Universitatea POLITEHNICA din București
Facultatea de Automatică și Calculatoare

Predicţia performanţei studenţilor unui curs folosind tehnici de Învăţare Automată

Autor:
Marin Cristian-Emil

Coordonator științific:
Şl.dr.ing. Popovici Matei

București
Iulie/Septembrie 2016

University POLITEHNICA of Bucharest
Faculty of Automatic Control and Computers

Predicting Student Course Performance with Machine Learning Techniques

Author:
Marin Cristian-Emil

Scientific Coordinator:
Şl.dr.ing. Popovici Matei

Bucharest
July/September 2016

Contents

Abstract

Scriem ceva abstract aici, maxim 150 cuvinte.

Keywords: grade prediction; machine learning; data analysis; neural networks; random forests; data classification; model entropy

Chapter 1

Introduction

1.1 Motivation

Lucrarea este structurată astfel: un fisier de configurare poate fi găsit în anexa A.1.

Mai jos urmează niște text de umplutură și un algoritm realizat cu algortihm2e.

Duis consectetur tempor volutpat. Nullam molestie, nibh fringilla egestas euismod, mi eros euismod augue, eu rutrum mi nibh vitae sapien. Sed ultricies vehicula est a tincidunt. Aenean nec ligula purus, quis porttitor velit. Donec ultrices, velit sed facilisis condimentum, ligula ligula facilisis mi, ac accumsan arcu tellus eu tortor. In congue mauris ac est aliquet a ullamcorper dui fermentum. Donec venenatis nulla ut lectus cursus elementum. Phasellus vulputate aliquet dolor convallis placerat. Nam pharetra, nibh ut mollis venenatis, purus nisl adipiscing diam, quis dignissim mauris risus non velit. Aliquam et mauris non tortor malesuada luctus. Integer quis justo turpis.

Textul următor este citat din.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce auctor ultrices sollicitudin. Ut sed lobortis nunc. Proin adipiscing erat vel dolor pharetra sed semper felis malesuada. Quisque imperdiet augue lectus. Nulla facilisi. Fusce tortor quam, pharetra et blandit et, pulvinar convallis risus. Nam id molestie dolor. Phasellus aliquam laoreet dolor, vel semper neque blandit sit amet. Quisque

```
1  function eval():
2  begin
3      if bubu = null then
4          return null
5      end if
6      bubu ← bibi
7      foreach foo in bar do
8          if foo is zaz then
9              bubu ← bubu['lili.lala']
10         else if term is Call then
11             bibi ← call(cucu)
12         end if
13     end foreach
14     return crtContext
15 end
```

Figure 1.1: Algoritmul de evaluare al crocobazilor.

quis tellus ipsum. Fusce fringilla, neque vitae tincidunt placerat, turpis metus dignissim justo, quis blandit justo est nec tellus. Nunc pretium, felis sed mattis euismod, metus leo tempus risus, a sodales augue lorem id libero.

1.1.1 Lorem ipsum

Duis consectetur tempor volutpat. Nullam molestie, nibh fringilla egestas euismod, mi eros euismod augue, eu rutrum mi nibh vitae sapien. Sed ultricies vehicula est a tincidunt. Aenean nec ligula purus, quis porttitor velit. Donec ultrices, velit sed facilisis condimentum, ligula ligula facilisis mi, ac accumsan arcu tellus eu tortor. In congue mauris ac est aliquet a ullamcorper dui fermentum. Donec venenatis nulla ut lectus cursus elementum. Phasellus vulputate aliquet dolor convallis placerat. Nam pharetra, nibh ut mollis venenatis, purus nisl adipiscing diam, quis dignissim mauris risus non velit. Aliquam et mauris non tortor malesuada luctus. Integer quis justo turpis.

