COMP 551

Assignment 3 report

Yang Lu

260631276

1.

Vocabulary saved to “yelp-vocab.txt” and “IMBD-vocab.txt” respectively.

Dataset saved as “yelp-train.txt”, “yelp-valid.txt” and “yelp-test.txt”.

Dataset saved as “IMDB-train.txt”, “IMDB -valid.txt” and “IMDB -test.txt”.

2.(a)

Random classifier:

```
confusion matrix for random classifier:
 [[ 31  29  22  25  36]
 [ 35  41  34  39  41]
 [ 61  70  50  59  60]
 [143 145 148 137 129]
 [124 138 132 136 135]]

F1 score: 0.17956502282729314
```

Majority classifier:

```
confusion matrix for majority classifier:
 [[  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 score: 0.10392301998519615
```

Based on the F1 measurement, the random classifier performs slightly better than the majority classifier.

Yelp Dataset (BBoW) Question 2

2.(b) Training

Bernoulli Naïve Bayes:

The Bernoulli Naïve Bayes classifier is employed. The only hyperparameter here is **the additive Laplace smoothing parameter k**.

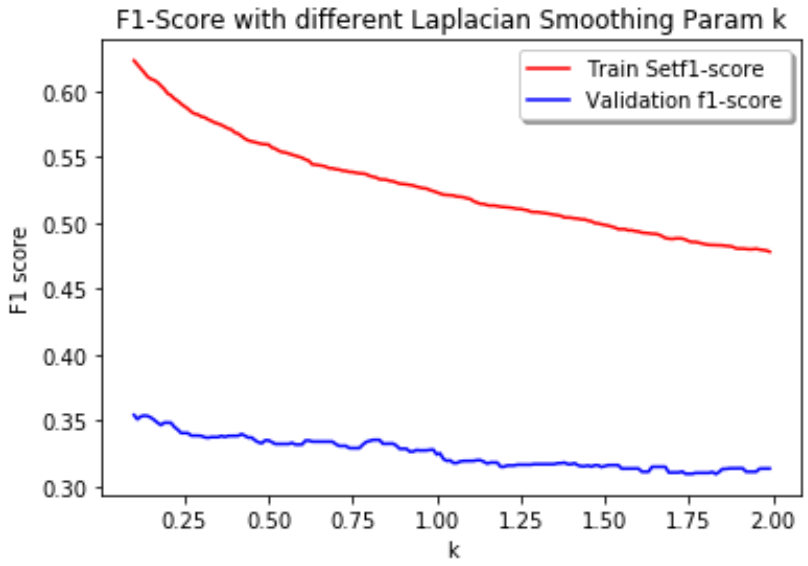
K value tried:

```
[0.01 0.03 0.05 0.07 0.09 0.11 0.13 0.15 0.17 0.19 0.21 0.23 0.25 0.27
0.29 0.31 0.33 0.35 0.37 0.39 0.41 0.43 0.45 0.47 0.49 0.51 0.53 0.55
0.57 0.59 0.61 0.63 0.65 0.67 0.69 0.71 0.73 0.75 0.77 0.79 0.81 0.83
0.85 0.87 0.89 0.91 0.93 0.95 0.97 0.99 1.01 1.03 1.05 1.07 1.09 1.11
1.13 1.15 1.17 1.19 1.21 1.23 1.25 1.27 1.29 1.31 1.33 1.35 1.37 1.39
1.41 1.43 1.45 1.47 1.49 1.51 1.53 1.55 1.57 1.59 1.61 1.63 1.65 1.67
1.69 1.71 1.73 1.75 1.77 1.79 1.81 1.83 1.85 1.87 1.89 1.91 1.93 1.95
1.97 1.99]
```

Corresponding F-1 score over validation set:

```
[0.3490164484683026, 0.35662981593128007, 0.3502980157079288, 0.35456952725554963, 0.35228767871493194, 0.35078929812721676, 0.3535188939181
6715, 0.35211420697975654, 0.3481658646704147, 0.3482151172970176, 0.3481504003699704, 0.34293683281719967, 0.3403100288331621, 0.3383224482
887961, 0.3383275478270594, 0.33709956580668254, 0.3372668585567843, 0.3372123174709923, 0.3373668750320813, 0.3381246309246309, 0.338199247
1437412, 0.33820627874628517, 0.3369326219146634, 0.3332357014463544, 0.3347650777788054, 0.33347752703434347, 0.3319847837255142, 0.3320287
355249263, 0.3328275962059243, 0.3315505724986394, 0.33459824727515736, 0.3338249203571357, 0.33379346693468354, 0.33379346693468354, 0.3337
9346693468354, 0.3305260888876669, 0.33053622074413636, 0.32888728810935763, 0.32884741654726374, 0.33349788610625597, 0.33504105136676854,
0.3351125727696186 4, 0.3322578787267427, 0.3322578787267427, 0.3283171823791656, 0.3283801284247326 3, 0.3262785092094312, 0.327083108833808
2, 0.3271503297128418, 0.3277411361216898, 0.32493425680173654, 0.31917780843390714, 0.31756963933168464, 0.3187304773025835, 0.319096681768
0527, 0.3191710913062198, 0.3197007254815164, 0.3171286913493695, 0.3179044390098629, 0.3147191976877795, 0.3156586769354394, 0.315658676935
4394, 0.3162808302180663 6, 0.3162808302180663 6, 0.3166811378314988, 0.3167017106132 4165, 0.3167017106132 4165, 0.3167017106132 4165, 0.3173950
01547593, 0.3166609772192097, 0.31737705587308, 0.3151017338491985 4, 0.3156492768362 3545, 0.3158103195261 2015, 0.3144967275035275, 0.3158918
7312436345, 0.315919610697267, 0.3132910723736 1054, 0.3133318369113963, 0.3133664921111727, 0.3108969104128125, 0.3108969104128125, 0.314644
40385244613, 0.3146444038524 4613, 0.3103260508387807, 0.3103826951057 8154, 0.3110766509860 1267, 0.3093373955166041, 0.3100409947473 4687, 0.3
1004099474734687, 0.3100539198654023, 0.3091653679617 7235, 0.3130465876705709, 0.3134647919617 0346, 0.3135106413476608, 0.3135582652829 5333,
0.3109058649877955, 0.3109107476506026, 0.3133625745497 8256, 0.3133295831103621]
```

Plot:



The overall trends in performance are decreasing as we increase k. Therefore, the **best** value for hyperparameter **k** would be at k=0.03.

