Machine Learning, Fall 2019: Project 3

Liam Li G48502460

Header:

Python 3, sublime text 3

Modules: matplotlib, numpy, sklearn(SVC, linearSVC, RandomForest, learning_curve, classification_report), statistics(for modes), info_gain, csv

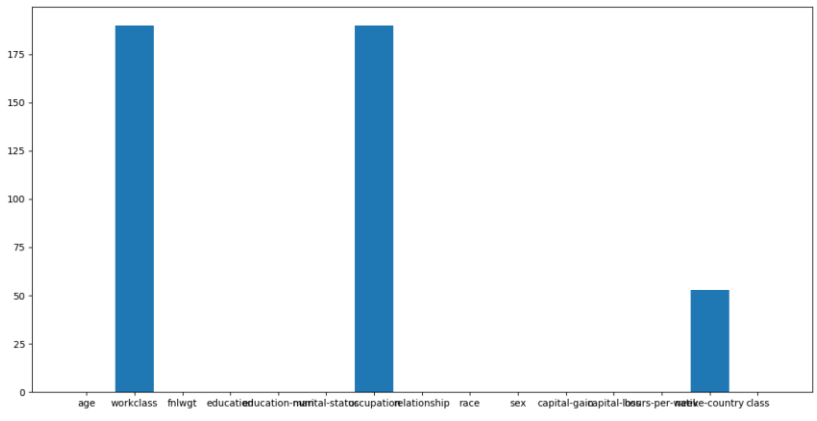
Dataset preprocessing

1. Missing features

a. Checking missing features

```
18  def showMissing(data):
19      nums=[0 for i in range(len(data[0]))]
20      columns=[]
21      for row in data:
22          for index, attr in enumerate(row):
23              if attr=='?':
24                  nums[index]+=1
25                  if index not in columns:
26                      columns.append(index)
27      print(columns)
28      plt.bar(titles,nums)
29      plt.show()
```

```
[13, 1, 6]
[Finished in 12.1s]
```



b. Replacing missing values with their modes.

```
def replaceMissing(data):
    # find modes for each attributes
    columns=[1,6,13]
    modes={}
    for index in columns:
        modes[index]=(statistics.mode([x[index] for x in data]))
    # replace missisng with modes
    for row in data:
        for index in columns:
            if row[index] =='?':
                row[index]=modes[index]
    return data
```

2. **Label Encoding**

It converts the nominal values into continuous number so the classifer can handle the data to train and test.

For some specific features like race and native country, united states and white are seen as 0 and other values are represented by 1 because of the small size of other values.

Work class	{'State-gov': 0, 'Self-emp-not-inc': 1, 'Local-gov': 2, 'Self-emp-inc': 3, 'Private': 4, 'Federal-gov': 5, 'Without-pay': 6}
Education	{'Bachelors': 0, 'HS-grad': 1, 'Assoc-acdm': 2, 'Some-college': 3, '7th-8th': 4, 'Assoc-voc': 5, '5th-6th': 6, '11th': 7, '9th': 8, 'Masters': 9, '12th': 10, '10th': 11, 'Prof-school': 12, 'Doctorate': 13, '1st-4th': 14, 'Preschool': 15}
Marital status	{'Never-married': 0, 'Married-civ-spouse': 1, 'Divorced': 2, 'Widowed': 3, 'Separated': 4, 'Married-spouse-absent': 5, 'Married-AF-spouse': 6}
Occupation	{'Adm-clerical': 0, 'Farming-fishing': 1, 'Protective-serv': 2, 'Prof-specialty': 3, 'Exec-managerial': 4, 'Machine-op-inspct': 5, 'Tech-support': 6, 'Other-service': 7, 'Craft-repair': 8, 'Transport-moving': 9, 'Sales': 10, 'Handlers-cleaners': 11, 'Priv-house-serv': 12}
Relationship	{'Not-in-family': 0, 'Own-child': 1, 'Husband': 2, 'Unmarried': 3, 'Other-relative': 4, 'Wife': 5}
Race	{'White': 0, 'Black': 1, 'Other': 1, 'Asian-Pac-Islander': 1, 'Amer-Indian-Eskimo': 1}
Sex	{'Male': 0, 'Female': 1}
Country	{'United-States': 0,'Others': 1}
Class	{'<=50K': 0, '>50K': 1}