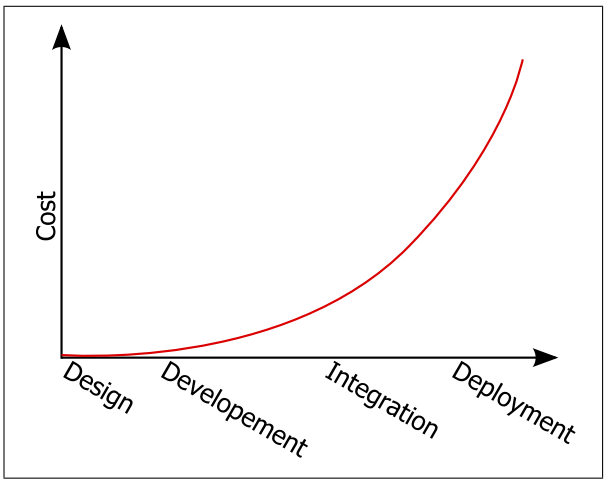

Figure 1.2: Variația costului de remediere a dudelor cu momentul descoperirii lor

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce auctor ultrices sollicitudin. Ut sed lobortis nunc. Proin adipiscing erat vel dolor pharetra sed semper felis malesuada. Quisque imperdiet augue lectus. Nulla facilisi. Fusce tortor quam, pharetra et blandit et, pulvinar convallis risus. Nam id molestie dolor. Phasellus aliquam laoreet dolor, vel semper neque blandit sit amet. Quisque quis tellus ipsum. Fusce fringilla, neque vitae tincidunt placerat, turpis metus dignissim justo, quis blandit justo est nec tellus. Nunc pretium, felis sed mattis euismod, metus leo tempus risus, a sodales augue lorem id libero.

Chapter 2

Background

2.1 State of the Art

2.1.1 A very short history of Machine Learning

"You have to know the past to understand the present." - Carl Sagan

The early beginnings of Machine Learning come not after the first electronic computers or after the first computer programs, as many may believe so. The fundamentals of this field took shape centuries ago with the discovery of the so-called Conditional Probability theories. **Bayes Theorem**, named after Thomas Bayes who first proposed a mathematical model for infering probabilities of conditioned events, and further developed by Pierre-Simon Laplace in his essay [1] in 1812, can be seen as one of the first models that can "learn" from given data and predict events based on correlated past events. Another old discovery that is the basis of today's regression models (e.g.: Linear Regression) is the "least squares method", credited to Carl Gauss, but first published by Adrien-Marie Legendre in 1805. This method was first applied in astronomy and allowed explorers to navigate oceans by aproximating the movement of celestial bodies.

Later on, in 1950, Alan Turing proposed in his paper[2] a *learning machine* that is able to learn and become intelligent and do well in the **Imitation Game** (now

[1] Théorie Analytique des Probabilités

[2] Turing, A.M. (1950). Computing machinery and intelligence. Mind, 59, 433-460.

generally called the **Turing Test**).

After this, the discovery of the Percetron and the **Neural Networks** around 1960s drew some attention in the field, but their current limitations had put Machine Learning on an impeding state for almost 10 years. It was only with the invention of the backpropagation algorithm in 1974 by Paul Werbos and the demonstration of its generalization by Geoffrey Hinton in 1986, that allowed it to be applied in multi-layered artificial neural networks. This also gave birth to a new sub-field of Machine Learning that today is called **Deep Learning.**

Along with the research in neural networks, some other models that were developed in that period are worth to mention. The most important ones are Support Vector Machines and kernels (models used for data classification and regression, that can be more time-efficient than neural networks are and provide good performance from a data perspective) and Decision Trees. The latter, in combination with Ensemble Methods helped researchers invent models like **Random Forests** and **Adaptive Boosting** that are now state-of-the-art algorithms for tree models used for a lot of tasks.

2.1.2 Current interests in the field

Coming back to Deep Learning, which is today's main subject of interest of the Machine Learning community, it is a general approach that combines several state-of-the-art models of Machine Learning to solve problems such as image classification, AI for computer games, natural language, etc. Its constituents include neural networks with many hidden layers, convolutional networks, deep belief networks and recurrent networks. Also, the Q-Learning algorithm[3] and the Monte-Carlo search used in combination with convolutional networks allowed researchers to build semi-supervised learning programs that could learn to play computer games by themselves[4] or beat professional human players at games, such as Go (Google AlphaGo's program first beat Lee Sedol in October 2015).

