Artigo
Danusa Calixto · Out. 18, 2022 17min de leitura

Open Exchange

Machine Learning Sustentável para o Concurso de Interoperabilidade da InterSystems

Olá a todos, sou um estudante francês que acabou de chegar em Praga para um intercâmbio acadêmico no meu quinto ano da faculdade de engenharia e aqui está minha participação no concurso de interoperabilidade.

Não tive muito tempo para programar desde a mudança da França para Praga e estou participando sozinho, então decidi criar um projeto que é mais um modelo do que um aplicativo.

Queria participar já que minha área (Ciência de Dados e IA) não é geralmente associada à sustentabilidade, e o concurso era uma maneira de me expressar nesse tema importante que é a sustentabilidade e o meio ambiente.

Como você sabe, a Inteligência Artificial está cada vez mais popular, com várias firmas conhecidas tentando seguir o movimento e vender ferramentas para criar, treinar e usar modelos de machine learning com facilidade. Tudo isso é prático e fácil, mas tem um custo, financeiro e também ambiental.

Treinar modelos enormes repetidamente pode exigir muitos recursos e produzir uma grande quantidade de CO2. Os supercomputadores são executados durante vários dias, e o período e o tamanho dos modelos estão aumentando exponencialmente, tomando mais espaço do que nunca. Todo esse esforço para ter alguma melhoria no desempenho, que em alguns casos não é nem garantida.

É claro que várias firmas precisam desse processo, em que até mesmo 0,1% de melhoria na precisão de um modelo pode salvar milhares de vidas. Por isso, esse modelo foi criado para usos mais comuns.

No entanto, como tive a oportunidade de trabalhar com Processamento de Linguagem Natural (PLN) ou Classificação de Imagens, percebi que alguns modelos e ferramentas já são quase utilizáveis no estado em que se encontram e podem nos ajudar a economizar centenas de horas de treinamento e, consequentemente, emissões de CO2 e consumo de eletricidade.

Por isso, decidi criar um modelo usando as tecnologias da InterSystems para desenvolver uma solução interoperável que resolvesse alguns problemas de sustentabilidade ao permitir facilmente que você, em alguns cliques, faça o download único de modelos pré-treinados da Internet, use de acordo com sua própria necessidade e, claro, ajuste esses modelos pré-treinados com novo conteúdo disponível na base de dados IRIS e adicione conteúdo ao modelo existente.

Dessa forma, no modelo, estamos pegando um modelo PLN, testando e treinando com dados para criar cinco novos rótulos no modelo para classificar a avaliação da internet.

Portanto, ao fazer isso, o resultado (se você tiver tempo e algum poder computacional) é um ótimo modelo que pode ser usado para prever a nota de avaliação da internet, que não tem custo e emite uma quantidade pequena de CO2.

Consulte o GitHub e a postagem do Open Exchange vinculada a este artigo.

Ou veja o ReadMe aqui:

1. Contest-Sustainability

Este modelo usa as tecnologias da InterSystems para desenvolver uma solução interoperável e resolver alguns problemas de sustentabilidade ao permitir facilmente que você, em alguns cliques, faça o download único de modelos pré-treinados da Internet, use de acordo com sua própria necessidade e, claro, ajuste esses modelos pré-treinados com novo conteúdo disponível na base de dados IRIS e adicione conteúdo ao modelo existente.

Neste exemplo, estamos pegando um modelo PLN, testando e treinando com dados para criar cinco novos rótulos no modelo para classificar a avaliação da internet. No processo, poupamos uma grande quantidade de recursos e emissões de CO2.

Veja alguns modelos de exemplo que você pode testar: https://huggingface.co/gpt2
https://huggingface.co/Jean-Baptiste/camembert-ner
https://huggingface.co/bert-base-uncased
https://huggingface.co/facebook/detr-resnet-50
https://huggingface.co/facebook/detr-resnet-50-panoptic

ÍNDICE:

2. Instalação

2.1. Início da produção

Na pasta "contest-sustainability", abra um terminal e insira:

```
docker-compose up
```

Na primeira vez, talvez leve alguns minutos para o build correto da imagem e a instalação de todos os módulos necessários para o Python.

