Using Decision Trees and Random Forests to Predict Autism and PDD

August 20, 2021

[9]:
```
# imports
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.model_selection import GridSearchCV
import sklearn.metrics
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import validation_curve
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
```

1 Distinguishing Autism from PDD with Trees

How predictable is Autism Spectrum Disorder at age 2, and can Random Forests and Decision Trees help discern it from PDD?

The outcome variable of interest is “bestest2”, which is a binary variable showing whether the child has Autism or PDD (Pervasive Developmental Disorder–generally considered as showing signs of autism, but below the threshold of being able to classify it as Autism)

1.1 Data Import

[10]:
```
autism = pd.read_csv("autism.csv")
autism
```

[10]:

	Unnamed: 0	childid	sicdegp	age2	vsae	gender	race	bestest2
0	1	1	high	0	6	male	white	pdd
1	2	1	high	1	7	male	white	pdd
2	3	1	high	3	18	male	white	pdd
3	4	1	high	7	25	male	white	pdd
4	5	1	high	11	27	male	white	pdd
..	…	…	…	…	…	…	…	…
599	608	97	high	11	50	male	nonwhite	pdd

```
600          609       99      low      0      9    male  nonwhite    autism
601          610       99      low      1     13    male  nonwhite    autism
602          611       99      low      3     10    male  nonwhite    autism
603          612       99      low      7     12    male  nonwhite    autism

[604 rows x 8 columns]
```

1.2 Data Cleaning

[12]:
```
mm = MinMaxScaler() # Fit a Scaler

cleaned_autism = pd.get_dummies(autism[['sicdegp', 'gender', 'race']]) # Get␣
↪dummies for categorical
mm.fit(autism[['age2', 'vsae']]) # Scale the two numeric variables
cleaned_autism[['age', 'vsae']] = mm.transform(autism[['age2', 'vsae']])
# For the sake of simplicity, coding outcome variable as 1 for autism, and PDD␣
↪as 0
y = autism['bestest2'].apply(lambda x : 1 if x == 'autism' else 0)
X = cleaned_autism.values
cleaned_autism.head(5)
```

[12]:
```
   sicdegp_high  sicdegp_low  sicdegp_med  gender_female  gender_male  \
0             1            0            0              0            1
1             1            0            0              0            1
2             1            0            0              0            1
3             1            0            0              0            1
4             1            0            0              0            1

   race_nonwhite  race_white       age      vsae
0              0           1  0.000000  0.025381
1              0           1  0.090909  0.030457
2              0           1  0.272727  0.086294
3              0           1  0.636364  0.121827
4              0           1  1.000000  0.131980
```

1.2.1 Train Test Split

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3)
```

1.3 Model Training

1.3.1 Basic Models

[ ]:
```
# Decision Tree
clf = tree.DecisionTreeClassifier()
clf.fit(X_train,y_train)
```

```
# Random Forest
clf_rf = RandomForestClassifier()
clf_rf.fit(X_train, y_train)
```

[ ]:
```
print('Accuracy before tuning for Decision Tree', clf.score(X_test, y_test))
print('Accuracy before tuning for Random Forests', clf_rf.score(X_test, y_test))
```

Not particularly good results, but lets see if the tuning can do better

1.3.2 Tuned Models

Decision Trees:

[ ]:
```
clf_param_grid = {'max_depth' : range(1,clf.tree_.max_depth),
              'max_features' : [.2, .4, .6, .8]}
grid_search = GridSearchCV(clf, clf_param_grid, cv = 8)
grid_search.fit(X_train, y_train)
print("Best Parameters for a CV Decision Tree:", grid_search.best_params_)
```

Random Forests:

[ ]:
```
clf_rf_param_grid = [{'n_estimators' : [3,10,30], 'max_features' : [2,4,6,8]},
              {'bootstrap' : [False], 'n_estimators' : [3,10], 'max_features' :␣
↪[2,3,4]}]

grid_search = GridSearchCV(clf_rf, clf_rf_param_grid, cv = 8)
grid_search.fit(X_train, y_train)
print("Best Parameters for a CV Random Forests Model:", grid_search.
↪best_params_)
```

1.4 Evaluation

[ ]:
```
best_clf = tree.DecisionTreeClassifier(max_depth = 5, max_features = 0.4)
best_clf.fit(X_train,y_train)
print('Accuracy after tuning for Decision Tree', best_clf.score(X_test, y_test))


best_clf_rf = RandomForestClassifier(max_features = 2, n_estimators = 10)
best_clf_rf.fit(X_train, y_train)
print('Accuracy after tuning for Random Forests', best_clf_rf.score(X_test,␣
↪y_test))
```

[ ]:
```
best_clf_predictions = best_clf.predict(X_test)
print("F1 for tuned Decision tree Model:",sklearn.metrics.f1_score(y_test,␣
↪best_clf_predictions))
print("Recall for tuned Decision tree Model:",sklearn.metrics.
↪recall_score(y_test, best_clf_predictions))
```

Surprisingly, the random forest model looks like it overfit to the training data more so than the Decision Tree model. We got a considerably better performance after tuning the Decision Tree model, and we got worse after tuning the Random Forest model. Still, 67% isn't particularly good accuracy. However, we still managed to achieve a very respectable F1 score of 78~% and an 87% Recall score, which is promising if the goal is simply to provide proper assistance to as many diagnosed with autism as possible.