

Université Cadi Ayyad

Faculté des Sciences

Département Informatique

---

Master Intelligence Artificielle

Module : MLOps

---

# End-to-End AutoML Platform Using Kubeflow + Katib

*Optimisation d'Hyperparamètres et Pipelines ML*

*Random Search, Bayesian Optimization & TPE*

**Réalisé par :**

Assoumana Souley Hadiza Dite Maa

Bouamir Assia

**Encadré par :**

Pr. Fahd Kalloubi

---

Année Universitaire 2024-2025

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Objectifs du Projet . . . . .	3
1.2	Architecture Globale . . . . .	3
<b>2</b>	<b>Installation de l'Infrastructure</b>	<b>4</b>
2.1	Étape 1 : Installation de Minikube . . . . .	4
2.1.1	Prérequis Système . . . . .	4
2.1.2	Installation sous Windows . . . . .	4
2.1.3	Vérification . . . . .	4
2.2	Étape 2 : Installation de Docker Desktop . . . . .	5
2.3	Étape 3 : Démarrage du Cluster Minikube . . . . .	6
2.4	Étape 4 : Installation de Kubeflow Pipelines . . . . .	6
2.5	Étape 5 : Installation de Katib . . . . .	6
2.6	Étape 6 : Vérification des Pods . . . . .	7
2.7	Étape 7 : Correction du Problème MinIO . . . . .	8
2.8	Étape 8 : Accès aux Interfaces Web . . . . .	8
2.9	Kubeflow Pipelines UI . . . . .	8
<b>3</b>	<b>Préparation du Code d'Entraînement</b>	<b>11</b>
3.1	Structure du Projet . . . . .	11
3.2	Script d'Entraînement (train.py) . . . . .	11
3.3	Dockerfile . . . . .	15
3.4	Requirements . . . . .	15
3.5	Construction de l'Image Docker . . . . .	15
<b>4</b>	<b>Pipeline 1 : Random Search</b>	<b>16</b>
4.1	Principe du Random Search . . . . .	16
4.2	Code Complet du Pipeline . . . . .	16
4.3	Lancement du Pipeline . . . . .	23
<b>5</b>	<b>Pipeline 2 : Optimisation Bayésienne</b>	<b>24</b>
5.1	Principe de l'Optimisation Bayésienne . . . . .	24
5.2	Code Complet du Pipeline . . . . .	24
<b>6</b>	<b>Pipeline 3 : Tree-structured Parzen Estimator (TPE)</b>	<b>28</b>
6.1	Principe de TPE . . . . .	28
6.2	Code Complet du Pipeline . . . . .	28
<b>7</b>	<b>Pipeline Complet : Comparaison des Algorithmes</b>	<b>32</b>
7.1	Objectif . . . . .	32
7.2	Code Complet . . . . .	32
<b>8</b>	<b>Exécution des Pipelines</b>	<b>38</b>
8.1	Compilation des Pipelines . . . . .	38
8.2	Soumission via l'Interface Web . . . . .	38
8.3	Soumission via CLI . . . . .	38
8.4	Monitoring de l'Exécution . . . . .	39

<b>9</b>	<b>Déploiement avec Flask</b>	<b>40</b>
9.1	Architecture de l'API . . . . .	40
9.2	Code de l'Application Fast . . . . .	40
9.3	Interface Web Moderne . . . . .	41
9.4	Lancement de l'Application . . . . .	44
<b>10</b>	<b>Visualisation et Analyse des Résultats</b>	<b>46</b>
10.1	Interface Katib . . . . .	46
10.2	Interface Kubeflow Pipelines . . . . .	46
10.3	Récupération du Modèle Sauvegardé . . . . .	47
<b>11</b>	<b>Résultats Attendus et Comparaison</b>	<b>49</b>
11.1	Comparaison des Algorithmes . . . . .	49
11.2	Avantages et Inconvénients . . . . .	49
11.3	Graphiques de Convergence . . . . .	49
<b>12</b>	<b>Dépannage et Problèmes Courants</b>	<b>50</b>
12.1	Problème 1 : Pods en ImagePullBackOff . . . . .	50
12.2	Problème 2 : Katib ne Collecte pas les Métriques . . . . .	50
12.3	Problème 3 : Timeout des Expériences . . . . .	50
12.4	Problème 4 : MinIO ou ML-Pipeline en CrashLoop . . . . .	51
12.5	Problème 5 : Pipeline ne Démarre pas . . . . .	51
<b>13</b>	<b>Extensions et Améliorations</b>	<b>53</b>
13.1	Ajouter d'Autres Algorithmes . . . . .	53
13.2	Early Stopping . . . . .	53
13.3	Monitoring Avancé avec Prometheus . . . . .	54
13.4	Utiliser des Datasets Personnalisés . . . . .	54
13.5	Multi-Objective Optimization . . . . .	54
<b>14</b>	<b>Bonnes Pratiques</b>	<b>56</b>
14.1	Gestion des Ressources . . . . .	56
14.2	Versioning des Modèles . . . . .	56
14.3	Logging Structuré . . . . .	56
14.4	Tests et Validation . . . . .	57
<b>15</b>	<b>Nettoyage et Maintenance</b>	<b>58</b>
15.1	Nettoyer les Expériences . . . . .	58
15.2	Nettoyer les Pipelines . . . . .	58
15.3	Arrêter et Redémarrer le Cluster . . . . .	58
15.4	Backup des Modèles . . . . .	59
<b>16</b>	<b>Conclusion</b>	<b>60</b>
16.1	Récapitulatif . . . . .	60
16.2	Avantages de cette Architecture . . . . .	60
16.3	Prochaines Étapes . . . . .	60
16.4	Ressources Supplémentaires . . . . .	60
16.5	Support et Communauté . . . . .	61
	<b>Annexes</b>	<b>62</b>

# 1 Introduction

Ce document présente un guide complet pour mettre en place une plateforme AutoML utilisant **Katib** pour l'optimisation d'hyperparamètres et **Kubeflow Pipelines** pour orchestrer le workflow d'entraînement.

## 1.1 Objectifs du Projet

- Déployer une infrastructure Kubernetes locale avec Minikube
- Installer et configurer Kubeflow Pipelines et Katib
- Implémenter des pipelines d'optimisation (Random, Bayésien et TPE)
- Comparer les algorithmes et sélectionner automatiquement le meilleur modèle
- Sauvegarder les artefacts dans Kubernetes

## 1.2 Architecture Globale

Le système AutoML se compose de plusieurs couches :

- **Infrastructure** : Minikube (Kubernetes local)
- **Orchestration** : Kubeflow Pipelines v2.14.3
- **Optimisation** : Katib v0.17.0
- **Stockage** : MinIO pour les artefacts
- **Conteneurisation** : Docker

### Information

#### Technologies utilisées :

- Kubernetes (Minikube)
- Kubeflow Pipelines v2.14.3
- Katib v0.17.0 (v1beta1)
- Docker Desktop
- Python 3.9
- scikit-learn, pandas, numpy

## 2 Installation de l'Infrastructure

### 2.1 Étape 1 : Installation de Minikube

#### 2.1.1 Prérequis Système

Vérifiez que votre système dispose de :

- Minimum 6 GB de RAM (recommandé : 8 GB)
- 4 CPUs
- 20 GB d'espace disque disponible
- Windows 10/11 avec PowerShell

#### 2.1.2 Installation sous Windows

Listing 1 – Installation Minikube - PowerShell (Administrateur)

```
1 # Créer le repertoire d'installation
2 New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force
3
4 # Telecharger Minikube
5 $ProgressPreference = 'SilentlyContinue'
6 Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' '
7   -Uri 'https://github.com/kubernetes/minikube/releases/latest/
8     download/minikube-windows-amd64.exe' '
   -UseBasicParsing
```

Listing 2 – Ajouter Minikube au PATH

```
1 # Ajouter au PATH systeme (PowerShell Administrateur)
2 $oldPath = [Environment]::GetEnvironmentVariable('Path', [
3   EnvironmentVariableTarget]::Machine)
4 if ($oldPath.Split(';') -notcontains 'C:\minikube'){
5   [Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube'
6     -f $oldPath), [EnvironmentVariableTarget]::Machine)
7 }
```

#### Attention

**Important :** Fermez et rouvrez votre terminal PowerShell après l'ajout au PATH !

#### 2.1.3 Vérification

Listing 3 – Vérifier l'installation

```
1 # Verifier la version
2 minikube version
```

## 2.2 Étape 2 : Installation de Docker Desktop

1. Télécharger Docker Desktop : <https://www.docker.com/products/docker-desktop>
2. Exécuter l'installateur
3. Activer WSL 2 (Windows Subsystem for Linux)
4. Redémarrer l'ordinateur
5. Démarrer Docker Desktop

Listing 4 – Vérification Docker

```
1 # Verifier Docker
2 docker --version
3
4 # Tester Docker
5 docker run hello-world
```

### Attention

Docker Desktop doit être en cours d'exécution avant de démarrer Minikube !

## 2.3 Étape 3 : Démarrage du Cluster Minikube

Listing 5 – Démarrer Minikube avec Docker

```
1 # Demarrer Minikube avec configuration optimale
2 minikube start --driver=docker --memory=6000 --cpus=4
3
4 # Verifier le statut
5 minikube status
```

### Succès

Si vous voyez minikube: Running, votre cluster est prêt !

## 2.4 Étape 4 : Installation de Kubeflow Pipelines

Listing 6 – Installation Kubeflow Pipelines v2.14.3

```
1 # Définir la version
2 $env:PIPELINE_VERSION = "2.14.3"
3
4 # Appliquer les ressources cluster-scoped
5 kubectl apply -k "github.com/kubeflow/pipelines/manifests/kustomize
   /cluster-scoped-resources?ref=$env:PIPELINE_VERSION"
6
7 # Attendre que les CRDs soient etablies
8 kubectl wait --for condition=established --timeout=60s crd/
   applications.app.k8s.io
9
10 # Installer les composants Kubeflow
11 kubectl apply -k "github.com/kubeflow/pipelines/manifests/kustomize
   /env/platform-agnostic?ref=$env:PIPELINE_VERSION"
```

### Information

Le déploiement complet prend environ 3 minutes. Soyez patient !

## 2.5 Étape 5 : Installation de Katib

Listing 7 – Installation Katib v0.17.0

```
1 # Installer Katib standalone
2 kubectl apply -k "github.com/kubeflow/katib.git/manifests/v1beta1/
   installs/katib-standalone?ref=v0.17.0"
3
4 # Installer le SDK Python Katib
5 pip install -U kubeflow-katib
```

## 2.6 Étape 6 : Vérification des Pods

Listing 8 – Vérifier tous les pods

```
1 # Verifier les pods Kubeflow et Katib
2 kubectl get pods -n kubeflow
3
4 # Attendre que tous soient en Running (sauf minio initialement)
5 kubectl get pods -n kubeflow -w
```



## 2.7 Étape 7 : Correction du Problème MinIO

MinIO pose un problème connu avec l'image latest. Voici la solution :

Listing 9 – Corriger MinIO

```
1 # Patcher l'image MinIO vers une version stable
2 kubectl patch deployment minio -n kubeflow --type='json' -p='[
3   {
4     "op": "replace",
5     "path": "/spec/template/spec/containers/0/image",
6     "value": "minio/minio:RELEASE.2021-06-17T00-10-46Z"
7   }
8 ]',
9
10 # Supprimer les pods MinIO pour forcer la recreation
11 kubectl delete pod -n kubeflow -l app=minio
12
13 # Supprimer les pods ml-pipeline pour qu'ils se reconnectent
14 kubectl delete pod -n kubeflow -l app=ml-pipeline
```

### Succès

Attendez 1-2 minutes et vérifiez que tous les pods sont en Running :

```
1 kubectl get pods -n kubeflow
```

## 2.8 Étape 8 : Accès aux Interfaces Web

Listing 10 – Port-forward pour Kubeflow UI

```
1 # Exposer Kubeflow Pipelines UI (dans un terminal separe)
2 kubectl port-forward -n kubeflow svc/ml-pipeline-ui 8080:80
```

Ouvrir dans le navigateur : <http://localhost:8080>

## 2.9 Kubeflow Pipelines UI

Listing 11 – Port-forward pour Kubeflow UI

```
1 # Exposer l'interface Kubeflow Pipelines
2 kubectl port-forward -n kubeflow svc/ml-pipeline-ui 8080:80
3
4 # Ouvrir dans le navigateur : http://localhost:8080
```

