

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

Nicolas Marcos de Moraes Oliveira

MLFLOW Aplicado à Soluções de Machine Learning

Rio de Janeiro, Brasil

2022

Nicolas Marcos de Moraes Oliveira

MLFLOW Aplicado à Soluções de Machine Learning

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina como requisito parcial à obtenção do título de especialista.

Rio de Janeiro, Brasil

2022

SUMÁRIO

1. Introdução.....	4
1.1. Contextualização	4
1.2. O problema proposto	4
2. Coleta de Dados	6
3. Análise e Exploração dos Dados	9
4. Processamento/Tratamento de Dados	12
5. Criação de Modelos de Machine Learning	14
6. Apresentação dos Resultados	19
7. Links	22
REFERÊNCIAS.....	23

1. Introdução

1.1. Contextualização

Este estudo tem por objetivo apresentar o emprego da plataforma de MLFLOW em soluções corporativas de Machine Learning, provendo um ambiente escalável, compartilhado, monitorável e metrificável. O estudo foi feito utilizando a implementação da plataforma através de ambiente Databricks.

1.2. O problema proposto

Conforme afirmado por Géron (2019, p. 24), ao trabalhar com Machine Learning e partir do princípio que a principal tarefa consiste em selecionar o algoritmo de aprendizagem e o treinar baseado em dados, dois possíveis pontos de erro são algoritmos ruins e dados ruins.

Para evitar algoritmos ruins e buscar as melhores métricas aos seus respectivos problemas, profissionais da área de Machine Learning realizam testes dentre diferentes algoritmos, com diferentes hiperparâmetros e técnicas. Corroborado por Vanschoren e Blockeel (2009, p.1), experimentação é alma de Machine Learning. Durante o trabalho de exploração, preparação, treinamento e testes de modelos; diversos experimentos são feitos por profissionais da área com o intuito de achar a melhor combinação de fatores inerentes ao trabalho que permitam os melhores desempenhos.

Deste modo, percebem-se duas questões que se tornam cotidianas aos profissionais de aprendizagem de máquina: como realizar a comparação mensurada dentre diferentes experimentos; e como realizar seu *tracking* monitorando diferentes versões a ajustes, bem como recuperar o experimento cuja combinação culminou no melhor desempenho?

Ao pensar sobre soluções corporativas, há dois novos desafios que podem ser adicionados ao contexto: como permitir reprodutibilidade destes experimentos

em diferentes ambientes e como permitir o trabalho colaborativo entre integrantes da mesma equipe com o mesmo fim?

Diante de todos estes desafios, surge o MLFLOW como alternativa capaz de atender tais desafios. O MLFLOW é uma plataforma Open Source de gerenciamento do ciclo de vida de machine learning, escalável para Big Data com Apache Spark, que permite o trabalho através de diferentes linguagens como Python e R, em diferentes clouds e com trabalho colaborativo entre diferentes usuários. O MLFLOW conta com os seguintes componentes:

- **MLFLOW Tracking:** componente responsável por armazenar e permitir consulta de experimentos, com seus códigos, dados e resultados. É possível qual código foi executado, sob quais dados (pode-se usar diferentes algoritmos para diferentes partes dos dados) e quais resultados foram obtidos.
- **MLFLOW Projects:** componente que permite empacotar projetos e posteriormente os compartilhar e/ou versionar. Deste modo, uma vez que o projeto faça uso de determinadas dependências, bibliotecas e demais configurações de ambiente podem ser reproduzidas em outras plataformas.
- **MLFLOW Models:** componente responsável por permitir deploy de modelos de machine learning, bem como os servir para uso de predições, seja em tempo real através de API REST ou inferência batch em ambiente Apache Spark.
- **MLFLOW Registry:** componente que permite o registro e armazenagem de modelos, bem como recuperar e disponibilizar versões específicas, controlando transições de estados, anotações e demais. Atua como um repositório centralizado de modelos, APIs e UI.

Este estudo da plataforma MLFLOW foi desenvolvido a utilizando no ambiente Databricks, que é um ambiente de Delta Lake provendo em um mesmo ambiente a possibilidade de usar MLFLOW com Big Data.

Quanto ao projeto de Machine Learning que servirá para o estudo da plataforma, será utilizando um modelo de classificação empregando algoritmo de

random forest sobre base de dados fictícia para tentar prever a probabilidade de um funcionário deixar ou não a empresa.

2. Coleta de Dados

Os dados trabalhados neste projeto foram obtidos em 10/06/2022 através da plataforma Kaggle no link que segue: <https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset?resource=download>. Esta base contém um case dados fictícios sobre funcionários de uma empresa e tem por objetivo tentar prever a possibilidade de um funcionário deixar ou não a companhia, dadas determinadas características.

O dataset obtido foi disponibilizado no github do autor do estudo através do link https://raw.githubusercontent.com/nicolasmarcos/TCC_PucMinas/main/dataset_hr.csv e conta com as seguintes colunas:

COLUNA	DESCRIÇÃO	TIPO
Age	Idade do funcionário	Numérico discreto
Attrition	Se o funcionário deixou ou não a companhia: YES / NO	Categórico nominal
BusinessTravel	Frequência com a qual o funcionário viaja	Categórico nominal
DailyRate	Taxa diária do funcionário	Numérico discreto
Department	Departamento do funcionário	Categórico nominal
DistanceFromHome	Distância entre a moradia do funcionário e seu trabalho	Numérico discreto
Education	Nível educacional do funcionário, sendo: 1 'Below College'; 2 'College'; 3 'Bachelor'; 4 'Master'; 5 'Doctor'.	Numérico discreto (representando categórico ordinal)
EducationField	Área de estudo do	Categórico nominal

	funcionário.	
EmployeeCount	Valor dummy para contagem de registros	Numérico discreto
EmployeeNumber	Número identificador do funcionário	Numérico discreto
EnvironmentSatisfaction	Nível de satisfação do funcionário com ambiente de trabalho, sendo: 1 'Low'; 2 'Medium'; 3 'High'; 4 'Very High'.	Numérico discreto (representando categórico ordinal)
Gender	Gênero do funcionário	Categórico nominal
HourlyRate	Taxa hora do funcionário	Numérico discreto
JobInvolvement	Nível de envolvimento do funcionário com o trabalho, sendo: 1 'Low', 2 'Medium', 3 'High', 4 'Very High'.	Numérico discreto (representando categórico ordinal)
JobLevel	Nível do funcionário	Numérico discreto
JobRole	Cargo do funcionário	Categórico nominal
JobSatisfaction	Nível de satisfação do funcionário com o trabalho, sendo: 1 'Low', 2 'Medium', 3 'High', 4 'Very High'.	Numérico discreto (representando categórico ordinal)
MaritalStatus	Status civil do funcionário	Categórico nominal
MonthlyIncome	Rendimento mensal do funcionário	Numérico discreto
MonthlyRate	Taxa mensal do funcionário	Numérico discreto
NumCompaniesWorked	Quantidade de companhias onde o funcionário já trabalhou	Numérico discreto
Over18	Se “Y” referente à “yes”,	Categórico nominal

	indica que o funcionário é maior de 18 anos	
OverTime	Indica se o funcionário faz horas extras como "Yes" ou "No"	Categórico nominal
PercentSalaryHike	Evolução salarial percentual	Numérico discreto
PerformanceRating	Nível de performance do funcionário, sendo: 1 'Low', 2 'Good', 3 'Excellent', 4 'Outstanding'	Numérico discreto (representando categórico ordinal)
RelationshipSatisfaction	Nível de satisfação do funcionário em relacionamento, sendo: 1 'Low', 2 'Medium', 3 'High', 4 'Very High'.	Numérico discreto (representando categórico ordinal)
StandardHours	Quantidade de horas padrões do funcionário	Numérico discreto
StockOptionLevel	Nível de opção de participação em ações	Numérico discreto
TotalWorkingYears	Total de anos trabalhados	Numérico discreto
TrainingTimesLastYear	Veze em que o funcionário foi treinado no último ano	Numérico discreto
WorkLifeBalance	Nível de equilíbrio entre vida pessoal e trabalho, sendo: 1 'Bad', 2 'Good', 3 'Better', 4 'Best'.	Numérico discreto (representando categórico ordinal)
YearsAtCompany	Anos do funcionário na companhia	Numérico discreto
YearsInCurrentRole	Anos do funcionário no cargo atual	Numérico discreto
YearsSinceLastPromotion	Anos do funcionário desde	Numérico discreto

	sua última promoção	
YearsWithCurrManager	Anos do funcionário com gestor atual	Numérico discreto

Observação: O dataset não dispunha de metadados a respeito de todas as colunas. Portanto, as descrições citadas são a respeito do entendimento do autor sobre as colunas, corroborados por pesquisas em canais e fóruns sobre estudos realizados nos mesmos dados.

3. Análise e Exploração dos Dados

Uma vez de posse dos dados, foram feitas análises com o objetivo de identificar suas naturezas. Através dos comandos que seguem, foi possível inferir que:

```

Cód 9
2) Estudo da base e do problema
Nossa meta neste momento é conhecer a base e seus dados.

Cód 10
1 df.shape

Out[5]: (1470, 35)

Command took 0.09 seconds -- by nicolasmarcos.t@gmail.com at 16/09/2022 11:43:08 on default

```

Figura 1 - O dataset dispõe de 1470 linhas e 35 colunas

```

1 # Teremos amostra dos dados do início
2 df.head()

```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Gender	HourlyRate	JobInvolvement	JobLevel	JobRole	JobSatisfaction	Mar
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	2	Female	94	3	2	Sales Executive	4	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	3	Male	61	2	2	Research Scientist	2	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	4	Male	92	2	1	Laboratory Technician	3	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	4	Female	56	3	1	Research Scientist	3	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	1	Male	40	3	1	Laboratory Technician	2	

Figura 2 - É possível ter uma ideia inicial dos dados

```

1 # Informações do dataset
2 df.describe()

```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	MonthlyRate	NumCompaniesWorked	Perce
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156	2.729932	2.063946	2.728571	6502.931293	14313.103401	2.693197	
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428	0.711561	1.106940	1.102846	4707.956783	7117.786044	2.498009	
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000	1.000000	1.000000	1.000000	1009.000000	2094.000000	0.000000	
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000	2.000000	1.000000	2.000000	2911.000000	8047.000000	1.000000	
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000	3.000000	2.000000	3.000000	4919.000000	14235.500000	2.000000	
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000	3.000000	3.000000	4.000000	8379.000000	20461.500000	4.000000	
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000	4.000000	5.000000	4.000000	19999.000000	26999.000000	9.000000	

Figura 3 - Não há valores nulos e é possível analisar sua distribuição

```

1 df["Attrition"].value_counts()

Out[11]: No      1233
Yes      237
Name: Attrition, dtype: int64

Command took 0.09 seconds -- by nicolasmarcos.ti@g

cmd 17

1 df["BusinessTravel"].value_counts()

Out[12]: Travel_Rarely      1043
Travel_Frequently      277
Non-Travel      150
Name: BusinessTravel, dtype: int64

Command took 0.09 seconds -- by nicolasmarcos.ti@g

cmd 18

1 df["Department"].value_counts()

Out[13]: Research & Development      961
Sales      446
Human Resources      63
Name: Department, dtype: int64

Command took 0.09 seconds -- by nicolasmarcos.ti@g

cmd 19

1 df["EducationField"].value_counts()

Out[14]: Life Sciences      606
Medical      464
Marketing      159
Technical Degree      132
Other      82
Human Resources      27
Name: EducationField, dtype: int64

