IST-2004-027173

Eu*Resist*

Integration of viral genomics with clinical data
to predict response to anti-HIV treatment

Instrument: STREP
Thematic Priority 2: “Information Society Technologies”

D4.1 - EuResist Prediction System Description

Due date of deliverable: 31 December 2007
Actual submission date: 12 January 2008

Start date of project: 1 Jan ‘06 Duration: 30 months

Lead contractor for this deliverable: UniKoeln
Author: Andre Altmann
Main contributions: Michal Rosen-Zvi, Mattia Prosperi
Revision: Final

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Content

List of Figures

List of Tables

Executive summary

This document briefly describes the research work done by the three groups (IBM, MPI, and RM3/Informa) to build a prediction engine to infer virologic response to combination antiretroviral therapy. The engines were all built and tested on the same data. The results of a detailed analysis of the single engine's performance carried out by IBM and MPI is given. Furthermore, approaches undertaken by MPI in combining the individual efforts are described and the gain in predictive performance by combining the single predictors is demonstrated on the same splits of the data. The comparison showed that taking the *mean* of the single engine's prediction is one of the best performing approaches. In addition, this approach does not require training, especially not retraining when one or more of the engines are updated, thus that was the chosen approach for the final EuResist prediction system

1. Introduction

Within the EuResist project four teams were independently developing models to predict response to combination antiretroviral therapy on the basis of the viral genotype and other clinical features. All engines follow a different design paradigm. One objective of the EuResist project is to combine the single efforts of the teams to single EuResist prediction system that should outperform all single approaches, and if not outperform, at least result in a more robust system than the single engines. Intuitively, if one consults two or more experts for one problem, it is more likely that a consensus (however it may be achieved) is more reliable than a single opinion. Chapter 2 will give a brief description of the single engine's layout and the features they use. Chapter 3 will describe the data that was available for this work. Chapter 4 will give a brief analysis of the single engine's results, followed by an introduction of methods we investigated for engine combination in chapter 5. Finally, chapter 6 will provide a summary of the results, and chapter 7 will describe the layout of the final EuResist system.

2. Data used for engine combination

As described in deliverable D3.2 the engine development teams agreed on a training and test set to be able to provided performance measures for the same data, and thus enable a fair comparison. For preparation of this deliverable we agreed to follow the same procedure, and in addition use the same cross-validation split for computation of predictions on the training data. This procedure allows us to optimally use the available amount of data. The training data is here again used (using the same cross-validation splits) to train and compare different methods for engine combination. In addition the different models are compared on the 10% held-out data that were neither used for the individual engines nor for the engine combination. For the classification task virological response was dichotomized to success and failure. A treatment is considered is successful if 2 months after onset of the treatment the viral load is reduced below the limit of detection (here 500 cp per ml) or a 100-fold reduction compared to a baseline measure obtained not longer than 3 months before the treatment. If none of the aforementioned events occurs, the treatment is considered to be a failure. Note, that for a therapy to be included in the data set a genotype obtained not longer than 3 months before the treatment has to be available.

Table 1: Overview about the database and data sets

	General information				Training data			Test data		
	Patients	Sequences	Viral load measures	Therapies	Total	Pos	Neg	Total	Pos	Neg
EuResist	18,467	22,006	240,795	64,864	2,722	1,822	900	301	202	99

3. Brief description of single engines

This section gives a brief overview of the individual engines developed by the teams. A table summarizing the used features for solving the classification and regression task can be found at the end of the section.

3.1. IBM

The prediction system developed by IBM follows a so-called generative discriminative approach. This approach uses a generative model (e.g. Naïve Bayes, or Bayesian Networks) to provide additional features for a standard discriminative approach. The current version uses a Bayesian Network based only on the putative regimen (minimal feature set) or on the putative regimen and information from previous therapies (maximal feature set). The probability of therapeutic success computed by the network is together with additional features input to a Logistic Regression. These Bayesian Networks are not trained on the training set but on a much larger fraction of the EuResist Integrated database. Since the Bayesian Networks do not apply any features related to the viral genotype, 17,000 therapies from the database could be used for training. The discriminative part of the approach is trained on the 2,722 instances of the training set like the other engines. When working with the minimal feature set the standard input consists of indicator variables for the current therapy (20 binary variables), indicator variables for mutations in the protease (20 binary variables), and indicator variables for mutations in the reverse transcriptase (25 binary variables). If the full feature set is available the standard input is augmented by 5 binary variables that indicate mutations observed in previous sequences derived from the patient's blood, the number of past treatment lines, and the logarithm of the baseline viral load.

3.2. MPI

The MPI engine focuses on the development / use of evolutionary features, since one major obstacle in HIV-1 treatment is the virus's escape from drug pressure by developing so-called resistance mutations. In order to accurately predict the outcome of an antiretroviral therapy information of the viral evolution has to be presented to the underlying model. Therefore, three different representations of the viral evolution were developed and investigated for use in prediction of treatment response. One is based on an exhaustive search of the mutational neighbourhood for possible escape mutants. The other two representations are based on mutagenetic [1][2] trees. Briefly, the mutagenetic tree is reconstructed from all pair wise joint probabilities of the defined events, where every event represents the occurrence of a mutation in the viral sequence. Our tests revealed that the Genetic Barrier among other proposed evolutionary features performs best in the setting [3]. Thus, the genetic barrier to drug resistance is computed for every single drug in the putative regimen given the viral sequences and is together with other features (putative treatment, viral genotype, treatment history, second order interactions, …) input for various statistical learning methods (for details see deliverable 3.2) where Logistic regression proved to be the most useful with respect to performance complexity relationship. Feature selection was done using support vector machines (SVMs): all features were used in a linear SVM to solve the classification task; first, the cost parameter is optimized; second, 25 different linear SVMs were generated (5 repetitions of 5-fold cross validation); finally, all features having a mean z-score larger than 2 were kept for use in further models.

3.3. RM3/Informa

The Roma TRE engine explored either local and global approximation, namely Instance Based Reasoning IBR, Logistic Regression LR, Decision Trees DT, Random Forests RF (for classification of virological success) and Multiple Linear Regression MLR (for regression of actual VL change). Feature derivation was investigated either with the definition of Fuzzy Rules for in-vivo resistance, or with the usage of second- and third-order variable interactions for global methods, or with the definition of new distance functions for IBR (rather than the Euclidean one). The Fuzzy rules were constructed translating existing medical knowledge published in the international guidelines using membership functions and logical aggregation operators. Different functions were explored, namely optimistic and pessimistic scores, taking into account also phenotypic resistance and simulation of viral replication through time using differential equations. Second- and third-order variable interactions were set up, taking into account the following mixed effects: (i) drug x mutations, (ii) mutations x mutations, (iii) drug x drug, (iv) drug x drug x drug. An additional set of variables was also included: information of last treatments, using either a continuous exponentially decreasing function or a binary indicator for each drug class; epidemiological information (such as risk factor, country of infection, age, sex…); clinical markers (viral load and CD4). Virtual phenotypes for in-vitro resistance assessment were derived through MLR. All the machine learners were fed with the clinical, epidemiological and all genomic (more than 500 mutations), with the derived features or the mixed effects added. The modelling required a strong work on feature selection, since the input attribute space ranged from hundreds to thousands variables. The techniques used were (i) filters: univariable analysis, Correlation-based Feature Selection (CFS) for all methods; (ii) embedded methods: AIC selection and Ridge Shrinkage for LR; (iii) wrapper methods: bubble selection for IBR. Either for classification or regression, 10-fold cross validation was executed multiple times, in order to retain Gaussian distribution that could be compared with t-statistic (adjusted for sample overlap and multiple testing), useful in model comparison or model selection. The model was tested using different feature spaces and different loss functions (accuracy, Area Under the ROC Curve). After comparisons, models trained with derived Fuzzy features or mixed effect resulted to improve significantly performance over comparisons with several naïve models. The final model chosen was a LR with mixed effects, the best in terms of performances and interpretability, a model that does not require complex feature derivation and allows for a restricted set of features.

Table 2: Summary of features used in the minimal and maxima feature set by each engine.