F1 score at k=0.03:

```
F1 score for train set: 0.6554711753880327
F1 score for valid set: 0.35662981593128007
F1 score for test set: 0.3721534723175413
```

Linear SVM classifier

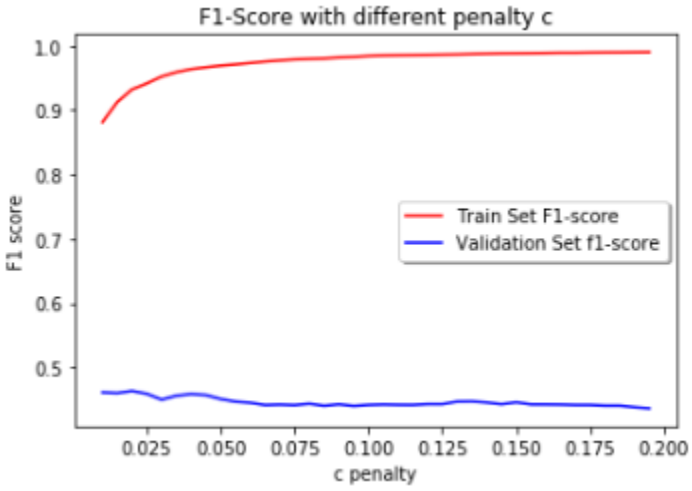
Hyperparameter: **penalty c**

Different value of c tried at :

```
[0.01  0.015 0.02  0.025 0.03  0.035 0.04  0.045 0.05  0.055 0.06  0.065
 0.07  0.075 0.08  0.085 0.09  0.095 0.1   0.105 0.11  0.115 0.12  0.125
 0.13  0.135 0.14  0.145 0.15  0.155 0.16  0.165 0.17  0.175 0.18  0.185
 0.19  0.195]
```

With corresponding F1 scores on validation set:

```
[0.46114395202301955, 0.4601515778808894, 0.46348321312717966, 0.4590312592952575, 0.44982439350831954, 0.45599911039317576, 0.4585334909958
7515, 0.45700181689760144, 0.4507473349810229, 0.4466827372256315, 0.44494269364390 86, 0.4415059281254961, 0.44211497978481784, 0.4413899923
3245474, 0.4434925511533262, 0.4403366823169847, 0.44255976753344745, 0.4400703129190776, 0.44163257286645363, 0.4422293633029627, 0.4418261
759301544, 0.44168947929580904, 0.4428784859751314, 0.4428511946486157, 0.4468079204965914, 0.4470904485474021, 0.44534875230414706, 0.44274
392747609886, 0.4456692714215661, 0.4424273661771279, 0.4423802030710897, 0.4421725392887842, 0.44159372417764364, 0.4415686568100751, 0.440
5208712751484, 0.4405208712751484, 0.43818916264337837, 0.43637199381875647]
```



According to the plot, we notice that the performance on validation set do not oscillate much as we change c. **Best penalty c** at c=0.02:

F1 score at c=0.02 :

```
F1 score for train set: 0.9322880355692773
F1 score for valid set: 0.46348321312717966
F1 score for test set: 0.43239419473847096
```

Decision tree classifier:

Hyperparameter:

1. **Max depth**.
2. **Min sample leaf**.
3. **Max leaf nodes**.

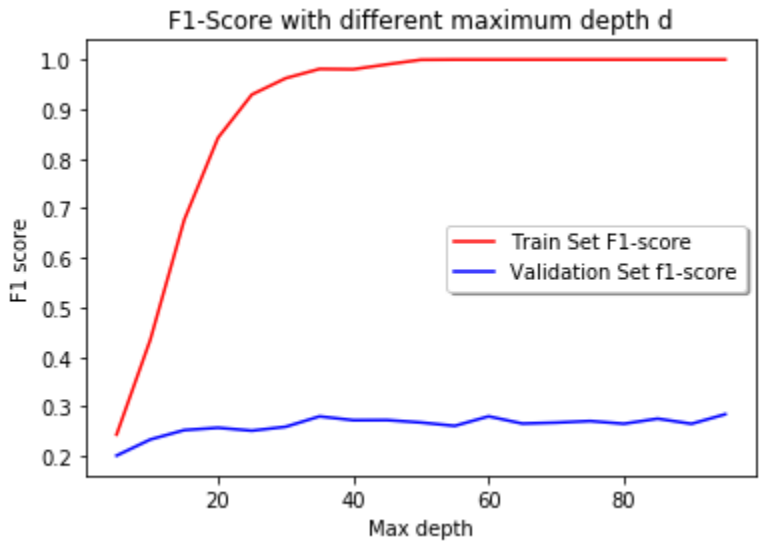
The plan is we first seek for the optimal max depth. Then, with the best max depth we find the optimal min sample leaf. And in the last step, we find the optimal max leaf nodes with the former two hyperparameter optimized.

a. **Max depth** tried:

```
[ 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95]
```

With corresponding validation F1 scores:

```
[0.20094033625948518, 0.2335349245695298, 0.252426418817649, 0.25716017957900994, 0.25128970336686673, 0.2591141537967784, 0.2800616757564673, 0.2724202009020372, 0.27262641320551817, 0.2677609979690324, 0.2609592230812535, 0.2801744952173279, 0.26557425940410073, 0.26754874818281227, 0.27069736696076074, 0.26523432115413426, 0.2755278675493626, 0.2651086971466265, 0.2844266978009614]
```



Best max depth at max_depth=95 with corresponding F1 scores over train set, validation set and test set.

F1 score:

```
F1 score for train set: 1.0
F1 score for valid set: 0.26728541371279546
F1 score for test set: 0.295048504420958
```

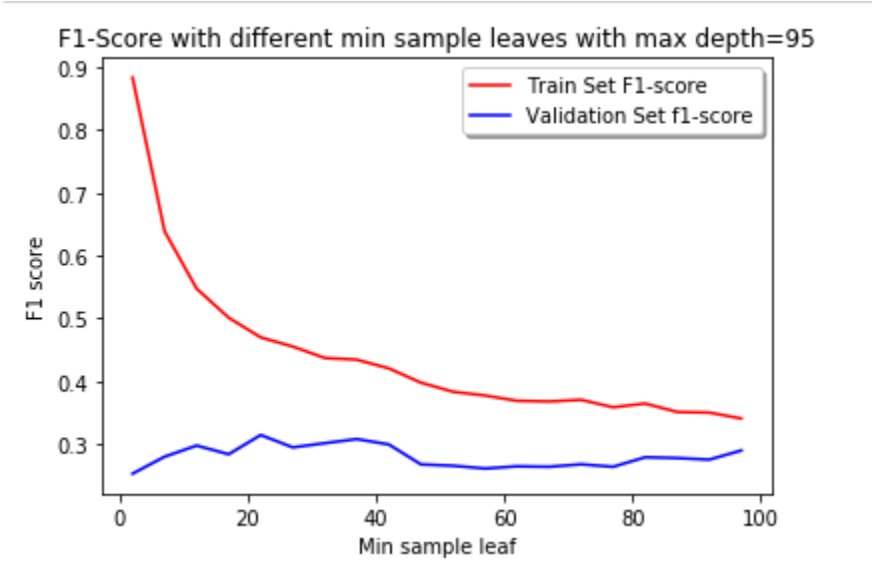
b. **Min sample leaf**:

With max_depth = 95, min_leaf tried at :

```
[ 2  7 12 17 22 27 32 37 42 47 52 57 62 67 72 77 82 87 92 97]
```

