COMP-551: Applied Machine Learning
Assignment #3
Mandana Samiei
Student ID: 260779555

Question 1

As a first a part of the assignment we were supposed to convert each dataset to two kinds of representation, binary bag of words and frequency bag of words. First, we needed to split data as review and target, and do some preprocessing on the reviews.

```
def split_data(data):
    reviews, targets = [], []
    splitted_txt = data.rstrip('\n').split('\n')
    for i in range(len(splitted_txt)):
        line = splitted_txt[i].split('\t')
        reviews.append(line[0])
        targets.append(line[1])
    return reviews, targets
```

```
def pre_process(data):
    rvw = []
    for x in data:
        rvw.append(x.translate(None, string.punctuation).lower())
    return rvw
```

```
def get_vocab(data):
    count = {}
    for x in data:
        rvw = x.split()
        for w in rvw:
            if w in count:
                count[w] +=1;
            else:
                count[w] = 1;
    v = sorted(count.items(), key=operator.itemgetter(1), reverse=True) # list of vocabularies
    return v
```

After splitting data, preprocessing and creating vocabulary, we need to do binary bag of words and frequency bag of words:

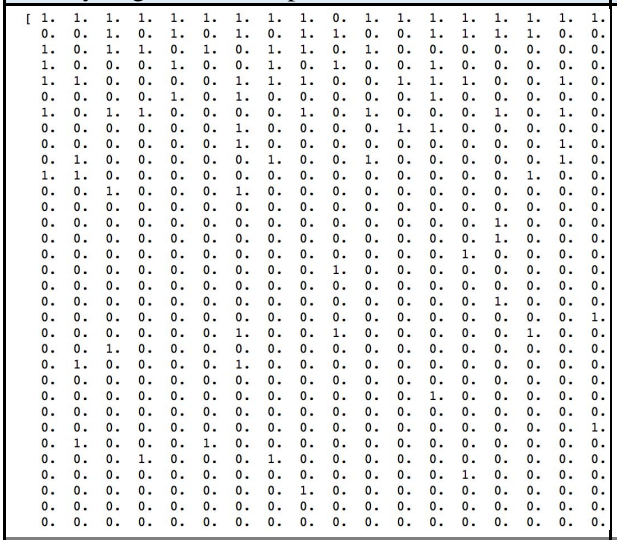
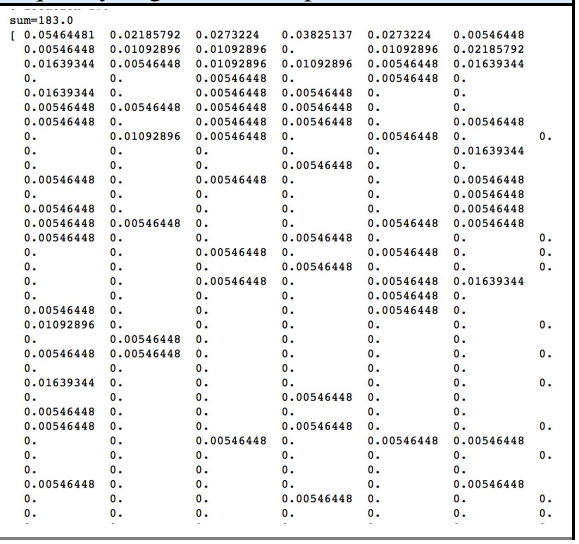
Binary bag-of-words representation

```
def bin_bow(rvw, f_set):
    new_dataset = np.zeros((len(rvw),len(f_set)))
    for i in range(len(rvw)):
        words = rvw[i].split()
        for w in words:
            try:
                new_dataset[i][f_set.index(w)] = 1;
            except ValueError:
                j=0
    return new_dataset
```

Frequency bag-of-words representation

```
def frq_bow(rvw, f_set):
    new_dataset = np.zeros((len(rvw),len(f_set)))
    fbow = np.zeros((len(rvw),len(f_set)))
    for i in range(len(rvw)):
        words = rvw[i].split()
        for w in words:
            try:
                new_dataset[i][f_set.index(w)] += 1.0;
            except ValueError:
                j=0
        total = sum(new_dataset[i])
        if total != 0: #in case that no words exist in a review, review number #3615 is empty and causes NaN
            fbow[i] = new_dataset[i][:]/float(total)
        else:
            fbow[i] = new_dataset[i][:]
    return fbow
```

Also, here you can see a part of these two representation on the train data of yelp dataset:

Binary bag of words representation	Frequency bag of words representation
	

Question 2- I used 'Macro' f1 because I want to do equally well on all classes. For each classifier I mentioned their hyperparameters, testing range and the best choice. I found the best range through different experiments, for example first I tried an initial range, then searched for the best value, when the best value was found, I limited the range around the best value from previous step. Besides hyperparameters, there are some parameters in each classifier that their variation affect the performance but their effect is not as much as hyperparameters. However, I also changed the value of those parameters to see how much improvement they may result. Here is the ordered list of hyperparameters for each classifier based on their importance on the f1 improvement.

Bernoulli Naive Bayes:

- Alpha

Gaussian Naive Bayes:

- There is no hyperparameter

SVM:

- C (Penalty Parameter)
- Loss function
- Tolerance

Decision Tree:

- Max_depth
- max_leaf_nodes
- min_sample_per_leaf
- criterion

The performance of random and majority-class classifier:

for majority-class classifier, we just predict class 4 for all examples, because class 4 has the most samples per classes.

Confusion Matrix of Random Classifier:	Confusion Matrix of Majority-class Classifier:
[[26 24 24 36 33] [50 38 43 35 24] [59 62 57 65 57] [145 156 130 142 129] [125 138 130 131 141]]	[[0 0 0 143 0] [0 0 0 190 0] [0 0 0 300 0] [0 0 0 702 0] [0 0 0 665 0]]

F1 measure of Random Classifier: 0.182202038978	F1 measure of Majority-class Classifier: 0.103923019985

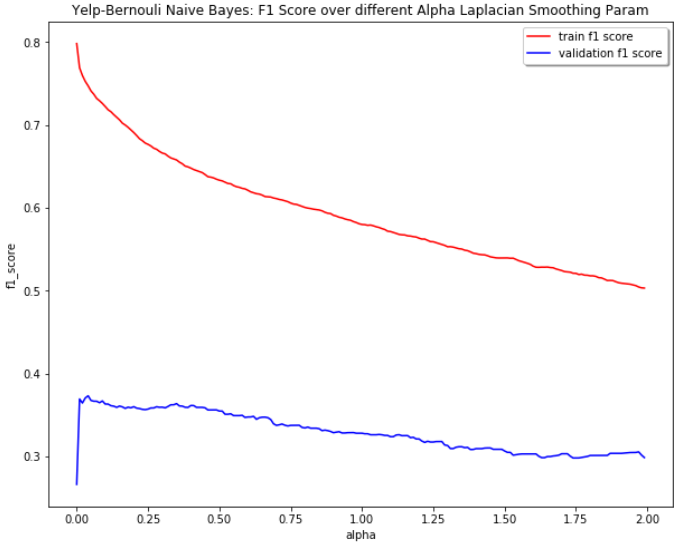
We expected a f1 score almost 20% for random classifier because yelp dataset has 5 classes and by randomness rule each class will get ⅕ of all occurance.
Random classifier result is better than Majority since the portion of class 4 of all data is less than ⅕.

