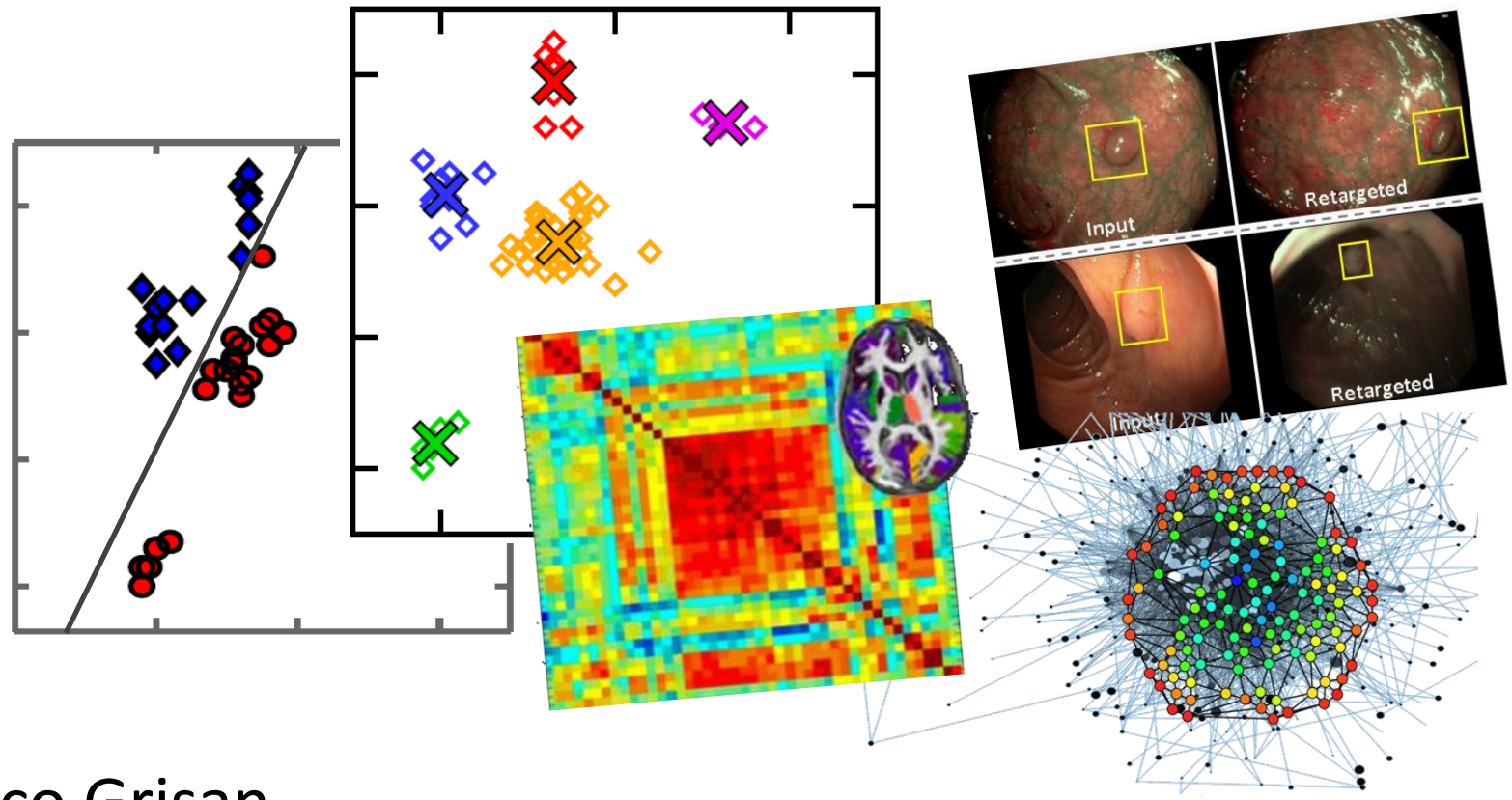
Machine Learning
CSI-7-MAL



Enrico Grisan

enrico.grisan@lsbu.ac.uk

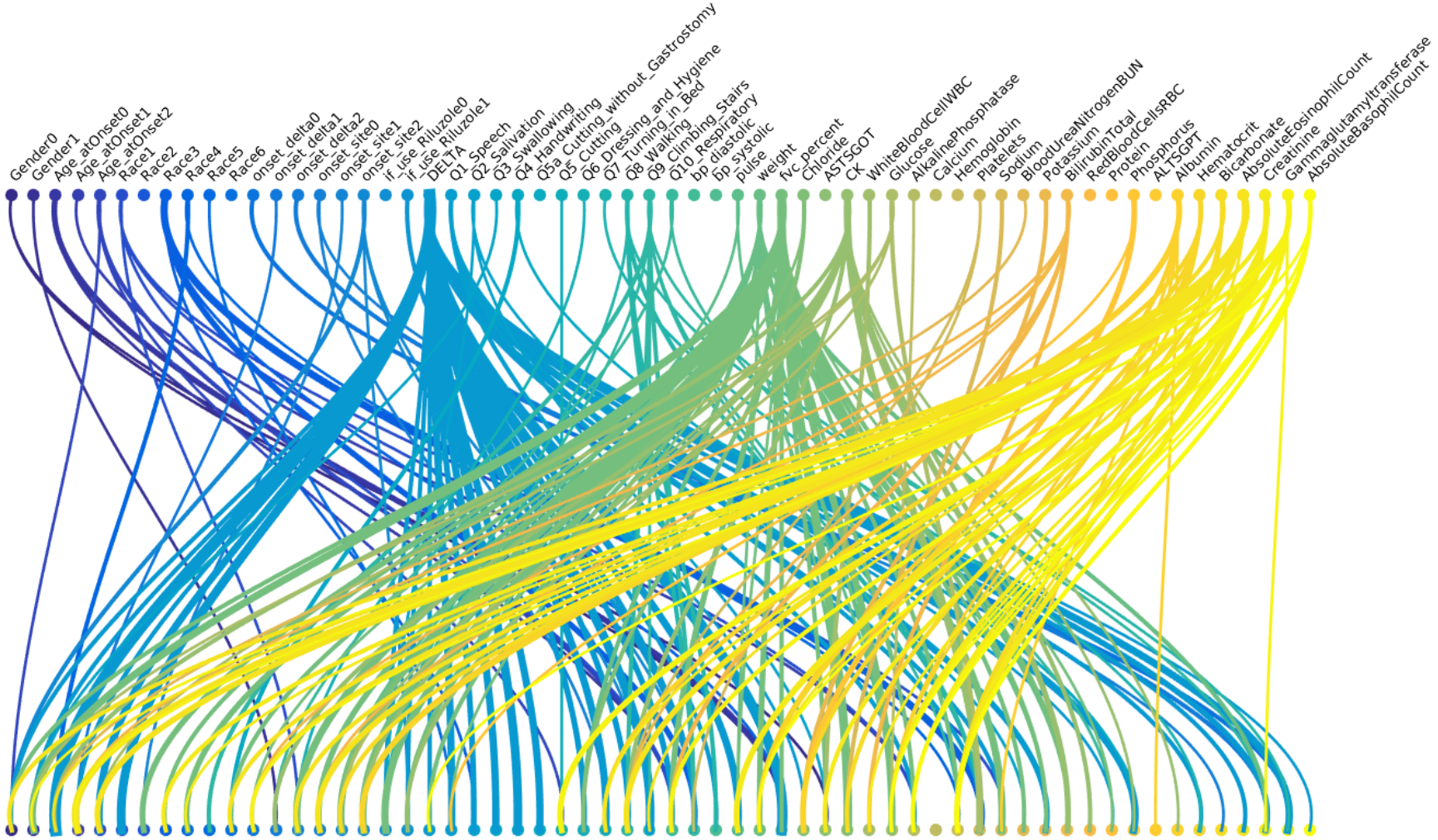
Week 7 tutorial

- Use random forests to predict survival
- Select the features used by the random forest and build a reduced classifier

PRO-ACT dataset

- Pooled Resource Open-Access ALS Clinical Trials (PRO-ACT) Database: https://nctu.partners.org/ProACT
- The full database: 10723 patients, multiple visits
- Pre-processing: removing all records with missing values
- Final data: 1936 patients and 16177 visits
- Time binning: a patient condition is evaluated every trimester for 6 trimesters after baseline (month 3 to month 18)

Features


Gender0
Gender1
Age_atOnset0
Age_atOnset1
Age_atOnset2
Race1
Race2
Race3
Race4
Race5
Race6
onset_delta0
onset_delta1
onset_delta2
onset_site0
onset_site1
onset_site2
if_use_Riluzole0
if_use_Riluzole1
DELTA
Q1_Speech
Q2_Salivation
Q3_Swallowing
Q4_Handwriting
Q5a_Cutting_without_Gastrostomy
Q5_Cutting
Q6_Dressing_and_Hygiene
Q7_Turning_in_Bed
Q8_Walking
Q9_Climbing_Stairs
Q10_Respiratory
bp_diastolic
bp_systolic
pulse
weight
fvc_percent
Chloride
ASTSGOT
CK
WhiteBloodCellWBC
Glucose
AlkalinePhosphatase
Calcium
Hemoglobin
Platelets
Sodium
BloodUreaNitrogenBUN
Potassium
BilirubinTotal
RedBloodCellsRBC
Protein
Phosphorus
ALTSGPT
Albumin
Hematocrit
Bicarbonate
AbsoluteEosinophilCount
Creatinine
GammaglutamyltransferaseAbsoluteBasophilCount

