

UNIVERSIDADE FEDERAL DO ABC

GRADUAÇÃO EM ENGENHARIA DE INSTRUMENTAÇÃO,  
AUTOMAÇÃO E ROBÓTICA

Thaís Moreira Vaz

**MÉTODOS DE APRENDIZADO DE MÁQUINA PARA  
AUXÍLIO DIAGNÓSTICO DE TRANSTORNO DO ESPECTRO  
AUTISTA**

SANTO ANDRÉ

2021

THAÍS MOREIRA VAZ

## **MÉTODOS DE APRENDIZADO DE MÁQUINA PARA AUXÍLIO DIAGNÓSTICO DE TRANSTORNO DO ESPECTRO AUTISTA**

Monografia apresentada ao curso de Engenharia de Instrumentação, Automação e Robótica, como requisito parcial para obtenção do título de Bacharel em Engenharia de Instrumentação, Automação e Robótica.

Orientadora: Prof<sup>a</sup>. Dr<sup>a</sup>. Marina Sparvoli de Medeiros

SANTO ANDRÉ

2021

## **AGRADECIMENTOS**

Agradeço a todas as pessoas que passaram por meu caminho e me inspiraram a ingressar no ensino superior, essas referências foram essenciais para que eu pudesse chegar até aqui.

Ao Murilo, que me incentivou desde quando a UFABC era um sonho distante para mim.

À minha orientadora, Professora Doutora Marina Sparvoli de Medeiros, que me direcionou no desenvolvimento deste projeto.

Ao Professor Doutor João Ricardo Sato, que me apresentou a base de dados aqui utilizada. Ao Professor Doutor Thiago Covões, que validou e adicionou ideias na parte de aprendizado de máquina.

À Professora Doutora Kátia Franklin Albertin Torres, pela disposição em avaliar este estudo.

À Universidade Federal do ABC, por fornecer uma formação pública, gratuita e de qualidade.

## RESUMO

O transtorno do espectro autista é um distúrbio de neurodesenvolvimento que afeta uma a cada 160 pessoas. O diagnóstico precoce é fundamental para que seja desenhado e executado um plano de tratamento que atenda às necessidades inerentes ao transtorno. O propósito deste trabalho é encontrar um método de aprendizado de máquina para auxílio diagnóstico de pessoas com transtorno do espectro autista. Neste projeto, foi utilizada a base de dados aberta ABIDE, que contém imagens de ressonância magnética funcional de pacientes em *resting state* (rs-fMRI). Alguns *Pipelines* de pré-processamento são disponibilizados para tratamento de dados e para este estudo foram aplicados os Atlas CC200 e CC400 para a identificação e divisão das áreas de interesse em 200 e 400 partes respectivamente. Esses pré-tratamentos são disponibilizados na base de dados pré-processados ABIDE. Foram extraídas as correlações de *Pearson* a partir das séries temporais entre as regiões e obtida a matriz de conectividade funcional. Com isso, aplicada a técnica de vetorização na matriz de conectividade para realizar a transformação linear da matriz e obter o dado 1D para cada observação. Após os tratamentos, a base foi separada entre treino e teste. Com isso, foram testadas algumas técnicas de aprendizado de máquina, aplicando-se a validação cruzada. A primeira técnica executada foi o *Support Vector Machine* com *kernel* linear, obtendo-se uma revocação de 78.05% nos dados de teste, com o atlas CC200. Outras técnicas foram testadas, como a aplicação do TPOT, um algoritmo genético de *autoML*. O TPOT realiza testes de otimização de hiperparâmetros e pré-processamentos, como o método PCA. Em uma execução de 24 horas pelo TPOT com o Atlas CC400, foi obtido um *Pipeline* que resultou em 73.17% de revocação, sendo este o segundo melhor dos 4 métodos empregados. Por fim, foi possível encontrar um modelo e um atlas que trouxesse bom desempenho, sendo alcançado o objetivo de propor um modelo e escolher um atlas que possibilite o auxílio-diagnóstico.

**Palavras-chave:** Aprendizado de Máquina, Inteligência Artificial, Transtorno do Espectro Autista.

## ABSTRACT

Autism spectrum disorder (ASD) is a neurodevelopmental disorder affecting one in 160 people. Early diagnosis is key for the design and execution of treatment plans that meets the inherent needs of the disorder. The proposal of this work is to find a machine learning method for assisting in the diagnosis of people with ASD. This Project used open ABIDE brain images database, which contains functional magnetic resonance images of patients in resting state (rs-fMRI). Available preprocessing Pipelines were used for data treatment, with Atlas CC200 and CC400 being applied to identify and divide areas of interest into 200 and 400 parts, respectively. These previous treatments were available in the pre-processed ABIDE database. Pearson correlations were extracted from the time series between the regions, with a functional connectivity matrix obtained. Vectorization technique was applied in the connectivity matrix to perform its linear transformation and obtaining a 1D data for each observation. After the treatments, the base was separated between training and testing data. Different machine learning techniques were tested using cross-validation. The first technique used was Support Vector Machine with linear kernel, resulting in a recall of 78.05% in the test data with CC200 Atlas. Other techniques were tested, including TPOT, an autoML genetic algorithm. TPOT performs hyperparameter optimization tests and pre-processing such as the PCA method. In a 24 hours long TPOT execution with Atlas CC400, a Pipeline that resulted in 73.17% recall has been found, the second best method among other 4 employed. The results demonstrated a good performance, being achieved the objective of proposing a model and choosing an Atlas for the diagnosis assistance of ASD.

**Keywords:** Machine Learning, Artificial Intelligence, Autism Spectrum Disorder.

## LISTA DE ILUSTRAÇÕES

Figura 1 - Fluxo típico para construção de um modelo de AM. ....	14
Figura 2 - Exemplo do TPOT aplicado. ....	15
Figura 3 - Equipamento de ressonância magnética .....	16
Figura 4 - Características tridimensionais de um voxel.....	17
Figura 5 - Pipelines de análise e processamento da CPAC.....	19
Figura 6 - Representação de nodos e arestas. ....	20
Figura 7 - Pipeline geral. ....	21
Figura 8 - Clusterização para quatro métodos de parcelamento funcionais.....	23
Figura 9 – Pré-processamento e análise de imagens de fMRI.....	25
Figura 10 – Pipeline fMRI.....	27
Figura 11 - Fluxograma do projeto. ....	28
Figura 12 – Matriz de Conectividade da primeira amostra da base Atlas CC200. ...	31
Figura 13 – Matriz de Conectividade da primeira amostra da base com TEA dado pré-processado com Atlas CC400. ....	31
Figura 14 – Separação entre treino e teste. ....	32
Figura 15 – Separação da porção de treino em 5 folds, para validação dos dados. 33	
Figura 16 - 48 anos TEA 29006. ....	34
Figura 17 - 41 anos TEA 29007. ....	35
Figura 18 - 48 anos Controle 29011.....	35
Figura 19 – Distribuição das classes.....	36
Figura 20 – Distribuição das classes por gênero. ....	36
Figura 21 – Distribuição dos dados por idade. ....	37
Figura 22 – Matriz de Confusão do modelo final com a base de dados completa de treino. ....	39
Figura 23 – Matriz de Confusão do modelo final com a base de dados completa de treino. ....	40
Figura 24 – Matriz de Confusão do modelo final com a base de dados completa de treino. ....	42
Figura 25 – Matriz de Confusão do modelo final com a base de dados completa de treino. ....	43

## LISTA DE TABELAS

Tabela 1 – Bibliotecas e configurações utilizadas no estudo. ....	29
Tabela 2 – Tabela de Contingência para análise da distribuição das classes por gênero. ....	37
Tabela 3 - Resultado na validação cruzada. ....	38
Tabela 4 - Resultado do modelo final SVC. ....	38
Tabela 5 - Resultado do Pipeline advindo do TPOT na validação cruzada.....	39
Tabela 6 - Resultado do modelo final do Pipeline advindo do TPOT. ....	40
Tabela 7 - Resultado na validação cruzada. ....	41
Tabela 8 - Resultado do modelo final SVC. ....	41
Tabela 9 - Resultado do Pipeline advindo do TPOT na validação cruzada.....	42
Tabela 10 - Resultado do modelo final do Pipeline advindo do TPOT. ....	43
Tabela 11 – Comparação dos resultados dos modelos. ....	44

## LISTA DE ABREVIATURAS E SIGLAS

TEA	Transtorno do Espectro Autista
ADOS	<i>Autism Diagnostic Observation Schedule</i>
AM	Aprendizado de Máquina
fMRI	<i>Functional Magnetic Ressonance Imaging</i>
DSM	<i>Diagnostic and Statistical Manual of Mental Disorders</i>
ADI-R	<i>Autism Diagnostic Interview-Revised</i>
DISCO	<i>Diagnostic Interview for Social and Communication Disorders</i>
CAD	<i>Computer-Aided Diagnosis</i>
MRI	<i>Magnetic Ressonance Imaging</i>
EEG	<i>Electroencephalogram</i>
ROI	<i>Region of Interest</i>
rs-fMRI	<i>Resting State Functional Magnetic Ressonance Imaging</i>
ABIDE	<i>Autism Brain Imaging Data Exchange</i>
CCS	<i>Connectome Computation System</i>
CPAC	<i>Configurable Pipeline for the Analysis of Connectomes</i>
DPARSF	<i>Data Processing Assistant for Resting-State fMRI</i>
NIAK	<i>Neuroimaging Analysis Kit</i>
PCP	<i>Preprocessed Connectomes Project</i>
INDI	<i>International Neuroimaging Datasharing Initiative</i>
CC200	<i>Craddock 200</i>
CC400	<i>Craddock 400</i>
PCA	<i>Principal Component Analysis</i>
BOLD	Blood Oxygenation Level Dependent
TPOT	<i>Tree-base Pipeline Optimization Tool</i>
AutoML	<i>Automatic Machine Learning</i>
SVC	<i>Support Vector Classifier</i>
VN	Verdadeiro Negativo
FN	Falso Negativo
VP	Verdadeiro Positivo
FP	Falso Positivo



# SUMÁRIO

1	<b>INTRODUÇÃO</b>	9
2	<b>OBJETIVOS</b>	10
2.1	Objetivos Gerais	10
2.2	Objetivos Específicos	10
3	<b>FUNDAMENTAÇÃO TEÓRICA</b>	10
3.1	Transtorno do Espectro Autista	10
3.2	Aprendizado de Máquina	12
3.3	SVC	12
3.4	TPOT	13
3.5	fMRI	15
3.6	Voxel	17
3.7	CPAC	18
3.8	ROI e Conectividade Funcional	20
3.9	Atlas Craddock 200 (CC200) e Atlas Craddock 400 (CC400)	22
3.10	Base de dados	23
4	<b>MATERIAIS E MÉTODOS</b>	25
4.1	Fluxograma	28
4.2	Recursos para replicabilidade	29
4.3	Tratamento dos dados	30
4.4	Matriz de conectividade	31
4.5	Separação dos dados	32
4.6	Construção dos modelos	32
4.6.1	Tpot	33
4.6.2	SVC	33
5	<b>RESULTADOS E DISCUSSÃO</b>	34
5.1	Análise Exploratória	34
5.2	Análise Exploratória dos dados	35
5.3	Resultados Dos Modelos	38
5.3.1	Atlas CC200 com SVC	38
5.3.2	Atlas CC200 com TPOT	39
5.3.3	Atlas CC400 com SVC	40

5.3.4	Atlas CC400 com TPOT .....	42
5.3.5	Comparação dos resultados dos modelos.....	43
6	<b>CONCLUSÕES E PERSPECTIVAS</b> .....	45
	<b>REFERÊNCIAS BIBLIOGRÁFICAS</b> .....	46
	APÊNDICE .....	49
	CÓDIGO EDA CC200/ CC400 .....	49
	CÓDIGO TPOT CC200 .....	53
	CÓDIGO TPOT CC400 .....	55
	CÓDIGOS DE TREINO E ANÁLISE DOS RESULTADOS DO SVC CC200, TPOT CC200, SVC CC400, TPOT CC400 .....	57

## 1 INTRODUÇÃO

O transtorno do espectro autista (TEA) é um distúrbio de neurodesenvolvimento que inclui déficits de comunicação e interação social, com a presença de comportamentos repetitivos, que afeta uma a cada 160 pessoas. Pessoas com autismo tendem a ficar socialmente isoladas devido as dificuldade de atenção e de comunicação que limitam a reciprocidade social (ELSABBAGH et al., 2012; RAHMAN et al., 2020).

Nesse contexto, o diagnóstico precoce do TEA é fundamental para que seja desenhado e executado um plano de tratamento que atenda às necessidades inerentes ao transtorno, a exemplo de terapias voltadas ao desenvolvimento das habilidades comportamentais, com o objetivo de mitigar os impactos da doença e melhorar a qualidade de vida do paciente (ROBINS et al., 2001).

No entanto, os métodos utilizados no diagnóstico baseiam-se principalmente em entrevistas com a criança e seus pais, a exemplo do método ADOS (do inglês, *Autism Diagnostic Observation Schedule*) que é o mais aceito atualmente para TEA. Nesse método, as conversas se iniciam a partir de um ano de idade. Estas entrevistas são realizadas por diversos profissionais, como pediatras, psicólogos, fonoaudiólogos e psiquiatras. Todo o processo depende de experiência médica, descrição correta dos comportamentos dos pais em relação às crianças, estado da criança no momento da entrevista e outras condições circunstanciais, que acarretam no diagnóstico demorado, com fatores pouco precisos (GOTHAM; PICKLES; LORD, 2009; LORD et al., 2018).

O aumento da capacidade computacional e o desenvolvimento de algoritmos de aprendizado de máquina (AM), que possibilitam a análise de grande quantidade de dados e a automatização do estudo de imagens, viabilizaram o diagnóstico mais rápido e preciso e vem favorecendo os estudos voltados a neuroimagens.

O propósito para o desenvolvimento deste trabalho foi utilizar métodos de AM para auxiliar no diagnóstico de pessoas com TEA, por meio de fMRI (do inglês, *Functional Magnetic Resonance Imaging*).

## 2 OBJETIVOS

Aplicar técnicas de AM para auxílio diagnóstico de TEA.

### 2.1 Objetivos Gerais

O propósito deste trabalho é, através de imagens de ressonância magnética funcional, utilizar método de AM para auxílio diagnóstico de pessoas com TEA.

### 2.2 Objetivos Específicos

- Identificar na literatura melhores tratamentos de dados fMRI para aplicação dos dados em técnicas de AM;
- Selecionar e testar métodos para classificação de autismo e determinar o melhor entre TPOT (do inglês, *Tree-based Pipeline Optimization Tool*) e SVC (do inglês, *Support Vector Classifier*);
- Testar essas abordagens de AM com o Atlas CC200 e o Atlas CC400.

## 3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo traz os principais fundamentos para melhor compreensão dos conceitos utilizados neste estudo.

### 3.1 Transtorno do Espectro Autista

O TEA é um transtorno do neurodesenvolvimento caracterizado por deficiências em comunicação social, combinada com atraso de linguagem e comportamento repetitivo (AMARAL; SCHUMANN; NORDAHL, 2008; FALCK-YTTER; VON HOFSTEN, 2011). O termo “espectro” no transtorno do espectro do autismo se refere à ampla faixa de gravidade (DUCHAN; PATEL, 2012).

Comportamentos comuns a um portador de TEA incluem: nervosismo por mudanças pequenas e circunstanciais, forte apego a bens, subestimação do perigo, repetição de frases ou movimentos, ausência de contato visual e tendência a ficar sozinho (LORD et al., 2018).

A suscetibilidade ao TEA não é afetada por etnia, raça ou status socioeconômico. Embora os sintomas de crianças autistas possam surgir no primeiro ano, os portadores geralmente são diagnosticado entre a idade de três a cinco anos, coincidindo com o surgimento de sintomas perceptíveis. Contudo, poucas crianças com TEA se desenvolvem normalmente durante o primeiro ano. A causa exata e cura para transtorno ainda são desconhecidos. (ZWAIGENBAUM et al., 2005; GESCHWIND; LEVITT, 2007).

O diagnóstico precoce e o desenvolvimento de um plano de tratamento personalizado são cruciais para o desenvolvimento de crianças autistas. As terapias educacionais, orientadas a habilidades comportamentais melhoram substancialmente o comportamento autista quando apresentado em uma idade precoce. A observação precoce pelos pais contribui para a redução da taxa de falsos positivos e reduz encaminhamentos desnecessários (ROBINS et al., 2001). No entanto, os métodos mais atuais de diagnóstico de TEA baseiam-se principalmente em entrevistas logo nos primeiros anos de idade (LORD et al., 2018).

