Projeto Machine Learning 2022.2

Automação de metrificação e padronização de modelos

Contexto 1. Empresas de IA

Muitos times e muitos projetos - cada projeto tem suas ferramentas, seus métodos de avaliação e metrificação. Os modelos tem estruturas diferentes e cada pipeline de integração de dados tem que ser construída de forma distinta.

Exemplo: Em um projeto de detecção de anomalias, um engenheiro cria uma pipeline com *kedro,* outro engenheiro cria uma pipeline com uma ferramenta nova que roda em kubernetes e um terceiro membro da equipe cria a pipeline em um jupyter notebook que só roda na máquina dele.

Contexto 2. Academia e competições de IA

Muitos alunos ou participantes criam modelos a partir dos mesmos dados porém cada modelo terá métodos distintos e não há padronização. Para compará-los entre si, tem que haver um trabalho manual de compreensão de cada um.

Exemplo: Em um desafio proposto pelo insper, alunos terão que prever algum acontecimento. Cada aluno cria o modelo de um jeito e em uma estrutura (Jupyter, script, contêiner) e com uma instrução específica de como rodar.

A proposta

Uma ferramenta que facilite a padronização de modelos para comparação, análise, mesclagem e rankeamento.

Playbooks

Playbooks são desafios de machine learning. Seja com dados de um cliente, seja com dados fictícios com intuito acadêmico.

Playbooks são delimitados com:
Nome
Métricas que serão utilizadas na avaliação
Dados (públicos e internos)

Dados públicos podem ser acessados por qualquer um, enquanto os dados internos não são disponibilizados e são usados pela ferramenta com fins de metrificação. Caso esses dados fossem vazados teríamos data leaking.

A criação de um playbook é dada por uma chamada de API

init

Para dar início a um estudo, um engenheiro roda o comando

```
camino init --playbook <NOME DO PLAYBOOK>
```

Esse comando cria um diretório no “working directory” com o arquivo “model.py” e a pasta “data” com os dados públicos de treino e validação do playbook, além do arquivo “meta.yml” com metadados e um requirements.txt.

```
# working dir

model.py
meta.yml
data
--train.csv
--validate.csv
requirements.txt
```

submit

Para validar um modelo, basta rodar o comando

```
camino submit
```

Esse comando manda uma requisição para a API enviando o ID do playbook trabalhado e a pasta com o modelo.

O servidor então aloca uma máquina na AWS (ou qualquer outra nuvem) e copia a pasta com o modelo pra essa instância. Ao invés de copiar a pasta “data” com os dados públicos, dessa vez irá mandar os dados internos de treino e validação.

O modelo será treinado e salvo em PKL. As métricas, junto com mais alguns dados como tempo de treinamento serão armazenadas em DB.

test

Para testar um modelo em desenvolvimento, basta rodar o comando

```
camino test
```

Esse comando testa o modelo com os dados de teste públicos. que já estão na máquina local.

compile

`camino compile --playbook <NOME DO PLAYBOOK>`

Esse comando treina um `VotingClassifier` (ou `Regressor`) com os 5 melhores modelos para cada uma das métricas de avaliação definidas no playbook.

funcionamento “model.py”

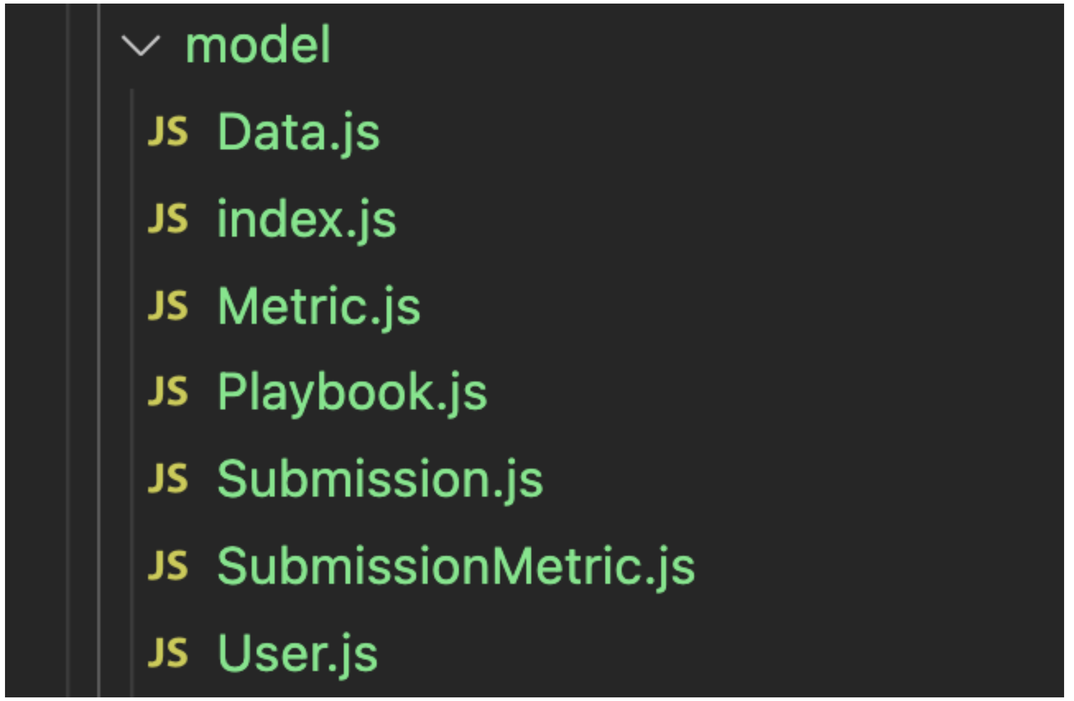
```
class Model:
    def __init__(self):
        self.model = None
        self.data = pd.read_csv(“/data/train.csv”)
        self.X = [<X columns here>]
        self.y = [<y columns here>]

    def train(self):
        # inits, trains and returns a model
        pass

    def predict(self, items):
        # returns list of predicted values
        pass

    def save(self):
        with open(model.pickle', 'wb') as f:
            pickle.dump(self, f)
```

tabelas e dados



Rodando na sua máquina

Instalação

Criação de playbooks

Dando início a um experimento

Testando o seu experimento

Instalação da CLI

Clonando o repositório

```
git clone https://github.com/jzsiggy/camino.git/
```

Instalando dependências da CLI

```
cd CLI \
pip install -r requirements.txt
```

Adicionando aliases

```
vim ~/.bashrc { .zshrc caso utilize ZSH }
alias camino="python3 ~/Desktop/INSPER/6o_semestre/ml/camino/cli/camino.py"
**utilizar o PATH para o seu camino.py no comando acima
source ~/.bashrc
```

Criação de um novo playbook

No diretório “`/camino/playbooks`” crie uma nova pasta com o nome do seu playbook

Dentro do novo playbook, crie um pasta “`/data`” com dois arquivos: “`train.csv`” e “`validation.csv`”. Esse arquivos devem conter os CSVs de treino e teste .

Crie o arquivo “`.test.py`”. Esse arquivo é padrão de uso interno da ferramenta. Pode copiar os conteúdos do arquivo “`.test.py`” de outro playbook e colar nele.

Crie o arquivo “`meta.yml`”. Esse arquivo deverá conter o atributo “`playbook`” com o nome do playbook e uma lista “`metrics`” com as metricas de avaliação.

Por último, o arquivo “`model.py`” deverá ser criado. Ele pode ser copiado de outro playbook, porém o “`__init__`” deverá ser alterado com as colunas X e y respectivas do novo dataset.

Dando início a um novo experimento

```
camino init --playbook iris
```

Com o diretório do novo playbook criado. O usuário deverá implementar os métodos train e predict.

No método `train`, o modelo deve ser iniciado e treinado enquanto no método `predict`, o modelo deverá fazer uma previsão tendo como entrada os argumentos do método.

O método de predict pode criar uma estratégia própria de predição (não precisa apenas chamar o método predict do modelo treinado)

```
def predict(self, items):
    for item in items:
        if item.feature1 > 5:
            self.specific_model_1.predict(item)
        else:
            self.specific_model_2.predict(item)
```

Testando o seu experimento

```
camino test
```

Esse comando irá testar o seu modelo e avaliá-lo pelas métricas definidas no playbook.

Demo

Classification

Link

Regression

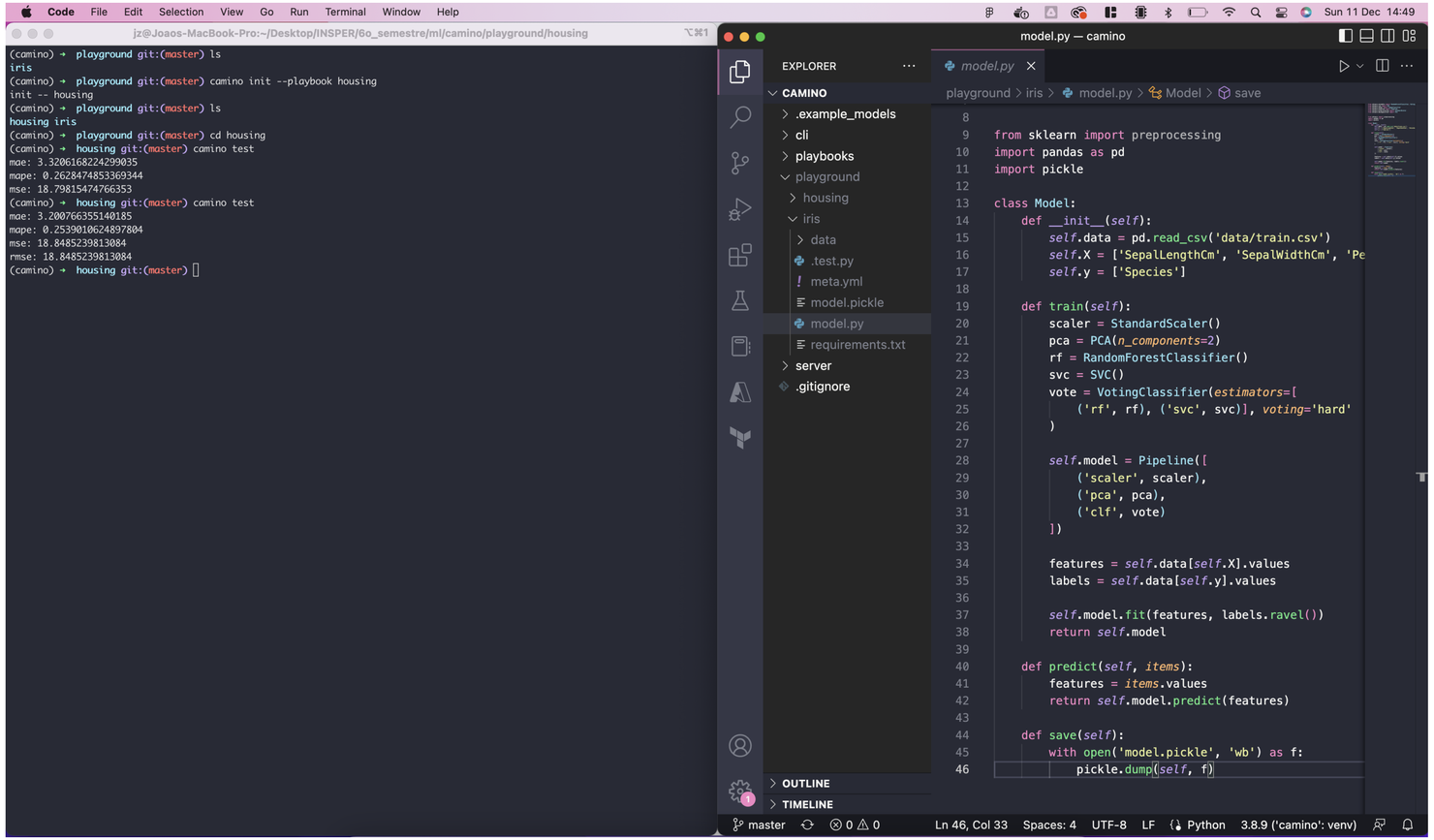
link

Próximos passos

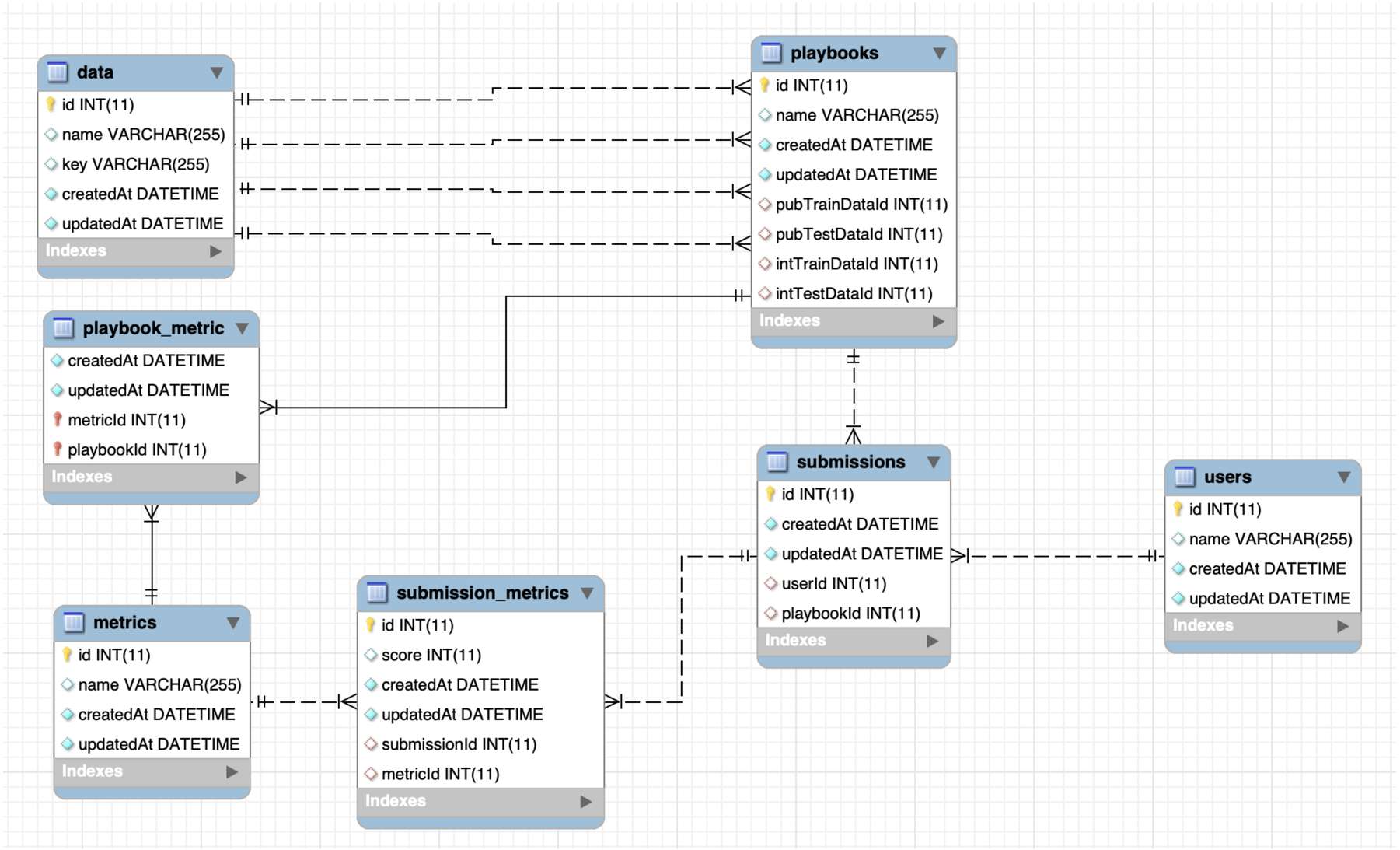
camino submit

camino compile

API



DB Strucure


data
id INT(11)
name VARCHAR(255)
key VARCHAR(255)
createdAt DATETIME
updatedAt DATETIME
Indexes
playbooks
id INT(11)
name VARCHAR(255)
createdAt DATETIME
updatedAt DATETIME
pubTrainDataId INT(11)
pubTestDataId INT(11)
intTrainDataId INT(11)
intTestDataId INT(11)
Indexes
playbook_metric
createdAt DATETIME
updatedAt DATETIME
metricId INT(11)
playbookId INT(11)
Indexes
submissions
id INT(11)
createdAt DATETIME
updatedAt DATETIME
userId INT(11)
playbookId INT(11)
Indexes
users
id INT(11)
name VARCHAR(255)
createdAt DATETIME
updatedAt DATETIME
Indexes
metrics
id INT(11)
name VARCHAR(255)
createdAt DATETIME
updatedAt DATETIME
Indexes
submission_metrics
id INT(11)
score INT(11)
createdAt DATETIME
updatedAt DATETIME
submissionId INT(11)
metricId INT(11)
Indexes