[3]Watkins, C.J.C.H. (1989). Learning from Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England

[4]https://www.cs.toronto.edu/ vmnih/docs/dqn.pdf

2.1.3 Trends in educational learning

All advances in the Machine Learning field also conducted in an incresing interest in educational learning and assessment. With the name of **Educational Data Mining**, this newly emerging discipline deals with studying machine learning and data mining models in order to gain important knowledge about the structure of an educational system data (final grades, performance indicators, course drop-outs). Educational data can be taken from schools and universities (the classical way and also, the way that this thesis explores), online courses (such as MOOCs[5]), or even collaborative learning.

With the use of Machine Learning techniques, student can better decide on what courses they could take (based on past related grades), whether they have a chance or not to pass an exam before taking it and what indicators are relevant to their final asessment. The course department also benefits from the "learned" data because it helps them to better plan the structure of their courses, whether they are on-line or taken at the university.

2.2 Related Work

This part of the chapter will focus on the work on other people about the study on student performance prediction and analysis. Some of their research is similar to that of this thesis, and some others treat only related issues.

(Mehdi Sajjadi et al., 2016)[1] did some work on approximating final grades of students in a course on algorithms. In the grading process they used the peer grading method[6], and then applied Machine Learning (both supervised and usupervised) to aggregate those grades into a final grade. Their results were not so good compared to the simple method of just using the mean of all peer grades per an assessment as the final grade.

(Siddharth Reddy et al., 2015)[2] worked on developing a representational model of combined students and educational content (assessments and lessons). This

[5]Massive Open Online Courses

[6]A process in which students grade work of other students based on a given guideline

representation is actually a semantic space[7] and it can be used to study the relation between course content and students. Several conclusions can be drawn from these representations, such as: probability of passing an assessment or course and knowledge gained from completing a lesson. This article aimed mostly at MOOCs platforms, like Coursera, EdX and Khan Academy, and the model described was tested on synthetic student data and also, on real data from Knewton. Their model's results can be used to personalize the learning process of a course for each student in order to maximize the educational performance. Also, this model successfully predicted assessment results.

(Michael Wu, 2015)[3] wrote an interesting Master Thesis in which describes a Machine Learning Model that simulates MOOC data. Working with data gathered from EdX, the model once trained, can be able to synthesize student data. The model was trained to learn about student types, habits and difficulty of course materials. One of the main results of the thesis was being able to classify students in 20 important clusters.

(Saeed Hosseini Teshnizi and Sayyed Mohhamad Taghi Ayatollahi, 2015)[4] did a comparasion between Logistic Regression and ANNs[8] on a dataset composed of 275 undergraduate students and 16 student characteristics (e.g.: age, gender, parent education, employment status, place of residence, etc.) in order to predict academic failure. They concluded that the neural network models had a better accuracy than Logistic Regression (84.3% versus 77.5%). They tested 9 ANNs from which the one with 15 neurons in the hidded layer provided the best results, so ANNs methods were appropiate to be used in their problem.

Other references[5],[6],[7] also treat prediction of student academic performance mostly with neural networks and provide good accuracy with this model (the accuracy is also dependent of the structure of the dataset, number of examples, number of characteristics and noise in the data).