Yelp: Binary Bag of Words: Bernoulli Naive Bayes Performance
In the bernoulli naive bayes, I considered alpha (Laplacian Smoothing parameter) as a hyperparameter:

Yelp:Binary bagging-Bernoulli Naive Bayes	**F1-measure**	**Best Alpha for each dataset**	**Range**
Train set	0.798360411795	0.0	np.arange(0,2,0.01)
Validation set	0.373354042012	0.04	np.arange(0,2,0.01)

According to the above table, in range 0 to 2 step by 0.01, I got the best validation set f1 measure with alpha = 0.04 so I calculated train and test f1 measure with alpha = 0.04:

Yelp: Binary bagging-Bernoulli Naive Bayes	**F1-measure**	**Best hyper-parameter configuration**
Train set	0.747396116481	0.04
Validation set	0.373354042012	0.04
Test set	0.385221910535	0.04



According to the diagram, as the value of alpha increases, the f1_score decreases. Alpha is laplacian smoothing parameter, and based on the data distribution of each class, its optimum value is the prior probability of that class.

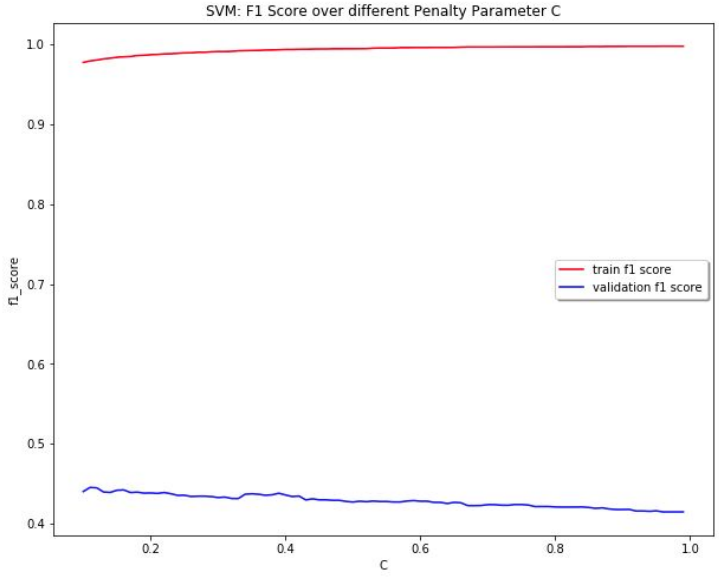
Yelp: Binary Bag of Words: Linear SVM Performance
For linear SVM, I considered C (penalty parameter) as a hyperparameter. There is many variable in the function which we can change them such as tolerance and loss function, I changed them as well and

observed the result, the improvement with the variation of these 2 variables is about 0.02 and it's not that much significant so I just reported variations of C. Also, I set dual to True since the number of feature in the Yelp dataset is more than number of examples.

Yelp: Binary bagging-Linear SVM	**F1-measure**	**Best C for each dataset**	**Dual**	**Range**
Train set	0.986925637104	0.195	True	np.arange(0.1, 1, 0.01)
Validation set	0.453039857496	0.01	True	np.arange(0.1, 1, 0.01)

I got the best validation set f1 measure with C = 0.01, in the following table I reported the f1 score for all datasets with best hyper-parameter configuration:

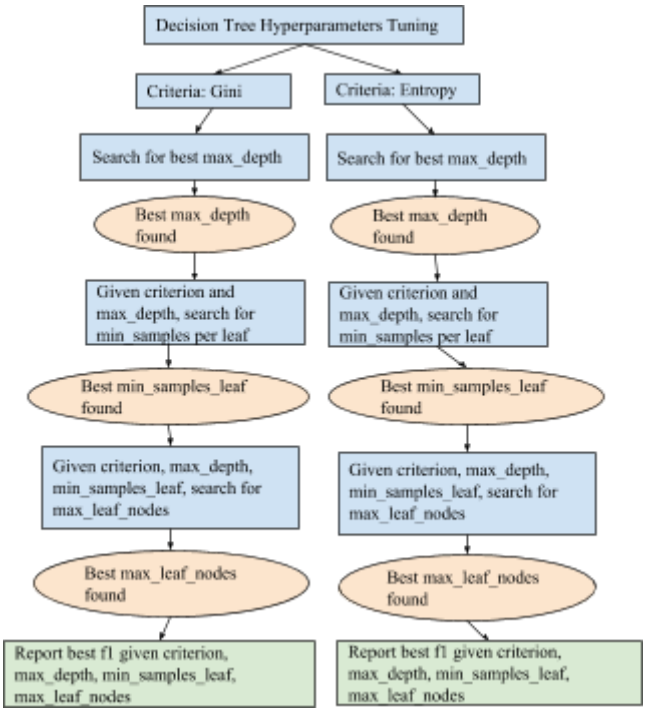
Yelp:Binary bagging- Linear SVM	**F1-measure**	**Best hyper-parameter configuration**	**Dual**	**Range**
Train set	0.853803311465	0.01	True	np.arange(0.1, 1, 0.01)
Validation set	0.453039857496	0.01	True	np.arange(0.1, 1, 0.01)
Test set	0.461663324968	0.01	True	np.arange(0.1, 1, 0.01)



Based on the above figure, in validation set as the value of C increases from range 0.1 to 1, f1- measure decreases. C shows how sensitive we are to wrong classifications, as we become more sensitive, f1 score get lower.

Yelp: Binary Bag of Words: Decision Tree Performance

In the decision tree, we can use Gini or Entropy as a criteria, So in this part I did f1 performance evaluation once for Gini and second for Entropy. Here is the hierarchy of my strategy for hyperparameter tuning:



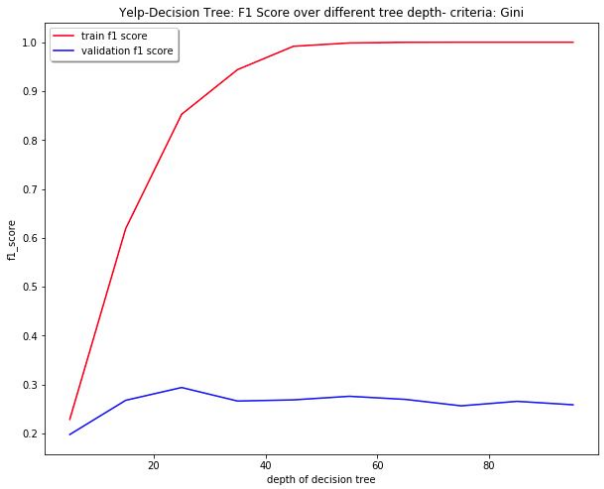
As a one example, I'll show the process of reaching to the best f1 score in decision tree in binary bagging representation of yelp but in the other case I'll show just the conditions of best f1 score(Not the procedure):

a-1) Criterion: Gini, search for max_depth:

Yelp: Binary bagging Decision Tree	**F1-measure**	**criterion**	**Best depth for each dataset**	**Range**
Train set	1.0 (overfitted)	Gini	75	np.arange(5,100,10)
Validation set	0.293787133684	Gini	25	np.arange(5,100,10)

I picked max_depth=25 as the best configuration

Yelp:Binary bagging-Decision Tree	**F1-measure**	**Best hyper-parameter configuration**	**Range**
Train set	0.851942484288	25	np.arange(5,100,10)
Validation set	0.445317421575	25	np.arange(5,100,10)
Test set	0.27427100026	25	np.arange(5,100,10)



According to the above figure as the depth of tree increases, training set reaches to a higher f1 measure while validation set stays in an almost same range.
max_depth = 25 has the best result for validation set.

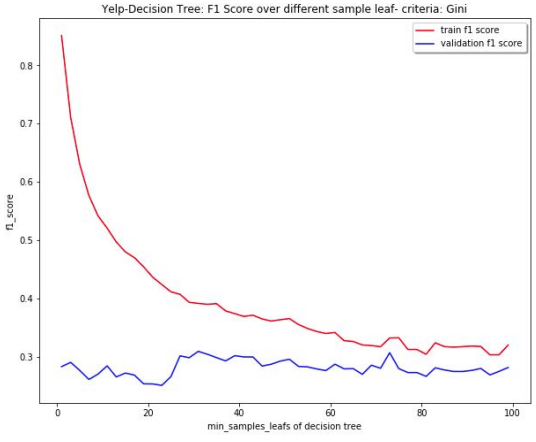
a-2) Criterion: Gini, max_depth=25, search for the best min_samples_leaf

In this part, given criterion, max_depth, I searched for the best min_samples_leaf in the range (1,100)

Yelp: Binary bagging Decision Tree	**F1-measure**	**criterion**	**Best min_samples_leafs for each dataset**	**Range**
Train set	0.850865792078	Gini	1	np.arange(1,100,2)
Validation set	0.309882851801	Gini	31	np.arange(1,100,2)

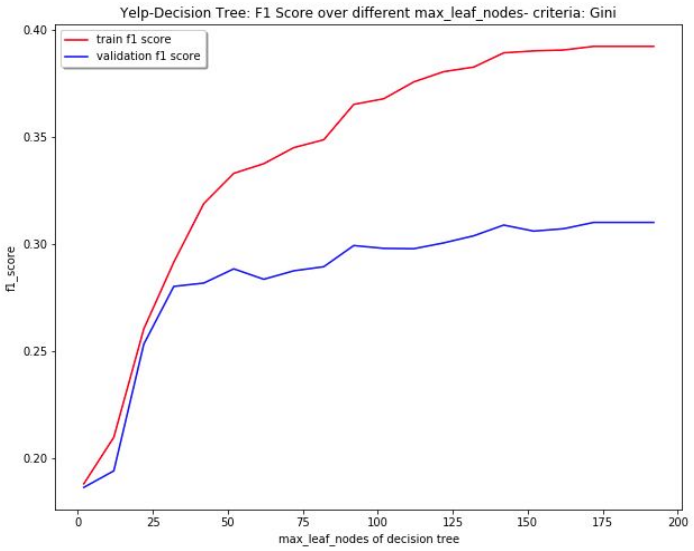
I picked min_Samples_leafs = 31 as the best configuration based on the validation set performance:

Yelp:Binary bagging- Decision Tree	**F1-measure**	**Criterion**	**max_depth**	**Best hyper-parameter configuration**	**Range**
Train set	0.392058433193	Gini	25	31	np.arange(1,100,2)
Validation set	0.309882851801	Gini	25	31	np.arange(1,100,2)
Test set	0.295984356499	Gini	25	31	np.arange(1,100,2)



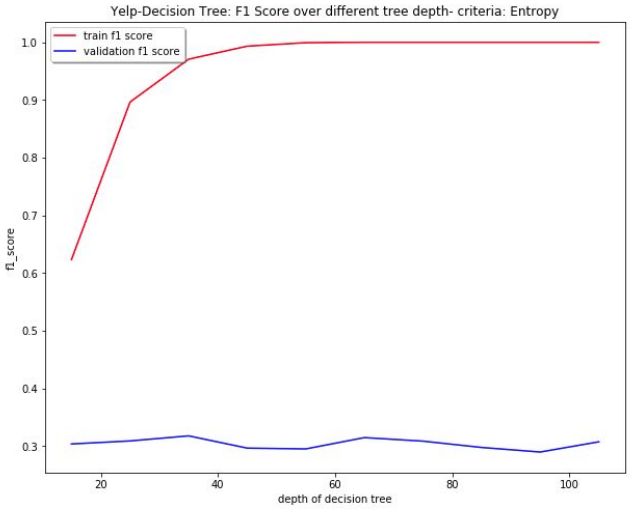
a-3) Criterion: Gini, max_depth=25, min_samples_leaf =31, search for best max_leaf_nodes
Here is my final f1 score for training, validation and test set as well as the best value for each parameter:

Yelp: Binary bagging Decision Tree	F1-measure	criterion	max_ depth	min_sample_ leafs	Best max_leaf_nodes for each dataset	Range
Train set	0.392058433193	Gini	25	31	172	np.arange(2,200,10)
Validation set	0.309882851801	Gini	25	31	172	np.arange(2,200,10)
Test set	0.295984356499	Gini	25	31	172	np.arange(2,200,10)



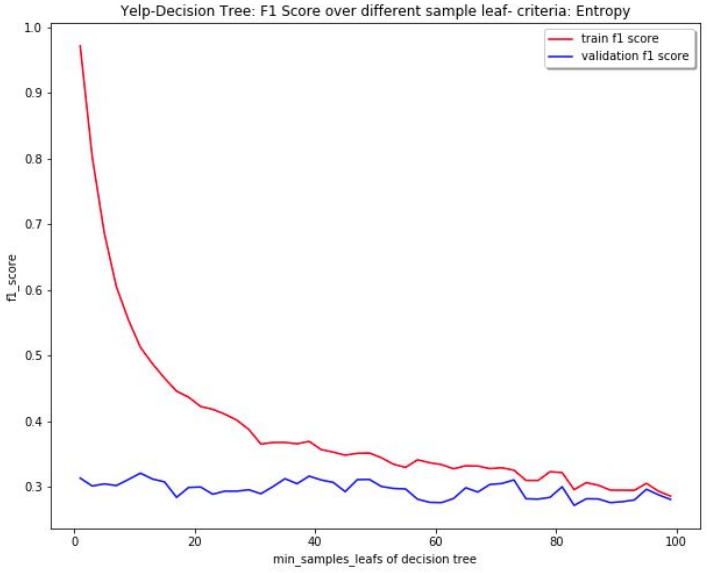
b-1) Criterion: Entropy, search for max_depth:
I picked max_depth=35 as the best configuration based on validation set:

Yelp:Binary bagging- Decision Tree	F1-measure	Criterion	Best hyper-parameter configuration	Range
Train set	0.969632868365	Entropy	35	np.arange(5,100,10)
Validation set	0.31767538649	Entropy	35	np.arange(5,100,10)
Test set	0.289533247147	Entropy	35	np.arange(5,100,10)



b-2) Criterion: Entropy, max_depth=35, search for the best min_sample_leaf

Yelp:Binary bagging-Decision Tree	F1-measure	Criterion	max_depth	Best hyper-parameter configuration	Range
Train set	0.512099859955	Entropy	35	11	np.arange(5,100,10)
Validation set	0.31767538649	Entropy	35	11	np.arange(5,100,10)
Test set	0.276521834892	Entropy	35	11	np.arange(5,100,10)

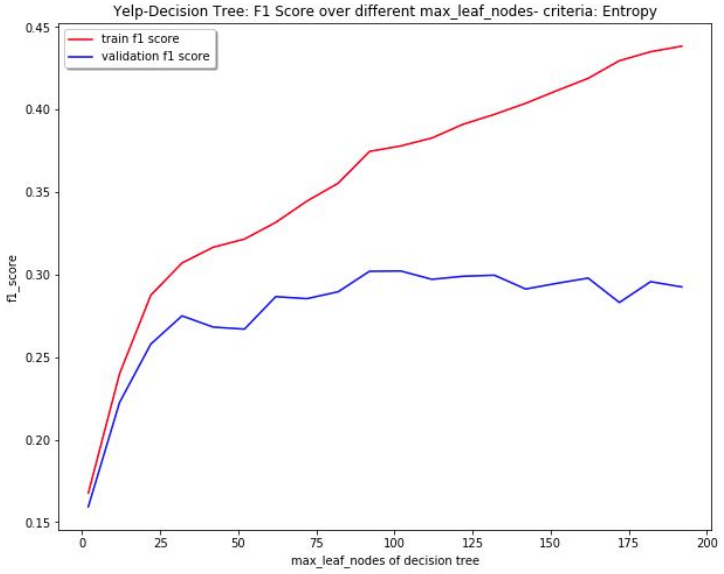


b-3) Criterion: Entropy, max_depth=35, min_sample_leaf=11, search for max_leaf_nodes

I picked max_leaf_nodes=102 as the best configuration since validation set has the maximum f1 score with this value of max_leaf_nodes param.

Yelp:Binary bagging-Decision Tree	F1-measure	Criterion	max_ depth	min_sample _leaf	max_leaf_nodes	Range
Train set	0.433610891187	Entropy	35	11	102	np.arange(2,200,10)

Validation set	0.302196408027	Entropy	35	11	102	np.arange(2,200,10)
Test set	0.275910008696	Entropy	35	11	102	np.arange(2,200,10)



Performance of different classifiers in binary bag of words representation:

Classifier	**Test f1_ score**
Bernoulli Naive Bayes	0.385221910535
SVM	0.461663324968
Decision Tree	0.295984356499

Comparison: SVM > Naive Bayes > Decision Tree
As we can see, SVM classifier works better among all of these classifiers and Decision tree is the worst. Basically, Naive bayes is a generative model and it looks at the distribution of data while SVM tries to maximize the distance between data points of different classes and decision tree tries to maximize information gain based on the separability of the data points. In this case, probably the data is not linearly separable and it could be the reason of low performance in decision tree. Furthermore, yelp has 5 different classes that make the classification process a way more difficult.

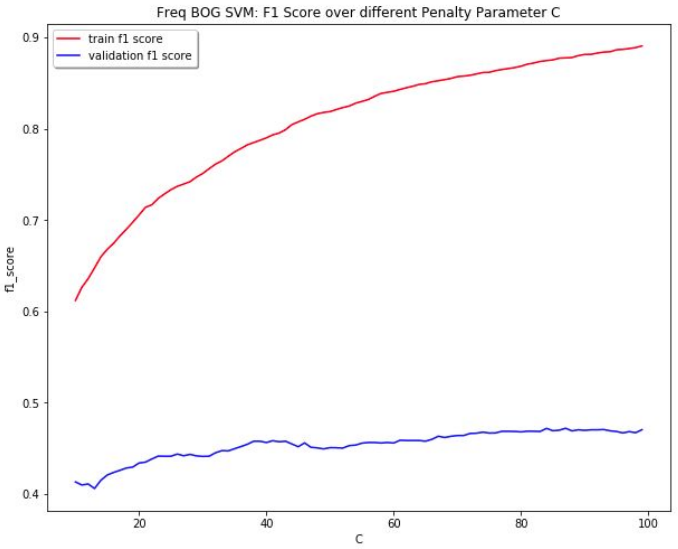
Question 3 - Yelp - Frequency Bag of words

There is no hyper parameter for Gaussian Naive Bayes. I used macro F1 as a performance evaluation.
Yelp: Frequency Bag of Words: Gaussian Naive Bayes Performance

Yelp- Frequency bagging: Gaussian Naive Bayes	**F1 score**
Training Set	0.785827048797
Validation Set	0.244660263781
Test Set	0.254895792069

Yelp: Frequency Bag of Words: SVM

Yelp: Frequency bagging- Linear SVM	F1-measure	Best hyper-parameter configuration	Dual	Range
Train set	0.877231072306	87	True	np.arange(10, 100, 1)
Validation set	0.472091184934	87	True	np.arange(10, 100, 1)
Test set	0.461221145438	87	True	np.arange(10, 100, 1)



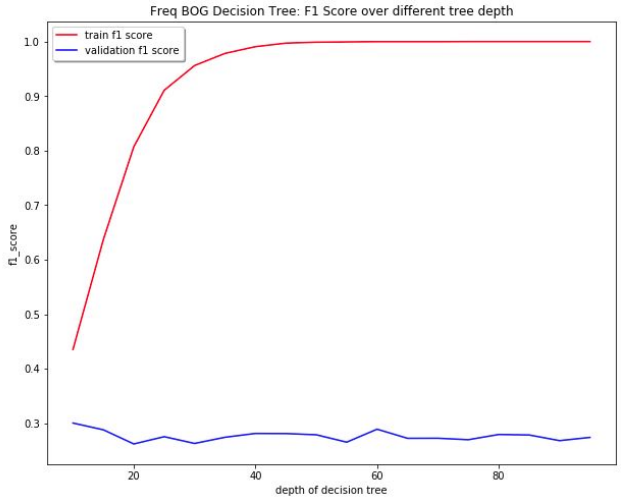
According to the above figure and my experiments, SVM in frequency bag of words need a higher value of C to have an almost same result to binary bag of words. The reason is, in frequency bagging, data are very small and all of them are close to zero so the chance of misclassification is more than binary bagging, SVM try to penalized the misclassification by increasing C.

