Machine Learning em Saúde

Professor: Alexandre Chiavegatto Filho

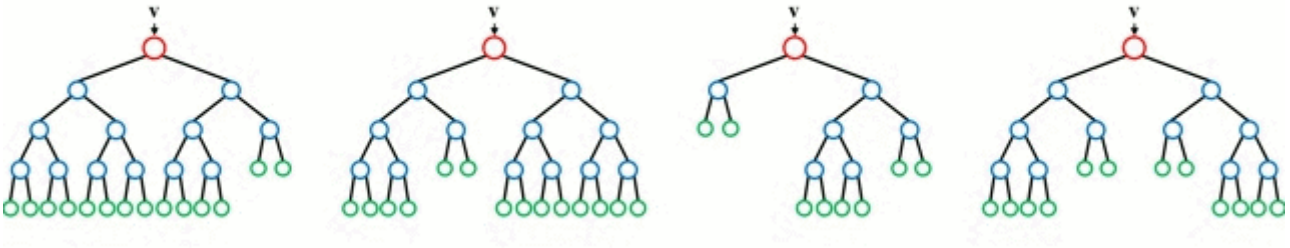
Monitoria: André Filipe de Moraes Batista

Fevereiro 2019

Classificação com Random Forest em Python

com alguns comandos comparados a comandos em R

Animação - Algoritmo Random Forest



Instalação de Pacotes necessários

In [0]:
```
### instala pacote dfply (dplyr-style Data Manipulation with Pipes in Python)
!pip install dfply

### pacote scikit-plot
!pip install scikit-plot
```

```
Collecting dfply
  Downloading https://files.pythonhosted.org/packages/53/91/18ab48c64661252dadff685f8ddbc6f456302923918f488714ee2345d49b/dfply-0.3.3-py3-none-any.whl (612kB)
    100% |████████████████████████████████| 614kB 19.5MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from dfply) (1.14.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from dfply) (0.22.0)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas->dfply) (2.5.3)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas->dfply) (2018.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas->dfply) (1.11.0)
Installing collected packages: dfply
Successfully installed dfply-0.3.3
Collecting scikit-plot
  Downloading https://files.pythonhosted.org/packages/7c/47/32520e259340c140a4ad27c1b97050dd3254fdc517b1d59974d47037510e/scikit_plot-0.3.7-py3-none-any.whl
Requirement already satisfied: matplotlib>=1.4.0 in /usr/local/lib/python3.6/dist-packages (from scikit-plot) (3.0.2)
Requirement already satisfied: joblib>=0.10 in /usr/local/lib/python3.6/dist-packages (from scikit-plot) (0.13.1)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.6/dist-packages (from scikit-plot) (0.20.2)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.6/dist-packages (from scikit-plot) (1.1.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=1.4.0->scikit-plot) (2.5.3)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=1.4.0->scikit-plot) (2.3.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.0.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=1.4.0->scikit-plot) (0.10.0)
Requirement already satisfied: numpy>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.14.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.1->matplotlib>=1.4.0->scikit-plot) (1.11.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from kiwisolver>=1.0.1->matplotlib>=1.4.0->scikit-plot) (40.8.0)
Installing collected packages: scikit-plot
Successfully installed scikit-plot-0.3.7
```

Importação de Bibliotecas

In [0]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from dfply import *
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import KFold, cross_val_score
from sklearn.model_selection import train_test_split, GridSearchCV, Rand
omizedSearchCV
from sklearn.ensemble import RandomForestClassifier
import time
from sklearn.metrics import classification_report
import random
import scikitplot as skplt
import warnings
warnings.filterwarnings("ignore")
```

In [0]:
```

```

Arquivo de dados

Fonte dos dados: https://github.com/laderast/cvdRiskData (https://github.com/laderast/cvdRiskData)

In [0]:
```
data = pd.read_csv('https://raw.githubusercontent.com/laderast/cvdRiskDa
ta/master/data-raw/fullPatientData.csv')
```

In [0]:
```
data.head(10)
```

Out[0]:

	patientID	age	htn	treat	smoking	race	t2d	gender	numAge	bmi	tchol	sbp
0	HHUID00076230	20-40	Y	Y	N	Asian/PI	N	M	21	26	176	179
1	HHUID00547835	70-90	N	N	N	White	N	M	86	23	244	123
2	HHUID00450841	20-40	Y	Y	N	White	N	M	29	22	189	165
3	HHUID00380788	20-40	Y	Y	N	White	N	M	24	24	218	172
4	HHUID00043423	20-40	N	N	N	Asian/PI	N	M	40	20	207	111
5	HHUID00103400	20-40	N	N	N	Asian/PI	N	M	35	24	192	118
6	HHUID00876576	40-55	N	N	Y	White	N	M	50	19	176	117
7	HHUID00383948	55-70	N	N	N	White	N	M	59	23	178	113
8	HHUID00324837	0-20	N	N	N	White	N	M	13	21	159	120
9	HHUID00688471	0-20	N	N	N	Asian/PI	N	M	12	23	178	112

Transformação de Dados

Comando em R:

```
cvd_patient_r <- cvd_patient_c %>%
        filter(numAge>55) %>%
        select(-c(patientID,age,treat))
```

em Python:

In [0]:
```
cvd_patient = (data >>
               mask(X.numAge > 55) >>
               drop(X.patientID, X.age, X.treat)
              )
```

In [0]:
```
cvd_patient.head(5)
```

Out[0]:

	htn	smoking	race	t2d	gender	numAge	bmi	tchol	sbp	cvd
1	N	N	White	N	M	86	23	244	123	N
7	N	N	White	N	M	59	23	178	113	N
10	N	N	Asian/PI	N	M	69	19	181	118	N
11	Y	Y	White	N	M	56	21	188	183	N
12	N	N	White	N	M	64	16	240	111	N

Dummies

Em R:

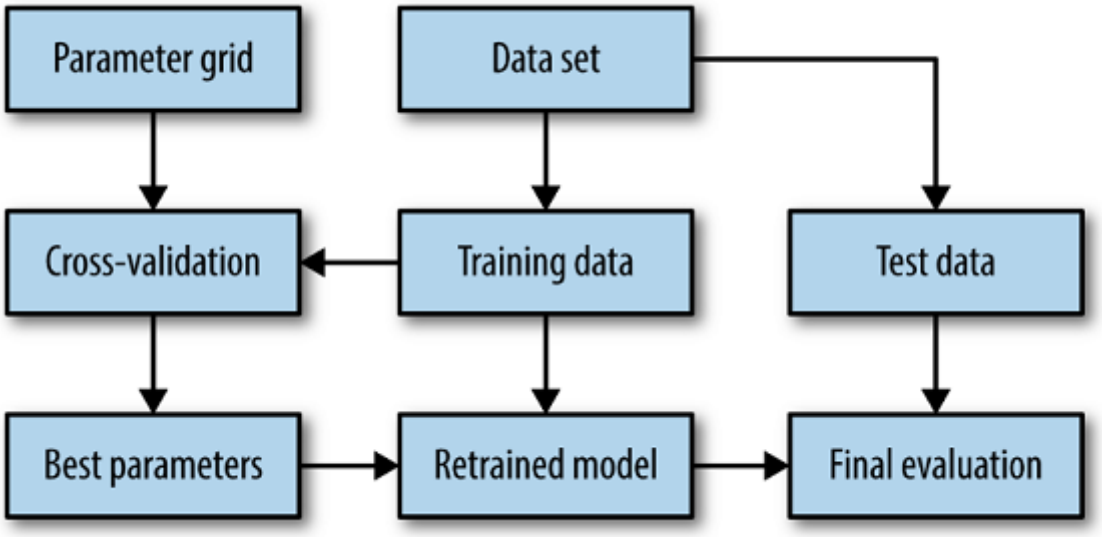
```
race_dummies <- dummy.data.frame(select(cvd_patient_r,race), names=names(sel
ect(cvd_patient_r,race)), sep="_")
names(race_dummies)
names(race_dummies)[2]<-"race_Asian_PI"
names(race_dummies)[3]<-"race_Black_AfAm"
banco_final<-cbind(cvd_patient_r,race_dummies)
banco_final_r<-select(banco_final,-c(race,race_White))
banco_final_r$race_AmInd<-as.factor(banco_final_r$race_AmInd)
banco_final_r$race_Asian_PI<-as.factor(banco_final_r$race_Asian_PI)
banco_final_r$race_Black_AfAm<-as.factor(banco_final_r$race_Black_AfAm)
```

Em Python:

In [0]:
```
cvd_patient = pd.get_dummies(cvd_patient, columns=['race'], drop_first=T
rue)
```

In [0]:
```
### scikit exige que demais categoricas sejam dummies


cvd_patient = pd.get_dummies(cvd_patient, columns=['htn', 'smoking', 't2
d', 'gender'], drop_first=True)
```

Entendendo o conceito



Conjunto de treinamento e de teste

Em R:

```
set.seed(1)
split1 <- createDataPartition(banco_final_r$cvd, p=.85)[[1]]
trainData <- banco_final_r[split1,]
testData <- banco_final_r[-split1,]
prop.table(table(trainData$cvd))
prop.table(table(testData$cvd))
```

Em Python:

In [0]:
```
feature_space = cvd_patient.iloc[:, cvd_patient.columns != 'cvd']
feature_class = cvd_patient.iloc[:, cvd_patient.columns == 'cvd']
X_train, X_test, y_train, y_test = train_test_split(feature_space,
                                                    feat
ure_class,
                                                    trai
n_size = 0.85,
                                                    rand
om_state = 42)
```

Random Forest Classifier

In [0]:
```
# criando a random forest

fit_rf = RandomForestClassifier(random_state=42, verbose=1)

#vendo os parametros que sao aceitos pela RF
fit_rf.get_params()
```

Out[0]:
```
{'bootstrap': True,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 'warn',
 'n_jobs': None,
 'oob_score': False,
 'random_state': 42,
 'verbose': 1,
 'warm_start': False}
```

In [0]:
```
X_train.shape
```

Out[0]:
```
(106534, 11)
```

In [0]:
```
X_test.shape
```

Out[0]:
```
(18801, 11)
```

Grid Search / RandomSearchCV

In [0]:
```
np.random.seed(42)

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1000, nu
m = 5)]
# Number of features to consider at every split
max_features = ['log2', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 20, num = 5)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the param grid
param_dist = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}

cv_rf = RandomizedSearchCV(fit_rf, n_iter=20, cv=5, verbose=5, param_dis
tributions=param_dist,
                     n_jobs = 1)

### Caso deseje rodar GridSearchCV, troque a linha acima pela que segue

#cv_rf = GridSearchCV(fit_rf, cv=10, param_grid=param_dist,
##                      n_jobs = -1)
```

In [0]:
```
print('Starting (Random) Grid Search ... \n')

cv_rf.fit(X_train, np.ravel(y_train), )
print('Best Parameters using grid search: \n',
      cv_rf.best_params_)
```

```
Starting (Random) Grid Search ...

Fitting 5 folds for each of 20 candidates, totalling 100 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.

[CV] n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    6.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.8s finished
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    7.5s remaining:
0.0s

[CV]  n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True, score=0.7613929694466607, tot
al=   6.5s
[CV] n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    6.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.8s finished
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:   14.9s remaining:
0.0s

[CV]  n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True, score=0.760782841319754, tota
l=   6.4s
[CV] n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    6.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.8s finished
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:   22.4s remaining:
0.0s
```

```
[CV]  n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True, score=0.7646313418125499, tot
al=   6.5s
[CV] n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    6.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.8s finished
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:   29.7s remaining:
0.0s

[CV]  n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True, score=0.7673534519172103, tot
al=   6.4s
[CV] n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    6.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.8s finished

[CV]  n_estimators=100, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True, score=0.7605838730873932, tot
al=   6.4s
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.9s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_
features=sqrt, max_depth=8, bootstrap=False, score=0.7610644389167879,
total= 1.4min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=False
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.9s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_
features=sqrt, max_depth=8, bootstrap=False, score=0.7597503167972967,
total= 1.4min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.8s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_
features=sqrt, max_depth=8, bootstrap=False, score=0.7638804148871263,
total= 1.4min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.9s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_
features=sqrt, max_depth=8, bootstrap=False, score=0.7662270615290749,
total= 1.4min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    8.0s finished
```

```
[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=4, max_
features=sqrt, max_depth=8, bootstrap=False, score=0.7607716136299634,
total= 1.4min
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=log2, max_depth=20, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:   14.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    2.1s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=log2, max_depth=20, bootstrap=False, score=0.7564650114985686,
total=  15.7s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=log2, max_depth=20, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:   15.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    2.1s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=log2, max_depth=20, bootstrap=False, score=0.75045759609518, to
tal=  15.7s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=log2, max_depth=20, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:   14.8s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    2.1s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=log2, max_depth=20, bootstrap=False, score=0.7588585910733562,
total=  15.5s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=log2, max_depth=20, bootstrap=False
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:   14.8s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    2.1s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=log2, max_depth=20, bootstrap=False, score=0.7582484629464495,
total=  15.5s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=log2, max_depth=20, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:   14.7s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    2.1s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=log2, max_depth=20, bootstrap=False, score=0.7513376513658124,
total=  15.5s
[CV] n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:   32.7s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    4.3s finished

[CV]  n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.760876707185432, tota
l=  34.2s
[CV] n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:   32.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    4.3s finished
```

