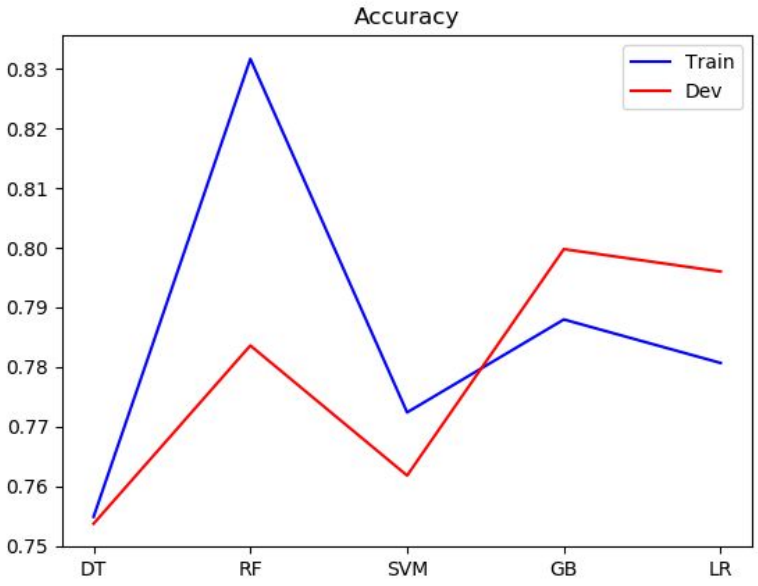


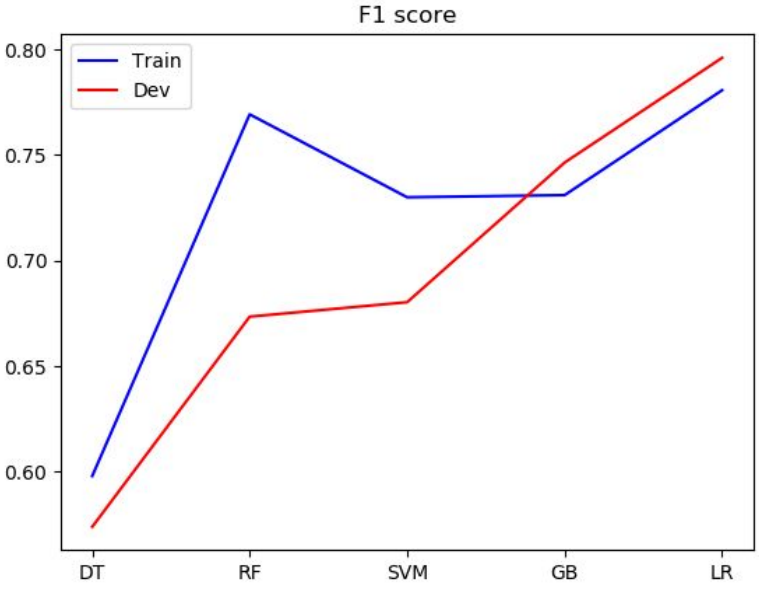
Classification Report

	precision	recall	f1-score
1	0.82	0.88	0.85
2	0.79	0.79	0.79
3	0.81	0.77	0.79
4	0.80	0.77	0.78
5	0.77	0.75	0.76
6	0.75	0.75	0.75
7	0.76	0.72	0.74
8	0.70	0.69	0.69
9	0.72	0.70	0.71
10	0.75	0.50	0.60
accuracy			0.80
macro avg	0.77	0.73	0.75
weighted avg	0.80	0.80	0.80

Result Table

Algorithm	**Train Accuracy(%)**	**Test Accuracy(%)**	**ROC CURVE area**
Decision Tree	75.48	75.42	0.89
Random Forest	**83.18**	78.43	**0.97**
Logistic Regression	78.06	**79.60**	**0.97**
SVM	77.23	76.17	0.96
Gradient Boosting	79.77	78.79	0.92


