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.
  --c C                 Penalty parameter C of the error term.
  --kernel {linear,rbf}
                        Specifies the kernel type to be used in the algorithm.
  --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.
  --c C                 Penalty parameter C of the error term.
  --kernel {linear,rbf}
                        Specifies the kernel type to be used in the algorithm.
```

Models' Report:

This section contains the analysis of using each model and how well it performs with different parameters. Every model is called from g07.py and the process is the following: 1. Waiting parameters. 2. Receiving parameters. 3. Creating the respectively model. 3. Generating and normalizing samples (some models differ in normalization methods). 4. Running the model for each round prediction and storing the samples predictions for later. 5. Generating an output file with each sample and its respectively round prediction. The output file is named with a prefix that comes on the 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. Also, all the experiments were ran with normalized samples covering the whole country, the samples were normalized and the labels were transformed to one hot encoding. The algorithm was implemented using Tensorflow and you can follow the process with specific commented functions at Logistic_Regression.py. 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 were:

Round	**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

> The loss is calculated by the function softmax_cross_entropy_with_logits given at Tensorflow.

According to the results for logistic regression using L1 and L2 regularization, we can conclude:

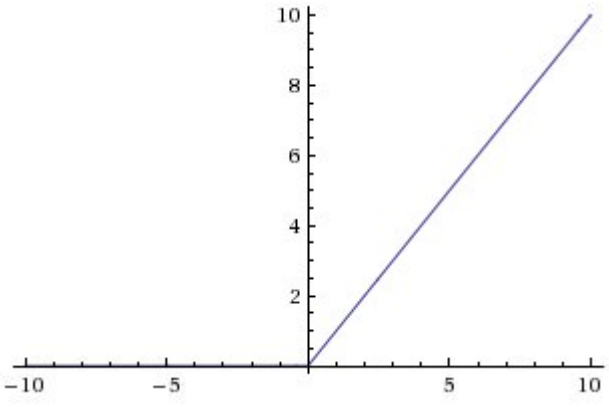
- For Round 1, the L1 regularization gives the minimal error in the prediction at test set. L1 specializes on sparse feature spaces, given the nature of the samples on round 1 (15 classes to predict, and 4 classes with the majority of samples) we can infer L1 performs better because we have the features space on round 1 being more sparse than other rounds. Sparsity on the space means that the dataset have a lot of gaps between different samples and prediction will be harder to get a reasonable accuracy.
- However, looking closely to the table, L2 on Round 1 is very close to L1. But, we can see the L2 regularization doing a little overfitting on training set, which not happens on L1.
- For Round 2 and Round 2 with round 1, we can see accuracy rates very close. The nature of these two groups of samples are the same (4 classes to predict, 2 classes with majority of samples) and with only one difference (Round 2 with round 1 add two attributes: numbers of voters in previous round and vote on previous round). For more details on samples used, go to Samples Generator.
- A reason provoking low accuracy rates on this specific model it may be because the space is linearly non-separable, so the error will never be near to zero. Linearly non-separable spaces are the ones that with a linear function you can't separate the classes perfectly. This concept is related with Logistic Regression because the implementation on this model is based on linear models.