With corresponding F1 scores on validation set:

```
[0.2524437120698094, 0.2794771291524004, 0.2972056372644848, 0.2833626023412531, 0.31406899392041643, 0.29429661853925176, 0.301019356979174
2, 0.30755357866602473, 0.298972031174547, 0.2671894793939599, 0.2650510595274297, 0.260610165093725, 0.2641553664773479, 0.2634054795479041
6, 0.26735551100225896, 0.2633850902572951, 0.2785929068240834, 0.2772966453369798, 0.2747937707635211, 0.28936990414832187]
```



Best min sample leaf is min_sample_leaf=22 when max depth =95, and F1 scores over train, validation and test set are:

F1 score:

```
F1 score for train set: 0.4696488723032758
F1 score for valid set: 0.31406899392041643
F1 score for test set: 0.3277245957780928
```

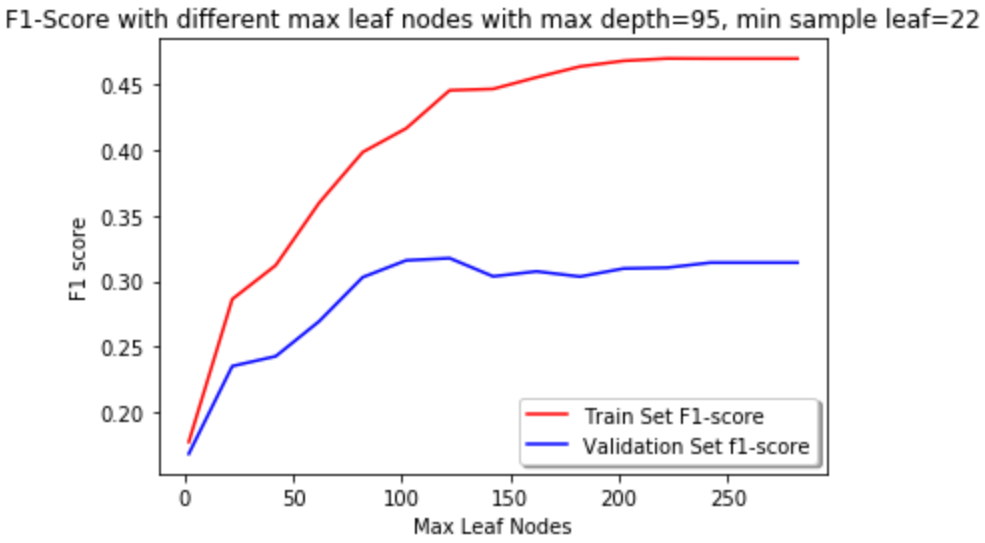
c. **max leaf nodes**:

With max_depth = 95, min_sample_leaf=22, max leaf nodes tried at:

```
[  2  22  42  62  82 102 122 142 162 182 202 222 242 262 282]
```

With corresponding F1scores on validation set:

```
[0.16825256582799963, 0.23514433473799926, 0.2426365139023664 3, 0.2693584223860499, 0.30292645349390485, 0.31579265070248164, 0.317456843548
84807, 0.3035865621467764, 0.30738177184682064, 0.3034072296627001, 0.3095806197525851, 0.31020932332414675, 0.31406899392041643, 0.31406899
392041643, 0.31406899392041643]
```



Best <u>max leaf nodes</u> is 122 when max depth=95, min sample leaf=22. And the F1 scores over train, validation and test set are

F1 score:

```
F1 score for train set: 0.44548570975002233
F1 score for valid set: 0.31745684354884807
F1 score for test set: 0.33523894706261753
```

2.c 2.d

Bernoulli Naïve Bayes: with Laplace smoothing parameter k =0.03 range=(0.1,2)

```
F1 score for train set: 0.6554711753880327
F1 score for valid set: 0.35662981593128007
F1 score for test set: 0.3721534723175413
```

Decision tree: with max depth=95|range=(5,100), min sample leaf=22|range=(2,100), max leaf nodes=122|range=(2,300)

```
F1 score for train set: 0.44548570975002233
F1 score for valid set: 0.31745684354884807
F1 score for test set: 0.33523894706261753
```

Linear SVM: with penalty c =0.02 range=(0.01,0.2)

```
F1 score for train set: 0.9322880355692773
F1 score for valid set: 0.46348321312717966
F1 score for test set: 0.43239419473847096
```

2.e

Comments: based on the F1 score over test set, using best hyperparameter found from validation set, the linear SVM classifier perform the best. This is because SVM is good when

dealing with data with high dimensionality. The Bernoulli naïve Bayes classifier perform slightly worse, probably because the assumption does not hold. and Decision tree classifier perform worst since the data is not well-organized, therefore overfitted. In conclusion: linear SVM classifier> Bernoulli naïve Bayes> Decision tree classifier> Random classifier> Majority classifier in this case using BBoW.

Yelp Dataset (FBoW) Question 3

3.a model training.

Gaussian Naïve Bayes

F1 score:

```
F1 score for train set: 0.7879844613865794
F1 score for valid set: 0.24561901520780366
F1 score for test set: 0.2477846680185553
```

Decision Tree

Hyperparameter:

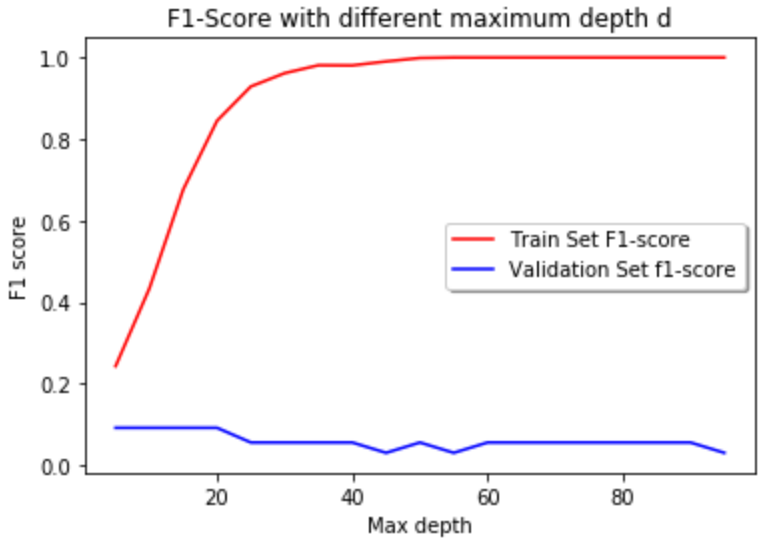
1. **Max depth**.
2. **Min sample leaf**.
3. **Max leaf nodes**.

