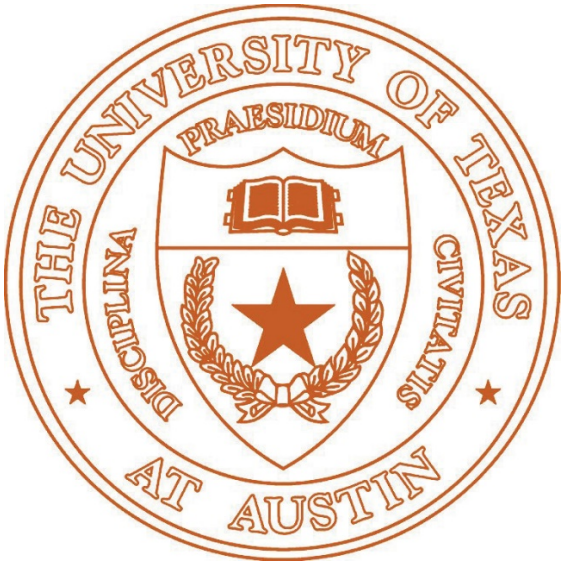
University of Texas at Austin, Cockrell School of Engineering
Data Mining – EE 380L



Problem Set # 3

April 11, 2016

Gabrielson Eapen
EID: EAPENGP

Discussed Homework with Following Students:
1. Mudra Gandhi
2. Rayo Landeros

Q1]

In [1]:
```
# Name: Gabe Eapen
# UT EID: eapengp
# PS3 - Q1
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import datasets, linear_model
from pandas import DataFrame, Series
import seaborn as sns
sns.set(style='ticks', palette='Set2')
```

In [3]:
```
def extract_int(some_string):
    int_as_string = (str(some_string)).split('.')[0]
    return int(int_as_string)
```

In [4]:
```
df=pd.read_stata("nes5200_processed_voters_realideo.dta")
df.shape
```

Out[4]:
```
(41498, 62)
```

In [5]:
```
print(df.columns)
```

```
Index([u'year', u'resid', u'weight1', u'weight2', u'weight3', u'age',
       u'gender', u'race', u'educ1', u'urban', u'region', u'income', u'occup1',
       u'union', u'religion', u'educ2', u'educ3', u'martial_status', u'occup2',
       u'icpsr_cty', u'fips_cty', u'partyid7', u'partyid3', u'partyid3_b',
       u'str_partyid', u'father_party', u'mother_party', u'dlikes', u'rlikes',
       u'dem_therm', u'rep_therm', u'regis', u'vote', u'regisvote',
       u'presvote', u'presvote_2party', u'presvote_intent', u'ideo_feel',
       u'ideo7', u'ideo', u'cd', u'state', u'inter_pre', u'inter_post',
       u'black', u'female', u'age_sq', u'rep_presvote', u'rep_pres_intent',
       u'south', u'real_ideo', u'presapprov', u'perfin1', u'perfin2',
       u'perfin', u'presadm', u'age_10', u'age_sq_10', u'newfathe', u'newmoth',
       u'parent_party', u'white'],
      dtype='object')
```

In [6]:
```
df_1992_raw = df[(df['year'] == 1992.0) & ((df['presvote'] == "1. democrat") | (df['presvote'] =
print df_1992_raw.shape
df_ed1992 = df.loc[(df['year'] == 1992.0) & ((df['presvote'] == "1. democrat") | (df['presvote']
print df_ed1992.shape
```

```
(1304, 62)
(1304, 62)
```

Part a]

In [7]:
```
df_vote_inc = pd.DataFrame(df_ed1992,columns=['presvote','income'])
print df_vote_inc.shape
print df_vote_inc.head()
df_clean = df_vote_inc.dropna(how='any')
print df_clean.shape
#print df_clean.dtypes
print df_clean.head()
```

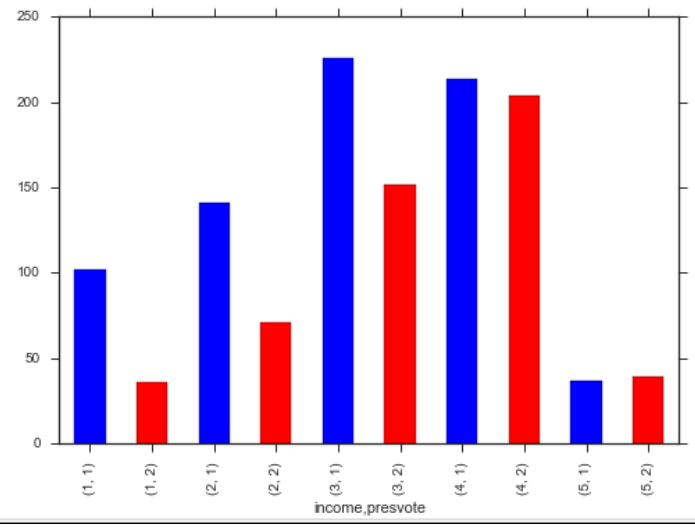
```
(1304, 2)
            presvote                  income
32092  2. republican  4. 68 to 95 percentile
32093  2. republican  2. 17 to 33 percentile
32095    1. democrat   1. 0 to 16 percentile
32096  2. republican  2. 17 to 33 percentile
32097    1. democrat  3. 34 to 67 percentile
(1222, 2)
presvote    category
income      category
dtype: object
            presvote                  income
32092  2. republican  4. 68 to 95 percentile
32093  2. republican  2. 17 to 33 percentile
32095    1. democrat   1. 0 to 16 percentile
32096  2. republican  2. 17 to 33 percentile
32097    1. democrat  3. 34 to 67 percentile
```

In [8]:
```
cat_columns = df_clean.select_dtypes(['category']).columns
#cat_columns
```

In [9]:
```
df_clean[cat_columns] = df_clean[cat_columns].apply(lambda x: x.cat.codes + 1)
#print vote_D.head()
#df_clean.head()
```

In [10]:
```
df_clean.groupby(['income','presvote']).size().plot(kind='bar',color=['blue','red'])
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0xa1db358>



```
In [11]: vote_D = df_clean.loc[(df_clean['presvote'] == 1) ]
         vote_R = df_clean.loc[(df_clean['presvote'] == 2) ]
         print vote_D.shape
         print vote_R.shape

         (720, 2)
         (502, 2)

In [13]: df_clean['presvote'].values

Out[13]: array([2, 2, 1, ..., 2, 1, 2], dtype=int8)
```

Part b]

```
In [14]: X = df_clean['income'].values
         Y = df_clean['presvote'].values
         # Reshape as X has only single feature
         #X.reshape(-1, 1)
         print X.shape
         print Y.shape
         # create the logistic regression object
         logreg = linear_model.LogisticRegression(fit_intercept=True,solver='liblinear',penalty='l1')
         logreg.fit(X.reshape(-1, 1), Y)

         (1222L,)
         (1222L,)

Out[14]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l1', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)

In [16]: print "Coeff:",logreg.coef_
         print "Intercept (B0)",logreg.intercept_

         Coeff: [[ 0.29072854]]
         Intercept (B0) [-1.25875036]
```

Part c]

```
In [56]: # Part C #

In [57]: print "Probability for Republican Inc level 2", logreg.predict_proba(2)[0][1]
         print "Probability for Republican Inc level 3", logreg.predict_proba(3)[0][1]

         Probability for Republican Inc level 2 0.336865682301
         Probability for Republican Inc level 3 0.404544542996

In [58]: XRepInc_2 = vote_R.loc[(vote_R['income'] == 2) ].values
         XRepInc_3 = vote_R.loc[(vote_R['income'] == 3) ].values

In [63]: print logreg.predict_proba(XRepInc_2.reshape(-1,1))

         [[ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
          [ 0.66313432  0.33686568]
```

```
In [64]: print logreg.predict_proba(XRepInc_3.reshape(-1,1))
```

```
[[ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
 [ 0.66313432  0.33686568]
 [ 0.59545546  0.40454454]
```

Q2]

```
In [1]: # Name: Gabe Eapen
        # UT EID: eapengp
        # PS3 - Q2
```

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        from sklearn import cross_validation
        from sklearn import datasets
        from sklearn import svm
```

```
In [3]: # Use only cars2010
        df=pd.read_stata("cars2010.dta")
```

```
In [4]: df['FE'].values

Out[4]: array([ 28.0198,  25.6094,  26.8   , ...,  30.4926,  29.7431,  26.2   ])
```

```
In [5]: #df.as_matrix(["FE"])
        X = df.as_matrix(['EngDispl','NumCyl'])
        y = df.as_matrix(['FE'])
        #y1 = y * 1000

In [8]: kf = cross_validation.KFold(len(X), n_folds=2)
        print kf

        sklearn.cross_validation.KFold(n=1107, n_folds=2, shuffle=False, random_state=None)

In [16]: for trn_idx, tst_idx in kf:
             X_trn, X_tst = X[trn_idx],X[tst_idx]
             y_trn, y_tst = y[trn_idx],y[tst_idx]

         print len(trn_idx), len(tst_idx)

         554 553

In [17]: X_trn = np.array(X_trn)
         y_trn = np.array(y_trn)
         clf = svm.SVC(kernel='linear',C=1).fit(X_trn, y_trn.astype(int))
         print "CLF Score:", clf.score(X_tst,y_tst.astype(int))

         CLF Score: 0.0759493670886
```

Here we split the cars2010 dataset with the cross_validation.kfold routine. Using parameter n_fold=2, denotes to split data into 2 sets. There was a total of 1107 samples and the kfold split resulted into a training set of 554 samples and a testing dataset of 553 samples which is roughly 50% each. In previous assignment we were using separate datasets as training and testing unlike here. I tried with n_fold= 4 which generates a larger training dataset and smaller testing dataset. The CLF score value decreased

```
In [21]: kf2 = cross_validation.KFold(len(X), n_folds=4)
         print kf2

         for trn_idx2, tst_idx2 in kf2:
             X_trn2, X_tst2 = X[trn_idx2],X[tst_idx2]
             y_trn2, y_tst2 = y[trn_idx2],y[tst_idx2]

         print len(trn_idx2), len(tst_idx2)

         sklearn.cross_validation.KFold(n=1107, n_folds=4, shuffle=False, random_state=None)
         831 276

In [22]: X_trn2 = np.array(X_trn2)
         y_trn2 = np.array(y_trn2)
         clf2 = svm.SVC(kernel='linear',C=1).fit(X_trn2, y_trn2.astype(int))
         print "CLF Score:", clf.score(X_tst2,y_tst2.astype(int))

         CLF Score: 0.054347826087
```