ANALYSIS

In [1]:

```
#uncomment this below code to install imblearn package
# !pip install imbalanced-learn
```

In [2]:

```
import pandas as pd
import numpy as np
import sklearn

#statistics
from scipy.stats import chi2_contingency, ttest_ind

import cudf #gpu-powered DataFrame (Pandas alternative)

#imbalance handling
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler, RepeatedEditedNearestNeighl
from imblearn.pipeline import Pipeline

#preprocessing
from sklearn import preprocessing
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder, LabelEncoder, MinI

#internal validation
from sklearn.model_selection import StratifiedKFold, KFold, RepeatedStratifiedKFol

#performance metrices
from sklearn.metrics import confusion_matrix, classification_report, f1_score, bal

#Models selection
from sklearn.naive_bayes import GaussianNB, ComplementNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from cuml.svm import SVC #gpu-powered SVM



#save and load trained model
import pickle

#visualisation
import matplotlib.pyplot as plt
import seaborn as sns

from collections import Counter
```

In [3]:

```
# Data loader
features = pd.read_csv('../FinalData/allFeatures2.csv')
outcomes = pd.read_csv("../FinalData/data_outcomes_29032022.csv")
outcomes = outcomes[outcomes.columns[1:]]
```

In [4]:

```
masterData = features.merge(outcomes, how = 'left', left_on='patid', right_on='pat
masterData = masterData.dropna() #NAs from Country
masterData = masterData.reset_index(drop=True)
print('original data shape: ', masterData.shape)
```

```
original data shape:  (313405, 63)
```

In [5]:

```
#Aggregate outcome for more than 3 months horizon

masterData["outcome_combined_6months"] = masterData.apply(lambda x: (x["outcome_3m
masterData["outcome_combined_9months"] = masterData.apply(lambda x: (x["outcome_co
masterData["outcome_combined_12months"] = masterData.apply(lambda x: (x["outcome_c
masterData["outcome_combined_15months"] = masterData.apply(lambda x: (x["outcome_c
masterData["outcome_combined_18months"] = masterData.apply(lambda x: (x["outcome_c
masterData["outcome_combined_24months"] = masterData.apply(lambda x: (x["outcome_c
```

In [6]:

```
#Positive vs negative class ratio

print('3 months -> 1 : ', round(masterData.outcome_3months.value_counts()[0]/maste
print('6 months -> 1 : ', round(masterData.outcome_combined_6months.value_counts()
print('9 months -> 1 : ', round(masterData.outcome_combined_9months.value_counts()
print('12 months -> 1 : ', round(masterData.outcome_combined_12months.value_counts
print('15 months -> 1 : ', round(masterData.outcome_combined_15months.value_counts
print('18 months -> 1 : ', round(masterData.outcome_combined_18months.value_counts
print('24 months -> 1 : ', round(masterData.outcome_combined_24months.value_counts
```

```
3 months -> 1 :  16.31
6 months -> 1 :  10.44
9 months -> 1 :  7.85
12 months -> 1 :  5.68
15 months -> 1 :  4.48
18 months -> 1 :  4.0
24 months -> 1 :  3.5
```

In [7]:

```
#Proportion of asthma attack in each outcome

print('3 months -> ', round(masterData.outcome_3months.value_counts()[1]/len(maste
print('6 months -> ', round(masterData.outcome_combined_6months.value_counts()[1]/
print('9 months -> ', round(masterData.outcome_combined_9months.value_counts()[1]/
print('12 months -> ', round(masterData.outcome_combined_12months.value_counts()[1
print('15 months -> ', round(masterData.outcome_combined_15months.value_counts()[1
print('18 months -> ', round(masterData.outcome_combined_18months.value_counts()[1
print('24 months -> ', round(masterData.outcome_combined_24months.value_counts()[1
```

```
3 months ->  5.78 %
6 months ->  8.74 %
9 months ->  11.3 %
12 months ->  14.98 %
15 months ->  18.24 %
18 months ->  20.0 %
24 months ->  22.24 %
```

In [8]:

```
#Data scenario
# 1: all data without ethnicity variable
# 2: all data with ethnicity variable (include all missing values in ethnicity as 
# 3: filter data based on ethnicity (exclude missing values)

scenario = 1 #change it based on the scenario

if scenario == 1:
    #Exclude ethnic column
    allData = masterData.drop('ethnic', axis=1)
elif scenario == 2:
    #include all data
    allData = masterData
elif scenario == 3:
    #exclude missing values for ethnic variable
    allData = masterData[masterData.ethnic!='0']
allData = allData.reset_index(drop=True)
print('Data shape for scenario', str(scenario), allData.shape)
```

```
Data shape for scenario 1 (313405, 68)
```

In [9]:

```
#change sex column to binary numeric, flag intersex as NAs

def sexConverter (x):
    if x == 'Female':
        return 0
    elif x == 'Male':
        return 1
    elif x == 'Intersex':
        return None
    else:
        return x

allData['sex'] = allData.apply(lambda x: sexConverter(x.sex), axis=1)
print('Intersex proportion: ', sum(allData['sex'].isnull())/allData.shape[0]*100,
allData = allData.dropna(subset=['sex']) #exclude missing values (intersex)
allData = allData.reset_index(drop=True)
print('Data shape after excluding missing values in sex variable: ', allData.shape
```

```
Intersex proportion:  0.003828911472375999 %
Data shape after excluding missing values in sex variable:  (313393, 68)
```

In [10]:

```
#Split data into training and evaluation set based on the country. Include only 18+

trainingData = allData[(allData.Country == 'England') & (allData.age>18)]
evaluationData = allData[((allData.Country == 'Scotland') | (allData.Country == 'Wa

#remove country variable
trainingData = trainingData.drop('Country', axis=1)
evaluationData = evaluationData.drop('Country', axis=1)

trainingData = trainingData.reset_index(drop=True)
evaluationData = evaluationData.reset_index(drop=True)

print('Training data shape:', trainingData.shape)
print('Evaluation data shape: ', evaluationData.shape)
```

```
Training data shape: (231121, 67)
Evaluation data shape:  (10268, 67)
```

In [11]:

```
#Identify categorical and continuous variables from the dataset for preprocessing p

summaryData = trainingData.describe().T
excludeVars = summaryData[summaryData['max'] == 0].index.to_list() #exclude variab
binaryVars = summaryData[summaryData['max'] == 1].index.to_list()
categoricalNonnumericVars = trainingData.select_dtypes(['object']).columns.to_list
categoricalNonnumericVars = categoricalNonnumericVars + ['BTS_step'] #BTS step is n
```

In [12]:

```
#Define feature candidates

features_columns = trainingData.columns.to_list()
exclude_columns = ['patid', 'practice_id', #identifier
                   'BMI', #use the categorical instead
                   'ICS_medication_possesion_ratio', #the max value is inf
                   'Spacer', 'numPCSAsthma',  #all zero
                   'outcome_3months', 'outcome_6months', 'outcome_9months', 'outcom
                   'outcome_21months', 'outcome_24months', 'outcome_combined_6month
                   'outcome_combined_15months', 'outcome_combined_18months', 'outco
                  ]
exclude_columns = exclude_columns + [x for x in features_columns if '_count' in x]
```

```
features_columns = [x for x in features_columns if x not in exclude_columns]
print('Features size: ', len(features_columns))
print(features_columns)
```

```
Features size:  34
['sex', 'age', 'smokingStatus', 'CharlsonScore', 'PEFStatus', 'EosinophilLevel',
'BTS_step', 'average_daily_dose_ICS', 'prescribed_daily_dose_ICS', 'DeviceType',
'numOCS', 'PriorEducation', 'numPCS', 'numAntibioticsEvents', 'numAntibioticswithL
RTI', 'numOCSEvents', 'numOCSwithLRTI', 'numAsthmaAttacks', 'numAcuteRespEvents',
'numHospEvents', 'BMI_cat', 'comorbid_anaphylaxis', 'comorbid_anxiety', 'comorbid_
cardiovascular disease', 'comorbid_rhinitis', 'comorbid_eczema', 'comorbid_heart f
ailure', 'comorbid_ischaemic heart disease', 'comorbid_nasal polyp', 'comorbid_pso
riasis', 'comorbid_diabetes mellitus', 'comedication_paracetamol', 'comedication_n
saids', 'comedication_betablocker']
```

