Safe Driver Experiment

An Inncretech Publication

In this experiment, we try to predict which drivers will be safest on the road based on a variety of metrics. Check out the steps we took and find the results below.

InnCreTech
Innovation Creativity Technology

In [1]:
```
train_file_path = "/train.csv"
import os
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.decomposition import PCA
#import warnings
#warnings.filterwarnings("ignore")

import os
import time
import numpy as np
import pandas as pd
#from matplotlib import pyplot as plt
from __future__ import division

from sklearn import svm
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, train_test_split, validation
_curve, learning_curve

def get_feature_importances(grid, X_test):
    #Returns a dataframe with feature importance info
    ls = list()
    ls.append({'feature':X_test.columns[a], 'importance':b})
    feature_importances = pd.DataFrame(ls).sort_values(by = ['importance'], as
cending=False)
    return(feature_importances)


def classify(grid, X_train, y_train, X_test, y_test):
    results = dict()

    #Training the model using grid search & cross validation
    start_time = time.time()
    grid.fit(X_train, y_train)
    end_time = time.time() - start_time
    results['training_time'] = end_time


    #Testing the model on the held out test data set
    start_time = time.time()
    grid_test = grid.predict(X_test)
    end_time = time.time() - start_time
    results['testing_time'] = end_time

    results['accuracy'] = metrics.accuracy_score(y_test, grid_test)
    results['report'] = metrics.classification_report(y_test, grid_test)
    results['matrix'] = metrics.confusion_matrix(y_test, grid_test)

    results['grid'] = grid
    results['grid_test'] = grid_test
```

```
    return(results)
```

In [2]:
```
data = (pd
         .read_csv(train_file_path)
#         .replace(-1, np.NaN)
        )

grouping = data[["id", "target"]].groupby("target").size()
class_0_num = int(round(grouping[0]*0.15))
class_1_num = int(round(grouping[1]*0.75))
```

In [3]:
```
train_sampled = pd.concat([data.loc[data.target==0].sample(class_0_num, random
_state=10),
                           data.loc[data.target==1].sample(class_1_num, random
_state=10)])

del data
```

In [4]:
```
mappings = {}
categorical_columns = filter(lambda column_name: column_name.endswith("_cat"),
 train_sampled.columns)
for column in categorical_columns:
    mappings[column] = column
```

In [5]:
```
train_encoded = pd.get_dummies(train_sampled, columns=categorical_columns, pre
fix=mappings, dummy_na=False)
train_encoded.shape
```

Out[5]:
```
(102299, 228)
```

In [6]:
```
nan_columns = filter(lambda column_name: column_name.endswith("-1"), train_enc
oded.columns)
train_encoded.drop(labels=nan_columns, axis=1, inplace=True)
train_encoded.shape


train, test = train_test_split(train_encoded,
                               test_size=0.3,
                               random_state=10,
                               stratify=train_encoded["target"])
```

In [7]:
```
X_train = train.ix[:, train_encoded.columns.difference(['id','target'])]
y_train = train.ix[:, 'target']
X_test = test.ix[:, train_encoded.columns.difference(['id','target'])]
y_test = test.ix[:, ['target']]
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:1: DeprecationWa
rning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-depre
cated
  """Entry point for launching an IPython kernel.
/usr/local/lib/python2.7/dist-packages/ipykernel_launcher.py:3: DeprecationWa
rning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexing.html#ix-indexer-is-depre
cated
  This is separate from the ipykernel package so we can avoid doing imports u
ntil
```

In [8]:
```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
```

In [15]:
```
C_OPTIONS = [0.01, 0.1, 1.0, 10.0, 100.0]

pipe = Pipeline([('reduce_dim', PCA(n_components=100)), ('classify', LogisticR
egression())])
param_grid = [{"classify__C":C_OPTIONS}]
```

In [16]:
```
grid = GridSearchCV(pipe, cv=5, n_jobs=1, param_grid=param_grid)
grid.fit(X_train, y_train)
```

Out[16]:
```
GridSearchCV(cv=5, error_score='raise',
       estimator=Pipeline(memory=None,
     steps=[('reduce_dim', PCA(copy=True, iterated_power='auto', n_components
=100, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)), ('classify', 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='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False))]),
       fit_params=None, iid=True, n_jobs=1,
       param_grid=[{'classify__C': [0.01, 0.1, 1.0, 10.0, 100.0]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [17]:
```
grid.best_estimator_
```

Out[17]:
```
Pipeline(memory=None,
     steps=[('reduce_dim', PCA(copy=True, iterated_power='auto', n_components
=100, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)), ('classify', LogisticRegression
(C=0.01, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False))])
```

In [18]:
```
grid_test = grid.predict(X_test)
```

In [19]:
```
print metrics.classification_report(y_test, grid_test)
```

```
             precision    recall  f1-score   support

          0       0.84      1.00      0.91     25809
          1       0.54      0.00      0.00      4881

avg / total       0.79      0.84      0.77     30690
```

In [26]:
```
print metrics.confusion_matrix(y_test, grid_test, labels=[0, 1])
```

```
[[25803     6]
 [ 4874     7]]
```

In [5]:
```
data[["id", "target"]].groupby("target").size()
```

Out[5]:
```
target
0    573518
1     21694
dtype: int64
```

In [29]:
```
7/13*100
```

Out[29]:
```
53.84615384615385
```

Thank You

If you'd like to learn more, or get in touch with our experts:

Email us: info@inncretech.com
Skype us: @voipgeek