Max depth tried at :

```
[ 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95]
```

With corresponding F1 scores over validation set:

```
[0.21481331680194202, 0.30750512066367985, 0.3032675878205417 5, 0.3066647730449856, 0.2907146926877023, 0.2941288615374931, 0.29390479721401963, 0.30196846349643314, 0.3087986761865267 3, 0.3075748137661375, 0.2951480627729559 7, 0.2982482229772897, 0.29895209626814373, 0.30051888672143434, 0.30010150245023137, 0.296531750967384 5, 0.29686873547490567, 0.3040040730173811, 0.3013400005993988]
```



Best <u>max depth</u>= 45, and the corresponding F1 scores on train set, validation set and test set are:

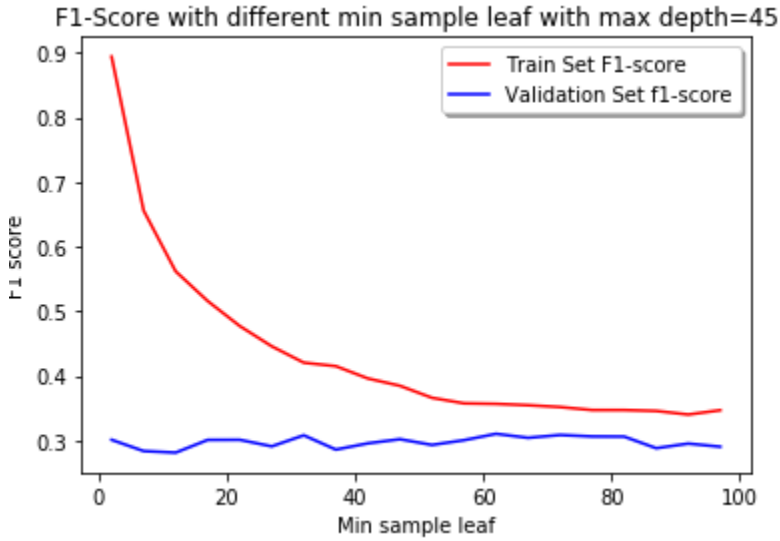
```
F1 score for train set: 0.9973054275336768
F1 score for valid set: 0.30743736994786797
F1 score for test set: 0.28590959758462653
```

<u>Min sample leaf</u>, with max depth =45, tried at:

```
[ 2  7 12 17 22 27 32 37 42 47 52 57 62 67 72 77 82 87 92 97]
```

With corresponding F1 scores over validation set:

```
[0.2868127052737695, 0.28593516461163015, 0.2812551524924649, 0.30095172630853634, 0.3011045248396671, 0.2909866887564311, 0.30639775068746
9, 0.28596786313994443, 0.29564472053877333, 0.3020564476502276, 0.29315702592009096, 0.30052950199458367, 0.31036705262312714, 0.3041882665
460673, 0.3085843021705025, 0.30629604588742704, 0.3060440545747247, 0.2881254771186147, 0.2952535259655825, 0.2903996363713224]
```



Best <u>min sample leaf</u>=62. And the F1 scores over train, validation and test sets with max depth =45, min sample leaf =62 are:

F1 Scores:

```
F1 score for train set: 0.3566998619855164
F1 score for valid set: 0.31036705262312714
F1 score for test set: 0.26927151388802584
```

With best max depth =45, best min sample leaf=62, the max leaf nodes tried at:

```
[  2  22  42  62  82 102 122 142 162 182 202 222 242 262 282]
```

With corresponding F1 scores over validation set:

```
[0.18618433440030127, 0.2990354723088583, 0.3051489125474637, 0.30846278103203, 0.30915180747510773, 0.31036705262312714, 0.3103670526231271
4, 0.31036705262312714, 0.31036705262312714, 0.31036705262312714, 0.31036705262312714, 0.31036705262312714, 0.31036705262312714, 0.31036705
262312714, 0.31036705262312714]
```

Best max leaf nodes =102, given best max depth=45, best min sample leaf=62. And the F1 scores over train, validation, and test set are following:

F1 scores:

```
F1 score for train set: 0.3566998619855164
F1 score for valid set: 0.31036705262312714
F1 score for test set: 0.26927151388802584
```

Linear SVM

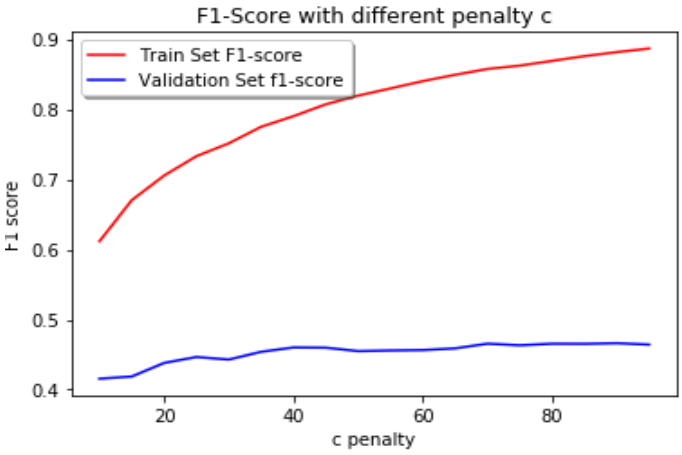
Hyperparameter: <u>penalty c</u>

c value tried at :

```
[10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95]
```

With corresponding F1 scores over validation set:

```
[0.41562269273763697, 0.4187208427153653, 0.43806749083734964, 0.44655165463673596, 0.44290718151435815, 0.45396950657367324, 0.460175822202
92075, 0.4598520627447665 3, 0.4549462440696873 7, 0.4558519311071718, 0.45642004861024504, 0.4588815734314049, 0.46556343736467537, 0.4633037
1055432573, 0.4655014002078294 3, 0.46536787403055246, 0.4662595033889982, 0.46426506890063485]
```



Best c penalty =90, and the F1 score :

F1 score:

```
F1 score for train set: 0.8820730541468915
F1 score for valid set: 0.4656633207681945
F1 score for test set: 0.46405533637129953
```

3.b,3.c

Gaussian Naïve Bayes: no hyperparameter

```
F1 score for train set: 0.7879844613865794
F1 score for valid set: 0.24561901520780366
F1 score for test set: 0.2477846680185553
```

Decision tree: with max depth=95|range=(5,100), min sample leaf=22|range=(2,100), max leaf nodes=122|range=(2,300)

```
F1 score for train set: 0.3566998619855164
F1 score for valid set: 0.31036705262312714
F1 score for test set: 0.26927151388802584
```

Linear SVM: with penalty c =0.02 range=(0.01,0.2)

```
F1 score for train set: 0.8820730541468915
F1 score for valid set: 0.4656633207681945
F1 score for test set: 0.46405533637129953
```

3.d

All models performed better than random classifier and majority classifier.
Linear SVM has the best test performance. The model performs better if we increase the penalty c.

3.e

Naïve Bayes perform better with better with BBoW(F1=0.37215) than with FBoW(F1=0.24778).

Decision tree perform slightly better with BBoW(F1=0.33) than with FBoW(F1=0.26927).

Linear SVM performs roughly the same. BBoW(F1=0.42329). FBoW(F1=0.464655).