Task	Engine	Minimal Feature Set	Maximal Feature Set
classification	MPI	genetic barrier for each drug (17); indicator for each drug (17); indicators for IAS mutations (49); 2nd order variables for: drug-drug (9), mutation-mutation (12), drug-mutation (18)	Genetic barrier for each drug (17); indicator for each drug (17); indicators for IAS mutations (49); logarithm (log) of baseline VL; indicator for previous use of drug (1); 2nd order variables for: drug-drug (6), mutation-mutation (7), drug-mutation indicators (xx), drug-"previous use of drug" (19)
	IBM	Indicator for each drug (20), indicators for specific mutations (45), prediction of a Bayesian Network using only the new treatment	Indicator for each drug (20); indicators for specific mutations (45); indicator for previously observed mutations (5); #past treatment lines; log of baseline VL; prediction of a Bayesian Network using previously used drugs and the new treatment
	RM3/ Informa	Like maximal feature set, missing values are replaced by mean or mode of the feature in the training data	Indicators for single mutations (22); indicators for single drugs (4); 2nd order variables for: drug-drug (3), mutation-mutation (5), drug-mutation (9); #past treatments; #drugs in new regimen; log of baseline VL; NRTI experienced indicator; indicator for "vertical transmission"; indicator for country of origin equals Italy
regression	MPI	genetic barrier for each drug (17); indicator for each drug (17); indicators for IAS mutations (49); 2nd order variables for: drug-drug (15), mutation-mutation (9), drug-mutation (8)	Genetic barrier for each drug (17); indicator for each drug (17); indicators for IAS mutations (49); log of baseline VL; indicator for previous use of drug (4); indicators for previous use of NRTIs and NNRTIs; 2nd order variables for: drug-drug (9), mutation-mutation (3), drug-mutation indicators (5), drug-"previous use of drug" (13)
	IBM	Indicator for each drug (20); indicators for specific mutations (45); prediction of a Bayesian Network using only the new treatment	Indicator for each drug (20); indicators for specific mutations (45); indicator for previously observed mutations (2); #past treatment lines, log of baseline VL, prediction of a Bayesian Network using previously used drugs and the new treatment
	RM3/ Informa	Like maximal feature set, missing values are replaced by mean or mode of the feature in the training data	Indicators for single mutations (32); indicators for single drugs (7); 2nd order variables for: drug-drug (11), mutation-mutation (23), drug-mutation (14); 3rd order variables for: drug-drug-drug (5); indicators for exposure to the three drug classes; #drugs in new regimen; #past treatment lines; log of baseline VL; local similarity to consensusB; indicator for "vertical transmission"; indicator for subtype equals CRF11_cpx

4. Single engine comparison

4.1. Performance of single engines

The performance of the single engines on the training data (10-fold cross validation) and on the test data are given in Table 2. Performance was measured by the Area under the ROC curve (AUC) and Accuracy. For every engine performance achieved with the minimal feature set consisting only of the putative treatment, the viral genotype and features derivable from these two, and the maximal feature set consisting of all available information is given. IBM and MPI have prediction engines trained with different sets of features, whereas RM3 trained a full engine and if features are not available they are simply replaced by the mean (or mode) of the feature in the training set.

Table 3: Performance of the single engines on the training set (10-fold cross vaildation) and on the test set using different feature sets. Performance is measure as the Area Under the ROC curve (AUC) and Accruracy.

		AUC		Accuracy = (1 – Error rate)	
		Train	Test	Train	Test
Minimal Feature Set	IBM	0.747 (0.027)	0.744	0.745 (0.024)	0.724
	MPI	0.766 (0.030)	0.768	0.754 (0.031)	0.748
	RM3/Informa	0.758 (0.019)	0.745	0.748 (0.031)	0.757
Maximal Feature Set	IBM	0.768 (0.025)	0.760	0.752 (0.028)	0.757
	MPI	0.789 (0.023)	0.804	0.780 (0.032)	0.751
	RM3/Informa	0.762 (0.021)	0.742	0.754 (0.030)	0.757

Table 4: Correlation (*r*) between actual and predicted drop in log(viral load) using the minimal and maximal feature set.

		Correlation (*r*) between actual and predicted change in log(VL)	
		Train	Test
Minimal Feature Set	IBM	*0.401 (0.033)**	*0.458**
	MPI	0.471 (0.020)	0.527
	RM3/Informa	*0.399 (0.038)***	*0.459***
Maximal Feature Set	IBM	0.658 (0.023)	0.657
	MPI	0.679 (0.020)	0.678
	RM3/Informa	0.664 (0.023)	0.642

* These values were calculated from the predicted follow-up viral load, therefore the performance might be much lower than it really is.

** The full model was used. Unavailable features were replaced by the mean (or mode) of the values in the training data.

4.2. Correlation analysis

As can be seen from last section the performance of the single engines is quite similar. On the one hand this is important for the combination, because predictors with poor performance might have a negative influence on the combination effort; on the other hand a certain amount of diversity is required to be able to exploit the presence of multiple engines.

Figure 1: Scatterplot (upper part of the matrix) and corrleation (lower part of the matrix) between predicted probabilities of success given by three different prediction engines and their mean on the training set.

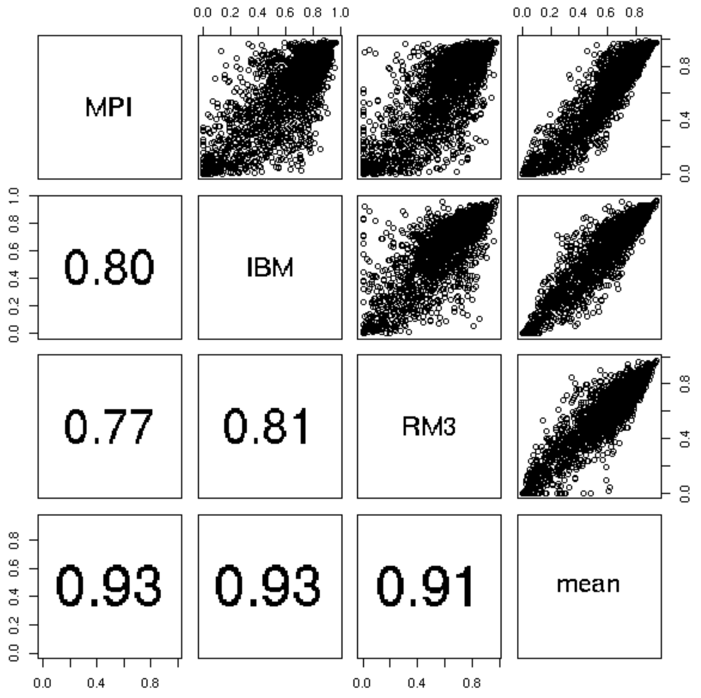


Figure 1 shows the correlation of the different engines on the training set. The three engines are highly correlated which can be expected since during training the same splits of the training data for were used and a logistic regression as statistical learning method was selected by all teams. However, the engines are not completely correlated and therefore might lead to a benefit in the combined system.

Table 5: Occurence of different decision profiles in the training set. The profile is given as "crisp" class labels and the first, second, and third digit correspond to MPI, IBM, and RM3, respectively.

Profile	000	001	010	011	100	101	110	111
Count	358	70	87	136	58	82	100	1831

Table 3 shows the decision profile and its occurrence on the training data. The two most occurring decision profiles represent the total agreement among the engines (111, and 000, respectively). The third and fourth most decision profiles represent an agreement between MPI and IBM (110) and IBM and RM3 (011) on the positive class, respectively.

There is disagreement in almost 20% of the instances in the training set. The agreement among the engines with respect to class labels is subject of the following section.

4.3. Agreement among engines

When studying the pie diagrams for the successful (figure 2) and failing (figure 3) therapies it is clear that the engines differ much more in the prediction of failing therapies. The pie diagramm for the successful therapies (figure 2) reveals that also simple strategies like a majority vote are likely to improve the overall predictive power. Since the fraction of cases predicted incorrectly is smaller than the fraction where only one engine is correct, thus the majority will correctly lable more successful cases.

