Problem 1

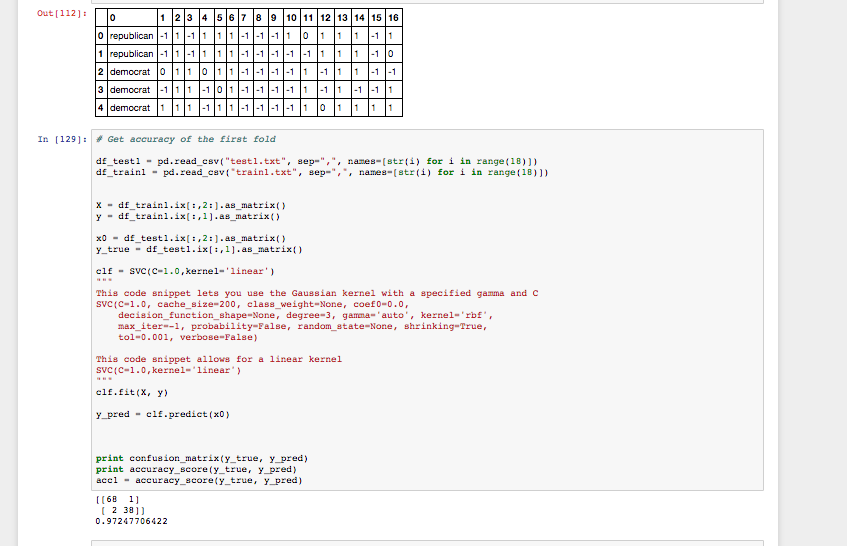
1. Opening, Converting, and Storing the Data

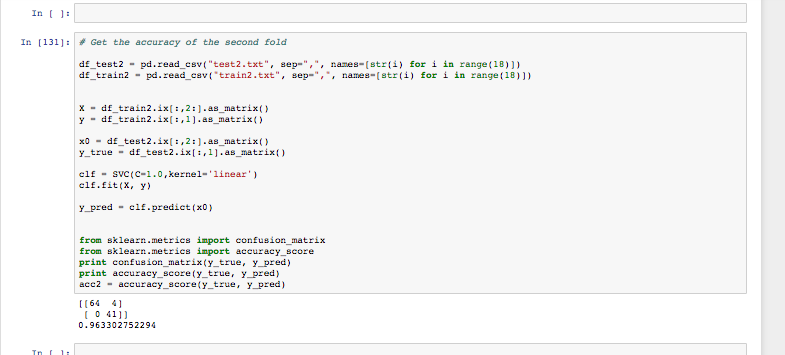
Dataset was converted into numeric and chopped into folds with the following code snippets:

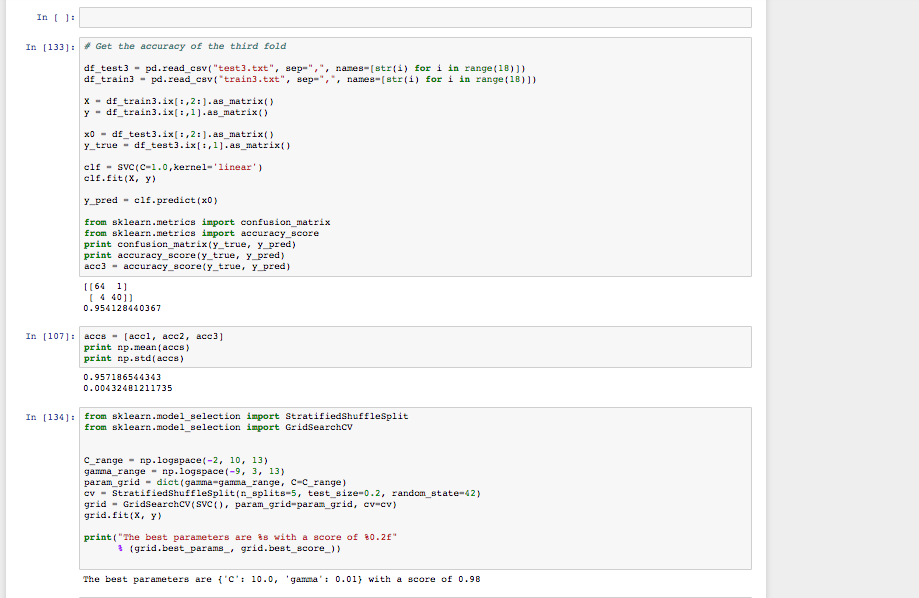
|  |
| --- |
| Snippet to get the dat and convert it to numeric  Snippet also contains how I chop the data file on a Unix system  f = open("house-votes-84.data.txt","r")  rows=[]  lines = f.readlines()  for line in lines:  arow = line.strip("\n").split(",")  for j in range(len(arow)):  if arow[j] is 'y':  arow[j] = "1"  elif arow[j] is 'n':  arow[j] = "-1"  elif arow[j] is '?':  arow[j] = "0"  rows.append(arow)  sed -n 1,327p CLEANED-house-votes-84.data.txt > train.txt  sed -n 328,435p CLEANED-house-votes-84.data.txt > validate.txt  sed -n 1,109p train.txt > test1.txt  sed -n 110,327p train.txt > train1.txt  sed -n 110,218p train.txt > test2.txt  sed -n 1,109p train.txt >> train2.txt  sed -n 219,327p train.txt >> train2.txt  sed -n 219,327p train.txt > test3.txt  sed -n 1,218p train.txt > train3.txt |