2.2. Acesso à produção

Seguindo este link, acesse a produção: Acessar a produção

2.3. Encerramento da produção

```
docker-compose down
```

3. Como funciona

Por enquanto, alguns modelos talvez não funcionem com essa implementação, já que tudo é feito automaticamente, ou seja, não importa o modelo de entrada, tentaremos fazer com que funcione usando a biblioteca transformers pipeline.

Pipeline é uma ferramenta poderosa da equipe HuggingFace que analisa a pasta em que o modelo foi transferido e entende qual biblioteca deve usar entre PyTorch, Keras, Tensorflow ou JAX. Em seguida, ela carrega esse modelo usando AutoModel.
Então, ao inserir a tarefa, o pipeline sabe o que fazer com o modelo, tokenizer ou até extrator de características nessa pasta e gerencia a entrada automaticamente, tokeniza, processa, transfere para o modelo e retorna um resultado decodificado que podemos usar diretamente.

4. API HuggingFace

Algumas pessoas ou sistemas não conseguem fazer o download de modelos ou usá-los devido a restrições. Por isso, é possível usar a API HuggingFace e chamar alguns modelos diretamente através desse serviço.
Veja uma explicação mais simples:

Primeiro, você precisa iniciar a demonstração, usando o botão verde Start, ou use Stop e Start novamente para aplicar as mudanças nas configurações.

Em seguida, ao clicar na operação Python.HFOperation escolhida e selecionar na guia à direita action, você pode aplicar test à demonstração.

Na janela test, selecione:

Tipo de solicitação: Grongier.PEX.Message

Em classname, insira:

```
msg.HFRequest
```

Para json, veja um exemplo de uma chamada para GPT2:

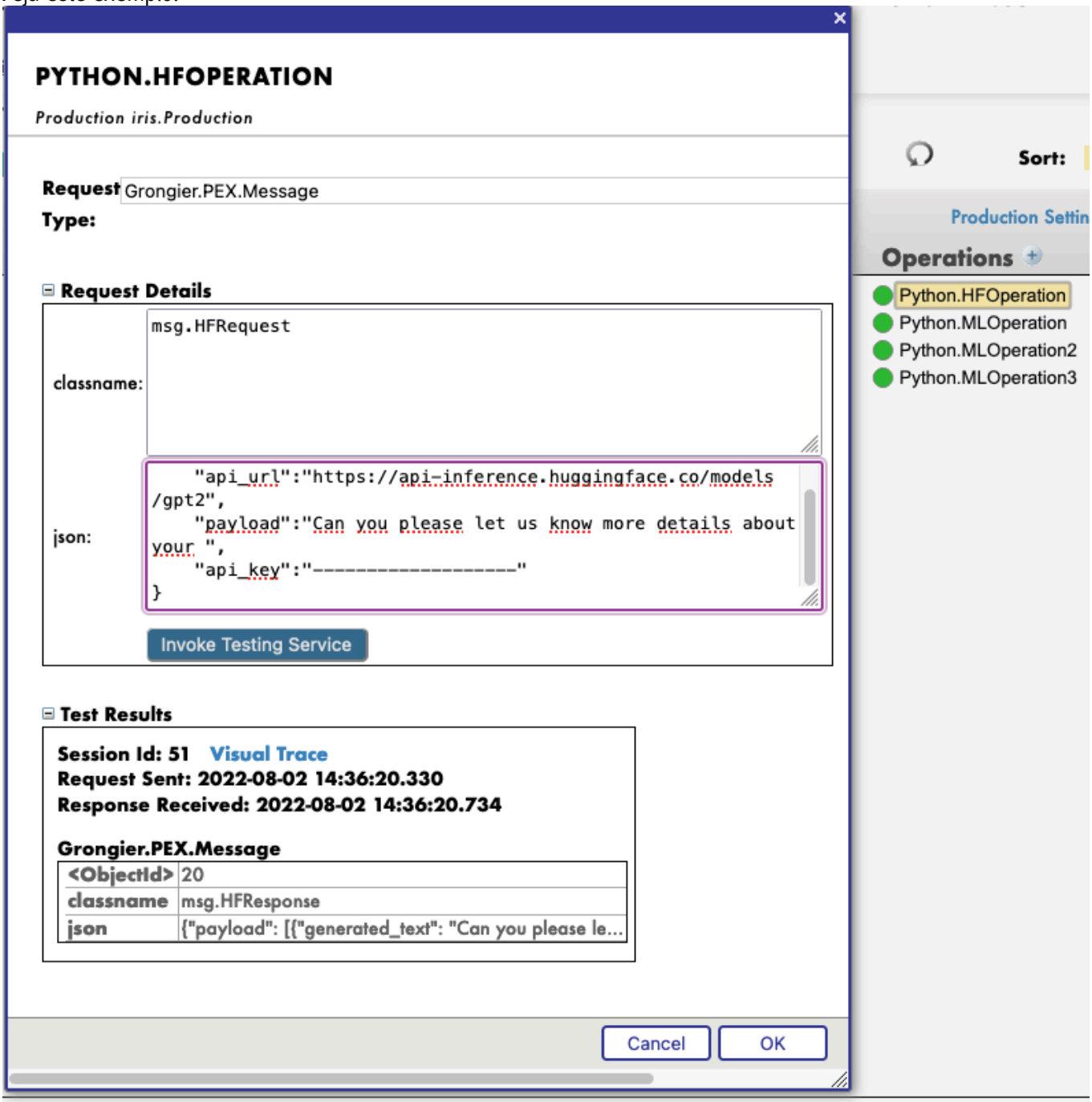
```
{
    "api_url":"https://api-inference.huggingface.co/models/gpt2",
    "payload":"Can you please let us know more details about your ",
    "api_key":"----------------------"
}
```