Accuracy
Train
Dev
0.83
0.82
0.81
0.80
0.79
0.78
0.77
0.76
0.75
DT
RF
SVM
GB
LR


F1 score
Train
Dev
0.80
0.75
0.70
0.65
0.60
DT
RF
SVM
GB
LR

Performance of Algorithms

- [**DL Model**]Our dev accuracy is quite low as compared to train accuracy. It is also much lower than other ML approaches because span prediction is a much harder problem due to the large no. of parameters and complex functions to be learned as compared to sentence index prediction. It is also much slower due to the inherent drawbacks of LSTMs being trained sequentially for each time step. Reduced training data due to pre processing and large no. of parameters led to overfitting of this model. It requires further hyperparameter tuning to reach a better level of accuracy. One notable thing during training was that freezing the embedding layer did not hurt performance which shows pretrained embedding does not require fine tuning for this specific task.
- Our features ml approach is not a common scale. Euclidean distance has no upper bound and varies in accordance with how far the vectors are from each other. Whereas, cosine similarity lies between 0 and 1 because of its inherent property. Therefore, Tree based classifiers work well with our features as they are invariant to feature scales.
- **[DT]** Varying the max depth from 5 to 10, we observed that the underfit decreases as we increase depth to 10. Speculating the train and test accuracy, we can see that it generalizes best among all other algorithms.
- **[RF]** Random forest also works good for our dataset. Although the generalization is not as good as the decision tree, it provides overall a better accuracy for both train and test. If we look at precision and recall scores, we can see that random forest has a better overall precision-recall score.
- **[LR]** One of the most interesting results that we got amidst our various tests of different algorithms is that of Logistic Regression. We observed a test accuracy more than that of training accuracy. Such behavior was not observed with any other algorithms. We speculated that the reason could be because the gradient descent model gets stuck on a local optima during training and that local optima tends to fit the test set better than train set. We were able to infer the reason for this to be the uneven feature scaling. To confirm this, we tried various scaling techniques. It was observed that with feature scaling improved the test and train accuracy. Reported accuracy is by scaling the data between 1st and 3rd quartile. Observed F1 score of the logistic regression is highest among all other algorithms.
- **[SVM]** works good for our extracted features when used with a polynomial kernel of degree 10. It underfits if the degree is less than 10, but it starts to fit the data as we increase the degree towards 10. It starts to overfit if we go beyond 10. SVM takes a very long time to converge for the complete data set for this complex kernel so the reported accuracy is after fixing the number of iterations to be 1million. As can be expected linear svm doesn't work well for our set as it is linearly inseparable.
- **[GB]** As we used gradient regression tree boosting, it works very well for our features. This provides the best accuracy while **maintaining generalization** among all the other algorithms that we tested. It works best for the setting with the number of leaves for each tree as 8 and the number of estimators as 60. With more leaves, it starts to over fit the

data. Interesting enough, gradient descent did not underfit for all the hyper parameter settings tested with our features. We speculate that this is so because gradient boost chooses the best function (h) at each step for the loss function.

Challenges

- One of the most challenging tasks was analysis of data to understand and decide on features that might be useful for our implementation.
- Surveying literature and finalizing deep learning architecture for span prediction tasks.
- Converting a dataset from char offset format to token offset requires testing multiple tokenizers. It needs to be consistent throughout preprocessing for deep learning methods as we need to fix the max length of tokens of input(question & paragraph) for running LSTM.
- Extraction of the features posed a great challenge as producing vector representation of a sentence comes from vector representation of each word. Adding the dimension of word vectors to make sentence vectors hurts the algorithm performance as it might be losing important semantic and syntactic information. Finally we used maxpool operation to solve this.
- As we fixed 10 classes for the sentence prediction task. We need to fill missing features in a paragraph with length less than 10, we have to fill missing values such that every feature has 10 values(one for each sentence in a paragraph). Filling missing values requires multiple experiments .
- Due to time constraints, we were not able to reprocess data with evenly scaled data as the process takes a long time.

Potential Improvements

As preprocessing and architecture selection took most of our limited time share.Having more time in hand, we would have implemented following points

- Replace LSTM layer to transformer based for parallelizing the creation of sentence representation.
- Using subword tokenizer to reduce vocabulary size and no. of parameters in the embedding layer.
- Extra features such as NER tags as most of the question starts with “What” and answers to such questions are named entities.

References

- Know What You Don’t Know: Unanswerable Questions for SQuAD[1]
- Question Answering on SQuAD 2.0 Dataset[2]
- SQuAD: 100,000+ Questions for Machine Comprehension of Text[3]

[1] "Know What You Don't Know: Unanswerable Questions for ...." 11 Jun. 2018, [1806.03822] Know What You Don't Know: Unanswerable Questions for SQuAD. Accessed 3 May. 2020.
[2] "Question Answering on SQuAD 2.0 Dataset - Stanford University." Question Answering on SQuAD 2.0 Dataset. Accessed 3 May. 2020.
[3] "SQuAD: 100000+ Questions for Machine Comprehension of ...." [1606.05250] SQuAD: 100,000+ Questions for Machine Comprehension of Text. Accessed 3 May. 2020.