B) Using Linear Kernel

* Three folds mean: 0.963302752294
* Three folds standard deviation: 0.00749079432044
* Best C: 10.0
* Best Gamma: 0.01







|  |
| --- |
| I used the following snippet to tune the gamma value and the C value for each fold  from sklearn.model\_selection import StratifiedShuffleSplit  from sklearn.model\_selection import GridSearchCV  C\_range = np.logspace(-2, 10, 13)  gamma\_range = np.logspace(-9, 3, 13)  param\_grid = dict(gamma=gamma\_range, C=C\_range)  cv = StratifiedShuffleSplit(n\_splits=5, test\_size=0.2, random\_state=42)  grid = GridSearchCV(SVC(), param\_grid=param\_grid, cv=cv)  grid.fit(X, y)  print("The best parameters are %s with a score of %0.2f"  % (grid.best\_params\_, grid.best\_score\_))  Fold1  OUTPUT:  The best parameters are {'C': 10.0, 'gamma': 0.10000000000000001} with a score of 0.98  Fold2  OUTPUT:  The best parameters are {'C': 1.0, 'gamma': 0.10000000000000001} with a score of 0.97  Fold3  OUTPUT:  The best parameters are {'C': 10.0, 'gamma': 0.01} with a score of 0.98 |

C) Using Gaussian Kernel

* Three folds mean: 0.957186544343
* Three folds standard deviation: 0.00432481211735
* Best C: 10.0
* Best Gamma 0.10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # First Fold  df\_test1 = pd.read\_csv("test1.txt", sep=",", names=[str(i) for i in range(18)])  df\_train1 = pd.read\_csv("train1.txt", sep=",", names=[str(i) for i in range(18)])  X = df\_train1.ix[:,2:].as\_matrix()  y = df\_train1.ix[:,1].as\_matrix()  x0 = df\_test1.ix[:,2:].as\_matrix()  y\_true = df\_test1.ix[:,1].as\_matrix()  clf = SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,  decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',  max\_iter=-1, probability=False, random\_state=None, shrinking=True,  tol=0.001, verbose=False)  """  This code snippet lets you use the Gaussian kernel with a specified gamma and C  SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,  decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',  max\_iter=-1, probability=False, random\_state=None, shrinking=True,  tol=0.001, verbose=False)  """  clf.fit(X, y)  y\_pred = clf.predict(x0)  print confusion\_matrix(y\_true, y\_pred)  print accuracy\_score(y\_true, y\_pred)  acc1 = accuracy\_score(y\_true, y\_pred)  OUTPUT:  [[66 3]  [ 2 38]]  0.954128440367 | | # Second Fold  df\_test2 = pd.read\_csv("test2.txt", sep=",", names=[str(i) for i in range(18)])  df\_train2 = pd.read\_csv("train2.txt", sep=",", names=[str(i) for i in range(18)])  X = df\_train2.ix[:,2:].as\_matrix()  y = df\_train2.ix[:,1].as\_matrix()  x0 = df\_test2.ix[:,2:].as\_matrix()  y\_true = df\_test2.ix[:,1].as\_matrix()  clf = SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,  decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',  max\_iter=-1, probability=False, random\_state=None, shrinking=True,  tol=0.001, verbose=False)  clf.fit(X, y)  y\_pred = clf.predict(x0)  from sklearn.metrics import confusion\_matrix  from sklearn.metrics import accuracy\_score  print confusion\_matrix(y\_true, y\_pred)  print accuracy\_score(y\_true, y\_pred)  acc2 = accuracy\_score(y\_true, y\_pred)  OUTPUT:  [[64 4]  [ 1 40]]  0.954128440367 | # Third Fold  df\_test3 = pd.read\_csv("test3.txt", sep=",", names=[str(i) for i in range(18)])  df\_train3 = pd.read\_csv("train3.txt", sep=",", names=[str(i) for i in range(18)])  X = df\_train3.ix[:,2:].as\_matrix()  y = df\_train3.ix[:,1].as\_matrix()  x0 = df\_test3.ix[:,2:].as\_matrix()  y\_true = df\_test3.ix[:,1].as\_matrix()  clf = SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,  decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',  max\_iter=-1, probability=False, random\_state=None, shrinking=True,  tol=0.001, verbose=False)  clf.fit(X, y)  y\_pred = clf.predict(x0)  from sklearn.metrics import confusion\_matrix  from sklearn.metrics import accuracy\_score  print confusion\_matrix(y\_true, y\_pred)  print accuracy\_score(y\_true, y\_pred)  acc3 = accuracy\_score(y\_true, y\_pred)  OUTPUT:  [[65 0]  [ 4 40]]  0.963302752294 | |
| accs = [acc1, acc2, acc3]  print np.mean(accs)  print np.std(accs)  OUTPUT:  0.957186544343  0.00432481211735 | | |