Para o diagnóstico de TEA, comumente são utilizados instrumentos comportamentais, como o Cronograma de Observação Diagnóstica de Autismo ADOS (GOTHAM; PICKLES; LORD, 2009), o Manual Americano de Diagnóstico e Estatística, da Associação Psiquiátrica de Transtornos Mentais (DSM)-5 (AMERICAN PSYCHIATRIC ASSOCIATION, 2013), o modelo Entrevista Diagnóstica de Autismo Revisada (ADI-R) (LAM; AMAN, 2007), e o Instrumento Diagnóstico para Distúrbios de Comunicação Social (DISCO) (NICE, 2012). Os instrumentos são utilizados em conjunto com o histórico e julgamento clínico de especialistas, como **pediatras, psicólogos, fonoaudiólogos ou psiquiatras**.

Dentre os métodos citados, o ADOS é considerado hoje o método mais eficaz para o diagnóstico e quantificação dos sintomas de pessoas no espectro. Esse método fornece abordagens padronizadas sob a condução de triagens diretas por profissionais treinados, permitindo a sua reprodutibilidade (GOTHAM; PICKLES; LORD, 2009).

Atualmente, os sistemas objetivos de diagnóstico auxiliado por computador (CAD) estão ganhando mais atenção pela sua potencialidade na compreensão do autismo. Várias técnicas são utilizadas nesses sistemas, como análises genéticas e de sangue, imagens por ressonância magnética (MRI) e dados provenientes de

eletroencefalograma (EEG), aplicando-se análise de dados, sensores e técnicas de visão computacional (ELSABBAGH et al., 2012).

### 3.2 Aprendizado de Máquina

Com o grande aumento na disponibilização de dados de diversas áreas, houve um aumento do interesse pela busca de padrões em grandes bases de dados, havendo uma maior aplicação das técnicas de AM.

Esta área é muito ampla possuindo aplicações em diversos ramos do conhecimento, como em filtragens de spam de e-mails utilizando algoritmos de AM (ASKI; SOURATI, 2016), em sistemas de recomendações (PORTUGAL; ALENCAR; COWAN, 2018) e em outras áreas de pesquisa e também de mercado.

AM é uma área da Inteligência Artificial que visa a criação de técnicas computacionais capazes de adquirir conhecimento de forma automática, ou seja, aprender (ALPAYDIN, 2014). Esta área envolve três principais técnicas: o Aprendizado Supervisionado, o Aprendizado Não supervisionado, e o Aprendizado Semi-supervisionado.

No Aprendizado Não-supervisionado não há conhecimento prévio dos rótulos das instâncias, sendo o objetivo encontrar padrões ou graus de similaridades que descrevam as instâncias de dados de maneira interpretativa. No Aprendizado Semi-supervisionado, o objetivo é procurar padrões nos dados utilizando um número limitado de exemplos rotulados, acrescentando também exemplos de dados não-rotulados. E por fim, Aprendizado Supervisionado, método empregado neste trabalho, visa obter um modelo capaz de prever um rótulo para uma nova instância/observação. Para este estudo é utilizada uma base de dados com neuroimagens com o rótulo TEA ou Controle, com isso a abordagem supervisionada se torna viável e interessante para esse contexto.

### 3.3 SVC

O SVM (do inglês, *Support Vector Machine*), ou SVC (do inglês, *Support Vector Classifier*) para o caso de classificação, é um método ~~algoritmo~~ usado para classificação binária. Esse método pode ser definido como um algoritmo de Aprendizado Supervisionado baseado em risco para classificar padrões de dados

identificando uma fronteira com uma margem máxima entre dados da mesma classe (PINEDA-JARAMILLO, 2019).

O SVC identifica um limite chamado de hiperplano. O hiperplano é uma linha que divide um plano em duas porções em um espaço bidimensional, onde cada classe se encontra em um lado da linha, sendo as classes divididas por essa linha.

Para alguns casos, o SVC classifica os dados projetando a variável resposta para um espaço de alta dimensão, onde as classes são linearmente separáveis.

Esse método foi criado em 1995 por (CORTES; VAPNIK, 1995) e tem sido aprimorado desde então.


### 3.4 TPOT

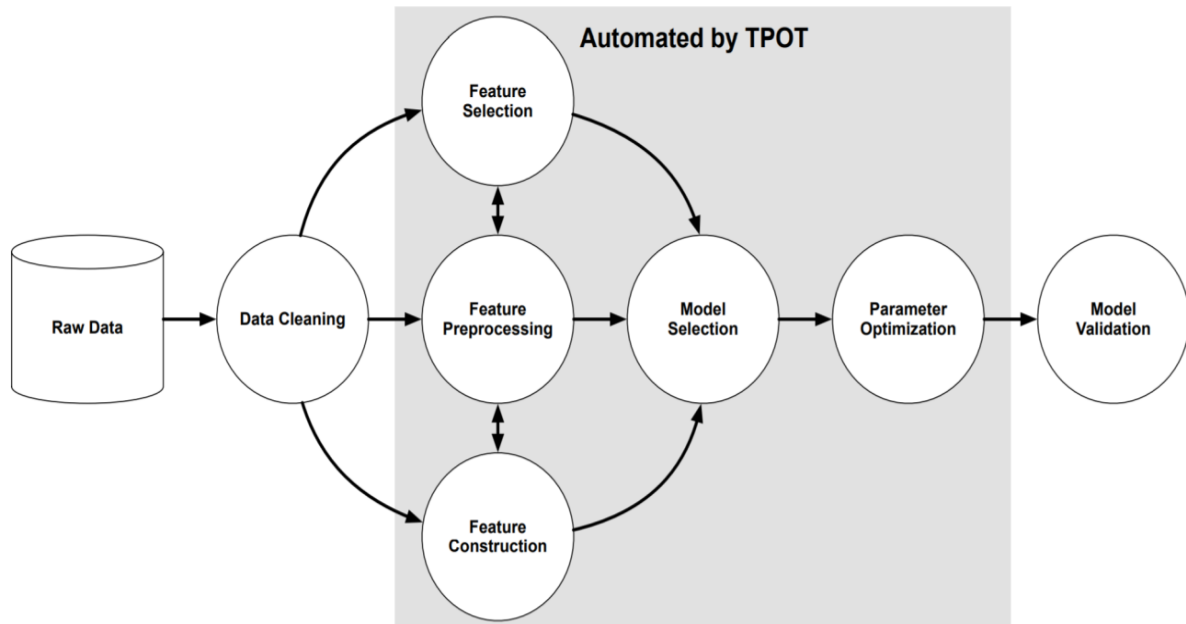
TPOT é um algoritmo evolutivo apresentado em 2016, que automatiza partes do processo de AM (OLSON et al., 2016).

Em um processo tradicional de AM, antes de aplicar técnicas para encontrar um modelo para os dados, é necessário que a base seja preparada. Inicialmente é feita a análise exploratória dos dados, em busca, por exemplo, de dados ausentes ou marcações incorretas, para corrigir ou remover registros (processo conhecido como limpeza de dados). Em seguida, é possível transformar os dados de modo a torná-los mais adequados para modelagem, por exemplo, normalizar as observações (processo conhecido como pré-processamento de dados), removendo atributos que não são úteis para a modelagem (processo conhecido como seleção de atributos) ou criando novos recursos a partir dos dados existentes (conhecido como construção de atributos).

Depois, deve-se selecionar um modelo de AM que melhor represente os dados (processo conhecido como seleção de modelo) e escolhe-se os parâmetros do modelo que permitam a classificação mais precisa dos dados (processo conhecido como otimização de parâmetros). Por último, deve-se validar o modelo para garantir que boas previsões sejam realizadas para novos conjuntos de dados, testando se esse é generalista o suficiente para dados que não foram utilizados no processo de ajuste do modelo (processo chamado de validação do modelo), por exemplo, testando o desempenho do modelo em um conjunto de dados de validação que foi excluído das fases anteriores do *Pipeline* (OLSON et al., 2016).

Na Figura 1 pode-se observar o fluxo de um processo típico para o desenvolvimento de um algoritmo de AM supervisionado. A área em cinza claro da figura indica as etapas no *Pipeline* que são automatizadas pelo TPOT.

Figura 1 - Fluxo típico para construção de um modelo de AM 



Fonte: (OLSON et al., 2016).

O TPOT é baseado na geração de candidatos de forma genética (LANGDON et al., 1970). A base de dados é copiada diversas vezes, sendo aplicada uma operação diferente para cada cópia, como um pré-processamento ou alguma forma de manipulação dos dados. Em seguida, os resultados são sumarizados e passam por um ciclo de seleção de variáveis.

Posteriormente, a base de dados já filtrada segue para o ciclo de construção do modelo de AM já com hiperparâmetros estabelecidos. Todo esse processo ocorre para a geração de um *Pipeline* e seu esquema pode ser visualizado na Figura 2 que representa os ciclos de geração de um *Pipeline*.



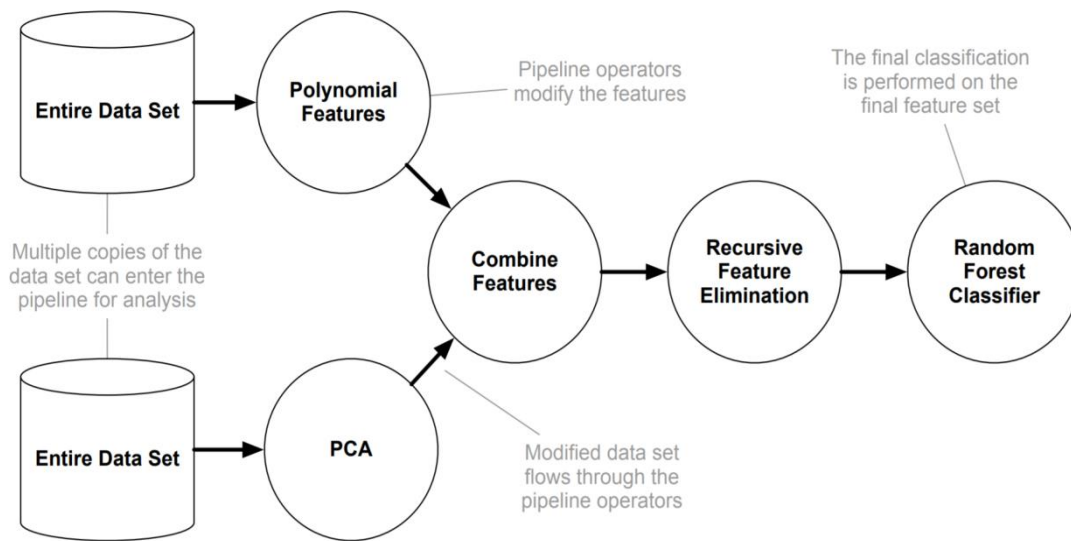
A Figura 2 apresenta um exemplo do TPOT  aplicado em que cada círculo representa uma parte do processo de construção de um modelo de AM e as setas indicam a direção do fluxo desse processo.



Figura 2 - Exemplo do TPOT aplicado 




Fonte: (OLSON et al., 2016).

Após a construção de diversos *Pipelines*, assim como um algoritmo genético comum, uma otimização dos *Pipelines* é realizada (LANGDON et al., 1970). Vários *Pipelines* são gerados, mas apenas os *Pipelines* com melhores resultados são armazenados, os demais da geração são descartados. Esse processo é o paralelo da seleção natural dos “genes melhor adaptados”.

Em seguida, os melhores *Pipelines* de cada geração são separados e passam por uma recombinação de genes, com uma técnica chamada de cruzamento, em que os *Pipelines* filhos são compostos por blocos dos *Pipelines* de seus pais, gerando assim uma nova safra de *Pipelines*. Essa geração passa pela etapa de seleção natural novamente separando os melhores resultados. O fluxo continua até que o critério de parada seja atingido, podendo ser o tempo máximo delimitado, por exemplo.

### 3.5 fMRI

Imagem por  IRI é uma técnica de neuroimagem que explora mudanças nas funções cerebrais ao longo do tempo. Esse é um método indireto de detecção de atividade, que mede a necessidade metabólica da ativação dos neurônios. Como a energia do cérebro é continuamente suprida pelo sangue, é a mudança do fluxo

sanguíneo em resposta à necessidade de oxigênio e glicose pelos neurônios que resulta em um sinal que pode ser medido por fMRI.

Um dos pontos a se analisar nesse método é se o sangue é ou não oxigenado. É possível capturar essa informação através da molécula de hemoglobina que têm propriedades magnéticas, podendo ser diamagnética (não tem momento magnético) ou paramagnética (tem momento magnético e pode ser atraído por campo magnético), a depender da oxigenação, se oxigenado (oxihemoglobina) ou não (deoxihemoglobina) (FIGUEIRA, 2013).

Quando as moléculas de hemoglobina perdem seu oxigênio para os neurônios, se transformam em deoxihemoglobina, incorrendo no aumento da sensibilidade magnética, que degenera seus spins quando expostos ao campo magnético da máquina do MRI da qual pode ser vista na Figura 3 (FIGUEIRA, 2013).

Figura 3 - Equipamento de ressonância magnética



Fonte: (SZAMEITAT; SHEN; STERR, 2009).

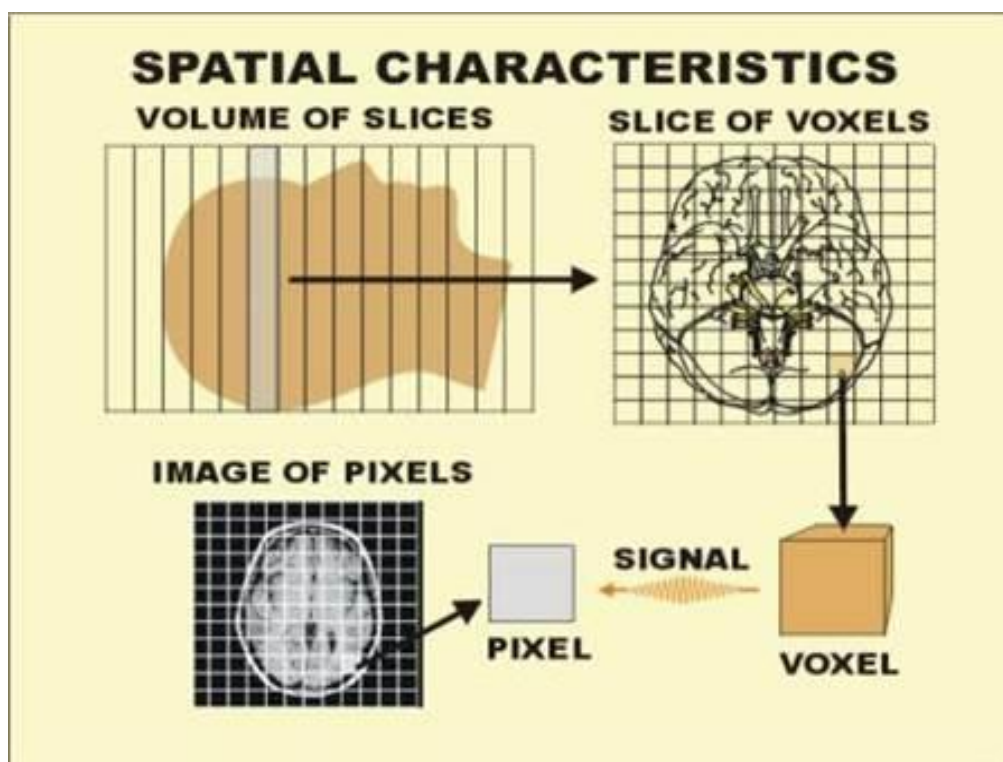
A mudança na relação de  $\text{HbO}_2/\text{deoxi-Hb}$  durante o aumento de fluxo sanguíneo oxigenado acarreta em um aumento do sinal de ressonância subjacente ao local onde ocorreu o aumento da atividade de neurônios. Essa defasagem resulta em uma deterioração da magnetização transversal, possibilitando a detecção por sequências de pulso MR sensíveis.

### 3.6 Voxel

O voxel é uma unidade tridimensional que incorpora os sinais em varreduras cerebrais (SMITH, 2004). Um voxel é um elemento de volume, que representa um valor no espaço tridimensional, correspondendo a um pixel para uma determinada espessura de fatia. Voxels são frequentemente usados na visualização e análise de dados médicos

Recentemente, técnicas de análise de imagens baseadas em voxels têm sido empregadas para a identificação de diferenças cerebrais por imagens de ressonância magnética estruturais. Essas técnicas envolvem um procedimento de normalização de imagens de exames em alta resolução em um espaço tridimensional e subdividido em elementos, como é possível observar na Figura 4. As imagens são segmentadas entre matéria cinza e branca, sendo aplicados alguns tratamentos como a suavização por *kernel* Gaussiano isotrópico. Por fim, são realizadas uma série de comparações entre voxels de matéria cinza e clara, em diferentes grupos de pacientes, com o objetivo de identificar as diferenças (MECHELLI et al., 2005).

Figura 4 - Características tridimensionais de um voxel.



Fonte: (SPRAWLS, 2000).

~~As imagens são segmentadas entre matéria cinza e branca, sendo aplicados alguns tratamentos como a suavização por kernel Gaussiano isotrópico. Por fim, são realizadas uma série de comparações entre voxels de matéria cinza e clara, em diferentes grupos de pacientes, com o objetivo de identificar as diferenças (MECHELLI et al., 2005).~~

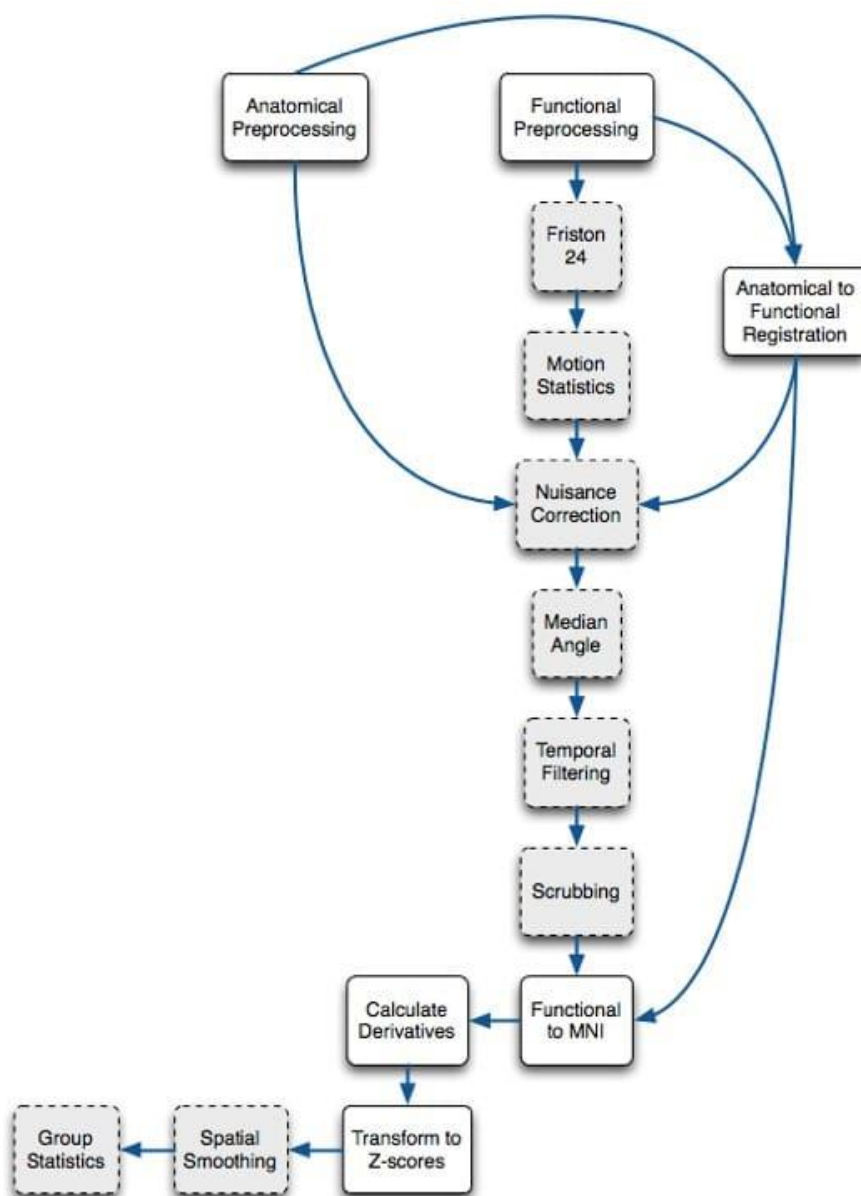
Técnicas de análise baseadas em voxels têm sido úteis na caracterização de mudanças sutis na estrutura cerebral em uma variedade de doenças associadas a disfunções neurológicas e psiquiátricas. Dentre elas, estão a esquizofrenia, distúrbios congênitos, epilepsia de lobo temporal e até mesmo dores de cabeça agudas. Além disso, as técnicas têm tido sucesso na identificação de anormalidades como as observadas na encefalite, na esclerose múltipla e na doença de Alzheimer (MECHELLI et al., 2005).

### 3.7 CPAC

A CPAC (do inglês, *Configurable Pipelines for the analysis of Connectomes*) é uma ferramenta concebida para possibilitar a realização de análises sobre a arquitetura funcional do cérebro e suas associações comportamentais. A ferramenta se caracteriza por ser uma solução para o processamento automático de *Pipelines* para dados de MRI, sendo configurável, de fonte aberta e baseada em softwares existentes. Com a CPAC, usuários podem realizar operações de pré-processamento e de análise de dados em larga escala de uma maneira facilitada, explorando o impacto de decisões em alterações de variáveis para o pré-processamento a partir da aplicação de múltiplos *Pipelines* simultâneos (CAMERON et al., 2013).

A CPAC é capaz de processar dados de centenas de milhares de observações em uma variedade de estratégias de pré-processamento em uma única execução, sendo otimizada para uso em grandes bases de dados como as publicadas pelo INDI (*do inglês, International Neuroimaging Datasharing Initiative*). Os *Pipelines* de análise e processamento da CPAC da Figura 5 são configurados através de um arquivo de configuração, permitindo a inclusão e exclusão de diferentes passos, bem como a aplicação de diferentes conjuntos de parâmetros (CAMERON et al., 2013).

Figura 5 - Pipelines de análise e processamento da CPAC.



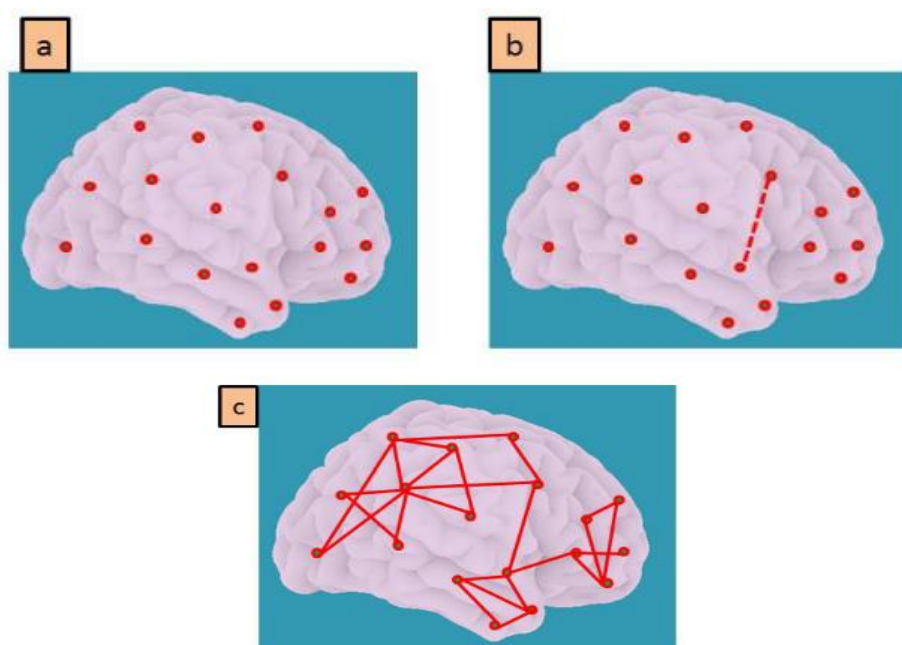
Fonte: (CAMERON et al., 2013).

A ferramenta é implementada em linguagem Python, utilizando a biblioteca de *Pipelines* Nipype. A Nipype fornece mecanismos para automatizar a detecção e a exploração de forma paralela em um dado *Pipeline*, possibilitando a iteração entre diferentes configurações de parâmetros. A CPAC proporciona uma extensão para as funcionalidades da Nipype, provendo fluxos de trabalho específicos para análises de conectividade, derivativos de conectividade funcionais, e análises não encontradas em outros pacotes de neuroimagens, oferecendo uma interface simplificada para a execução de *Pipelines* (CAMERON et al., 2013).

### 3.8 ROI e Conectividade Funcional

Diferentes regiões do cérebro humano trabalham em sincronia, mesmo que anatomicamente separadas, sugerindo que existem conexões funcionais e estruturais. Isso implica, que é possível assumir que o cérebro pode ser estudado em relação às diferenças entre indivíduos e tarefas, em que os nodos são diferentes regiões e as arestas são medidas de conectividade funcional entre séries temporais de um sinal de ressonância magnética associado a cada região (PAMPLONA, 2014). Esse cenário é ilustrado na Figura 6, com (a) representando os nodos, (b) os nodos com uma aresta e (c) alguns nodos entre si conectados através das arestas.

Figura 6 - Representação de nodos e arestas.



Fonte: (PAMPLONA, 2014).

Diferentes métodos têm sido utilizados para avaliar a Conectividade Funcional. Comumente, faz-se uso de métricas que representam a dependência estatística entre séries temporais ou na forma de mapas espaciais, após agrupar voxels ou regiões com base na série temporal latente. Entre elas têm-se correlação simples ou parcial, coerência cruzada e correlação canônica. Nesses métodos são comparadas as séries temporais por pares, com uma considerada modelo e outra proveniente de região semente, o que dá a essa abordagem a denominação de métodos dependentes ou modelo de semente (SALMON; LEONI, 2019).

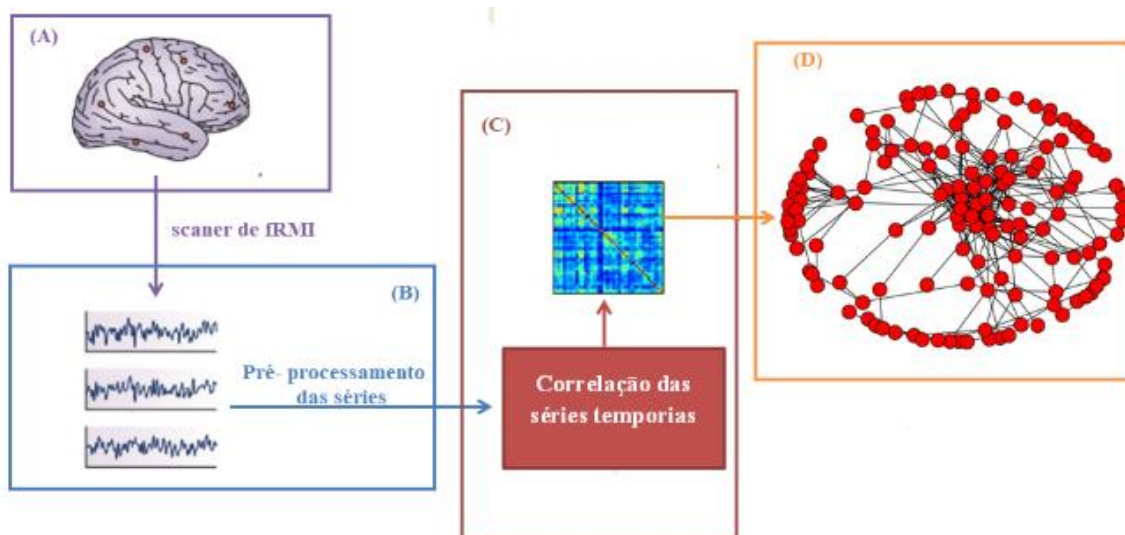


Há distintas estratégias para a determinação da região semente, que pode ser um único voxel ou toda uma região de interesse (ROI, do inglês *Region of Interest*). Finalmente chega-se em um dos pontos mais importantes de todo o *Pipeline* que consiste em através do método da semente, obter-se uma matriz de conectividade ou de associação cuja interpretação depende da escolha das sementes e das ROIs associadas a um determinado atlas anatômico (SALMON; LEONI, 2019).

A conectividade das arestas pode ser representada por grafos, com uma distinção de conectividades funcionais e conectividades não funcionais.

A conectividade entre os nós da seção anterior é representada por arestas em grafos. As arestas podem ser diferenciadas pela ausência ou presença de direção, de modo que grafos não direcionados possuem matrizes simétricas. Adiciona-se ainda que conectividades funcionais originam grafos não direcionais enquanto as efetivas dão origem a grafos direcionados. Desse modo, o peso nas redes funcionais representa a magnitude da interação, a qual pode variar de -1 a +1 e redes sem peso são denominadas de binárias, apresentando valores de 0 a 1, sendo que 1 representa regiões conectadas e 0 quando não há conexão. Esse cenário com todo *Pipeline*, incluindo também a matriz de correlação e os gráficos pode ser visualizado na Figura 7 (ALVES, 2019).

Figura 7 - Pipeline geral.



Fonte: (ALVES, 2019).

### 3.9 Atlas Craddock 200 (CC200) e Atlas Craddock 400 (CC400)

Regiões de Interesse, em análises de funções cerebrais, são definidas pela seleção de regiões do cérebro para investigação de funcionalidades neurais. Seja na construção de modelos computacionais para funções cerebrais, na exploração de conectomas humanos, ou na investigação da conectividade funcional em doenças, um requisito necessário é a definição de regiões do cérebro para a realização de análises e modelagens (CRADDOCK et al., 2012).

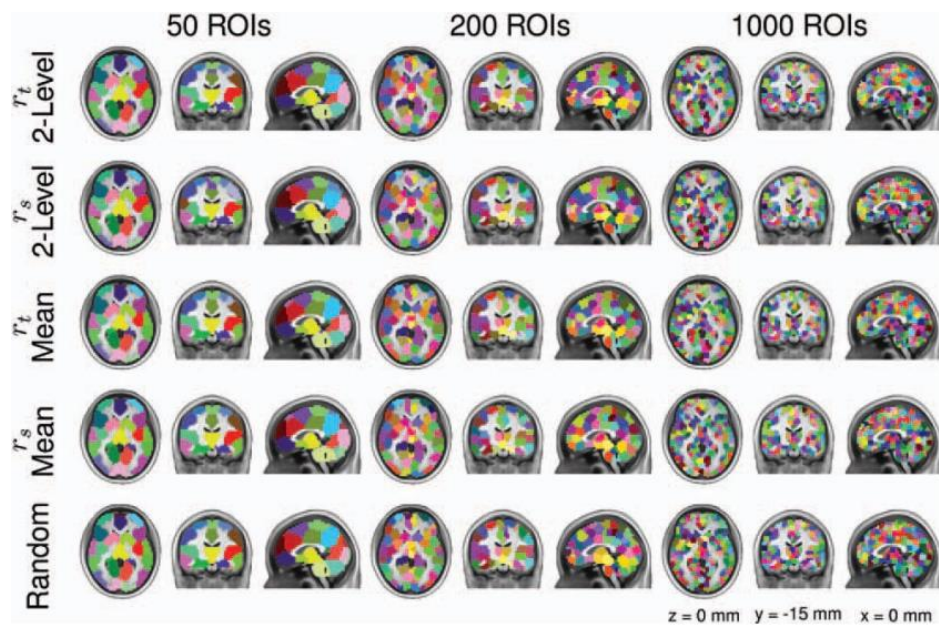
Vários métodos foram utilizados para se desenvolver conjuntos de ROIs para análises de funções cerebrais. Um dos métodos consiste em manualmente definir ROIs por meio de templates ou imagens anatômicas, requerendo que sejam criadas hipóteses sobre a forma, o tamanho e a localização de áreas funcionais do cérebro baseadas em literatura neuroanatômica ou em resultados de análises funcionais de neuroimagens (CRADDOCK et al., 2009; DOSENBACH et al., 2010). Esse método é um processo trabalhoso e propenso a vieses e erros (CRADDOCK et al., 2012).

Diversos outros métodos foram desenvolvidos para a definição de ROIs, dentre eles métodos de agrupamentos, ou clusterização, baseados em dados de fMRI em estado de repouso. A abordagem utilizada neste trabalho foi a determinação de ROIs, conforme disponibilizado por Craddock, tendo sido implementadas as soluções com 200 e 400 ROIs. Craddock desenvolveu um algoritmo de clusterização com restrições espaciais para se definir ROIs baseadas na estrutura de correlação de dados fMRI sem a necessidade de se definir grupos anatômicos a princípio. A abordagem introduziu um método para se gerar um atlas de ROI por meio do parcelamento de dados de fMRI do cérebro em estado de repouso em diferentes regiões espaciais com conectividades funcionais homogêneas (CRADDOCK et al., 2012).

Na Figura 8 pode ser observada a clusterização para quatro métodos de parcelamento funcionais. No método de parcelamento aleatório contendo 50, 200, ou 100 ROIs, cada parcelamento é retratado em três visões ortogonais. Códigos de cores foram arbitrariamente utilizados para melhor enfatizar os limites das ROI. As cores das ROIs foram igualadas entre os resultados dos parcelamentos para se enfatizar similaridades na identificação de ROI entre os métodos (CRADDOCK et al., 2012).



Figura 8 - Clusterização para quatro métodos de parcelamento funcionais.



Fonte: (CRADDOCK et al., 2012).

### 3.10 Base de dados

Para esse estudo, foram utilizados dados advindos da *Resting State Functional* MRI (rs-fMRI), em português, Ressonância Magnética Funcional em Estado de Repouso, que é uma ferramenta vastamente utilizada para investigar a conectividade funcional e distúrbios cerebrais. Contudo, essa abordagem é sensível a ruídos estruturados e aleatórios de fontes que não sejam neurais, como por exemplo fontes fisiológicas. Desse modo, é primordial que se realize um pré processamento inicial para remoção de ruídos nos dados (DIAO et al., 2021).

Adiciona-se ainda que, independentemente do paradigma de aquisição das imagens e da técnica de pré-processamento, todo o processo é relativamente longo, consumindo na ordem de minutos. Assim, é inevitável a existência de uma variabilidade indesejável nas séries temporais adquiridas, seja por motivos intrínsecos da técnica de aquisição ou pelos ruídos fisiológicos, como mencionado anteriormente (SALMON; LEONI, 2019).

Complementa-se a esse contexto que na literatura não há um consenso sobre os melhores métodos para o pré-processamento de dados oriundos de fMRI em estado de repouso. Dessa forma, a ABIDE (do inglês, *Autism Brain Imaging Data Exchange*), comenta que ao invés de favorecer uma única estratégia de processamento, o processamento pode ser realizado através de quatro diferentes

*Pipelines*, ou seja, utilizando-se de 4 distintos fluxos de processos, os quais são (ABIDE, 2019):

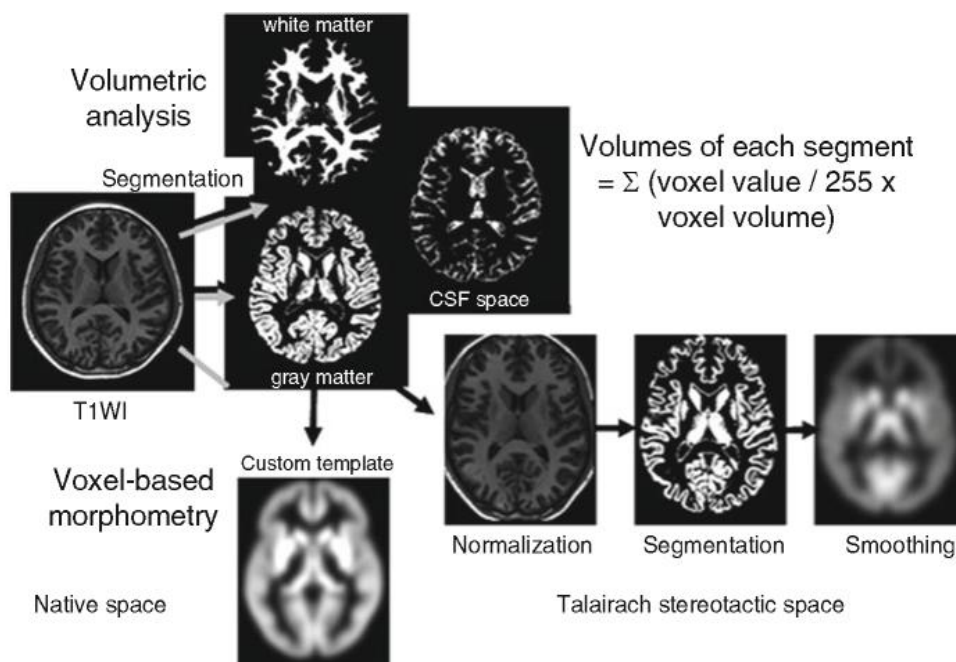
- *Connectome Computation System (CCS)*;
- *Configurable Pipeline for the Analysis of Connectomes (CPAC)*;
- *Data Processing Assistant for Resting-State fMRI (DPARSF)*;
- *Neuroimaging Analysis Kit (NIAK)*.

Isso é possível, porque os passos implementados em cada um dos *Pipelines* são suficientemente similares, e estão descritos a seguir (YUMI, 2018):

- *Segmentação*: Separação entre substância branca, cinzenta e licor;
- *Normalização espacial*: Utiliza-se de rotação, translação e modificação na escala (KIM, 2010);
- *Suavização espacial*: Consiste em borrar a imagem para padronizar os dados dos diferentes formatos de cérebro, atingindo-se assim um modelo padrão, o qual comumente utilizado é o MNI152;
- *Voxel based morphometry*: Utilização de inferência bayesiana, para cada voxel estima a probabilidade de ser substância branca, cinzenta ou licor.

Tem-se um exemplo desse *Pipeline* na Figura 9.

Figura 9 – Pré-processamento e análise de imagens de fMRI.



Fonte: (KOBATAKE et al., 2017).

Este trabalho utilizou dados abertos de neuroimagem pré-processados do ABIDE, dos quais foram extraídos do *Preprocessed Connectomes Project* (do inglês, PCP), compartilhados através da iniciativa do consórcio da INDI, sendo a base ABIDE uma colaboração de 16 sites internacionais de imagem que agregaram e compartilharam abertamente dados de neuroimagem de 539 indivíduos com TEA e 573 controles típicos. Essas 1.2 neuroimagens são compostas de dados estruturais e fMRI em estado de repouso, juntamente com informações fenotípicas.

#### 4 MATERIAIS E MÉTODOS

Há duas abordagens muito utilizadas para realizar a classificação de neuroimagens. Uma é a utilização de séries temporais, matriz de conectividade e redes complexas, a outra é a utilização de rede neural convolucional (WEN et al., 2018; ALVES, 2019).

Para a realização deste estudo, foi realizada a busca por uma base de dados pública disponível para obtenção de neuroimagens dos grupos de controle e neuroatípicos. Com isso, foi encontrada e utilizada a base de dados ABIDE pré-processada. Para a extração dos dados, utilizou-se o Python 3.0, com o auxílio da biblioteca *nilearn*.

Aplicou-se o *Pipeline* CPAC (YANG; SCHRADER; ZHANG, 2020), em que são eliminados alguns ruídos e outros pré-tratamentos das imagens, conforme citado na fundamentação teórica. Conforme sugerido em (YANG; SCHRADER; ZHANG, 2020) o atlas ROI CC400 foi aplicado, porém, para analisar se a complexidade da dimensão causada pela utilização do atlas ROI CC400 influencia nos resultados dos métodos de AM empregados, também foi separado um base de dados com a aplicação do atlas ROI CC200.

Para a extração da base de dados, usou-se do parâmetro *quality\_checked*, que separa somente às observações que passaram na avaliação de qualidade de todos os avaliadores (NIELSEN et al., 2013). A aquisição dos dados pré-processados foi efetuada através da biblioteca *nilearn*, pelo método *fetch\_abide\_pcp*.

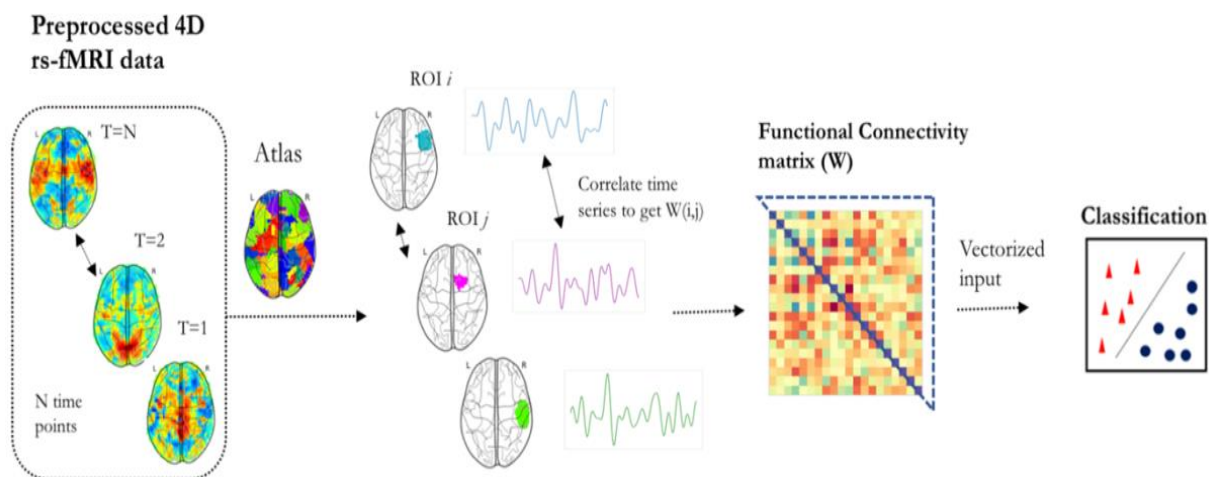
Após obtenção das séries temporais de cada ROI, para cada neuroimagem, foi utilizado o método *Connectivity Measure* do *nilearn*, aplicando o tipo “*correlation*”. Esse método cria uma matriz de conectividade baseada na correlação de Pearson, em que cada elemento representa uma região e como ela se relaciona às demais regiões, método similar ao utilizado na obtenção da correlação de Pearson entre variáveis em métodos de análise de dados.

Obtendo-se a matriz de conectividade, tem-se, para cada observação, uma matriz 2x2. Para possibilitar o uso de diferentes técnicas de AM, foi aplicado o método *vectorize*, que realiza uma transformação linear de 2D para 1D nos dados.

Antes da aplicação dos modelos, 10% dos dados foram separados para teste. Já na etapa dos métodos de AM, o primeiro método testado foi o SVC com *kernel* linear, utilizando-se da técnica de *cross validation*, conhecida como *K-fold*, disponível na biblioteca *sklearn*, com 5 *folds*.

A ideia principal do fluxo deste projeto pode ser visualizada na Figura 10. Parte-se da base *resting state* fMRI (rs-fMRI).

Figura 10 – Pipeline fMRI.



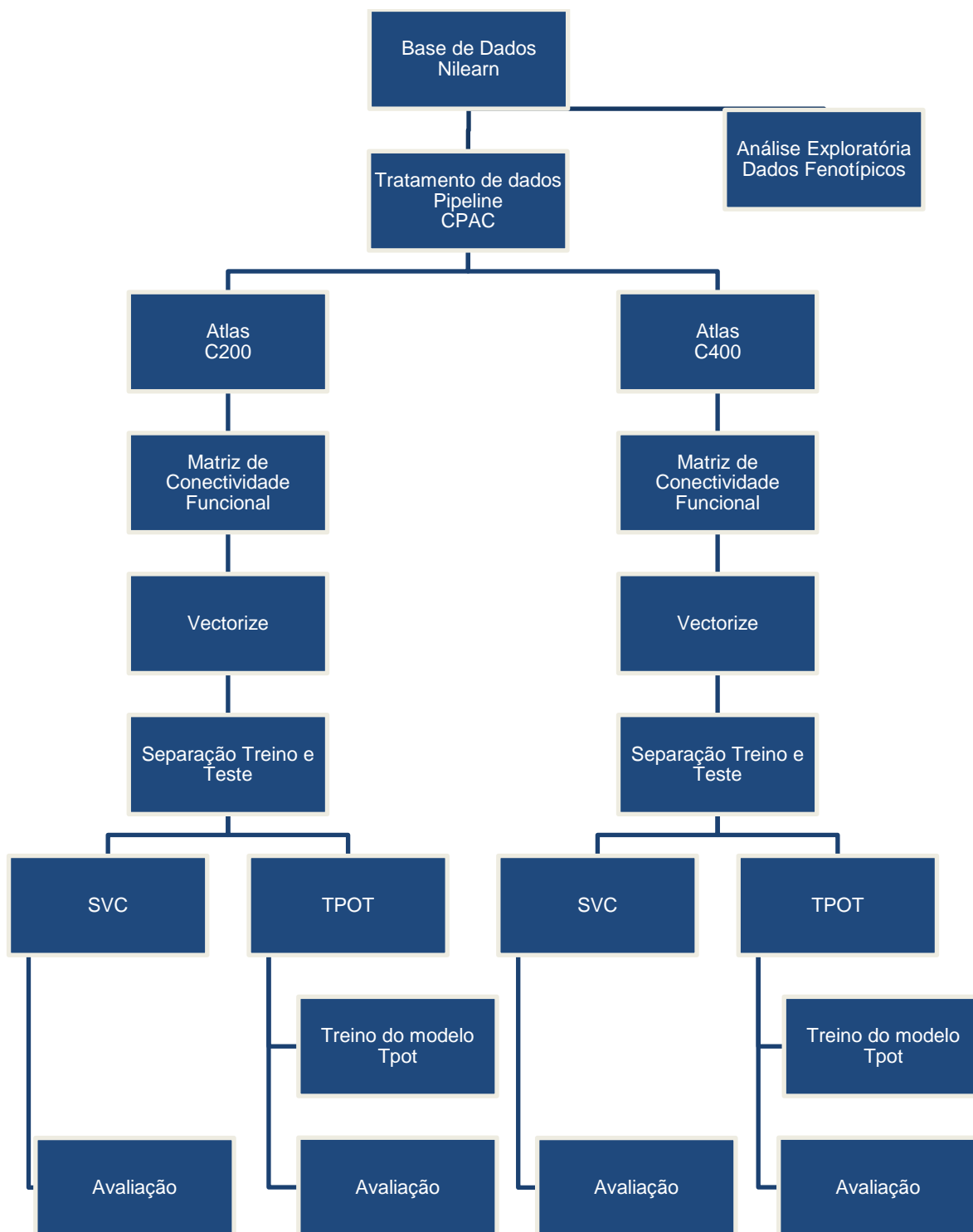
Fonte: Figura adaptada de (KHOSLA et al., 2019).

Extrai-se o fMRI, aplica-se um atlas para identificar as ROIs, extrai-se a matriz de conectividade funcional a partir das séries temporais de cada ROI para cada imagem e aplica-se o *vectorize*. Os dados resultantes são aptos para aplicação de técnicas de AM, obtendo-se modelos de classificação de autismo.

## 4.1 Fluxograma

Na Figura 11, pode-se observar o fluxograma com os passos empregados neste estudo.

Figura 11 - Fluxograma do projeto.



Fonte: Autoria Própria.

## 4.2 Recursos para replicabilidade

Os testes foram realizados com o Python versão 3, através do Anaconda, que trata-se de uma plataforma de distribuição da linguagem de programação Python para computação científica, que objetiva simplificar o gerenciamento e implantação de pacotes.

Todos os testes foram realizados utilizando a configuração de computador Intel i7-6700k 4.20Ghz, 16GB RAM DDR4, Placa de vídeo RTX 3060Ti.

Todos os procedimentos em Python foram executados com a *seed* 19, a fim de possibilitar a replicabilidade deste estudo.

### 4.2.1 Bibliotecas Python

Para realizar os testes em Python, foram utilizadas as bibliotecas com suas respectivas versões conforme Tabela 1.

Tabela 1 - Bibliotecas e configurações utilizadas no estudo.

<b>Biblioteca</b>	<b>Versão</b>
category-encoders	2.3.0
certifi	2021.5.30
chardet	4.0.0
colorama	0.4.4
confluent-kafka	1.6.0
cucim	21.8.1
cudf	21.8.3
cudf-kafka	21.8.3
cugraph	21.8.4
cuml	21.8.2
cusignal	21.8.0
cuspatial	21.8.1
custreamz	21.8.3
cuxfilter	21.8.0
cycler	0.10.0
dask-cuda	21.8.0
dask-cudf	21.8.3
deap	1.3.1
decorator	4.3.0
idna	2.10
ipython-genutils	0.2.0
joblib	1.0.1
kiwisolver	1.3.2

matplotlib	3.4.3
nibabel	3.2.1
nilearn	0.8.0
numpy	1.19.5
packaging	21.0
pandas	1.3.0
patsy	0.5.2
Pillow	8.3.2
pyparsing	2.4.7
python-dateutil	2.8.1
pytz	2021.1
requests	2.25.1
scikit-image	0.18.1
scikit-learn	0.24.2
scipy	1.7.0
six	1.16.0
sklearn	0.0
statsmodels	0.13.0
stopit	1.1.2
threadpoolctl	2.2.0
torch	1.10.0
TPOT	0.11.1
tqdm	4.62.2
update-checker	0.18.0
urllib3	1.26.6
xgboost	1.4.2

---

Fonte: Autoria Própria.

### 4.3 Tratamento dos dados

O *Pipeline* CPAC, já disponível para extração pelo *nilearn* ABIDE Pré-processado.

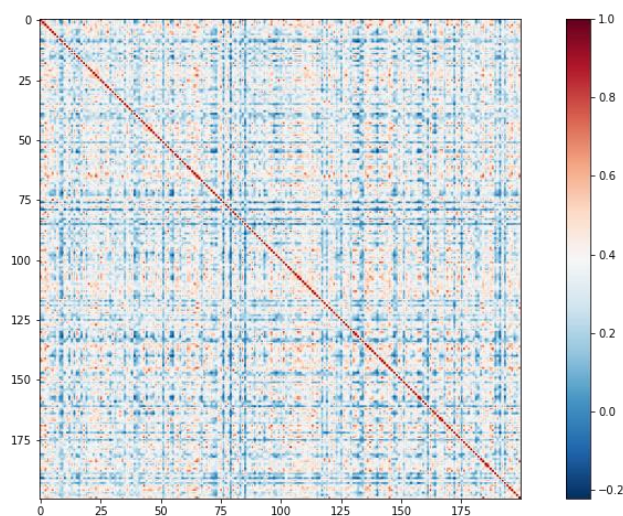
É importante notar que têm-se duas bases de dados diferentes, resultantes dos Atlas CC200 e CC400, que dividem o fluxograma. Após essa divisão, as bases de dados passam pelas mesmas etapas até a separação dos dados em treino e teste. Em seguida, tanto para a base de dados CC200, quanto para o CC400, é aplicado o TPOT e o SVC com *kernel* linear, resultando assim em 4 resultados diferentes: TPOT com CC200, SVC com CC200, TPOT com CC400 e SVC com CC400.



#### 4.4 Matriz de conectividade

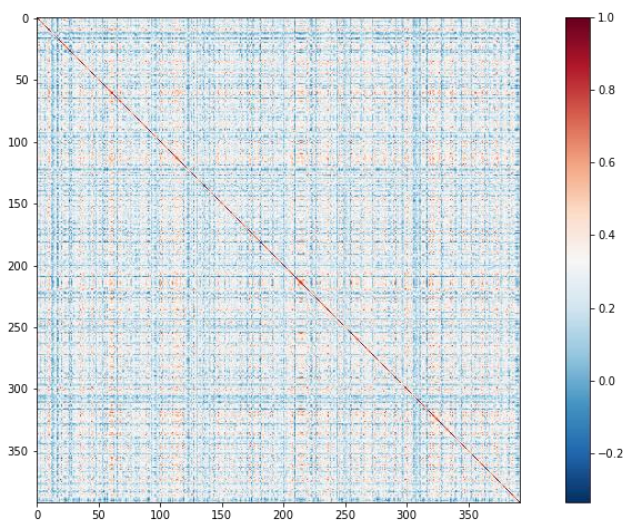
Para a construção da matriz de conectividade, utiliza-se os dados já tratados pelo CPAC e pelos Atlas. A Figura 12 traz a matriz de conectividade para o Atlas CC200, essa tem sua matriz com dimensão 200 x 200. Já na Figura 13, encontra-se a matriz de conectividade para o CC400, essa tem sua matriz com dimensão 400 x 400.

Figura 12 – Matriz de Conectividade da primeira amostra da base Atlas CC200.



Fonte: Autoria Própria.

Figura 13 – Matriz de Conectividade da primeira amostra da base com TEA dado pré-processado com Atlas CC400.



Fonte: Autoria Própria.

Nota-se que a diagonal da matriz tem a cor vermelha, o que representa máxima conectividade. Isso ocorre, porque é na diagonal que a matriz traz a correlação do ROI com o próprio ROI. Por isso, há o tratamento de retirar a diagonal para separar os dados para passar ao modelo.

Ademais, nota-se que a matriz é simétrica, sendo assim, há também o tratamento de extrair somente a informação do triângulo inferior da matriz.

Antes de passar os dados, como citado, é aplicado o *vectorize*, que realiza a transformação linear de 2D para 1D nos dados. Após esses tratamentos, tem-se para cada observação um vetor de tamanho 19.900 para dados advindos do CC200 e um vetor de 79.800 para dados advindos do CC400. O que resulta em maior complexidade computacional para os modelos que utilizam do Atlas CC400.

#### 4.5 Separação dos dados

Os dados foram separados em treino e teste, sendo 10% dos dados separados para teste e 90% dos dados separados para o treino, como pode ser observado na Figura 14.

Figura 14 – Separação entre treino e teste.



Fonte: Autoria Própria.

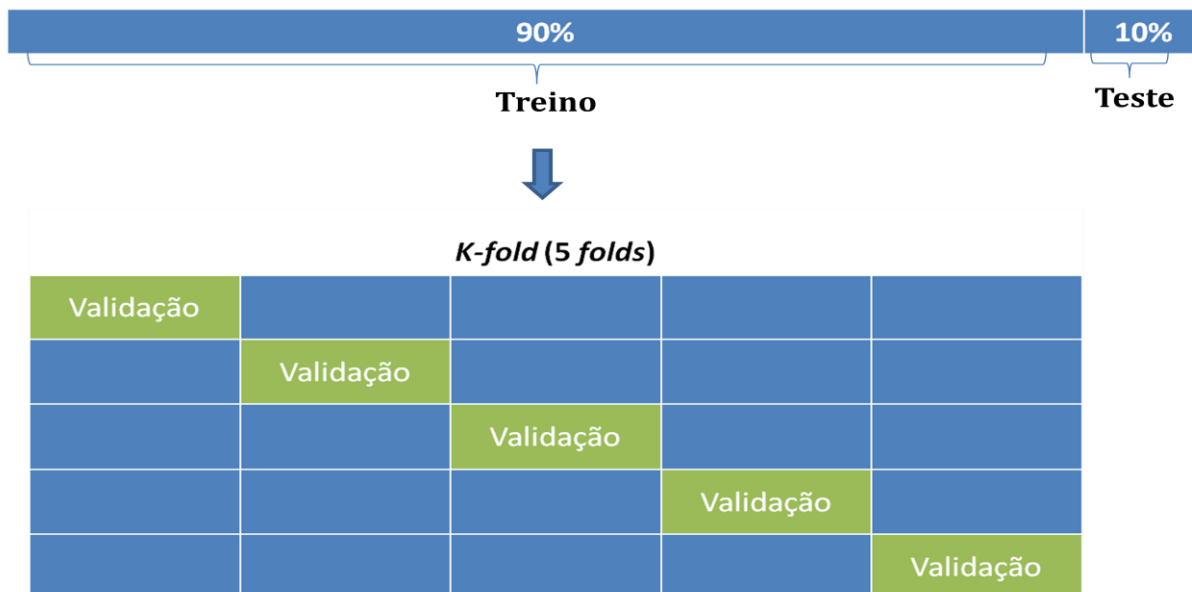
#### 4.6 Construção dos modelos

Para o desenvolvimento dos modelos, foi aplicado o método *K-fold* de validação cruzada. Esse método consiste em dividir o modelo em K vezes (BENGIO; GRANDVALET, 2004).

Ao final da iteração, após o modelo ser treinado por 5 vezes, é possível extrair um *score* verdadeiro de como o modelo está generalizando, em geral ao se consultar a média e desvio padrão de todos os treinos realizados. Esse processo demora mais que somente treinar o modelo, mas é essencial para assegurar a generalização dos dados. No exemplo da Figura 15, a base de dados foi separada

em 5 partes, em que para cada uma dessas partes o modelo treina com 4 partes (K-1) e valida com a parte restante.

Figura 15 – Separação da porção de treino em 5 folds, para validação dos dados.



Fonte: Autoria Própria.

Esse método foi empregado tanto para extração e melhor modelo advindo do autoML Tpot, quanto na geração do modelo via SVC.

#### 4.6.1 Tpot

No Tpot, foi utilizado o *f1 score* como métrica de otimização dos resultados, essa métrica nada mais é do que a média harmônica entre a precisão e a revocação. O tempo limite foi de 1.440 minutos, o equivalente a 24h, com limitação de 5 minutos para avaliação de cada *Pipeline* gerado. E 100 indivíduos a serem retidos na população da programação genética a cada geração, geralmente, o TPOT funciona melhor quando fornecido mais observações para otimizar o *Pipeline*.

#### 4.6.2 SVC

No SVC, foi utilizada a penalização e a tolerância padrões do algoritmo no scikit-learn, com C igual a 0.01, sendo esse parâmetro inversamente proporcional a

regularização, indica que foi aplicada uma alta regularização. O número máximo de iterações para execução foi de 10.000.

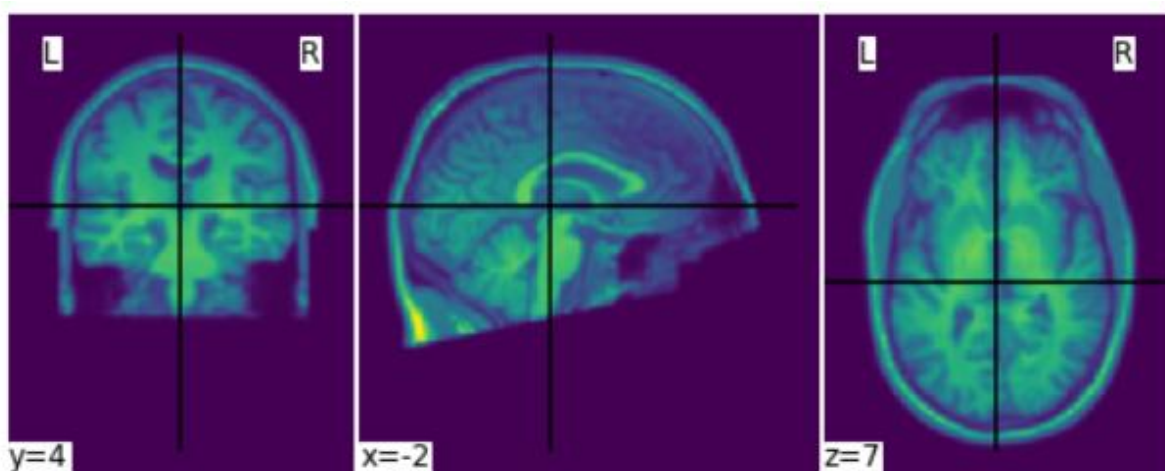
## 5 RESULTADOS E DISCUSSÃO

### 5.1 Análise Exploratória

A Análise Exploratória é um processo importante dentro do *Pipeline* da construção de modelos, porque é onde se tenta entender os dados, a fim de pensar em possibilidades para os próximos passos. São analisados quais tipos de tratamentos aplicar no dado, identificar algum viés, ou qualquer outro ponto que possa atrapalhar na criação de um modelo genérico o suficiente para conseguir capturar os padrões dos dados sem copiar esses dados.

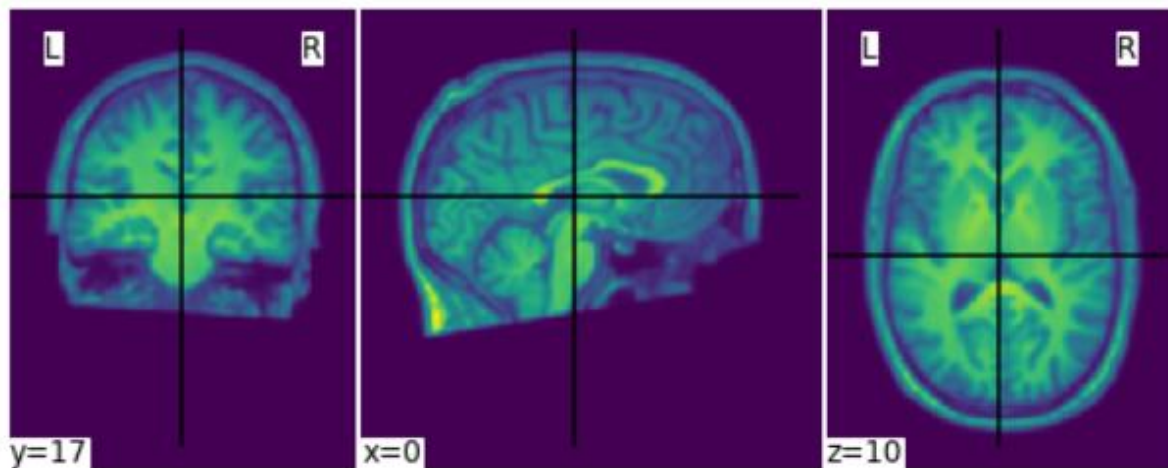
Para essa parte, a fim de verificar por inspeção visual possíveis padrões nas imagens, verificar se a presença de alguma diferença visual entre autismo e controle e analisar o quão diferentes as imagens estavam entre si, em termos de ruídos e borrados, por exemplo, foi utilizado o python com a biblioteca *nilearn*, que possibilitou a construção das Figuras 16, 17 e 18, que apresentam consecutivamente as imagens de homens de 48, 41 e 48 anos, sendo os dois primeiros TEA e o último parte do grupo de controle.

Figura 16 - 48 anos TEA 29006.



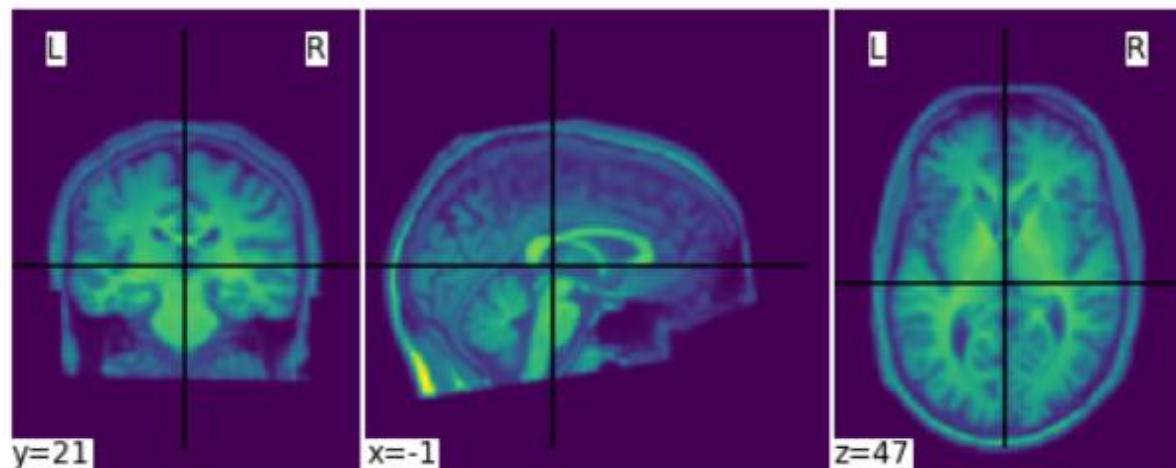
Fonte: Autoria Própria.

Figura 17 - 41 anos TEA 29007.



Fonte: Autoria Própria.

Figura 18 - 48 anos Controle 29011.



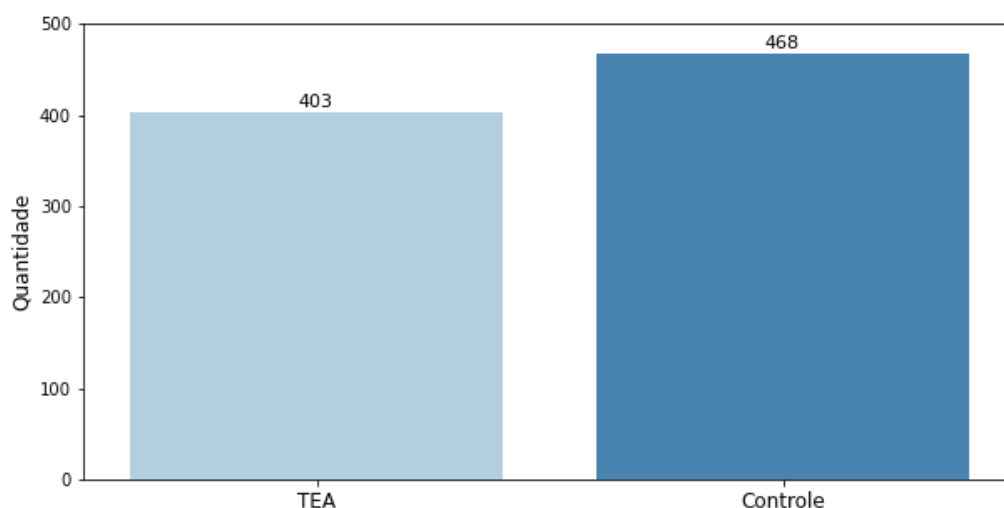
Fonte: Autoria Própria.

## 5.2 Análise Exploratória dos dados

A modelagem preditiva de classificação visa prever um rótulo de classe para cada observação. Este estudo objetiva prever se o fMRI é ou não advindo de um indivíduo com TEA, separando em classe TEA e classe Controle.

Após filtragem de parâmetros como o *quality-check* na base de dados ABIDE pré-processada e outros pré-tratamentos já citados, tem-se a redução da amostra de 1.112 para 871 neuroimagens. Como pode ser observado na Figura 19, têm-se na base 403 neuroimagens da classe TEA e 468 neuroimagens da classe Controle.

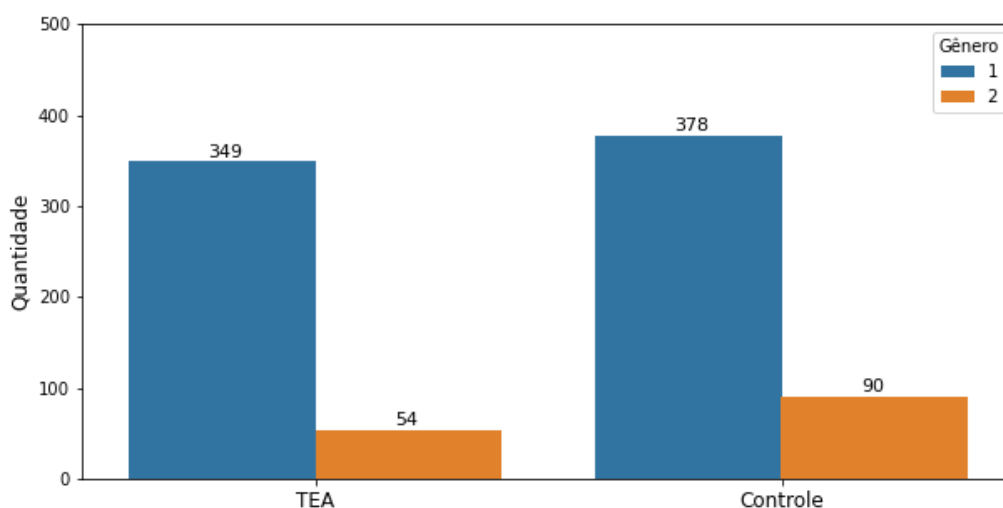
Figura 19 – Distribuição das classes.



Fonte: Autoria Própria.

Ao analisar os dados fenotípicos das imagens, Figura 20, nota-se que há o desbalanceamento dos dados em relação ao sexo. Sendo 144 neuroimagens advindas de observações do sexo feminino e 727 neuroimagens advindas do sexo masculino.

Figura 20 – Distribuição das classes por gênero.



Fonte: Autoria Própria.

Para analisar se a variável sexo (feminino ou masculino) e a variável *target* (TEA ou Controle) são estatisticamente independentes, utilizou-se do teste Chi-Quadrado de Pearson, por tratarem-se de variáveis categóricas.

O teste Chi-Quadrado de Pearson necessita da tabela de contingência, que encontra-se na Tabela 2 e foi construída no Python com a função *pd.crosstab*.

Tabela 2 - Tabela de Contingência para análise da distribuição das classes por gênero.

	Controle	TEA
Masculino	349	378
Feminino	54	90

Fonte: Autoria Própria.

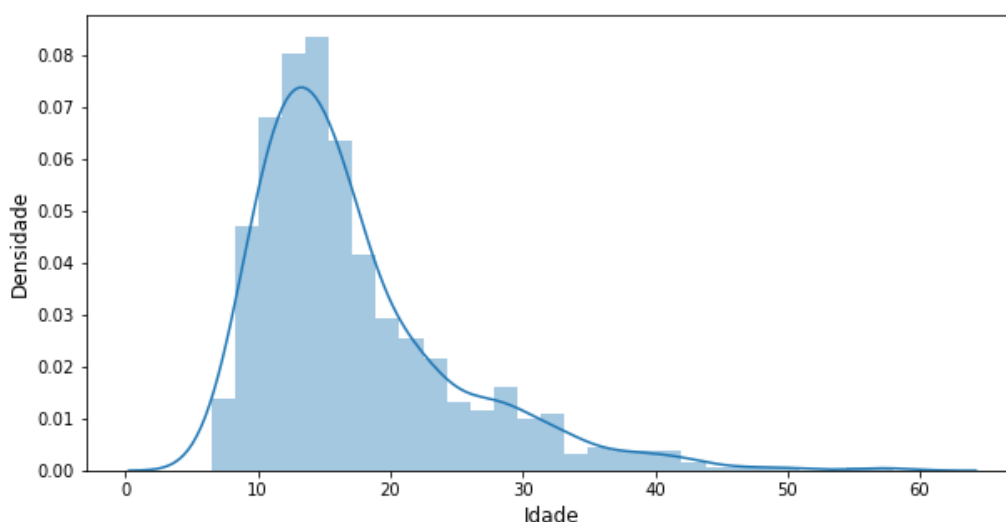
Após, passou-se a tabela de contingência para a função *chi2\_contingency* também do Python, que resultou em um *p-value* de 0.0265.

Como a hipótese nula é de que as variáveis são estatisticamente independentes. Considerando 95% de confiança, tendo encontrado um *p-value* menor que 0.05, tem-se que a hipótese nula pode ser descartada e que há relação entre as variáveis, ou seja, elas não são independentes.

Apesar disso, considerando que há poucas amostras para o estudo, decidiu-se por manter os neurorimagens advindas do sexo feminino. Em um estudo futuro, pode-se recolher mais dados e realizar um estudo sobre o autismo voltado somente para o sexo feminino.

Outro atributo analisado foi a distribuição dos dados por idade, na Figura 21 pode-se notar uma maior concentração de idades entre 10 e 30 anos. E pouco ou nenhum dado antes dos 9 e depois dos 45 anos.

Figura 21 – Distribuição dos dados por idade.



Fonte: Autoria Própria.



## 5.3 Resultados Dos Modelos

### 5.3.1 Atlas CC200 com SVC

O Atlas utilizado nos resultados da tabela 3, tabela 4 e Figura 22 foi o CC200, com aplicação do método de AM SVC com *kernel* linear com os parâmetros explicitados na metodologia.

Na tabela 3, tem-se o resultado das principais métricas após treino do modelo com o *cross validation*, sendo que esses valores tratam-se da média do resultado da porção de validação para cada *fold*, no caso 5 *folds*. Com o desvio padrão nota-se que há uma pequena variação dos resultados, porém não significativa em detrimento da média.

Tabela 3 - Resultado na validação cruzada.

	Média	Desvio Padrão
<b>Acurácia</b>	0.6540	0.0485
<b>Recall</b>	0.5775	0.0511
<b>Precision</b>	0.6400	0.0569
<b>F1</b>	0.6069	0.0523

Fonte: Autoria Própria.

Após treino do modelo final com a base de dados total de treino, aplica-se o modelo para predizer as classes na porção de teste. Comparando-se a classe predita com a classe real, tem-se na tabela 4 os resultados das métricas de classificação.

Tabela 4 - Resultado do modelo final SVC.

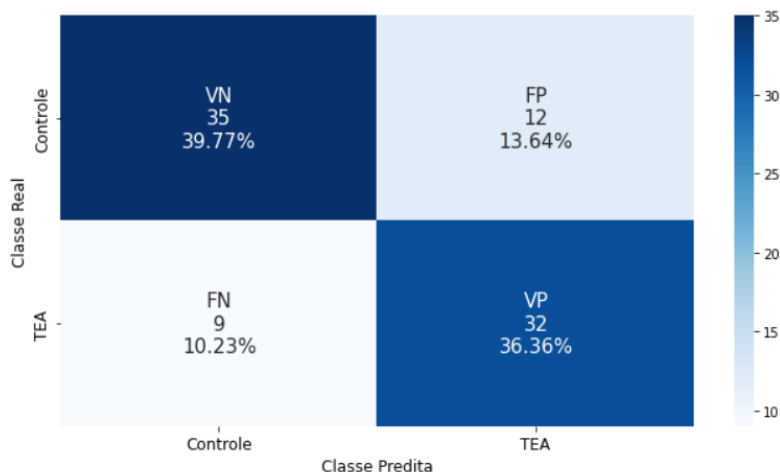
	SVC_200
<b>Acurácia</b>	0.7614
<b>Recall</b>	0.7805
<b>Precision</b>	0.7273
<b>F1</b>	0.7529

Fonte: Autoria Própria.



Na Figura 22, são apresentadas as quantidades de: verdadeiro negativo (VN), verdadeiro positivo (VP), falso negativo (FN) e falso positivo (FP) para as predições efetuadas na porção de teste.

Figura 22 – Matriz de Confusão do modelo final com a base de dados completa de treino.



Fonte: Autoria Própria.

### 5.3.2 Atlas CC200 com TPOT

O Atlas utilizado nos resultados da tabela 5, tabela 6 e Figura 23 foi o CC200, com execução do autoML TPOT para obtenção do melhor *Pipeline* de AM.

Na tabela 5, tem-se o resultado das principais métricas após treino do modelo com o *cross validation*, sendo que esses valores tratam-se da média do resultado da porção de validação para cada *fold*, no caso 5 *folds*. Com o desvio padrão nota-se que há uma pequena variação dos resultados, porém não significativa em detrimento da média.

Tabela 5 - Resultado do Pipeline advindo do TPOT na validação cruzada.

	Média	Desvio Padrão
<b>Acurácia</b>	0.6475	0.0246
<b>Recall</b>	0.5027	0.0704
<b>Precision</b>	0.6536	0.0212
<b>F1</b>	0.5667	0.0509

Fonte: Autoria Própria.

Após treino do modelo final com a base de dados total de treino, aplica-se o modelo para prever as classes na porção de teste. Comparando-se a classe predita com a classe real, tem-se na tabela 6 os resultados das métricas de classificação.

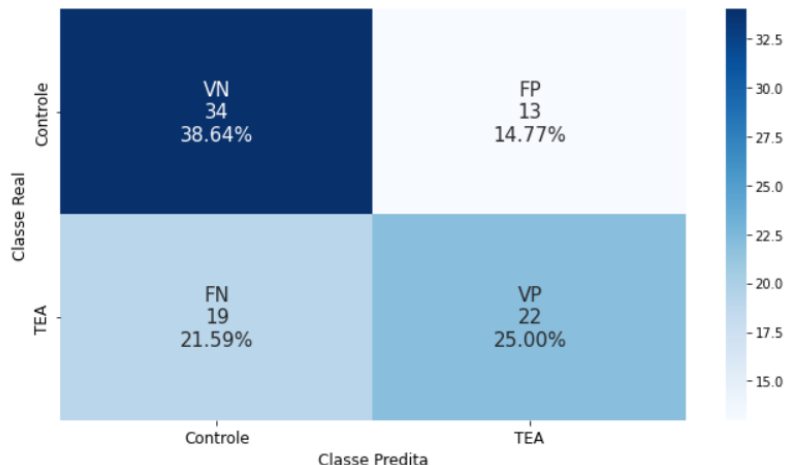
Tabela 6 - Resultado do modelo final do Pipeline advindo do TPOT.

TPOT_200	
<b>Acurácia</b>	0.6364
<b>Recall</b>	0.5366
<b>Precision</b>	0.6286
<b>F1</b>	0.5789

Fonte: Autoria Própria.

Na Figura 23, são apresentadas as quantidades de VN, VP, FN e FP para as predições efetuadas na porção de teste.

Figura 23 – Matriz de Confusão do modelo final com a base de dados completa de treino.



Fonte: Autoria Própria.

### 5.3.3 Atlas CC400 com SVC

O Atlas utilizado nos resultados da tabela 7, tabela 8 e Figura 24 foi o CC400, com aplicação do método de AM SVC com *kernel* linear com os parâmetros explicitados na metodologia.

Na tabela 7, tem-se o resultado das principais métricas após treino do modelo com o *cross validation*, sendo que esses valores tratam-se da média do resultado da porção de validação para cada *fold*, no caso 5 *folds*. Com o desvio padrão nota-se que há uma pequena variação dos resultados, porém não significativa em detrimento da média.

Tabela 7 - Resultado na validação cruzada.

	Média	Desvio Padrão
<b>Acurácia</b>	0.6718	0.0275
<b>Recall</b>	0.5912	0.0411
<b>Precision</b>	0.6639	0.0367
<b>F1</b>	0.6247	0.0304

Fonte: Autoria Própria.

Após treino do modelo final com a base de dados total de treino, aplica-se o modelo para prever as classes na porção de teste. Comparando-se a classe predita com a classe real, tem-se na tabela 4 os resultados das métricas de classificação.

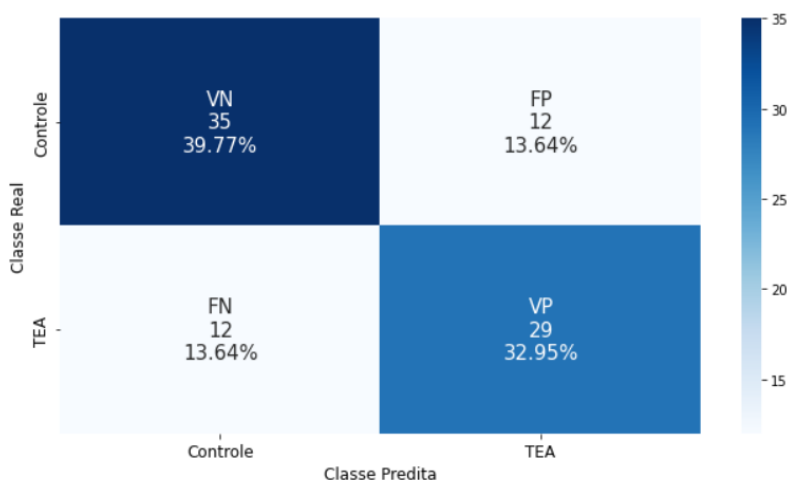
Tabela 8 - Resultado do modelo final SVC.

	SVC_400
<b>Acurácia</b>	0.7273
<b>Recall</b>	0.7073
<b>Precision</b>	0.7073
<b>F1</b>	0.7073

Fonte: Autoria Própria.

Na Figura 24, são apresentadas as quantidades de VN, VP, FN e FP para as predições efetuadas na porção de teste.

Figura 24 – Matriz de Confusão do modelo final com a base de dados completa de treino.



Fonte: Autoria Própria.

### 5.3.4 Atlas CC400 com TPOT

O Atlas utilizado nos resultados da tabela 9, tabela 10 e Figura 25 foi o CC400, com execução do autoML TPOT para obtenção do melhor *Pipeline* de AM.

Na tabela 9, tem-se o resultado das principais métricas após treino do modelo com o *cross validation*, sendo que esses valores tratam-se da média do resultado da porção de validação para cada *fold*, no caso 5 *folds*. Com o desvio padrão nota-se que há uma pequena variação dos resultados, porém não significativa em detrimento da média.

Tabela 9 - Resultado do Pipeline advindo do TPOT na validação cruzada.

	Média	Desvio Padrão
<b>Acurácia</b>	0.6731	0.0365
<b>Recall</b>	0.6135	0.0694
<b>Precision</b>	0.6557	0.0322
<b>F1</b>	0.6332	0.0502

Fonte: Autoria Própria.

Após treino do modelo final com a base de dados total de treino, aplica-se o modelo para prever as classes na porção de teste. Comparando-se a classe

predita com a classe real, tem-se na tabela 10 os resultados das métricas de classificação.

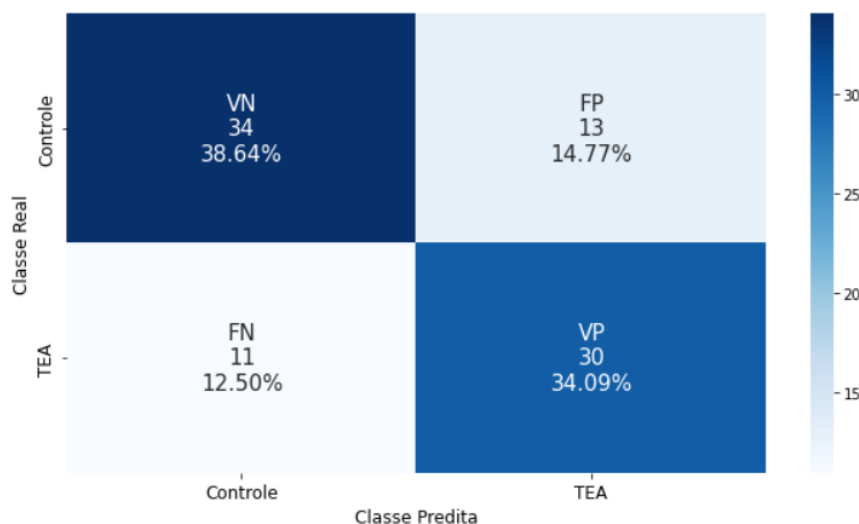
Tabela 10 - Resultado do modelo final do Pipeline advindo do TPOT.

TPOT_400	
<b>Acurácia</b>	0.7273
<b>Recall</b>	0.7317
<b>Precision</b>	0.6977
<b>F1</b>	0.7143

Fonte: Autoria Própria.

Na Figura 25, são apresentadas as quantidades de VN, VP, FN e FP para as predições efetuadas na porção de teste.

Figura 25 – Matriz de Confusão do modelo final com a base de dados completa de treino.



Fonte: Autoria Própria.

### 5.3.5 Comparação dos resultados dos modelos

A Revocação, ou o *recall*, é a frequência em que o classificador encontra os exemplos de uma classe. Ou seja, de todas as observações da classe TEA, quantas o modelo conseguiu prever corretamente? Essa métrica é importante para aplicações onde a presença de FN tem maior necessidade ser mapeada e evitada do que a de FP.

Para este estudo, considerou-se como importante que a maioria da classe TEA seja classificada da forma correta, para que os tratamentos ocorram o mais rápido possível e que o mínimo da classe TEA seja predita como Controle, para evitar o atraso no diagnóstico e consequentemente no tratamento. Isso posto, para os resultados a métrica considerada mais relevante será a recall.

Na tabela 11, tem-se a comparação das métricas dos modelos para cada Atlas, totalizando 4 resultados diferentes.

Tabela 11 - Comparação dos resultados dos modelos.

	SVC_200	TPOT_200	SVC_400	TPOT_400
<b>Acurácia</b>	0.7614	0.6364	0.7273	0.7273
<b>Recall</b>	0.7805	0.5366	0.7073	0.7317
<b>Precision</b>	0.7273	0.6286	0.7073	0.6977
<b>F1</b>	0.7529	0.5789	0.7073	0.7143

Fonte: Autoria Própria.

Dos resultados, tem-se que o SVC com atlas CC200 atingiu o melhor resultado dentre as combinações de modelos e atlas. Obteve uma revocação de 0.7805, o que significa que esse método mapeou 78,05% dos casos de TEA. Esse resultado em relação ao segundo melhor resultado, chega a uma diferença de 5 pontos.

Além disso, nota-se que as demais métricas do SVC com CC200 estão com bons resultados, a exemplo da precisão, que está com 0.7317, que significa que de todas as observações que o modelo predisse como verdadeiras 73,17% eram de fato verdadeiras. O que não acontece, por exemplo, no TPOT com CC200, onde se há uma diferença de quase 10 pontos da revocação para a precisão, além das duas métricas apresentarem resultados inferiores ao SVC com CC200.

## 6 CONCLUSÕES E PERSPECTIVAS

Por fim, foi possível determinar um modelo e um atlas que trouxesse bom desempenho, sendo alcançado o objetivo de propor um modelo e escolher um atlas que possibilite o auxílio-diagnóstico de TEA. Esse método de AM, pode ser um aliado no diagnóstico de TEA, acelerando o início de tratamentos e trazendo benefícios no desenvolvimento e na qualidade de vida dessas pessoas.

Há possibilidades para melhorias desses métodos, como por exemplo, tentar executar o TPOT por mais tempo, poderia prover um desempenho superior ao método escolhido.

Em trabalhos futuros, pode-se utilizar de métricas de grafos para construção de *features*, partindo da matriz de correlação resultante das séries temporais, estabelecendo-se um limite, obtendo-se a matriz de adjacência e extraíndo essas métricas como variáveis para a construção de modelos.

Outra possibilidade, seria a pesquisa na literatura de possíveis biomarcadores em pesquisas relacionados ao autismo que apontem para atividades não típicas de determinadas regiões do cérebro. Com isso, pode-se, através dos ROIs, identificar qual ROI estaria associado a cada biomarcador e utilizar essas regiões específicas para estudo e seleção de atributos que possam trazer mais informações e possivelmente chegar a um modelo com melhor desempenho.

Por último, uma ideia interessante seria identificar de que ROI vem a *feature* mais importante do modelo e com isso tentar identificar se aquela região é comum de observar alterações em pacientes com autismo, facilitando assim no diagnóstico e na busca de compreender melhor sobre o autismo.

## REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, C. L. **Diagnóstico de doenças mentais baseado em mineração de dados e redes complexas**. 2019. Universidade de São Paulo, São Carlos, 2019.

AMARAL, D. G.; SCHUMANN, C. M.; NORDAHL, C. W. Neuroanatomy of autism. **Trends in Neurosciences**, v. 31, n. 3, p. 137–145, mar. 2008.

AMERICAN PSYCHIATRIC ASSOCIATION. **Diagnostic and Statistical Manual of Mental Disorders**. [s.l.] American Psychiatric Association, 2013.

ASKI, A. S.; SOURATI, N. K. Proposed efficient algorithm to filter spam using machine learning techniques. **Pacific Science Review A: Natural Science and Engineering**, v. 18, n. 2, p. 145–149, jul. 2016.

CAMERON, C. et al. Towards Automated Analysis of Connectomes: The Configurable Pipeline for the Analysis of Connectomes (C-PAC). **Frontiers in Neuroinformatics**, v. 7, 2013.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, set. 1995.

CRADDOCK, R. C. et al. Disease state prediction from resting state functional connectivity. **Magnetic Resonance in Medicine**, v. 62, n. 6, p. 1619–1628, dez. 2009.

CRADDOCK, R. C. et al. A whole brain fMRI atlas generated via spatially constrained spectral clustering. **Human Brain Mapping**, v. 33, n. 8, p. 1914–1928, ago. 2012.

DIAO, Y. et al. PIRACY: An Optimized Pipeline for Functional Connectivity Analysis in the Rat Brain. **Frontiers in Neuroscience**, v. 15, p. 1–19, 26 mar. 2021.

DOSENBACH, N. U. F. et al. Prediction of Individual Brain Maturity Using fMRI. **Science**, v. 329, n. 5997, p. 1358–1361, 10 set. 2010.

DUCHAN, E.; PATEL, D. R. Epidemiology of Autism Spectrum Disorders. **Pediatric Clinics of North America**, v. 59, n. 1, p. 27–43, fev. 2012.

ELSABBAGH, M. et al. Global Prevalence of Autism and Other Pervasive Developmental Disorders. **Autism Research**, v. 5, n. 3, p. 160–179, jun. 2012.

FALCK-YTTER, T.; VON HOFSTEN, C. How special is social looking in ASD. In: **Progress in Brain Research**. 1. ed. [s.l.] Elsevier B.V., 2011. 189p. 209–222.

FIGUEIRA, A. Human brain networks: an investigation on cortical thickness resting-state fMRI and structural connectivity as assessed by tractography. 2013.

GESCHWIND, D. H.; LEVITT, P. Autism spectrum disorders: developmental



disconnection syndromes. **Current Opinion in Neurobiology**, v. 17, n. 1, p. 103–111, fev. 2007.

GOTHAM, K.; PICKLES, A.; LORD, C. Standardizing ADOS Scores for a Measure of Severity in Autism Spectrum Disorders. **Journal of Autism and Developmental Disorders**, v. 39, n. 5, p. 693–705, 12 maio 2009.

KHOSLA, M. et al. Machine learning in resting-state fMRI analysis. **Magnetic Resonance Imaging**, v. 64, p. 101–121, dez. 2019.

KIM, H. Y. **Introdução sobre fMRI**. Disponível em: <<http://www.lps.usp.br/hae/apostila/fmri.pdf>>. Acesso em: 16 mar. 2021.

KOBATAKE, H. et al. **Computational Anatomy Based on Whole Body Imaging**. Tokyo: Springer Japan, 2017.

LAM, K. S. L.; AMAN, M. G. The Repetitive Behavior Scale-Revised: Independent Validation in Individuals with Autism Spectrum Disorders. **Journal of Autism and Developmental Disorders**, v. 37, n. 5, p. 855–866, 17 maio 2007.

LANGDON, W. B. et al. Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications. In: [s.l: s.n.]p. 927–1028.

LORD, C. et al. Autism spectrum disorder. **The Lancet**, v. 392, n. 10146, p. 508–520, 2018.

MECHELLI, A. et al. Voxel-Based Morphometry of the Human Brain: Methods and Applications. **Current Medical Imaging Reviews**, v. 1, n. 2, p. 105–113, 1 jun. 2005.

NICE. National Institute for Health and Care Excellence, Autism spectrum disorder in adults: diagnosis and management. 2012.

NIELSEN, J. A. et al. Multisite functional connectivity MRI classification of autism: ABIDE results. **Frontiers in Human Neuroscience**, v. 7, 2013.

OLSON, R. S. et al. Evaluation of a tree-based pipeline optimization tool for automating data science. **GECCO 2016 - Proceedings of the 2016 Genetic and Evolutionary Computation Conference**, p. 485–492, 2016.

PAMPLONA, G. S. P. **Conectividade funcional no cérebro: uma análise das associações com desempenho intelectual e atenção sustentada usando imagens por ressonância magnética**. 2014. Universidade de São Paulo, Ribeirão Preto, 2014.

PINEDA-JARAMILLO, J. D. A review of Machine Learning (ML) algorithms used for modeling travel mode choice. **DYNA**, v. 86, n. 211, p. 32–41, 1 out. 2019.

PORTUGAL, I.; ALENCAR, P.; COWAN, D. The use of machine learning algorithms in recommender systems: A systematic review. **Expert Systems with Applications**,

v. 97, p. 205–227, maio 2018.

ROBINS, D. L. et al. The Modified Checklist for Autism in Toddlers: An Initial Study Investigating the Early Detection of Autism and Pervasive Developmental Disorders. **Journal of Autism and Developmental Disorders**, v. 31, n. 2, p. 131–144, 2001.

SALMON, C. E. G.; LEONI, R. F. Conectividade Funcional Cerebral utilizando Técnicas de Imagens por Ressonância Magnética. **Revista Brasileira de Física Médica**, v. 13, n. 1, p. 66, 2019.

SMITH, S. M. Overview of fMRI analysis. **The British Journal of Radiology**, v. 77, n. suppl\_2, p. S167–S175, dez. 2004.

SPRAWLS, P. Magnetic Resonance Image Characteristics. In: [s.l: s.n.]

SZAMEITAT, A. J.; SHEN, S.; STERR, A. The functional magnetic resonance imaging (fMRI) procedure as experienced by healthy participants and stroke patients – A pilot study. **BMC Medical Imaging**, v. 9, n. 1, p. 14, 31 dez. 2009.

WEN, D. et al. Deep Learning Methods to Process fMRI Data and Their Application in the Diagnosis of Cognitive Impairment: A Brief Overview and Our Opinion. **Frontiers in Neuroinformatics**, v. 12, 26 abr. 2018.

YANG, X.; SCHRADER, P. T.; ZHANG, N. A deep neural network study of the ABIDE repository on autism spectrum classification. **International Journal of Advanced Computer Science and Applications**, v. 11, n. 4, p. 1–6, 2020.

YUMI, A. **FMRI**. Disponível em: <[https://amandayumi.netlify.app/aulas/8\\_fmri/](https://amandayumi.netlify.app/aulas/8_fmri/)>. Acesso em: 16 mar. 2021.

ZWAIGENBAUM, L. et al. Behavioral manifestations of autism in the first year of life. **International Journal of Developmental Neuroscience**, v. 23, n. 2–3, p. 143–152, 2 abr. 2005.

## APÊNDICE

### CÓDIGO EDA CC200/ CC400

```
# %matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from nilearn.datasets import fetch_abide_pcp
pd.set_option('display.max_columns', 200)

atlas='EDA200/'

# n_subjects

abide=fetch_abide_pcp(pipeline='cpac', derivatives=['rois_cc200'], band_pass_filtering=False,
global_signal_regression=False, quality_checked=True)

print(abide.rois_cc200[0].shape)

abide_pheno=pd.DataFrame(abide.phenotypic)

abide_pheno.shape

# Balanceamento das classes

# Quantidade de cada classe
# 1=autism
# 2=control

abide_pheno.DX_GROUP.value_counts()

# % de cada classe
abide_pheno.DX_GROUP.value_counts(normalize=True)

# 1=autism
# 2=control
```

```
plt.figure(figsize=(10,5))
ax=sns.countplot(x='DX_GROUP',data=abide_pheno, palette='Blues')

plt.ylabel('Quantidade', fontsize=12)
plt.xlabel('', fontsize=12)
ax.set_xticklabels(['TEA','Controle'], fontsize=12)
ax.set(ylim=(0, 500))

for p in ax.patches:
    ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.36, p.get_height()+5), fontsize=11)

plt.savefig(atlas+'1.distribuicao_classes.png')
```

# Mulheres e Homens

# 1=male

# 2=female

```
plt.figure(figsize=(10,5))
ax=sns.countplot(x='DX_GROUP', data=abide_pheno, hue='SEX')

plt.legend(title="Gênero")
plt.xlabel('', fontsize=12)
plt.ylabel('Quantidade', fontsize=12)
ax.set_xticklabels(['TEA','Controle'], fontsize=12)
ax.set(ylim=(0, 500))

for p in ax.patches:
    ax.annotate('{:.0f}'.format(p.get_height()), (p.get_x()+0.17, p.get_height()+5), fontsize=11)

plt.savefig(atlas+'2.distribui_genero.png')
```

# Percentual de cada sexo no espectro

```
abide_pheno[['SEX','DX_GROUP']].groupby(['SEX']).count()
```

### Teste Qui-Quadrado

<https://ichi.pro/pt/como-executar-o-teste-qui-quadrado-em-python-86446368105629>

```
from scipy.stats import chi2_contingency
```

```

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# visualizando tabela de contingência
# 1=male
# 2=female
contingency_img=pd.crosstab(abide_pheno['SEX'], abide_pheno['DX_GROUP'], normalize='index')
plt.figure(figsize=(7,5))
ax=sns.heatmap(contingency_img, annot=True, cmap="YlGnBu")

ax.set_xticklabels(['Homem','Mulher'], fontsize=12)
ax.set_yticklabels(['Homem','Mulher'], fontsize=12)
plt.xlabel("", fontsize=12)
plt.ylabel("", fontsize=12)

plt.savefig(atlas+'3.tabela_contingencia.png')

# criando tabela de contingência
contingency= pd.crosstab(abide_pheno['SEX'], abide_pheno['DX_GROUP'])
contingency

# passando tabela de contingência para a função chi2
c, p, dof, expected = chi2_contingency(contingency)

# p-value
print(p)

# Distribuição de faixa etária por classe

# distributions and plot the estimated PDF over the data / distribution with a kernel density estimate

plt.figure(figsize=(10,5))
# sns.distplot(abide_pheno['AGE_AT_SCAN']).set_title("Distribuição de Idade")
sns.distplot(abide_pheno['AGE_AT_SCAN'])
plt.xlabel('Idade', fontsize=12)
plt.ylabel('Densidade', fontsize=12)

plt.savefig(atlas+'4.distribuição_idade.png')

```

```
# distribuição de idades TEA
```

```
plt.figure(figsize=(10,5))
sns.distplot(abide_pheno[abide_pheno.DX_GROUP==1][['AGE_AT_SCAN']])
plt.xlabel('Idade - Amostra TEA', fontsize=12)
plt.ylabel('Densidade', fontsize=12)

plt.savefig(atlas+'5.distribuição_idade_tea.png')
```

```
# distribuição de idades Controle
```

```
plt.figure(figsize=(10,5))
sns.distplot(abide_pheno[abide_pheno.DX_GROUP==2][['AGE_AT_SCAN']])
plt.xlabel('Idade', fontsize=12)
plt.ylabel('Densidade - Amostra Controle', fontsize=12)

plt.savefig(atlas+'6.distribuição_idade_controle.png')
```

```
# Matriz conectividade
```

```
from nilearn.connectome import ConnectivityMeasure

conn_est2 = ConnectivityMeasure(kind='correlation')
conn_matrice2 = conn_est2.fit_transform(abide.rois_cc200)

X=conn_matrice2
X.shape
```

```
# TEA
```

```
abide_pheno['DX_GROUP'][0]

plt.figure(figsize=(15,8))
plt.imshow(conn_matrice2[0], cmap='RdBu_r')
plt.colorbar()
# plt.title('Matriz de conectividade - Atlas CC200')

plt.savefig(atlas+'7.matriz_conectidade.png')
```

```
# Outras Análises
```

```
# Quantidade de amostras

test=np.array(abide.rois_cc200)
test.shape

abide_pheno.AGE_AT_SCAN.shape[0] # todos os dados têm info de idade

abide_pheno.describe()

# Primeiras análises

def count_unique(df): #remover features que são únicas
    print("Quantidade de valores únicos para cada feature no conjunto de treinamento")
    for i in df.columns:
        print(f"{i}: {df[i].nunique()}")

count_unique(abide_pheno)
```

## **CÓDIGO TPOT CC200**

```
# Importar Bibliotecas

import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

#%matplotlib inline

import random
random.seed(19)

# Leitura das Bases

from nilearn.datasets import fetch_abide_pcp

# n_subjects
```

```

abide = fetch_abide_pcp(pipeline='cpac', derivatives=['rois_cc200'], band_pass_filtering=False,
global_signal_regression=False, quality_checked=True)

abide_pheno = pd.DataFrame(abide.phenotypic)

test=np.array(abide.rois_cc200)
test.shape

print(abide.rois_cc200[0].shape)

from nilearn.connectome import ConnectivityMeasure

conn_est2 = ConnectivityMeasure(kind='correlation', vectorize=True, discard_diagonal=True)
conn_matrice2 = conn_est2.fit_transform(abide.rois_cc200)

X=conn_matrice2
X.shape

X.shape

abide_pheno['DX_GROUP']=np.where(abide_pheno['DX_GROUP']==1,1,0)

# 1=autism
# 2=control

y=abide_pheno['DX_GROUP']

# Separar dado para validação

from sklearn.model_selection import train_test_split

# Separa 10% para teste
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1,stratify=y, random_state=19)

# Tpot

import tpot
from tpot import TPOTClassifier

#print('tpot: %s' % tpot._version_)

```



```
tpot = TPOTClassifier(scoring='f1_score', verbosity=2, max_eval_time_mins=5, max_time_mins=1440,
population_size=100, cv=5, config_dict='TPOT cuML', memory='/media/pc/Barracuda/tpot_memory/')

```

```
tpot.fit(X_train, y_train)

```

```
tpot.score(X_test, y_test)

```

```
tpot.export('tpot_pipeline_20024hs.py')

```

```
y_pred = tpot.predict(X_test)

```

```
from sklearn.metrics import average_precision_score, recall_score, precision_score, f1_score,
accuracy_score, roc_auc_score

```

```
average_precision_score(y_test, y_pred)

```

```
recall_score(y_test, y_pred)

```

```
precision_score(y_test, y_pred)

```

```
f1_score(y_test, y_pred)

```

```
accuracy_score(y_test, y_pred)

```

```
roc_auc_score(y_test, y_pred)

```

## CÓDIGO TPOT CC400

```
# Importar Bibliotecas

```

```
import numpy as np

```

```
import pandas as pd

```

```
import warnings

```

```
warnings.filterwarnings("ignore")

```

```
#%%matplotlib inline

```

```
import random

```

```
random.seed(19)

```

```
# Leitura das Bases

```

```

from nilearn.datasets import fetch_abide_pcp

# n_subjects

abide = fetch_abide_pcp(pipeline='cpac', derivatives=['rois_cc400'], band_pass_filtering=False,
global_signal_regression=False, quality_checked=True)

abide_pheno = pd.DataFrame(abide.phenotypic)

test=np.array(abide.rois_cc400)
test.shape

print(abide.rois_cc400[0].shape)

from nilearn.connectome import ConnectivityMeasure

conn_est2 = ConnectivityMeasure(kind='correlation', vectorize=True, discard_diagonal=True)
conn_matrice2 = conn_est2.fit_transform(abide.rois_cc400)

X=conn_matrice2
X.shape

X.shape

abide_pheno['DX_GROUP']=np.where(abide_pheno['DX_GROUP']==1,1,0)

# 1=autism
# 2=control

y=abide_pheno['DX_GROUP']

# Separar dado para validação

from sklearn.model_selection import train_test_split

# Separa 10% para teste
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.1,stratify=y, random_state=19)

# Tpot

```

```

import tpot
from tpot import TPOTClassifier

#print('tpot: %s' % tpot._version_)

tpot = TPOTClassifier(scoring='f1_score', verbosity=2, max_eval_time_mins=5, max_time_mins=1440,
population_size=100, cv=5, config_dict='TPOT cuML', memory='/media/pc/Barracuda/tpot_memory/')

tpot.fit(X_train, y_train)

tpot.score(X_test, y_test)

tpot.export('tpot_pipeline_40024hs.py')

y_pred = tpot.predict(X_test)

from sklearn.metrics import average_precision_score, recall_score, precision_score, f1_score,
accuracy_score, roc_auc_score

average_precision_score(y_test, y_pred)
recall_score(y_test, y_pred)
precision_score(y_test, y_pred)
f1_score(y_test, y_pred)
accuracy_score(y_test, y_pred)
roc_auc_score(y_test, y_pred)

```

## **CÓDIGOS DE TREINO E ANÁLISE DOS RESULTADOS DO SVC CC200, TPOT CC200, SVC CC400, TPOT CC400**

```
# Importar Bibliotecas
```

```
import numpy as np
import pandas as pd
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
%matplotlib inline
```

```
import random
```

```

random.seed(19)

# Funções

# FUNÇÃO AVALIA MODELO

def avalia(model_name, y_test, y_pred):
    df = pd.DataFrame()
    from sklearn.metrics import recall_score, precision_score, f1_score, accuracy_score

    df.loc['Acurácia', model_name]=accuracy_score(y_test, y_pred)
    df.loc['Recall', model_name]=recall_score(y_test, y_pred)
    df.loc['Precision', model_name]=precision_score(y_test, y_pred)
    df.loc['F1', model_name]=f1_score(y_test, y_pred)

    df=round(df,4)
    return df

# FUNÇÃO AVALIA ACC

df_metrics_acc = pd.DataFrame()

def avalia_acc(df, model_name, y_test, y_pred):

    from sklearn.metrics import recall_score, precision_score, f1_score, accuracy_score

    df.loc['Acurácia', model_name]=accuracy_score(y_test, y_pred)
    df.loc['Recall', model_name]=recall_score(y_test, y_pred)
    df.loc['Precision', model_name]=precision_score(y_test, y_pred)
    df.loc['F1', model_name]=f1_score(y_test, y_pred)

    df=round(df,4)
    return df

import seaborn as sns
from sklearn.metrics import confusion_matrix
class_names = ['TEA', 'Controle']

def plot_confusion(cf_matrix, model_name):

```

```

plt.figure(figsize=(9,5))

group_names = ['VN','FP','FP','VP']
group_counts = ["{0:0.0f}".format(value) for value in
                 cf_matrix.flatten()]

group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten()/np.sum(cf_matrix)]

labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)

ax=sns.heatmap(cf_matrix, annot=labels, fmt="", cmap='Blues',annot_kws={"fontsize":15})
classes=['Controle', 'TEA']
ax.set_xticklabels(classes, fontsize=12, fontname="Special Elite")
ax.set_yticklabels(classes, fontsize=12, fontname="Special Elite")

plt.tight_layout()
plt.ylabel('Classe Real',fontsize=12,fontname="Special Elite")
plt.xlabel('Classe Predita',fontsize=12,fontname="Special Elite")

plt.savefig('drive/My Drive/00.TCC/00_Entrega/matrizes/'+model_name+'.png')

# Leitura das Bases - CC200

from nilearn.datasets import fetch_abide_pcp

abide = fetch_abide_pcp(pipeline='cpac', derivatives=['rois_cc200'],
band_pass_filtering=False,global_signal_regression=False, quality_checked=True)

abide_pheno = pd.DataFrame(abide.phenotypic)

abide.rois_cc200[0].shape

# Matriz de Conectividade

from nilearn.connectome import ConnectivityMeasure

conn_est2 = ConnectivityMeasure(kind='correlation', vectorize=True, discard_diagonal=True)

```

```

conn_matrice2 = conn_est2.fit_transform(abide.rois_cc200)
X=conn_matrice2

X.shape

# 1=autism
# 2=control --->0
abide_pheno['DX_GROUP']=np.where(abide_pheno['DX_GROUP']==1,1,0)
y=abide_pheno['DX_GROUP']

y.shape

# Split

# Separar dado para teste
from sklearn.model_selection import train_test_split

# Separa 10% para teste
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.1, stratify=y, random_state=19)

abide.rois_cc200[0].shape

abide_pheno.shape

# 1. SVC CC200

from sklearn.svm import LinearSVC

clf_svc200 = LinearSVC(penalty='l2', C=0.1, dual=True, tol=0.0001, max_iter=10000,
random_state=19)

# Cross validated

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate

scores = cross_validate(clf_svc200, X_train, y_train, cv=5, scoring=['accuracy','recall','precision','f1'])

resultados = pd.DataFrame(scores)

```

```

data = {
    "Média" :
[resultados['test_accuracy'].mean(),resultados['test_recall'].mean(),resultados['test_precision'].mean(),re
sultados['test_f1'].mean()],
    "Desvio Padrão" :
[resultados['test_accuracy'].std(),resultados['test_recall'].std(),resultados['test_precision'].std(),resultado
s['test_f1'].std()]
}

```

```

df_cv_svc200 = pd.DataFrame(data, index = ['Acurácia','Recall','Precision','F1'])
df_cv_svc200.round(4)

```

```
# Modelo Final
```

```
clf_svc200.fit(X_train, y_train)
```

```
y_pred_train_svc200 = clf_svc200.predict(X_train)
```

```
y_pred_test_svc200 = clf_svc200.predict(X_test)
```

```
df_svc200=avalia('SVC_200', y_test, y_pred_test_svc200)
```

```
df_svc200
```

```
avalia_acc(df_metrics_acc, 'SVC_200', y_test, y_pred_test_svc200)
```

```
df_metrics_acc
```

```
# 2. TPOT 200
```

```
from sklearn.pipeline import make_pipeline
```

```
from sklearn.preprocessing import Normalizer
```

```
from xgboost import XGBClassifier
```

```

clf_tpot200 = XGBClassifier(alpha=1, learning_rate=0.01, max_depth=9, min_child_weight=4,
n_estimators=100, n_jobs=1, subsample=0.55, tree_method="gpu_hist", verbosity=0)

```

```
# Cross validated
```

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn.model_selection import cross_validate
```

```
scores = cross_validate(clf_tpot200, X_train, y_train, cv=5, scoring=['accuracy','recall','precision','f1'])
```

```

resultados = pd.DataFrame(scores)

data = {
    "Média" :
[resultados['test_accuracy'].mean(),resultados['test_recall'].mean(),resultados['test_precision'].mean(),re
sultados['test_f1'].mean()],
    "Desvio Padrão" :
[resultados['test_accuracy'].std(),resultados['test_recall'].std(),resultados['test_precision'].std(),resultado
s['test_f1'].std()]
}

df_cv_tpot200 = pd.DataFrame(data, index = ['Acurácia','Recall','Precision','F1'])
df_cv_tpot200.round(4)

# Modelo Final

clf_tpot200.fit(X_train, y_train)

y_pred_train_tpot200 = clf_tpot200.predict(X_train)
y_pred_test_tpot200 = clf_tpot200.predict(X_test)

df_tpot200=avalia('TPOT_200', y_test, y_pred_test_tpot200)
df_tpot200

avalia_acc(df_metrics_acc, 'TPOT_200', y_test, y_pred_test_tpot200)
df_metrics_acc

# Leitura das Bases - CC400

from nilearn.datasets import fetch_abide_pcp

abide = fetch_abide_pcp(pipeline='cpac', derivatives=['rois_cc400'], band_pass_filtering=False,
global_signal_regression=False, quality_checked=True)

abide_pheno = pd.DataFrame(abide.phenotypic)

abide.rois_cc400[0].shape

#Matriz de Conectividade

```



```

from nilearn.connectome import ConnectivityMeasure

conn_est2 = ConnectivityMeasure(kind='correlation', vectorize=True, discard_diagonal=True)
conn_matrice2 = conn_est2.fit_transform(abide.rois_cc400)
X=conn_matrice2

X.shape

# 1=autism
# 2=control --->0
abide_pheno['DX_GROUP']=np.where(abide_pheno['DX_GROUP']==1,1,0)
y=abide_pheno['DX_GROUP']

y.shape

# Split

# Separar dado para teste
from sklearn.model_selection import train_test_split

# Separa 10% para teste
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.1, stratify=y, random_state=19)

# 3. SVC 400

from sklearn.svm import LinearSVC

clf_svc400 = LinearSVC(penalty='l2', C=0.1, dual=True, tol=0.0001, max_iter=10000,
random_state=19)

# Cross validated

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate

scores = cross_validate(clf_svc400, X_train, y_train, cv=5, scoring=['accuracy','recall','precision','f1'])

resultados = pd.DataFrame(scores)

```

```
data = {
    "Média" :
    [resultados['test_accuracy'].mean(),resultados['test_recall'].mean(),resultados['test_precision'].mean(),re
    sultados['test_f1'].mean()],
    "Desvio Padrão" :
    [resultados['test_accuracy'].std(),resultados['test_recall'].std(),resultados['test_precision'].std(),resultado
    s['test_f1'].std()]
}
```

```
df_cv_svc400 = pd.DataFrame(data, index = ['Acurácia','Recall','Precision','F1'])
df_cv_svc400.round(4)
```

```
# Modelo Final
```

```
clf_svc400.fit(X_train, y_train)
```

```
y_pred_train_svc400 = clf_svc400.predict(X_train)
```

```
y_pred_test_svc400 = clf_svc400.predict(X_test)
```

```
df_svc400=avalia('SVC_400', y_test, y_pred_test_svc400)
```

```
df_svc400.round(4)
```

```
avalia_acc(df_metrics_acc, 'SVC_400', y_test, y_pred_test_svc400)
```

```
df_metrics_acc
```

```
# 4. TPOT 400
```

```
from sklearn.pipeline import make_pipeline
```

```
from sklearn.preprocessing import Normalizer
```

```
from xgboost import XGBClassifier
```

```
from tpot.builtins import StackingEstimator
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
clf_tpot400 = make_pipeline(
```

```
    StackingEstimator(estimator=LogisticRegression(C=0.001, penalty="l2")),
```

```
    MinMaxScaler(),
```

```
    XGBClassifier(alpha=1, learning_rate=0.01, max_depth=3, min_child_weight=2, n_estimators=100,
    subsample=0.45, tree_method="gpu_hist", verbosity=0)
```

```
)
```

```

# Cross validated

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate

scores = cross_validate(clf_tpot400, X_train, y_train, cv=5, scoring=['accuracy','recall','precision','f1'])

resultados = pd.DataFrame(scores)

data = {
    "Média" :
[resultados['test_accuracy'].mean(),resultados['test_recall'].mean(),resultados['test_precision'].mean(),re
sultados['test_f1'].mean()],
    "Desvio Padrão" :
[resultados['test_accuracy'].std(),resultados['test_recall'].std(),resultados['test_precision'].std(),resultado
s['test_f1'].std()]
}

df_cv_tpot400 = pd.DataFrame(data, index = ['Acurácia','Recall','Precision','F1'])
df_cv_tpot400.round(4)

# Modelo Final

clf_tpot400.fit(X_train, y_train)

y_pred_train_tpot400 = clf_tpot400.predict(X_train)
y_pred_test_tpot400 = clf_tpot400.predict(X_test)

df_tpot400=avalia('TPOT_400', y_test, y_pred_test_tpot400)
df_tpot400

avalia_acc(df_metrics_acc, 'TPOT_400', y_test, y_pred_test_tpot400)
df_metrics_acc

round(df_metrics_acc,4)

# Matriz de confusão

import matplotlib.pyplot as plt

```

```
cnf_matrix1 = confusion_matrix(y_test, y_pred_test_svc200)
```

```
print(cnf_matrix1)
```

```
plot_confusion(cnf_matrix1, model_name='SVC_200')
```

```
cnf_matrix2 = confusion_matrix(y_test, y_pred_test_svc400)
```

```
print(cnf_matrix2)
```

```
plot_confusion(cnf_matrix2, model_name='SVC_400')
```

```
cnf_matrix3 = confusion_matrix(y_test, y_pred_test_tpot200)
```

```
print(cnf_matrix3)
```

```
plot_confusion(cnf_matrix3, model_name='TPOT_200')
```

```
cnf_matrix4 = confusion_matrix(y_test, y_pred_test_tpot400)
```

```
print(cnf_matrix4)
```

```
plot_confusion(cnf_matrix4, model_name='TPOT_400')
```