Machine Learning with Salesforce and Apache PredictionIO (incubating) in the Academic World

Luciano Straga, *Saleforce Certified Professional, University of Palermo (Argentina)*

***Abstract*—Artificial Intelligence and Machine Learning have become trendy in software development.Is not a matter only of data scientists or mathematicians. Apache PredictionIO makes Machine Learning available for any software developer from different technologies and backgrounds. This paper demostrates how PredictioIO, in conjunction with Salesforce can introduce the concept of Prediction as a Service, and help to solve problematics of the academic world.**

***Index Terms*—AI, Aritifitial Inteligence, Machine Learning, PredictionIO (incubating), Spark, Salesforce, Heroku, Heroku Connect, Heroku Postgres, Scikit Learn.**

I. Introduction

THE purpose of this paper is to demonstrate that tasks and problematics of the Academic World could be handled with the smart use of Machine Learning [2][3][4][5]. Administrative tasks, such as calculate and determine the number courses to open about an specific subject, based in historical data; or infer the number of students that will pass an exam based on its background are machine learning problems.

A. The Challenges

The core of Machine Learning, are its powerful algorithms that build models that can predict reality with certain grade of accuracy. Most of machine learning algorithms are not new, and have statistics and maths fundamentals that are sometimes complex for non scientists users or technical users.

Each algorithm is useful for a specific problem; so the key lies on choosing the right algorithm. But it is just only the first challenge. The play actually begins at the moment of develop a scalable, efficient and effective strategy for feeding the algorithms. These days are Big Data[6] days, so the whole Machine Learning Framework has to be solid and consistant, and Big Data ready.

B. The desired formula

We aim to build a simple and stable enviroment that could handle tons of terabytes, that has to be scalable for even more data, in which machine learning algorithms could be easily available for third parties; and finally with external and non-technical users that may take advantage of all of this from the intuitive software that they use and love.

The approach: Salesforce[7] + Apache Spark[8][9] + PredictionIO (incubating)[10][11]

Luciano Straga(26)[1] is a software developer form Argentina, working with Salesforce(4X Certified) since 2012. Two times speaker at Dreamforce Conference, San Francisco, USA (2015-2016). University of Palermo, Buenos Aires.

E-mail: lucianostraga@icloud.com

C. Preceding Projects

There are no previous projects listed about PredictionIO involved in specific academic tasks, or formal and documented intents of using machine learning for solving and predicting that kind of problems, intrinsic to the academic world. There are a couple of papers[12] about presenting PredictionIO as practical machine learning framework, but applied to the academic field.

Despite the lack of formal intents, there are many websites that put this necessity on the table. OnlineUniversities.com[13] presents Ten Ways Artificial Intelligence Can Reinvent Education. The same way TimesHigherEducation[14] enumerates Four ways that artificial intelligence can benefit universities. Both basically summarize different areas in which artificial intelligence may introduce drastical changes in classic education tasks. Some of theme are:

- Artificial intelligence can automate basic activities in education, like grading
- Students could get additional support from AI tutors
- AI-driven programs can give students and educators helpful feedback
- AI can make trial-and-error learning less intimidating
- It could change the role of teachers

II. Machine Learning for Universities

Every quarter, University of Palermo[15], needs to estimate the number of courses that have to be opened for fulfilling students needs.

A. The Problematic to Solve

Across all Engineering careers and related others, University of Palermo has a big concern. Last years we have detected a worrying **dropout rate in a specific subject:** *Programming Fundamentals*.

This course is an introduction to the basics of programming and coding. It brings students the base knowledge for start programming, like coding structures, *if-else* conditionings, *for-while* sentences, etc. based in the Python programming language. This introductory course is usually easy for students that have already programming skills. However, might be a nightmare for non-experienced students or with some weak math-logic skills. One of the worst last examples, are courses in which the drop-out rate exceeds the 50% after the first exam and students that are repeating for 4 or 5 times. We needed to rethink how to group students in new courses, by predicting in advance drop-out rates, based on their skills and background. Machine Learning algorithms will be the tool.

B. An initial internal survey to feed the algorithm

We created an internal survey targeted to students of Engineering careers that include that course. We evaluated the following eight variables:

- Sex
- Engineering career
- Mathematics performance at High School
- If they like programming (coding)
- If they learned to code at High School
- If their job is related with Software Development
- How many times they enrolled into Programming Fundamentals
- **If they already completed the course and passed the final exam**

The survey covered a total of 104 students. The pending steps were: **choosing a reliable Machine Learning Algorithm** that could build a predictive model, **deciding carefully the number of variables to use** (twelve originally) to get the most powerful and accurate model, and **designing a data transformation** to the data obtained from the survey.

III. The Naive Bayes Algorithm

Since the need of estimate the dropout rate of a course, we first need to estimate the dropout rate of each student, and that is classification task. In other words, classify each student (predict the completion or not of the course).

Naive Bayes[16] algorithm is a supervised classification[17] algorithm based on Bayes Theorem[18]. There is particular property of this algorithm that makes it fit perfectly in our use case. **Naive Bayes assumes the independence of all the features involved, that predicts the label**. The presence of a single feature in a class is unrelated to the presence of any other feature. A Naive Bayes predictive model is easy to build and understand, and often performs better than other sophisticated classification techniques.

A. Naive Bayes Representation Model

The model is represented by probabilities.

- **Class Probabilities**: The probabilities of each class in the training dataset.
- **Conditional Probabilities**: The conditional probabilities of each input value given each class value (likelihood).

Training a Naive Bayes model is fast. Class probabilities and conditional probabilities are the only calculation needed. No coefficients need to be fitted by other optimization procedures.

Naive Bayes predictive model works with the calculation of the posterior probability, based on Bayes Rule.

B. Our Use Case Model

The data collected in the survey is composed mostly by real independent variables. For instance, *Sex* and *Mathematics Performance at High School* are not related. Variables in Machine Learning problems are called **features**, and the predicted values are called **labels**. There are different Naive Bayes models, like the Gaussian[16], Bernoulli[16] and Multinomial[16]; usually based on type of features and its distribution. **We will use the Multinomial model, since our features may take different values**.

IV. Apache PredicitonIO (Incubating)

Apache PredictionIO is an **open source machine learning server designed for developers and data scientists**. PredictionIO was acquired by Salesforce and donated to the Apache Software Foundation in 2016; At this moment, is part of an incubator program, surrounded by other industry-leading open source projects such as Apache Hadoop[19][20], Apache Spark, Apache Cassandra[21][22] and Apache Lucene[23][24].

Prediction IO follows the *As a Service*[**?**] philosophy; in this case, Prediction as a service. It is conceived to be a convenient **framework for creating and deploying predictive engines for any machine learning task**.

A. PredictionIO Basic Features

PredictionIO allows developers and data scientist to:

- Build and deploy an engine as a web service based on customizable templates
- Respond to dynamic queries inreal-timeonce deployed as a web service
- Evaluate multiple engines systematically
- Unify data from multiple platforms in batch or in real-time
- Support Machine Learning libraries such as Spark MLLib and OpenNLP
- Bundle the server with Apache Spark, MLlib, HBase, Spray and Elasticsearch

B. PredictionIO Components

PredictionIO is composed by the following components:

- PredictionIO Platform
- Event Server
- Engines Deployed Based on customizable templates

1) PredictionIO Platform:

Open source machine learning stack for building, evaluating and deploying engines with machine learning algorithms.