(Emaan Abdul Majeed and Khurum Nazir Junejo)[8] had some great results using Machine Learning models for predicting student's performance. They claim that they were capable of predicting the final grade with an accuracy of 96%. With a final number of 2500 student records and about 10 attributes for each record,

[7]Semantic similarity between objects represented as a kind of "metric" in space

[8]Artificial Neural Networks

they managed to predict the value of the final Grade attribute (which is a Class Variable that can have 6 values: A, B+, B, C+, C, Fail). Four classifier models were used in their study and we can see in Figure 2.1[8] their performance:

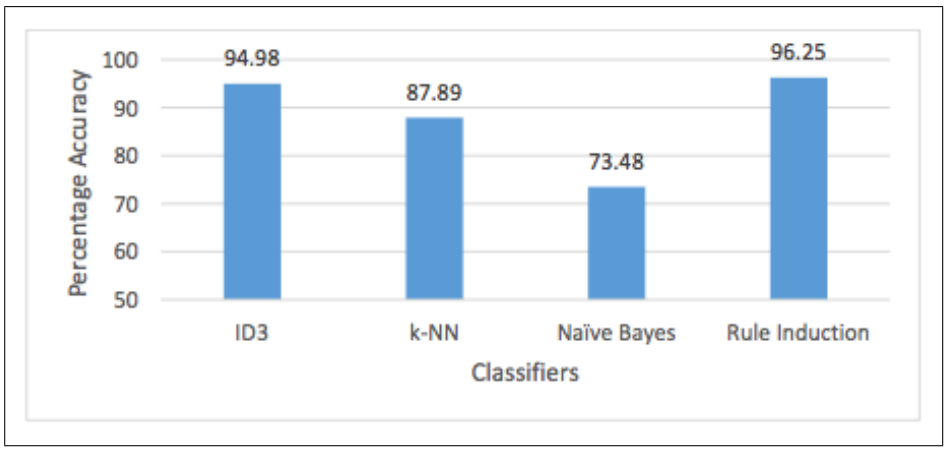

Figure 2.1: Classifiers accuracy

Chapter 3

Model Design and Methods Used

3.1 Getting and Pre-processing the Data

The data that was used in this thesis comes from our Computer Science Undergraduate course, i.e. Analysis of Algorithms from 2^{nd} year during the Fall 2015 semester. For that semester, we have a total record of 180 registered students, but we removed those whose attributes were not relevant. Since we are interested in classifying students based on their exam grade, also implying the classification in *passed* and *failed* classes, we also removed those student that had a poor grade during the semester and could not participate at the final exam. Having said this, we are left with a total of 137 relevant student records. These records must further be split into two sets: the training set - the set that is used for building the model and the testing set - the set used for testing the accuracy and measure the performance indicators of the model. There is not a standard recipe of how to divide these two sets, but it is recommended that the training set should have a proportion between 60% and 80% of the total dataset's number of examples.

3.1.1 Dataset Structure

In Table 3.1 there is a listing with all the attributes (features) used for the training models. There are a total of 10 initial features (will address later on this problem with the number of features). The weight characteristic of the features represents

the actual maximum points that a student can get from that assignment (they are all equal, summing for a total of 5 semester points). This was not used in the project, since the value is constant for all features. From the table, we can see that the range for the test and homework grades is not the same. This was resolved by dividing the test grades range with a factor of 20, in order to have a better uniformization of the datased before applying standard pre-processing techniques that are required from learning models.

Feature	Description	Type	Range	Weight
t_1	Test 1 grade	Float	0-10	0.5
t_2	Test 2 grade	Float	0-10	0.5
t_3	Test 3 grade	Float	0-10	0.5
t_4	Test 4 grade	Float	0-10	0.5
t_5	Test 5 grade	Float	0-10	0.5
t_6	Test 6 grade	Float	0-10	0.5
hw_1	Homework 1 grade	Float	0-0.5	0.5
hw_2	Homework 2 grade	Float	0-0.5	0.5
hw_3	Homework 3 grade	Float	0-0.5	0.5
hw_4	Homework 4 grade	Float	0-0.5	0.5

Table 3.1: Dataset Attribute Details

Next, we need to provide the values used for labelling the examples. Since this thesis treats both classification and regression problems, different types of labels must be present in the dataset. For classification, we use the final grade (integer between 4 and 10) and the exam grade as integer values. For regression, only the exam grade (which is a real number between 0 and 4) is used and it will be represented as a float number. Table 3.2 gives a structured view of how the students dataset labels are used in the studied models. For classifying the students into 4 classes based on their exam points the real continuous interval $(0, 4]$ was split into 4 equal subintervals and for each subinterval was assigned a class label (0, 1, 2 and 3, respectively).

