Thesis_over_sampling_data-with-weigths

November 2, 2019

1 Imports

```
In [1]: import pandas as pd
        import psycopg2

        %matplotlib inline

        import matplotlib
        import numpy as np
        import matplotlib.pyplot as plt
        from datetime import datetime
        from sklearn import neighbors, datasets
        from matplotlib.colors import ListedColormap
```

2 Read Data Set

```
In [2]: df = pd.read_csv("mergeData_v4.csv")
        df.head()
```

```
Out[2]:    speed_total_mean  steering_total_mean  brake_total_mean  \
        0          5.919151             0.503649          0.965743
        1          7.580378             0.499771          0.891302
        2          9.474048             0.494557          0.952182
        3         11.669419             0.500661          0.891913
        4         12.187044             0.499769          0.861132

           throttle_total_mean  acceleration_total_mean  speed_total_var  \
        0             0.820576                 0.030731        13.796202
        1             0.878839                -0.026652        31.451253
        2             0.781126                 0.006292        53.873833
        3             0.522365                 0.008028        47.209285
        4             0.558120                 0.001881        42.031423

           steering_total_var  brake_total_var  throttle_total_var  \
        0            0.000655         0.014468            0.028719
        1            0.000345         0.058767            0.010391
```

```
2           0.001231         0.022506            0.045416
3           0.000396         0.055982            0.112551
4           0.000430         0.102442            0.079023

   acceleration_total_var  total_time  distancePed  max_speed       PKE  \
0                0.039370   15.405173    89.992450  11.669766  1.932290
1                0.063480   11.412381    85.063860  13.499710  0.878493
2                0.106281  102.356492   789.212800  25.851397  2.857169
3                0.159198    7.505478    88.011610  20.055070  2.969647
4                0.158822    8.681609   105.973686  19.697004  4.033468

   PKE_Steering  speed_react  reaction_time  hadCollision
0     -0.000150     7.754880       1.048791             0
1      0.000274    13.472353       2.106615             0
2      0.000108    25.585112       0.079211             1
3     -0.000258    19.412087       1.161592             0
4      0.000066    18.461056       1.275896             0
```

2.0.1 Distribution

```
In [3]: num_obs = len(df)
        num_true = len(df.loc[df['hadCollision'] == 1])
        num_false = len(df.loc[df['hadCollision'] == 0])
        print("Number of True cases:  {0} ({1:2.2f}%)".format(num_true, (num_true/num_obs) * 1
        print("Number of False cases: {0} ({1:2.2f}%)".format(num_false, (num_false/num_obs) *
```

```
Number of True cases:  54 (9.66%)
Number of False cases: 505 (90.34%)
```

2.1 Split data set

```
In [50]: from sklearn.model_selection import train_test_split

         data = df.copy()
         X = data.drop('hadCollision', axis=1)
         Y = data['hadCollision']

         X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.4, random_state

In [51]: print("Training True  : {0} ({1:0.2f}%)".format(len(y_train[y_train[:] == 1]), (len(y
         print("Training False : {0} ({1:0.2f}%)".format(len(y_train[y_train[:] == 0]), (len(y
         print("")
         print("Test True      : {0} ({1:0.2f}%)".format(len(y_test[y_test[:] == 1]), (len(y_t
         print("Test False     : {0} ({1:0.2f}%)".format(len(y_test[y_test[:] == 0]), (len(y_t
```

```
Training True  : 32 (9.55%)
Training False : 303 (90.45%)
```

```
Test True       : 22 (9.82%)
Test False      : 202 (90.18%)
```

3 Over sampling data

3.1 Random Over Sampler

```
In [52]: from imblearn.over_sampling import RandomOverSampler
         ros = RandomOverSampler(random_state=0)
         X_resampled, y_resampled = ros.fit_sample(X_train, y_train)

         print("Training True  : {0} ({1:0.2f}%)".format(len(y_resampled[y_resampled[:] == 1])
         print("Training False : {0} ({1:0.2f}%)".format(len(y_resampled[y_resampled[:] == 0])
```

```
Training True  : 303 (50.00%)
Training False : 303 (50.00%)
```

3.2 Random forest

```
In [59]: from sklearn.ensemble import RandomForestClassifier
         rf_model = RandomForestClassifier(random_state=42, n_estimators=10, class_weight={0:1
         rf_model.fit(X_resampled, y_resampled.ravel())
```

```
Out[59]: RandomForestClassifier(bootstrap=True, class_weight={0: 10, 1: 1},
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, min_impurity_decrease=0.0,
                     min_impurity_split=None, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     n_estimators=10, n_jobs=1, oob_score=False, random_state=42,
                     verbose=0, warm_start=False)
```

3.2.1 Predict Test Data

```
In [60]: from sklearn import metrics

         rf_predict_test = rf_model.predict(X_test)

         # training metrics
         print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, rf_predict_test)))
```

```
Accuracy: 0.9286
```

```
In [61]: print(metrics.confusion_matrix(y_test, rf_predict_test) )
         print("")
         print("Classification Report")
         print(metrics.classification_report(y_test, rf_predict_test))
```

```
[[196   6]
 [ 10  12]]

Classification Report
             precision    recall  f1-score   support

          0       0.95      0.97      0.96       202
          1       0.67      0.55      0.60        22

avg / total       0.92      0.93      0.93       224
```

3.3 SMOT

```
In [62]: from imblearn.over_sampling import SMOTE, ADASYN
         X_resampled, y_resampled = SMOTE().fit_sample(X_train, y_train)

         print("Training True  : {0} ({1:0.2f}%)".format(len(y_resampled[y_resampled[:] == 1])
         print("Training False : {0} ({1:0.2f}%)".format(len(y_resampled[y_resampled[:] == 0])
```

```
Training True  : 303 (50.00%)
Training False : 303 (50.00%)
```

3.4 Random forest

```
In [63]: rf_model = RandomForestClassifier(random_state=42, n_estimators=10, class_weight={0:1
         rf_model.fit(X_resampled, y_resampled.ravel())
```

```
Out[63]: RandomForestClassifier(bootstrap=True, class_weight={0: 10, 1: 1},
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, min_impurity_decrease=0.0,
                     min_impurity_split=None, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     n_estimators=10, n_jobs=1, oob_score=False, random_state=42,
                     verbose=0, warm_start=False)
```

3.4.1 Predict Test Data

```
In [64]: rf_predict_test = rf_model.predict(X_test)

         # training metrics
         print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, rf_predict_test)))
```

```
Accuracy: 0.8973
```

```
In [65]: print(metrics.confusion_matrix(y_test, rf_predict_test) )
         print("")
         print("Classification Report")
         print(metrics.classification_report(y_test, rf_predict_test))

[[187  15]
 [  8  14]]

Classification Report
             precision    recall  f1-score   support

          0       0.96      0.93      0.94       202
          1       0.48      0.64      0.55        22

avg / total       0.91      0.90      0.90       224
```

3.5 ADASYN

```
In [66]: from imblearn.over_sampling import SMOTE, ADASYN

         X_resampled, y_resampled = ADASYN().fit_sample(X_train, y_train)

         print("Training True  : {0} ({1:0.2f}%)".format(len(y_resampled[y_resampled[:] == 1])
         print("Training False : {0} ({1:0.2f}%)".format(len(y_resampled[y_resampled[:] == 0])

Training True  : 314 (50.89%)
Training False : 303 (49.11%)
```

3.6 Random forest

```
In [67]: rf_model = RandomForestClassifier(random_state=42, n_estimators=10, class_weight={0:1
         rf_model.fit(X_resampled, y_resampled.ravel())

Out[67]: RandomForestClassifier(bootstrap=True, class_weight={0: 10, 1: 1},
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, min_impurity_decrease=0.0,
                     min_impurity_split=None, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     n_estimators=10, n_jobs=1, oob_score=False, random_state=42,
                     verbose=0, warm_start=False)
```

3.6.1 Predict Test Data

```
In [68]: rf_predict_test = rf_model.predict(X_test)

         # training metrics
         print("Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test, rf_predict_test)))
```

```
Accuracy: 0.8125
```

```
In [69]: print(metrics.confusion_matrix(y_test, rf_predict_test) )
         print("")
         print("Classification Report")
         print(metrics.classification_report(y_test, rf_predict_test))
```

```
[[161  41]
 [  1  21]]

Classification Report
             precision    recall  f1-score   support

          0       0.99      0.80      0.88       202
          1       0.34      0.95      0.50        22

avg / total       0.93      0.81      0.85       224
```