```
[CV]  n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.7603604449242033, tot
al=  34.5s
[CV] n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:   33.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    4.3s finished

[CV]  n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.7648190735439058, tot
al=  34.5s
[CV] n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:   32.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    4.3s finished

[CV]  n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.7667433237903036, tot
al=  33.7s
[CV] n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:   33.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    4.3s finished

[CV]  n_estimators=550, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.760912419036891, tota
l=  34.5s
[CV] n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fea
tures=log2, max_depth=16, bootstrap=True
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:   32.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    1.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    6.7s finished

[CV]  n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fe
atures=log2, max_depth=16, bootstrap=True, score=0.7598441826629746, to
tal=  34.3s
[CV] n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fea
tures=log2, max_depth=16, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:   32.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    1.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    6.7s finished

[CV]  n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fe
atures=log2, max_depth=16, bootstrap=True, score=0.7554324869761111, to
tal=  34.1s
[CV] n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fea
tures=log2, max_depth=16, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:   32.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    1.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    6.7s finished

[CV]  n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fe
atures=log2, max_depth=16, bootstrap=True, score=0.7625193598347961, to
tal=  33.8s
[CV] n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fea
tures=log2, max_depth=16, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:   31.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    1.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    5.8s finished
```

```
[CV]  n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fe
atures=log2, max_depth=16, bootstrap=True, score=0.7624724269019572, to
tal=  33.7s
[CV] n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fea
tures=log2, max_depth=16, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:   32.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    1.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 325 out of 325 | elapsed:    6.7s finished

[CV]  n_estimators=325, min_samples_split=2, min_samples_leaf=2, max_fe
atures=log2, max_depth=16, bootstrap=True, score=0.756406646015207, tot
al=  34.7s
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_f
eatures=sqrt, max_depth=20, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.8min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    5.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   20.9s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_
features=sqrt, max_depth=20, bootstrap=True, score=0.759093255737551, t
otal= 1.9min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_f
eatures=sqrt, max_depth=20, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.8min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    5.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   21.0s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_
features=sqrt, max_depth=20, bootstrap=True, score=0.7533205049983573,
total= 1.9min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_f
eatures=sqrt, max_depth=20, bootstrap=True
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.8min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    5.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   21.4s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_
features=sqrt, max_depth=20, bootstrap=True, score=0.7597503167972967,
total= 1.9min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_f
eatures=sqrt, max_depth=20, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.8min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    5.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   20.1s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_
features=sqrt, max_depth=20, bootstrap=True, score=0.7605951095883982,
total= 1.9min
[CV] n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_f
eatures=sqrt, max_depth=20, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.8min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    5.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   20.7s finished

[CV]  n_estimators=1000, min_samples_split=10, min_samples_leaf=2, max_
features=sqrt, max_depth=20, bootstrap=True, score=0.7540129540974373,
total= 1.9min
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_fe
atures=log2, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished
```

```
[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_f
eatures=log2, max_depth=5, bootstrap=True, score=0.757403670155348, tot
al=   4.9s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_fe
atures=log2, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_f
eatures=log2, max_depth=5, bootstrap=True, score=0.7547284929835265, to
tal=   4.8s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_fe
atures=log2, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_f
eatures=log2, max_depth=5, bootstrap=True, score=0.7612052377153048, to
tal=   4.8s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_fe
atures=log2, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_f
eatures=log2, max_depth=5, bootstrap=True, score=0.7599380485286525, to
tal=   4.8s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_fe
atures=log2, max_depth=5, bootstrap=True
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=2, max_f
eatures=log2, max_depth=5, bootstrap=True, score=0.7557964892518539, to
tal=   4.8s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=12, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  2.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    3.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   11.5s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=12, bootstrap=False, score=0.7622377622377622,
total= 2.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=12, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.9min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    3.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   11.7s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=12, bootstrap=False, score=0.759093255737551, t
otal= 2.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=12, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.9min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    3.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   11.7s finished
```

```
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=12, bootstrap=False, score=0.7648190735439058,
total= 2.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=12, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.9min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   11.6s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=12, bootstrap=False, score=0.7659923968648801,
total= 2.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=12, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.9min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    3.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   11.8s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=12, bootstrap=False, score=0.7594574298319722,
total= 2.0min
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   47.7s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.1s finished

[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.7611583047824658, tot
al=  49.9s
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   48.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.3s finished

[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.7601257802600084, tot
al=  50.3s
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   47.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.2s finished

[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.7648190735439058, tot
al=  49.7s
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   47.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.3s finished

[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.7665555920589477, tot
al=  49.5s
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=log2, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   47.8s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.3s finished
```

```
[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=log2, max_depth=8, bootstrap=True, score=0.760912419036891, tota
l=  50.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   58.7s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    6.1s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=False, score=0.7560895480358567, t
otal= 1.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   59.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    6.1s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=False, score=0.7543999624536537, t
otal= 1.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   57.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    6.1s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=False, score=0.7602196461256864, t
otal= 1.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=False
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   58.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    6.1s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=False, score=0.7594217862674238, t
otal= 1.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   59.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    6.1s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=False, score=0.7557964892518539, t
otal= 1.0min
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=True, score=0.7576852677523819, to
tal=   4.8s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished
```

```
[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=True, score=0.7547284929835265, to
tal=   4.9s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=True, score=0.7611583047824658, to
tal=   4.8s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=True, score=0.7598911155958136, to
tal=   4.9s
[CV] n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=5, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    4.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.6s finished

[CV]  n_estimators=100, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=5, bootstrap=True, score=0.7559372946587816, to
tal=   4.8s
[CV] n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   48.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.2s finished

[CV]  n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7616745670436945, to
tal=  50.1s
[CV] n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   47.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.1s finished

[CV]  n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7600319143943305, to
tal=  50.1s
[CV] n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   46.8s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.1s finished

[CV]  n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7642089454169991, to
tal=  48.9s
[CV] n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   46.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.1s finished
```

```
[CV]  n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7664617261932698, to
tal=  49.0s
[CV] n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   47.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    6.1s finished

[CV]  n_estimators=775, min_samples_split=10, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7603022622735379, to
tal=  49.5s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.8s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.760923640118271, tot
al= 1.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   59.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.9s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7602196461256864, to
tal= 1.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   59.5s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.8s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7651006711409396, to
tal= 1.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   59.8s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.8s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7665555920589477, to
tal= 1.0min
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=8, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   59.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.8s finished

[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=8, bootstrap=True, score=0.7607246784943208, to
tal= 1.0min
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=20, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:  2.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    4.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   16.8s finished
```

```
[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=20, bootstrap=False, score=0.7558079504388229, t
otal= 2.0min
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=20, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:  2.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    4.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   16.8s finished

