ML HW3 Zoe Zhou

October 17, 2018

1. Data Processing:

a) Import the data: Only keep numeric data (pandas has tools to do this!). Drop "PHONE" and "COUNTRY_SSA" as well.

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.decomposition import PCA
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import confusion_matrix, classification_report
```

```
In [2]: provider=pd.read_csv('/Users/zoezhou/Desktop/UCHICAGO FALL2018/Machine Learning/Provide
```

```
In [3]: newprovider=provider._get_numeric_data()
```

```
In [4]: newprovider.drop(["PHONE","COUNTY_SSA"],axis=1,inplace=True)
```

```
In [5]: newprovider.head(5)
```

```
Out[5]:        ZIP  BEDCERT  RESTOT  OVERALL_RATING  SURVEY_RATING  QUALITY_RATING  \
        0  35653.0     57.0    51.5             5.0            5.0             5.0
        1  35150.0     85.0    74.2             3.0            3.0             5.0
        2  35768.0     50.0     NaN             1.0            2.0             2.0
        3  35206.0     92.0    79.8             2.0            2.0             4.0
        4  35111.0    103.0    98.1             3.0            3.0             4.0

           STAFFING_RATING  RN_STAFFING_RATING   AIDHRD   VOCHRD       ...        \
        0              4.0                 4.0  3.43572  1.16495       ...
        1              1.0                 1.0      NaN      NaN       ...
        2              1.0                 1.0      NaN      NaN       ...
        3              3.0                 3.0  2.32722  0.82104       ...
        4              3.0                 2.0  2.33617  0.92407       ...

           ADJ_AIDE  ADJ_LPN   ADJ_RN  ADJ_TOTAL  INCIDENT_CNT  CMPLNT_CNT  FINE_CNT  \
        0   3.11741  1.24750  0.83853    5.13047           0.0         0.0       0.0
        1       NaN      NaN      NaN        NaN           0.0         0.0       1.0
```

```
2        NaN      NaN      NaN         NaN            0.0          0.0        0.0
3    2.40074  0.86962  0.56463     3.83026            0.0          1.0        0.0
4    2.55126  1.08955  0.30360     3.95709            0.0          0.0        0.0

   FINE_TOT  PAYDEN_CNT  TOT_PENLTY_CNT
0       0.0         0.0             0.0
1   15259.0         1.0             2.0
2       0.0         0.0             0.0
3       0.0         0.0             0.0
4       0.0         0.0             0.0

[5 rows x 28 columns]
```

b) This data is extra messy and has some NaN and NaT values. NaT values should be replaced by "np.nan." After this step, remove any rows that have an NaN value.

```
In [6]: newprovider.replace(["NaN", 'NaT'], np.nan, inplace = True)
```

```
In [7]: cleaned_df= newprovider.dropna(how='any', axis = 0)
```

```
In [8]: cleaned_df.head(5)
```

```
Out[8]:        ZIP  BEDCERT  RESTOT  OVERALL_RATING  SURVEY_RATING  QUALITY_RATING  \
        0  35653.0     57.0    51.5             5.0            5.0             5.0
        3  35206.0     92.0    79.8             2.0            2.0             4.0
        4  35111.0    103.0    98.1             3.0            3.0             4.0
        5  35611.0    149.0   119.7             5.0            3.0             5.0
        6  36025.0    124.0    96.0             5.0            4.0             5.0

           STAFFING_RATING  RN_STAFFING_RATING   AIDHRD   VOCHRD       ...        \
        0              4.0                 4.0  3.43572  1.16495       ...
        3              3.0                 3.0  2.32722  0.82104       ...
        4              3.0                 2.0  2.33617  0.92407       ...
        5              4.0                 3.0  2.57869  1.01443       ...
        6              3.0                 4.0  1.99985  0.62768       ...

           ADJ_AIDE  ADJ_LPN   ADJ_RN  ADJ_TOTAL  INCIDENT_CNT  CMPLNT_CNT  FINE_CNT  \
        0   3.11741  1.24750  0.83853    5.13047           0.0         0.0       0.0
        3   2.40074  0.86962  0.56463    3.83026           0.0         1.0       0.0
        4   2.55126  1.08955  0.30360    3.95709           0.0         0.0       0.0
        5   2.56783  1.04823  0.46444    4.07866           0.0         1.0       0.0
        6   2.12102  0.70311  0.75448    3.52979           1.0         1.0       0.0

           FINE_TOT  PAYDEN_CNT  TOT_PENLTY_CNT
        0       0.0         0.0             0.0
        3       0.0         0.0             0.0
        4       0.0         0.0             0.0
        5       0.0         0.0             0.0
        6       0.0         0.0             0.0
```

```
[5 rows x 28 columns]
```

c) Split into train / test set using an 80/20 split.

```
In [9]: train, test = train_test_split(cleaned_df, test_size=0.2)

In [10]: X_train=train.loc[:,train.columns !="OVERALL_RATING"]

In [11]: Y_train=train[["OVERALL_RATING"]]

In [12]: Y_train = np.asarray(Y_train).reshape((len(Y_train),1))

In [13]: X_test=test.loc[:,test.columns !="OVERALL_RATING"]

In [14]: Y_test=test[["OVERALL_RATING"]]

In [16]: Y_test = np.asarray(Y_test).reshape((len(Y_test),1))
```

d)Scale all input features (NOT THE TARGET VARIABLE) #Only scale X

```
In [18]: scaling_tool=StandardScaler()

In [19]: X_train_scaled=scaling_tool.fit_transform(X_train)

In [20]: X_test_scaled=scaling_tool.transform(X_test)
```

2. Model #1: Logistic Regression

```
In [21]: logisticRegr = LogisticRegression()

In [22]: logisticRegr.fit(X_train_scaled,Y_train)

/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: Futu
  FutureWarning)
/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:752: DataConve
  y = column_or_1d(y, warn=True)
/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:459: Futu
  "this warning.", FutureWarning)


Out[22]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

b)

```
In [23]: logisticRegr.score(X_train_scaled,Y_train)

Out[23]: 0.699871189351653
```

c) Calculate the confusion matrix and classification report (both are in sklearn.metrics).

```
In [24]: Y_train_pred = logisticRegr.predict(X_train_scaled)
         Y_test_pred = logisticRegr.predict(X_test_scaled)

In [25]: #Train
         confusion_matrix(Y_train,Y_train_pred)

Out[25]: array([[1076,  292,    1,    0,    0],
                [ 244, 1677,  348,   49,    0],
                [   0,  789,  305,  830,    0],
                [   0,  397,   19, 1762,  440],
                [   0,    0,    0,   86, 3330]])

In [26]: #Test
         confusion_matrix(Y_test,Y_test_pred)

Out[26]: array([[261,  69,   0,   0,   0],
                [ 85, 404,  78,  13,   0],
                [  0, 189,  57, 223,   0],
                [  0, 109,   6, 468, 117],
                [  0,   0,   0,  21, 812]])

In [27]: #Train
         print(classification_report(Y_train, Y_train_pred, target_names=['1-Rating', '2-Rating

              precision    recall  f1-score   support

    1-Rating       0.82      0.79      0.80      1369
    2-Rating       0.53      0.72      0.61      2318
    3-Rating       0.45      0.16      0.23      1924
    4-Rating       0.65      0.67      0.66      2618
    5-Rating       0.88      0.97      0.93      3416

   micro avg       0.70      0.70      0.70     11645
   macro avg       0.67      0.66      0.65     11645
weighted avg       0.68      0.70      0.67     11645



In [28]: #Test
         print(classification_report(Y_test, Y_test_pred, target_names=['1-Rating', '2-Rating'

              precision    recall  f1-score   support

    1-Rating       0.75      0.79      0.77       330
    2-Rating       0.52      0.70      0.60       580
    3-Rating       0.40      0.12      0.19       469
    4-Rating       0.65      0.67      0.66       700
```

```
   5-Rating       0.87      0.97      0.92       833

   micro avg       0.69      0.69      0.69      2912
   macro avg       0.64      0.65      0.63      2912
weighted avg       0.66      0.69      0.66      2912
```

3. Model #2: PCA(n_components = 2) + Logistic Regression

a)Pick up from step d in Problem 1 (use the same data that has been scaled): We will now transform the X_train & X_test data using PCA with 2 components.

```
In [34]: logisticRegr = LogisticRegression()
         pca_two = PCA(n_components=2)
```

```
In [43]: X_train_pca_2 = pca_two.fit_transform(X_train_scaled)
         X_test_pca_2=pca_two.transform(X_test_scaled)
```

b) Then use the transformed data (X_train_pca) to fit a Logistic Regression model.

```
In [38]: logisticRegr_pca_2 = LogisticRegression()
         logisticRegr_pca_2.fit(X_train_pca_2, Y_train)
```

```
/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: Futu
  FutureWarning)
/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:752: DataConve
  y = column_or_1d(y, warn=True)
/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:459: Futu
  "this warning.", FutureWarning)
```

```
Out[38]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)
```