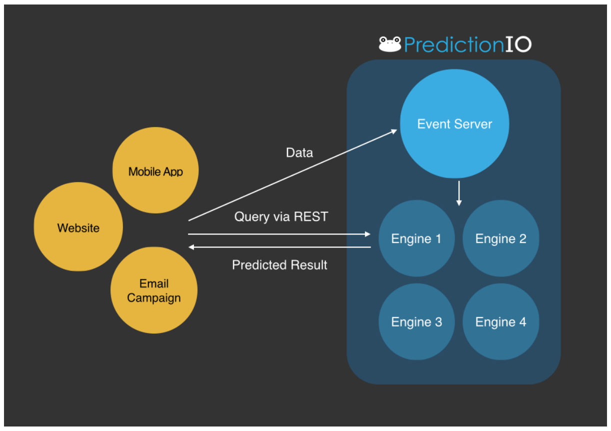

Fig. 1. PredictionIO Components

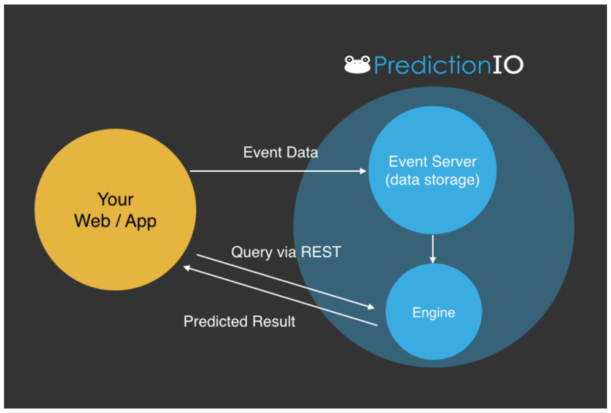

Fig. 2. Event Server

2) Event Server:
Event Server collects data from different sources and can host different applications, in real-time or in batch. It can also unify data from multiple platforms. After data is collected, it mainly serves two purposes:

- Provide data to Engine(s) for model training and evaluation
- Offer a unified view for data analysis

3) Engine:
Engine is responsible for making prediction. It contains one or more machine learning algorithms. An engine reads training data and build predictive model. It is then deployed as a web service and responds to prediction queries through REST API in real-time. PredictionIO's template gallery offers Engine Templates for all kinds of machine learning tasks.

The components of a template, namely Data Source, Data Preparator, Algorithm(s), and Serving, are all customizable for your specific needs.

C. PredictionIO Architecture

Apache PredictionIO works in conjunction with **Apache Spark**. Spark is a large-scale data processing framework that powers the data preparation and input to the algorithm, training, and serving processing. PredictionIO allows for different engines to be used in training but many algorithms come from Spark's MLlib. For data store, we decided to simply by using PostgreSQL 9.1, but there are more complex schemas with HBase and Elasticsearch. Apache Hadoop is needed since Spark uses HDFS as file system, regardless the way that Spark is implemented.

1) Spark MLlib:
MLlib [25] is a machine learning library provided as a module with Spark. Includes common learning algorithms such as classification, regression, clustering, and collaborative filtering. Most of the templates provided by Apache PredictionIO are based on MLlib algorithm. **In this case, the Naive Bayes Algorithm we used to build the predictive model**. Spark MLlib's algorithms supports RDD[26](Resilient Distributed Datasets) and DataFrames[27] APIs, and this is one of the key reasons about why are they convenient for large-scale processing. PredictionIO

V. Building the Model

After concluded the survey, we obtained **104 study cases (observations)**. The collected data is composed by seven features and one label and in a plain text file.

A. Feature Selection

Initially the survey contained eleven features. The first and mandatory step was **removing dependent features based on Naive Bayes independence nature**. Total independence between features is utopian and unreal. However, we tried to strongly avoid correlated features. After obtaining an independent featured model, we proceed to **discard features that were not relevant** and dont expose significant difference and weight between occurrences. For instance, the original survey evaluated if the person knows about the concept of software development. We obtained within the whole universe a 99% of positive answers, so we decided to not to consider this feature. Common sense in this stage is the key. Finally, and probably the most difficult problem, was to deal with the **zero-frequency problem intrinsic to Naive Bayes**[28]. One of algorithm weakness, is the scenario in which there are no combinations of specific features for obtaining specific labels. This brings a probability zero for the missing combination and affects negatively the performance of the predictive model.

B. Training the model

All the features collected were categorical. Naive Bayes algorithm is commonly used in text classification and performs well with categorical data. Apache Spark RDDs used by MLlib algorithms usually works with Labeled Points[29]. Labeled Points are a tuple that contains one label and a Vector of features. Vectors of features and labels in Labeled Points are of data type Double. **We needed to assign each of the seven features to numeric value**.

TABLE I
FEATURES CODIFICATION

Categorical Features	Answers	Code
Sex	Male	0
	Female	1
	Other	2
Career	Electronic degree	0
	Industrial degree	1
	Informatics degree	2
	Informatics bachelor	3
	Communications and Networks bachelor	4
	Informatics Systems Admin. bachelor	5
	Technology of Information bachelor	6
Mathematics perf.	Excellent	0
	Very good	1
	Regular	2
	Bad	3
Like Coding?	No	0
	Yes	1
Code at High School?	No	0
	Yes	1
Code at work?	No	0
	Yes	1
Times coursing (#)	1-10	1-10

TABLE II
LABELS CODIFICATION

Categorical Label	Code
Passed Course	0
Failed Course	1

C. Cross Validation & Accuracy[**?**]

In order to evaluate the accuracy of the predictive model, the complete data set was **splitted in training set and a test set, with a 80-20 ratio**. We did ten random splits with an **average accuracy of 81.08%**. This score, for a data set of 104 observation is more than acceptable. This evaluation task was accomplished with Apache Spark, by converting the raw data set, and converting in different RDDs. The following code snippets in python illustrate the task.

```
sc = SparkContext("local",
        "Cross␣Validation")

def parseLine(line):
    label = float(...)
    features = Vectors.dense(...)
    return LabeledPoint(label, features)

data = sc.textFile('DATASET_PATH')
                                .map(parseLine)

training, test =
    data.randomSplit([0.8, 0.2], seed=x)
```

Once the data set is splitted, the next step is to train the algorithm and to evaluate its performance.

VI. ANALYZING THE PREDICTING MODEL BUILT

Our model was trained with a data set of seven features. Theres no graphical representation possible above three dimen-

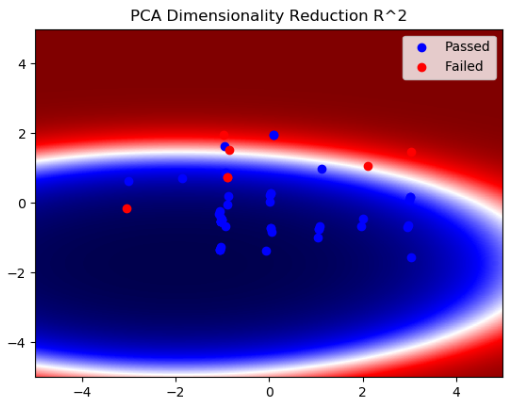

Fig. 3. Decision boundary calculated with PCA of a dataset splitted randomly with accuracy 0.79

sion. For that reason, we needed to perform a **dimensionality reduction**[30] to start interpreting the model we built. Our first metric (Figure 3) was to considering the **decision boundary of the algorithm**. The decision boundary shows if the points (combination of features) were correctly classified in the correct classes (labels). Analyzing a decision boundary is useful to contrast the accuracy score obtained by model. All of the following analytics, as well as the dimensionality reductions, were created with **Scikit Learn Pyhthon library**[31]. Dimensionality reductions over the data set were transformations to (x,y) two dimensionals plain. In plots without transformation, superposition of points with different labels makes the color vary.

A. Dimensionality Reduction with PCA

Principal Components Analysis (**PCA**)[32] is one of the most common dimensionality reduction techniques(Linear Discriminant Analysis). Similar to LDA, Principal Components Analysis (Figre 4) works best on linear data, but with the benefit of being an unsupervised method. PCA produces a set of linearly uncorrelated variables called principal components. The first component is determined by its contribution to the greatest variance in the data. Then all subsequent components are found by the same greatest-variability constraint, while also being orthogonal to the previous one. PCA has a close mathematical relationship to other popular methods such as factor analysis and k-Means Clustering. **PCA transformation output usually is two-dimensional features with negative values**. Multinomial Naive Bayes algorithm provided by Apache Spark does not support negative terms. For this case, we used the **Gaussian Naive Bayes algorithm available on Scikit Learn library** which expects a continuous input. Only multinomial and Bernoulli models are available in Apache Spark.

B. Dimensionality Reduction with t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding)[33] is a machine learning algorithm for dimensionality reduction.