3.f

BBoW representation is better with respect to Yelp dataset. BBoW performs better than or at least as well as FBoW for all classifiers in our study. Notice that using frequency instead of binary to the represent the training vectors would lead to overall drop in classifiers' performance.

Q4 IMDB Dataset BBoW Representation

4.a

Random classifier

```
confusion matrix for random classifier:
 [[6331 6169]
 [6274 6226]]
```

F1 score for random classifier is: 0.502271220064322

Majority classifier

```
confusion matrix for majority classifier:
 [[12500     0]
 [12500     0]]
```

F1 score for majority classifier is: 0.3333333333333333

4.b

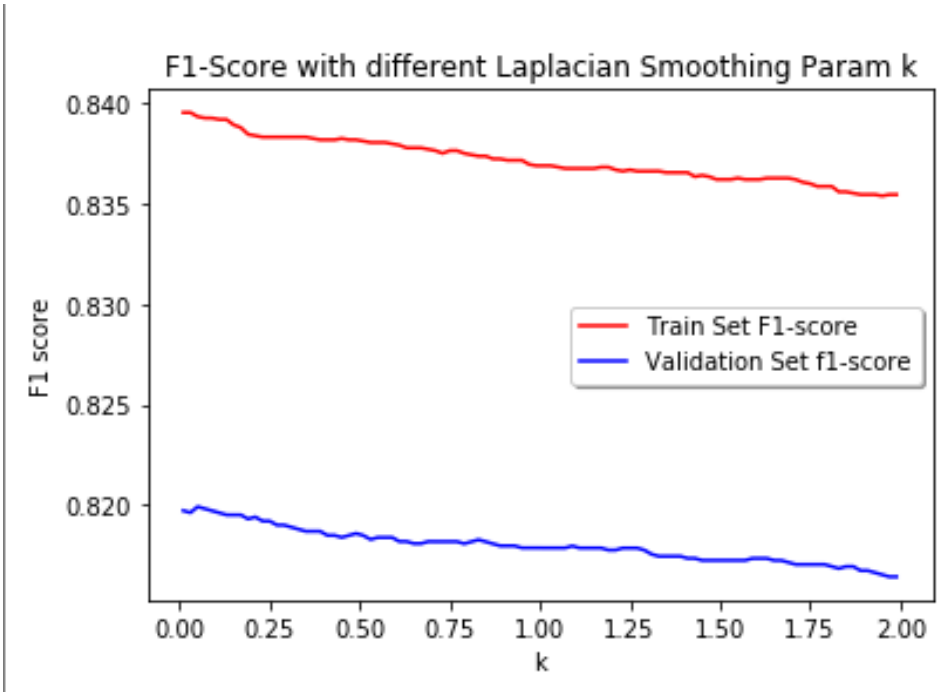
Bernoulli naïve Bayes:

Hyperparameter Additive Laplace smoothing k tried at:

```
[0.01 0.03 0.05 0.07 0.09 0.11 0.13 0.15 0.17 0.19 0.21 0.23 0.25 0.27
 0.29 0.31 0.33 0.35 0.37 0.39 0.41 0.43 0.45 0.47 0.49 0.51 0.53 0.55
 0.57 0.59 0.61 0.63 0.65 0.67 0.69 0.71 0.73 0.75 0.77 0.79 0.81 0.83
 0.85 0.87 0.89 0.91 0.93 0.95 0.97 0.99 1.01 1.03 1.05 1.07 1.09 1.11
 1.13 1.15 1.17 1.19 1.21 1.23 1.25 1.27 1.29 1.31 1.33 1.35 1.37 1.39
 1.41 1.43 1.45 1.47 1.49 1.51 1.53 1.55 1.57 1.59 1.61 1.63 1.65 1.67
 1.69 1.71 1.73 1.75 1.77 1.79 1.81 1.83 1.85 1.87 1.89 1.91 1.93 1.95
 1.97 1.99]
```

with corresponding F1 scores over validation set:

```
[0.8197286122420135, 0.8196270578602247, 0.8199232869196121, 0.8198217164428289, 0.8197201407577066, 0.8196185598580186
0.8195169737375363, 0.819514103119906, 0.819514103119906, 0.8193137858092734, 0.8194125028987398, 0.8192092867251469,
0.8192092867251469, 0.8190089646051748, 0.8190089646051748, 0.8189044230431142, 0.8188027912783944, 0.8187011542345941,
0.8187011542345941, 0.8187011542345941, 0.8184978642848031, 0.8184978642848031, 0.8183962113663358, 0.8184948899099387,
0.8185935664383633, 0.8184919008922505, 0.8182945531438348, 0.8183932280247116, 0.8183932280247116, 0.8183932280247116,
0.81819288961106, 0.81819288961106, 0.8180912207617698, 0.8180912207617698, 0.8181898883105925, 0.8181868723408892,
0.8181868723408892, 0.8181868723408892, 0.8181868723408892, 0.8180851854986164, 0.8181838417004945, 0.8182824959081505,
0.8181807963879467, 0.8180790915174367, 0.8179773812903729, 0.8179773812903729, 0.8179773812903729, 0.8178756657005062,
0.8178756657005062, 0.8178756657005062, 0.8178725786069163, 0.8178725786069163, 0.8178725786069163, 0.8178725786069163,
0.8179712104989637, 0.8178694768117389, 0.8178694768117389, 0.8178694768117389, 0.8178694768117389, 0.8177677377385413,
0.8177677377385413, 0.8178663603134766, 0.8178663603134766, 0.8178663603134766, 0.8177646121698836, 0.8175610996743661,
0.8174593353099289, 0.8174593353099289, 0.8174593353099289, 0.8174593353099289, 0.817357565526029, 0.817357565526029,
0.8172557903164084, 0.8172557903164084, 0.8172557903164084, 0.8172557903164084, 0.8172557903164084, 0.8172557903164084,
0.8172557903164084, 0.8173543888422867, 0.8173543888422867, 0.8173543888422867, 0.8172526045110714, 0.8172526045110714,
0.8171540096748072, 0.8170522235949651, 0.8170522235949651, 0.8170522235949651, 0.8170522235949651, 0.8170522235949651,
0.8169536305073275, 0.8168518426788588, 0.8169504320706209, 0.8169504320706209, 0.8167468326633772, 0.8167468326633772,
0.8166450247679506, 0.8165432114029687, 0.8164413925621652, 0.8164413925621652]
```



Best k=0.05.

