AI APPLIED IN RAISING CATTLE PREDICTION

DEMIAN Horia
University of Oradea (Management Marketing Department, Faculty of Economics), Oradea, Romania
horia_demian@yahoo.com

Abstract: *This paper presents the results obtained from the testing of an algorithm based on artificial intelligence applied to data on cattle breeding, over a period of several years. Following the application of this algorithm can be made predictions regarding the weight gain of cattle according to their age, their type, the input weight and the number of days we propose to keep him fattening. Prediction can help us make decisions about future sales contracts, simply by the fact that we can know a weight that we can achieve for existing cattle after a certain number of days.*

Keywords: *Artificial Intelligence; raising cattle.*

JEL Classification: *D83.*

1. Introduction

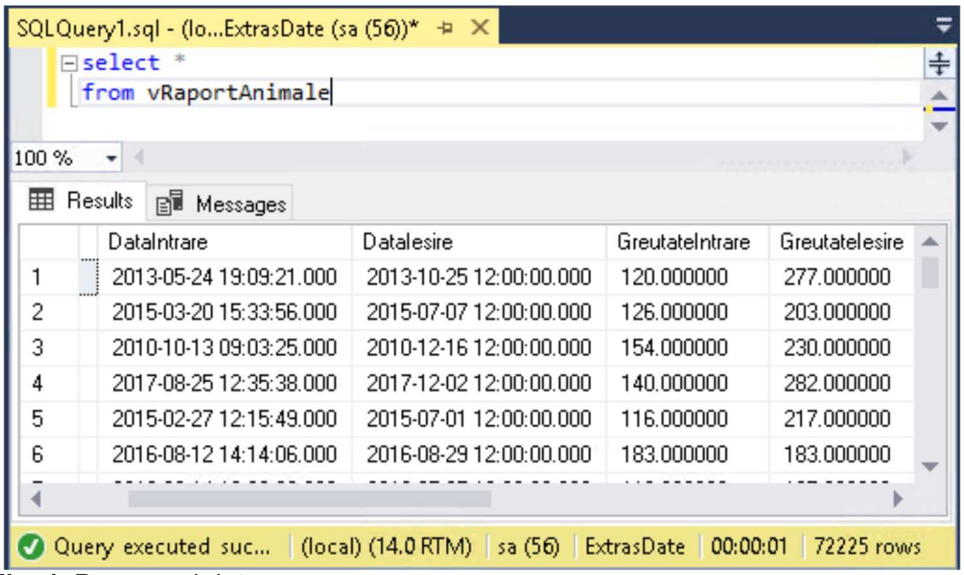
In livestock farms, where their purchase is made for fattening and subsequent sale, it is important to know when we make the purchase, depending on the breed, gender, age and weight of acquisition what the yield we can achieve for this calf in a period of time. Depending on this forecast yield we can also make a price offer for its purchase. Most farms currently use a price grid per kilogram depending on the type of calf and the input weight without taking into account the breed and age. By using artificial intelligence we propose that based on the data available in such a farm we can make predictions on the individual evolution of a calf.

2. Machine learning

2.1. Getting the data that participates in the learning process

It was started by identifying the objective, namely the prediction of the weight that a calf can achieve according to race, gender, input weight and age in a certain period of time, in order to refine the nomenclature used in their acquisition.
Following the identification of this objective, we continued with the identification, extraction and processing of data. In the first phase 72,225 cattle for which the date of birth is known, gender, input weight, weight at the time of sale and number of fattening days.
For each calf at the time of its acquisition, data on the date of birth, the birth-tag, the genus, the breed and its mass weight on the scale are processed. All this data is operated and stored in the database. For the calculation of the fattening period and the weight that the calf accumulated over a period of time, the date on which the calf was sold and its weight at the time of sale, obtained from a weight, were taken into account. This data was extracted from the sales invoice.

