Pós-Graduação em Ciência da Computação

“Seleção Ativa de Exemplos de Treinamento para Meta-Aprendizado”

Por

Arthur Fernandes Minduca de Sousa

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, SETEMBRO/2013

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARTHUR FERNANDES MINDUCA DE SOUSA

“Seleção Ativa de Exemplos de Treinamento para Meta-Aprendizado"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: Prof. Dr. Ricardo Bastos Cavalcante Prudêncio

RECIFE, SETEMBRO/2013

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

Sousa, Arthur Fernandes Minduca de
Seleção ativa de exemplos de treinamento para meta-aprendizado / Arthur Fernandes Minduca de Sousa. - Recife: O Autor, 2013.
72 f.: il., fig., tab.

Orientador: Ricardo Bastos Cavalcante Prudêncio.
Dissertação (mestrado) - Universidade Federal de Pernambuco. CIn, Ciência da Computação, 2013.

Inclui referências.

1. Inteligência artificial. 2. Aprendizado de máquina. 3. Meta - aprendizado. I. Prudêncio, Ricardo Bastos Cavalcante (orientador). II. Título.

006.3 CDD (23. ed.) MEI2013 – 114

Dissertação de Mestrado apresentada por **Arthur Fernandes Minduca de Sousa** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Seleção Ativa de Exemplos de Treinamento Para Meta-Aprendizado**" orientada pelo **Prof. Ricardo Bastos Cavalcante Prudêncio** e aprovada pela Banca Examinadora formada pelos professores:

_____

Profa. Renata Maria Cardoso Rodrigues de Souza
Centro de Informática / UFPE

_____

Profa. Anne Magaly de Paula Canuno
Departamento de Informática e Matemática Aplicada / UFRN

_____

Prof. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 29 de julho de 2013

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Dedico esta dissertação aos meus pais,
José Tadeu Fernandes de Sousa e Severina Maria Minduca.

AGRADECIMENTOS

Agradeço aos meus pais pela participação ativa na construção dos valores que alicerçam meu caráter e, de uma maneira mais abrangente, minha formação como indivíduo. Agradeço à minha namorada por seu apoio constante e por me manter resiliente nos momentos mais desafiadores da pesquisa acadêmica. Em especial, agradeço ao meu orientador, Ricardo Prudêncio, pela paciência virtuosa ao guiar orientandos incipientes, porém ávidos por conhecimento, nesse primeiro nível de aprofundamento *stricto sensu*. Agradeço ao meu orientador também por se dispor como fonte bem-servida de sabedoria e conhecimento, sendo capaz de repassar sua experiência com excelente didática.

RESUMO

Várias abordagens têm sido aplicadas à tarefa de seleção de algoritmos. Nesse contexto, Meta-Aprendizado surge como uma abordagem eficiente para predizer o desempenho de algoritmos adotando uma estratégia supervisionada. Os exemplos de treinamento de Meta-Aprendizado (ou meta-exemplos) são construídos a partir de um repositório de instâncias de problemas (como, por exemplo, um repositório de bases de dados de classificação). Cada meta-exemplo armazena características descritivas de uma instância de problema e um rótulo indicando o melhor algoritmo para o problema (empiricamente identificado entre um conjunto de algoritmos candidatos). Os melhores algoritmos para novos problemas podem ser preditos se baseando apenas em suas características descritivas, sem a necessidade de qualquer avaliação empírica adicional dos algoritmos candidatos. Apesar dos resultados Meta-Aprendizado requererem a implementação de um número suficiente de instâncias de problemas para produzir um conjunto rico de meta-exemplos. Abordagens recentes para gerar conjuntos de dados sintéticos ou manipulado foram adotados com sucesso no contexto de Meta-Aprendizado. Essas propostas incluem a abordagem de Datasetoids, que é uma técnica simples de manipulação de dados que permite a geração de novos conjuntos de dados a partir de bases existentes. Apesar dessas propostas produzirem dados relevantes para Meta-Aprendizado, eles podem eventualmente produzir instâncias de problemas redundantes ou até mesmo irrelevantes. Meta-Aprendizado Ativo surge nesse contexto para selecionar somente as instâncias mais informativas para a geração de meta-exemplos. Neste trabalho, investigamos o uso de Meta-Aprendizado Ativo combinado com Datasetoids, focando no uso do algoritmo Random forest em Meta-Aprendizado. Para selecionar as instâncias de problemas, implementamos um critério de incerteza baseado em entropia, específico para o Random forest. Também investigamos o uso de uma técnica de detecção de *outliers* a fim de remover a priori os problemas considerados *outliers*, objetivando melhorar o desempenho dos métodos de Aprendizagem Ativa. Nossos experimentos revelaram uma melhora no desempenho do Meta-Aprendizado e uma redução no custo computacional para a geração de meta-exemplos.

Palavras-chave: Meta-Aprendizado, Seleção de Algoritmos, Aprendizagem Ativa, *Uncertainty Sampling*, Detecção de *Outliers*.

ABSTRACT

Several approaches have been applied to the task of algorithm selection. In this context, Meta-Learning arises as an efficient approach to predict algorithm performance by adopting a supervised strategy. The training examples for Meta-Learning (the meta-examples) are built from a pool of problem instances (e.g., a pool of classification datasets). Each meta-example stores the descriptive features of a problem instance and a label indicating the best algorithm for the problem (empirically identified among a set of candidates). The best algorithms for new problems can be predicted based on their descriptive features, without the need of empirical evaluation of the candidate algorithms. Despite its promising results, Meta-Learning requires an adequate number of instances to produce a rich set of meta-examples. Recent approaches to generate synthetic or manipulated datasets have been adopted with success in the context of Meta-Learning. These proposals include the datasetoids approach, a simple data manipulation technique that generates new datasets from existing ones. Although such proposals can actually produce relevant datasets, they can eventually produce redundant, or even irrelevant, problem instances. Active Meta-Learning arises in this context to select only the most informative instances for meta-example generation. In this work, we investigate the use of Active Meta-Learning combined to datasetoids, focusing on using the Random forest algorithm to meta-learning. In order to select problem instances, we implemented an uncertainty sampling method based on entropy, specific for Random forest. We also investigated the use of Outlier Detection to remove a priori those problems considered as outliers, aiming to improve the performance of the Active Learning methods. Our experiments revealed a significant improvement in Meta-Learning performance, at same time reducing the computational cost of generating meta-examples.

Keywords: *Meta-Learning, Algorithm Selection, Active-Learning, Uncertainty Sampling, Outlier Detection*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Em diferentes áreas de conhecimento observa-se uma grande variedade de algoritmos que competem entre si com o objetivo de resolver problemas específicos com as mais diversas finalidades. Em aprendizagem de máquina, algoritmos de redes neurais, máquinas de vetores de suporte, árvores de decisão, entre muitas outras opções, são algoritmos candidatos à resolução de problemas de regressão e classificação. Em alguns contextos, algoritmos também podem apresentar variações decorrentes de parametrizações ou até mesmo detalhes de implementação que podem vir a afetar significativamente seu comportamento.

Dadas as várias possibilidades, a seleção de um bom algoritmo para resolver um determinado problema é uma questão decisiva para o sucesso de uma aplicação. A análise de desempenho de algoritmos e a decisão de qual algoritmo apresenta o melhor resultado em um determinado problema não são tarefas simples de serem resolvidas. O principal motivo é que não existe um único algoritmo que seja considerado o mais eficiente em todas as tarefas possíveis. Além disso, apesar da variedade de algoritmos, não existe um padrão consistente que informe qual método é o mais adequado para um determinado problema. A escolha do melhor algoritmo para um determinado problema é efetuada levando em consideração características específicas da tarefa a ser realizada [5] como, por exemplo, interpretabilidade dos modelos produzidos ou disponibilidade de ferramentas. Sendo assim, dado uma instância de problema (por exemplo, uma tarefa de classificação) e um conjunto de algoritmos candidatos a resolvê-lo, o objetivo principal é selecionar o algoritmo que tem a maior probabilidade de obter o melhor desempenho em sua execução no problema proposto. Abordagens tradicionais geralmente envolvem avaliações empíricas custosas, como tentativa e erro, ou conhecimento de especialista, o que nem sempre está disponível [6].

A necessidade de resolver a tarefa de seleção de algoritmos de maneira eficiente levou a comunidade acadêmica a pesquisar uma maneira de aprender sobre o próprio processo de aprendizado e extrair informações de desempenho de execuções de algoritmos em problemas de aprendizagem. O acesso a esse contexto permite que, dado um novo problema de aprendizado e uma lista de algoritmos candidatos a resolvê-lo, possamos analisar um histórico de execuções desses algoritmos candidatos

em tarefas semelhantes e inferir automaticamente o melhor algoritmo para a tarefa em questão.

Meta-Aprendizado surge nesse contexto como uma proposta eficiente para a tarefa da seleção de algoritmos, provendo abordagens automatizadas para predição de desempenho de algoritmos através de uma estratégia de aprendizado supervisionado [4], [20]. Em Meta-Aprendizado, um algoritmo de aprendizagem supervisionada é utilizado em nível meta para realizar um mapeamento entre características de problemas (entropia, número de exemplos de treinamento, número de atributos, etc.) e os algoritmos que apresentam o melhor desempenho em problemas de características semelhantes. Isso é realizado através da aquisição de conhecimento a partir de um conjunto de meta-exemplos gerado de um repositório de instâncias de problemas. Cada meta-exemplo armazena características de um problema e um atributo alvo indicando o melhor algoritmo (definido empiricamente).

Um problema conhecido em Meta-Aprendizado está relacionado à baixa disponibilidade de instâncias de problemas reais ou conjuntos de dados para produzir um rico conjunto de meta-exemplos [5]. Isso é verdade no contexto de seleção de algoritmos para problemas de classificação e regressão, assim como em outros domínios onde Meta-Aprendizado tem sido aplicado mais recentemente [20]. Esse assunto tem atraído a atenção da comunidade acadêmica nos últimos anos, através da geração de novas instâncias de bases de dados sintéticas ou manipuladas [13], [22], [21].

Além do problema da indisponibilidade de dados para a construção dos meta-exemplos, a geração de meta-dados para seleção de algoritmos é geralmente um procedimento de alto custo computacional. Para poder gerar os meta-exemplos a partir de uma instância de problema, é preciso antes realizar uma avaliação empírica de cada algoritmo candidato sobre o conjunto de dados referente ao problema. Consequentemente, observa-se um elevado custo computacional quando existe uma grande quantidade de problemas disponível. Dessa forma, produzir e selecionar somente os meta-exemplos mais informativos e não redundantes é um problema de importância significativa em Meta-Aprendizado. Essa questão tem sido resolvida com o uso de técnicas de Aprendizagem Ativa para dar suporte à geração de meta-exemplos, em uma abordagem denominada Meta-Aprendizado Ativo [14]. Aprendizagem Ativa permite que o algoritmo de aprendizagem possa selecionar os exemplos mais

relevantes para incorporar à base de treinamento e desconsiderar os exemplos menos informativos.

Em [15], Meta-Aprendizado Ativo foi combinado com uma técnica de manipulação de dados denominada *datasetoids* [21] com o objetivo aumentar a quantidade de conjuntos de dados e reduzir o custo computacional para coleta de meta-dados. Os experimentos apresentaram taxas de acerto semelhantes ao que foi observado em [21], mas usando menos de 1/3 da quantidade de meta-exemplos disponíveis. Os resultados obtidos mostram que é possível aumentar o tamanho da base e manter (ou até mesmo melhorar) a taxa de acerto do algoritmo de aprendizagem usando uma quantidade menor de exemplos de treinamento. Apesar do sucesso da aplicação, alguns pontos motivam uma investigação mais aprofundada em Meta-Aprendizado. Os experimentos focaram no k-NN, algoritmo que apresentou certa instabilidade no seu processo de seleção de exemplos. Além disso, a seleção de meta-exemplos foi realizada utilizando um método de Aprendizagem Ativa baseado em incerteza. Entretanto, essa abordagem apresenta uma sensibilidade a *outliers*. Apesar da técnica de *datasetoids* ter uma tendência a aumentar a quantidade de *outliers* na base, não foi realizado um procedimento para amenizar os efeitos desses *outliers*, a fim de realizar uma seleção mais consistente de exemplos de treinamento.

1.1 TRABALHO REALIZADO

Em [15], o algoritmo utilizado como meta-aprendiz foi o k-NN. Este algoritmo, apesar de ser aplicado extensivamente nos mais diversos problemas de aprendizagem devido a sua simplicidade e eficiência, obteve resultados inferiores aos do Random forest no trabalho original de *datasetoids* [21]. Nesse caso, Random forest apresentou uma taxa de acerto superior a de todos os demais algoritmos investigados.

Neste trabalho, realizamos uma investigação mais profunda da combinação de abordagens de manipulação para geração de novas bases de dados e Aprendizagem Ativa para dar suporte à geração de meta-exemplos [15], combinando Meta-Aprendizagem Ativa com uma técnica de manipulação de dados denominada *datasetoids* [21] e remoção de *outliers*, focando no algoritmo Random forest [8], [10] como meta-aprendiz.

Inicialmente, a técnica de *datasetoids* é adotada a fim de produzir um grande repositório de instâncias de problemas. Em seguida, uma técnica de Aprendizagem Ativa é utilizada para selecionar deste repositório apenas as instâncias mais relevantes. Também investigamos o efeito de *outliers* no desempenho do método *uncertainty sampling* [28], um problema conhecido em Aprendizagem Ativa [27]. Uma técnica de detecção de *outliers* foi usada a fim de melhorar o desempenho do *uncertainty sampling* através da remoção de dados indesejados da lista de problemas disponíveis para geração de meta-exemplos. O objetivo principal desta combinação é reduzir o custo computacional relacionado à coleta de meta-dados realizando uma seleção ativa dos problemas mais informativos.

Os experimentos apresentados nesta dissertação foram realizados usando os algoritmos k-NN e Random forest como meta-aprendizes. A escolha do Random Forest foi motivada por seu notável desempenho obtido em outros trabalhos na literatura [9],[21]. Também foi utilizada uma técnica de detecção de *outliers* [26] para eliminar ruído. Nos experimentos de Meta-Aprendizado realizados, a acurácia do Random Forest foi avaliada em dois cenários: (1) adotando um critério de incerteza baseado em entropia para selecionar as instâncias de problemas para a geração de meta-exemplos e (2) adotando uma estratégia de seleção aleatória para servir como *baseline*. No método baseado em entropia, o meta-aprendiz Random Forest obteve o nível desejável de acurácia utilizando apenas 25% do repositório total de instâncias de problemas. Os resultados apresentam um ganho de desempenho em relação aos resultados obtidos em [15], onde foi utilizado um meta-aprendiz baseado em instância. Os experimentos também revelaram que o desempenho de Meta-Aprendizado Ativo pode ser melhorado com a remoção de problemas considerados *outliers*. Estudos anteriores em [28] também apresentaram bons resultados na aplicação de detecção de *outliers* para remover ruído e dados irrelevantes.

1.2 OBJETIVOS

A partir do que foi exposto, o principal objetivo deste trabalho é melhorar o desempenho dos modelos de Meta-Aprendizado e reduzir o custo computacional relacionado à aquisição de meta-exemplos.

Como objetivos específicos, podemos citar:

- Reproduzir os experimentos de [15] usando uma versão mais estável do algoritmo k-NN.
- Usar um algoritmo Random forest como meta-aprendiz aplicando um critério de incerteza baseado em entropia para realizar uma seleção ativa de meta-exemplos.
- Aplicar uma técnica de manipulação de dados para aumentar a disponibilidade de meta-dados.
- Usar uma técnica de detecção de *outliers* para eliminar ruído e aumentar a qualidade do conjunto de dados utilizado pelo modelo.
- Obter resultados positivos em cenários não bem sucedidos nos trabalhos anteriores.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

Além deste capítulo, esta dissertação será apresentada em mais cinco capítulos que estão organizados da seguinte forma:

Capítulo 02: Este capítulo apresenta Meta-Aprendizado no contexto de seleção de algoritmos, sua relevância e abordagens mais comuns para resolução dessa tarefa. Em seguida, são discutidas abordagens de geração de dados.

Capítulo 03: Neste capítulo é realizada uma revisão de Aprendizagem Ativa, focando em sua motivação, cenários de execução e estratégias de *queries* mais usadas na literatura.

Capítulo 04: Neste capítulo é apresentado o trabalho desenvolvido: uma investigação mais profunda no uso de Meta-Aprendizado Ativo combinado com *Datasetoids*, focando no uso do algoritmo Random forest como meta-aprendiz. Além disso, é detalhada uma técnica de detecção de *outliers* que é combinada ao modelo de Meta-Aprendizado.

Capítulo 05: Neste capítulo é detalhado todo o arranjo experimental utilizado, desde a construção dos meta-exemplos até a configuração do modelo de Meta-Aprendizado Ativo. Ainda neste capítulo são analisados os resultados obtidos a partir dos cenários implementados.

Capítulo 06: Aqui são detalhadas as conclusões do trabalho desenvolvido e é realizado um levantamento de propostas de trabalhos futuros.

2 META-APRENDIZADO APLICADO À SELEÇÃO DE ALGORITMOS

O problema da seleção de algoritmos abriu possibilidades para a investigação da possibilidade de elevar o nível de abstração do processo de aprendizado e fazer com que o modelo aprenda sobre o próprio processo de aprendizagem em si, tornando possível o acúmulo de conhecimento decorrente de processos de aprendizado anteriores. Com isso, o modelo de aprendizagem pode usar suas experiências anteriores de aprendizado para selecionar algoritmos de forma automática.

Neste capítulo, apresentamos o problema da seleção de algoritmo. Em seguida, fazemos uma revisão de Meta-Aprendizado e como a técnica pode ser utilizada para resolver a tarefa em questão.

