Procedimento para escolha de métodos de classificação em Aprendizado de Máquina

Felipe Henrique Faria
Universidade Federal de São João Del Rei
felipevisu@gmail.com

Abstract

Este trabalho desenvolve uma metodologia para escolha e comparação de métodos de classificação em aprendizado de máquina. Para isto são comparados três algoritmos diferentes, aplicando-se técnicas de pré processamento, calibragem dos parâmetros e análise dos resultados.

1 Introdução

Aprendizado de Máquina (AM) é um tecnologia que permite aos computadores desenvolverem a capacidade de realizar determinada tarefa, sem que tenham sido especificamente programados para isto.

Apesar de os algoritmos de AM existirem a bastante tempo é com o advento da Big Data que o aprendizado de máquina ganhou força e se tornou uma área de intensa pesquisa nos últimos anos. A necessidade de extrair informações em grandes quantidades de dados, de forma que o computador possa aprender a tomar decisões, motiva este estudo.

Diversas áreas da indústria aplicam o aprendizado de máquina para optimizar suas tarefas. Dentre elas é possível citar: filtro de SPAN na caixa de e-mails, aprovação automática de crédito para empréstimos em bancos, predição de valores em aplicações na bolsa, reconhecimento de padrões em fotos e textos, entre outros.

Com esta enorme variedade de cenários de aplicação, configurar corretamente um sistema de aprendizado de máquina não é uma tarefa fácil. Os métodos de AM se concentram principalmente em duas categorias: aprendizado supervisionado e aprendizado não supervisionado. Cada segmento possui uma gama de algoritmos que podem ser aplicados a diversos cenários diferentes. A aplicação de um algoritmo de AM para um cenário especifico implica ainda encontrar qual a melhor configuração dos parâmetros do algoritmo, isto é, para qual combinação de valores nos parâmetros de entrada, o algoritmo obtém sua melhor performance.

Além disso, os dados de entrada para o aprendizado do algoritmo escolhido não saem de seu ambiente prontos para serem utilizados. Diversos dados precisam passar por uma etapa de pré-processamento para que o AM seja capaz de interpreta-los de forma eficiente e obter bons resultados.

Configurados os parâmetros e pré-processados os dados o AM ainda precisa ser treinado e avaliado. Existem diversos métodos para a treinamento, e estes serão abordados na próxima seção. Após todo este processo, o cientista de dados precisa também de conhecimento suficiente para avaliar os resultados obtidos e então ser capaz de escolher o melhor algoritmo para o problema em que trabalha.

Com foi visto até aqui, configurar um ambiente de aprendizado de máquina é um processo que envolve várias etapas que são dependentes entre si. Este trabalho tem o objetivo de guiar o processo de aplicação de técnicas de AM, servindo de referencia para aqueles que buscam iniciar seus estudos no assunto.

1.1 Algoritmos escolhidos

Este trabalho estuda algoritmos de AM do tipo supervisionados, mais precisamente, algoritmos de classificação. Os algoritmos escolhidos para análise são:

- Árvores de decisão;
- Máquina de vetor de suporte;
- Vizinho mais próximo.

1.2 Base de dados

Para testes dos algoritmos escolhidos, foi utilizada a base de dados Titanic. Esta base possui informações sobre sobreviventes ao famoso naufrágio, baseando-se em informações como, sexo, idade, classe da viagem. A base possui um total de 890 elementos.

O resultado da aplicação do AM é um modelo de predição que retorna se determinada pessoa sobreviveria ou não ao naufrágio, com base nas informações descritas à cima.

A escolha da base de dados foi motivada pela variedade de tipos de dados presente nela, o que possibilita aplicação de diversas técnicas de pré-processamento, além de obter bons resultados em diversos algoritmos. A base de dados encontra-se disponível em: https://www.kaggle.com/c/titanic.

2 Fundamentação teórica

2.1 Aprendizado supervisionado

O aprendizado supervisionado é o tipo em que dado um conjunto de valores de entrada para o algoritmo, os valores de saída que os resultados da análise podem assumir, são previamente conhecidos. Dentro deste paradigma ainda existem outras duas abordagens: a classificação e a regressão.