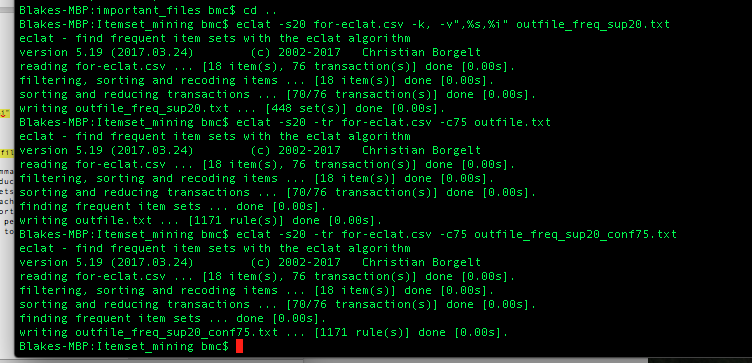
|  |
| --- |
| I used the following snippet to tune the gamma value and the C value for each fold  from sklearn.model\_selection import StratifiedShuffleSplit  from sklearn.model\_selection import GridSearchCV  C\_range = np.logspace(-2, 10, 13)  gamma\_range = np.logspace(-9, 3, 13)  param\_grid = dict(gamma=gamma\_range, C=C\_range)  cv = StratifiedShuffleSplit(n\_splits=5, test\_size=0.2, random\_state=42)  grid = GridSearchCV(SVC(), param\_grid=param\_grid, cv=cv)  grid.fit(X, y)  print("The best parameters are %s with a score of %0.2f"  % (grid.best\_params\_, grid.best\_score\_))  Fold1  OUTPUT:  The best parameters are {'C': 10.0, 'gamma': 0.10000000000000001} with a score of 0.98  Fold2  OUTPUT:  The best parameters are {'C': 1.0, 'gamma': 0.10000000000000001} with a score of 0.97  Fold3  OUTPUT:  The best parameters are {'C': 10.0, 'gamma': 0.01} with a score of 0.98 |