F1 scores:

```
F1 score for train set: 0.8393111199590175
F1 score for valid set: 0.8199232869196121
F1 score for test set: 0.8035944569794238
```

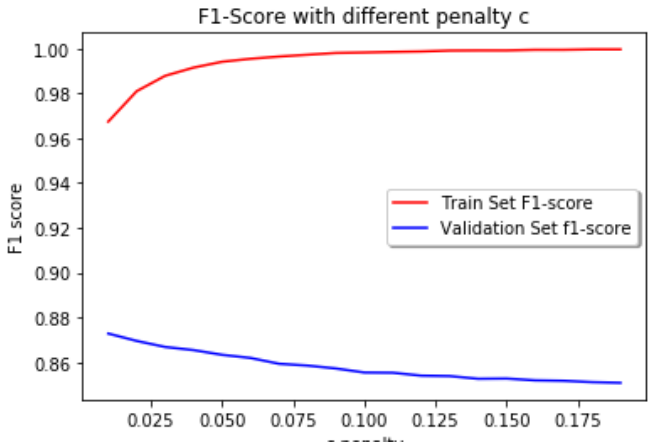
Linear SVM

Hyper parameter penalty c tried :

```
[0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1  0.11 0.12 0.13 0.14
 0.15 0.16 0.17 0.18 0.19]
```

with corresponding F1 scores over validation set:

```
[0.8727981632054767, 0.8694973573214858, 0.8667998667998668, 0.8653999784639965, 0.8632999876969989, 0.8618999323309668,
0.8592998860329076, 0.8584997608645959, 0.8571995373265009, 0.8553990224973921, 0.8552990956193476, 0.8539997897596971,
0.8537994151976609, 0.852599286580547, 0.8526992207788779, 0.8518983871734362, 0.8516985748233041, 0.8510981759526555,
0.8507982752280616]
```



Best c=0.01

F1 scores:

```
F1 score for train set: 0.9673998781479889
F1 score for valid set: 0.8727981632054767
F1 score for test set: 0.8683184417331121
```

Decision tree

Hyperparameter: max depth, min sample leaf, max leaf nodes.

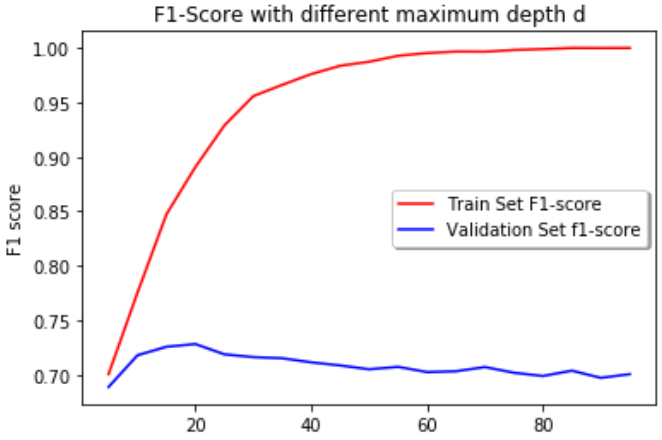
Max depth:

Tried max depth at :

```
[ 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95]
```

With corresponding F1 scores over validation set:

```
[0.6887017465164107, 0.7178417615753537, 0.7256269474686565, 0.7280978940690265, 0.718668269867207, 0.7161341508549633,
0.7150492610591002, 0.7113297492077673, 0.7084627736115624, 0.7049560856637902, 0.707175215310224, 0.7023626643726189,
0.703146697020917, 0.7069957690189046, 0.7017758438433512, 0.6987733836161764, 0.703681506762837, 0.6970666015928256,
0.7004740940043904]
```



Best max depth=20,

F1 scores:

```
F1 score for train set: 0.8908220057319984
F1 score for valid set: 0.7201655527890622
F1 score for test set: 0.7243765343871711
```

Min sample leaf:

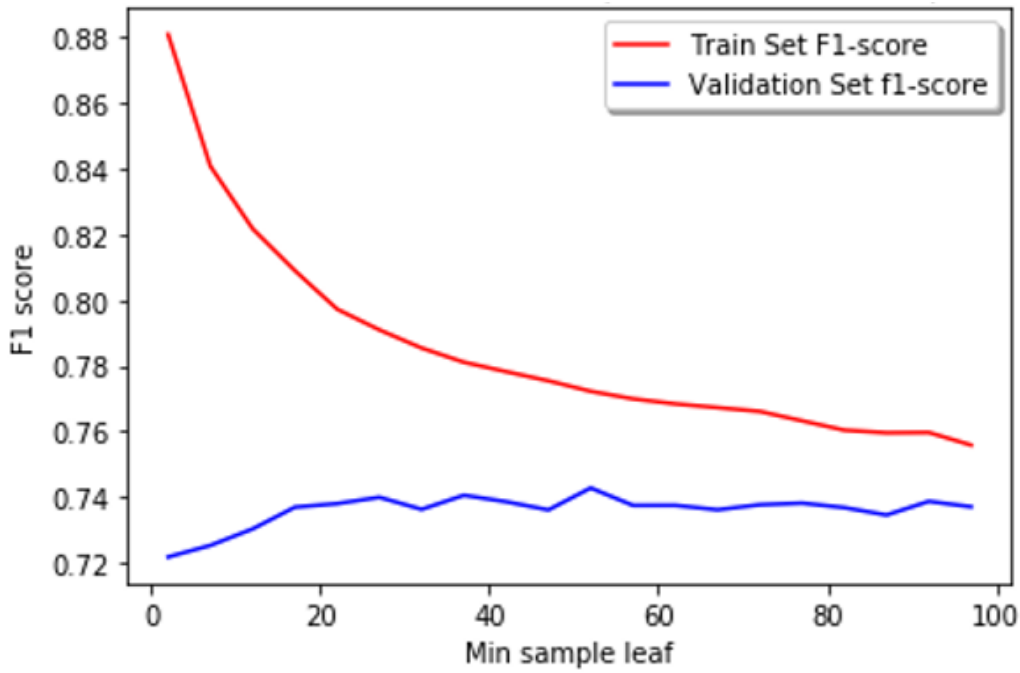
With best max depth =20, min sample leaf tried at:

```
[ 2  7 12 17 22 27 32 37 42 47 52 57 62 67 72 77 82 87 92 97]
```

Corresponding F1 scores over validation set:

```
[0.7217148008236158, 0.7252455903321103, 0.7302706518425823, 0.7368977666746699, 0.7379764533717175, 0.7399032313730947,
0.7362369349253243, 0.7405242395408145, 0.7385478527173465, 0.7360670840345878, 0.7427505234245484, 0.7374356886615255,
0.7374444324189986, 0.7361122817156915, 0.7376351389487512, 0.7381372250034524, 0.7367600656382485, 0.7345117782598977,
0.7386591730228824, 0.7370205121295608]
```

Plot:



Best min sample leaf =52, given best max depth= 20.