We continued with the identification of the cattle for which we have all the necessary data in the learning process, both inbound and outgoing. This data was included in a new database, in a backgammon, only with the necessary information: Veal Birth-tag, Date of acquisition (date of receipt note), Date of exit (date of sales invoice), Input weight (Weight weighed at the time of receipt), Weight of exit (weight weighed at the time of sale), Sex, Race and Date of Birth (from the animal's passport). The database was created using the Microsoft SQL Server database server.



```
select *
from vRaportAnimale
```

	DataIntrare	DataIesire	GreutateIntrare	GreutateIesire
1	2013-05-24 19:09:21.000	2013-10-25 12:00:00.000	120.000000	277.000000
2	2015-03-20 15:33:56.000	2015-07-07 12:00:00.000	126.000000	203.000000
3	2010-10-13 09:03:25.000	2010-12-16 12:00:00.000	154.000000	230.000000
4	2017-08-25 12:35:38.000	2017-12-02 12:00:00.000	140.000000	282.000000
5	2015-02-27 12:15:49.000	2015-07-01 12:00:00.000	116.000000	217.000000
6	2016-08-12 14:14:06.000	2016-08-29 12:00:00.000	183.000000	183.000000

Query executed suc... | (local) (14.0 RTM) | sa (56) | ExtrasDate | 00:00:01 | 72225 rows

Fig. 1. Processed data

We also tried to identify particular cases, since on the farm there were also animals bought not only for fattening. Some animals participate in the reproductive process, or are large enough to be resold directly, without the need to go through the fattening process. We also wanted to study how filtering this data leads to an improvement in learning and in what way. We started from the main objective of the company namely the purchase of cattle up to one year old and weights up to 300 kg.

2.2. Developing the software

To software developing we have used C# and the existing AI libraries. Not being a commercial software, we placed the necessary objects in a single interface. This Interface has two functionalities namely: Learning and Application of what we have learned.

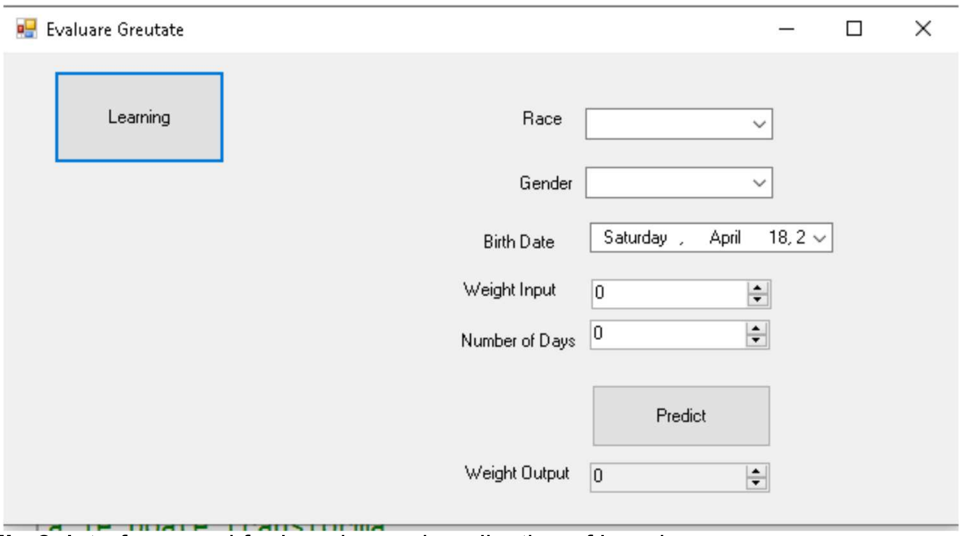


Fig.2. Interface used for learning and application of learning

On the left side we placed the Generation Model button which serves as an application and testing of learning. The right side is used to make a prediction based on the data specified here. It specifies the race, the sex, the date of birth, the input weight and the number of days we intend to keep fattening, until the moment of a possible sale. After completing this data, after pressing the predict button you get the weight that the calf will have at the end of this fattening process.
The algorithm chosen for the prediction was the FastTree algorithm. As can be observed some of the characteristics that will be taken into account for learning contain numerical values. These features are Entry Weight and Number of Days between the date of entry and the date of sale of the animal.
In the case of the date of birth, in order to convert it into a number we took into account the number of days obtained from the difference between the date of purchase and the date of birth of the calf expressed in days. Thus the date of birth was converted into a numerical value.
For the Gender feature that contains only two values, we used OneHotEncoding to convert data.
In the case of our data for the Breed feature because we have only two breeds of animals purchased we have also used OneHotEncoding to convert them into numeric values.
We have defined the Bull and WeightPrediction class as follows:

```
public class Bull{
[LoadColumn(0)]
public float WeightInput { get; set; }
[LoadColumn(1)]
public float WeightOutput { get; set; }
[LoadColumn(2)]
```

```
public class WeightPrediction
 {
  [ColumnName("Score")]
  public float WeightOutput { get; set; }
 }
 public float DaysNumber { get; set; }
 [LoadColumn(3)]
 public string Gender { get; set; }
 [LoadColumn(4)]
 public string Race { get; set; }
 [LoadColumn(5)]
 public float DaysSinceBirthDate { get; set; }
 public string DataIntrare { get; set; }
 public string DataIesire { get; set; } }
```