Problem 2

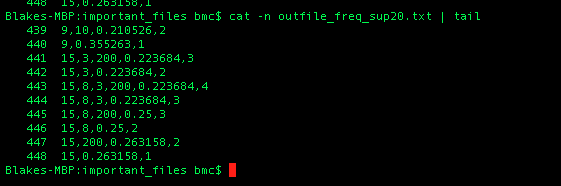
Commands

eclat -s20 for-eclat.csv -k, -v",%s,%i" outfile\_freq\_sup20.txt

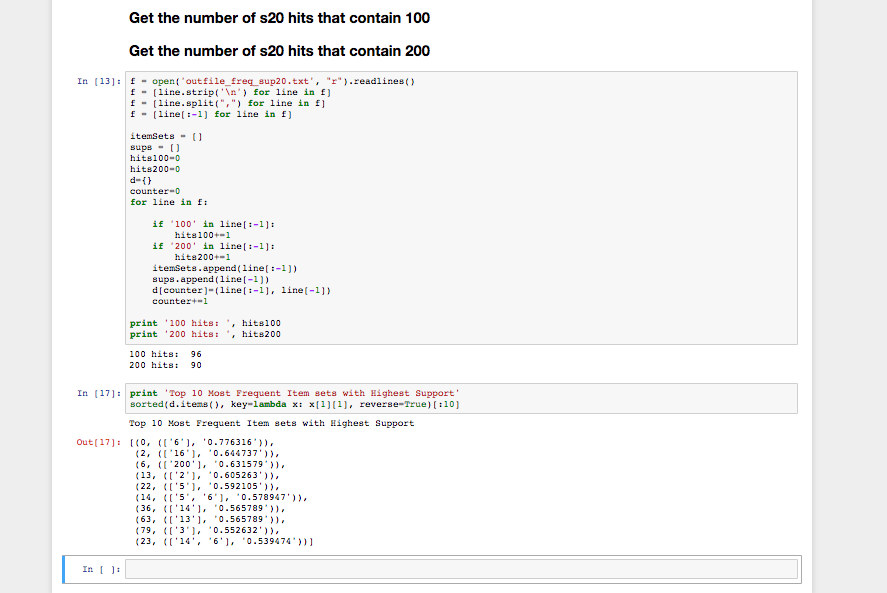
eclat -s20 -tr for-eclat.csv -c75 outfile.txt



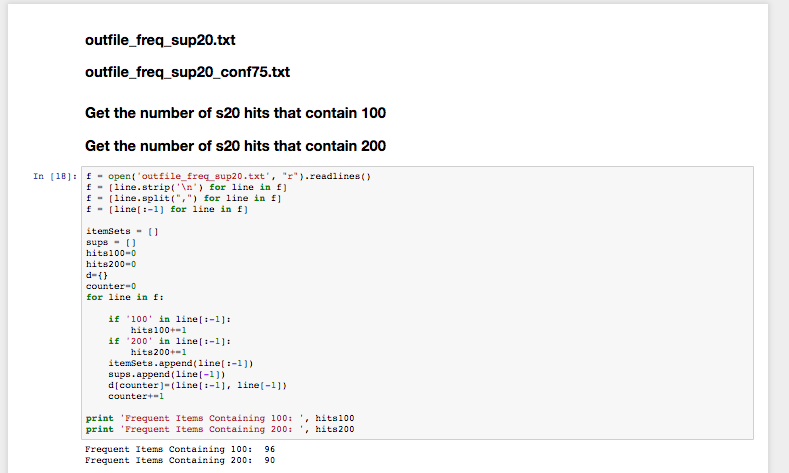
1. Number of frequent item sets with > 20% support: 448



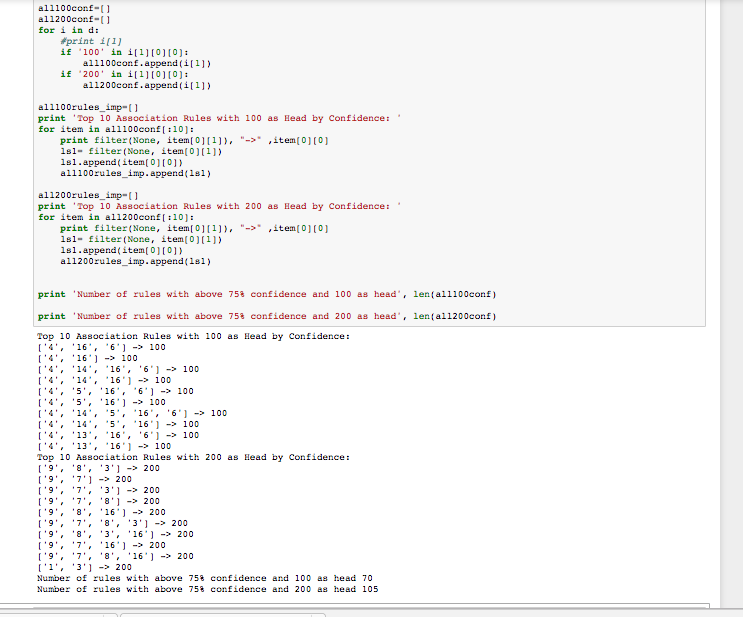
1. Top 10 item sets with highest support



1. Number of frequent items containing 100: 96
2. Number of frequent items containing 200: 90



(E,F,G,H) in the following image:



1. Top 10 frequent item sets with 100 as head:

['4', '16', '6'] -> 100

['4', '16'] -> 100

['4', '14', '16', '6'] -> 100

['4', '14', '16'] -> 100

['4', '5', '16', '6'] -> 100

['4', '5', '16'] -> 100

['4', '14', '5', '16', '6'] -> 100

['4', '14', '5', '16'] -> 100

['4', '13', '16', '6'] -> 100

['4', '13', '16'] -> 100

1. Rules with head 100 which the confidence value is more than 75%: 70
2. Top 10 frequent item sets with 200 as head:

['9', '8', '3'] -> 200

['9', '7'] -> 200

['9', '7', '3'] -> 200

['9', '7', '8'] -> 200

['9', '8', '16'] -> 200

['9', '7', '8', '3'] -> 200

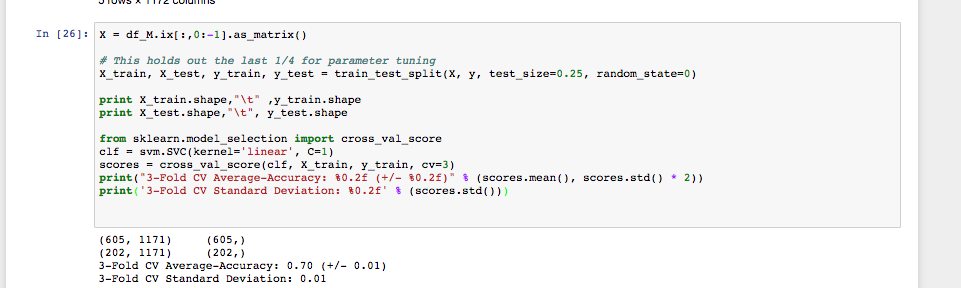
['9', '8', '3', '16'] -> 200

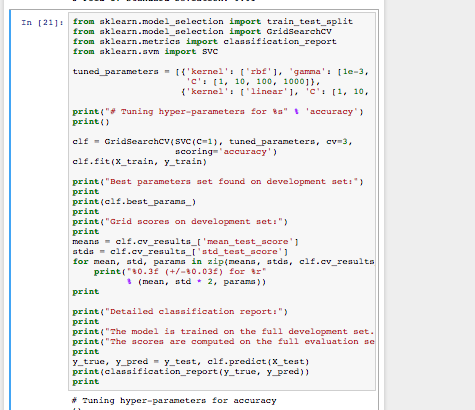
['9', '7', '16'] -> 200

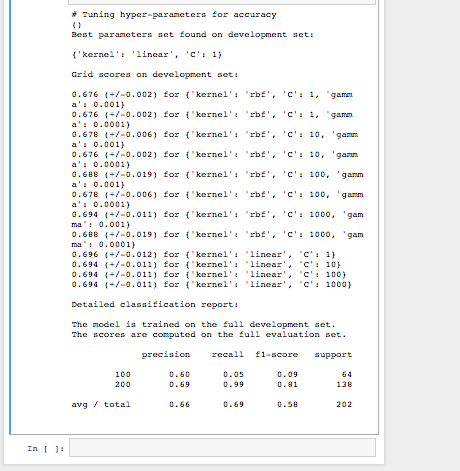
['9', '7', '8', '16'] -> 200

['1', '3'] -> 200

1. Rules with head 200 which the confidence value is more than 75%: 105
2. The following images reflect my findings:
   1. 3-Fold CV Average-Accuracy: 0.70 (+/- 0.01)
   2. 3-Fold CV Standard Deviation: 0.01
   3. Highest Accuracy for Parameters was at: C=1000, Gamma=0.0001







Problem 3)

* The 10 best feature body’s (columns) in terms of odds-ratio:

['4', '5', '16']

['4', '13', '16', '6']

['4', '13', '16']

['4', '14', '5', '16', '6']

['4', '14', '5', '16']

['9', '7', '8', '3']

['9', '8', '3', '16']

['9', '7', '16']

['9', '7', '8', '16']

['1', '3']

Problem 4) FP-Growth by hand