F1 scores:

```
F1 score for train set: 0.7721918378874133
F1 score for valid set: 0.7427505234245484
F1 score for test set: 0.7446305104354195
```

Max leaf nodes

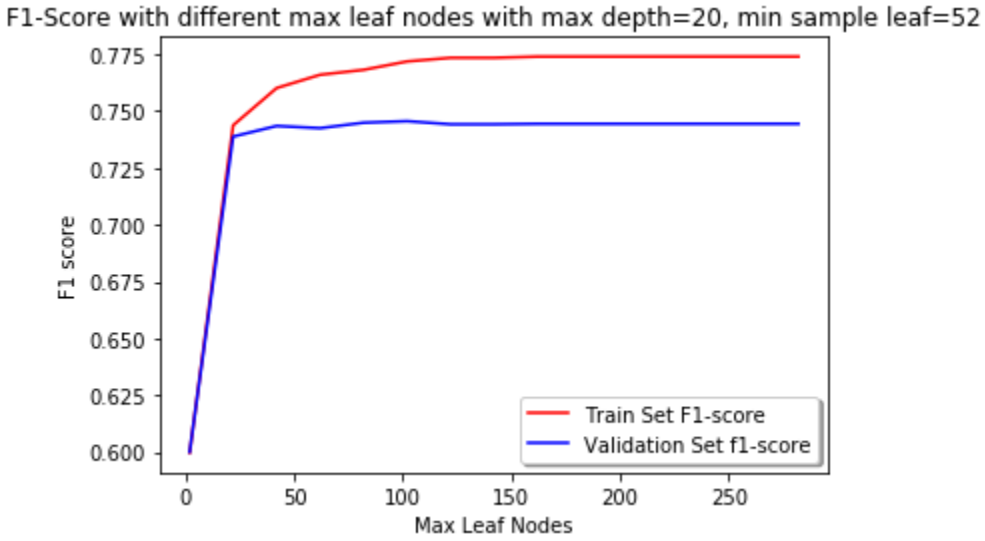
Given best max depth=20, best min sample leaf=52, Max leaf nodes tried at:

```
[  2  22  42  62  82 102 122 142 162 182 202 222 242 262 282]
```

Corresponding F1 scores over validation set:

```
[0.6000310443989368, 0.7387818336612177, 0.7434525732127475, 0.7424859147758187, 0.7448887907664503, 0.7455578047683311,
0.7441949372959218, 0.7441949372959218, 0.7443862714377218, 0.7443862714377218, 0.7443862714377218, 0.7443862714377218,
0.7443862714377218, 0.7443862714377218, 0.7443862714377218]
```

Plot:



Best max leaf nodes=102, with best max depth=20, and best min sample leaf=52.

F1 scores:

```
F1 score for train set: 0.7718279714511586
F1 score for valid set: 0.7455578047683311
F1 score for test set: 0.7481516130468167
```

4.c 4.d

Bernoulli Naïve Bayes: with Laplace smoothing parameter k =0.05 range=(0.1,2)

```
F1 score for train set: 0.8393111199590175
F1 score for valid set: 0.8199232869196121
F1 score for test set: 0.8035944569794238
```

Decision tree: with max depth=20|range=(5,100), min sample leaf=52|range=(2,100), max leaf nodes=102|range=(2,300)

```
F1 score for train set: 0.7718279714511586
F1 score for valid set: 0.7455578047683311
F1 score for test set: 0.7481516130468167
```

Linear SVM: with penalty c =0.01 range=(0.01,0.2)

```
F1 score for train set: 0.9673998781479889
F1 score for valid set: 0.8727981632054767
F1 score for test set: 0.8683184417331121
```

4.e

For IMDB dataset with BBoW representation, both Linear SVM and Naïve Bayes classifier have good performance(F1 >0.8).

In terms of Naïve Bayes classifier, we notice that , as we increases the smoothing parameter, the F1 scores drop. Therefore, the additive Laplace smoothing parameter may not be favored in this case.

IMDB Dataset FBoW Representation Q5

5.a

Gaussian Naïve Bayes

F1 score:

```
F1 score over train set: 0.8627017402201094
F1 score over validation set: 0.7593849791768671
F1 score over test set: 0.6920485233021636
```

Decision Tree

Hyperparameter:

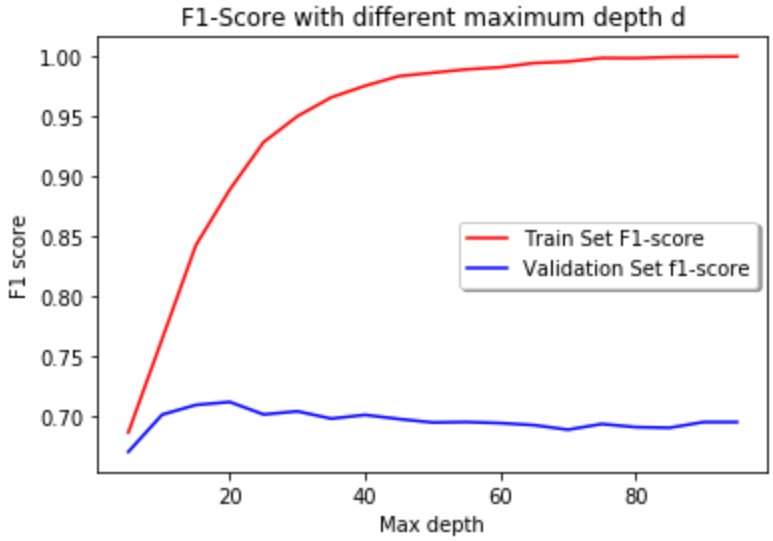
1. **Max depth**.
2. **Min sample leaf**.
3. **Max leaf nodes**.

Max depth tried at :

```
[ 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95]
```

With corresponding F1 scores over validation set:

```
[0.670444115925511, 0.7014723690560629, 0.709541704478762, 0.7120116282369533, 0.7016218474646028, 0.7042151344590369,
0.6981227194161705, 0.7012318778804756, 0.6977995648313734, 0.6948991182584519, 0.6952948779068977, 0.6943992176619973,
0.6926878027788923, 0.6888925303096528, 0.6936622561265774, 0.6910385753791712, 0.6904720651038756, 0.695222949313816,
0.6952508470091141]
```



Best <u>max depth</u>= 20, and the corresponding F1 scores on train set, validation set and test set are:

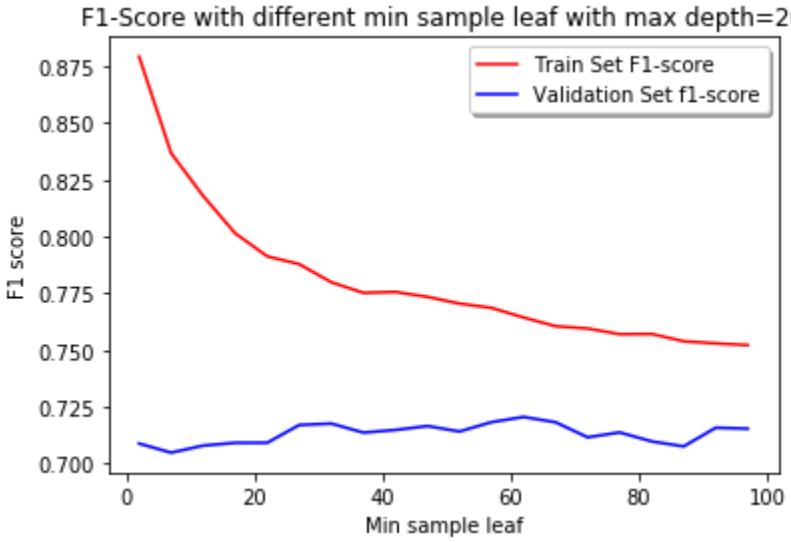
```
F1 score for train set: 0.8884281155369583
F1 score for valid set: 0.7104900575581642
F1 score for test set: 0.7127756204543972
```