Yelp: Frequency Bag of Words: Decision Tree

I used the same hierarchical strategy as binary bag of words to determine f1 score.

a-1) Criterion: Gini, search for max_depth:

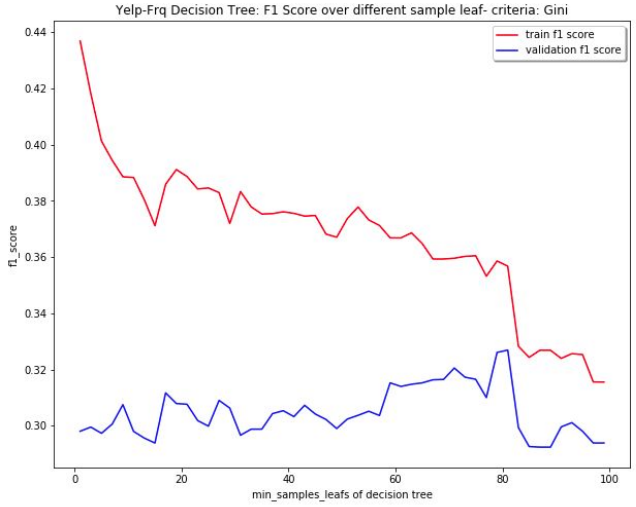
Yelp:Frequency bagging- Decision Tree	F1-measure	Best hyper-parameter configuration	Range
Train set	0.43537430174	10	np.arange(5,100,10)
Validation set	0.445317421575	10	np.arange(5,100,10)
Test set	0.267768590963	10	np.arange(5,100,10)



a-2) Criterion: Gini, max_depth: 10, search for the best min_samples_leaf

In this part, given criterion, max_depth, I searched for the best min_samples_leaf in the range (1,100)
I picked min_Samples_leafs = 81 as the best configuration as the validation set has the most f1 score:

Yelp:Frequency bagging-Decision Tree	**F1-measure**	**Criterion**	**max_depth**	**Best hyper-parameter configuration**	**Range**
Train set	0.356726062557	Gini	10	81	np.arange(1,100,2)
Validation set	0.326885557701	Gini	10	81	np.arange(1,100,2)
Test set	0.285983288876	Gini	10	81	np.arange(1,100,2)

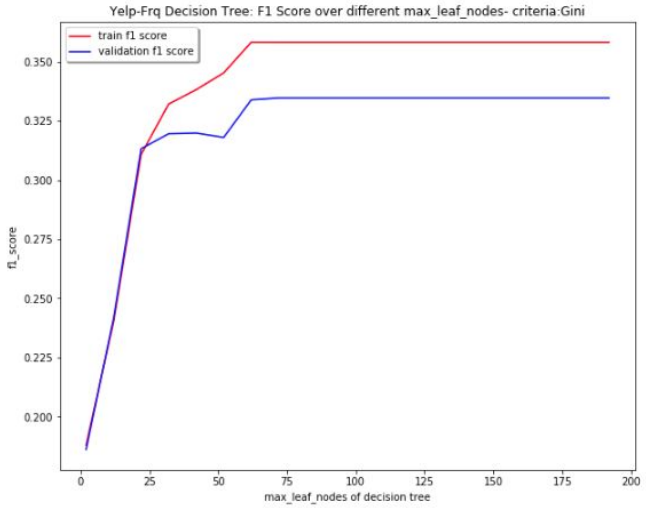


a-3) Criterion: Gini, max_depth=10, min_samples_leaf =81 search for best max_leaf_nodes

Here is my final f1 score for training, validation and test:

Yelp: Frequency bagging Decision Tree	**F1-measure**	**criterion**	**max_ depth**	**min_sample_ leafs**	**Best max_leaf_nodes for each dataset**	**Range**

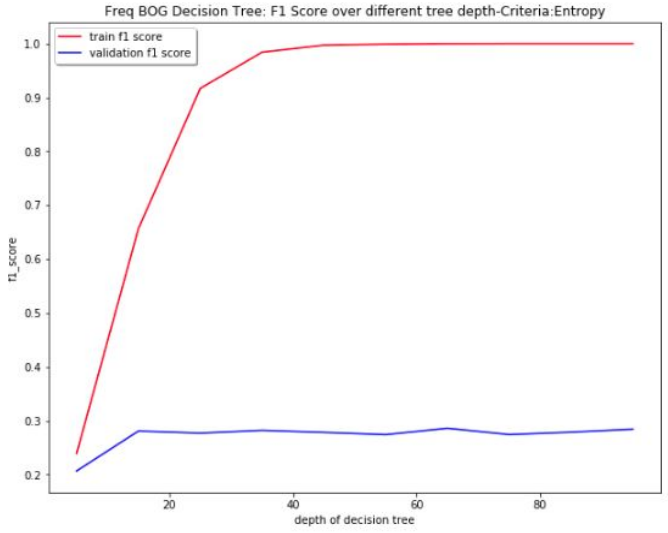
Train set	0.358194634116	Gini	10	81	72	np.arange(2,200,10)
Validation set	0.334681714619	Gini	10	81	72	np.arange(2,200,10)
Test set	0.287384219859	Gini	10	81	172	np.arange(2,200,10)



b-1) Criterion: Entropy, search for the best max depth

I picked max_depth=65 as the best configuration based on the performance of validation set:

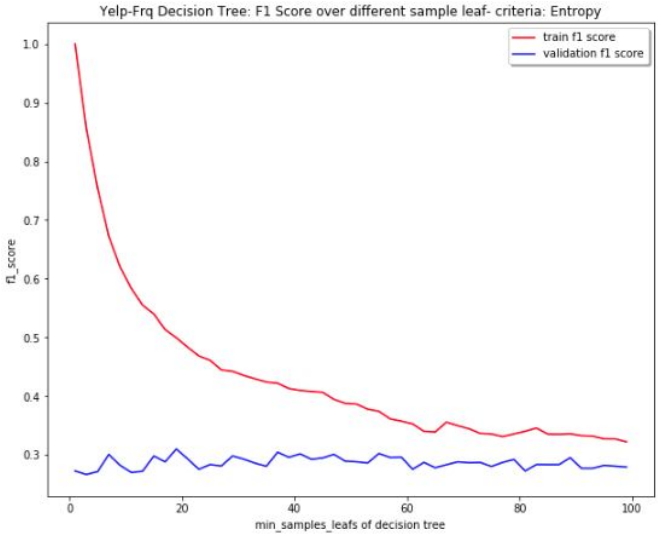
Yelp:Frequency bagging- Decision Tree	**F1-measure**	**Criterion**	**Best hyper-parameter configuration**	**Range**
Train set	0.999766095815	Entropy	65	np.arange(5,100,10)
Validation set	0.285982945	Entropy	65	np.arange(5,100,10)
Test set	0.301008561307	Entropy	65	np.arange(5,100,10)



b-2) Criterion: Entropy, max_depth=65, search for the best min_sample_leaf

I picked min_sample_leaf =19 as the best configuration based on the performance of validation set:

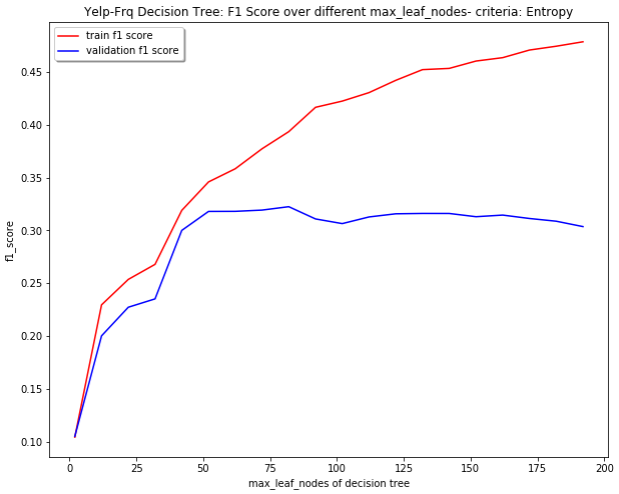
Yelp:Frequency bagging-Decision Tree	F1-measure	Criterion	max_depth	Best hyper-parameter configuration	Range
Train set	0.499598115023	Entropy	65	19	np.arange(1,100,2)
Validation set	0.310286351813	Entropy	65	19	np.arange(1,100,2)
Test set	0.299815850055	Entropy	65	19	np.arange(1,100,2)



b-3) Criterion: Entropy, max_depth=65, min_sample_leaf=19, search for max_leaf_nodes

I picked max_leaf_nodes=82 as the best configuration since validation set has the maximum f1 score with this value of max_leaf_nodes.

Yelp:Frequency Bagging-Decision Tree	F1-measure	Criterion	max_ depth	min_sample _leaf	max_leaf_nodes	Range
Train set	0.393509869069	Entropy	65	19	82	np.arange(2,200,10)
Validation set	0.32245995998	Entropy	65	19	82	np.arange(2,200,10)
Test set	0.298799903353	Entropy	65	19	82	np.arange(2,200,10)



Performance of different classifiers in frequency bag of words representation:

Classifier	Binary Bag of words, Test f1_score	Frequency Bag of words, Test f1_ score
Naive Bayes	0.385221910535 (Bernoulli)	0.254895792069 (Gaussian)
Decision Tree	0.295984356499	0.298799903353
SVM	0.461663324968	0.461221145438

*** Compare the performance with the binary bag-of-words based classifiers. Do you see a huge improvement?**

No, there is no significant improvement in frequency bagging in comparison with binary bag of words. SVM and Decision tree have an almost similar performance in both kind of bagging but Naive bayes has a lower performance in frequency bag of words. The reason is GNB assumes features to follow a normal distribution while bernoulli is an event based model. In case of frequency bagging we just calculated the frequency of occurrence of the corresponding word which can't be a good estimation of normal distribution.

Question 4 - IMDB dataset

Random Classifier:

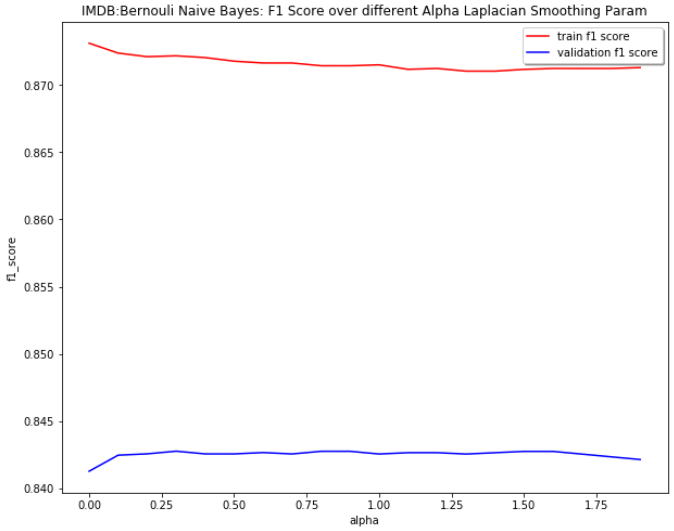
IMDB Confusion Matrix of Random Classifier:
[[6204 6296]
[6219 6281]]
IMDB F1 measure of Random Classifier: 0.499395251063

IMDB: Binary Bag of Words: Bernoulli Naive Bayes Performance

In the bernoulli naive bayes, I considered alpha (Laplacian Smoothing parameter) as a hyperparameter: I got the best validation set f1 measure with alpha = 0.3 and then I calculated train and test f1 measure with this alpha:

IMDB: Binary bagging- Bernoulli Naive Bayes	F1-measure	Best hyper-parameter configuration

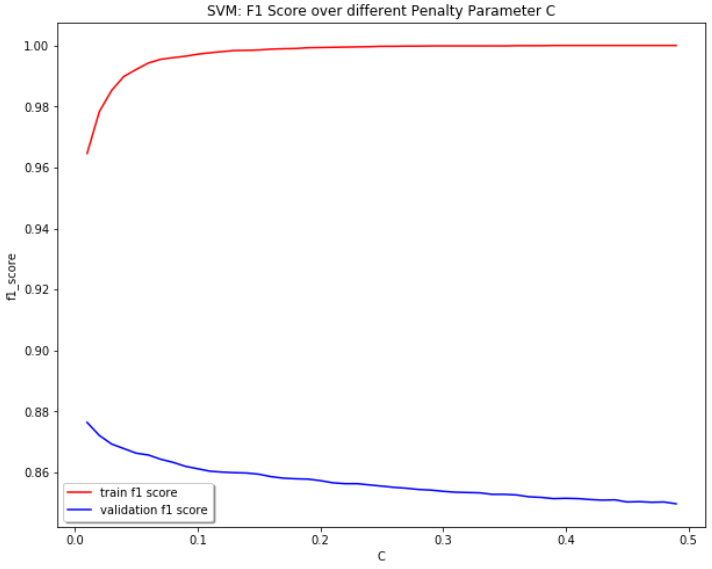
Train set	0.827755233198	0.3
Validation set	0.842770918909	0.3
Test set	0.836094836146	0.3



IMDB: Binary Bag of Words: Linear SVM Performance

I got the best validation set f1 measure with C = 0.01, in the following table I reported the f1 score for all datasets with best hyper-parameter configuration, I set dual to false since the number of examples is more than the number of features in IMDB.