2.1 O PROBLEMA DA SELEÇÃO DE ALGORITMOS

Existe uma grande variedade de algoritmos que resolvem um mesmo problema. Na área de aprendizagem de máquina, por exemplo, algoritmos de classificação, como redes neurais artificiais, árvores de decisão e máquinas de vetores de suporte, podem ser aplicados com o objetivo de auxiliar a tomada de decisão de, por exemplo, diagnóstico médico ou análise de crédito. Em alguns cenários, observa-se também que variações de um determinado algoritmo podem ser geradas facilmente, devido a detalhes de implementação ou configuração de parâmetros. Figura 2.1 apresenta a definição formal do problema da seleção de algoritmos criada por Rice em [33]. Em seu modelo, observa-se o espaço de problemas P, que contém todo o conjunto de instâncias de problemas. O espaço de características F que contém características adquiridas de P e realizadas por um extrator de características qualquer. O espaço de algoritmos A, que contém todos os algoritmos capazes de resolver um determinado problema. O espaço de medidas de desempenho D, que contém todas as medidas de desempenho possíveis. Dado um problema $x \in P$ que contém características $f(x) \in F$, a tarefa consiste em encontrar um mapeamento $S(f(x))$ no espaço de algoritmos A de forma que um algoritmo candidato selecionado $\propto \in A$ apresente o maior desempenho $d(\propto (x)) \in D$.

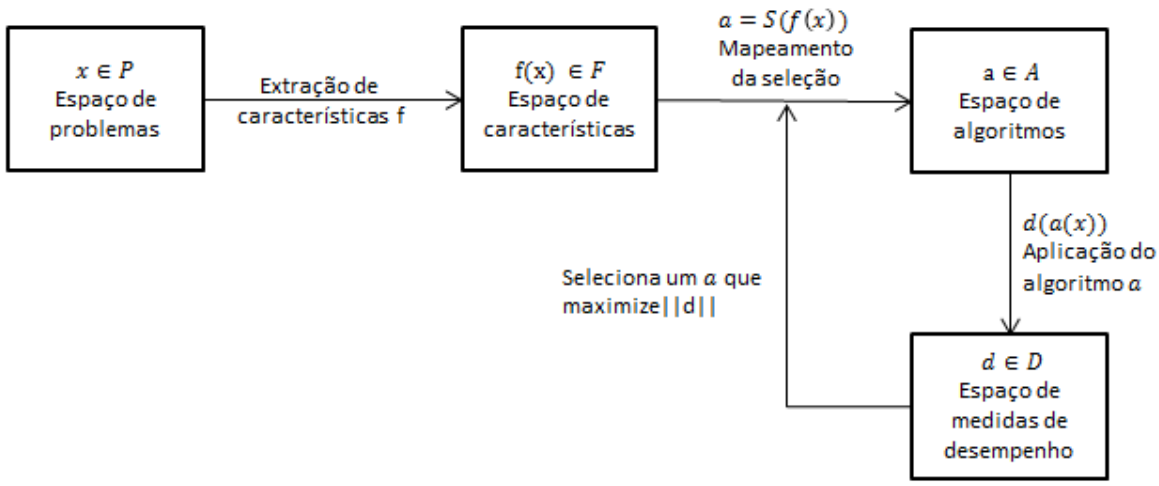

Figura 2.1: Modelo do problema da seleção de algoritmos (adaptado de [33])

A análise do que é um bom algoritmo, assim como a decisão de que um determinado algoritmo é o melhor para um dado problema não são tarefas de natureza trivial. De uma forma geral, a definição de "melhor" no contexto de algoritmos é encontrada a partir da relação entre alguns atributos chave. No contexto de meta-heurísticas, por exemplo, podemos destacar três aspectos de interesse [29]: custo computacional, qualidade da solução e robustez.

O custo computacional está relacionado ao tempo de execução e consumo de memória. Como os limites atuais de memória tendem a ser mais que suficientes para um grande número de implementações, requisitos de memória geralmente são excluídos da análise de desempenho e o custo computacional muitas vezes é focado nos requisitos de tempo de execução das operações.

Tempo, contudo, não é o único fator relevante. A qualidade da solução obtida em tempo hábil muitas vezes é a métrica mais importante. Uma vez que uma melhor solução geralmente está relacionada a um maior tempo de execução, em meta-heurísticas, assim como em boa parte dos problemas onde se aplica técnicas de Aprendizagem Ativa, podemos cair no caso especial de que, com tempo suficiente, o algoritmo é capaz de cobrir todo o espaço de problemas e definitivamente encontrar a melhor solução para o problema. O mesmo ocorre com o problema da definição do critério de parada de alguns algoritmos, uma vez que muitas vezes é associado um número máximo de iterações ou até mesmo um tempo de execução limite. De uma forma geral, o cenário de explorar todo o espaço de soluções geralmente é indesejado (em alguns casos, impraticável de ser computado), pois demanda uso máximo de custo

computacional. Dessa forma, a forma como o custo computacional e a qualidade da solução estão relacionados é uma questão de natureza relevante.

Além dos critérios mencionados, a robustez define o quanto um algoritmo escala para variações de um determinado problema. No contexto de Meta-Aprendizado, por exemplo, um algoritmo é executado em uma série de problemas distintos usando, em sua configuração mais comum, uma parametrização padrão. A robustez do algoritmo pode estar relacionada não somente à natureza de sua implementação, mas também a um *tunning* de seus parâmetros e usabilidade em uma gama de instâncias de problemas.

2.1.1 O TEOREMA "*NO FREE LUNCH*"

O teorema *No Free Lunch* (NFL) de Wolpert e Macready [32] afirma que dois algoritmos são equivalentes quando seu desempenho é medido entre todos os problemas possíveis. O teorema faz pesquisadores se perguntarem se é mesmo possível escolher um bom algoritmo para uma tarefa particular, já que, segundo a afirmação acima, de uma maneira geral todos os métodos apresentam desempenho médio semelhante. Segundo o teorema, o ganho do algoritmo em um determinado problema é eventualmente compensado em outro. Partindo do princípio de que não existe um único algoritmo que possa ser considerado o melhor em todos os domínios de problemas, o NFL leva a entender que a tarefa de encontrar o melhor algoritmo para resolver uma determinada tarefa requer um entendimento das características do problema.

2.2 META-APRENDIZADO

De uma maneira geral, os desenvolvimentos realizados na área de aprendizagem de máquina e mineração de dados não deixam claros quais métodos são ideais para determinados tipos de problema. Dessa forma, como os problemas reais de aprendizagem de máquina trazem consigo pouca ou nenhuma informação relevante sobre o método ideal a ser aplicado em seu contexto, pesquisadores utilizam sua experiência – e em algumas situações utilizam procedimentos de tentativa e erro – a fim de selecionar alguns poucos algoritmos e identificar qual deles se adequa melhor às características dos dados.

O problema de seleção de modelos abriu possibilidades para investigar se o processo de aprendizado pode ocorrer em um nível de abstração em que é possível aprender sobre o próprio processo de aprendizado e gerar conhecimento a partir da experiência de execuções anteriores. Dessa forma, torna-se possível a realização de uma seleção automática de modelos tomando como base o conhecimento adquirido e a características de um novo problema a ser resolvido.

Esta seção tem como objetivo realizar uma revisão em uma subárea de aprendizagem de máquina e mineração de dados conhecida como *Meta-Aprendizado*. Uma motivação desta abordagem é permitir que usuários que não possuem o nível de conhecimento necessário para selecionar o modelo que melhor se adapta às características do problema possam utilizar sistemas de Meta-Aprendizado para auxiliar nessa tomada de decisão de forma sistemática e automática. Além disso, a técnica também permite que possamos tirar proveito do uso repetitivo de um método em tarefas de características semelhantes para realizar uma adaptação contínua. Esse conceito faz parte de uma área de pesquisa conhecida como *learning to learn* e possibilita uma reaprendizagem por parte do mecanismo de aprendizado [34], [35].

Meta-Aprendizado é o processo de aquisição automática de conhecimento (ou meta-conhecimento) a partir de uma avaliação empírica de algoritmos de treinamento em diferentes problemas de aprendizagem (conjuntos de dados). Um sistema de Meta-Aprendizado resolve o problema de seleção de modelos efetivamente através da realização de um mapeamento entre um determinado problema e um modelo preditivo (ou uma combinação de modelos). A função preditiva é modelada na forma de um problema de aprendizagem supervisionada. O processo difere da abordagem tradicional de aprendizagem de máquina principalmente no nível de abstração em que a adaptação do modelo de aprendizagem ocorre. A aprendizagem dos métodos tradicionais foca na adaptação do modelo em uma tarefa específica. Meta-Aprendizado, por outro lado, utiliza modelos de aprendizado induzidos a partir de conhecimento em nível meta, tendo como objetivo o acúmulo de experiência resultante de várias aplicações de um sistema de aprendizado [25].

Um sistema de Meta-Aprendizado armazena conhecimento sobre o processo de aprendizado, conhecido como *meta-conhecimento*. A geração de meta-conhecimento pode variar de sistema em sistema, mas ele é composto por basicamente qualquer informação extraída a partir da execução de um modelo preditivo em uma tarefa de aprendizado que pode ser utilizado para a criação de um modelo que

relacione características de problemas a uma classe de saída. Sendo assim, a qualidade de um modelo de Meta-Aprendizado está intimamente ligada à qualidade do meta-conhecimento obtido durante a etapa de aprendizagem. Em um contexto de recomendação de algoritmos, por exemplo, o meta-conhecimento é utilizado para construir a função de mapeamento entre características de problemas e desempenho de algoritmos [36].

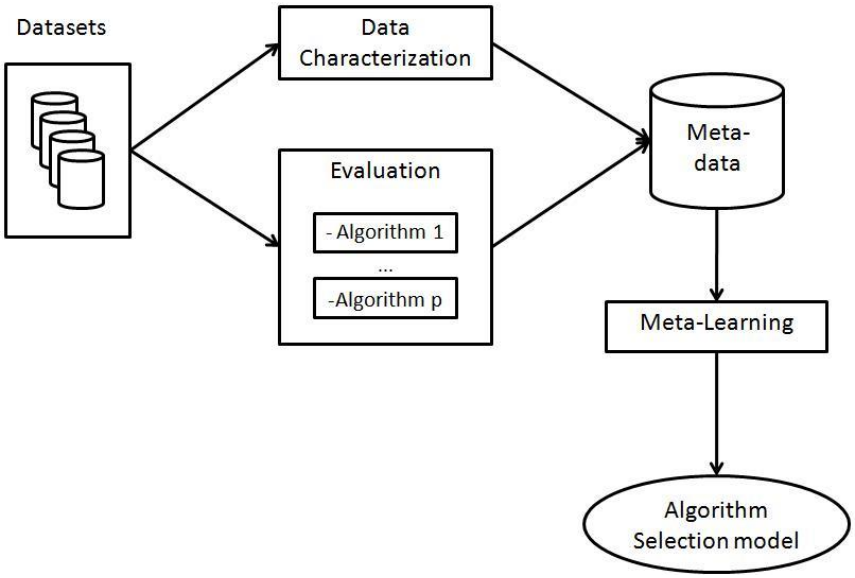

Figura 2.2: Processo de Meta-Aprendizado para seleção de algoritmo (adaptado de [4]).

A abordagem de Meta-Aprendizado aplicada à seleção de algoritmo é sumarizada na Figura 2.2. Os meta-dados armazenam descrições de um conjunto de bases de dados, contendo: (1) estimativas de desempenho de um conjunto de algoritmos candidatos nos respectivos conjuntos de dados; (2) atributos preditivos denominados meta-atributos, cada um correspondendo a uma característica do problema (como, por exemplo, número de exemplos de treinamento no conjunto de dados, entropia do atributo correspondente à sua classe, correção entre os atributos numéricos). Os exemplos em Meta-Aprendizado (ou meta-exemplos) compõem o conjunto de meta-dados e são descritos por meta-atributos e pelo meta-atributo alvo. Um algoritmo de aprendizagem de máquina (o então chamado meta-aprendiz) é aplicado a essa base de dados com o objetivo de induzir um modelo que relaciona valores de meta-atributos com o desempenho de um algoritmo candidato que empiricamente apresentou os resultados mais satisfatórios entre os demais algoritmos

aplicados. Após a etapa de treinamento, o modelo será capaz de identificar, para um novo problema apresentado, qual o algoritmo mais adequado para a tarefa em questão.

Meta-Aprendizado começou a ser investigado originalmente no contexto de aprendizagem supervisionada visando selecionar algoritmos para tarefas de classificação e regressão. Nos últimos anos, a técnica de Meta-Aprendizado tem sido aplicada à tarefa de recomendação de algoritmos em vários domínios de aplicação [20]. Para que o modelo gere saídas relevantes para o usuário, os algoritmos candidatos devem ser escolhidos cuidadosamente levando em conta a natureza das tarefas que serão executadas. Isso porque um algoritmo pode apresentar desempenho melhor que os demais candidatos somente em uma determinada classe de problemas. Esse comportamento inerente aos algoritmos é denominado superioridade seletiva [41]. Dessa forma, existe uma relação direta entre a característica dos dados com o *bias* indutivo de um algoritmo de aprendizagem de máquina.

Maiores informações sobre Meta-Aprendizado para recomendação de algoritmos podem ser lidas em [4], [20], [25] e referências citadas.

2.2.1 META-DADOS

Os meta-dados assumem um papel importante em Meta-Aprendizado, pois o desempenho de um modelo depende da qualidade dos dados de entrada na etapa de treinamento. Dentre os dois fatores que mais influenciam no desempenho dos modelos de Meta-Aprendizado, destacam-se a quantidade de meta-exemplos e a escolha dos meta-atributos.

A base de meta-exemplos contém os dados de entrada para treinamento do modelo. Esse conjunto de meta-exemplos deve ser suficientemente numeroso e diversificado para que o modelo possa ser considerado confiável. Uma base de dados composta por uma quantidade insatisfatória de meta-exemplos pode levar o modelo a falhar em generalizar as regras de indução. A fim de amenizar essa limitação e gerar uma base de meta-exemplos maior e mais confiável, técnicas de geração de meta-dados artificiais podem ser aplicadas [2], [11]. Essas abordagens tem recebido maior atenção ao longo dos anos devido ao sucesso obtido em sua aplicação em problemas de natureza semelhante. Dado a importância desse tema, o assunto é discutido com maior aprofundamento na seção 2.2.3.

A escolha dos meta-atributos ideais na etapa de geração dos meta-exemplos é uma tarefa de grande importância, pois o desempenho do modelo de Meta-Aprendizado está intimamente relacionado às informações descritas neles. De acordo com [4], as informações contidas nos meta-atributos devem descrever um problema de forma a garantir que dois problemas distintos com o mesmo conjunto de meta-atributos apresentem valores diferentes de atributo-alvo. Além disso, dependendo do cenário, alguns meta-atributos podem se mostrar menos relevantes que outros. Outra consideração importante acerca dos meta-atributos é que a complexidade computacional para extraí-los não pode ser alta. Caso contrário, esse alto custo poderia resultar na situação em que a realização do teste empírico de executar todos os algoritmos candidatos no problema proposto se mostra mais vantajoso.

A caracterização das bases de dados devem conter informações que viabilizem a determinação do desempenho relativo de algoritmos e apresentar baixo custo computacional e [37]. Três propostas de caracterização de meta-atributos são apresentadas em [4]: a de caracterização direta, a abordagem baseada em *landmarking* e a caracterização através de modelos.

A abordagem de caracterização direta é definida pela extração de características do problema a partir de medidas descritivas simples, estatísticas e baseadas em teoria da informação. Um dos primeiros esforços nessa linha foi realizado no projeto STATLOG [38]. A Tabela 2.1 apresenta todas as medidas utilizadas no projeto STATLOG para caracterização dos dados.

Medidas	Definição
Simples	Número de exemplos Número de atributos Número de classes Número de atributos binários
Estatísticas	Razão média do desvio dos atributos Correlação média absoluta entre atributos por classe Primeira correlação canônica Proporção de variância explicada pelo 1º discriminante canônico

	Assimetria média absoluta dos atributos Curtose média dos atributos
Informação	Entropia normalizada das classes Entropia média dos atributos Informação média dos atributos Informação mútua média entre classes e atributos Razão sinal/ruído

Tabela 2.1: Medidas usadas no STATLOG para caracterizar bases de dados de classificação.

A abordagem de *Landmarking* permite detectar um grau de proximidade entre conjuntos de dados. A proximidade das bases é calculada a partir da aplicação de *landmarkers*, algoritmos simples de classificação que constroem vizinhanças de áreas de competência. O desempenho dos *landmarkers* define o nível de proximidade entre os problemas. Dessa forma, a tendência é que bases de dados de natureza semelhante estejam próximas e dentro de uma mesma área de competência. Em [39] os autores selecionam sete algoritmos para realizar caracterização baseada em *landmarking* e 10 algoritmos complexos para servirem como rótulo de classe. Na abordagem proposta, um modelo de Meta-Aprendizagem utilizou os *landmarkers* como meta-atributo a fim de sugerirem o algoritmo final mais apropriado para o rótulo de classe.

A caracterização através de modelos [40] é baseada em informações do próprio classificador construído como, por exemplo, número de folhas em uma árvore de decisão. Dentre os algoritmos de classificação, as árvores de decisão têm sido bastante utilizadas para caracterização dos dados [40]. Isso porque elas apresentam comportamento determinístico, o que facilita seu entendimento, e podem ter vários meta-atributos relacionados (número de nós por atributo, profundidade máxima da árvore, etc.).

2.2.2 FORMAS DE RECOMENDAÇÃO

Em [4] são abordadas três propostas para determinar a saída do modelo de Meta-Aprendizado: encontrar o melhor algoritmo para o problema, um subconjunto dos algoritmos disponíveis ou realizar um ranking de algoritmos.

Na abordagem de encontrar o melhor algoritmo, a saída do modelo é o algoritmo cujo desempenho é o melhor dentre os candidatos disponíveis. Nesse tipo de abordagem, o problema pode ser pensado como uma simples tarefa de classificação. Uma desvantagem desta proposta reside na falta de tolerância a falhas do modelo, pois uma recomendação pode não ser a melhor dentre as opções disponíveis. No pior caso, uma recomendação pode estar longe de ser uma boa solução.

Outra proposta consiste em retornar um subconjunto de algoritmos que contém os métodos mais propensos a resolver o problema com o melhor desempenho. Nessa abordagem, o modelo pode definir um limiar mínimo de desempenho a ser atendido. Em [4], o autor aborda uma estratégia para cálculo do intervalo do limiar através da seguinte equação:

$$\left[e_{min}, e_{min} + k\sqrt{\frac{e_{min}(1 - e_{min})}{n}} \right) \tag{2.1}$$

Na fórmula acima, e_{min} representa o erro do melhor algoritmo, k é uma constante definida em nível de usuário para influenciar no limiar de desempenho mínimo e n é o número de exemplos. Apesar da vantagem de ter um grupo maior de algoritmos, a disposição das recomendações não é ordenada, de forma a dificultar o entendimento de qual das propostas é a melhor.