Problem type	Label Description		
	Label type	Label Value	Label used
Binary Classification	Integer	0|1	Final grade
4-class Classification	Integer	0|1|2|3	Exam grade
Regression	Float	0.0-4.0	Exam grade

Table 3.2: Label Structure

3.1.2 Data Standardization

In Machine Learning, standardization of data is a common requierement for a lot of models (e.g.: neural networks and SVMs). This means that, before applying our learning models, we must first scale the features from our dataset. This scaling implies that the data should have **mean** 0 and **standard deviation** 1. This is done by subtracting the mean value of each feature, then scale the data by dividing the features by their standard deviation (or variance, because standard deviation is the square root of variance, so they are equal).

$$X_{scaled} = \frac{X - \overline{X}}{\sigma} \tag{3.1}$$

where X is a vector of values from one feature, $\overline{X}$ is the mean of X and σ represents the standard deviation of X.

In Figure 3.1[1] we can see a straight-forward visualization of how the data is represented before and after the preprocessing techniques:

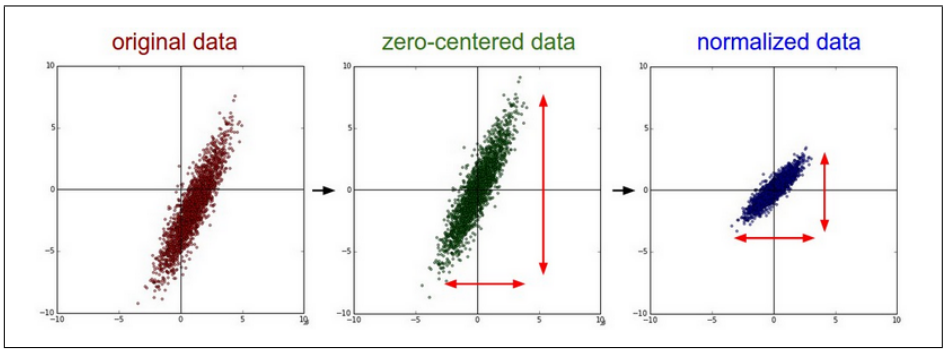

Figure 3.1: Data preprocessing pipeline

[1]http://cs231n.github.io/neural-networks-2

It is worth noting that the mean and standard deviation values must come only from the training data and the transformation must be done on both training and testing sets, for meaningful results. The reason is that, in general, the built Machine Learning model is applied on unseen data (on real-time data), which is not available when the model is built. So, for accurate calculation on the model's performance and generalization, we must restrict the computation of mean and variance only on the training examples.

3.2 Using Polynomial Features

3.2.1 Curse of Dimensionality

In general, the number of examples and the dimensionality of each example (the number of features per example) are correlated, taking into consideration the accuracy of the trained model. The *Hughes phenomenon*[9] tells us that if we have a constant number of training examples, the ability of the model's prediction decreses when the dimensionality increses over an optimal value. This is also called the **Curse of Dimensionality** and it can lead to *overfitting* the dataset - the model has a low power of generalization. Finding the best number of features can be a very hard problem (as it requires a lot of manual testing). Actually, this is an *intractable* problem, because we need to generate all possible combinations of features and find the optimal one. This could easily be avoided now by using Feature Selection tools. For example, **Random Forests** are very good models at selecting features that provide the best accuracy to the model. This will be analyzed with more details in the next sections of the thesis.

So, supposing that we have M number of examples in the training set and the feature tensor is one-dimensional, if we add another dimension to the tensor (another feature), ideally, we need to square the training examples. By induction, the number of training example grows exponentially with the dimension of the feature tensor.

Figure 3.2[2] shows a representation of how the number of feature dimensions

[2]http://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/

affects the quality of the learning model. It can be seen that keeping the training examples constant and only increasing the number of features, the accuracy drops by an exponential rate.