Figure 2: Pie chart showing the agreement of the engines for all successful therapies in the training and test set.

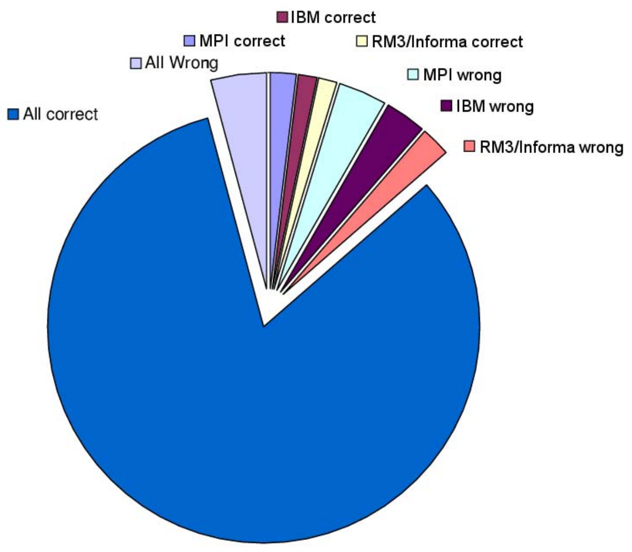
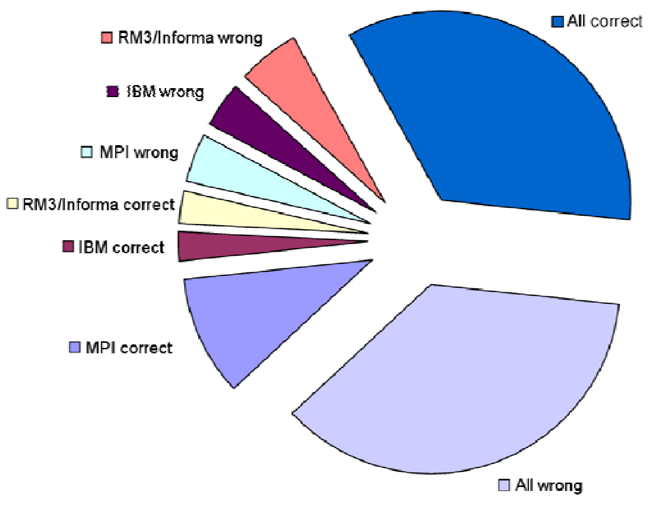


Figure 3: Pie chart showing the agreement of the engines for all failing therapies in the training and test set.



Moreover in the current data set a large fraction (350 instances) of therapies are predicted to be successful by all three systems but are labelled as failure. A closer look at these cases reveals that a large amount (=145) of these failing therapies labelled as success by all engine indeed achieves a viral load below 500 cp per ml during the whole course of the therapy. For comparison, among the remaining cases (=550) this occurs less frequently (100 instances). This difference is according to a Fisher's exact test highly significant (p=4.773x10^{-14}). On the test set the same trend can be observed. However, due to reduced sample size the p-value is smaller (p=0.0111).

4.4. Performance differences with respect to input features

Naturally an engine's performance varies with respect to input features. It is interesting to see how the performance of the multiple engines varies since they mainly differ in the used set of features. Figure 4 shows the variation between the engines with respect to a specific drug in the regimen. In most of the cases the ranking of the prediction engines is the same. However, for drug ddC the system from RM3 outperforms the remaining two engines.

Figure 4: AUC on the training set for all treatments containing a specific drug. Engines from left to right are MPI, IBM, and RM3.

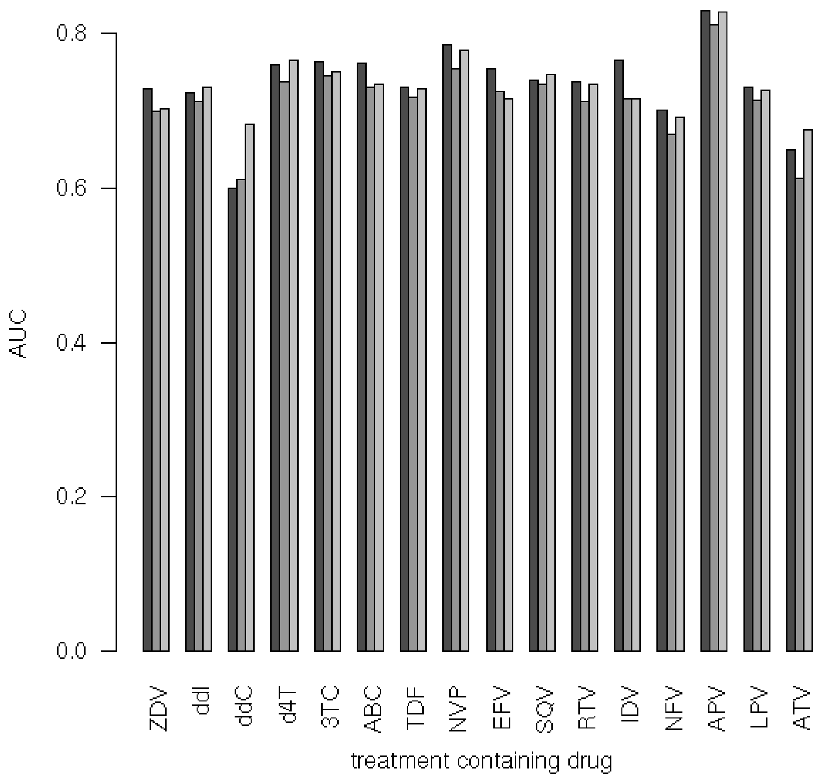
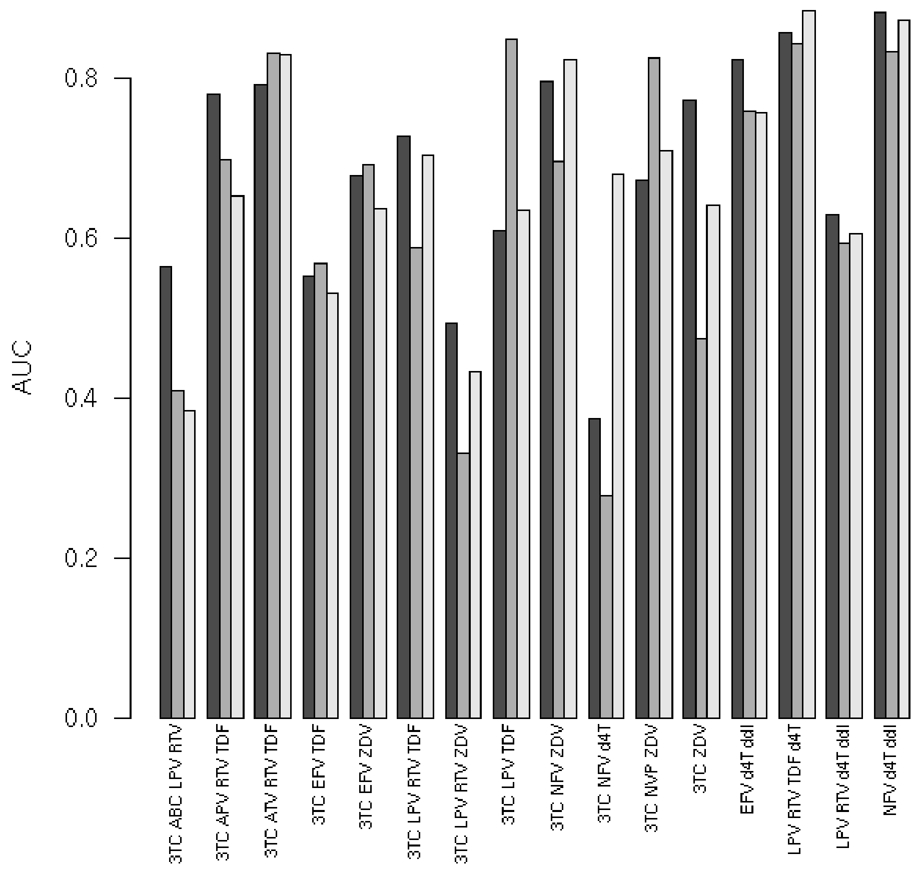


Figure 5 shows the difference in performance with respect to a fixed drug combination rather than a single drug. Every treatment listed here occurs at least with 10 failing and 10 succeeding cases in the training data. Here the performance varies a lot between the engines, and the ranking of the engines changes from one drug combination to the next. For example, the combination "3TC NFV d4T" is much better predicted by RM3, whereas the combination "3TC LPV TDF" is much better predicted by IBM. A combination of the engines could be targeted to exploit these effects.