Classificação

O processo de classificação consiste em definir a que categoria ou grupo se encaixa a entidade à qual pertence os dados de entrada. O resultado portanto, assume sempre valores discretos previamente conhecidos, uma lista de categorias ou valor binário: sim ou não.

Regressão

Já os resultados de uma regressão, assumem valores contínuos (preço, peso, temperatura, ...), no entanto este não é o foco deste trabalho, que tem fonte de estudo algoritmos de classificação.

2.2 Aprendizado não supervisionado

Esta abordagem ocorre em cenários onde apenas existem os valores de entrada, não são conhecidos os valores que podem ser assumidos na execução, sendo este então o objetivo: encontrar grupos que relacionem as entidades entre si, o que também é chamado de clusterização, que é o processo de encontrar perfis, reconhecer padrões e itens semelhantes.

Embora não seja de interesse, neste, caso, ao realizar a comparação entre os tipos de aprendizado, alguns altores defendem que as técnicas de aprendizado supervisionado geram melhores resultados que os do aprendizado não supervisionado [1].

2.3 Algoritmos de classificação

Árvores de decisão

Em computação, árvores são uma estrutura de dados bastante utilizada. Composta por um conjunto de nós que armazenam as informações, e um conjunto de arestas que conectam os nós entre si através de ramificações. Esta estrutura é ilustrada na figura 1.

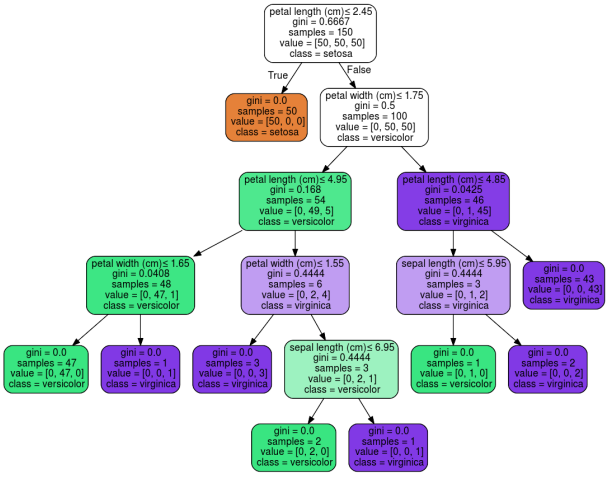

Figure 1: Árvore de decisão para o dataset Iris. Fonte: Scikit-Learn

É possível observar que árvore é composta por um nó raiz e ao final de cada ramificação existem os nós terminais, ou nós folha. Em uma árvore de decisão, uma decisão é tomada percorrendo um caminho que vai da raiz à alguma determina folha. O nó folha contém o resultado da analise de predição realizada.

Dentre as vantagens deste algoritmo estão a facilidade de compreensão e ilustração de seu método e uma boa sensibilidade a ruídos e dados faltantes.

A principal desvantagem é a ocorrência de overfitting, é o co que ocorre quando uma instância analisada não consegue caminhar até um nó folha, ficando sem resultado.

Máquina de vetor de suporte, SVM

Este algoritmo bastante utilizado em problemas de classificação pode ser descrito da seguinte forma: dadas duas classes e um conjunto de pontos que pertencem a essas classes, uma SVM determina o hiperplano que separa os pontos de forma a colocar o maior número de pontos da mesma classe do mesmo lado, enquanto maximiza a distância de cada classe a este hiperplano.

A menor distância entre uma classe e um hiperplano é denominada margem de separação. O hiperplano gerado pela SVM é determinado por um subconjunto dos pontos das duas classes, chamado vetores suporte.

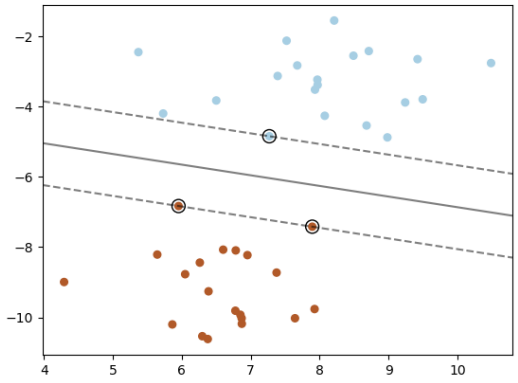

Figure 2: Máquina de vetor de suporte. Fonte: Scikit-Learn

A figura 2 apresenta o SVM conforme descrito: contendo duas classes separadas por um vetor. Apesar de a imagem ilustrar o SVM contendo apenas um hiperplano, sua representação pode abranger também um espaço multidimensional, o que justifica sua utilização em problemas de classificação, sendo o destaque o reconhecimento de dígitos manuscritos.

Vizinho mais próximo (KNN)

Em resumo, o método do vizinho mais próximo é um método que calcula a distância entre dois elementos com base em sua diferença euclidiana. Este algoritmo pode ser utilizado tanto para aprendizado supervisionado quanto não supervisionado.

Em aprendizado supervisionado, pode ainda ser usado tanto para classificação quanto para regressão, conforme visto em [2]. O que faz deste algoritmo bastante versátil.

Em se tratando de classificação, que é o objetivo deste trabalho, o KNN opera armazenando uma estrutura com o dados das instâncias utilizadas no treinamento. A classificação de uma nova instância é feita calculando-se seu ponto de inserção e atribuindo-a a classe de dados que possui o mairo

número de representantes nos vizinhos mais próximos ao ponto calculado.

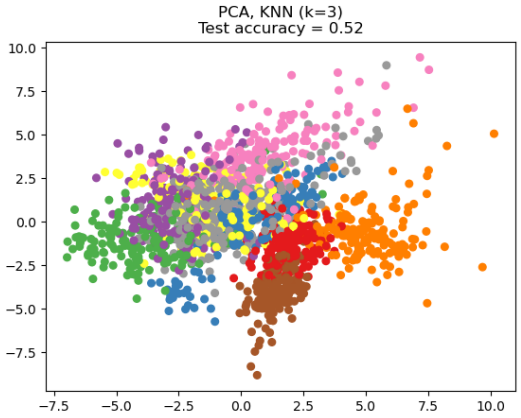

Figure 3: Vizinho mais próximo. Fonte: Scikit-Learn

2.4 Pré processamento

Dentre as etapas úteis de pré processamento dos dados úteis a este trabalho, destacam-se: discretização, tratamento de valores valores ausentes e codificação de categorias.

Discretização

A discretização consiste em transformar uma variável contínua em discreta. Isto é feito analisando-se todos os valores encontrados e dividindo-os em intervalos de partes iguais. Cada intervalos é nomeado e torna-se então uma nova variável.

Exemplo: uma variável que mede a idade em anos é do tipo contínua, ela pode ser convertida para uma variável discreta dividindo-se a idade em faixas etárias, como, criança, adolescente, adulto e idoso.

A discretização pode ser um recurso interessante em valores como idade, peso, altura, temperatura, entre outros. Porém isto irá depender do algoritmo que esta sendo utilizado e do contexto do problema.

Valores ausentes

Valores ausentes são um problema comum em AM. Diversas técnicas são utilizadas para tratar esta questão. Em variáveis contínuas, uma técnica bastante utilizada é o cálculo da média ou mediana de todos os valores e inserção deste valor nos campos ausentes.

Em bases de dados muito grandes, uma variável que possui poucas entidades com valor nulo pode ser desconsiderada por fins de praticidade. Em outros casos, no lugar dos valores ausentes pode ser simplesmente inserido um valor à critério do analista de dados.

Codificação

A codificação é a conversão de uma variável categórica, com dados em texto, para uma variável de valor numérico que possua a mesma representatividade.

As tabelas 1 e 2 representam respectivamente codificações binárias e decimais. A escolha da melhor opção depende do algoritmo de AM utilizado e da da quantidade de valores possíveis a variável pode assumir. Uma variável com muitos valores diferentes quando convertido para uma codificação binária pode aumentar significantemente o tamanho da base de dados, o que pode comprometer o desempenho.

ID	Estação	Estação_Num
1	Primavera	1
1	Outono	3
1	Primavera	1
1	Verão	2
1	Inverno	4

Table 1: Codificação decimal

ID	Estação	Est_Pri	Est_Ver	Est_Out	Est_Inv
1	Primavera	1	0	0	0
1	Outono	0	0	1	0
1	Primavera	1	0	0	0
1	Verão	0	1	0	0
1	Inverno	0	0	0	1

Table 2: Codificação binária

2.5 Calibragem parâmetros

Algoritmos de AM possuem diversos parâmetros a serem configurados. Árvores de decisão por exemplos possuem valores como: critério de qualidade (Gini, entropia), máximo de nós folhas, máxima de profundidade, entre outros.

O algoritmo do vizinho mais próximo deve considerar valores como: número de vizinhos, algoritmo de posicionamento (ball_tree, kd_tree, brute...), e pesos (uniforme, distancia). O vetor de suporte, por sua vez, possui valores como: parâmetro regularizador, kernel e gamma.

A solução encontrada para seleção da melhor configuração de parâmetros no problema proposta é um experimento fatorial completo, que testa todas as combinações possíveis de valores para cada algoritmo e retorna a configuração que apresenta melhor desempenho.

2.6 Métodos de treinamento

O treinamento do algoritmo pode ser feito utilizando três abordagens:

Treinar e testar com os mesmos dados

Esta abordagem é simples e de fácil implementação, porém gera uma falsa impressão sobre a qualidade dos resultados. Treinar e testar com os mesmos dados não previne casos de overfiting e o algoritmo quando sujeito a dados nunca antes vistos pode apresentar um péssimo desempenho.

Treinar e testar com dados diferentes

Um método mais sofisticado em relação ao anterior e dividir a base de dados em duas partes: treinamento e teste. Desta forma os dados testados serão sempre diferentes dos dados utilizados na aprendizagem fornecendo um resultado mais real sobre a qualidade da solução.

A porcentagem dos dados utilizada em cada etapa fica à critério do desenvolvedor. Um fator importante a ser considerado é ordenação dos dados. Os dados podem estar ordenados em relação a uma variável que será analisada no algoritmo, o que comprometeria os resultados. Embaralhar os dados neste caso é uma boa medida.

K-fold cross-validation

Por último e mais sofisticado o método cross-validation também divide a base de dados em treino e teste. Porém este processo é realizado N vezes e em cada etapa um conjunto de teste diferente é utilizado, conforme ilustrado na imagem a seguir.

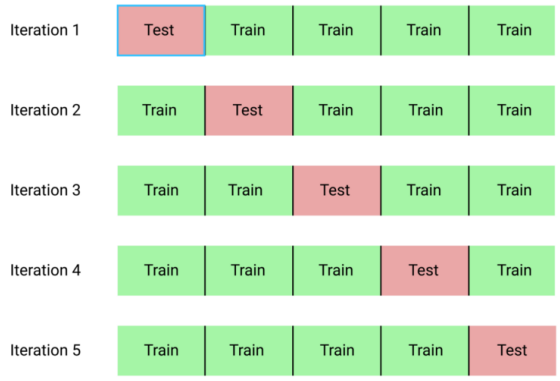

Figure 4: Treinamento com cross-validation

Um bom valor de N para o cross-validation fica entre 5 e 10 repartições. Através de um vetor com os resultados é possível obter a média e variância das execuções.

2.7 Análise

A análise dos resultados em um problema de classificação é feita medindo a acurácia de uma execução de teste e calculando a média e variância de uma sequência de execuções.

3 Desenvolvimento

Este projeto foi desenvolvido utilizando a linguagem de programação Python 3.9. Esta linguagem dispõe da biblioteca Scikit-Lean [2], que prove a implementação de todos algoritmos utilizados, além das funcionalidades para teste utilizando o método cross-validation. Demais bibliotecas como numpy e pandas também são utilizadas e fornecem recursos necessários para leitura e pré processamento dos dados.

3.1 Fluxograma

O fluxograma da figura 7 apresenta o processo de aprendizado de máquina desenvolvido neste projeto. Cada etapa é descrita nas seções seguintes.

3.2 Pré-processamento

Seleção dos atributos

Como descrito na seção 1.2 nossa base de dados consiste em características dos sobreviventes do titanic, e nosso objetivo

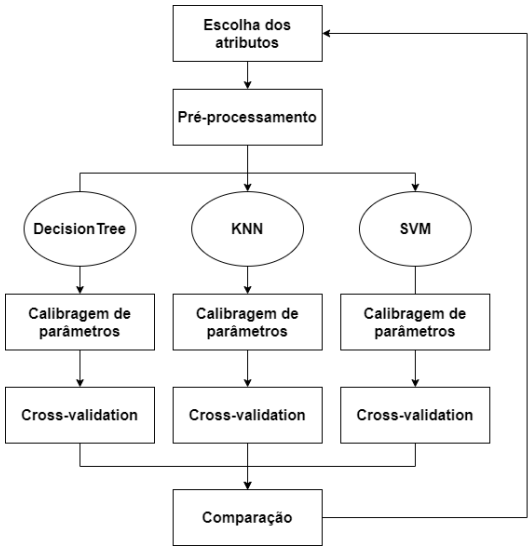

Figure 5: Fluxograma de execução

é classificar se uma pessoa é sobrevivente ou não, com base em suas características.

A tebela 3 mostra a lista de atributos da base de dados e suas descrições. Inicialmente os primeiros atributos eliminados para execução do algoritmo são: Name e PassengerId, visto que são atributos únicos a cada passageiro e não fornecem dados relevantes para o AM.

Atributo	Descrição
PassengerId	ID do passageiro
Survived	Resultado, se sobrevive ou não
Pclass	Classe da acomodação
Name	Nome do passageiro
Sex	Sexo
Age	Idade
Ticket	Número do ingresso
Fare	Tarifa
Cabin	Identificação da cabine
Embarked	Número do portão de embarque

Table 3: Lista de atributos

Tratamento de valores ausentes

A tabela 4 mostra quantas entidades possuem valores ausentes para cada atributo. O atributo Cabin possui 687 valores ausentes, isto mostra que mais da metade dos objetos não possuem este valor, sendo a melhor opção descartar este atributo da execução do AM.

O atributo Embarked possui apenas dois objetos com valores faltantes. Como a base possui 890 objetos, a opção mais prática é a remoção destes dois objetos identificados. Isto não interfere na qualidade dos resultados.

O último atributo com valores ausentes é o atributo Age (idade). Neste caso a solução adotada foi o cálculo da média de todos os valores disponíveis e a inserção deste valor nos campos ausentes.

Atributo	Ausentes
PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	177
Ticket	0
Fare	0
Cabin	687
Embarked	2

Table 4: Valores ausentes

Discretização de valores

Os atributos idade e tarifas foram discretizados para obtenção de melhores resultados. As idades foram divididas em quatro categorias: Children, Teenage, Adult e Elder. Já as tarifas foram divididas em 5 intervalos diferentes, de forma que cada um tivesse a mesma quantidade de itens.

A tabela 5 mostra a conversão dos valores de idade para cinco instâncias dos dados.

Id	Age	AgeClass
1	24	Adult
2	53	Elder
3	18	Teenage
3	32	Adult
3	7	Children

Table 5: Discretização da idade

Codificação

Os atributos: Sex, Embarked, Age e Fare foram codificados para valores decimais. Os atributos Age e Fare são portanto submetidos a discretização e codificação antes que possam de fato serem utilizados.

3.3 Calibragem dos parâmetros

```
Experimento fatorial:

0.803 (+/-0.043) for {'C': 1, 'decision_function_shape': 'ovo', 'gamma': 'scale'}
0.801 (+/-0.046) for {'C': 1, 'decision_function_shape': 'ovo', 'gamma': 'auto'}
0.803 (+/-0.043) for {'C': 1, 'decision_function_shape': 'ovr', 'gamma': 'scale'}
0.801 (+/-0.046) for {'C': 1, 'decision_function_shape': 'ovr', 'gamma': 'auto'}
0.801 (+/-0.048) for {'C': 3, 'decision_function_shape': 'ovo', 'gamma': 'scale'}
0.794 (+/-0.055) for {'C': 3, 'decision_function_shape': 'ovo', 'gamma': 'auto'}
0.801 (+/-0.048) for {'C': 3, 'decision_function_shape': 'ovr', 'gamma': 'scale'}
0.794 (+/-0.055) for {'C': 3, 'decision_function_shape': 'ovr', 'gamma': 'auto'}
0.795 (+/-0.051) for {'C': 5, 'decision_function_shape': 'ovo', 'gamma': 'scale'}
0.810 (+/-0.069) for {'C': 5, 'decision_function_shape': 'ovo', 'gamma': 'auto'}
0.795 (+/-0.051) for {'C': 5, 'decision_function_shape': 'ovr', 'gamma': 'scale'}
0.810 (+/-0.069) for {'C': 5, 'decision_function_shape': 'ovr', 'gamma': 'auto'}

Melhor calibragem: {'C': 5, 'decision_function_shape': 'ovo', 'gamma': 'auto'}
Melhor accuracia: 0.80988253319714

Pontuação final: 0.80988253319714
```

Figure 6: Experimento fatorial completo

A figura 7 mostra o resultado de um experimento fatorial completo aplicado ao algoritmo SVM para os parâmetros:

- C: [1, 3, 5];
- Shape: [’ovo’, ’ovr’];
- Gamma: [’scale’, ’auto’].

Este procedimento é aplicados aos três algoritmos estudados afim de compara-los em suas melhores performances. Os melhores resultados para cada algoritmo são obtidos com as seguintes configurações:

SVM

- C: 5;
- decision function shape: ovo;
- gamma: auto;

KNN

- algorithm: auto;
- n neighbors: 15;
- weights: distance;

Tree

- criterion: gini;
- min samples leaf: 1;
- min samples split: 6;
- random state: 4;
- splitter: random

4 Resultados

Selecionados os atributos, pré-processados e calibrados os parâmetros, agora é possível realizar a comparação e escolha do melhor método para o problema proposto. Ambos algoritmos são avaliados com o método cross-validation particionado em dez partes iguais.

O resultado de cada avaliação é um vetor de dez elementos, sendo estes elementos a acurácia de cada uma das dez execuções para cada algoritmo. A acurácia nada mais é que a porcentagem de testes bem sucedidos durante a execução. A imagem mostra um gráfico boxplot onde é possível analisar e comparar os algoritmos em detalhes.

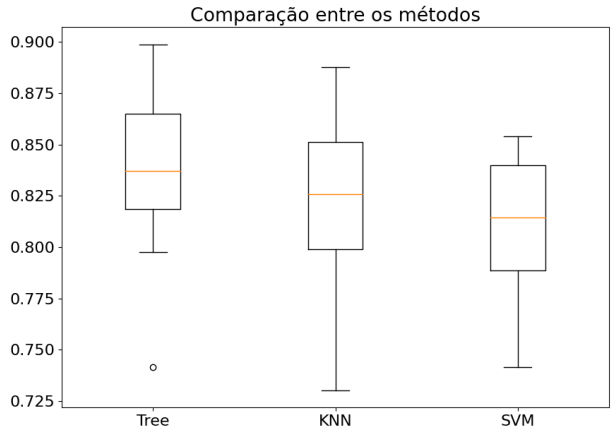

Figure 7: Comparação entre os métodos

Da imagem é possível observar que o KNN possui o pior resultado. Apesar de seu melhor resultado ser superior ao SVM, sua variância é a maior dentre os três. O algoritmo árvore de decisão possui o melhor resultado, em comparação aos piores resultados o a árvore de decisão apresenta o melhor desempenho em relação aos três. Além disso a árvore de decisão apresenta a menor variância e a maior média. Fica fácil concluir que a utilização de árvores de decisão é melhor opção para o problema proposto.

E finalmente, a média de acurácia para cada um dos algoritmos analisados é mostrada na tabela 6

Algoritmo	**Média**	**Variância**	**Desvio padrão**
DescisionTree	84%	0.0023	0.0478
SVM	80%	0.0013	0.0361
KNN	82%	0.0020	0.0450

Table 6: Resultados finais

5 Conclusão

Este trabalho apresenta um método para seleção de modelos de classificação para problemas de aprendizado de máquina. Analisando as etapas abordadas fica evidente que além da implementação dos algoritmos, a seleção de atributos e pré processamento se mostram igualmente relevantes.

Em problemas de classificação a discretização de variáveis continuas levou a melhoras nos resultados, assim como a aplicação da média geral aos valores ausentes.

O trabalho proporcionou um panorama sobre escolha e aplicação de algoritmos de classificação em aprendizado de máquina supervisionado. O leitor bem como o autor possuem agora um passo a passo para obtenção de bons resultados e tomadas de decisões na área de estudo aborada.

References

[1] Tom M Mitchell et al. Machine learning. 1997.

[2] 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.