[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=20, bootstrap=False, score=0.75045759609518, tot
al= 2.0min
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=20, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:  2.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    4.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   16.7s finished

[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=20, bootstrap=False, score=0.7576383348195429, t
otal= 2.0min
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=20, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:  2.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    4.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   17.1s finished

[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=20, bootstrap=False, score=0.7584361946778054, t
otal= 2.0min
[CV] n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fea
tures=sqrt, max_depth=20, bootstrap=False
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:  2.0min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:    4.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 775 out of 775 | elapsed:   16.0s finished

[CV]  n_estimators=775, min_samples_split=2, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=20, bootstrap=False, score=0.7509621702806721, t
otal= 2.0min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_fe
atures=log2, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    8.0s finished

[CV]  n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_f
eatures=log2, max_depth=8, bootstrap=False, score=0.7610175059839489, t
otal= 1.5min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_fe
atures=log2, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    8.1s finished

[CV]  n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_f
eatures=log2, max_depth=8, bootstrap=False, score=0.7599849814614915, t
otal= 1.5min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_fe
atures=log2, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.9s finished
```

```
[CV]  n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_f
eatures=log2, max_depth=8, bootstrap=False, score=0.7638804148871263, t
otal= 1.5min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_fe
atures=log2, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    7.9s finished

[CV]  n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_f
eatures=log2, max_depth=8, bootstrap=False, score=0.766180128596236, to
tal= 1.5min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_fe
atures=log2, max_depth=8, bootstrap=False

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.4min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    2.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    8.0s finished

[CV]  n_estimators=1000, min_samples_split=5, min_samples_leaf=2, max_f
eatures=log2, max_depth=8, bootstrap=False, score=0.7605369379517507, t
otal= 1.5min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=16, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.6min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    4.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   17.2s finished

[CV]  n_estimators=1000, min_samples_split=5, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=16, bootstrap=True, score=0.7611113718496268, t
otal= 1.7min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=16, bootstrap=True
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.6min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    4.1s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   17.0s finished

[CV]  n_estimators=1000, min_samples_split=5, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=16, bootstrap=True, score=0.7573567372225091, t
otal= 1.7min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=16, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:  1.6min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:    4.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed:   16.9s finished

[CV]  n_estimators=1000, min_samples_split=5, min_samples_leaf=4, max_f
eatures=sqrt, max_depth=16, bootstrap=True, score=0.7652414699394565, t
otal= 1.7min
[CV] n_estimators=1000, min_samples_split=5, min_samples_leaf=4, max_fe
atures=sqrt, max_depth=16, bootstrap=True

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
```

Função auxiliar: verificar resultados do GridSearch

In [0]:
```
# Utility function to report best scores
def report(results, n_top=3):
    for i in range(1, n_top + 1):
        candidates = np.flatnonzero(results['rank_test_score'] == i)
        for candidate in candidates:
            print("Model with rank: {0}".format(i))
            print("Mean validation score: {0:.3f} (std: {1:.3f})".format
(
                  results['mean_test_score'][candidate],
                  results['std_test_score'][candidate]))
            print("Parameters: {0}".format(results['params'][candidate
]))
            print("")
```

In [0]:
```
report(cv_rf.cv_results_)
```

```
Model with rank: 1
Mean validation score: 0.763 (std: 0.001)
Parameters: {'n_estimators': 550, 'min_samples_split': 2, 'min_samples_
leaf': 4, 'max_features': 'log2', 'max_depth': 8, 'bootstrap': True}

Model with rank: 2
Mean validation score: 0.763 (std: 0.002)
Parameters: {'n_estimators': 775, 'min_samples_split': 2, 'min_samples_
leaf': 4, 'max_features': 'log2', 'max_depth': 8, 'bootstrap': True}

Model with rank: 3
Mean validation score: 0.763 (std: 0.002)
Parameters: {'n_estimators': 100, 'min_samples_split': 5, 'min_samples_
leaf': 4, 'max_features': 'sqrt', 'max_depth': 8, 'bootstrap': True}
```

- Resultados obtidos com execução em outra máquina (GridSearch)

Model with rank: 1 Mean validation score: 0.763 (std: 0.001) Parameters: {'bootstrap': True, 'max_features': 'sqrt', 'n_estimators': 775, 'max_depth': 10, 'min_samples_split': 2, 'min_samples_leaf': 2}

Treinamento do modelo

In [0]:
```
#### definimos o modelo com a configuracao de hiperparametros que aprese
ntou melhor desempenho
fit_rf = cv_rf.best_estimator_
```

In [0]:
```
### treinamos o modelo
fit_rf.fit(X_train, np.ravel(y_train))
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:   42.0s finished
```

Out[0]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gi
ni',
            max_depth=8, max_features='log2', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=4, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=550, n_jobs=Non
e,
            oob_score=False, random_state=42, verbose=1, warm_start=Fal
se)
```

Importância das variáveis

In [0]:
```
## agora que ja treinamos nos valores mais 'apropriados', vamos verifica
r a importancia dos atributos
importances_rf = fit_rf.feature_importances_
```

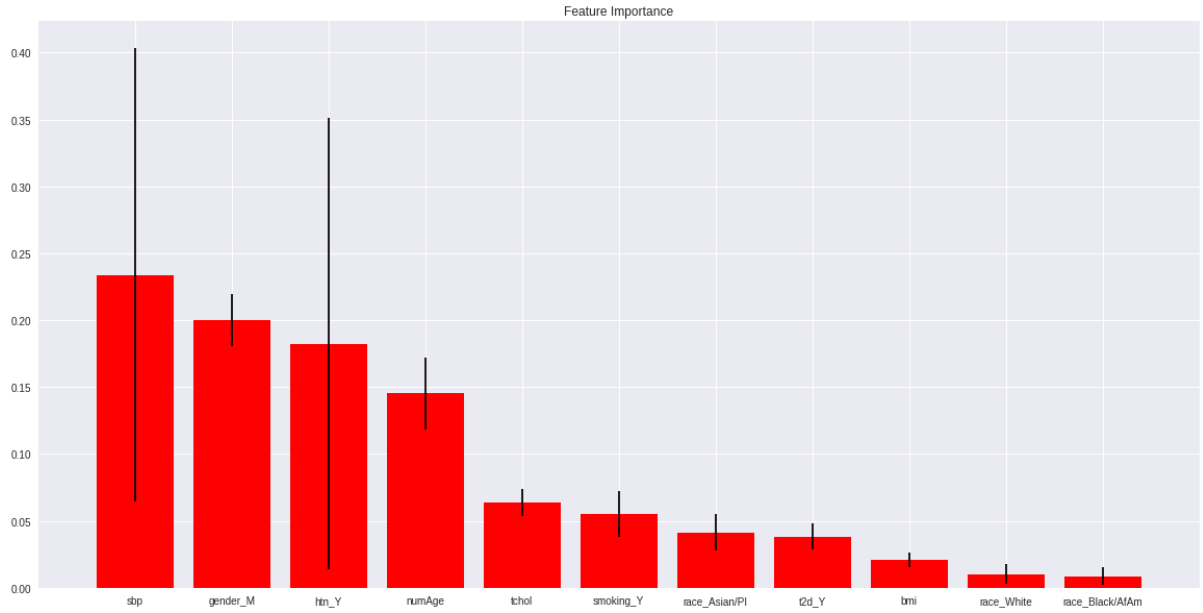
In [0]:
```
from tabulate import tabulate
headers = ["name", "score"]
values = sorted(zip(X_train.columns, fit_rf.feature_importances_), key=l
ambda x: x[1] * -1)
print(tabulate(values, headers, tablefmt="plain"))
```

```
name                 score
sbp               0.233855
gender_M          0.200063
htn_Y             0.18249
numAge            0.14534
tchol             0.0637295
smoking_Y         0.0549396
race_Asian/PI     0.0414388
t2d_Y             0.0383116
bmi               0.0209652
race_White        0.010159
race_Black/AfAm   0.00870817
```

Visualização gráfica da importância das variáveis

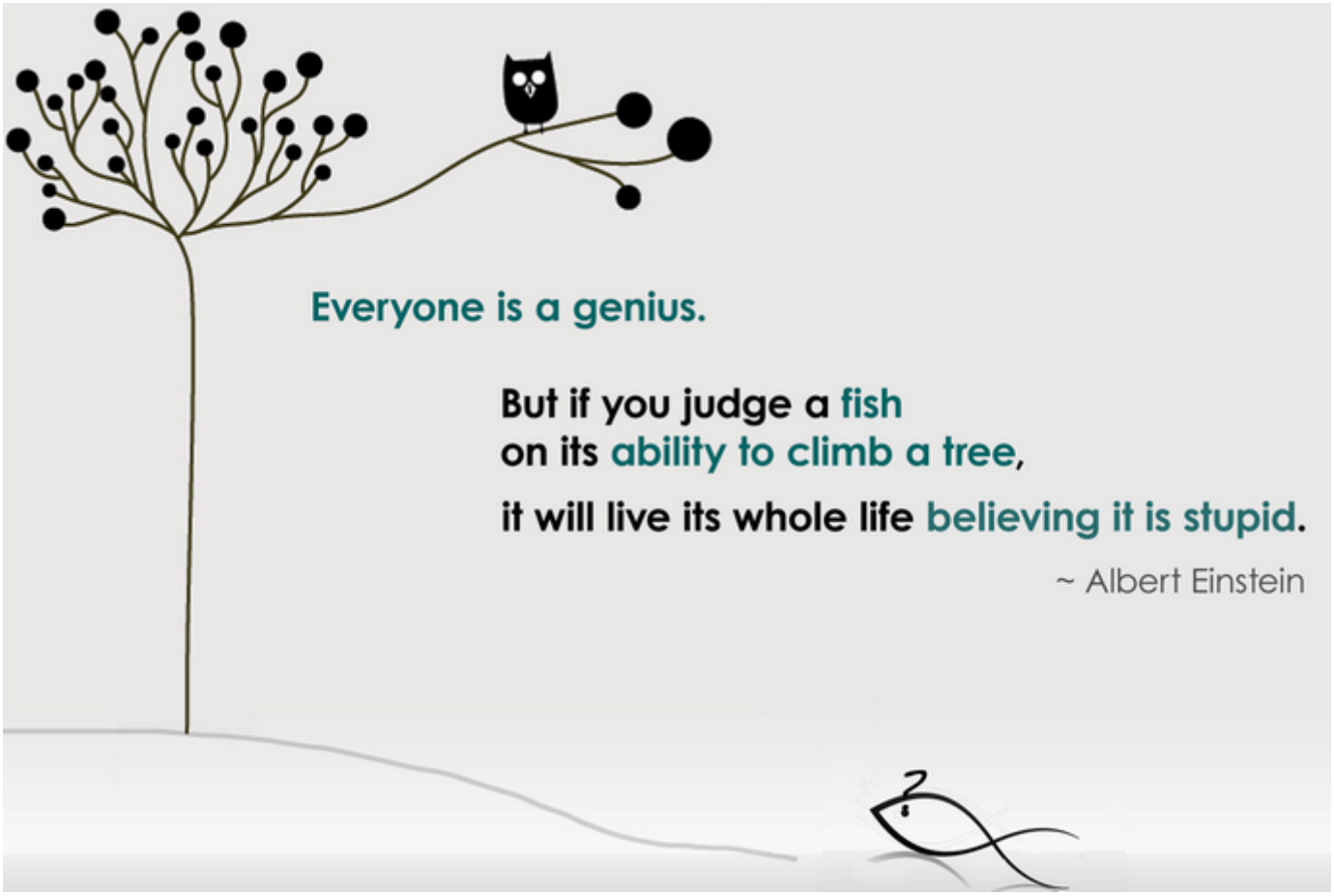
In [0]:
```
skplt.estimators.plot_feature_importances(fit_rf, feature_names=X_train.
columns, figsize=(20,10))

plt.show()
```

Out[0]:



Avaliando o desempenho do algoritmo

Reflexão sobre a importância da escolha de uma boa métrica:



Testando o modelo (conjunto de testes)

In [0]:
```
predictions_rf = fit_rf.predict(X_test)
print(classification_report(y_test, predictions_rf))

### Para entender o conceito de micro e macro avg: http://tinyurl.com/y45cn5pn
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.0s finished
```

	precision	recall	f1-score	support
N	0.78	0.93	0.85	13458
Y	0.65	0.32	0.43	5343
micro avg	0.76	0.76	0.76	18801
macro avg	0.71	0.63	0.64	18801
weighted avg	0.74	0.76	0.73	18801

