Metodologia para Aplicabilidade de Jurimetria na Prova da Probabilidade do Direito

Renan Winter Spatin[1], Sandro Roberto Fernandes[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais – *Campus* Juiz de Fora

[2]Núcleo de Informática - Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais – *Campus* Juiz de Fora

rwspatin@gmail.com, sandro.fernandes@ifsudestemg.edu.br

Abstract. *This article aims to present a scientific method in order to demonstrate the applicability of jurimetrics in law probability proof through Artificial Intelligence (AI) using a jurisprudence database from Minas Gerais Court to the training of AI, which contains information such as matter class, decision date, matter location, among others, and it was possible to have a satisfactory result.*

Resumo. *Este artigo tem como objetivo apresentar uma metodologia com intuito de demonstrar a aplicabilidade da jurimetria na prova da probabilidade do direito através de Inteligência Artificial (IA) utilizando um banco de dados de jurisprudências do Tribunal de Justiça de Minas Gerais para o treinamento, o qual possui informações como classes processuais, data de decisão, comarca entre outros, onde foi possível obter resultados satisfatórios.*

1. Introdução

A Inteligência Artificial vem sendo um potencializador de muitas áreas de estudo, auxiliando, por exemplo, Big Data, automação de atividades, análises de doenças baseadas em imagens de radiografia, economia e investimentos, predizendo uma possível valorização ou desvalorização de ações, entre outros. No Direito também há promissoras aplicações da Inteligência Artificial (AI) como melhoria de processos e auxílio no trabalho dos profissionais. O advogado americano Lee Loevinger, já antecipava o uso da Inteligência Artificial no Direito. Em seu livro '*Jurimetrics--The Next Step Forward*' escrito em 1949 citou pela primeira vez o termo "*jurimetria"* como a aplicação de técnicas computacionais no Direito e baseando este

conceito na Econometria (aplicação de estatística nos estudos de Economia). Por isso é considerado por alguns como "pai da Jurimetria", citado por Rodrigo Garcia em '*La Jurimetria - Una Breve Aproximación*' (2003) destacando como data de surgimento da "jurimetria" a data do livro escrito por Loevinger.

A Jurimetria, de forma simples, pode ser considerada a aplicação de técnicas estatísticas no ramo do Direito. De acordo com Karina Moacyr Jr "a Jurimetria interpreta o Direito através de porcentagens obtidas com o estudo e coleta de informações sobre casos pretéritos, escolhidos de forma aleatória" (MOACYR, 2019, p. 113). Umas das formas de aplicação da Jurimetria no universo jurídico que pode ser citada, ocorre nos casos enquadrados nas Tutelas Provisórias do Código de Processo Civil. A aplicabilidade desta tecnologia pode ser utilizada para o cumprimento de um dos requisitos destas tutelas conhecido como *fumus boni iuris*, também conhecido como Probabilidade do Direito.

As Tutelas Provisórias no Universo Jurídico surgiram com o Novo Código de Processo Civil de 2015, sendo distribuída entre os artigos 294 e 311, trazendo consigo a possibilidade do adiantamento de uma tutela do juiz que, em regra geral, somente seria dada no fim do processo. Este processo judicial tem o objetivo de alcançar uma "adequada distribuição do ônus do tempo no processo" (MARINONI et al, 2015, p. 195). Dentre os artigos que abrangem este tema, é importante a análise do art. 300 CPC, onde são demonstrados os requisitos essenciais para as Tutelas Provisórias sendo eles a Probabilidade do Direito (*fumus boni iuris*) e o perigo de dano ou risco do resultado útil do processo.

Dentre as possíveis situações que exigem a necessidade da aplicação das Tutelas Provisórias podemos citar os casos de família referente a pensão alimentícia. Este tipo de processo decorre do não pagamento deste auxílio financeiro pela parte que não possui a guarda da(s) criança(s) em questão. O que pode causar um endividamento da outra parte para que consiga sustentar a casa e as necessidades de saúde e educação que uma criança deve possuir. Este cenário exige uma urgência na decisão do juiz para que não exista uma situação que prejudique a família. Porém, apesar da urgência, quando existe somente as fases processuais comuns, é sabido pelo "homem médio" que existe uma demora até chegar na fase final do processo e com isso esta família pode ter seu bem estar prejudicado até chegar ao fim do julgamento. As Tutelas vem para resolver essas situações tornando possível no início do processo ocorrer um julgamento prévio garantindo assim a pensão alimentícia no cenário descrito até que exista a decisão final.

Quando nos referimos a fase de análise do requisito da prova da Probabilidade do Direito, para um melhor entendimento, pode-se levar em consideração o entendimento de

Schaiane da Silva e Yasmin Zanoni onde descrevem que a análise deste requisito pelos juízes deverá ser baseada "não dá certeza de que o direito existe, mas de mera aparência", sendo assim o mesmo deverá se convencer da existência do provável do direito. Tal qual a descrição utilizada por Silva e Zanoni pode-se analisar a definição utilizada por Caroline de Souza em que afirma que a Probabilidade do Direito "nada mais é do que a demonstração da probabilidade de existência do direito da parte" (SOUZA, p.5).

Dentre os requisitos da Tutela Provisória, como já anteriormente citado, este trabalho foca no requisito do *fumus boni iuris*, validando tecnicamente a teoria defendida no TCC "Utilização de Jurimetria na Prova da Probabilidade do Direito", onde é discutido o cenário de Tutela Provisória que possibilita um julgamento prévio à decisão final do juiz. Para que seja válida essa possibilidade há a necessidade de existir dois requisitos essenciais como citados anteriormente, que são o "Perigo de Dano" e a "Prova da Probabilidade do Direito". A teoria defendida se refere ao método utilizado para comprovar a Probabilidade do Direito, devido ao fato de no cenário atual ser demonstrado através de discussões meramente textuais envolvendo doutrinas, jurisprudências, entre outros métodos existentes no universo jurídico. Com isso o trabalho demonstrou a possibilidade de utilização da Jurimetria na prova deste requisito, trazendo uma análise de uma quantidade de dados maior do que uma pessoa conseguiria analisar e, gerando desta forma, qual o resultado mais provável para um determinado caso concreto.

Diante do exposto, o presente trabalho tem por objetivo responder ao questionamento da viabilidade técnica da teoria defendida pelo autor no Bacharelado em Direito com tema de "Utilização de Jurimetria na Prova da Probabilidade do Direito", comprovando a aplicabilidade da Jurimetria no Direito no setor de Tutelas no quesito prova da Probabilidade do Direito.

Para comprovar a teoria, foi gerada uma base de dados oriunda de Jurisprudências online. Foi utilizada a linguagem Python com Flask para a leitura dos dados e o retorno da resposta solicitada. Para que seja possível uma visualização mais "amigável" deste processo, foi desenvolvida uma interface para uso da API que está publicada no Heroku, uma plataforma em nuvem que suporta a linguagem de programação utilizada. Utilizando esta API, é possível, dado os *inputs*, mostrar qual a resolução mais provável entre Procedente e Improcedente, sendo estas resoluções os julgamentos possíveis dos juízes podendo variar entre um julgamento positivo (procedente) ou um julgamento negativo (improcedente).

2. Metodologia

O processo de viabilização técnica da proposta necessitou de algumas aplicações que possibilitasse a geração de uma base de dados para que posteriormente fosse criado uma estrutura de IA com o objetivo de analisar a Probabilidade do Direito. Estas aplicações foram divididas em soluções para captura de dados e sua inserção em um banco de dados, solução de Inteligência Artificial que leia este banco de dados e gere arquivos com os *encoders* e com o modelo de IA para possibilitar que a API em *Python* com *Flask,* leia esses dados e seja capaz de a partir de um *request* que retorne a resposta prevista mais provável, sendo todas estas soluções versionadas através do GitHub dentro de uma organização criada com o nome Jurisprudence.

2.1 – Visão Geral

A imagem abaixo representa o diagrama da solução criada onde o item 1, *Data Capturing Structure on Heroku*, representa o conjunto de soluções de captura de dados para geração do dataset. Este conjunto de soluções é composto de um agendador em Node.js, um sistema de crawling em .Net Core 3.1 dentro de um container Docker. Uma solução para salvamento dos dados capturados pela solução de crawling feita em Node.js e os salvando em um banco Postgresql, onde todas estas soluções comunicam entre si através do sistema de mensageria do RabbitMQ. Todas estas soluções deste item foram publicadas no Heroku para que o funcionamento e o processo de captura de dados ocorresse de forma efetiva. Para que os códigos fossem constantemente atualizados neste ambiente, foi criado um sistema de CI/CD para cada uma das soluções onde a cada commit enviado para a branch principal o sistema era automaticamente republicado.

O item 2, IA, representa o modelo de Inteligência Artificial criado em Júpiter Notebook que lê o dataset gerado anteriormente, faz os tratamentos necessários nos dados e salva os modelos treinados e os encoders. Os modelos e os encoders são utilizados posteriormente no item 3, API. Desta forma é possível utilizar de forma mais rápida a metodologia criada para entregar as respostas das requisições feitas através do swagger, tanto nos testes feitos quanto no uso real da metodologia.

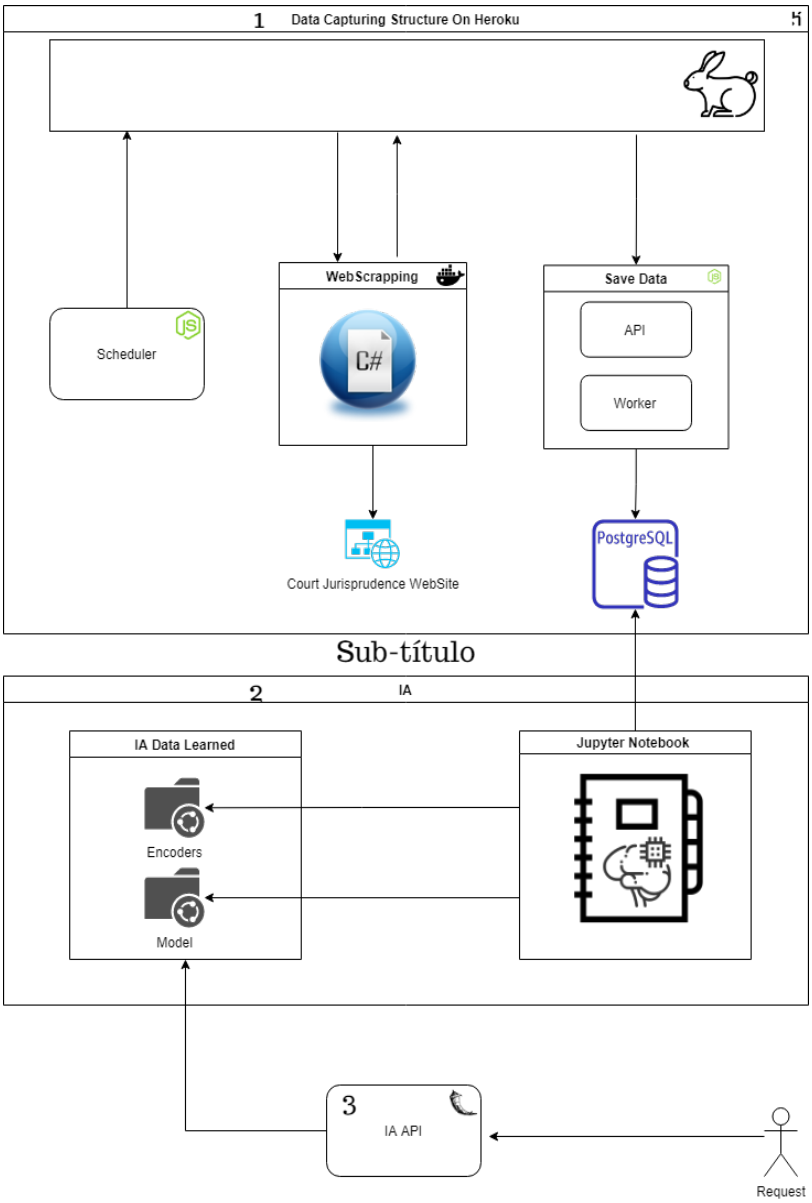

Imagem 01 - Diagrama Estrutural da Solução – Visão Geral. Fonte: O Autor.

As soluções de Agendador (nome da solução: *schedule-jurimetry-data*) que é responsável por efetuar uma atualização constante dos dados sem a necessidade de interação de uma pessoa, mantendo assim, o dado atualizado diariamente para que o algoritmo de IA ser sempre o mais condizente com o cenário atual; Crawling (nome da solução: *drivers-jurimetry-data*); API de Salvamento de dados (nome da solução: *save-jurimetry-data*) e API de IA (nome da solução: *jurimetry-predict-api*) foram todas publicadas no Heroku como citadas anteriormente conforme pode ser visto na imagem abaixo:

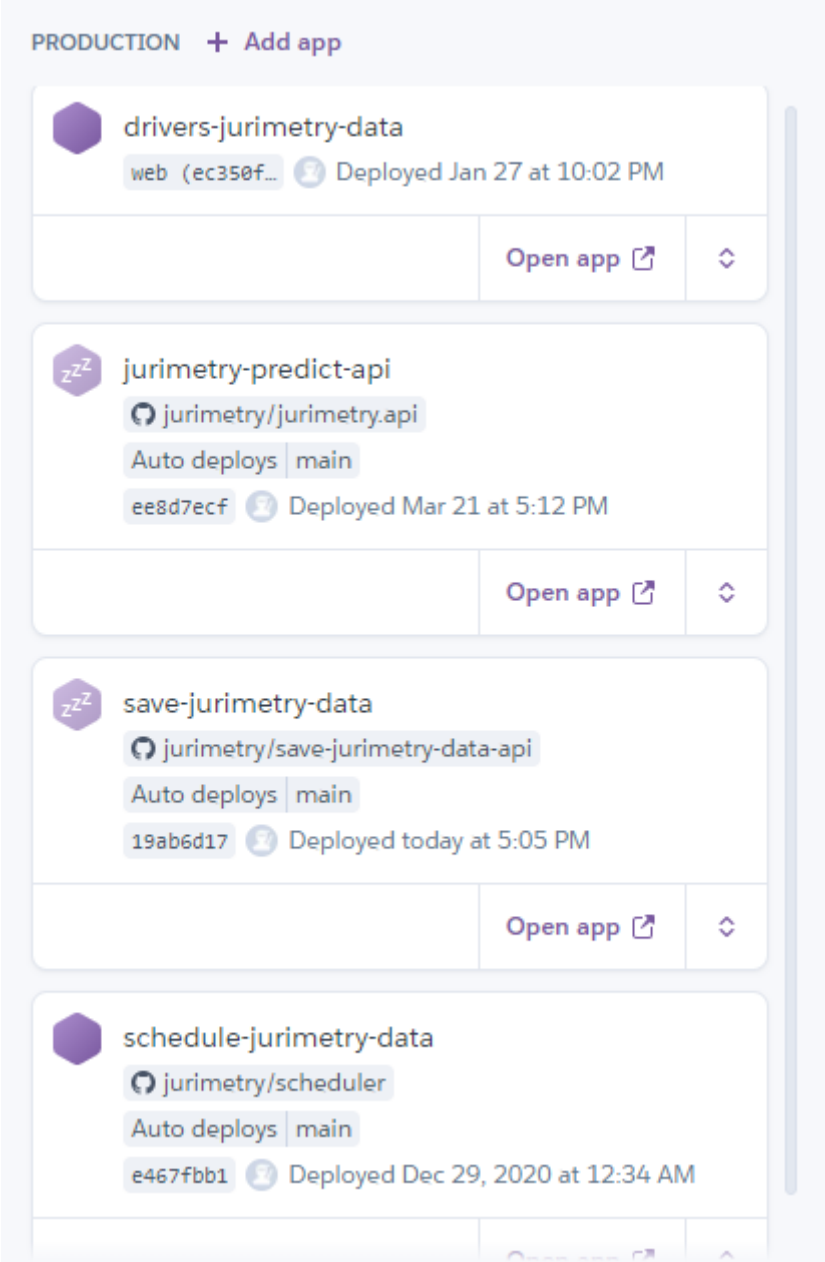

Imagem 02 - Soluções publicadas no Heroku. Fonte: O Autor.

2.2 – Sistema de Captura de Dados

A etapa executada antes do desenvolvimento da API e da ferramenta de front-end, foi a criação da Base de Dados e a análise de qual dado jurídico seria capturado que permitisse seu uso para demonstrar a aplicação da Jurimetria na prova da Probabilidade do Direito. Após algumas análises, foi decidido que o melhor caminho seria a utilização de jurisprudências, esta decisão é justificada pelo motivo de as mesmas serem decisões anteriores que ocorreram em processos possuindo os dados dos pedidos, ou seja, o que gerou o processo e a decisão final.

Dessa forma existem os dados necessários para as análises posteriores pelo algoritmo de IA e como demonstrado por Masha Medvedeva (2019), este surgimento de bases de dados de jurisprudências tornou possível este tipo de análise no contexto jurídico. As mesmas utilizadas nas análises de seu trabalho em cima do Tribunal Europeu..

Para o processo de captura de dados foram criadas três soluções visando que a comunicação entre elas ocorresse de forma simples. Para isso foi utilizado a tecnologia de Mensageria RabbitMQ com intuito de possibilitar uma comunicação assíncrona entre as aplicações tornando a comunicação mais efetiva e evitando tempo de processamento desnecessário das aplicações. Estas soluções foram divididas em um agendador utilizando *Node.Js*[1] com intuito de viabilizar consultas diárias nos sites dos tribunais de forma que as bases de dados estejam constantemente atualizadas com as novas jurisprudências. Para isso a solução do agendador executa o envio de uma solicitação na fila do *RabbitMQ* no canal com nome '*scheduler-jurimetry*' solicitando que a aplicação de captura de dados seja executada imediatamente. Em seguida fica em modo de espera até o mesmo horário no dia seguinte.

```
schedule.scheduleJob('* * * * *', ()=>{
    amqp.connect(url).then((connection) => {
        connection.createChannel().then((channel) => {
            const queueName = 'schedule-jurimetry';
            channel.assertQueue(queueName, {durable: true});
            setInterval(() => {
                var message = {
                    "court": "tjmg",
                    "requiredDate": dataAtualFormatada()
                };

                console.log('send to queue', message);
                channel.sendToQueue(queueName, Buffer.from(JSON.stringify(message)), {
                    persistent: true
                });
            }, INTERVAL_DURATION)
        });
    }).catch((err) => {
        console.error('Connect amqp failed', err);
```

Imagem 03 - Código do Agendador. Fonte: O Autor.

A segunda solução é referente ao processo de captura de informações no site do Tribunal de Justiça de Minas Gerais (TJMG) em sua página de jurisprudências. O site do TJMG foi escolhido devido a suas consultas serem públicas e sem restrições, como é o caso das *captchas*. Foi utilizado o termo de busca "Tutelas" onde encontram-se os requisitos necessários para o tema jurídico objetivo deste trabalho. Para um bom processo de busca foi utilizado a

[1] Node.Js ou Node é um interpretador server-side em JavaScript. - Por Stefan Tilkov p.1

estratégia de combinação entre Órgão Julgador e os Juízes. Em cada órgão julgador foi efetuado a busca de processos vinculados a todos juízes disponíveis no site do Tribunal. Este processo de combinação de busca foi feito utilizando paralelismo no código com um limite de 10 *threads* por execução para que este processo ocorresse de forma mais rápida mas que também não fosse prejudicial ao site.

A solução de captura foi criada em .Net Core 3.1 dentro de um container *Docker*. Este código de captura no site do tribunal é iniciado a partir da mensagem do agendador citado anteriormente, recebendo a requisição no canal '*scheduler-jurimetry*' através do RabbitMQ que indica que o processo de captura deve ser iniciado. Sua execução tem o intuito de capturar dados do site, transformá-los em um formato mais apresentável e enviar cada um dos processos capturados para ser salvo posteriormente utilizando outro canal do *RabbitMQ* chamado '*save-data*' para que seja salvo pela aplicação responsável. Para facilitar o processo de captura foi utilizado uma biblioteca de *crawling* de autoria do autor e seus amigos nomeada de *GorticLib* em que possui facilitadores para o processo captura das informações nos sites disponível em https://www.nuget.org/packages/GorticLib/.

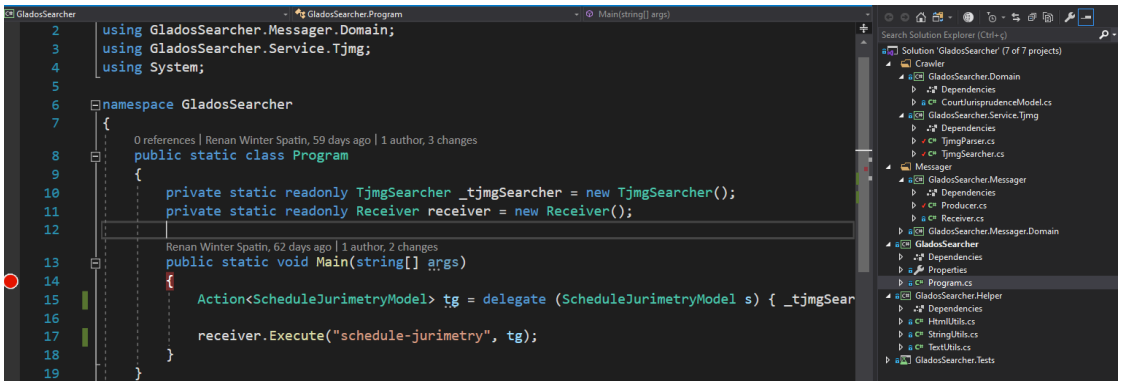

Imagem 04 - Código do Robô de Captura. Fonte: O Autor.

A terceira solução referente a geração dos *datasets* é uma aplicação em *Node.js* que possui uma API e um *Worker* para que seja possível efetuar a inserção dos dados no banco através de duas formas de execução. Uma feita pela API onde ocorre um *request* da API REST e a outra através do Worker, que é um consumer da fila do RabbitMQ com nome '*save-data*'. Ao final de cada processamento esta aplicação salva os dados requisitados em um banco relacional *Postgresql que foi escolhido devido ao fato de ser um banco gratuito e de fácil utilização sendo criada uma instância do mesmo na plataforma Heroku*.

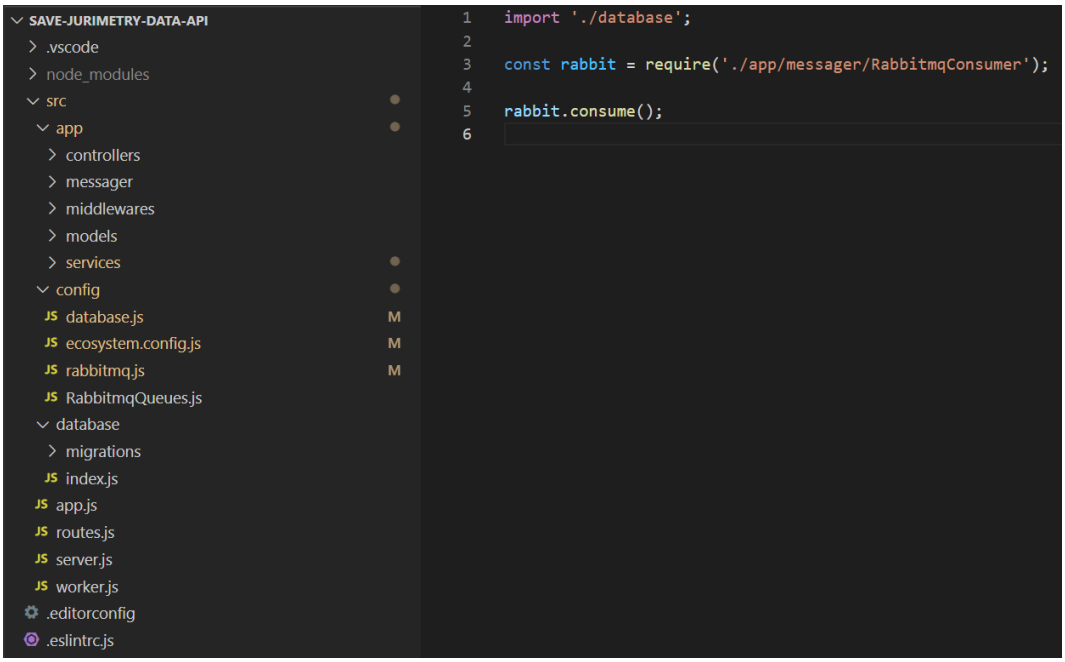

Imagem 05 - Código da API de salvamento de dados. Fonte: O Autor.

As três soluções supracitadas fazem parte do ciclo de captura de dados para geração do *dataset* e foram publicadas em um ambiente do *Heroku,* como citado anteriormente. Para a criação do *Continuous Integration* e *Continuous Delivery* (CI/CD) foi utilizado o *GitHub Actions* de forma que o código seja constantemente atualizado sempre houve alterações na *branch* principal (*branch main* no caso do *GitHub*).

2.3 – IA

Após o desenvolvimento do *Dataset* foi iniciado a criação da Inteligência Artificial. Como primeiro passo foi criado um código no *Jupyter Notebook*[2] com a linguagem *Python* utilizando a biblioteca *matplotlib* para analisar na distribuição de dados qual variação existente nas classes processuais e nos juízes.

A aplicação do *Jupyter Notebook* foi utilizada devido a facilidade na criação de códigos em sua interface, facilitando a execução de blocos de códigos separadamente. A biblioteca do *Matplotlib*[3] foi utilizada para geração gráfica dos dados a serem analisados.

A distribuição dos dados pode ser vista na Imagem 06 abaixo:

---

[2] Jupyter Notebook é uma aplicação web open-source que permite a criação e compartilhamento de documentos que contêm códigos em tempo real, equações, visualizações e textos narrativos. - por jupyter.org

[3] Matplotlib é uma biblioteca fácil de compreender para criação de visualizações estáticas, animadas e interativas em Python - Por matplotlib.org

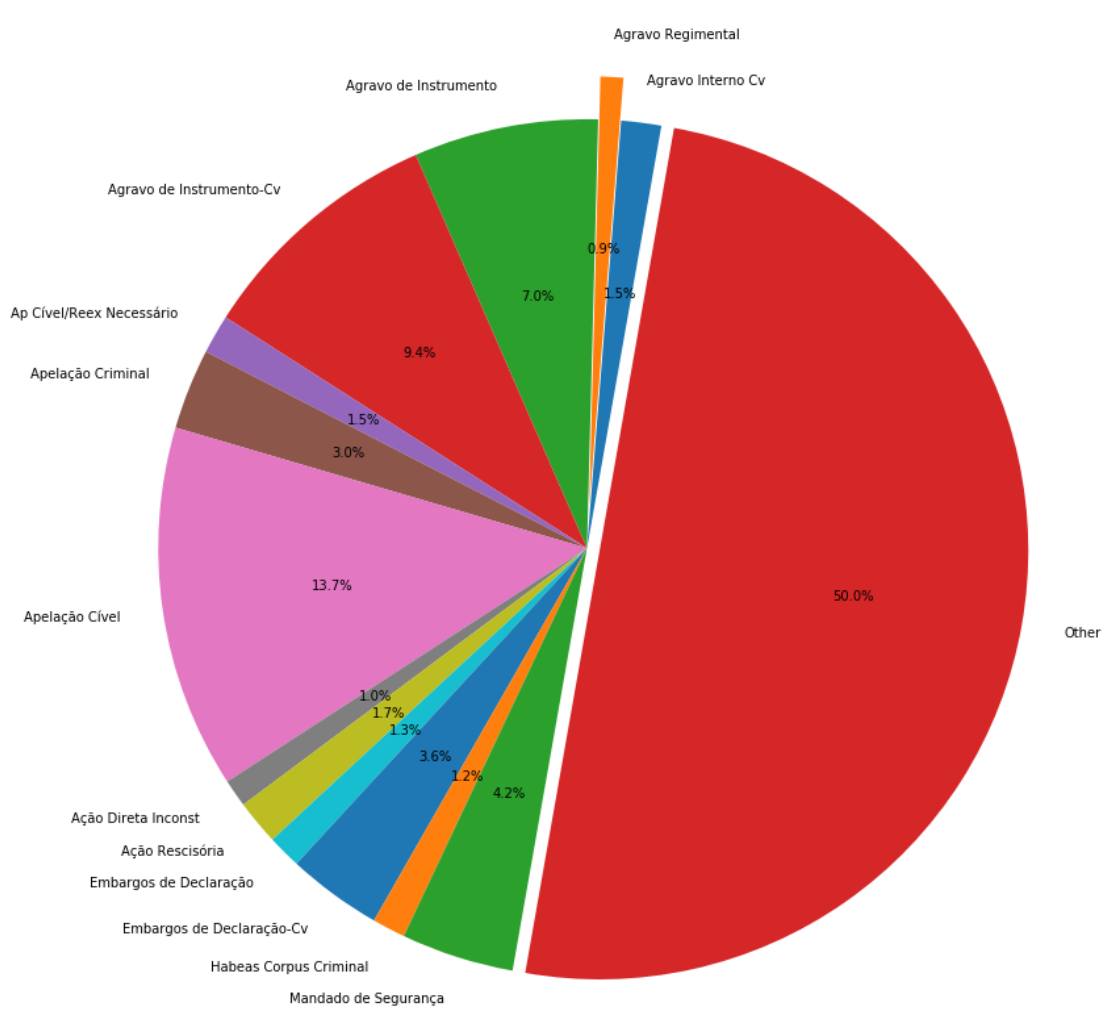

Imagem 06 - Gráfico de Pizza de distribuição das classes processuais encontradas na Base de Dados. Fonte: O Autor.

Analisando a Imagem 06 é possível verificar a quantidade variada de classes processuais, o que garante uma variedade satisfatória para a análise da IA posteriormente. Estas classes processuais podem ser consideradas categorias das fases processuais, assim um processo com classe de "Embargos de Declaração" quer dizer que está ocorrendo o recurso do tipo Embargo de Declaração no processo analisado.

Para que estes dados capturados fossem incluídos na base de dados criada em *Postgresql*[4], foi feito um pré-processamento que eliminou informações como data, mantendo somente o ano do processo; e a separação dos dados considerados de entrada e saída. Esse pré-

[4] PostgreSQL é uma base de dados relacional de código aberto com uma forte reputação em performance, confiabilidade. - por postgresql.org

processamento foi necessário para que fosse possível testar mais de um algoritmo de treinamento e avaliar os melhores resultados para *SKLearning*.

Os primeiros testes possuíam resultados aproximados de 30% (trinta por cento) de assertividade. Em uma primeira análise foi verificado a existência de numerosos dados de saída, o que não permitiu efetuar a predição de forma eficiente. Na imagem 07 abaixo está demonstrando uma parte do código feito para a análise através da IA efetuando o treinamento e salvando os modelos

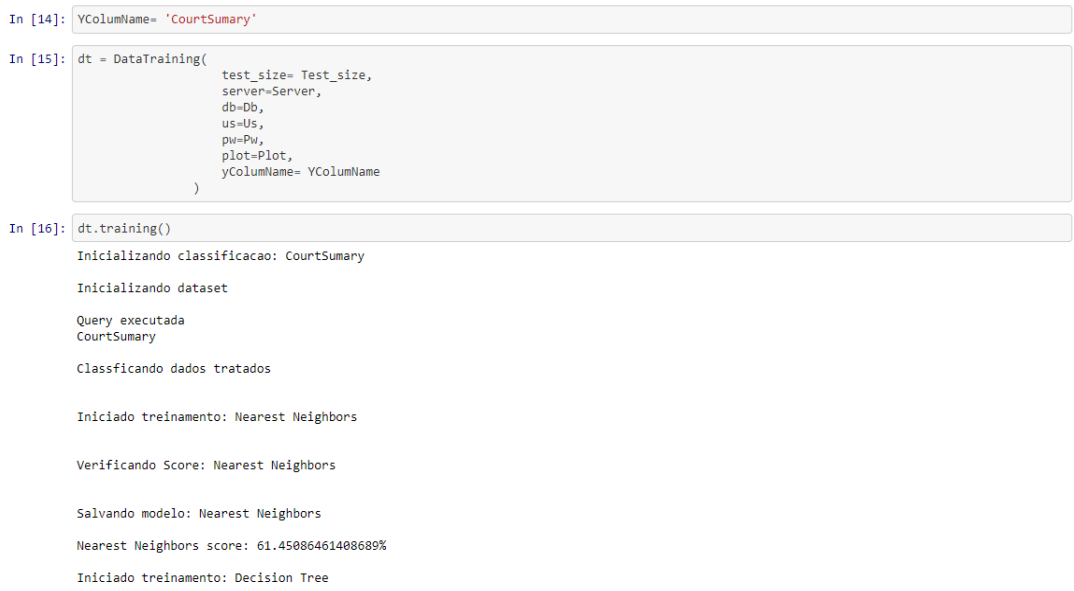

Imagem 07 - Parte do Código do Jupyter Notebook efetuando o treinamento dos modelos de IA. Fonte: O Autor.

Após a análise do causador da assertividade negativa do modelo, foi executado um processo manual para o tratamento de dados capturando as saídas que mais se repetiam. Foi feita a classificação dos dados de “Provido” e “Improvido”, simplificando desta forma a variação de saídas na resposta do algoritmo. Em um novo treinamento foi obtido a classificação de aproximadamente de 48% (quarenta e oito por cento) dos dados. Esse novo resultado é apresentado na Imagem 08 abaixo:

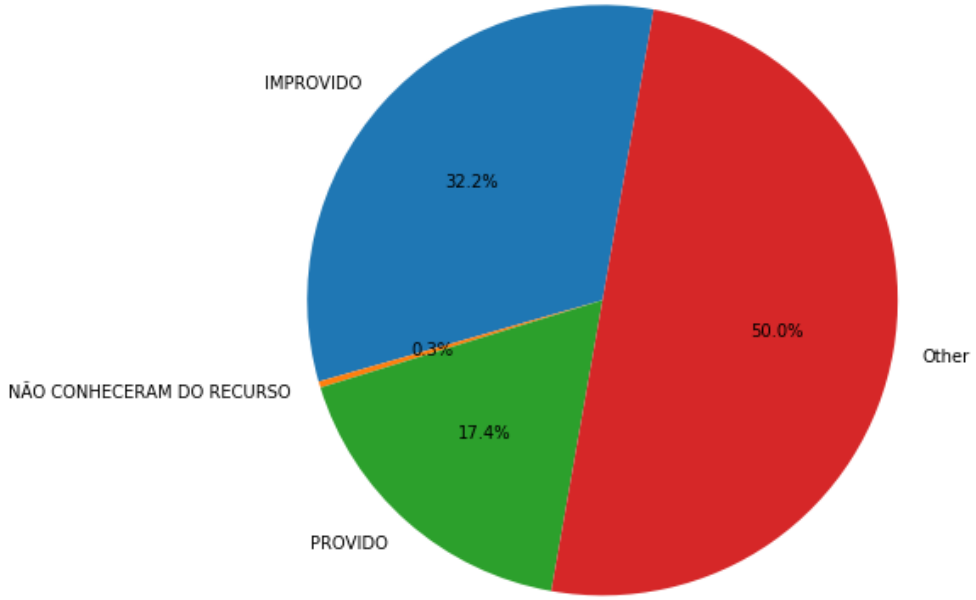

Imagem 08 - Gráfico de Pizza referente à distribuição das decisões dos juízes existentes no Dataset após tratamento de dados e classificação para “Provido” e “Improvido”. Fonte: O Autor.

Com a nova, foi possível alcançar uma assertividade de 45% (quarenta e cinco por cento) sem alterações nos hiperparâmetros do modelo ou no tamanho de 0.33 da divisão do teste e de treino. Este processo foi se repetindo até alcançar uma assertividade com valores acima de 60% (sessenta por cento), o que resultou em respostas mais precisas e serviu para validar o validar a metodologia proposta.

2.4 – Micro-framework e API

Após o desenvolvimento do modelo proposto, visando sua automatização e uso futuro, foi criado uma API em *Python*. Para este desenvolvimento utilizou-se o micro-framework *Flask*. O *Flask* foi escolhido devido a maior integração do *Python* para bibliotecas como Pandas e *SKLear*n, que são usadas pelo modelo para o treinamento e as predições.

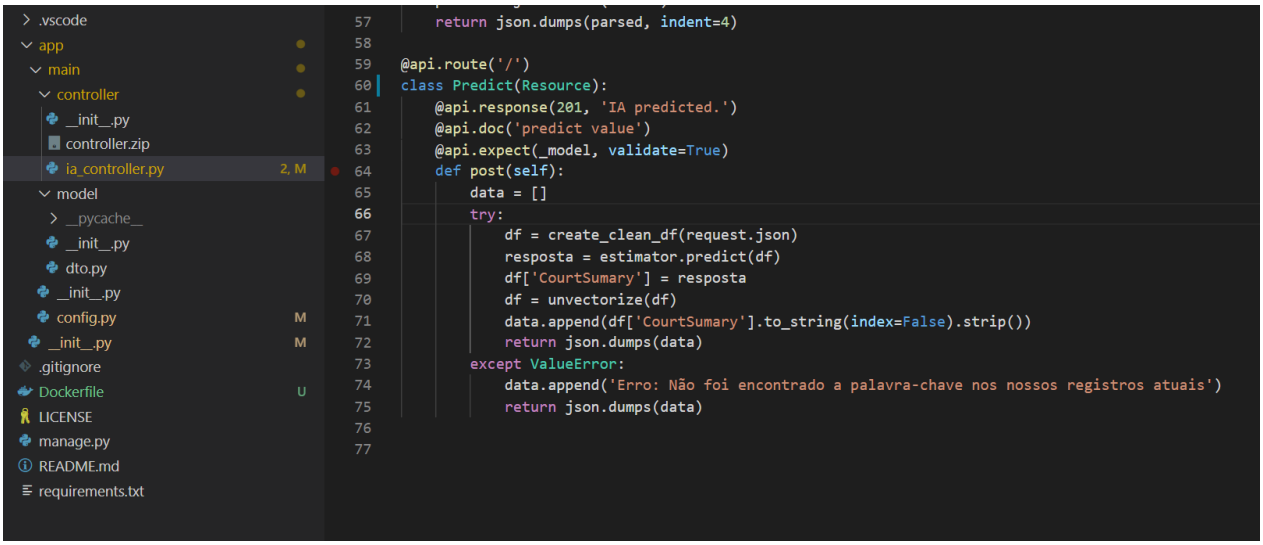

Imagem 09 - Código da API de predições em *Flask*. Fonte: O Autor.

Visando disponibilizar o uso da API de forma mais simplificada, foi adicionado um *Swagger* na API através da biblioteca *Blueprint* do *Python*. O *Swagger* é um framework que independe de linguagem de programação e funciona como um facilitador de testes em APIs REST. Também proporciona ao desenvolvedor uma forma de documentação e mapeamento de todas as requisições disponíveis na API de forma que facilite sua utilização.

Na Imagem 10 abaixo pode ser visto o mapeamento das requisições presentes na API. No POST criado, o nome de *action* é "*predict*". Uma das vantagens do swagger é que, caso seja adicionada uma nova *action,* independente ser em um *controller* distinto, o mesmo será mapeado automaticamente.

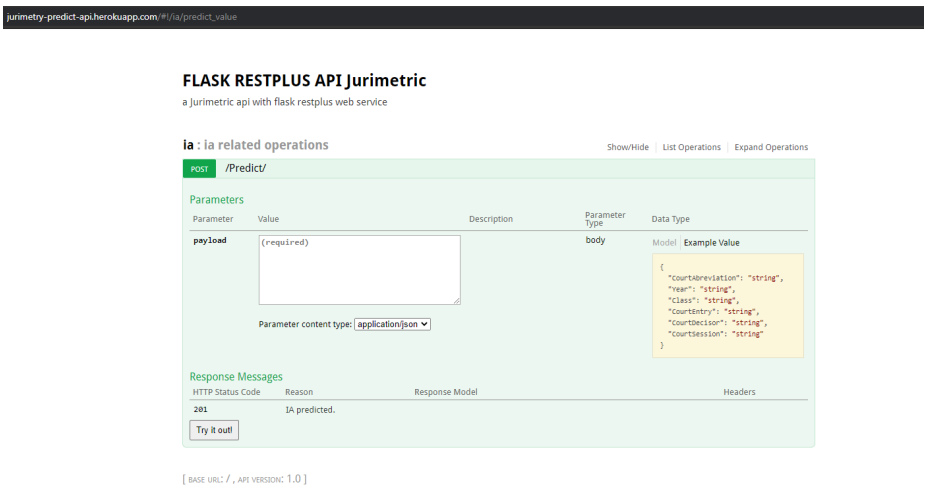

Imagem 10 - *Swagger* da API com a *action* de predição publicada no *Heroku*. Fonte: O Autor.

Na Imagem 10 é possível identificar qual o "*controller"* que a *action* analisada se encontra possuindo o nome "ia", sendo assim, o *Swagger* agrupa as *actions* de acordo com o contexto do "*controller" em* que a mesma está inserida.

Para reduzir o tamanho da API o código foi adaptado para que o modelo e os *encoders* fossem salvos em uma pasta .zip sempre na primeira vez que a API rodar. Após uma publicação o código irá descompactar os arquivos e irá deixá-los em *cache* para uma maior velocidade na resposta.

3. Resultados

Após um tratamento manual dos dados existentes na base de dados, aproximadamente 60% dos dados mais representativos foram utilizados para o treinamento da IA. Desta forma foi alcançada resultados com uma assertividade adequada para o modelo. A biblioteca *SKLearn*, que foi utilizada para o aprendizado de máquina, possui opção de escolha do algoritmo a ser utilizado no processo. Foram escolhidos cinco deles para comparação dos melhores resultados no treinamento: *Random Forest*, *Decision Tree*, *Nearest Neighbors*, *AdaBoost* e *Naive Bayes*. O resultado da assertividade encontrada para cada um dos algoritmos pode ser encontrado na Tabela 01 abaixo:

Tabela 01 – Comparação entre os treinamentos.

Algoritmo	Assertividade (%)
Random Forest	65.31
Decision Tree	64.10
Nearest Neighbors	62.41
AdaBoost	60.83
Naive Bayes	60.57

Para uso posterior do treinamento, cada um dos resultados encontrados foi salvo para uso na API criada. Foi escolhida a extensão "*.save*" para estes arquivos e a extensão "*.plk*" para os *encoders* salvos.

Com a API desenvolvida e com os treinamentos salvos, a mesma foi hospedada na plataforma Heroku com o *Swagger* visando seu uso por terceiros. Esta API se encontra na URL: http://jurimetry-predict-api.herokuapp.com onde ao acessá-la irá carregar automaticamente o *Swagger* da aplicação facilitando assim seu uso. Um exemplo de seu uso pode ser visto nos passos a seguir:

a) Selecione "ia" para visualizar as "*actions*" existentes. Em seguida clique na "*action*" "*Predict*". Após estes passos terá um campo para a entrada de dados escolhida pelo usuário para o teste. Visando facilitar seu uso foi disponibilizado um exemplo do modelo de dados, necessitando somente do preenchimento dos valores nos campos. Isso pode ser visto na Imagem 11 abaixo.

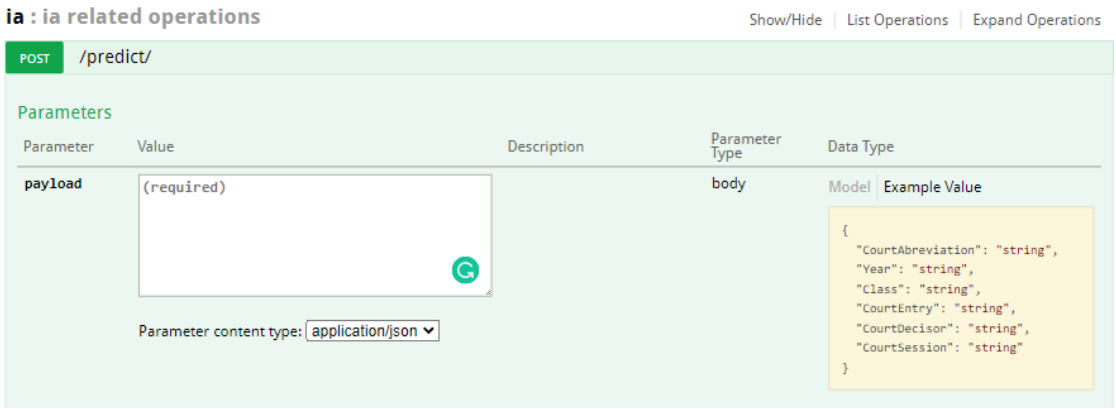

Imagem 11 - *Swagger* com corpo da requisição para efetuar o *request*. Fonte: O Autor.

b) Para este teste utilizamos os valores apresentados na Imagem 12 abaixo, para o *request json*. O "*CourtAbreviation"* representa a sigla do órgão em questão; o "*Year"* representa o ano do processo; o *Class* representa a classe processual; o "*CourtEntry"* se refere aos dados principais do processo, como o caso dos resumos das jurisprudências que pode ser visto facilmente no site Jusbrasil por exemplo; o "*CourtDecisor"* representa o Juiz(a)/Desembargador(a) que está relacionado ao processo; e por último o "*CourtSession"* se refere ao local do processo que no caso do exemplo acima o processo está localizado na 6ª Câmara Cível.

```
{
 "CourtAbreviation": "tjmg",
 "Year": "2005",
 "Class": "Embargos de Declaração",
 "CourtEntry": "EMBARGOS DE DECLARAÇÃO. OMISSÃO. INOCORRÊNCIA. REJEIÇÃO.",
 "CourtDecisor": "Des.(a) *",
 "CourtSession": "Câmaras Cíveis Isoladas / 6ª CÂMARA CÍVEL"
}
```


Imagem 12 – Dados utilizados no exemplo de requisição da API de IA. Fonte: O Autor.

c) Ao executar a API com as informações desejadas, o usuário terá um dos prováveis retornos: Provido, Improvido ou uma mensagem de erro.

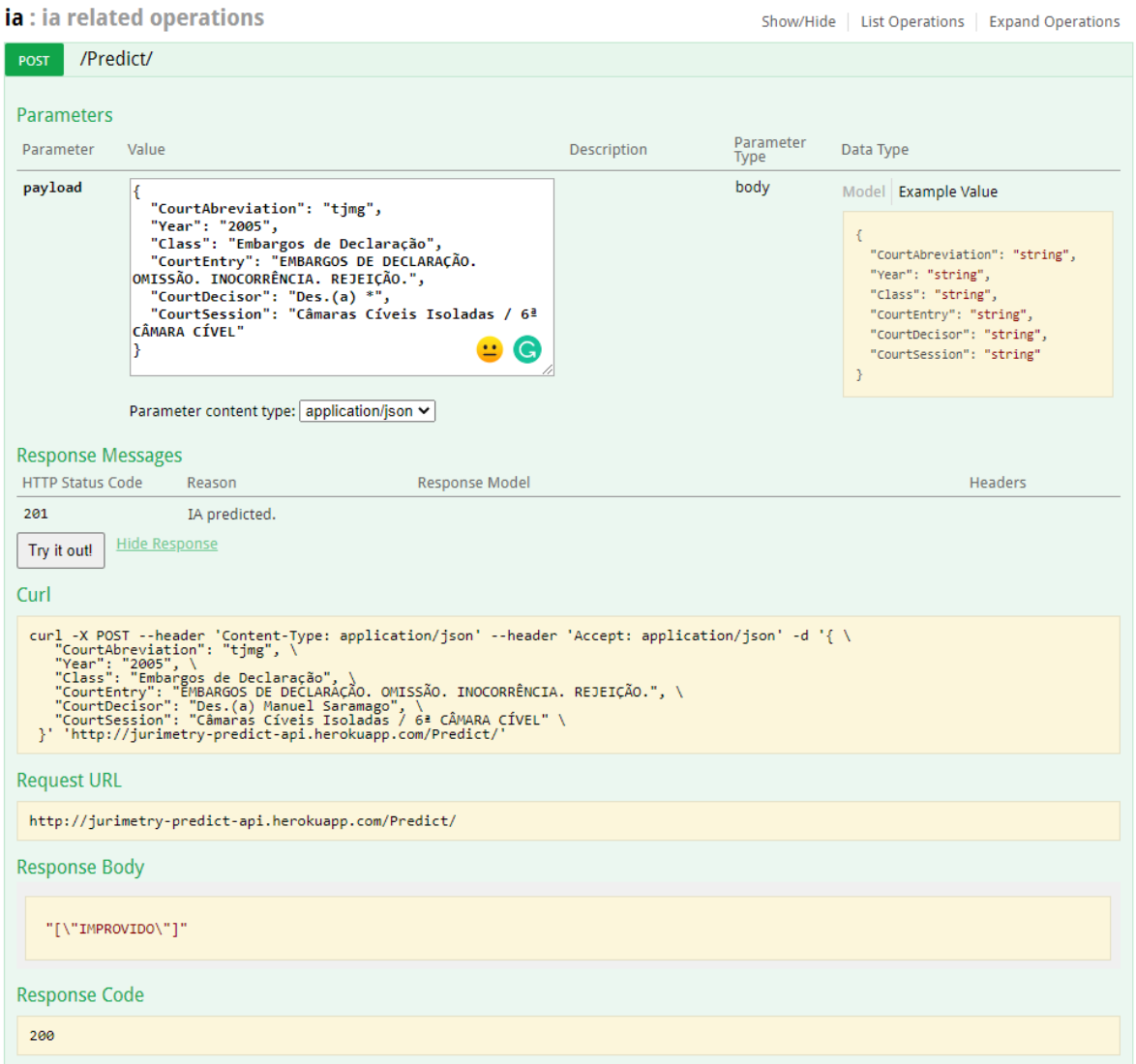

Imagem 13 - Resposta da Requisição de Predição. Fonte: O Autor.

Como pode ser observado na Imagem 13, o resultado para a requisição utilizada como exemplo, foi de que o processo possui uma maior probabilidade de ser Improvido.

Os códigos existentes neste trabalho estão disponibilizados no GitHub visando uma constante atualização e disponibilização para seus usuários. O projeto possui o nome de "*Jurimetry*" e pode ser acessada pelo link https://github.com/jurimetry.

4. Análise dos Resultados

Ao analisar os algoritmos de IA utilizados para o treinamento, percebe-se que, em ordem de assertividade, os três principais foram: *Random Forest*, *Decision Tree* e *Nearest Neighbors*. A princípio foi feito uma tentativa de utilizar o *Random Forest* já que o mesmo possui uma assertividade 1% maior que o segundo colocado, porém na medida que foi necessário adicionar este modelo na API de *Python* criada foi encontrado alguns empecilhos que levaram a análise de se de fato o melhor modelo para este uso seria o *Random Forest*.

A desvantagem encontrada no *Random Forest* foi a de possuir uma variação em seu funcionamento influenciado pela arquitetura do sistema. Ou seja, se for executado em uma arquitetura x86 o mesmo não irá funcionar em uma arquitetura x64. Desta forma foi decidido utilizar o de segunda maior assertividade, *Decision Tree*. A diferença entre a sua assertividade e a do algoritmo *Random Forest*, não é significativa e não possui restrições em relação à arquitetura. Existe uma forma de evitar que este requisito da arquitetura não seja um problema que é a utilização de um *Docker*, porém este fator iria aumentar mais a complexidade do projeto e não era o objetivo do presente trabalho.

Com uma assertividade de 64.1% e possuindo a classificação manual de aproximadamente 60% dos valores presentes no banco de dados com mais de 10.000 (dez mil) jurisprudências, validamos a afirmativa de que é viável uma análise de Probabilidade do Direito, com técnicas de inteligência artificial. Vale ressaltar que quanto maior a quantidade de dados classificados, melhor será a assertividade da predição.

Com relação ao tempo de resposta da API em *Python*, o resultado foi satisfatório. Demonstrando a eficiência da API para os casos de IA. Para os testes foram executadas 100 (cem) requisições para API em o tempo de resposta foi de 50 ms (cinquenta milissegundos), aproximadamente.

5. Conclusões

Através dos resultados obtidos foi possível validar a possibilidade da prova da Probabilidade do Direito através de inteligência artificial demonstrando que é viável tecnicamente e sendo possível alcançar assertividades acima de 60% (sessenta por cento). Foi observado que os tratamentos e classificações manuais executados na base de dados permitiu alcançar resultados mais positivos no treinamento.

O modelo de IA utilizado atualmente não permite que novas inserções, diferentes da base de dados utilizada nos treinos, sejam reconhecidas. Assim seriam necessários novos treinos a cada nova entrada. Este problema foi parcialmente sanado com a estrutura de Agendamentos desenvolvida. Ao final de cada dia, após serem publicadas as novas jurisprudências, as mesmas são adicionadas na base de dados, o que permite com que seja executado novamente o treinamento e os modelos sejam salvos e substituídos na API, possuindo apenas, como fases manuais, os passos de classificação das saídas para "Provido" e "Improvido" para melhoria da assertividade e a execução do treinamento pelo Jupyter Notebook. Desta forma evita-se a supervisão constante e este agendamento contribui para que a assertividade possa aumentar, ou pelo menos se manter.

No contexto jurídico esta ferramenta tem utilidade no contexto do Advogado e no contexto do Juiz/Desembargador. Para o advogado, a ferramenta auxilia no momento em que o mesmo for criar o pedido de uma Tutela Provisória em sua Petição Inicial, onde existe a necessidade de demonstrar uma probabilidade maior de o Pedido ter uma sentença como "Provida" no final do processo. No caso do Juiz/Desembargador, o mesmo poderá utilizar a ferramenta para analisar o pedido feito pelo advogado em sua peça processual, de forma a averiguar a probabilidade de esta Tutela ser correta, dado que a probabilidade de a sentença final ser "Provida" é alta.

Quanto ao contexto de tecnologia este trabalho demonstra uma maneira de predição de dados categóricos textuais com mais de uma entrada e uma saída, possuindo um resultado satisfatório para o mesmo, e, portanto, pode ser utilizado em estudos semelhantes até mesmo fora do contexto jurídico. Além da parte de construção da inteligência, traz um formato de arquitetura que pode ser utilizado para captura de dados e construção de *datasets* com intuito de uma análise posterior destes dados.

6. Trabalhos futuros

Visando alcançar resultados melhores na assertividade o tratamento de dados, feito de forma manual, deve ser feito de forma mais abrangente. Há resultados, como o "Parcialmente Provido" que não foram incluídos dado o volume de dados analisados. Outra solução para esta limitação é a inclusão de uma análise semântica dos resultados, visando tornar o modelo mais dinâmico para incluir "conclusões" diferentes dos processos.

Um estudo exploratório, contendo uma avaliação de algoritmos utilizados para o treinamento de máquina que inclui algoritmos que não foram utilizados neste trabalho. Desta forma pode-se determinar a existência de possíveis desempenhos superiores aos encontrados.

Por fim, a inclusão de técnicas de *Deep Learning* que possam identificar novas palavras ou novas frases, sem a necessidade de novos treinamentos.

Referências

DA SILVA, Schaiane Gauer; ZANONI, Yasmin. **Tutela provisória de urgência e a possibilidade de sua estabilização de acordo com o novo código de processo civil. p.** 364-366

FAVINI, Caroline; DE SOUZA, Maria Carolina Rosa. **Os Pressupostos para a Concessão da Tutela de Urgência e da Tutela de Evidência no Novo Código de Processo Civil**. Acesso em: 27/03/2021. Disponível em: <https://soac.imed.edu.br/index.php/mic/ixmic/paper/viewFile/143/25>

GARCIA, Rodrigo Javier Moya. **La Jurimetría – Una Breve Aproximación**. Revista Chilena de Informática Jurídica, Jurimetria N° 2 (2003) pág.: 5-9. Disponível em: http://repositorio.uchile.cl/bitstream/handle/2250/126835/La-jurimetriauna-breve-aproximacion.pdf?sequence=1

Hartmann, Jochen et al. **Comparing automated text classification methods**. ed. Elsevier B.V.: 2018. Acesso em: 20/02/2021 Disponível em: <https://reader.elsevier.com/reader/sd/pii/S0167811618300545?token=00894E305F35533362050A6685E19FF238B39263FCA022981F1710C88B596711043DF3A3C91C35A4D44EB7EE3694AFD6>

JUPYTER TEAM. The Jupyter Notebook. Acesso em: 03/04/2021. Disponível em: <https://jupyter-notebook.readthedocs.io/en/stable/notebook.html>

LOEVINGER, Lee. **Jurimetrics--The Next Step Forward (1949)**. Minnesota Law Review. 1796. Disponível em: <https://scholarship.law.umn.edu/mlr/1796>

MARINONI, Luiz Guilherme, et al. **Novo curso de Processo Civil**, vol. 2, Revista dos Tribunais: São Paulo, 2015.

MATPLOTLIB DEVELOP TEAM, Matplotlib: Visualization with Python. Acesso em: 03/04/2021. Disponível em: <https://matplotlib.org/>

MEDVEDEVA, Masha, VOLS, Michel, WIELING, Martins. Using Machine Learning to Predict Decisions of the European Court of Human Rights. Artificial Inteligence and Law, 2019.

MOACYR, Karina Reis. **Jurimetria. A Estatística e a Importância da Previsão de Comportamentos No Direito**. 2019. Disponível em: <http://pidcc.com.br/06022019.pdf>

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL: The World's Most Advanced Open Source Relational Database. Acesso em: 03/04/2021. Disponível em: <https://www.postgresql.org/>

TILKOV, Stefan; VINOSKI, Steve. Node.js: Using JavaScript to Build High-Performance Network Programs.Vol. 4, IEEE Internet Computing, 2010, p. 80-83. Acesso em: 02/04/2021. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5617064>.

TRIBUNAL DE JUSTIÇA DE MINAS GERAIS, **Pesquisa por Jurisprudência**. Acesso em: 18/12/2020 Disponível em: <https://www5.tjmg.jus.br/jurisprudencia/sentenca.do>

SPATIN, Renan Winter. **A Aplicação de Jurimetria na Prova da Probabilidade do Direito**. Orientador: Guilherme Madeira. 2019. 15f. TCC (Graduação) - Bacharelado em Direito, Rede de Ensino Doctum, Juiz de Fora, 2019.