Predicting Legal Proceedings Status: an Approach Based on Sequential Text Data

Felipe Maia Polo[1], Itamar Ciochetti[2] and Emerson Bertolo[3]

***Abstract*— Machine learning applications in the legal field are numerous and diverse. In order to make contribution to both the machine learning community and the legal community, we have made efforts to create a model compatible with the classification of text sequences, valuing the interpretability of the results. The purpose of this paper is to classify legal proceedings in three possible status classes, which are (i) archived proceedings, (ii) active proceedings and (iii) suspended proceedings. Our approach is composed by natural language processing, supervised and unsupervised deep learning models and performed remarkably well in the classification task. Furthermore we had some insights regarding the patterns learned by the neural network applying tools to make the results more interpretable.**

I. Introduction

Machine Learning is present in many areas and is capable of performing the most diverse tasks with quality. One area that is already undergoing major changes and where there is still much room to work in the coming years is the area of law and justice. Many companies and researchers are developing new technologies to make legal processes increasingly efficient, creating great value for firms and consumers, especially democratizing services in developing countries. Law is a very wide area regarding its sub-areas and tasks, however Machine Learning applications have been proving to be very versatile, making a good deal in many of them. Examples of notable applications in the field of law would be reviewing documents, making text-based classifications or anticipating legal outcomes. 1, a recent review paper, provides insights on how Machine Learning can relate to various legal tasks, and the author highlights the importance that smart tools have gained in recent years assisting legal professionals.

In this paper we make extensive use of natural language processing (NLP) and machine learning tools to classify legal proceedings. Applications of NLP in the legal context are often challenging because legal texts are rhetoric, directed to persuasion and rarely descriptive, using figures of speech and other compositional techniques that challenge and twist a hypothetical "plain" and "neutral" meaning of terms, expressions and phrases. Therefore, our efforts were directed also to develop a classification model about a concrete fact – the status of legal proceedings according to a practical view –, disregarding theoretical discussions and queries on law about the nature of this status. We believe that the major contribution of this work is precisely the way we solve an important problem, described better in Section III, combining several types of techniques to analyze sequences of texts in chronological order, which are so common in the legal context. The results obtained were satisfactory both in terms of classification performance and interpretability, which also brings importance to this work.

[1]Department of Statistics, University of São Paulo, Brazil. Email: felipemaiapolo@gmail.com. Website: https://felipemaiapolo.github.io/

[2]Itamar Ciochetti, Tikal Tech, Brazil. Email: itamar@tikal.tech

[3]Emerson Bertolo, Tikal Tech, Brazil. Email: emerson@tikal.tech

II. Related work

Although there are some efforts to apply machine learning in the legal world, there have not been any, as far as we know, to solve a problem similar to ours, then we are going to talk about some applications that inspired us. Regarding the use of classic machine learning algorithms in the legal area we have the following examples. 2 make use of natural language processing tools to extract features such as N-Grams and Topics and then perform a binary classification task using Support Vector Machines (SVM) on whether cases referred to the European Court of Human Rights (ECHR) contain, in its report, any violated human rights article - the most optimistic accuracy rate was 84%. The authors of 3, on the other hand, were intended to perform three main prediction tasks that relate to cases judged by the French Supreme Court: (i) predict the legal area of a case, (ii) predict the court's decision based on the case description and (iii) estimate when the case description and a decision were issued. Results were 0.9 F1 score in the prediction of the legal area of a case, 0.96 F1 score in the prediction of a case decision and 0.76 F1 score for the third task. The methodology was composed by a Bag of Words feature extraction and Support Vector Machine (SVM) models for classification.

When using Deep Learning models, often the features extraction procedure is made in a more data driven way, endogenously and optimally[1]. If the amount of data is satisfactory and the computational power is not a limitation, a good use of these tools tend to give better results than those achieved by classic machine learning. 4, a recent Brazilian study, makes use of Convolutional Neural Networks to classify documents analyzed by the Brazilian Supreme Court (STF), achieving significant results. The classification proposed by the authors is made for six different classes, which were not translated to English by the authors. The authors then reached a result of 90.35 % accuracy and 0.91 F1 score. 5, another work, uses Multi-task deep learning models to accomplish three very important tasks when it comes to analyzing written documents: (i) translation, (ii) summarization, and

[1]With use of supervised and unsupervised algorithms.

(iii) document classification. The authors' approach, creating a multitasking model, allowed better performance in all three tasks compared to the performance obtained by isolated models.

III. Objective and practical importance of this work

The objective of this paper is to develop a model for the classification of legal proceedings in three possible classes of status: (i) archived proceedings, (ii) active proceedings and (iii) suspended proceedings. Each proceeding is made up of a sequence of short texts written by the courts that we will call "motions", which relate to the current state of proceedings, but not necessarily to their status. The three possible classes are given in a certain instant in time, which may be temporary or permanent, and are decided by the courts. In addition to focusing on the construction of a good classifier, we will also value the interpretability of the results achieved, given the importance of understanding the decisions made by models in the legal area.

These objectives and criteria have been chosen because they are a key feature to any task related to legal proceedings in Brazil. Although there are 90 different Courts in Brazil (State, Labour, Federal and others) – plus the Supreme Court –, all legal proceedings in Brazil must be included in one of the three presented classes (Archived, Active, Suspended). According to the National Council of Justice (CNJ) report[2], in the end of 2018 there were 64.6 million active proceedings and 14.1 were suspended in that year. In the same period of 2018, 31.9 million legal proceedings were definitively closed, that is, archived.

The three labels of interest (Archived, Active, Suspended) reflect the most practical classifications of the status of proceedings. Although in Procedural Law there may be some other subtle categories of analog status for proceedings (such as extinction and dismissal), the status of being archived, active or suspended is related to the activities of all personnel involved with these proceedings. For example, the suspension of a proceeding means that, even if not extinct (and therefore subject to reactivation of the same lawsuit), and from a practical view these proceedings are out of the judiciary routine of certain portfolios from courts, law firms, civil associations or legal aid organizations. In spite of the status of a proceeding being an objective information, sometimes it can be hard for public or private organizations to track it due to the size of their portfolios and because the information are mainly non-structured and can be spread in hundreds of separate individual Courts' web pages. It must also be noted that approximately half of the proceedings in Brazil have a small number of big players as plaintiffs of defendants, as usual in a contemporary mass litigation society. Thus, our work may help big public and private organizations to better handle their portfolios and will add value to Brazilian society as a whole.

[2]The report can be found in https://www.cnj.jus.br/wp-content/uploads/conteudo/arquivo/2019/08/justica_em_numeros20190919.pdf

IV. Data

Our data is composed by two datasets: a dataset of $3 \cdot 10^6$ unlabelled motions and a dataset containing 6449 legal proceedings, each with an individual and variable number of motions, but which have been labeled by law experts. As long as the motions have an specific format, we will give an example of a motion contained in our datasets. The sample motion is:

"Type of Motion: Ordinary Act Practiced Description: Be aware of the Court's record. Wait for the interested party's manifestation. Nothing being requested, the records will be forwarded to DIPEA."

The datasets we used are random samples from the first (São Paulo) and third (Rio de Janeiro) biggest State Courts. State Courts handle the most variable types of cases throughout the Courts in Brazil, and are responsible for 80% of the total amount of lawsuits. Therefore, these datasets are a good representation of a very significative portion of the use of language and expressions in Courts vocabulary in Brazil. Since classifying sets of texts is a complex task and our dataset of labeled proceedings is not very large, we used the unlabelled texts dataset for the embedding learning of words and expressions in the legal context and we used the second dataset to create a model for the legal proceedings classification. The distribution of the legal proceedings' labels can be seen in Table I. [3]

TABLE I: Distribution of legal proceedings' labels

	%	N
Archived (class 1)	47.14%	3040
Active (class 2)	45.23%	2917
Suspended (class 3)	7.63%	492
Total	100%	6449

V. Methodology

A. Text preprocessing

The first step before applying any Natural Language Processing or Machine Learning model to text is to preprocess the raw data obtained in text form. This step is crucial to the success of the application, avoiding, among other things, over-parameterization of the models used, which can undermine their performance. We applied the three points below, which are standard in the literature of NLP:

- *Uppercase to lowercase conversion*: uppercase and lowercase strings are understood as different things by the computer when applying the models. Often, having or not capitalized words in your body does not change the meaning of the text, as is the case at the beginning of sentences. In order to avoid the problem of model over-parameterization, we will standardize the words in the body of the texts by converting the uppercase characters to lowercase;

[3]The data can be found in https://doi.org/10.6084/m9.figshare.11750061.v1.

- *Stop words removal*: in many cases, some words add little information to the texts. We will evaluate which words will be removed without much loss of information in order to avoid the problem of over-parameterization. Words like "a" and "the" are some examples;
- *Noise removal and standardization of expressions*: noise removal or standardization is the removal of undesirable elements or standardization of expressions that may be intrinsic to the raw texts or arise by obtaining the data from the court's website. Examples are the conversion of the terms "state law" and "federal law" to "law" and the removal of punctuation and other undesired symbols;

B. Embedding learning

The construction of words and expressions embeddings in this work is completely unsupervised, given the small number of labeled text sequences - we then used a mass of $3 \cdot 10^6$ motion texts, all from unlabelled proceedings. Once we have the mass of preprocessed texts, the next step is to tokenize them - in this step we use the method proposed in [6] in order to identify presence words that generally appear together and which should be considered as unique tokens[4] Applying this methodology twice in sequence, with threshold=1, we could identify which sets of 2 to 4 words should be considered as unique tokens. After the tokenization of the texts, we then use the model specified in [7] (Continuous Bag of Words Word2Vec) (size=100, window=5)[5] and extract the vector representations for each of the tokens in the vocabulary. After obtaining each of the vector representations of terms and expressions, we normalize them to have a unitary euclidean norm, which will facilitate the interpretability of the classification model as we will show in Section V-E.

C. Representation of texts in a matrix form

Before describing the neural network used in the classification task, we need to understand what each text will look like after learning the embedded representations for the tokens in our vocabulary. First of all, it is important to remember that each legal proceeding we want to classify is consisted by a sequence of texts of varying length called motions. We are now interested in knowing the format of each of the texts in question. Each motion will be represented by a matrix of dimensions $R \times D$ where R is the maximum number of tokens allowed for each of the texts and D the size of the embeddings - in our case $D = 100$. We have noticed that over 90% of the motions have a maximum of 30 tokens, so we decided to set a ceiling of $R = 30$ tokens, selecting the first tokens, and completing texts that have less than R tokens using null vectors of dimension D (*zero-padding*). One can see in Figure 1 how we converted each of the texts to matrix form.

[4]This method is implemented in the Gensim package `https://radimrehurek.com/gensim/`.

[5]We tested many configurations, e.g. windows=5, 10, 15 and size=50, 100, 150, and we chose to work with the more parsimonious and most performing one, according to the classification results.

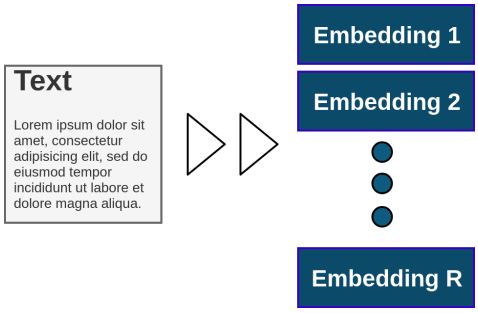

Fig. 1. Representation of texts in matrix form

D. Construction of the Neural Network for legal proceeding classification

Now that we know the format of the representation of each text that will use to feed our neural network, we can better explain how we developed a classification model that combines a Recurrent Neural Network (RNN) with Long Short-Term Memory units (LSTM) [8]) with convolutional filters [9] works. We mentioned that each legal proceeding is composed of a sequence of motions/texts and, as in the case of tokens, we needed to impose a ceiling on the number of motions/texts we would use. Our legal experience is that the last 5 motions contain enough information for our purpose, then we separated the last five (5) motions/texts from each of the legal proceedings and put them in chronological order, always putting the most current motions closer to the output layer, which is a Softmax function - those proceedings that had less than 5 motions available were completed by zero-padding matrices.

To extract features from each motion we used a convolutional layer with K[6] filters that run through each text. Onde the features are extracted by the filters, they pass through a ReLU activation function and then are selected according to the *max-over-time pooling* procedure proposed in [10], that is, we kept only one feature per filter - the one with the highest value. Each motion/text will be represented by only K numbers[7], that feed the Recurrent Neural Network with LSTM units with hidden state size H[8]. After processing the data using the RNN, the legal proceeding is then classified taking the greatest class probability according to the Softmax function. In order to give an interpretable appeal to the solution, in the learning process of the neural network, we constrain the euclidean norm of filters to be equal to one. Later in Section V-E and in Section VI we will show that we can easily compare filters learned by the network with the embeddings representations of tokens present in our vocabulary. There is an illustration of the neural network used in Figure 2.

[6]It will be determined through a cross validation procedure.

[7]Thus, each legal proceeding is represented by $5K$ numbers (5 motions and K features per motion)

[8]Also determined in a cross validation procedure.

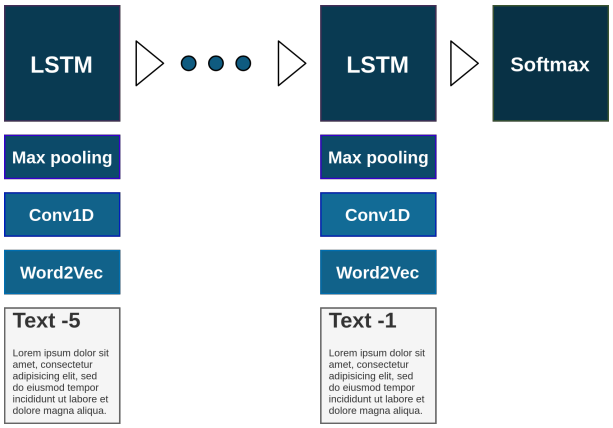

Fig. 2. Neural Network Architecture

1) Describing how our neural network works with more details: Now that we know how the network works in classifying legal proceedings, it's important that we present those ideas in mathematical language, which will help us make things clearer. Let (i) i be the index of a legal proceeding [9], (ii) $t \in \{-5, \dots, -1\}$ a index for a text/motion of i proceeding, where -1 denotes the most current text and -5 the least current text taken into account, (iii) $n \in [30]$ an index [10] of embedded tokens in the text t from proceeding i and (iv) $\mathbf{f}_k \in \mathbb{R}^{100}$ is the vector representing the k-th convolutional filter, $k \in [K]$. We then define the following quantity z_{itnk}, which is the feature extracted by the filter $\mathbf{f}_k$ from tokem $\mathbf{x}_{itn} \in \mathbb{R}^{100}$, that is, n-th tokem from t-th motion/text from i-th proceeding:

$$z_{itnk} = \text{ReLU}(\mathbf{x}_{itn} \cdot \mathbf{f}_k) \tag{1}$$

Note that we removed the constant neuron, which represents the bias. Furthermore, the final feature extracted by the $\mathbf{f}_k$ filter from the t-th motion/text from i-th proceeding right after applying *max-over-time pooling* procedure is given by the quantity z^*_{itk} as follows:

$$z^*_{itk} = \max\left\{z_{itnk}\right\}_{n=1}^{30} \tag{2}$$

Grouping those quantities through index k in an array, we have the following vector that we will use to feed our recurrent neural network with LSTM units:

$$\mathbf{z}^*_{i,t} = (z^*_{it1}, \dots, z^*_{itK}) \tag{3}$$

The probability vector of i-th legal proceeding belonging to one of the three possible classes/status, $\mathbf{p}_i$, is given by the function $\boldsymbol{h}$ which is a recurrent neural network (RNN/LSTM) with a time depth of 5:

$$\mathbf{p}_i = \boldsymbol{h}(\mathbf{z}^*_{i,-1}, \dots, \mathbf{z}^*_{i,-5}) \tag{4}$$

[9] i can represent an out of sample proceeding.
[10] Consider $[N] = \{1, \dots, N\}$, $N \in \mathbb{N}$.

Given that $\mathbf{z}^*_{i,-1}$ refers to the most current network input and $\mathbf{z}^*_{i,-5}$ refers to the least current input. For a class $j \in [3]$, we can also write the individual predicted probability as $p_{ij} = h_j(\mathbf{z}^*_{i,-1}, \dots, \mathbf{z}^*_{i,-5})$. It's not explicit, but this time, as well as all the others not mentioned, we included the constant neuron to take the bias into account.

E. Interpretability

1) What are the filters looking for?: In the process of feature extraction performed by the convolutional layer of the network, we have that each of the K filters go through all 30 embedding representations of tokens present in each text performing scalar products. As we discussed earlier, each of the embeddings representations and filters were constrained to have unitary euclidean norm and that means the scalar product between the filters and embeddings representations will give us the value of the cosine of the shortest angle formed between the vectors, i.e. the cosine similarity between they. Mathematically, we have:

$$\mathbf{x}_{itn} \cdot \mathbf{f}_k = \|\mathbf{x}_{itn}\| \, \|\mathbf{f}_k\| \cos(\theta_{itnk}) \tag{5}$$
$$= \cos(\theta_{itnk}) \tag{6}$$

If θ_{itnk} is the shortest angle formed between the vectors $\mathbf{x}_{itn}$ and $\mathbf{f}_k$. Thus, we can rewrite z_{itnk} as below:

$$z_{itnk} = \text{ReLU}\big[\cos(\theta_{itnk})\big] \tag{7}$$

Which equals to $\cos(\theta_{itnk})$ if $\theta_{itnk} \in [0, \pi/2]$. In the learning process, the network learns representations for filters that tend to minimize the loss (cross entropy) function when classifying. By constraining the vectors to have unitary euclidean norms, while learning the best weights for the convolutional layer, the network aligns[11] the filters representations to those representations of the tokens that help the most in the task of classifying legal proceedings. Then, analyzing the final representations of the filters, we can have insights on the patterns that the network looks for in the texts. In order to better understand what those patterns are, we are going to take a look at the tokens which have the closest representations to the filters according to cosine similarity.

2) How do features extracted by each filter relate to classification?: To interpret how each filter relates to the classification task, we will use the Partial Dependence Plots [12]. To explain the concept, we will first introduce a new notation. If y_i is a random variable that denotes the class of the i-th proceeding, then we can rewrite p_{ij} as follows:

$$p_{ij} = \hat{\mathbb{P}}\big(\mathrm{y}_i = j \;\big|\; \mathbf{z}^*_{i,-1}, \dots, \mathbf{z}^*_{i,-5}\big) \tag{8}$$
$$= \hat{\mathbb{P}}\big(\mathrm{y}_i = j \;\big|\; z^*_{i,-1,1}, \dots, z^*_{i,-5,K}\big) \tag{9}$$

Moreover, in order to help us define the partial dependence function, we will write $\mathbf{z}^*_i = (\mathbf{z}^*_{i,-1}, \dots, \mathbf{z}^*_{i,-5})$ as the concatenation of the vectors. When we want to talk about

[11] By 'aligning' we mean approximating according to the cosine similarity metric.
[12] See 11 for a more detailed explanation.

the features themselves, i.e. random variables/vectors, and not their instances in the i individual, we can rewrite z^*_{itk} as z^*_{tk}, $\mathbf{z}^*_{i,-1}$ as $\mathbf{z}^*_{-1}$ and $\mathbf{z}^*_i$ as $\mathbf{z}^*$. Given all these notations, the partial dependence function on z^*_{tk} feature predicting j class probability, with $t = -1$ and $k = 1$ for example, is given by:

$$g_{j,\mathrm{z}^*_{-1,1}}(z) = \mathbb{E}_{\mathbf{z}^* \setminus \mathrm{z}^*_{-1,1}}\Big[\mathbb{P}\big(\mathrm{y} = j \mid z, \mathrm{z}^*_{-1,2}, \dots, \mathrm{z}^*_{-5,K}\big)\Big] \tag{10}$$

With $\mathbf{z} \setminus \mathrm{z}^*_{-1,1}$ denoting the $\mathbf{z}^*$ vector without the first original feature. Here we work with the $\mathrm{z}^*_{-1,1}$ feature for pure practicality, but the definition is valid for any of the features. The empirical version of the partial dependence function for the same feature is given by the following:

$$\hat{g}_{j,\mathrm{z}^*_{-1,1}}(z) = \frac{1}{m}\sum_{i=1}^{m} \hat{\mathbb{P}}\big(\mathrm{y}_i = j \mid z, z^*_{i,-1,2}, \dots, z^*_{i,-5,K}\big) \tag{11}$$

In this paper, we will calculate this function according to the test set data and center it on zero, so it is easier to make comparisons between plots - then, we will be interested in average variations in the predicted probabilities of the j class given variations in an specific feature.

F. Hyperparameter tuning

Hyperparameters are parameters used to control the behaviour of algorithms and are not learned by the algorithms themselves [12]. We have chosen to keep some of the hyperparameters fixed and to tune the rest of them in a simple cross-validation procedure using the grid search approach. Table II shows a summary about the values tested or fixed for the hyperparameters that we worried about while building the model. In total, we had 280 combinations of hyperparameters.

TABLE II: Hyperparameters for the classification model

Hyperparameter	Values tested/fixed
Optimizer	Adam
Beta 1 (Adam)	0.9
Beta 2 (Adam)	0.999
Learning rate	0.005
# Epochs	50
Batch size	500
# Convolutional filters (K)	3, 5, 8, 12
LSTM hidden state size (H)	10, 30, 50, 75, 100
LSTM weights l1 penalization strength (λ)	.0, .0001, .0003, .0005, .0007, .0009, .0011, .0013, .0015, .0016, .0018, .002, .0025, .003

G. Training, validation and test sets

In order to train and assess our classifier, we splitted at random our labeled dataset in three parts: training set (70%), validation set (10%) and test set (20%). We used the training set to fit the model, the validation set to choose the best hyperparameters and the test set just to check the performance of the final model.

VI. Results

A. Hyperparameters

Our criteria to choose the best combination of hyperparameters was to choose those values who gave us the higher accuracy in the validation set. Out of 280 possible combinations of values we chose the following values for the tuned hyperparameters: (i) $K = 12$, (ii) $H = 10$ and (iii) l1-$\lambda = .0001$.

B. Proceeding classification task performance

The classifications were made by selecting the most likely class, given the features, according to the estimated model. In the process of training our final model, it was not possible to verify accentuated overfitting, due to the parsimonious architecture of the network, counting with only 2153 trainable parameters. Now it is necessary to evaluate the performance in the test set. The most straightforward way to get an overview of classifier performance in this case is by using the confusion matrix, which compares true labels with those predicted by the model. Here our classes are: archived proceedings (1), active proceedings (2) and suspended proceedings (3). It is possible to check in Figure 3 the joint distribution that characterizes the confusion matrix. It is possible to notice a great performance of the model with accuracy of 0.93 ± 0.01.

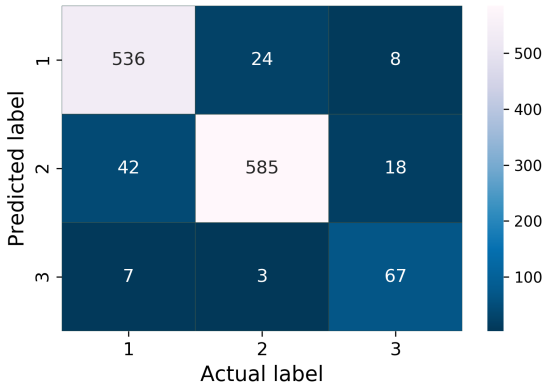

Fig. 3. Confusion matrix in the test set

In order to present more results and compare them to those obtained by similar alternatives, we will consider two other ways to extract features from the texts (other than convolutional filters), maintaining the recurrent neural network part, as it is important for us to take into account the chronological order of the facts. The other two ways to extract features are applications of the Doc2Vec [13] and TFIDF [14] models, which were trained beforehand in the unlabeled dataset. For the Doc2Vec alternative we kept the specifications for the Word2Vec model that we discussed in Section V-B. For the TFIDF alternative, we imposed a ceiling of 2000 tokens, keeping the more frequents in the corpus. For both alternatives we applied the processing steps described in Sections V-A and V-B. The hyperparameters for the alternative models were tuned as it is described in the

Supplementaty Material. First, we will assess the accuracy of the models in Table III, comparing our main model (CNN) with the benchmarks[13]:

TABLE III: Evaluation metrics by class

Feature extraction	Accuracy
CNN	**0.93 ±0.01**
Doc2Vec	0.85 ±0.02
TFIDF	0.92 ±0.02

It is possible to notice that our main model obtained the best accuracy, although it is not very different from the second best option. Next, one can look at Table IV which contains summary measures that characterize the model as a whole, which are the simple (macro-averaging) and weighted (micro-averaging) averages of other important metrics[14].

In general, we obtained excellent results with our main model proposal as well as the second best alternative, which is using the TFIDF model to extract features from texts. Despite our main proposal achieving a result more or less similar to another option, it is in its simplicity and interpretability that this solution stands out, as we will see next. First thing to point out is the number of learnable weights for the classification model according to each approach. In Table V one can see that our main model is much simpler, then less prone to overfitting and easier/faster to train.

TABLE V: Quantity of learnable weights by approach

Feature extraction	# Learnable weights
CNN	**2,153**
Doc2Vec	15,813
TFIDF	243,813

In the next section we will show how interpretable results can be easily obtained using our main approach.

C. Interpretability of results

1) What are the filters looking for?: In order to better understand what are the patterns extracted by the convolutional layer of the neural network, let's look at the embedding representations of tokens in our vocabulary which have the closest representations to the filters according to cosine similarity. As long as we have 12 filters in our model, which is a big quantity, we are going to focus in three specific filters (1, 9 and 11), which bring interesting results - the full results will be available in the Supplementary Material. In Table VII one can see which tokens[15] most closely resemble our filters after they are learned.

[13]The 0.95 confidence intervals were calculated using a bootstrap procedure.

[14]The 0.95 confidence intervals were calculated using a bootstrap procedure.

[15]Tokens were translated from Portuguese to English.

TABLE VI: Similarity between filters and their most similar tokens

Filters	Tokens	$\cos(\theta)$
1	*final storage of docket*	0.46
	final remittance to origin	0.45
	form registered in book	0.42
9	*emitted*	0.47
	certificate	0.43
	granted injunctions	0.42
11	*temporarily stored docket*	0.55
	docket remain in clerk	0.5
	return after granted period	0.45

One can see that the patterns sought by the neural network do have to do with the classifications we want to make, specially when looking to filters 1 and 11. For example, the expressions 'final storage of docket' and 'final remittance to origin' indicate archiving of proceedings (class 1) and the expression 'temporarily stored docket' may indicate suspension (class 3). We chose to present results for the filter 9, because it seems it is not looking up for very important patters and this will be clear in the next section.

2) How do features extracted by each filter relate to classification?: The patterns extracted by filter 1, in Figure 4, explain which legal proceedings are likely to be archived but not suspended or active, which can easily make sense when one sees those expressions linked to filter 1, e.g. 'final storage of docket' and 'final remittance to origin'. Regarding to filter 11, it is possible to notice that the partial dependence functions are decreasing in all plots but the one related to the suspended proceedings. This is understandable because the expressions linked to filter 11, as seen in Table VII, are more common to appear when a proceeding is suspended, e.g. 'temporarily stored docket'. On the other hand, patterns extracted by filter 9, presented in Figure 4, have almost no impact in the decision of the neural network as expected.

To conclude this section, we would like to highlight two points that we find most interesting regarding these results: (i) the results were very intuitive regarding the link between patterns search by the network in the texts and the output of the classification model and (ii) it is possible to notice that more recent information tends to have greater importance in the decision of the neural network, which makes sense in the legal context.

VII. Conclusion

This work aimed to develop a model for the classification of legal processes composed of sequential texts. During the development of the model, we wanted to have a model that performed very well on the classification task, had a parsimonious architecture and that we could gain insight into how decisions are made. We believe that the major contribution of this work is precisely the way we solve an important problem, which is classifying legal proceedings' status, combining several types of techniques to analyze

TABLE IV: Aggregate analysis of evaluation metrics

	Macro averaging			Micro averaging		
Feature extraction	F1 Score	Precision	Recall	F1 Score	Precision	Recall
CNN	**0.89 ±0.02**	0.92 ±0.02	**0.87 ±0.03**	**0.93 ±0.01**	**0.93 ±0.01**	**0.93 ±0.01**
Doc2Vec	0.82 ±0.03	0.85 ±0.03	0.8 ±0.03	0.85 ±0.01	0.86 ±0.02	0.85 ±0.02
TFIDF	0.88 ±0.02	**0.93 ±0.02**	0.85 ±0.03	0.91 ±0.01	0.92 ±0.01	0.92 ±0.02

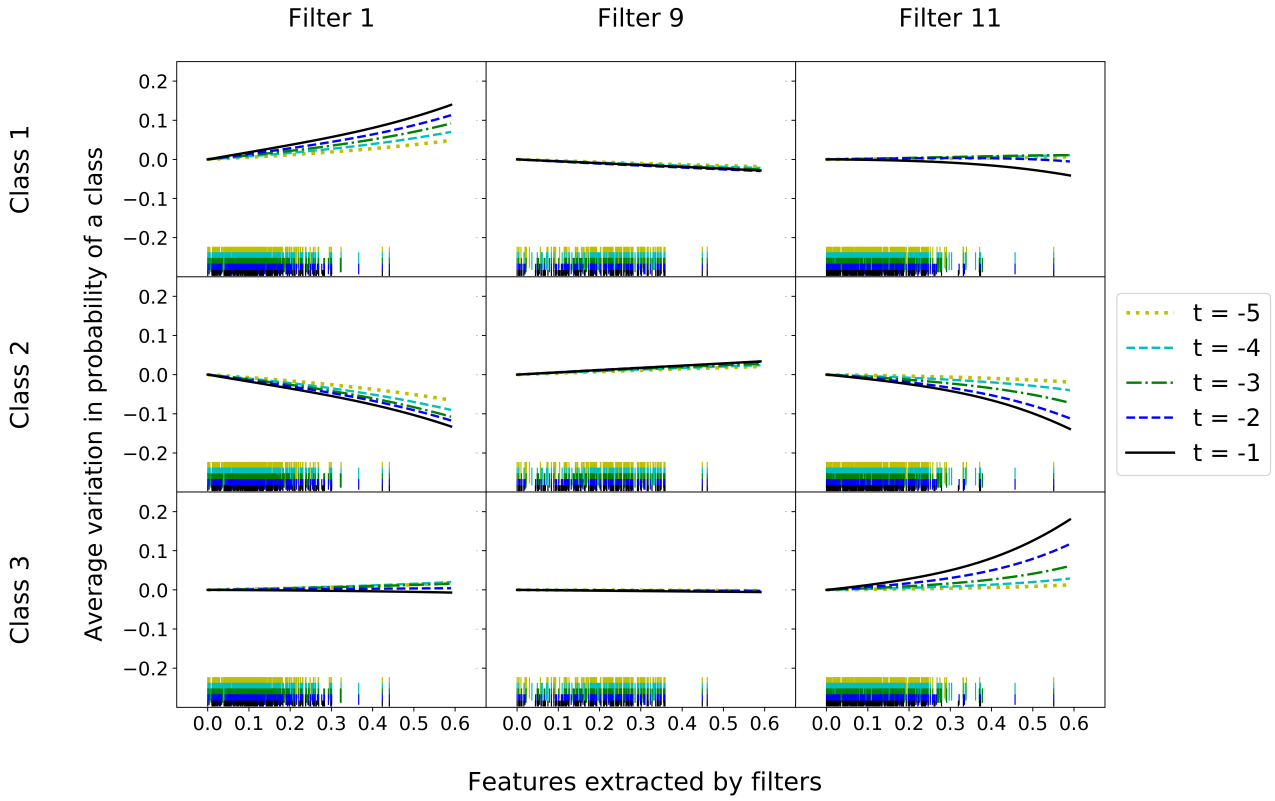

Fig. 4. Partial dependence plots: varying features extracted by filters 1, 9 and 11

sequences of texts in chronological order, which are so common in the legal context. The results obtained were satisfactory both in terms of classification and interpretability, which also brings importance to this work.

VIII. Code and Datasets

The code (Jupyter Notebooks) used in this work as well as the datasets can be found in `https://bit.ly/36yJZY3`. The data can also be found in `https://doi.org/10.6084/m9.figshare.11750061.v1`.

IX. Computing Infrastructure

Google Cloud VM instance with 24 vCPUs Intel Haswell, 90 GB of memory.

X. Acknowledgments

We would like to thank *Ana Carolina Domingues Borges*, *Andrews Adriani Angeli* and *Nathália Caroline Juarez Delgado* from Tikal Tech for helping us to obtain the datasets. This work would not be possible without their efforts.

References

[1] H. Surden, "Machine learning and law," *Wash. L. Rev.*, vol. 89, p. 87, 2014.

[2] N. Aletras, D. Tsarapatsanis, D. Preoţiuc-Pietro, and V. Lampos, "Predicting judicial decisions of the european court of human rights: A natural language processing perspective," *PeerJ Computer Science*, vol. 2, p. e93, 2016.

[3] O.-M. Sulea, M. Zampieri, M. Vela, and J. van Genabith, "Predicting the law area and decisions of french supreme court cases," *arXiv preprint arXiv:1708.01681*, 2017.

[4] N. C. da Silva, F. Braz, D. Gusmão, F. Chaves, D. Mendes, D. Bezerra, G. Ziegler, L. Horinouchi, M. Ferreira, P. Inazawam, *et al.*, "Document type classification for brazil's supreme court using a convolutional neural network," 2018.

[5] A. Elnaggar, C. Gebendorfer, I. Glaser, and F. Matthes, "Multi-task deep learning for legal document translation, summarization and multi-label classification," *arXiv preprint arXiv:1810.07513*, 2018.

[6] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.
[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, pp. 2493–2537, 2011.
[11] C. Molnar, *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.
[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
[13] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, 2014.
[14] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986.

XI. Supplementary Material

A. Hyperparameter tuning of benchmark models

The models used as performance benchmarks for our main model also had their set of hyperparameters tuner or fixed beforehand. In both models we fixed or tested the same combinations of values for the hyperparameters using the same values used in the paper for our main model. As long as we are not using a convolutional layer for the benchmark models, we only are interested in tuning the LSTM hidden state size (H) and the strength of the $l1$ regularizarion (λ) for the weights of the RNN/LSTM. For the model we used the Doc2Vec alternative to extract features the values chosen in order to maximize the accuracy in the validation set were $H_{\text{Doc2Vec}} = 30$ and $\lambda_{\text{Doc2Vec}} = .005$. On the other hand, when we used TFIDF to extract the features, the values chosen in order to maximize the accuracy in the validation set were $H_{\text{TFIDF}} = 30$ and $\lambda_{\text{TFIDF}} = .0016$.

B. Interpretability

In this part we will present the full results for the interpretability part. Although there are too much information in this section, there is nothing actually new, since we could present the most interesting patterns and results in the paper. In Table VII one can see which tokens[16] most closely resemble our filters after they are learned. The tokens marked with "*" were not related to the legal vocabulary or were not fully understood by us. In Figures 5, 6, 7 and 8 one can see how the features extracted by the filters from 1 to 12 relate to the classification task.

TABLE VII: Similarity between most similar tokens and filters

Filter	Tokens	cos(θ)
1	*final storage of docket*	0.46
	final remittance to origin	0.45
	form registered in book	0.42
2	*clerk*	0.51
	interlocutory appeal	0.48
	*são paulo clerk**	0.48
3	*non-legal - name of a certain clerk**	0.54
	*non-legal - name of a certain clerk**	0.51
	*non-legal - name of a certain clerk**	0.51
4	*temporarily stored docket*	0.47
	suspended	0.42
	docket received from storage	0.42
5	*originals*	0.56
	docket	0.52
	submitted	0.51
6	*Itaquaquecetuba County*	0.42
	*wvpv**	0.41
	original clerk	0.4
7	*interlocutory appeal*	0.45
	*non-legal**	0.43
	*non-legal - name of a certain clerk**	0.42
8	*final remittance to origin*	0.45
	final storage of docket	0.44
	remittance to origin	0.39
9	*emitted*	0.47
	certificate	0.43
	granted injunctions	0.42
10	*small claims courts*	0.44
	defense entered	0.41
	lack of standing from this point of view	0.41
11	*temporarily stored docket*	0.55
	docket remain in clerk	0.5
	return after granted period	0.45
12	*final storage central storage*	0.51
	final storage of docket	0.51
	non-reactivated proceeding	0.51

[16]Tokens were translated from Portuguese to English.

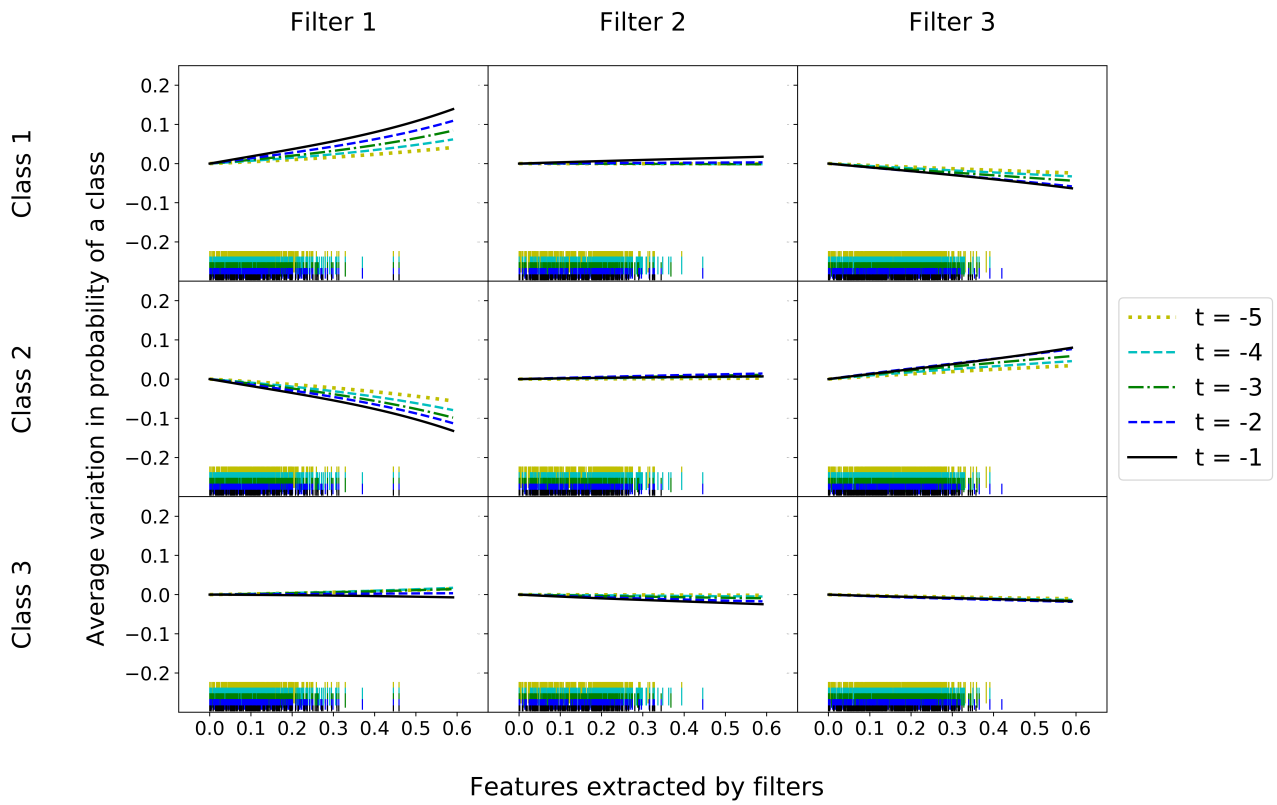

Fig. 5. Partial dependence plots: varying features extracted by filters 1, 2 and 3

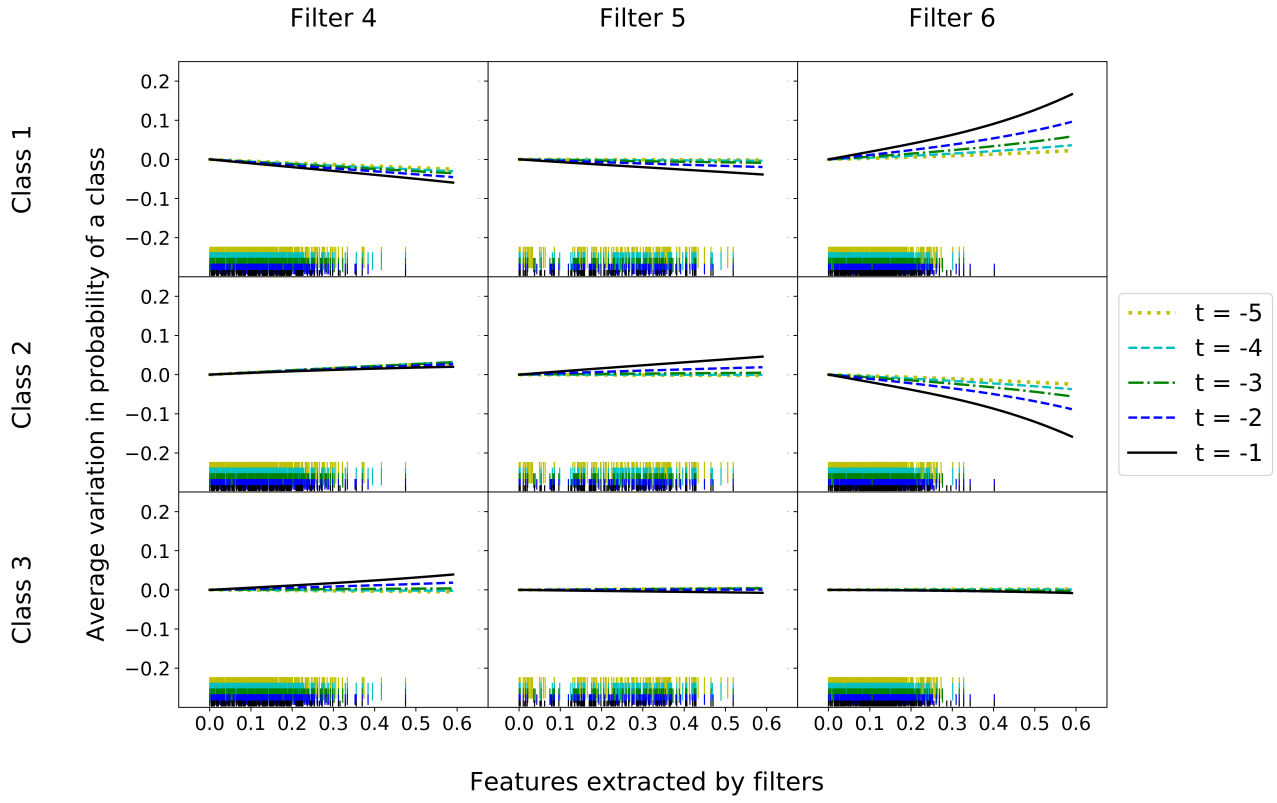

Fig. 6. Partial dependence plots: varying features extracted by filters 4, 5 and 6

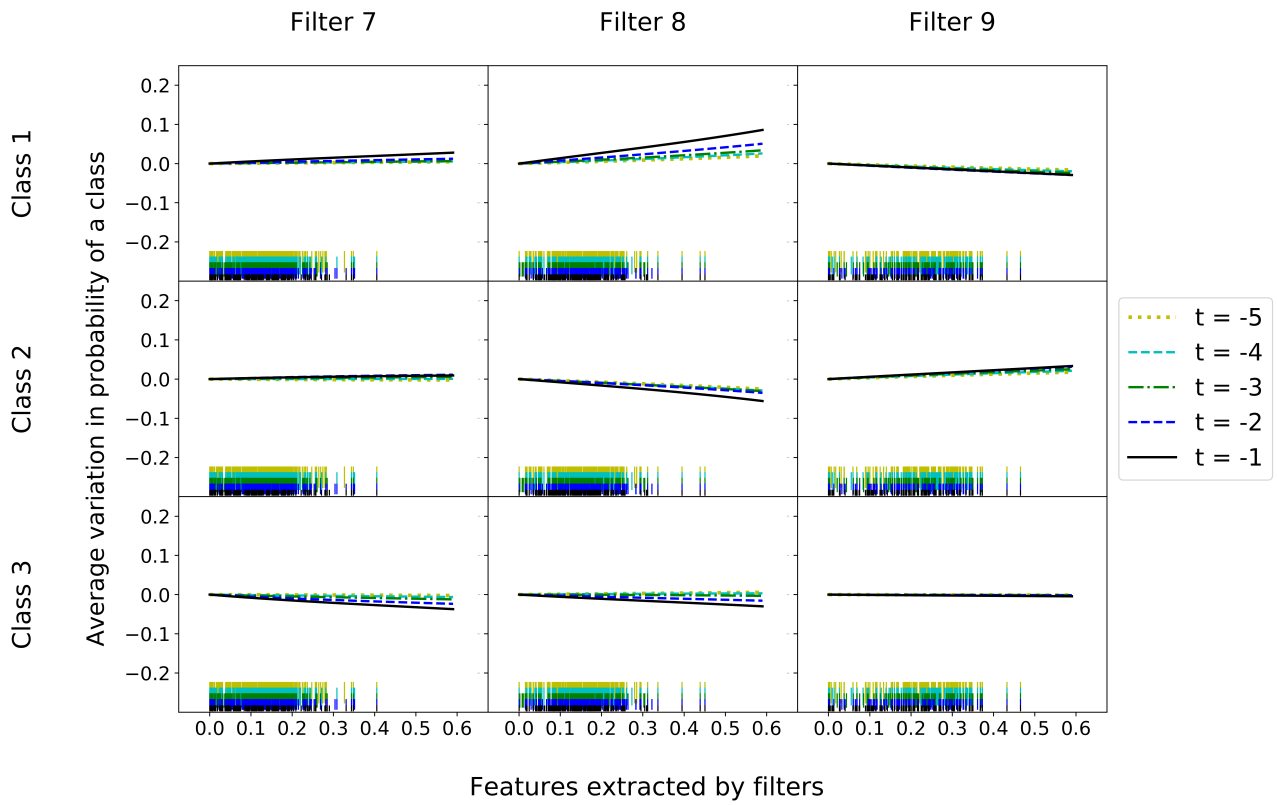

Fig. 7. Partial dependence plots: varying features extracted by filters 7, 8 and 9

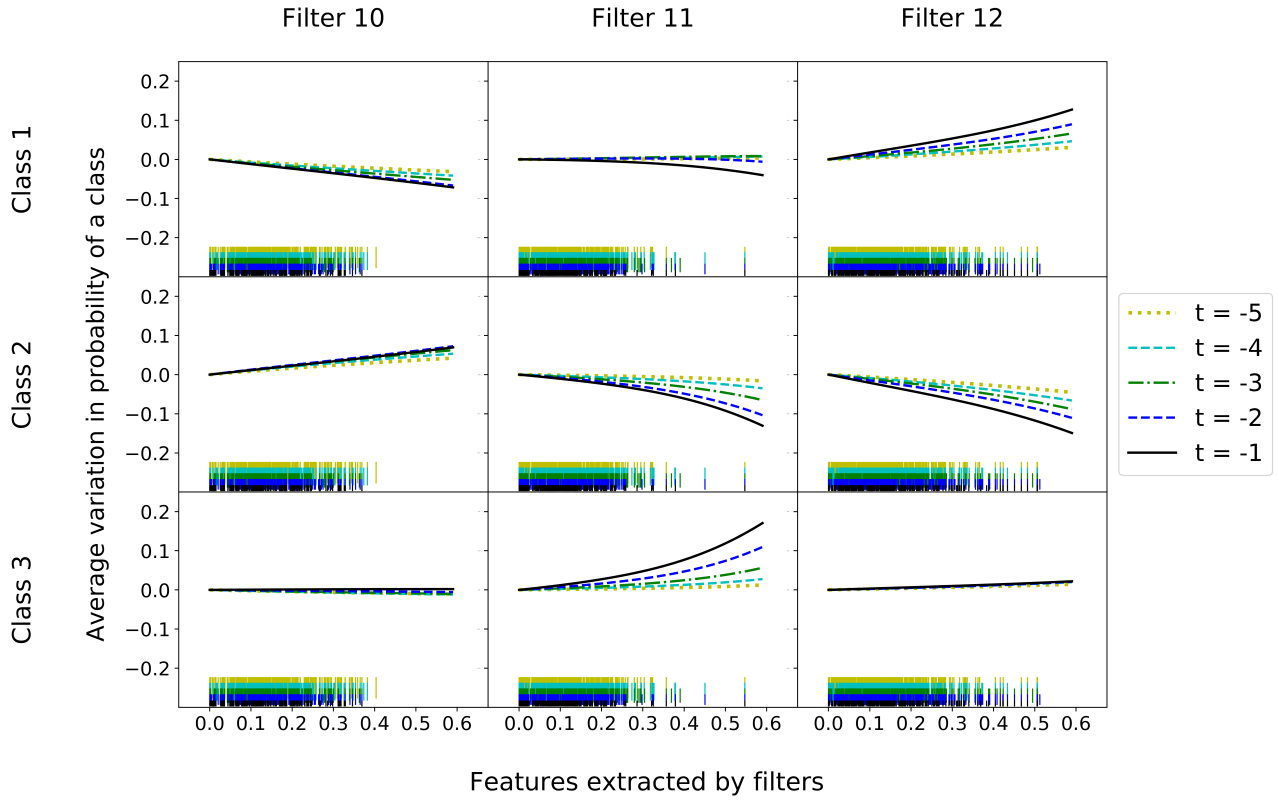

Fig. 8. Partial dependence plots: varying features extracted by filters 10, 11 and 12