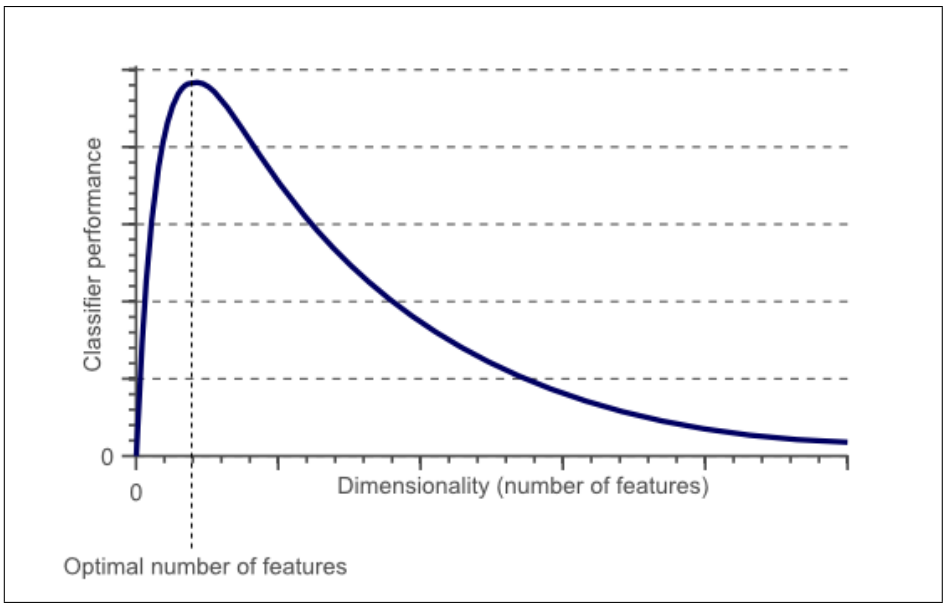

Figure 3.2: The Curse of Dimensionality

3.2.2 Adding Complexity to the Model

There are situations in which is better to add complexity to our features. For example, when our dataset in not linearly separable using one, two, or more features, we can add extra dimensions to the feature tensor in order to make that data separable. **SVMs** make use of this approach, by using multiple *kernel functions* that have the role to transform an input space in order to easily process data. Intuitively, a kernel is a "shortcut" that allows us to do certain computations, but without being directly involved in higher-dimensional calculations. Kernels can be linear functions, polynomial functions or even sigmoid functions. Using SVMs, we implicitly add complexity to our model.

Having tested some simple models on the dataset, such as Linear Regression or the Perceptron, adding some complexity to the models was not at all a bad idea. A good method was the one of using **Polynomial Features**, that combines the initial features into new nonlinear features. Polynomial Features adds more dimensions

to the feature space, but the key here is that they are correlated, so this can help in achieving better prediction accuracy. Here, there are two different options to consider:

- The first one is generating a list of features (a polynomial of a certain degree) from the current features. Example: If we have the input given as (X_1, X_2), after the polynomial tranformation the example becomes $(1, X_1, X_2, X_1 * X_2, X_1^2, X_2^2)$
- The second approach was to consider only interactions between the features for building a polynomial with the same degree as the number of initial features. Example: The features (X_1, X_2, X_3) are tranformed by the polynomial into: $(1, X_1, X_2, X_3, X_1 * X_2, X_1 * X_3, X_2 * X_3, X_1 * X_2 * X_3)$, resulting in a total of 2^N final features, where N is the original dimension of the input space.

