UNIVERSITY of the
WESTERN CAPE

University of the Western Cape
Department of Computer Science
Private Bag X17, Bellville, 7535
South Africa
Tel: +27 21 959 3010
E-mail: aismail@ uwc.ac.za
tuyinuel@gmail.com
iahmed@uwc.ac.za
mghaziasgar@uwc.ac.za

# CSC311 – Machine Learning – Final Project

The final phase of the CSC311 Machine Learning course is a project involving the application of the classification techniques that you've learnt in the course to a real-world classification problem. The aim of this project is to allow you to "put it all together" and go from **data-set** right through to **testing to produce results**, and **presentation**.

Note that the project is worth 40% of the overall mark for CSC311 Machine Learning component.

## 1. General Project Structure

You will be randomly organized into groups of 3 students. Each group will be given a data set to work with. The data sets will be provided to you in a few days time. Each student in the group will be assigned to apply one of the following three techniques to the data set: logistic regression; neural networks; support vector machines. Therefore, each group will end up applying all three techniques to the data set they have been assigned.

The project implementation will have the following distinct phases:

1. **Phase 1: Training**
    a. Dividing up your data set.
    b. Training and optimization.

2. **Phase 2: Testing**
    a. Testing and results generation.
    b. Demo creation.

3. **Phase 3: Presentation and comparison**

All of the work should be done in Jupyter Notebooks running the version of **Python 3.6 or newer.** Make sure to do all the work in the notebook in a very well-documented and systematic manner. Always assume that someone will eventually read over your work and you will need to put in text as you go along to explain what each part of the notebook is doing, as you go along.

This project brief will provide further detail into exactly what each phase above entails and how it should be done below. There will be three consultations (one each week) on a day and time that will be communicated with you shortly (Dr Ibrahim will conduct a survey on Thursday morning to find out the best time for everybody). In the these sessions, teams will be expected to discuss their work with one of the lecturers. Each weekly consultation will be dedicated to each phase of the project. This will allow the team to explain the problems they faced pertaining to the specific phase of the project, and receive feedback to fine tune their work. There will be marks assigned to progress made and effort put in during the weekly meetings.

## 2. Techniques To Be Used / Compared

You've learnt how to use all three techniques. In terms of specifics, use the following:

**Support Vector Machines:**
Use the sklearn implementation of SVMs. Use the Gaussian (aka Radial-Basis Function) kernel only.

**Neural Networks:**
Use the sklearn implementation of NNs. Use a three-layer NN with one input, one hidden and one output layer.

**Logistic Regression:**
Use the sklearn implementation of Logistic Regression. As you know, the machinery of Logistic Regression allows for a large number of different feature variations e.g. higher-order features and/or combination of features.

Using these three techniques, you will implement each of the phases described in the next section over three weeks.

## 3. Project Phases

As described before, the project is implemented in three phases. The following three sections cover each of the project phases in detail.

### 3.1 Phase 1: Training

The following two subsections describe Phases 1a and 1b that need to be implemented.

#### 3.1.1 Phase 1a: Dividing Up Your Data Set

Please note that in this part (Phase 1a), the team needs to create a Jupyter Notebook (call it "groupx_dividedata.ipynb") and put the code to divide up the data set in this file. All three team members should work on this together.

You hopefully know by now that you, first and foremost, need to divide up your data set into **Training**, **Cross-Validation (CV)**, and **Testing** portions (feel free to refer to the last part of the Linear Regression slides for more details on this). The **Training** and **CV** portions of the data set will be used in Phase 1b ("Training and Optimization") described below, while the **Testing** portion of the data should be stored away for Phase 2 ("Testing") described later on. Please ensure that the **Testing** portion is not used in any way in the "Training and Optimization" phase.

When the notebook is run, it should read the original data file in, split it up appropriately and write the resulting splits into three separate files: "groupx_train.dat", "groupx_cv.dat" and "groupx_test.dat". These files will be read in and used in the next parts of the project.

Implementation note: you don't need to re-invent the wheel when it comes to dividing up your data set, despite that fact that it should be very easy to do so. Numpy provides a split() function on matrices/arrays that can help you do this OR you can look into the train_test_split() function of sklearn.model_selection (although you would need to be clever about it – train_test_split() only divides into 2 portions, not 3) OR you can look on StackOverFlow for further solutions.

How you divide up your data is left to you (as a team). As you know, you can use anything from 50-90% of the data for training, and the remaining portion (depending on how much you used for training) split equally between CV and Testing. You should know that there is a trade-off between

the size of the data used between training and testing: Using a much bigger portion for training makes it more likely to obtain a converged and accurate model, but leaves very little to test on, so it becomes difficult to say for certain whether or not the model generalizes to new (unseen) data; on the other hand, using a much bigger portion of data for testing makes it possible to say for certain how the model generalizes, but this could potentially mean that not enough data was used during training resulting in a lack of convergence and poor testing accuracies. Choose wisely.

Each team should make a decision about how to divide up the data set and justify it. The code used to divide up the data will be the same for all three team members, and should be replicated into the top portion of each of the three notebooks of team members. Therefore, all three team members will end up with the same **Training**, **CV**, and **Testing** sets. From this point on, each team member will then continue on to Phase 1b as described below and work on their own technique.

### 3.1.2 Phase 1b: Training and Optimization

Please note that in this part (Phase 1b), each team member needs to create a separate Jupyter Notebook that: carries out optimization; and then trains on the optimal parameters for his/her specific assigned technique. Therefore, there should be **3 separate** Jupyter Notebooks produced at the end of this procedure; one for SVMs (call it "groupx_train_svm.ipynb"), one for NNs (call it "groupx_train_nn.ipynb") and one for Logistic Regression (call it "groupx_train_logreg.ipynb"). Each of these files should load in the "groupx_train.dat" and "groupx_cv.dat" files created in the previous part. These will be used as below.

Before anything, keep in mind that your data needs to be scaled before use. If the sklearn function(s) that you are using to train with scale your data automatically, then great. Otherwise, make sure to scale your data yourself (sklearn also has functions to scale data for you).

As you've learnt extensively in class, the first phase of a ML project involves training with a ML technique, and doing this requires optimization. Different techniques have different parameters that can be optimized. You need to tune the parameters described below for each technique.

**Support Vector Machines:** Cost $C$ and Gaussian kernel parameter $\gamma$. To do this, you need to systematically step through a variety of different $(C, \gamma)$ combinations, train your SVM using the Train portion of the data set on the specific $(C, \gamma)$ combination, and then apply the CV set to obtain a CV accuracy. We suggest that you try the following range of values: $C$: $\{2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}, 2^{7}, 2^{9}, 2^{11}\}$ and $\gamma$: $\{2^{-11}, 2^{-9}, 2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}\}$. So there are 8 possibilities for each parameter, meaning that there are 64 unique combinations. At the end of this procedure, you should have a table of CV accuracies corresponding to each $(C, \gamma)$ combination; a total of 64 entries in the table. At the end of the day, you can pick the $(C, \gamma)$ combination that obtains the highest CV accuracy.

**Neural Networks:** The learning rate $\alpha$ and the number of nodes in the (one and only) hidden layer. To do this, it may be a good idea to start with, say, 5 nodes in the hidden layer, and then increase in steps of, say, 2, 3 or 5 nodes, until it is clear that your NN converges. For each number of (hidden) nodes, train your NN on the Train portion of the data set. Make sure you use a graph to ensure that your NN has converged each time. Then, pass your CV portion of the data set into each trained model to obtain a CV accuracy. At the end of the day, you should end up with a table that summarizes the number of hidden nodes vs CV accuracy. Then, you can pick the number of hidden nodes that results in the highest CV accuracy.

**Logistic Regression:** The features. As mentioned before, you can get really creative with features when it comes to logistic (and linear) regression. For this project, look into second-order features for some features, as well as combining some of the features as well. All-in-all, try nine different

sets of features that include higher-order features of some of the features and/or combination of features as well, as well as the original set of features as they were. This should result in ten unique models. For each of the models, carry out training on the Train portion of the data set, and then pass in the CV portion of the data set to obtain an accuracy. At the end of the procedure, you should end up with ten CV accuracies corresponding to each of the models. You can then pick the model with the highest CV accuracy as the model to use.

At the end of this procedure, use the ideal parameters of each technique to carry out your final training using ALL of the training data (the **Training** + **CV** portions of the data) to produce your final models. You should end up with three models which will be used in the testing phase described next.

So at the end of Phase 1, you should end up with **4 Jupyter Notebooks** with names as mentioned before; the first notebooks divides up the data into three parts, and then the next three notebooks each: carry out optimization on the relevant technique; and carry out final training of the relevant technique using the ideal parameters resulting from optimization to produce a final model.

## 3.2 Phase 2: Testing

In this phase, all three team members should produce a single Jupyter Notebook called "groupx_testing.ipynb" that will produce and demonstrate all of the testing results as described in Phase 2a below, and another single Jupyter Notebook called "groupx_demo.ipynb" that allows users to "try out" the final system using the final trained models as described in Phase 2b below.

### 3.2.1 Phase 2a: Testing and Results Generation

The next part of an ML project (once you have optimized your technique(s) and produced your final optimized model(s)) is to carry out objective testing to gauge how well your model generalizes.

When it comes to testing, the same strategy can be applied to any (classification) technique. For each of the three techniques, use the Testing portion of the data set that you set aside earlier on to produce the results described below, and any other tests that you deem to be relevant.

1. **Overall Accuracy:** determine the overall number of testing data samples that are correctly classified as a percentage of the total number of testing data samples.

2. **Per-Class Accuracy:** for each of the classes that you are classifying, determine the number of samples of that class that were correctly recognized as a percentage of the total number of testing data samples of that class. For example, if you are working on the problem of classifying 5 species of butterflies, per-class accuracy here means the accuracy of each species of butterfly.

3. **Confusion Matrix:** produce a confusion matrix that helps determine which classes were confused with which other classes.

4. **Additional Relevant Analyses:** use the information above to make statements about how well your model generalizes, and attempt to determine where the sources of error may have been. Use any visual aids that you think are relevant e.g. bar graphs, pie charts etc. For example, a bar graph of the accuracies per class can be very useful. Also, if you see from your confusion matrix that a specific class is consistently confused with another class, try to look at the data samples to determine if they "look" similar. This may not always be possible. Make statements about this if you can.

Note that all of these results need to be generated separately for each of the three techniques, but need to be done in **a single Jupyter Notebook ("groupx_testing.ipynb")**. So this notebook, once run from top to bottom, should systematically produce each of the results for each of the three techniques in a systematic and meaningful manner e.g. first for SVMs, then again for NNs, and then again for Logistic Regression.

### 3.2.2 Phase 2b: Demo Creation

Now that you've produced optimized models, produced testing results and analysed the results, what remains is to use the three trained models to create a demo script that allows users to actually use your system in real-time. This should be in a separate notebook called "groupx_demo.ipynb". So this means that you will need to include some form of an interface (it could be a command-line interface or a GUI – whichever you think makes more sense). The interface will need to allow the user to provide input pertaining to the classification problem i.e. most likely features to be classified, and then pass those features into each of the three trained models to carry out three separate predictions. It should then display the prediction to the user.

As a very simple example: assume that we are working on recognizing the species of fish based on the following features: [the average length, width and height of the fish, the number of fins, the dominant colour, the secondary colour, the eye-to-body ratio of the fish]; you have trained and optimized a SVM, NN and logistic regression model to recognize/classify the species of fish based on these features. The live demo should ask the user to input all of these features and then use the input features to make a prediction using each of the ML techniques, and finally display the predictions back to the user. Get creative. Maybe it won't be all that easy/intuitive for the user (especially during a demo) to type in each of the features one-by-one if they are many; maybe it make more sense to allow the user to put in a comma-delimited piece of text e.g. "2,3,5,2,green,yellow,0.12", and the text will be automatically parsed to read in the features and make the relevant prediction. Again, get creative. The goal is to demonstrate that your models work.

## 3.3 Phase 3: Presentation and Comparison

The final phase involves producing a presentation that presents all of the aforementioned phases i.e.:

1. **Training:** Present and demonstrate how you divided up your data set, and how you optimized and trained each of your three techniques. Each team member should cover his/her section.

2. **Testing:** Present and demonstrate how you carried out testing, what you tested, and what your results were. Very importantly, use this part of the presentation to compare the performance of the three techniques. Feel free to compare the techniques in whichever manner makes the most sense and sells them the best. This part has a strong element of "salesmanship". It is all about demonstrating your work and your results in a manner that shows how amazing your work is. Use the below as a guideline but, again, get creative:

    a. **Overall Accuracy:** Compare the techniques in terms of their overall accuracies. Which is doing the best overall / on average?

    b. **Per-Class Accuracy:** Ideally, it is desirable for the ML technique to be consistent across classes i.e. hopefully it doesn't do very well on some classes but poorly on some other classes; rather, hopefully all the classes are more-or-less close in terms of accuracy.

This shows that the technique is not class-specific. So, for each of the classes that you are classifying, compare the accuracies of the three techniques per class. Consider using a combined bar graph that shows the accuracy of each class for each technique. Make comments about how the techniques compare in this regard.

c. **Confusion Matrix:** If there are some classes that are being consistently confused with some other classes, is this the same case across all three techniques? If so, this may mean that the classes are intrinsically similar. If not, it may mean a specific technique is confusing them for some reason. Make comments about this.

d. **Any Other Relevant Analyses:** This is left to your (team) discretion. This phase is a bit of an art and involves "selling" your work. Get creative and show that your work is amazing.

3. **Demo:** Finally, demo your system live. Again, get creative about how you do this. Make sure that you clearly demonstrate how well you have set this up, and convince us that your models work.

## 4. Summary of Files to Be Created

For you benefit, the files to be created are listed below. At the end of the day, you should have all of the files below.

- groupx_dividedata.ipynb
  - groupx_train.dat
  - groupx_cv.dat
  - groupx_test.dat
- groupx_train_svm.ipynb
- groupx_train_nn.ipynb
- groupx_train_logreg.ipynb
- groupx_test.ipynb
- groupx_demo.ipynb

## 5. Weekly Consultations and Assessments, and Presentation

As mentioned before, there will be a total of three consultations / assessments over three weeks. Following the three weeks will be a presentation as described in the previous section. Below is the break down of content and mark allocation for these assessments:

**Week 1:**
This week will be more focused on consultation. Students will be expected to have made significant progress on both Phase 1a and Phase 1b. Lecturers will offer advice to students and students can seek help/advice on Phases 1a and 1b. Marks will be allocated to progress made but not necessarily to correctness/completeness of solution.

Phase 1a and 1b progress: **2 marks**

**Week 2:**
In this week, Phases 1a and 1b will be assessed as being complete and marks will be allocated to them.

Furthermore, students will be expected to have made significant progress on Phases 2a and 2b. Lecturers will offer advice to students and students can seek help/advice on Phases 2a and 2b. Marks will be allocated to progress made but not necessarily to correctness/completeness of solution of Phases 2a and 2b.

Phase 1a and 1b completeness and correctness: **13 marks**
Phase 2a and 2b progress: **2 marks**

**Week 3:**
In this week, Phases 2a and 2b will be assessed as being complete and marks will be allocated to them. Furthermore, students can consult with lecturers on their presentations if they will have made some progress on them. This will not be allocated marks, however.

Phase 2a and 2b completeness and correctness: **13 marks**
Consultation on Presentation: **0 marks**

**Week 4:**
This will be the final presentation. Team numbers will be randomly picked to speak in the session. The presentation will be worth a total of **10%** of the overall project. The presentation will be marked as follows:

Introduction to Team Members +
Who did what +
Introduction and Background on Your Classification Problem: **10%**

Description of Phase 1a Tasks: **10%**
Description of Phase 1b Tasks: **20%**
Description of Phase 2a Tasks: **20%**
Demo: **20%**
Q&A: **10%**
Timing: **10%**

Note that each of the relevant categories above will be marked in terms of:

1. Clarity
2. Audio/Visuals
3. Team Coordination

## 6. Final Mark Computation

As mentioned earlier, the project is worth 40% of the course (ML) mark. Therefore, to make up this 40%, your final project mark (out of 40%) = [Week 1 mark + Week 2 mark + Week 3 mark + (Presentation percentage / 10)]%. If you did everything right, you'll end up with 40%.