In [13]:

```
#ONE HOT encoding for categorical data

categoricalNonnumericVars = pd.Series(list(set(categoricalNonnumericVars).intersec

# define one hot encoder
categoricalEncoder = OneHotEncoder(sparse=False)

# transform data
result = categoricalEncoder.fit_transform(trainingData[categoricalNonnumericVars])
result = pd.DataFrame(result, columns=categoricalEncoder.get_feature_names_out())

#save encoder
pickle.dump(categoricalEncoder, open('./models/categoricalEncoder.pkl', 'wb'))

# replace categorical variables in the original data with the one hot version
trainingData = pd.concat([trainingData.loc[:, ~trainingData.columns.isin(categorica
print('Data shape after one-hot encoding: ', trainingData.shape)
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/preprocessing/_encoder
s.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 an
d will be removed in 1.4. `sparse_output` is ignored unless you leave `sparse` to
its default value.
  warnings.warn(
Data shape after one-hot encoding:  (231121, 86)
```

In [14]:

```
#Scaling continous variable into 0-1 range

# summaryData = allData.describe().T
continuous_vars = summaryData[summaryData['max'] >5].index.to_list() + ['numHospEve
continuous_vars = pd.Series(list(set(continuous_vars).intersection(set(features_col

# define scaler
scaler = MinMaxScaler()

#save scaler
pickle.dump(scaler, open('./models/scaler.pkl', 'wb'))


# transform data
result = scaler.fit_transform(trainingData[continuous_vars])
result = pd.DataFrame(result, columns=scaler.get_feature_names_out())

allData = pd.concat([trainingData.loc[:,~trainingData.columns.isin(continuous_vars

print('Data shape after scaling: ', trainingData.shape)
```

```
Data shape after scaling:  (231121, 86)
```

In [15]:

```
#Update feature candidates
```

```
features_columns = trainingData.columns.to_list()
exclude_columns = ['patid', 'practice_id', #identifier
                   'BMI', #use the categorical instead
                   'ICS_medication_possesion_ratio', #the max value is inf
                   'Spacer', 'numPCSAsthma',  #all zero
                   'outcome_3months', 'outcome_6months', 'outcome_9months', 'outcom
                   'outcome_21months', 'outcome_24months', 'outcome_combined_6month
                   'outcome_combined_15months', 'outcome_combined_18months', 'outc
                  ]
exclude_columns = exclude_columns + [x for x in features_columns if '_count' in x]
features_columns = [x for x in features_columns if x not in exclude_columns]
print('Features size: ', len(features_columns))
print(features_columns)
```

```
Features size:  53
['sex', 'age', 'CharlsonScore', 'average_daily_dose_ICS', 'prescribed_daily_dose_I
CS', 'numOCS', 'PriorEducation', 'numPCS', 'numAntibioticsEvents', 'numAntibiotics
withLRTI', 'numOCSEvents', 'numOCSwithLRTI', 'numAsthmaAttacks', 'numAcuteRespEven
ts', 'numHospEvents', 'comorbid_anaphylaxis', 'comorbid_anxiety', 'comorbid_cardio
vascular disease', 'comorbid_rhinitis', 'comorbid_eczema', 'comorbid_heart failur
e', 'comorbid_ischaemic heart disease', 'comorbid_nasal polyp', 'comorbid_psoriasi
s', 'comorbid_diabetes mellitus', 'comedication_paracetamol', 'comedication_nsaid
s', 'comedication_betablocker', 'PEFStatus_PEF_60-80', 'PEFStatus_PEF_less than 6
0', 'PEFStatus_PEF_more than 80', 'PEFStatus_PEF_not recroded', 'smokingStatus_Smo
king_current', 'smokingStatus_Smoking_former', 'smokingStatus_Smoking_never', 'Dev
iceType_DeviceType_BAI', 'DeviceType_DeviceType_DPI', 'DeviceType_DeviceType_NEB',
'DeviceType_DeviceType_pMDI', 'DeviceType_DeviceType_unknown', 'BMI_cat_Normalweig
ht', 'BMI_cat_Obese', 'BMI_cat_Overweight', 'BMI_cat_Underweight', 'EosinophilLeve
l_Eosinophil_high', 'EosinophilLevel_Eosinophil_normal', 'EosinophilLevel_Eosinoph
il_unknown', 'BTS_step_0.0', 'BTS_step_1.0', 'BTS_step_2.0', 'BTS_step_3.0', 'BTS_
step_4.0', 'BTS_step_5.0']
```

In [16]:

```
#ONE HOT encoding for evaluation dataset

# transform data
result = categoricalEncoder.transform(evaluationData[categoricalNonnumericVars])
result = pd.DataFrame(result, columns=categoricalEncoder.get_feature_names_out())

# replace categorical variables in the original data with the one hot version
evaluationData = pd.concat([evaluationData.loc[:, ~evaluationData.columns.isin(cat
print('Data shape after one-hot encoding: ', evaluationData.shape)
```

```
Data shape after one-hot encoding:  (10268, 86)
```

In [17]:

```
#Scaling continous variable into 0-1 range for evaluation dataset


# transform data
result = scaler.transform(evaluationData[continuous_vars])
result = pd.DataFrame(result, columns=scaler.get_feature_names_out())

evaluationData = pd.concat([evaluationData.loc[:,~evaluationData.columns.isin(cont

print('Data shape after scaling: ', evaluationData.shape)
```

```
Data shape after scaling:  (10268, 86)
```

In [ ]:

```
excludeDesc_columns = ['patid', 'practice_id', #identifier
                   'BMI', #use the categorical instead
                   'ICS_medication_possesion_ratio', #the max value is inf
                   'Spacer', 'numPCSAsthma',  #all zero
                   'outcome_6months', 'outcome_9months', 'outcome_12months', 'outc
                   'outcome_21months', 'outcome_24months', 'outcome_combined_9month
                   'outcome_combined_15months', 'outcome_combined_18months', 'outc
                   'Country',#used for train-tes split only
                   ]
excludeDesc_columns = excludeDesc_columns + [x for x in features_columns if '_coun
descData = masterData[masterData.columns.difference(excludeDesc_columns)]
```

In [ ]:
```
summaryData = descData.describe().T
```

In [ ]:
```
cat_vars = summaryData[summaryData['max'] <= 5].index.to_list()
cat_vars.remove('numHospEvents')
cat_vars = cat_vars + categoricalNonnumericVars
cont_vars = summaryData[summaryData['max'] > 5].index.to_list() + ['numHospEvents'
```

In [ ]:
```
# writer = pd.ExcelWriter('../../code/descriptive_cat.xlsx', engine='xlsxwriter')
outcomes = ['outcome_combined_12months']
# cat_vars = ['gender_x', 'language', 'marital_status', 'ethnicity', 'admission_loc
for target_outcome in outcomes:
    desc_table = []
    print(target_outcome)
    for var in cat_vars:
        chi,pval,df,tab = chi2_contingency(pd.crosstab(descData[var].values, descDa
        desc_table.append((var  + ' (n, % of total)','-' ,'-'))
        for group in descData[var].unique():
            noAsthma = descData[(descData[var]==group)&(descData[target_outcome]==(
            noAsthmaPercent = round(noAsthma/sum(descData[target_outcome]==0)*100,:
            asthma = descData[(descData[var]==group)&(descData[target_outcome]==1)
            asthmaPercent = round(asthma/sum(descData[target_outcome]==1)*100,2)
            desc_table.append((group, str(noAsthma) + ' (' + str(noAsthmaPercent) +
    descriptive_cat = pd.DataFrame(desc_table, columns=['var','No asthma attack',
    # descriptive_cat.to_excel(writer, sheet_name=target_outcome)
    print('writing to Excel done!!')
# writer.save()
```

In [ ]:
```
for target_outcome in outcomes:
    desc_table_cont = []
    print(target_outcome)
    for var in descData.columns:
        if (var in cont_vars):
            tval,pval = ttest_ind(descData[var],descData[target_outcome])
            noAsthmaMean = np.round(np.mean(descData[var][descData[target_outcome]=
            noAsthmaSD = np.round(np.std(descData[var][descData[target_outcome]==0
            asthmaMean = np.round(np.mean(descData[var][descData[target_outcome]==:
            asthmaSD = np.round(np.std(descData[var][descData[target_outcome]==1])
            desc_table_cont.append((var  + ' (mean, std)', str(noAsthmaMean)  + '
    descriptive_cont = pd.DataFrame(desc_table_cont, columns=['var','No asthma att
    # descriptive_cont.to_excel(writer, sheet_name=target_outcome)
    print('writing to Excel done!!')
# writer.save()
```

In [ ]:
```
pd.concat([descriptive_cat, descriptive_cont]).to_csv(target_outcome + '.csv', ind
```