IMDB:Binary bagging- Linear SVM	**F1-measure**	**Best hyper-parameter configuration**	**Dual**	**Range**
Train set	0.964599681397	0.01	False	np.arange(0.01, 0.5, 0.01)
Validation set	0.876396123782	0.01	False	np.arange(0.01, 0.5, 0.01)
Test set	0.870638539561	0.01	False	np.arange(0.01, 0.5, 0.01)



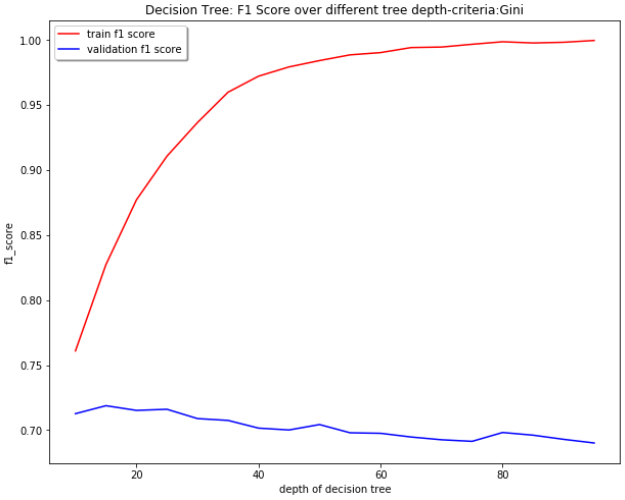
IMDB: Binary Bag of Words: Decision Tree Performance

I used the same strategy as the one with yelp, a hierarchical hyperparameter tuning.

a-1) Criterion: Gini, search for max_depth:

I picked max_depth=15 as the best configuration based on the performance of validation set:

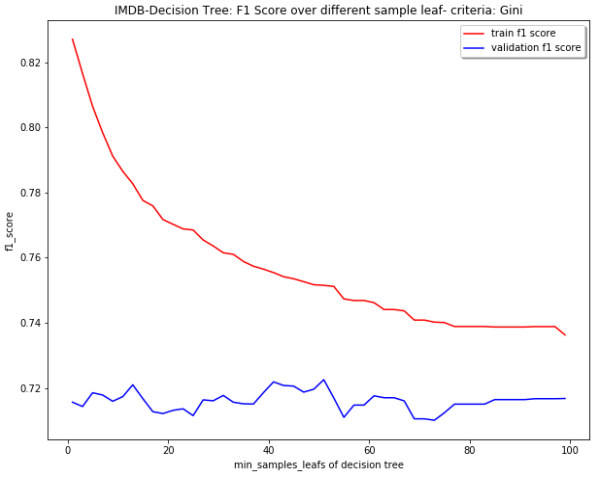
IMDB:Binary bagging-Decision Tree	F1-measure	Best hyper-parameter configuration	Range
Train set	0.827755233198	15	np.arange(10,100,5)
Validation set	0.718770183921	15	np.arange(10,100,5)
Test set	0.726110871834	15	np.arange(10,100,5)



a-2) Criterion: Gini, max_depth=25, search for the best min_samples_leaf

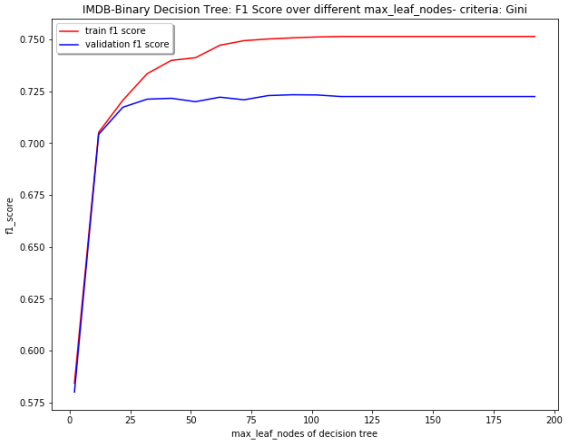
In this part, given criterion, max_depth, I searched for the best min_samples_leaf in the range (1,100)

IMDB:Binary bagging-Decision Tree	F1-measure	Criterion	max_depth	Best hyper-parameter configuration	Range
Train set	0.827755233198	Gini	15	51	np.arange(1,100,2)
Validation set	0.722511004245	Gini	15	51	np.arange(1,100,2)
Test set	0.726066031883	Gini	15	51	np.arange(1,100,2)



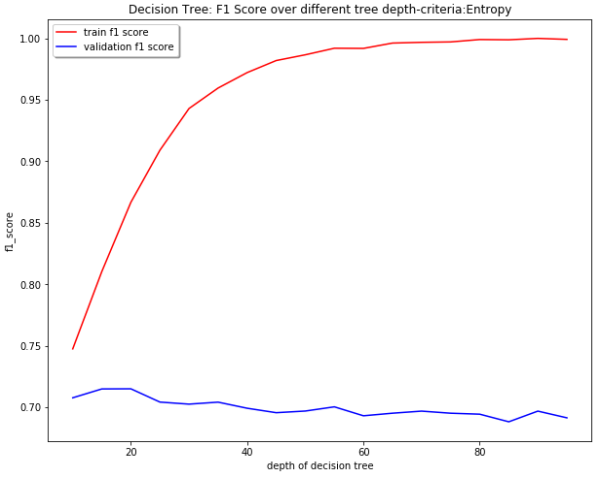
a-3) Criterion: Gini, max_depth=10, min_samples_leaf =81 search for best max_leaf_nodes
Here is the final f1 score for training, validation and test set with the corresponding hyperparameters value:

IMDB: Binary bagging Decision Tree	F1-measure	criterion	max_ depth	min_sample_ leafs	Best max_leaf_nodes for each dataset	Range
Train set	0.750810961476	Gini	15	51	92	np.arange(2,200,10)
Validation set	0.723377425928	Gini	15	51	92	np.arange(2,200,10)
Test set	0.727187525687	Gini	15	51	92	np.arange(2,200,10)



b-1) Criterion: Entropy, search for max_depth:
I picked max_depth=35 as the best configuration based on the performance of validation set:

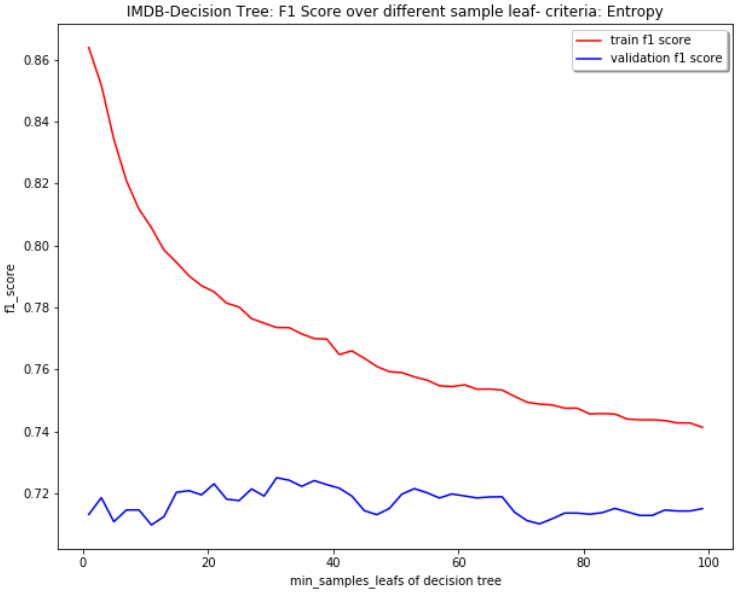
IMDB:Binary bagging- Decision Tree	F1-measure	Criterion	Best hyper-parameter configuration	Range
Train set	0.865584217266	Entropy	20	np.arange(10,100,5)
Validation set	0.715007328161	Entropy	20	np.arange(10,100,5)
Test set	0.717827244932	Entropy	20	np.arange(10,100,5)



b-2) Criterion: Entropy, max_depth=20 search for the best min_sample_leaf

I picked min_sample_leaf =31 as the best configuration based on the performance of validation set:

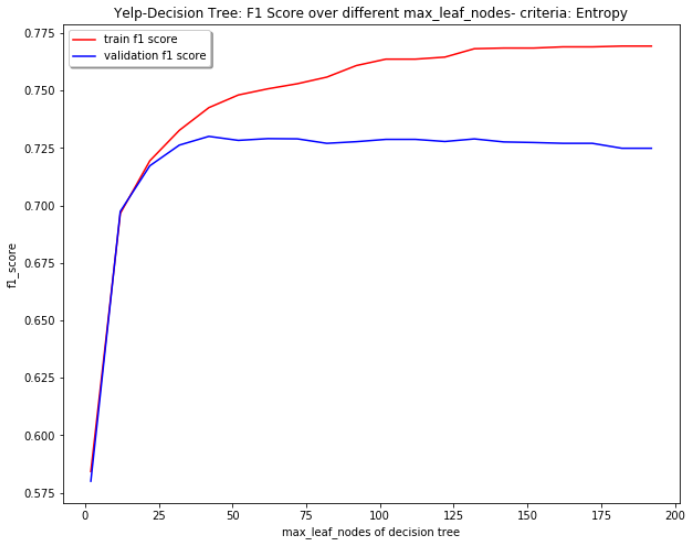
IMDB:Binary bagging-Decision Tree	F1-measure	Criterion	max_depth	Best hyper-parameter configuration	Range
Train set	0.512099859955	Entropy	20	31	np.arange(1,100,2)
Validation set	0.725031663415	Entropy	20	31	np.arange(1,100,2)
Test set	0.730862267338	Entropy	20	31	np.arange(1,100,2)



b-3) Criterion: Entropy, max_depth=35, min_sample_leaf=11, search for max_leaf_nodes

I picked max_leaf_nodes=42 as the best configuration since validation set has the maximum f1 score with this value of max_leaf_nodes.

IMDB:Binary bagging-Decision Tree	F1-measure	Criterion	max_ depth	min_sample _leaf	max_leaf_nodes	Range
Train set	0.742599907336	Entropy	20	31	42	np.arange(2,200,10)
Validation set	0.730098572221	Entropy	20	31	42	np.arange(2,200,10)
Test set	0.728079984337	Entropy	20	31	42	np.arange(2,200,10)



IMDB - Frequency Bag of words

IMDB: Frequency bag of words: Gaussian Naive Bayes

There is no hyper parameter for Gaussian Naive Bayes. I used macro F1 as a performance evaluation.

IMDB- Frequency bagging: Gaussian Naive Bayes	F1 score
Training Set	0.863360059737
Validation Set	0.76141198613
Test Set	0.695851692996

IMDB: Frequency Bag of Words: SVM

I set dual=False (Primal problem) since number of examples in the IMDB dataset is more than number of features.

IMDB: Frequency bagging-Linear SVM	F1-measure	Best C for each dataset	Dual	Range
Train set	0.928331987567	99	False	np.arange(1,50, 1)
Validation set	0.87698913076	89	False	np.arange(1,50, 1)

I picked C=47 as the best configuration of the hyperparameter:

Yelp: Frequency bagging- Linear SVM	F1-measure	Best hyper-parameter configuration	Dual	Range
Train set	0.943732372032	89	False	np.arange(1,50, 1)
Validation set	0.880090501969	89	False	np.arange(1,50, 1)
Test set	0.875238816466	89	False	np.arange(1,50, 1)

IMDB: Frequency Bag of Words: Decision Tree

Here is the final f1 score for training, validation and test set:

IMDB: Frequency bagging Decision Tree	F1-measure	criterion	max_depth	min_sample _leafs	Best max_leaf_nodes for each dataset	Range
Train set	0.750810961476	Gini	20	61	52	np.arange(2,200,10)
Validation set	0.727642736885	Gini	20	61	52	np.arange(2,200,10)
Test set	0.727997363478	Gini	20	61	52	np.arange(2,200,10)

IMDB: Frequency bagging Decision Tree	F1-measure	criterion	max_depth	min_sample _leafs	Best max_leaf_nodes for each dataset	Range
Train set	0.741569268754	Entropy	15	91	52	np.arange(2,200,10)
Validation set	0.719117356061	Entropy	15	91	52	np.arange(2,200,10)
Test set	0.715308697704	Entropy	15	91	52	np.arange(2,200,10)

*** Do your observations about the relative performance of the classifiers change as you change the dataset?**

Dataset	Classifier	Binary bag of words F1-measure	Frequency bag of words F1-measure
Yelp	Random	0.182202038978	
	Majority	0.103923019985	
	Naive Bayes	0.385221910535	0.254895792069
	SVM	0.461663324968	0.461221145438
	Decision Tree	0.295984356499	0.298799903353
IMDB	Random	0.499395251063	
	Naive Bayes	0.836094836146	0.695851692996
	SVM	0.870638539561	0.875238816466
	Decision Tree	0.728079984337	0.727997363478

Yelp Vs. IMDB:
In general IMDB has a pretty much better performance than yelp, and one of the reason is that yelp dataset has 5 classes and classification in this condition is more difficult than IMDB which has only 2 classes. According to the above table, we can see SVM has the best performance in both dataset either yelp or IMDB. Here is an order based on the classifiers performance:

Comparison:
Yelp: Binary: SVM > Bernoulli Naive Bayes > Decision Tree > Random > Majority
Frequency: SVM > Decision Tree > Gaussian Naive Bayes > Random > Majority
IMDB: Binary: SVM > Bernoulli Naive Bayes > Decision Tree > Random
Frequency: SVM > Decision Tree > Gaussian Naive Bayes > Random

SVM always leads to a better performance in both datasets and different kinds of bagging:
SVM uses a kernel to project feature space into kernel space and making classes linearly separable. It's the reason that SVM works better in all above cases, while decision tree tries to separate even a non-linearly separable data by defining rules to maximize information gain (entropy) and all it can do is defining some linear lines to make data separated as much as possible.

Gaussian Naive Bayes doesn't work as well as Bernoulli Naive Bayes:
5 class classification in yelp also can be a reason of low performance of gaussian naive bayes as well as unbalanced data since GNB cannot get a proper intuition of the data distribution (it has intrinsic bias). Also, there is no hyper parameter in GNB to penalize misclassifications.