Imports

```
import matplotlib.pyplot as plt
import numpy as np

from scipy import io
from sklearn.datasets import fetch_openml
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

Load data

Predicting the survival situation for the 4th trimester using random forest

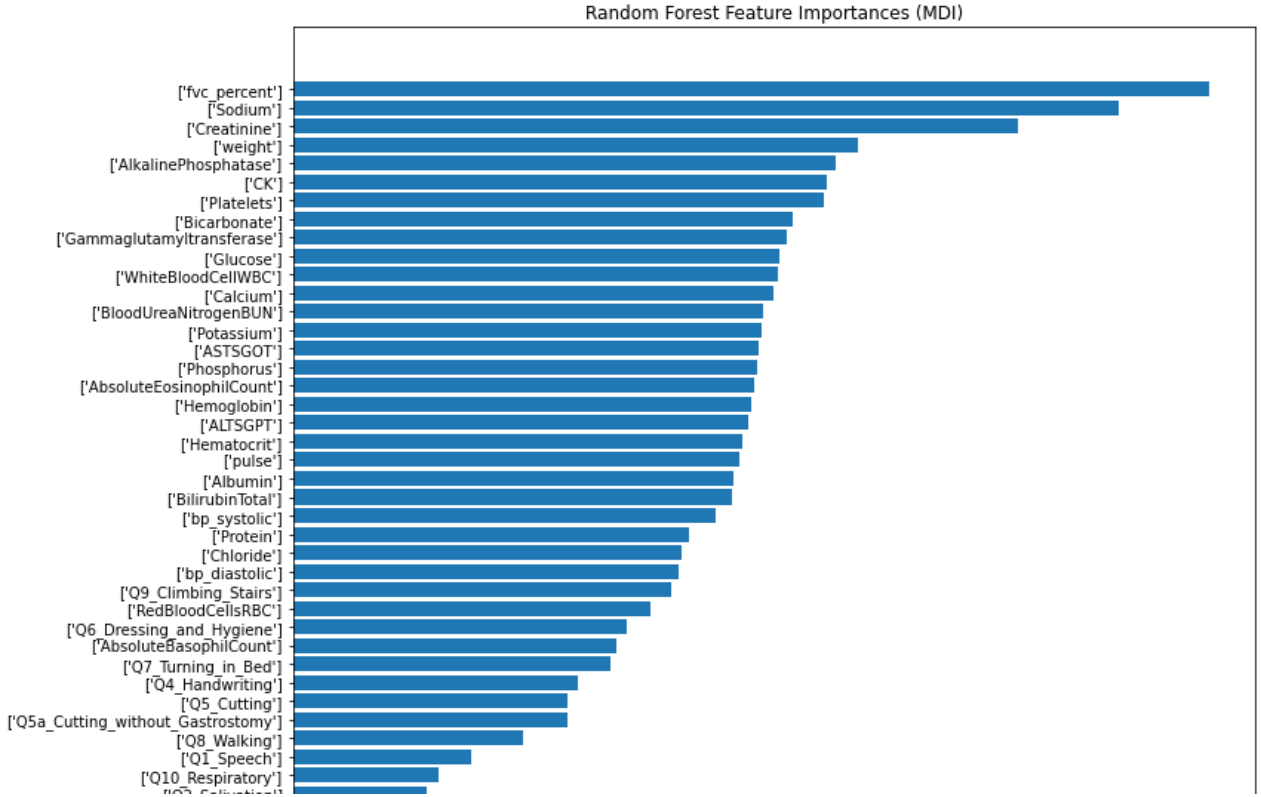
```
#Load data
mat_data = io.loadmat('ALS_data.mat')

data=mat_data['data']
target=mat_data['survived']
Dt=mat_data['Dt']
labels=mat_data['labels'][0]

Xtrain,Xtest,Ytrain,Ytest=train_test_split(data,target[:,3],test_size=0.3)
```

Random forest

```
RF= RandomForestClassifier(random_state=42, oob_score=True, n_estimators=100)

RF.fit(Xtrain[Ytrain>-1,:],Ytrain[Ytrain>-1])

Yest=RF.predict(Xtest)

oob_error = 1 - RF.oob_score_

print(oob_error)
print("RF train accuracy: %0.3f" % RF.score(Xtrain, Ytrain))
print("RF test accuracy: %0.3f" % RF.score(Xtest, Ytest))
```

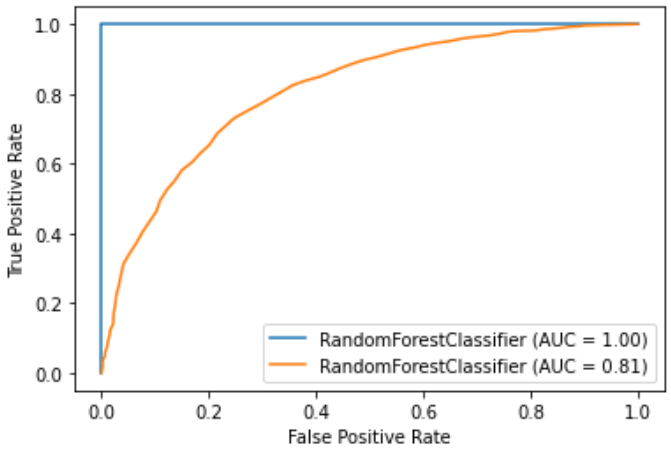
Feature importance

```
tree_feature_importances = RF.feature_importances_
sorted_idx = tree_feature_importances.argsort()

y_ticks = np.arange(0, 59)
fig=plt.figure(figsize=(12,12))
ax = fig.subplots()
ax.barh(y_ticks, tree_feature_importances[sorted_idx])
ax.set_yticks(y_ticks)
ax.set_yticklabels(labels[sorted_idx])
ax.set_title("Random Forest Feature Importances (MDI)")
fig.tight_layout()
plt.show()
```



Showing a ROC curve

```
from sklearn import metrics
fig, ax = plt.subplots()

# Use PLOT_ROC_CURVE
metrics.plot_roc_curve(RF, Xtrain[Ytrain>-1,:], Ytrain[Ytrain>-1], ax=ax)
metrics.plot_roc_curve(RF, Xtest[Ytest>-1,:], Ytest[Ytest>-1], ax=ax)
plt.show()

# COMPUTE ROC AND PLOT
ptrain=RF.predict_proba(Xtrain[Ytrain>-1,:])
fpr, tpr, _ = metrics.roc_curve(Ytrain[Ytrain>-1], ptrain[:,1])
roc_auc = metrics.auc(fpr, tpr)
fig, ax = plt.subplots()
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)

ptest=RF.predict_proba(Xtest[Ytest>-1,:])
fpr, tpr, _ = metrics.roc_curve(Ytest[Ytest>-1], ptest[:,1])
roc_auc = metrics.auc(fpr, tpr)
plt.plot(fpr, tpr, label='ROC curve (area = %0.2f)' % roc_auc)
plt.legend(loc="lower right")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```



Growing the forest

```
n_trees=np.arange(1,500,10)
error_rate=np.zeros(n_trees.shape)
test_rate=np.zeros(n_trees.shape)
for ind in range(n_trees.shape[0]):
    RF= RandomForestClassifier(random_state=42, oob_score=True, n_estimators=n_trees[ind])
    RF.fit(Xtrain[Ytrain>-1,:],Ytrain[Ytrain>-1])
    Yest=RF.predict(Xtest)
    oob_error = 1 - RF.oob_score_
    test_error=1-sum(Yest[Ytest>-1]==Ytest[Ytest>-1])/sum(Ytest>-1)
    error_rate[ind]=oob_error
    test_rate[ind]=test_error

plt.plot(n_trees, error_rate,label='OOB error')
plt.plot(n_trees, test_rate,label='Test error')
plt.xlabel("n_trees")
plt.ylabel("Error rate")
plt.legend(loc="upper right")
plt.show()
```

Ex 1

Build a random forest classifier for each of the target trimester window for prediction.

Do the performance change significantly when the prediction is further in the future?

Ex 2

Use AdaBoost to build a classifier for each of the target trimester window for prediction.

- Look into the scikit-learn documentation for AdaBoost

- You will need to import:

```
from sklearn.ensemble import AdaBoostClassifier
```

Homework 1

Using random forest, look into the scikit-learn documentation and try to regularize the trees that are grown by changing the default parameters for the maximum depth of the trees and the minimum samples per final node (leaf).

Does the test error becomes more similar to the training error?

Homework 2

How much the random forest performance change if you train one on the best 25 features, based on the ranking from the initial random forest trained on the full set of features?

Homework 3

Look into the documentation for AdaBoost:

- how can you extract the features selected in each weak classifier?

-can you think of a way to evaluate the feature importance for AdaBoost similar to what is done for Random Forests?

-try coding the feature importance evaluation for AdaBoost