curva ROC

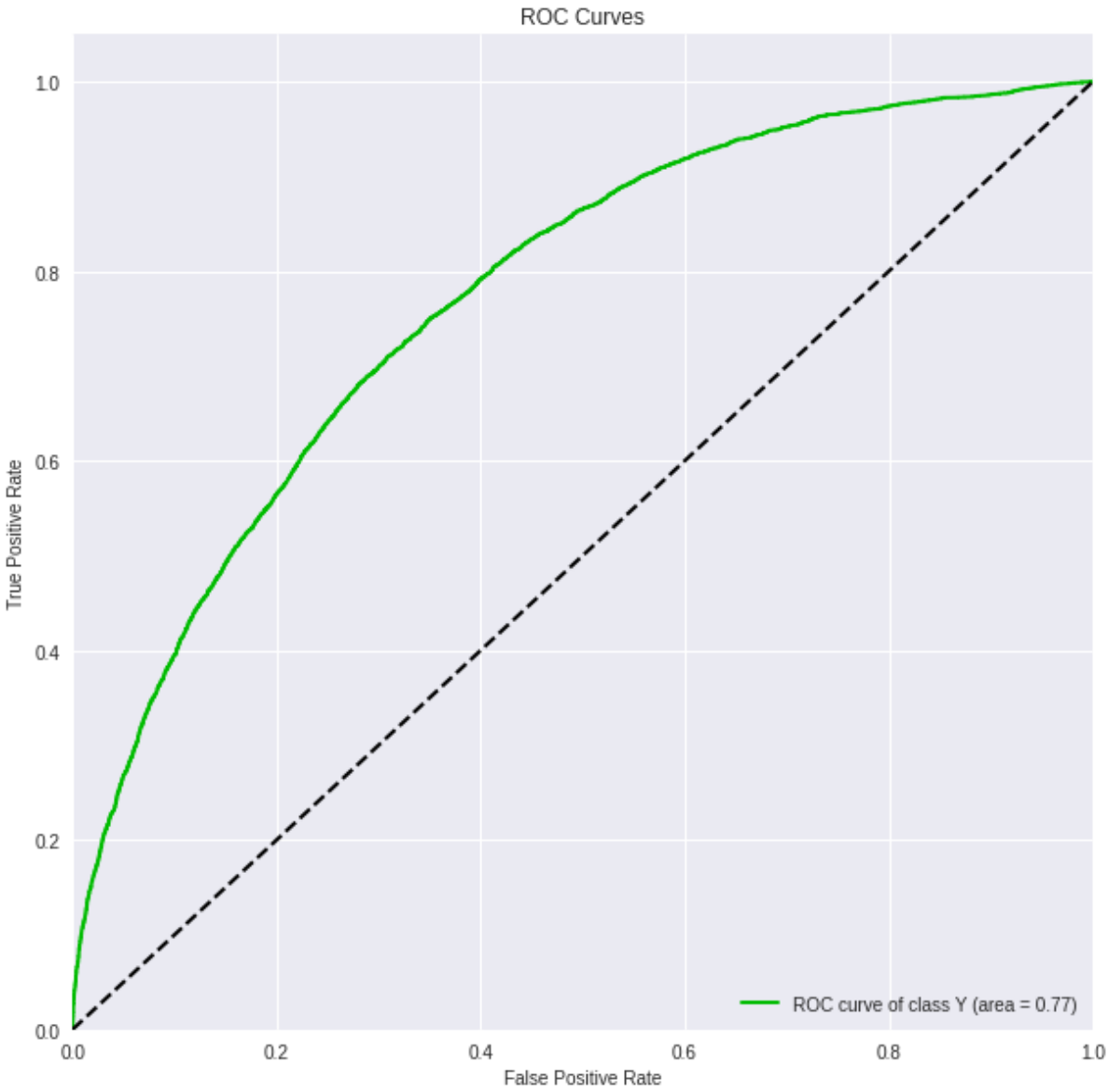
In [0]:
```
probas = fit_rf.predict_proba(X_test)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.0s finished
```

In [0]:
```
skplt.metrics.plot_roc(np.ravel(y_test), probas, plot_micro=False, plot_
macro=False, classes_to_plot='Y', figsize=(10,10))
```

Out[0]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc94b3f2be0>

Out[0]:



Curva de calibração

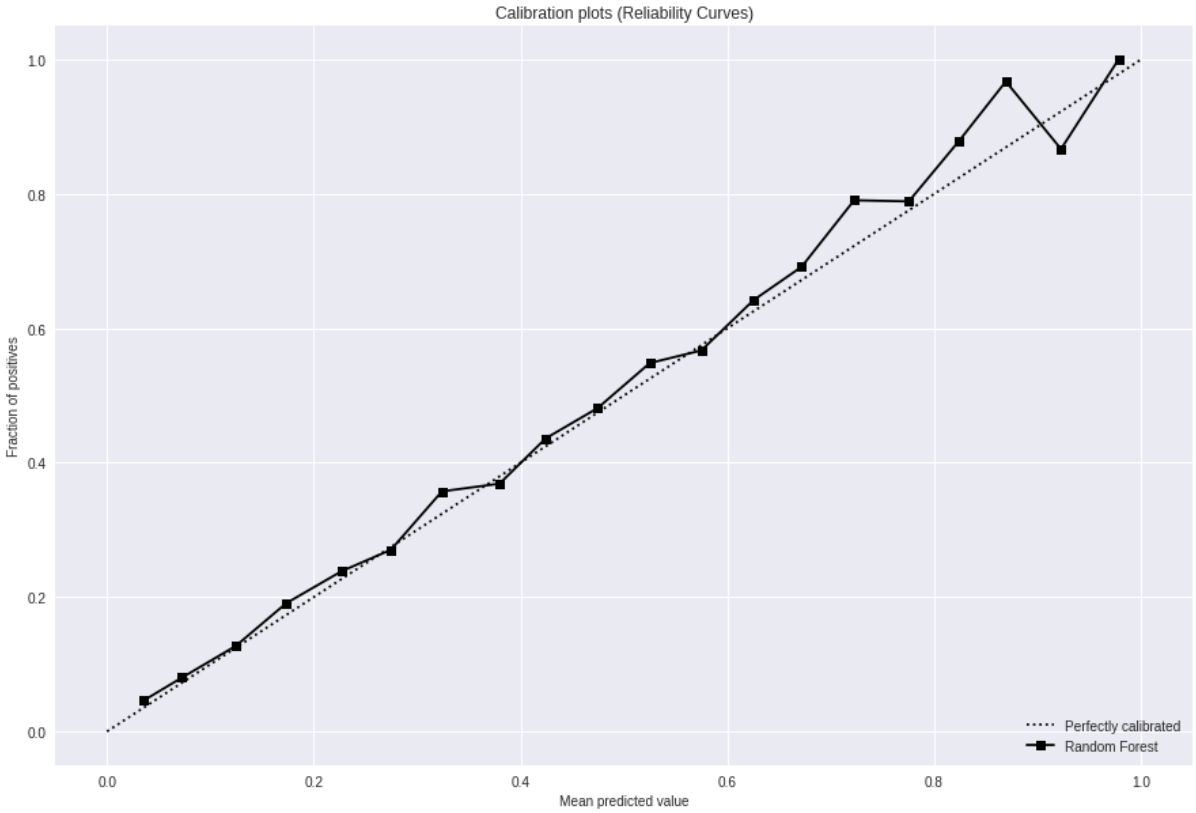
In [0]:
```
rf_probas =  fit_rf.predict_proba(X_test)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.0s finished
```