```

Figura 4 - Pode-se analisar ocorrências distintas para atributos categóricos.

```

1 df["Gender"].value_counts()

Out[15]: Male      882
        Female    588
        Name: Gender, dtype: int64

Command took 0.09 seconds -- by nicolasmarcos.ti@gmail.com at 10/09/2022 11:43:07 on default

```

Cmd 21

```

1 df["JobRole"].value_counts()

Out[16]: Sales Executive      326
        Research Scientist    292
        Laboratory Technician  259
        Manufacturing Director 145
        Healthcare Representative 131
        Manager               102
        Sales Representative    83
        Research Director       80
        Human Resources         52
        Name: JobRole, dtype: int64

Command took 0.09 seconds -- by nicolasmarcos.ti@gmail.com at 10/09/2022 11:43:07 on default

```

Cmd 22

```

1 df["MaritalStatus"].value_counts()

Out[17]: Married      673
        Single      470
        Divorced     327
        Name: MaritalStatus, dtype: int64

Command took 0.09 seconds -- by nicolasmarcos.ti@gmail.com at 10/09/2022 11:43:07 on default

```

Cmd 23

```

1 # Este atributo indica se a pessoa tem mais de 18 anos. Como todos possuem, não o utilizaremos
2 df["Over18"].value_counts()

Out[18]: Y      1470
        Name: Over18, dtype: int64

Command took 0.09 seconds -- by nicolasmarcos.ti@gmail.com at 10/09/2022 11:43:09 on default

```

Cmd 24

```

1 # Este atributo indica se a pessoa trabalha horas extras
2 df["OverTime"].value_counts()

Out[19]: No      1054
        Yes      416
        Name: OverTime, dtype: int64

```

Figura 5 - Pode-se analisar ocorrências distintas para demais atributos categóricos.

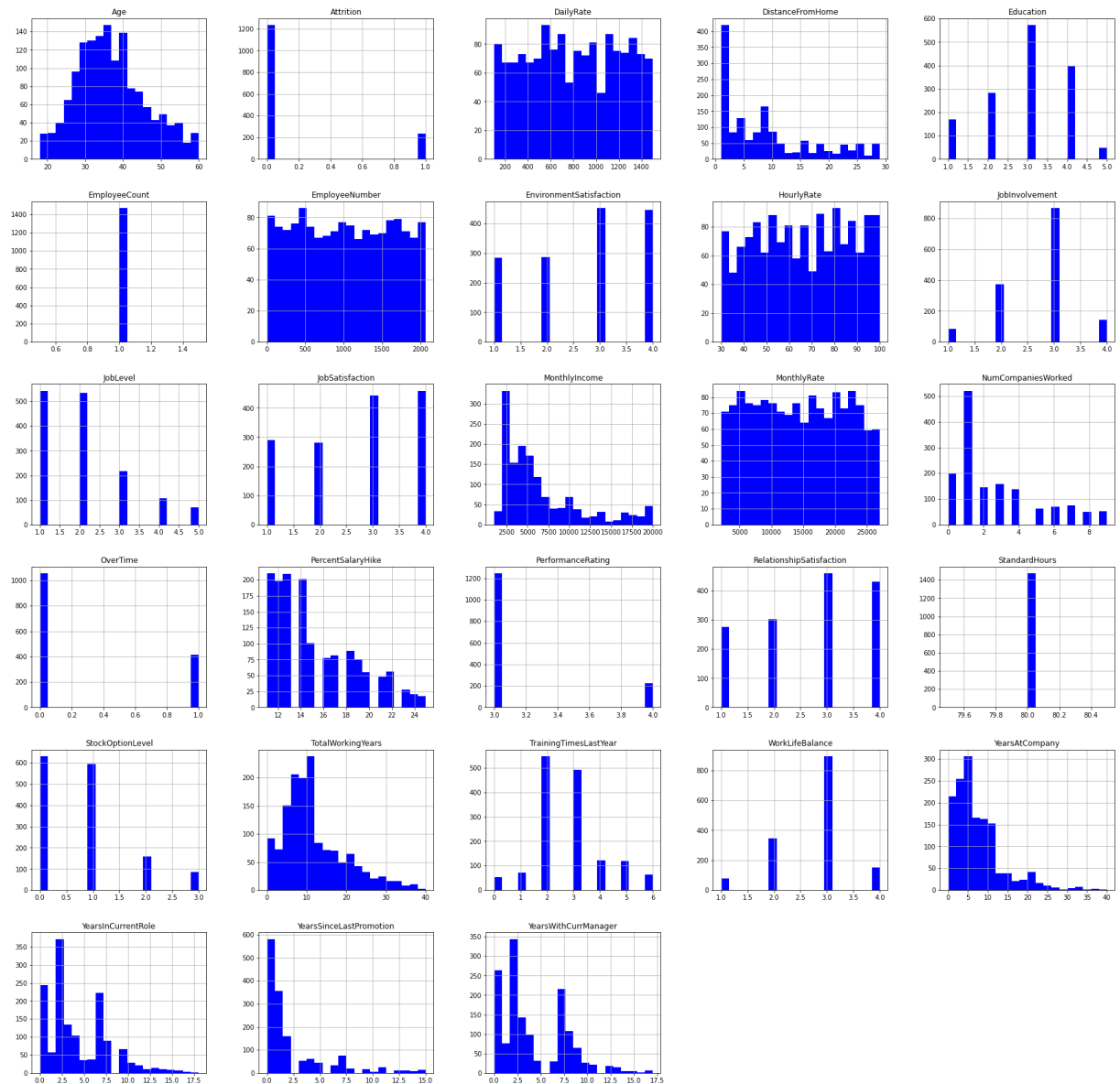


Figura 6 - Pode-se analisar distribuição de features numéricas

4. Processamento/Tratamento de Dados

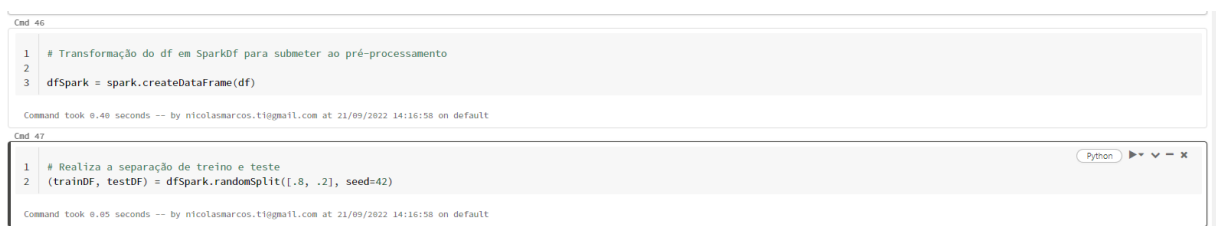
Após as análises realizadas na etapa de exploração de dados, identificou-se que:

- Todos os funcionários possuem mais de 18 anos. Portanto, a coluna “Over18” não se faz necessária para o treinamento do modelo e foi retirada do dataset. O mesmo raciocínio foi empregue para as colunas “StandardHours” e “EmployeeCount”, visto que todos os registros tinham o mesmo valor.

- A coluna “EmployeeNumber” foi retirada por ser apenas um ID do funcionário, sem agregar significado ao modelo.
- Uma vez que a classe “Attrition” a ser prevista pelo modelo é binária, seus valores foram alterados para 1 quando “Yes” e 0 quando “No”. O mesmo foi empregue para a feature “OverTime”.
- Não foram encontrados outliers ou dados faltantes que requeressem tratamentos.

Realizados os ajustes, é necessário preparar o dataset para o processo de Machine Learning. Dado que o algoritmo a ser utilizado será um algoritmo de Random Forest, não se faz necessário etapa de normalização dos dados. Todavia, será necessário encoding de dados para tratar os dados categóricos. O processo será realizado com o uso de OneHotEncoder().

Uma vez que o objetivo deste estudo consiste em explorar o potencial do MLFLOW para soluções corporativas e reproduzíveis, todas as etapas de pré-processamento dos dados foram atribuídas à um pipeline de processamento, de modo que mediante entrada de novos dados seja possível reproduzir as etapas com o mínimo de esforço. Para tal, foi convertido o dataframe pandas que estava sendo utilizado até então para análises, feita a divisão em treinamento e teste nos percentuais de 80% e 20% do dataset, respectivamente. Posteriormente, o dataset de treino será submetido às etapas do pipeline.



```
Cmd 46
1 # Transformação do df em SparkDf para submeter ao pré-processamento
2
3 dfSpark = spark.createDataFrame(df)

Command took 0.40 seconds -- by nicolasmarcos.tl@gmail.com at 21/09/2022 14:16:58 on default

Cmd 47
1 # Realiza a separação de treino e teste
2 (trainDF, testDF) = dfSpark.randomSplit([.8, .2], seed=42)

Python
Command took 0.05 seconds -- by nicolasmarcos.tl@gmail.com at 21/09/2022 14:16:58 on default
```

Figura 7 - Separação dos dados de treino e teste

A primeira etapa do pipeline será a utilização do StringIndexer() que será responsável por indexar a lista de colunas categóricas antes de submetê-las ao OneHotEncoder(), que é executado na sequência e gera uma lista de colunas encodadas.

A terceira etapa do processo ocorre com o `VectorAssembler()` que recebe as colunas categóricas encodadas e as une com as colunas numéricas, gerando como saída uma coluna de features que serão usadas pelo modelo de aprendizado.

A etapa seguinte do pipeline é o modelo de `RandomForestClassifier`. Nesta etapa há a instanciação do modelo e `MulticlassClassificationEvaluator` que será a estratégia utilizada para análise do sucesso do modelo. Todavia, nesta etapa não são passados hiperparâmetros ao modelo. Isto ocorrerá na hiperotimização que será feita através do MLFLOW, usando uma função objetiva de otimização de resultado.

```
# Define pipeline e métrica de avaliação

# Esta função gera uma lambda que capta o nome dos campos onde o tipo do dado for string
categoricalCols = [field for (field,dataType) in trainDF.dtypes if ((dataType == "string"))]
# Gera um array com o nome das colunas com sufixo Index
indexOutputCols = [x + "Index" for x in categoricalCols]
# Instancia o stringIndexer e caso hajam valores inválidos, pulará
stringIndexer = StringIndexer(inputCols=categoricalCols, outputCols=indexOutputCols, handleInvalid="skip")

# Gera um array com o nome das colunas com sufixo OHE
oheOutputCols = [x + "OHE" for x in categoricalCols]
# Instância o OHE
oheEncoder = OneHotEncoder(inputCols=indexOutputCols, outputCols=oheOutputCols)

# Pegará as colunas numéricas que não são a classe para dps combinarmos os inputs do assembler
numericCols = [field for (field, dataType) in trainDF.dtypes if ((dataType != "string") & (field != "Attrition"))]
# Unifica colunas categóricas e numéricas
assemblerInputs = oheOutputCols + numericCols
# Gera coluna de features para o algoritmo
vecAssembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")

# Instancia o modelo informando a classe
rf = RandomForestClassifier(labelCol="Attrition", seed=42)

stages = [stringIndexer, oheEncoder, vecAssembler, rf]
pipeline = Pipeline(stages=stages)

evaluator = MulticlassClassificationEvaluator(labelCol="Attrition", predictionCol="prediction", metricName="accuracy")
```

Figura 8 - Definição do Pipeline

5. Criação de Modelos de Machine Learning

Uma vez definido o pipeline, a próxima etapa consiste em definir a objective function, que será a função utilizada dentro do MLFLOW na busca dos melhores hiperparâmetros do modelo. Nesta etapa ocorrem os seguintes passos:

- Definem-se os parâmetros a serem otimizados no modelo;

- Instancia-se o grid com a classe ParamGridBuilder() que será usada na otimização do modelo;
- Instancia-se o CrossValidator() que usará como estimador (ou seja, dataset de entrada) o resultado do pipeline gerado nos passos anteriores, com o grid de parâmetros a serem otimizados, a quantidade desejada de folds e objeto evaluator que será usado no treinamento.
- Geração do modelo após treinamento usando dataset de treino;
- Registro da acurácia. Perceba que a acurácia é negativada, pois será usado algoritmo fmin() para a otimização e este algoritmo busca otimizar o menor valor de perda encontrado. Ou seja, negativando a acurácia, será buscado o menor valor encontrado que por sua vez corresponderá a melhor acurácia para o modelo.

```

1 def objective_function(params):
2
3     # Instancia como array os parâmetros a serem tunados e coloca em variável de mesmo nome
4     max_depth = params["max_depth"]
5     num_trees = params["num_trees"]
6     max_bins = params["max_bins"]
7
8     # Instancia o grid com os parâmetros e valores
9     grid = (ParamGridBuilder()
10             .addGrid(rf.maxDepth, [max_depth])
11             .addGrid(rf.numTrees, [num_trees])
12             .addGrid(rf.maxBins, [max_bins])
13             .build())
14
15     # Instancia o Cross Validator COM O PIPELINE, mas o cross validator está recebendo o grid instanciado acima
16     # E o grid instanciado acima está recebendo uma lista de valores
17     cv = CrossValidator(estimator=pipeline, estimatorParamMaps=grid, parallelism=4, evaluator=evaluator, numFolds=3)
18     cvModel = cv.fit(trainDF)
19
20
21     accuracy = cvModel.avgMetrics[0]
22
23     return {"loss": -accuracy, "status": STATUS_OK}

```

Figura 9 - Geração Objective Function

Após a geração da Objective Function, define-se o espaço de buscas que o algoritmo de otimização usará para cada um dos hiperparâmetros. Buscando evitar overfitting de dados, foi considerada poda (max_depth) de até 10 níveis. A quantidade de árvores no modelo foi testada de 10 a 300 árvores. Para o parâmetro max_bins que corresponde a quantidade de subdivisões e compartimentos do modelo, segundo a documentação oficial da biblioteca:

“(...) Número máximo de compartimentos para discretizar recursos contínuos. Deve ser ≥ 2 e \geq número de categorias para qualquer recurso categórico. (...)”.

Traduzido de

<https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.ml.classification.RandomForestClassifier.html#pyspark.ml.classification.RandomForestClassifier.maxBins>. Acessado em: 24 de set. de 2022.

Diante deste fato, gerou-se mecanismo capaz de identificar a quantidade de registros distintos dentro os atributos categóricos. Para testes e otimização do parâmetro, utilizou-se espaço de busca de hiperparâmetros entre este limiar mínimo e até 3 vezes a quantidade de valores categóricos.

```
1 valoresDistintosCategoricos = 0
2
3 for col in categoricalCols:
4     valoresDistintosCategoricos += df[col].value_counts().shape[0]
5
6 valoresDistintosCategoricos
```

Out[72]: 26

Figura 10 - Valores distintos de features categóricas

```
# Gera-se números aleatórios baseado em um intervalo para cada parâmetro.
search_space = {
    "max_depth": hp.randint("max_depth", 2, 10),
    "num_trees": hp.randint("num_trees", 10, 300),
    "max_bins": hp.randint("max_bins", valoresDistintosCategoricos, valoresDistintosCategoricos*3)
}
```

Figura 11 - Espaço de busca dos hiperparâmetros

A partir deste momento será instanciado o experimento MLFLOW. Uma instancia sem um nome especificado no ambiente Databricks criará por padrão um experimento com o nome do notebook que o executa.

Na instância, define a quantidade de tentativas e o objeto Trials() que serão usados pela função fmin() para encontrar o melhor hiperparâmetro do modelo. Por sua vez, a função fmin() recebe o espaço de busca definido anteriormente; o algoritmo de busca TPE para otimização dos parâmetros e o parâmetro rstate para permitir reprodutibilidade do código. Esta função será responsável por encontrar os melhores hiperparâmetros do modelo.

Na sequência são alimentadas variáveis com estes hiperparâmetros. Em seguida estes melhores parâmetros encontrados são definidos na instância do modelo de Random Forest e ocorre o treinamento do pipeline com os dados de treinamento, usando os melhores parâmetros encontrados.

Por fim, valida-se o modelo o aplicando aos dados que foram separados para teste e há verificação da acurácia. Ocorre então o registro no MLFLOW dos parâmetros, métricas e o modelo.

```

1  # Instancia Execução do MLFLOW
2  with mlflow.start_run():
3      num_evals = 50 # Indica o número de vezes que ele tentará sozinho achar o melhor hiperparâmetro nos intervalos especificados
4      trials = Trials() # Instancia objetos Trials()
5      best_hyperparam = fmin(fn=objective_function, # fmin com a função objetiva que criamos
6                             space=search_space, # no space passamos o space de amostragem de cada parâmetro
7                             algo=tpe.suggest, # No algo passamos o algoritmo da estratégia de TPE para otimização dos parâmetros
8                             max_evals=num_evals, # No max_eval a quantidade de avaliações que serão realizadas
9                             trials=trials, # No trials o objeto Trials das tentativas
10                             rstate=np.random.default_rng(42)
11                             )
12
13     # Recebe a lista dos hiperparâmetros
14     best_max_depth = best_hyperparam["max_depth"]
15     best_num_trees = best_hyperparam["num_trees"]
16     best_max_bins = best_hyperparam["max_bins"]
17
18     # Encontrará o melhor hiperparâmetro e o setará para o rf
19     rf.setMaxDepth(best_max_depth)
20     rf.setNumTrees(best_num_trees)
21     rf.setMaxBins(best_max_bins)
22
23     # Treina o pipeline com todo o conjunto de dados de treinamento, já utilizando os melhores hiperparâmetros encontrados.
24     pipelineModel = pipeline.fit(trainDF)
25
26
27     # Testará os dados e fará evaluate
28     predDF = pipelineModel.transform(testDF)
29     acc = evaluator.evaluate(predDF)
30
31     # Loga os parâmetros encontrados, métrica e modelo
32     mlflow.log_param("max_depth", best_max_depth)
33     mlflow.log_param("numTrees", best_num_trees)
34     mlflow.log_param("max_bins", best_max_bins)
35     mlflow.log_metric("accuracy", acc)
36     mlflow.spark.log_model(pipelineModel, "model")
37

```

Figura 12 - Execução MLFLOW

A partir deste momento, o modelo já está foi registrado e é possível o importar em outros projetos, bem como acompanhar o tracking de seus experimentos. Por fim, é possível também salvar o modelo em um diretório e o importar para realização de previsões.

Como uma das principais funcionalidades do MLFLOW é permitir a comparação de modelos, ajustaremos o espaço de pesquisa dos hiperparâmetros e executaremos novamente o modelo a fim de checar se foram encontrados novos hiperparâmetros melhores. Estes resultados poderão ser acompanhados na UI do MLFLOW. Perceba que foi obtida uma melhor acurácia.

```
# Gera-se números aleatórios baseado em um intervalo para cada parâmetro.
search_space = {
    "max_depth": hp.randint("max_depth", 2, 20),
    "num_trees": hp.randint("num_trees", 10, 1000),
    "max_bins": hp.randint("max_bins", valoresDistintosCategoricos, valoresDistintosCategoricos*5)
}

wand took 0.09 seconds -- by nicolasmarcos.ti@gmail.com at 25/09/2022 21:32:14 on default
7

# Instancia Execução do MLFLOW
with mlflow.start_run():
    num_evals = 10 # Indica o número de vezes que ele tentará sozinho achar o melhor hiperparâmetro nos intervalos especificados
    trials = Trials() # Instancia objetos Trials()
    best_hyperparam = fmin(fn=objective_function, # fmin com a função objetiva que criamos
                           space=search_space, # no space passamos o space de amostragem de cada parâmetro
                           algo=tpe.suggest, # No algo passamos o algoritmo da estratégia de TPE para otimização dos parâmetros
                           max_evals=num_evals, # No max_eval a quantidade de avaliações que serão realizadas
                           trials=trials, # No trials o objeto Trials das tentativas
                           rstate=np.random.default_rng(42)
                           )

    # Recebe a lista dos hiperparâmetros
    best_max_depth = best_hyperparam["max_depth"]
    best_num_trees = best_hyperparam["num_trees"]
    best_max_bins = best_hyperparam["max_bins"]

    # Encontrará o melhor hiperparâmetro e o setará para o rf
    rf.setMaxDepth(best_max_depth)
    rf.setNumTrees(best_num_trees)
    rf.setMaxBins(best_max_bins)

    # Treina o pipeline com todo o conjunto de dados de treinamento, já utilizando os melhores hiperparâmetros encontrados.
    pipelineModel = pipeline.fit(trainDF)

    # Testará os dados e fará evaluate
    predDF = pipelineModel.transform(testDF)
    acc = evaluator.evaluate(predDF)

    # Loga os parâmetros encontrados, métrica e modelo
    mlflow.log_param("max_depth", best_max_depth)
    mlflow.log_param("numTrees", best_num_trees)
    mlflow.log_param("max_bins", best_max_bins)
    mlflow.log_metric("accuracy", acc)
    mlflow.spark.log_model(pipelineModel, "model")
```

Figura 13 - Segunda tentativa de hiperotimização

Refresh

Compare

Delete

Download CSV

Created

All time

Columns

Only show differences

metrics.rmsc < 1 ans

Showing 23 matching runs

								Metrics	Parameters		
<input type="checkbox"/>	Created	Duration	Run Name	User	Source	Version	Models	accuracy	max_bins	max_depth	numTrees
<input type="checkbox"/>	2 hours ago	2.1h	-	nicolasmarc...	TCC Final	-	spark	0.864	65	15	736
<input type="checkbox"/>	3 hours ago	1.6h	-	nicolasmarc...	TCC Final	-	spark	0.861	37	7	102
<input type="checkbox"/>	15 hours ago	-	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	16 hours ago	1.6h	-	nicolasmarc...	TCC Final	-	spark	0.861	37	7	102
<input type="checkbox"/>	1 day ago	-	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	1 day ago	1.5h	-	nicolasmarc...	TCC Final	-	spark	0.861	37	7	102
<input type="checkbox"/>	1 day ago	1.5h	-	nicolasmarc...	TCC Final	-	-	0.861	37	7	102
<input type="checkbox"/>	1 day ago	1.5h	-	nicolasmarc...	TCC Final	-	-	0.861	37	7	102
<input type="checkbox"/>	1 day ago	4.3min	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	1 day ago	1.5h	-	nicolasmarc...	TCC Final	-	-	0.871	-	8	98
<input type="checkbox"/>	4 days ago	1.5h	-	nicolasmarc...	TCC Final	-	-	0.868	-	9	203
<input type="checkbox"/>	4 days ago	1.4h	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	4 days ago	1.5h	-	nicolasmarc...	TCC Final	-	-	0.573	-	9	11
<input type="checkbox"/>	4 days ago	349ms	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	4 days ago	340ms	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	4 days ago	327ms	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	4 days ago	1.4s	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	4 days ago	1.3s	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	4 days ago	1.4s	-	nicolasmarc...	TCC Final	-	-	-	-	-	-
<input type="checkbox"/>	4 days ago	1.4s	-	nicolasmarc...	TCC Final	-	-	-	-	-	-

Columns

Only show differences

Q metrics.rmse < 1 and params.model = "tree"

Search

Filter

Clear

Showing 22 matching runs

								Metrics	Parameters		
	Created	Duration	Run Name	User	Source	Version	Models	accuracy	max_bins	max_depth	numTrees
	1 minute ago	-	-	nic...	TCC Final	-	-	-	-	-	-
	11 hours ago	-	-	nic...	TCC Final	-	-	-	-	-	-
	12 hours ago	1.6h	-	nic...	TCC Final	-	spark	0.861	37	7	102
	1 day ago	-	-	nic...	TCC Final	-	-	-	-	-	-
	1 day ago	1.5h	-	nic...	TCC Final	-	spark	0.861	37	7	102
	1 day ago	1.5h	-	nic...	TCC Final	-	-	0.861	37	7	102
	1 day ago	1.5h	-	nic...	TCC Final	-	-	0.861	37	7	102
	1 day ago	4.3min	-	nic...	TCC Final	-	-	-	-	-	-
	1 day ago	1.5h	-	nic...	TCC Final	-	-	0.871	-	8	98
	4 days ago	1.5h	-	nic...	TCC Final	-	-	0.868	-	9	203
	4 days ago	1.4h	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	1.5h	-	nic...	TCC Final	-	-	0.573	-	9	11
	4 days ago	349ms	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	340ms	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	327ms	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	1.4s	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	1.3s	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	1.4s	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	1.4s	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	1.6s	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	354ms	-	nic...	TCC Final	-	-	-	-	-	-
	4 days ago	0.5s	-	nic...	TCC Final	-	-	-	-	-	-

Load more

Figura 16 - Lista de Experimentos MLFLOW

Run 481cda3c7ba24da3b7f23b50f5c755f3

Provide Feedback

Reproduce Run

Run ID: 481cda3c7ba24da3b7f23b50f5c755f3

Date: 2022-09-25 09:01:52

Source: TCC Final

User: nicolasmarcos.ti@gmail.com

Duration: 1.6h

Status: FINISHED

Lifecycle Stage: active

Description

Edit

Parameters (3)

Name	Value
max_bins	37
max_depth	7
numTrees	102

Metrics (1)

Name	Value
accuracy	0.861

Tags

Artifacts

Figura 17 - Métricas e Parâmetros de Run Específica

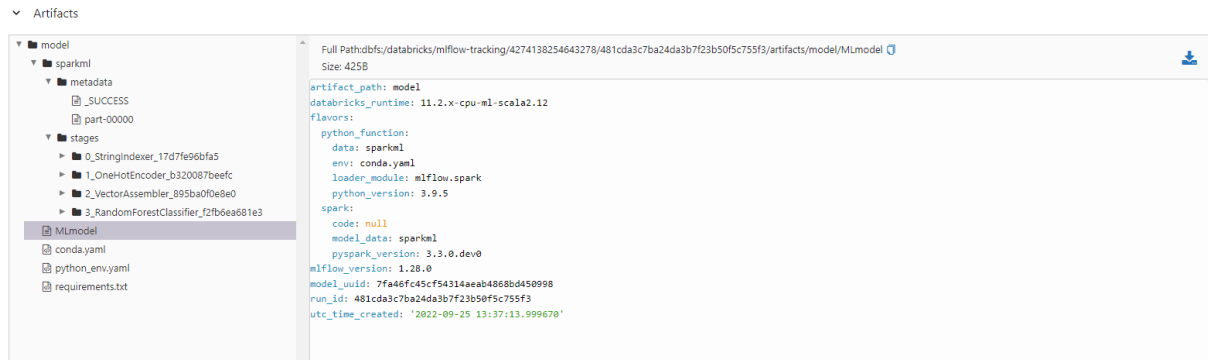


Figura 18 - Artefatos do Modelo em Run Específica

Através da API provida pelo `MlflowClient()` é possível interagir com o MLFLOW. Os comandos abaixo permitem listar os experimentos (os mesmos acima listados via UI) e obtermos informações sobre estes.

```

cmd 1
1 from mlflow.tracking import MlflowClient
2
3 mlflowClient = MlflowClient()

Command took 0.08 seconds -- by nicolasmarcos.ti@gmail.com at 25/09/2022 21:39:05 on default

cmd 2
1 # Lista os experimentos logados
2
3 mlflowClient.list_experiments()

Out[5]: [(<Experiment: artifact_location='dbfs:/databricks/mlflow-tracking/2637492296523236', experiment_id='2637492296523236', lifecycle_stage='active', name='/Users/nicolasmarcos.ti@gmail.com/Ibme
c Barra/ML Regressão Linear Databricks', tags={'mlflow.experiment.sourceName': '/Users/nicolasmarcos.ti@gmail.com/Ibme
c Barra/ML Regressão Linear Databricks',
'mlflow.experimentType': 'NOTEBOOK',
'mlflow.ownerEmail': 'nicolasmarcos.ti@gmail.com',
'mlflow.ownerId': '6594287489259527'}>),
(<Experiment: artifact_location='dbfs:/databricks/mlflow-tracking/3276328354489950', experiment_id='3276328354489950', lifecycle_stage='active', name='/Users/nicolasmarcos.ti@gmail.com/ALMWD-1.0.0/
03 Model Generalization', tags={'mlflow.experiment.sourceName': '/Users/nicolasmarcos.ti@gmail.com/ALMWD-1.0.0/03
Model Generalization',
'mlflow.experimentType': 'NOTEBOOK',
'mlflow.ownerEmail': 'nicolasmarcos.ti@gmail.com',
'mlflow.ownerId': '6594287489259527'}>),
(<Experiment: artifact_location='dbfs:/databricks/mlflow-tracking/3276328354489122', experiment_id='3276328354489122', lifecycle_stage='active', name='/Users/nicolasmarcos.ti@gmail.com/ALMWD-1.0.0/
01 Linear Regression 1', tags={'mlflow.experiment.sourceName': '/Users/nicolasmarcos.ti@gmail.com/ALMWD-1.0.0/01
Linear Regression 1',
'mlflow.experimentType': 'NOTEBOOK',
'mlflow.ownerEmail': 'nicolasmarcos.ti@gmail.com',
'mlflow.ownerId': '6594287489259527'}>),
(<Experiment: artifact_location='dbfs:/databricks/mlflow-tracking/3276328354489154', experiment_id='3276328354489154', lifecycle_stage='active', name='/Users/nicolasmarcos.ti@gmail.com/ALMWD-1.0.0/
02 Linear Regression 2', tags={'mlflow.experiment.sourceName': '/Users/nicolasmarcos.ti@gmail.com/ALMWD-1.0.0/02
Linear Regression 2',
'mlflow.experimentType': 'NOTEBOOK',
'mlflow.ownerEmail': 'nicolasmarcos.ti@gmail.com',
'mlflow.ownerId': '6594287489259527'}>)]

Command took 0.56 seconds -- by nicolasmarcos.ti@gmail.com at 25/09/2022 21:39:06 on default

cmd 3
1 experiment_id = "4274138254643278"

```

Figura 19 - Interação MlflowClient

1 import mlflow

Command took 0.12 seconds -- by nicolasmarcos.t@gmail.com at 25/09/2022 21:39:13 on default

id 5

1 # Obtém a run com a métrica mais alta
2 runs = mlflowClient.search_runs(experiment_id, order_by=["metrics.accuracy desc"], max_results=1)
3 # Apresenta a métrica da run
4 runs[0].data.metrics

Out[8]: {'accuracy': 0.8711864406779661}

Command took 0.38 seconds -- by nicolasmarcos.t@gmail.com at 25/09/2022 21:39:13 on default

id 6

1 # Permite retornar um DF sob experiment_id específicos
2 runs_df = mlflow.search_runs(experiment_id)
3 runs_df.display()

	run_id	experiment_id	status	artifact_uri	start_time	end_time	me
1	c89e0ae3161485082f0141c7566e8ab	4274138254643278	RUNNING	dbfs:/databricks/mlflow-tracking/4274138254643278/c89e0ae3161485082f0141c7566e8ab/artifacts	2022-09-25T13:38:14.963+0000	null	nul
2	481cda3c7ba24da3b7f23b50f5c755f3	4274138254643278	FINISHED	dbfs:/databricks/mlflow-tracking/4274138254643278/481cda3c7ba24da3b7f23b50f5c755f3/artifacts	2022-09-25T12:01:52.255+0000	2022-09-25T13:37:43.872+0000	0.8
3	f29aa82136f04cc0bf65d0b9663de33c	4274138254643278	RUNNING	dbfs:/databricks/mlflow-tracking/4274138254643278/f29aa82136f04cc0bf65d0b9663de33c/artifacts	2022-09-24T21:52:19.764+0000	null	nul
4	09174f1ec17c4f8d91e9c6ee7ab0d5f4	4274138254643278	FINISHED	dbfs:/databricks/mlflow-tracking/4274138254643278/09174f1ec17c4f8d91e9c6ee7ab0d5f4/artifacts	2022-09-24T20:21:09.633+0000	2022-09-24T21:52:19.219+0000	0.8
5	d77c31dc3c76438681fea2f2fb443632	4274138254643278	FINISHED	dbfs:/databricks/mlflow-tracking/4274138254643278/d77c31dc3c76438681fea2f2fb443632/artifacts	2022-09-24T18:43:04.373+0000	2022-09-24T20:13:10.361+0000	0.8

Showing all 21 rows.

Showing all 21 rows.

Figura 20 - Interação API Mlflow

7. Links

Repositório do projeto: https://github.com/nicolasmarcos/TCC_PucMinas

Fonte original dos dados:

<https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset?resource=download>

REFERÊNCIAS

GÉRON, Aurélien. **Hands-on Machine Learning with Scikit-learning, Keras & Tensorflow**. Estados Unidos da América, O'REILLY, 2019.

VANSCHOREN, JOAQUIN; BLOCKEEL, HENDRIK. **A Community-Based Platform for Machine Learning Experimentation**. Leuven, Leuven, Bélgica: Computer Science Dept., K.U, 2009.

Apache Spark, Acessado em: 24 de set. de 2022. Link: <https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.ml.classification.RandomForestClassifier.html#pyspark.ml.classification.RandomForestClassifier.maxBins>