So, for Linear Regression, the input features are transformed using the first method (to generate non-linear functions like polynomials of degree 2 or 3). This "trick" allows us to use simple linear models that are trained on actual non-linear combinations of the data and are faster than other complex non-linear models. Supposing that we want to train our student's dataset using Linear Regression with Polynomial Features:

Let $\tilde{y}$ be the output vector of the linear model, x the input tensor, $\omega \in \mathbb{R}^{M \cdot N}$ the coefficient tensor (a two-dimensional vector) and β the vector bias. The model computes the following equation, making use of the "least squares" method for calculating ω and β:

$$\tilde{y}(\omega, x) = \omega \cdot x + \beta \tag{3.2}$$

where $x = (x_1, x_2, \ldots, x_{10})$. When we add the polynomial features, x is transformed like this:

$$x_{nonlinear} = (x_1, x_2, \ldots, x_{10}, \ldots, x_i \cdot x_j, \ldots, x_1^2, x_2^2, \ldots, x_{10}^2); i, j = \overline{1, 10}, i < j$$

The size of the new feature space is: $10 + 10 + C_{10}^2 = 65$
Substituting $x_{nonlinear}$ in Equation 3.2, we obtain:

$$\tilde{y}(\omega', x_{nonlinear}) = \omega' \cdot x_{nonlinear} + \beta' \tag{3.3}$$

We can observe from the above equation that the linearity is still preserved and the model can fit more complicated data now.

On the other hand, the Perceptron model was tested using both methods, although the second method turned to be more appropiate, i.e. the *interaction features* method. This method was also used for the MLP[3] neural network model. With the size of the feature space of 10, adding interaction features provided a total of $2^{10} = 1024$ features.

3.3 Visualizing the dataset

[3]Multi-Layer Perceptron

Appendix A

Source code

A.1 Fișierele de configurare

A.1.1 Configurarea serviciilor

```
<?xml version="1.0" encoding="UTF-8"?>
<services-config>
  <services>
    <service-include file-path="remoting-config.xml" />
    <default-channels>
      <channel ref="my-amf"/>
    </default-channels>
  </services>
</services-config>
```

List of Figures

Bibliography

[1] Mehdi S. M. Sajjadi, Morteza Alamgir, Ulrike von Luxburg. *Peer Grading in a Course on Algorithms and Data Structures: Machine Learning Algorithms do not Improve over Simple Baselines.* In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale,* pp. 369-378, 2016.

[2] Siddharth Reddy, Igor Labutov, Thorsten Joachims. *Learning Representations of Student Knowledge and Educational Content. Cornell University,* 2015.

[3] Michael Wu. *The Synthetic Student: A Machine Learning Model to Simulate MOOC Data,* MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2015.

[4] Saeed Hosseini Teshnizi and Sayyed Mohhamad Taghi Ayatollahi. *A Comparison of Logistic Regression Model and Artificial Neural Networks in Predicting of Student's Academic Failure.* In *Acta Inform Med.,* 23(5): 296–300, 2015 Oct.

[5] Jeng-Fung Chen, Ho-Nien Hsieh and Quang Hung Do. *Predicting Student Academic Performance: A Comparison of Two Meta-Heuristic Algorithms Inspired by Cuckoo Birds for Training Neural Networks*. In *Algorithms,* 7(4), 538-553, 2014.

[6] B.A., Kalejaye, O., Folorunso and O.L., Usman. *Predicting Students' Grade Scores Using Training Functions of Artificial Neural Network*. In *Journal of Natural Sciences, Engineering and Technology, Volume 14,* 2015.

[7] V.O. Oladokun, Ph.D., A.T. Adebanjo, B.Sc., and O.E. Charles-Owaba, Ph.D. *Predicting Students' Academic Performance using Artificial Neural Network: A Case Study of an Engineering Course*. In *The Pacific Journal of Science and Technology, Volume 9,* pp. 72-79, 2008.

[8] Emaan Abdul Majeed and Khurum Nazir Junejo. *GRADE PREDICTION USING SUPERVISED MACHINE LEARNING TECHNIQUES.* In *E-Proceedin*

of the 4th Global Summit on Education, pp. 224-234, 2016.

[9] G. Hughes. *On the mean accuracy of statistical pattern recognizers*. *In IEEE Transactions on Information Theory, Volume 14*, pp. 55-63, 1968.