In [0]:
```
probas_list = [rf_probas]
clf_names = ['Random Forest']
skplt.metrics.plot_calibration_curve(np.ravel(y_test), probas_list,clf_n
ames, figsize=(15,10), n_bins=20)
```

Out[0]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc94de31978>

Out[0]:



Visualizando o conjunto de testes juntamente com a predição

In [0]:
```
y_hats = fit_rf.predict(X_test)
y_test['preds'] = y_hats

df_out = pd.merge(X_test,y_test[['preds']],how = 'left',left_index = True, right_index = True)
df_out.head(20)
```

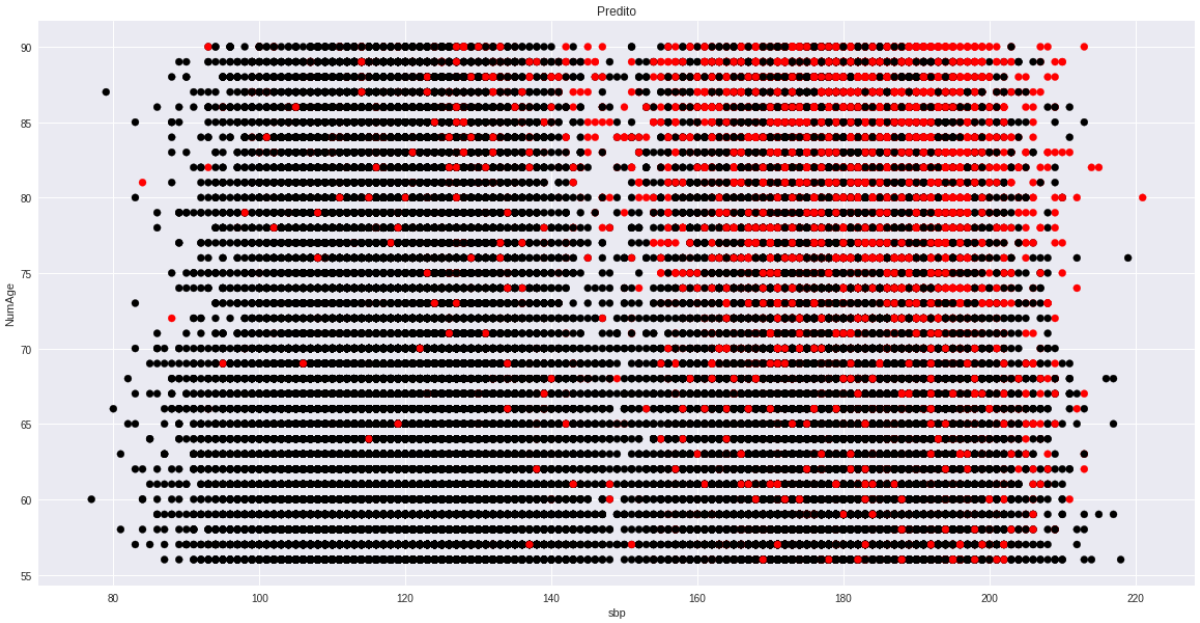
```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 550 out of 550 | elapsed:    1.0s finished
```

Out[0]:

	numAge	bmi	tchol	sbp	race_Asian/PI	race_Black/AfAm	race_White	htn_Y	smoking_
28034	74	18	183	197	0	0	1	1	
100868	62	23	196	112	0	0	1	0	
36958	62	26	178	112	0	0	1	0	
109568	61	16	185	173	0	0	1	1	
74016	70	23	158	121	0	0	1	0	
397613	63	17	156	131	0	0	1	0	
338855	85	21	212	182	0	0	1	1	
186361	68	24	196	127	0	0	1	0	
120512	63	23	157	184	0	0	1	1	
232074	87	23	190	158	1	0	0	1	
152540	69	17	179	128	1	0	0	0	
331032	66	25	245	167	0	0	1	1	
54068	63	19	240	182	0	0	1	1	
197742	58	17	196	178	0	0	1	1	
42306	80	23	159	107	0	0	1	0	
238564	86	22	155	106	1	0	0	0	
292242	84	23	155	177	1	0	0	1	
302920	76	22	157	127	0	0	1	0	
371456	74	36	243	112	0	0	1	0	
259479	83	19	241	187	0	1	0	1	

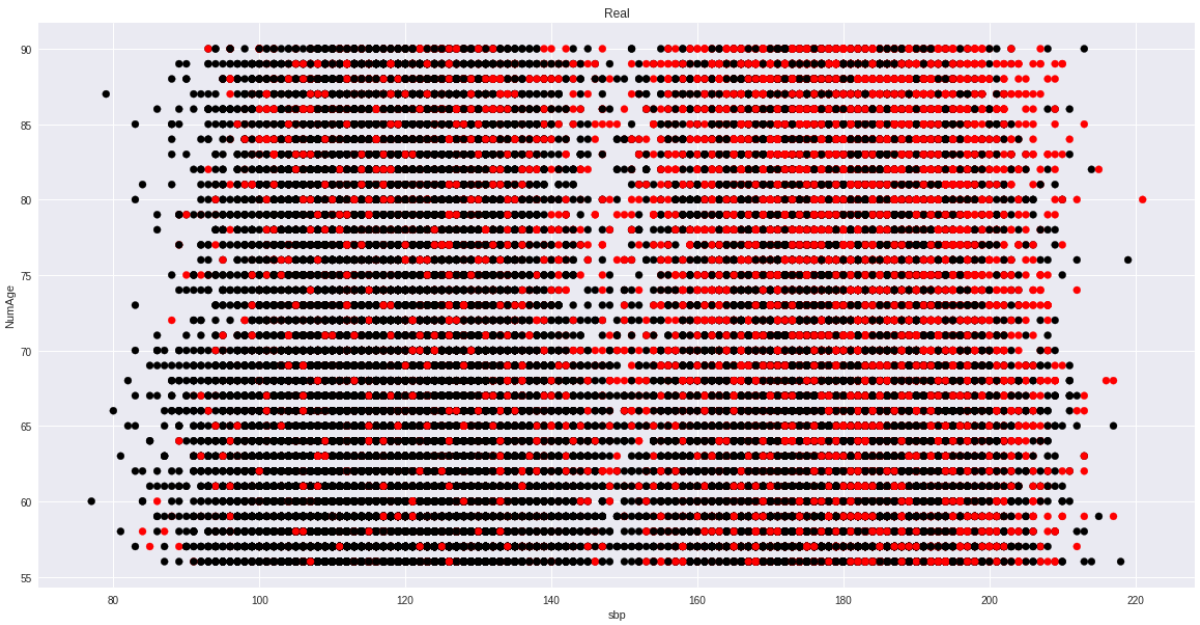
In [0]:
```
plt.figure(figsize=(20,10))
color = ['red' if l == 'Y' else 'black' for l in df_out['preds']]
plt.scatter(df_out['sbp'],df_out['numAge'],c=color)
plt.xlabel('sbp')
plt.ylabel('NumAge')
plt.title('Predito')
plt.show()
```