Agora, você pode clicar em Visual Trace para ver nos detalhes o que aconteceu e visualizar os registros.

OBSERVE que você precisa ter uma chave de API do HuggingFace antes de usar esta Operação (as chaves de API são gratuitas, basta fazer a inscrição no HF)

OBSERVE que você pode mudar o URL para testar outros modelos do HuggingFace. Talvez seja necessário mudar o payload.

Veja este exemplo:



5. Use qualquer modelo da Web

Nesta seção, vamos ensinar você a usar praticamente qualquer modelo pré-treinado da internet, HuggingFace ou não, para poupar recursos ou simplesmente usar esses modelos dentro do IRIS.

5.1. PRIMEIRO EXEMPLO: VOCÊ TEM SEU PRÓPRIO MODELO

Nesse caso, você precisa copiar e colar seu modelo, com config, tokenizer.json etc. dentro de uma pasta na pasta do modelo.
Caminho: src/model/yourmodelname/

Em seguida, você precisa criar uma nova operação, chame-a como quiser e acesse os parâmetros dessa operação.
Então, acesse settings na guia à direita, na parte Python e na parte %settings. Aqui, você pode inserir ou modificar quaisquer parâmetros (não se esqueça de pressionar apply depois de terminar).
Veja a configuração padrão para esse caso:
%settings

```
name=yourmodelname
task=text-generation
```

OBSERVE que qualquer configuração que não for name ou modelurl entrará nas configurações PIPELINE.

Agora você pode clicar duas vezes na operação e executar o start. Você precisa ver na parte Log a inicialização do modelo.

Em seguida, criamos um PIPELINE utilizando transformers que usam o arquivo config na pasta como vimos antes.

Para chamar esse pipeline, clique na operação e selecione na guia à direita action. Você pode aplicar test à demonstração.

Na janela test, selecione:

Tipo de solicitação: Grongier.PEX.Message

Em classname, insira:

```
msg.MLRequest
```

Para json, você precisa inserir todos os argumentos necessários para o modelo.
Veja um exemplo de uma chamada para GPT2:

```
{
    "text_inputs":"Unfortunately, the outcome",
    "max_length":100,
    "num_return_sequences":3
}
```

Clique em Invoke Testing Service e aguarde a operação do modelo.

Veja este exemplo:

Agora, você pode clicar em Visual Trace para ver nos detalhes o que aconteceu e visualizar os registros.

Veja este exemplo :

5.2. SEGUNDO EXEMPLO: VOCÊ QUER FAZER O DOWNLOAD DE UM MODELO DO HUGGINGFACE

Nesse caso, você precisa encontrar o URL do modelo no HuggingFace.
Encontre um modelo que faça o que você busca e use-o sem gastar recursos utilizando as tecnologias da InterSystems.

5.2.1. Configurações

Vá até os parâmetros do Hugging.
Clique na operação HuggingFace escolhida e acesse settings na guia à direita, na parte Python e na parte %settings. Aqui, você pode inserir ou modificar quaisquer parâmetros (não se esqueça de pressionar apply depois de terminar).
Veja um exemplo de configuração para alguns modelos que encontramos no HuggingFace:

%settings para gpt2

```
model_url=https://huggingface.co/gpt2
name=gpt2
task=text-generation
```

%settings para camembert-ner

```
name=camembert-ner
model_url=https://huggingface.co/Jean-Baptiste/camembert-ner
task=ner
aggregation_strategy=simple
```

%settings para bert-base-uncased

```
name=bert-base-uncased
model_url=https://huggingface.co/bert-base-uncased
task=fill-mask
```

%settings para detr-resnet-50

```
name=detr-resnet-50
model_url=https://huggingface.co/facebook/detr-resnet-50
task=object-detection
```

%settings para detr-resnet-50-protnic

```
name=detr-resnet-50-panoptic
model_url=https://huggingface.co/facebook/detr-resnet-50-panoptic
task=image-segmentation
```

OBSERVE que qualquer configuração que não for name ou modelurl entrará nas configurações PIPELINE. Então, no segundo exemplo, o pipeline camembert-ner requer a especificação de aggregationstrategy e task, enquanto gpt2 requer apenas uma task.

Veja este exemplo:

Agora você pode clicar duas vezes na operação e executar o start.
Você precisa ver na parte Log a inicialização e o download do modelo.
OBSERVAÇÃO: Você pode atualizar os registros a cada x segundos para ver o avanço dos downloads.

Em seguida, criamos um PIPELINE utilizando transformers que usam o arquivo config na pasta como vimos antes.

5.2.2. Teste

Para chamar esse pipeline, clique na operação e selecione na guia à direita action. Você pode aplicar test à demonstração.

Na janela test, selecione:

Tipo de solicitação: Grongier.PEX.Message

Em classname, insira:

```
msg.MLRequest
```

Para json, você precisa inserir todos os argumentos necessários para o modelo.
Veja um exemplo de uma chamada para GPT2:

```
{
    "text_inputs":"George Washington lived",
    "max_length":30,
    "num_return_sequences":3
}
```

Aqui está um exemplo de uma chamada para Camembert-ner:

```
{
    "inputs":"George Washington lived in washington"
}
```

Aqui está um exemplo de uma chamada para bert-base-uncased:

```
{
    "inputs":"George Washington lived in [MASK]."
}
```

Aqui está um exemplo de uma chamada para detr-resnet-50 usando um URL online:

```
{
    "url":"http://images.cocodataset.org/val2017/000000039769.jpg"
}
```

Aqui está um exemplo de uma chamada para detr-resnet-50-panoptic usando um URL como caminho:

```
{
    "url":"/irisdev/app/misc/000000039769.jpg"
}
```

Clique em Invoke Testing Service e aguarde a operação do modelo.
Agora, você pode clicar em Visual Trace para ver nos detalhes o que aconteceu e visualizar os registros.

OBSERVE que, após fazer pela primeira vez o download de um modelo, a produção não fará o download novamente, mas usará os arquivos em cache encontrados em src/model/TheModelName/.
Se alguns arquivos estiverem ausentes, a Produção fará o download novamente.

Veja este exemplo:

Veja este exemplo:

6. Ajuste dos modelos

Nesta parte, tentamos ajustar um modelo para reaproveitá-lo e torná-lo ainda melhor sem usar muitos recursos.

6.1. Personalização do modelo

6.1.1. Download do modelo

Para utilizar esse GitHub, você precisa ter um modelo do HuggingFace compatível com pipeline para usar e treinar, além de um conjunto de dados para treinar seu modelo.

Para ajudar, oferecemos a possibilidade de usar script Python em src/utils/downloadbert.py. Ele fará o download para você do modelo "https://huggingface.co/bert-base-cased" e colocará dentro da pasta src/model/bert-base-cased se já não estiver lá.
Além disso, também disponibilizamos um conjunto de dados para treinar o modelo bert. Esse conjunto de dados já estava carregado dentro da base de dados IRIS e nenhuma ação adicional é necessária se você quiser usá-lo. (Para acessá-lo, vá até a parte SQL do portal, o namespace da base de dados iris e depois a tabela de revisão)

Para usar o script, se você estiver no contêiner, pode executá-lo sem se preocupar. Se você estiver no local, talvez seja necessário aplicar pip3 install requests e pip3 install beautifulsoup4

Veja o resultado:

6.1.2. Configurações

Se você quiser usar o modelo bert-base-cased e já fez o download usando o script, não é necessário adicionar mais nada às configurações, e você pode avançar para o treinamento do modelo.

Se você quiser treinar seu próprio modelo, clique em Python.TuningOperation e selecione Settings na guia à direita, depois Python e, na parte %settings, insira o caminho do modelo, o nome da pasta e o número do rótulo que você quer para o treinamento.

Exemplo:

```
path=/irisdev/app/src/model/
model_name=bert-base-cased
num_labels=5
```

6.1.3. Treinamento do modelo

Para treinar o modelo, você precisa acessar Production neste link:

```
http://localhost:52795/csp/irisapp/EnsPortal.ProductionConfig.zen?PRODUCTION=iris.Production
```

E conectar usando:
SuperUser como nome de usuário e SYS como senha.

Para chamar o treinamento, clique em Python.TuningOperation e selecione na guia à direita Actions. Você pode aplicar Test à demonstração.

Na janela de teste, selecione:

Tipo de solicitação: Grongier.PEX.Message

Em classname, insira:

```
msg.TrainRequest
```

Para json, você precisa inserir todos os argumentos necessários para o treinamento. Aqui está um exemplo que treina com as primeiras 20 linhas (não é um treinamento adequado, mas é rápido):

```
{
    "columns":"ReviewLabel,ReviewText",
    "table":"iris.Review",
    "limit":20,
    "p_of_train":0.8,

    "output_dir":"/irisdev/app/src/model/checkpoints",
    "evaluation_strategy":"steps",
    "learning_rate":0.01,
    "num_train_epochs":1
}
```

Veja este exemplo:

Como pode ver, é preciso inserir
- a table usada.
- as columns usadas (primeiro é label e segundo é input para a tokenização)
- o limit de linhas compreendidas (se você não especificar um número de linhas, todos os dados serão usados)
- poftrain, a porcentagem de dados de treinamento usados do conjunto de dados, e 1 - poftrain, a porcentagem de dados de teste usados do conjunto de dados.

Depois disso, os outros parâmetros cabem a você e podem variar de acordo com os parâmetros https://huggingface.co/docs/transformers/mainclasses/trainer.

OBSERVE que o tamanho do lote para treinamento e teste será calculado automaticamente se não for inserido na solicitação. (É o maior divisor do número de linhas que é menor do que a raiz quadrada do número de linhas e do que 32)

Clique em "Invoke Testing Service" e feche a janela de teste sem esperar.
Agora acesse Python.TuningOperation e selecione na guia à direita log. Aqui você pode ver o avanço do treinamento e das avaliações.
Após a conclusão, você verá um log dizendo que o novo modelo foi salvo em uma pasta temporária.
Agora acesse Python.TuningOperation, selecione na guia à direita message e clique no cabeçalho do último item para selecioná-lo. Aqui você pode ver o avanço do treinamento e das avaliações e, ao final, é possível acessar as Métricas do modelo novo e antigo para comparação.

6.1.4. Substituição do modelo

Se você quiser manter o modelo antigo, nenhuma ação é necessária: o antigo permanecerá na pasta não temporária e ainda será carregado para treinamento adicional.

Se você quiser manter o modelo novo, clique em Python.TuningOperation, selecione na guia à direita Actions e teste. Na janela de teste, selecione:

Tipo de solicitação: Grongier.PEX.Message

Em classname, insira:

```
msg.OverrideRequest
```

Para json, chaves vazias:

```
{}
```

Clique em Invoke Testing Service e veja a mensagem de resposta. O novo modelo foi movido da pasta temporária para a não temporária.

6.2. Uso do modelo

Treinar um modelo é interessante, mas você também pode testá-lo.

6.2.1. Configurações

Se você quiser usar o modelo bert-base-cased e já fez o download usando o script, não é necessário adicionar mais nada às configurações, e você pode avançar para o teste do modelo.

Se você quiser treinar seu próprio modelo, clique em Python.TuningOperation e selecione Settings na guia à direita, depois Python e, na parte %settings, insira o parâmetro para adicionar ao pipeline.

6.2.2. Teste do modelo

Para testar o modelo, você precisa acessar Production neste link:

```
http://localhost:52795/csp/irisapp/EnsPortal.ProductionConfig.zen?PRODUCTION=iris.Production
```

E conectar usando:
SuperUser como nome de usuário e SYS como senha.

Para chamar o teste, clique em Python.MLOperation e selecione na guia à direita Actions. Você pode aplicar Test à demonstração.

Na janela de teste, selecione:

Tipo de solicitação: Grongier.PEX.Message

Em classname, insira:

```
msg.MLRequest
```

Para json, você precisa inserir todos os argumentos necessários para o modelo funcionar

```
{
    "inputs":"This was a really bad experience"
}
```

Pressione Call test services e veja o resultado.

7. Observação importante

O ajuste fino de modelos pode exigir MUITO tempo e recursos, mas sempre consumirá menos recursos do que o treinamento de um modelo do zero.
Você pode ver que já está levando muito tempo e poder computacional para o ajuste fino do modelo, então imagine o custo se tivesse que treiná-lo do zero e começar de novo várias vezes para obter resultados ideais.

8. Solução de problemas

Se você tiver problemas, a leitura é o primeiro conselho que podemos dar a você. A maioria dos erros são facilmente compreendidos apenas ao ler os logs, pois quase todos os erros são capturados por um try/catch e registrados.

Se você precisar instalar um novo módulo ou dependência do Python, abra um terminal dentro do contêiner e digite, por exemplo: "pip install new-module"
Há várias formas de abrir um terminal,
- Se você usa os plugins da InterSystems, pode clicar na barra abaixo no VSCode, que se parece com docker:iris:52795[IRISAPP], e selecionar Open Shell in Docker.
- Em qualquer terminal local, digite: docker-compose exec -it iris bash
- No Docker-Desktop, encontre o contêiner IRIS e clique em Open in terminal

Alguns modelos talvez exijam algumas alterações no pipeline ou nas configurações, por exemplo. É sua responsabilidade adicionar as informações corretas nas configurações e na solicitação.

9. Conclusão

Daqui em diante, você poderá usar modelos de que talvez precise no IRIS e ajustá-los como desejar.
Esse modelo precisa ser modificado para atender às suas necessidades e foi criado como base para qualquer projeto de IA e IRIS que considere a sustentabilidade e interoperabilidade.

Por falta de tempo, não consegui adicionar uma API que usasse um Diretor para se comunicar diretamente com a produção e permitir que os usuários fizessem solicitações aos modelos.
No entanto, se você ainda tiver interesse em uma API IRIS usando esse módulo do Python, confira meu GitHub ou acesse meu exemplo de API no Python para IRIS.

Link para meu perfil do DC: https://community.intersystems.com/user/lucas-enard-0

#API #Bancos de dados #Docker #Interoperabilidade #Python #InterSystems IRIS
Confira o aplicativo relacionado no InterSystems Open Exchange