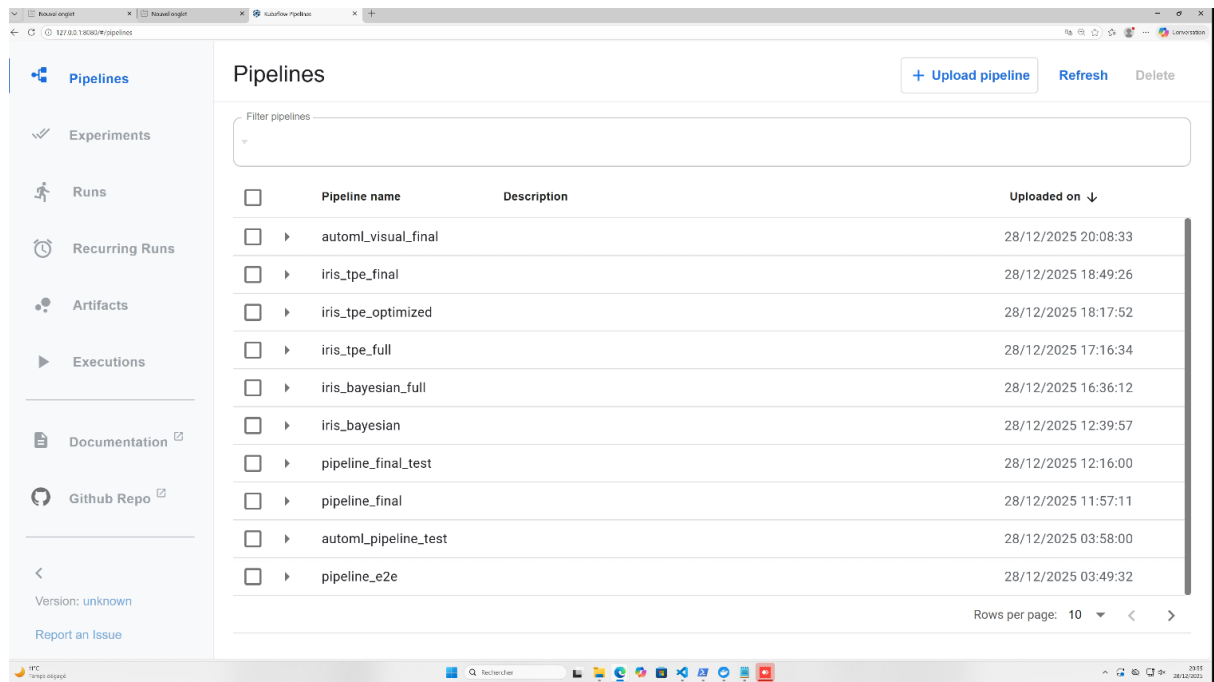


FIGURE 1 – Interface principale de Kubeflow Pipelines - Vue des pipelines

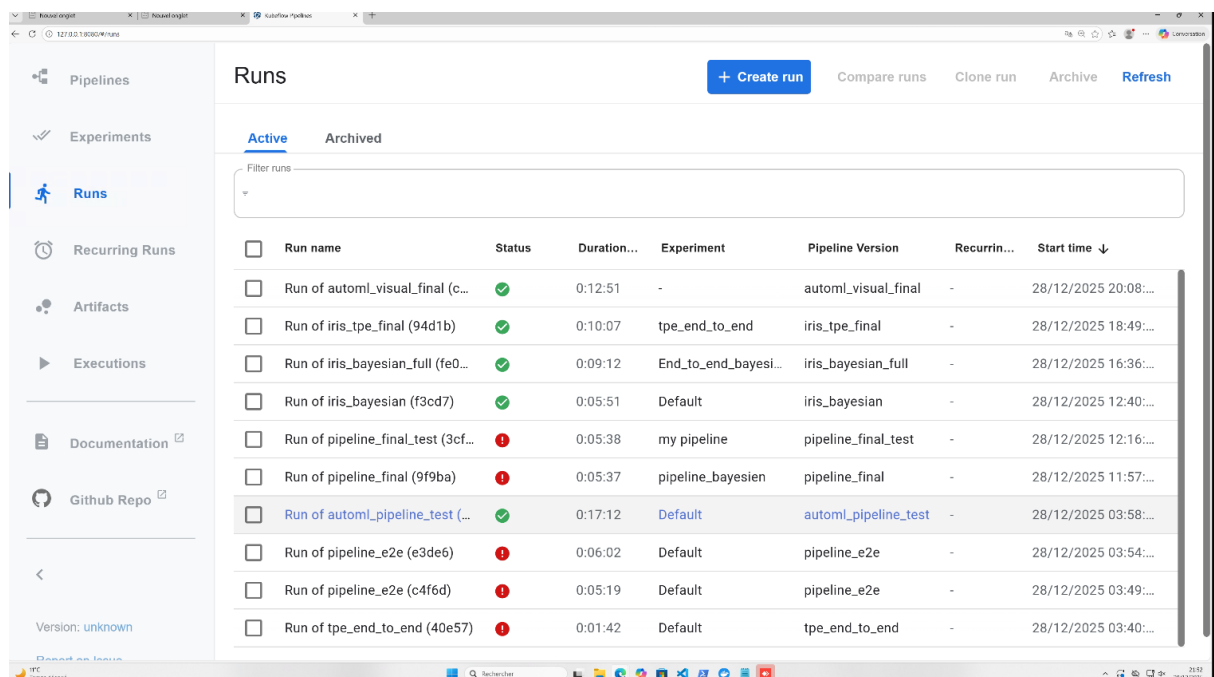


FIGURE 2 – Vue des exécutions (Runs) dans Kubeflow Pipelines

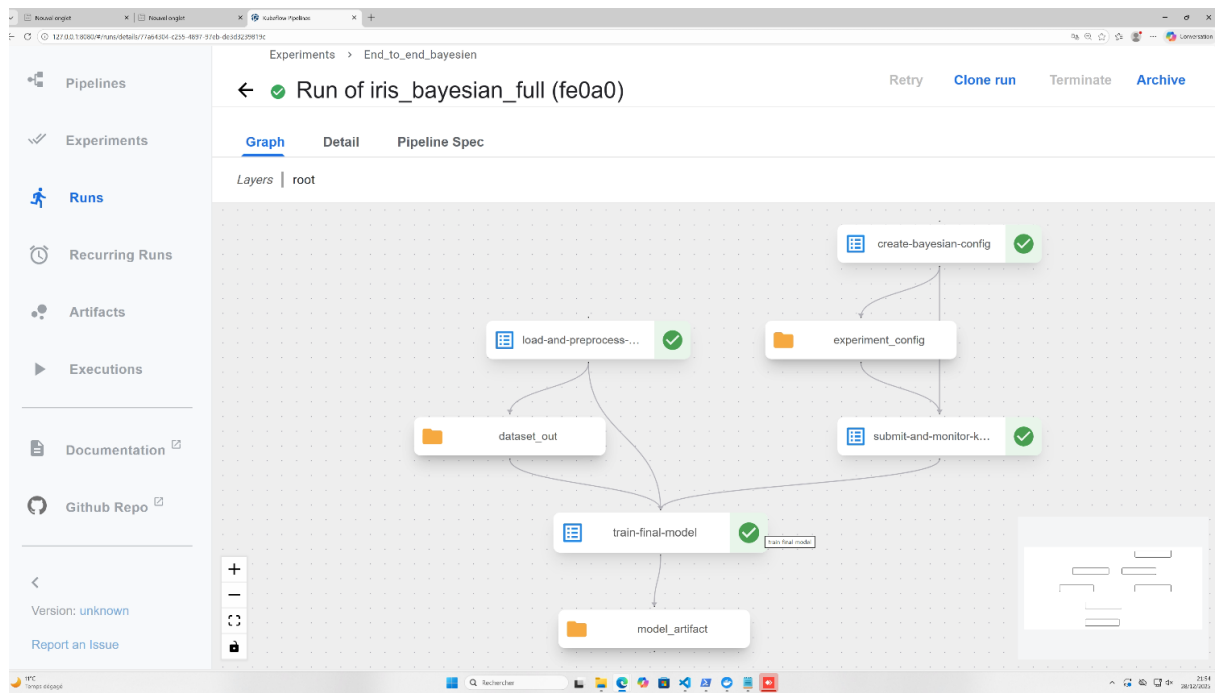


FIGURE 3 – Graphe d'exécution d'un pipeline dans Kubeflow

## Listing 12 – Port-forward pour Katib UI

```
1 # Exposer Katib UI (dans un autre terminal)
2 kubectl port-forward -n kubeflow svc/katib-ui 8081:80
```

Ouvrir dans le navigateur : <http://localhost:8081/katib/>

**Information**

Laissez ces terminaux ouverts pendant toute la durée du travail!

## 3 Préparation du Code d'Entraînement

### 3.1 Structure du Projet

```
project/  
  src/  
    training/  
      train.py          # Script d'entraînement  
  pipelines/  
    random_pipeline.py  # Pipeline Random Search  
    bayesian_pipeline.py # Pipeline Bayesian  
    tpe_pipeline.py     # Pipeline TPE  
    complete_pipeline.py # Pipeline complet  
  Dockerfile  
  requirements.txt
```

### 3.2 Script d'Entraînement (train.py)

Listing 13 – src/training/train.py - Script complet

```
1 import argparse  
2 import numpy as np  
3 from sklearn.datasets import load_iris  
4 from sklearn.model_selection import train_test_split  
5 from sklearn.neural_network import MLPClassifier  
6 from sklearn.metrics import accuracy_score  
7 import json  
8 import os  
9  
10 def train_model(learning_rate, hidden_units, optimizer):  
11     """  
12     Entraîne un modele MLP sur le dataset Iris  
13  
14     Args:  
15         learning_rate: Taux d'apprentissage  
16         hidden_units: Nombre d'unités dans la couche cachée  
17         optimizer: Algorithme d'optimisation (adam ou sgd)  
18  
19     Returns:  
20         accuracy: Precision du modele  
21     """  
22  
23     # Charger les données Iris  
24     print("Chargement du dataset Iris...")  
25     X, y = load_iris(return_X_y=True)  
26  
27     # Split train/test  
28     X_train, X_test, y_train, y_test = train_test_split(  
29         X, y, test_size=0.2, random_state=42, stratify=y  
30     )  
31
```

```
32 # Normalisation des donnees
33 mean = X_train.mean(axis=0)
34 std = X_train.std(axis=0) + 1e-8
35 X_train = (X_train - mean) / std
36 X_test = (X_test - mean) / std
37
38 print(f"Parametres d'entrainement:")
39 print(f"  - Learning Rate: {learning_rate}")
40 print(f"  - Hidden Units: {hidden_units}")
41 print(f"  - Optimizer: {optimizer}")
42
43 # Creation du modele
44 model = MLPClassifier(
45     hidden_layer_sizes=(hidden_units,),
46     learning_rate_init=learning_rate,
47     solver=optimizer,
48     max_iter=1000,
49     random_state=42,
50     early_stopping=True,
51     validation_fraction=0.1,
52     n_iter_no_change=10
53 )
54
55 # Entrainement
56 print("Debut de l'entrainement...")
57 model.fit(X_train, y_train)
58
59 # Evaluation
60 y_pred_train = model.predict(X_train)
61 y_pred_test = model.predict(X_test)
62
63 train_accuracy = accuracy_score(y_train, y_pred_train)
64 test_accuracy = accuracy_score(y_test, y_pred_test)
65 loss = 1 - test_accuracy
66
67 print(f"Resultats:")
68 print(f"  - Train Accuracy: {train_accuracy:.4f}")
69 print(f"  - Test Accuracy: {test_accuracy:.4f}")
70 print(f"  - Loss: {loss:.4f}")
71
72 # CRITIQUE : Ecriture des metriques pour Katib
73 # Le format DOIT etre: metric_name=value
74 metrics_path = "/tmp/katib-metrics.log"
75 with open(metrics_path, "w") as f:
76     f.write(f"accuracy={test_accuracy}\n")
77     f.write(f"loss={loss}\n")
78
79 print(f"Metriques ecrites dans {metrics_path}")
80
81 # Sauvegarder les parametres et resultats
82 results = {
```

```
83     "learning_rate": learning_rate,
84     "hidden_units": hidden_units,
85     "optimizer": optimizer,
86     "train_accuracy": float(train_accuracy),
87     "test_accuracy": float(test_accuracy),
88     "loss": float(loss),
89     "n_iterations": model.n_iter_
90 }
91
92 results_path = "/tmp/training_results.json"
93 with open(results_path, "w") as f:
94     json.dump(results, f, indent=2)
95
96 print(f"Resultats sauvegardes dans {results_path}")
97 print("Entrainement termine avec succes!")
98
99 return test_accuracy
100
101 def main():
102     parser = argparse.ArgumentParser(
103         description="Entrainement MLP sur Iris avec hyperparametres
104             variables"
105     )
106     parser.add_argument(
107         "--learning-rate",
108         type=float,
109         default=0.01,
110         help="Taux d'apprentissage"
111     )
112     parser.add_argument(
113         "--hidden-units",
114         type=int,
115         default=64,
116         help="Nombre d'unites dans la couche cachee"
117     )
118     parser.add_argument(
119         "--optimizer",
120         type=str,
121         default="adam",
122         choices=["adam", "sgd"],
123         help="Algorithme d'optimisation"
124     )
125
126     args = parser.parse_args()
127
128     # Lancer l'entrainement
129     accuracy = train_model(
130         learning_rate=args.learning_rate,
131         hidden_units=args.hidden_units,
132         optimizer=args.optimizer
```

```
133
134     print(f"\n=== ENTRAINEMENT TERMINE ===")
135     print(f"Accuracy finale: {accuracy:.4f}")
136
137 if __name__ == "__main__":
138     main()
```

### Attention

**Point critique :** Le fichier `/tmp/katib-metrics.log` doit respecter strictement le format `metric_name=value` (une métrique par ligne) pour que Katib puisse parser les résultats !

### 3.3 Dockerfile

Listing 14 – Dockerfile pour l'image d'entraînement

```
1 FROM python:3.9-slim
2
3 WORKDIR /app
4
5 # Installation des dependances systeme
6 RUN apt-get update && apt-get install -y --no-install-recommends \
7     build-essential \
8     && rm -rf /var/lib/apt/lists/*
9
10 # Installation des bibliotheques Python
11 COPY requirements.txt .
12 RUN pip install --no-cache-dir -r requirements.txt
13
14 # Copier le code d'entraînement
15 COPY src/ /app/src/
16
17 # Point d'entree
18 CMD ["python", "-u", "/app/src/training/train.py"]
```

### 3.4 Requirements

Listing 15 – requirements.txt

```
1 scikit-learn==1.3.2
2 numpy==1.24.3
3 pandas==2.0.3
```

### 3.5 Construction de l'Image Docker

Listing 16 – Build et chargement dans Minikube

```
1 # Builder l'image
2 docker build -t automl-training:v1.0 .
3
4 # Charger l'image dans Minikube
5 minikube image load automl-training:v1.0
6
7 # Verifier que l'image est presente
8 minikube image ls | findstr automl
```

#### Succès

L'image est maintenant disponible dans le cluster Minikube et peut être utilisée par les pods Katib!



## 4 Pipeline 1 : Random Search

### 4.1 Principe du Random Search

Le Random Search explore l'espace des hyperparamètres de manière aléatoire. C'est simple mais souvent efficace comme baseline.

### 4.2 Code Complet du Pipeline

Listing 17 – pipelines/random\_pipeline.py

```
1 import kfp
2 from kfp import dsl, compiler
3 from kfp.dsl import component, InputPath, OutputPath
4 from typing import NamedTuple
5 import json
6
7 @component(
8     base_image="python:3.9",
9     packages_to_install=["pandas==2.0.3", "scikit-learn==1.3.2"]
10 )
11 def load_and_preprocess_data(dataset_out: OutputPath("csv")):
12     """Charge et préprocesse le dataset Iris"""
13     import pandas as pd
14     from sklearn.datasets import load_iris
15
16     print("Chargement du dataset Iris...")
17     iris = load_iris()
18     df = pd.DataFrame(iris.data, columns=iris.feature_names)
19     df['target'] = iris.target
20
21     df.to_csv(dataset_out, index=False)
22     print(f"Dataset sauvegarde: {len(df)} lignes, {len(df.columns)} colonnes")
23
24 @component(
25     base_image="python:3.9",
26     packages_to_install=["pyyaml==6.0.1"]
27 )
28 def create_random_config(
29     experiment_name: str,
30     namespace: str,
31     training_image: str,
32     experiment_config: OutputPath("yaml"),
33 ) -> str:
34     """Cree la configuration Katib pour Random Search"""
35     import yaml
36
37     config = {
38         "apiVersion": "kubeflow.org/v1beta1",
39         "kind": "Experiment",
40         "metadata": {
```

```
41     "name": experiment_name,
42     "namespace": namespace
43 },
44 "spec": {
45     "algorithm": {
46         "algorithmName": "random"
47     },
48     "objective": {
49         "type": "maximize",
50         "objectiveMetricName": "accuracy",
51         "additionalMetricNames": ["loss"]
52     },
53     "parameters": [
54         {
55             "name": "learning_rate",
56             "parameterType": "double",
57             "feasibleSpace": {
58                 "min": "0.001",
59                 "max": "0.05"
60             }
61         },
62         {
63             "name": "hidden_units",
64             "parameterType": "int",
65             "feasibleSpace": {
66                 "min": "32",
67                 "max": "128"
68             }
69         },
70         {
71             "name": "optimizer",
72             "parameterType": "categorical",
73             "feasibleSpace": {
74                 "list": ["adam", "sgd"]
75             }
76         }
77     ],
78     "parallelTrialCount": 2,
79     "maxTrialCount": 10,
80     "metricsCollectorSpec": {
81         "collector": {"kind": "File"},
82         "source": {
83             "filePath": {
84                 "path": "/tmp/katib-metrics.log",
85                 "kind": "File"
86             }
87         }
88     },
89     "trialTemplate": {
90         "primaryContainerName": "training-container",
91         "trialParameters": [
```

```

92         {"name": "lr", "reference": "learning_rate"},
93         {"name": "hu", "reference": "hidden_units"},
94         {"name": "opt", "reference": "optimizer"}
95     ],
96     "trialSpec": {
97         "apiVersion": "batch/v1",
98         "kind": "Job",
99         "spec": {
100             "template": {
101                 "metadata": {
102                     "annotations": {
103                         "sidecar.istio.io/inject": "
104                             false"
105                     }
106                 },
107                 "spec": {
108                     "containers": [{
109                         "name": "training-container",
110                         "image": training_image,
111                         "imagePullPolicy": "
112                             IfNotPresent",
113                         "command": [
114                             "python",
115                             "-u",
116                             "/app/src/training/train.py
117                             "
118                         ],
119                         "args": [
120                             "--learning-rate=${
121                                 trialParameters.lr}",
122                             "--hidden-units=${
123                                 trialParameters.hu}",
124                             "--optimizer=${
125                                 trialParameters.opt}"
126                         ],
127                         "resources": {
128                             "limits": {
129                                 "cpu": "500m",
130                                 "memory": "512Mi"
131                             },
132                             "requests": {
133                                 "cpu": "100m",
134                                 "memory": "256Mi"
135                             }
136                         }
137                     }
138                 }
139             }
140         },
141         "restartPolicy": "Never"
142     }
143 }

```

```

137         }
138     }
139 }
140
141 with open(experiment_config, "w") as f:
142     yaml.dump(config, f)
143
144 print(f"Configuration Random Search creee pour {experiment_name}
145     ")
146 return experiment_name
147
148 @component(
149     base_image="python:3.9",
150     packages_to_install=["kubernetes==28.1.0", "pyyaml==6.0.1"]
151 )
152 def submit_and_monitor_katib(
153     experiment_name: str,
154     namespace: str,
155     experiment_config: InputPath("yaml"),
156 ) -> NamedTuple("Outputs", [
157     ("best_params", str),
158     ("best_accuracy", float)
159 ]):
160     """Soumet l'experience Katib et monitore son execution"""
161     import yaml
162     import time
163     import json
164     from kubernetes import client, config
165
166     # Charger la config Kubernetes
167     try:
168         config.load_incluster_config()
169     except:
170         config.load_kube_config()
171
172     custom_api = client.CustomObjectsApi()
173     group = "kubeflow.org"
174     version = "v1beta1"
175     plural = "experiments"
176
177     # Charger la config de l'experience
178     with open(experiment_config, "r") as f:
179         exp_dict = yaml.safe_load(f)
180
181     # Supprimer l'experience si elle existe deja
182     try:
183         custom_api.delete_namespaced_custom_object(
184             group, version, namespace, plural, experiment_name
185         )
186         print(f"Experience existante {experiment_name} supprimee")
187         time.sleep(5)

```

```
187     except:
188         pass
189
190     # Creer l'experience
191     custom_api.create_namespaced_custom_object(
192         group, version, namespace, plural, body=exp_dict
193     )
194     print(f"Experience {experiment_name} lancee avec succes")
195
196     # Monitoring en boucle
197     max_wait_time = 3600 # 1 heure max
198     start_time = time.time()
199
200     while True:
201         if time.time() - start_time > max_wait_time:
202             raise Exception("Timeout: l'experience a depasse 1
203                             heure")
204
205         exp = custom_api.get_namespaced_custom_object(
206             group, version, namespace, plural, experiment_name
207         )
208
209         status = exp.get("status", {})
210         conditions = status.get("conditions", [])
211
212         # Afficher l'etat actuel
213         trials_running = status.get("trialsRunning", 0)
214         trials_succeeded = status.get("trialsSucceeded", 0)
215         trials_failed = status.get("trialsFailed", 0)
216
217         print(f"Status - Running: {trials_running}, "
218               f"Succeeded: {trials_succeeded}, Failed: {
219                   trials_failed}")
220
221         # Verifier si l'experience est terminee
222         for condition in conditions:
223             if (condition.get("type") == "Succeeded" and
224                 condition.get("status") == "True"):
225
226                 print("Experience terminee avec succes!")
227
228                 optimal = status.get("currentOptimalTrial", {})
229                 assignments = optimal.get("parameterAssignments",
230                                           [])
231
232                 if assignments:
233                     params = {p["name"]: p["value"] for p in
234                               assignments}
235                     metrics = optimal.get("observation", {}).get("
236                                   metrics", [])
237                     acc = next(
```

```

233         (float(m["latest"]) for m in metrics
234         if m["name"] == "accuracy"),
235         0.0
236     )
237
238     print(f"Meilleurs parametres: {params}")
239     print(f"Meilleure accuracy: {acc:.4f}")
240
241     return (json.dumps(params), acc)
242
243     if (condition.get("type") == "Failed" and
244         condition.get("status") == "True"):
245         raise Exception("L'experience Katib a echoue.")
246
247     time.sleep(20)
248
249 @component(
250     base_image="python:3.9",
251     packages_to_install=[
252         "pandas==2.0.3",
253         "scikit-learn==1.3.2",
254         "joblib==1.3.2"
255     ]
256 )
257 def train_final_model(
258     best_params: str,
259     dataset_in: InputPath("csv"),
260     model_artifact: OutputPath("Model")
261 ):
262     """Entraîne le modèle final avec les meilleurs hyperparametres"""
263
264     import pandas as pd
265     import json
266     import joblib
267     from sklearn.neural_network import MLPClassifier
268     from sklearn.model_selection import train_test_split
269     from sklearn.metrics import accuracy_score,
270         classification_report
271
272     # Parser les parametres
273     params = json.loads(best_params)
274     lr = float(params.get('learning_rate', 0.01))
275     hu = int(float(params.get('hidden_units', 64)))
276     opt = params.get('optimizer', 'adam')
277
278     print(f"Entraînement final avec:")
279     print(f"  Learning Rate: {lr}")
280     print(f"  Hidden Units: {hu}")
281     print(f"  Optimizer: {opt}")
282
283     # Charger les donnees

