Capstone Project - Telecom Network Service Disruption

Build Models with FULL dataset

This is Part 3 of the project documentation. (File Name 03-FINAL-TelstraModels-FullDataset)

Please see Part 1 for Project Details and Executive Summary. (File Name 01-FINAL -TelstraEDA)

Develop Models

Data

- Dependent variable is a 3 class variable **fault_severity**

- 5 features used **log_feature** (386 classes), **resource_type** (3 classes), **event_type** (6 classes), **location** (32 classes), **severity_type** (5 classes),

Estimators

- 5 classifier estimators have been selected for comparison – RandomForest Classifier, KNeighbors Classifier,
  DecisionTress Classifier, AdaBoost Classfier, GradientBoostClassifer

Hyperparameters

- *Hyperparameters for all estimators derived using sklearn.model_selection.RandomizedSearchCV*
  Please see file 02-FINAL-TelstraHyperparameters for details on optimization with hyperparameters

Scenarios

- Each estimator was run for the following scenarios

- All features used – hyperparameters given

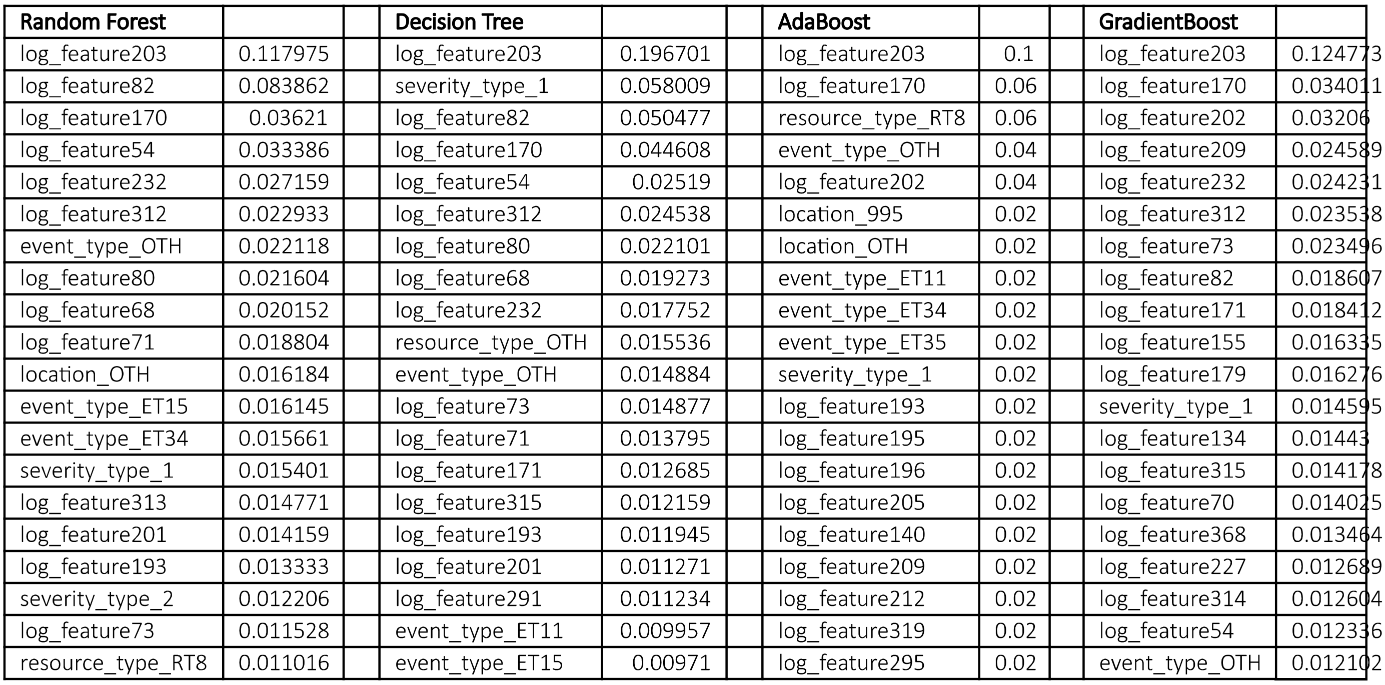
- All features used – default hyperparameters

Results

Estimator	All features			Features except logfeature	
	Default params	Optimum params		Default params	Optimum params
Random Forest Classifer	72.28%	75.49%		59.95%	62.53%
KNeigbors Classifer	71.33%	73.81%		59.37%	63.66%
DecisionTree Classifer	71.38%	70.11%		58.83%	58.19%
AdaBoost Classifer	72.42%	73.81%		66.23%	65.73%
GradientBoost Classifer	76.16%	76.48%		66.41%	65.96%
Baseline Accuracy	64.82%				

As can be seen, there is an improvement in accuracy, across all estimators except DecisionTree, when hyperparameters are provided to the model.

Top 20 features by Estimator



Random Forest			Decision Tree			AdaBoost			GradientBoost	
log_feature203	0.117975		log_feature203	0.196701		log_feature203	0.1		log_feature203	0.124773
log_feature82	0.083862		severity_type_1	0.058009		log_feature170	0.06		log_feature170	0.034011
log_feature170	0.03621		log_feature82	0.050477		resource_type_RT8	0.06		log_feature202	0.03206
log_feature54	0.033386		log_feature170	0.044608		event_type_OTH	0.04		log_feature209	0.024589
log_feature232	0.027159		log_feature54	0.02519		log_feature202	0.04		log_feature232	0.024231
log_feature312	0.022933		log_feature312	0.024538		location_995	0.02		log_feature312	0.023538
event_type_OTH	0.022118		log_feature80	0.022101		location_OTH	0.02		log_feature73	0.023496
log_feature80	0.021604		log_feature68	0.019273		event_type_ET11	0.02		log_feature82	0.018607
log_feature68	0.020152		log_feature232	0.017752		event_type_ET34	0.02		log_feature171	0.018412
log_feature71	0.018804		resource_type_OTH	0.015536		event_type_ET35	0.02		log_feature155	0.016335
location_OTH	0.016184		event_type_OTH	0.014884		severity_type_1	0.02		log_feature179	0.016276
event_type_ET15	0.016145		log_feature73	0.014877		log_feature193	0.02		severity_type_1	0.014595
event_type_ET34	0.015661		log_feature71	0.013795		log_feature195	0.02		log_feature134	0.01443
severity_type_1	0.015401		log_feature171	0.012685		log_feature196	0.02		log_feature315	0.014178
log_feature313	0.014771		log_feature315	0.012159		log_feature205	0.02		log_feature70	0.014025
log_feature201	0.014159		log_feature193	0.011945		log_feature140	0.02		log_feature368	0.013464
log_feature193	0.013333		log_feature201	0.011271		log_feature209	0.02		log_feature227	0.012689
severity_type_2	0.012206		log_feature291	0.011234		log_feature212	0.02		log_feature314	0.012604
log_feature73	0.011528		event_type_ET11	0.009957		log_feature319	0.02		log_feature54	0.012336
resource_type_RT8	0.011016		event_type_ET15	0.00971		log_feature295	0.02		event_type_OTH	0.012102

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sys
import time
```

Load Data

In [100]:

```
# Full dataset
train_id = pd.read_csv('./Data/train_id.csv')
test_id = pd.read_csv('./Data/test_id.csv')
```

In [101]:

```
print('Dataframe train - number of rows columns', train_id.shape)
print('Dataframe test - number of rows columns', test_id.shape)
```

```
Dataframe train - number of rows columns (7381, 434)
Dataframe test - number of rows columns (11171, 434)
```

Develop Models

In [102]:

```
#################################
# Import classifier model modules
#################################
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
#Gridsearch and scoring
from sklearn.grid_search import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
```

In [103]:

```
#################################
# Prepare data for modelling
#################################
# Create target dataframe
target = train_id['fault_severity']
# Drop fault_severity from train so we don't have to specify column names
train_id.drop(['fault_severity'], axis=1, inplace=True)
test_id.drop(['fault_severity'], axis=1, inplace=True)
train_id.drop(['id'], axis=1, inplace=True)
test_id.drop(['id'], axis=1, inplace=True)
```

In [104]:

```
#################################
# Assign values to lists
#################################
XTrain = train_id
yTrain = target
XTest = test_id
# Split training dataset into "train" and "validate" to compare model accuracy
X_train, X_test, y_train, y_test = train_test_split(XTrain,yTrain, test_size=0.3, random_st
```

In [105]:

```
#################################
# Baseline accuracy on the percentage for largest class in training dataset
#################################
baseline = target.value_counts()/len(target)
print('Baseline accuracy', round(baseline[0]*100,2))
```

```
Baseline accuracy 64.82
```

In [106]:

```
#############################################################
# Run models on "train" and "test" and get relative scores
#############################################################
# Random Forest Classifer with params
###
model = RandomForestClassifier(bootstrap=False,max_depth=70,max_features='auto',min_samples
                              n_estimators=1200)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
x = model.feature_importances_
print ('Random Forest Classifer accuracy - params:',round(accuracy_score(y_test, y_pred),4)
###
# Random Forest Classifer default
###
model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print ('Random Forest Classifer accuracy - default:',round(accuracy_score(y_test, y_pred),4
###
# Model feature importance
###
x = model.feature_importances_
zRF = pd.Series(data=x,index=X_train.columns)
print('Top 20 features')
print(zRF.sort_values(ascending=False).head(20))
```

```
Random Forest Classifer accuracy - params: 0.7549
Random Forest Classifer accuracy - default: 0.7228
Top 20 features
log_feature203      0.117975
log_feature82       0.083862
log_feature170      0.036210
log_feature54       0.033386
log_feature232      0.027159
log_feature312      0.022933
event_type_OTH      0.022118
log_feature80       0.021604
log_feature68       0.020152
log_feature71       0.018804
location_OTH        0.016184
event_type_ET15     0.016145
event_type_ET34     0.015661
severity_type_1     0.015401
log_feature313      0.014771
log_feature201      0.014159
log_feature193      0.013333
severity_type_2     0.012206
log_feature73       0.011528
resource_type_RT8   0.011016
dtype: float64
```

In [107]:

```
###
# KNeighbors Classifier params
###
model = KNeighborsClassifier(algorithm='kd_tree',metric='minkowski',leaf_size=10,p=4,weight
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print ('KNeighbors Classifer accuracy - params:',round(accuracy_score(y_test, y_pred),4))
###
# KNeighbors Classifier params default
###
model = KNeighborsClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print ('KNeighbors Classifer accuracy - default:',round(accuracy_score(y_test, y_pred),4))
###
# Model feature importance not available in KNN
###
```

```
KNeighbors Classifer accuracy - params: 0.7381
KNeighbors Classifer accuracy - default: 0.7133
```

In [109]:

```
###
# DecisionTree Classifier params
###
model = DecisionTreeClassifier(class_weight='balanced',criterion='gini',max_features=None,s
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print ('DecisionTree Classifer accuracy - params:',round(accuracy_score(y_test, y_pred),4))
###
# DecisionTree Classifier default
###
model = DecisionTreeClassifier()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print ('DecisionTree Classifer accuracy - default:',round(accuracy_score(y_test, y_pred),4)
###
# Model feature importance
###
x = model.feature_importances_
zDT = pd.Series(data=x,index=X_train.columns)
print('Top 20 features')
print(zDT.sort_values(ascending=False).head(20))
```

```
DecisionTree Classifer accuracy - params: 0.7011
DecisionTree Classifer accuracy - default: 0.7138
Top 20 features
log_feature203       0.196701
severity_type_1      0.058009
log_feature82        0.050477
log_feature170       0.044608
log_feature54        0.025190
log_feature312       0.024538
log_feature80        0.022101
log_feature68        0.019273
log_feature232       0.017752
resource_type_OTH    0.015536
event_type_OTH       0.014884
log_feature73        0.014877
log_feature71        0.013795
log_feature171       0.012685
log_feature315       0.012159
log_feature193       0.011945
log_feature201       0.011271
log_feature291       0.011234
event_type_ET11      0.009957
event_type_ET15      0.009710
dtype: float64
```

In [110]:

```
###
# AdaBoost Classifier params
###
model = AdaBoostClassifier(algorithm='SAMME.R',learning_rate=1.0,n_estimators=90,random_sta
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print ('AdaBoost Classifer accuracy - params:',round(accuracy_score(y_test, y_pred),4))
###
# AdaBoost Classifier default
###
model = AdaBoostClassifier()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print ('AdaBoost Classifer accuracy - default:',round(accuracy_score(y_test, y_pred),4))
###
# Model feature importance
###
x = model.feature_importances_
zAB = pd.Series(data=x,index=X_train.columns)
print('Top 20 features')
print(zAB.sort_values(ascending=False).head(20))
```

```
AdaBoost Classifer accuracy - params: 0.7381
AdaBoost Classifer accuracy - default: 0.7242
Top 20 features
log_feature203      0.10
log_feature170      0.06
resource_type_RT8   0.06
event_type_OTH      0.04
log_feature202      0.04
location_995        0.02
location_OTH        0.02
event_type_ET11     0.02
event_type_ET34     0.02
event_type_ET35     0.02
severity_type_1     0.02
log_feature193      0.02
log_feature195      0.02
log_feature196      0.02
log_feature205      0.02
log_feature140      0.02
```

In [111]:

```
###
# GradientBoost Classifier params
###
model = GradientBoostingClassifier(criterion='friedman_mse',init=None,learning_rate=0.1,max
                                  n_estimators=150,random_state=88)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print ('GradientBoost Classifer accuracy - params:',round(accuracy_score(y_test, y_pred),4)
###
# GradientBoost Classifier params
###
model = GradientBoostingClassifier()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print ('GradientBoost Classifer accuracy - default:',round(accuracy_score(y_test, y_pred),4
###
# Model feature importance
###
x = model.feature_importances_
zGB = pd.Series(data=x,index=X_train.columns)
print('Top 20 features')
print(zGB.sort_values(ascending=False).head(20))
```

```
GradientBoost Classifer accuracy - params: 0.7616
GradientBoost Classifer accuracy - default: 0.7648
Top 20 features
log_feature203     0.124773
log_feature170     0.034011
log_feature202     0.032060
log_feature209     0.024589
log_feature232     0.024231
log_feature312     0.023538
log_feature73      0.023496
log_feature82      0.018607
log_feature171     0.018412
log_feature155     0.016335
log_feature179     0.016276
severity_type_1    0.014595
log_feature134     0.014430
log_feature315     0.014178
log_feature70      0.014025
log_feature368     0.013464
log_feature227     0.012689
log_feature314     0.012604
log_feature54      0.012336
event_type_OTH     0.012102
dtype: float64
```

In [115]:

```
X_test['log_feature203'][X_test.log_feature203 > 0].count()
```

Out[115]:

```
337
```

In [116]:

```
X_test['log_feature170'][X_test.log_feature170 > 0].count()
```

Out[116]:

```
177
```

In [117]:

```
X_test['log_feature82'][X_test.log_feature82 > 0].count()
```

Out[117]:

```
410
```

In [ ]:

```

```