As can be seen the characteristics of this class that participate in the learning process are WeightInput, WeightOutput, DaysNumber, Gender, Race and DaysSinceBirthDate.
Effective learning involves starting from an existing list of cattle in a list defined as follows: List<ModelDate.Bull> lstBulls = new List<ModelDate.Bull>();
The list of elements was initialized with the data obtained by reading from the database in a SQLDataReader object.The actual learning involves normalizing data, training data for learning and testing, applying the learning algorithm and then applying the testing to the test.

```
 var pipeline = context.Transforms.CopyColumns(outputColumnName: "Label", inputColumnName: "WeightOutput")
   .Append(context.Transforms.Categorical.OneHotEncoding("GenderOneHot", "Gender"))
   .Append(context.Transforms.Categorical.OneHotEncoding("RaceOneHot", "Race"))
   .Append(context.Transforms.Concatenate("Features", "WeightInput", "DaysNumber", "DaysSinceBirthDate", "GenderOneHot", "RaceOneHot"))
  .Append(context.Regression.Trainers.FastTree());

IDataView trainData;
IDataView testData;

var split = context.Data.TrainTestSplit(data, testFraction: 0.1);

//obtaining data used for trainning
trainData = context.Data.LoadFromEnumerable(context.Data.CreateEnumerable<ModelDate.Bull>(split.TrainSet, reuseRowObject: false));
//obtaining the data used for testing
testData = context.Data.LoadFromEnumerable(context.Data.CreateEnumerable<ModelDate.Bull>(split.TestSet, reuseRowObject: false));

//begin the process of learning
var model = pipeline.Fit(trainData);
//save the model
context.Model.Save(model, data.Schema, _modelPath);

//verifying
```

```
RegressionMetrics trainedModelMetrics =
context.Regression.Evaluate(model.Transform(testData));
double rSquared = trainedModelMetrics.RSquared;
```

From the total amount of data we have user 10% for testing and rSquare calculation.

No.	Data set	Condition for data filtering				Records count	Dimension of output learned model	Learning time (sec)	RSquare
		Input Weight		Age in days at purchase time					
		Min	Max	Min	Max				
1	complete	-	-	-	-	72 225	82KB	4.5858125	0.77297
2	filtered	40	300	-	-	60832	82KB	4.2149553	0.75561
3	filtered	-	-	7	365	54491	82KB	3.8190093	0.71919
4	filtered	40	300	7	365	44734	82KB	3.7729948	0.67347

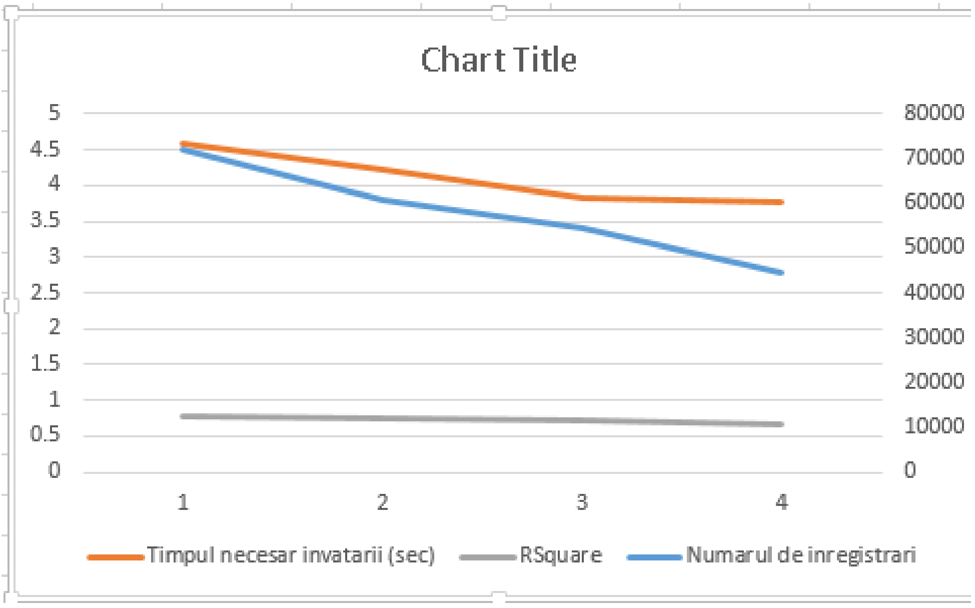
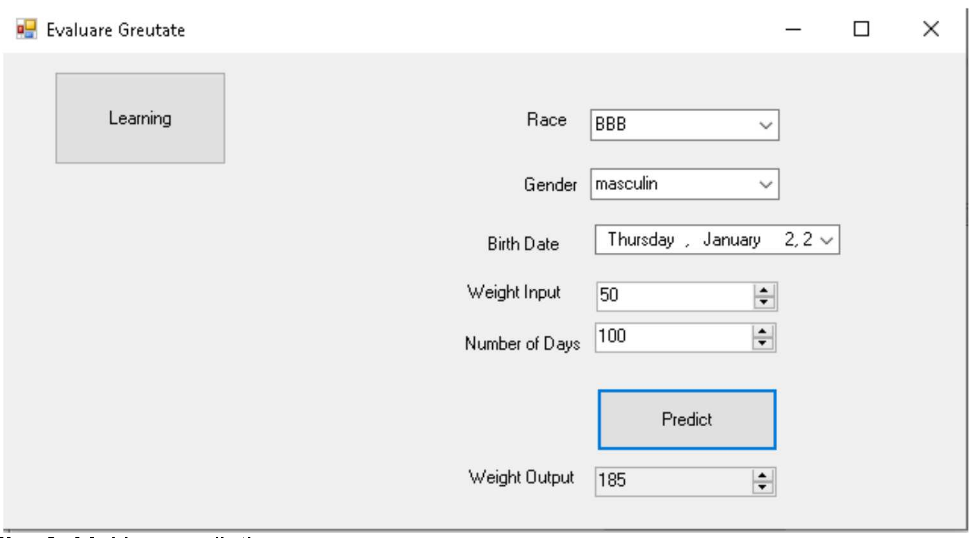


Fig. 2. Evolution of RSquare according to test data and learning times

2.3. Prediction

The realization of the prediction involves first loading the model obtained from learning.Then you will have to call with a breed, sex, date of birth, the weight that the calf has at the entrance and the number of days we expect for fattening.It will result from the prediction of the output weight after that number of days.
For example, a male bull, the BBB breed, born on 2 Jan 2020, having an input weight of 50 KG and which we want to increase his weight over a period of 100 days will have at the exit a weight of 185 KG.



Race: BBB
Gender: masculin
Birth Date: Thursday , January 2, 2
Weight Input: 50
Number of Days: 100
Weight Output: 185

Fig. 3. Making prediction

Making the prediction involves first loading the learned model, reading the input data and applying the model to them in order to obtain the result. The following part of code does that.

```
DataViewSchema predictionPipelineSchema;
ITransformer predictionPipeline = null;
MLContext context = new MLContext();

MdelDate.Bull el = new ModelDate.Bull();
//after initialization of el object’s properties we continue with load of the model and prediction
predictionPipeline = context.Model.Load(_modelPath, out predictionPipelineSchema);
//obtaining the prediction in the following variables
float InOutpurWeight = PredictieIngrasare(context, predictionPipeline, el);
```

3. Conclusion

The reduction in the number of records participating in learning did not lead to higher scores in the process of verifying the accuracy of learning but on the contrary. Therefore, filtering of learning data according to the company's objective was not necessary.

The decrease in the number of records has led to a decrease in learning time, but the size of the archive has not necessarily decreased.

Using the FastTree algorithm had very good learning times for our case.

Bibliography:

https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet