UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC)
– BARCELONATECH

BACHELOR'S THESIS

Using Random Fourier Features with Random Forest

Author:
Albert RIBES

Supervisor:
Lluís A. BELANCHE

Computer Science

April 14, 2019

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BARCELONATECH

Abstract

Facultat d'Informàtica de Barcelona (FIB)
Computer Science

Bachelor Degree in Computer Science

Using Random Fourier Features with Random Forest

by Albert RIBES

Kernel methods are powerful tools to capture nonlinear patters behind the data. They perform an implicit mapping to a high (even infinite) dimensional feature space, but their running time increases significantly with the size of the dataset. In contrast, linear methods are much faster to train, but they have a more limited representational power. Fortunately, there are some ways to construct a random mapping into a relatively low-dimensional feature space which allows models to capture nonlinear patters with linear methods. In this project we study how Random Fourier Features and the Nyström method can be used with Logistic Regression, Support Vector Machine and Decision Tree to speed up both a single model and an ensemble of estimators. We empirically show that they can be used to successfully increase the accuracy of Logistic Regression and Support Vector Machine and we study different ways to train and ensemble with this estimators using Random Fourier Features and Nyström.

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BARCELONATECH

Abstract

Facultat d'Informàtica de Barcelona (FIB)
Computer Science

Bachelor Degree in Computer Science

Using Random Fourier Features with Random Forest

by Albert RIBES

Los métodos kernel son herramientas muy potentes para capturar patrones no lineales tras los datos. Para ello realizan una transformación implícita a un espacio de alta (incluso infinita) dimensionalidad, pero su tiempo de ejecución crece significativamente con el tamaño del conjunto de datos. Por contra, los métodos lineales son más rápidos de entrenar, pero tienen una capacidad de representación más limitada. Afortunadamente, hay formas de construir una transformación aleatoria a un espacio de baja dimensionalidad que permite a los modelos capturar patrones no lineales con métodos lineales. En este proyecto estudiamos como las Random Fourier Features y el método Nyströmse pueden usar junto con Regresión Logística, Máquina de Vectores de Soporte y Árbol de Decisión para mejorar un modelo solo y también un conjunto de estimadores. Mostramos empíricamente que se pueden usar para aumentar con éxito la precisión de Regresión Logística y Máquina de Vectores de Soporte y estudiamos varias formas de entrenar un conjunto de estimadores usando Random Fourier Features y Nyström.

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BARCELONATECH

Abstract

Facultat d'Informàtica de Barcelona (FIB)
Computer Science

Bachelor Degree in Computer Science

Using Random Fourier Features with Random Forest

by Albert RIBES

Els mètodes kernel són eines molt potents per capturar patrons no lineals rere les dades. Per fer-ho fan una transformació implícita cap a un espai de alta (fins i tot infinita) dimensionalitat, però el seu temps d'execució augmenta significativament amb la grandaria del conjunt de dades. En contraposició, el mètodes lineals són més ràpids d'entrenar, però tenen una capacitat de representació més limitada. Afortunadament, hi ha maneres de construir una transformació aleatòria cap a un un espai de baixa dimensionalitat que permeti als models capturar patrons no lineals amb mètodes lineals. En aquest projecte estudiem com les Random Fourier Features y el mètode Nyström es poden fer servir juntament amb Regressió Logística, Màquina de Vectors de Suport i Arbre de Decisió per a millorar un sol model i també un conjunt d'estimadors. Mostrem empíricament que es poden fer servir per augmentar satisfactòriament la precisió de Regressió Logística i Màquina de Vectors de Suport i estudiem algunes maneres de entrenar un conjunt d'estimadors fent servir Random Fourier Featuers i Nyström.

Contents

Chapter 1

Introduction

1.1 Problem to study

Supervised Learning uses statistical and mathematical models to predict a response variable from a classification or regression problem. Usually it does so by trying to minimise an error function. The better a model is, the higher the accuracy it will obtain on new, unseen data.

For classification problems the error function might be proportion of instances that are missclassified, and for regression problems the Mean Squared Error is usually used. If y_i is the correct response variable for instance i and the predicted one is $\hat{y}_i$, it is defined as $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, where n is the number of instances.

Reducing the value of the error function of the models is one of the main topics in the field. But it is not easy to achieve it. Usually, it comes at the cost of increasing the computation time to produce the model. Therefore, a trade-off needs to be made between the accuracy obtained and the training time.

In this project we study some recent approaches to improve the trade-off of currently used Machine Learning methods. In particular, we study how Kernel approximation techniques could make some procedures feasible in very large datasets and also how to increase the accuracy of some models at the expense of some more time.

1.2 Project proposal

The current development of Machine Learning has opened many fronts and techniques trying to solve many of the difficulties that the field has.

One known issue is the Bias–Variance dilemma[13]. While solving a classification or regression problem, the expected generalization error of a model is the sum of three error terms: the squared bias, the variance and the irreducible error. While the latter, as the name suggests, cannot be reduced, since it is caused by the inherent random noise in the data, the other two terms seem to have an inverse proportion: trying to reduce one of them increases the other most of the times. Ensemble methods[22] were developed in an attempt to reduce both of them (or at least their sum). Although they can outperform some models their usage is mostly restricted to a small subset of all available models. This is because they only increase significantly the accuracy on unstable methods.

Another advance in Machine Learning has been the usage of kernel methods[10]. They are useful to implicitly transform the data into another feature space with better properties, such as a clear dividing margin between classes of data. They are very effective, but their high computational costs has caused their use to be limited to just some specific problems or with a small number of instances. There are some

less expensive approaches to approximate these methods, but they are not widely used[23][29].

And then there is a collection of classical algorithms which have the advantage of being very simple and straightforward, although they don't usually get the highest scores.

There is a collection of techniques that have shown some good results by their own, but still they haven't been tested in combination with the others. If we could mix some of these methods, maybe we could find new Machine Learning methods, with better accuracy or trade-off.

In this project, we try some combinations of currently known techniques which could produce better results or show new useful model designs. On the one hand, we try to extend the usage of ensemble methods to new basic models. Currently, it doesn't make much sense to train an ensemble of Support Vector Machines (SVM) or of Logistic Regression models, because they are so stable and most of the estimators would predict the same answer. We propose the use of random kernel approximations such as Random Fourier Features**??** (RFF) or the Nyström method[29] to increase the instability of these models and thus be able to succesfuly train and ensemble with them, hopefully increasing the accuracy of a single one.

On the other hand, the usage of these Random Kernel approximations could allow us to use some methods which right now are not accessible. Support Vector Machines cannot be used with non-linear Kernels such as the Radial Basis Function (RBF) kernel on big datasets, since the cost is $\mathcal{O}(n^3)$ with the number of instances. But if we transform the data to some space almost equivalent to the one of the RBF kernel, we can use a Linear Support Vector Machine, which is less expensive to train, and achieve a similar accuracy. In fact, with the usage of an ensemble, the results could be improved. Similar approaches have already been studied, and they have showed good results [25] [31] [23].

For this study, we have formulated some hypothesis and we try to confirm or refuse them based on the experimental results. The hypothesis are:

1. An SVM (explained in 2.2.3) using RFF or Nyström could achieve an accuracy similar to using the Radial Basis Function kernel (explained in 2.4) but with much less training time.

2. Training an ensemble of SVMs or Logistic Regression (explained in 2.2.2) models using RFF or Nyström could increase the accuracy of a single estimator.

3. Mixing a random mapping with the Bootstrap (from Bagging, explained in 2.3) could cause an excess of randomness and hence a bad accuracy compared to just using the random mapping.

4. Basic algorithms which are not based on the dot product of the input data such as the Decision Tree (see 2.2.1) will not benefit so much of the usage of RFF and Nyström than those that do, like SVM or Logistic Regression.

In the following pages we show an experimental set up to check these hypothesis and the results of the experiments. In Chapter 2 we explain some of the Machine Learning concepts which are needed to understand the rest of the project. In 3 we explain with more detail how this study is developed and what experiments are executed. In 4 we show the results obtained with the experiments and discuss the hypothesis suggested based on them. In 5 we present the conclusions of this project and propose some future work related to this topic. Finally, in Chapter 6 we present a sustainability report of this project.

Chapter 2

Background Information and Theory

2.1 Machine Learning

Machine Learning uses statistical and mathematical models to give computational answers based on data to problems when there is no known formula of procedure.

In the subfield of Supervised Learning, the objective is to predict a numerical or categorical variable in response to some input data, and the way of doing it is to feed the model with lots of different examples for which we already know the correct answer, and we expect the models to be able to predict the correct answer to instances that it hasn't seen before. When it does, we say that the model is able to generalize.

When a model is trained with some data, there is always a risk of overfitting [18]. For a model to overfit means that it adjusts very well to the data that is has seen, but can't predict the correct answer to new, unseen data. This happens because it has not only fitted the relevant information, but also random noise present in the data sample, and thus it generalizes poorly. In the extreme case of ovefitting, the model tends to memorise the data sample.

For this reason, when a Machine Learning algorithm is trained the data is split in two subsets, a *Training dataset* and a *Testing dataset*. The training dataset will be used to train the model, while the Testing datasets will be used only to check it. If a model has generalized well, it will achieve a good accuracy score on both the training and the testing dataset, but if it has overfitted it will show good results in the training dataset and bad ones in the testing dataset.

Many models need some parameters to tune the behaviour of the algorithm. For example, some of them are used to adjust how much a model will fit to the data. We usually call these "hyperparameters". The correct value for them is not straightforward, and it is normally chosen with a resampling process called "cross-validation"[16]. This process consists of splitting the training dataset in many subsets and check many possible values for the hyperparameters in order to see which one gets a higher accuracy with unseen data.

2.2 Some currently used Machine Learning models

2.2.1 Decision Tree

Decision Tree[6][20] is a predictive model which uses the training data to build a tree where each node splits the data in two sets according to some feature, and the leafs contain the set of instances that belong to some class (in classification problems) or that has a similar numerical response variable (for regression problems).

To predict the answer to a new instance, it uses the features to "decide" the nodes to cross until it reaches a leaf. The response given is the most prevailing class in the leaf for classification problems, or the mean of the values of the rest of the instances in the leaf.

To decide what feature to use to split a node in two subsets, it uses the Gini impurity: it will pick the feature that minimises the sum of the Gini impurity of the two child nodes. Given a node with instances belonging to k classes, if p_i is the proportion of instances that belong to class i, the Gini impurity of the node is $1 - \sum_{i=1}^{k} p_i^2$.

Decision Trees have the advantages that it is easy to interpret the tree produced and that it is very fast to build the tree. The way to avoid overfitting is to limit its growth.

These models are very unstable. This means that small differences in the training data can produce very different Decision Trees. This property is very useful to build an ensemble of estimators to produce better answers. Random Forest is an algorithm that trains many Decision Trees with some sort of randomization.

2.2.2 Logistic Regression

Logistic Regression[11] models the probability that an instance belongs to a class, and predicts the class with a higher probability. To do so it uses the *logistic sigmoid function*[17], defined by:

$$\sigma(a) = \frac{1}{1 + exp(-a)} \tag{2.1}$$

Once the vector $w \in \mathbb{R}^d$ has been found, the predicted probability that an instance $\boldsymbol{x} \in \mathbb{R}^d$ belongs to a class is $y(\boldsymbol{x}) = \sigma(w^\intercal \boldsymbol{x})$.

Given $D = \{\boldsymbol{\chi}, \boldsymbol{t}\}$, where $\boldsymbol{\chi} = \{\boldsymbol{x}_1, \dots \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{t} = \{0, 1\}^n$ it tries to maximize a likelihood function that can be written

$$p(\boldsymbol{t}|w) = \prod_{i=1}^{n} y_i^{\boldsymbol{t}_i} (1 - y_i)^{1 - \boldsymbol{t}_i} \tag{2.2}$$

where $y_i = \sigma(w^\intercal \boldsymbol{x}_i)$.

2.2.3 Support Vector Machines

Support Vector Machine[10] (SVM) is a model that finds in hyperplane that divides the data in two sets. In two-class classification problems, each side of the hyperplane contains the instances of each of the classes. It does so by converting the problem to an optimization one.

Given some data $D = \{\boldsymbol{\chi}, \boldsymbol{y}\}$, where $\boldsymbol{\chi} = \{\boldsymbol{x}_1, \dots \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $\boldsymbol{y} = \{-1, +1\}^n$, the optimization problem consists on finding $\boldsymbol{\alpha} \in \mathbb{R}^n$ the maximises

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\intercal \boldsymbol{x}_j \tag{2.3}$$

subject to

$$0 \leq \alpha_i \leq C; \forall i \tag{2.4}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.5}$$

C is an hyper-parameter to tune the amount of penalization for missclassified instances or instances located within the margin zone.

If we compute

$$w = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \tag{2.6}$$

and

$$b = y_i - w\boldsymbol{x}_i \tag{2.7}$$

for any i so that $\alpha_i \neq 0$, we can compute the class of $\boldsymbol{x}_0$ with

$$sign(w\boldsymbol{x}_0 + b) \tag{2.8}$$

Note that this algorithm just uses the dot product of the input data, not the data itself. This property allows us to use the Kernel Trick with them. See 2.4

2.3 Ensemble Methods

Ensemble methods[22] are a technique used in Machine Learning to reduce the overall accuracy error of a basic classification or regression model. The idea is that a commimtee of models is expected to learn better than a single one.

Some ensemble methods are focused on decreasing the error caused by the variance of the model. One example is *Bagging*[5]. Others are focused on decreasing the bias error, like *Boosting*[15].

In Bagging, every model in the ensemble vote with equal weight. Thus, it is important to promote the variance among each of the models, since not doing it would be equivalent to training just one model. Ideally, one would train each of the models with totally different datasets, with no correlation among them. But in practice this is not always possible, because of a limited number of instances to train. One alternative is to use a technique called *Bootstrap*[14]. Bootstrap allows to generate many different instances of a dataset by performing a resampling.

Given a dataset D of size n, Bootstrap generates m new datasets D_i of size n by sampling instances from D uniformly and with replacement. This means that some of the instances in D may be repeated in D_i, and others may not appear at all. With a large n, it is expected that each dataset D_i will contain 63.2% of the instances in D.

Theoretically Bagging could be used with any kind of method. However, for most of them Bootstrap is not enough to decorrelate the estimators. In practice, Bagging is mostly used with Decision Tree, given that this method produces very different trees with a small variation in the data. Random Forest[7] is an algorithm that trains many Decision Trees with a Bagging. Instead of building the tree in a deterministic way, in each split it chooses a random subset of features on which to perform the separation. Besides, it lets the estimators overfit, since it has a positive impact in reducing the overall variance of the Forest.

2.4 The kernel trick

A Kernel[3] is a function that equals to the inner product of inputs mapped into some Hilbert Space [1] , i.e:

$$\kappa(x, y) = \langle \phi(x)\phi(y) \rangle \tag{2.9}$$

They are interesting in Machine Learning because we don't need to know the explicit function $\phi(\cdot)$. In fact, $\phi(\cdot)$ could map the data to a Hilbert Space with infinite dimensions, and we could still compute $\phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y})$ through the kernel κ

Support Vector Machines (explained in 2.2.3) can benefit a lot of Kernel Functions. SVMs solve an optimization problem to maximise

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\intercal \boldsymbol{x}_j \tag{2.10}$$

in order to find an hyperplane that separates the data points in two classes. But with some problems there may not exist such hyperplane, and so it would be needed to map the data to a different feature space. If we did that, then the function to maximise would be

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)^\intercal \phi(\boldsymbol{x}_j) \tag{2.11}$$

As we said previously, SVMs don't work with the data points alone, but just with their inner products. Thus, a Kernel could be used to define the optimization problem as

$$L = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{2.12}$$

This approach has one big advantage: as long as the learning technique relies only on the inner product of the input, the underlying mapping $\phi(\cdot)$ does not need to be explicitly calculated and can, in fact, be unknown[8].

Kernel functions can be characterised with the Mercer's condition [21]. It says that given a function $\kappa(\boldsymbol{x}, \boldsymbol{y})$, there exists a mapping $\phi(\cdot)$ so that $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{y})$ if and only if for any $g(\boldsymbol{x})$ such that $\int g(\boldsymbol{x})^2 d\boldsymbol{x}$ is finite then $\int \kappa(\boldsymbol{x}, \boldsymbol{y}) g(\boldsymbol{x}) g(\boldsymbol{y}) \geq 0$.

There are many known Kernels. One that is very popular is the Radial Basis Function Kernel[27], RBF. This kernel is defined as:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = e^{-\gamma \left\| x - y \right\|^2} \tag{2.13}$$

where $\gamma > 0$ is a free parameter. The value of this Kernel decreases with the euclidean distance of the parameters, so it can be interpreted as a measure of similarity. The feature space of this kernel has infinite number of dimensions.

When a kernel is used with an SVM, the answer can be computed with

$$sign \left(\sum_{i=1}^{n} \alpha_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) \right) \tag{2.14}$$

[1] A Hilbert space is a generalization of the Euclidean Space which contains the structure of an inner product that allows length and angle to be measured.

SVMs using the RBF kernel have a huge ability to fit to the data, and is able to separate classes for very difficult problems. The problem is that the optimization of the function

2.5 Random Fourier Features

A kernel function $\kappa(\boldsymbol{x}, \boldsymbol{y})$ with $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ equals the inner product of inputs mapped with some function $\phi(\cdot)$, so that $\phi(\boldsymbol{x})^\intercal \phi(\boldsymbol{y}) = \kappa(\boldsymbol{x}, \boldsymbol{y})$. But $\phi(\cdot)$ could be a mapping to an infinitely-dimensional space, so calculating $\phi(\boldsymbol{x})$ is not possible for some kernels.

Random Fourier Features[23] provide a way to, given a kernel $\kappa(\boldsymbol{x}, \boldsymbol{y})$, explicitly map the data to a low-dimensional Euclidean inner product space using a randomized feature map $z : \mathbb{R}^d \mapsto \mathbb{R}^D$ so that the inner product between a pair of transformed points approximates their kernel evaluation, i.e:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^\intercal \phi(\boldsymbol{y}) \approx z(\boldsymbol{x})^\intercal z(\boldsymbol{y}) \tag{2.15}$$

To approximate the RBF kernel, it uses the Bochner's Theorem, which says:

Theorem 1. *[24] A continuous kernel $\kappa(x, y) = \kappa(x - y)$ on $\mathbb{R}^D$ is positive definite if and only if $k(\delta)$ is the Fourier Transform of a non-negative measure.*

Since it is known that RBF is shift-invariant and positive definite, then its Fourier transform is a proper probability distribution, and so

$$\kappa(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{R}^D} p(w) e^{iw^\intercal(\boldsymbol{x}-\boldsymbol{y})} = \int_{\mathbb{R}^D} p(w) cos\left(w^\intercal(\boldsymbol{x} - \boldsymbol{y})\right) \tag{2.16}$$

A random feature can be obtained by picking $w \sim \{\mathcal{N}(0, 2\gamma)\}^d$ and $b \sim \mathcal{U}(0, 2\pi)$ and computing $\sqrt{2} cos(w^\intercal \boldsymbol{x} + b)$. To generate a lower variance approximation of $\phi(\boldsymbol{x})$ with D features we can concatenate D randomly chosen features $(f_1, \ldots, f_D)$ into a column vector and normalize each component by $\sqrt{D}$.

It is guaranteed an exponentially fast convergence in D between $z(\boldsymbol{x})^\intercal z(\boldsymbol{y})$ and $\kappa(\boldsymbol{x}, \boldsymbol{y})$.

2.6 Nyström

The Nyström[29] method is a general method for low-rank approximations of kernels. It achieves this by subsampling the data on which the kernel is evaluated.

In kernel methods the data can be represented in a kernel matrix K, where $K_{i,j} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The problem of these methods is their high computational cost associated with the kernel matrix: with non-linear kernels, the cost of training the model is cubic with the number of instances, something unacceptable for large-scale problems.

The Nyström method consists on generating an approximation of the kernel matrix of ranq q, where q can be a lot smaller than the number of instances, without any significant decrease in the accuracy of the solution. This way, if there are n instances in a dataset, the complexity can be reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(nq^2)$.

With Nyström, given a kernel $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})\phi(\boldsymbol{y})$, one can construct a mapping $z : \mathbb{R}^d \mapsto \mathbb{R}^q$ so that $z(\boldsymbol{x}) \approx \phi(\boldsymbol{x})$. This function defines each component j as $z_j(\boldsymbol{y}) = \frac{1}{q} \sum_{i=1}^{q} \kappa(\boldsymbol{y}, \boldsymbol{x}_i) g_i(\boldsymbol{x}_i)$, where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_q$ are some chosen instances and $g_i(\cdot)$ comes from a column from the Singular Value Decomposition of the approximated kernel matrix.

Chapter 3

Project Development

3.1 General Idea

RFF and Nyström can be very useful for two main reasons:

- They allow us to perform an explicit mapping approximating the Kernel feature space.
- They can be used to generate many different equally valid datasets with the required number of features from a single one.

We will study how these advantages could be used to increase the accuracy of some models. In particular, we expect they will be useful in two different ways. First, some methods could achieve a higher accuracy if they were trained with data in a Kernel feature space instead of the general one. Second, generating many datasets could make it possible to increase the accuracy of some models by training an ensemble.

Although both approaches are expected to increase the accuracy of some models, they will also increase a little bit the training time. Performing a random mapping has a linear cost with the number of instances, and training an ensemble clearly multiplies the amount of work to be done. Thus, the benefit of the methods presented will depend on the circumstances of each problem. It should be noted that both increases in time are linear with the number of instances, so they can scale well with very large datasets. This is in contrast to training SVM with non-linear kernels, whose cost is cubic with the number of instances.