Out[0]:



In [0]:
```
plt.figure(figsize=(20,10))
color = ['red' if l == 'Y' else 'black' for l in y_test['cvd']]
plt.scatter(df_out['sbp'],df_out['numAge'],c=color)
plt.xlabel('sbp')
plt.ylabel('NumAge')
plt.title('Real')
plt.show()
```

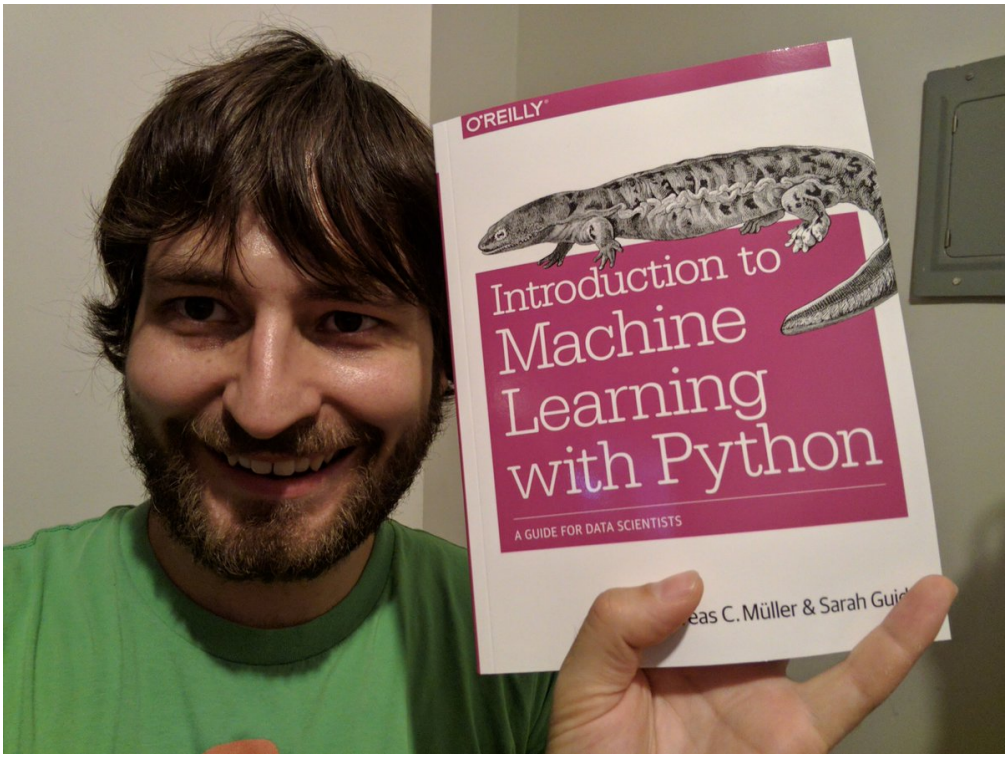
Out[0]:

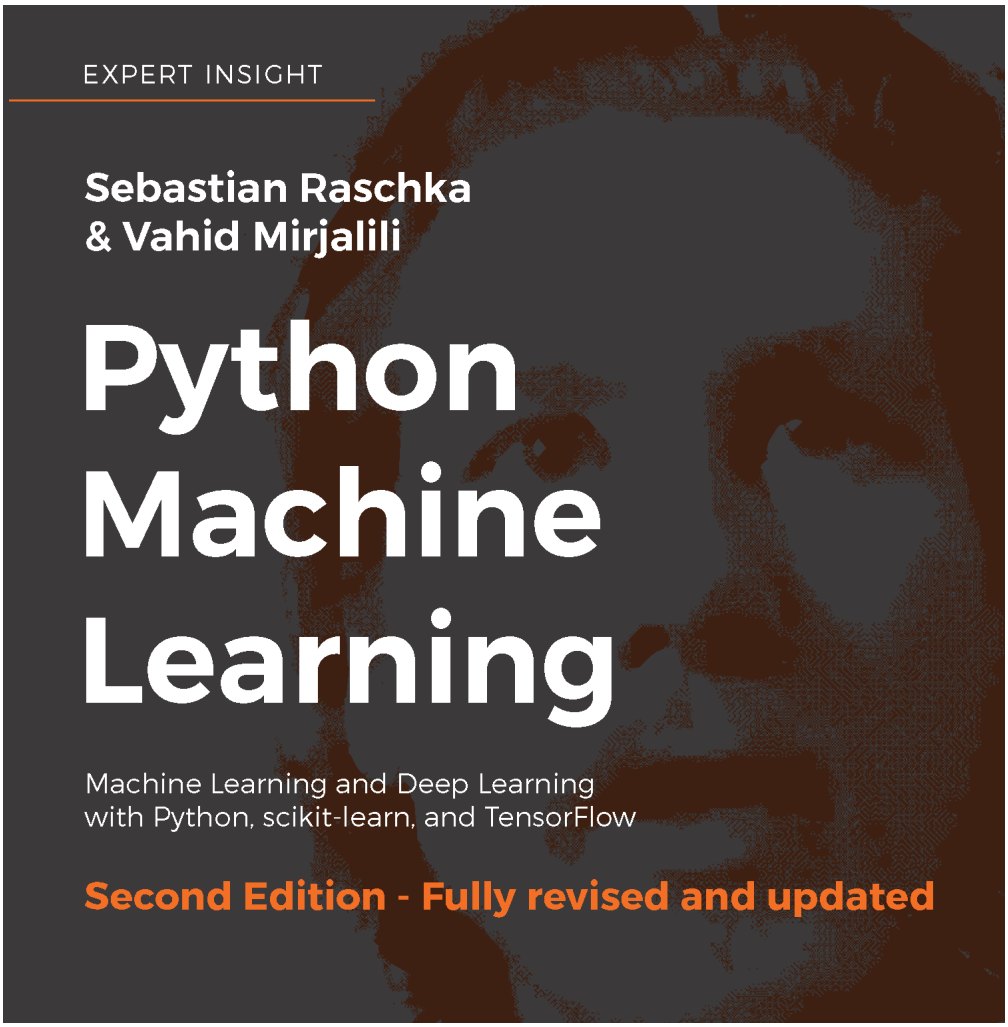


In [0]:

Referências Sugeridas

In [0]:


O'REILLY
Introduction to
Machine
Learning
with Python
A GUIDE FOR DATA SCIENTISTS


EXPERT INSIGHT
Sebastian Raschka
& Vahid Mirjalili
Python
Machine
Learning
Machine Learning and Deep Learning
with Python, scikit-learn, and TensorFlow
Second Edition - Fully revised and updated

In [0]: