CS 412: Introduction to Machine Learning
Spring 2022
Programming Assignment 2
Due: April 28, 11:59:59pm

Description

In this assignment you will implement and evaluate several different machine learning algorithms. You will explore how different training techniques can affect performance, how we can transform a regression problem to classification and, finally, how a multi-class classification problem can be viewed as binary positive/negative taxonomy!

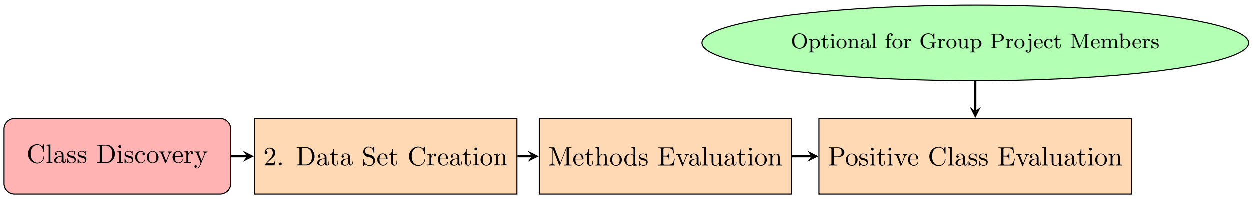


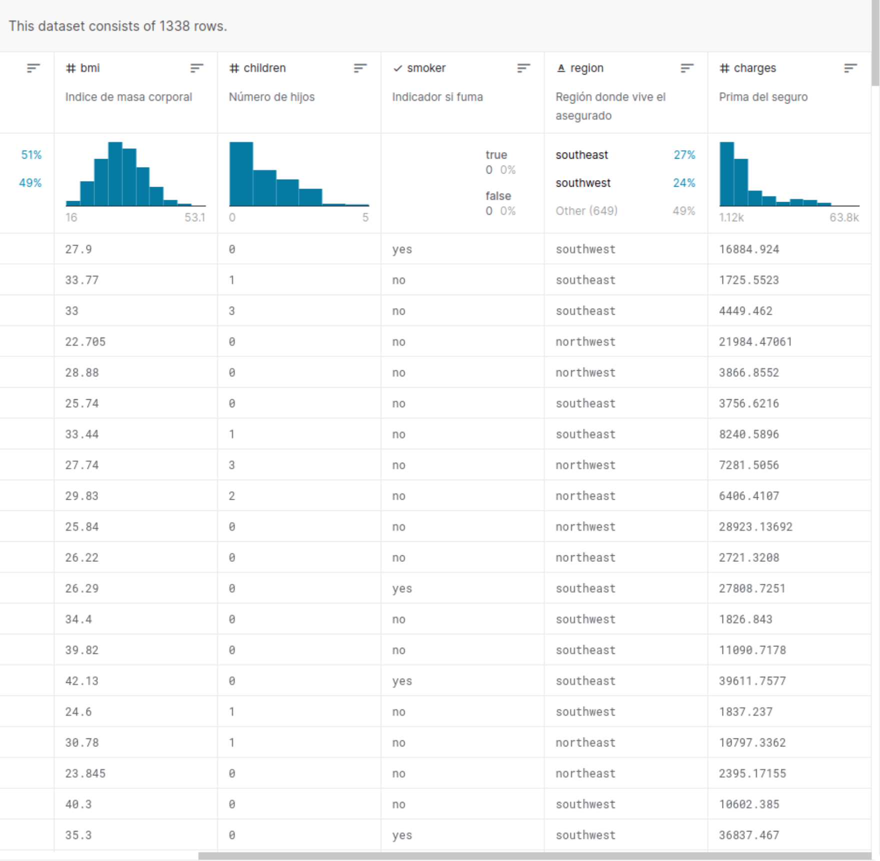
Figure 1: Overview of the required task chain

We will use the Medical Cost Dataset, found at Kaggle. This is a multivariate dataset, where the proposed goal is to predict the medical expenses an individual is expected to pay given 6 predictor variables (features). A snapshot of the data can be seen in figure 2. Instead of the proposed task however, you are asked to **classify** the expected medical cost in three distinct classes: **low, average, high**. The required tasks of this project, will guide you towards evaluating the performance of the algorithms presented in table, by first formating the dataset and then utilizing k-fold-cross validation. In this project you will be asked, in Task 3, to implement **all** classifiers labeled as 'Core' in table 1, then, if you **do not have a group project**, pick one from the elective methods, and **optionally** implement an Multi Layer Perceptron (or any Deep Net you like) for 10 bonus points.
Note: The methods in the core set **MUST** be implemented by you, no packages can be used, similarly for the (bonus) DNN; you can use PyTorch, but you must declare the architecture, write the training procedure etc. For the elective methods, you can use any package.

Type	Approach	Source
Core	Naive Bayes (NB)	Your implementation
	Maximum Likelihood (ML)[1]	Your implementation
	Maximum Aposteriori (MAP)[2]	Your implementation
	K- Nearest Neighbors (KNN)	Your implementation
Elective	Random Forests (RF)	Any Package
	Support Vector Machines (SVM)	Any Package or your implementation!
Bonus	Multi-layer Perceptron (MLP)	Your PyTorch implementation

Table 1: Algorithms for evaluation

[1] Refer to the Appendix for details
[2] Refer to the Appendix for details



This dataset consists of 1338 rows.

	# bmi	# children	✓ smoker	⚠ region	# charges
	Indice de masa corporal	Número de hijos	Indicador si fuma	Región donde vive el asegurado	Prima del seguro
51% 49%	16 53.1	0 5	true 0 0% false 0 0%	southeast 27% southwest 24% Other (649) 49%	1.12k 63.8k
	27.9	0	yes	southwest	16884.924
	33.77	1	no	southeast	1725.5523
	33	3	no	southeast	4449.462
	22.705	0	no	northwest	21984.47061
	28.88	0	no	northwest	3866.8552
	25.74	0	no	southeast	3756.6216
	33.44	1	no	southeast	8240.5896
	27.74	3	no	northwest	7281.5056
	29.83	2	no	northeast	6406.4107
	25.84	0	no	northwest	28923.13692
	26.22	0	no	northeast	2721.3208
	26.29	0	yes	southeast	27808.7251
	34.4	0	no	southwest	1826.843
	39.82	0	no	southeast	11090.7178
	42.13	0	yes	southeast	39611.7577
	24.6	1	no	southwest	1837.237
	30.78	1	no	northeast	10797.3362
	23.845	0	no	northeast	2395.17155
	40.3	0	no	southwest	10602.385
	35.3	0	yes	southwest	36837.467

Figure 2: Subset of the Data: Last columns holds the target.

Tasks

Task 1: Class Discovery

In this task, you are required to form classes out of the given data. In the provided dataset, you will notice that the target variable is given as a continuous dollar value, more suitable to regression than classification. However, as the task is to determine whether a patient will be expected to pay a 1) Low, 2) Medium or 3) High price during their visit, we need a way to map costs to classes.

(a) Load the data

(b) Gather all the values for the target variable.

(c) Develop a method that classifies an input value it into one of three classes: Low, Medium or High

(d) Your Method should be soundly rooted on the target data's underlying statistical profile [Hint: k-means with 3 classes (can be any package), assume Gaussian distribution and classify according to distance from mean (within 1 standard deviation can be Medium etc)]. **(5 points)**

(e) **(Report)** Explain your methodology. See written deliverables **(5 points)**

Autograder Expectations

> *Input: a Pytorch Long tensor that contains all the class labels as ”Low”=0. ”Medium”=1, ”High” =2, in order of original appearance. That is, sample 0 should appear first in your returned tensor.*

Task 2: Dataset Creation

In this step, you are required split the available data, using the class labels discovered in Task 1, into a Training, Validation and Evaluation(Test) sets. Develop a method that given a parameter $0.01 <= k <= 1$, splits the overall data into 3 sets: Training, Validation and Evaluation such that $|D| = 0.9 * (1 - k)Train + 0.1 * (1 - K)Validation + kEvaluation$. In the extreme case when K = 1, do a version of the Leave-one-out-cross-validation scheme, where the validation and evaluation sets are just one sample. When performing evaluation with this procedure, a method should be evaluated on each point once. Concretely, each data point should be assigned to the evaluation set exactly one time. For example, if k = 0.33 (alternative view of 3-fold), that means that 33% of the points are used as an evaluation set in each fold, so the procedure would be repeated 3 times where each 1/3 of the dataset considered exactly once as an evaluation set. **(10 points)**

Optional: Make sure there is proportional representation of all classes across the sets; i.e if class 'Low' is 10% of the overall data set, then both training, validation and evaluation sets should have 10% of the cardinality as 'Low'.

Note: Categorical data needs to be transformed, if they are going to be used in a meaningful way; a common approach is 1-hot encoding for data that does not accept an ordering, or enumeration for data that does.

Note 2: It is recommended that you normalize your data.

Autograder Expectations

> *Input: 3 PyTorch Tensors or 3 numpy ndarrays that are: TrainSet, ValidationSet, TestSet*

Task 3: Methods Evaluation

In this step, you are required to predict the class of each sample using the classifiers listed in table 1, using the class labels discovered in task 1. You can assume categorical features are independent of all continuous ones, for all those approaches that require the computation of covariance matrices.

For each classifier:

(a) Perform k = [0.33, 0.2] fold cross validation (by Task 2's function requirements)**(20 points)**

(b) Hyper parameters (such as Knn's K parameter, Learning Rate etc) should be chosen using the validation set

(c) Report the accuracy, per k-fold level **(10 points)**

(d) **(Report)** Jointly plot the results for each k-fold level on a single plot, for all classifers**(10 points)**

(e) **(Report)** Discussion. See written deliverables. **(10 points)**

Autograder Expectations

Input A python dictionary of the form returnDict = ['nb':[[acc list for k:0.3], [acc list for k:0.3], 'mle':[...], 'map':[...], 'knn':[...] + any chosen methods]

Task 4: Positive Class Evaluation (Optional for students with group projects)

In this step, we assume that we are only interested in predicting if a patient is expected to pay a high cost. We assume all samples with class labels 'High' are the positive class, whereas all the other are the negative class. For each classifier:

(a) Pick the best k-fold value and hyperparameters, as discovered in Task 2

(b) Evaluate the pre-trained classifier from Task 2, by using Precision, Recall and F1-scores as metrics **(10 points)**

(c) Retrain the best classifier on the new data, where class labels are only Positive, Negative. Report Precision, Recall and F1-scores. **(10 points)**

(d) **(Report)** Jointly plot the results for both pre-trained and re-trained models on a single plot, for all classifiers for the F-1 Score **(5 points)**

(e) **(Report)** Discussion. Is there any difference in their performance? If so, why? If not, why not? **(5 points)**

Autograder Expectations

*Input: A python nested dictionary of the form returnDict = {**'nb'**:{**'pretrained'**:{'precision': float, 'recall': float, 'f1': float}, **'retrained'**:{'precision': float, 'recall': float, 'f1': float}}, **'mle'**:[...], **'map'**:[...], **'knn'**:[...] + any chosen methods}*

Deliverables

Item 1: Code Submission to Gradescope

As in the previous assignment, submit a .py named "Lab_2.py" that contains all the function and necessary machinery for your code to run.

Item 2: Written Report

A two-Column Style, Eligible, well-written **PDF** file of up to two (2) pages, that satisfies the following criteria:

1. State whether you are working towards a group project.

2. Has a structure of:

   (a) **Dataset Creation**: Briefly Explain the methodology of your clustering.

   (b) **Methods Description:** Brief Implementation overview. What is chosen distance function for Knn? How did you handle categorical data? How did you implement MLE/MAP? How did you handle categorical data in Naive Bayes? What is your DNN architecture? SVM or RF what package did you use?

   (c) **Methods Evaluation:** For all methods: i) hyperparameters chosen; ii) The requested plot in that task iii) **Briefly** state why each method might fail or not, which is the best performing and why.

   (d) **Positive Class Evaluation**: i) State the best k-fold value and hyper parameters for each method ii) Joint plot (one plot only) of all the pretrained methods iii) Join plot of all the retrained methods iv) Discuss if there is any difference between the performance of ach method in the two cases, if you expected and why you think that is the case.

3. Experiments are well presented, in **tables** and **graphs**, properly labeled, annotated and captioned. For example if in deliverable requirement 2.b you chose to present your hyperparameters as a table, make sure the table is properly annotated to convey which method has which parameters and which k-fold lead to that choice.

Appendix

MLE and MAP Details

You can use your logistic regression (LinearPotentials) method from Lab 1 to perform Maximum Likelihood Estimation and Max Aposteriori. Below are three ways you can do this.

Method 1: Discriminant Functions:

1. Train a Logistic Regression model of each of the 'Low', 'Medium', 'High' Classes. For each class, treat the class of interest as 1, the rest as 0. Use binary cross entropy loss and Stochastic Gradient Descent to train.

2. To classify a sample, simply assign it at as the class that labeled it '1' with the highest confidence.

3. This method will require an alteration to you LinearPotentials class to output a scalar. You will then need to train 3 distinct models.

Method 2: Multiclass Classification:

1. Train a Logistic Regression that instead of outputting 0 or 1, it outputs a 1-hot representation for all classes. For 3 classes, a vector could look like [0.3, 0.6, 0.9]. Use negative log likelihood and Gradient Descent to train.

2. To classify a sample, simply use Softmax and assign it to the class that has the highest confidence. I.e if the models output is [0.3, 0.6, 0.9], using the softmax and selecting the index of the max element would yield [0,0,1] which is the third class of 'High'.

3. For this method you can use your LinearPotentials class.

Method 3: Gaussian Model Assumption:

1. Compute mean vectors μ_c and Covariance matrices Σ_c for each class $c \in C$

2. Produce a likelihood for each class, given a Gaussian multinomial Distribution

3. Assign a sample to the class with the highest likelihood.

For the **MAP** case you can weight the output of you model by the prior of each class.
For methods 1, 3 you should weight the output of each model before any comparisons to determine the winner.
For method 2, you should do this **before** selecting the max element and after softmax.