For this project we have used 9 different datasets to check the hypothesis. 7 of them are quite small (5000 instances approx.) and the other 2 have a much bigger size (70000 instances). For this reason, the first ones will be useful to compare the accuracies obtained, but will show a very big increase in the training time. In contrast, the bigger datasets will show the real strength of these methods.

We propose two ways of using the random mappings. The first one is very straightforward: simply using a random mapping of the data to train a single model, and use the same mapping in the prediction step with the instances in the input. The second one involves mixing the mapping with some ensemble algorithm.

There are many ways to mix these two methods. The ones that we have studied are based on the Random Forest and are the most straightforward, but other ways could be studied in a future work.

The first thing to decide is where to place the random mapping in the whole process. We have taken two different approaches: we can think of an ensemble as a Black Box, where we can only affect the inputs and the outputs, or we can differentiate the different parts it contain, like in a White Box.

Then, there's what kind of ensemble to use. We've chosen to use the Bagging technique, which is the one that Random Forest uses, but we've also defined a modified version which doesn't perform the Bootstrap. This is because we expected that using Bootstrap together with a random mapping would produce too much randomness in the data, affecting negatively the overall accuracy. We've called the first method a "Bag", and the second method an "Ensemble".

From the combination of these approaches we have defined four different ways of mixing, and we have called them "Black Bag", "Black Ensemble", "White Bag" and "White ensemble". (See 3.1):

Black Bag First a single random mapping is generated and then it is used to train a Bagging, which performs a Bootstrap with the data.

Black Ensemble A single random mapping is generated and then exactly the same data is used to train an ensemble of models.

White Bag Many different random mappings are generated and then a Bootstrap resamples each of the mappings to each of the estimators.

White Ensemble Many different random mappings are generated and then each estimator is trained with one of them.

For this project we have used Random Fourier Features and the Nyström method to approximate the Radial Basis Function (RBF) Kernel with tree well-known Machine Learning algorithms: Logistic Regression, Support Vector Machine and Decision Tree. We study how we can use them to increase the accuracy that we can achieve with them and the computational costs.

3.2 Hyper-parameters

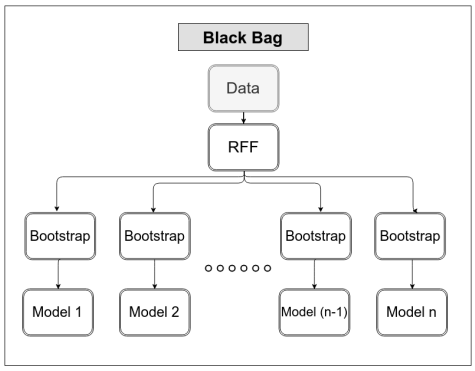
With the models defined in this project there are many hyper-parameters to tune the models. These are the hyper-parameters that have been used in the experiments:

Number of features extracted from the kernel The higher this value is, the better the approximation of the kernel function. We have fixed a value of 500, which is enough given that there is an exponentially fast convergence in the number of features between the approximation and the real kernel[23].

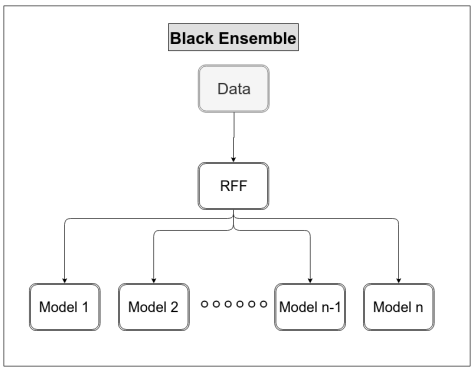
Amount of estimators Having a large number of estimator doesn't affect negatively the accuracy obtained, but increases the computation time, so the ideal number depends on the computational resources available. For this project we have picked 50 estimators for each Bag/Ensemble.

Gamma parameter of the RBF Kernel A higher value will generate a higher overfit. There is a fast method to find a suitable value for this parameter, explained in [9] . It is $\frac{1}{2\sigma^2}$, where σ^2 is the mean of the 0.1 and 0.9 quantile of $\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2$ of each pair of instances i and j. We have chosen this estimation.

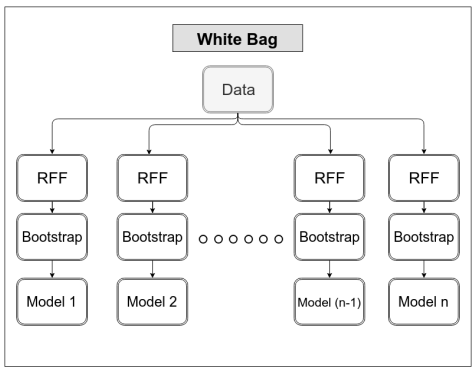
Parameters of the simple models Decision Tree use *min_impurity_decrease* to tune the overfit, and SVM use a penalty C to do the same. When we train these models without any ensemble, we use Cross-Validation to find a suitable value. When we train an ensemble of these models, as we want them to overfit we set *min_impurity_decrease* to 0 and C to 1000, which is enough to achieve it.



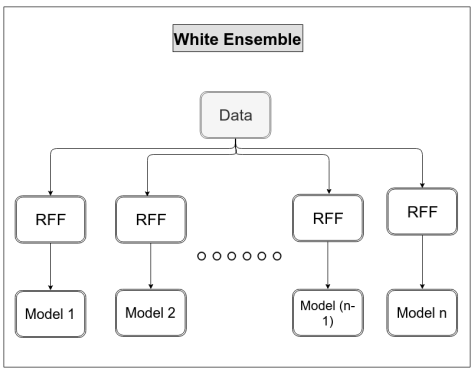
Black Bag. Each model recieve data from the same Random Mapping, but they take a different resampling with with replacement.



Black Ensemble. Each model recieve exactly the same data, from just one Random Mapping. Only useful for randomized models.



White Bag. Each model recieves data from a different Random Mapping, which has then had a resampling with replacement.



White Ensemble. Each model recieves data from a different Random Mapping.

FIGURE 3.1:
Four different ways to mix Random Fourier Features with ensemble methods.

3.3 Hypothesis

We had proposed these four hypothesis:

1. **It is possible to achieve an accuracy close to using the RBF Kernel but with a lower cost**

   When the number of instances available is too big it is not possible to use an SVM with the RBF kernel, because the cost is cubic with the number of instances, and the optimization problem is too complex. A linear Kernel needs less time, but it may not be suitable for some problems, since data may not be easy to separate.

   If we could first map the data to the new feature space, we could then feed a Linear SVM with it and have the same accuracy with less costs. But this can't be done with the RBF kernel, since the new feature space has infinite dimensions. However, with the use of RFF and Nyström, we can get an approximation of the feature space of the RBF. Using them with a Linear SVM could increase the accuracy on some datasets at almost the same cost.

2. **It could make sense to train ensembles of SVM and Logistic Regression algorithms**

Since these models are very stable, having an ensemble of them is useless: all of them will always predict the same answer. There are some methods to randomize a little bit the data, such as Bootstrap, but with these models it is not enough.

Since RFF and Nyström generate a random mapping of the data, we can achieve a higher level of randomization of the data, while still being a good representation of the real data. Random Mapping can allow us to build ensembles with these two models, increasing the overall accuracy at the expense of some computation time.

3. **Bootstrap together with a Random Mapping may be too much randomization**

   With a simple mix of Bagging with RFF there are two different sources of randomness. For the one hand, Bootstrap generates a random sample of the data with replacement, and on the other hand, RFF and Nyström perform a Random Mapping of the data to a different feature space.

   It is possible that for some models, this is too much randomization of the data, and it could have a bad effect on the learning process.

4. **Decision Tree does not benefit from RFF and Nyström as much as Logistic Regression and SVM do**

   Kernels were originally used on Support Vector Machines because they were a fast way to implicitly compute the inner product of two vectors in a feature space where data was separable by an hyper-plane. They were useful because SVM just needed the inner products of their input to work.

   RFF and Nyström are ways to explicitly compute an approximation of that mapping, which doesn't necessarily fits the requirements of Decision Tree, which has nothing to do with the inner products. That's the reason why Decision Tree may not benefit so much of these Random Mappings.

3.3.1 Experiments Proposal

There are two factors we need to check in order to accept of refute the hypothesis: we want to know if these methods increase the accuracy of the models (and if so, how much) and also at what costs does it comes, the increase of training time. The 7 first datasets (Covertype, Digits, Fall Detection, Pen Digits, Satellite, Segment and Vowel) are suitable to measure the increase of the accuracy, but not to measure the time, since the overhead caused by these techniques is too much compared with training with a problem as small as that. The other two datasets (MNIST and Fashion MNIST) have a larger number of instances, and are expected to reflect better the benefits of using these methods. However, some of the models proposed in the experiments are very expensive to train on very large datasets, and it was not possible to run the whole training with them. Some simplifications have been made to extrapolate the real training time.

To check the hypothesis previously suggested we have proposed a set of experiments:

Hypothesis 1

In this hypothesis we assumed that we could train an SVM with results comparable to the ones using the RBF Kernel, but with much less training time. To check that

we have defined the experiment 1.1, where we compare four different models: an SVM without a kernel, an SVM with the RBF kernel and two SVM without kernel but using the random mappings, RFF and Nyström. To consider that results support the hypothesis we need to see that the models using RFF and Nyström significantly increase the accuracy compared to the single SVM.

For the method to be useful it would be needed for the training time to be much less using the random mappings compared to the RBF-SVM on large datasets. To check that we will need to focus on MNIST and Fashion-MNIST, since they are the only large datasets.

Theoretically, for RBF-SVM a cross-validation should have been performed to find a suitable value for C, but the training time needed to do that for the large datasets is too much. To avoid that, we have used the same value found for the SVM with RFF in MNIST and Fashion-MNIST, and have multiplied the resulting time by 5 in order to approximately match the real required time with cross-validation.

Results of this experiment can be seen in Appendix A

Hypothesis 2

For this hypothesis it was asserted that training an ensemble of SVM and Logistic Regression using a random mapping could increase significantly the accuracy obtained. Nevertheless, the training time is expected to be significantly greater since training an ensemble requires a lot more steps.

Experiments 2.1 through 2.4 were defined to check that. The first two are for Logistic Regression and the others for SVM. 2.1 and 2.3 will compare a single model alone against a single model using a random mapping. 2.2 and 2.4 will compare a single model against the model with Black Bag, White Bag and White Ensemble. The Black Ensemble model is not tested because these estimators are not randomized, and each one would predict the same answer.

For the hypothesis to be supported we need to see that we can achieve an increase in the accuracy using any of the ensemble methods proposed. Then we will need to study the increase in the training time, and consider if it is worth it.

Results of these experiments can be seen in Appendices B, C, D and E.

Hypothesis 3

With this hypothesis we wanted to know if using Bootstrap together with a random mapping will produce worse results than just using the random mapping. To check that we will compare the "Bags" against the "Ensembles". Experiments 3.1 and 3.3 will compare "White Bag" and "White ensemble" for Logistic Regression and SVM respectively, and 3.2 and 3.4 will compare "Black Bag" and "Black Ensemble". To make it fairer only one estimator has been trained for the "Black models" , since "Black Ensemble" doesn't benefit from using more.

To support this hypothesis we would need to see a meaningful differences in the accuracy of these models.

Results of these experiments can be seen in Appendices F, G, H and I.

Hypothesis 4

In this hypothesis we asserted that Decision Tree, which doesn't work with the inner product of the inputs, would not experiment a meaningful increase in the accuracy

TABLE 3.1: Information on the datasets used in this project

Dataset	N. Instances	N. Features	N. Classes
Covertype[4]	4900	12	7
Digits[2]	5000	64	10
Fall Detection[32]	5000	6	6
MNIST[19]	70000	784	10
Pen Digits[1]	5000	16	10
Satellite[26]	5000	36	6
Segment[28]	2310	19	7
Vowel[12]	990	10	11
Fashion MNIST[30]	70000	784	10

with the usage of a random mapping approximating the RBF feature space. Decision Tree can normally increase their accuracy by using a normal Bagging. That algorithm is called Random Forest. For this reason the ensemble methods using a random mapping should be compared against the Random Forest instead of a single Decision Tree.

Experiment 4.1 compares a single Decision Tree with one that uses a random mapping, and 4.2 compares the Random Forest with "Black Bag", "Black Ensemble", "White Bag" and "White Ensemble".

To support this hypothesis we should see that, compared with the normal algorithm, using a random mapping doesn't outperform too much the model.

Results of these experiments can be seen in Appendices J and K

3.4 Datasets

This project is focused on studying the effects of Random Fourier Features and Nyström on classification problems, but there is nothing that prevents their usage for regression. It is proposed as a future work to perform the same experiments on regression problems.

The RBF kernel is just designed for numerical values of the features. Thus, we wanted all the variables on the datasets to be numerical. When that was not possible, categorical variables were transformed to integers, as in these circumstances that seemed preferable to using One Hot Encoding.

Table 3.1 shows some information of the datasets used in this project. Most of them have few instances compared to some large-scale problems, but we expect the accuracy results will to be representative for bigger problems. Regarding the computation time, smaller datasets may not be the most appropriated to study the performance; for that it will be better to focus on MNIST and Fashion MNIST.

Some preprocessing was done with the dataset Covertype, since the target classes were very unbalanced. As working with unbalanced classes is out of the scope of this project, we took a random subset of instances of each of the classes and discarded the rest.

The features in all datasets have been normalised to have a mean of 0 and a variance of 1. This is a common practice to normalise the contribution of the features of the problem, and also helps to reduce the time of some optimization problems.

Here follows a short description of the datasets used in this project:

Covertype Given some information from some terrain such as the elevation, the slope, the distance to water, etc. the problem is to predict the type designation of the forest cover.

Digits Each instance is an image of 8×8 pixels of a hand-written digit from 0 to 9. The problem is to predict the digit that is represented.

Fall Detection The problem consists on predicting in which position a patient is (Standing, Walking, Falling, etc.) based on data collected with a wearable device, such as the sugar level, the blood pressure, etc.

MNIST Images of 28×28 pixels in a greyscale which represent hand-written digits from 0 to 9.

Pen Digits In this dataset each pair of points of an instance are coordinates. This way, the 16 features represent 8 points in a plane through which a pen crossed to write a digit. The problem is to predict what digits was written.

Satellite From color information of the pixels in a photo, the problem is to find out the type of terrain of the photo.

Segment The problem is to differentiate 7 types of outdoor images based on information on the pixels such as the contrast of adjacent pixels or the average of the intensity.

Vowel The problem is to distinguish what was the vowel pronounced by an English native speaker (there are 11 vowels in English) through some observations of a recording.

Fashion MNIST Very similar to MNIST, but instead of being images of digits, they are of different types of clothes, like trousers, shirt, sneakers, etc.

Chapter 4

Experimental Results

In this chapter we will study the results obtained from the experiments and discuss the hypothesis that were considered for this project. All the results can be looked up in the appendices.

Hypothesis 1

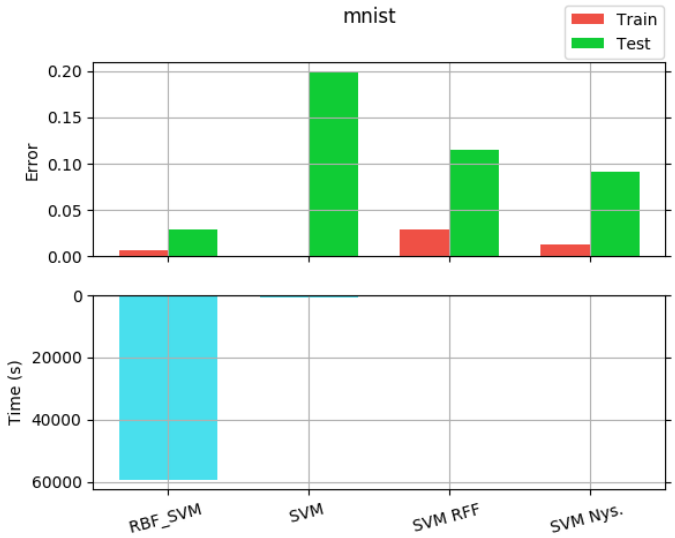


FIGURE 4.1: Exp. 1.1 with MNIST. SVM is outperformed

Hypothesis 1 claimed that Linear SVMs could achieve the accuracy of an RBF-SVM but with much less training time. To check that we ran experiment 1.1, whose results can be seen in A.

We can see that the models using Random Fourier Features or Nyström (shorted as RFF and Nys.) reduce the error compared to a single Support Vector Machine in the cases were using the real RBF kernel increased the accuracy. In fact, the error incurred is always very close to the one of RBF-SVM. See Figure 4.1.

Regarding the training time, we observe that in most of the datasets it was needed more time to train a single SVM using a random mapping than to train an SVM with the RBF kernel. The only situations were the random mapping approach saved us a lot of time is with datasets MNIST and Fashion MNIST.

We suggest that the reason why that happens is that the overhead of using the random mapping is too big for small datasets compared to using the RBF kernel, but becomes less relevant with bigger datasets, since the cost of the random mapping is

linear with the number of instances while the cost of the optimising with the RBF kernel is cubic.

We conclude that the results obtained provide evidence to assert that the hypothesis 1 is true for large datasets.

Hypothesis 2

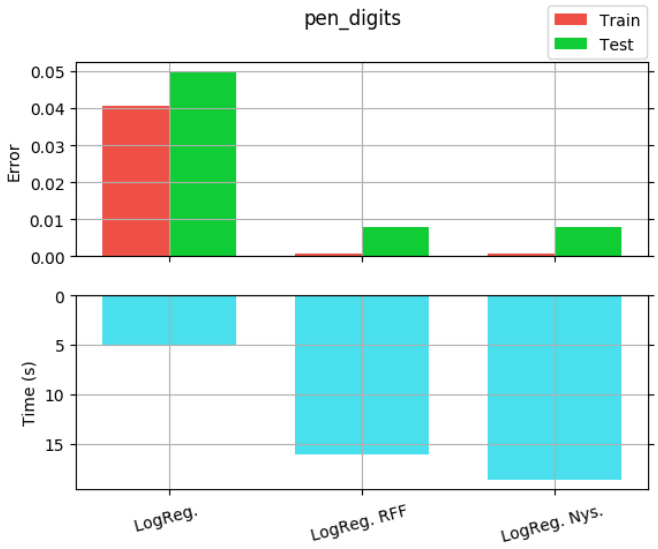


FIGURE 4.2: Exp 2.1 with Pen Digits. Error is decreased by 4% approx.

Hypothesis 2 suggested that using a random mapping to train an ensemble of Logistic Regression models and Support Vector Machines could improve the accuracy of a single model. To give support to this hypothesis we proposed the experiments 2.1, 2.2, 2.3 and 2.4. The results of these experiments can be looked up in B, C, D and E, respectively.

First, we wanted to see what was the effect of using the random mapping with a single model, and then with an ensemble.

Experiments 2.1 and 2.2 were defined for Logistic Regression. Except for Fall Detection and Segment, we can see an increase of the accuracy when using the random mapping on a single estimator. With most of the datasets the increase is between 2 and 4 %, and with Vowel it is about 26 % of increase. See 4.2.

When used with an ensemble, all of the Box Models show a very similar improvement to a single estimator. To see better the differences, they have been put together in a separate chart. See 4.3

2.3 and 2.4 are the equivalent experiments for SVM.

With a single model, we can see there is a meaningful improve in the accuracy with all datasets except for Fall Detection. The improvements are from 2 % (with Digits and Segment) to 35 % with Vowel.

With an Ensemble of SVMs the increases are more of less the same as with a single estimator. The differences can be seen in a separate chart.

Although we can see there are some benefits of using an ensemble of these models, the training time is a lot higher than with a single one. Given that the difference between the accuracy of a single model using a random mapping and an ensemble

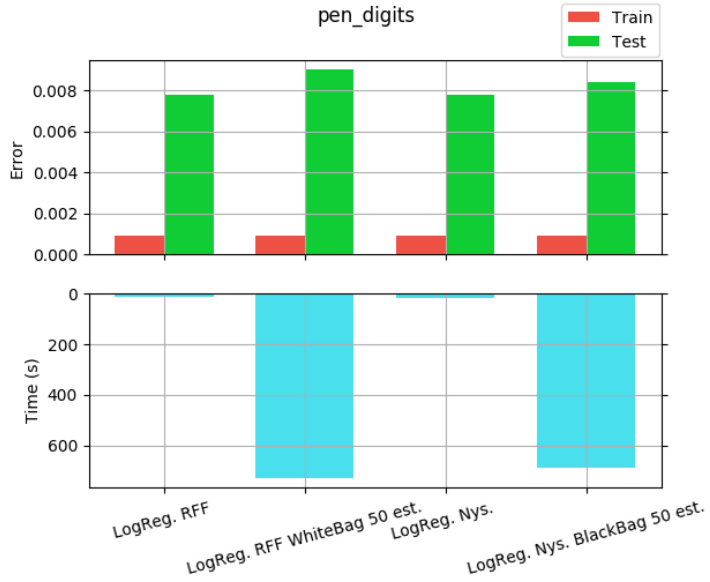


FIGURE 4.3: Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Pen Digits.

of them is so tiny, there are not evidences that training an ensemble of Logistic Regression or SVM with random mapping is worth it. However, in situations were that tiny difference is very important it could make sense to use it, specially if one can afford it.

Hypothesis 3

For this hypothesis we expected that the "Ensemble" models would get better results than the "Bags" given that Bootstrap with random mapping could cause too much randomization. To check that we proposed experiments 3.1, 3.2, 3.3 and 3.4. The results can be seen in F, G, H and I.