Figure 5: AUC on the training set for specific treatments with at least 10 failures and 10 successes. Engines from left to right are MPI, IBM, and RM3.



5. Methods for engine combination

In principle there are two main directions in approaching the combination of multiple classifier systems, namely fusion and selection [4]. In classifier fusion every individual system has complete knowledge about the complete feature space and all outputs from the system have to be combined, whereas in classifier selection every system is an expert in a specific domain of the feature space and the local expert alone decides for the output of the ensemble. However, the individual systems of the EuResist engines where designed to be global experts thus classifier fusion is the direction we follow to combine the individual approaches.

Methods for classifier fusion can operate on the "crisp" class labels or continuous values (e.g. support) provided by every classifier. The methods range from simple methods, so-called non-trainable combiners like the *majority* vote, to very sophisticated methods that require an additional training step. Within the limited time of the project we did not aim at developing a completely novel approach but use approaches successfully published in literature to classifier combination. Therefore, we compare several approaches ranging from simple methods to more sophisticated ones. Moreover, we study two different alternatives to combining the individual efforts of the prediction engine teams. The first way is to combine the engines output by classifier fusion; the second is to combine the engines at the feature level. All results will be compared to a combination that has access to an *oracle* telling which engine knows the truth (if known by any of the engines). Intuitively, this oracle represents the upper bound that can be achieved by engine combination.

5.1. Non-trainable combiners

As mentioned above there is a number of simple methods to combine outputs from multiple classifiers. The most intuitive way in doing this is a simple *majority vote*, here every single engine computes a class label (in this case *success* or *failure*) and the label that receives the most votes is taken as the output of the ensemble. This approach is also implemented by so-called ensemble methods in classification. However, with an even number of single engines ties can occur, we decided to predict a *failure* in this case, usually ties are resolved in favour of the majority class. However, here we consider the safety option to reject a putative treatment. As stated above, the single engines output is not only restricted to the predicted class label, but also a probability of observing a success is given. By combining the engines on this level, a handling of ties is obsolete. This continuous measure can be exploited by further simple functions: *mean* will give the mean probability of success by the four engines [5]; *min* will give the minimal probability of success (a pessimistic measure); *max* results in the maximal predicted probability of success (an optimistic measure); *prod* computes the product of the posterior probabilities [6].

5.2. Trainable Combiners

5.2.1. Meta classifiers

The use of so-called meta classifiers is a more sophisticated way of engine combination. Where all the simple methods described above did not honour a classifier's reliability because no training procedure is needed to apply them. As the name meta classifier suggests, in this approach the single engines output are the input for a second classification step. This allows for e.g. weighting the single classifier's outputs. For example in [7] Decision Trees were used combine outputs from several tools that predict 2nd structure of proteins to achieve a more robust result. In this work we apply Quadratic Discriminant Analysis, Logistic Regression, Decision Trees, and Naïve Bayes (operating on the "crisp" class labels) as meta classifiers. Since all these methods represent different paradigms in statistical learning. Moreover, all methods require a training step we use the same split of the data that was available for training the engines.

5.2.2. Weighted means

Instead of combining the classifiers by using the simple average (mean) it is also possible to weight the influence of every classifier with respect to its error. Assume that we have *J* classes and *I* classifiers, the output is then combined by following rule:

$$\mu_i(\vec{x}) = \sum_j w_i d_{i,j}(\vec{x})$$

Where $\vec{x}$ is the instance to be classified, and μ_i is the support for each class, and $d_{i,j}$ is the support by classifier *i* for class *j*. w_i are the weights. The class that has the highest combined support is the output of the ensemble. If all w_i are set to 1 then the above formula is the same as the simple average. However, setting the $w_i = 1 - E_i$ where E_i is the error of the classifier *i* then the weighting is based on the accuracy of every classifier. Setting the $w_i = E_i^{E_i}(1-E_i)^{1-E_i}$ results in the uniform noise model. Setting the $w_i = E_i^{-1}$ minimizes the ensemble error [8]. We will refer to these versions of weighted means as *accuracy, uniform noise,* and *minimum error,* respectively.

5.2.3. Decision Templates

The idea of the *decision templates* combiner [9] is to remember the most typical decision profile for each class ω_j, called *decision template*, DT_j, and then compare it with the current decision profile $DP(\vec{x})$ using a similarity measure *S*. The closest match will label $\vec{x}$. A decision profile contains supports for all classes by all classifiers given a query instance $\vec{x}$. Having *J* classes and *I* classifiers each decision profile is a IxJ matrix. Training of decision templates combiner is very simple: the mean of the decision profiles of all instances belonging to the same class in the training data forms the decision template. Briefly the decision template combiner is a nearest-mean classifier that operates in the decision space rather than on the feature space. Decision Templates were reported to outperform other combiners in a number of studies e.g. [9][10] .

5.2.4. Dempster-Shafer Combination

In [11] a combination method motivated by the evidence combination of Dempster-Shafer theory is proposed. The training of the Dempster-Shafer combiner is straight forward to the training of the decision template combiner, but instead of computing the similarity between a decision template and the decision profile following steps are carried out:

1) Let DT_j^i denote the ith row of decision template DT_j. Denote by $D_i(\vec{x})$ the output of D_i, that is $D_i(\vec{x}) = [d_{i,1}(\vec{x}), \ldots, d_{i,J}(\vec{x})]^T$: the ith row of the decision profile $DP(\vec{x})$. We calculate the "proximity" Φ between DT_j^i and the output of classifier D_i for the input $\vec{x}$.

$$\Phi_{i,j}(\vec{x}) = \frac{(1 + \|DT_j^i - D_i(\vec{x})\|^2)^{-1}}{\sum_{k=1}^{J}(1 + \|DT_k^i - D_i(\vec{x})\|^2)^{-1}}$$

   Where $\|\cdot\|$ is any matrix norm. For example, we can use the Euclidean distances. Thus for each decision template we have I proximities.

2) Using the $\Phi_{i,j}$, we calculate for every class, j = 1, …, J; and for every classifier, i = 1, …, I the following *belief* degrees:

$$b_j(D_i(\vec{x})) = \frac{\Phi_{j,i}(\vec{x}) \prod_{k \neq j}(1 - \Phi_{k,i}(\vec{x}))}{1 - \Phi_{j,i}(\vec{x})[1 - \prod_{k \neq j}(1 - \Phi_{k,i}(\vec{x}))]}$$

3) The final degrees of support are

$$\mu_j(\vec{x}) = K \prod_{i=1}^{I} b_j(D_i(\vec{x}))$$

where *K* is a normalizing constant.

5.3. Finding strong regions of engines and exploit them

As seen in section 4.4, the performance of the single classifiers differs with respect to different input features. Thus the full benefit might be achieved by combining the ideas of classifier selection and classifier fusion: identify regions (in the feature space) where the classifiers have a different reliability than in the overall feature space. In these regions different combination methods can be trained, e.g. a logistic regression. The basic idea is that in regions where the classifiers are differently reliable should be exploited by the combination method, an approach direct in this direction is the estimation of local accuracy described in section 5.3.2. On the other hand it might be that an engine is more reliable when predicting failing regimens, an approach that should discover interesting regions in the decision space is given in section 5.3.1.

