Thesis_over_sampling_data

November 1, 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	throttle_total_mean	acceleration_total_mean	speed_total_var	steering_total_var	brake_total_var	throttle_total_var
0	5.919151	0.503649	0.965743	0.820576	0.030731	13.796202	0.000655	0.014468	0.028719
1	7.580378	0.499771	0.891302	0.878839	-0.026652	31.451253	0.000345	0.058767	0.010391
2	9.474048	0.494557	0.952182	0.781126	0.006292	53.873833			
3	11.669419	0.500661	0.891913	0.522365	0.008028	47.209285			
4	12.187044	0.499769	0.861132	0.558120	0.001881	42.031423			

```
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) * 10
        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 [45]: 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=0.5, random_state=

In [46]: 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_te
         print("Test False     : {0} ({1:0.2f}%)".format(len(y_test[y_test[:] == 0]), (len(y_te
```

```
Training True  : 27 (9.68%)
Training False : 252 (90.32%)
```

```
Test True      : 27 (9.64%)
Test False     : 253 (90.36%)
```

3 Over sampling data

3.1 Random Over Sampler

```
In [47]: 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  : 252 (50.00%)
Training False : 252 (50.00%)
```

3.2 Random forest

```
In [48]: from sklearn.ensemble import RandomForestClassifier
         rf_model = RandomForestClassifier(random_state=42, n_estimators=10)      # Create ran
         rf_model.fit(X_resampled, y_resampled.ravel())
```

```
Out[48]: RandomForestClassifier(bootstrap=True, class_weight=None, 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 [49]: rf_predict_test = rf_model.predict(X_test)

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

```
Accuracy: 0.9179
```

```
In [50]: print(metrics.confusion_matrix(y_test, rf_predict_test) )
         print("")
         print("Classification Report")
         print(metrics.classification_report(y_test, rf_predict_test))
```

```
[[249   4]
 [ 19   8]]
```

```
Classification Report
             precision    recall  f1-score   support

          0       0.93      0.98      0.96       253
          1       0.67      0.30      0.41        27

avg / total       0.90      0.92      0.90       280
```

3.3 SMOT

```
In [51]: 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  : 252 (50.00%)
Training False : 252 (50.00%)
```

3.4 Random forest

```
In [52]: rf_model = RandomForestClassifier(random_state=42, n_estimators=10)      # Create ran
         rf_model.fit(X_resampled, y_resampled.ravel())

Out[52]: RandomForestClassifier(bootstrap=True, class_weight=None, 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 [53]: rf_predict_test = rf_model.predict(X_test)

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

Accuracy: 0.9107
```

```
In [54]: print(metrics.confusion_matrix(y_test, rf_predict_test) )
         print("")
         print("Classification Report")
         print(metrics.classification_report(y_test, rf_predict_test))
```

```
[[241  12]
 [ 13  14]]

Classification Report
             precision    recall  f1-score   support

          0       0.95      0.95      0.95       253
          1       0.54      0.52      0.53        27

avg / total       0.91      0.91      0.91       280
```

3.5 ADASYN

```
In [55]: 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  : 254 (50.20%)
Training False : 252 (49.80%)
```

3.6 Random forest

```
In [56]: rf_model = RandomForestClassifier(random_state=42, n_estimators=10)      # Create ran
         rf_model.fit(X_resampled, y_resampled.ravel())

Out[56]: RandomForestClassifier(bootstrap=True, class_weight=None, 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 [57]: rf_predict_test = rf_model.predict(X_test)

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

```
Accuracy: 0.8536
```

```
In [58]: print(metrics.confusion_matrix(y_test, rf_predict_test) )
         print("")
         print("Classification Report")
         print(metrics.classification_report(y_test, rf_predict_test))
```

```
[[220  33]
 [  8  19]]

Classification Report
             precision    recall  f1-score   support

          0       0.96      0.87      0.91       253
          1       0.37      0.70      0.48        27

avg / total       0.91      0.85      0.87       280
```