Uma abordagem mais robusta que as anteriores é retornar uma lista em ordem decrescente de relevância, de forma a indicar uma sugestão ordenada dos algoritmos mais propícios a atender a tarefa em questão com bom desempenho. Essa ordem pode ser linear e completa, o que significa que a lista possui tamanho fixo e a única informação provida pelo modelo é uma ordem linear das sugestões sem qualquer distinção de empates. Outra maneira é representar a ordem como fraca e completa. Essa abordagem é parecida com a anterior, com o adicional de prover ao usuário a informação de eventuais empates. A outra maneira é tratar a ordem como linear e incompleta. Nessa abordagem, a lista tem tamanho variável, pois são eliminados dela

entradas algoritmos que o modelo prevê que não resultará em uma solução considerada boa. Nessa estratégia, existe a possibilidade do modelo não realizar uma recomendação de algoritmo para um determinado meta-exemplo. Por último, a ordem pode ser fraca e incompleta. Nesse caso, ela é idêntica ao cenário anterior, mas podendo informar empates.

2.2.3 TÉCNICAS DE GERAÇÃO DE BASES DE DADOS

Uma questão importante em Meta-Aprendizado se refere à disponibilidade de um número suficiente de instâncias de problemas (ou conjuntos de dados) para obtenção de meta-dados para a construção de um modelo de (meta-) indução confiável. Conforme mencionado em [21], por exemplo, o *UCI Repository* [1] é a fonte mais comum de instâncias de problemas para Meta-Aprendizado. Contudo, ela contém um pouco mais de 160 bases de dados de classificação. Dessa forma, a maioria das pesquisas de Meta-Aprendizado é realizada com uma quantidade pequena de meta-exemplos, comprometendo a confiabilidade dos meta-aprendizes. A baixa disponibilidade de problemas também pode ser observada em outros domínios de aplicação [58]. Nos últimos anos, essa questão tem recebido uma crescente quantidade de atenção, sendo resolvida de forma satisfatória com técnicas de geração de dados.

Existem basicamente duas abordagens tradicionais para aumentar o número de conjunto de dados para Meta-Aprendizado. Uma utiliza o modelo de geração de bases de dados sintéticas [2],[13],[31],[58]. A outra abordagem está relacionada à manipulação de conjuntos de dados existentes [11],[3],[22].

A abordagem de geração de bases sintéticas [2],[13],[31],[58] geralmente envolve um modelo de geração de dados que permite produzir novas bases com meta-atributos específicos. Em [2], os autores desenvolveram algoritmos genéticos para produzir conjuntos de dados sintéticos tomando como base valores de meta-atributos providos pelo usuário. Cada conjunto de dados é representado como um cromossomo em algoritmo genético. A função que representa o *fitness* do modelo é definida levando em conta similaridades entre os meta-atributos obtidos a partir dos conjuntos de dados produzidos e os valores desejados. A flexibilidade deste modelo permite que uma grande variedade de conjuntos de dados possa ser produzida à medida que um gerador de dados está disponível. A técnica também é utilizada em outros domínios de

aplicação, como geração de dados de ventos para séries temporais [31], aplicações na área de estudos biológicos [58], dentre outros.

Apesar do sucesso dessas técnicas, uma limitação geral da geração de conjuntos de dados sintéticos é a necessidade de ter que assumir, a priori, conhecimento ou modelo sobre como os dados são produzidos, criando uma forte dependência de decisões humanas.

Outra abordagem tradicional para aumentar o número de meta-exemplos na base de dados envolve a manipulação de conjuntos de dados existentes [21],[11],[3],[22]. Nesta abordagem, instâncias reais de problemas são utilizadas para produzir novas bases de dados através de uma alteração nos dados ou manipulação da estrutura dessas bases de dados de referência.

2.2.4 *DATASETOIDS*

Para este trabalho, que é focado na seleção de algoritmos para tarefas de classificação, adotamos um método para manipulação de conjuntos de dados chamada *datasetoids*, proposta recentemente em [21]. Motivada pela baixa disponibilidade de instâncias de problemas em Mata-Aprendizado e a necessidade de aumentar esse conjunto, *Datasetoids* é uma técnica de simples implementação que realiza uma manipulação dos conjuntos de dados existentes para formar novos elementos na base, causando um aumento eficaz do conjunto de dados inicial e melhorando a qualidade da predição dos modelos de Meta-Aprendizado. A técnica, inclusive, pode ser utilizada em outros domínios de aplicação.

Nesse método, uma nova base de dados (chamado *datasetoid*) é gerada a partir de uma instância real problema (ou seja, uma base de dados de classificação) através da alteração do atributo alvo com um atributo descritor simbólico (Figura 2.3). Para poder gerar *datasetoids* para classificação o processo de manipulação é repetido para todos os atributos descritores simbólicos do conjunto de dados. Dessa forma, a quantidade de *datasetoids* gerados está diretamente relacionada com a quantidade de atributos simbólicos existentes na base de dados.

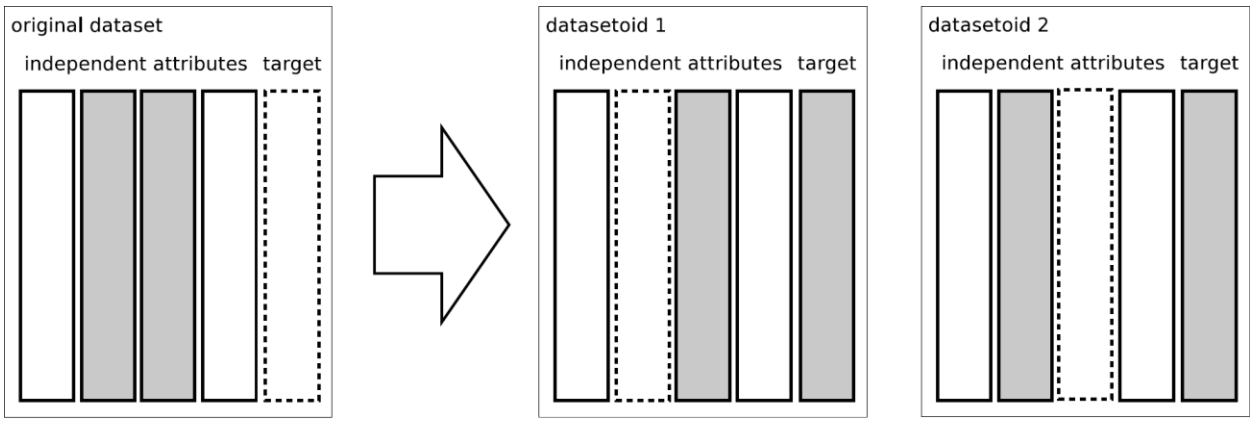

Figura 2.3: Ilustração da geração de dois *datasetoids* de classificação a partir de uma base de dados com dois atributos simbólicos (reproduzido de [21]).

Experimentos no problema de decidir quando podar uma árvore de decisão utilizando meta-dados contendo *datasetoids* obtiveram resultados melhores do que os realizados com meta-dados que só possuíam bases de dados reais [21]. Nesses experimentos, utilizando somente bases de dados do UCI, o modelo de Meta-Aprendizado obteve uma acurácia que varia de 41% a 45%, a qual foi relativamente parecida com a acurácia padrão (que considerava a classe majoritária dos meta-exemplos). Utilizando *datasetoids*, por outro lado, foi observada uma acurácia que varia de 48% a 62% (o melhor desempenho foi obtido utilizando o algoritmo Random forest como meta-aprendiz), representando um ganho de desempenho.

A técnica, contudo, apresenta alguns aspectos negativos que devem ser levados em consideração. Atributos com valores faltantes acabam sendo replicados para os datasetoids. Uma forma possível de sobrepor essa limitação é remover os exemplos que contém atributos faltantes. Outro problema que a técnica pode causar na base é a injeção de *outliers*. Uma forma de amenizar os efeitos desse comportamento é aplicar uma técnica de detecção e remoção de elementos que sejam considerados *outliers*.

2.3 CONSIDERAÇÕES FINAIS

Este capítulo discutiu Meta-Aprendizado sob a perspectiva do problema da seleção de algoritmos. Apesar de ser um problema antigo, somente nos últimos anos ele foi resolvido [20][30] usando uma abordagem de Meta-Aprendizado. A proposta citada resolve o problema através da criação de uma base de meta-exemplos que

contém o mapeamento entre as características de problemas analisados anteriormente e os algoritmos mais eficientes para cada tarefa. Dessa forma, dada uma nova instância de problema, um modelo de aprendizagem poderia facilmente inferir o melhor algoritmo para a tarefa em questão sem a necessidade de ter todos os algoritmos candidatos executados no problema.

Além disso, foi visto a importância de utilizar técnicas de geração e manipulação de conjuntos de dados, assim como conceitos e aplicações utilizadas com sucesso na literatura. Sabendo que a disponibilidade de meta-exemplos é geralmente baixa nos problemas de Meta-Aprendizado, essas técnicas permitem o aumento da base de meta-exemplos.

Meta-Aprendizado, contudo, contém um custo relativamente alto para construir sua base de meta-exemplos. Uma boa alternativa para resolver esse problema é a realização de uma seleção ativa das instâncias para compor a base de dados. No capítulo seguinte, é realizada uma revisão em Aprendizagem Ativa, descrevendo os cenários de aplicação e as estratégias mais utilizadas para selecionar somente os elementos mais informativos de um conjunto de dados.

3 APRENDIZAGEM ATIVA

Aprendizagem supervisionada é uma tarefa de aprendizagem de máquina largamente utilizada que induz o modelo através de dados de treinamento rotulados. O algoritmo de aprendizagem supervisionada recebe inicialmente um conjunto de elementos rotulados para realizar treinamento do modelo e ajuste da função de mapeamento. Após treinamento, o modelo gerado usa sua função de mapeamento ajustada para predizer a classe de novos elementos. Figura 3.1 apresenta uma visão geral do framework de aprendizagem supervisionada.

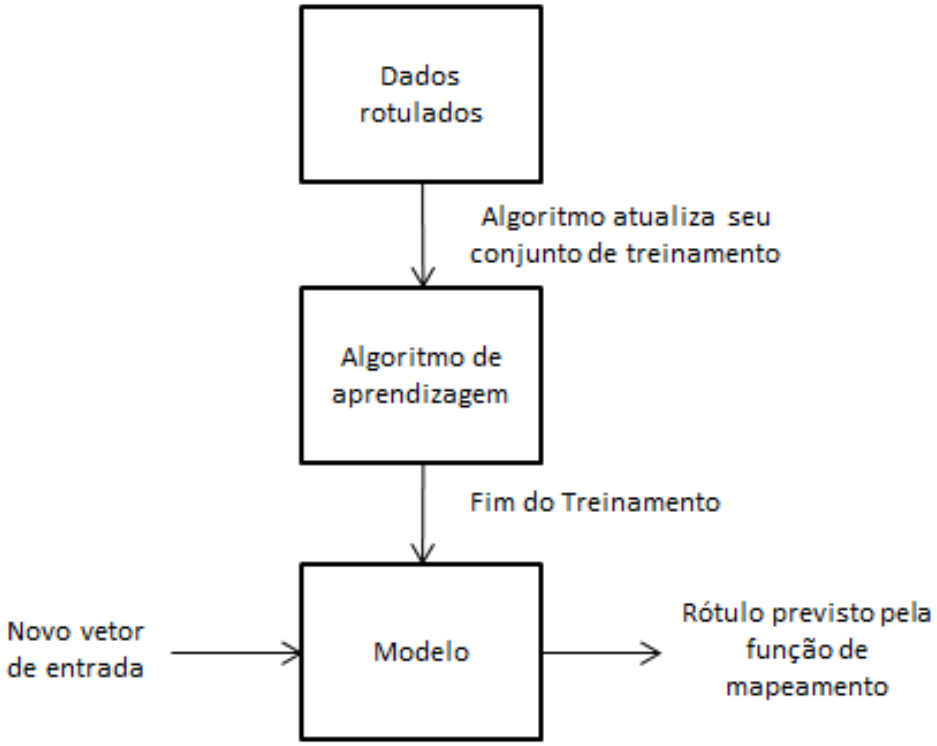

Figura 3.1: Framework de aprendizagem supervisionada

A construção de um modelo de aprendizagem supervisionada confiável requer uma quantidade considerável de exemplos de treinamento. Na prática, o processo de aquisição de rotular exemplos de treinamento pode ter um custo alto associado. Na tarefa de categorização de texto, por exemplo, o processo de classificação requer uma quantidade considerável de tempo, dinheiro ou recursos humanos para ler e informar a classificação (política, esportes, entretenimento, etc.).

Aprendizagem Ativa surge nesse contexto como uma forma de manter (e até mesmo melhorar) o desempenho de algoritmos de treinamento demandando uma quantidade menor de exemplos de treinamento [23]. Trata-se de uma abordagem em Aprendizagem de Máquina que possibilita um controle sobre os exemplos utilizados para treinamento por parte do algoritmo de aprendizado [7], sendo aplicado em domínios nos quais a aquisição de exemplos rotulados é um processo custoso.

Diferentemente de um algoritmo de aprendizagem que é inicializado com todo o conjunto de dados rotulados, um algoritmo de aprendizagem ativa é inicializado com um conjunto pequeno de exemplos de treinamento rotulados. Dado um repositório de exemplos não rotulados, o método escolhe o exemplo mais informativo e solicita seu rótulo de classe para um determinado oráculo. O processo se repete fazendo com que a base de treinamento cresça gradativamente ao passo que a taxa de erro do modelo tende a diminuir cada vez mais. Ao final de sua execução, o modelo gerado possui um nível de desempenho satisfatório com uma quantidade menor de exemplos de treinamento, reduzindo o custo da construção de uma base de meta-exemplos.

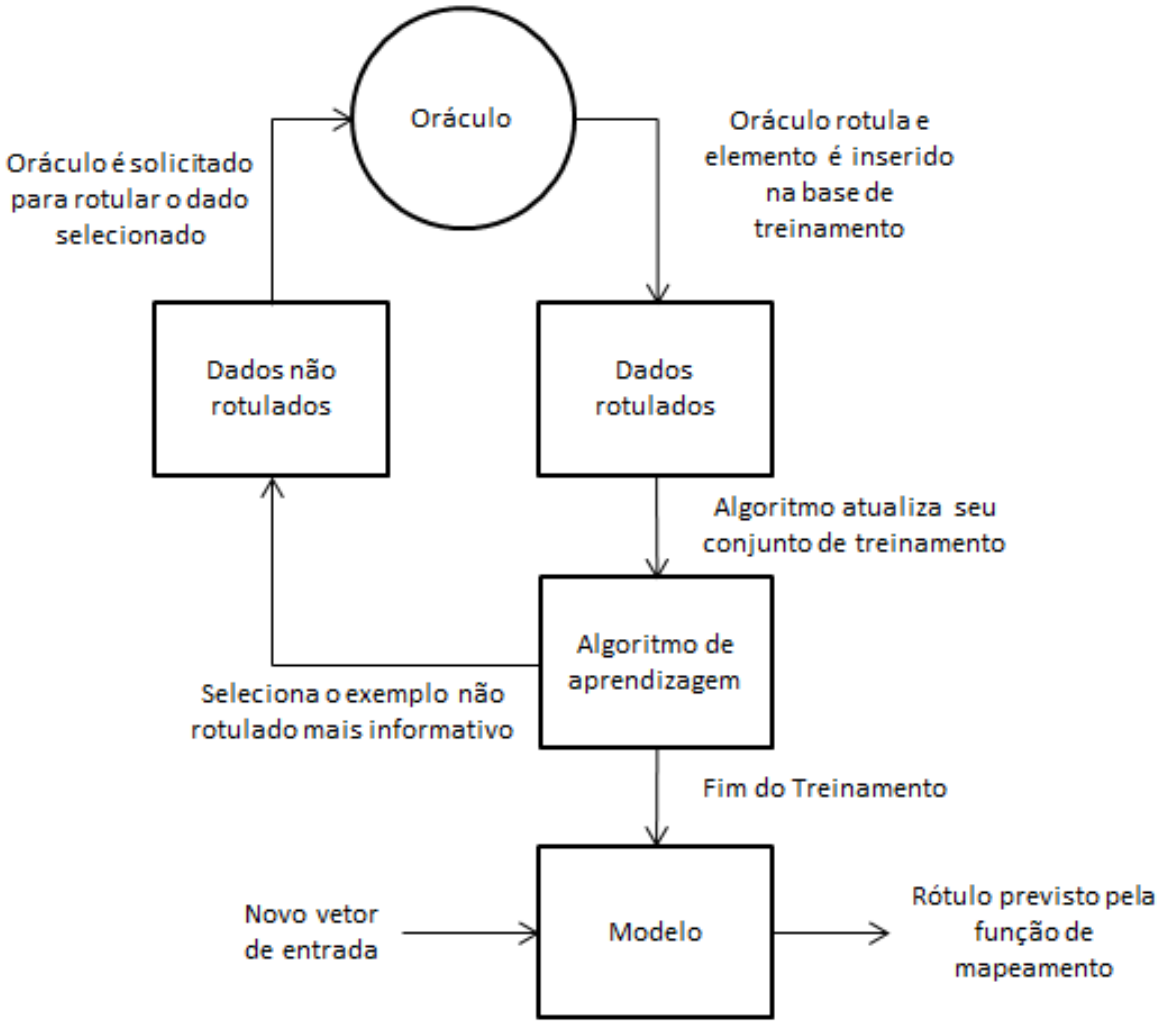

Figura 3.2: Framework de aprendizagem ativa.

A Figura 3.2 apresenta o framework de aprendizagem ativa. A técnica permite a construção de modelos de aprendizagem de alto desempenho usando em geral uma quantidade de exemplos rotulados bem menor. Por causa dessas vantagens, esse método tem sido largamente utilizado em várias aplicações práticas, como classificação de texto [45], [42], diagnóstico de câncer [43], classificação de imagens [44], reconhecimento de voz [17], dentre outras [12], [16], [18].

3.1 CENÁRIOS

A técnica de Aprendizagem Ativa pode ser aplicada em diferentes cenários de seleção de novos exemplos. As abordagens mais comumente utilizadas incluem *membership query*, *stream-based selective sampling* e *pool-based sampling*. Figura 3.3 ilustra as diferenças entre os cenários mencionados.

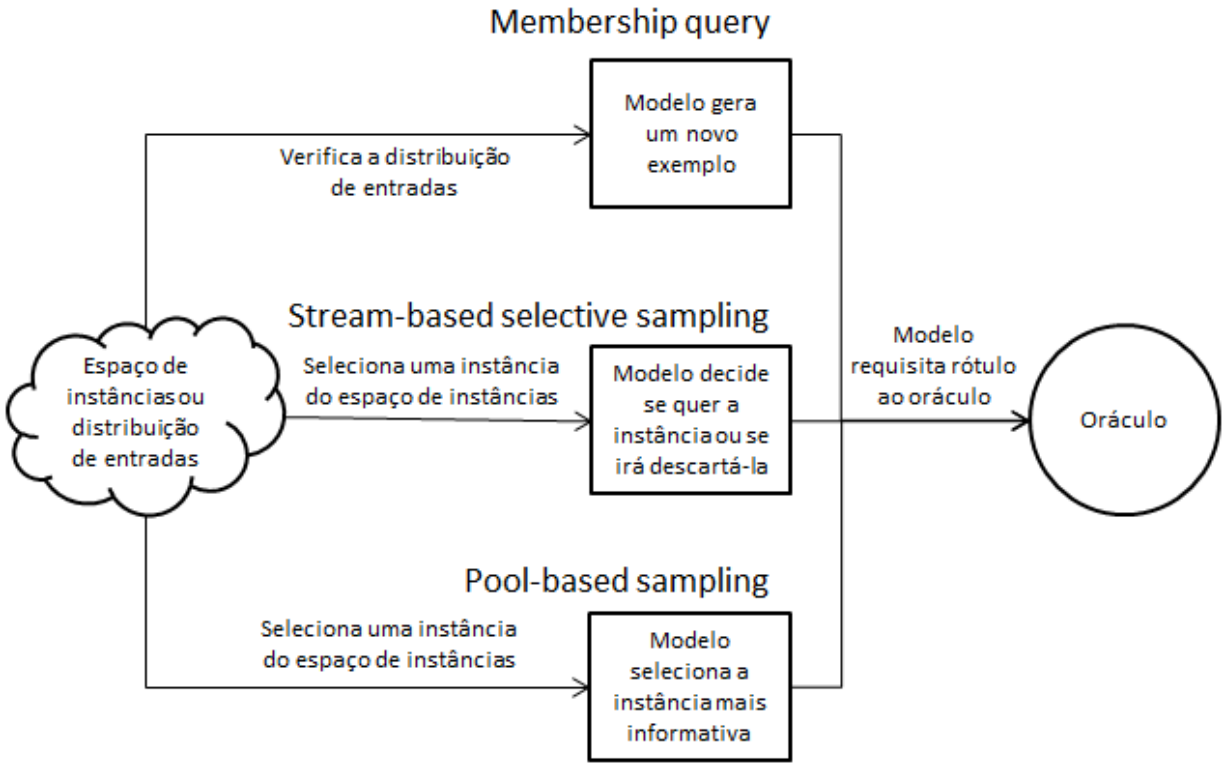

Figura 3.3: Diferença entre os três cenários mais comuns de aprendizagem ativa

3.1.1 *MEMBERSHIP QUERY*

A estratégia de *membership query* foi proposta em [46]. Nessa abordagem, o classificador é inicializado com um conjunto de exemplos rotulados de treinamento. O aprendiz ativo pode adquirir novos exemplos através da geração de exemplos sintéticos e requisitar ao oráculo o rótulo correspondente.

A vantagem desta abordagem é que como os exemplos são gerados pelo próprio modelo, eles podem ser feitos de forma a sempre serem os exemplos mais

informativos possível, aumentando o desempenho da predição do modelo. Algumas estratégias obtiveram resultados satisfatórios em alguns domínios de problemas [47].

Apesar de mostrar-se vantajoso em vários cenários de aplicação, rotular instâncias arbitrariamente pode apresentar-se inviável quando o oráculo é um humano [23]. Devido ao fato de geralmente não existir garantias que o exemplo gerado seja passível a uma interpretação amigável em nível de usuário, valores estranhos de atributos podem dificultar a análise e interpretação do dado de entrada por parte do oráculo humano, dificultando a definição do rótulo apropriado. No pior caso, problemas dessa natureza podem acabar interrompendo o processo de aprendizagem.

3.1.2 *STREAM-BASED SELECTIVE SAMPLING*

A abordagem *stream-based selective sampling*, também conhecida como *sequential sampling* [48], assume que a obtenção de um exemplo não rotulado é uma tarefa de custo desprezível. Dessa forma, o aprendiz seleciona um exemplo por vez do conjunto de dados e o algoritmo de aprendizagem decide se quer o exemplo ou se irá descartá-lo. A busca é realizada de forma sequencial e a decisão da *query* é feita de forma individual;

A decisão por parte do algoritmo de aprendizagem de descartar ou não um exemplo novo pode levar em conta uma série de fatores. Uma abordagem comum é definir uma “medida de informatividade” e realizar uma decisão aleatória e enviesada pelas instâncias mais informativas para serem selecionadas [49]. Outras propostas envolvem um cálculo de região de incerteza e selecionar somente as instâncias que fizerem parte dessa região [48].

Algumas aplicações reais obtiveram sucesso no uso desta abordagem, como busca de informações [51], aplicações de sensores [50], etc.

3.1.3 *POOL-BASED SAMPLING*

A estratégia de *pool-based sampling* [52] recebe esse nome porque ela gerencia um repositório (também conhecido como *pool*) que contém uma quantidade geralmente expressiva de exemplos não rotulados. Essa abordagem é bastante utilizada em estudos acadêmicos e aplicações reais devido a sua eficácia e facilidade

de implementação. Nessa abordagem, o algoritmo de aprendizado possui um conjunto pequeno de instâncias rotuladas. O método de aprendizagem ativa realiza um *ranking* de todo o repositório e somente depois seleciona desse repositório o exemplo mais informativo para ter seu rótulo definido pelo oráculo.

O cenário *pool-based* é aplicado em muitas tarefas reais de aprendizagem de máquina, como classificação de vídeos [55], classificação de texto [53], extração de informações [54], dentre outros. Apesar do cenário *pool-based* ser bastante utilizado, o cenário *stream-based* é mais vantajoso nos casos que existe baixa disponibilidade de recursos de máquina [23], como memória ou processamento. Isso ocorre, por exemplo, em dispositivos móveis e embarcados.

3.2 ESTRATÉGIAS DE *QUERIES*

A definição de como será medido o grau de informatividade de uma instância, tenha ela sido gerada sinteticamente ou proveniente de um repositório de instâncias, é uma questão importante para o sucesso da aplicação. De uma maneira geral, o grau de informatividade de um exemplo está intimamente relacionado ao desempenho do modelo, de forma que os exemplos mais informativos tendem a ser os que maximizam o desempenho do modelo. Existem muitas estratégias de *queries* propostas na literatura e esta seção tem como objetivo realizar uma introdução nas estratégias mais utilizadas.

3.2.1 *UNCERTAINTY SAMPLING*

A técnica de *uncertainty sampling* [52] é amplamente utilizada devido a sua simplicidade e eficácia. Nessa abordagem, o exemplo não rotulado considerado mais informativo é o que o modelo tem a menor certeza de predição. Sendo assim, caso o algoritmo de aprendizagem tenha certeza de que sabe predizer a classe de um determinado elemento ainda não rotulado, esse elemento não irá agregar valor ao modelo. Por outro lado, se o modelo não tiver certeza sobre a predição do novo elemento (ou seja, se o algoritmo de aprendizagem não tiver certeza que sabe classificar o exemplo não rotulado), existem boas chances de essa instância agregar valor ao modelo caso ela seja rotulada pelo oráculo e adicionada ao conjunto de treinamento.

Settles discorre sobre três abordagens de medição do grau de incerteza de um exemplo não rotulado: *least confident*, *margin sampling* e *entropy*. Seja U o conjunto de todos os exemplos não rotulados e Y todos os valores possíveis de classificação, o exemplo mais incerto x^* é definido da seguinte forma na abordagem *least confidente*:

$$x^* = \underset{x \in U}{argmin}\{P(y_1|x)\} \; onde \; y_1 = \underset{y \in Y}{argmax}\{P(y|x)\} \tag{3.1}$$

Um problema na estratégia de *least confident* é que ela desconsidera informação sobre os demais rótulos possíveis, considerando apenas o mais provável. Uma forma de resolver essa limitação é usar a técnica de *margin sampling*. Nessa abordagem, a incerteza é calculada através da diferença de probabilidade entre as duas classes mais prováveis, conforme fórmula abaixo:

$$\begin{gathered} x^* = \underset{x \in U}{argmin}\{P(y_1|x) - P(y_2|x)\} \\ onde \; y_1 = \underset{y \in Y}{argmax}\{P(y|x)\}, y_2 = \underset{y \in Y}{argmax}\{P(y|x)\} \end{gathered} \tag{3.2}$$

A abordagem de *margin sampling* resolve as limitações da abordagem *least confident*. Instâncias com margem mais estreita são consideradas mais incertas (e consequentemente mais difíceis de rotular) do que as que apresentam margens maiores. Mesmo assim, para conjuntos de dados com muitas possibilidades de classificação a técnica de *margin sampling* ainda ignora muita informação relevante. *Entropy* é uma estratégia que resolve essa limitação, pois itera sobre todos os rótulos possíveis. Essa técnica é provavelmente a estratégia mais comum para medição de incerteza. A abordagem, como o próprio nome enfatiza, considera como incerto o exemplo que possui a maior entropia. Formalmente, a incerteza é calculada da seguinte forma:

$$x^* = \underset{x \in U}{argmax} \left\{ - \sum_{y \in U} P(y|x) \log P(y|x) \right\} \tag{3.3}$$

As três abordagens mencionadas apresentam comportamento distinto em problemas multi-classes. A abordagem baseada em entropia, especificamente, generaliza mais facilmente para tarefas que apresentem mais de dois rótulos de classificação. Por outro lado, em problemas de saída binária esses métodos funcionam de forma relativamente semelhante [23]. Nesse último caso, todos eles vão selecionar o exemplo com incerteza próxima a 0.5.

3.2.2 *DENSITY-BASED SAMPLING*

Density-based Sampling é uma estratégia que considera como mais informativo os exemplos de densidade mais alta. A motivação dessa estratégia é fazer com que um modelo de aprendizado consiga realizar predições corretas mesmo em um espaço denso de instâncias. Dessa forma, os exemplos que apresentam densidade mais alta são selecionados pelo modelo para serem rotulados pelo oráculo e em seguida integrados ao conjunto de treinamento do algoritmo de aprendizado.

Essa técnica, contudo, geralmente requer uma combinação com critérios adicionais de seleção de exemplos. Isso porque se o algoritmo de aprendizagem selecionar sempre o exemplo com maior densidade em cada iteração existe uma probabilidade relativamente alta de o modelo selecionar exemplos somente de uma mesma área. Esse comportamento pode fazer com que o algoritmo de aprendizado atinja um estado de sobreajustamento, fazendo com que o modelo só consiga classificar corretamente instâncias que residam no espaço de alta densidade. Para evitar esse comportamento indesejado, a técnica *density-based* pode ser combinada com outra técnica, como *uncertainty sampling*. Nesse caso, um balanceamento é criado utilizando os dois critérios, de forma a selecionar os exemplos que maximizam o modelo de aprendizado [56]. Outra estratégia consiste em clusterizar os elementos não rotulados e então selecionar os exemplos mais representativos em cada cluster durante o treinamento [57].

3.2.3 *EXPECTED ERROR REDUCTION*

A estratégia de *expected error reduction* [27] seleciona os exemplos de forma a minimizar o erro de predição de classe. É uma abordagem bastante natural, se

considerarmos que a maior motivação de Aprendizagem Ativa é aumentar o desempenho dos modelos de aprendizado usando menos instâncias.

Diferentemente dos métodos anteriores, que utilizam critérios indiretos para seleção do elemento (no caso, incerteza e densidade), a abordagem de *expected error reduction* foca em reduzir o erro de generalização do modelo. Para atingir esse objetivo, cada entrada do repositório de exemplos não rotulados possui seu rótulo predito e em seguida é gerado um novo modelo que contém a base de treinamento mais o exemplo selecionado. Ao final, cada modelo é avaliado e o que apresentar a maior taxa de acerto tem seu exemplo correspondente considerado o mais informativo e é selecionado para fazer parte da base de dados.

Apesar de sua simplicidade e praticidade, esse método tem uma desvantagem: como os rótulos de todos os exemplos do repositório são desconhecidos, eles são gerados a partir da predição do algoritmo de aprendizagem, que por sua vez possui sua função de mapeamento ajustada apenas com os dados de treinamento que adquiriu até o momento, podendo resultar em estimativas erradas.

3.3 CONSIDERAÇÕES FINAIS

Este capítulo realizou uma revisão em Aprendizagem Ativa, discutindo desde a motivação da técnica, até os conceitos, cenários de execuções e estratégias de *queries* mais usadas.

Dado que Aprendizagem Ativa pode ser aplicada em diferentes cenários nos quais os algoritmos de aprendizagem podem realizar *queries*, detalhamos os cenários mais utilizados na literatura. Além disso, dentro do contexto de um cenário qualquer, existe a necessidade de medir o grau de informatividade de instâncias não rotuladas. Diferentes estratégias foram propostas com o objetivo de medir e definir a instância mais informativa, que variam entre técnicas que retiram uma amostra de um repositório usando uma determinada distribuição e técnicas que geram instâncias novas.

Como já foi dito anteriormente, um dos problemas de Meta-Aprendizado está relacionado ao alto custo computacional de criar da base de meta-exemplos. Isso porque, dada uma nova instância de problema, é preciso executar nela todos os algoritmos candidatos para armazenar no meta-exemplo as informações de desempenho desses algoritmos. Tendo em vista os benefícios da Aprendizagem Ativa

no que diz respeito a manter (e em alguns casos, aumentar) as taxas de acerto de algoritmos de aprendizado usando menos exemplos de treinamento, a técnica pode ser aplicada em combinação com Meta-Aprendizado a fim de reduzir o custo computacional da construção da base de meta-exemplos sem comprometer a taxa de acerto do modelo.

O capítulo seguinte detalha o modelo usado nos experimentos. É realizada uma discussão sobre os algoritmos meta-aprendizes, assim como o uso de Meta-Aprendizado em combinação com Aprendizagem Ativa e detalhes da técnica de detecção de *outliers* implementada.

4 APRENDIZAGEM ATIVA E *DATASETOIDS* EM META-APRENDIZADO PARA SELEÇÃO DE ALGORITMOS

O uso de técnicas de geração de dados – seja gerando bases de dados sintéticas ou manipuladas – aumentam a quantidade de dados para produzir meta-exemplos, causando um forte impacto positivo no desempenho de Meta-Aprendizado. Entretanto, o uso desse tipo de técnica pode introduzir outras dificuldades. Primeiramente, não obstante ocorre a produção de bases de dados que trazem consigo ruído ou informações redundantes [21]. Além disso, conforme mencionado, os rótulos gerados em Meta-Aprendizado são adquiridos através da avaliação empírica dos algoritmos candidatos nos conjuntos de dados disponíveis, o que pode ser um procedimento computacionalmente caro. Por esse motivo, uma vez que pode haver uma quantidade expressiva de informações redundantes e ruídos, executar os algoritmos em *todos* os conjuntos de dados não é a melhor estratégia para produzir meta-exemplos. Isso faz com que os problemas de selecionar conjuntos de dados e remover *outliers* sejam uma questão importante em Meta-Aprendizado.

Baseando-se na motivação acima, Aprendizagem Ativa tem sido integrada ao contexto de Meta-Aprendizado com o objetivo de selecionar conjuntos de dados para a geração de meta-exemplos [14]. Como já foi dito antes, Aprendizagem Ativa é ideal para domínios nos quais a aquisição de exemplos rotulados é um processo custoso. Isso faz com que Meta-Aprendizado seja um bom candidato para a técnica mencionada.

Figura 4.1 ilustra o processo de Meta-Aprendizado combinado com Aprendizagem Ativa, o qual é denominado Meta-Aprendizado Ativo (*Active Meta-Learning*). Um repositório de bases de dados candidatas é usada para produzir um conjunto de meta-exemplos sem rótulos (ou seja, exemplos compostos por meta-atributos descritivos, mas sem rótulo de classe). Este repositório de bases é composto por conjuntos originais e *datasetoids* (para maiores detalhes sobre *datasetoids*, consultar seção 2.2.4). Dado um conjunto inicial de meta-exemplos rotulados, o módulo de Aprendizagem Ativa seleciona o próximo exemplo sem rótulo para ser classificado e incluído na base de meta-exemplos. O processo de classificação é realizado através da execução dos algoritmos candidatos na base de dados selecionada. O algoritmo que obtém a melhor solução se torna o rótulo do respectivo meta-exemplo. Após inclusão

do novo elemento na base de treinamento, o modelo é treinado novamente. O procedimento itera até que alguma condição de parada ocorra.

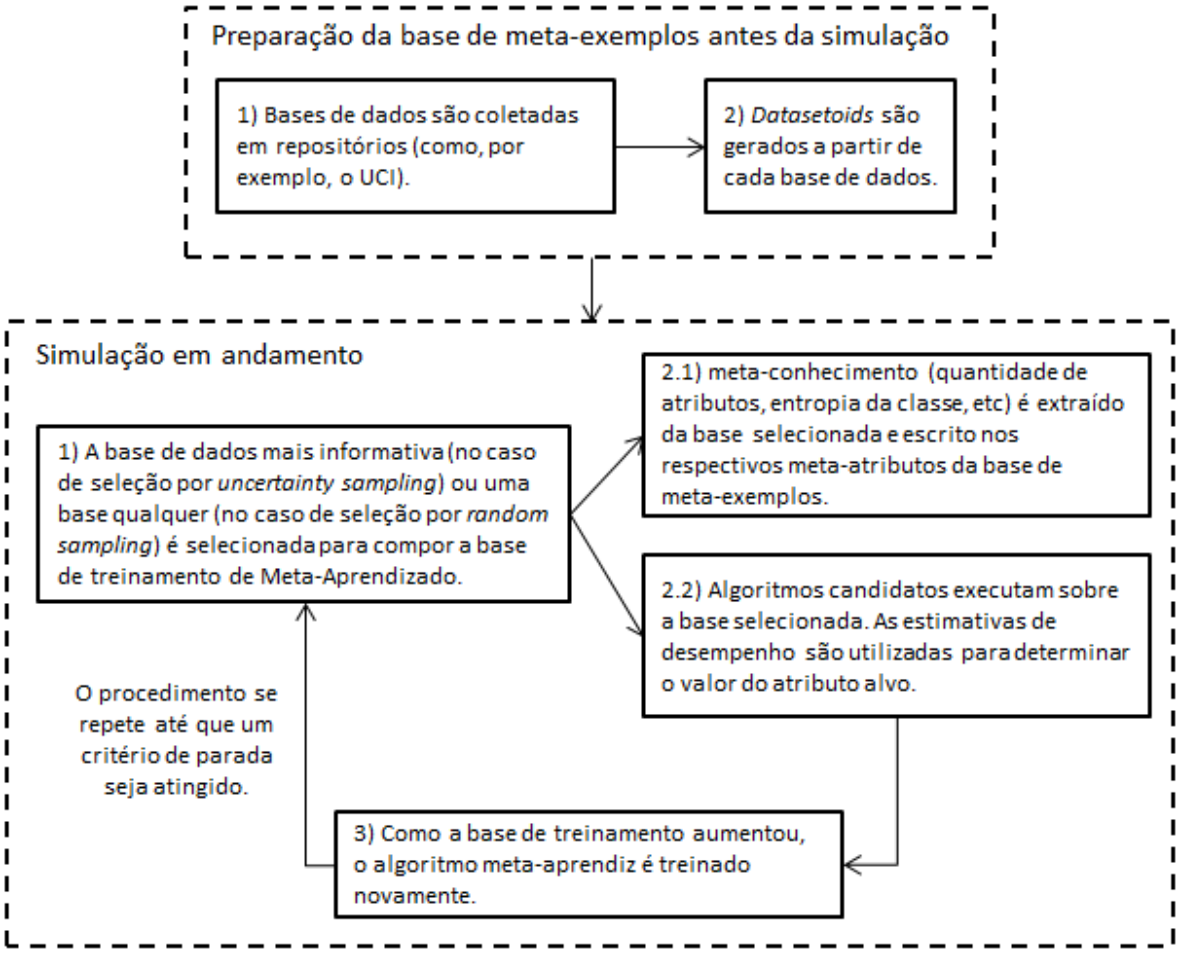

Figura 4.1: Processo de Meta-Aprendizado Ativo (adaptado de [15])

Em [15], Meta-Aprendizado Ativo foi investigado no contexto de *datasetoids* para aumentar o número de conjuntos de dados disponíveis para produzir meta-exemplos e ao mesmo tempo evitar o excessivo custo computacional para coletar meta-dados. Nesse trabalho, um repositório de problemas candidatos foi composto por um conjunto de bases de dados reais coletadas do UCI e seus *datasetoids* correspondentes.

Diferentes experimentos foram realizados em [15] através da adoção de método baseado em critério de incerteza (*uncertainty sampling*) para a aprendizagem ativa e o algoritmo k-Nearest Neighbor como meta-aprendiz. Nesses experimentos, foi possível observar taxas de acerto similares aos resultados originais obtidos com *datasetoids* apresentados em [21] usando, contudo, menos de 1/3 do repositório de

problemas candidatos. Os resultados obtidos mostraram que é possível produzir um maior conjunto de meta-exemplos sem aumentar o custo computacional.

Apesar dos resultados promissores, algumas limitações podem ser apontadas de forma a motivar uma investigação mais profunda. Primeiramente, os experimentos anteriores focaram em um único meta-aprendiz (k-NN) e método de aprendizagem ativa, o que apresentou um comportamento instável no processo de seleção. Além disso, o processo de aprendizagem ativa tem se mostrado muito sensível ao conjunto inicial de meta-exemplos. Em alguns cenários de experimentos, o método de *uncertainty sampling* não conseguiu obter resultados mais satisfatórios do que os atingidos por um processo simples de seleção aleatório de conjuntos de dados. Outra limitação de métodos baseados em incerteza é que eles dão margem à seleção de exemplos considerados *outliers* [27]. Exemplos considerados *outliers* podem ter alto grau de incerteza, mas não devem ser considerados informativos.

O algoritmo Random forest tem se mostrado um classificador de eficiência bastante satisfatória em diferentes contextos. De fato, no trabalho original de *datasetoids*, o Random forest obteve a melhor taxa de desempenho dentre todos os meta-aprendizes avaliados. No nosso trabalho, experimentos foram realizados usando *uncertainty sampling* baseado em entropia para o algoritmo Random forest e k-NN. Diferentes cenários de experimentos foram considerados com o objetivo de avaliar o impacto do conjunto inicial de meta-exemplos rotulados. Conforme será apresentado, no trabalho corrente observamos resultados positivos em cenários que representaram um desafio nos trabalhos anteriores.

Nesta dissertação, realizamos uma investigação mais profunda em Meta-Aprendizado Ativo, combinado com *datasetoids* e um procedimento de detecção de *outliers*. Primeiramente, são removidos meta-exemplos não rotulados considerados *outliers*, e então é usada uma técnica de *uncertainty sampling* para selecionar progressivamente os meta-exemplos não rotulados mais informativos para serem rotulados.

4.1 META-APRENDIZES

Os algoritmos utilizados como meta-aprendizes foram o k-NN e o Random forest. O k-NN está entre os mais simples e intuitivos métodos de classificação e é bastante utilizado em problemas de classificação na área de aprendizagem de máquina, não somente por causa da sua facilidade de implementação, mas também devido à sua capacidade de aprender funções complexas e, principalmente, a sua eficácia. Trata-se de um algoritmo baseado em instâncias capaz de classificar um novo exemplo tomando como base os exemplos mais próximos no espaço de características. No contexto de Meta-Aprendizado, dada uma nova instância de problema, o algoritmo calcula a distância euclidiana entre os atributos da nova instância e os atributos de cada exemplo de treinamento rotulado. Conhecendo a distância entre os exemplos, o rótulo da nova instância de problema (ou seja, a estratégia recomendada pelo meta-aprendiz para resolver o problema proposto) será inferido a partir do rótulo predominante entre seus k vizinhos mais próximos. Nos nossos experimentos, o k-NN foi avaliado com $k = 5$, valor determinado empiricamente.

Apesar das vantagens do k-NN, uma desvantagem inerente ao seu processo é que ruído pode comprometer a precisão da classificação. Isso significa que um atributo com dados inconsistentes ou menos relevantes podem facilmente fazer com que o meta-aprendiz passe a fazer recomendações erradas. Na busca por um meta-aprendiz mais robusto, o algoritmo Random forest [8] surge como um candidato mais preciso e tolerante a falhas. Seu processo de indução é eficiente (rápido de treinar e realizar predições, além de permitir paralelismo durante todo o treinamento). É resistente a *outliers*, os dados não precisam de pré-processamento e sua modelagem resolve automaticamente valores faltantes. O Random forest é um conjunto de classificadores que pode ser visto como um classificador por proximidade de vizinhos. A ideia por trás do algoritmo é controlar um conjunto de árvores de decisão para induzir um modelo. Cada árvore é construída utilizando um diferente e aleatório subconjunto do conjunto de treinamento, chamado *bootstrap sample*. Em seguida, cada árvore de decisão constrói sua solução de forma independente, em um processo chamado *bootstrap aggregation* [59]. Dada uma nova instância de problema, a abordagem utilizada pelo meta-aprendiz para recomendar o rótulo é realizada em um processo de votação onde cada árvore recomenda seu rótulo. O valor final de classificação é definido pelo rótulo

que apresentar o maior número de votos. A implementação do Random forest utilizada foi do pacote Weka e parametrizado nos nossos experimentos com 100 árvores.

4.2 APRENDIZAGEM ATIVA

Meta-Aprendizado Ativo foi avaliado no nosso trabalho através do uso do método *uncertainty sampling* para o algoritmo Random forest e k-NN. Esse método seleciona exemplos não rotulados os quais o classificador tem a maior incerteza de classificação, ou seja, a menor capacidade de predição do rótulo da classe. A seleção de exemplos baseado em incerteza tem sido bastante utilizada na literatura de aprendizagem ativa [23], uma vez que eles tendem a reduzir a incerteza do classificador já que os exemplos mais difíceis de classificar são rotulados e assim utilizados na base de dados de treinamento do classificador. Usamos uma estratégia de incerteza que busca a instância cuja predição contém a maior entropia. *Query* baseada em entropia é um modelo que generaliza mais facilmente modelos probabilísticos para instâncias de estrutura mais complexas.

Seja $E = \{e_1, \dots, e_2\}$ o conjunto de n problemas usados para gerar um conjunto de n meta-exemplos $ME = \{me_1, \dots, me_n\}$. Cada meta-exemplo $me_i \in ME$ está relacionado a um problema e armazena os valores de p características $X_1, \dots, X_p$ para o problema, assim como o valor do atributo classe C, que representa a informação de desempenho.

Seja $C = \{c_1, \dots, c_L\}$ o domínio de atributos classe C, onde L representa a quantidade de rótulos possíveis de classe que c pode assumir, cada meta-exemplo $me_i \in ME$ é representado pelo par $(x_i, C(e_i))$ armazenando: (1) a descrição x_i para o problema e_i, onde $x_i = (x_i^1, \dots x_i^p)$ e $x_i^j = X_j(e_i)$; e (2) o rótulo de classe associado a e_i, ou seja, $C(e_i) = c_l$, onde $c_l \in C$.

A probabilidade de distribuição para um problema não rotulado $\tilde{e}$ pode ser representada como:

$$pc(\tilde{e}|E) = (p(C(\tilde{e}) = c_1|E), \dots, (p(C(\tilde{e}) = c_L|E) \quad (4.1)$$

No Random forest, a distribuição de probabilidade de classe para um dado exemplo é estimado como sendo o número de votos que cada rótulo de classe recebe

dentre o número total de árvores (100 para os experimentos apresentados neste trabalho). No k-NN, a distribuição de probabilidade é calculada como a razão entre a quantidade de votos do rótulo mais expressivo (ou seja, o rótulo que recebeu o maior número de votos) e o valor de k parametrizado no algoritmo.

De acordo com [23], a entropia da distribuição de probabilidade reflete a incerteza do classificador em predizer o valor da classe. A entropia da probabilidade de distribuição é computada da seguinte forma:

$$Ent(\tilde{e}|E) = -\sum_{l=1}^{L} p(C(\tilde{e}) = c_l|E) * \log_2 p(C(\tilde{e}) = c_l|E) \quad (4.2)$$

Se a distribuição de probabilidade é muito espalhada, o valor de entropia será alto, indicando que o classificador não é certo de sua predição. Por outro lado, se a distribuição de probabilidade é muito focada em um único rótulo de classe, a entropia é baixa, indicando baixo grau de incerteza na predição do valor de classe.

A instância mais informativa (ou seja, o exemplo com o maior grau de incerteza) é definida por uma seleção probabilística realizada pelo algoritmo da roleta (*roulette-wheel algorithm*). Como classificações errôneas comprometem a acurácia do classificador, encontrar a instância mais informativa para o treinamento é uma tarefa de relevância significativa. Seleção determinística tende a injetar *outliers* no conjunto de dados com mais frequência do que uma seleção probabilística [24]. Assim, o algoritmo da roleta aumenta a robustez do modelo contra *outliers* realizando uma seleção aleatória enviesada pela incerteza de predição da classe por parte do classificador.

O algoritmo da roleta itera os exemplos não rotulados de acordo com suas probabilidades associadas. Nesse método, a probabilidade de um meta-exemplo ser selecionado é proporcional a sua incerteza. O meta-exemplo selecionado é rotulado (isto é, o valor de classe $C(\tilde{e})$ é defindo) através da estimativa de desempenho dos algoritmos candidatos na base de dados correspondente. Por último, o novo meta-exemplo rotulado $(\tilde{x}, C(\tilde{e}))$ é incluído nos meta-dados.

4.3 DETECÇÃO DE *OUTLIERS*

No protótipo desenvolvido, implementamos o mesmo modelo que foi usado em [28], que constitui basicamente da adaptação do método baseado em distância proposto em [26] para detecção de *outliers*. No modelo proposto, foi eliminado do conjunto de meta-exemplos não rotulados os que mais desviavam dos demais em termos de distância.

Seja $\tilde{E} = \{\tilde{e}_1, \dots, \tilde{e}_m\}$ o conjunto de m problemas associados ao conjunto disponível de m meta-exemplos não rotulados $\widetilde{ME} = \{\mu, \dots, \widetilde{me}_m\}$. Cada $\widetilde{me}_i \in \widetilde{ME}$ armazena a descrição de $\tilde{x}_i$ de um problema $\tilde{e}_i \in \tilde{E}$. Para detectar *outliers*, primeiramente calculamos a distância média entre os meta-exemplos em $\widetilde{ME}$. Formalmente, para cada par diferente $(\widetilde{me}_i, \widetilde{me}_j)$, primeiramente calculamos a distância: $dist(\tilde{x}_i, \tilde{x}_j)$. Em seguida, computamos a média dessas distâncias conforme a seguir:

$$\mu_{dist} = \frac{1}{m * (m-1)} \sum_{\widetilde{me}_i, \widetilde{me}_j \in \widetilde{ME}, i \neq j}^{L} dist(\tilde{x}_i, \tilde{x}_j) \tag{4.3}$$

A medição do quanto um meta-exemplo não rotulado é considerado um outlier é computada a partir da proporção de outros meta-exemplos não rotulados em $\widetilde{ME}$ cuja distância do meta-exemplo analisado é no mínimo o valor de referência μ_{dist}. Formalmente, seja $G_i = \{\widetilde{me}_j \in \widetilde{ME} | i \neq j, dist(\tilde{x}_i, \tilde{x}_j) \geq \mu_{dist}\}$ o conjunto de meta-exemplos não rotulados que são distantes de $\widetilde{me}_i$, o grau de outlier ($OutlierDegree$) é definido como sendo:

$$OutlierDegree(\widetilde{me}_i) = \frac{G_i}{m-1} \tag{4.4}$$

Os meta-exemplos não rotulados podem ser ordenados usando essa medida, de forma que os meta-exemplos com os maiores valores de $OutlierDegree$ sejam considerados *outliers*. No nosso protótipo, os primeiros 10% dos meta-exemplos não rotulados nesse rank são removidos do conjunto de candidatos para gerar os meta-exemplos rotulados.

4.4 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados detalhes de como o modelo de Meta-Aprendizado proposto é implementado. Detalhamos os algoritmos utilizados como meta-aprendizes, assim como as vantagens de cada um. O k-NN, apesar de ser muito utilizado na literatura, possui limitações que motivam a investigação de outros classificadores para realizar a tarefa de meta-aprendiz. O Random forest surge nesse contexto como uma alternativa para suprir as limitações do k-NN e ainda aumentar o desempenho do modelo.

A combinação do aprendizado em nível meta com Aprendizagem Ativa (ou Meta-Aprendizado Ativo) permite que tenhamos um ganho expressivo no desempenho do modelo. Isso porque, como já foi dito anteriormente, o custo de rotular meta-exemplos é alto e a estratégia proposta permite realizar o treinamento utilizando um subconjunto de treinamento menor do que a quantidade total de meta-exemplos disponíveis. Esse subconjunto contém somente os meta-exemplos que tendem a agregar mais valor ao processo de indução do modelo. Aliando os conceitos mencionados a uma técnica de manipulação de dados, amenizamos os efeitos de outro problema conhecido em Meta-Aprendizado: a baixa disponibilidade de conjuntos de dados para a construção de meta-exemplos. Além disso, como contribuição adicional ao fato de termos incorporado o Random forest como meta-aprendiz no processo de Meta-Aprendizado Ativo, foi investigado a aplicação de uma técnica de detecção de *outliers* com o objetivo de reduzir o ruído e redundâncias provenientes da técnica de *datasetoids*.

No capítulo seguinte, apresentaremos os resultados dos experimentos realizados sobre o modelo proposto. Os resultados apontam um ganho de desempenho quando comparado aos obtidos nos trabalhos anteriores.

5 EXPERIMENTOS E RESULTADOS

O modelo proposto foi avaliado empiricamente na mesma tarefa de Meta-Aprendizado usada em [21]. Ela consiste em predizer, a priori, se podar uma árvore de decisão irá melhorar a qualidade do modelo ou não. Existem três classes, p, u ou t, significando, respectivamente, se o vencedor é a árvore podada, a árvore sem poda ou se eles estão empatados.

Os meta-atributos utilizados para caracterizar as bases de dados (e *datasetoids*) foram a entropia da classe e a entropia média dos atributos [5]. Apesar de certamente existirem outros meta-atributos que poderiam contribuir para melhorar os resultados de Meta-Aprendizado, o uso de medidas que obtiveram bons resultados em experimentos anteriores permite que o foco do trabalho seja testar a combinação de Meta-Aprendizado Ativo com *datasetoids*.

O conjunto de problemas usados para gerar os meta-exemplos foi extraído de 64 bases de dados de classificação do UCI. Realizando uma troca entre o atributo alvo com todos os atributos nominais (maiores detalhes sobre *datasetoids* na seção 2.2.4), 983 *datasetoids* foram obtidos. A Tabela 5.1 contém a distribuição de classes, tanto para os meta-dados obtidos por bases de dados reais quanto para os meta-dados obtidos por *datasetoids*.

Meta-dados	Árvore podada ganha (p)	Árvore sem poda ganha (u)	Empate (t)
Bases de dados	36%	23%	41%
Datasetoids	37%	10%	53%

Tabela 5.1: Distribuição de classes dos meta-dados (para bases de dados originais e *datasetoids*, respectivamente)

5.1.1 ESTIMATIVAS DE DESEMPENHO EM NÍVEL META

Considerando que os *datasetoids* são gerados a partir de conjuntos de dados, eles não podem ser tratados como problemas independentes. Em outras palavras, *datasetoids* geram meta-exemplos os quais não são independentes dos meta-

exemplos que representam as bases de dados correspondentes. Além disso, usando a mesma metodologia que [21], os *datasetoids* gerados não são considerados para o conjunto de testes. Isso significa que somente os conjuntos de dados originais são usados para verificar o modelo. Isso porque a predição de qual algoritmo é o melhor nos *datasetoids* não é relevante sob o ponto de vista de aplicação. Além disso, os *datasetoids* gerados a partir dos meta-exemplos que fazem parte do conjunto de testes são removidos da base de treinamento. O objetivo por trás dessa estratégia é garantir a independência entre os conjuntos de treinamento e teste.