UTILS

In [18]:
```
#Model evaluation function

def summariseResult (testX, testY, model):
    preds = model.predict(testX)
    tn, fp, fn, tp = confusion_matrix(testY, preds).ravel()
    specificity = tn / (tn+fp)
    sensitivity = tp / (tp+fn)
    ppv = 100*tp/(tp+fp)
    npv = 100*tn/(fn+tn)
    acc = accuracy_score(testY, preds)
    f1score = f1_score(testY, preds, average = 'binary')
    balanceacc = balanced_accuracy_score(testY, preds)
    fpr, tpr, thresholds = roc_curve(testY, preds, pos_label=1)
    aucscore = auc(fpr, tpr)
    # auc = roc_auc_score(testY, preds)
    auprc = average_precision_score(testY, preds)
    # plot_confusion_matrix(model, testX, testY, cmap='viridis')
    return np.round(acc,4), np.round(specificity,4), np.round(sensitivity,4), np.r
```

In [19]:

```
#Fix model name for visualisation

def modelNameFixer(x):
    if 'liblinear' in x:
        return 'Lasso'
    elif 'GaussianNB' in x:
        return 'GNB'
    elif 'SVC' in x:
        return 'SVC'
    elif 'RandomForest' in x:
        return 'RF'
    elif 'XGB' in x:
        return 'XGBoost'
    elif 'DecisionTree' in x:
        return 'DT'
    else:
        return 'LR'
```

In [20]:

```
#Define number of split in k-fold

n_splits = 10
```

In [21]:

```
# instantiate the model (using the default parameters)
def build_models (X_train, y_train, params, split_counter):
    models = [] #list to store all the models
    model_counter = 0
    print("Building models . . . .")

    #LR
    lr_model = LogisticRegression(class_weight='balanced', penalty='l2', random_st
    lr_model.fit(X_train,y_train)
    modelname =str(split_counter) + 'LRModel'
    models.append([modelname, y_train.value_counts()[1]/y_train.value_counts()[0]]
    model_counter+=1
    pickle.dump(lr_model, open('./models/'+ target_outcome + '/'+ modelname + '.sa
    print("LR done")

    #Lasso
    lasso_model = LogisticRegression(class_weight='balanced', penalty='l1', solver
    lasso_model.fit(X_train, y_train)
    modelname =str(split_counter) + 'LassoModel'
    models.append([modelname, y_train.value_counts()[1]/y_train.value_counts()[0]]
    model_counter+=1
    pickle.dump(lasso_model, open('./models/'+ target_outcome + '/'+ modelname + '
    print("LR done")

    #GNB
    gnb_model = GaussianNB()
    gnb_model.fit(X_train, y_train)
    modelname =str(split_counter) + 'GNBModel'
    models.append([modelname, y_train.value_counts()[1]/y_train.value_counts()[0]]
    model_counter+=1
    pickle.dump(gnb_model, open('./models/'+ target_outcome + '/'+ modelname + '.s
    print("GNB done")

    #SVM
    svc_model = SVC(class_weight='balanced', C = 0.7, degree=2, kernel='poly', ran
    svc_model.fit(X_train,y_train)
    modelname =str(split_counter) + 'SVCModel'
    models.append([modelname, y_train.value_counts()[1]/y_train.value_counts()[0]]
    model_counter+=1
    pickle.dump(svc_model, open('./models/'+ target_outcome + '/'+ modelname + '.s
    print("SVM done")

    #DT
    dt_model = DecisionTreeClassifier(class_weight='balanced', random_state=1234)
    dt_model.fit(X_train, y_train)
    modelname =str(split_counter) + 'DTModel'
    models.append([modelname, y_train.value_counts()[1]/y_train.value_counts()[0]]
    model_counter+=1
    pickle.dump(dt_model, open('./models/'+ target_outcome + '/'+ modelname + '.sa
    print("DT done")

    #RF
    rf_model = RandomForestClassifier(class_weight='balanced', n_estimators=500, r
    rf_model.fit(X_train, y_train)
    modelname =str(split_counter) + 'RFModel'
    models.append([modelname, y_train.value_counts()[1]/y_train.value_counts()[0]]
    model_counter+=1
    pickle.dump(rf_model, open('./models/'+ target_outcome + '/'+ modelname + '.sa
    print("RF done")



    #XGB
    scale_pos_ratio = y_train.value_counts()[0]/y_train.value_counts()[1]
    xgb_model = xgb.XGBClassifier(objective ='binary:logistic', max_depth = params
                                 importance_type = 'gain', scale_pos_weight = scal
    # xgb_model = xgb.XGBClassifier(objective ='binary:logistic', learning_rate = 0
    xgb_model.fit(X_train,y_train)
    #save model
    modelname = str(split_counter) + 'XGBoostModel'
    models.append([modelname,  y_train.value_counts()[1]/y_train.value_counts()[0]
    pickle.dump(xgb_model, open('./models/'+ target_outcome + '/'+ modelname + '.s
    model_counter+=1
    print("XGB done")

    return models
    # return [xgb_model]
```

3months

In [49]:

```
#Create X set for model development

target_outcome = 'outcome_3months'
X = trainingData[features_columns]
y = trainingData[[target_outcome]]
print('X shape: ', X.shape)
print('y shape: ', y.shape)

#model parameters
params = {'xgb_lr': 0.6,
         'xgb_maxdepth': 7}
```

```
X shape:  (231121, 53)
y shape:  (231121, 1)
```

In [50]:

```
%%time

#EXECUTE model training

kf = StratifiedKFold(n_splits=n_splits, random_state=1234, shuffle=True)
kf.get_n_splits(X)
models1 = pd.DataFrame(columns=['modelname', 'class_ratio'])
summary_result1 = []
cols = ['model_name', 'class_ratio', 'acc','spec','sens','auc', 'auprc', 'balance_a
split_counter = 0

#train model
for train_index, test_index in kf.split(X, y):
    #split data
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    #Build models -> it can be commented if the models have been trained
    models_temp = pd.DataFrame(build_models(X_train, y_train[target_outcome], parar
    models1 = pd.concat([models1,models_temp]).reset_index(drop=True)
    split_counter+=1

#evaluate model
for modelname, classratio in models1.values:
    # print('=====================================================================
    print(modelname)
    model = pickle.load(open('./models/'+ target_outcome + '/'+ modelname + '.sav'
    summary_result1.append((str(model), classratio, ) + summariseResult (X_test, y_


summary_result1 = pd.DataFrame(summary_result1, columns=cols)
summary_result1['model_num'] = summary_result1.index
```

```
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
0LRModel
0LassoModel
0GNBModel
0SVCModel
0DTModel
0RFModel
0XGBoostModel
1LRModel
1LassoModel
1GNBModel
1SVCModel
1DTModel
1RFModel
1XGBoostModel
2LRModel
2LassoModel
2GNBModel
2SVCModel
2DTModel
2RFModel
2XGBoostModel
3LRModel
3LassoModel
3GNBModel
3SVCModel
3DTModel
3RFModel
3XGBoostModel
4LRModel
4LassoModel
4GNBModel
4SVCModel
4DTModel
4RFModel
4XGBoostModel
5LRModel
5LassoModel
5GNBModel
5SVCModel
5DTModel
5RFModel
5XGBoostModel
6LRModel
6LassoModel
6GNBModel
6SVCModel
6DTModel
6RFModel
6XGBoostModel
7LRModel
7LassoModel
7GNBModel
7SVCModel
7DTModel
7RFModel
7XGBoostModel
8LRModel
```

```
8LassoModel
8GNBModel
8SVCModel
8DTModel
8RFModel
8XGBoostModel
9LRModel
9LassoModel
9GNBModel
9SVCModel
9DTModel
9RFModel
9XGBoostModel
CPU times: user 1h 1min 58s, sys: 1min 37s, total: 1h 3min 36s
Wall time: 1h 4min 33s
```

In [51]:

```
print(target_outcome)
summary_result1['model_name'] = summary_result1.apply(lambda x: modelNameFixer(x.m
summary_result1.groupby('model_name').mean().sort_values(['auc'], ascending=False)
```

```
outcome_3months
```

Out[51]:

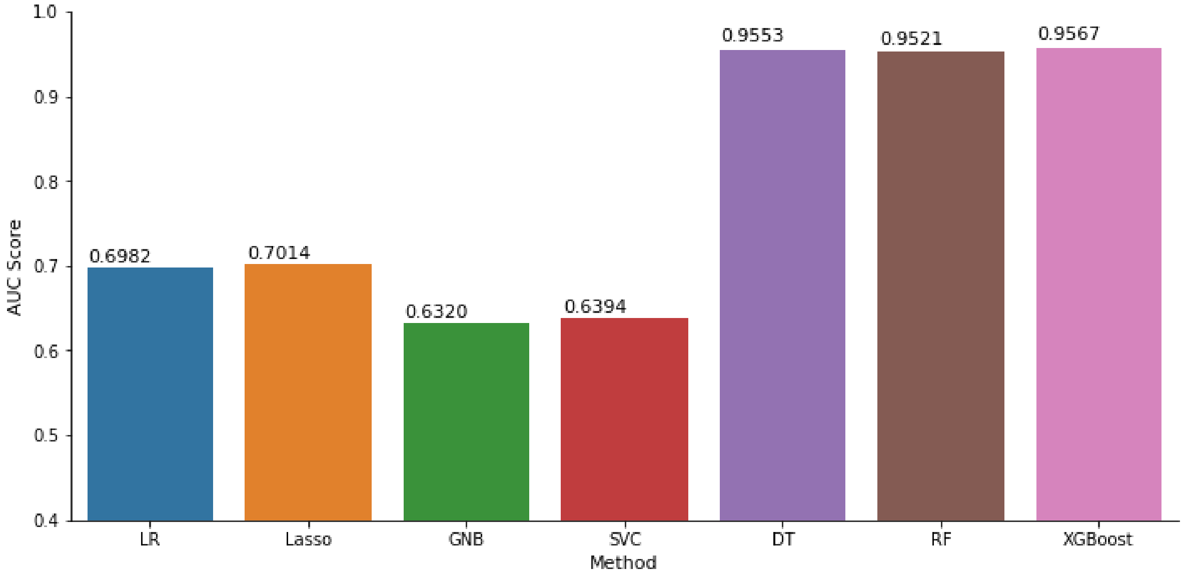
model_name	class_ratio	acc	spec	sens	auc	auprc	balance_accuracy	f1_score
XGBoost	0.065697	0.99182	0.99676	0.91656	0.95666	0.90875	0.95666	0.91965
DT	0.065697	0.98948	0.99428	0.91635	0.95532	0.90718	0.95532	0.91576
RF	0.065697	0.99393	0.99981	0.90449	0.95215	0.90807	0.95215	0.90805
Lasso	0.065697	0.75722	0.76507	0.63769	0.70137	0.11886	0.70137	0.24465
LR	0.065697	0.75650	0.76470	0.63173	0.69821	0.11742	0.69821	0.24239
SVC	0.065697	0.84904	0.87852	0.40042	0.63944	0.10825	0.63944	0.24649
GNB	0.065697	0.87659	0.91100	0.35307	0.63203	0.11289	0.63203	0.26080

In [52]:

```
summary_result1.to_csv("summaryResult_outcome1.csv")
summary_result1 = pd.read_csv("summaryResult_outcome1.csv")

bar = sns.catplot(x = "model_name",       # x variable name
            y = "auc",        # y variable name
            data = summary_result1,     # dataframe to plot
            kind = "bar",
            height=5,
            aspect=5/2.5,
            ci = None)
ax = bar.facet_axis(0,0)
for p in ax.patches:
    ax.text(p.get_x() + 0.01,
            p.get_height() * 1.01,
            '{0:.4f}'.format(p.get_height()),
            color='black', rotation='horizontal', fontsize=11)

# listOf_Yticks = np.arange(0.5, 0.7, 0.05)
ax.set_ylim(0.4, 1)
ax.set_ylabel('AUC Score', fontsize=11)
ax.set_xlabel('Method', fontsize=11)
```

Out[52]:

```
Text(0.5, 6.79999999999999, 'Method')
```

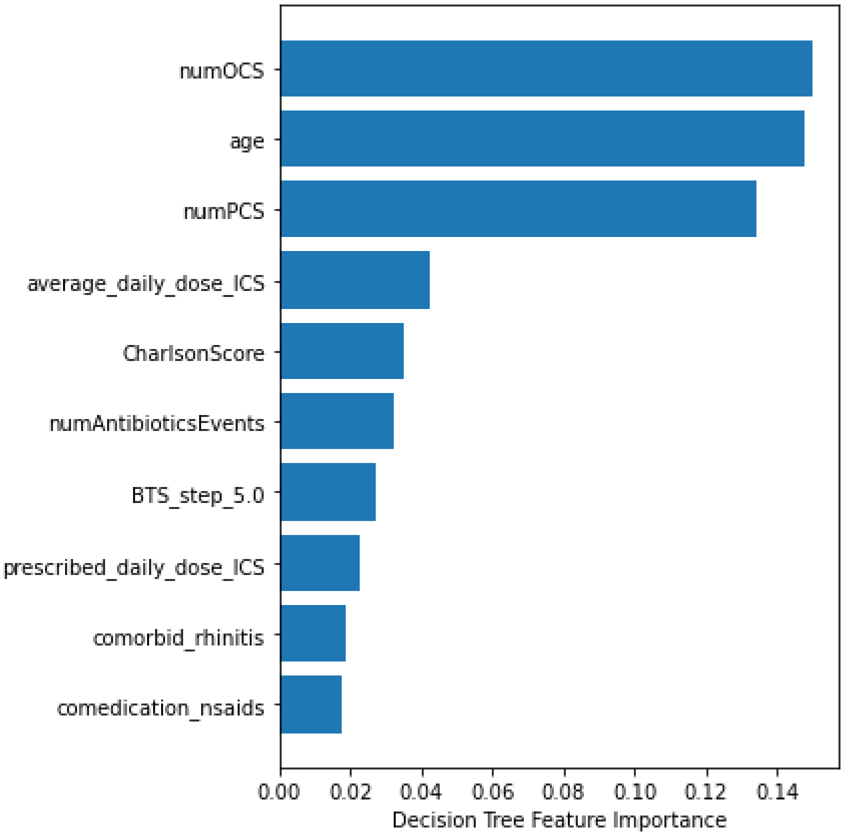


In [53]:

```
# kf = StratifiedKFold(n_splits=2, random_state=1234, shuffle=True)
# kf.get_n_splits(X)
# for train_index, test_index in kf.split(X, y):
#     #split data
#     X_train, X_test = X.iloc[train_index], X.iloc[test_index]
#     y_train, y_test = y.iloc[train_index], y.iloc[test_index]
#     trymodel = SVC(class_weight='balanced', C = 0.7, degree=2, kernel='poly', ran
#     trymodel.fit(X_train,y_train)
#     print(summariseResult(X_test, y_test, trymodel))
```

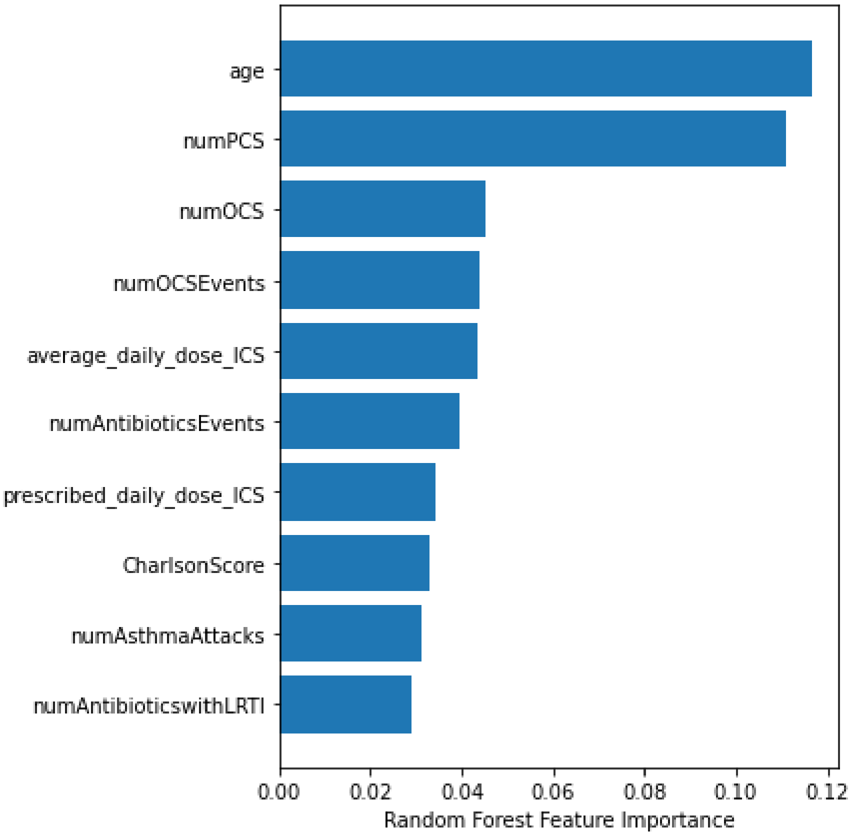
In [58]:

```
best_model1 = pickle.load(open('./models/outcome_3months/0DTModel.sav', 'rb'))

# pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_values
sorted_idx = best_model1.feature_importances_.argsort()
plt.figure(figsize=(5,7))
plt.barh(X.columns[sorted_idx][-10:], best_model1.feature_importances_[sorted_idx]
plt.xlabel("Decision Tree Feature Importance")
plt.show()
```