In 3.1 we compare the White Box Models with Logistic Regression. It can be seen that the results are very similar. We can just see a very little improve in using an Ensemble in Digits, Pen Digits and Vowel (with Nyström).

With the Black Models we can see an improve in more datasets. Digits, MNIST, Pen Digits, Segment and Vowel present better results with the Ensemble models.

Regarding the SVMs (in 3.3 and 3.4), the difference is only seen with Vowel, either in the White and Black Models. See 4.4.

Based on the results, we can say that whether to perform a bootstrap or not together with a random mapping doesn't make much difference. However, seen that Bootstrap does not benefit the models, it may be better to avoid using it.

Hypothesis 4

Hypothesis 4 claimed that Decision Tree could not benefit of using Random Fourier Features or Nyström. To check that, we proposed experiments 4.1 and 4.2. The first one, which can be seen in J was to compare a single tree, and the second one, in K,

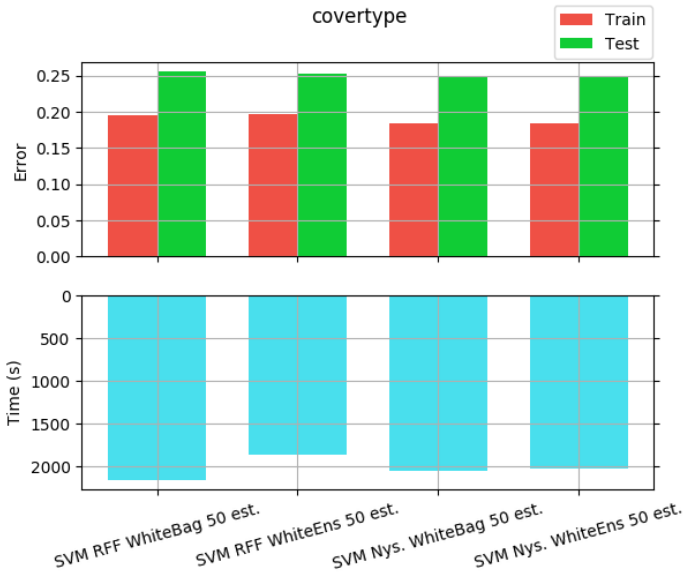


FIGURE 4.4: Exp. 3.3 with Covertype. White Box Models with Support Vector Machine. There is not much difference between Box Models

an ensemble. For 4.2 a Random Forest was used instead of a Decision Tree since they benefit with Bagging.

With a single model, we can see no improvement of using a random mapping for any of the datasets. Besides, the training time is increased. See 4.5.

With ensembles, however, there is a very little improvement with Pen Digits and Vowel. But as for the majority of the datasets the error is increased, we can say that using RFF or Nyström with Decision Tree is in general a bad choice. See 4.6

We observe there are big differences in the training time of a Random Forest and the Ensembles of Decision Tree with random mapping. We guess that the reason for that is that we used a full-fledged implementation of Random Forest, which may already by highly optmised, whilst for the Decision Tree we stacked together many different methods and may have a naive implementation.

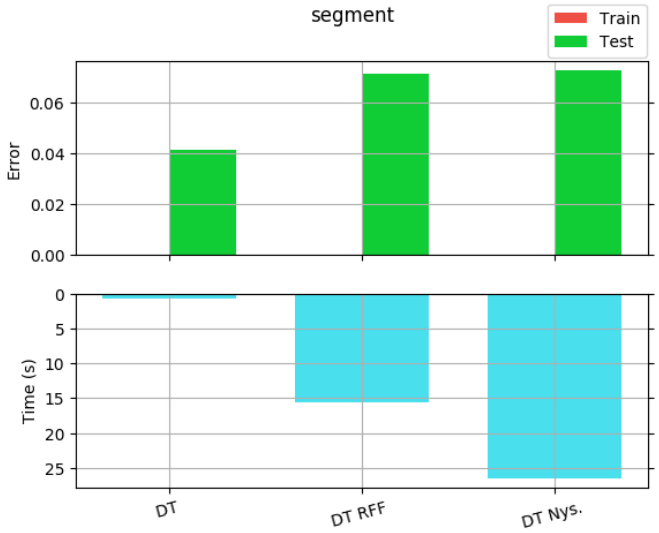


FIGURE 4.5: Exp. 4.1 with Segment. Random Samplers increase the error on Decision Tree.

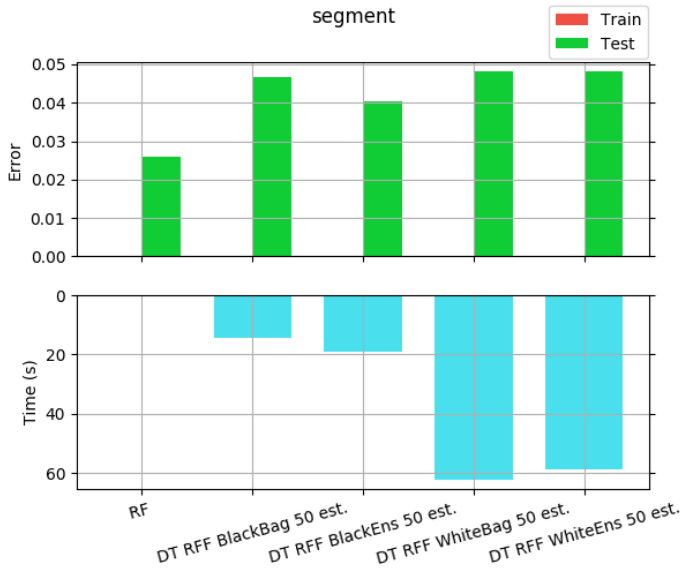


FIGURE 4.6: Exp. 4.2 with Segment. Random Forest outperforms Ensembles of Decision Tree with RFF.

Chapter 5

Conclusion and Future Directions

We have studied many ways of using Random Fourier Features and Nyström method to improve the trade-off between accuracy and training time of some Machine Learning Methods.

Regarding Support Vector Machines, we've seen that a single SVM using random features can match the accuracy of an SVM using the RBF Kernel, but it is only worth the time for datasets with a lot of instances. In some situations an ensemble of SVMs with random features can increase a little bit the accuracy, but the additional needed training time is too much, and may not be suitable for most of the situations.

We've seen that Logistic Regression can also benefit from Random Fourier Features and Nyström, achieving better accuracy than a single one. Like with SVMs, Ensembles of Logistic Regression barely increase the accuracy and is much more expensive.

We've also verified that a single Decision Tree doesn't benefit from using random features. For some problems an ensemble of Decision Trees using random features outperformed the Random Forest a little bit, but in others the accuracy was worse, so results are not conclusive.

We've checked that using Bootstrap together with random features is not too much randomness for the models studied, and in fact maybe it should even be increased to benefit from them.

Finally, we've not observed a clear difference in the performance of using Random Fourier Features compared to Nyström, so it seems that choosing one over the other doesn't make a real difference.

Here are some ideas to extend this work for future studies:

- This study was carried out only for classification problems. It may be interesting to run the same experiments regression problems.
- Using Random Fourier Features or Nyström to approximate other kernels than the RBF.
- Test these ideas with other types of ensembles, like Boosting.
- Study or characterise what problems can show a higher accuracy using an ensemble of Decision Trees with random features than using a Random Forest.

Chapter 6

Sustainability Report

6.1 Environmental

6.2 Economic

6.3 Social

6.3.1 Impacto Personal

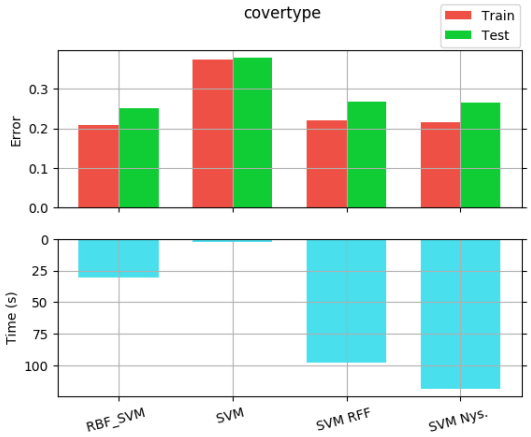
6.3.2 Impacto Social

6.3.3 Riesgos Sociales

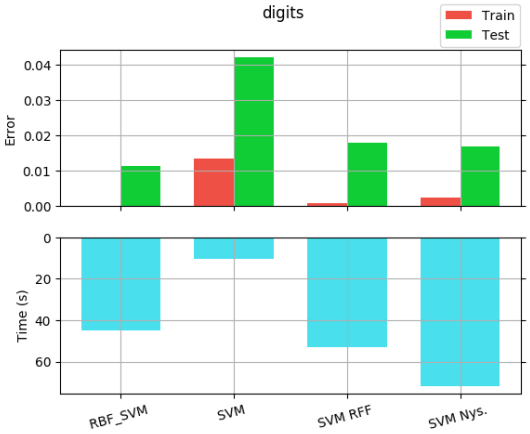
Appendix A

Results of experiment 1.1

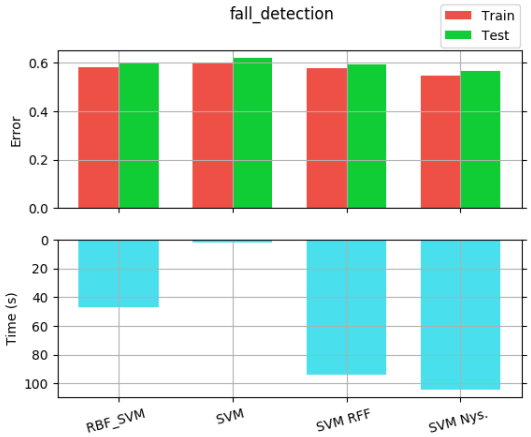
These experiments are discussed here



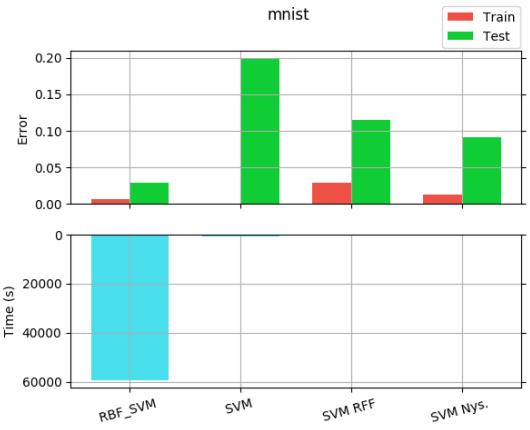
Exp. 1.1 with Covertype. SVM is outperformed



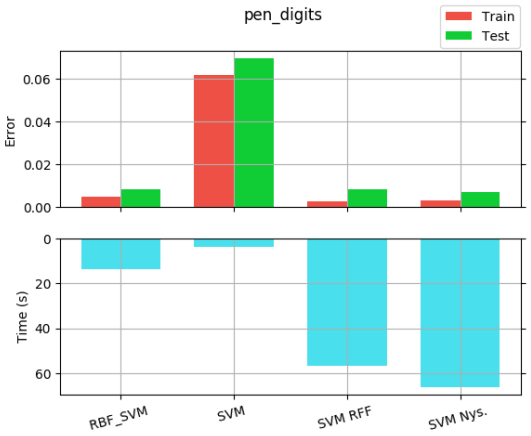
Exp. 1.1 with Digits. SVM is outperformed



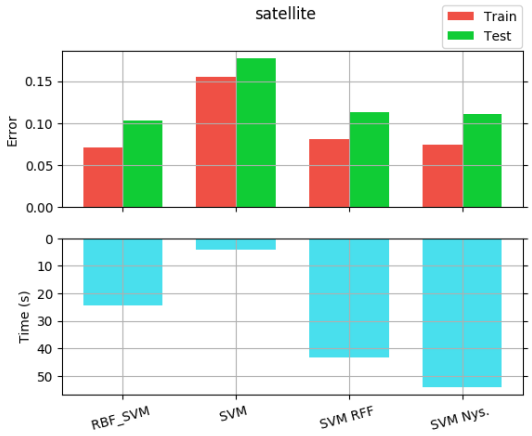
Exp. 1.1 with Fall Detection. RFF or Nyström can't outperform SVM



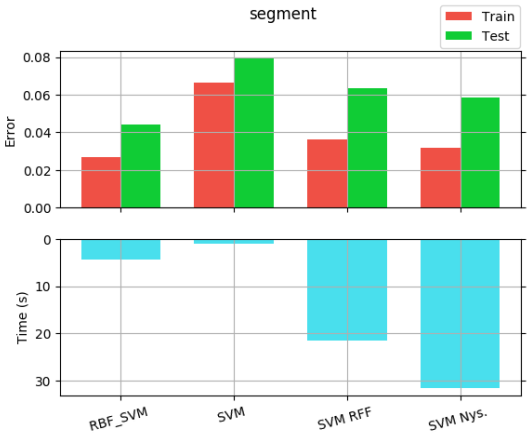
Exp. 1.1 with MNIST. SVM is outperformed



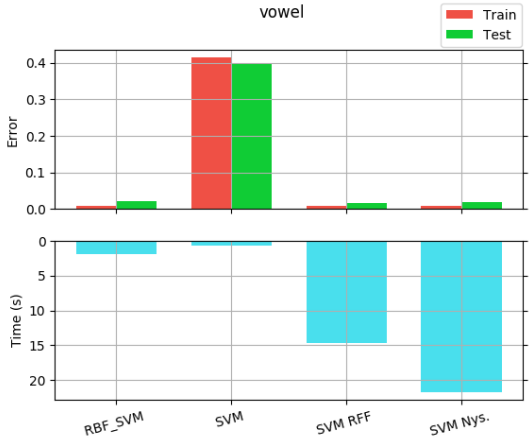
Exp. 1.1 with Pen Digits. SVM is outperformed



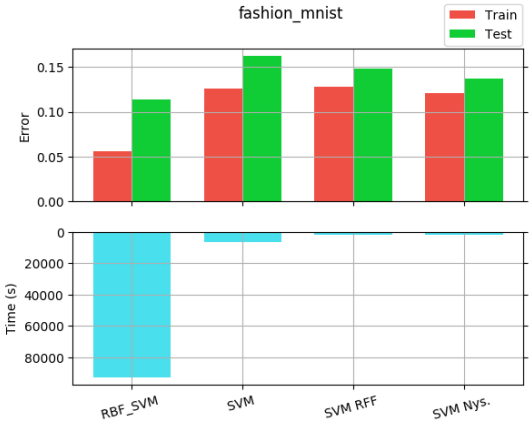
Exp. 1.1 with Satellite. SVM is outperformed



Exp. 1.1 with Segment. SVM is outperformed



Exp. 1.1 with Vowel. SVM is outperformed

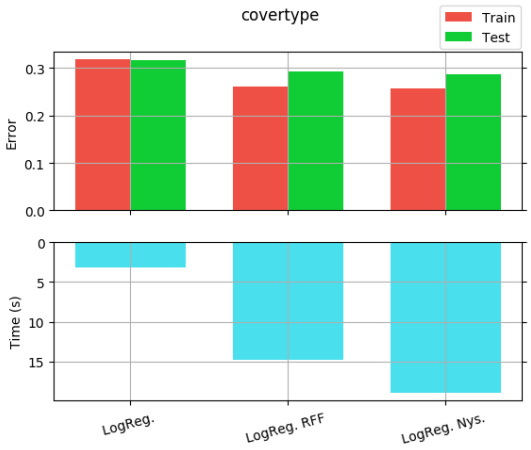


Exp. 1.1 with Fashion MNIST. SVM is outperformed

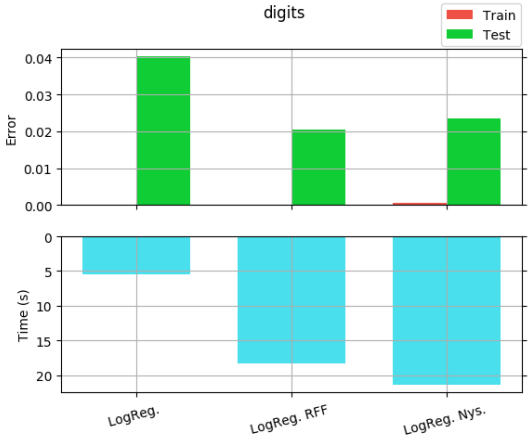
Appendix B

Results of experiment 2.1

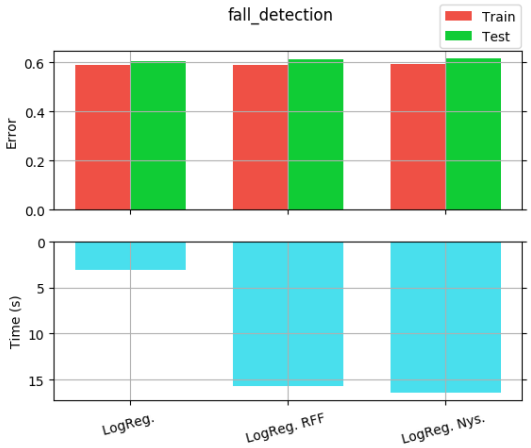
These experiments are discussed here



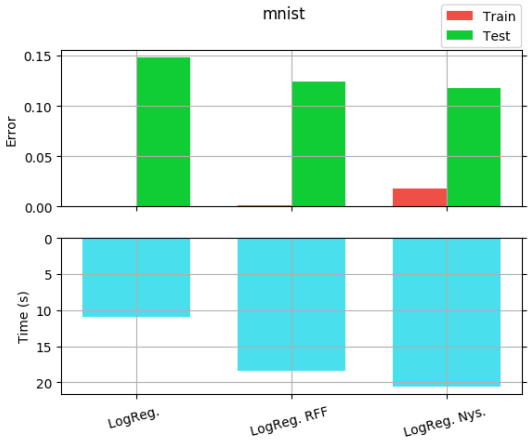
Exp 2.1 with Covertype. Error is decreased by 1% approx.



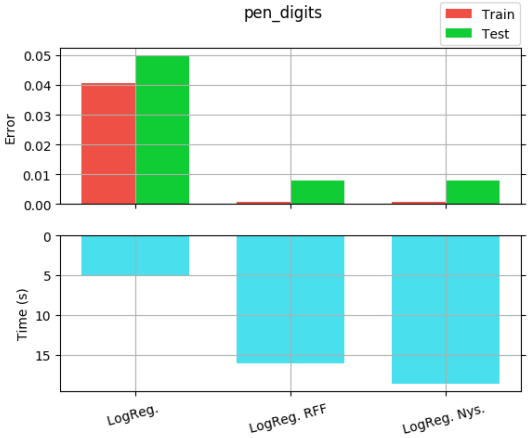
Exp 2.1 with Digits. Error is decreased by 2% approx.



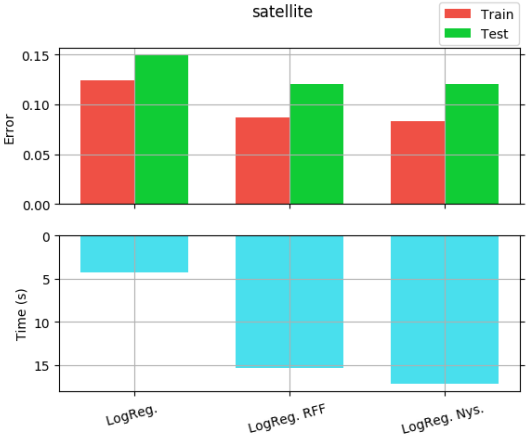
Exp 2.1 with Fall Detection. Error is not decreased.



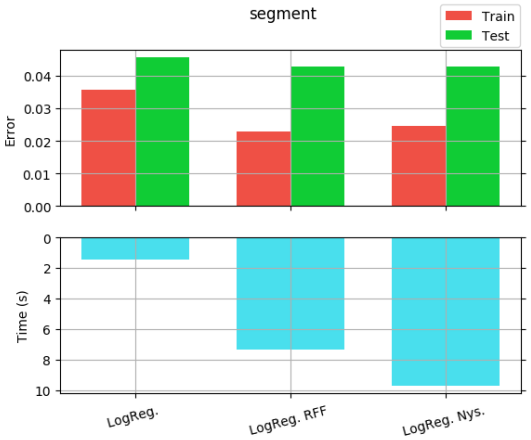
Exp 2.1 with MNIST. Error is decreased by 3% approx.



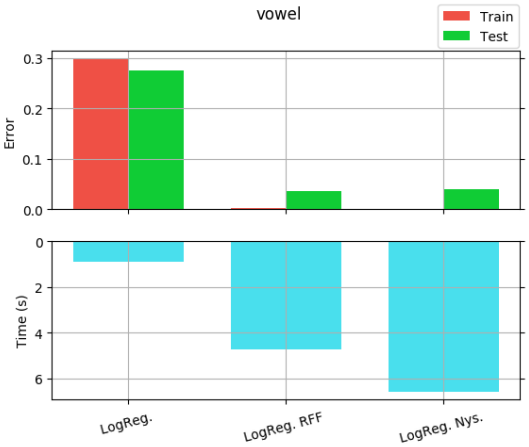
Exp 2.1 with Pen Digits. Error is decreased by 4% approx.



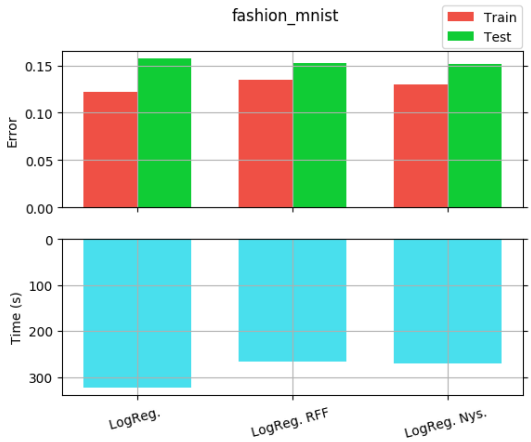
Exp 2.1 with Satellite. Error is decreased by 3% approx.