Usamos a abordagem de *leave-one-out* (LOO) como em [21], significando que a um único conjunto de dados é usado como teste cada iteração. Os *datasetoids* correspondentes ao dataset de teste são desconsiderados da base de treinamento. Além disso, a medida de desempenho do modelo de Meta-Aprendizado é a acurácia de classificação. Figura 5.1 ilustra o processo de LOO.

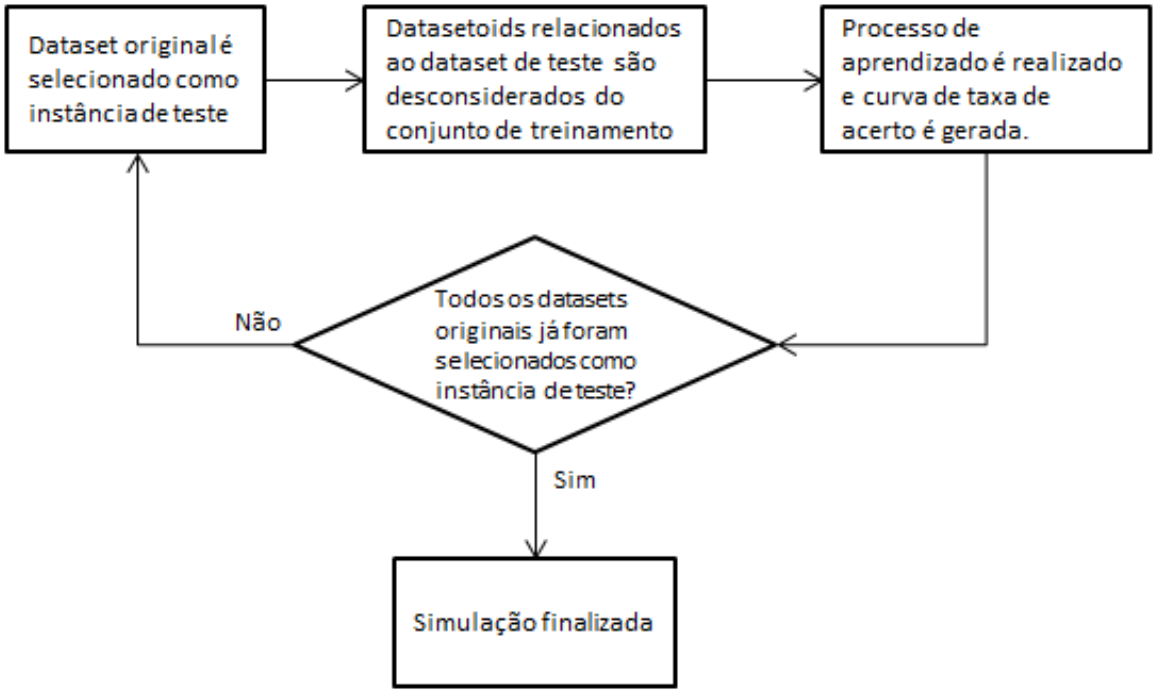

Figura 5.1: Fluxo de execução do LOO e construção dos conjuntos de treinamento e teste do modelo.

5.1.2 CONFIGURAÇÃO E *BASELINE* DO MODELO DE META-APRENDIZADO ATIVO

O desempenho obtido pelos meta-aprendizes foi sendo avaliado na medida em que o método de aprendizagem ativa ia progressivamente selecionando e rotulando meta-exemplos. Como base de comparação para o modelo com aprendizagem ativa, testamos o método de seleção aleatória para selecionar exemplos não rotulados. Apesar de sua simplicidade, o método de seleção aleatória tem a vantagem de realizar uma exploração totalmente uniforma do espaço de exemplos [12]. Consideramos dois diferentes cenários para o conjunto inicial de problemas rotulados e não rotulados. Figura 5.2 apresenta o diagrama de fluxo do algoritmo usado nos experimentos. No primeiro cenário, começamos com um conjunto vazio de meta-exemplos rotulados e em seguida selecionamos exemplos não rotulados de um conjunto contendo todas as bases de dados e *datasetoids*. Nesse cenário, o primeiro meta-exemplo é selecionado aleatoriamente enquanto os próximos são selecionados usando Aprendizagem Ativa. Sendo mais específico, é aplicado um critério de incerteza baseado em entropia. No segundo cenário, começamos a simulação com um conjunto de treinamento composto por meta-exemplos rotulados produzidos diretamente de todas as bases de dados e então usamos Aprendizagem Ativa para selecionar os exemplos não rotulados de um repositório composto apenas por *datasetoids*. Como pode ser visto, a diferença fundamental entre os dois cenários de simulação é que no segundo a simulação é iniciada com todos os conjuntos de dados originais. A motivação por trás desse cenário é a investigação do grau de influência dos conjuntos de dados originais sobre o processo de indução do modelo.

Também consideramos um cenário que possui a mesma configuração que o segundo, mas incluindo o procedimento de detecção de *outliers*. Nesse contexto, 10% dos problemas não rotulados são definidos como *outliers* e removidos do processo de seleção. A técnica de detecção de *outliers* foi desenvolvida tanto para a seleção aleatória quanto para a seleção ativa.

Em todos os cenários, permitimos que o modelo de Meta-Aprendizado Ativo selecione e rotule até 300 meta-exemplos de treinamento (critério de parada, cerca de 1/3 dos problemas candidatos disponíveis), número escolhido empiricamente após análise com objetivo de detectar a quantidade ideal de meta-exemplos para melhor observação da evolução dos modelos de Meta-Aprendizado. Repetimos os

experimentos 100 vezes para reduzir a influência do comportamento probabilístico da execução do algoritmo. Finalmente, como referência de limite superior de desempenho, avaliamos o meta-aprendiz usando os meta-exemplos produzidos por todos os conjuntos de dados originais (sem Aprendizagem Ativa). Avaliamos também usando os meta-exemplos de todos os conjuntos de dados (com bases originais e *datasetoids*). A motivação por trás dessa avaliação de limite superior de desempenho é poder analisar o ganho que existe em usar um modelo de aprendizagem que selecione os exemplos de treinamento ativamente ao invés de usar o repositório inteiro. Sendo assim, o fato do modelo de aprendizagem ativa atingir a mesma taxa de acerto que o limite superior pode ser vista como indicador de que a técnica consegue realizar um treinamento com a mesma qualidade que uma indução feita com todos os conjuntos de dados, mas usando um subconjunto menor do todo, reduzindo o custo computacional.

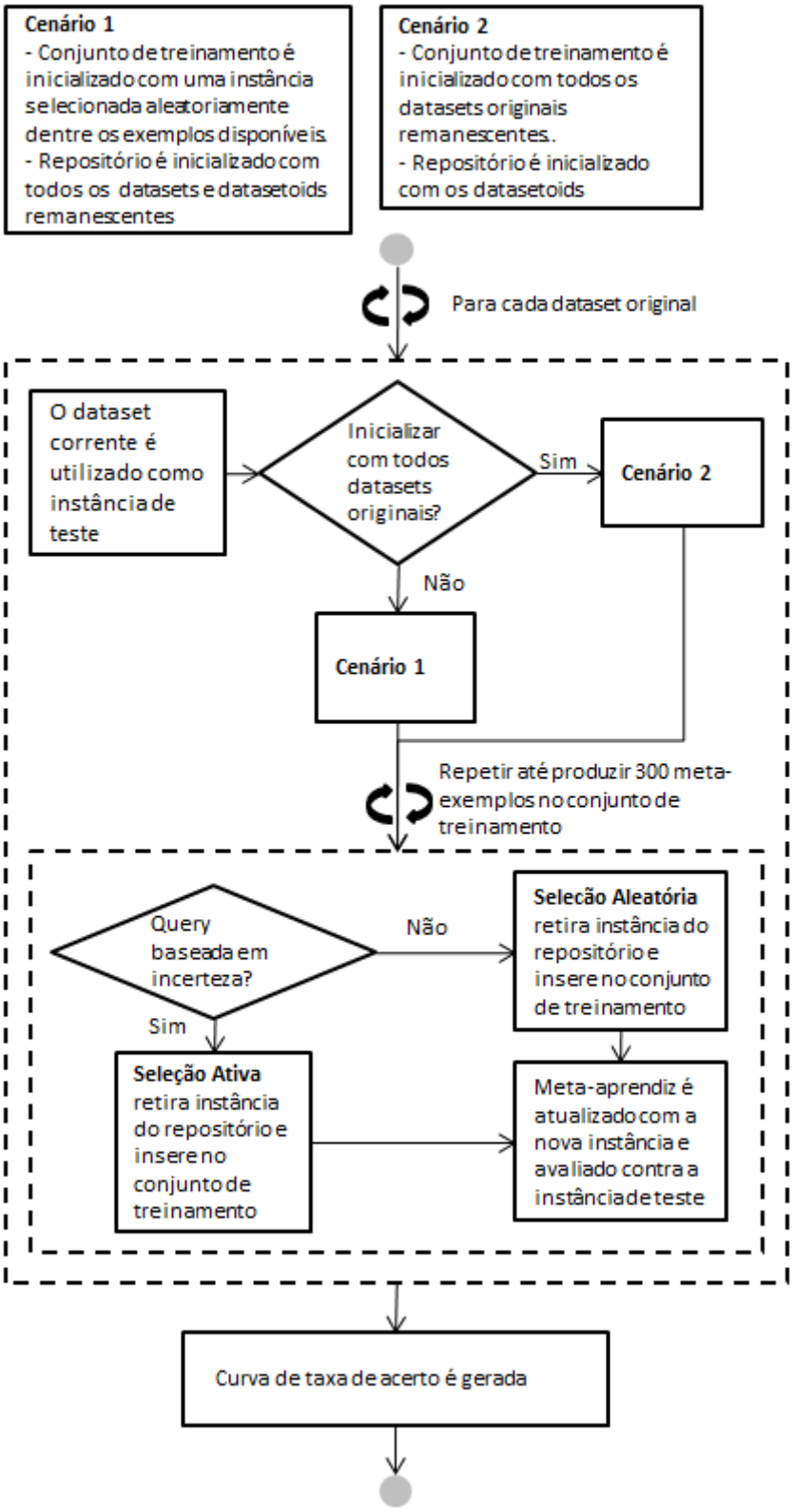

Figura 5.2: Diagrama de fluxo do algoritmo de Meta-Aprendizado Ativo adaptado aos experimentos.

5.2 RESULTADOS

Os cenários apresentados nesta seção têm como objetivo avaliar empiricamente o modelo proposto. Todos os cenários foram investigados no contexto do k-NN e Random forest como meta-aprendizes. Os gráficos apresentados nesta seção apresentam a taxa de acerto do modelo de Meta-Aprendizado na medida em que a base de meta-exemplos cresce com a inserção de novos elementos selecionados pela aprendizagem ativa. O eixo X apresenta a quantidade de exemplos de treinamento na base de meta-exemplos e o eixo Y apresenta a taxa de acerto.

Em todos os gráficos de taxa de acerto do k-NN, a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* é de 58,06% e a taxa de acerto usando somente os conjuntos de dados é de 52,21%. Já no Random forest, a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* é de 67,19% e a taxa de acerto usando somente os conjuntos de dados é de 54,84%.

Os gráficos são acompanhados por uma verificação estatística obtida através da realização de um teste de wilcoxon. Para cada ponto do eixo X (que indica a quantidade de exemplos de treinamento na base de dados) é verificada a taxa de acerto do meta-aprendiz e o teste de wilcoxon é utilizado para verificar se nesse determinado ponto da curva existe uma diferença estatística entre os gráficos exibidos (verificado através da rejeição da hipótese nula). O objetivo desse teste é verificar a quantidade de pontos em que um gráfico é melhor que o outro.

5.2.1 CENÁRIO 01 – *RANDOM SAMPLING* E *UNCERTAINTY SAMPLING* INICIALIZADOS COM CONJUNTO DE TREINAMENTO VAZIO

Figura 5.3 e Figura 5.4 apresentam as curvas médias de desempenho obtidas pelo k-NN e Random forest, respectivamente, usando *uncertainty sampling* e *random sampling* em um conjunto de treinamento incialmente vazio. Em ambos os casos, as curvas apresentaram uma tendência positiva, ou seja, a taxa de acerto do meta-aprendiz cresce enquanto o número de meta-exemplos rotulados aumenta. Contudo, a curva de Meta-Aprendizado Ativo está claramente acima da curva relacionada ao método de seleção aleatória. Isso significa que o método de Aprendizagem Ativa é capaz de identificar exemplos que são mais informativos para o meta-aprendiz.

O *uncertainty sampling* alcançou resultados melhores que o *random sampling*. No caso do k-NN, após a seleção de 300 instâncias, foi obtida uma taxa de acerto de 54,36% usando *random sampling*, contra 58,06% usando *uncertainty sampling*. Estatisticamente, foi verificado que o *uncertainty sampling* é significativamente melhor que o *random sampling* em 34% da curva (102 pontos). Já no caso do Random forest, o algoritmo obteve uma taxa de acerto de 63,58% após a seleção de 300 conjuntos de dados usando *random sampling*, contra 66,89% obtido pelo *uncertainty sampling*. Estatisticamente, a vantagem significativa do *uncertainty sampling* ocorre em 22% (66 pontos) da curva. Selecionando apenas 197 meta-exemplos, o método de *uncertainty sampling* obteve o mesmo nível de acurácia obtido pelo *random sampling* após a seleção de 300 meta-exemplos. Esses resultados demonstram uma melhoria na taxa de acerto obtida pelo modelo de Meta-aprendizado quando o Random forest é utilizado como meta-aprendiz ao invés do k-NN.

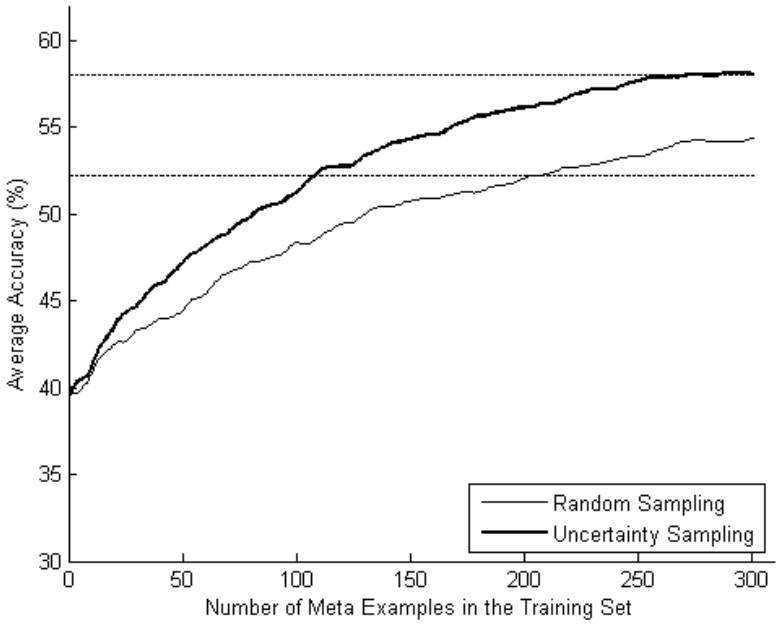

Figura 5.3: Cenário 01: Taxa de acerto média obtida pelo k-NN usando *uncertainty* e *random sampling*. Algoritmo inicializado com um conjunto de treinamento vazio, selecionando problemas não rotulados de um repositório composto por todas as bases de dados e *datasetoids*. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

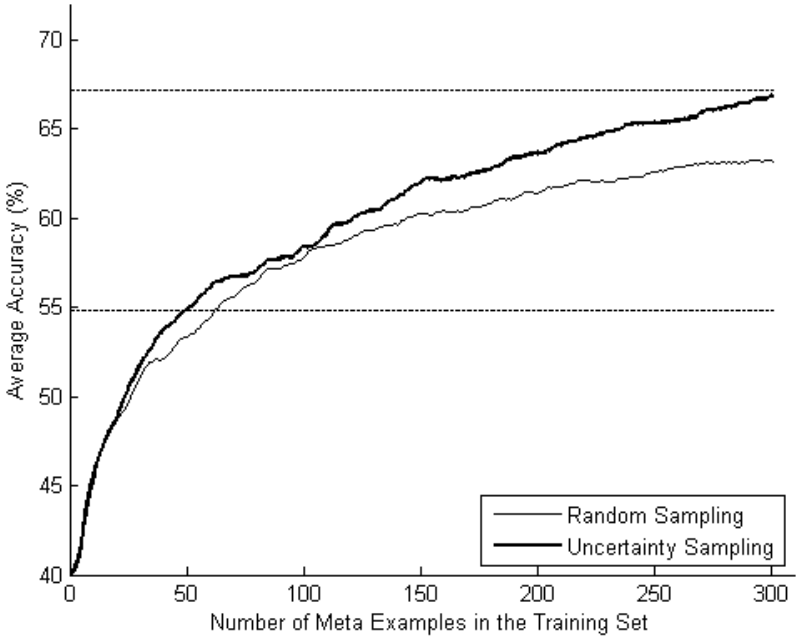

Figura 5.4: Cenário 01: Taxa de acerto média obtida pelo Random forest usando *uncertainty* e *random sampling*. Algoritmo inicializado com um conjunto de treinamento vazio, selecionando problemas não rotulados de um repositório composto por todas as bases de dados e *datasetoids*. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

5.2.2 CENÁRIO 02 – *RANDOM SAMPLING* E *UNCERTAINTY SAMPLING* INICIALIZADOS COM OS CONJUNTOS DE DADOS ORIGINAIS

Figura 5.5 e Figura 5.6 apresentam as curvas de acerto obtidas pelo k-NN e Random forest com as bases de treinamento inicializadas com todos os conjuntos de dados originais. Como pode ser visto, o *uncertainty sampling* permaneceu melhor do que a técnica de *random sampling* durante as curvas de acerto, apesar de no caso do Random forest os resultados obtidos terem sido relativamente semelhantes em alguns pontos da curva. O melhor resultado obtido pelo Random forest usando *random sampling* foi 67,19% no limite de 300 meta-exemplos, contra 67,89% obtidos pelo *uncertainty sampling*. A mesma taxa de acerto obtida pelo *random sampling* foi alcançada pelo *uncertainty sampling* usando 209 meta-exemplos rotulados, o que representa menos de 25% do total de meta-exemplos disponíveis. Os testes estatísticos apontaram uma melhora significativa do *uncertainty sampling* em relação ao *random sampling* em 16% (38 pontos) da curva. Também é importante notar que o resultado obtido pelo *uncertainty sampling* foi ligeiramente melhor que a taxa de acerto

baseline (onde todos os conjuntos de dados e *datasetoids* foram considerados na base de treinamento).