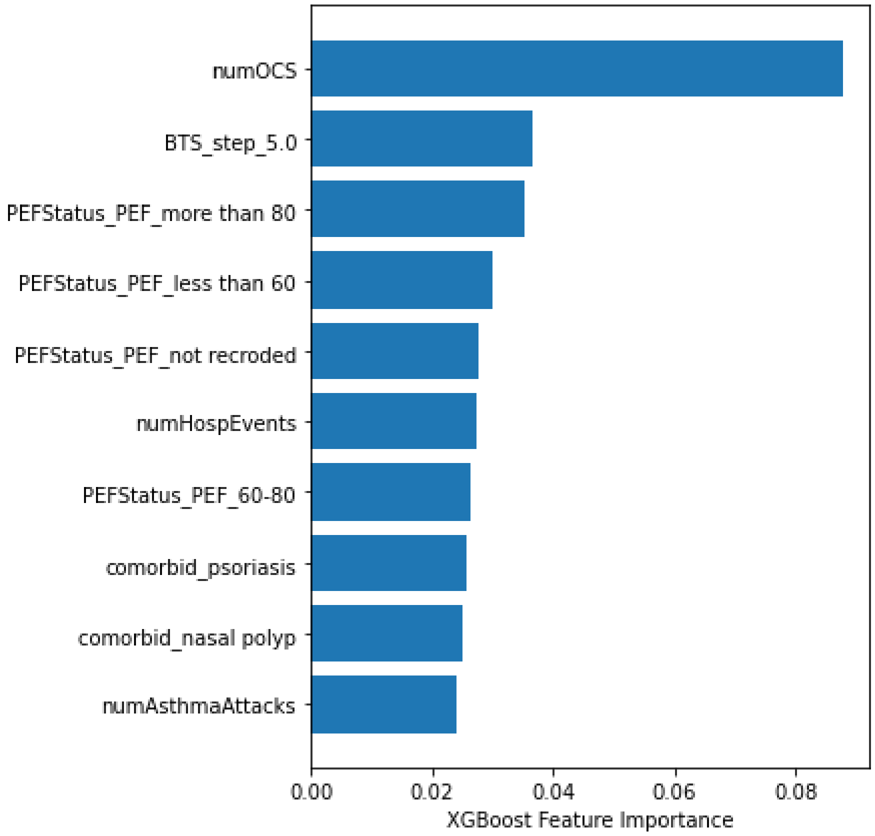


In [55]:

```
best_model1 = pickle.load(open('./models/outcome_3months/0RFModel.sav', 'rb'))

# pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_value
sorted_idx = best_model1.feature_importances_.argsort()
plt.figure(figsize=(5,7))
plt.barh(X.columns[sorted_idx][-10:], best_model1.feature_importances_[sorted_idx]
plt.xlabel("Random Forest Feature Importance")
plt.show()
```



In [59]:
```
best_model1 = pickle.load(open('./models/outcome_3months/0XGBoostModel.sav', 'rb')

# pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_values
sorted_idx = best_model1.feature_importances_.argsort()
plt.figure(figsize=(5,7))
plt.barh(X.columns[sorted_idx][-10:], best_model1.feature_importances_[sorted_idx]
plt.xlabel("XGBoost Feature Importance")
plt.show()
```



6months

In [33]:
```
target_outcome = 'outcome_combined_6months'
y = trainingData[[target_outcome]]

#model parameters
params = {'xgb_lr': 0.6,
          'xgb_maxdepth': 7}
```

In [34]:
```
%%time

#EXECUTE model training

kf = StratifiedKFold(n_splits=n_splits, random_state=1234, shuffle=True)
kf.get_n_splits(X)
models2 = pd.DataFrame(columns=['modelname', 'class_ratio'])
summary_result2 = []
cols = ['model_name', 'class_ratio', 'acc','spec','sens','auc', 'auprc', 'balance_
split_counter = 0

#train model
for train_index, test_index in kf.split(X, y):
    #split data
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    #Build models -> it can be commented if the models have been trained
    models_temp = pd.DataFrame(build_models(X_train, y_train[target_outcome], para
    models2 = pd.concat([models2,models_temp]).reset_index(drop=True)
    split_counter+=1

#evaluate model
for modelname, classratio in models2.values:
    # print('=================================================================
    print(modelname)
    model = pickle.load(open('./models/'+ target_outcome + '/'+ modelname + '.sav'
    summary_result2.append((str(model), classratio, ) + summariseResult (X_test, y_


summary_result2 = pd.DataFrame(summary_result2, columns=cols)
summary_result2['model_num'] = summary_result2.index
# summary_result1['method_name'] = summary_result1.apply(lambda x: 'LR' if x.model_
```

```
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
0LRModel
0LassoModel
0GNBModel
0SVCModel
0DTModel
0RFModel
0XGBoostModel
1LRModel
1LassoModel
1GNBModel
1SVCModel
1DTModel
1RFModel
1XGBoostModel
2LRModel
2LassoModel
2GNBModel
2SVCModel
2DTModel
2RFModel
2XGBoostModel
3LRModel
3LassoModel
3GNBModel
3SVCModel
3DTModel
3RFModel
3XGBoostModel
4LRModel
4LassoModel
4GNBModel
4SVCModel
4DTModel
4RFModel
4XGBoostModel
5LRModel
5LassoModel
5GNBModel
5SVCModel
5DTModel
5RFModel
5XGBoostModel
6LRModel
6LassoModel
6GNBModel
6SVCModel
6DTModel
6RFModel
6XGBoostModel
7LRModel
7LassoModel
7GNBModel
7SVCModel
7DTModel
7RFModel
7XGBoostModel
8LRModel
```

```
8LassoModel
8GNBModel
8SVCModel
8DTModel
8RFModel
8XGBoostModel
9LRModel
9LassoModel
9GNBModel
9SVCModel
9DTModel
9RFModel
9XGBoostModel
CPU times: user 1h 11min 28s, sys: 2min 6s, total: 1h 13min 34s
Wall time: 1h 14min 55s
```

In [35]:

```
print(target_outcome)
summary_result2['model_name'] = summary_result2.apply(lambda x: modelNameFixer(x.m
summary_result2.groupby('model_name').mean().sort_values(['auc'], ascending=False)
```

```
outcome_combined_6months
```

Out[35]:

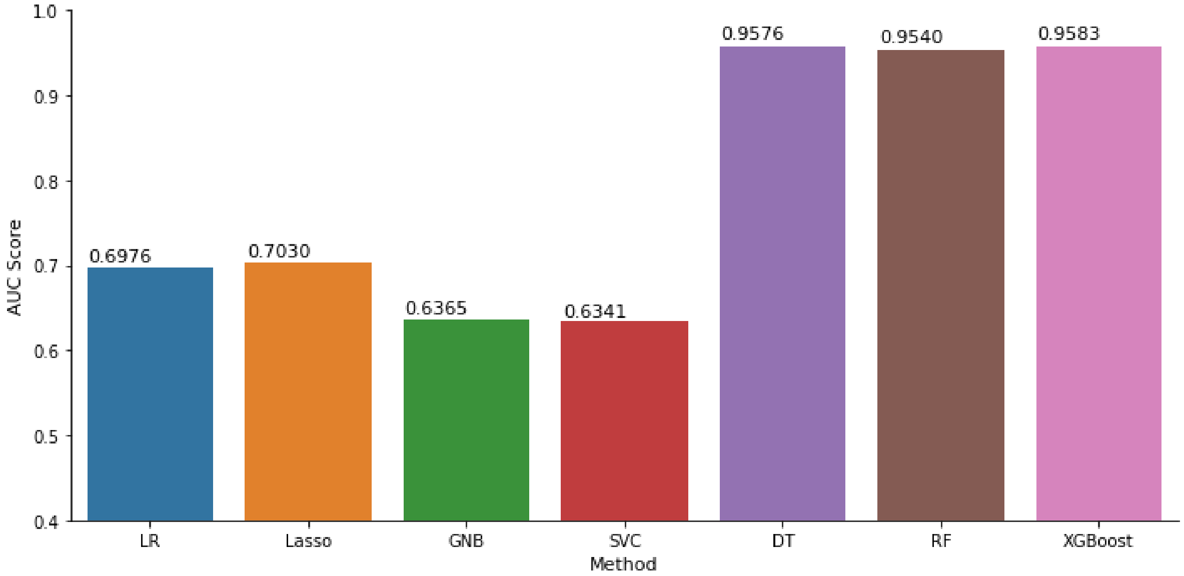
	class_ratio	acc	spec	sens	auc	auprc	balance_accuracy	f1_score
model_name								
XGBoost	0.102161	0.98771	0.99442	0.92217	0.95829	0.91211	0.95829	0.92447
DT	0.102161	0.98565	0.99204	0.92315	0.95759	0.91203	0.95759	0.92287
RF	0.102161	0.99114	0.99959	0.90849	0.95404	0.91383	0.95404	0.91493
Lasso	0.102161	0.75989	0.77285	0.63308	0.70299	0.17438	0.70299	0.32839
LR	0.102161	0.76090	0.77531	0.61998	0.69763	0.17161	0.69763	0.32472
GNB	0.102161	0.85759	0.90792	0.36503	0.63648	0.16414	0.63648	0.32220
SVC	0.102161	0.83690	0.88307	0.38516	0.63410	0.15400	0.63410	0.30454

In [36]:

```
summary_result2.to_csv("summaryResult_outcome2.csv")
summary_result2 = pd.read_csv("summaryResult_outcome2.csv")

bar = sns.catplot(x = "model_name",       # x variable name
            y = "auc",        # y variable name
            data = summary_result2,     # dataframe to plot
            kind = "bar",
            height=5,
            aspect=5/2.5,
            ci = None)
ax = bar.facet_axis(0,0)
for p in ax.patches:
    ax.text(p.get_x() + 0.01,
            p.get_height() * 1.01,
            '{0:.4f}'.format(p.get_height()),
            color='black', rotation='horizontal', fontsize=11)

# listOf_Yticks = np.arange(0.5, 0.7, 0.05)
ax.set_ylim(0.4, 1)
ax.set_ylabel('AUC Score', fontsize=11)
ax.set_xlabel('Method', fontsize=11)
```

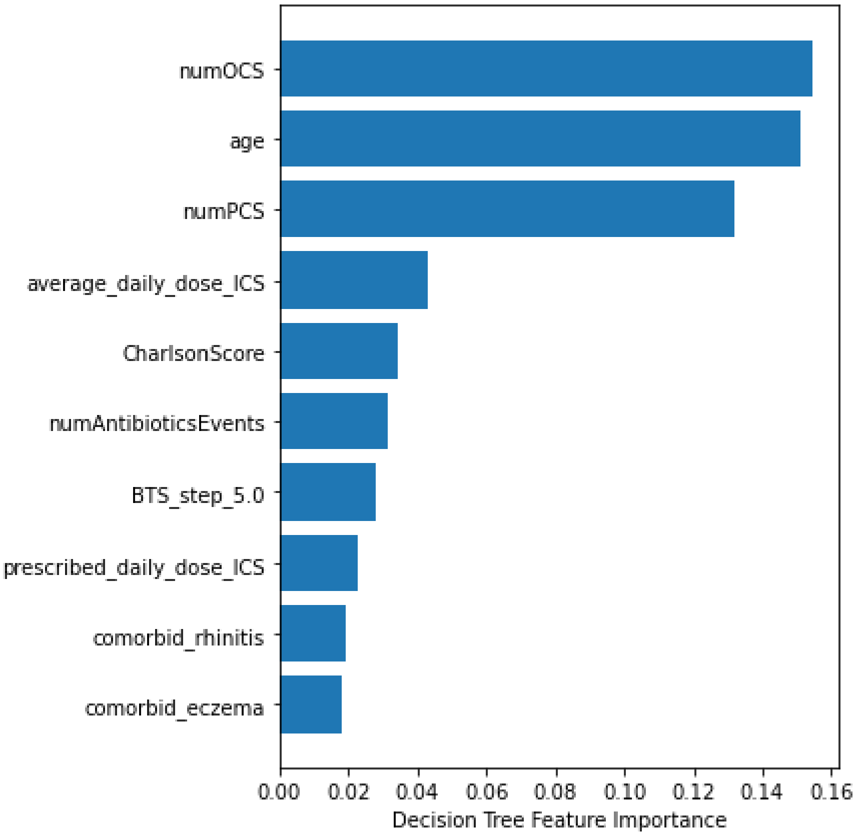
Out[36]:

```
Text(0.5, 6.79999999999999, 'Method')
```



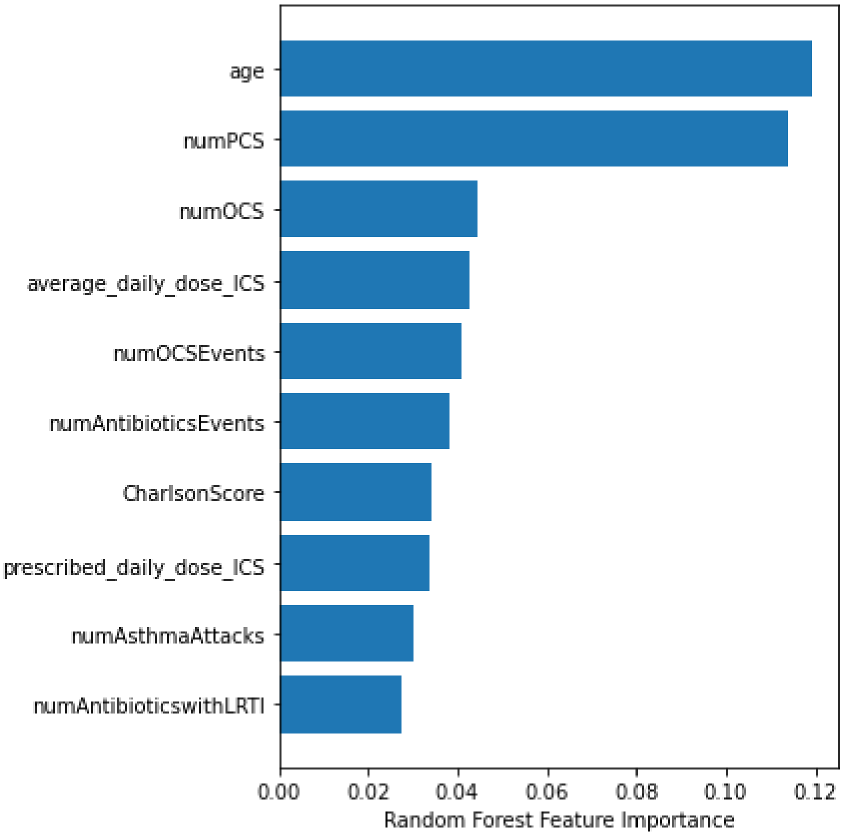
In [60]:

```
best_model2 = pickle.load(open('./models/outcome_combined_6months/0DTModel.sav', '

# pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_value
sorted_idx = best_model2.feature_importances_.argsort()
plt.figure(figsize=(5,7))
plt.barh(X.columns[sorted_idx][-10:], best_model2.feature_importances_[sorted_idx]
plt.xlabel("Decision Tree Feature Importance")
plt.show()
```

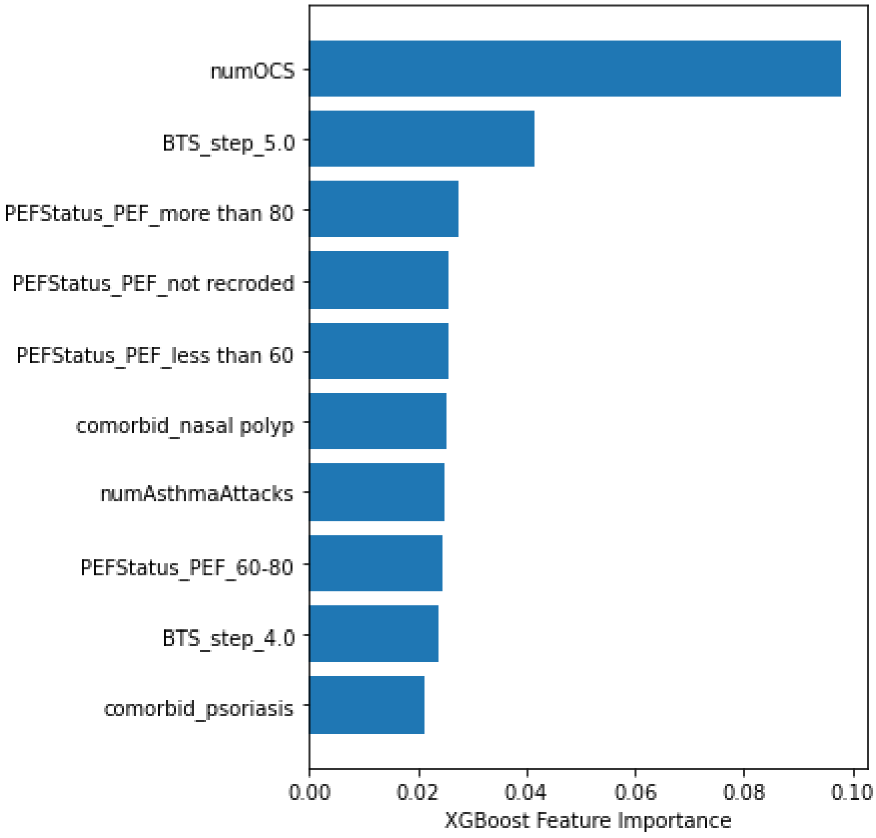


In [38]:

```
best_model2 = pickle.load(open('./models/outcome_combined_6months/0RFModel.sav', '

# pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_value
sorted_idx = best_model2.feature_importances_.argsort()
plt.figure(figsize=(5,7))
plt.barh(X.columns[sorted_idx][-10:], best_model2.feature_importances_[sorted_idx]
plt.xlabel("Random Forest Feature Importance")
plt.show()
```