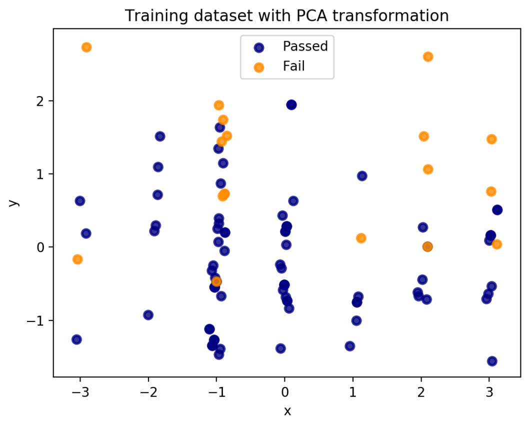

Fig. 4. The complete training dataset after PCA transformation

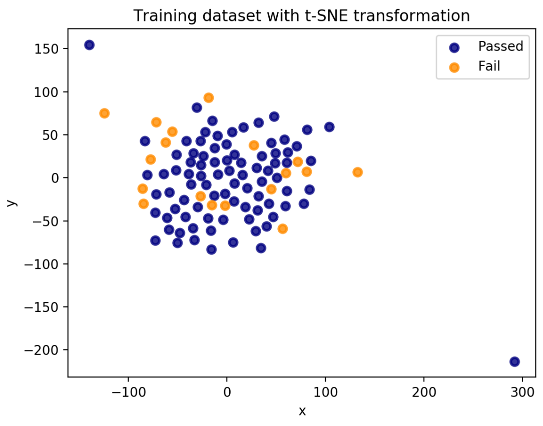

Fig. 5. The complete training dataset after t-SNE transformation

t-SNE is a non-linear method and performs better on data where the underlying relationship is not linear. It is particularly well-suited for embedding high-dimensional data into a space of two or three dimensions, which can then be visualized in a scatter plot. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points. As well as PCA usuarlly, t-SNE (Figure 5) returns negative features and could not be used to train a multinomial model.

C. Intercepting Features

Sometimes, is interesting to **isolate certain features and evaluating them in pairs. No transformation is necessary for analytics, if we reduced the training dataset to two features**. In addition, without adding a transformation, we keep the significance of that feature and the analysis makes more sense. The following examples are show what we obtained by isolating some features:

1) Sex VS Maths Performance at Hight School: Some curious pattern was discovered. As shown on the picture (Figure 6), only males (x=0) with excellent performance in

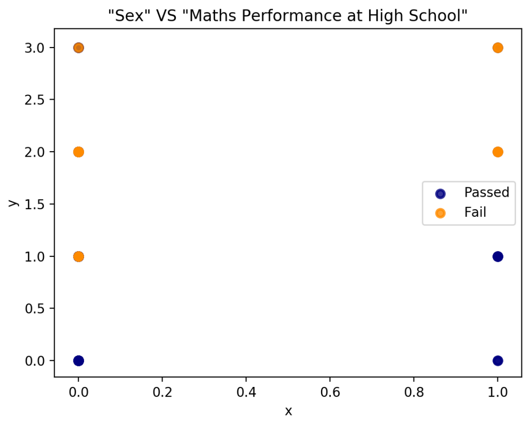

Fig. 6. Indicates the vinculation of 'Sex' VS 'Maths Performance' features.

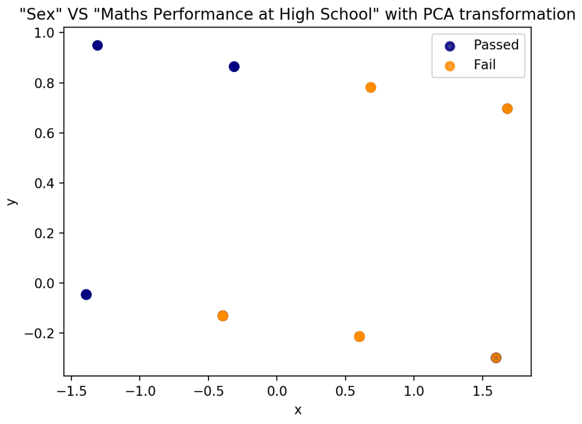

Fig. 7. Indicates the rotation of the features performed by PCA.

Mathematics at High school passes the course. Males with lower performance used to fail. The females pass the course has excellent or very good performance at mathematics.

2) Sex VS Maths Performance at Hight School with PCA: The picture (Figure 7) exposes the transformation made by PCA, and represented as a rotation. Categorical features can't be analyzed this way but the pattern is highly similar to the original dataset.

3) Sex VS Maths Performance at Hight School with t-SNE: The picture (Figure 8) exposes the transformation made by t-SNE. In this case, since the method is not linear, the dataset plotted looks totally different.

4) Times Coursed VS Career: The picture (Figure 9) is one of the most expressive. Indicates the tendency of not passing the course after the second attempt regardless the career of the student.

5) Times Coursed VS Career with PCA: Again, the pictures (Figure 10) shows the rotation performed by PCA.

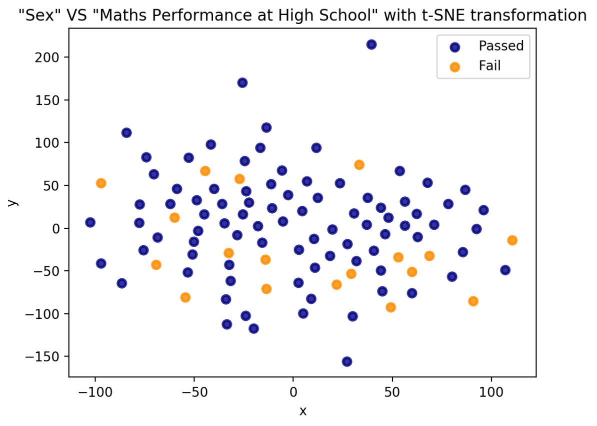

Fig. 8. Transformation of th features performed by TSNE.

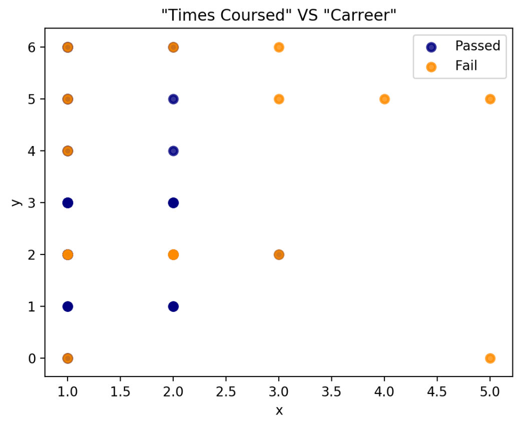

Fig. 9. Times coursing and Career type features

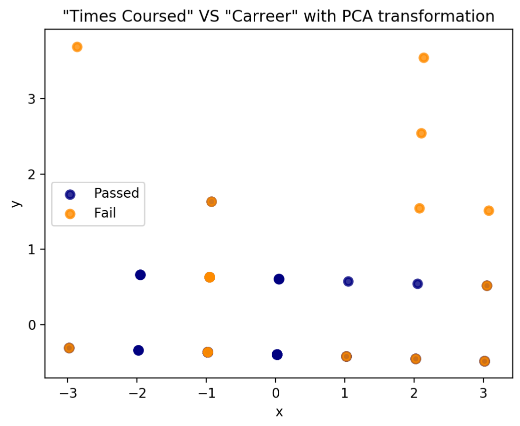

Fig. 10. Indicates the rotation of the features performed by PCA.

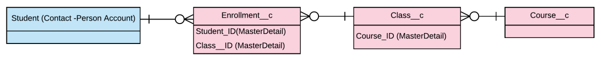

Fig. 11. Simplified data model in Salesforce. Standard objects in blue, custom objects in pink

D. *Dimensionality Reduction and Accuracy*

After getting some insights about the model, by applying dimensionality reduction to the dataset, we decide evaluate if PCA and t-SNE could help us to improve our original accuracy. We knew that we could not train Apache Spark multinomial Naive Bayes model with a negative and continuous input, but we decided to study how would perform in the Scikit Learn Gaussian model. The dataset splits were the same for all the techniques, and generated randomly ten times.

With PCA we obtained a slightly improvement, obtaining an average score of 85.7%. **With t-SNE the improvement was radical. We obtained an average score 90.4%**. Basically, both transformations improved the prediction accuracy of the model.

VII. The Salesforce layer

Salesforce is cloud computing PAAS (Platform as a Service).Known as the world leader CRM and development platform for enterprise applications, **Salesforce works as an interface for interacting with the predictive model**. Since its user adoption, Salesforce is ideal software and platform for interacting with PredictionIO, not only making predictive queries, but also for feeding and improving the model.

A. *University of Palermo simplified data model*

University of Palermo is evaluating Salesforce in a pilot program, for account relationship management and students tracking. Basic standard objects are slightly customized to store students information, and other core business entities are modeled easily with custom objects (Figure 11).

B. *From Salesforce to Apache PredictionIO*

There are different ways to integrate Salesforce with the predictive model. For this prototype we integrated Salesforce for **consuming the predictive model (sending queries to PredictionIO) and for training the model**.

1) *Live Queries to PredictionIO:*

Since business requirements demand, Salesforce needs to consume synchronously the PredictionIO services for making predictions in real time. Taking advantage of PredictionIO architecture, **Salesforce interacts with PredictionIO predictive engines via its REST APIs**. **University of Palermo designed a business flow for launching classes before confirming enrollments. With a combination of Apex triggers and callouts, Salesforce predicts the success or failures of all the enrolments and groups them in a smarter way. Basically, Salesforce balances students enrollments so as to control dropout rate in advance**.

2) Training the model:
For this goal there are several alternatives: The easiest way is to **interact with PredictionIO Event Server via REST callouts** as did before for consuming services.

Another workaround is by using **Heroku Connect with Heroku Postgres**[34]. Since PredictionIO support PostgreSQL for storing events, Salesforce could connect with a Heroku Postgres cloud database in a point and click integration with Heroku Connect. **Salesforce can map the features an labels in external objects** and synchronize them with Salesforce in a seamless way. Even **PredictionIO might also be hosted in Heroku as a Scala app**, although Univesity of Palermo decided to host it privately and public some specific domain to REST APIs interaction.

In a more complex integration schema, in which we expect high data volumes, we may imagine Salesforce as streamer of training data. Recently, **Salesforce in Summer '17 released Platform Events**[35]. This is a different solution, similar to Streaming API, in which with a custom development on the PredictionIO side, you can turn PredictionIO as subscriber of Salesforce topics and make it learn constantly.

Finally, other simpler approaches might be **manual exports based on text files and import scripts**, in a non automated way.

VIII. Conclusion

After concluding and during this project, we learned important things.

First of all PredictionIO is a fascinating place for start working and being involved with machine learning; specially if you don't have previous experience. As a Salesforce professional, I discovered PredictionIO within the App Cloud echosystem, and lets you make your first baby steps with artificitial inteligence without technical science knowledge. So, from the PredictionIO perspective, the experience was so fruitfull, and it is notorious that is designed for developers that are starting with machine learning. Apart from that, since its core is Apache Spark, its framework is super solid and scalable for big data processing. It gives the 'as a service touch' that machine learning was needing to be more popular and available to every one. Personally, I believe that the Spark MLlib is a great tool that fits perfect with big data needs and structures (Data Frames and RDDs). However, it cannot be compared with other machine learning libraries, such us Scikit Learn for Python. The variety of algorithm is bigger and documentation is easier and better. Since, from PredictionIO is possible to use another algorithms from external libraries, that task is out of scope of this project and definitely not easy. A gaussian Naive Bayes model would have worked better for fitting PCA and t-SNE training sets. Any way, it is important to mention, that PredictionIO is an incubating project, so we expect more enhancements, and more templates sooner.

One of things that took us the attention, was the fact that there were not so much precedents about using artificitial inteligence for universities administrative flows. So, that made this project more challenging.

Finally, the use of Salesforce was natural. Is the platform and the technology I know, and the most accurate place so as to interact with PredictionIO. Users fell confident about Salesforce, and that means easier adoption.

Acknowledgment

I really would like to thank to University of Palermo, for letting me design and consider this prototype, with the tools that I choose and recommend, for trusting me and bring me the necessary support to distribute our initial survey and start this project.