Os resultados positivos obtidos pelo Random forest se mostraram superiores aos resultados obtidos pelo k-NN. Neste último, a taxa de acerto foi de 56,67% com *random sampling* e 59,46% com *uncertainty sampling*. Testes estatísticos indicam que o *uncertainty sampling* é significativamente melhor do que o *random sampling* em 41% (98 pontos) da curva. Em [15], os resultados foram ainda mais diferentes: a curva de acerto obtida pelo k-NN com *uncertainty sampling* foi praticamente flat, sem ganho significativo quando novos meta-exemplos foram selecionados. A melhor taxa de acerto obtida usando o *random sampling* em [15] foi 58% com 300 meta-exemplos, valor superior a melhor taxa de certo obtida pelo k-NN com *uncertainty sampling*. Entretanto, esse comportamento pode ser justificado pela diferença de parametrização dos algoritmos na simulação. De toda forma, a seleção de parâmetros ainda será investigada com maior profundidade em trabalhos futuros.

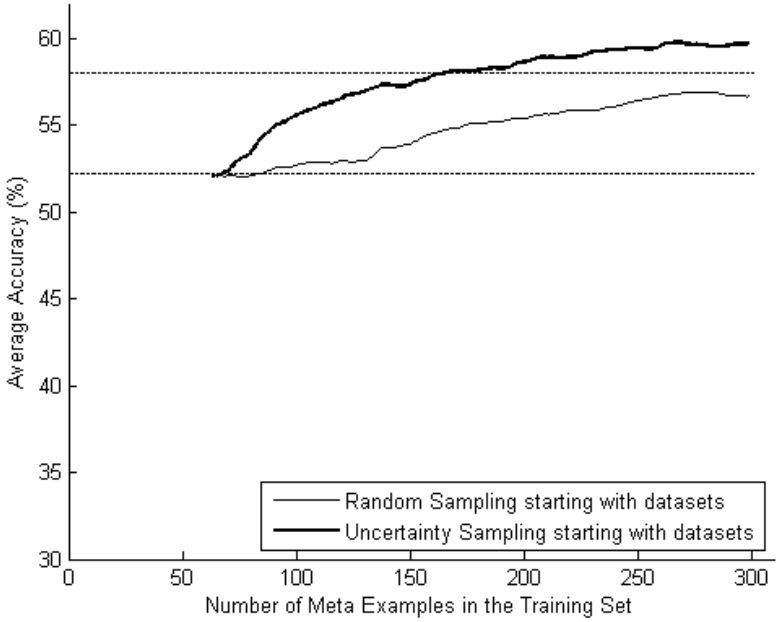

Figura 5.5: Cenário 02: Taxa de acerto média obtida pelo k-NN usando *uncertainty* e *random sampling*. Algoritmo inicializado com um conjunto de treinamento contendo todos os 64 conjuntos de dados originais, restando apenas *datasetoids* para seleção ativa. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

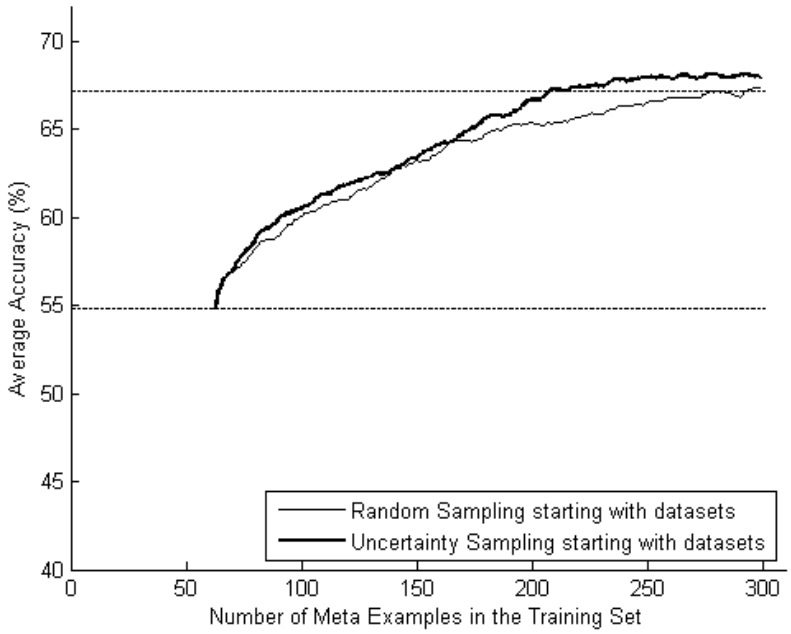

Figura 5.6: Cenário 02: Taxa de acerto média obtida pelo Random forest usando *uncertainty* e *random sampling*. Algoritmo inicializado com um conjunto de treinamento contendo todos os 64 conjuntos de dados originais, restando apenas *datasetoids* para seleção ativa. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

Em ambos os cenários, a taxa de classificação obtida pelo Random forest quando comparada com o k-NN como meta-aprendiz representa um ganho de desempenho no modelo de Meta-Aprendizado. A Tabela 5.2 sumariza as taxas de acerto obtidas pelo *random sampling* e *uncertainty sampling*, tanto pelo k-NN quanto pelo Random forest.

	k-NN		Random forest	
	Random	Uncertainty	Random	Uncertainty
Cenário 01	54,36%	58,06%	63,58%	66,89%
Cenário 02	56,67%	59,46%	67,19%	67,89%

Tabela 5.2: Taxa de acerto média (%) obtida pelos algoritmos k-NN e Random forest após seleção de 300 meta-exemplos

Figura 5.7 e Figura 5.8 apresentam os resultados obtidos pelo k-NN e Random forest nos dois cenários de experimentos, com o objetivo de demonstrar o quanto um

conjunto de treinamento inicializado com as bases de dados originais melhora a taxa de acerto do meta-aprendiz.

No caso do k-NN, o *uncertainty sampling* apresentou uma vantagem estatística significativa em 21% (49 pontos) em relação ao *random sampling*. Já no Random forest, no início da simulação as instâncias mais informativas selecionadas pela Aprendizagem Ativa com um conjunto de dados vazio (cenário 1) foi capaz de produzir um conjunto de treinamento capaz de alcançar um desempenho melhor do que a Aprendizagem Ativa com o conjunto de treinamento inicializado com todos os 64 conjuntos de dados (cenário 2). Esse comportamento persiste até a o 69º meta-exemplo ser selecionado pela Aprendizagem Ativa. Uma explicação cabível para esta situação é que em condições de *underfitting* as instâncias mais informativas possam ser outras que não as bases de dados originais. Entretanto, de médio a longo prazo, os conjuntos de dados originais ajustam o classificador para uma taxa de acerto melhor, contribuindo para seleção posterior da Aprendizagem Ativa. Este assunto será investigado mais profundamente em trabalhos futuros. De toda forma, testes estatísticos apontam uma vantagem significativa do *uncertainty* sobre o *random sampling* em 59% (141 pontos) da curva.

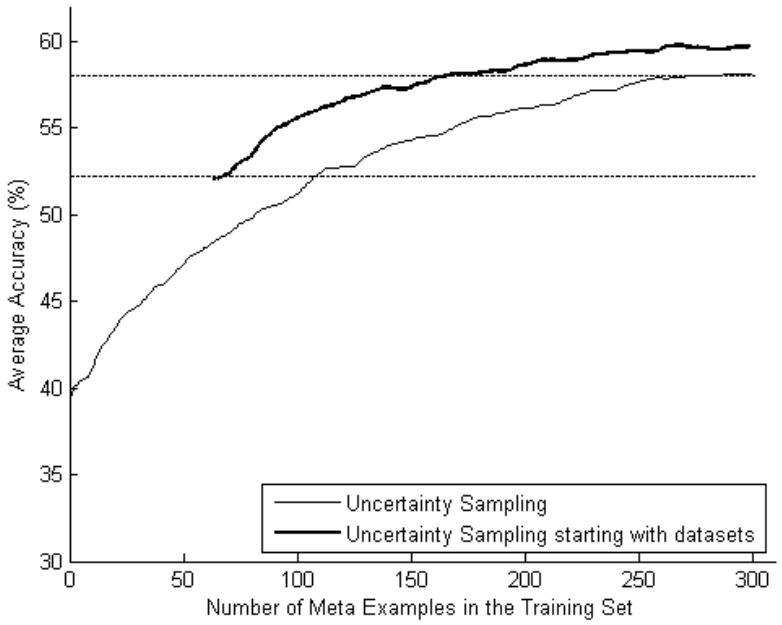

Figura 5.7: Cenário 02: Taxa de acerto média obtida pelo k-NN usando *uncertainty sampling*. Algoritmo inicializado com um conjunto de treinamento vazio e um conjunto de treinamento com todos os 64 conjuntos de dados originais, respectivamente. As linhas pontilhadas representam a taxa de acerto obtida

usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

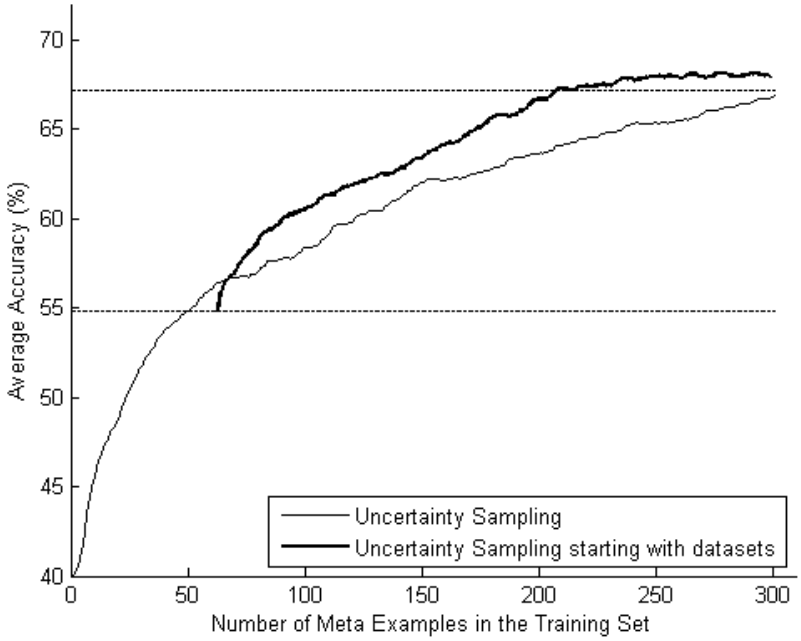

Figura 5.8: Cenário 02: Taxa de acerto média obtida pelo Random forest usando *uncertainty sampling*. Algoritmo inicializado com um conjunto de treinamento vazio e um conjunto de treinamento com todos os 64 conjuntos de dados originais, respectivamente. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

5.2.3 CENÁRIO 03 – *RANDOM SAMPLING* E *UNCERTAINTY SAMPLING* COM DETECÇÃO DE *OUTLIERS*

Figura 5.9 e Figura 5.10 apresentam as curvas de desempenho obtidas pelo k-NN usando uma técnica de detecção de *outliers* e seleção de exemplos com *random sampling* e *uncertainty sampling*, respectivamente. Figura 5.11 e Figura 5.12 apresenta o resultado dos mesmos cenários usando o Random forest como meta-aprendiz. Como pode ser visto, resultados melhores são quando a detecção de *outliers* é aplicada. No caso do Random forest, os resultados apresentam algumas similaridades em alguns pontos da curva.

Na Figura 5.9, o k-NN obteve 59,28% de taxa de acerto usando o *random sampling* com detecção de *outliers*, contra 56,67% sem a aplicação da técnica mencionada. Estatisticamente, nesse caso a detecção de *outliers* apresenta ganho

significativo em 57% (136 pontos) da curva. Na Figura 5.10, o k-NN apresentou uma acurácia de 61,38% aplicando uma técnica de detecção de *outliers* em conjunto com a seleção ativa usando *uncertainty sampling*. É um bom resultado, considerando que a execução sem a remoção de ruído gerou um acerto de 59,46%. Nesse caso, a vantagem estatística da detecção de *outliers* é significativa em 24% (59 pontos) da curva. Já no caso do Random forest, o resultado obtido (Figura 5.11) com *random sampling* e inicializado com todos os conjuntos de dados foi 67,87% quando combinado com a técnica de detecção de *outliers*, contra 67,19% obtido sem a combinação. Estatisticamente, a vantagem estatística da detecção de *outliers* ocorreu de forma significativa em 26% (61 pontos) da curva. A Figura 5.12 mostra que as curvas do Random forest apresentaram uma tendência de crescimento usando *uncertainty sampling*. Contudo, o Meta-Aprendizado Ativo aliado ao procedimento de detecção de *outliers* está acima da curva representada pelo mesmo método sem a detecção de *outliers*. O desempenho obtido pelo *uncertainty sampling* com detecção de *outliers* foi 70,76%, contra 67,89% obtido sem o método de remoção de ruído. Esse comportamento indica que a técnica combinada consegue eliminar algum ruído e melhorar a qualidade dos meta-dados utilizados pelo meta-aprendiz. Estatisticamente, nesse último caso a vantagem estatística da detecção de *outliers* é significativa em 27% (64 pontos) da curva.

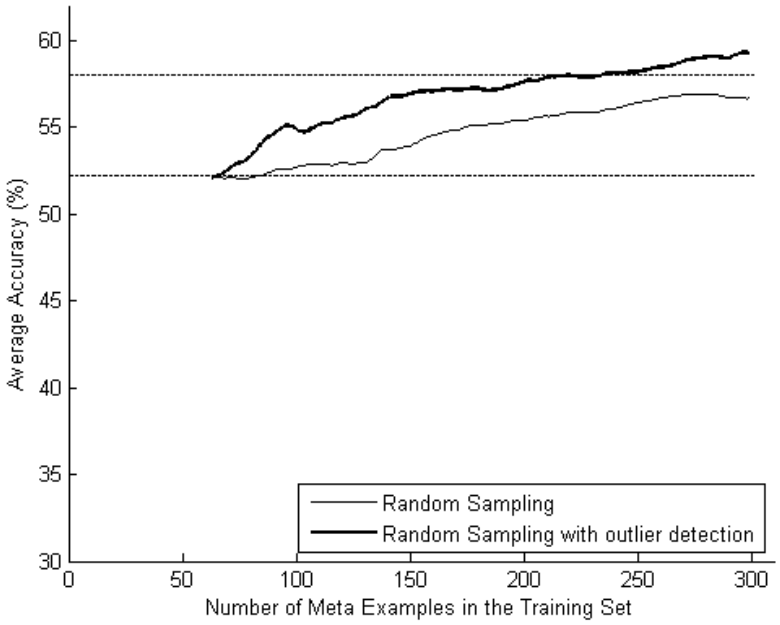

Figura 5.9: Cenário 03: Taxa de acerto média obtida pelo k-NN usando *random sampling*. Algoritmo inicializado com um conjunto de treinamento contendo todos

os 64 conjuntos de dados originais, com e sem detecção de *outliers*, respectivamente. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

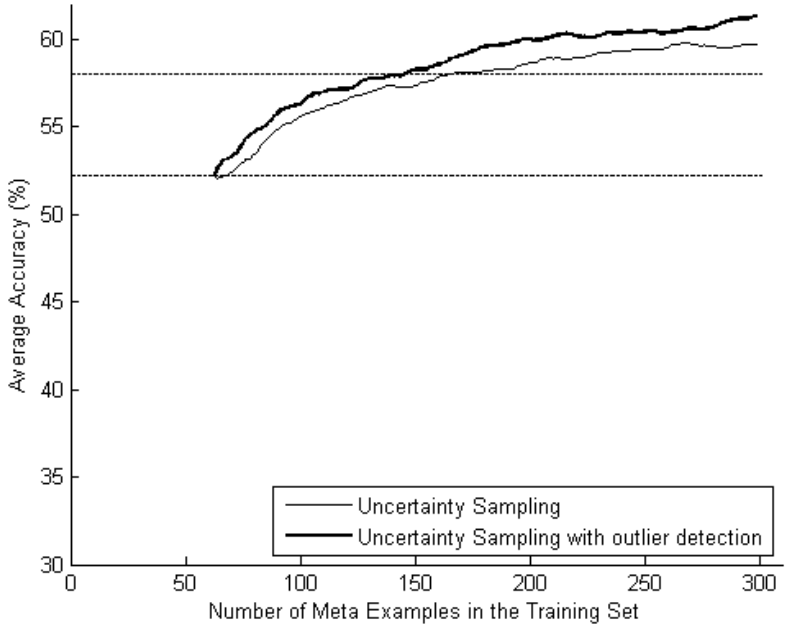

Figura 5.10: Cenário 03: Taxa de acerto média obtida pelo k-NN usando *uncertainty sampling*. Algoritmo inicializado com um conjunto de treinamento contendo todos os 64 conjuntos de dados originais, com e sem detecção de *outliers*, respectivamente. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

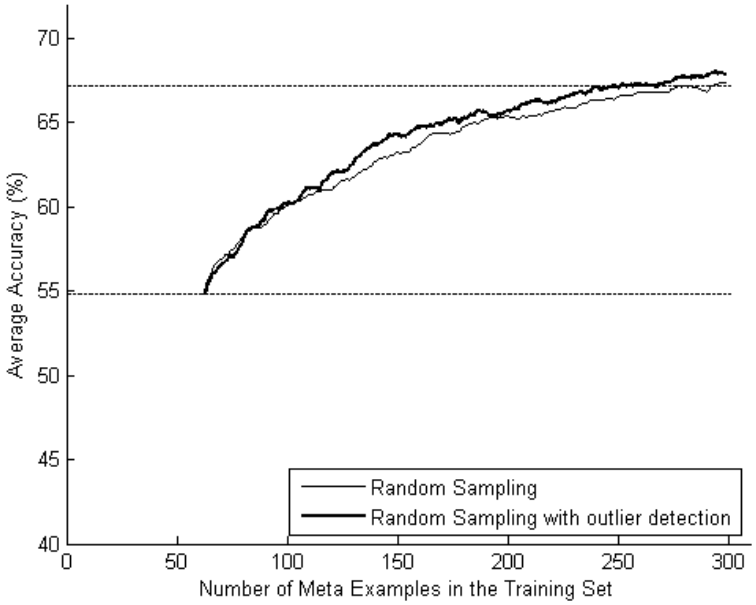

Figura 5.11: Cenário 03: Taxa de acerto média obtida pelo Random forest usando *random sampling*. Algoritmo inicializado com um conjunto de treinamento contendo todos os 64 conjuntos de dados originais, com e sem detecção de *outliers*, respectivamente. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e *datasetoids* (topo) ou somente os conjuntos de dados (base).

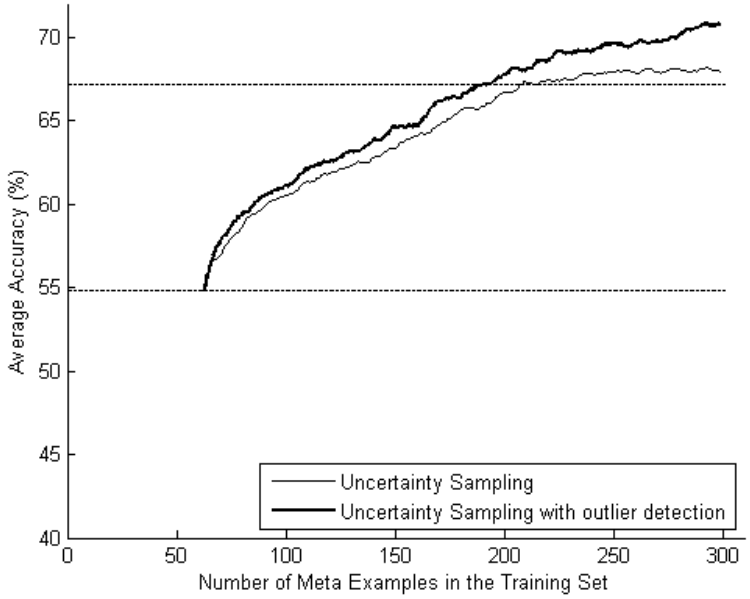

Figura 5.12: Cenário 03: Taxa de acerto média obtida pelo Random forest usando uncertainty sampling. Algoritmo inicializado com um conjunto de treinamento contendo todos os 64 conjuntos de dados originais, com e sem detecção de *outliers*, respectivamente. As linhas pontilhadas representam a taxa de acerto obtida usando todos os conjuntos de dados e datasetoids (topo) ou somente os conjuntos de dados (base).

5.3 CONSIDERAÇÕES FINAIS

Os cenários apresentados neste capítulo avaliaram o modelo de Meta-Aprendizado usando o k-NN e Random forest como meta-aprendizes. Os resultados demonstraram que o Random forest apresenta um desempenho superior ao do k-NN. As curvas médias de desempenho também demonstraram que o método de Aprendizagem Ativa é capaz de identificar exemplos que são mais informativos para o meta-aprendiz, permitindo que seja obtida a mesma taxa de acerto que o limite superior (conjunto de treinamento contendo todas as bases originais e *datasetoids*) usando menos de 30% da base. Além disso, foi observado que a escolha do conjunto de dados

inicial pode influenciar na qualidade da indução do modelo. Os resultados obtidos indicaram também que a técnica de detecção de *outliers* é capaz de eliminar elementos indesejáveis, deixando a base de dados mais consistente e informativa, aumentando assim a qualidade do modelo de Meta-Aprendizado e sua taxa de acerto.

6 CONCLUSÃO

Esta dissertação teve como objetivo melhorar o desempenho dos modelos de Meta-Aprendizado e reduzir o custo computacional relacionado à construção da base de meta-exemplos no contexto da tarefa de seleção de algoritmos. Ao longo dos capítulos, foram analisados em detalhes os problemas de Meta-Aprendizado que se referem principalmente à baixa disponibilidade de instâncias para construir uma base consistente de meta-exemplos e o alto custo computacional para classificar um meta-exemplo, assim como possíveis estratégias para resolver esse problema.

Criamos um modelo de Meta-Aprendizado no qual utilizamos os algoritmos k-NN e Random forest como meta-aprendizes combinados a uma abordagem de Aprendizagem Ativa com estratégia de *query* usando um critério de incerteza baseado em entropia. Além disso, o modelo proposto foi combinado com uma técnica simples e relativamente recente de manipulação de dados chamada *datasetoids*. Como foi apresentado, essa combinação obteve ganho no desempenho dos modelos de Meta-Aprendizado, amenizando de forma bastante eficaz as limitações mencionadas na abordagem quando aplicada à tarefa de seleção de algoritmos.

6.1 CONTRIBUIÇÕES REALIZADAS

A abordagem proposta conseguiu aumentar de forma expressiva a quantidade de conjuntos de dados para a geração de meta-exemplos e ainda diminuiu o custo computacional para a coleta de meta-dados. Nossos resultados mostraram que o algoritmo Random forest pode obter resultados muito satisfatórios como meta-aprendiz. Além disso, o conjunto inicial de treinamento pode impactar as seleções subsequentes de meta-exemplos realizadas pelo método de *uncertainty sampling*.

Sistematicamente, as principais contribuições do trabalho desenvolvido foram:

- Revisão de técnicas de Aprendizagem Ativa e geração de dados sintéticos e manipulados.
- Revisão do estado da arte sobre o problema de seleção de algoritmos no contexto de Meta-Aprendizado.

- Reprodução dos experimentos de [15] usando uma versão do algoritmo k-NN mais estável e com parâmetros melhores ajustados.
- Implementação do algoritmo Random forest como meta-aprendiz combinado a uma abordagem de aprendizagem ativa para realizar a seleção de meta-exemplos.
- Aplicação de técnica para manipular dados existentes e criar novas entradas na base de dados a fim de aumentar a disponibilidade de meta-dados.
- Implementação de uma técnica de detecção de *outliers* para eliminar ruído e aumentar a qualidade do conjunto de dados utilizado pelo modelo.
- Levantamento de novos pontos a serem aprofundados.

Além dos tópicos acima, um artigo foi publicado, validando a pesquisa realizada:

- SOUSA, A., PRUDENCIO, R., SOARES, C, LUDERMIR, T.: Active Selection of Training Instances for a Random Forest Meta-Learner. International Joint Conference on Neural Networks. Proceedings, 2013.

6.2 LIMITAÇÕES E TRABALHOS FUTUROS

Apesar dos resultados positivos desta dissertação, existem muitos pontos que podem ser pesquisados mais a fundo. Os resultados foram obtidos usando um método padrão de Aprendizagem Ativa na literatura que não é robusto a *outliers*. Além disso, o a tarefa de Meta-Aprendizado estudada é conceitualmente muito simples e talvez inadequada para avaliar bem o modelo proposto.

Abaixo, uma lista de pontos que podem se tornar trabalhos futuros:

- Adaptar o modelo proposto para utilizá-lo em outros domínios de aplicação, como tarefas de regressão, otimização meta-heurística e aprendizado supervisionado.
- Investigar uma seleção da melhor configuração de parâmetros para os meta-aprendizes k-NN e Random forest.

- Aprimorar a combinação do modelo de Meta-Aprendizado Ativo, implementando métodos de mais complexos de Aprendizagem Ativa e menos sensíveis a *outliers*.
- Validar o comportamento do modelo sobre uma tarefa mais complexa de Meta-Aprendizado, envolvendo diferentes problemas e uma quantidade maior de meta-atributos.
- Investigar outros algoritmos de aprendizagem para atuarem como meta-aprendizes.
- Implementar estratégias mais robustas e menos ruidosas para aumentar a disponibilidade de meta-dados e gerar novas entradas no respositório.

Além dos pontos levantados, os experimentos realizados ainda podem ser avaliados mais a fundo. No início da simulação do algoritmo Random forest com *uncertainty sampling* e inicializado com todos os conjuntos de dados originais (Figura 5.8), a versão sem inicialização dos conjuntos originais apresentou um desempenho inesperadamente melhor no início da simulação. Apesar de termos sugerido uma razão para o acontecimento, cabe uma análise mais profunda para entender esse comportamento com mais detalhes.

REFERÊNCIAS

[1] Assuncion, A., Newmanm D.: UCI machine learning repository (2007).

[2] Reif, M.: Dataset Generation for Meta-Learning Poster and Demo Track of the 35th German Conference on Artificial Intelligence (KI-2012), pp. 69-73, 2012.

[3] Blockeel, H., Vanschoren, J.: Experiment databases: Towards an Improved Experimental Methodology in Machine Learning. LNCS 4702, pp. 6-17 (2007).

[4] Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Cognitive Technologies, Springer (2009).

[5] Brazdil, P., Soares, C., da Costa, J: Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. Machine Learning 50(3), pp. 251-277 (2003).

[6] Giraud-Carrier, C., Vilalta, R., Brazdil, P.: Introduction to the Special Issue on Meta-Learning. Machine Learning 54, pp. 187-193 (2004).

[7] Cohn, D., Atlas, L., Ladner, R.: Improving Generalization with Active Learning. Machine Learning 15, mpp. 201-221 (1994).

[8] Breiman, Leo.: Random Forests. Machine Leargning 45.1, pp. 5-32 (2001).

[9] Caruana, R., Karampatziakis, N., Yessenalina, A.: An Empirical Evaluation of Supervisioned Learning in High Dimensions. Proceedings of the 25th International Conference on Machine Learning, pp. 96-103 (2008).

[10] Liaw, A., Wiener, M.: Classification and Regression by Random Forest. R news 2(3), pp. 18-22 (2002).

[11] Hilario, M., Kalousis, A.: Quantifying the resilience of inductive classification algorithms. In: Zighed, D.A., Komorowski, J., ˙Zytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 106–115. Springer, Heidelberg (2000).

[12] Lindenbaum, M., Markovitch, S., Rusakov, D.: Selective Sampling for Nearest Neighbor Classifiers. Machine Learning 54, pp. 125-152 (2004).

[13] Macià, N., Orriols-Puig, A., Bernadó-Mansilla, E.: Genetic-based Synthetic Data Sets for the Analysis of Classifiers Behavior. In: Proceedings of 15th International Conference on Hybrid Intelligent Systems, pp. 507-512 (2008).

[14] Prudêncio, R.B.C., Ludermir, T.B.: Selective Generation of Training Examples in Active Meta-Learning. Intern. Journal of Hybrid Intelligent Systems 5, pp. 59-70 (2008).

[15] Prudêncio, R.B.C., Soares, C., Ludermir, T.B.: Uncertainty Sampling Methods for Selecting Datasets in Active Meta-Learning, Proceedings of International joint Conference on Neural Networks, pp. 1082-1089 (2011).

[16] Raghavan, H., Madani, O., Jones, R.: Active Learning with feedback on both features and instances. Pattern Recognition Letters 7, pp. 1655-1686 (2006).

[17] Riccardi, G., Hakkani-Tur, D.: Active Learning - Theory and Applications to Automatic Speech Recognition. IEEE Transactions on Speech and Audio Processing 13(4), pp. 504-511 (2005).

[18] Sampaio, I., Ramalho, G., Corruble, V., Prudêncio, R.: Acquiring the Preferences of New Users in Recommender Systems - the role of item controversy. In: Proceedings of the ECAI 2006 Workshop on Recommender Systems, pp. 107-110 (2006).

[19] Prudêncio, R.B.C., Ludermir, T.B.: Meta-Learning Approaches for Selecting Time Series Models, Neurocomputing Journal 61, pp. 121-137 (2004).

[20] Smith-Miles, K.: Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection. ACM Computing Surveys 41(1), pp. 1-25 (2008).

[21] Soares, C.: Uci++, Improved Support for Algorithm Selection Using Datasetoids. Lecture Notes in Computer Science 5476, pp. 499-506 (2009).

[22] Kanda, J.Y.; de Carvalho, A.C.P.L.F.; Hruschka, E.R.; Soares, C.; Using Meta-Learning to Recommend Meta-heuristics for the Traveling Salesman Problem, 10th International Conference on Machine Learning and Applications and Workshops (ICMLA), vol.1, no., pp. 346--351 (2011).

[23] Settles, B.: Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison (2009).

[24] Namata, G., London, B., Getoor, L., Huang, B.: Query-driven Active Surveying for Collective Classification. 10th International Workshop on Mining and Learning with Graphs, (2012).

[25] Vilalta, R., Drissi, Y.: A Perspective View and Survey of Meta-Learning. Journal of Artificial Intelligence Review 18(2), pp. 77--95 (2002).

[26] Knorr, E., Ng, R.: A unified notion of outliers: Properties and computation. Proceedings of the KDD. (1997)

[27] Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. Proc. 18th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, pp. 441-448 (2001)

[28] Prudêncio, R.B.C., Ludermir, T.B.: Combining Uncertainty Sampling Methods for Supporting the Generation of Meta-Examples. Information Sciences, v. 196, pp. 1-14, 2012.

[29] Barr, R.S., Golden, B.L., Kelly, J. P., Resende, M. G., Stewart, W. R.: Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics, pp 9–32, 1995.

[30] Smith-Miles, K., Lopes, L.: Review: Measuring instance difficulty for combinatorial optimization problems. Computers and Operations Research, v. 39, n.5, p. 875-889, 2012

[31] Liang, L., Zhong, J., Liu, J., Li, P., Zhan, C., & Meng, Z.: An implementation of synthetic generation of wind data series. In Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES (pp. 1-6). IEEE, 2013

[32] Wolpert, D. H., Macready, W. G.: No free lunch theorems for optimization. IEEE, Transactions on Evolutionary Computation, pp. 67–82, 1997.

[33] J. R. Rice. The algorithm selection problem. Advances in Computers, 15:65–118, 1976.

[34] Thrun, S.: Lifelong Learning Algorithms. In S. Thrun and L. Pratt, editors, Learning to Learn, chapter 8, pp 181–209. Kluwer Academic Publishers, MA, 1998.

[35] Caruana, R.: Multitask Learning. Machine Learning, Second Special Issue on Inductive Transfer, 28(1):41–75, 1991.

[36] Brazdil, P., Soares, C., Costa, J.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning, v.50, pp. 251-257, 2003.

[37] Soares, C., Brazdil, P. B., Kuba, P.: A meta-learning method to select the kernel width in support vector regression, Machine Learning, v.54, n.3, p. 125-209, 2004.

[38] Michele, D., Spiegelhalter, D. J., Taylor, C. C.: Campbell, J.: Machine learning, neural and statistical classification. Ellis Horwood, 1994.

[39] Bensusan, H.; Giraud-Carrier, C.: Casa batlo is in passeig de Gracia or landmarking the expertise space. Proceedings of the ECML'2000 workshop on Meta-Learning: Building Automatic advice Strategies for Model Selection and Method Combination, pp. 109-113, 2000.

[40] Bensusanm H., Giraud-Carrier, C., Kennedy, C.: A higher-order approach to meta-learning. Proceedings of the ECML'2000 workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination, pp 109-117, 2000.

[41] Brodley, C.E.: Recursive automatic bias selection for classifier construction. Machine Learning, v.20, n.1-2, pp. 63-94.

[42] Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research, 2:45-66, 2002.

[43] Liu., Y.: Active Learning with support vector machine applied to gene expression data for cancer classification. Journal of Chemistry Information and Computer Science, 44:1963-1941, 2004.

[44] Zhang, C., Chen, T.: An active learning framework for content-based information retrieval. Multimedia, IEEE Transactions on, 4(2):260-268, 2002.

[45] Mccallum A., Nigam, K.: Employing em and pool-based active learning for text classification. ICML'98: Proceedings of the Fifteenth International Conference on Machine Learning, pp 350-358, 1998.

[46] Angluin, D.: Queries and concept learning. Machine Learning, 2(4):319-342, 1988.

[47] Angluin, D.: Queries revisited. In Proceedings of the International Conference on Algorithmic Learning Theory, pp 12-31, 2001.

[48] Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning, 15(2):201-221, 1994.

[49] Dagan, I.m Engelson, S.: Committee-based sampling for training probabilistic classifiers. In Proceedings of the International Conference on Machine Learning, pp 150-157, 1995

[50] Krishnamurthy, V.: Algorithms for optimal scheduling and management of hidden markov models sensors. IEEE Transactions on Signal Processing, 50(6):1382-1397, 2002.

[51] Yu, H.: SVM selective sampling for ranking with application to data retrieval. In Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD), pp 354-363, 2005.

[52] Lewis, D., Gale, W., A sequential algorithm for training text classifiers. In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, pp 3-12, 1994.

[53] Hoi, S.C.H., Jin, R., Lyu, M.R.: Large-scale text categorization by batch mode active learning. In Proceedings of the International Conference on the World Wibe Web, pp 663-342, 2006.

[54] Settles, B., Craven, M.: An analysis of active learning strategies for sequence labeling tasks. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1069-1078, 2008.

[55] Hauptmann, A., Lin, W., Yan, R., Yang, J., Chen, M.Y.: Extreme video retrieval: joint maximization of human and computer performance. In Proceedings of the ACM Workshop on Multimedia Image Retrieval, pp 385-394, 2006.

[56] Donmez, P., Carbonell, J.G., Bennet, P.N.: Dual strategy active learning. In ECML'07: Proceedings of the 18th European conference on Machine Learning, pp. 116-127, 2007.

[57] Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In ICML'04: Proceedings of the twenty-first international conference on Machine learning, pp. 79+, 2004.

[58] Khot, L. R., Panigrahi, S., Doetkott, C., Chang, Y., Glower, J., Amamcharla, J., Logue, C., Sherwood, J.: Evaluation of technique to overcome small dataset problems during neural-network based contamination classification of packaged beef using integrated olfactory sensor system. LWT-Food Science and Technology, 45(2), 233-240, 2012.

[59] Breiman L.: Bagging Predictors. Machine Learning 26, pp 123-140, 1996.