```

```
282 df = pd.read_csv(dataset_in)
283 X = df.drop('target', axis=1)
284 y = df['target']
285
286 # Split pour evaluation finale
287 X_train, X_test, y_train, y_test = train_test_split(
288     X, y, test_size=0.2, random_state=42, stratify=y
289 )
290
291 # Normalisation
292 mean = X_train.mean(axis=0)
293 std = X_train.std(axis=0) + 1e-8
294 X_train = (X_train - mean) / std
295 X_test = (X_test - mean) / std
296
297 # Entrainement
298 model = MLPClassifier(
299     learning_rate_init=lr,
300     hidden_layer_sizes=(hu,),
301     solver=opt,
302     max_iter=1000,
303     random_state=42
304 )
305 model.fit(X_train, y_train)
306
307 # Evaluation
308 y_pred = model.predict(X_test)
309 accuracy = accuracy_score(y_test, y_pred)
310
311 print(f"\nResultats finaux:")
312 print(f"Accuracy: {accuracy:.4f}")
313 print("\nClassification Report:")
314 print(classification_report(y_test, y_pred))
315
316 # Sauvegarder le modele
317 joblib.dump(model, model_artifact)
318 print(f"Modele final sauvegarde avec succes!")
319
320 @dsl.pipeline(
321     name="Iris-Random-Search-AutoML",
322     description="Pipeline AutoML avec Random Search"
323 )
324 def iris_random_pipeline(
325     training_image: str,
326     namespace: str = "kubeflow",
327     exp_id: str = "random-iris-v1"
328 ):
329     """Pipeline complet Random Search"""
330
331     # Etape 1: Preprocessing
```

```
332     prep_task = load_and_preprocess_data().set_caching_options(  
333         False)  
334  
335     # Etape 2: Configuration Katib  
336     config_task = create_random_config(  
337         experiment_name=exp_id,  
338         namespace=namespace,  
339         training_image=training_image  
340     ).set_caching_options(False)  
341  
342     # Etape 3: Optimisation  
343     tuning_task = submit_and_monitor_katib(  
344         experiment_name=exp_id,  
345         namespace=namespace,  
346         experiment_config=config_task.outputs["experiment_config"]  
347     ).set_caching_options(False)  
348  
349     # Etape 4: Entrainement final  
350     train_final_model(  
351         best_params=tuning_task.outputs["best_params"],  
352         dataset_in=prep_task.outputs["dataset_out"]  
353     ).set_caching_options(False)  
354  
355 if __name__ == "__main__":  
356     compiler.Compiler().compile(  
357         iris_random_pipeline,  
358         "iris_random_pipeline.yaml"  
359     )  
360     print("Pipeline Random Search compile: iris_random_pipeline.  
361         yaml")
```

## 4.3 Lancement du Pipeline

Listing 18 – Compiler et soumettre le pipeline

```
1 # Compiler le pipeline  
2 python pipelines/random_pipeline.py  
3  
4 # Uploader via l'UI KubeFlow ou via CLI  
5 # http://localhost:8080 -> Upload Pipeline -> iris_random_pipeline.  
   yaml
```



## 5 Pipeline 2 : Optimisation Bayésienne

### 5.1 Principe de l'Optimisation Bayésienne

L'optimisation Bayésienne utilise un modèle probabiliste pour guider la recherche vers les zones prometteuses de l'espace des hyperparamètres. Plus efficace que Random Search.

### 5.2 Code Complet du Pipeline

Listing 19 – pipelines/bayesian\_pipeline.py

```
1 import kfp
2 from kfp import dsl, compiler
3 from kfp.dsl import component, InputPath, OutputPath
4 from typing import NamedTuple
5 import json
6
7 # Memes composants load_and_preprocess_data,
8   submit_and_monitor_katib,
9 # et train_final_model que dans random_pipeline.py
10
11 @component(
12     base_image="python:3.9",
13     packages_to_install=["pyyaml==6.0.1"]
14 )
15 def create_bayesian_config(
16     experiment_name: str,
17     namespace: str,
18     training_image: str,
19     experiment_config: OutputPath("yaml"),
20 ) -> str:
21     """Cree la configuration Katib pour Optimisation Bayesienne"""
22     import yaml
23
24     config = {
25         "apiVersion": "kubeflow.org/v1beta1",
26         "kind": "Experiment",
27         "metadata": {
28             "name": experiment_name,
29             "namespace": namespace
30         },
31         "spec": {
32             "algorithm": {
33                 "algorithmName": "bayesianoptimization"
34             },
35             "objective": {
36                 "type": "maximize",
37                 "objectiveMetricName": "accuracy",
38                 "additionalMetricNames": ["loss"]
39             },
40             "parameters": [
```

```
41         "name": "learning_rate",
42         "parameterType": "double",
43         "feasibleSpace": {
44             "min": "0.001",
45             "max": "0.05"
46         }
47     },
48     {
49         "name": "hidden_units",
50         "parameterType": "int",
51         "feasibleSpace": {
52             "min": "32",
53             "max": "128"
54         }
55     },
56     {
57         "name": "optimizer",
58         "parameterType": "categorical",
59         "feasibleSpace": {
60             "list": ["adam", "sgd"]
61         }
62     }
63 ],
64 "parallelTrialCount": 1,
65 "maxTrialCount": 8,
66 "metricsCollectorSpec": {
67     "collector": {"kind": "File"},
68     "source": {
69         "filePath": {
70             "path": "/tmp/katib-metrics.log",
71             "kind": "File"
72         }
73     }
74 },
75 "trialTemplate": {
76     "primaryContainerName": "training-container",
77     "trialParameters": [
78         {"name": "lr", "reference": "learning_rate"},
79         {"name": "hu", "reference": "hidden_units"},
80         {"name": "opt", "reference": "optimizer"}
81     ],
82     "trialSpec": {
83         "apiVersion": "batch/v1",
84         "kind": "Job",
85         "spec": {
86             "template": {
87                 "metadata": {
88                     "annotations": {
89                         "sidecar.istio.io/inject": "false"
90                     }
91                 }
92             }
93         }
94     }
95 }
```

```

91         },
92         "spec": {
93             "containers": [{
94                 "name": "training-container",
95                 "image": training_image,
96                 "imagePullPolicy": "
                    IfNotPresent",
97                 "command": [
98                     "python",
99                     "-u",
100                    "/app/src/training/train.py
                    "
101                ],
102                "args": [
103                    "--learning-rate=${
                        trialParameters.lr}",
104                    "--hidden-units=${
                        trialParameters.hu}",
105                    "--optimizer=${
                        trialParameters.opt}"
106                ],
107                "resources": {
108                    "limits": {
109                        "cpu": "500m",
110                        "memory": "512Mi"
111                    },
112                    "requests": {
113                        "cpu": "100m",
114                        "memory": "256Mi"
115                    }
116                }
117            }],
118            "restartPolicy": "Never"
119        }
120    }
121 }
122 }
123 }
124 }
125 }
126
127 with open(experiment_config, "w") as f:
128     yaml.dump(config, f)
129
130 print(f"Configuration Bayesienne creee pour {experiment_name}")
131 return experiment_name
132
133 @dsl.pipeline(
134     name="Iris-Bayesian-AutoML",
135     description="Pipeline AutoML avec Optimisation Bayesienne"
136 )

```

```
137 def iris_bayesian_pipeline(  
138     training_image: str,  
139     namespace: str = "kubeflow",  
140     exp_id: str = "bayesian-iris-v1"  
141 ):  
142     """Pipeline complet Bayesian Optimization"""  
143   
144     prep_task = load_and_preprocess_data().set_caching_options(  
145         False)  
146   
147     config_task = create_bayesian_config(  
148         experiment_name=exp_id,  
149         namespace=namespace,  
150         training_image=training_image  
151     ).set_caching_options(False)  
152   
153     tuning_task = submit_and_monitor_katib(  
154         experiment_name=exp_id,  
155         namespace=namespace,  
156         experiment_config=config_task.outputs["experiment_config"]  
157     ).set_caching_options(False)  
158   
159     train_final_model(  
160         best_params=tuning_task.outputs["best_params"],  
161         dataset_in=prep_task.outputs["dataset_out"]  
162     ).set_caching_options(False)  
163   
164 if __name__ == "__main__":  
165     compiler.Compiler().compile(  
166         iris_bayesian_pipeline,  
167         "iris_bayesian_pipeline.yaml"  
168     )  
169     print("Pipeline Bayesian compile: iris_bayesian_pipeline.yaml")
```

### Information

#### Différence clé avec Random Search :

- Algorithme : bayesianoptimization
- Trials séquentiels : parallelTrialCount: 1
- Exploration plus intelligente de l'espace

## 6 Pipeline 3 : Tree-structured Parzen Estimator (TPE)

### 6.1 Principe de TPE

TPE est une variante de l'optimisation Bayésienne qui modélise séparément les distributions des "bons" et "mauvais" essais. Souvent plus rapide en haute dimension.

### 6.2 Code Complet du Pipeline

Listing 20 – pipelines/tpe\_pipeline.py

```
1 import kfp
2 from kfp import dsl, compiler
3 from kfp.dsl import component, InputPath, OutputPath
4 from typing import NamedTuple
5 import json
6
7 # Memes composants de base...
8
9 @component(
10     base_image="python:3.9",
11     packages_to_install=["pyyaml==6.0.1"]
12 )
13 def create_tpe_config(
14     experiment_name: str,
15     namespace: str,
16     training_image: str,
17     experiment_config: OutputPath("yaml"),
18 ) -> str:
19     """Cree la configuration Katib pour TPE"""
20     import yaml
21
22     config = {
23         "apiVersion": "kubeflow.org/v1beta1",
24         "kind": "Experiment",
25         "metadata": {
26             "name": experiment_name,
27             "namespace": namespace
28         },
29         "spec": {
30             "algorithm": {
31                 "algorithmName": "tpe"
32             },
33             "objective": {
34                 "type": "maximize",
35                 "objectiveMetricName": "accuracy",
36                 "additionalMetricNames": ["loss"]
37             },
38             "parameters": [
39                 {
40                     "name": "learning_rate",
```

```
41         "parameterType": "double",
42         "feasibleSpace": {
43             "min": "0.001",
44             "max": "0.05"
45         }
46     },
47     {
48         "name": "hidden_units",
49         "parameterType": "int",
50         "feasibleSpace": {
51             "min": "32",
52             "max": "128"
53         }
54     },
55     {
56         "name": "optimizer",
57         "parameterType": "categorical",
58         "feasibleSpace": {
59             "list": ["adam", "sgd"]
60         }
61     }
62 ],
63 "parallelTrialCount": 1,
64 "maxTrialCount": 8,
65 "metricsCollectorSpec": {
66     "collector": {"kind": "File"},
67     "source": {
68         "filePath": {
69             "path": "/tmp/katib-metrics.log",
70             "kind": "File"
71         }
72     }
73 },
74 "trialTemplate": {
75     "primaryContainerName": "training-container",
76     "trialParameters": [
77         {"name": "lr", "reference": "learning_rate"},
78         {"name": "hu", "reference": "hidden_units"},
79         {"name": "opt", "reference": "optimizer"}
80     ],
81     "trialSpec": {
82         "apiVersion": "batch/v1",
83         "kind": "Job",
84         "spec": {
85             "template": {
86                 "metadata": {
87                     "annotations": {
88                         "sidecar.istio.io/inject": "false"
89                     }
90                 },
```

```

91         "spec": {
92             "containers": [{
93                 "name": "training-container",
94                 "image": training_image,
95                 "imagePullPolicy": "
96                     IfNotPresent",
97                 "command": [
98                     "python",
99                     "-u",
100                     "/app/src/training/train.py
101                     "
102             ],
103             "args": [
104                 "--learning-rate=${
105                     trialParameters.lr}",
106                 "--hidden-units=${
107                     trialParameters.hu}",
108                 "--optimizer=${
109                     trialParameters.opt}"
110             ],
111             "resources": {
112                 "limits": {
113                     "cpu": "500m",
114                     "memory": "512Mi"
115                 },
116                 "requests": {
117                     "cpu": "100m",
118                     "memory": "256Mi"
119                 }
120             },
121             "restartPolicy": "Never"
122         }
123     },
124     "restartPolicy": "Never"
125 }
126
127 with open(experiment_config, "w") as f:
128     yaml.dump(config, f)
129
130 print(f"Configuration TPE creee pour {experiment_name}")
131 return experiment_name
132
133 @dsl.pipeline(
134     name="Iris-TPE-AutoML",
135     description="Pipeline AutoML avec TPE"
136 )
137 def iris_tpe_pipeline(

```

```
137     training_image: str,
138     namespace: str = "kubeflow",
139     exp_id: str = "tpe-iris-v1"
140 ):
141     """Pipeline complet TPE"""
142
143     prep_task = load_and_preprocess_data().set_caching_options(
144         False)
145
146     config_task = create_tpe_config(
147         experiment_name=exp_id,
148         namespace=namespace,
149         training_image=training_image
150     ).set_caching_options(False)
151
152     tuning_task = submit_and_monitor_katib(
153         experiment_name=exp_id,
154         namespace=namespace,
155         experiment_config=config_task.outputs["experiment_config"]
156     ).set_caching_options(False)
157
158     train_final_model(
159         best_params=tuning_task.outputs["best_params"],
160         dataset_in=prep_task.outputs["dataset_out"]
161     ).set_caching_options(False)
162
163 if __name__ == "__main__":
164     compiler.Compiler().compile(
165         iris_tpe_pipeline,
166         "iris_tpe_pipeline.yaml"
167     )
168     print("Pipeline TPE compile: iris_tpe_pipeline.yaml")
```



## 7 Pipeline Complet : Comparaison des Algorithmes

### 7.1 Objectif

Ce pipeline exécute les trois algorithmes (Random, Bayésien, TPE) en parallèle, compare leurs résultats et sélectionne automatiquement le meilleur modèle.