Exp 2.1 with Segment. Error is not decreased.



Exp 2.1 with Vowel. Error is decreased by 25% approx.

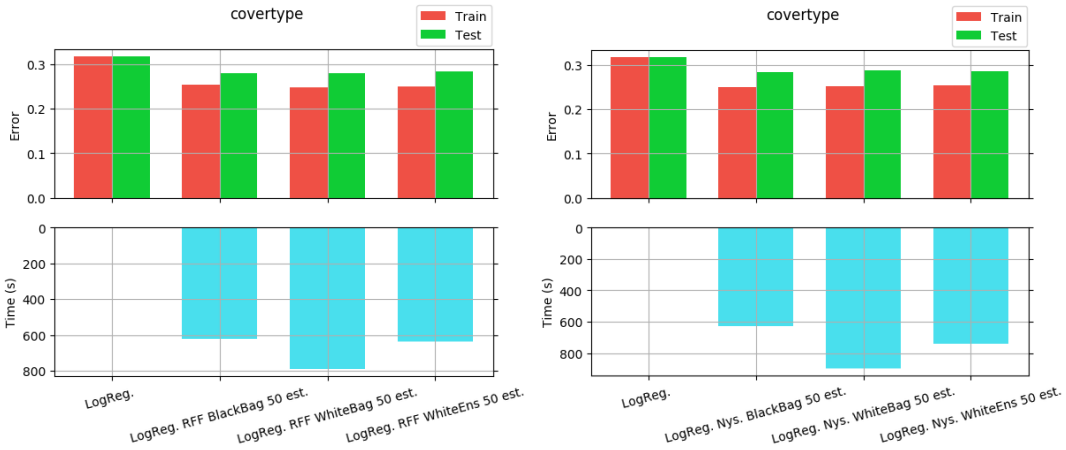


Exp 2.1 with Fashion MNIST. Error is decreased by 1% approx.

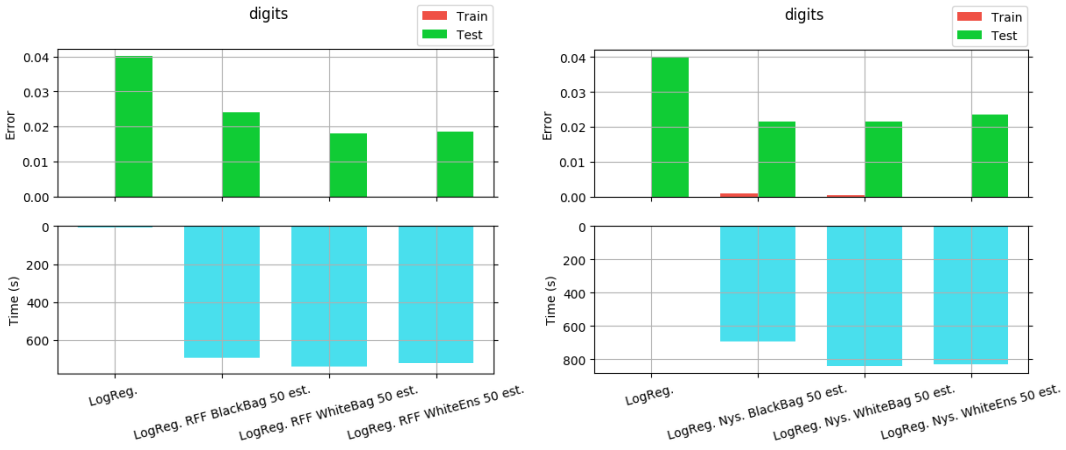
Appendix C

Results of experiment 2.2

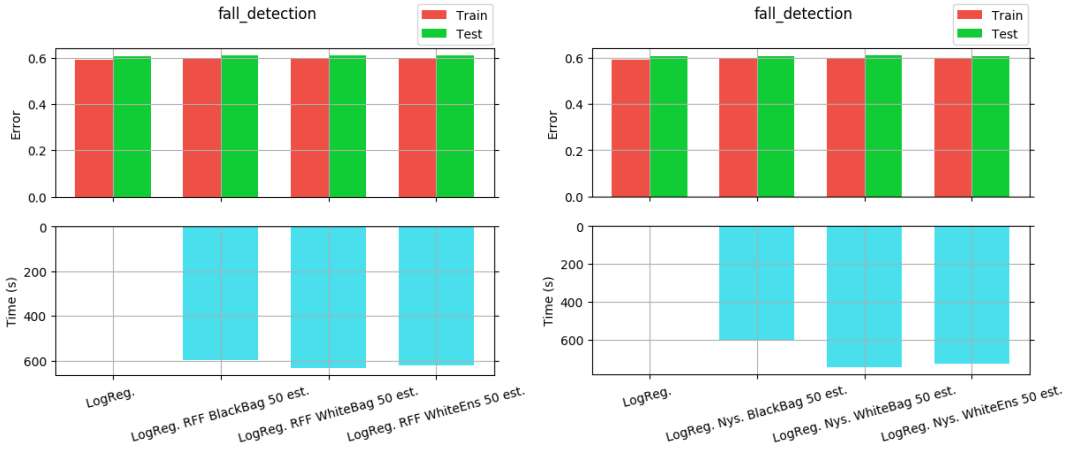
These experiments are discussed here



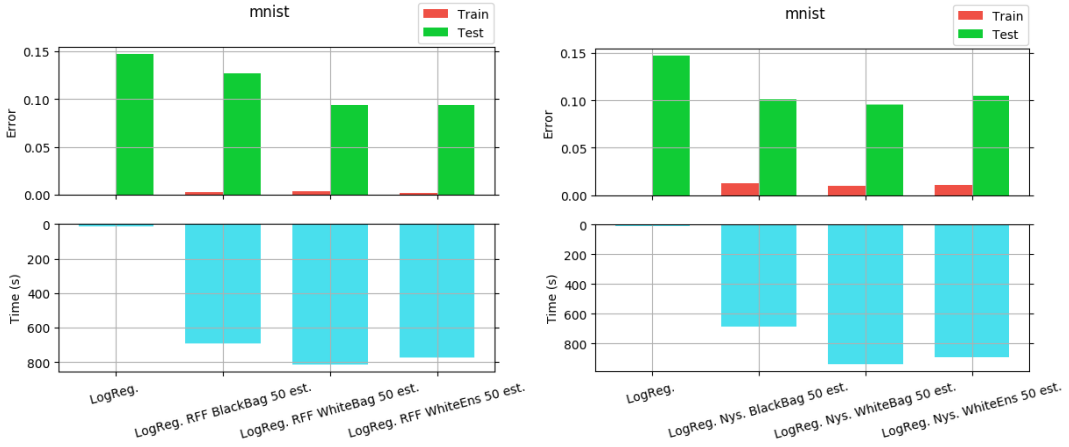
Exp 2.2 with Random Fourier Features (left) and Nyström (right) with Covertype. Error is decreased by 2% approx.



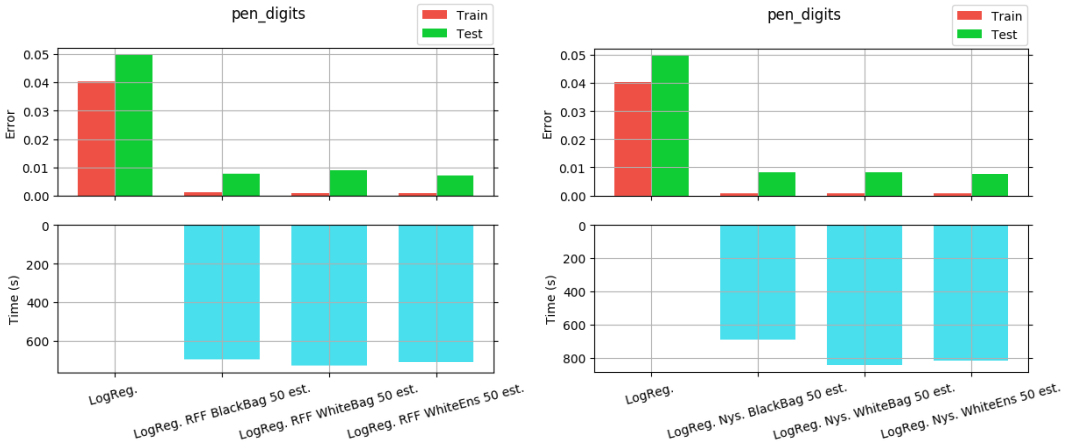
Exp 2.2 with Random Fourier Feautures (left) and Nyström (right) with Digits. Error is decreased by 2% approx.



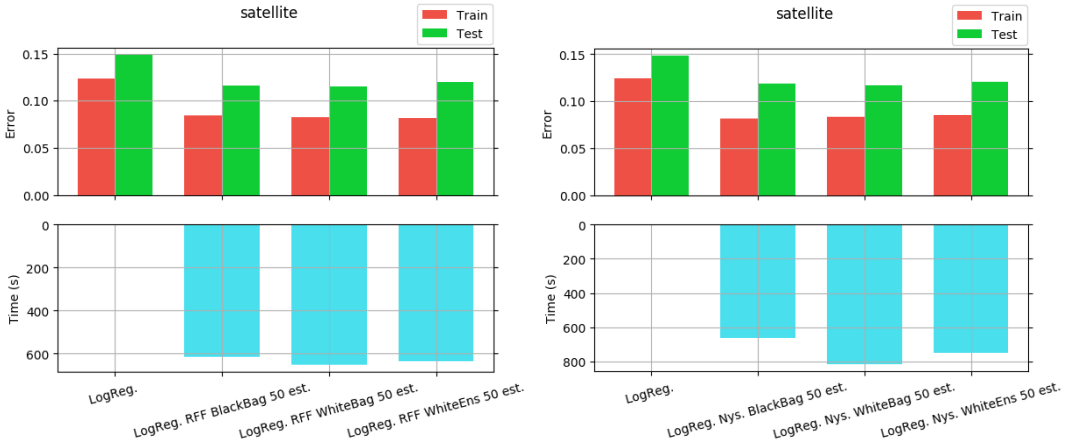
Exp 2.2 with Random Fourier Feautures (left) and Nyström (right) with Fall Detection. Error is not decreased.



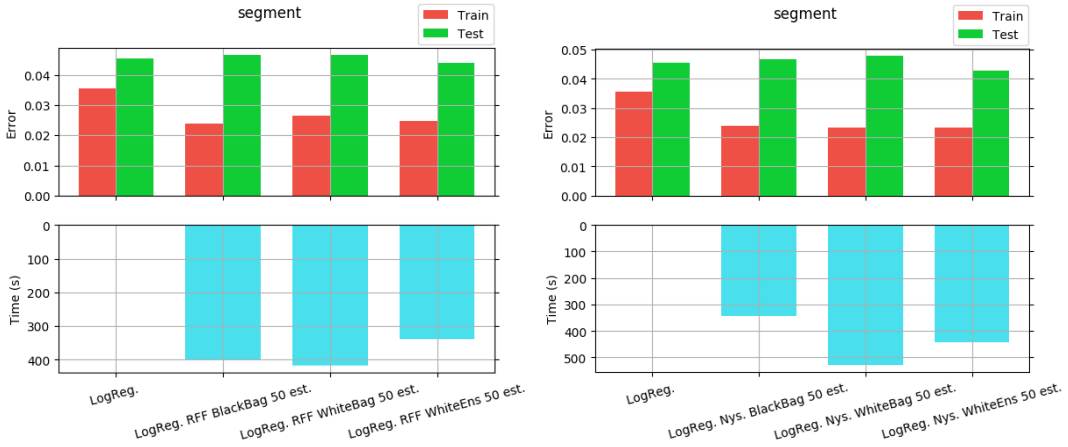
Exp 2.2 with Random Fourier Feautures (left) and Nyström (right) with MNIST. Error is decreased by 5% approx.



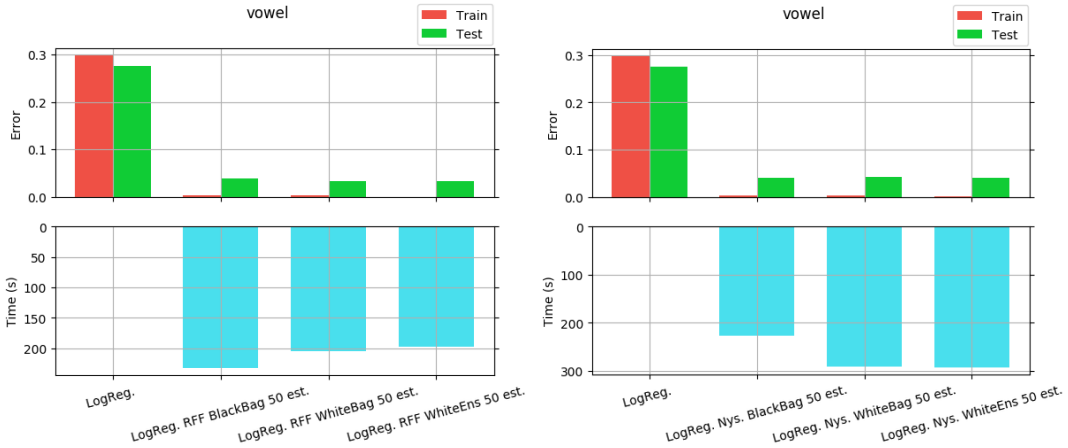
Exp 2.2 with Random Fourier Feautures (left) and Nyström (right) with Pen Digits. Error is decreased by 4% approx.



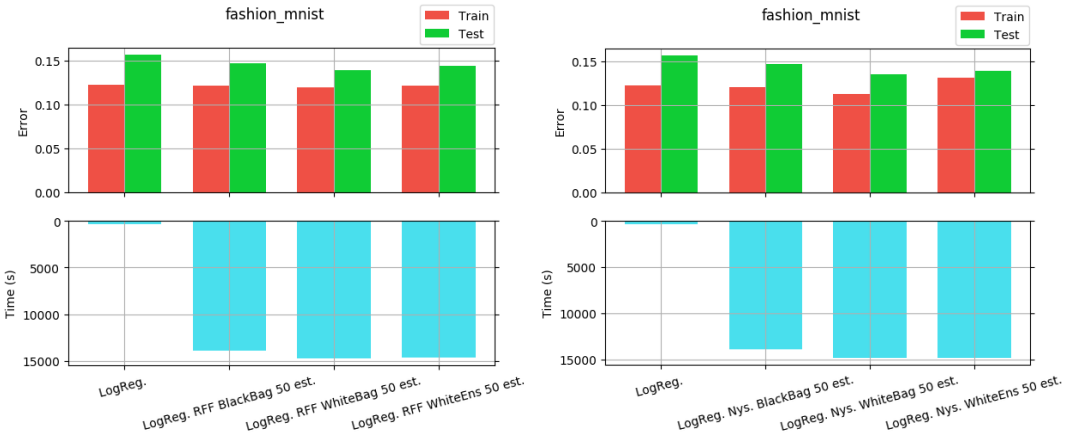
Exp 2.2 with Random Fourier Feautures (left) and Nyström (right) with Satellite. Error is decreased by 3% approx.



Exp 2.2 with Random Fourier Feautures (left) and Nyström (right) with Segment. Error is not decreased.

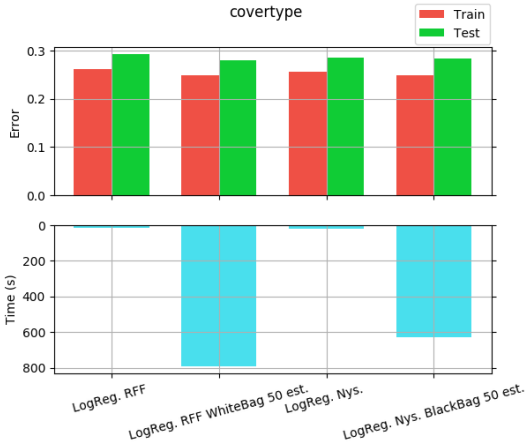


Exp 2.2 with Random Fourier Feautures (left) and Nyström (right) with Vowel. Error is decreased by 25% approx.

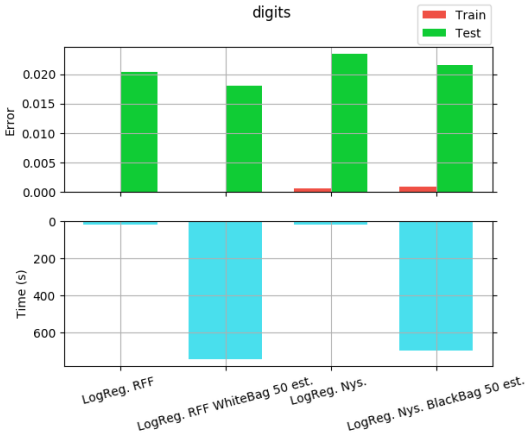


Exp 2.2 with Random Fourier Feautures (left) and Nyström (right) with Fashion MNIST. Error is decreased by 3% approx.

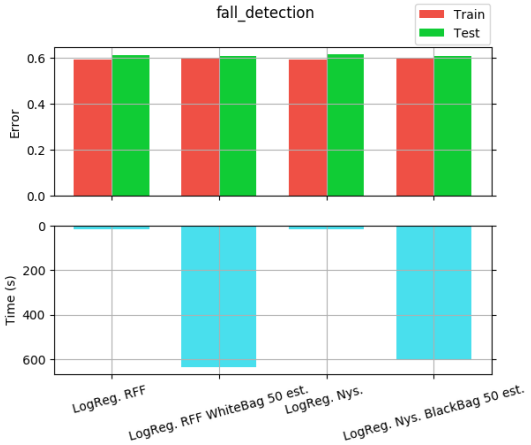
Comparison of Single Model and Ensemble both using a random mapping



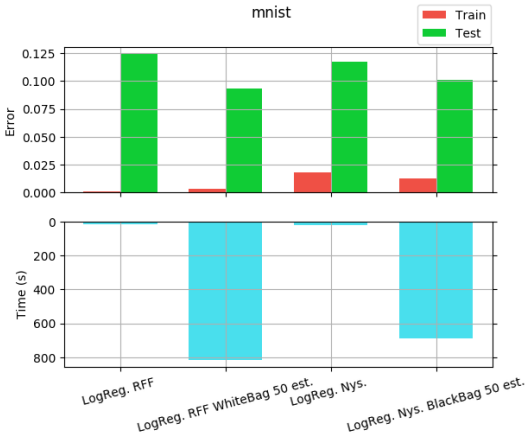
Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Covertype.



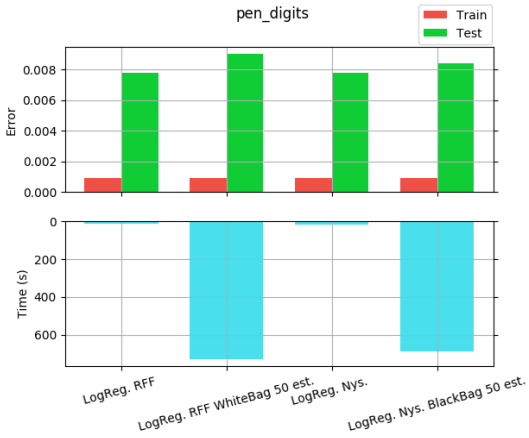
Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Digits.



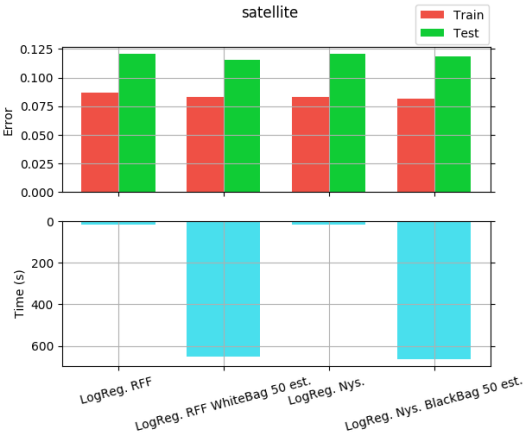
Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Fall Detection.



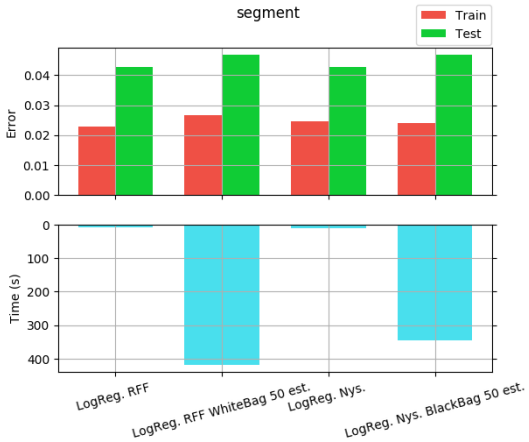
Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with MNIST.



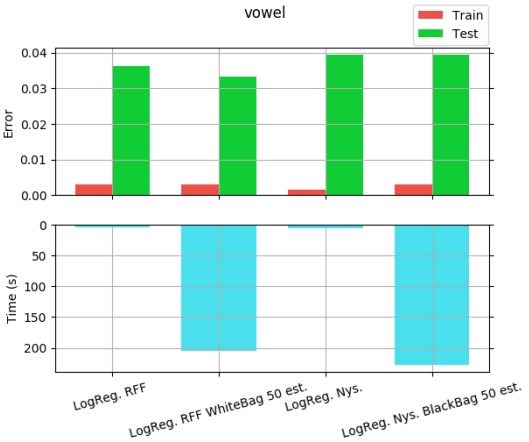
Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Pen Digits.



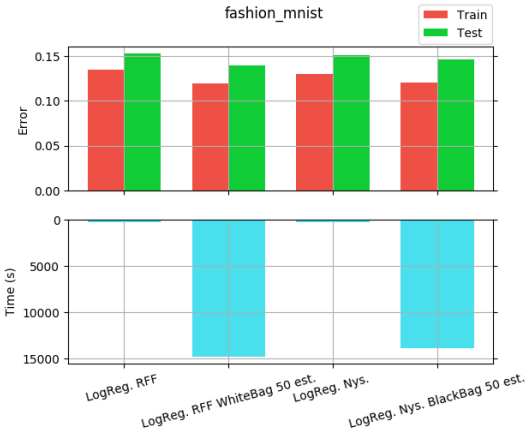
Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Satellite.



Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Segment.



Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Vowel

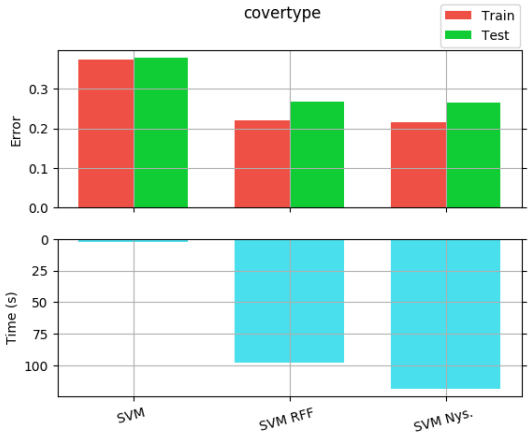


Logistic Regression with random mapping. A single model vs. an Ensemble. There is not big difference with Fashion MNIST.

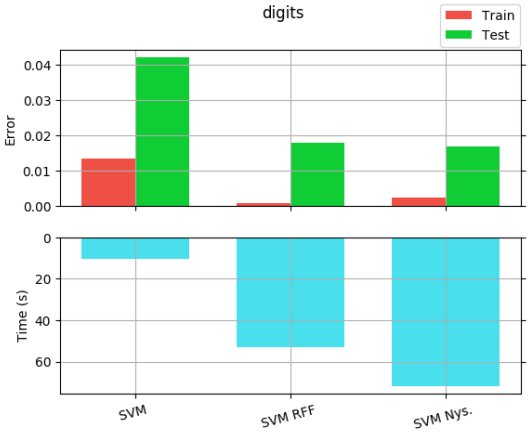
Appendix D

Results of experiment 2.3

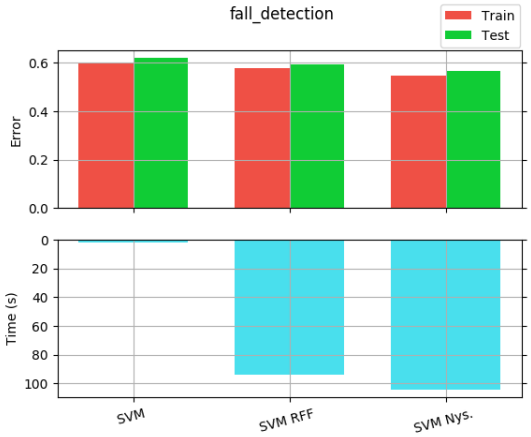
These experiments are discussed here



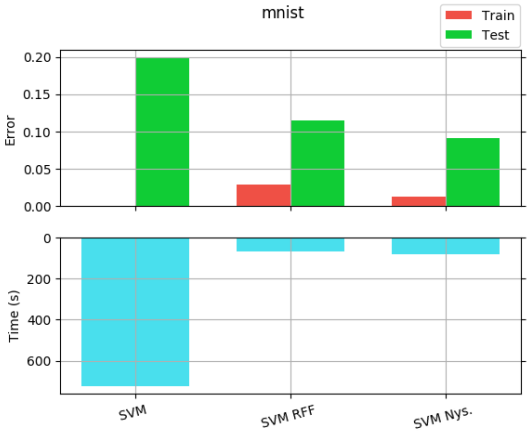
Exp 2.3 with Covertype. Error is decreased by 10% approx.



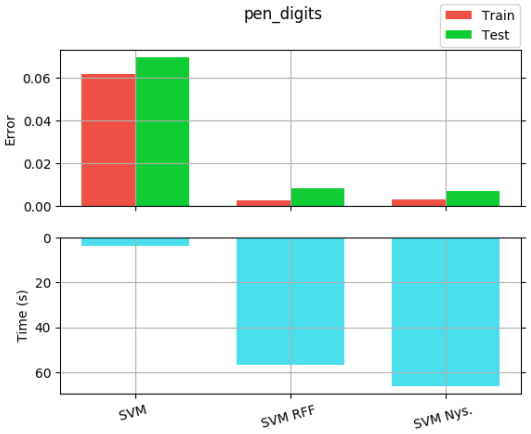
Exp 2.3 with Digits. Error is decreased by 2% approx.



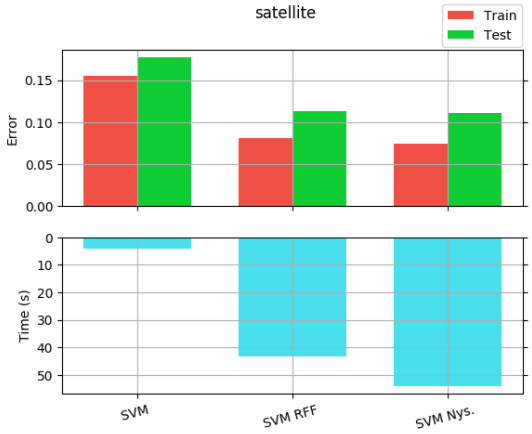
Exp 2.3 with Fall Detection. Error is not decreased.



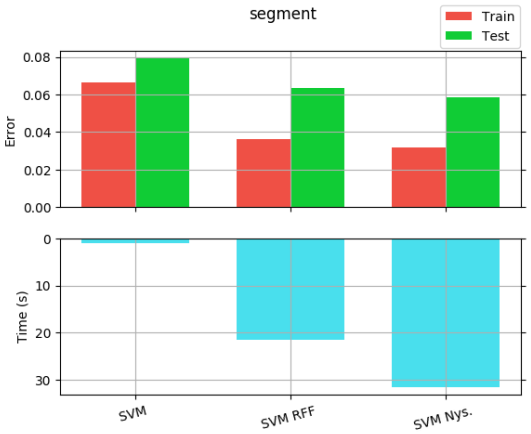
Exp 2.3 with MNIST. Error is decreased by 10% approx.



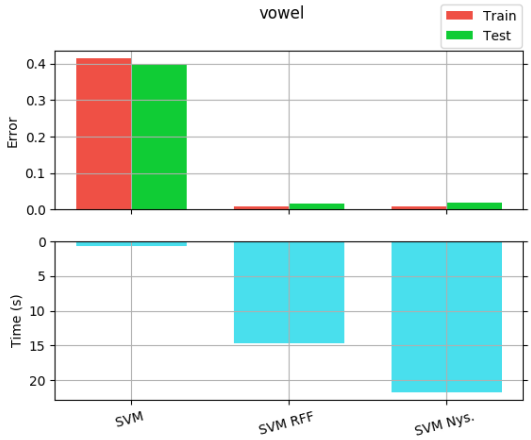
Exp 2.3 with Pen Digits. Error is decreased by 5% approx.



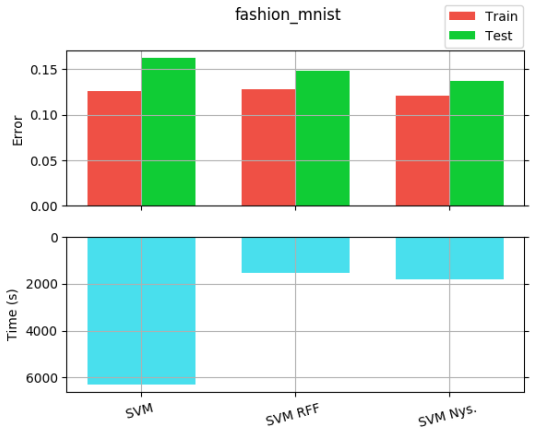
Exp 2.3 with Satellite. Error is decreased by 7% approx.



Exp 2.3 with Segment. Error is decreased by 2% approx.



Exp 2.3 with Vowel. Error is decreased by 35% approx.

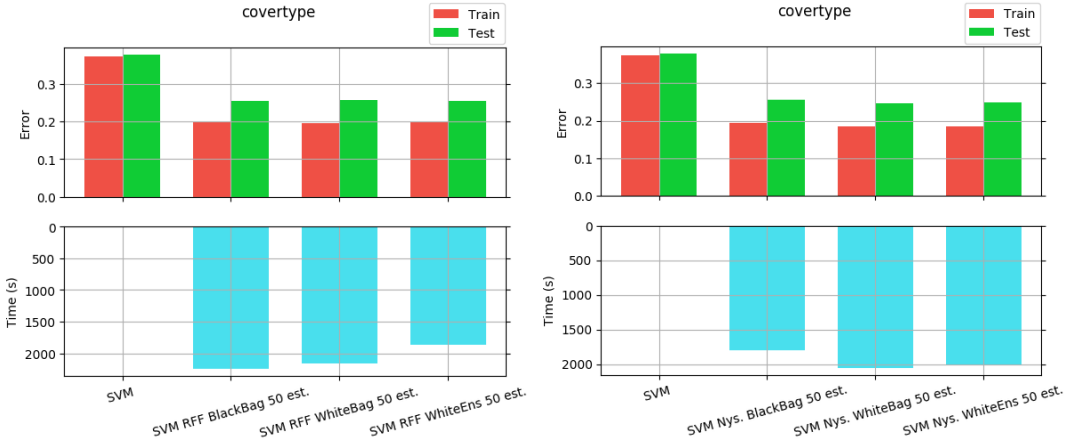


Exp 2.3 with Fashion MNIST. Error is decreased by 2% approx.

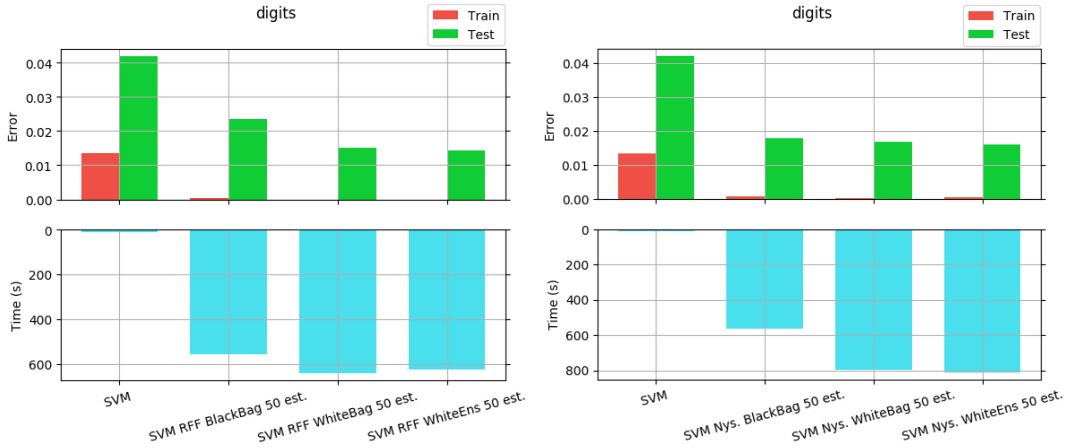
Appendix E

Results of experiment 2.4

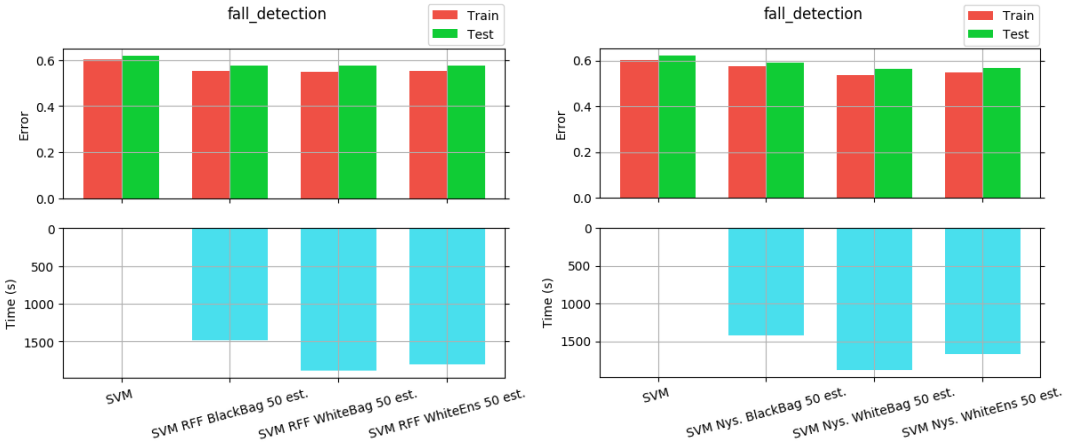
These experiments are discussed here



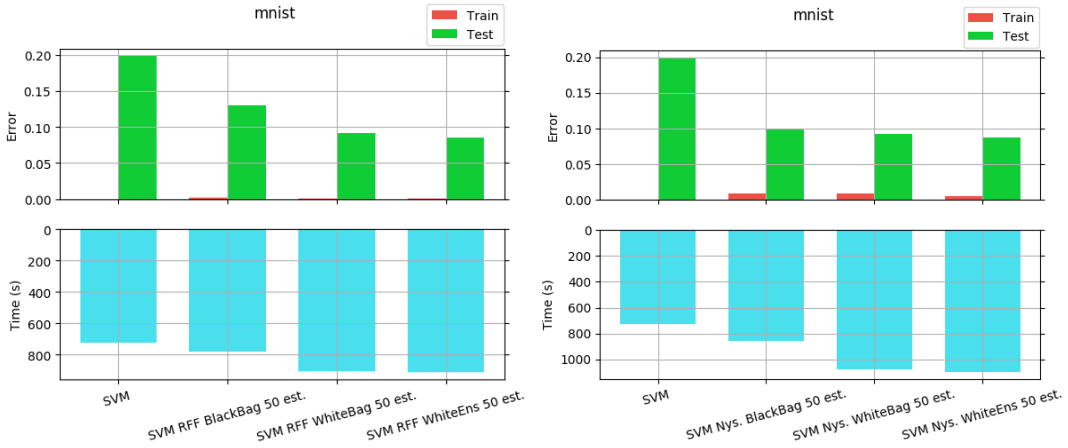
Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with Covertype. Error is decreased by 10% approx.



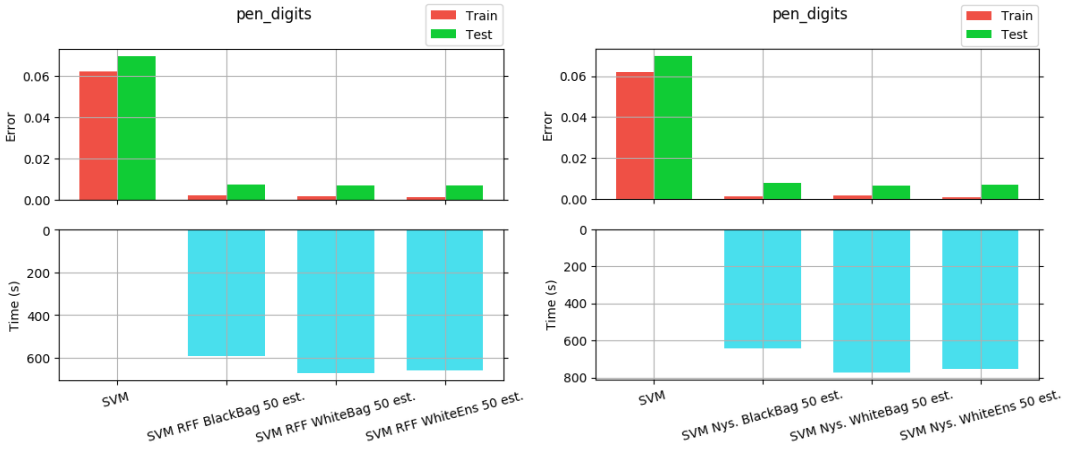
Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with Digits. Error is decreased by 2% approx.



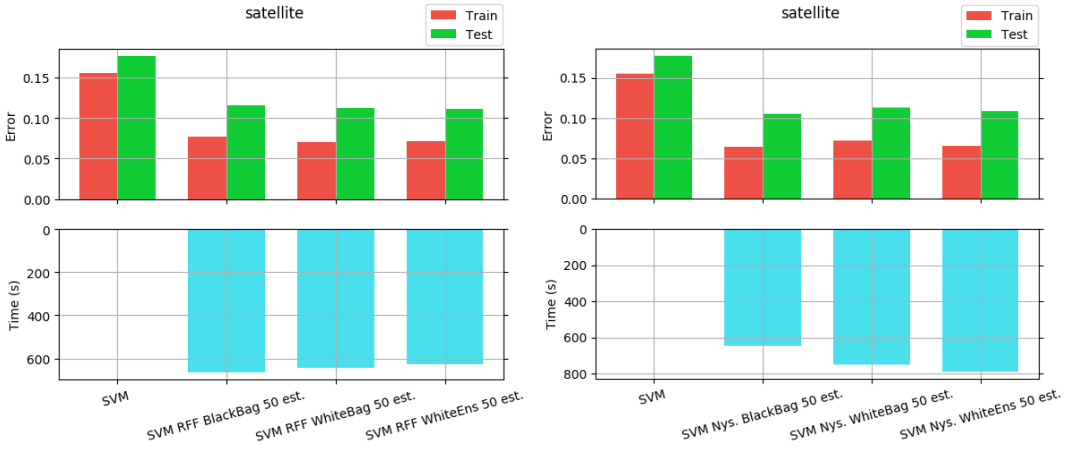
Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with Fall Detection. Error is decreased by 2% approx.



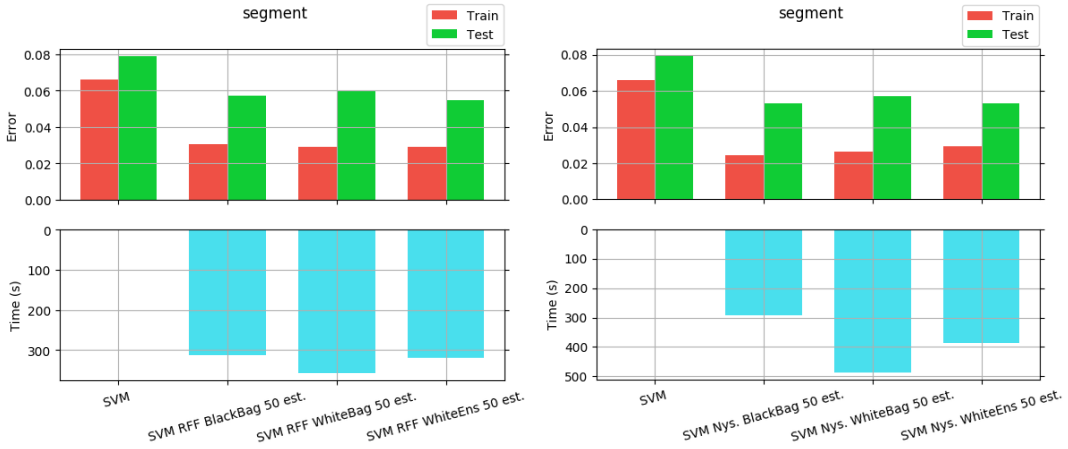
Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with MNIST. Error is decreased by 10% approx.



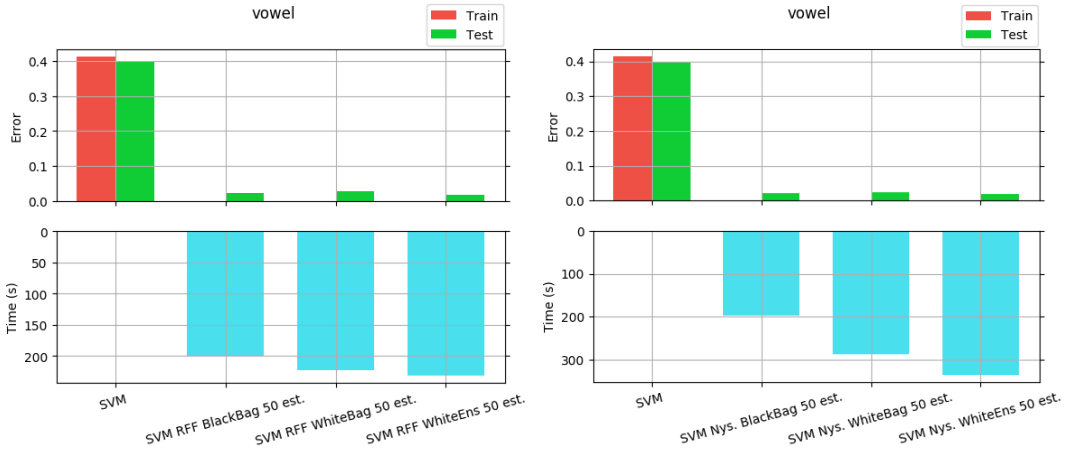
Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with Pen Digits. Error is decreased by 5% approx.



