Machine Learning techniques for malicious PDF detection

January 13, 2023

Lorenzo Ippolito, Mattia Rosso, Martino Picasso

Abstract

This project presents a thorough analysis of machine learning techniques for detecting malicious PDF files. By comparing the performance of four binary classification models - Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Decision Trees and Random Forests - we demonstrate the effectiveness of these approaches in achieving high accuracy and low false negative rates. What makes this project stand out is its focus on the machine learning models themselves, including the careful selection of hyperparameters to optimize their performance. Additionally, the inclusion of KNN as a valid model is rare in the field, and the analysis of evasion techniques adds another layer of depth to the study. Overall this paper, realized for the MALIS course at EURECOM (year 2022/2023), offers a comprehensive look at the use of machine learning for malicious PDF detection and the potential vulnerabilities of these models.

1 Introduction

PDF (Portable Document Format) is a file format used for exchanging and sharing documents. It can contain various types of elements, such as text, images, multimedia, and executable code. Keywords, also known as actions or triggers, specify the different types of contents and they can be used to detect whether the file is malicious or benign. In addition, PDF files may include also fonts, annotations, and form fields, which can be exploited for malicious purposes. We implemented four different binary classifiers to analyze their performances and to prove which one achieves the best results in terms of Accuracy and False Negative Rate. The task is a binary classification, with malicious PDFs labelled as Positive and benign PDFs being labelled as Negative. We also investigated evasion techniques against our classifiers concluding the work with the implementation of a countermeasuere algorithm known as Adversarial Learning.

2 Related work

Most of the publications on this topic make use of the Contagio dataset([1]) that is the one we have adopted as benchmark. We have considered some of them. Maiorca et al. [2] preprocesses data with PDFiD([3]) and uses SVM and Random Forest, Cuan et al. [4] uses both PDFiD as preprocessing and train a RBF SVM, they also explain how to perform evasion attacks and countermeasures. In [5] Li et al. explain the iterative Gradient Descent attack algorithm against RBF SVM.

3 Datasets and features

As first step in building our model, we needed to obtain a dataset of PDFs to use for training and evaluation and we went for the Contagio dataset. It is a collection of malicious PDF files that has been widely used for research and testing purposes. The dataset was created by the security firm Mandiant and contains about 20000 PDF samples, including malware, phishing attacks, and other types of malicious content. The samples in the dataset are representative of the types of threats that are commonly encountered.

We chose to perform a static analysis, which does not require opening the PDFs in a reader. Indeed, we used a tool called PDFiD to extract relevant information from the PDFs. PDFiD characterizes each PDF by extracting the number of occurrences of specific keywords. This allowed us to map each PDF into an array of 21 elements, with each element representing the count of a specific keyword thus we worked with samples of 21 integer features.

Among the keywords that we selected, some are related to the structure of the PDF file, such as *obj* and *endobj*, which mark the beginning and end of an object, *stream* and *endstream*, which mark the beginning and end of a stream of data. We also included *xref*, *trailer*, and *startxref*, which pertain to the organization of the file. Next, there are features related to the content of the PDF: we added */Page*, which specifies a page object in the file, and */Encrypt*. We also included */ObjStm*, which denotes an object stream, */JS* and */JavaScript*, which indicate the presence of JavaScript code. Additionally, we selected features related to actions that can be triggered by the PDF, such as */AA*, */OpenAction*, */XFA* and */AcroForm*, that could potentially exfiltrate information from users. We also included the features related to the types of data that may be present in the file, such as */JBIG2Decode*, */RichMedia*, */Launch* and */EmbeddedFile*. Finally we also chose */Colors*, which specifies the presence of more than 16 million colors.

The output of PDFiD is then converted into a CSV file, which can be easily accessed using the `numpy` and `pandas` Python libraries. This CSV file serves as the basis for our data extraction pipeline, which we have implemented using the functions provided by PDFiD. It is important to note that the keywords we selected are well known to be discriminant for PDF malware analysis but they do not represent the full set that may appear in a PDF file.

The plot in 2 shows that the features distribution is much different between Benign and Malicious PDFs. This will TODO

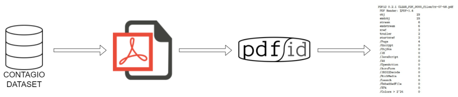

Figure 1: Features extraction pipeline

3.0.1 Model validation

We statically split the dataset in Train and Test using a stratified approach supplied by the `sklearn` library ([6]). The Training Set, which consists of 80% of the samples, is used to train and validate the models with a k-fold cross validation approach. The

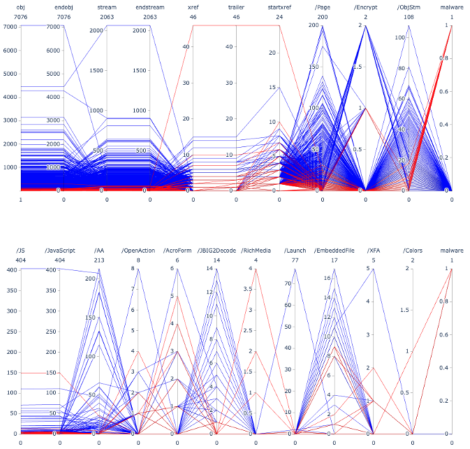

Figure 2: The parallel plot highlights the difference between the features of Benign PDFs (Blue) and Malicious PDFs (Red)

Test Set, which consists of 20% of the entire dataset, is used to make final considerations about the performance of the models trained with the best parameters/hyperparameters. To evaluate the performance of our models, we used two measures: the False Negative Rate (FNR) and the Accuracy where the FNR is defined as:

$$FNR = \frac{FN}{FN+TP}$$

We aimed to minimize the FNR, as False Negatives (Malicious PDFs classified as Benign) can be particularly damaging in this context. At the same time, we also took into consideration the value of the Accuracy, as the penalization of False Negative errors should not result in an excessively high number of False Positives that may lead to unnecessary alerts.

4 Support vector machines

Support Vector Machines are linear classifiers that look for maximum margin separation hyperplanes. The soft margin SVM problem adds a penalty term $C\sum_{i=1}^{n}\xi_i$ to the hard margin SVM to relax the constraint of having points inside the margin. The hyperparameter C is meant to define the strength of this penalization.

4.1 *Linear SVM*

We implemented a linear SVM classifier using the `sklearn` library([6]). The samples x_i are features vectors of 21 elements (the ones extracted from PDFiD).

4.1.1 *Experiments*

We have cross-validated the linear model by trying 10 different values for the C hyperparameter in the range $[10^{-6}, 10]$. As explained in section 3 we optimized for the False Negative Rate (FNR) with the constraint of obtaining a high accuracy anyway.

The plot in Figure 3 shows that the variations in terms of Accuracy and FNR when different values of C are used to train the linear SVM are not that evident. Here the results obtained with $C = 10^{-6}$ where not reported because of the significatively worse results (88.67% of Accuracy and 2.94% of FNR).

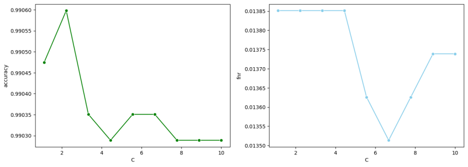

Figure 3: The plots show how the accuracy (left) and the FNR (right) change when the value of C hyperparameter changes

C	Accuracy	FNR
2.22	99.06%	1.38%
6.67	99.03%	1.35%

Table 1: Best model parameters for Linear SVM

Optimizing for the FNR the hyperparameter we have selected is $C = 6.67$.

4.2 *Kernel SVM*

What it makes the SVM particularly powerful is the way the dual problem is defined becuase it allows to easily introduce a kernel function to express the problem in a higher dimensional features space. Thus, we implemented a Radial Basis Function (RBF) kernel that is defined as:

$$k(x, y) = e^{-\gamma\|x-y\|^2}$$

4.2.1 *Experiments*

The cross-validation of the model took into account two different hyperparameters (γ for the RBF kernel and C for the dual SVM problem). This time we exploited a grid search k-fold cross validation that makes use of the CV to train the model with each possible combinations of hyperparameters. The hyperparameters values we tried are:

- γ: 10 values in $[10^{-6}, 1]$
- C: 10 values in $[10^{-6}, 10]$

We investigated the influence of both hyperparameters. What it is not displayed and analyzed in the plots is what happens with $C = 10^{-6}$ and $\gamma = 10^{-6}$, parameters under which the classifier performs very poorly in terms of accuracy (around 54%) despite achieving a FPR equal to 0%. However, this is not better than a dummy classifier that safely detects every PDF as malicious so it's not of interest in this project.

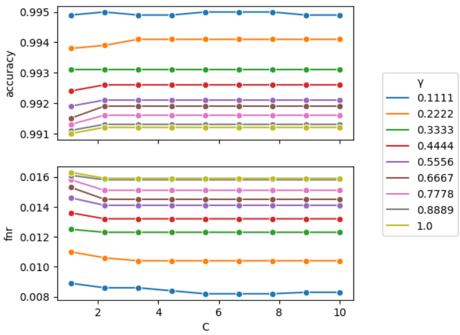

Figure 4: Accuracy (top) and FNR (bottom) when γ (colored lines) changes and C changes

In Figure 4 it's clear that smaller values of γ lead to obtain better results both in terms of Accuracy and FNR. The γ parameter of the RBF kernel controls its width and the shape of the decision boundary. A lower value of gamma means a larger, less concentrated kernel, which can lead to smoother decision boundaries. In contrast, the value of the C hyperparameter has clearly less influence on the model than γ as we can observe in Figure 4. What it does is controlling the trade-off between the simplicity of the model (low bias, high variance) and the training error (low variance, high bias). A higher value of C means a higher penalty for misclassification and the opposite for a lower value. When γ is sufficiently small the Accuracy reaches the best values and the same happens for the FNR.

Overall the best values we have found from cross validation are:

$$C = 5.56 \quad \gamma = 0.11 \quad Accuracy = 99.50\% \quad FNR = 0.82\%$$

4.3 Performances on the test set

Finally, given the best hyperparameters found in section 4.2.1 we assessed the performances achieved on the Test Set.

4.3.1 Linear SVM

The obtained results are:

$$C = 6.67 \quad Accuracy = 99.06\% \quad FNR = 1.58\%$$

The scores obtained are comparable with the ones achieved on the Training Set. Despite a smaller improvement in the Accuracy a slightly worse FNR has been obtained

4.3.2 Kernel SVM

The RBF SVM on the entire Test Set achieves these results:

$$C = 5.56 \quad \gamma = 0.11 \quad Accuracy = 99.73\% \quad FNR = 0.36\%$$

Here instead, we obtained better results over both Accuracy and FNR.

5 Decision tree and random forest

5.1 Decision Tree

The Decision Tree is a classification technique in which predictions are made in a sequence of single-attribute tests. Each internal node represents a test on an attribute, and each branch represents the outcome of the test. The paths from the root to the leaf nodes represent classification rules, and the leaf nodes represent class labels.

5.1.1 Experiments

We performed a k-fold cross validation with 5 folds in order to learn the best model parameters on the Training Set with more focus on *max_depth*. To summurize, the results we have obtained are reported in table 2. Despite the model achieves the best Accuracy with $max\_depth = 9$ we chose as best parameter the one that achieves the lowest FNR that is $max\_depth = 15$.

max depth	**Accuracy**	**FNR**
15	99.72%	0.21%
9	99.74%	0.28%

Table 2: Decision Tree - Optimal depths from cross validation

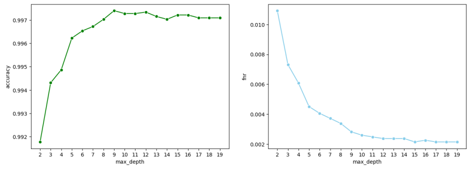

Figure 5: Accuracy (left) and FNR (right) when *max_depth* changes

5.2 Random forests

Random Forests is an ensemble method that combines the predictions of multiple decision tree models to make a more accurate and stable predictions. To train a Random Forests model, multiple Decision Trees are trained on different samples of the data and their predictions are combined. This helps to reduce overfitting and improve the model's generalization ability.

5.2.1 Experiments

We performed a k-fold cross validation with 5 folds in order to learn the best model parameters on the Training Set with more focus on *max_depth*. The plot in 6 shows how both the Accuracy and the FNR improve as the depth of the tree increases. Choosing a not too high value of depth is crucial to avoid overfitting.

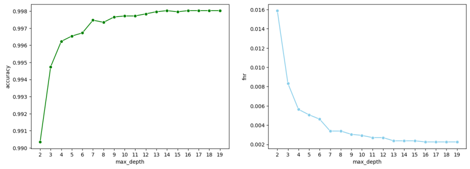

Figure 6: Accuracy (left) and FNR (right) when *max_depth* changes

The best obtained result is:

$$max\_depth = 16 \quad Accuracy = 99.80\% \quad FNR = 0.22\%$$

5.2.2 Results on the test set

The results obtained on the Test Set of both Decision Trees and Random Forests using the best model parameters discussed in 5.1.1 and 5.2.1 are:

Decision Trees

$$max\_depth = 15 \quad Accuracy = 99.75\% \quad FNR = 0.22\%$$

Random Forests

$$max\_depth = 16 \quad Accuracy = 99.90\% \quad FNR = 0.13\%$$

6 K-Nearest Neighbors

K-Nearest Neighbors (KNN) works by identifying the K number of training samples that are closest in distance to the input sample, and then assigning the input sample to the class of the majority of the K nearest neighbors.

6.1 Experiments

We performed a k-fold cross-validation with 5 folds. To determine the optimal value of K (the number of nearest neighbors to consider), we conducted a grid search over a range of possible values ($K \in [1, 8]$). The results of this search shows, as expected, that the best performances were achieved with $K = 1$. Since we have selected very discriminant features and we dispose of a large and diverse dataset, we expect that the possible overfitting effect will not be really high. To visualize the effect of increasing K on the performance of our model, we plotted in Figure 7 the accuracy and FNR as a function of K. The performance of the model decreases slightly as K increases. This is to be expected, as increasing K allows the model to consider more neighbors, which can introduce more noise and reduce the model's ability to accurately classify samples. The best result obtained is:

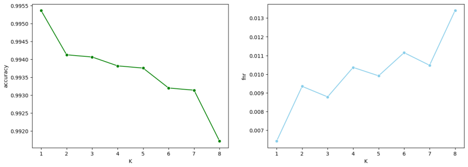

Figure 7: Accuracy (left) and FNR (right) when K changes

$$K = 1 \quad Accuracy = 99.67\% \quad FNR = 0.82\%$$

6.2 Performances on the test set

The performances on the Test Set are:

$$K = 1 \quad Accuracy = 99.68\% \quad FNR = 0.36\%$$

We also obtained excellent results on the Test Set. We could explain the fact that $K = 1$ is still the best model parameter by noticing that, despite the dataset is made of PDFs that are unique, PDFiD is able to extract such discriminant features that different PDFs result to be equal in terms of the extracted features vector. Having overlapped samples in the features space is really helpful for the way KNN classifies because the nearest neighbor of a test sample will probably consist in different overlapped samples, most of them belonging to the same class. Our results demonstrate that KNN is an effective technique for detecting malicious PDFs, and that it can achieve high level performances with relatively little tuning.

7 Final results

We have summarized the results of all our classifiers in table 7.

The table shows that, as widely discussed in the previos sections, the model that achieves the best results is the Random Forests. Actually, all of them reach an impressively high accuracy and low FNR. This is also coherent with the results obtained in Corona et al. ([2]) that proved Random Forests to be the best performing model. Also the other works cited achieve comparable results.

Finally we also plotted the ROC curves for all the models. In Figure 8 we can confirm the results showed in table 7.

8 Attacking the classifiers

In this section, we present two evasion attacks against the classifiers we built. We increased the number of objects in malicious PDFs so that they will be detected as Benign by the classifiers (still preserving their malicious behaviour). We operated in a condition in which the attacker is equipped with the Training Set, the type of classifier and the model parameters used to train it.

	Train		***Test***	
Model	**Accuracy**	**FNR**	**Accuracy**	**FNR**
Tree *depth* = 15	99.72%	0.21%	99.75%	0.22%
Forest *depth* = 16	99.80%	0.22%	99.90%	0.13%
KNN $k = 1$	99.67%	0.82%	99.68%	0.36%
LSVM $C = 6.67$	99.03%	1.35%	99.06%	1.58%
KSVM $C = 5.56$ $\gamma = 0.11$	99.50%	0.82%	99.73%	0.36%

Table 3: Overall results of all the models (LSVM: linear SVM, KSVM: RBF Kernel SVM). Results on Train are the ones obtained through k-fold cross validation

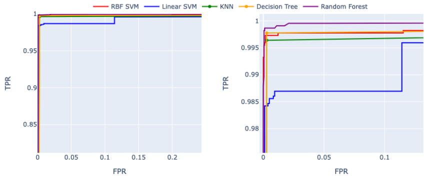

Figure 8: ROC curve comparing all the different models.

8.1 Attack against SVM

We have studied an evasion attack based on the Gradient Descent algorithm following the idea of Cuan et al. in [4]. Given a feature vector x characterizing a malicious PDF correctly classified, our goal is to find a x' that will be detected as Benign. The constraint is that x' is obtained with a minimal perturbation over x. The function to be minimized is the L1 distance between x' and x:

$$\|x - x'\| = \sum_{i=1}^{d} x_i - x'_i$$

The Gradient Descent attack converges to the minimum iteratively. Following the approach of Li et al. in [5] the feature vector is moved in the negative direction of the gradient, that means that the Gradient Descent update step becomes:

$$x_{t+1} = x_t - \mu_t \nabla_x f_\Phi(x_t)$$

By recovering that $w = \nabla_x(w^T x_t + w_0)$ for a linear SVM it can be derived that $w = \nabla_x f_\Phi(x_t)$ in a kernel SVM. We obtained the expression for a RBF kernel SVM from the dual problem definition:

$$\nabla_x f_\Phi(x_t) = \textstyle\sum_{i \in SV} \alpha_i z_i \nabla_x k(x_i, x_t) \text{ where}$$
$$\nabla_x k(x_i, x_t) = \nabla_{x_t} e^{-\gamma\|x_i - x_t\|^2} = -2\gamma\alpha_i z_i (x_i - x_t) e^{-\gamma\|x_i - x_t\|^2}$$

To better get the idea behind the attack we trained a RBF SVM ($C = 1$, $\gamma = 0.1$) on the dataset preprocessing it with PCA (with 2 components) just to visualize it. In Figure 9 a random malicious PDF is perturbated iteratively (using a learning rate for the Gradient Descent of $\mu = 0.03$) and it is successfully attacked. We only inspected the working mechanism of this

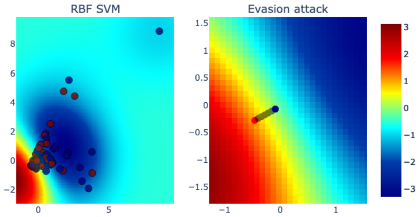

Figure 9: Decision boundary of RBF SVM($C = 1$, $\gamma = 0.1$) on PCA=2 dataset (left), Gradient Descent attack on a random malicious PDF (right). (Blue: benign, Red: malicious)

evasion attack by trying to successfully craft benign PDFs from malicious ones.

8.2 *Attacks against Decision Trees*

We also investigated an attack against the Decision Trees classifier. Firstly we found a way to craft malicious PDFs that are detected as Benign (using a different approach with respect to the one used in 8.1 that is explained in the Pseudocode in 1).

Algorithm 1 Evasion Attack

```
Require: Train DecisionTree using $X_{train}$
  $M \leftarrow X_{train}[malicious]$
  $t \leftarrow 50$
  for $m_i \in M$ do
      for $feature_j \in m_i$ do
          while $feature_j < t$ do
              $feature_j \leftarrow feature_j + 1$
              $m_i[j] \leftarrow feature_j$
              $benign \leftarrow predict(m_i)$
              if $benign$ is $True$ then
                  $break$                                  ▷ PDF evaded
              end if
          end while
      end for                                          ▷ PDF not evaded
  end for
```

With a threshold $t = 50$ we obtained a success rate of evasion attack on the Training Set equal to 91.28%.

8.3 *Adversarial learning*

In order to improve the strenght of the Decision Trees classifier against the attack procedure in 8.2 we implemented a defense mechanism similar to what has been implemented by Cuan et al. in [4]. It is called Adversarial Learning and the algorithm is shown in Pseudocode 2.

The results we obtain are overall satisfying because the classifier is able to defend perfectly against our evasion algroithm.

The plot in Figure 10 shows that the number of PDFs that can be evaded in the Training Set reaches 0 at iteration 69.
We can also see in Figure 11 how the features of the improved dataset (the ones obtained adding the crafted PDFs to the original Training Dataset) have been modified to defend against this attack.

Algorithm 2 Adversarial Learning

```
Require: Train DecisionTree using $X_{train}$
  $t \leftarrow 50$
  $X^{imp}_{train} \leftarrow X_{train}$
  $M \leftarrow X_{train}[malicious]$
  while $\#PDFevaded > 0$ do
      $evaded, X^{evaded}_{train} \leftarrow evade(M, t)$
      if $evaded$ is $True$ then
          $X^{imp}_{train} \leftarrow X^{imp}_{train} + X^{evaded}_{train}$
      end if
      $train(X^{imp}_{train})$
  end while
```

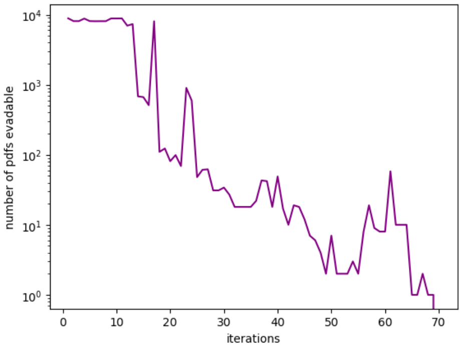

Figure 10: Adversarial Learning defense effectivness in different epochs: although the different spikes the number of evadable PDFs of Training Set reduces after each iteration

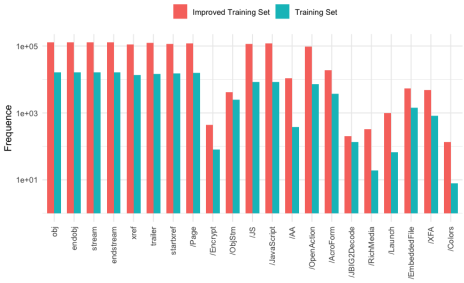

Figure 11: Difference of features frequencies between X_{train} and X^{imp}_{train} (logarithmic scale)

9 Conclusions and future works

Classifiers To conclude, the model that performs better is the Random Forests both in terms of Accuracy and FNR. Comparable results are achieved by the others and the use of PDFiD makes all the algorithms efficient in detecting patters using the most discriminant features. Further improvement could be done in order to achieve even lower values of FNR. Dimensionality reduction techniques could be exploited toghether with other types of metrics that take into account both the Accuracy and the FNR.
Evasion Attacks The evasion attack against the Decision Trees showed a really high success rate on the Training Set. We thus implemented the defense mechanism that proved to be very effective using Adversarial Learning. We could investigate more the size of the improved training dataset nedded to mitigate the attack.

10 Contributions

Contributions of each member:

- L. Ippolito worked on Decision Trees and Random Forests classifiers carefully exploring all the model parameters.
- M. Picasso studied the KNN classifier to better understand the reason why $K = 1$ is the best performing parameter.
- M. Rosso worked on SVM classifier investigating the role of the different hyperparameters and the dual formulation of the soft margin SVM to be used in the evasion attacks.

We collaborated both on features extraction and choice of the model validation approach. After having assessed the performances of the models we collaborated on designing the evasion attacks algorithms and countermeasueres.

References

[1] Contagio dataset. `http://contagiodump.blogspot.com/`. Accessed: 2022-01-16.

[2] Davide Maiorca, Giorgio Giacinto, and Igino Corona. A pattern recognition system for malicious pdf files detection, Jan 1970. URL `https://link.springer.com/chapter/10.1007/978-3-642-31537-4_40`.

[3] Stevens, d.: Pdf tools. `https://blog.didierstevens.com/programs/pdf-tools/`. Accessed: 2022-01-16.

[4] Bonan Cuan, Aliénor Damien, Claire Delaplace, and Mathieu Valois. Malware detection in pdf files using machine learning. *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications*, 2018. doi: 10.5220/0006884705780585.

[5] Wanman Li, Xiaozhang Liu, Anli Yan, and Jie Yang. Kernel-based adversarial attacks and defenses on support vector classification. *Digital Communications and Networks*, 8(4): 492–497, 2022. doi: 10.1016/j.dcan.2021.12.003.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.