

Bootcamp: Analista de Machine Learning

Atividade Modular

Módulo: MLOps e Gerenciamento de Projetos em Machine Learning

Objetivos de Ensino

Exercitar os seguintes conceitos trabalhados no Módulo:

1. Ciclo de vida de modelos de Machine Learning
2. Treinamento e validação de modelos
3. Gerenciamento projetos com MFLOW

Enunciado

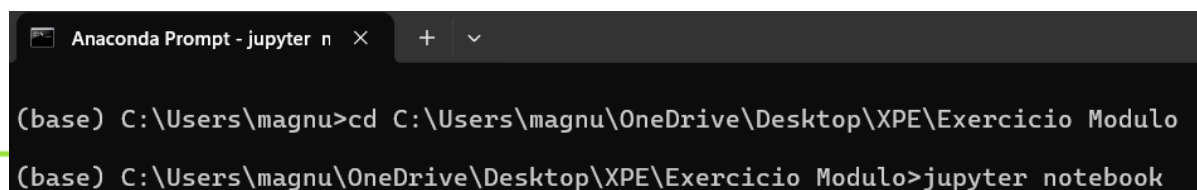
A atividade prática consiste em gerenciar o desenvolvimento de um algoritmo de machine learning com MLFLOW. Para isso, vamos criar pipeline de treinamento e validação, registrando todos os resultados dos experimentos no MLFLOW.

Atividades

Os alunos deverão desempenhar as seguintes atividades:

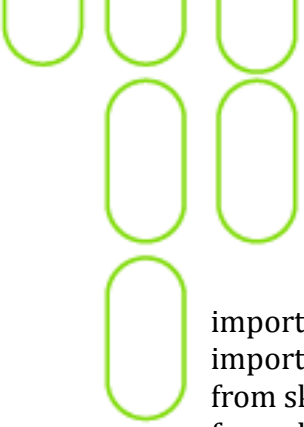
1. Abrir o notebook python

Entre onde foi instalado o python e dê o seguinte comando:



```
Anaconda Prompt - jupyter n  X + v
(base) C:\Users\magnu>cd C:\Users\magnu\OneDrive\Desktop\XPE\Exercicio Modulo
(base) C:\Users\magnu\OneDrive\Desktop\XPE\Exercicio Modulo>jupyter notebook
```

2. Após abrir o jupyter notebook, importe as seguintes bibliotecas



```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import (
    accuracy_score, recall_score, precision_score, f1_score,
    roc_auc_score, log_loss, confusion_matrix, ConfusionMatrixDisplay,
    RocCurveDisplay, roc_curve
)
import mlflow
import mlflow.sklearn
import matplotlib.pyplot as plt
```

Essas bibliotecas serão necessárias para realizamos etapas de: Coleta dos dados, Limpeza e manipulação dos dados, treinamento e validação do conjunto, além de realizar o deploy no MLFLOW3.

3. Importe do conjunto de dados

```
credito = pd.read_csv('Credit.csv')
```


O conjunto de dados consiste em dados bancários, simulando clientes que estão inadimplentes e clientes que têm um bom relacionamento com a instituição financeira


4. Tratamento e Limpeza do Conjunto de dados

O script abaixo é apresentado como transformar as colunas que estão do type object para tipo categoria e separando o conjunto de treino e teste.

```
for col in credito.columns:
    if credito[col].dtype == 'object':
        credito[col] = credito[col].astype('category').cat.codes

previsores = credito.iloc[:,0:20].values
classe = credito.iloc[:,20].values
```





```
X_treinamento, X_teste, y_treinamento, y_teste = train_test_split(previsores, classe,
                                                                    test_size=0.3, random_state=123)
```

5. Pipeline de Treinamento do Algoritmo junto com validação e registrando resultados no MFLOW

A função abaixo é um conjunto de treinamento do algoritmo utilizando um pipeline automatizado para treinar diversos algoritmos e registrar os resultados no MLFLOW. Para isso, foi utilizada a biblioteca do Sklearn e, para calibragem dos algoritmos, o GridSearch com cross validation

```
def treina_modelo(modelo, parametros):
    mlflow.set_experiment("Atividade_Modular")

    pipeline = Pipeline([
        ('clf', modelo)
    ])

    grid = GridSearchCV(pipeline, param_grid=parametros, cv=5, scoring='accuracy')

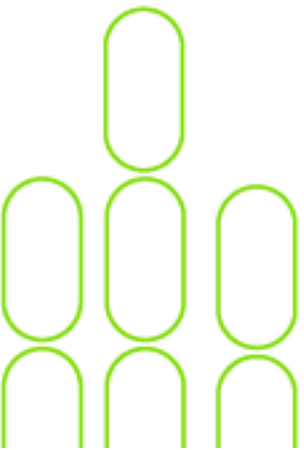
    with mlflow.start_run():
        grid.fit(X_treinamento, y_treinamento)

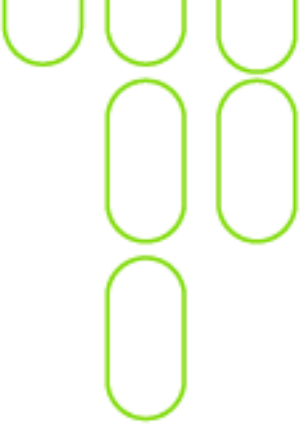
        best_model = grid.best_estimator_
        previsoes_train = best_model.predict(X_treinamento)
        previsoes_teste = best_model.predict(X_teste)

        # Registro de hiperparâmetros e parâmetros selecionados
        mlflow.log_params(grid.best_params_)

        # Métricas --- Treino
        acuracia_train = accuracy_score(y_treinamento, previsoes_train)
        recall_train = recall_score(y_treinamento, previsoes_train)
        precision_train = precision_score(y_treinamento, previsoes_train)
        f1_train = f1_score(y_treinamento, previsoes_train)
        auc_train = roc_auc_score(y_treinamento, previsoes_train)
        log_train = log_loss(y_treinamento, previsoes_train)

        # Registrar métricas
        mlflow.log_metric("acuracia_train", acuracia_train)
        mlflow.log_metric("recall_train", recall_train)
        mlflow.log_metric("precision_train", precision_train)
```





```
mlflow.log_metric("f1_train", f1_train)
mlflow.log_metric("auc_train", auc_train)
mlflow.log_metric("log_train", log_train)
```

```
# Matriz de Confusão
cm = confusion_matrix(y_treinamento, previsoes_train)
confusion_disp = ConfusionMatrixDisplay(confusion_matrix=cm)
confusion_disp.plot()
plt.savefig("confusion_matrix_train.png")
mlflow.log_artifact("confusion_matrix_train.png")
```

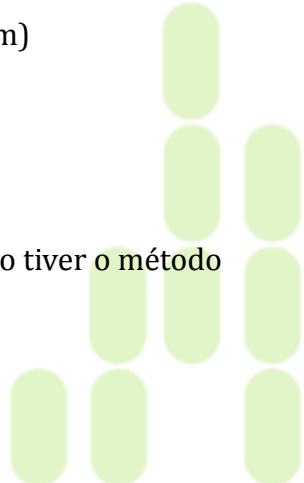
```
# Curva ROC
if hasattr(best_model, "predict_proba"): # Somente se o modelo tiver o método
`predict_proba`
    fpr, tpr, _ = roc_curve(y_treinamento,
best_model.predict_proba(X_treinamento)[: , 1])
    roc_disp = RocCurveDisplay(fpr=fpr, tpr=tpr)
    roc_disp.plot()
    plt.savefig("roc_curve_train.png")
    mlflow.log_artifact("roc_curve_train.png")
```


```
# Métricas --- Teste
acuracia_test = accuracy_score(y_teste, previsoes_teste)
recall_test = recall_score(y_teste, previsoes_teste)
precision_test = precision_score(y_teste, previsoes_teste)
f1_test = f1_score(y_teste, previsoes_teste)
auc_test = roc_auc_score(y_teste, previsoes_teste)
log_test = log_loss(y_teste, previsoes_teste)
```

```
# Registrar métricas
mlflow.log_metric("acuracia_test", acuracia_test)
mlflow.log_metric("recall_test", recall_test)
mlflow.log_metric("precision_test", precision_test)
mlflow.log_metric("f1_test", f1_test)
mlflow.log_metric("auc_test", auc_test)
mlflow.log_metric("log_test", log_test)
```

```
# Matriz de Confusão
cm = confusion_matrix(y_teste, previsoes_teste)
confusion_disp = ConfusionMatrixDisplay(confusion_matrix=cm)
confusion_disp.plot()
plt.savefig("confusion_matrix_test.png")
mlflow.log_artifact("confusion_matrix_test.png")
```

```
# Curva ROC
if hasattr(best_model, "predict_proba"): # Somente se o modelo tiver o método
`predict_proba`
```





```

fpr, tpr, _ = roc_curve(y_teste, best_model.predict_proba(X_teste)[:, 1])
roc_disp = RocCurveDisplay(fpr=fpr, tpr=tpr)
roc_disp.plot()
plt.savefig("roc_curve_test.png")
mlflow.log_artifact("roc_curve_test.png")

# Logar o modelo
mlflow.sklearn.log_model(best_model, "Modelo")

print("Modelo treinado: ", mlflow.active_run().info.run_uuid)

mlflow.end_run()

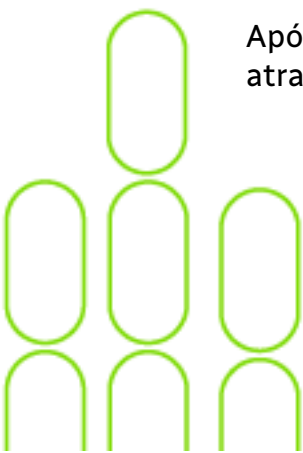
# Configuração de modelos e hiperparâmetros
modelos_e_parametros = {
    RandomForestClassifier(): {
        'clf_n_estimators': [50, 100, 500]
    },
    LogisticRegression(max_iter=500): {
        'clf_C': [0.1, 1, 10],
        'clf_penalty': ['l2']
    },
    KNeighborsClassifier(): {
        'clf_n_neighbors': [3, 5, 7],
        'clf_weights': ['uniform', 'distance']
    }
}

# Loop para treinar e registrar cada modelo com diferentes parâmetros
for modelo, parametros in modelos_e_parametros.items():
    print("Modelo:" , modelo)
    print("Parametros:" , parametros)
    treina_modelo(modelo, parametros)

```

6. Acessar MLFLOW

Após executar script da etapa superior, entrar, através do terminal, o MLFLOW, através do seguinte comando e clique no url.



```
Anaconda Prompt (anaconda) X + v

(base) C:\Users\lucia>mflow ui
'mflow' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

(base) C:\Users\lucia>cd C:\Users\lucia\OneDrive\Área de Trabalho\XPE\EXERCICIO MODULAR

(base) C:\Users\lucia\OneDrive\Área de Trabalho\XPE\EXERCICIO MODULAR>mlflow ui
INFO:waitress:erving on http://127.0.0.1:5000
```

7. Visualização dos Resultados

Após realizar a etapa acima, é possível visualizar todos os resultados do treinamento

The screenshot shows the mlflow 2.16.2 web interface. The top navigation bar includes the mlflow logo, version 2.16.2, and tabs for 'Experiments' and 'Models'. On the right, there are links for 'GitHub' and 'Docs'. The left sidebar shows a list of experiments under the 'Experiments' tab, with 'Atividade_Modular' selected. The main content area displays the 'Atividade_Modular' experiment details. It includes a search bar, a filter for 'State: Active', and a table of runs. The table has columns for 'Run Name', 'Created', 'Dataset', 'Duration', 'Source', and 'Model'. Three runs are listed: 'handsome-calf-704' (4.6s), 'unequaled-shoot-302' (10.7s), and 'righteous-shark-503' (21.7s).

Run Name	Created	Dataset	Duration	Source	Model
handsome-calf-704	17 minutes ago	-	4.6s	C:\Users...	
unequaled-shoot-302	17 minutes ago	-	10.7s	C:\Users...	
righteous-shark-503	18 minutes ago	-	21.7s	C:\Users...	