<u>Min sample leaf</u>, with max depth =20, tried at:

```
[ 2  7 12 17 22 27 32 37 42 47 52 57 62 67 72 77 82 87 92 97]
```

With corresponding F1 scores over validation set:

```
[0.7087936241450896, 0.704780736263616, 0.7078899310420821, 0.7091129363912386, 0.7091382244019617, 0.7169747445461799,
0.7175968627450982, 0.7136259533676779, 0.7147699448467155, 0.7165133269367315, 0.7141004648962854, 0.7182274667881523,
0.7205173178433107, 0.7181884271435623, 0.7115418352956691, 0.7136966444886604, 0.7097352669528716, 0.7075341951939186,
0.715766157125729, 0.7153506015830831]
```



Best <u>min sample leaf</u>=47. And the F1 scores over train, validation and test sets with max depth =20, min sample leaf =47 are:

F1 Scores:

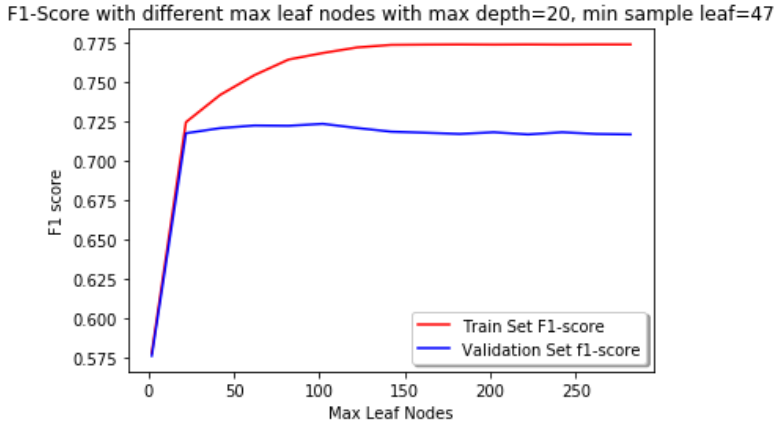
```
F1 score for train set: 0.7734452741982862
F1 score for valid set: 0.7168190136663104
F1 score for test set: 0.7251833449883782
```

With best max depth =20, best min sample leaf=47, the max leaf nodes tried at:

```
[  2  22  42  62  82 102 122 142 162 182 202 222 242 262 282]
```

With corresponding F1 scores over validation set:

```
[0.5763827509464391, 0.717268828888385, 0.7204987673995642, 0.7221733175254152, 0.7219899052223672, 0.7232246479103936,
0.720596314713426, 0.7182704778817497, 0.717594321269285, 0.7168190136663104, 0.7179001956118884, 0.7165133269367315,
0.7179001956118884, 0.7168190136663104, 0.7165133269367315]
```



Best max leaf nodes =102, given best max depth=20, best min sample leaf=47. And the F1 scores over train, validation, and test set are following:

F1 scores:

```
F1 score for train set: 0.7685600802140135
F1 score for valid set: 0.7226150508593256
F1 score for test set: 0.7283815324385501
```

Linear SVM

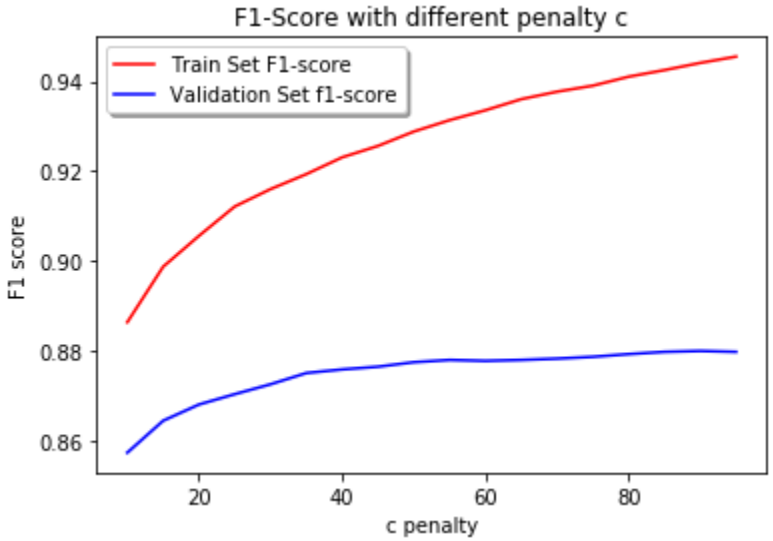
Hyperparameter: <u>penalty c</u>

c value tried at :

```
[10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95]
```

With corresponding F1 scores over validation set:

```
[0.8573720449208044, 0.8644722859377515, 0.8680793874042819, 0.8703825586770956, 0.8725822583536531, 0.8750823104059766,
0.875885253927019, 0.8764878945785476, 0.8774884728904143, 0.8779896730459266, 0.8777896561164937, 0.877990117199493,
0.8782885481694973, 0.8786890516869147, 0.8792900040052316, 0.8797893781894568, 0.8799934284401414, 0.879792686587052]
```



Best c penalty =90, and the F1 score :

F1 score:

```
F1 score for train set: 0.9441327125856955
F1 score for valid set: 0.8798932439949747
F1 score for test set: 0.8751597553131204
```

5.b 5.c

Gaussian Naïve Bayes:

```
F1 score over train set: 0.8627017402201094
F1 score over validation set: 0.7593849791768671
F1 score over test set: 0.6920485233021636
```

Decision tree: with max depth=20 |range=(5,100), min sample leaf=47|range=(2,100), max leaf nodes=102|range=(2,300)

```
F1 score for train set: 0.7685600802140135
F1 score for valid set: 0.7226150508593256
F1 score for test set: 0.7283815324385501
```

Linear SVM: with penalty c =90 range=(10,100)

```
F1 score for train set: 0.9441327125856955
F1 score for valid set: 0.8798932439949747
F1 score for test set: 0.8751597553131204
```

5.d the linear SVM classifier perform the best with IMDB data in FBoW representation. This is probably because our dataset in different group can be linearly separated.

5.e

For both BBoW representation and FBoW representation, the SVM classifiers performed well. For BBoW representation, Bayes classifier performs as well as the SVM classifier does.

5.f

It's safe to conclude that BBoW representation is since every classifier perform better on BBoW instead of on FBoW. This is probability because for documentation classifier, additional information about frequency does not help.

5.g we get such a high overall performance on IMDB sets simply because it only has 2 target classes and more training/validation data. Also due to the property of both yelp and IMDB dataset, linear SVM can always do best.