Neural Network

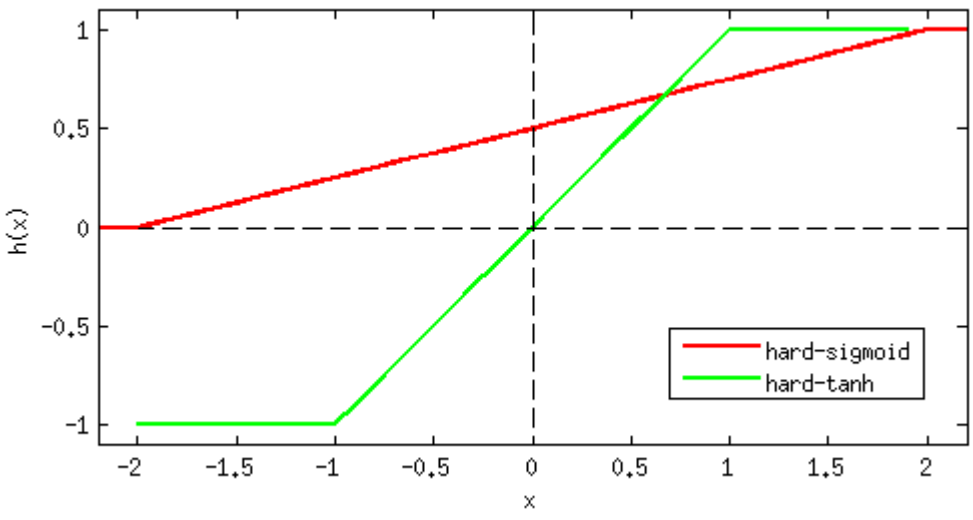
For neural network model we had to compare how it performs with different structures and different activation functions. All the experiment combinations were ran 10 times and the value in the table is the mean. Also, all the experiments were ran with normalized samples covering the whole country, the samples were normalized and the labels were transformed to one hot encoding. The algorithm was implemented using Keras on top of Tensorflow and you can follow the process with specific commented functions at Neural_Network.py. In these tests we used the next hyper-parameters to get the best results: * Learning rate = 0.01 * Training epochs = 3000 * Batch size = 500

Structures had the input layer's and output layer's amount of units accordingly to the specific prediction round (e.g. the round 1 had 15 units on output layer). We test different hidden layers amount: * 1 hidden layers with 20 units. * 2 hidden layers fully connected with 30 and 20 respectively. * 3 hidden layers fully connected with 30, 20 and 10 respectively.

The activation functions tested were: * ReLU



* Hard Sigmoid (**the red one**)



The results were:

- First structure: 1 hidden layers with 20 units.

Round	**ReLU**		**Hard Sigmoid**	
	Acurracy	Loss	Acurracy	Loss

Round	ReLU				Hard Sigmoid			
	Train	Test	Train	Test	Train	Test	Train	Test
r1	0.2955	**0.27985**	1.87434	1.8835	0.26577	0.26475	1.9004	1.88837
r2	0.63767	0.6221	0.69096	0.70923	0.62474	**0.6299**	0.70856	0.70884
r2 with r1	0.6384	0.6187	0.69025	0.712	0.62685	**0.62925**	0.7065	0.70788

- Second structure: 2 hidden layers fully connected with 30 and 20 respectively.

Round	ReLU				Hard Sigmoid			
	Acurracy		Loss		Acurracy		Loss	
	Train	Test	Train	Test	Train	Test	Train	Test
r1	0.28854	**0.2773**	1.88636	1.8903	0.24879	0.24765	1.90916	1.91535
r2	0.63499	**0.61385**	0.69893	0.71269	0.59294	0.5923	0.72854	0.73106
r2 with r1	0.63827	**0.61195**	0.6934	0.71736	0.59294	0.5923	0.72854	0.73106

- Third structure: 3 hidden layers fully connected with 30, 20 and 10 respectively.

Round	ReLU				Hard Sigmoid			
	Acurracy		Loss		Acurracy		Loss	
	Train	Test	Train	Test	Train	Test	Train	Test
r1	0.24814	**0.2513**	1.90623	1.90478	0.2468	0.25565	1.91678	1.91866
r2	0.59471	**0.596**	0.73018	0.73422	0.5958	0.5923	0.73265	0.72856
r2 with r1	0.59471	**0.596**	0.73018	0.7342	0.5958	0.5923	0.73265	0.72856

> The loss is calculated by categorical crossentropy, sopported by Keras.

According to the results, we can conclude:

- The samples' nature, explained before, affects different algorithms in a way that it only can achieve a certain amount of accuracy, making the model non-optimal.
- With structures with two or more hidden layers, ReLU performs better than hard sigmoid at transforming the approximated function into non-linear. However, with structures with one hidden layer, hard sigmoid have the capability to approximate a better function at least when there are a little amount of outputs.
- The best results were given on the first structure, one hidden layer with 20 units, proving that simple models are best than complicated. Nevertheless, maybe the more complicated structures could performed better if more time were applied on parameter tunning.
- The first and second structures were overfitting a little bit in the training data. Only the last structure, three hidden layer with 30, 20 and 10 units, was performed better on every test set for each prediction round, independently of which activation function were chosen. However, this is the structure with worst accuracy.
- The tested structures came to a point were they gave the same results for r2 and r2 with r1 predictions. That means, for those structures the extra attributes dont't affect at all the final prediction. Being specific, the loss and accuracy change but only a few decimals after e-06.

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 tested 10 times and the value in the table is the mean of all. The decision tree model was implemented from scratch and you can follow the algorithm with specific commented functions at Decision_Tree.py.

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. We make this to see the behavior of the accuracy as it goes down the threshold, comparing the training set with test set.

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 were:

Round 1	**Original**	**0.80**	**0.60**	**0.40**	**0.20**	**0.10**	**0.05**	**0.02**
Training	0.99814	0.78849	0.42965	0.34932	0.28290	0.27717	0.27482	0.27289
Test	0.18880	0.20530	0.23864	0.25755	0.26715	0.27040	0.26385	**0.27155**
Round 2	**Original**	**0.80**	**0.60**	**0.40**	**0.20**	**0.10**	**0.05**	**0.02**
Training	0.99911	0.95178	0.83981	0.72201	0.65539	0.63788	0.63394	0.62916
Test	0.53316	0.54505	0.57860	0.60045	0.61740	0.61880	0.61995	**0.62510**
R2 with R1	**Original**	**0.80**	**0.60**	**0.40**	**0.20**	**0.10**	**0.05**	**0.02**
Training	0.99983	0.90336	0.79884	0.71436	0.65473	0.63694	0.63415	0.62975
Test	0.53975	0.56239	0.58720	0.60085	0.61870	0.61715	0.62195	**0.62555**

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 set and the test set 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 votes of the first round to estimate the votes 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 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	0.26450	0.73903	0.73894	0.35012	0.55992	0.56211
Test	0.26070	0.73855	0.73800	0.35336	0.55745	0.55735

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 for generating samples, 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	0.27905	0.62752	0.62801	0.27261	0.62396	0.62412

Threshold	**Without Pruning**			**0.02**		
Test	0.26705	0.62085	0.62160	**0.26980**	**0.62315**	**0.62320**

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

For the Nearest Neighbors with k-d Tree model we had to compare how it performs with different k values, being k the number of nearest neighbors to be analized, and different amounts of attributes (r1, r2 and r2 with r1) (as well as with the other models). All the experiment combinations were tested 10 times and the value that appears in the table is the mean of all. The k-d tree model was implemented from scratch and you can follow the algorithm with specific commented functions at K_Nearest_Neighbors.py.

First we compared the accuracy of the k-d tree without with different k values, with the country results. Including the classification r1, r2 and r2 with r1. We make this to see the behavior of the accuracy as the k value changes, comparing the training set with test set.

The results are:

Round 1	**k=3**	**k=5**	**k=7**
Training	0.85445	0.60084	0.54519
Test	0.19490	0.21905	**0.22070**
Round 2	**k=3**	**k=5**	**k=7**
Training	0.77504	0.72000	0.69761
Test	0.55555	0.56925	**0.57645**
R2 with R1	**k=3**	**k=5**	**k=7**
Training	0.77769	0.71920	0.69674
Test	0.54825	0.56960	**0.58100**

According to the results obtained with the k value change, we can conclude that:

- The accuracy of the k-d tree increases as the k value does too, which indicates that the model performs better with higher k values and might even work even better with values higher than 7 (it was only possible to test it with 3 different k values due to processing time).
- As the k value increased, the performance of the training set is reduced, while the performance of

the test set increases gradually.

- Due to the high amount of dimensions (54 in the case of round 1 or 2, and 56 in the case of round 2 with round 1), the model suffers from the curse of dimensionality so the nearest neighbors are not really near and it affects its accuracy in comparison with the rest of the models.
- Due to the fact that k-d trees are only appropriate when there are many more examples than dimensions, preferably 2^n examples, being n the number of dimensions, we would need to test the model with approximately 17 billion samples to make it perform optimally. If there aren't enough examples, literature states that k-d trees lookup is not faster than linear scan of the whole data set, so the average of processing time of our k-d tree for 10000 samples, calculating the three types of predictions (executing three times) and calculating the error of the whole training set and test set ended up being on average about 2 hours, 15 minutes on an average computer.
- Although the k-d tree tries to be faster by looking for the closest points only in only branch, sometimes it is neccessary to look for them in the other branch (some of the closest points might be in the ignored branch) according to the k-d tree algorithm, and it slows down the processing time a bit more, making the initial advantage disappear.
- Even when the k-d tree is trained with a set of data, and then tested with the same data (training set), it still fails to predict them, that due to the curse of dimensionality. We can state that because the tree succeeds to find the nearest point almost always but it gets confused with the rest of near points, because they are quite far.
- As well as in the case of the Decision Tree, we can observe that the r2 and r2_with_r1 modes have similar behavior, the accuracy values of the k-d tree between the two modes differ in a range of 0.00035-0.0073. Including the votes of the first round to estimate the votes 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.

As we mentioned before in the Decision Tree section, we observed the behavior by provinces, and found out that the accuracy was notably better in the case of Cartago, not only in comparison to Puntarenas, but also in comparison to the accuracy measured with data from all the country.

Next, we will see the behavior of the accuracy when it is trained with only data from Cartago. As for k, it will only be configured with the values of 3 and 7, because it was checked above that accuracy increased as k did too, and these two values showed a bigger difference.

Cartago		**k=3**			**k=7**	
Round	r1	r2	r2 with r1	r1	r2	r2 with r1
Training	0.86342	0.81441	0.81359	0.53148	0.76129	0.76274
Test	0.18264	0.67100	0.66510	**0.20200**	**0.70445**	**0.70010**

According to the results obtained, we can conclude that:

- In the table of Cartago we can notice some behaviors different to the estimation behavior by country. The predictions of Cartago surpass significantly every accuracy value in r2 and r2_with_r1 in the previous experiment with k=3 and k-7. As we mentioned before, Cartago was the province that in its two rounds of voting had the lowest proportion of abstinence.

- This is other example that shows that by taking only the people who voted, Cartago's predictions are more accurate because there is more data from the entire province, and because the indicators used for generating samples includes the whole population of the province, which makes the data connect pretty well.

As an overall conclusion, it can be observed that round 1 predictions have almost always low accuracy. It might be due to some attributes that bring some noise to the data set, or because plenty of factors affected the vote choice for that round and some or them were not included in the sample generator.

Support Vector Machine

For support vector machine we compared how it performs with linear and rbf kernels. All the experiment combinations were ran 10 times and the value in the table is the mean. Also, all the experiments were ran with normalized samples covering the whole country, the samples were normalized and even the labels were normalized. The algorithm was implemented using Scikit and you can follow the process with specific commented functions at Support_Vector_Machine.py. In these tests we used the next hyper-parameters to get the best results: * C = 1 (but you can configured it as a parameter) * Decision function shapeTraining = ovr

The results were:

Round	**Linear**				**Rbf**			
	Acurracy		Loss		Acurracy		Loss	
	Train	Test	Train	Test	Train	Test	Train	Test
r1	0.29086	**0.28845**	37.19956	37.2352	0.38024	0.286	31.29876	37.01385
r2	0.62129	0.6238	0.41412	0.4149	0.67651	**0.62945**	0.35705	0.40515
r2 with r1	0.6215	0.62425	0.41387	0.4146	0.68087	**0.6306**	0.35286	0.40445

The loss is calculated by the function mean_squared_error given at Scikit.

According to the results for support vector machine using linear and rbf kernels, we can conclude:

- With more time of parameter tunning this algorithm would performed better, that is because we only adjust one parameter and this model gave the **best overall results** of all the algorithms tested on this project.
- We consider rbf better than linear, because:
  - It has the fewest loss of the kernels.
  - Better accuracy was achieved.
  - At r1 linear kernel won, but only on 0.02 percent.
- However, you can see it performs better on training set than test set, but that allows to lower the mean squared error between ground truth and prediction.

Samples Generator

The creation of samples was done by using a Python module developed by our work team called tec.ic.ia.pc1.g07, which contains all the logic necessary to recreate a population of N Costa Ricans and their the vote for the first and second round of the 2018 electoral process. The discribed module tec.ic.ia.pc1.g07 can be found on pc1 or can be installed using pip: `pip install tec.ic.ia.pc1.g07`

This module needs three auxiliary files to produce the samples: Juntas.csv, VotosxPartidoxJunta.csv and Indicadores_x_Canton.csv, because each of them has important details regarding the population that helps the generator to be as precise as possible and to be faithful to the reality of Costa Rica's population. These three mentioned files contain data from the scrutiny records of the elections, the maping of the voting boards to their locations, and the location indicators (regarding its population). First, in Juntas.csv each of its rows represents a voting board, and its columns represent the following data in the same order:

```
1.  Province
2.  Canton
3.  District
4.  Neighborhood
5.  Board Number
6.  Number of Electors
7.  Received Votes (First Round)
8.  Received Votes (Second Round).
```

In the case of VotosxPartidoxJunta.csv, each row represents the votes from each voting board for each party, and the columns represent the following data in the same order:

```
1. Board Number
2. Accesibilidad sin Exclusión
3. Acción Ciudadana
4. Alianza Demócrata Cristiana
5. De Los Trabajadores
6. Frente Amplio
7. Integración Nacional
8. Liberación Nacional
9. Movimiento Libertario
10. Nueva Generación
11. Renovación Costarricense
12. Republicano Social Cristiano
13. Restauración Nacional
14. Unidad Social Cristiana
15. Null Votes, Blank Votes
16. Acción Ciudadana (Second Round)
17. Restauración Nacional (Second Round)
18. Null Votes (Second Round)
19. Blank Votes (Second Round).
```

Lastly, in Indicadores_x_Canton.csv each row represents a canton in Costa Rica, and each column

represents the following data in the same order:

```
1. Province
2. Canton
3. Total Population
4. Canton Surface (Km2)
5. Density (People per Km2)
6. Urban Population Percentage
7. Urban Population
8. Non-urban Population
9. Relation Men-Women (Men per 100 Women)
10. Number of Women
11. Number of Men
12. Demographic Dependency Relation: Dependent people (People youger than 15 years
old or older than 65 years old)
13. Women in the age range of 15 to 19
14. Women in the age range of 20 to 24
15. Women in the age range of 25 to 29
16. Women in the age range of 30 to 34
17. Women in the age range of 35 to 39
18. Women in the age range of 40 to 44
19. Women in the age range of 45 to 49
20. Women in the age range of 50 to 54
21. Women in the age range of 55 to 59
22. Women in the age range of 60 to 64
23. Women in the age range of 65 to 69
24. Women in the age range of 70 to 74
25. Women in the age range of 75 to 79
26. Women in the age range of 80 to 84
27. Women in the age range of 85 or more
28. Men in the age range of 15 to 19
29. Men in the age range of 20 to 24
30. Men in the age range of 25 to 29
31. Men in the age range of 30 to 34
32. Men in the age range of 35 to 39
33. Men in the age range of 40 to 44
34. Men in the age range of 45 to 49
35. Men in the age range of 50 to 54
36. Men in the age range of 55 to 59
37. Men in the age range of 60 to 64
38. Men in the age range of 65 to 69
39. Men in the age range of 70 to 74
40. Men in the age range of 75 to 7
41. Men in the age range of 80 to 84
42. Men in the age range of 85 or more
43. Individual Occupied Houses
44. Average Ocupants by Individual Occupied House
45. Average Houses in Good Condition
46. Number of Houses in Good Condition
47. Number of Houses in Bad Condition
48. Crowded Houses Percentage (more than 3 people by dormitory per each 100
occupied houses).
49. Number of Crowded Houses
50. Number of Non-crowded Houses
51. Literacy Percentage
52. Literacy Percentage in 10-24 Year Olds
```

53. Literacy Percentage in 25+ Year Olds
54. Average Education (Average Approved Years In Regular Education)
55. Average Education in 25-49 Year Olds
56. Average Education in 50+ Year Olds
57. Assistance Percentage in Regular Education
58. Assistance Percentage in Regular Education in 0-5 Year Olds
59. Assistance Percentage in Regular Education in 5-17 Year Olds
60. Assistance Percentage in Regular Education in 18-24 Year Olds
61. Assistance Percentage in Regular Education in 25+ Years Olds
62. People With No Education
63. People With Incomplete Primary School
64. People With Complete Primary School
65. People With Incomplete High School
66. People With Complete High School
67. People With Superior Education
68. Percentage of People Outside of the Work Force
69. Retired People
70. Rentier People
71. Students
72. People in Domestic Work
73. People with Other Reason of Unemployment
74. Percentaje of People Inside the Work Force
75. Percentaje of Men Inside the Work Force
76. Percentaje of Women Inside the Work Force
77. People Working In The Primary Sector
78. People Working In The Secondary Sector
79. People Working In The Third Sector
80. Percentage of Not Insured Occupied People
81. Percentage of People Born Abroad
82. Number of People Born Abroad
83. Number of People Not Born Abroad
84. Percentage of People With Disability
85. Number of People With Disability
86. Number of People Without Disability
87. Percentage of Not Insured People
88. Number of Not Insured People
89. Number of Insured People
90. Direct Insured People
91. Indirect Insured People
92. Insured People by Other Way
93. Percentage of Houses With Female Head
94. Percentage of Houses With Shared Head
95. People With Cell Phone
96. People With House Phone
97. People With Computer
98. People With Internet
99. People With Electricity
100. People With Toilet
101. People With Water

Note:

Voting boards abroad were not taken into account.

Now, the general operation of the generator is going to be explained by the steps it goes through to create the samples:

- It contains two different functions in case the samples must belong to the country or a single province, so one of the must be called and be told how many samples must be created (and the name of the province if needed).
  - If the samples must belong to the whole country, the function opens the files mentioned above, and for each sample it chooses a random voting board. This is done by taking the number of electors of the voting boards, calculating their ranges, generating a random number and classifying it in a range.
  - If the samples must belong to a single province, the function opens the files mentioned above, calculates the indexes for the province data, and for each sample it chooses a random voting board. This is done by taking the number of electors of the voting boards, calculating their ranges, generating a random number and classifying it in a range.
- For each sample, the generator locates the data from the selected voting board, and that way it gets the demographic and geographic information for it. The general canton information that doesn't need to be calculated is just taken from the files and copied to the sample.
- For each sample, the attributes that must be calculated and randomized are calculated the same way the number of voting board is: taking the values for each attribute from the files, calculating their ranges, generating a random number and classifying it in a range.
- All the samples are returned.

The following list shows all the attributes generated for each sample:

```
1.  Province
2.  Canton
3.  District
4.  Neighborhood
5.  Board Number
6.  Number of Electors
7.  Received Votes (First Round)
8.  Received Votes (Second Round).
9. Total Population
10. Canton Surface (Km2)
11. Density (People per Km2)
12. Urban Population Percentage
13. If The Sample Belongs to Urban Population or Not
14. Relation Men-Women (Men per 100 Women)
15. Man/Woman
16. Demographic Dependency Relation: Dependent people (People youger than 15 years
old or older than 65 years old)
17. Age Range According to Their Gender
18. Individual Occupied Houses
19. Average Ocupants by Individual Occupied House
20. Average Houses in Good Condition
21. If the Sample Lives in a House in Good/Bad Condition
22. Crowded Houses Percentage (more than 3 people by dormitory per each 100
occupied houses).
23. If the Sample Lives in a Corwded House or not
24. Education Level
```

```
25. Literacy Percentage
26. Literacy Percentage in Their Age Range
27. If The Sample Is Literate Or Not (it is only considered the possibility of
illiteracy if the sample doesn't have an education or has incomplete primary school
studies).
28. Average Education (Average Approved Years In Regular Education)
29. Average Education in Their Age Range (the 25-49 years range is taken as if it
started at 18 because of lack of data)
30. Assistance Percentage in Regular Education
31. Assistance Percentage in Regular Education in Their Age Range (the 5-17 years
range is taken as the 15-19 years range, and the 18-24 years range as the 20-24
years range, because of lack of data accuracy).
32. Percentage of People Outside of the Work Force
33. Percentaje of People Inside the Work Force
34. Percentaje of Men Inside the Work Force
35. Percentaje of Women Inside the Work Force
36. If The Sample Is Inside/Outside the Work Force
37. Reason for Being Outside The Work Force (if its the case), or The Sector in
Which The Sample Works (if its the case)
38. Percentage of Not Insured Occupied People
39. Percentage of People Born Abroad
40. If the Sample Was Born Abroad Or Not
41. Percentage of People With Disability
42. If The Sample Has a Disability or Not
43. Percentage of Not Insured People
44. If The Sample Is Insured or Not
45. Insurance Way (if its the case)
46. Percentage of Houses With Female Head
47. Percentage of Houses With Shared Head
48. Type of Head (lidership) In The Samples House
49. If The Sample Has A Cell Phone
50. If The Sample Has A House Phone
51. If The Sample Has A Computer
52. If The Sample Has Internet
53. If The Sample Has A Electricity
54. If The Sample Has A Toilet
55. If The Sample Has A Water
56. Sample's First Round Vote
57. Sample's Second Round Vote
```

For the prediction models developed, the indexes 2, 3, 4, 5, 6, 10, 11, 16, 20, 22, 25, 26, 28, 31, 32, 33, 38, 39, 41, 45 where ignored as they were not significant and may produce noise.

> Note:
>
> It is only considered the possibility of a generated sample to be illiterate if they don't have an education, or they have incomplete primary school studies, because the Costa Rican education system ensures that a person with complete primary school studies cannot be illiterate.
>
> To generate the Average Education For Their Age, the 25-49 years range is taken as if it started at 18 because of lack of data for ages lower than 25 years.

To generate the Assistance Percentage to Regular Education for Their Age, the 5-17 years range is taken as the 15-19 years range, and the 18-24 years range as the 20-24 years range, because of lack of data accuracy between the age ranges for the samples, and the age ranges from the Assistance Percentage to Regular Education Information used.

It is only considered the possibility of a person being outside of the work force because of being retired if their age exceeds 60 years, that due to the Costa Rican legislation in the matter.

Unit Testing

For this project we created unit testing for every function that were created by us and some functions that involves frameworks. We emphasize on Decision Tree and K-Nearest Neighbors models, because that ones were implemented from scratch. All the test can be located at test_models.py were every test is documented with an objective, parameters, requirements and desired outputs.