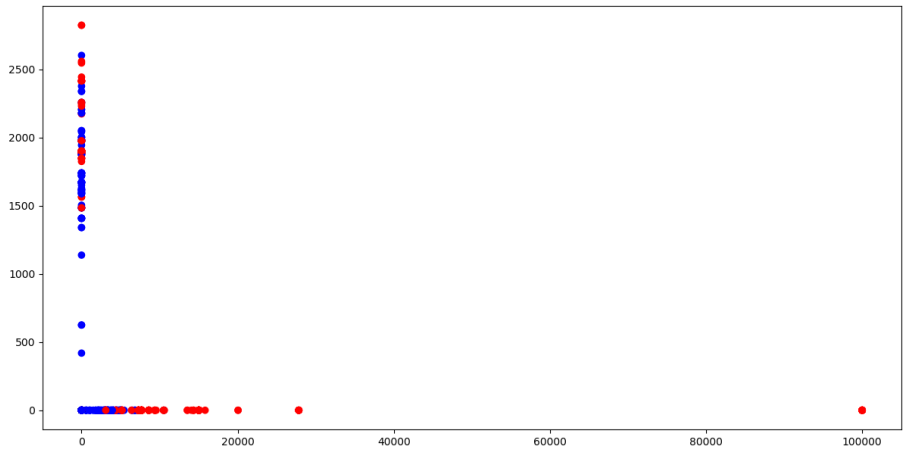
3. 10-fold-cross validation

```
def crossValidation(data):
    k=int(len(data)/10)
    y_train=[]
    y_test=[]
    y_tes=[]
    for i in range(10):
        # splitting data set
        training=data[:]
        validation=training[i*k:i*k+k]
        del training[i*k:i*k+k]

        y_train.extend(training)
        y_test.extend(validation)

    return y_train,y_test
```

4. Information gain

The two highest information gain features are capital gain and capital loss.

```
age 0.013876598032084228
workclass 0.010082958893954325
fnlwgt 0.04521579283127425
education 0.021977760107035445
education-num 0.021977760107035477
marital-status 0.0622378848062946
occupation 0.01920568626582297
relationship 0.05447460106576111
race 0.0066887238763815365
sex 0.02730496405460196
capital-gain 0.09313271554174522
capital-loss 0.07200042456868828
hours-per-week 0.014648950817164661
native-country 0.014266215563705501
[Finished in 6.4s]
```

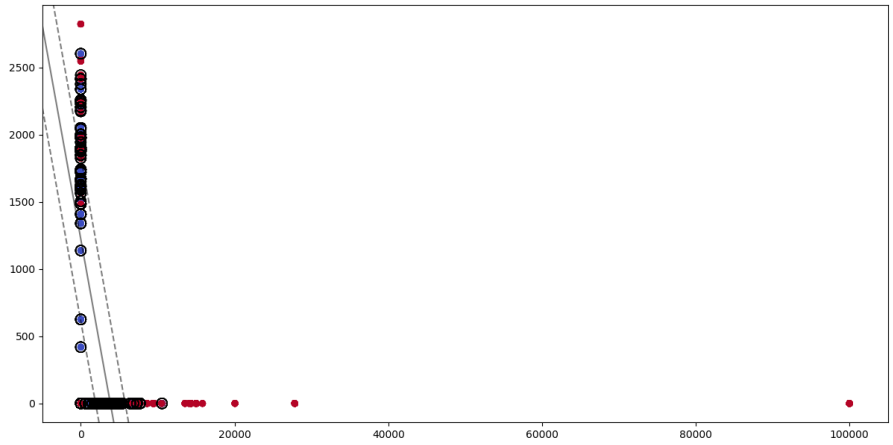


Linear soft margin SVM

1. **Boundary visualization**

   clf = SVC(C=1.0,gamma='scale',kernel=linear)

   Support vectors are circled points.



   Some of the support vectors:

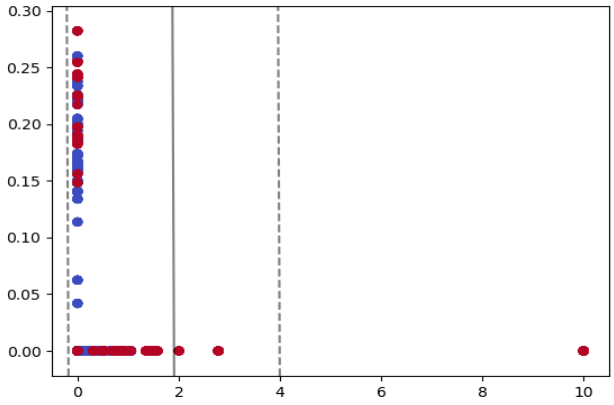
```
[[3674.    0.]
 [6849.    0.]
 [5013.    0.]
 ...
 [   0.    0.]
 [   0.    0.]
 [   0. 1977.]]
[Finished in 6.3s]
```

2. **Margins with different C**

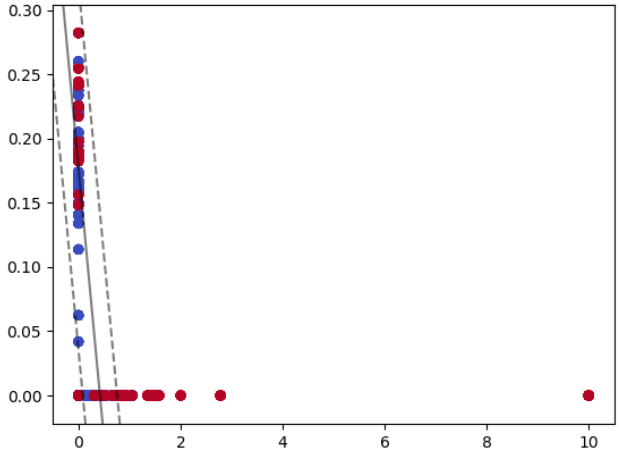
   When I change the values of C from 10^-5 to 10^5 exponentially, the performance changes as follow. As we can see, the accuracy starts merging when C=0.1.

```
C= 1e-05 accuracy:0.7652307692307693
C= 0.0001 accuracy:0.768
C= 0.001 accuracy:0.7972307692307692
C= 0.01 accuracy:0.7956923076923077
C= 0.1 accuracy:0.8006153846153846
C= 1 accuracy:0.8086153846153846
C= 10 accuracy:0.8086153846153846
C= 100 accuracy:0.8086153846153846
C= 1000 accuracy:0.8086153846153846
C= 10000 accuracy:0.8086153846153846
[Finished in 5.9s]
```

   Decision boundary and margins when C=1e-4.



Decision boundary and margins when C=1.0.



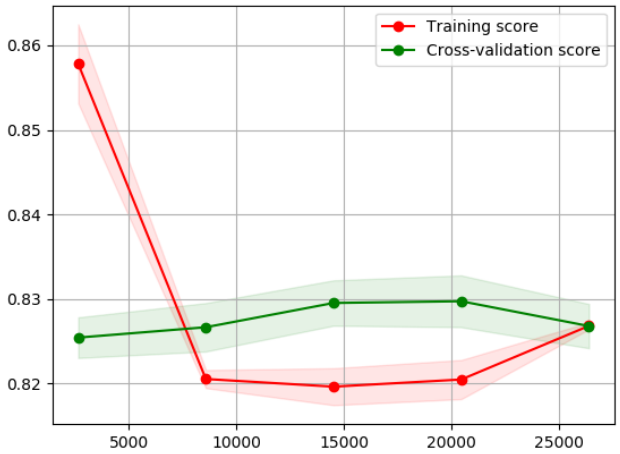
Effect: the smaller C brings larger margin and more misclassifications.

3. SVM with all features

Since we already know that C is convergence, I iteration different c values exponentially to get the optimal C for all features.

```
C= 1e-05 accuracy:0.7615384615384615
C= 0.0001 accuracy:0.7732307692307693
C= 0.001 accuracy:0.8126153846153846
C= 0.01 accuracy:0.8276923076923077
C= 0.1 accuracy:0.8273846153846154
C= 1 accuracy:0.8276923076923077
C= 10 accuracy:0.827076923076923
C= 100 accuracy:0.8276923076923077
C= 1000 accuracy:0.8276923076923077
C= 10000 accuracy:0.827076923076923
[Finished in 5.5s]
```

Learning curve when C=10000



SVM WITH KERNEL

1. Performance comparison

Linear:clf = SVC(C=1.0, gamma='scale',kernel='linear')

```
              precision    recall  f1-score   support

           0       0.81      0.95      0.88      2472
           1       0.66      0.28      0.39       778

    accuracy                           0.79      3250
   macro avg       0.74      0.62      0.64      3250
weighted avg       0.77      0.79      0.76      3250

0.7935384615384615
[Finished in 8.5s]
```

Polynomial: clf = SVC(C=1.0, gamma='scale',kernel='poly',degree=3)

```
              precision    recall  f1-score   support

           0       0.79      1.00      0.88      2472
           1       0.95      0.16      0.28       778

    accuracy                           0.80      3250
   macro avg       0.87      0.58      0.58      3250
weighted avg       0.83      0.80      0.74      3250

0.7972307692307692
[Finished in 10.5s]
```

RBF: clf = SVC(C=1.0, gamma='auto',kernel='rbf')

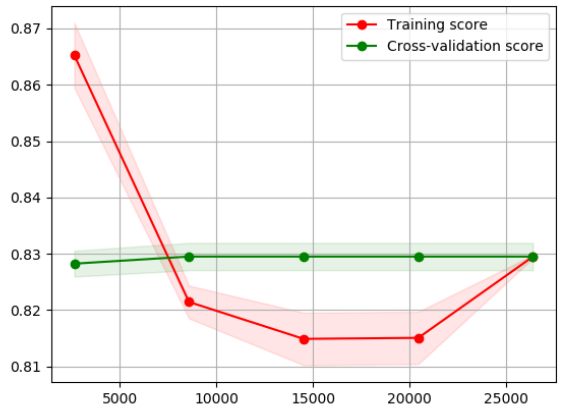
```
              precision    recall  f1-score   support

           0       0.81      0.97      0.88      2472
           1       0.72      0.25      0.38       778

    accuracy                           0.80      3250
   macro avg       0.76      0.61      0.63      3250
weighted avg       0.79      0.80      0.76      3250

0.7981538461538461
[Finished in 11.4s]
```

2. Better performance

Random forest: clf=RandomForestClassifier(criterion='entropy').

```
clf=RandomForestClassifier(criterion='entropy',n_estimators=10)
clf.fit(x,y)
accuracy,report=evaluation(validation,clf)
print(accuracy)
print(report)
learningCurve(RandomForestClassifier(criterion='entropy',n_estimators=10),x,y)
```

```
              0       0.82      1.00      0.90      2472
              1       0.97      0.30      0.46       778

    accuracy                            0.83      3250
   macro avg       0.90      0.65      0.68      3250
weighted avg       0.86      0.83      0.79      3250

0.8304615384615385
[Finished in 2.5s]
```



The cross validation curve doesn't drop in the end compared with the linear SVM classfier, and the training score is slightly higher in the end.