References

[1] LinkedIn, "Luciano Straga," https://www.linkedin.com/in/luciano-straga-121a9167/, 2017.
[2] Wikipedia, "Machine Learning," https://en.wikipedia.org/wiki/Machine_learning, 2017.
[3] V. V. DataCamp, "Introduction to Machine Learning," https://www.datacamp.com/courses/introduction-to-machine-learning-with-r, 2017, [Free Online Course].
[4] Udacity, "Intro to Machine Learning," https://www.udacity.com/course/intro-to-machine-learning--ud120, 2017, [Free Online Course].
[5] S. University, "Machine Learning," https://www.coursera.org/learn/machine-learning, 2017, [Free Online Course in Coursera].
[6] Wikipedia, "Big Data," https://es.wikipedia.org/wiki/Big_data, 2017.
[7] Salesforce.com, "What is Salesforce?" https://www.salesforce.com/products/what-is-salesforce/, 2017.
[8] Wikipedia, "Apache Spark," https://en.wikipedia.org/wiki/Apache_Spark, 2017.
[9] A. S. Fundation, "Apache Spark," https://spark.apache.org/, 2017.
[10] A. Wiki, "PredictionIO," https://wiki.apache.org/incubator/PredictionIO, 2016.
[11] A. S. Fundation, "Welcome to Apache PredictionIO (incubating)!" https://predictionio.incubator.apache.org/, 2017.
[12] S. Chan, T. Stone, K. P. Szeto, and K. H. Chan, "Predictionio: a distributed machine learning server for practical software development," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, pp. 2493–2496.
[13] OnlineUniversities.com, "10 Ways Artificial Intelligence Can Reinvent Education," http://www.onlineuniversities.com/blog/2012/10/10-ways-artificial-intelligence-can-reinvent-education/, 2012, [Online; accessed 30-October-2012].
[14] TimesHigherEducation.com, "Four ways that artificial intelligence can benefit universities," https://www.timeshighereducation.com/blog/four-ways-artificial-intelligence-can-benefit-universities, 2016, [Online; accessed 9-August-2016].
[15] U. de Palermo, "Univesidad de Palermo, Buenos Aires Argentina," http://www.palermo.edu/, 2017.
[16] Wikipedia, "Naive Bayes classifier," https://en.wikipedia.org/wiki/Naive_Bayes_classifier, 2017.
[17] ——, "Supervised Learning," https://en.wikipedia.org/wiki/Supervised_learning, 2017.
[18] ——, "Bayes' theorem," https://en.wikipedia.org/wiki/Bayes%27_theorem, 2017.
[19] ——, "Apache Hadoop," https://en.wikipedia.org/wiki/Apache_Hadoop, 2017.
[20] A. S. Fundation, "Welcome to Apache Hadoop!" http://hadoop.apache.org/, 2017.
[21] Wikipedia, "Apache Cassandra," https://en.wikipedia.org/wiki/Apache_Cassandra, 2017.
[22] A. S. Fundation, "Apache Cassandra," http://cassandra.apache.org/, 2016.
[23] ——, "Apache Lucene Corea," https://lucene.apache.org/core/, 2016.
[24] Wikipedia, "Apache Cassandra," https://en.wikipedia.org/wiki/Apache_Lucene, 2017.
[25] A. Spark, "Machine Learning Library (MLlib)," https://spark.apache.org/docs/latest/ml-guide.html, 2017.
[26] T. Point, "Resilient Distributed Datasets," https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm, 2014.
[27] A. Spark, "Datasets and DataFrames," https://spark.apache.org/docs/latest/sql-programming-guide.html#datasets-and-dataframes, 2017.
[28] D. S. Sayad, "Naive Bayesian," http://www.saedsayad.com/naive_bayesian.htm, 2014.

[29] A. Spark, “Labeled point, howpublished = ”https://spark.apache.org/docs/latest/mllib-data-types.html#labeled-point”, year = 2017.”
[30] Wikipedia, “Dimensionality reduction,” https://en.wikipedia.org/wiki/Dimensionality_reduction, 2017.
[31] S. Learn, “scikit-learn,” http://scikit-learn.org/stable/, 2016.
[32] Wikipedia, “Principal component analysis,” https://en.wikipedia.org/wiki/Principal_component_analysis, 2017.
[33] ——, “t-distributed stochastic neighbor embedding,” https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding, 2017.
[34] Salesforce, “Heroku Connect,” https://www.heroku.com/connect, 2017.
[35] ——, “Platform Events Developer Guide,” https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_intro.htm, 2017.