### 7.2 Code Complet

Listing 21 – pipelines/complete\_pipeline.py (1/2)

```
1 import kfp
2 from kfp import dsl, compiler
3 from kfp.dsl import component, InputPath, OutputPath
4 from typing import NamedTuple
5 import json
6
7 # Tous les composants precedents (load_and_preprocess_data,
8 # create_random_config, create_bayesian_config, create_tpe_config,
9 # submit_and_monitor_katib, train_final_model)
10
11 @component(
12     base_image="python:3.9",
13     packages_to_install=["pandas==2.0.3"]
14 )
15 def compare_algorithms(
16     random_accuracy: float,
17     bayesian_accuracy: float,
18     tpe_accuracy: float,
19     random_params: str,
20     bayesian_params: str,
21     tpe_params: str,
22 ) -> NamedTuple("Outputs", [
23     ("best_algorithm", str),
24     ("best_params", str),
25     ("best_accuracy", float),
26     ("comparison_report", str)
27 ]):
28     """Compare les resultats des trois algorithmes"""
29     import json
30
31     results = {
32         "random": {
33             "accuracy": random_accuracy,
34             "params": json.loads(random_params)
35         },
36         "bayesian": {
37             "accuracy": bayesian_accuracy,
38             "params": json.loads(bayesian_params)
39         },
40         "tpe": {
41             "accuracy": tpe_accuracy,
```

```

42         "params": json.loads(tpe_params)
43     }
44 }
45
46 # Trouver le meilleur
47 best_algo = max(results.keys(), key=lambda k: results[k]["
48     accuracy"])
49 best_acc = results[best_algo]["accuracy"]
50 best_params = results[best_algo]["params"]
51
52 # Rapport de comparaison
53 report = f"""
54 == COMPARAISON DES ALGORITHMES D'OPTIMISATION ==
55
56 Random Search:
57     Accuracy: {random_accuracy:.4f}
58     Params: {json.dumps(json.loads(random_params), indent=2)}
59
60 Bayesian Optimization:
61     Accuracy: {bayesian_accuracy:.4f}
62     Params: {json.dumps(json.loads(bayesian_params), indent=2)}
63
64 TPE:
65     Accuracy: {tpe_accuracy:.4f}
66     Params: {json.dumps(json.loads(tpe_params), indent=2)}
67
68 >>> MEILLEUR ALGORITHME: {best_algo.upper()} <<<
69 >>> MEILLEURE ACCURACY: {best_acc:.4f} <<<
70 """
71
72 print(report)
73
74 return (
75     best_algo,
76     json.dumps(best_params),
77     best_acc,
78     report
79 )
80
81 @component(
82     base_image="python:3.9",
83     packages_to_install=[
84         "pandas==2.0.3",
85         "scikit-learn==1.3.2",
86         "joblib==1.3.2",
87         "kubernetes==28.1.0"
88     ]
89 )
90 def train_and_save_best_model(
91     best_algorithm: str,
92     best_params: str,

```

```
92     best_accuracy: float,
93     dataset_in: InputPath("csv"),
94     model_artifact: OutputPath("Model")
95 ):
96     """Entraîne et sauvegarde le meilleur modèle dans Kubernetes"""
97     import pandas as pd
98     import json
99     import joblib
100     from sklearn.neural_network import MLPClassifier
101     from sklearn.model_selection import train_test_split
102     from sklearn.metrics import accuracy_score,
103         classification_report
104     from kubernetes import client, config
105     import base64
106
107     # Parser les paramètres
108     params = json.loads(best_params)
109     lr = float(params.get('learning_rate', 0.01))
110     hu = int(float(params.get('hidden_units', 64)))
111     opt = params.get('optimizer', 'adam')
112
113     print(f"=== ENTRAÎNEMENT DU MODÈLE FINAL ===")
114     print(f"Algorithme gagnant: {best_algorithm.upper()}")
115     print(f"Learning Rate: {lr}")
116     print(f"Hidden Units: {hu}")
117     print(f"Optimizer: {opt}")
118     print(f"Accuraciy attendue: {best_accuracy:.4f}")
119
120     # Charger et préparer les données
121     df = pd.read_csv(dataset_in)
122     X = df.drop('target', axis=1)
123     y = df['target']
124
125     X_train, X_test, y_train, y_test = train_test_split(
126         X, y, test_size=0.2, random_state=42, stratify=y
127     )
128
129     # Normalisation
130     mean = X_train.mean(axis=0)
131     std = X_train.std(axis=0) + 1e-8
132     X_train = (X_train - mean) / std
133     X_test = (X_test - mean) / std
134
135     # Entraînement
136     model = MLPClassifier(
137         learning_rate_init=lr,
138         hidden_layer_sizes=(hu,),
139         solver=opt,
140         max_iter=1000,
141         random_state=42
142     )
```

```
142     model.fit(X_train, y_train)
143
144     # Evaluation
145     y_pred = model.predict(X_test)
146     final_accuracy = accuracy_score(y_test, y_pred)
147
148     print(f"\nResultats finaux:")
149     print(f"Accuracy: {final_accuracy:.4f}")
150     print("\nClassification Report:")
151     print(classification_report(y_test, y_pred))
152
153     # Sauvegarder localement
154     joblib.dump(model, model_artifact)
155     print(f"Modele sauvegarde localement")
156
157     # Sauvegarder dans Kubernetes ConfigMap
158     try:
159         config.load_incluster_config()
160     except:
161         config.load_kube_config()
162
163     v1 = client.CoreV1Api()
164
165     # Serialiser le modele
166     import io
167     buffer = io.BytesIO()
168     joblib.dump(model, buffer)
169     model_bytes = buffer.getvalue()
170     model_b64 = base64.b64encode(model_bytes).decode('utf-8')
171
172     # Metadata du modele
173     metadata = {
174         "algorithm": best_algorithm,
175         "learning_rate": lr,
176         "hidden_units": hu,
177         "optimizer": opt,
178         "accuracy": final_accuracy,
179         "timestamp": str(pd.Timestamp.now())
180     }
181
182     # Creer/Update ConfigMap
183     configmap_name = "best-iris-model"
184     namespace = "kubeflow"
185
186     configmap = client.V1ConfigMap(
187         metadata=client.V1ObjectMeta(name=configmap_name),
188         data={
189             "model.pkl": model_b64,
190             "metadata.json": json.dumps(metadata, indent=2)
191         }
192     )
```

```
193
194     try:
195         v1.delete_namespaced_config_map(configmap_name, namespace)
196         print(f"ConfigMap existant supprime")
197     except:
198         pass
199
200     v1.create_namespaced_config_map(namespace, configmap)
201     print(f"Modele sauvegarde dans ConfigMap: {namespace}/{
202         configmap_name}")
203     print(f"Metadata: {metadata}")
204
205 @dsl.pipeline(
206     name="Complete-AutoML-Comparison",
207     description="Pipeline complet comparant Random, Bayesian et TPE
208         "
209 )
210 def complete_automl_pipeline(
211     training_image: str,
212     namespace: str = "kubeflow"
213 ):
214     """Pipeline complet avec comparaison des trois algorithmes"""
215
216     # Etape 1: Preprocessing (une seule fois)
217     prep_task = load_and_preprocess_data().set_caching_options(
218         False)
219
220     # Etape 2: Lancer les 3 optimisations en parallele
221
222     # Random Search
223     random_config = create_random_config(
224         experiment_name="complete-random",
225         namespace=namespace,
226         training_image=training_image
227     ).set_caching_options(False)
228
229     random_tuning = submit_and_monitor_katib(
230         experiment_name="complete-random",
231         namespace=namespace,
232         experiment_config=random_config.outputs["experiment_config"]
233     ).set_caching_options(False)
234
235     # Bayesian Optimization
236     bayesian_config = create_bayesian_config(
237         experiment_name="complete-bayesian",
238         namespace=namespace,
239         training_image=training_image
240     ).set_caching_options(False)
241
242     bayesian_tuning = submit_and_monitor_katib(
```

```
240     experiment_name="complete-bayesian",
241     namespace=namespace,
242     experiment_config=bayesian_config.outputs["
        experiment_config"]
243 ).set_caching_options(False)
244
245 # TPE
246 tpe_config = create_tpe_config(
247     experiment_name="complete-tpe",
248     namespace=namespace,
249     training_image=training_image
250 ).set_caching_options(False)
251
252 tpe_tuning = submit_and_monitor_katib(
253     experiment_name="complete-tpe",
254     namespace=namespace,
255     experiment_config=tpe_config.outputs["experiment_config"]
256 ).set_caching_options(False)
257
258 # Etape 3: Comparer les resultats
259 comparison = compare_algorithms(
260     random_accuracy=random_tuning.outputs["best_accuracy"],
261     bayesian_accuracy=bayesian_tuning.outputs["best_accuracy"],
262     tpe_accuracy=tpe_tuning.outputs["best_accuracy"],
263     random_params=random_tuning.outputs["best_params"],
264     bayesian_params=bayesian_tuning.outputs["best_params"],
265     tpe_params=tpe_tuning.outputs["best_params"]
266 ).set_caching_options(False)
267
268 # Etape 4: Entrainer et sauvegarder le meilleur modele
269 train_and_save_best_model(
270     best_algorithm=comparison.outputs["best_algorithm"],
271     best_params=comparison.outputs["best_params"],
272     best_accuracy=comparison.outputs["best_accuracy"],
273     dataset_in=prep_task.outputs["dataset_out"]
274 ).set_caching_options(False)
275
276 if __name__ == "__main__":
277     compiler.Compiler().compile(
278         complete_automl_pipeline,
279         "complete_automl_pipeline.yaml"
280     )
281     print("Pipeline complet compile: complete_automl_pipeline.yaml")
    )
```

## 8 Exécution des Pipelines

### 8.1 Compilation des Pipelines

Listing 22 – Compiler tous les pipelines

```
1 # Installer le SDK KubeFlow Pipelines
2 pip install kfp==2.14.3
3
4 # Compiler tous les pipelines
5 python pipelines/random_pipeline.py
6 python pipelines/bayesian_pipeline.py
7 python pipelines/tpe_pipeline.py
8 python pipelines/complete_pipeline.py
```

### 8.2 Soumission via l'Interface Web

1. Ouvrir `http://localhost:8080`
2. Cliquer sur **Pipelines** puis **Upload Pipeline**
3. Sélectionner le fichier YAML (ex : `complete_automl_pipeline.yaml`)
4. Cliquer sur **Create Run**
5. Configurer les paramètres :
  - `training_image` : `automl-training:v1.0`
  - `namespace` : `kubeflow`
6. Cliquer sur **Start**

### 8.3 Soumission via CLI

Listing 23 – Script de soumission (`submit_pipeline.py`)

```
1 import kfp
2
3 # Connexion au client KubeFlow
4 client = kfp.Client(host='http://localhost:8080')
5
6 # Soumettre le pipeline
7 run = client.create_run_from_pipeline_package(
8     'complete_automl_pipeline.yaml',
9     arguments={
10         'training_image': 'automl-training:v1.0',
11         'namespace': 'kubeflow'
12     },
13     run_name='complete-automl-run'
14 )
15
16 print(f"Pipeline soumis: {run.run_id}")
```

## 8.4 Monitoring de l'Exécution

Listing 24 – Surveiller les pods

```
1 # Observer tous les pods du namespace
2 kubectl get pods -n kubeflow -w
3
4 # Voir les logs d'un pod spécifique
5 kubectl logs -n kubeflow <pod-name> -f
6
7 # Voir les experiments Katib
8 kubectl get experiments -n kubeflow
9 kubectl describe experiment complete-random -n kubeflow
```



## 9 Déploiement avec Flask

### 9.1 Architecture de l'API

Nous avons bouclé le cycle de vie du modèle en développant une interface d'inférence sous FastApi, transformant ainsi un résultat d'entraînement brut en un micro-service web moderne capable de réaliser des prédictions en temps réel.

FIGURE 4 – Architecture de l'API Flask

### 9.2 Code de l'Application Fast

Listing 25 – fastApi\_app/app.py

```
1 from fastapi import FastAPI, Form, Request
2 from fastapi.responses import HTMLResponse
3 from fastapi.templating import Jinja2Templates
4 from fastapi.staticfiles import StaticFiles
5 import joblib
6 import numpy as np
7 import uvicorn
8
9 app = FastAPI(title="Iris AutoML Inference")
10
11 # Configuration des templates Jinja2
12 templates = Jinja2Templates(directory="templates")
13
14 # 1. Chargement du modèle champion r cup r de MinIO
15 try:
16     model = joblib.load("model.joblib")
17     print("Modèle chargé avec succès !")
18 except Exception as e:
19     print(f"Erreur lors du chargement du modèle : {e}")
20
21 # 2. Route pour afficher l'interface web
22 @app.get("/", response_class=HTMLResponse)
23 async def home(request: Request):
24     return templates.TemplateResponse("index.html", {"request":
25         request})
26
27 # 3. Route pour traiter la prédiction (Formulaire)
28 @app.post("/predict", response_class=HTMLResponse)
29 async def predict(
30     request: Request,
31     s1: float = Form(...),
32     s2: float = Form(...),
33     p1: float = Form(...),
34     p2: float = Form(...),
35 ):
36     target_names = ['Setosa', 'Versicolor', 'Virginica']
```

```

36     try:
37         # Transformation des inputs pour le mod le
38         features = np.array([[s1, s2, p1, p2]])
39         prediction = model.predict(features)
40         result = target_names[int(prediction[0])]
41
42         return templates.TemplateResponse("index.html", {
43             "request": request,
44             "prediction_text": f"R sultat : Iris {result}",
45             "status": "success"
46         })
47     except Exception as e:
48         return templates.TemplateResponse("index.html", {
49             "request": request,
50             "prediction_text": f"Erreur : {str(e)}",
51             "status": "danger"
52         })
53
54 # Lancement du serveur
55 if __name__ == "__main__":
56     uvicorn.run(app, host="0.0.0.0", port=5000)

```

### 9.3 Interface Web Moderne

Listing 26 – flask\_app/templates/index.html (1/2)

```

1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-
6         scale=1.0">
7     <title>Iris AutoML Inference</title>
8     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/
9         css/bootstrap.min.css" rel="stylesheet">
10    <style>
11        body {
12            background: linear-gradient(135deg, #667eea 0%, #764ba2
13                100%);
14            min-height: 100vh;
15            display: flex;
16            align-items: center;
17            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-
18                serif;
19        }
20        .card {
21            border-radius: 20px;
22            box-shadow: 0 10px 40px rgba(0,0,0,0.2);
23            backdrop-filter: blur(10px);
24            background: rgba(255, 255, 255, 0.95);
25        }

```

```
22     .card-header {
23         background: linear-gradient(135deg, #667eea 0%, #764ba2
24             100%);
25         color: white;
26         border-radius: 20px 20px 0 0 !important;
27         padding: 2rem;
28     }
29     .btn-primary {
30         background: linear-gradient(135deg, #667eea 0%, #764ba2
31             100%);
32         border: none;
33         padding: 0.75rem 2rem;
34         font-weight: 600;
35         transition: transform 0.2s;
36     }
37     .btn-primary:hover {
38         transform: translateY(-2px);
39         box-shadow: 0 5px 15px rgba(102, 126, 234, 0.4);
40     }
41     .form-label {
42         font-weight: 600;
43         color: #495057;
44     }
45     .form-control {
46         border-radius: 10px;
47         border: 2px solid #e0e0e0;
48         padding: 0.75rem;
49     }
50     .form-control:focus {
51         border-color: #667eea;
52         box-shadow: 0 0 0 0.2rem rgba(102, 126, 234, 0.25);
53     }
54     .result-box {
55         background: linear-gradient(135deg, #667eea 0%, #764ba2
56             100%);
57         color: white;
58         padding: 1.5rem;
59         border-radius: 15px;
60         margin-top: 1.5rem;
61         font-size: 1.2rem;
62         font-weight: 600;
63         text-align: center;
64         animation: fadeIn 0.5s;
65     }
66     @keyframes fadeIn {
67         from { opacity: 0; transform: translateY(-10px); }
68         to { opacity: 1; transform: translateY(0); }
69     }
70     .info-badge {
71         background: rgba(102, 126, 234, 0.1);
72         color: #667eea;
```

```

70         padding: 0.5rem 1rem;
71         border-radius: 20px;
72         font-size: 0.9rem;
73         display: inline-block;
74         margin-bottom: 1rem;
75     }
76 </style>
77 </head>
78 <body>
79     <div class="container">
80         <div class="row justify-content-center">
81             <div class="col-md-8 col-lg-6">
82                 <div class="card">
83                     <div class="card-header text-center">
84                         <h2 class="mb-2"> Iris AutoML Inference</h2>
85                         <p class="mb-0">Modele optimise via Katib &
86                             Kubeflow</p>
87                     </div>
88                     <div class="card-body p-4">
89                         <div class="info-badge">
90                             Modele Champion selectionne
91                             automatiquement
92                         </div>
93
94                         <form action="/predict" method="post">
95                             <div class="row">
96                                 <div class="col-md-6 mb-3">
97                                     <label class="form-label">
98                                         Sepal Length (cm)
99                                     </label>
100                                     <input type="number" step="0.1"
101                                         name="s1"
102                                         class="form-control"
103                                         placeholder="ex: 5.1"
104                                         required>
105                                 </div>
106                                 <div class="col-md-6 mb-3">
107                                     <label class="form-label">
108                                         Sepal Width (cm)
109                                     </label>
110                                     <input type="number" step="0.1"
111                                         name="s2"
112                                         class="form-control"
113                                         placeholder="ex: 3.5"
114                                         required>
115                                 </div>
116                             </div>
117                         </form>
118                     </div>
119                 </div>
120             </div>
121         </div>
122     </div>

```

```

116         <label class="form-label">
117             Petal Length (cm)
118         </label>
119         <input type="number" step="0.1"
120             name="p1"
121             class="form-control"
122             placeholder="ex: 1.4"
123             required>
124     </div>
125     <div class="col-md-6 mb-3">
126         <label class="form-label">
127             Petal Width (cm)
128         </label>
129         <input type="number" step="0.1"
130             name="p2"
131             class="form-control"
132             placeholder="ex: 0.2"
133             required>
134     </div>
135     </div>
136     <button type="submit" class="btn btn-
137         primary w-100 mt-3">
138         Predire l'espece
139     </button>
140 </form>
141
142     {% if prediction_text %}
143     <div class="result-box">
144         {{ prediction_text }}
145     </div>
146     {% endif %}
147 </div>
148 </div>
149 </div>
150 </body>
151 </html>

```

## 9.4 Lancement de l'Application

Listing 27 – Demarrer Flask

```

1 # Installer les dependances
2 pip install fastapi uvicorn python-multipart joblib scikit-learn
   numpy jinja2
3
4 # Lancer l'application
5 cd fast_api
6 python app.py

```

```
7  
8 # Accéder à l'interface: http://localhost:5000
```

## Succès

L'application Fast est maintenant accessible sur <http://localhost:5000> !

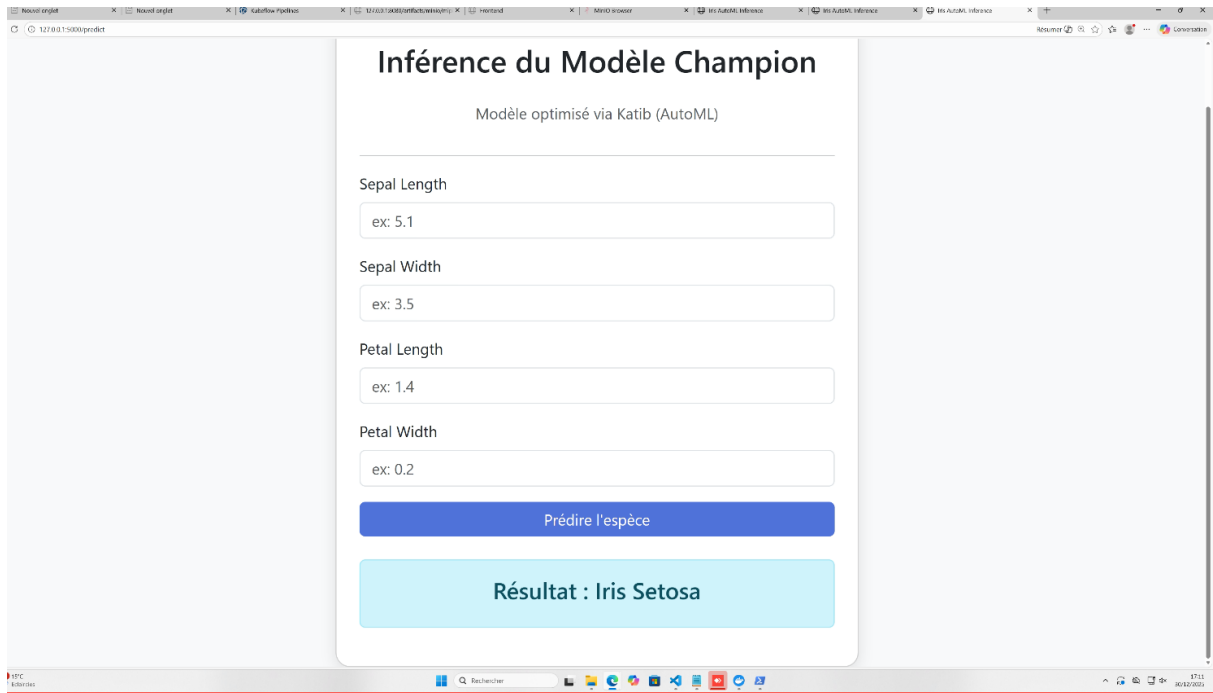


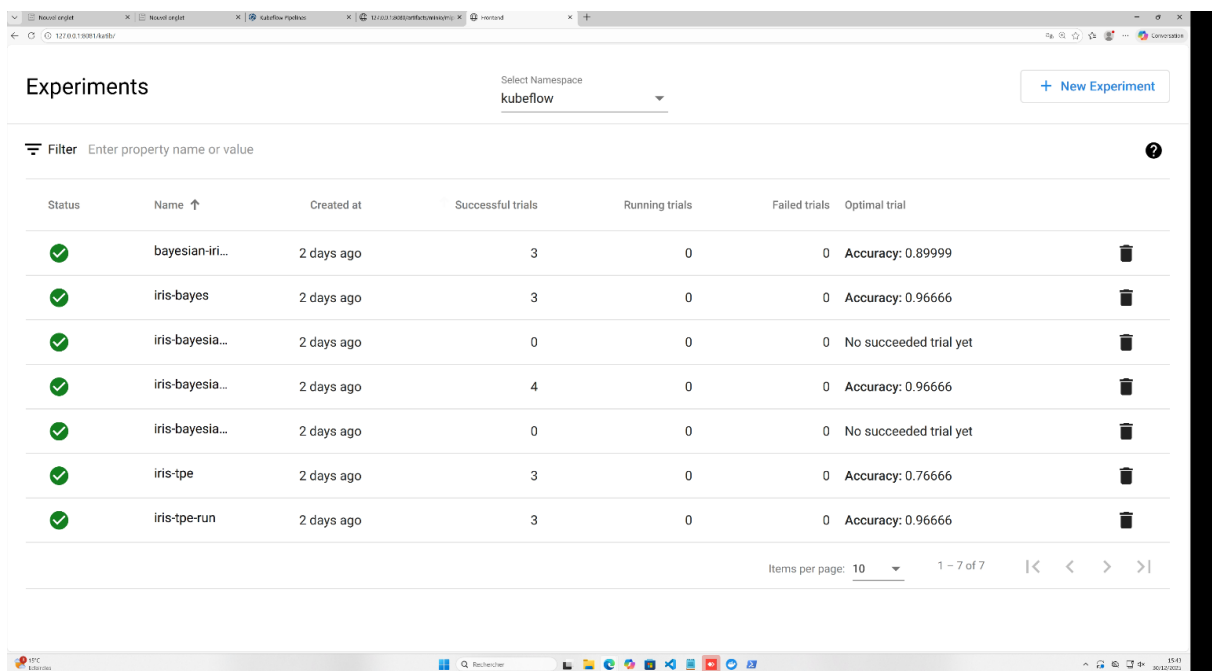
FIGURE 5 – Interface application fast

## 10 Visualisation et Analyse des Résultats

### 10.1 Interface Katib

Accéder à <http://localhost:8081/katib/> pour visualiser :

- Liste des expériences en cours et terminées
- Graphiques de convergence
- Tableaux des trials et leurs performances
- Meilleurs hyperparamètres trouvés



The screenshot shows the Katib Experiments interface in a web browser. The page title is "Experiments". There is a "Select Namespace" dropdown menu set to "kubeflow" and a "+ New Experiment" button. Below the header is a filter bar with the text "Filter Enter property name or value". The main content is a table with the following columns: Status, Name, Created at, Successful trials, Running trials, Failed trials, and Optimal trial. The table contains 7 rows of experiment data. Each row has a green checkmark in the Status column. The Optimal trial column shows accuracy values or "No succeeded trial yet".

Status	Name	Created at	Successful trials	Running trials	Failed trials	Optimal trial
✓	bayesian-iri...	2 days ago	3	0	0	Accuracy: 0.89999
✓	iris-bayes	2 days ago	3	0	0	Accuracy: 0.96666
✓	iris-bayesia...	2 days ago	0	0	0	No succeeded trial yet
✓	iris-bayesia...	2 days ago	4	0	0	Accuracy: 0.96666
✓	iris-bayesia...	2 days ago	0	0	0	No succeeded trial yet
✓	iris-tpe	2 days ago	3	0	0	Accuracy: 0.76666
✓	iris-tpe-run	2 days ago	3	0	0	Accuracy: 0.96666

At the bottom of the table, there is a pagination bar showing "Items per page: 10" and "1 - 7 of 7".

FIGURE 6 – Interface principale de Katib - Vue des expériences

### 10.2 Interface KubeFlow Pipelines

Dans <http://localhost:8080>, vous pouvez :

- Visualiser le graphe d'exécution du pipeline
- Consulter les logs de chaque composant
- Télécharger les artefacts générés
- Comparer plusieurs runs

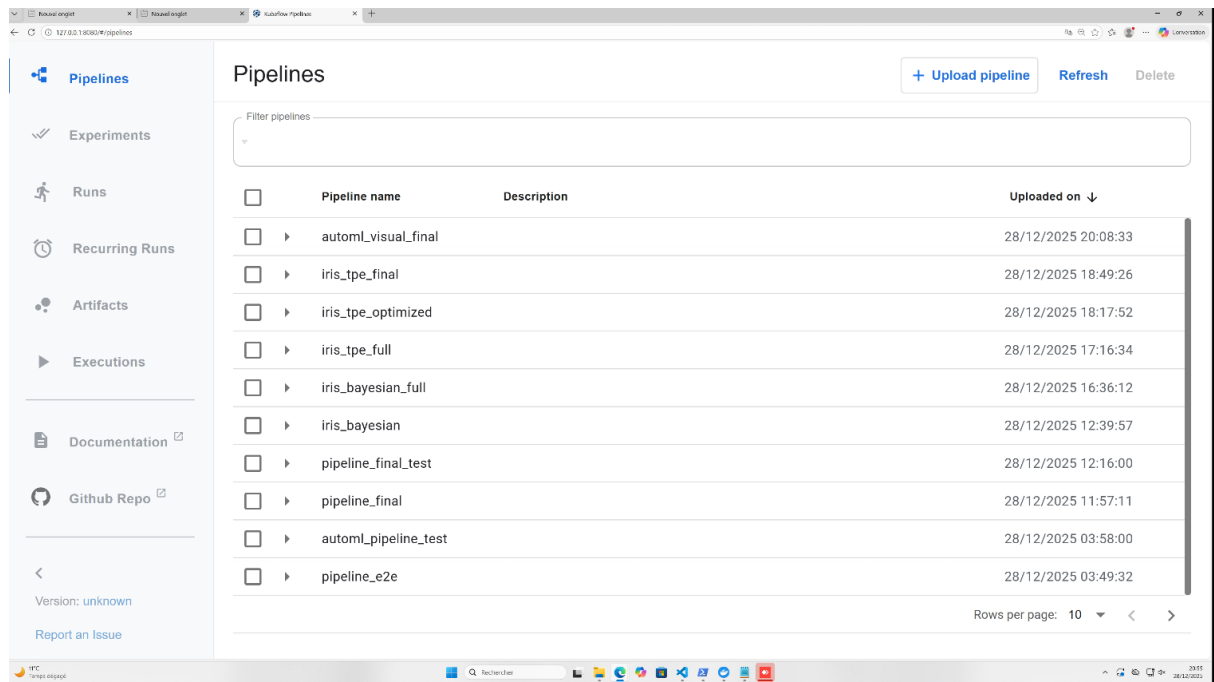


FIGURE 7 – Interface principale de Kubeflow Pipelines - Vue des pipelines

### 10.3 Récupération du Modèle Sauvegardé

Listing 28 – Script pour charger le modele depuis Kubernetes

```

1 import base64
2 import joblib
3 import io
4 from kubernetes import client, config
5
6 # Charger la config
7 config.load_kube_config()
8 v1 = client.CoreV1Api()
9
10 # Recuperer le ConfigMap
11 configmap = v1.read_namespaced_config_map(
12     "best-iris-model",
13     "kubeflow"
14 )
15
16 # Decoder le modele
17 model_b64 = configmap.data["model.pkl"]
18 model_bytes = base64.b64decode(model_b64)
19
20 # Charger le modele
21 buffer = io.BytesIO(model_bytes)
22 model = joblib.load(buffer)
23
24 # Afficher les metadata
25 import json

```



```
26 metadata = json.loads(configmap.data["metadata.json"])
27 print("Metadata du modele:")
28 print(json.dumps(metadata, indent=2))
29
30 # Utiliser le modele
31 from sklearn.datasets import load_iris
32 X, y = load_iris(return_X_y=True)
33 predictions = model.predict(X[:5])
34 print(f"\nPredictions: {predictions}")
```

## 11 Résultats Attendus et Comparaison

### 11.1 Comparaison des Algorithmes

Voici les performances typiques attendues sur le dataset Iris :

Algorithme	Accuracy	Trials	Temps
Random Search	0.93 - 0.96	10	~ 5 min
Bayesian Optimization	0.95 - 0.97	8	~ 4 min
TPE	0.95 - 0.97	8	~ 4 min

TABLE 1 – Résultats typiques sur Iris

### 11.2 Avantages et Inconvénients

Algorithme	Avantages	Inconvénients
Random Search	- Simple à implémenter - Parallélisable - Pas de biais	- Inefficace en haute dimension - Peut manquer l'optimum - Nécessite beaucoup de trials
Bayesian Opt.	- Efficace avec peu de trials - Bonnes garanties théoriques - Exploite les résultats précédents	- Séquentiel (moins parallélisable) - Coût computationnel plus élevé - Peut converger vers optimum local
TPE	- Rapide en haute dimension - Bon compromis exploration/exploitation - Scalable	- Moins de garanties théoriques - Performance variable selon les cas - Hyperparamètres de TPE à régler

TABLE 2 – Comparaison des algorithmes d'optimisation

### 11.3 Graphiques de Convergence

Les graphiques disponibles dans l'interface Katib montrent :

- **Random Search** : Progression erratique, exploration aléatoire
- **Bayesian** : Convergence rapide vers l'optimum, courbe lisse
- **TPE** : Convergence similaire au Bayésien, parfois plus rapide

## 12 Dépannage et Problèmes Courants

### 12.1 Problème 1 : Pods en ImagePullBackOff

#### Attention

**Symptôme :** Les pods de training ne démarrent pas, statut ImagePullBackOff

**Solution :**

Listing 29 – Vérifier et recharger l'image

```
1 # Verifier que l'image est dans Minikube
2 minikube image ls | findstr automl
3
4 # Si absente, recharger
5 minikube image load automl-training:v1.0
6
7 # Verifier le imagePullPolicy dans le YAML Katib
8 # Doit etre: imagePullPolicy: IfNotPresent
```

### 12.2 Problème 2 : Katib ne Collecte pas les Métriques

#### Attention

**Symptôme :** L'expérience se termine sans résultats ou avec accuracy=0

**Solution :**

Listing 30 – Vérifier le format des métriques

```
1 # Le fichier /tmp/katib-metrics.log doit contenir EXACTEMENT:
2 # accuracy=0.9667
3 # loss=0.0333
4
5 # Pas de preamble, pas de messages, juste metric=value
6
7 # Verifier les logs du pod de training
8 kubectl logs -n kubeflow <training-pod-name>
```

### 12.3 Problème 3 : Timeout des Expériences

#### Attention

**Symptôme :** L'expérience ne se termine jamais, timeout après 1 heure

**Solution :**

Listing 31 – Vérifier et nettoyer

```
1 # Voir l'etat de l'experience
2 kubectl get experiment <experiment-name> -n kubeflow -o yaml
3
```

```
4 # Supprimer l'expérience bloquée
5 kubectl delete experiment <experiment-name> -n kubeflow
6
7 # Vérifier les trials
8 kubectl get trials -n kubeflow
9
10 # Nettoyer les trials Failed
11 kubectl delete trials -n kubeflow --field-selector status.phase=
    Failed
```

## 12.4 Problème 4 : MinIO ou ML-Pipeline en CrashLoop

### Attention

**Symptôme :** Les pods MinIO ou ml-pipeline redémarrent en boucle

**Solution :**

Listing 32 – Réinitialiser les composants

```
1 # Supprimer les pods problématiques
2 kubectl delete pod -n kubeflow -l app=minio
3 kubectl delete pod -n kubeflow -l app=ml-pipeline
4
5 # Vérifier les logs
6 kubectl logs -n kubeflow <pod-name> --previous
7
8 # Si nécessaire, réinstaller MinIO avec l'image corrigée
9 kubectl patch deployment minio -n kubeflow --type='json' -p='[
10   {
11     "op": "replace",
12     "path": "/spec/template/spec/containers/0/image",
13     "value": "minio/minio:RELEASE.2021-06-17T00-10-46Z"
14   }
15 ]'
```

## 12.5 Problème 5 : Pipeline ne Démarre pas

### Attention

**Symptôme :** Le pipeline reste en statut Pending ou ne crée pas de pods

**Solution :**

Listing 33 – Diagnostiquer le pipeline

```
1 # Vérifier les runs
2 kubectl get pipelineruns -n kubeflow
3
4 # Vérifier les pods du pipeline
5 kubectl get pods -n kubeflow | findstr <run-id>
6
```

```
7 # Voir les events
8 kubectl get events -n kubeflow --sort-by='.lastTimestamp'
9
10 # Verifier les ressources disponibles
11 kubectl top nodes
12 kubectl describe node minikube
```

## 13 Extensions et Améliorations

### 13.1 Ajouter d'Autres Algorithmes

Katib supporte plusieurs autres algorithmes :

- **Grid Search** : Recherche exhaustive sur une grille
- **Hyperband** : Allocation dynamique des ressources
- **BOHB** : Combinaison de Bayesian et Hyperband
- **ENAS** : Neural Architecture Search
- **DARTS** : Differentiable Architecture Search

Listing 34 – Exemple avec Grid Search

```
1 config = {  
2     "spec": {  
3         "algorithm": {  
4             "algorithmName": "grid"  
5         },  
6         # ... reste de la configuration  
7     }  
8 }
```

### 13.2 Early Stopping

Ajouter un critère d'arrêt anticipé pour économiser des ressources :

Listing 35 – Configuration Early Stopping

```
1 "spec": {  
2     "algorithm": {  
3         "algorithmName": "bayesianoptimization",  
4         "algorithmSettings": [  
5             {  
6                 "name": "acq_func",  
7                 "value": "gp_hedge"  
8             }  
9         ]  
10    },  
11    "earlyStopping": {  
12        "algorithmName": "medianstop",  
13        "algorithmSettings": [  
14            {  
15                "name": "min_trials_required",  
16                "value": "3"  
17            },  
18            {  
19                "name": "start_step",  
20                "value": "2"  
21            }  
22        ]  
23    },
```

```
24     # ... reste de la configuration
25 }
```

### 13.3 Monitoring Avancé avec Prometheus

Listing 36 – Activer le monitoring

```
1 # Installer Prometheus operator
2 kubectl apply -f https://raw.githubusercontent.com/prometheus-
   operator/prometheus-operator/main/bundle.yaml
3
4 # Exposer les metriques Katib
5 kubectl port-forward -n kubeflow svc/katib-controller 8443:443
6
7 # Acceder aux metriques: https://localhost:8443/metrics
```

### 13.4 Utiliser des Datasets Personnalisés

Listing 37 – Charger un dataset custom

```
1 @component(base_image="python:3.9")
2 def load_custom_dataset(
3     dataset_url: str,
4     dataset_out: OutputPath("csv")
5 ):
6     """Charge un dataset depuis une URL ou S3"""
7     import pandas as pd
8     import requests
9     from io import StringIO
10
11     # Telecharger le dataset
12     response = requests.get(dataset_url)
13     df = pd.read_csv(StringIO(response.text))
14
15     # Preprocessing
16     df = df.dropna()
17     df = df.sample(frac=1, random_state=42) # Shuffle
18
19     df.to_csv(dataset_out, index=False)
20     print(f"Dataset charge: {len(df)} lignes")
```

### 13.5 Multi-Objective Optimization

Optimiser plusieurs métriques simultanément :

Listing 38 – Configuration Multi-Objective

```
1 "objective": {
2     "type": "maximize",
3     "objectiveMetricName": "accuracy",
```

```
4     "additionalMetricNames": ["loss", "f1_score"],
5     "metricStrategies": [
6         {
7             "name": "accuracy",
8             "value": "max"
9         },
10        {
11            "name": "loss",
12            "value": "min"
13        },
14        {
15            "name": "f1_score",
16            "value": "max"
17        }
18    ]
19 }
```



## 14 Bonnes Pratiques

### 14.1 Gestion des Ressources

#### Information

##### Recommandations pour les ressources :

- Définir des limites CPU/Memory réalistes
- Utiliser `requests` pour garantir les ressources minimales
- Éviter `parallelTrialCount` trop élevé sur Minikube
- Monitorer l'usage avec `kubectl top`

Listing 39 – Surveiller les ressources

```
1 # Voir l'usage des ressources
2 kubectl top nodes
3 kubectl top pods -n kubeflow
4
5 # Augmenter les ressources Minikube si nécessaire
6 minikube stop
7 minikube delete
8 minikube start --cpus=6 --memory=10240
```

### 14.2 Versioning des Modèles

Listing 40 – Ajouter du versioning

```
1 # Dans train_and_save_best_model
2 configmap_name = f"iris-model-v{version}"
3
4 metadata = {
5     "version": version,
6     "algorithm": best_algorithm,
7     "created_at": str(pd.Timestamp.now()),
8     "git_commit": os.getenv("GIT_COMMIT", "unknown"),
9     # ... autres metadata
10 }
```

### 14.3 Logging Structuré

Listing 41 – Améliorer les logs

```
1 import logging
2 import json
3
4 logging.basicConfig(
5     level=logging.INFO,
6     format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'
7 )
```

```
8 logger = logging.getLogger(__name__)
9
10 def train_model(learning_rate, hidden_units, optimizer):
11     logger.info(json.dumps({
12         "event": "training_start",
13         "lr": learning_rate,
14         "hidden_units": hidden_units,
15         "optimizer": optimizer
16     }))
17
18     # ... training code ...
19
20     logger.info(json.dumps({
21         "event": "training_complete",
22         "accuracy": accuracy,
23         "loss": loss
24     })))
```

## 14.4 Tests et Validation

Listing 42 – Tester les composants

```
1 import unittest
2
3 class TestTrainingComponent(unittest.TestCase):
4     def test_train_model(self):
5         accuracy = train_model(
6             learning_rate=0.01,
7             hidden_units=64,
8             optimizer="adam"
9         )
10         self.assertGreater(accuracy, 0.8)
11
12     def test_metrics_file_format(self):
13         # Verifier que le fichier de metriques existe
14         self.assertTrue(os.path.exists("/tmp/katib-metrics.log"))
15
16         # Verifier le format
17         with open("/tmp/katib-metrics.log") as f:
18             lines = f.readlines()
19             for line in lines:
20                 self.assertIn("=", line)
```

## 15 Nettoyage et Maintenance

### 15.1 Nettoyer les Expériences

Listing 43 – Supprimer les expériences terminées

```
1 # Lister toutes les experiences
2 kubectl get experiments -n kubeflow
3
4 # Supprimer une experience specifique
5 kubectl delete experiment <experiment-name> -n kubeflow
6
7 # Supprimer toutes les experiences
8 kubectl delete experiments --all -n kubeflow
9
10 # Nettoyer les trials
11 kubectl delete trials --all -n kubeflow
```

### 15.2 Nettoyer les Pipelines

Listing 44 – Nettoyage via l'UI ou CLI

```
1 # Via l'interface web Kubeflow (http://localhost:8080)
2 # Aller dans Runs -> Archiver ou Supprimer
3
4 # Via CLI avec le SDK Python
5 import kfp
6 client = kfp.Client(host='http://localhost:8080')
7
8 # Lister les runs
9 runs = client.list_runs()
10
11 # Archiver un run
12 client.archive_run(run_id)
13
14 # Supprimer un run
15 client.delete_run(run_id)
```

### 15.3 Arrêter et Redémarrer le Cluster

Listing 45 – Gestion du cluster Minikube

```
1 # Arrêter Minikube (preserve les donnees)
2 minikube stop
3
4 # Redémarrer
5 minikube start
6
7 # Supprimer complètement le cluster
8 minikube delete
```

```
9
10 # Recreer depuis zero
11 minikube start --cpus=4 --memory=6000 --driver=docker
```

## 15.4 Backup des Modèles

Listing 46 – Script de backup

```
1 from kubernetes import client, config
2 import json
3 import os
4 from datetime import datetime
5
6 config.load_kube_config()
7 v1 = client.CoreV1Api()
8
9 # Repertoire de backup
10 backup_dir = f"backups/{datetime.now().strftime('%Y%m%d_%H%M%S')}"
11 os.makedirs(backup_dir, exist_ok=True)
12
13 # Recuperer tous les ConfigMaps de modeles
14 configmaps = v1.list_namespaced_config_map("kubeflow")
15
16 for cm in configmaps.items:
17     if "model" in cm.metadata.name:
18         # Sauvegarder
19         filepath = os.path.join(backup_dir, f"{cm.metadata.name}.json")
20         with open(filepath, "w") as f:
21             json.dump(cm.data, f, indent=2)
22         print(f"Backup: {filepath}")
```

## 16 Conclusion

### 16.1 Récapitulatif

Ce guide complet a couvert :

1. **Installation** : Minikube, Docker, Kubeflow Pipelines, Katib
2. **Configuration** : Correction de MinIO, accès aux interfaces
3. **Développement** : Scripts d'entraînement, Dockerfile, pipelines
4. **Optimisation** : 3 algorithmes (Random, Bayesian, TPE)
5. **Comparaison** : Pipeline complet avec sélection automatique
6. **Sauvegarde** : Persistance dans Kubernetes ConfigMaps
7. **Monitoring** : Visualisation via Katib et Kubeflow UIs
8. **Maintenance** : Dépannage, nettoyage, bonnes pratiques

### 16.2 Avantages de cette Architecture

#### Succès

##### Points forts :

- **Reproductibilité** : Tout est défini en code (pipelines as code)
- **Scalabilité** : Prêt pour le cloud (AWS EKS, GKE, AKS)
- **Flexibilité** : Facile d'ajouter de nouveaux algorithmes
- **Traçabilité** : Historique complet dans Kubeflow
- **Automatisation** : Du preprocessing au déploiement

### 16.3 Prochaines Étapes

Pour aller plus loin :

1. **Production** : Déployer sur un cluster Kubernetes cloud
2. **CI/CD** : Intégrer avec Jenkins/GitLab CI
3. **Serving** : Ajouter KFServing pour le déploiement de modèles
4. **Monitoring** : Prometheus + Grafana pour le monitoring
5. **Feature Store** : Feast pour la gestion des features
6. **Drift Detection** : Surveiller la dérive des modèles

### 16.4 Ressources Supplémentaires

- Documentation Kubeflow : <https://www.kubeflow.org/docs/>
- Documentation Katib : <https://www.kubeflow.org/docs/components/katib/>
- Minikube Docs : <https://minikube.sigs.k8s.io/docs/>
- Kubernetes Docs : <https://kubernetes.io/docs/>
- KFP SDK : <https://kubeflow-pipelines.readthedocs.io/>

## 16.5 Support et Communauté

- Kubeflow Slack : <https://kubeflow.slack.com>
- GitHub Issues : <https://github.com/kubeflow/>
- Stack Overflow : Tag `kubeflow` ou `katib`
- Forum Kubeflow : <https://discuss.kubeflow.org/>

## Annexes

### Annexe A : Commandes Utiles

Listing 47 – Aide-mémoire des commandes

```

1 # === MINIKUBE ===
2 minikube start --cpus=4 --memory=6000 --driver=docker
3 minikube stop
4 minikube delete
5 minikube status
6 minikube dashboard
7 minikube image ls
8
9 # === KUBECTL ===
10 kubectl get pods -n kubeflow
11 kubectl get experiments -n kubeflow
12 kubectl get trials -n kubeflow
13 kubectl logs -n kubeflow <pod-name> -f
14 kubectl describe pod -n kubeflow <pod-name>
15 kubectl delete pod -n kubeflow <pod-name>
16 kubectl top nodes
17 kubectl top pods -n kubeflow
18
19 # === PORT-FORWARD ===
20 kubectl port-forward -n kubeflow svc/ml-pipeline-ui 8080:80
21 kubectl port-forward -n kubeflow svc/katib-ui 8081:80
22
23 # === DOCKER ===
24 docker build -t automl-training:v1.0 .
25 docker images
26 minikube image load automl-training:v1.0
27
28 # === PIPELINE ===
29 python pipelines/complete_pipeline.py

```

### Annexe B : Structure Complète du Projet

Plateforme\_autoML\_end\_to\_end/

README.md	# Documentation principale
USAGE.md	# Guide d'utilisation
requirements.txt	# Dépendances Python
setup.py	# Configuration du package
.gitignore	# Fichiers à ignorer par Git
config/	# Configuration
config.yaml	# Configuration principale
katib/	# Configurations Katib

```
README.md
hyperparameter-tuning/  # Expériences HP tuning
  bayesian-optimization.yaml
  tpe-optimization.yaml
  random-search.yaml
experiments/           # Exemples
  README.md

pipelines/              # Kubeflow Pipelines
  README.md
  kfp/
    __init__.py
    automl_pipeline.py  # Pipeline principal

src/                    # Code source
  training/             # Scripts d'entraînement
    __init__.py
    train.py           # Training standard
  models/               # Gestion des modèles
    __init__.py
    base_model.py      # Classe de base
    model_registry.py  # Registre des modèles
  utils/                # Utilitaires
    __init__.py
    config.py           # Gestion config
    logger.py           # Logging
  serving/              # Model serving
    __init__.py

scripts/                # Scripts utilitaires
  README.md
  submit_katib_experiment.py
  deploy_kfserving.py
  compile_pipeline.py

docker/                 # Dockerfiles
  Dockerfile.training
  Dockerfile.serving
  docker-compose.yml

notebooks/              # Notebooks Jupyter
  example_katib_experiment.ipynb

docs/                   # Documentation
  GETTING_STARTED.md
  ARCHITECTURE.md
  EXAMPLES.md
```



## Annexe C : Variables d'Environnement

Listing 48 – Fichier .env

```
1 # Kubernetes
2 NAMESPACE=kubeflow
3 KUBECONFIG=~/.kube/config
4
5 # Docker
6 DOCKER_REGISTRY=docker.io
7 IMAGE_NAME=automl-training
8 IMAGE_TAG=v1.0
9
10 # Kubeflow
11 KF_PIPELINES_ENDPOINT=http://localhost:8080
12 KATIB_UI_ENDPOINT=http://localhost:8081
13
14 # Training
15 DATASET_URL=https://example.com/data.csv
16 MAX_TRIALS=10
17 PARALLEL_TRIALS=2
18
19 # Monitoring
20 LOG_LEVEL=INFO
21 METRICS_PATH=/tmp/katib-metrics.log
```

## Annexe D : Exemple de Metadata

Listing 49 – Format des métadonnées du modèle

```
1 {
2   "model_id": "iris-mlp-2024-01-15-1234",
3   "version": "1.0.0",
4   "algorithm": "bayesian",
5   "framework": "scikit-learn",
6   "model_type": "MLPClassifier",
7   "dataset": {
8     "name": "iris",
9     "samples": 150,
10    "features": 4,
11    "classes": 3
12  },
13  "hyperparameters": {
14    "learning_rate": 0.0234,
15    "hidden_units": 96,
16    "optimizer": "adam",
17    "max_iter": 1000
18  },
19  "metrics": {
20    "train_accuracy": 0.9833,
21    "test_accuracy": 0.9667,
```

```
22     "loss": 0.0333,  
23     "f1_score": 0.9662  
24 },  
25 "training": {  
26     "trials_run": 8,  
27     "best_trial": 6,  
28     "duration_seconds": 240,  
29     "timestamp": "2024-01-15T14:30:45Z"  
30 },  
31 "environment": {  
32     "python_version": "3.9.18",  
33     "sklearn_version": "1.3.2",  
34     "kubernetes": "v1.28.0",  
35     "cluster": "minikube"  
36 }  
37 }
```