Voting Prediction

Description: This repository contains the first project of Artificial Intelligence course from Instituto Tecnológico de Costa Rica, imparted by the professor Juan Manuel Esquivel. The project consists on a comparison and analysis between 5 models (Logistic Regression / Neural Networks / Decision Trees / K-Nearest Neighbors / Support Vector Machines) to predict the president electoral votes for Costa Rica on 2018.

Content:

Installation:

Before using the project, first you have to install all the project dependencies.

- Python 3.5 or greater, and it has to be 64-bit.
- Numpy:
  - Install it with pip: `python -m pip install --user numpy`
- Scipy:
  - Install it with pip: `python -m pip install --user scipy`
- Tensorflow:
  - Install it with pip: `pip install --upgrade tensorflow`
- Keras:
  - For installing Keras you must have a installation of Tensorflow, Theano or CNTK.
  - Install it with pip: `pip install keras`
  - For storing the models you can install h5py: `pip install h5py`
  - For visualizing the model's graph: `pip install pydot`
- Scikit:
  - For installing Scikit you must have Python 3.3+, Numpy 1.8.2+ and Scipy 0.13.3+.
  - Install it with pip: `pip install -U scikit-learn`

Now that all dependencies are installed. You can install the project using:

```
pip install -U tec.ic.ia.p1.g07
```

Or you can clone the repository by:

```
git clone https://github.com/mamemo/Voting-Prediction.git
```

Usage:

When you have the directory with the repository, you have to search the repository on a console and execute the instruction:

```
python -m tec.ic.ia.p1.g07.py -h
```

This will display all the flags that you can work with:

```
-h, --help            show this help message and exit
--regresion-logistica
                      Logistic Regression Model.
--l1                  L1 regularization.
--l2                  L2 regularization.
--red-neuronal        Neural Network Model.
--numero-capas NUMERO_CAPAS
                      Number of Layers.
--unidades-por-capa UNIDADES_POR_CAPA
                      Number of Units per Layer.
--funcion-activacion {softmax,elu,selu,softplus,softsign,relu,tanh,sigmoid,hard_sigmoid,linear}
                      Activation Function.
--arbol               Decision Tree Model.
--umbral-poda UMBRAL_PODA
                      Minimum information gain required to do a partition.
--knn                 K Nearest Neighbors Model.
--k K                 Number of Layers.
--svm                 Support Vector Machine Model.
--prefijo PREFIJO     Prefix of all generated files.
--poblacion POBLACION
                      Number of Samples.
--porcentaje-pruebas PORCENTAJE_PRUEBAS
                      Percentage of samples to use on final validation.
--muestras {PAIS,SAN_JOSE,ALAJUELA,CARTAGO,HEREDIA,GUANACASTE,PUNTARENAS,LIMON}
                      The function to called when generating samples.
```

Each model uses different flags, but they have four in common, you can see the description next to each flag:

```
--prefijo PREFIJO     Prefix of all generated files.
--poblacion POBLACION
                      Number of Samples.
--porcentaje-pruebas PORCENTAJE_PRUEBAS
                      Percentage of samples to use on final validation.
--muestras {PAIS,SAN_JOSE,ALAJUELA,CARTAGO,HEREDIA,GUANACASTE,PUNTARENAS,LIMON}
                      The function to called when generating samples.
```

To run logistic regression you will need:

```
--regresion-logistica
                      Logistic Regression Model.
--l1                  L1 regularization.
--l2                  L2 regularization.
```

To run neural network you will need:

```
--red-neuronal        Neural Network Model.
--numero-capas NUMERO_CAPAS
                      Number of Layers.
--unidades-por-capa UNIDADES_POR_CAPA
                      Number of Units per Layer.
--funcion-activacion {softmax,elu,selu,softplus,softsign,relu,tanh,sigmoid,hard_sigmoid,linear}
                      Activation Function.
```

To run decision tree you will need:

```
--arbol               Decision Tree Model.
--umbral-poda UMBRAL_PODA
                      Minimum information gain required to do a partition.
```

To run k-nearest neighbors you will need:

```
--knn                 K Nearest Neighbors Model.
--k K                 Number of Layers.
```

To run support vector machine you will need:

```
--svm                 Support Vector Machine Model.
```

Models' Report:

This section contains the analysis of using each model and how well it performs with different parameters.

Logistic Regression

For logistic regression we had to compare how it performs with regularization L1 and L2. All the experiment combinations were ran 10 times and the value in the table is the mean. This algorithm uses the normalized samples NOMBRE. In these tests we used the next hyper-parameters to get the best results:

- Learning rate = 0.01
- Training epochs = 5000
- Batch size = 1000
- Regularization Coefficient = 0.01

The results are:

Prediction	**L1**				**L2**			
	Acurracy		Loss		Acurracy		Loss	
	Train	Test	Train	Test	Train	Test	Train	Test
r1	0.24674	**0.25455**	2.59636	2.58917	0.25614	0.252	2.55583	2.55691
r2	0.611325	0.6159	1.15681	1.15294	0.61821	**0.61945**	1.12866	1.12633
r2 with r1	0.60874	0.61335	1.15408	1.14953	0.61834	**0.62045**	1.12618	1.12462

HABLADA DICIENDO PORQUE LOS RESULTADOS DIERON ASI

Neural Network

Decision Tree

For the decision tree we had to compare how it performs with different thresholds, different amounts of attributes (r1, r2 and r2 with r1) and other combinations. All the experiment combinations were ran 10 times and the value in the table is the mean. This algorithm uses the normalized samples NOMBRE.

First we compared the accuracy of the tree without pruning with different thresholds, with the country results. Including the classification r1, r2 and r2 with r1. This is to see the behavior of the accuracy as it goes down the threshold, comparing the set of training with test.

The threshold is in the range of 0 to 1, where 1 is 100%. It is important to mention that as the node of a tree classifies the data, how closer to 1 is its deviation (value of the chi square), the classification will be worse.

The results are:

Round 1	Original	0.80	0.60	0.40	0.20	0.10	0.05	0.02
Training	99.814%	78,849%	42,965%	34,932%	28,290%	27,717%	27,482%	27,289%
Test	18,880%	20,530%	23,864%	25,755%	26,715%	27,040%	26,385%	27,155%

Round 2	Original	0.80	0.60	0.40	0.20	0.10	0.05	0.02
Training	99,911%	95,178%	83,981%	72,201%	65,539%	63,788%	63,394%	62,916%
Test	53,316%	54,505%	57,860%	60,045%	61,740%	61,880%	61,995%	62,510%

R2 with R1	Original	0.80	0.60	0.40	0.20	0.10	0.05	0.02
Training	99,983%	90,336%	79,884%	71,436%	65,473%	63,694%	63,415%	62,975%
Test	53,975%	56,239%	58,720%	60,085%	61,870%	61,715%	62,195%	62,555%

According to the results obtained with the threshold change and without pruning, we can conclude that:

- The accuracy of the tree without pruning, with the training set is greater than 99.8%, which indicates that there is an overfitting in the data, the accuracy of the test set is well below 99%. For this reason, an analysis of different values of thresholds for tree pruning is included.
- As the threshold is decreased, the performance of the training set is reduced, while the performance of the test set increases gradually.
- It can be seen that in each estimate vote, if the threshold value is close to 0, the performance of the training test and the test test is reasonably similar.
- With a threshold of 0.02, the performance of the model increases almost ten percent of its original accuracy with the tree without pruning.
- We can observe that the r2 and r2_with_r1 have a similar behavior, the accuracy of the tree without pruning is 53% and with a threshold of 0.02, 62.5%. Including the vote of the first round to estimate the vote of the second round has no direct effect, the classification of the second round that does not take into consideration the first round behaves practically the same.
- Although we can observe that there are two accuracy decreases (r1 with 0.05 and r2_with_r1 with 0.10) as the threshold increases, the highest accuracy results can always be observed at the 0.02 threshold.

Now we see some particular behaviors that have been found in the model. First, we will see the behavior by provinces, specifically Cartago and Puntarenas, to analyze if there are differences in accuracy. Next we will see the behavior of the accuracy when it is trained, but with the restriction that it can not repeat attributes in the whole tree.

In the following table the threshold is chosen 0.02 because it is the one that returns a better accuracy when pruning the tree according to the tables analyzed previously. The result of the tree without pruning is not analyzed.

Province	Cartago			Puntarenas		
Round	r1	r2	r2 with r1	r1	r2	r2 with r1
Training	26.450%	73.903%	73.894%	35.012%	55.992%	56.211%
Test	26.070%	73.855%	73.800%	35.336%	55.745%	55.735%

In the table of provinces we can notice some behaviors different to the estimation behavior by country. The predictions of Cartago surpass the 73% percent accuracy in r2 and r2_with_r1, surpassing by 10% the predictions by country. On the other hand, the opposite effect is seen in Puntarenas, where the accuracy is 55%, being 10% lower than the predictions per country.

In the r1 prediction of Puntarenas, with a threshold of 0.02 the tree is pruned completely, therefore all the outputs are Restauracion Nacional in most of the occasions. We see an accuracy of 35% that obeys the proportion of votes obtained by the aforementioned RN.

It is important to mention what differentiates the provinces to understand the results:

- Cartago was the province that in its two rounds of voting had the lowest proportion of abstinence, while Puntarenas was one of the provinces with the highest proportion of abstinence.

How does that difference affect? By taking only the people who voted, Cartago is more accurate because there is more data from the entire province, but in Puntarenas you have data from a smaller sector, so the data contains noise when you match the indicators of the entire population of Puntarenas. The indicators used include the population that did not vote, which also affect the model.

The following table also uses the 0.02 threshold because it is the one that returns the best accuracy when pruning the tree according to the tables analyzed previously. The difference of the following prediction to the ones analyzed above, is that the tree was trained with a restriction that will be mentioned later. In this case the accuracy of the unpruned tree is shown, because the results have a different behavior.

Threshold	Without Pruning			0.02		
Round	r1	r2	r2 with r1	r1	r2	r2 with r1
Training	27.905%	62.752%	62.801%	27.261%	62.396%	62.412%
Test	26.705%	62.085%	62.160%	26.980%	62.315%	62.320%

We can see that training a tree with a restriction can cause the accuracy to increase considerably, to the point that by applying the 0.02 pruning (when before it was the threshold that caused the highest accuracy) the accuracy can decrease instead of increase.

The restriction is that an attribute can be used only once, not repeatedly as in the previous iterations. With only the training, the performance of the training and testing set is similar, which indicates that there is no overfitting as it exists when the tree is trained allowing repeating attributes.

It can be concluded that, including the restriction, there is no increase in the overall performance of the predictions, but there is a clear decrease in overfitting in their initial training.

K-Nearest Neighbors

Support Vector Machine

Samples Generator