5.3.1. Using clusters in decision space

Interesting regions in the decision space are those where the engines do not agree on the same class. Therefore we propose following method that finds clusters in the decision space and learns a separate logistic regression for every cluster that fuses the classifier outputs. The success probabilities provided by the engines are rewritten to be able to express these differences in their decision. Let d_i denote the predicted success provided by classifier i. Then $a_{i,j} = 0$ if classifier i and j agree on the class label, if they disagree, then $a_{i,j} = d_i – d_j$. $a_{i,j}$ is computed for all i,j where j > i. These *agreements* are used as input to K-medoid clustering. For each of the k clusters found by the method a single Logistic regression is learned that operates on the original decisions provided by the engines. When a new instance has to be classified then first the engine’s output is re-encoded and to find the appropriate cluster, afterwards the logistic regression associated with the cluster is used to fuse the output of the classifiers. K the number of clusters is the only parameter of this method and has to be tuned to provide optimal performance. For k=1 the method is identical to fitting a single logistic regression on all available instances.

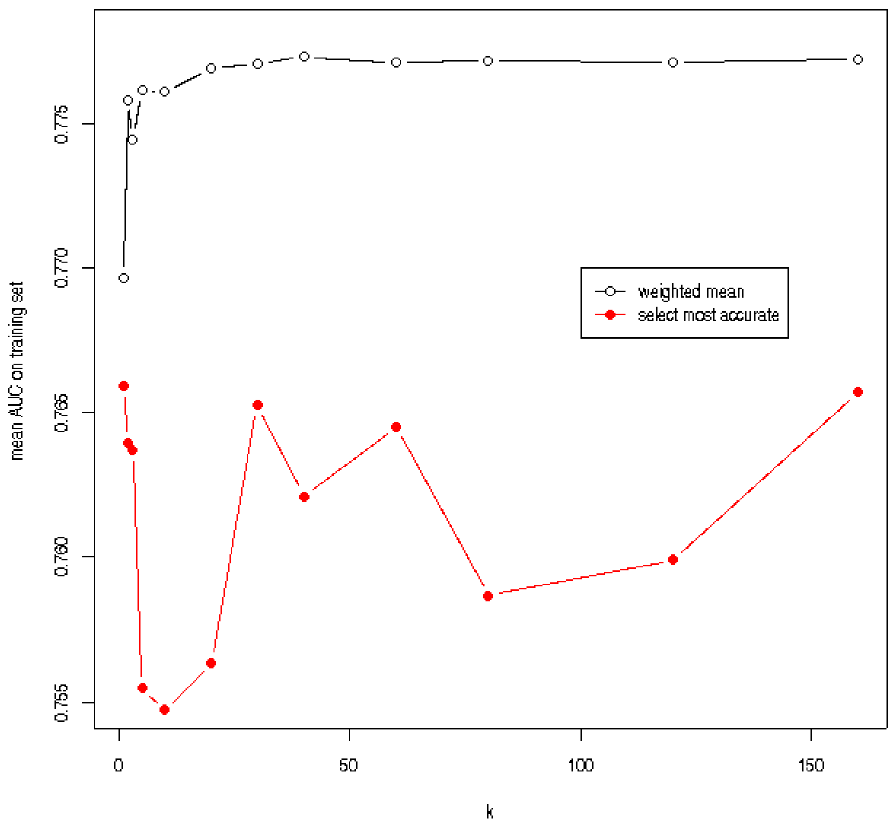
5.3.2. Weighing on basis of local accuracy

The weight a classifier receives when fusing different classifier outputs can also be adjusted according to its reliability in the current region of the feature space. In [12] a method is proposed that uses a k-nearest-neighbour (knn) classifier to select the most reliable classifier from the ensemble. Here we re-weight the classifiers by their reliability rather than selecting the most reliable classifier. For every classifier in the ensemble we train a knn classifier that solves a binary classification problem. Precisely, the knn is trained with the instances of the original training set using the feature set applied by the original engines. Now the class labels do not correspond to the success of a therapy but report whether the prediction engine classified the particular sample correctly in the 10-fold cross-validation setting. When a new instance has to be classified by the ensemble, then first the knn classifier for every prediction engine is used to determine the local reliability, denoted by r_i. The output by the ensemble is then defined by:

$$o = \frac{\sum_i r_i d_i}{\sum_i r_i}$$

where d_i is the support for the positive class. By giving only the output of the currently most reliable engine this approach is transformed in a pure classifier selection approach. Figure 6 depicts the (mean) performance (measured in AUC) of the weighted fusion and the selection approach on the training set. It is obvious that the weighted mean outperforms the selection approach with respect to all sizes of the neighbourhood. Moreover, the results of the selection approach seem to be more unstable, whereas the weighted mean is very stable. The best performance is reached with k=40.

Figure 6: Mean AUC on the training set using different sizes of the neighborhood to select the best classifier (red line) or to compute a weighted mean (black line).



5.4. Combining engines on feature level

In contrast to the combination of the engines output, the efforts of the teams can alternatively be combined on the feature level. Briefly, as described in Chapter 2 every team developed its own list of derived features, therefore using all individually selected sets of features as input to one statistical learning method might outperform the single classifiers or even a combination of single classification results. Prediction methods used here will not be more sophisticated than methods examined for the single engines. Therefore, we will focus on Logistic Regression, Support Vector Machines, and Random Forest for classification.

6. Results

This section describes the results obtained on the training and test set by applying the described combination methods on the engines using either the minimal feature set or the maximal feature set. For the method weighted means we always used the *minimum error* method, since it provided the best results. The oracle performance is computed by taking the *min* and *max* of all the engine's outputs if a failing and successful therapy has to be rated, respectively.

6.1. Accuracy and ROC based measures with classical labelling

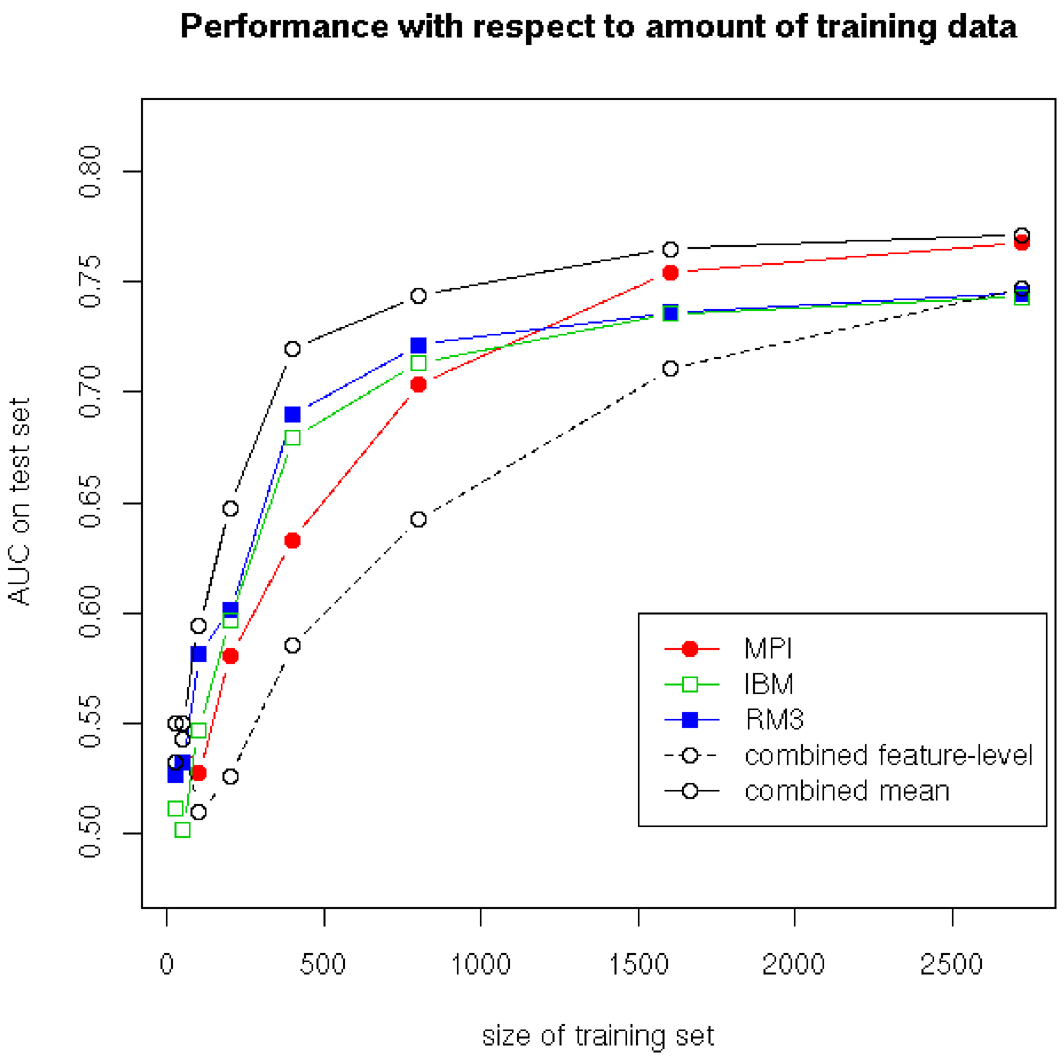
6.1.1. with minimal feature set

Table 4 summarizes the results obtained by using different combination methods with prediction results obtained by the minimal feature set. As can be seen from the table, the performance of the combination methods does not differ a lot and in most of the cases improves the performance from the ensemble above the single best method. However, all methods are quite far from the performance of the oracle. Among the non-trainable methods the *mean* combiner has the best performance. The majority vote has usually a worse AUC but simply because the output of the has only two possible cut-offs 0 and 1, therefore the ROC curve for the majority combiner consists only of one point with connections to the lower left, and upper right corner. Among the Meta classifiers usually the logistic regression provides the best performance. However, the gain with respect to the *mean* is not significant. Among the trainable methods the decision templates and the Dempster-Shafer combiner show very similar performance, and can not improve above the logistic regression. Approaches designed to exploit strong regions of the prediction engines do not perform better than the mean either. Finally, when combining the engines on the feature level a different behaviour is visible. All three selected approaches have a worse performance than the single best method on the training set. But on the test set the SVM has the same performance as the decision template combiner and the Random Forest is slightly better with respect to AUC. Figure 7 shows the learning curve of the single engines and their combination using the *mean* combiner and the combination on feature level using logistic regression. The results were computed using 5 repetitions of 5-fold cross-validation. In every repetition a subset of the training set was randomly selected and used for training the engines. Performance was measure on the full test set. Remarkably the performance for the mean combiner is increasing faster than the performance of any of the single engines, and reaches a good performance with already 400 instances. The learning curve for the mean clearly shows that the output of the ensemble is not dominated by the single best classifier as results on the complete training set might indicate. Moreover, a combination on feature level shows the worst learning behaviour, and is significantly worse than the mean of the single engines although the results using the complete training set are similar.

Table 6: Performance achieved by applying different combination methods for engines using the minimal feature set. Performance is measured in AUC and Accruracy on training set and test set.

		AUC		Accuracy	
		Train	Test	Train	Test
Reference	Single Best	0.766 (0.030)	0.768	0.754 (0.031)	0.748
	Oracle	0.914 (0.015)	0.911	0.842 (0.025)	0.844
Non-trainable Methods	Min	0.771 (0.020)	0.765	0.746 (0.027)	0.761
	Max	0.760 (0.023)	0.765	0.742 (0.030)	0.731
	Median	0.773 (0.020)	0.766	0.759 (0.027)	0.766
	Mean	0.777 (0.020)	0.772	0.760 (0.024)	0.744
	Majority	0.683 (0.023)	0.660	0.759 (0.027)	0.738
	Product	0.776 (0.020)	0.772	0.759 (0.025)	0.744
Meta Classifiers	QDA	0.771 (0.020)	0.763	0.755 (0.031)	0.738
	Logistic	0.778 (0.021)	0.774	0.762 (0.028)	0.744
	Decision Trees	0.718 (0.044)	0.741	0.748 (0.032)	0.757
	Naïve Bayes	0.732 (0.027)	0.740	0.759 (0.027)	0.738
Trainable Methods	Decision Templates	0.777 (0.021)	0.774	0.755 (0.027)	0.754
	Dempster-Shafer	0.777 (0.021)	0.772	0.755 (0.024)	0.754
	Weighted means	0.777 (0.020)	0.772	0.760 (0.024)	0.748
Exploit Strong Regions	Clustering k=2	0.775 (0.019)	0.773	0.758 (0.029)	0.741
	Local accuracy K = 40	0.777 (0.020)	0.771	0.761 (0.025)	0.741
Combine on feature level	Logistic	0.750 (0.026)	0.747	0.745 (0.029)	0.751
	Random Forest	0.764 (0.027)	0.782	0.743 (0.025)	0.751
	SVM	0.763 (0.016)	0.774	0.757 (0.025)	0.756

Figure 7: Learning curves for the three single engines, the mean combiner, and the combination of feature level using the minimal feature set. All models use logistic regression.



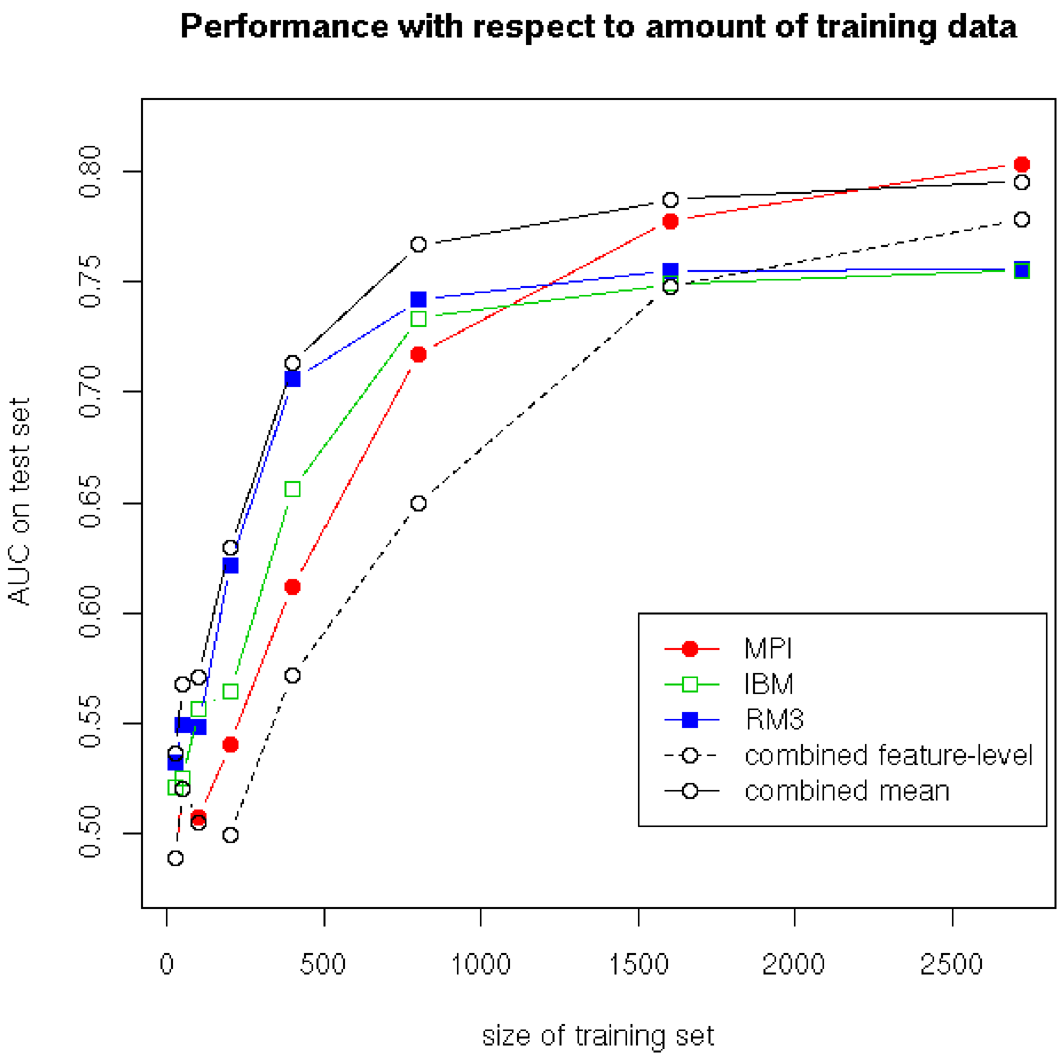
6.1.2. with maximal feature set

The results using the maximal feature set do not differ qualitatively from the results obtained with the minimal feature set. Precisely, the AUC on the training set is improved by 0.025 and on the test set by 0.04 compared to the minimal set. All results are summarized in table 5. The learning curves depicted in figure 8 show a similar behaviour as well.

Table 7: Performance achieved by applying different combination methods for engines using the maximal feature set. Performance is measured in AUC and Accruracy on training set and test set.

		AUC		Accuracy	
		Train	Test	Train	Test
Reference	Single Best	0.789 (0.023)	0.804	0.780 (0.032)	0.751
	Oracle	0.917 (0.013)	0.920	0.850 (0.022)	0.860
Non-trainable Methods	Min	0.792 (0.021)	0.793	0.760 (0.030)	0.764
	Max	0.779 (0.021)	0.779	0.757 (0.030)	0.741
	Median	0.789 (0.029)	0.786	0.768 (0.029)	0.761
	Mean	0.794 (0.019)	0.793	0.780 (0.028)	0.781
	Majority	0.697 (0.027)	0.683	0.768 (0.029)	0.761
	Product	0.794 (0.019)	0.795	0.780 (0.027)	0.771
Meta Classifiers	QDA	0.790 (0.022)	0.794	0.769 (0.027)	0.764
	Logistic	0.798 (0.020)	0.805	0.781 (0.030)	0.771
	Decision Trees	0.722 (0.033)	0.678	0.777 (0.032)	0.757
	Naïve Bayes	0.752 (0.278)	0.753	0.768 (0.029)	0.761
Trainable Methods	Decision Templates	0.796 (0.019)	0.797	0.766 (0.026)	0.767
	Dempster-Shafer	0.796 (0.019)	0.796	0.767 (0.026)	0.764
	Weighted means	0.795 (0.019)	0.795	0.783 (0.028)	0.777
Exploit Strong Regions	Clustering k=2	0.797 (0.018)	0.800	0.783 (0.028)	0.784
	Local accuracy K=20	0.795 (0.019)	0.791	0.781 (0.029)	0.777
Combine on feature level	Logistic	0.786 (0.021)	0.779	0.780 (0.029)	0.767
	Random Forest	0.800 (0.022)	0.810	0.772 (0.025)	0.786
	SVM	0.798 (0.013)	0.804	0.783 (0.024)	0.781

Figure 8: Learning curves for the three single engines, the mean combiner, and the combination of feature level using the maximal feature set. All models use logistic regression.



6.2. Correlation of predicted and expected change in VL

Most of the advanced methods that were introduced in section 5 were designed to compute a consensus for classification results.

6.2.1. with minimal feature set

Table 8: Regression performance achieved by applying different combination methods for engines using the minimal feature set.

		Correlation (*r*) between actual and predicted change in log(VL)	
		Train	Test
Reference	Single Best	0.471 (0.020)	0.527
	Oracle	0.645 (0.029)	0.719
Non-trainable Methods	Min	0.466 (0.021)	0.534
	Max	0.428 (0.038)	0.481
	Median	0.464 (0.027)	0.515
	Mean	0.475 (0.023)	0.531
	Product	0.455 (0.022)	0.535
Meta Classifiers	Linear Regression	0.482 (0.027)	0.541

6.2.2. with maximal feature set

Table 9: Regression performance achieved by applying different combination methods for engines using the maximal feature set.

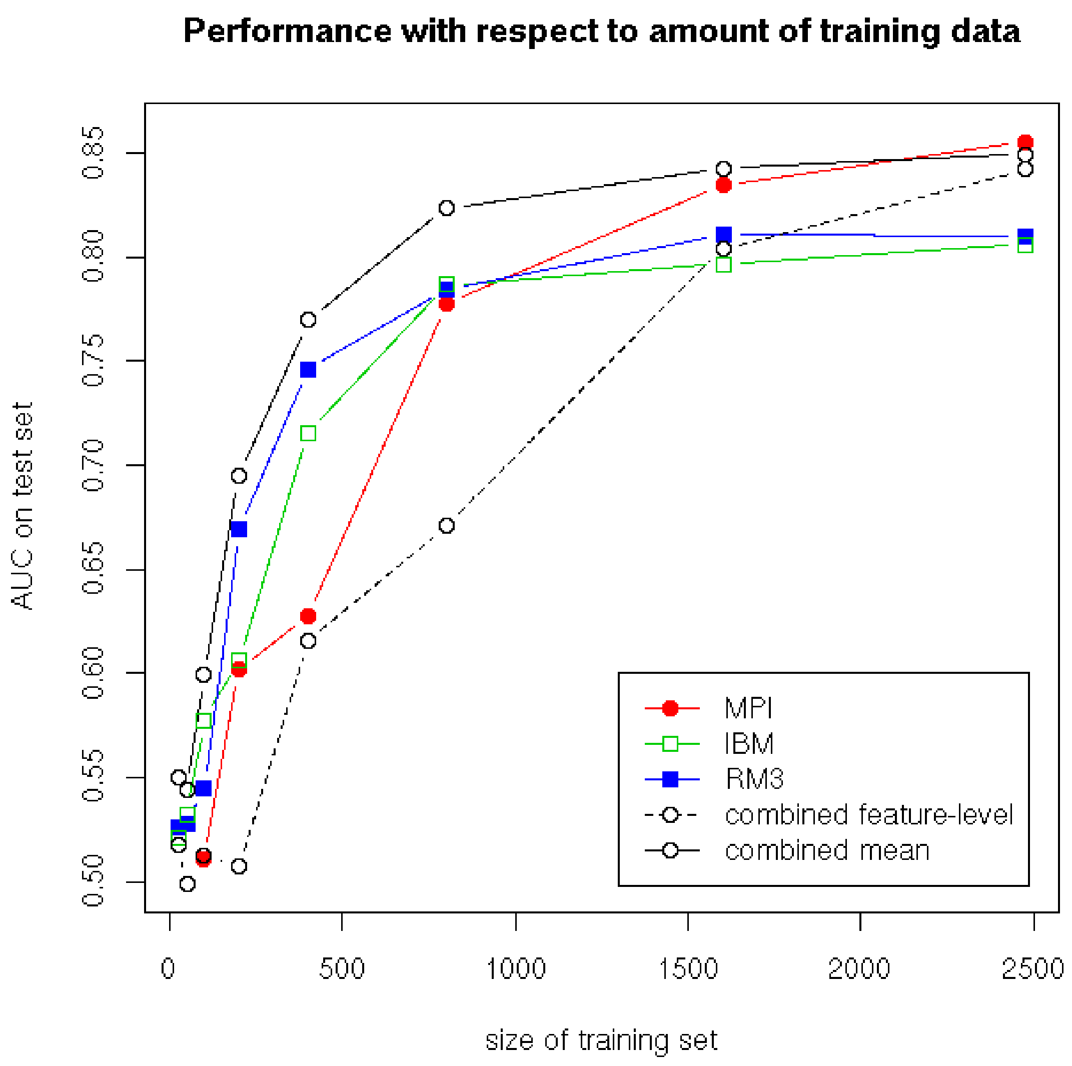
		Correlation (*r*) between actual and predicted change in log(VL)	
		Train	Test
Reference	Single Best	0.679 (0.020)	0.678
	Oracle	0.834 (0.012)	0.814
Non-trainable Methods	Min	0.679 (0.021)	0.676
	Max	0.676 (0.020)	0.667
	Median	0.686 (0.020)	0.674
	Mean	0.691 (0.019)	0.681
	Product	0.595 (0.028)	0.608
Meta Classifiers	Linear Regression	0.693 (0.019)	0.682

7. Detailed description of selected approach

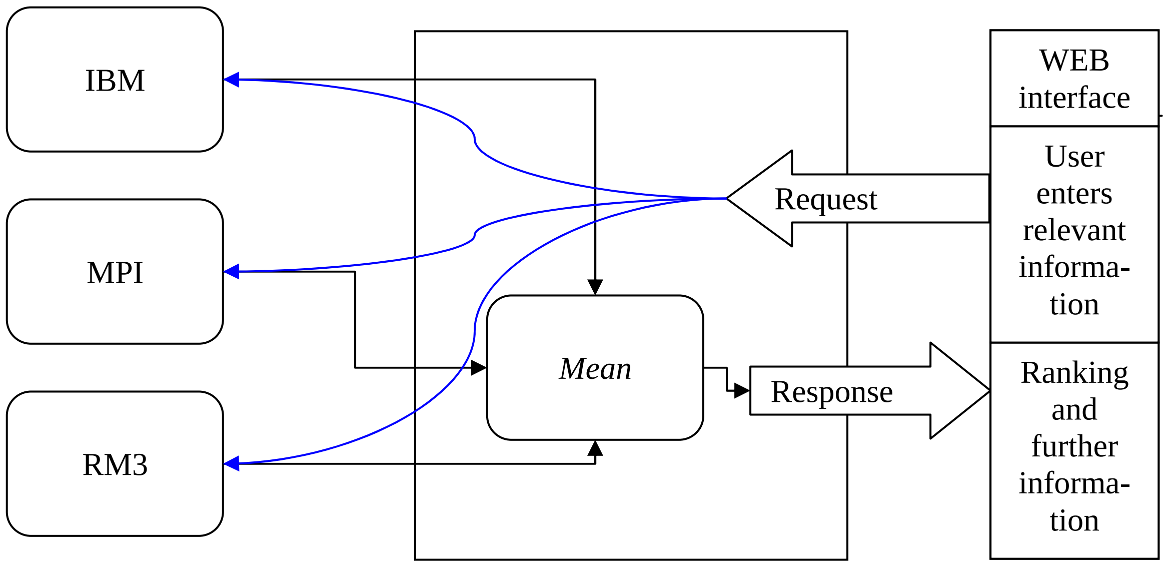
We examined a number of methods to combine the predictions of the singe engines. The methods range from simple, so-called non-trainable ones, to more sophisticated approaches that required an additional training step. On both sets of available features, minimal and maximal, the performance of all methods was very similar and only improved a little above the single best method on the test set. It turned out the simple non-trainable methods perform quite well, especially the mean (or sum) and the product combiner. This fact is frequently reported in the classifier fusion literature, e.g. in [9, 10, 13]. Here we focused on finding the best combination strategy for a particular learning scenario. The advantages of the mean combiner is that is does not require an additional training set, although range among the best fusion methods, and is an easy to explain combination strategy to the user of the system. Moreover, the learning curves for the minimal and maximal feature set show that the mean is not dominated by the single best approach, but even for a limited amount of training samples (n=200 minimal features, n=400 maximal features) reaches a significantly better performance than any single approach using the same amount of data. This shows a more robust behaviour of the combined system, even though the performance achieved with the full amount of training data is close to the single best approach.

In section 4.3. we reported that all engines frequently agree that a particular therapy should be successful when indeed it was labelled as failure. The close investigation revealed that a large fraction of these instances indeed reaches a viral load below 500 cp per ml during the course of the treatment. This shows a further benefit of the combined system: if it was only a single prediction engine these cases were some among the remaining failures, but three engines agreeing on the wrong label raised suspicion. However, if the definition of failure is changed so that at no point the viral load is allowed to drop below the threshold of 500 cp per ml, and training and testing is repeated with the reduced dataset then performance reaches an AUC of 0.82 and 0.85 for the minimal and the maximal feature set using the mean combiner, respectively. Figure 9 shows the learning curves for the reduced training and test set when using the maximal feature set and logistic regression. It is possible that some of the trainable approaches might benefit from the changed failure definition as well. However, the good performance of the oracle leads to the conclusion that the diversity of predictions provided by the three systems is sufficient, although using the same statistical learning methods.

Figure 9: Learning curves for the three single engines, the mean combiner, and the combination of feature level using the maximal feature set on the reduced dataset. All models use logistic regression.



Based on these results we propose following setup for the EuResist prediction engine:


IBM
MPI
RM3
Mean
Request
Response
WEB interface
User enters relevant informa-tion
Ranking and further informa-tion

The user enters the available data (viral sequences, previous treatments, clinical markers,…) on a web interface (which is subject to another deliverable), this information is transferred to the single prediction systems using SOAP. Each of the engines decides now, based on the available information, which prediction model should be applied and/or replaces missing information. Every engine separately computes the probability of success for a fixed list of regimens; note that this list is augmented by specific drug combinations the user wants to rate. The list of success probabilities is sent to the combiner (using SOAP). The combiner computes the consensus for every treatment by taking the *mean* of the predicted success probabilities provided by the three systems. The consensus prediction is used to compute a ranking of all the regimens that have to be rated. The regimen predicted to be most effective with respect to the provided information is at the top of the ranking followed by the remaining ones. This ranking is then part of the visible output on the web interface along with warnings concerning use of specific drugs (details in deliverable D6.1).

The intended setup allows independent retraining of the individual models without the need of retraining the combiner. Moreover, the combination process is robust, effective (although no training is required), and intuitive for the end user.

8. References

1. Beerenwinkel N, Rahnenfuhrer J, Daumer M, Hoffmann D, Kaiser R, Selbig J & Lengauer T. Learning multiple evolutionary pathways from cross-sectional data. *J Comput Biol* 2005; **12:**584-598.
2. Beerenwinkel N, Daumer M, Sing T, Rahnenfuhrer J, Lengauer T, Selbig J, Hoffmann D & Kaiser R. Estimating HIV evolutionary pathways and the genetic barrier to drug resistance. *J Infect Dis* 2005; **191:**1953-1960.
3. Altmann A, Beerenwinkel N, Sing T, Savenkov I, Daumer M, Kaiser R, Rhee S, Fessel W, Shafer R & Lengauer T. Improved prediction of response to antiretroviral combination therapy using the genetic barrier to drug resistance. *ANTIVIRAL THERAPY* 2007; **12:**169-178.
4. Kuncheva LI. *Combining Pattern Classifiers: Methods and Algorithms*. 2004: Wiley-Interscience.
5. Kittler J, Hatef M, Duin RPW & Matas J. On combining classifiers. *Ieee Transactions on Pattern Analysis and Machine Intelligence* 1998; **20:**226-239.
6. Tax DMJ, van Breukelen M, Duin RPW & Kittler J. Combining multiple classifiers by averaging or by multiplying? *Pattern Recognition* 2000; **33:**1475-1485.
7. Selbig J, Mevissen T & Lengauer T. Decision tree-based formation of consensus protein secondary structure prediction. *Bioinformatics* 1999; **15:**1039-1046.
8. Fumera G & Roli F. Performance Analysis and Comparison of Linear Combiners for Classifier Fusion. *Structural, Syntactic, and Statistical Pattern Recognition: Joint Iapr International Workshops Sspr 2002 and Spr 2002, Windsor, Ontario, Canada, August 6-9, 2002: Proceedings* 2002.
9. Kuncheva LI, Bezdek JC & Duin RPW. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition* 2001; **34:**299-314.
10. Kuncheva L. Combining classifiers by clustering, selection and decision templates. Edited by Editor|. Year|; p.^pp. Pages|. City|: Publisher|.
11. Rogova G. Combining the Results of Several Neural-Network Classifiers. *Neural Networks* 1994; **7:**777-781.
12. Woods K, Kegelmeyer WP & Bowyer K. Combination of multiple classifiers using local accuracy estimates. *Ieee Transactions on Pattern Analysis and Machine Intelligence* 1997; **19:**405-410.
13. Liu R & Yuan B. Multiple classifiers combination by clustering and selection. *Information Fusion* 2001; **2:**163-168.