Universidade Federal de São Carlos
Centro de Ciências Exatas e Tecnologia

ECID lite: uma aplicação para previsão de explosões solares

Carlos Gomes de Carvalho Junior

São Carlos - SP
2022

Carlos Gomes de Carvalho Junior

ECID lite: uma aplicação para previsão de explosões solares

Trabalho de conclusão de curso apresentado ao Programa de Graduação em Engenharia de Computação, ao Departamento de Computação da Universidade Federal de São Carlos, como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação.

Orientadora: Profa. Dra. Marcela Xavier Ribeiro

São Carlos-SP
2022

Dedico este trabalho à minha família por todo apoio durante minha trajetória acadêmica.

AGRADECIMENTOS

Primeiramente agradeço a Deus por permitir que tudo acontecesse. Agradeço também aos meus pais e à minha irmã, por todos os ensinamentos e por estarem sempre presentes.

Agradeço aos colegas de curso, que tornaram a experiência da graduação mais divertida.

Agradeço à professora Marcela e a todo pessoal das explosões solares pelo acompanhamento e feedback ao longo do projeto.

“Dedication is a talent all on its own”
(Alphonse Elric)

RESUMO

Explosões solares são fenômenos que acontecem no Sol, que consistem em uma súbita liberação de uma grande quantidade de energia. Essa grande liberação de radiação solar causa distúrbios eletromagnéticos na terra, afetando comunicações, interferindo em frequências de rádio e transmissões de rede elétrica. Um método de previsão pode detectar com antecedência ou amenizar tais consequências. Neste projeto buscou-se aprimorar o método Ensemble of Classifiers for Imbalanced Datasets (ECID), tornando-o mais eficiente e flexível. Para isso, foi feito um estudo do método teórico e sua implementação, identificando pontos de melhoria. O método foi aprimorado a partir da proposição de uma nova arquitetura, refatoração e documentação da base de código. Os testes com a nova versão resultaram em uma redução de 22% no tempo de execução e menor complexidade de código, medida a partir da complexidade ciclomática antes e depois das alterações. Neste trabalho se criou uma API que disponibiliza os resultados do método, além do desenvolvimento de uma prova de conceito para o uso da API desenvolvida na forma da aplicação web ECID lite, que permite, de forma simples, a análise de um conjunto de previsões do ECID.

Palavras-chave: Previsão de Explosões Solares. Aprendizado de Máquina. Ensemble Of Classifiers For Imbalanced Datasets. Desenvolvimento de Software.

ABSTRACT

A solar flare is a phenomenon that occurs on the Sun, consisting of a sudden release of a colossal amount of energy. This large release of solar radiation causes electromagnetic disturbances on earth, affecting communications, interfering with radio frequencies, and power grid transmissions. Forecasting methods are a way to prevent or mitigate such consequences. In this project we sought to improve the Ensemble of Classifiers for Imbalanced Datasets (ECID) method, making it more efficient and flexible. To this end, a study of the theoretical method and its implementation was performed, identifying points of improvement. The method was improved by proposing a new architecture, refactoring and documenting the code base. The tests with the new version resulted in a 22% reduction in execution time and less code complexity, measured by the cyclomatic complexity before and after the changes. This work also created an API that makes the results of the method available. Furthermore, the web application ECID lite is developed as a proof of concept of the API usage, which allows a simple way to analyze a set of ECID predictions.

Keyword: Solar Flare Forecasting. Machine Learning. Ensemble Of Classifiers For Imbalanced Datasets. Software Development.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 - Introdução

Explosões solares têm um histórico de danos graves aos sistemas de comunicação que datam desde a década de 1850, com a erupção solar Carrington (CARRINGTON, 1859). Eventos tão recentes, como a explosão de julho de 2012, poderiam ter causado danos na escala dos bilhões de dólares se estivessem direcionados à Terra, além de levar meses a anos para serem reparados (LLOYD'S, 2013). Felizmente, no incidente de julho de 2012, o pior cenário foi evitado por cerca de uma semana (BAKER et al., 2013).

Um método confiável para previsão desse tipo de evento pode ser a diferença entre uma interrupção momentânea de serviços, e milhões de dólares gastos com reparo de infraestrutura tecnológica danificada. Um alerta seria suficiente para operadores de satélites executarem rotinas de emergência para evitar situações críticas; companhias aéreas poderiam cancelar voos ou alterar o trajeto, evitando regiões com comunicação prejudicada; operadores da rede elétrica poderiam fazer planos para minimizar o tempo que o serviço se encontra fora do ar (GREEN, ODENWALD, 2008). Esses são apenas alguns exemplos de situações que necessitam de um método de previsão confiável. Dito isso, estabelecemos que um método de previsão de explosões solares é uma maneira de amenizar seus efeitos negativos. Entretanto, a causa desses fenômenos não é precisamente definida (BORBA; COUVIDAT, 2015), aumentando a complexidade do problema da previsão.

Aplicando técnicas de mineração de dados em conjuntos de informações da atividade solar coletadas por satélite, como níveis de raios X, e fluxo magnético, já foi possível criar algoritmos capazes de prever a ocorrência de tais explosões em um horizonte de alguns dias. Entretanto, essa ainda é uma tarefa difícil, devido ao conjunto de dados ser bastante desbalanceado, e ser difícil distinguir com precisão a intensidade das explosões solares.

Assim, faz-se necessário que um método de previsão robusto e aplicável para esse problema trate essas dificuldades, ao passo que seja capaz de entregar resultados utilizando um horizonte de previsão que ofereça um tempo hábil para serem aplicadas medidas para prevenir danos. Outra característica que afeta a viabilidade do método de previsão é a acessibilidade de seus resultados: uma

plataforma de fácil consulta e utilização auxilia tanto pesquisadores buscando compreender melhor o fenômeno das explosões solares quanto outras partes interessadas na prevenção das suas consequências.

1.1 - Objetivos

1.1.1 - Geral

Este estudo usa como base um método de previsão de explosões solares chamado *Ensemble of Classifiers for Imbalanced Datasets* (ECID), proposto por (DÍSCOLA JR, 2019). Trata-se de uma técnica robusta que aborda algumas peculiaridades do domínio do problema, como o conjunto de dados desbalanceado, e o fato de que explosões solares de classes diferentes podem ser difíceis de distinguir.

O presente trabalho busca aprimorar a implementação original do autor, e criar uma API RESTful que torna os resultados do método disponíveis para acesso a pesquisadores e interessados. O trabalho também se propõe a criar uma aplicação web que consome a API como forma de prova de conceito da sua utilização.

1.1.2 - Específico

Os seguintes tópicos são objetivos específicos deste trabalho:

- Estudo sobre o método de previsão de explosões solares ECID, e o domínio do problema que ele busca resolver;
- Refatoração e extensão da base de código já existente do método ECID, de forma a torná-lo completamente automatizado e facilitar modificações futuras;
- Documentação da base de código, auxiliando futuros colaboradores a compreenderem e contribuírem com o projeto;
- Criação de uma *Application Programming Interface* (API) como forma de disponibilizar os resultados de uma instância treinada do método ECID;
- Desenvolvimento de uma aplicação web como prova de conceito da utilização da API desenvolvida.

1.2 - Organização do trabalho

O restante desta monografia está dividido em mais 4 capítulos: no Capítulo 2 são apresentados alguns conceitos importantes para compreensão do domínio da pesquisa e do trabalho desenvolvido. O Capítulo 3 detalha a metodologia utilizada para alcançar os objetivos propostos, sendo apresentadas a nova implementação desenvolvida para o método ECID, a API, e a aplicação web ECID lite. O Capítulo 4 apresenta uma comparação entre a versão original e a nova do método ECID, além de uma explicação sobre as funcionalidades da aplicação web. Por fim, o Capítulo 5 contém a conclusão, juntamente com uma análise crítica, e indicação para possíveis trabalhos futuros.

2 - Revisão teórica

2.1 - Explosões solares

De acordo com HOLMAN (2006), explosões solares são súbitas liberações de energia das regiões ativas da atmosfera solar. Tais fenômenos podem durar de segundos até algumas horas e, dependendo da intensidade, produzem consequências nas atividades na Terra, como interferência no funcionamento de equipamentos que utilizam ondas eletromagnéticas de alta frequência, sendo a comunicação via rádio e satélite alguns exemplos (TSURUTANI et al., 2009; BASU et al., 2010).

Uma instância desse fenômeno pode ser identificada a partir do monitoramento do fluxo solar de raios X no comprimento de onda entre 1 e 8 Angstroms (raios X1-8 Å), e é caracterizada pelo crescimento abrupto em um determinado período de tempo. A *Space Weather Prediction Center* (SWPC) estabelece esse período como 4 minutos (DÍSCOLA JR, 2019).

Salvo algumas exceções, uma explosão solar só acontece caso exista uma variação abrupta, ou seja, na maioria das vezes o aumento de forma gradual do fluxo de raios X1-8 Å não caracteriza uma explosão. O fluxo máximo emitido durante o evento é o que determina a classe da explosão solar. A Tabela 1 ilustra a classificação, mostrando também os possíveis danos associados a uma explosão de cada classe.

Tabela 1 - Classes de explosões solares.

Classe	**Fluxo de raios X 1-8 Å máximo observado – I (W/m^2)**	**Danos**
A	$I < 10^{-7}$	Nenhum
B	$10^{-7} <= I < 10^{-6}$	Nenhum
C	$10^{-6} <= I < 10^{-5}$	Possíveis efeitos em missões espaciais
M	$10^{-5} <= I < 10^{-4}$	Blecaute nas transmissões de rádio e possíveis danos à astronautas fora de espaçonaves

X	$I >= 10^{-4}$	Danos a satélites, sistemas de comunicação, estações de distribuição de energia e equipamentos eletrônicos

Fonte: Adaptado de (DÍSCOLA JR, 2019)

Além dessa classificação, cada classe pode ser expandida em subníveis indo de 1.0 até 9.9. Estes subníveis representam a intensidade de uma explosão solar dentro do intervalo da classe. Por exemplo, um evento com emissão de 3.0×10^{-7} W/m^2 seria classificado como B3.0 (DÍSCOLA JR, 2019). Essa característica faz com que a distinção de eventos pertencentes a classes adjacentes seja particularmente difícil, já que uma explosão de classe B9.9 tem intensidade semelhante a explosões de classe C1.0, apesar de pertencerem a classes diferentes.

Neste trabalho, explosões solares de classes A e B são consideradas pertencentes ao grupo AB, já que não causam danos. As demais classes pertencem ao grupo com o seu respectivo nome.

Apesar das explosões solares serem facilmente identificáveis utilizando o fluxo de raios X 1-8 Å, não se tem uma certeza sobre quais fatores são responsáveis pela sua ocorrência (BORBA; COUVIDAT, 2015). Isso torna a tarefa da sua previsão bastante desafiadora. Além disso, outra característica do domínio que torna esse problema especialmente difícil é o desbalanceamento dos dados. Explosões solares de intensidade baixa são muito mais abundantes quando comparadas às mais intensas. Se não for tratado de maneira adequada esse desbalanceamento pode afetar negativamente a previsão, tornando a solução enviesada, ou até incapaz de prever a ocorrência das classes mais raras.

2.2 - Aprendizado de máquina

Esta seção descreve conceitos relacionados ao tema de aprendizado de máquina que foram utilizados para o desenvolvimento do trabalho.

2.2.1 - Classificação com árvores de decisão

Classificadores são sistemas capazes de analisar dados de entrada, e a partir deles aplicar uma estratégia para relacionar esses dados a um rótulo categórico. Logo, a tarefa de classificação pode ser entendida como a busca de uma função, modelo ou hipótese responsável por essa relação (FACELI *et al*., 2011).

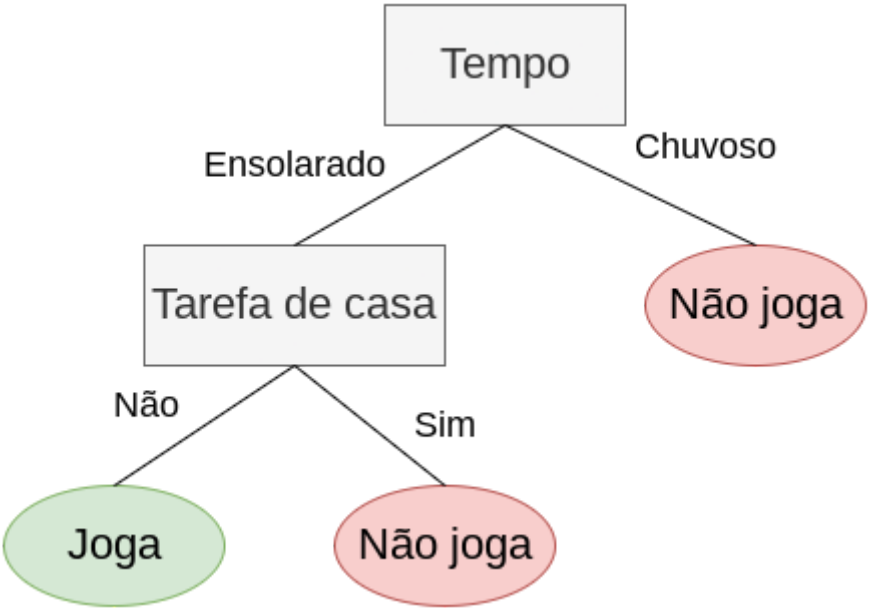
Para seu funcionamento, o método ECID utiliza um conjunto de classificadores binários, ou seja, especializados para problemas que possuem apenas dois rótulos categóricos. Na implementação do método esses classificadores binários são árvores de decisão.

Árvores de decisão são classificadores que utilizam uma estratégia de divisão e conquista para dividir todo o espaço de previsão em regiões mutuamente excludentes, possibilitando que algoritmos desse tipo façam previsões para qualquer exemplo de entrada (FACELI *et al.*, 2011).

Elas podem ser entendidas como um grafo acíclico direcionado, em que cada nó de divisão possui um teste condicional sobre um atributo, e cada nó folha representa uma região no espaço de atributos. Por se tratarem de algoritmos de aprendizado supervisionado, precisam ter acesso aos rótulos dos objetos durante o treinamento.

Na Figura 1 temos um exemplo de uma árvore de decisão simples para prever se um garoto irá jogar bola ou não. Nesse caso o espaço de previsão consiste nos dois resultados possíveis: “Joga” e “Não joga”, por isso trata-se de classificador binário. A árvore realiza testes condicionais nos atributos “Tempo” e “Tarefa de casa”, e a partir dos resultados toma uma decisão.

Figura 1 - Exemplo de árvore de decisão.



Fonte: Próprio autor

No exemplo da Figura 1, os atributos “Tempo” e “Tarefa de casa” têm valores categóricos, ou seja, valores dentro de um conjunto finito: sim/não para “Tarefa de casa”, e ensolarado/chuvoso para “Tempo”. Apesar disso, árvores de decisão também podem ser aplicadas em atributos contínuos, que assumem valores dentro dos números reais. Nessa situação, o teste condicional é alterado, e passa a ser, por exemplo, uma comparação se um valor excede ou não um limiar. Valores faltantes apresentam um problema para a árvore, já que não é possível tomar uma decisão sobre eles. Isso exige que eles sejam tratados de alguma forma, como substituição pelo valor mais comum ou por um valor válido aleatório.

Durante a construção de uma árvore de decisão deve existir um critério que irá definir quais serão os testes realizados nos nós de decisão, e por consequência, como serão as regiões de decisão. Uma abordagem possível é ter um critério que mede quão bem uma divisão discrimina as possíveis classes; dessa forma, no processo de construção da árvore o sistema “olha um passo à frente", avaliando os testes possíveis de acordo com esse critério, e escolhendo o teste que apresenta a melhor discriminação de classes. Essa é a estratégia adotada em algumas árvores de decisão populares, como o C4.5 (QINLAN, 1993) utilizado neste trabalho.

Um problema que deve ser tratado ao se trabalhar com árvores de decisão é a possibilidade de *overfitting*. Essa condição indica que o algoritmo se adaptou de

forma excessiva aos dados de entrada, o que pode causar uma perda de poder de classificação para outros dados (HAWKINS, 2004). Árvores de decisão são especialmente sensíveis a essa condição (IBM, 2021), uma vez que seu método de funcionamento se baseia em sucessivas divisões do espaço de decisão: sem uma condição de parada, o algoritmo pode fazer divisões extremamente específicas para se adequar ao conjunto de dados de entrada.

Como forma de contornar o problema de *overfitting,* surgem as estratégias de poda da árvore: uma prática que busca melhorar a capacidade de generalização para novos dados eliminando partes consideradas super ajustadas. Esse processo de eliminação pode acontecer enquanto a árvore é construída (pré-poda) ou ser aplicado em uma árvore já produzindo *overfitting* (pós-poda).

Para aumentar as chances de se obter resultados satisfatórios ao utilizar árvores de decisão, é importante considerar os problemas levantados nesta subseção.

2.2.2 - Classificação utilizando um conjunto de classificadores

Uma abordagem possível para qualquer problema de classificação no qual temos disponível um conjunto de classificadores é a combinação dos seus resultados. Essa estratégia é chamada de *ensemble*, e nessa configuração o conjunto de classificadores individuais é chamado de classificadores base. É preferível que os classificadores base utilizados realizem suas previsões de forma independente (CHANDRA; YAO, 2006).

A ideia básica dessa abordagem é reduzir a variância de um sistema de tomada de decisões, tornando-o mais robusto (ZHANG; MA, 2012). A combinação dos resultados de cada classificador pode ocorrer de duas maneiras: por seriação ou por votação. As duas subseções a seguir apresentam algumas definições de acordo com (FACELI *et al.*, 2011).

2.2.2.1 - Métodos de votação

A estratégia de votação é a mais comum, e pode ser utilizada quando cada um dos classificadores base tem como resultado um rótulo de classe (voto). Os votos, dentre outras maneiras, podem ser considerados de maneira uniforme, ou utilizando pesos ajustados ao longo do tempo como forma de recompensar um bom

classificador. O presente trabalho utiliza um método de votação adaptado que é explicado na Subseção 2.3.2.

2.2.2.2 - Métodos de seriação

A estratégia de seriação é aplicada quando cada classificador base tem como saída uma distribuição de probabilidade entre os possíveis valores da classe. A partir do conjunto de distribuições aplica-se uma regra que será usada para definir qual classe será escolhida como rótulo. Alguns exemplos de regras são:

- Soma das probabilidades de cada classe, escolhendo a classe com maior soma;
- Cálculo da média de probabilidade por classe, escolhendo a classe que teve maior média;
- Multiplicação das probabilidades por classe, escolhendo a classe com maior valor;

2.2.3 - Classificação na previsão de séries temporais

Podemos interpretar o problema da previsão de valores futuros de uma série temporal como um problema de classificação.

Considerando existência de um conjunto de observações $y_1, y_2, \ldots, y_t$, sendo y_i o valor do atributo-alvo Y no tempo i; e a tarefa preditiva como descobrir o valor do atributo Y no tempo futuro $t + h$; podemos assumir a existência de uma função f capaz de mapear valores passados ao valor futuro y_{t+h}, tal que $y_{t+h} = f(AtributosPassados)$, sendo $AtributosPassados$ informações obtidas em tempo igual ou anterior a t (OLIVEIRA; TORGO, 2015). Dessa forma, a tarefa de classificação se transforma em estimar tal função f.

Este trabalho utiliza diretamente essa ideia com a aplicação do algoritmo SS, detalhado na Subseção 2.3.1.

2.3 - Ensemble of Classifiers for Imbalanced Datasets - ECID

O método *Ensemble of Classifiers for Imbalanced Datasets* - ECID (DÍSCOLA JR, 2019), surge como alternativa para tratar dois principais problemas da

classificação de explosões solares: a natureza desbalanceada dos dados, e a dificuldade de definir com precisão a gravidade de uma explosão. Essa estratégia trata-se de uma extensão do método de previsão SeMiner, e ambos serão detalhados nesta seção.

2.3.1 - *Sequence Miner* - SeMiner

O Sequence Miner - SeMiner (DÍSCOLA JR. *et al*. 2018) trata-se de um método de previsão de explosões solares antecessor ao ECID. Ele aborda problemas diferentes na questão da previsão, focando em incorporar o conhecimento dos especialistas do domínio no modelo preditivo, e em considerar a evolução temporal dos parâmetros utilizados na previsão. Ele é aplicado como forma de processamento dos dados antes de serem encaminhados ao método ECID.

A ideia básica é elaborada na Subseção 2.2.3, e consiste em transformar um conjunto de dados presentes ou passados de forma que ele fique adequado para prever se ocorrerá alguma explosão solar de classe mais elevada em um momento futuro. O algoritmo permite, por exemplo, utilizar dados de dois dias atrás para prever se ocorrerá uma explosão solar de classe alta daqui a três dias. Assumindo que o atual dia da semana é quarta-feira, neste exemplo estaríamos usando dados de segunda e terça-feira para prever se ocorrerá uma explosão no sábado.

O funcionamento do SeMiner se baseia no algoritmo *Series to Sequence* (SS), que necessita da definição de três intervalos de tempo: *current window*, *jump* e *future window*. Esses intervalos podem ser configurados manualmente, e são cruciais para o bom funcionamento do algoritmo. Nessa etapa pode-se aplicar o conhecimento do especialista do domínio na seleção dos intervalos.

O primeiro intervalo é a *current window*, e ela contém os dados que serão utilizados no processo de previsão. No exemplo citado anteriormente, esse intervalo corresponde a segunda e terça-feira. O segundo intervalo é o *jump*, que representa o deslocamento de tempo entre as informações utilizadas para previsão e a data sobre a qual se deseja fazer a previsão. No exemplo isso corresponde a quarta, quinta e sexta-feira. Por fim, a *future window* representa o intervalo de tempo sobre o qual queremos prever se ocorrerá uma explosão ou não. No exemplo ele corresponde ao dia de sábado.

A entrada do algoritmo SS é uma série temporal contendo o valor de um ou mais atributos observados em cada instante *t*, e a classe da explosão solar (*Solar flare class*) observada no mesmo instante. A figura 2 exemplifica o funcionamento do SS em uma série temporal com informações de raios X e classe da explosão solar observada coletadas a cada 5 minutos. Os intervalos considerados são: 20 minutos para a *current window*, 20 minutos para *jump* e 20 minutos para a *future window*, formando uma janela (*window*) total de 1 hora. A cada iteração do algoritmo a janela é deslocada por um fator *step*, que também é configurável, e nesse caso é igual 5 minutos.

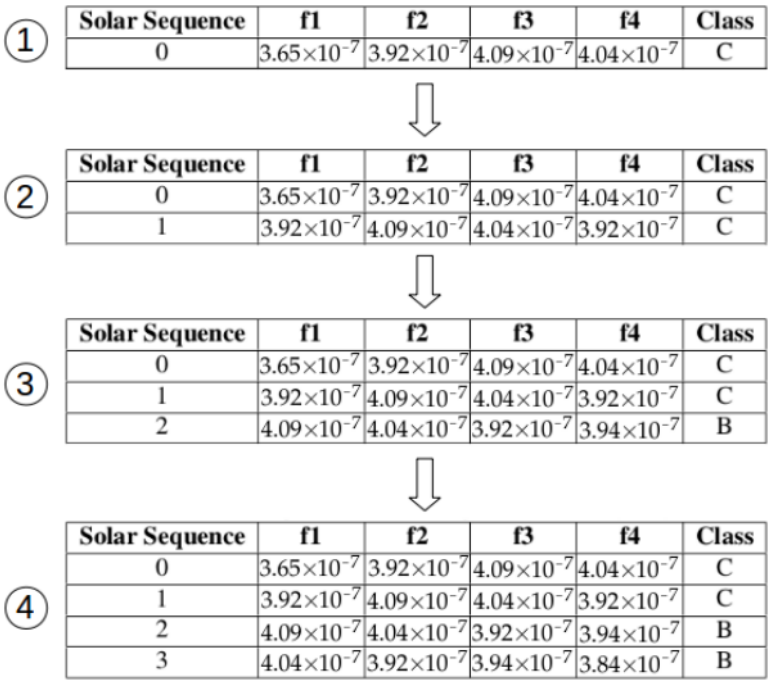
Figura 2 - Funcionamento do algoritmo SS.

X-ray Time Series						
t	**x(t)**	**Solar flare class**	***window-0***	***window-1***	***window-2***	***window-3***
0	3.65E-7	B	current window	step	step	step
5	3.92E-7	B		current window	current window	current window
10	4.09E-7	B				
15	4.04E-7	B				
20	3.92E-7	B	jump	jump	jump	jump
25	3.94E-7	B				
30	3.84E-7	B				
35	3.80E-7	B				
40	3.80E-7	B	future window	future window	future window	future window
45	3.83E-6	C				
50	3.84E-7	B				
55	3.90E-7	B				
60	6.47E-7	B				
65	6.75E-7	B				
70	5.24E-7	B				

Fonte: (DÍSCOLA JR, 2019)

Em cada iteração, o SS produz um mapeamento dos valores observados na *current window*, para a máxima classe de explosão solar observada na *future window*, criando um novo conjunto de dados. A construção desse conjunto, iteração por iteração, pode ser observada na Figura 3.

Figura 3 - Execução do algoritmo SS, iteração por iteração.



①

Solar Sequence	f1	f2	f3	f4	Class
0	3.65×10^{-7}	3.92×10^{-7}	4.09×10^{-7}	4.04×10^{-7}	C

②

Solar Sequence	f1	f2	f3	f4	Class
0	3.65×10^{-7}	3.92×10^{-7}	4.09×10^{-7}	4.04×10^{-7}	C
1	3.92×10^{-7}	4.09×10^{-7}	4.04×10^{-7}	3.92×10^{-7}	C

③

Solar Sequence	f1	f2	f3	f4	Class
0	3.65×10^{-7}	3.92×10^{-7}	4.09×10^{-7}	4.04×10^{-7}	C
1	3.92×10^{-7}	4.09×10^{-7}	4.04×10^{-7}	3.92×10^{-7}	C
2	4.09×10^{-7}	4.04×10^{-7}	3.92×10^{-7}	3.94×10^{-7}	B

④

Solar Sequence	f1	f2	f3	f4	Class
0	3.65×10^{-7}	3.92×10^{-7}	4.09×10^{-7}	4.04×10^{-7}	C
1	3.92×10^{-7}	4.09×10^{-7}	4.04×10^{-7}	3.92×10^{-7}	C
2	4.09×10^{-7}	4.04×10^{-7}	3.92×10^{-7}	3.94×10^{-7}	B
3	4.04×10^{-7}	3.92×10^{-7}	3.94×10^{-7}	3.84×10^{-7}	B

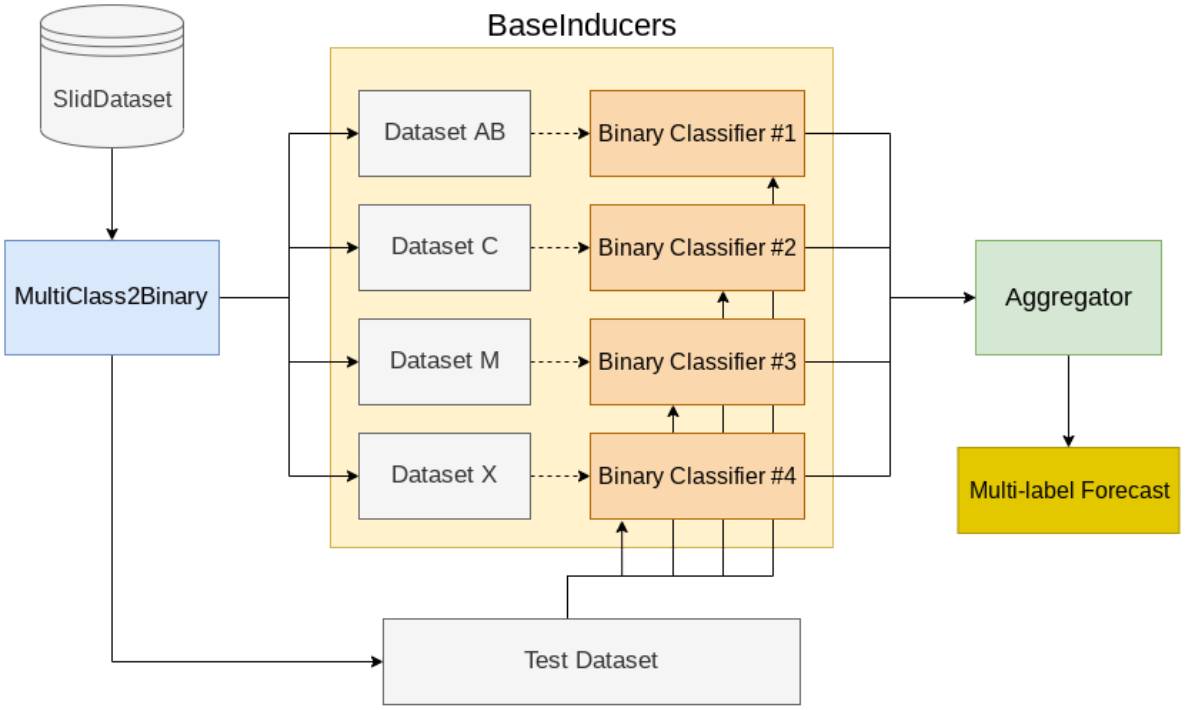
Fonte: (DÍSCOLA JR, 2019)

O SeMiner então encaminha essa saída, chamada de *Slid Dataset* (SD), para uma etapa de extração de características, e por fim para classificadores binários tradicionais, tornando possível a obtenção de previsões binárias: explosões de classe maior ou igual a C são positivas e as outras são negativas. O ECID, por outro lado, utiliza o SD como dados de entrada para o módulo *MultiClass2Binary*, como é explicado na subseção 2.3.2.

2.3.2 - Funcionamento do método ECID

O método ECID tem como dois de seus principais objetivos tratar a natureza desbalanceada dos dados do domínio das explosões solares, e a dificuldade de definir com precisão a gravidade de uma explosão. Apesar de ser voltado para o problema da previsão de explosões solares, ele pode ser aplicado a quaisquer outros problemas que tratem um domínio com dados desbalanceados e que permita que um único objeto receba vários rótulos de classe. A Figura 4 ilustra o seu modelo teórico.

Figura 4 - Modelo teórico ECID.

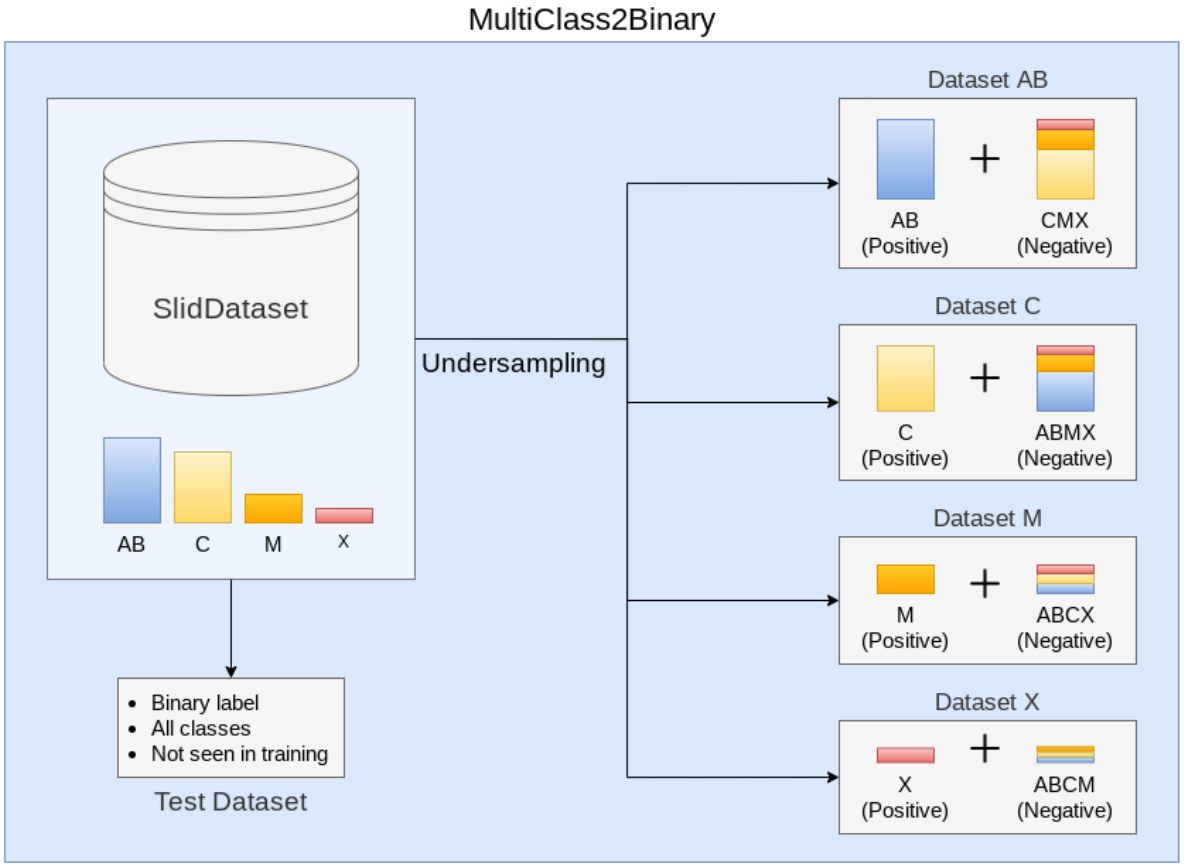


Fonte:Adaptado de (DÍSCOLA JR, 2019)

A primeira etapa do método é a divisão dos dados produzidos pelo algoritmo SS (explicado na Subseção 2.3.1) em quatro conjuntos balanceados e o conjunto de testes. Os conjuntos balanceados são gerados a partir de uma estratégia de *undersampling* aleatória, e o módulo *MultiClass2Binary*, representado na Figura 4 pelo bloco à esquerda, é o responsável por essa etapa. Além disso, esse módulo também separa parte dos dados disponíveis para o conjunto de testes.

Tratando-se dos conjuntos balanceados, cada um deles consiste em 50% de exemplos pertencentes à uma classe, e 50% de exemplos pertencentes às outras classes. Além disso, eles são mapeados para um formato binário. Por exemplo, o conjunto referente à classe AB tem 50% de exemplos da classe AB, e a classe AB recebe o rótulo binário “positivo”, enquanto os outros exemplos do conjunto são das classes C, M, e X, e recebem o rótulo binário “negativo”. A Figura 5 ilustra o funcionamento do módulo.

Figura 5 - Módulo MultiClass2Binary.

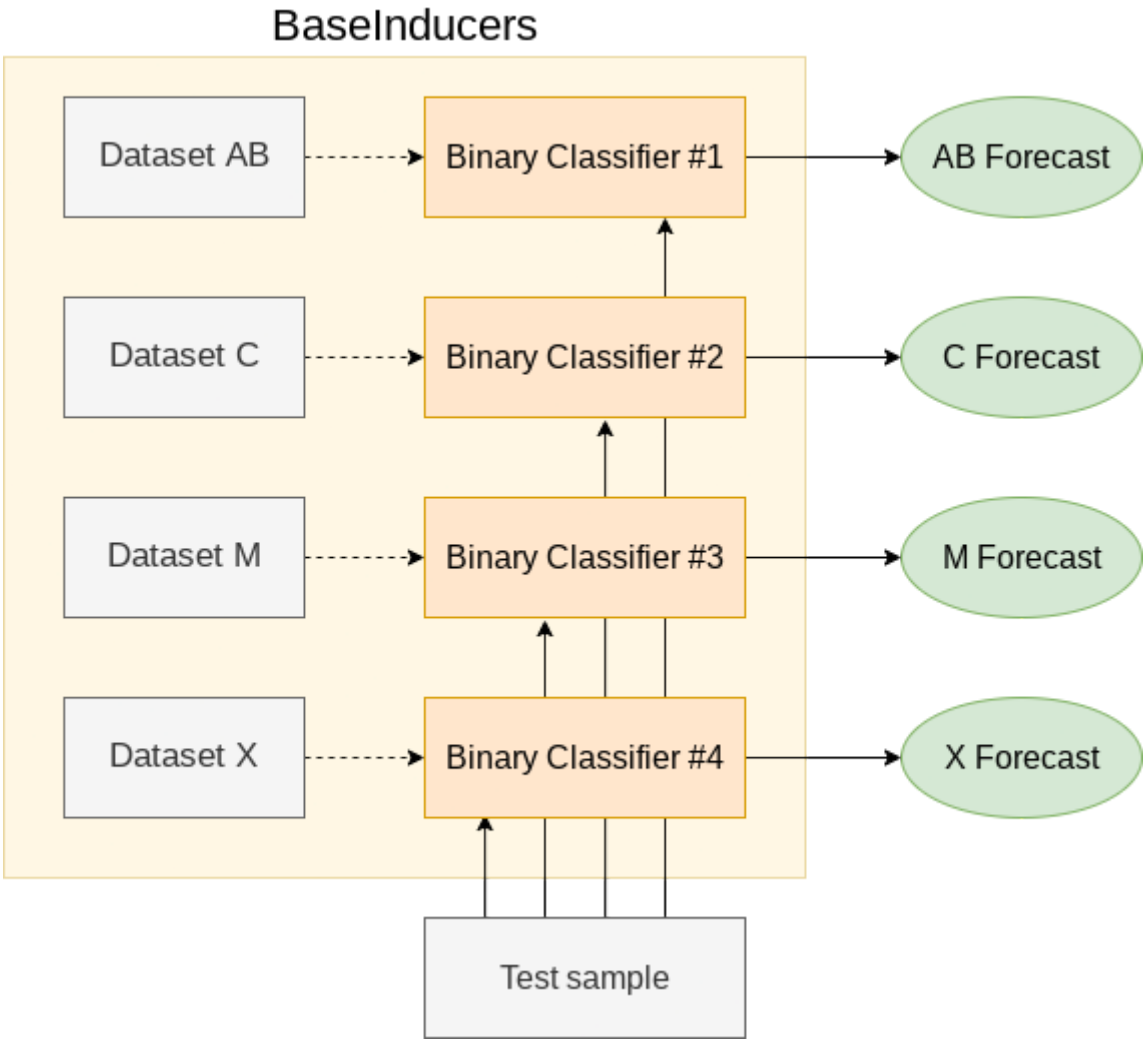


Fonte: Adaptado de (DÍSCOLA JR, 2019)

Como o número de exemplos de cada classe de explosão solar no *Slid Dataset* é diferente, os conjuntos balanceados resultantes terão tamanhos diferentes. Isso acontece por causa da estratégia de *undersampling*, que busca um equilíbrio a partir da classe positiva: se existem muitos exemplos positivos, deverão existir muitos exemplos negativos, e se existem poucos exemplos positivos, deverão existir poucos exemplos negativos.

Em seguida, cada conjunto balanceado é usado para o treinamento de um classificador binário, resultando em quatro classificadores (AB, C, M, X). Cada classificador é responsável por avaliar se o dado de entrada reflete uma explosão solar futura da sua respectiva classe. Como o resultado de todos os classificadores deve ser considerado, cada dado de teste é passado aos quatro classificadores. Todo esse processo acontece no módulo *BaseInducers*, e é ilustrado na Figura 6.

Figura 6 - Quatro classificadores.



Fonte: Adaptado de (DÍSCOLA JR, 2019)

As previsões dos classificadores são contabilizadas como votos para a previsão final. Essa tarefa é realizada pelo módulo *Aggregator*. Se para determinada instância do conjunto de testes, o classificador #1 resultou positivo, será contabilizado um ponto para a classe AB. Se para a mesma instância, o classificador #2 também resultou positivo, a classe C também ganha um ponto. Repetindo esse processo para várias instâncias do conjunto de testes, e considerando que o *Slid Dataset* foi gerado a partir de uma série temporal medida a cada 12 minutos, tem-se a situação da Tabela 2.

Tabela 2 - Previsões individuais dos classificadores.

instant	AB_y	C_y	M_y	X_y
Day_1_Instant_00:00	1	1	0	0
Day_1_Instant_00:12	...	...	...	...
Day_1_Instant_00:24	...	...	...	...

Fonte: (DÍSCOLA JR, 2019)

A Tabela 3 mostra os votos já agregados e organizados de forma a termos uma representação dividida em dias.

Tabela 3 - Agregação das previsões individuais.

instant	AB_y	C_y	M_y	X_y
Day_1	80	40	0	0
Day_2	0	0	20	100
Day_3	0	2	90	28

Fonte: (DÍSCOLA JR, 2019)

A escolha das classes da previsão final é então feita de maneira parametrizada. São definidos os parâmetros p1, p2 e p3, que correspondem ao número mínimo de votos que a primeira, segunda e terceira classe mais votadas devem possuir para serem associadas à previsão final. A Tabela 4 ilustra essa escolha para os parâmetros (p1 = 24, p2 = 6, p3 = 3).

Tabela 4 - Cálculo do resultado da previsão.

Instant	1th Voted	2nd Voted	3rd Voted	Forecasting Result
Day_1	24 <= 80 (AB)	6 < 40 (C)	-	ABC
Day_2	24 <= 100 (X)	6 <= 20 (M)	-	X
Day_3	24 <= 90 (M)	6 <= 28 (X)	3 <= 2 (C)	MX

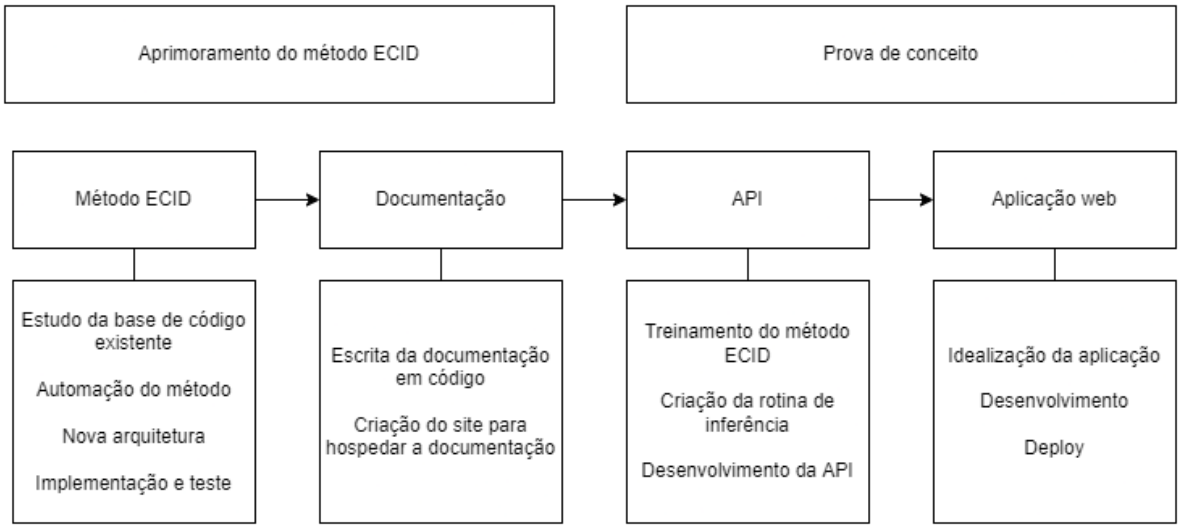
Fonte: (DÍSCOLA JR, 2019)

Quando mais de uma classe é selecionada no processo de previsão, obtém-se uma classificação multirrótulo, ou seja, objetos podem receber múltiplos rótulos de classe (AB, C, M, X), e não apenas um. Essa característica permite, por exemplo, que um especialista do domínio possua algumas alternativas para a escolha final quando classes adjacentes são previstas.

3 - Metodologia

De maneira geral, os objetivos deste trabalho podem ser simplificados em duas tarefas: aprimoramento do método ECID, e construção da aplicação como prova de conceito do uso da API. Dito isso, o fluxograma da Figura 7 foi seguido para o desenvolvimento das atividades.

Figura 7 - Fluxograma de trabalho.



Fonte: Próprio Autor

3.1 - Aprimoramento da base de código

O processo de aprimoramento do método ECID se iniciou com a replicação dos resultados presentes no trabalho de (DÍSCOLA JR, 2019). O autor do texto foi consultado e tanto a base de dados quanto a base de código do método foram disponibilizadas, mas não foi concedida permissão para compartilhamento do código fonte. O projeto utiliza a linguagem Java 8 e a biblioteca Weka 3.8 (FRANK, 2016) para sua implementação.

A versão recebida do autor possuía alguns elementos do ECID não implementados, como a parte relacionada ao módulo *MultiClass2Binary*. Os resultados alcançados na tese foram obtidos executando algumas etapas do ECID com auxílio de ferramentas externas, como o uso da interface gráfica da própria biblioteca Weka (FRANK, 2016).

A partir de então, iniciou-se o estudo da aplicação em detalhe. O método de funcionamento consistia no usuário utilizar a linha de comando para escolher a

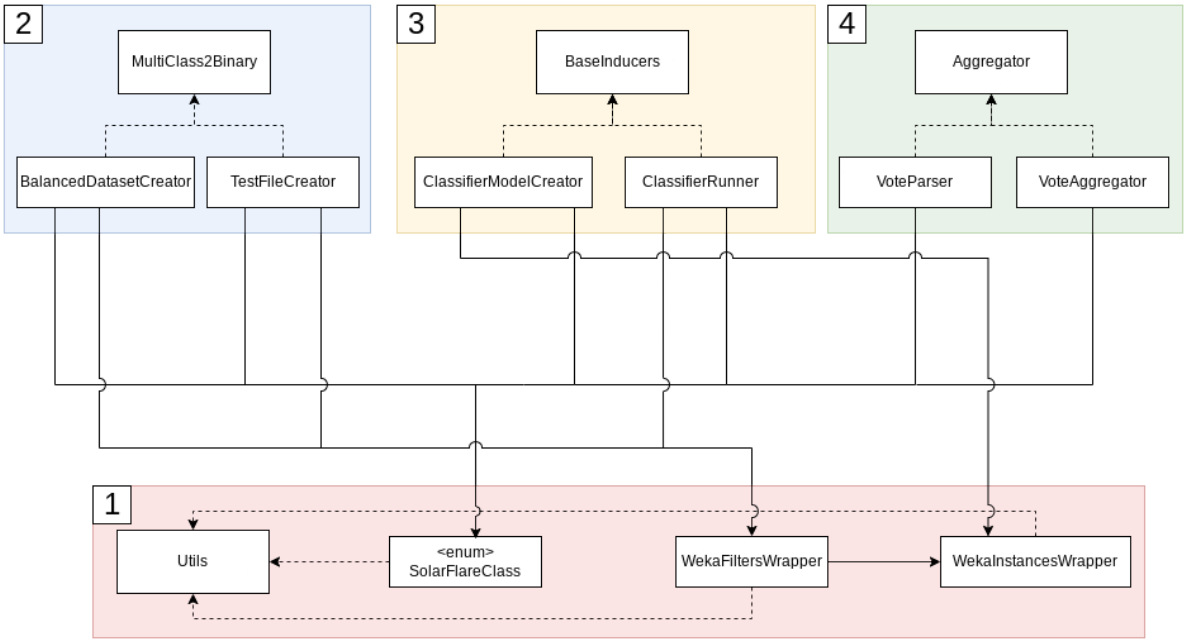
operação que o ECID deveria realizar, por exemplo: “Treinar classificadores” ou “Realizar contagem de votos”. A integração entre essas operações se dava por meio de arquivos: uma operação recebia, processava e retornava um ou mais arquivos. Sabendo disso, foi adotada uma estratégia de *black-box testing*, que trata o programa a ser testado como uma caixa preta, e se preocupa mais com as entradas e saídas (MYERS, 2011). Os testes foram implementados em bash, tomando os arquivos de entrada e saída de cada operação como referência. Essas rotinas foram então incorporadas no processo de desenvolvimento como uma medida para detecção de erros.

O próximo passo foi identificar pontos de melhoria na aplicação. Algumas questões foram observadas:

- Leitura e escrita de dados: Em sua maioria, os arquivos tratados pela aplicação são *Comma Separated Values* (CSV) e *Attribute-Relation File Format* (ARFF), um formato específico da biblioteca Weka. Tais operações nestes arquivos envolviam uma lógica convoluta que era replicada em diversas partes da base de código;
- Processamento de dados: Operações comuns em um mesmo conjunto de dados não eram devidamente reaproveitadas em código;
- Representação das classes de explosões solares: Cada rotina assumia a existência das 4 classes de explosões solares (AB, C, M, X), gerando vários pontos de erro, caso isso mude no futuro (exemplo: diferenciar classes A e B).

Foi então proposta e implementada uma nova arquitetura para o método ECID, que busca tratar essas questões. Ilustrada na Figura 8, a arquitetura tenta se aproximar ao máximo do modelo teórico do autor, ao passo que se mantém compatível com o modo de funcionamento da aplicação original, utilizando majoritariamente arquivos para fluxo de informações. As setas contínuas indicam a relação "utiliza", e as setas pontilhadas indicam uma relação de pertencimento ao módulo.

Figura 8 - Nova arquitetura proposta para o ECID.



Fonte: Próprio autor

3.1.1 - Módulo *Utils*

A primeira parte da arquitetura implementada foi o módulo *Utils* (1 na Figura 8), que busca aumentar o isolamento entre a lógica de funcionamento do método ECID e a implementação. Ele contém classes que serão utilizadas pelos outros módulos do programa para tarefas como leitura e escrita de dados, processamento desses dados, e representação de conceitos específicos do domínio; atacando diretamente as questões levantadas anteriormente.

Ter a lógica de funcionamento separada da implementação permite a criação de sistemas mais flexíveis. Dito isso, a implementação ainda utiliza a biblioteca Weka para processos de leitura e escrita de dados, treinamento de classificadores, e teste desses classificadores. A contribuição, nesse sentido, está na maior facilidade de substituí-la por outra que execute funções semelhantes.

A classe *WekaInstancesWrapper* simplifica as tarefas de leitura e escrita de dados, além de fornecer algumas utilidades para conversão e formatação entre os principais tipos de arquivo utilizados: CSV e ARFF.

A classe *WekaFiltersWrapper* busca tornar as operações de processamento de dados mais eficientes. Ela utiliza a funcionalidade de filtros da biblioteca Weka para fazer modificações em elementos de um conjunto de dados que obedeçam a determinada condição. Além disso, é permitida a definição de dois tipos de filtros: os

comuns e os específicos. Antes da aplicação dos filtros específicos, todos os filtros comuns são aplicados. Esse estado é salvo e restaurado após a aplicação de cada filtro específico, gerando reaproveitamento. Além disso, a *WekaFiltersWrapper* utiliza as funcionalidades da *WekaInstancesWrapper*, oferecendo uma forma fácil de gravar os resultados após o processamento.

A enumeração *SolarFlareClass* tem como objetivo unificar as referências a esse tipo de informação no código, e aumentar a legibilidade.

3.1.2 - Módulo *MultiClass2Binary*

A segunda parte da arquitetura a ser implementada, o módulo *MultiClass2Binary* (2 na Figura 8), foi construída tendo apenas como referência suas entradas e saídas, já que ela não existia na implementação original. Esse módulo consiste nas classes *BalancedDatasetCreator* e *TestFileCreator*, que usam a funcionalidade de filtros do Weka a partir da classe *WekaFiltersWrapper* para processar os dados no formato esperado pelos classificadores (binário). A diferença entre elas é que a primeira produz quatro conjuntos de dados balanceados, que serão utilizados para treinamento dos classificadores, enquanto a segunda gera os conjuntos que serão utilizados na etapa de teste e não se preocupa com o balanceamento. As principais operações aplicadas pelos filtros são: criação de um subconjunto aleatório com uma determinada distribuição de classes e alteração dos rótulos dos exemplos para refletir uma classificação binária.

3.1.3 - Módulo *BaseInducers*

O terceiro módulo implementado foi o *BaseInducers* (3 na Figura 8). Esse módulo é responsável pela criação e teste dos classificadores binários. Para a criação e treinamento dos classificadores, temos a classe *ClassifierModelCreator*, que usa a classe *WekaInstancesWrapper* para formatar os dados de entrada da maneira adequada para manipulação pela biblioteca Weka. Os classificadores binários utilizados são a implementação do algoritmo de árvore de decisão J48 do Weka, baseado no algoritmo C4.5 de (QUINLAN, 1993), e os dados de treinamento usados são o resultado do módulo *MultiClass2Binary*.

A classe *ClassifierRunner* realiza a inferência de cada exemplo do conjunto de teste em todos os quatro classificadores, e usa a classe *WekaFiltersWrapper* para organização e escrita de resultados intermediários. Os resultados finais da

inferência são escritos em um formato específico da biblioteca Weka, e que muda de acordo com o tipo de classificador binário utilizado.

3.1.4 - Módulo *Aggregator*

O quarto módulo implementado é chamado *Aggregator* (4 na Figura 8). Ele usa a classe *VoteParser* como forma de processamento intermediário dos resultados finais do módulo *BaseInducers*, organizando-os em uma lista. Essa abordagem foi adotada em detrimento do armazenamento em arquivos para manter a compatibilidade com a rotina do método original que executa uma função semelhante. A próxima classe, *VoteAggregator*, utiliza a lista construída anteriormente, agrega os dados disponíveis em dias, e aplica a lógica de decisão do método explicada ao final da Subseção 2.3.2 para produzir as previsões finais.

3.1.5 - Considerações

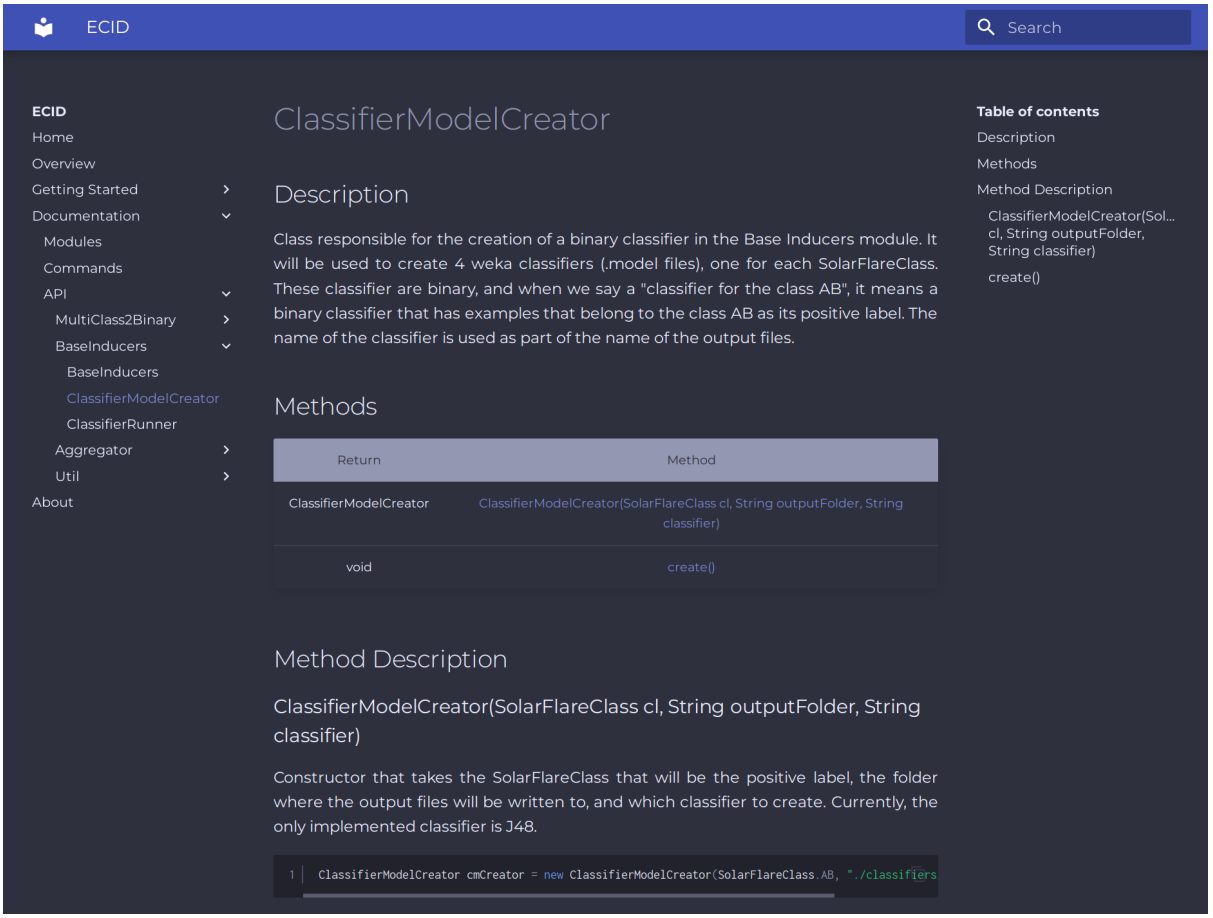
A equivalência da versão aprimorada com a original foi garantida por meio dos testes de caixa preta utilizados durante o desenvolvimento, que comparavam o resultado das duas versões para diferentes entradas. Foi observada uma divergência máxima de um voto durante a etapa de teste, o que é considerado aceitável, visto que essa divergência não foi suficiente para alterar a previsão final de nenhuma instância, e pode ser atribuída a pequenas diferenças de versão ou configuração de alguma ferramenta utilizada, como a biblioteca Weka, a linguagem Java ou até mesmo o *hardware*.

Ao longo de todo o processo de implementação, foram feitas diversas melhorias no código, como remoção de trechos duplicados, escolha de estruturas de dados mais eficientes, tratamento de erros, e adição de comentários e nomes de variáveis mais significativos. Apesar disso, ainda há espaço para melhorias, tanto arquiteturais quanto referentes à implementação. Um exemplo disso é a questão do acoplamento da biblioteca Weka à lógica de funcionamento do código. Apesar da nova arquitetura amenizar um pouco essa questão, ela ainda persiste, especialmente no módulo *Aggregator*, que trabalha diretamente com resultados que dependem da implementação do classificador binário utilizado no módulo *BaseInducers*.

3.2 - Documentação

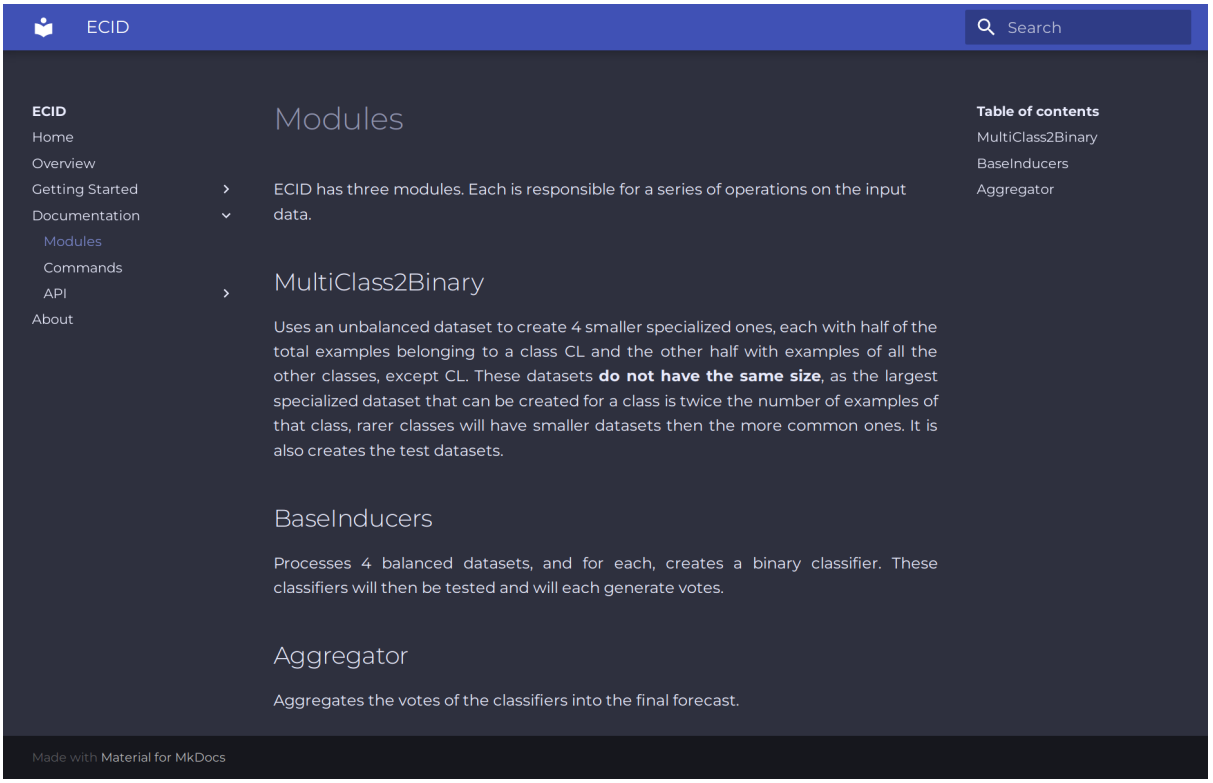
Ao longo do desenvolvimento, o código foi documentado seguindo o padrão da ferramenta Javadoc. Isso aconteceu tanto para as novas funcionalidades, quanto para funcionalidades já existentes que não possuíam documentação. Feito isso, a ferramenta de geração de sites estáticos MkDocs foi utilizada para criação de uma página da web para o projeto. O objetivo dessa página foi agregar informações como instruções de como configurar e treinar o ECID localmente, explicações sobre as classes utilizadas na implementação (quais funcionalidades elas oferecem e como usá-las), visão geral da arquitetura e mais detalhes do ECID como modelo teórico. O site foi hospedado na plataforma GitLab para uso dos colaboradores do projeto, e as Figuras 9 e 10 mostram algumas de suas telas.

Figura 9 - Seção da classe *ClassifierModelCreator* no site da documentação.



Fonte: Próprio autor

Figura 10 - Seção *Modules* no site da documentação.



Fonte: Próprio autor

3.3 - API RESTful

De maneira simplificada, uma API (*Application Programming Interface*) é um conjunto de funcionalidades disponibilizadas de forma organizada por uma aplicação para que outra aplicação possa utilizá-las. Como um dos objetivos deste trabalho é tornar os resultados do método ECID disponíveis, uma API simples que se adequa às restrições REST (*Representational State Transfer*) foi desenvolvida.

Essa API se encontra no nível 1 de maturidade de acordo com a classificação de Richardson (FOWLER, 2010), e trata cada previsão realizada pelo método como um recurso. Ela fornece acesso a operações como consulta, listagem, filtro e inferência de um recurso ou conjunto de recursos.

A Tabela 5 indica os *endpoints* para cada funcionalidade da API, o método HTTP necessário para acioná-los, e oferece um breve resumo da operação executada.

Tabela 5 - Operações da API e endpoints.

Endpoint	Método	Funcionalidade
/predictions	**GET**	Lista *dummy* de previsões

/prediction/DATA	**GET**	Consulta o elemento DATA da lista dummy
/predictions/q/ANO	**GET**	Filtra a lista *dummy* por ANO
/predict	**POST**	Inferência

Fonte: Próprio autor

Os endpoints /predictions, /prediction/DATA e /predictions/q/ANO ilustram, respectivamente, as operações de consulta de um conjunto de previsões, consulta de um única previsão, e aplicação de filtros sobre um conjunto de previsões. A maior contribuição desses endpoints foi para testes durante o período de desenvolvimento, e para representar funcionalidades que podem ser implementadas no futuro. O endpoint /predict é o responsável por realizar a inferência ou previsão do ECID em um novo exemplo.

A inferência acontece a partir de uma data passada por um objeto JSON (*Javascript Object Notation*) na requisição HTTP do tipo POST ao endpoint /predict. O JSON deve possuir um atributo chamado *date*, e seu conteúdo deve ser uma sequência de caracteres no formato YYYY.MM.DD, sendo cada Y um dígito do ano, cada M um dígito do mês, e cada D um dígito do dia, como é ilustrado na figura 11.

Figura 11 - Exemplo de objeto a ser enviado ao endpoint /predict.

```
{
    "date": "2017.10.21"
}
```

Fonte: Próprio autor

A API então realiza algumas validações na data recebida, e monta corretamente a consulta a uma base de dados MongoDB que contém as informações necessárias para realizar a previsão. Para isso é levada em consideração a configuração sob a qual a instância do ECID que será consultada foi treinada. Caso as informações requeridas não existam na base, é acusado erro, e não há inferência. Caso contrário, a inferência ocorre.

O processo de inferência consiste na execução apenas das partes relevantes do método ECID completo. Uma instância treinada do ECID já possui um conjunto de classificadores binários. Com isso, é possível utilizar a classe *ClassifierRunner* do módulo *BaseInducers* para alimentar os dados consultados do MongoDB aos

classificadores, e em seguida realizar a contagem de votos com as classes *VoteParser* e *VoteAggregator* do módulo *Aggregator*, gerando a previsão final.

Os resultados dessa previsão são então retornados no formato JSON. Uma previsão realizada com sucesso retorna informações como o rótulo previsto (“Prediction”), o número de votos positivos (Y) e negativos (N) que cada classe recebeu e a data a qual se refere a previsão (“date”). Caso outras informações a respeito da data em questão estejam disponíveis na base de dados, elas também serão retornadas.

O retorno de uma previsão realizada com sucesso pode ser observado na Figura 12. Esse é o formato de um “recurso” da API, e todos os endpoints retornam ou um recurso ou um conjunto de recursos. Para implementação da API foi utilizado o framework *Express* para *Node.js*.

Figura 12 - Exemplo de JSON retornado pela API após uma inferência realizada com sucesso.

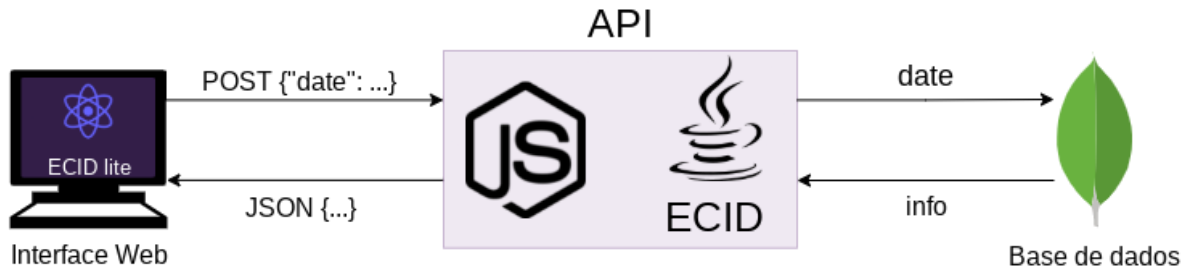

```
{
    "Prediction": "AB",
    "RealClass": "AB",
    "abN": "0",
    "abY": "120",
    "bgLevel": "A7.4",
    "cN": "0",
    "cY": "120",
    "date": "2017.10.21",
    "mN": "0",
    "mY": "120",
    "magConfig": "Alpha",
    "radio": "77",
    "xN": "0",
    "xY": "120"
}
```


Fonte: Próprio autor

3.4 - Aplicação web - ECID lite

Utilizando a biblioteca React da linguagem Javascript e a API desenvolvida anteriormente, foi construída a aplicação web ECID lite. A figura 13 ilustra sua arquitetura.

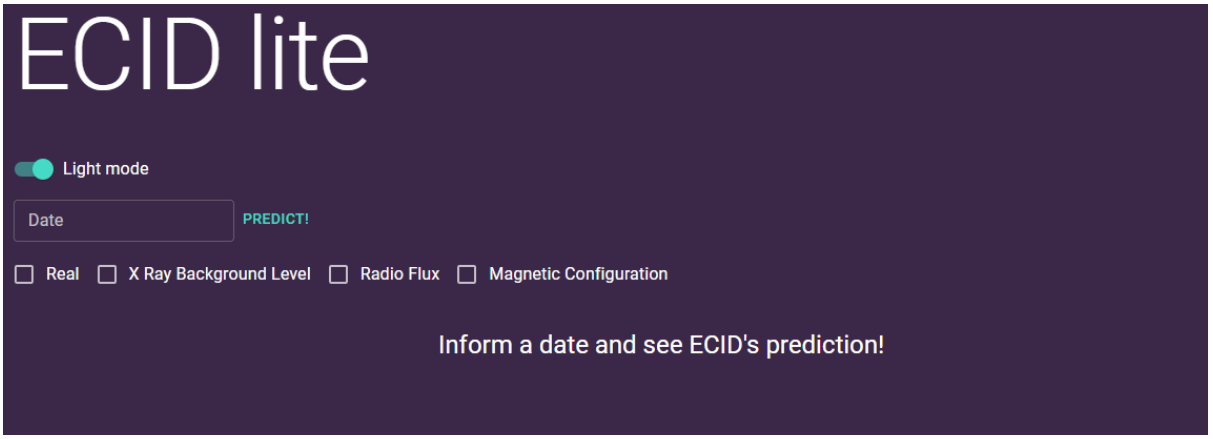
Figura 13 - Arquitetura da aplicação web.



Fonte: Próprio autor

É uma aplicação simples que oferece uma maneira fácil de acessar os resultados de uma instância do ECID treinada, permitindo que pessoas sem grande conhecimento técnico sejam capazes de trabalhar diretamente com um modelo para previsão de explosões solares. Ela permite ao usuário informar uma data e conferir a previsão do ECID para tal data.

Figura 14 - Tela do ECID lite sem previsões.

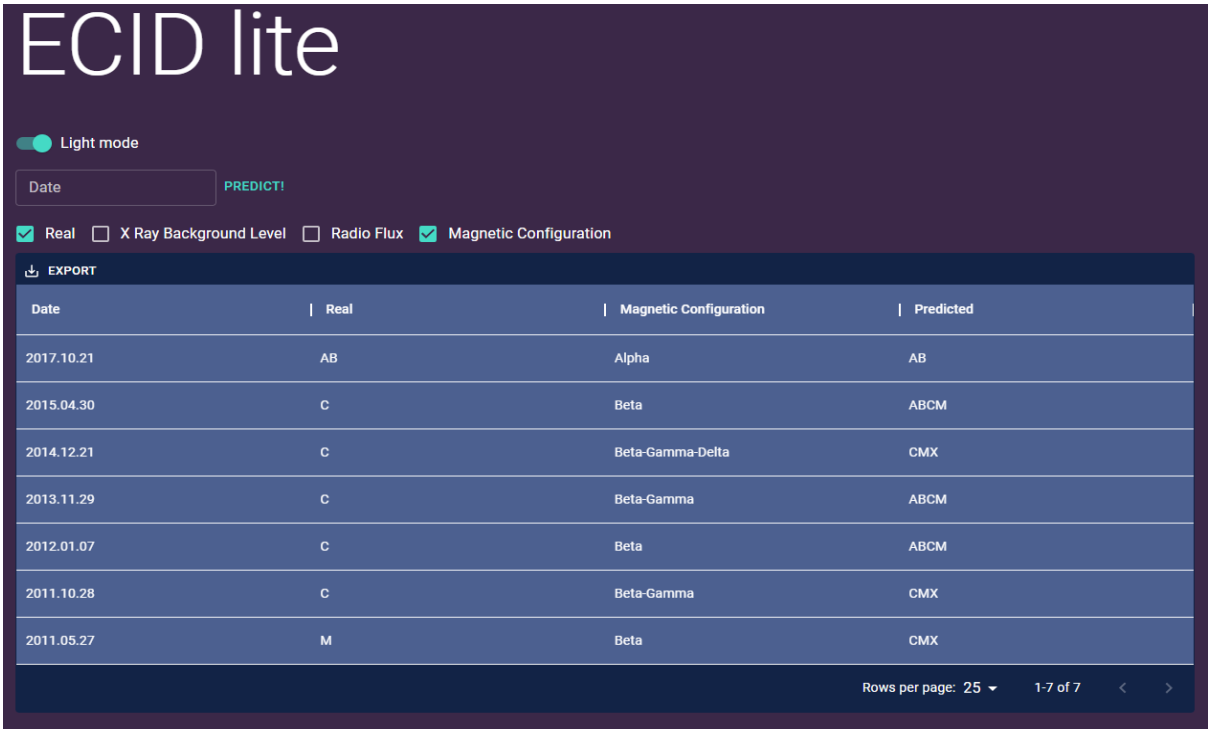


Fonte: Próprio autor

As datas consultadas são mantidas em um histórico no formato de tabela, e podem ser exportadas juntamente com informações de cada previsão em um arquivo CSV. Os arquivos gerados pela aplicação tem as informações:

- *Real*: verdadeira classe da explosão solar que aconteceu na data informada;
- *Magnetic Configuration*: máxima classificação magnética de *Mount Wilson* (HALE et al., 1919) no dia;
- *Background level*: valor máximo de raio-x de fundo observado no dia;
- *Radio flux*: valor máximo do fluxo de rádio solar no comprimento de onda de 10.7cm no dia.

Figura 15 - Tela do ECID lite com previsões.



Fonte: Próprio autor

O modelo do ECID atualmente disponível para ser utilizado por meio da aplicação utiliza informações de dois dias para suas previsões, e tem horizonte de zero dias. Isso significa que, para cada data consultada, o resultado é calculado utilizando informações de dois dias antes da data indicada: se o dia 10 for consultado, serão utilizadas informações dos dias 8 e 9 para prever a classe de explosão do dia 10.

Optando pela simplicidade e facilidade de acesso, a única informação requerida por parte do usuário para realizar uma previsão é a data desejada. Essa característica traz uma desvantagem: caso não exista informação suficiente para realizar a previsão de tal data na base de dados consultada pela API, não há previsão.

A base de dados utilizada atualmente conta com informações desde meados de 2010 até o fim de 2017, sendo que nem todas as datas dentro desse intervalo possuem informação completa para previsão. Uma lista com todas as datas válidas para consulta pode ser encontrada no Apêndice A.

O serviço de *Platform as a Service* (PaaS) Heroku foi utilizado para hospedagem da aplicação ECID lite. Ela pode ser acessada a partir da seguinte URL: https://tranquil-bayou-64910.herokuapp.com/.

4 - Resultados e discussão

Este capítulo apresenta os resultados deste trabalho. Os resultados que envolvem processamento computacional foram obtidos utilizando a seguinte configuração: CPU Intel(R) Core(™) i7-7500U @ 2.70GHz, 12GB de memória RAM, sistema operacional Ubuntu 20.04.

4.1 - Versão aprimorada do método ECID

Para quantificar o resultado do aprimoramento feito ao método ECID, foram utilizadas algumas métricas de avaliação estática de complexidade de código. Como a versão aprimorada contém funcionalidades que a versão anterior não possui, a comparação considerou apenas as partes equivalentes das versões. A base de código original foi reorganizada na mesma estrutura de módulos da base aprimorada, como forma de facilitar as comparações.

A primeira métrica analisada é o número de linhas de código fonte da linguagem Java. Essa métrica representa, de maneira geral, o tamanho do projeto. Um número maior geralmente indica maior complexidade.

Tabela 6 - Comparação do número de linhas de código, separado por módulo entre a versão original e a aprimorada.

Versão	Módulo	Número de linhas	Total
Original	*BaseInducers*	477	1575
	Aggregator	1082	
	Utils	16	
Aprimorada	*BaseInducers*	398	1425
	Aggregator	561	
	Utils	466	

Fonte: Próprio autor

A versão aprimorada reduziu o número de linhas de código do programa em 150, cerca de 10%. Os módulos *BaseInducers* e *Aggregator* sofreram redução, enquanto o módulo *Utils* sofreu um aumento para 466 linhas. Este resultado é esperado, já que agora o módulo *Utils* contém classes que são utilizadas em todo o programa. Se compararmos o número de linhas dedicadas à lógica do programa

(excluir o módulo *Utils*), temos 1559 linhas da versão anterior contra 959 linhas da versão aprimorada.

Utilizando o *plugin* Statistic (TOPINKA, 2009) da ferramenta IntelliJ, é possível ter uma visão mais detalhada da métrica de linhas de código, separando as linhas correspondentes a código fonte, comentário e em branco. Isso é ilustrado na Tabela 7.

Tabela 7 - Comparação da porcentagem de linhas de código, separada por tipo entre a versão original e a aprimorada.

	Porcentagem de linhas de código		
Versão	Código Fonte	Comentário	Em branco
Original	71%	14%	15%
Aprimorada	63%	20%	17%

Fonte: Próprio autor

A cobertura de comentários da versão aprimorada aumentou 6% quando comparada com a versão original. Isso pode ser atribuído à tarefa de documentação. A porcentagem de código fonte sofreu uma redução de 8%, provavelmente causada pelos esforços na redução de código duplicado.

A segunda métrica a ser analisada é a complexidade ciclomática. Essa métrica busca medir a complexidade de um programa a partir do número de caminhos de execução diferentes. O plugin CodeMetrics (KISS, 2019) da ferramenta IntelliJ foi utilizado para o cálculo, e os resultados são mostrados na Tabela 8.

Tabela 8 - Comparação da complexidade ciclomática, separada por módulo entre a versão original e a aprimorada.

		Complexidade ciclomática	
Versão	Módulo	Média	Total
Original	*BaseInducers*	4.80	48
	Aggregator	6.87	108
	Utils	3.00	3
Aprimorada	*BaseInducers*	2.05	43
	Aggregator	4.83	61
	Utils	1.96	55

Fonte: Próprio autor

Apesar da soma dos valores de complexidade total ser a mesma entre a aplicação original e a aprimorada (159), observou-se uma redução significativa da complexidade ciclomática média nos módulos da aplicação aprimorada: os módulos *BaseInducers, Aggregator e Utils* apresentaram, respectivamente, valores 57%, 29% e 34% menores. Isso foi acompanhado de um grande aumento da complexidade total do módulo Utils, resultado esperado considerando as modificações feitas.

A terceira métrica a ser analisada é o tempo de execução completa do método ECID. Processos de aprendizado de máquina são altamente iterativos, ou seja, até a produção do modelo final, diversos ciclos de treinamento e teste dos resultados são executados. Uma redução no tempo necessário para concluir uma iteração permite que os pesquisadores consigam executar mais ciclos no mesmo período.

A metodologia adotada para os testes a seguir foi criar rotinas capazes de executar tanto o método original quanto o aprimorado do início (*MultiClass2Binary)* ao fim (*Aggregator*). Como o método original não possuía implementação do módulo *MultiClass2Binary*, a implementação da versão aprimorada foi utilizada. O tempo necessário para concluir uma rotina consiste em uma medição; e a média aritmética de cinco medições consiste em um *batch* de testes. A ideia dessa abordagem é reduzir a variância das medições. As medições foram feitas com o comando *time* do UNIX. Foram realizados cinco *batches* em cada versão do método ECID, e os resultados estão explicitados na Tabela 9.

Tabela 9 - Comparação do tempo de execução.

	Tempo (s)		
Batch	**Original**	**Aprimorada**	**Diferença**
1	153.83	118.93	22.69%
2	152.81	119.91	21.53%
3	152.52	119.51	21.64%
4	152.27	120.02	21.18%
5	153.30	119.20	22.24%

Fonte: Próprio autor

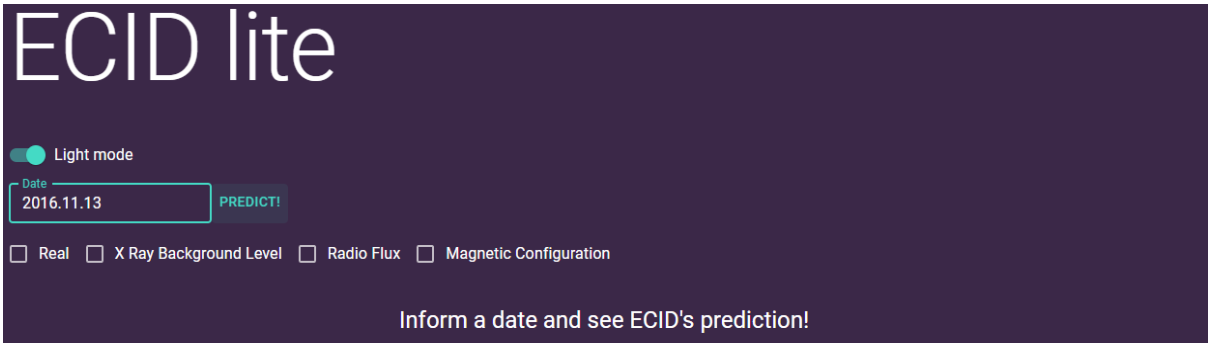
O tempo de execução da versão aprimorada fica entre 118 e 121 segundos, enquanto a versão original demora de 152 a 154 segundos. Isso se traduz em um ganho de desempenho de aproximadamente 21%, indicando que as alterações feitas na base de código tiveram impacto positivo na performance.

4.2 - Utilizando a aplicação ECID lite

Esta seção exemplifica como utilizar a aplicação ECID lite para realizar consultas, organizar e customizar essas consultas, e fazer o download do arquivo CSV contendo os resultados. Ela pode ser acessada a partir da URL https://tranquil-bayou-64910.herokuapp.com/.

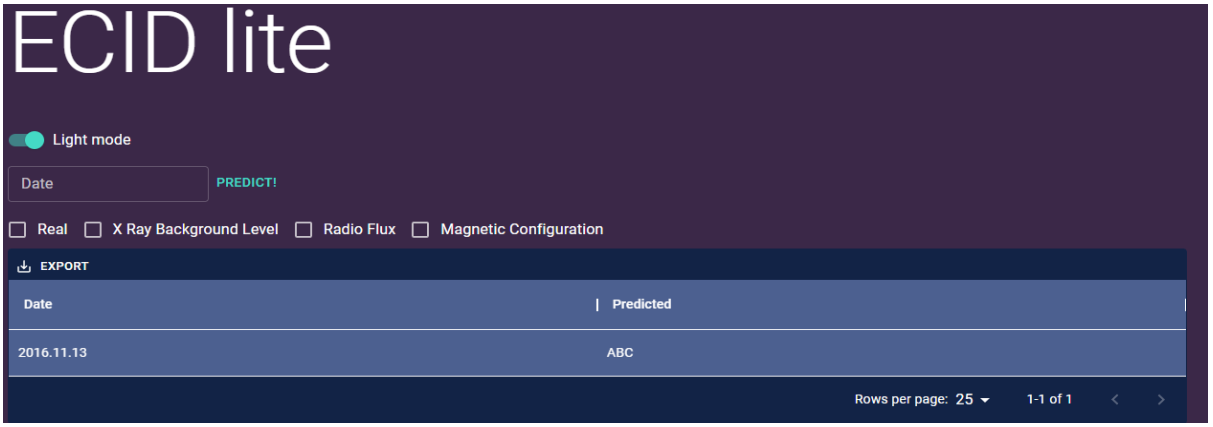
Inicialmente insira no campo "*Date*" a data que deseja consultar no formato YYYY.MM.DD. Em seguida, clique no botão "*Predict!*". Após alguns segundos, se o servidor possuir as informações necessárias para realizar a previsão nesta data, uma tabela aparecerá, e a data escolhida constará como uma entrada na tabela. Caso contrário, no topo da tela aparecerá uma mensagem de erro.

Figura 16 - Processo de consulta para a data 13 de novembro de 2016.



Fonte: Próprio autor

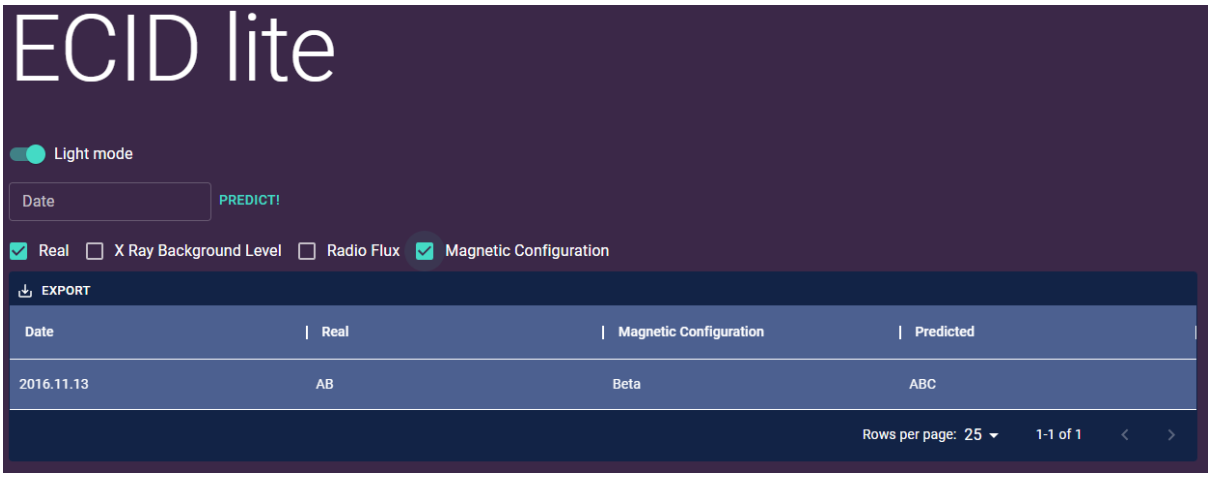
Figura 17 - Resultado da consulta para a data 13 de novembro de 2016.



Fonte: Próprio autor

A tabela gerada sempre possuirá duas colunas: a coluna “*Date*”, que representa a data consultada, e a coluna “*Predicted*”, que informa a previsão para essa data. Mais colunas podem ser adicionadas ou removidas ativando ou desativando as caixas de seleção acima da tabela.

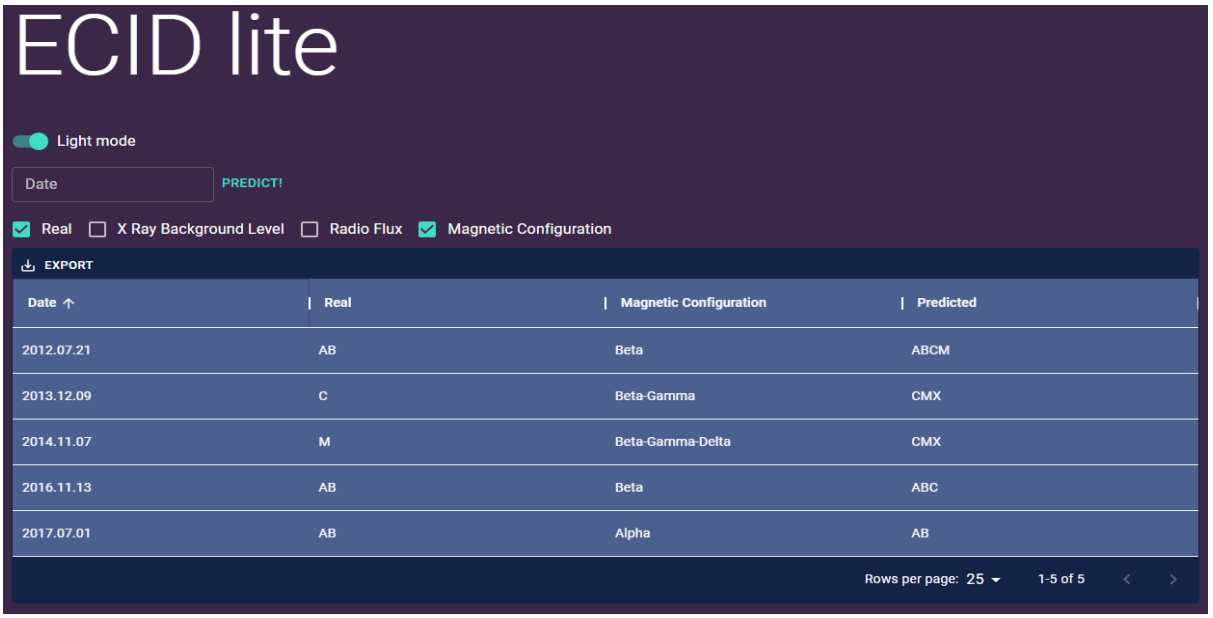
Figura 18 - Previsão com as caixas de seleção correspondentes à configuração magnética, e à real intensidade da explosão selecionadas.



Fonte: Próprio autor

Cada consulta de uma data válida adicionará uma nova entrada na tabela. Essas entradas podem ser ordenadas tendo como base uma das colunas disponíveis. Para fazer isso basta clicar na seta ao lado do nome da coluna. Um primeiro clique ordenará os dados de forma ascendente, um segundo clique ordenará de forma descendente, e um terceiro removerá a ordenação.

Figura 19 - Previsões ordenadas por data.



Fonte: Próprio autor

No canto esquerdo, na parte superior da tabela, fica o botão “*Export*”. Ao clicar nele, o usuário pode escolher a opção de exportar os dados como um arquivo CSV. Os dados exportados preservam a ordenação feita no aplicativo.

Figura 20 - Visualização do arquivo CSV gerado pelo ECID lite.

	A	B	C	D
1	Date	Real	Magnetic Configuration	Predicted
2	2012.07.21	AB	Beta	ABCM
3	2013.12.09	C	Beta-Gamma	CMX
4	2014.11.07	M	Beta-Gamma-Delta	CMX
5	2016.11.13	AB	Beta	ABC
6	2017.07.01	AB	Alpha	AB

Fonte: Próprio autor

5 - Conclusão

Este trabalho buscou aprimorar o método de previsão de explosões solares apresentado por (DÍSCOLA JR, 2019) no sentido de torná-lo completamente automatizado, mais eficiente, flexível e robusto à alterações futuras. Considerando isso, os resultados foram satisfatórios: a nova arquitetura e implementação propostas resultaram em um ganho de desempenho de cerca de 21% no tempo de execução, e a complexidade do código teve uma redução significativa, evidenciada pela queda da complexidade ciclomática média por módulo. Além disso, foi produzido um novo site para hospedar a documentação e informações adicionais relacionadas do projeto; algo que facilitará a integração de novos membros no futuro.

Entretanto, apesar desses avanços, restaram alguns pontos que ainda podem ser melhorados: algumas partes do código continuam bem acopladas à implementação, principalmente no módulo *Aggregator*; o funcionamento do método ECID ainda depende bastante do uso de arquivos para o fluxo de dados; e as implementações para as operações existentes dependem, em sua grande maioria, de uma única biblioteca. Esses pontos ainda impedem que o método ECID seja considerado totalmente flexível, e podem ser abordados em trabalhos futuros.

Com as modificações feitas no método, foi também desenvolvida uma API RESTful que expõe os resultados de um modelo ECID treinado para uso, mas apesar de cumprir com o objetivo primário de fornecer um *endpoint* voltado para previsões, a API ainda é muito imatura, oferecendo pouca flexibilidade para a criação de aplicações que a consomem.

Utilizando a API e a versão aprimorada do método ECID, foi desenvolvida a ECID lite: aplicação que permite ao usuário consultar e gerar relatórios sobre as previsões do ECID em diversas datas. Tudo isso de forma bastante simples, sendo necessária apenas a informação da data desejada. Entretanto, essa simplificação trouxe alguns problemas, sendo o maior deles a necessidade de uma base de dados atualizada com frequência que contém as informações necessárias para a previsão. Sem isso, a ECID lite perde parte da sua utilidade como ferramenta de previsão, sendo melhor aplicada como ferramenta de pesquisa para analisar resultados históricos.

Dito isso, existe um claro caminho para aprimoramento do trabalho desenvolvido no formato da criação de uma aplicação capaz de realizar previsões em tempo real ou algo próximo disso; e expansão da API desenvolvida adicionando novas funcionalidades. Outras vertentes que também podem ser consideradas são: melhora do tempo de resposta da aplicação ECID lite implementando um sistema de *cache*; melhora da interface gráfica buscando aprimorar a experiência de usuário; entre outras.

6 - Referências

BAKER, D. N. et al. **A major solar eruptive event in July 2012: Defining extreme space weather scenarios**. Space Weather. v. 11, n. 10, set 2013.

BASU, S. et al. **Specification of the occurrence of equatorial ionospheric scintillations during the main phase of large magnetic storms within solar cycle 23**. Radio Science. v. 45, n. 5, out 2010.

BORBA, M. G.; COUVIDAT, S. **Solar flare prediction using sdo /hmi vector magnetic field data with a machine-learning algorithm**. The Astrophysical Journal. v. 798, n. 2, p. 135, jan 2015.

CARRINGTON, R. C. **Description of a Singular Appearance seen in the Sun on September 1, 1859**. Monthly Notices of the Royal Astronomical Society. Oxford University Press. v. 20, n. 1, p. 13–15, nov 1859.

CHANDRA, A.; YAO, X. **Evolving hybrid ensembles of learning machines for better generalisation.** Neurocomputing. v. 69, n. 7–9, p. 686–700, mar 2006.

DÍSCOLA JR., S. L. **ENHANCING SOLAR FLARE FORECASTING: A MULTI-CLASS AND MULTILABEL CLASSIFICATION APPROACH TO HANDLE IMBALANCED TIME SERIES**. (Doutorado em Ciência da Computação). Universidade Federal de São Carlos. São Carlos, p.179. 2019.

DÍSCOLA JR, S. L. *et al*. **SeMiner: A Flexible Sequence Miner Method to Forecast Solar Time Series**. Information. v. 9, n. 1, 2018.

FACELI, K. *et al*. **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. 1. ed. Rio de Janeiro: LTC, 2011.

FOWLER, M. **Richardson maturity model**. Martin Fowler website, 2010. Disponível em: <https://martinfowler.com/articles/richardsonMaturityModel.html>. Acesso em: 28 out. 2021.

FRANK, E. et al. **The WEKA Workbench**. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques". 4 ed. [s.l.]:Morgan Kaufmann, 2016.

GREEN, J. L.; ODENWALD, S. F. **Bracing for a Solar Superstorm**. Scientific American. v. 299, n. 2, p. 80-87, ago 2008.

HALE, G. E. *et al*. **The magnetic polarity of sun-spots**. The Astrophysical Journal. v. *49*, p.153, 1919.

HAWKINS, D. M. **The Problem of Overfitting**. Journal of Chemical Information and Computer Sciences. v. 44, n.1, p. 1–12, dez 2003.

HOLMAN, G. D. **The Mysterious Origins of Solar Flares**. Scientific American. v. 294, n. 4, p. 38–45, abr 2006.

IBM. **Bagging**. IBM Cloud Education, 2021. Disponível em: <https://www.ibm.com/cloud/learn/bagging>. Acesso em: 4 abr. 2022.

KISS, T. **CodeMetrics**. JetBrains Marketplace, 2019. Disponível em: <https://plugins.jetbrains.com/plugin/12159-codemetrics>. Acesso em: 5 abr. 2022.

LLOYD'S. 2013. **Solar storm risk to the North American electric grid**. Disponível em : <https://assets.lloyds.com/assets/pdf-solar-storm-risk-to-the-north-american-electric-grid/1/pdf-Solar-Storm-Risk-to-the-North-American-Electric-Grid.pdf>. Acesso em: 4 abr. 2022.

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The Art of Software Testing**. 3. ed. New York: Wiley, 2011.

OLIVEIRA, M.; TORGO, L. **Ensembles for Time Series Forecasting**. Proceedings of the Sixth Asian Conference on Machine Learning. v. 39, p. 360–370, 2015.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. 1. ed. San Mateo: Morgan Kaufmann Publishers Inc., 1993.

TANDBERG-HANSSEN, E.; EMSLIE, A. G. **The Physics of Solar Flares**. 1. ed. Nova Iorque: Cambridge University Press, 2009. 288 p. ISBN 9780521115520.

TOPINKA, T. **Statistic**. JetBrains Marketplace, 2009. Disponível em: <https://plugins.jetbrains.com/plugin/4509-statistic>. Acesso em: 5 abr. 2022.

TSURUTANI, B. T. et al. **A brief review of “solar flare effects” on the ionosphere**. Radio Science. v. 44, n. 1, fev 2009.

ZHANG, C.; MA, Y. **Ensemble Machine Learning: Methods and Applications**. 1. ed. Boston: Springer, 2012

Apêndice A

Lista de datas válidas para a aplicação ECID lite:

2010.05.07	2010.12.19	2011.05.14
2010.05.08	2010.12.20	2011.05.15
2010.05.28	2010.12.24	2011.05.16
2010.05.29	2010.12.31	2011.05.21
2010.06.04	2011.01.01	2011.05.22
2010.06.05	2011.01.02	2011.05.23
2010.06.11	2011.01.03	2011.05.27
2010.06.17	2011.01.07	2011.05.28
2010.06.18	2011.01.08	2011.05.29
2010.06.19	2011.01.09	2011.06.03
2010.06.24	2011.01.10	2011.06.04
2010.06.25	2011.01.14	2011.06.05
2010.06.26	2011.01.15	2011.06.06
2010.07.01	2011.01.16	2011.06.07
2010.07.02	2011.01.28	2011.06.11
2010.07.03	2011.01.29	2011.06.17
2010.07.08	2011.01.30	2011.06.18
2010.07.16	2011.01.31	2011.06.19
2010.07.17	2011.02.04	2011.06.20
2010.07.28	2011.02.05	2011.06.24
2010.07.29	2011.02.06	2011.07.01
2010.07.30	2011.02.11	2011.07.02
2010.07.31	2011.02.12	2011.07.03
2010.08.05	2011.02.13	2011.07.15
2010.08.06	2011.02.14	2011.07.16
2010.08.07	2011.02.18	2011.07.29
2010.08.14	2011.02.19	2011.08.05
2010.10.16	2011.04.15	2011.08.06
2010.10.22	2011.04.16	2011.08.13
2010.10.23	2011.04.22	2011.08.14
2010.11.12	2011.04.23	2011.08.15
2010.11.13	2011.04.24	2011.08.19
2010.11.19	2011.04.25	2011.08.20
2010.11.20	2011.04.29	2011.08.26
2010.11.21	2011.04.30	2011.08.27
2010.11.22	2011.05.06	2011.08.28
2010.11.26	2011.05.07	2011.08.29
2010.11.27	2011.05.08	2011.10.21
2010.12.17	2011.05.09	2011.10.22
2010.12.18	2011.05.13	2011.10.23

2011.10.28
2011.10.29
2011.10.30
2011.10.31
2011.11.04
2011.11.05
2011.11.06
2011.11.07
2011.11.11
2011.11.12
2011.11.13
2011.11.14
2011.11.18
2011.11.19
2011.11.20
2011.11.21
2011.11.25
2011.11.26
2011.11.27
2011.11.28
2011.12.03
2011.12.04
2011.12.10
2011.12.11
2011.12.12
2011.12.17
2011.12.18
2011.12.19
2011.12.25
2011.12.26
2011.12.30
2011.12.31
2012.01.01
2012.01.02
2012.01.06
2012.01.07
2012.01.08
2012.01.14
2012.01.15
2012.01.23
2012.01.27
2012.02.05
2012.02.06
2012.02.10
2012.02.11
2012.02.12
2012.02.13
2012.02.18
2012.04.13
2012.04.14
2012.04.15
2012.04.22
2012.04.23
2012.04.27
2012.04.28
2012.05.12
2012.05.13
2012.05.21
2012.05.25
2012.05.26
2012.05.27
2012.06.01
2012.06.02
2012.06.08
2012.06.15
2012.06.16
2012.06.17
2012.06.23
2012.06.24
2012.06.25
2012.06.29
2012.06.30
2012.07.13
2012.07.14
2012.07.15
2012.07.16
2012.07.21
2012.07.22
2012.07.27
2012.07.28
2012.07.29
2012.07.30
2012.08.03
2012.08.04
2012.08.17
2012.08.18
2012.08.19
2012.08.24
2012.08.25
2012.08.26
2012.08.27
2012.10.26
2012.10.27
2012.10.28
2012.10.29
2012.11.24
2012.11.25
2012.11.26
2012.12.01
2012.12.02
2012.12.07
2012.12.14
2012.12.23
2012.12.24
2012.12.28
2012.12.29
2012.12.30
2012.12.31
2013.01.04
2013.01.05
2013.01.11
2013.01.12
2013.01.18
2013.01.19
2013.02.01
2013.02.02
2013.02.03
2013.02.04
2013.02.08
2013.02.09
2013.02.17
2013.02.18
2013.02.22
2013.04.13
2013.04.14
2013.04.19
2013.04.26
2013.05.03
2013.05.04
2013.05.05
2013.05.10
2013.05.11

2013.05.12
2013.05.17
2013.05.18
2013.05.19
2013.05.24
2013.05.25
2013.05.31
2013.06.01
2013.06.06
2013.06.07
2013.06.14
2013.06.15
2013.06.16
2013.06.22
2013.06.23
2013.06.24
2013.06.28
2013.06.29
2013.06.30
2013.07.05
2013.07.06
2013.07.07
2013.07.08
2013.07.12
2013.07.13
2013.07.19
2013.07.28
2013.08.03
2013.08.09
2013.08.10
2013.08.11
2013.08.16
2013.08.17
2013.08.23
2013.08.24
2013.08.25
2013.10.18
2013.10.19
2013.10.20
2013.10.25
2013.10.26
2013.10.27
2013.11.01
2013.11.02
2013.11.08
2013.11.09
2013.11.10
2013.11.15
2013.11.16
2013.11.17
2013.11.22
2013.11.23
2013.11.29
2013.12.06
2013.12.07
2013.12.08
2013.12.09
2013.12.13
2013.12.14
2013.12.15
2013.12.21
2013.12.22
2013.12.23
2013.12.27
2013.12.28
2013.12.29
2014.01.04
2014.01.10
2014.01.11
2014.01.17
2014.01.18
2014.01.19
2014.02.14
2014.02.15
2014.02.16
2014.04.13
2014.04.18
2014.04.19
2014.04.24
2014.04.25
2014.05.02
2014.05.03
2014.05.23
2014.05.24
2014.05.25
2014.05.30
2014.06.06
2014.06.07
2014.06.08
2014.06.13
2014.06.27
2014.07.11
2014.07.12
2014.07.13
2014.07.18
2014.07.19
2014.07.20
2014.07.27
2014.08.01
2014.08.02
2014.08.03
2014.08.08
2014.08.22
2014.08.23
2014.08.24
2014.10.18
2014.10.31
2014.11.01
2014.11.07
2014.11.08
2014.11.09
2014.11.14
2014.11.15
2014.11.28
2014.11.29
2014.11.30
2014.12.05
2014.12.06
2014.12.07
2014.12.21
2014.12.27
2014.12.28
2015.01.02
2015.01.09
2015.01.10
2015.01.17
2015.01.18
2015.01.23
2015.01.24
2015.01.30
2015.01.31
2015.02.01

2015.02.06
2015.02.07
2015.02.08
2015.02.20
2015.04.25
2015.04.26
2015.04.27
2015.04.28
2015.04.29
2015.04.30
2015.05.01
2015.05.08
2015.05.09
2015.05.10
2015.05.15
2015.05.16
2015.05.17
2015.06.12
2015.06.13
2015.07.03
2015.07.04
2015.07.05
2015.07.11
2015.07.12
2015.07.17
2015.07.24
2015.07.25
2015.07.26
2015.08.01
2015.08.02
2015.08.07
2015.08.08
2015.08.09
2015.08.21
2015.08.22
2015.10.18
2015.10.23
2015.10.24
2015.10.25
2015.10.30
2015.10.31
2015.11.01
2015.11.06
2015.11.13
2015.11.20
2015.11.21
2015.11.27
2015.11.28
2015.11.29
2015.12.05
2015.12.06
2015.12.11
2015.12.12
2016.01.01
2016.01.08
2016.01.09
2016.01.15
2016.01.16
2016.01.22
2016.01.23
2016.01.24
2016.02.05
2016.02.06
2016.02.07
2016.02.13
2016.04.15
2016.04.16
2016.04.29
2016.05.21
2016.05.27
2016.05.28
2016.06.03
2016.06.04
2016.06.17
2016.06.18
2016.06.19
2016.06.24
2016.06.25
2016.07.01
2016.07.02
2016.07.08
2016.07.09
2016.07.15
2016.07.16
2016.07.22
2016.07.23
2016.07.29
2016.07.30
2016.10.22
2016.11.06
2016.11.11
2016.11.12
2016.11.13
2016.11.18
2016.11.19
2016.11.20
2016.11.25
2016.11.26
2016.11.27
2016.12.03
2016.12.04
2016.12.18
2016.12.23
2016.12.24
2016.12.25
2016.12.30
2016.12.31
2017.01.01
2017.01.06
2017.01.07
2017.01.20
2017.01.21
2017.01.22
2017.01.28
2017.01.29
2017.02.03
2017.04.15
2017.04.16
2017.04.21
2017.04.22
2017.04.23
2017.04.28
2017.04.29
2017.04.30
2017.05.20
2017.05.21
2017.05.26
2017.05.27
2017.05.28
2017.06.02
2017.06.03
2017.06.04

2017.06.09	2017.07.09	2017.10.27
2017.06.16	2017.07.14	2017.10.28
2017.06.17	2017.07.15	2017.10.29
2017.06.18	2017.07.16	2017.11.17
2017.06.24	2017.08.04	2017.11.18
2017.06.25	2017.08.05	2017.11.19
2017.06.30	2017.10.20	2017.11.24
2017.07.01	2017.10.21	2017.11.25
2017.07.08	2017.10.22	2017.11.26