Predicting Children's Food Allergies

By Dara Estrada

Business Problem & Understanding

Can we predict a child's future food allergies based on their current ones?

The prevalence of food allergies in children has increased at a rapid rate within the past couple of decades. "Between 1997 and 2011, food allergies among children increased 50% and now affect 6 million US children" [1]:[source (https://community.kidswithfoodallergies.org/blog/10-shareable-images-for-food-allergy-awareness-week-1)](https://community.kidswithfoodallergies.org/blog/10-shareable-images-for-food-allergy-awareness-week-1). As a millenial born in 1991, my elementary school experience was void of strict protocols on outside food, unlike the current school landscape. Classmates often had cupcakes or pizza brought in to celebrate birthdays and peanut butter crackers were the unofficial snack of field trips.

The stakeholder is Food Allergy Research & Education (FARE). The objective is to use existing data to predict and model patients' future food allergies. The ability to predict an increase of a specific food allergy identifies new trends and allows research to pivot and address them.

Data Understanding

The dataset was obtained from [zenodo.org (https://zenodo.org/record/44529#.YmAFlZPMIiz)](https://zenodo.org/record/44529#.YmAFlZPMIiz). It entails information on food allergies alongside preexisting conditions on a peer reviewd study from the Children's Hospital of Philadelphia of patients born in 1983-2012. There were 333,200 individuals in the dataset. The columns were then pared down as listed below.

Data Preparation

In [1]:

```
# import neccessary libraries

import pandas as pd
import seaborn as sns
import numpy as np
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier, DummyRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.preprocessing import StandardScaler, MinMaxScaler, OneHotE
from sklearn.feature_selection import RFE
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, make_scorer, recall_score,
from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
from sklearn.tree import DecisionTreeClassifier, plot_tree
import re
from imblearn.over_sampling import SMOTE
```

In [2]:

```
# read in csv
df = pd.read_csv('EA/food-allergy-analysis-Zenodo (1).csv')
df
```

4	5	2006	S1 - Female	R1 - Black	E0 - Non-Hispanic
...	...	...	...	...	...
333195	333196	2006	S0 - Male	R0 - White	E0 - Non-Hispanic
333196	333197	2006	S1 - Female	R1 - Black	E0 - Non-Hispanic
333197	333198	2006	S0 - Male	R0 - White	E0 - Non-Hispanic
333198	333199	2006	S0 - Male	R3 - Other	E0 - Non-Hispanic
333199	333200	2006	S1 - Female	R0 - White	E0 - Non-Hispanic

333200 rows × 50 columns

Exploratory Data Analysis

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 333200 entries, 0 to 333199
Data columns (total 50 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   SUBJECT_ID                 333200 non-null  int64
 1   BIRTH_YEAR                 333200 non-null  int64
 2   GENDER_FACTOR              333200 non-null  object
 3   RACE_FACTOR                333200 non-null  object
 4   ETHNICITY_FACTOR           333200 non-null  object
 5   PAYER_FACTOR               333200 non-null  object
 6   ATOPIC_MARCH_COHORT        333200 non-null  bool
 7   AGE_START_YEARS            333200 non-null  float64
 8   AGE_END_YEARS              333200 non-null  float64
 9   SHELLFISH_ALG_START        5246 non-null    float64
 10  SHELLFISH_ALG_END          1051 non-null    float64
 11  FISH_ALG_START             1796 non-null    float64
 12  FISH_ALG_END               527  non-null    float64
 13  MILK_ALG_START             7289 non-null    float64
```

In [4]:
```
df.describe()
```

Out[4]:

	SUBJECT_ID	BIRTH_YEAR	AGE_START_YEARS	AGE_END_YEARS	SHELLFISH_ALG_STAR
count	333200.000000	333200.000000	333200.000000	333200.000000	5246.00000
mean	166600.500000	2001.261191	3.942140	10.336654	8.72407
std	96186.699184	6.603479	4.646174	5.623426	5.27309
min	1.000000	1983.000000	-4.312115	1.002053	0.09308
25%	83300.750000	1996.000000	0.021903	5.289528	3.97535
50%	166600.500000	2002.000000	1.763176	10.193018	8.36139
75%	249900.250000	2007.000000	7.208761	15.616701	13.07802
max	333200.000000	2012.000000	17.984942	18.997947	24.29842

8 rows × 45 columns

The patients insurance status, the start allergy date and a column with only one value were all dropped. Nan values are filled with 0. `GENDER_FACTOR`, `RACE_FACTOR`, `ETHNICITY_FACTOR` were parsed out to binomials. Target variable was set from the sum of the allergy only columns. The specific level of food allergy severity is not being considered in this analysis. If a patient was in a march cohort was also dropped. New datasets for separating out pre-existing conditions from food allergies are created and start values are also dropped.

In [5]:

```python
# Drop payer_factor & treenut
df = df.drop(['PAYER_FACTOR','TREENUT_ALG_START','TREENUT_ALG_END','ATO
```

In [6]:

```python
#replacing nans
df = df.fillna(0)
```

In [7]:

```python
# parsing out GENDER_FACTOR, RACE_FACTOR, ETHNICITY_FACTOR
df_GenRaceEth = df.loc[:,'GENDER_FACTOR':'ETHNICITY_FACTOR']

df['GENDER_FACTOR'] = df_GenRaceEth['GENDER_FACTOR'].apply(lambda x: x.
df['RACE_FACTOR'] = df_GenRaceEth['RACE_FACTOR'].apply(lambda x: x.repl
df['ETHNICITY_FACTOR'] = df_GenRaceEth['ETHNICITY_FACTOR'].apply(lambda
df
```

Out[7]:

	SUBJECT_ID	BIRTH_YEAR	GENDER_FACTOR	RACE_FACTOR	ETHNICITY_FACTOR	AGE_S
0	1	2006	1	1	0	
1	2	1994	1	0	0	
2	3	2006	0	0	1	
3	4	2004	0	4	1	
4	5	2006	1	1	0	
...	...	...	...	...	...	
333195	333196	2006	0	0	0	
333196	333197	2006	1	1	0	
333197	333198	2006	0	0	0	
333198	333199	2006	0	3	0	
333199	333200	2006	1	0	0	

333200 rows × 46 columns

In [8]:

```
# allergy only sublist
df_ALG = df.loc[:,'SHELLFISH_ALG_START':'CASHEW_ALG_END']
df_ALG
```

Out[8]:

	SHELLFISH_ALG_START	SHELLFISH_ALG_END	FISH_ALG_START	FISH_ALG_END	MILK_AL
0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	
...	...	...	...	...	
333195	0.0	0.0	0.0	0.0	
333196	0.0	0.0	0.0	0.0	
333197	0.0	0.0	0.0	0.0	
333198	0.0	0.0	0.0	0.0	
333199	0.0	0.0	0.0	0.0	

333200 rows × 30 columns

In [9]:

```
# dropping starts
df.columns.str.endswith('START')
df = df.loc[:,~df.columns.str.endswith('START')]
```

In [10]:

```
# dropping firsts
df.columns.str.startswith('FIRST')
df = df.loc[:,~df.columns.str.startswith('FIRST')]
```

In [11]:

```
# creating allergy only subset and setting target
# adding df_ALF to main df
df['ALG_TOTAL'] = df_ALG.sum(axis=1)

# zero = no allergy | greater than zero = Allergy
df['ALG_target'] = ['Allergy' if x > 0 else 'No_Allergy' for x in df['A

# turning it into a boolean
df['ALG_target'] = [0 if x > 0 else 1 for x in df['ALG_TOTAL']]
```

In [12]:

```
df['ALG_target'].value_counts()
```

Out[12]:

```
1    310602
0     22598
Name: ALG_target, dtype: int64
```

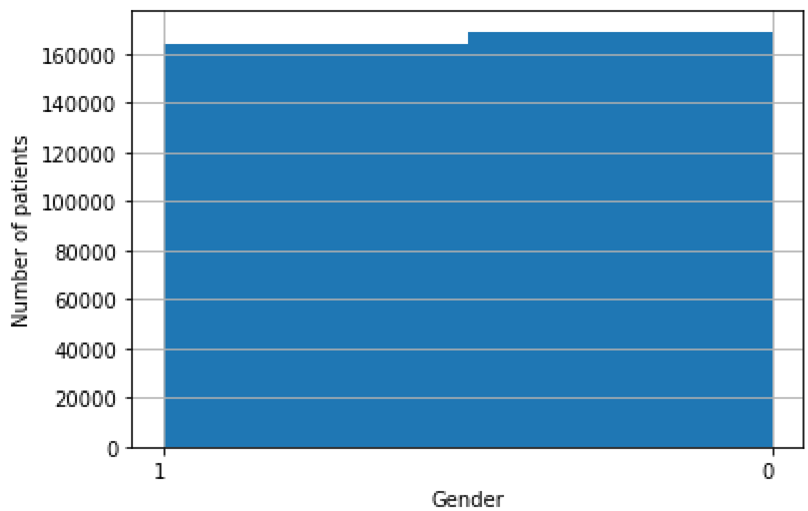
Exploring the dataset.

In [13]:

```
# amount of male and female
df['GENDER_FACTOR'].hist(bins=2)
plt.xlabel('Gender')
plt.ylabel('Number of patients');
df['GENDER_FACTOR'].value_counts()
```

Out[13]:

```
0     169032
1     164168
Name: GENDER_FACTOR, dtype: int64
```

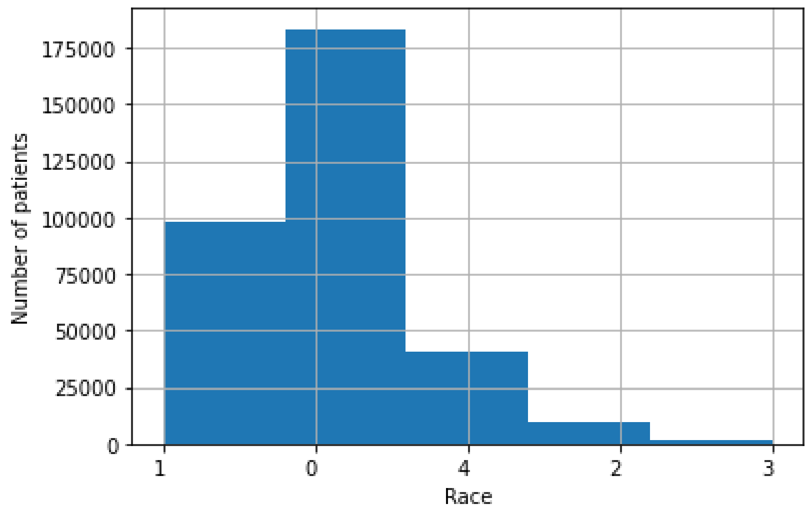


In [14]:

```
# amount of each race
df['RACE_FACTOR'].hist(bins=5)
plt.xlabel('Race')
plt.ylabel('Number of patients');
df['RACE_FACTOR'].value_counts()
# 0- White, 1- Black, 3- Other, 4- Unknown
```

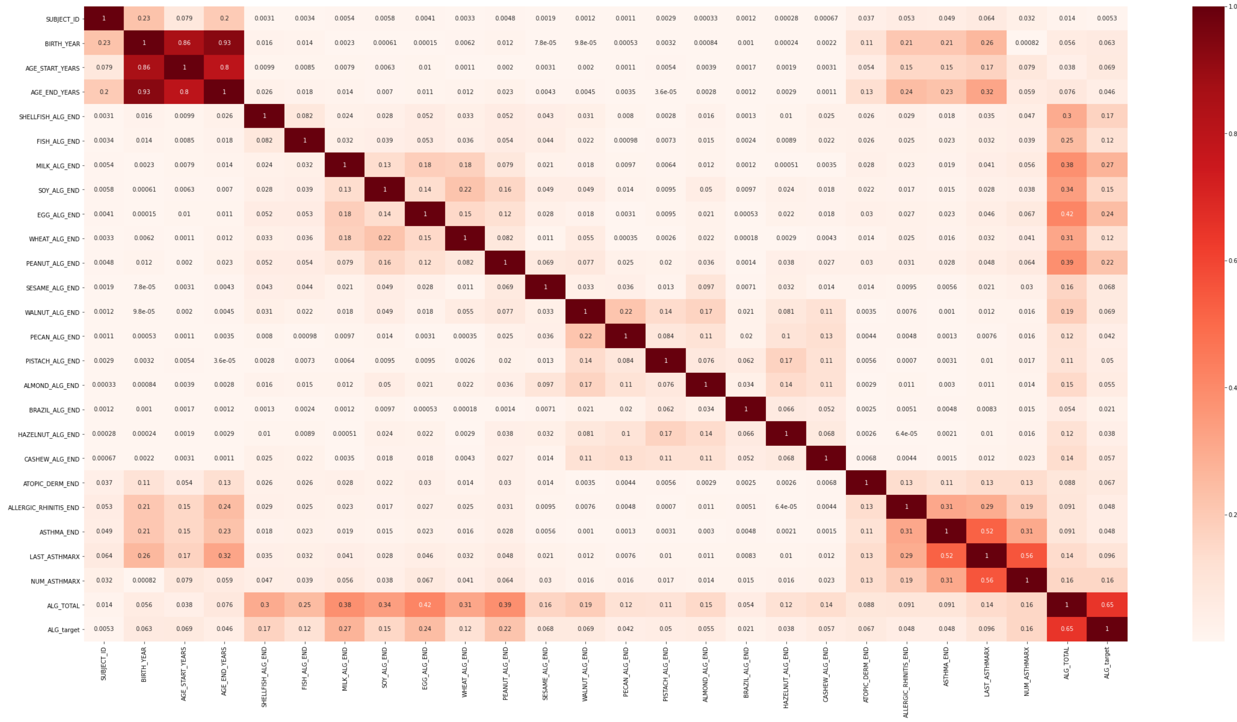
Out[14]:

```
0    183308
1     97795
4     40940
2      9152
3      2005
Name: RACE_FACTOR, dtype: int64
```



In [15]:

```
corr = df.corr().abs()
fig, ax=plt.subplots(figsize=(40,20))
sns.heatmap(corr, cmap='Reds', annot=True);
```



Modeling

Function that prints out training/test scores for each metric of training and test data and corresponding confusion matrix.

In [16]:
```
# Credit to Peter Voung
def score_matrix_printer(model, X_train, y_train, X_test, y_test):
    train_pred = model.predict(X_train)
    test_pred = model.predict(X_test)

    # Cleaning up scores to be more visually appealing
    ascore_train = round((accuracy_score(y_train, train_pred) * 100), 2
    pscore_train = round((precision_score(y_train, train_pred) * 100),

    ascore_test = round((accuracy_score(y_test, test_pred) * 100), 2)
    pscore_test = round((precision_score(y_test, test_pred) * 100), 2)

    conf_mat = plot_confusion_matrix(model, X_test, y_test)
    roc_cirve = plot_roc_curve(model, X_test, y_test)

    print(f"""
    Train Accuracy: {ascore_train}%
    Train Precision: {pscore_train}%
    ------------------------------
    Test Accuracy: {ascore_test}%
    Test Precision: {pscore_test}%
    """)
```

Split, scaled, transformed and smote

In [17]:
```
# train test split
y = df['ALG_target']
X = df.drop(['ALG_target'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0

# scale
scale = StandardScaler()
X_train_scaled = scale.fit_transform(X_train)
X_test_scaled = scale.transform(X_test)

# SMOTE data to achieve target variable balance
sm = SMOTE(sampling_strategy='minority', random_state=15)
X_train_scaled, y_train = sm.fit_resample(X_train_scaled, y_train)
```

In [18]:
```
y_train.value_counts()
```

Out[18]:
```
1    217439
0    217439
Name: ALG_target, dtype: int64
```

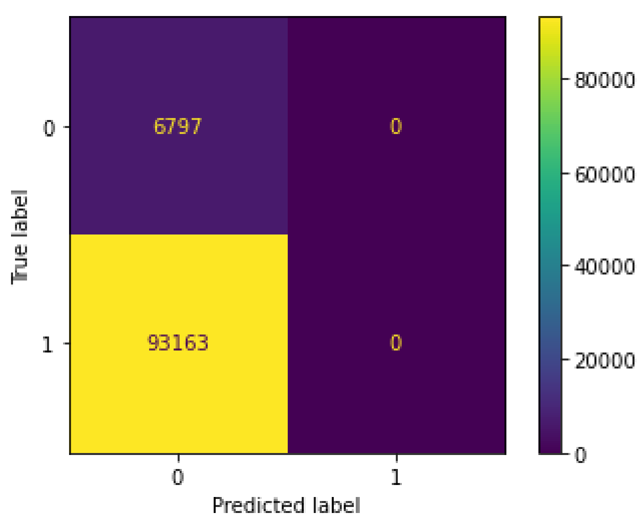
Dummy Baseline Model

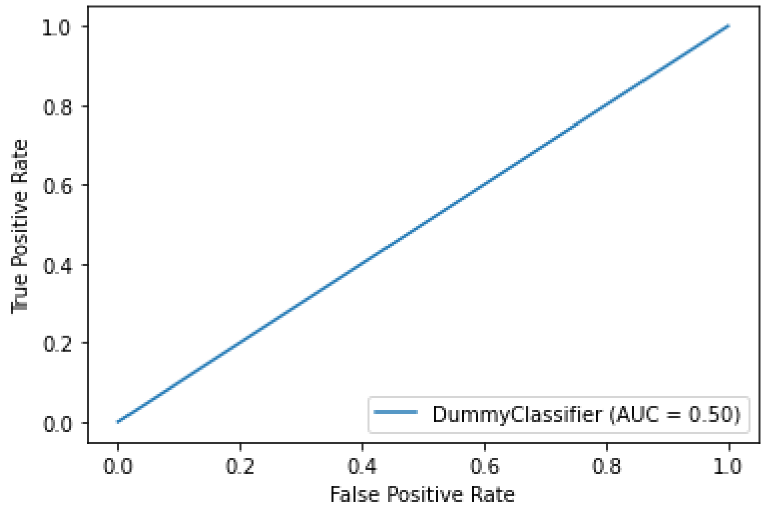
In [19]:

```
# Instantiated, fit, and ran dummy model

dum = DummyClassifier(strategy="most_frequent")
dum.fit(X_train_scaled, y_train)
y_hat_train = dum.predict(X_train_scaled)
y_hat_test = dum.predict(X_test_scaled)
print(f'Train {accuracy_score(y_train, y_hat_train)}')
print(f'Test {accuracy_score(y_test, y_hat_test)}')

# Plotted confusion matrix and ROC AUC for dummy model
score_matrix_printer(dum, X_train_scaled, y_train, X_test_scaled, y_tes
```

```
Train 0.5
Test 0.06799719887955182

/Users/darla/opt/anaconda3/envs/learn-env/lib/python3.8/site-packages/skl
earn/metrics/_classification.py:1221: UndefinedMetricWarning: Precision i
s ill-defined and being set to 0.0 due to no predicted samples. Use `zero
_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
Train Accuracy: 50.0%
Train Precision: 0.0%
------------------------------
Test Accuracy: 6.8%
Test Precision: 0.0%
```





One Hot Encoded

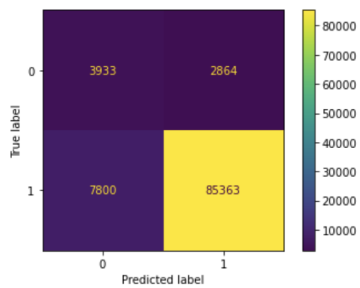
In [20]:
```
ohe = OneHotEncoder(handle_unknown='ignore', sparse=False)
ohe.fit_transform(X_train)
```

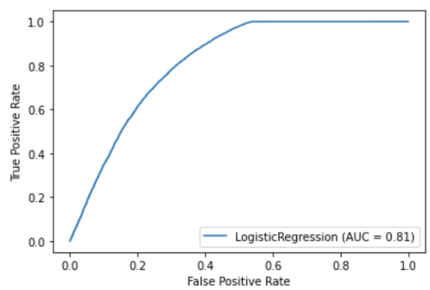
Out[20]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Logistic Regression Model

In [30]:
```
# instatiate logistic model
#fit to training
#score on test
```

In [31]:
```
logreg = LogisticRegression()
logreg.fit(X_train_scaled, y_train)
y_hat_train = logreg.predict(X_train_scaled)
y_hat_test = logreg.predict(X_test_scaled)

plot_confusion_matrix(logreg, X_test_scaled, y_test)
plot_roc_curve(logreg, X_test_scaled, y_test);
```





In [32]:
```
score_matrix_printer(logreg, X_train_scaled, y_train, X_test_scaled, y_test)
```

```
Train Accuracy: 75.45%
Train Precision: 69.25%
------------------------------
Test Accuracy: 89.33%
Test Precision: 96.75%
```

Decision Tree Classifier

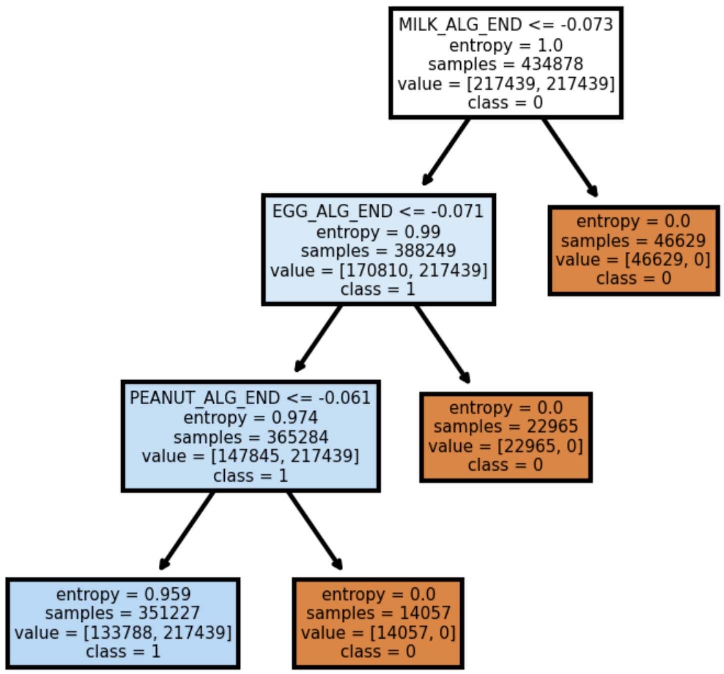
Finding which allergens have the highest frequencies.

Decision Tree Classifier

In [33]:
```python
from sklearn.tree import DecisionTreeClassifier, plot_tree

DTC = DecisionTreeClassifier(criterion='entropy', max_depth = 3)
DTC.fit(X_train_scaled, y_train)
```

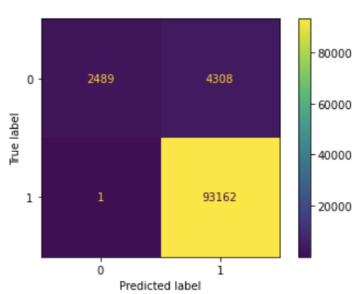
Out[33]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

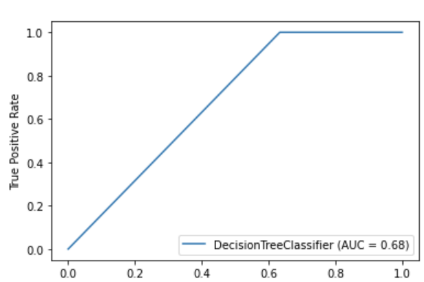
In [34]:
```python
fig, axes = plt.subplots(nrows = 1,ncols =1 , figsize = (3,3), dpi=300)
plot_tree(DTC,
               feature_names = X_train.columns,
               class_names=np.unique(y).astype('str'),
               filled = True)
plt.show()
```



In [36]:
```python
score_matrix_printer(DTC, X_train_scaled, y_train, X_test_scaled, y_test)
```

```
Train Accuracy: 69.24%
Train Precision: 61.91%
------------------------------
Test Accuracy: 95.69%
Test Precision: 95.58%
```





Evaluation

In the future I would like to run an additional target also utilizing pre-existing traits of asthma and eczema as well in addition to having an adult dataset. Finally I would like to implement a risk level range estimation prediction.