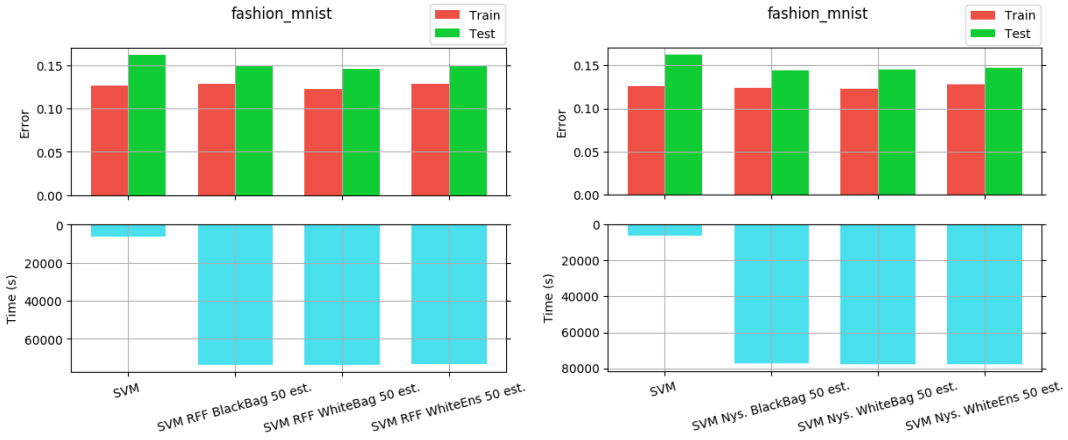
Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with Satellite. Error is decreased by 7% approx.



Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with Segment. Error is decreased by 2% approx.

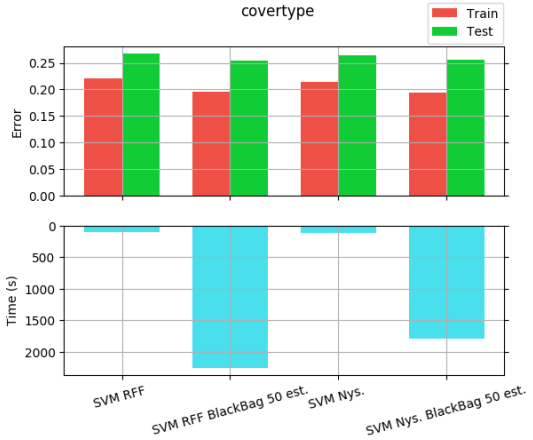


Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with Vowel. Error is decreased by 38% approx.

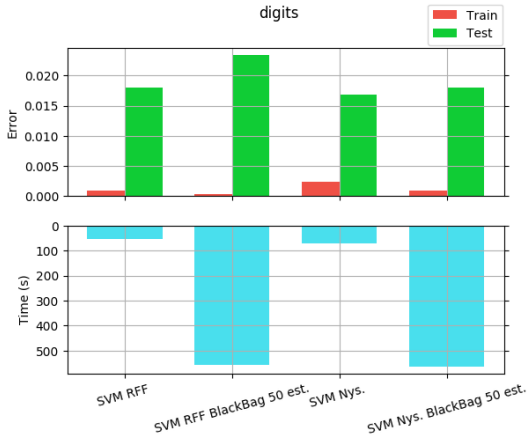


Exp 2.4 with Random Fourier Feautures (left) and Nyström (right) with Fashion MNIST. Error is decreased by 2% approx.

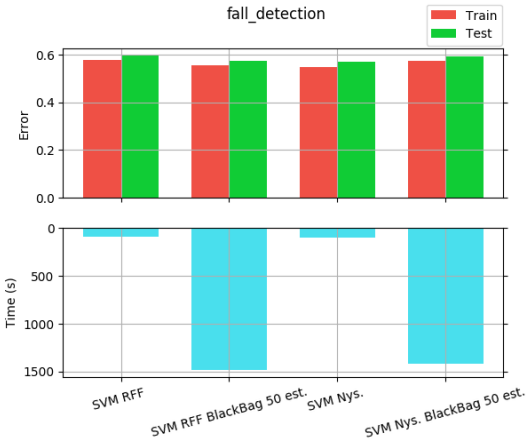
Comparison of Single Model and Ensemble both using a random mapping



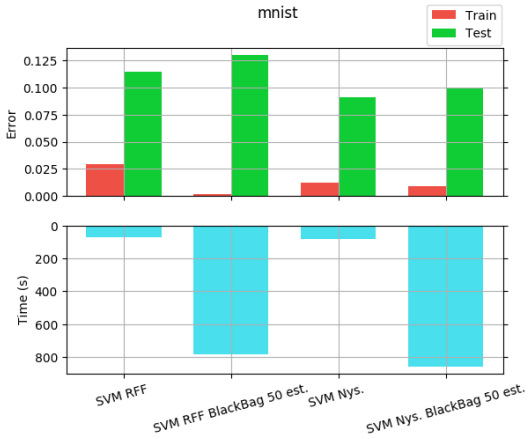
Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with Covertype.



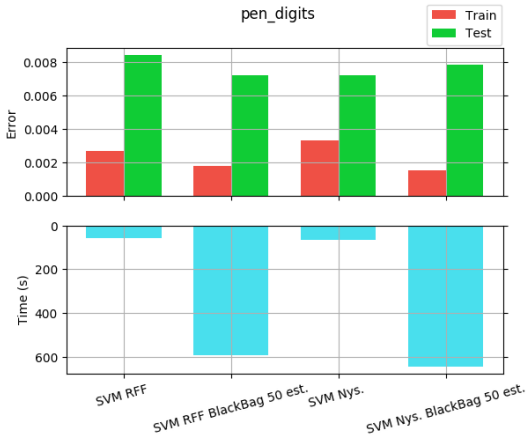
Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with Digits.



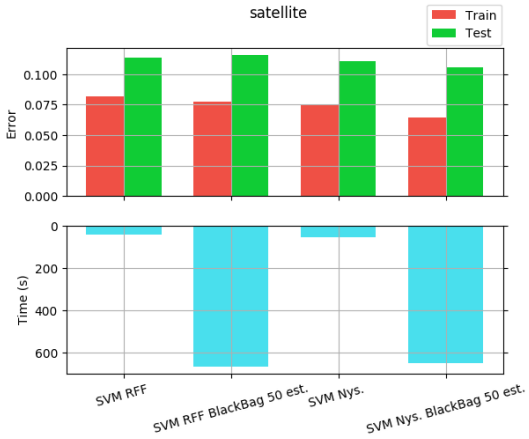
Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with Fall Detection.



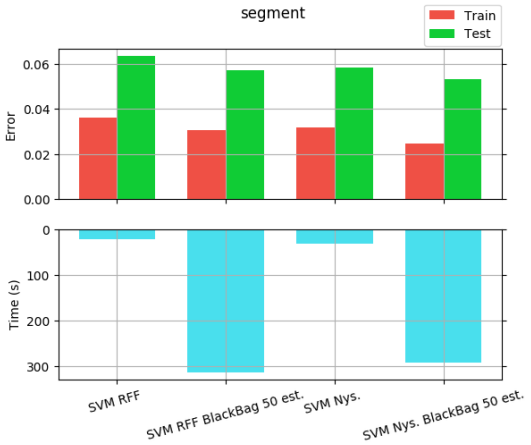
Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with MNIST.



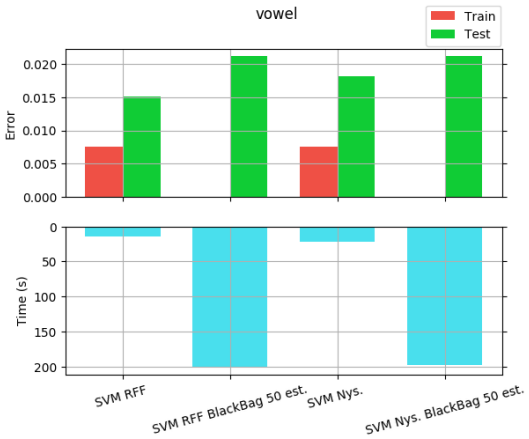
Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with Pen Digits.



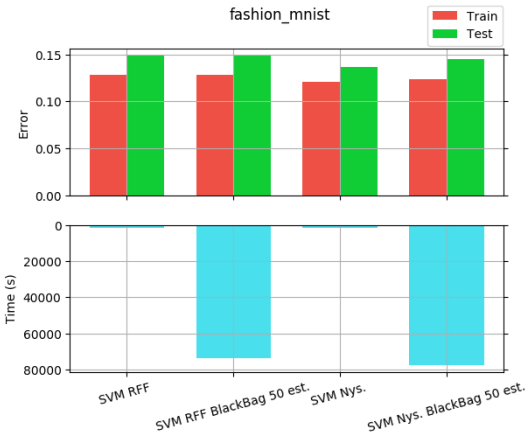
Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with Satellite.



Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with Segment.



Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with Vowel.

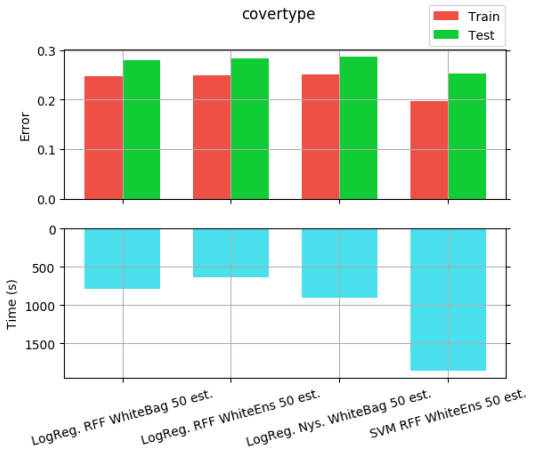


Support Vector Machine with random mapping. A single model vs. an Ensemble. There is not big difference with Fashion MNIST.

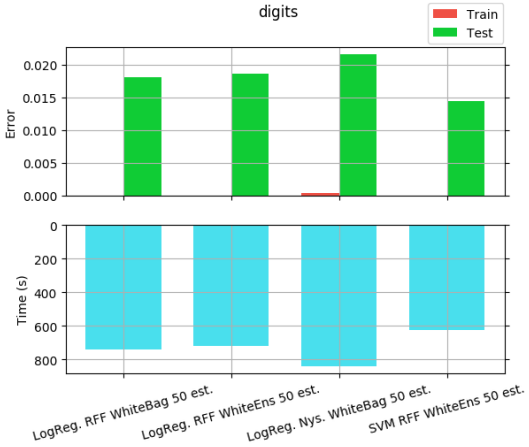
Appendix F

Results of experiment 3.1

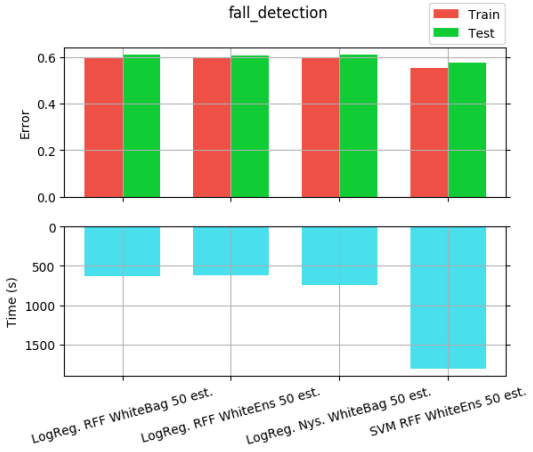
These experiments are discussed here



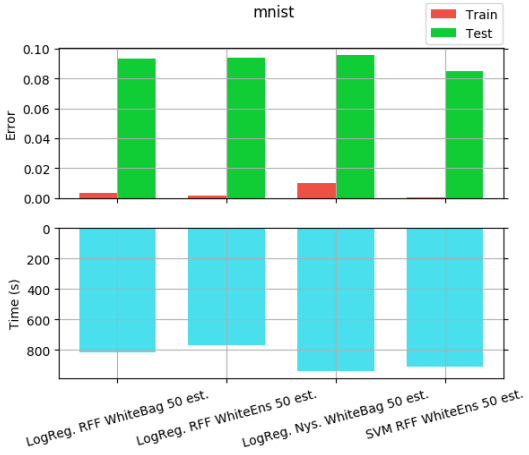
Exp. 3.1 with Covertype. White Box Models with Logistic Regression. There is not much difference between Box Models



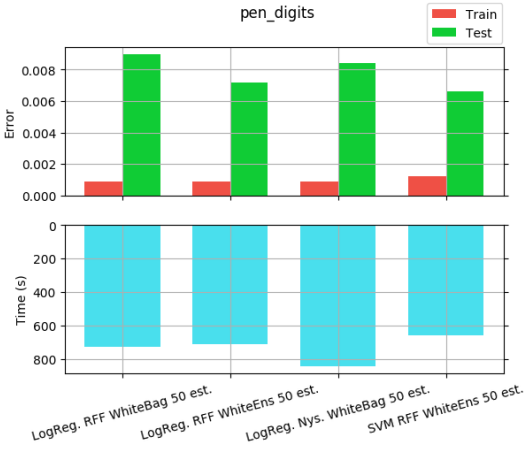
Exp. 3.1 with Digits. White Box Models with Logistic Regression. There is not much difference between Box Models



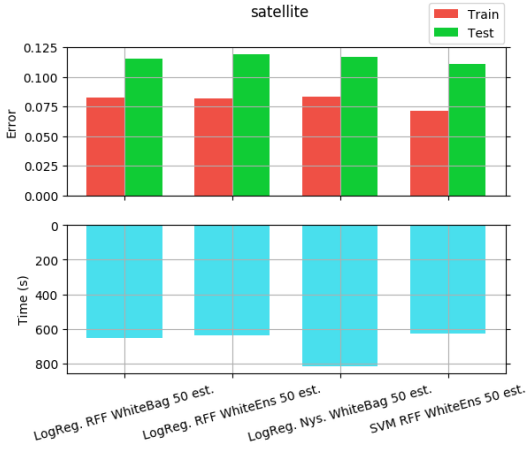
Exp. 3.1 with Fall Detection. White Box Models with Logistic Regression. There is not much difference between Box Models



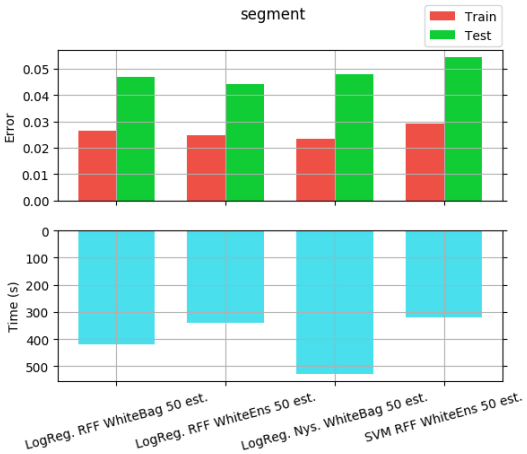
Exp. 3.1 with MNIST. White Box Models with Logistic Regression. There is not much difference between Box Models



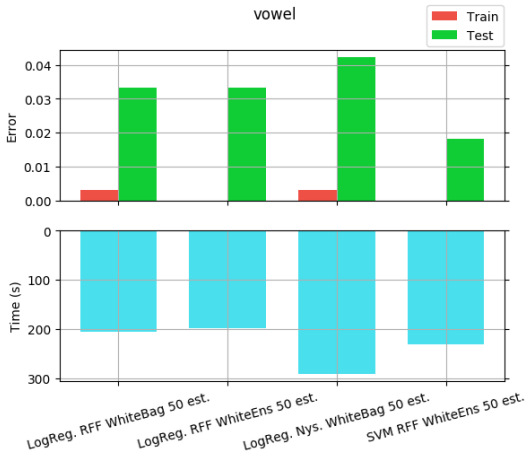
Exp. 3.1 with Pen Digits. White Box Models with Logistic Regression. There is not much difference between Box Models



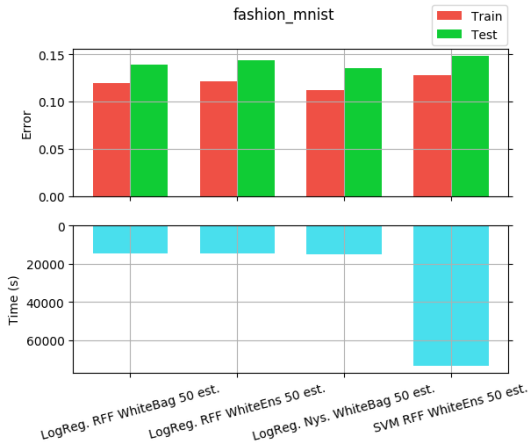
Exp. 3.1 with Satelite. White Box Models with Logistic Regression. There is not much difference between Box Models



Exp. 3.1 with Segment. White Box Models with Logistic Regression. There is not much difference between Box Models



Exp. 3.1 with Vowel. White Box Models with Logistic Regression. Ensemble with Nyström decreases error by 2%

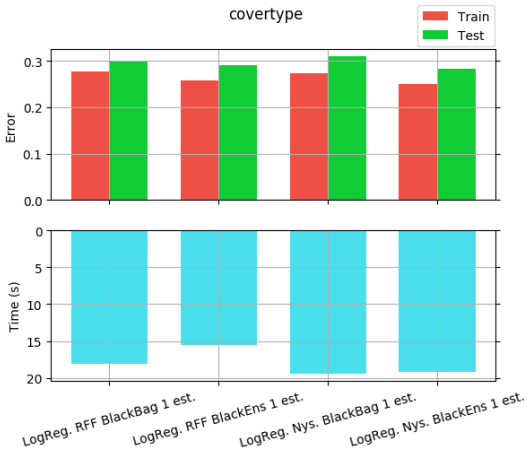


Exp. 3.1 with Fashion MNIST. White Box Models with Logistic Regression. There is not much difference between Box Models

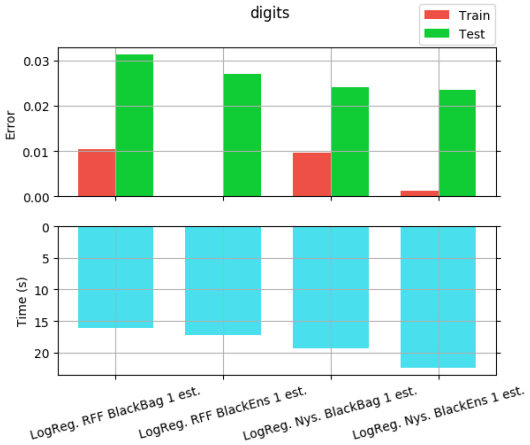
Appendix G

Results of experiment 3.2

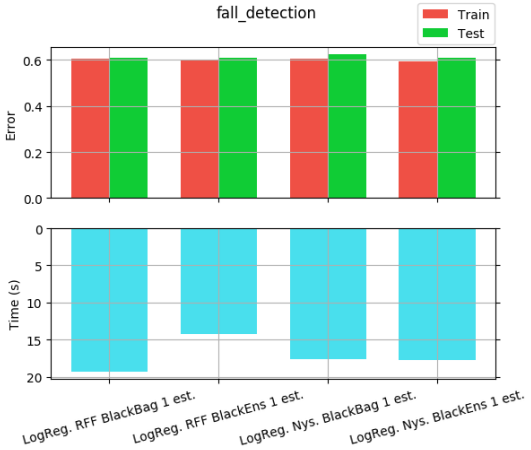
These experiments are discussed here



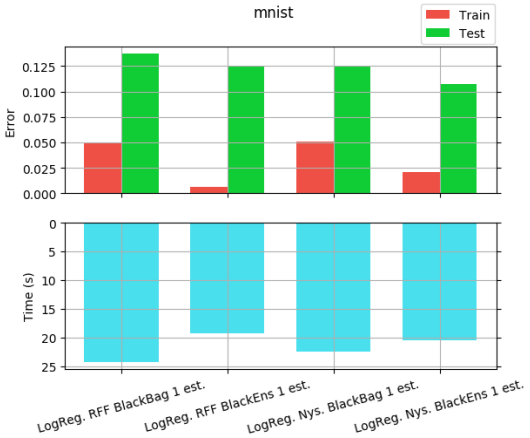
Exp. 3.2 with Covertype. Black Box Models with Logistic Regression. There is not much difference between Box Models



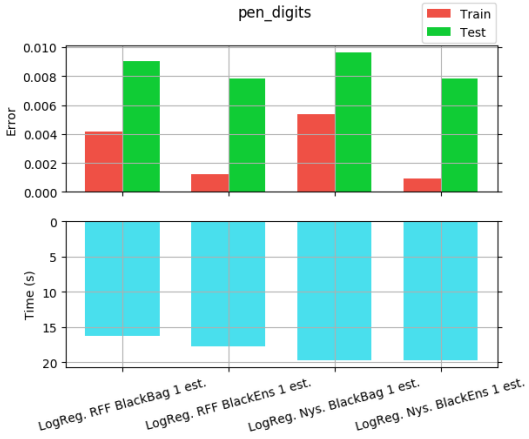
Exp. 3.2 with Digits. Black Box Models with Logistic Regression. There is not much difference between Box Models



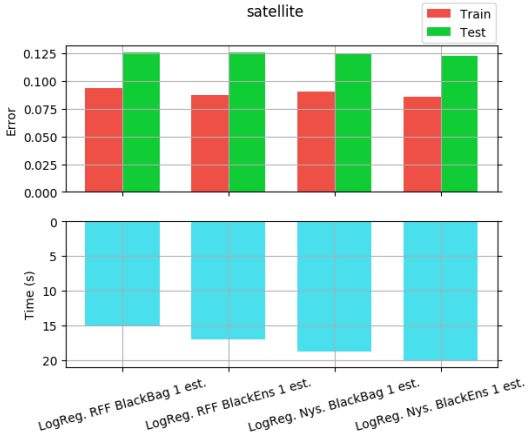
Exp. 3.2 with Fall Detection. Black Box Models with Logistic Regression. There is not much difference between Box Models



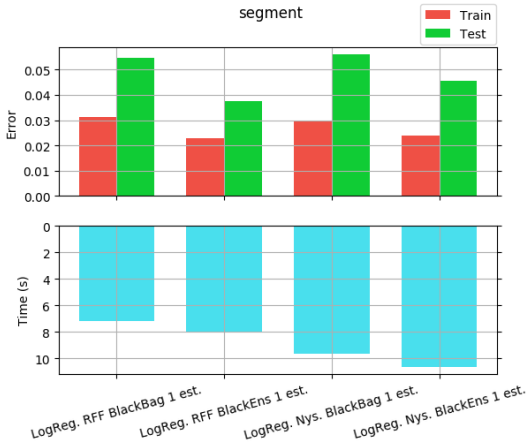
Exp. 3.2 with MNIST. Black Box Models with Logistic Regression. There is not much difference between Box Models



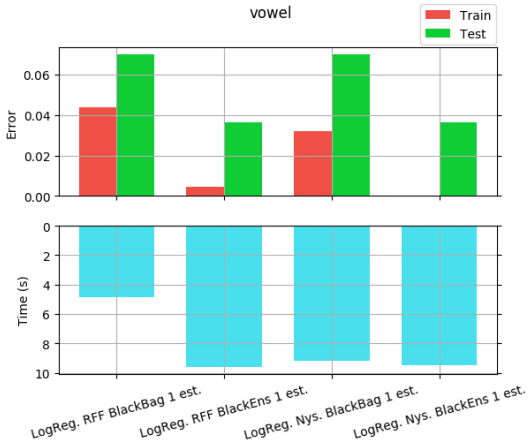
Exp. 3.2 with Pen Digits. Black Box Models with Logistic Regression. There is not much difference between Box Models



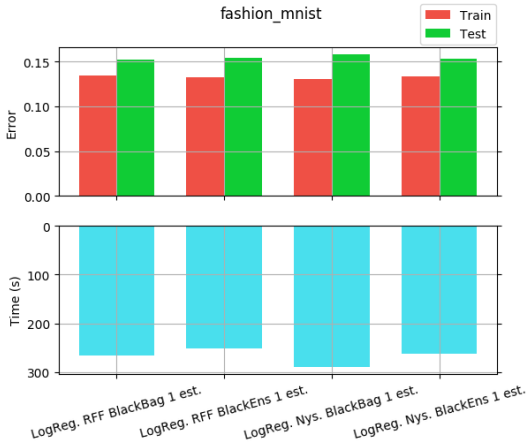
Exp. 3.2 with Satellite. Black Box Models with Logistic Regression. There is not much difference between Box Models



Exp. 3.2 with Segment. Black Box Models with Logistic Regression. There is not much difference between Box Models



Exp. 3.2 with Vowel. Black Box Models with Logistic Regression. Ensemble with decreases error by 2%

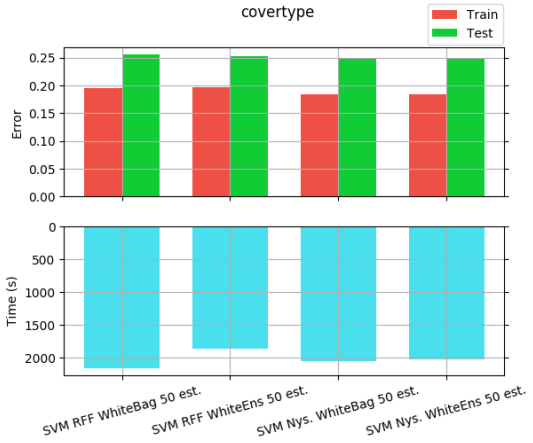


Exp. 3.2 with Fashion MNIST. Black Box Models with Logistic Regression. There is not much difference between Box Models

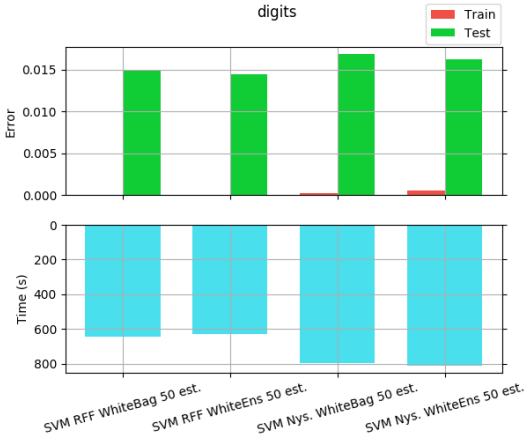
Appendix H

Results of experiment 3.3

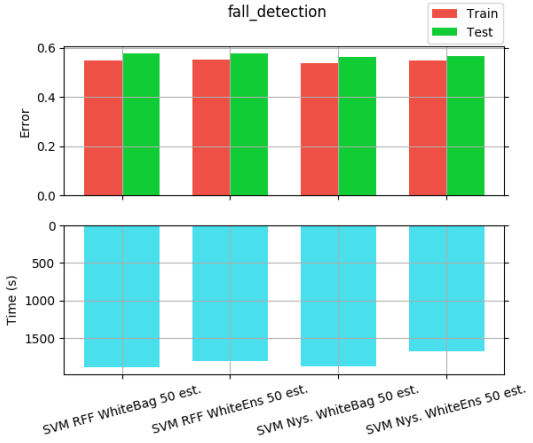
These experiments are discussed here



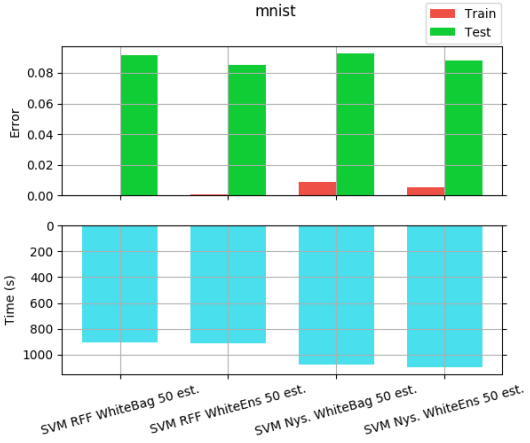
Exp. 3.3 with Covertype. White Box Models with Support Vector Machine. There is not much difference between Box Models



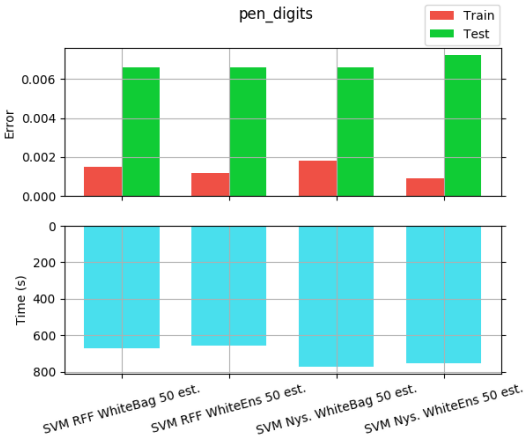
Exp. 3.3 with Digits. White Box Models with Support Vector Machine. There is not much difference between Box Models



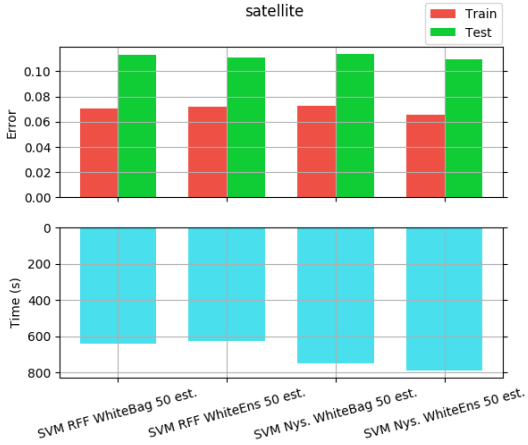
Exp. 3.3 with Fall Detection. White Box Models with Support Vector Machine. There is not much difference between Box Models



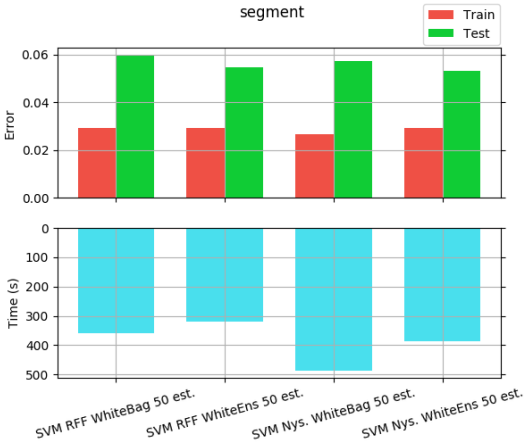
Exp. 3.3 with MNIST. White Box Models with Support Vector Machine. There is not much difference between Box Models



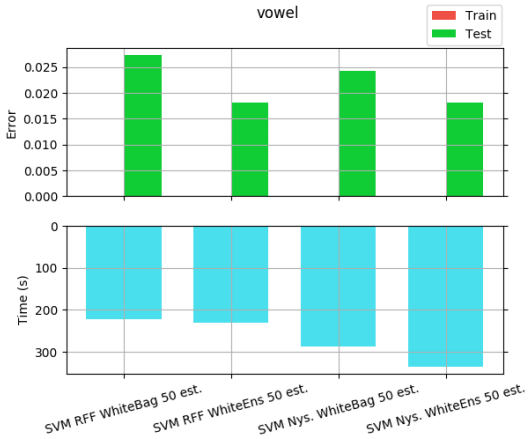
Exp. 3.3 with Pen Digits. White Box Models with Support Vector Machine. There is not much difference between Box Models



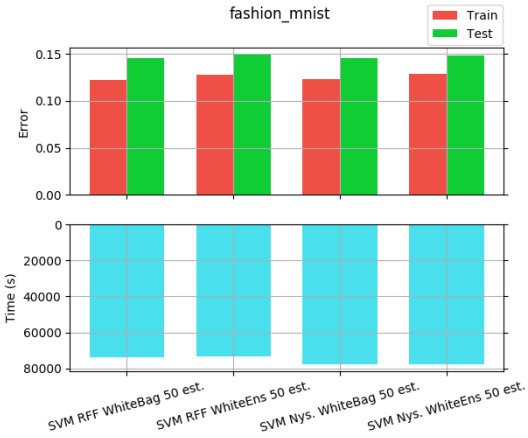
Exp. 3.3 with Satellite. White Box Models with Support Vector Machine. There is not much difference between Box Models



Exp. 3.3 with Segment. White Box Models with Support Vector Machine. There is not much difference between Box Models



Exp. 3.3 with Vowel. White Box Models with Support Vector Machine. There is not much difference between Box Models

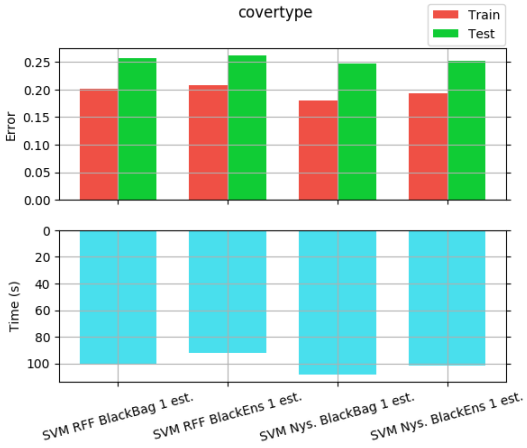


Exp. 3.3 with Fashion MNIST. White Box Models with Support Vector Machine. There is not much difference between Box Models

Appendix I

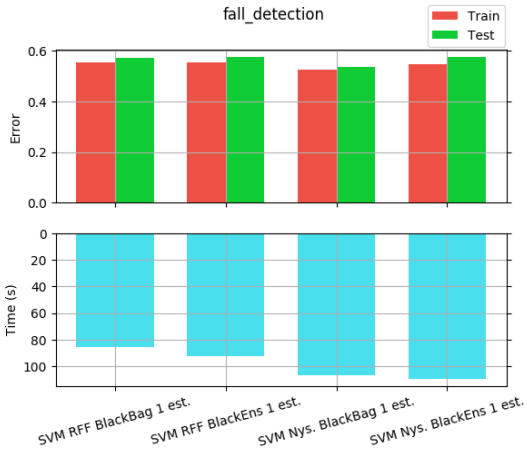
Results of experiment 3.4

These experiments are discussed here

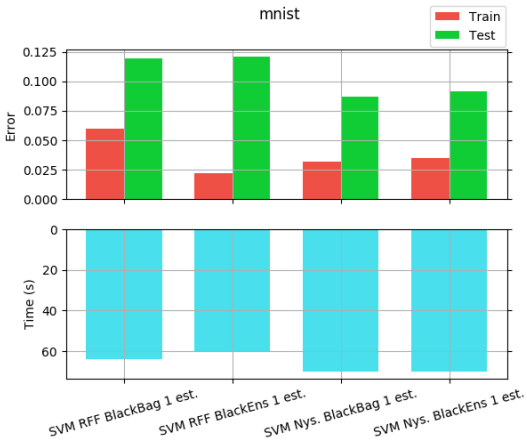


Exp. 3.4 with Covertype. Black Box Models with Support Vector Machine. There is not much difference between Box Models

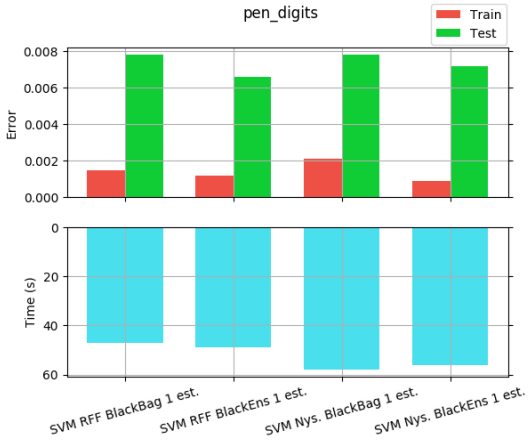
Exp. 3.4 with Digits. Black Box Models with Support Vector Machine. There is not much difference between Box Models



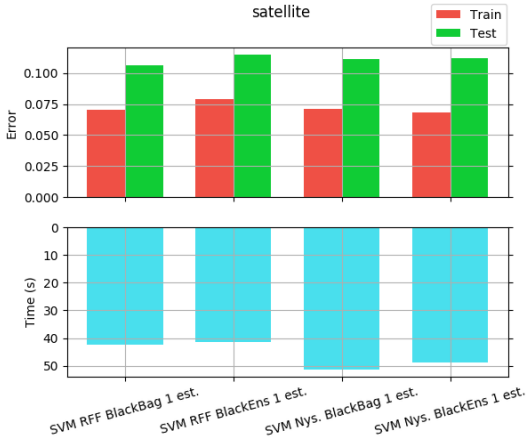
Exp. 3.4 with Fall Detection. Black Box Models with Support Vector Machine. There is not much difference between Box Models



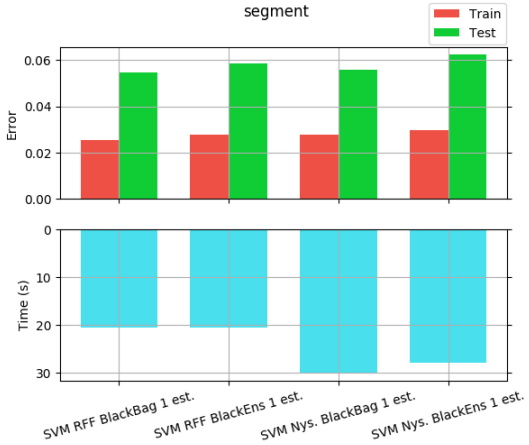
Exp. 3.4 with MNIST. Black Box Models with Support Vector Machine. There is not much difference between Box Models



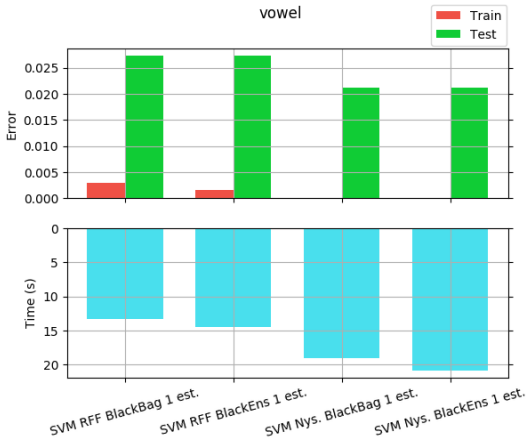
Exp. 3.4 with Pen Digits. Black Box Models with Support Vector Machine. There is not much difference between Box Models



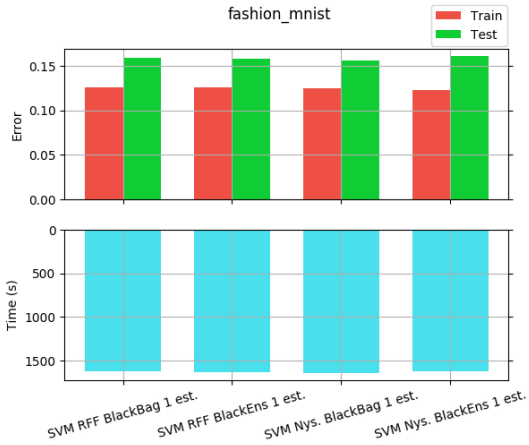
Exp. 3.4 with Satellite. Black Box Models with Support Vector Machine. There is not much difference between Box Models



Exp. 3.4 with Segment. Black Box Models with Support Vector Machine. There is not much difference between Box Models



Exp. 3.4 with Vowel. Black Box Models with Support Vector Machine. There is not much difference between Box Models

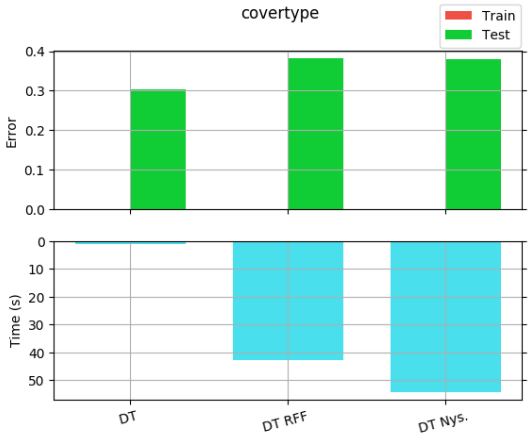


Exp. 3.4 with Fashion MNIST. Black Box Models with Support Vector Machine. There is not much difference between Box Models

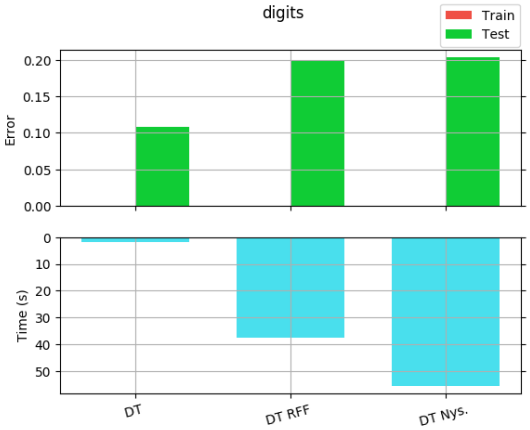
Appendix J

Results of experiment 4.1

These experiments are discussed here

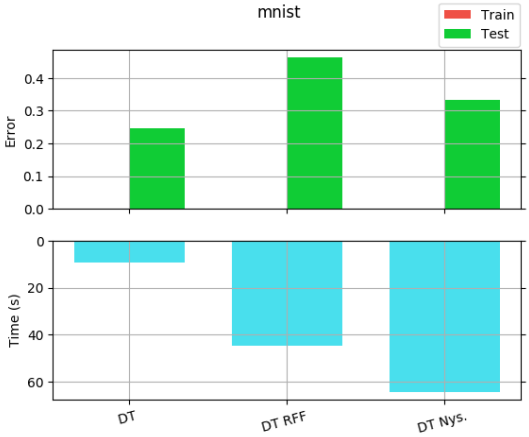


Exp. 4.1 with Covertype. Random Samplers increase the error on Decision Tree.

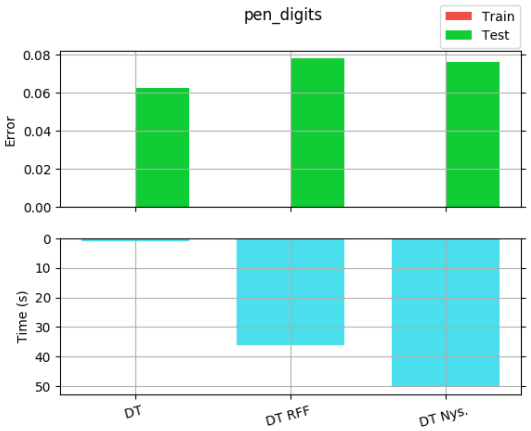


Exp. 4.1 with Digits. Random Samplers increase the error on Decision Tree.

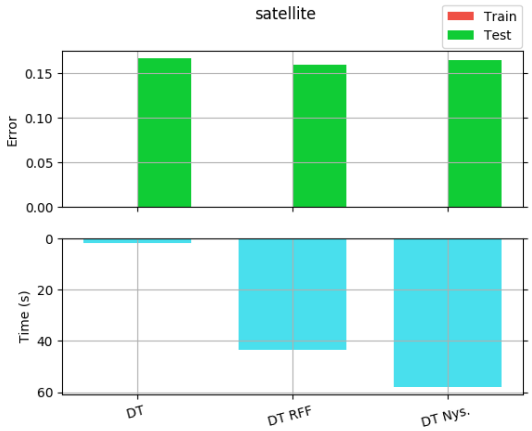
Exp. 4.1 with Fall Detection. Random Samplers increase the error on Decision Tree.



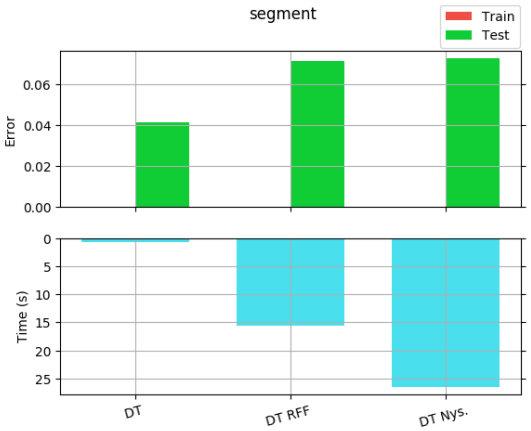
Exp. 4.1 with MNIST. Random Samplers increase the error on Decision Tree.



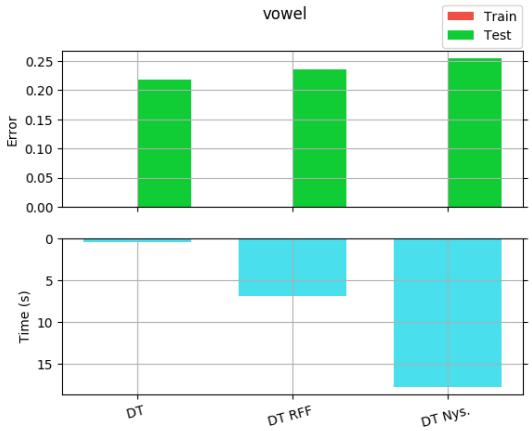
Exp. 4.1 with Pen Digits. Random Samplers increase the error on Decision Tree.



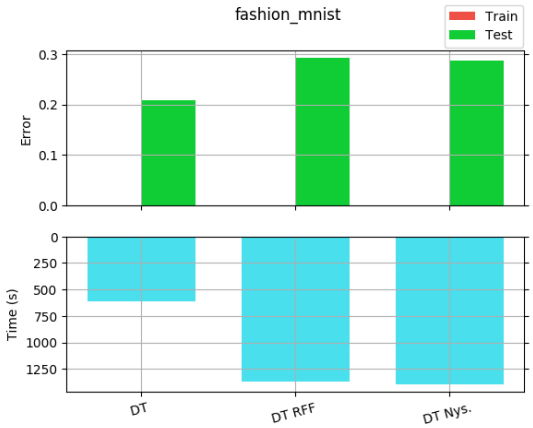
Exp. 4.1 with Satellite. Random Samplers make no difference on Decision Tree.



Exp. 4.1 with Segment. Random Samplers increase the error on Decision Tree.



Exp. 4.1 with Vowel. Random Samplers increase the error on Decision Tree.

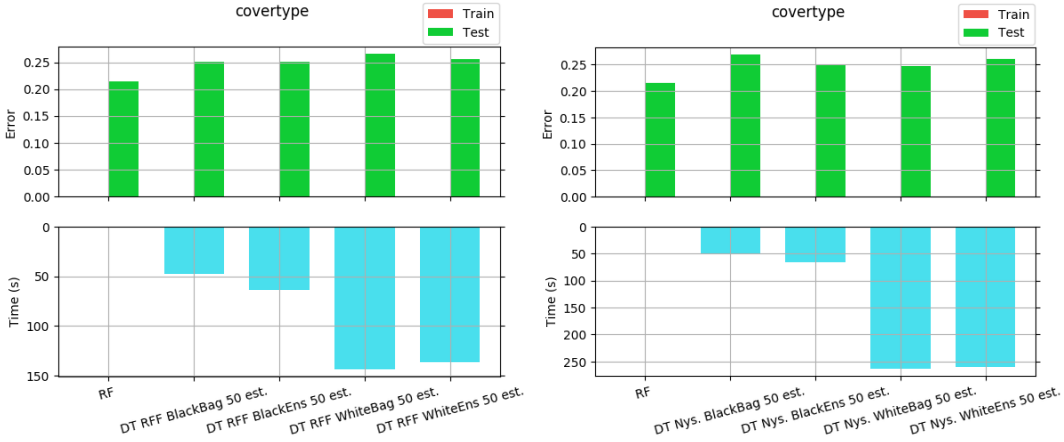


Exp. 4.1 with Fashion MNIST. Random Samplers increase the error on Decision Tree.

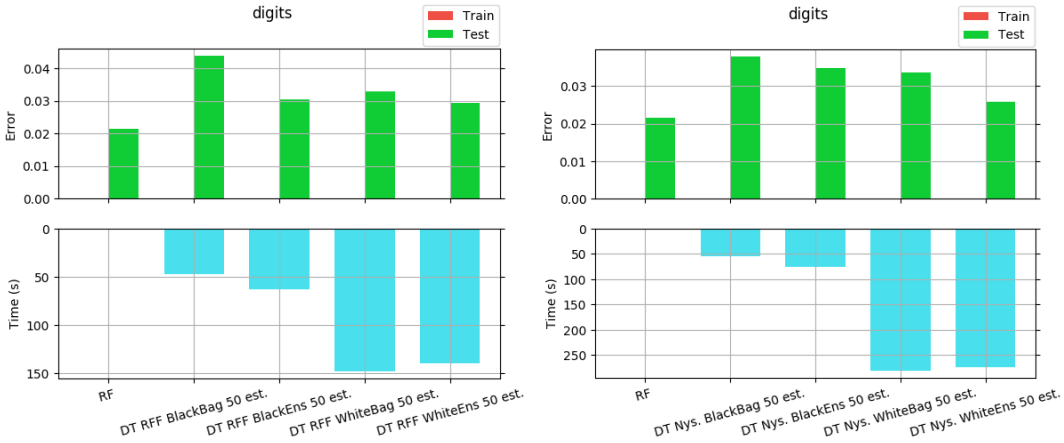
Appendix K

Results of experiment 4.2

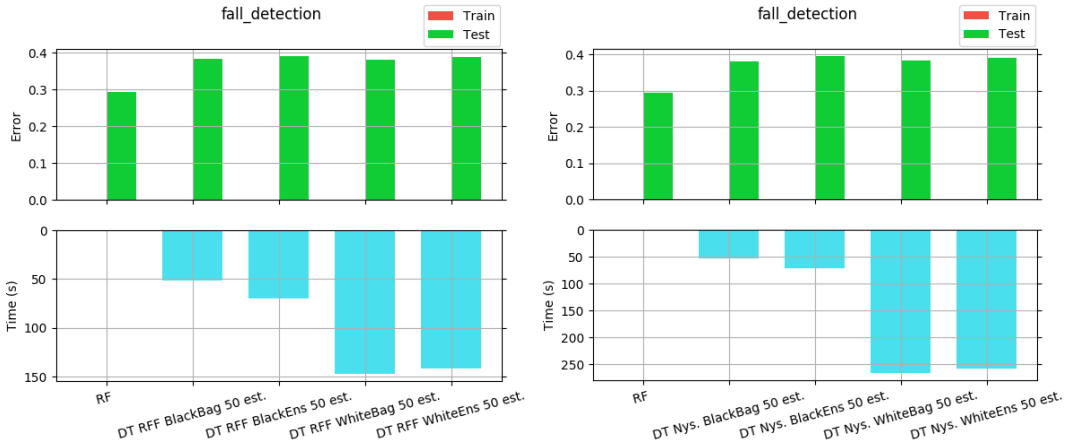
These experiments are discussed here



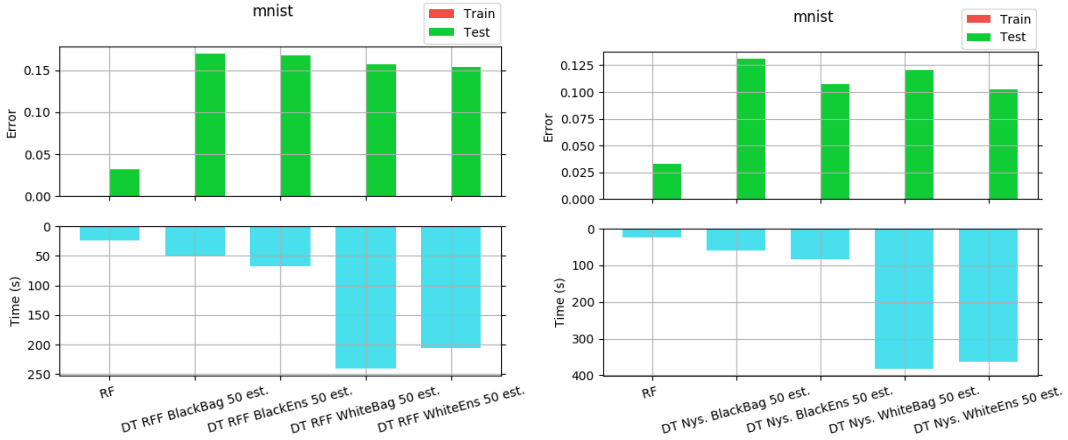
Exp. 4.2 with Covertype. Random Forest outperforms Ensembles of Decision Tree with RFF (left) and Nyström (right).



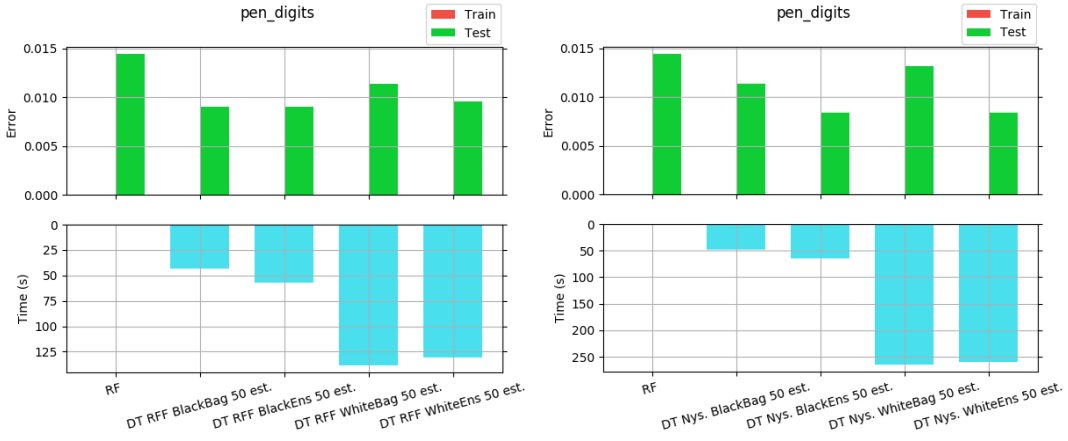
Exp. 4.2 with Digits. Random Forest outperforms Ensembles of Decision Tree with RFF (left) and Nyström (right).



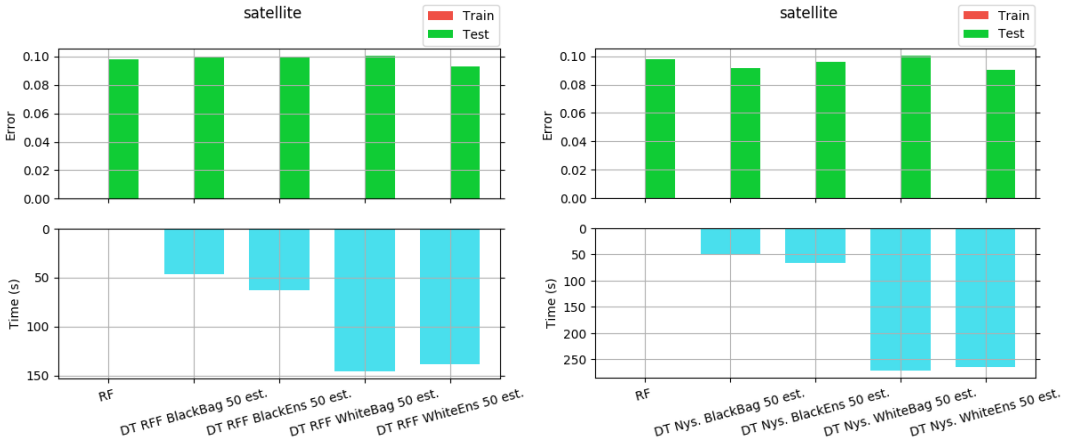
Exp. 4.2 with Fall Detection. Random Forest outperforms Ensembles of Decision Tree with RFF (left) and Nyström (right).



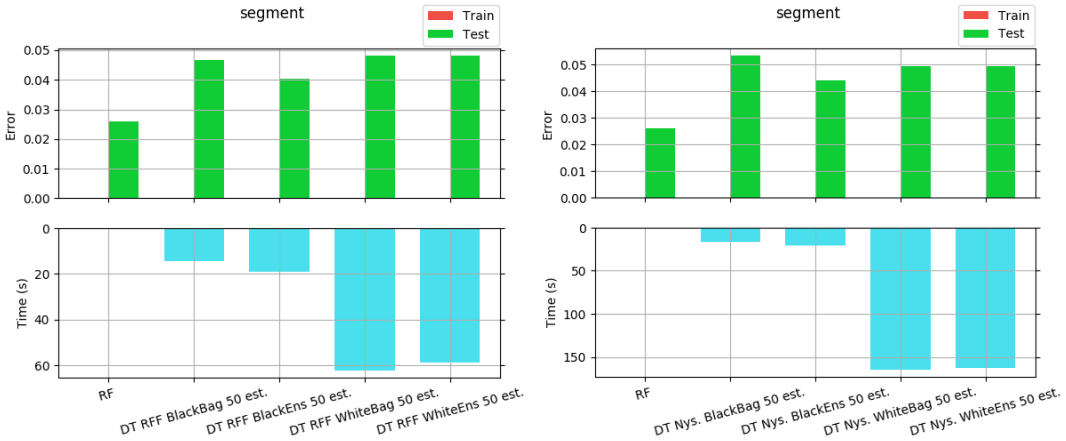
Exp. 4.2 with MNIST. Random Forest outperforms Ensembles of Decision Tree with RFF (left) and Nyström (right).



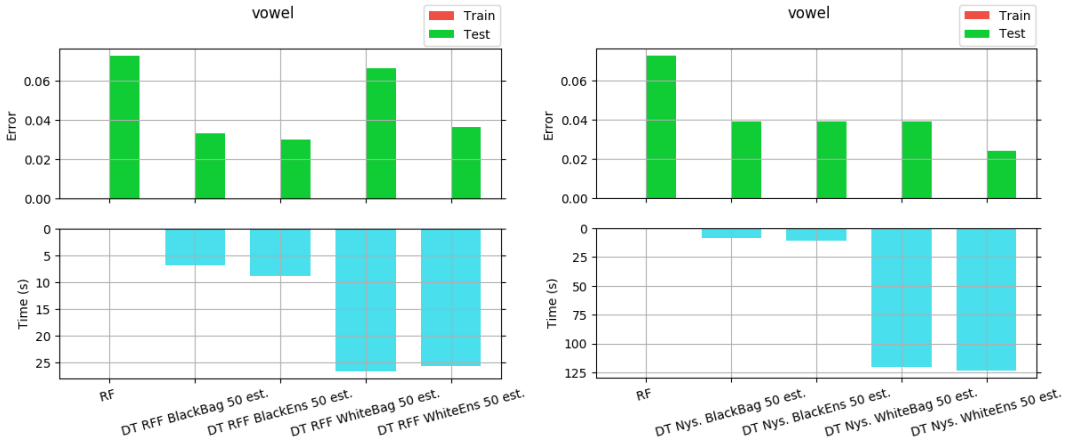
Exp. 4.2 with Pen Digits. Random Forest and Ensembles of Decision Tree with RFF (left) and Nyström (right) have the same accuracy.



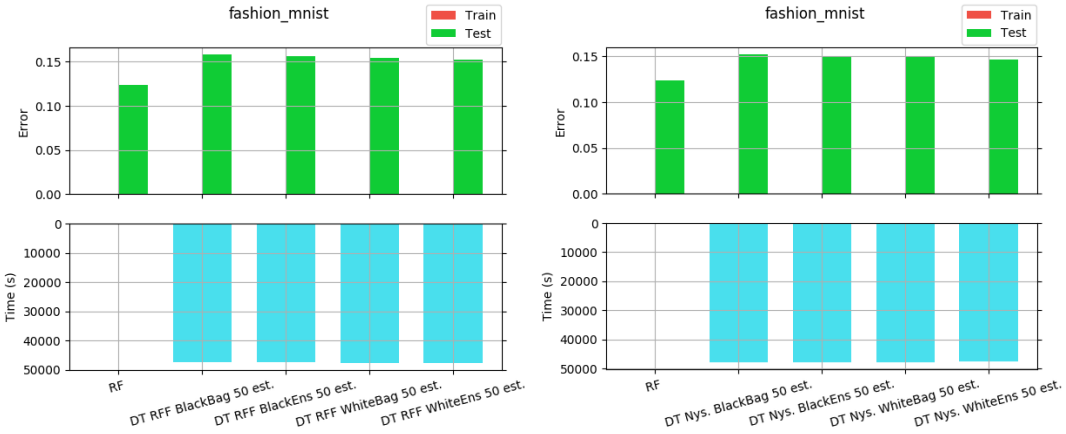
Exp. 4.2 with Satellite. Random Forest and Ensembles of Decision Tree with RFF (left) and Nyström (right) have the same accuracy.



Exp. 4.2 with Segment. Random Forest outperforms Ensembles of Decision Tree with RFF (left) and Nyström (right).



Exp. 4.2 with Vowel. Ensembles of Decision Tree with RFF (left) and Nyström (right) outperform Random Forest.



Exp. 4.2 with Fashion MNIST. Random Forest outperforms Ensembles of Decision Tree with RFF (left) and Nyström (right).

References

[3] Stefan Bergman. *The kernel function and conformal mapping*. Vol. 5. American Mathematical Soc., 1970.

[5] Leo Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[6] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[7] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[8] Christopher JC Burges. "A tutorial on support vector machines for pattern recognition". In: *Data mining and knowledge discovery* 2.2 (1998), pp. 121–167.

[9] Barbara Caputo et al. "Appearance-based object recognition using SVMs: which kernel should I use?" In: *Proc of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler*. Vol. 2002. 2002.

[10] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018. URL: https://doi.org/10.1007/BF00994018.

[11] David R Cox. "The regression analysis of binary sequences". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 20.2 (1958), pp. 215–232.

[13] Pedro Domingos. "A unified bias-variance decomposition". In: *Proceedings of 17th International Conference on Machine Learning*. 2000, pp. 231–238.

[14] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[15] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[16] Seymour Geisser. *Predictive inference*. Routledge, 2017.

[17] Jun Han and Claudio Moraga. "The influence of the sigmoid function parameters on the speed of backpropagation learning". In: *International Workshop on Artificial Neural Networks*. Springer. 1995, pp. 195–201.

[18] Douglas M Hawkins. "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1 (2004), pp. 1–12.

[20] Roger J Lewis. "An introduction to classification and regression tree (CART) analysis". In: *Annual meeting of the society for academic emergency medicine in San Francisco, California*. Vol. 14. 2000.

[21] J Mercer. *Functions of positive and negative type and their connection with the theory of integral equations, Philosophical Transsaction of the Royal Society of London, Ser*. 1909.

[22] Robi Polikar. "Ensemble based systems in decision making". In: *IEEE Circuits and systems magazine* 6.3 (2006), pp. 21–45.

[23] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems*. 2008, pp. 1177–1184.

[24] Walter Rudin. *Fourier analysis on groups*. Vol. 121967. Wiley Online Library, 1962.

[25] Yauheni Selivonchyk. "Speeding up Support Vector Machines with Random Fourier Features". In: (2016).

[27] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. "A primer on kernel methods". In: *Kernel methods in computational biology* 47 (2004), pp. 35–70.

[29] Christopher K. I. Williams and Matthias Seeger. "Using the Nyström Method to Speed Up Kernel Machines". In: *Advances in Neural Information Processing Systems 13*. Ed. by T. K. Leen, T. G. Dietterich, and V. Tresp. MIT Press, 2001, pp. 682–688. URL: http://papers.nips.cc/paper/1866-using-the-nystrom-method-to-speed-up-kernel-machines.pdf.

[31] Shuai Zhang et al. "Stacked Kernel Network". In: *CoRR* abs/1711.09219 (2017).

Datasets

[1] E. Alpaydin and Fevzi. Alimoglu. *Pen-Based Recognition of Handwritten Digits*. July 1998. URL: https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits.

[2] E. Alpaydin and C. Kaynak. *Optical Recognition of Handwritten Digits*. July 1998. URL: http://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits.

[4] Jock A. Blackard, Dr. Denis J. Dean, and Dr. Charles W. Anderson. *Forest Covertype data*. Aug. 1998. URL: https://archive.ics.uci.edu/ml/datasets/covertype.

[12] David Deterding, Mahesan Niranjan, and Tony Robinson. *Vowel Recognition (Deterding data)*. URL: https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Vowel+Recognition+-+Deterding+Data).

[19] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *MNIST database*. URL: http://yann.lecun.com/exdb/mnist/.

[26] Ashwin Srinivasan. *Statlog (Landsat Satellite)*. Feb. 1993. URL: https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite).

[28] University of Massachusetts Vision Group. *Image Segmentation data*. Nov. 1990. URL: http://archive.ics.uci.edu/ml/datasets/image+segmentation.

[30] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[32] Özdemir, Ahmet Turan, and Billur Barshan. *Detecting Falls with Wearable Sensors Using Machine Learning Techniques*. 2017. URL: https://www.kaggle.com/pitasr/falldata.