In [61]:
```
best_model2 = pickle.load(open('./models/outcome_combined_6months/0XGBoostModel.sa

# pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_value
sorted_idx = best_model2.feature_importances_.argsort()
plt.figure(figsize=(5,7))
plt.barh(X.columns[sorted_idx][-10:], best_model2.feature_importances_[sorted_idx]
plt.xlabel("XGBoost Feature Importance")
plt.show()
```



12 months

In [42]:

```
target_outcome = 'outcome_combined_12months'
y = trainingData[[target_outcome]]

#model parameters
params = {'xgb_lr': 0.6,
          'xgb_maxdepth': 10}
```

In [43]:

```
%%time

#EXECUTE model training

kf = StratifiedKFold(n_splits=n_splits, random_state=1234, shuffle=True)
kf.get_n_splits(X)
models3 = pd.DataFrame(columns=['modelname', 'class_ratio'])
summary_result3 = []
cols = ['model_name', 'class_ratio', 'acc','spec','sens','auc', 'auprc', 'balance_a
split_counter = 0

#train model
for train_index, test_index in kf.split(X, y):
    #split data
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    #Build models -> it can be commented if the models have been trained
    models_temp = pd.DataFrame(build_models(X_train, y_train[target_outcome], paran
    models3 = pd.concat([models3,models_temp]).reset_index(drop=True)
    split_counter+=1

#evaluate model
for modelname, classratio in models3.values:
    # print('=====================================================================
    print(modelname)
    model = pickle.load(open('./models/'+ target_outcome + '/'+ modelname + '.sav'
    summary_result3.append((str(model), classratio, ) + summariseResult (X_test, y_


summary_result3 = pd.DataFrame(summary_result3, columns=cols)
summary_result3['model_num'] = summary_result3.index
# summary_result1['method_name'] = summary_result1.apply(lambda x: 'LR' if x.model_
```

```
Building models . . . .
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
Building models . . . .
```

```
/opt/conda/envs/rapids/lib/python3.8/site-packages/sklearn/linear_model/_logistic.
py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR done
LR done
GNB done
SVM done
DT done
RF done
XGB done
0LRModel
0LassoModel
0GNBModel
0SVCModel
0DTModel
0RFModel
0XGBoostModel
1LRModel
1LassoModel
1GNBModel
1SVCModel
1DTModel
1RFModel
1XGBoostModel
2LRModel
2LassoModel
2GNBModel
2SVCModel
2DTModel
2RFModel
2XGBoostModel
3LRModel
3LassoModel
3GNBModel
3SVCModel
3DTModel
3RFModel
3XGBoostModel
4LRModel
4LassoModel
4GNBModel
4SVCModel
4DTModel
4RFModel
4XGBoostModel
5LRModel
5LassoModel
5GNBModel
5SVCModel
5DTModel
5RFModel
5XGBoostModel
6LRModel
6LassoModel
6GNBModel
6SVCModel
6DTModel
6RFModel
6XGBoostModel
7LRModel
7LassoModel
7GNBModel
7SVCModel
7DTModel
7RFModel
7XGBoostModel
8LRModel
```

```
8LassoModel
8GNBModel
8SVCModel
8DTModel
8RFModel
8XGBoostModel
9LRModel
9LassoModel
9GNBModel
9SVCModel
9DTModel
9RFModel
9XGBoostModel
CPU times: user 1h 18min 45s, sys: 3min 33s, total: 1h 22min 19s
Wall time: 1h 24min
```

In [44]:

```python
print(target_outcome)
summary_result3['model_name'] = summary_result3.apply(lambda x: modelNameFixer(x.m
summary_result3.groupby('model_name').mean().sort_values(['auc'], ascending=False)
```

```
outcome_combined_12months
```

Out[44]:

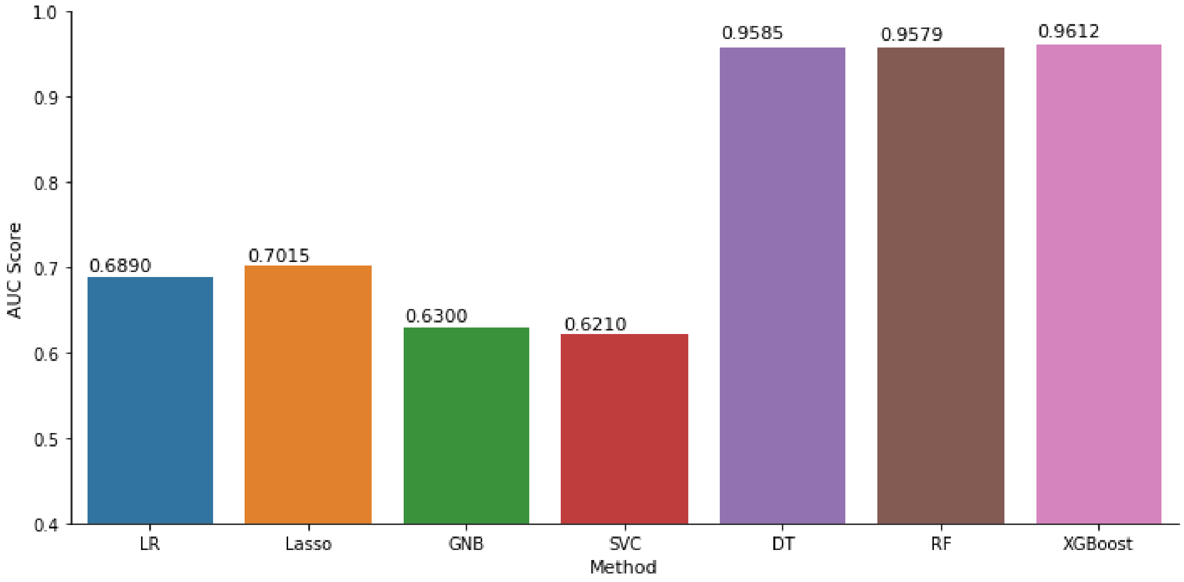
	class_ratio	acc	spec	sens	auc	auprc	balance_accuracy	f1_score
model_name								
XGBoost	0.186556	0.98205	0.99162	0.93068	0.96117	0.92242	0.96117	0.93460
DT	0.186556	0.97807	0.98705	0.92986	0.95848	0.91898	0.95848	0.92956
RF	0.186556	0.98546	0.99812	0.91756	0.95786	0.92349	0.95786	0.92719
Lasso	0.186556	0.75600	0.78099	0.62207	0.70152	0.27490	0.70152	0.44497
LR	0.186556	0.74390	0.76906	0.60891	0.68899	0.26230	0.68899	0.42782
GNB	0.186556	0.82133	0.90913	0.35089	0.62999	0.24897	0.62999	0.38181
SVC	0.186556	0.80298	0.88647	0.35558	0.62103	0.23245	0.62103	0.36208

In [45]:

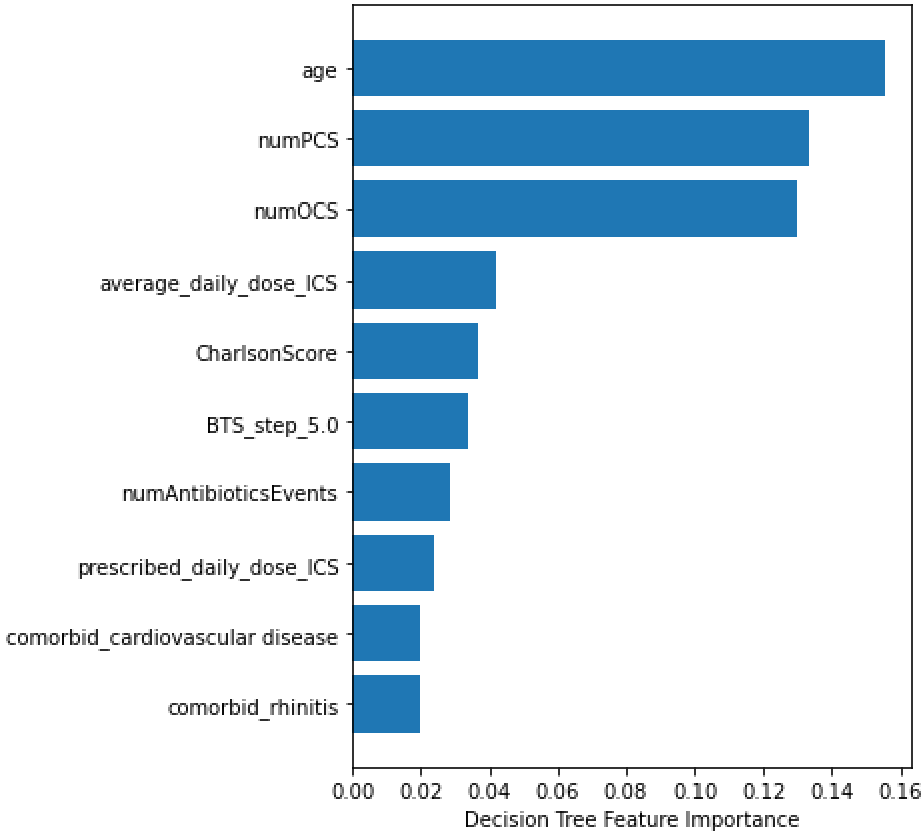
```python
summary_result3.to_csv("summaryResult_outcome3.csv")
summary_result3 = pd.read_csv("summaryResult_outcome3.csv")

bar = sns.catplot(x = "model_name",       # x variable name
            y = "auc",        # y variable name
            data = summary_result3,     # dataframe to plot
            kind = "bar",
            height=5,
            aspect=5/2.5,
            ci = None)
ax = bar.facet_axis(0,0)
for p in ax.patches:
    ax.text(p.get_x() + 0.01,
            p.get_height() * 1.01,
            '{0:.4f}'.format(p.get_height()),
            color='black', rotation='horizontal', fontsize=11)

# listOf_Yticks = np.arange(0.5, 0.7, 0.05)
ax.set_ylim(0.4, 1)
ax.set_ylabel('AUC Score', fontsize=11)
ax.set_xlabel('Method', fontsize=11)
```

Out[45]:

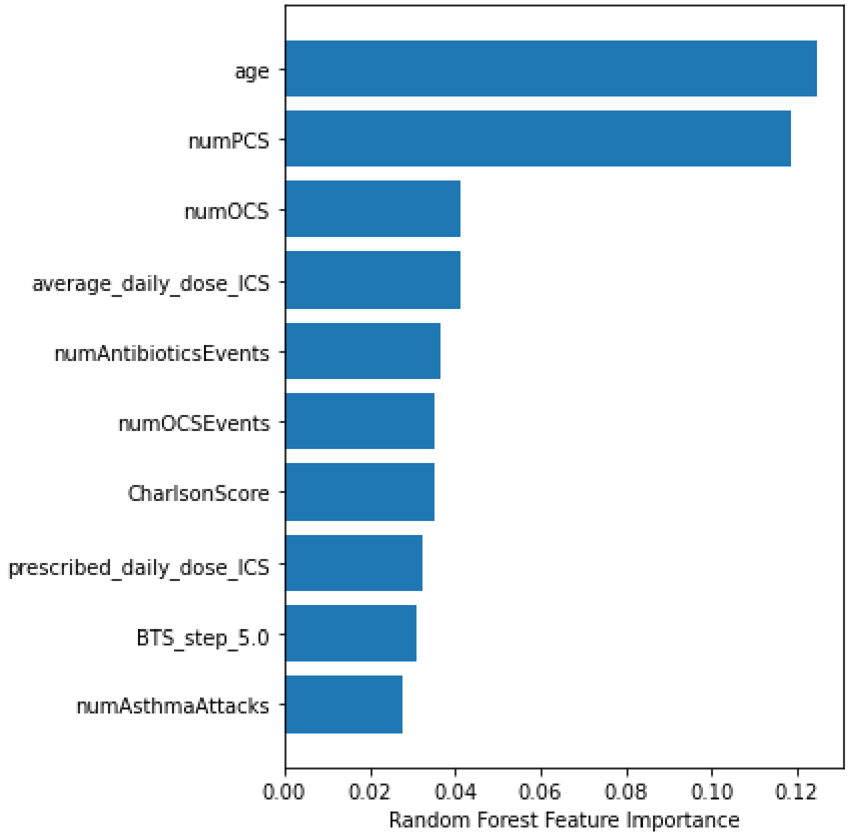
```
Text(0.5, 6.79999999999999, 'Method')
```



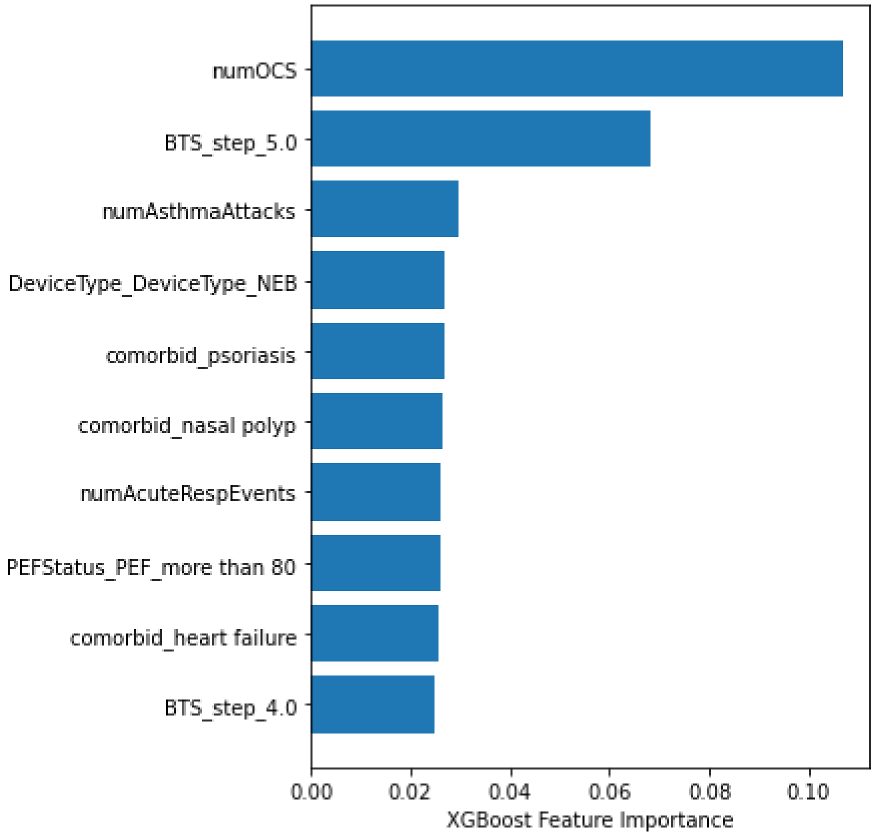
In [62]:
```
best_model3 = pickle.load(open('./models/outcome_combined_12months/0DTModel.sav',

# pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_value
sorted_idx = best_model3.feature_importances_.argsort()
plt.figure(figsize=(5,7))
plt.barh(X.columns[sorted_idx][-10:], best_model3.feature_importances_[sorted_idx]
plt.xlabel("Decision Tree Feature Importance")
plt.show()
```



In [47]:
```
best_model3 = pickle.load(open('./models/outcome_combined_12months/0RFModel.sav',

# pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_value
sorted_idx = best_model3.feature_importances_.argsort()
plt.figure(figsize=(5,7))
plt.barh(X.columns[sorted_idx][-10:], best_model3.feature_importances_[sorted_idx]
plt.xlabel("Random Forest Feature Importance")
plt.show()
```



```
In [63]: best_model3 = pickle.load(open('./models/outcome_combined_12months/0XGBoostModel.sa

         # pd.DataFrame([best_model3.feature_importances_], columns=X.columns).T.sort_value
         sorted_idx = best_model3.feature_importances_.argsort()
         plt.figure(figsize=(5,7))
         plt.barh(X.columns[sorted_idx][-10:], best_model3.feature_importances_[sorted_idx]
         plt.xlabel("XGBoost Feature Importance")
         plt.show()
```


numOCS
BTS_step_5.0
numAsthmaAttacks
DeviceType_DeviceType_NEB
comorbid_psoriasis
comorbid_nasal polyp
numAcuteRespEvents
PEFStatus_PEF_more than 80
comorbid_heart failure
BTS_step_4.0
XGBoost Feature Importance