c) Calculate the same error metrics as those from Model #1.

```
In [40]: logisticRegr_pca_2.score(X_train_pca_2, Y_train)
```

```
Out[40]: 0.38428510090167456
```

```
In [47]: Y_train_pred_pca_2 = logisticRegr_pca_2.predict(X_train_pca_2)
         Y_test_pred_pca_2 = logisticRegr_pca_2.predict(X_test_pca_2)
```

```
In [49]: confusion_matrix(Y_train_pred_pca_2,Y_train)
```

```
Out[49]: array([[ 548,  393,  175,   87,   42],
               [ 659,  990,  782,  624,  437],
               [   0,    3,    1,    0,    0],
               [  27,   70,   64,   75,   76],
               [ 135,  862,  902, 1832, 2861]])
```

```
In [50]: confusion_matrix(Y_test_pred_pca_2,Y_test)

Out[50]: array([[142, 120,  37,  30,   5],
                [144, 241, 174, 155, 106],
                [  2,   0,   0,   0,   1],
                [  9,  21,  17,  12,  27],
                [ 33, 198, 241, 503, 694]])
```

```
In [51]: #Train
         print(classification_report(Y_train, Y_train_pred_pca_2, target_names=['1-Rating', '2-

              precision    recall  f1-score   support

    1-Rating       0.44      0.40      0.42      1369
    2-Rating       0.28      0.43      0.34      2318
    3-Rating       0.25      0.00      0.00      1924
    4-Rating       0.24      0.03      0.05      2618
    5-Rating       0.43      0.84      0.57      3416

   micro avg       0.38      0.38      0.38     11645
   macro avg       0.33      0.34      0.28     11645
weighted avg       0.33      0.38      0.30     11645
```

```
In [52]: #Test
         print(classification_report(Y_test, Y_test_pred_pca_2, target_names=['1-Rating', '2-Ra

              precision    recall  f1-score   support

    1-Rating       0.43      0.43      0.43       330
    2-Rating       0.29      0.42      0.34       580
    3-Rating       0.00      0.00      0.00       469
    4-Rating       0.14      0.02      0.03       700
    5-Rating       0.42      0.83      0.55       833

   micro avg       0.37      0.37      0.37      2912
   macro avg       0.25      0.34      0.27      2912
weighted avg       0.26      0.37      0.28      2912
```

4. Model #3: PCA(n_components = 16) + Logistic Regression

a) Pick up from step d in Problem 1 (use the same data that has been scaled): We will now transform the X_train & X_test data using PCA with 16 components.

```
In [53]: pca_sixteen = PCA(n_components=16)
```

```
In [57]: X_train_pca_sixteen = pca_sixteen.fit_transform(X_train_scaled)
         X_test_pca_sixteen=pca_sixteen.fit_transform(X_test_scaled)
```

b) Then use the transformed data (X_train_pca) to fit a Logistic Regression model.

```
In [55]: logisticRegr_pca_16 = LogisticRegression()
         logisticRegr_pca_16.fit(X_train_pca_sixteen, Y_train)

/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432: Futu
  FutureWarning)
/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:752: DataConve
  y = column_or_1d(y, warn=True)
/Users/zoezhou/anaconda3/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:459: Futu
  "this warning.", FutureWarning)


Out[55]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

c)Calculate the same error metrics as those from Model #1.

```
In [56]: logisticRegr_pca_16.score(X_train_pca_sixteen, Y_train)

Out[56]: 0.693774151996565

In [58]: Y_train_pred_pca_16 = logisticRegr_pca_16.predict(X_train_pca_sixteen)
         Y_test_pred_pca_16 = logisticRegr_pca_16.predict(X_test_pca_sixteen)

In [59]: #Train
         confusion_matrix(Y_train_pred_pca_16,Y_train)

Out[59]: array([[1072,  252,    0,    0,    0],
               [ 296, 1680,  792,  397,    0],
               [   1,  340,  287,   11,    0],
               [   0,   46,  845, 1709,   85],
               [   0,    0,    0,  501, 3331]])

In [60]: #Test
         confusion_matrix(Y_test_pred_pca_16,Y_test)

Out[60]: array([[ 54, 103,  37,  21,  24],
               [ 99, 133,  77,  88, 104],
               [ 33,  76,  49,  73,  38],
               [ 66, 142, 163, 309, 444],
               [ 78, 126, 143, 209, 223]])

In [61]: #Train
         print(classification_report(Y_train, Y_train_pred_pca_16, target_names=['1-Rating', '2
```

```
              precision    recall  f1-score   support

    1-Rating       0.81      0.78      0.80      1369
    2-Rating       0.53      0.72      0.61      2318
    3-Rating       0.45      0.15      0.22      1924
    4-Rating       0.64      0.65      0.64      2618
    5-Rating       0.87      0.98      0.92      3416

   micro avg       0.69      0.69      0.69     11645
   macro avg       0.66      0.66      0.64     11645
weighted avg       0.67      0.69      0.67     11645
```

```
In [62]: #Test
         print(classification_report(Y_test, Y_test_pred_pca_16, target_names=['1-Rating', '2-R
```

```
              precision    recall  f1-score   support

    1-Rating       0.23      0.16      0.19       330
    2-Rating       0.27      0.23      0.25       580
    3-Rating       0.18      0.10      0.13       469
    4-Rating       0.27      0.44      0.34       700
    5-Rating       0.29      0.27      0.28       833

   micro avg       0.26      0.26      0.26      2912
   macro avg       0.25      0.24      0.24      2912
weighted avg       0.26      0.26      0.25      2912
```

5. Between Model #2 and Model #3, which performed the best?

Overall, logistic regression performs the best.