

# Trabalho Prático 2

## Regressão Logística

**CCF 726**

Nome: Matheus Freitas Martins

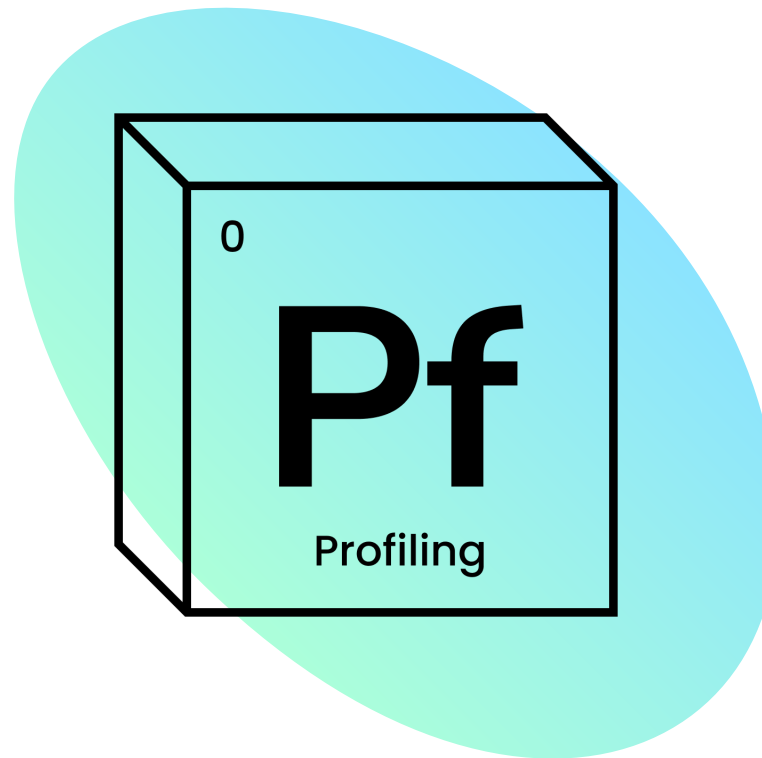
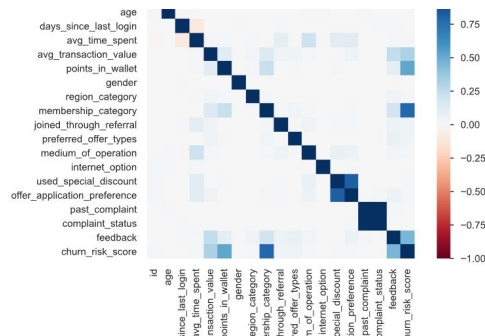
Matrícula: ES111281

Inicialmente, foi idealizado visualizar os dados de forma simples

```
df.isnull().sum()
df.info()
df.describe()
df.head()
```

Porém, foi visualizado que esse dataframe é um dataframe problemático, porque vários atributos demonstraram ter inserções não coerentes. Assim, foi escolhido utilizar **ydata-profiling** para visualizar e lidar melhor com as correlações, dados faltantes, nulos, negativos, inconsistentes, etc.

Overview	Alerts	Reproduction
<b>Alerts</b>		
security_no has a high cardinality: 30880 distinct values	High cardinality	
posting_date has a high cardinality: 1086 distinct values	High cardinality	
referrer_id has a high cardinality: 13359 distinct values	High cardinality	
last_visit_time has a high cardinality: 30101 distinct values	High cardinality	
avg_frequency_login_days has a high cardinality: 1654 distinct values	High cardinality	
points_in_wallet is highly overall correlated with churn_risk_score	High correlation	
membership_category is highly overall correlated with churn_risk_score	High correlation	
used_special_discount is highly overall correlated with offer_application_preference	High correlation	
offer_application_preference is highly overall correlated with used_special_discount	High correlation	
past_complaint is highly overall correlated with complaint_status	High correlation	
complaint_status is highly overall correlated with past_complaint	High correlation	
churn_risk_score is highly overall correlated with points_in_wallet and 1 other fields	High correlation	
avg_frequency_login_days is highly imbalanced (0.7%)	Imbalanced	
region_category has 5428 (4.7%) missing values	Missing	
points_in_wallet has 3443 (3.3%) missing values	Missing	
id is uniformly distributed	Uniform	
security_no is uniformly distributed	Uniform	
last_visit_time is uniformly distributed	Uniform	
id has unique values	Unique	
security_no has unique values	Unique	



## Unnamed

Foi decidido remover 'Unnamed' do conjunto de dados porque ela não fornece informações úteis para a análise ou para a construção do modelo. Muitas vezes, essa coluna 'Unnamed' é gerada ao salvar um DataFrame pandas em um arquivo CSV, onde o índice do DataFrame é salvo como uma coluna adicional sem nome.

```
df = drop_cols(df, ['Unnamed: 0'])
```

## Gender

A decisão de incluir uma categoria de 'non-binary' para a variável 'gender' pode ser uma consideração importante. Em termos de modelagem de dados, adicionar uma categoria de 'non-binary' pode nos ajudar a capturar mais nuances em nossos dados. Além disso, as categorias 'M', 'F' e 'Non-Binary' foram transformadas em números (0, 1, 2).

```
df['gender'] = df['gender'].replace('Unknown', 'Non-binary')
df['gender'] = df['gender'].replace({'M': 0, 'F': 1, 'Non-binary': 2})
```

## security\_no

'security\_no' parece ser uma identificação única para cada registro, similar a uma chave primária em um banco de dados. Portanto, pode ser removida do conjunto de dados porque não fornece informações significativas para a modelagem.

```
df = df.drop(columns=['security_no'])
```

## region\_category

No contexto de divisões de territórios, existem vários termos que podem ser usados para descrever diferentes tipos de comunidades além de Town, City e Village. Por exemplo: Hamlet, Metropolis, Megalopolis, District, etc. Além disso, foi decidido converter os valores categóricos para numéricos.

```
df['region_category'] = df['region_category'].fillna('Other')
df['region_category'] = df['region_category'].map({'Village': 0,
'Town': 1, 'City': 2, 'Other': 3})
```

## joined\_through\_referral

'?' representa um dado desconhecido e compõe uma proporção significativa dos dados (14,7%). Nesse caso, será substituído '?' por 'Not\_Informed' para representar explicitamente que esses dados não foram informados. Além disso, foi decidido converter os valores categóricos para numéricos.

```
df['joined_through_referral'].replace('?', 'Not_Informed', inplace=True)
df['joined_through_referral'] = df['joined_through_referral'].map({'Yes': 1, 'No': 0, 'Not_Informed': -1})
```

## referral\_id

referral\_id indica não ser relevante para o modelo, portanto faz sentido removê-lo, pois IDs únicos ou quase únicos não fornecem informações úteis para um modelo.

```
df = df.drop(columns=[ 'referral_id' ])
```

## preferred\_offer\_types

A coluna preferred\_offer\_types provavelmente se refere aos tipos de ofertas preferidos pelos usuários ou clientes. Estes podem incluir várias categorias de ofertas que são oferecidas aos usuários, como "Gift Vouchers/Coupons", "Credit/Debit Card Offers", "Without Offers" e outros. Foi considerado transformar essa coluna em variável numérica.

```
df['preferred_offer_types'].fillna('Other', inplace=True)
df['preferred_offer_types'] = df['preferred_offer_types'].map({'Gift Vouchers/Coupons': 0, 'Credit/Debit Card Offers': 1, 'Without Offers': 2, 'Other': 3})
```

## medium\_of\_operation

Os dados da coluna medium\_of\_operation possuem um número significativo de valores desconhecidos '?' (14.6%). Como há outros tipos de dispositivos, como Tablet, faz sentido considerar os valores desconhecidos com o tipo 'Other'. Foi considerado transformar essa coluna em variável numérica.

```
df['medium_of_operation'].replace('?', 'Other', inplace=True)
df['medium_of_operation'] = df['medium_of_operation'].map({'Desktop': 0, 'Smartphone': 1, 'Both': 2, 'Other': 3})
```

## last\_visit\_time

last\_visit\_time tem alta cardinalidade, o que significa que ela tem muitos valores únicos. Nesse caso, 81.4% dos valores são únicos, o que é bastante alto. Além disso, esta coluna representa o horário da última visita, que pode não ter uma relação clara ou direta com a taxa de churn. Por estas razões, faz sentido remover a coluna last\_visit\_time.

```
df = df.drop(columns=['last_visit_time'])
```

## days\_since\_last\_login

Aparentemente -999 representa um valor específico (como o usuário nunca fez login), portanto faz sentido substituí-lo pelo valor 0.

```
df['days_since_last_login'] =  
df['days_since_last_login'].replace(-999, 0)
```

## avg\_time\_spent

O campo 'avg\_time\_spent' provavelmente representa o tempo médio gasto por um usuário no uso de um serviço. Neste contexto, não faria sentido ter um valor de tempo médio negativo, pois o tempo gasto não pode ser menor que zero. O fato de haver valores negativos pode ser devido a um erro de codificação ou de entrada de dados. Portanto, é plausível considerar que o valor absoluto desses números é o tempo correto.

```
df['avg_time_spent'] = df['avg_time_spent'].abs()
```

## avg\_frequency\_login\_days

Substituir os valores de 'Error' na coluna 'avg\_frequency\_login\_days' pela mediana, média, ou qualquer outro valor pode introduzir ruído nos dados que não correspondem à realidade do comportamento do usuário. 'Error' pode ser uma indicativa de algum tipo de problema ou anomalia com a conta do usuário (por exemplo, a conta está inativa, foi excluída, etc.), então substituir essas marcas de 'Error' por um valor numérico poderia potencialmente nos levar a tirar conclusões imprecisas sobre esses usuários.

```
df['avg_frequency_login_days'].replace('Error', np.nan, inplace=True)  
df.dropna(subset=['avg_frequency_login_days'], inplace=True)
```

## points\_in\_wallet

No contexto de uma plataforma de recompensas, valores negativos podem representar situações onde os usuários têm uma dívida de pontos ou um déficit de pontos. Assim, vou decidido manter os valores negativos. Além disso, considerando que temos 9,3% dos valores faltantes em 'points\_in\_wallet', preencher esses valores com a média ou a mediana pode introduzir um viés significativo. Portanto, é uma decisão mais segura remover esses valores faltantes em vez de preenchê-los, garantindo assim que nosso modelo seja treinado apenas em dados confiáveis.

```
df = df.dropna(subset=['points_in_wallet'])
```

## Joining\_date

Foi decidido realizar (one-hot) nesta coluna, separando o ano, mes e dia.

Internet\_option

Used\_special\_discount

Past\_complaint

Complaint\_status

Feedback

Membership\_category

Offer\_application\_preference

Foram convertidos em atributos numéricos

- Essas correlações representam a medida da relação linear entre cada variável e o target (**churn\_risk\_score**). Valores mais próximos de 1 indicam uma correlação positiva forte, ou seja, à medida que o valor da variável aumenta, o **churn\_risk\_score** também tende a aumentar.
- Observando as correlações, **podemos identificar quais variáveis têm uma relação mais forte com o churn\_risk\_score**.
- No caso, a **membership\_category** tem a maior correlação (0.71), indicando uma relação positiva significativa com o risco de churn. Outras variáveis como **feedback**, **points\_in\_wallet** e **avg\_transaction\_value** também têm correlações positivas moderadas.

churn_risk_score	1.000000
membership_category	0.717771
feedback	0.349802
points_in_wallet	0.307454
avg_transaction_value	0.212212
preferred_offer_types	0.044235
region_category	0.016678
used_special_discount	0.015832
joined_through_referral	0.015492
offer_application_preference	0.014414
complaint_status	0.013175
past_complaint	0.011327
avg_time_spent	0.009602
joining_day	0.008829
days_since_last_login	0.008061
joining_year	0.007395
internet_option	0.006349
age	0.005420
joining_month	0.004838
gender	0.004268
medium_of_operation	0.000909
Name: churn_risk_score, dtype: float64	

# Resultados (baseline x aperfeiçoado)

Acurácia do modelo **baseline**: 77.96079723274583

Acurácia do modelo **aperfeiçoado**: 0.8932630538626256

---

- Os resultados parecem ter melhorado significativamente entre o modelo baseline e o modelo aperfeiçoado.
- O modelo baseline, que utilizou uma abordagem simplista de Regressão Logística sem ajustes de hiperparâmetros ou pré-processamento de dados, obteve uma acurácia de aproximadamente **77.96%**.
- No entanto, após o aperfeiçoamento, com a seleção de features mais relevantes, normalização dos dados com **QuantileTransformer** e **ajuste fino dos hiperparâmetros usando GridSearchCV**, a acurácia aumentou para cerca de **89.33%**.
- **Esse aumento é substancial e sugere que as alterações feitas no processo de modelagem resultaram em uma melhoria notável na capacidade do modelo de prever corretamente a variável de interesse 'churn\_risk\_score'.** A seleção de características e a otimização de hiperparâmetros parecem ter contribuído para um modelo mais eficiente.

Foi utilizado o conjunto de dados ajustado com as melhores correlações (**membership\_category, feedback, points\_in\_wallet**) para ambos os modelos.



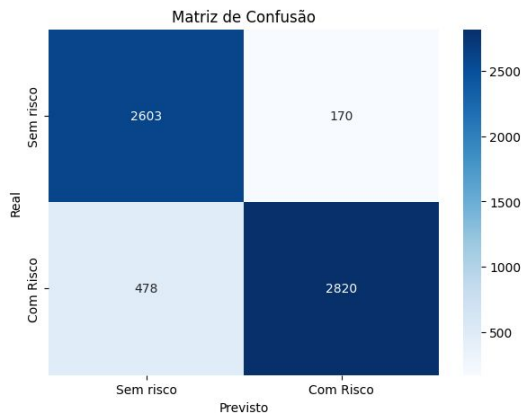
**classifier\_\_penalty:** Determina a função de penalidade utilizada no algoritmo de otimização da regressão logística. Neste caso, foi considerado duas opções: 'l1' (penalidade L1, também conhecida como **regularização Lasso**) e 'l2' (penalidade L2, também conhecida como **regularização Ridge**).

**classifier\_\_solver:** Define o algoritmo utilizado para resolver o problema de otimização durante o treinamento do modelo de regressão logística. Foi utilizado várias opções, como 'liblinear', 'saga', 'newton-cg', 'lbfgs' e 'sag'.

**classifier\_\_C:** Parâmetro de regularização, que controla a intensidade da penalidade aplicada aos coeficientes do modelo. Valores menores de C aumentam a regularização, levando a modelos mais simples, enquanto valores maiores de C diminuem a regularização, permitindo que o modelo se ajuste mais aos dados de treinamento. **Neste caso, foi utilizado um espaço de busca logarítmico entre  $10^{-4}$  e  $10^4$ , com 50 valores distintos.**

**classifier\_\_class\_weight:** Define a estratégia de ponderação dos pesos das classes. A opção 'None' indica que **todas as classes têm o mesmo peso**, enquanto a opção 'balanced' **ajusta automaticamente os pesos das classes** inversamente proporcionais às frequências das classes no conjunto de treinamento.

	precision	recall	f1-score	support
0	0.84	0.94	0.89	2773
1	0.94	0.86	0.90	3298
accuracy			0.89	6071
macro avg	0.89	0.90	0.89	6071
weighted avg	0.90	0.89	0.89	6071



Analisando os resultados para o modelo aperfeiçoado:

Classe 0 (não churn):

- **Precisão:** 84%. Isso significa que quando o modelo previu a classe 0, estava correto 84% das vezes.
- **Recall:** 94%. Isso significa que o modelo foi capaz de identificar 94% das instâncias da classe 0 corretamente.
- **F1-score:** 89%. Isso sugere que o equilíbrio entre precisão e recall para a classe 0 é bastante bom.

Classe 1 (churn):

- **Precisão:** 94%. Isso significa que quando o modelo previu a classe 1, estava correto 94% das vezes.
- **Recall:** 86%. Isso significa que o modelo foi capaz de identificar 86% das instâncias da classe 1 corretamente.
- **F1-score:** 90%. Isso sugere que o equilíbrio entre precisão e recall para a classe 1 é bom.

**Acurácia total:** 89%. Isso significa que, em geral, o modelo fez a previsão correta para 89% das instâncias.

Esses resultados sugerem que o modelo aperfeiçoado teve um bom desempenho. É especialmente forte na identificação da classe 0 (não churn), como evidenciado pela alta pontuação de recall. Por outro lado, embora ainda seja bom, o recall é um pouco mais baixo para a classe 1 (churn), o que significa que há mais casos de falsos negativos (casos em que a realidade é 1, mas o modelo previu 0).

Abaixo é possível visualizar todas as tentativas realizadas pelo modelo em sua validação cruzada. É exibido o score obtido para diferentes combinações de hiperparâmetros. Os valores mostrados indicam o desempenho médio do modelo em cada iteração da validação cruzada.

```
0.45961863886603604 {'classifier_C': 0.0001, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.5323088734898124 {'classifier_C': 0.0001, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.0001, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.49998107708679373 {'classifier_C': 0.0001, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.00014563484775012445, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.5323088734898124 {'classifier_C': 0.00014563484775012445, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.00014563484775012445, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.5080914105573578 {'classifier_C': 0.00014563484775012445, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.00021209508879201905, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.540381361133964 {'classifier_C': 0.00021209508879201905, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.00021209508879201905, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.5241985400192484 {'classifier_C': 0.00021209508879201905, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.00030888435964774815, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.540381361133964 {'classifier_C': 0.00030888435964774815, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.00030888435964774815, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.49190858944264215 {'classifier_C': 0.00030888435964774815, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.0004498432668969444, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.7773155643334964 {'classifier_C': 0.0004498432668969444, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.0004498432668969444, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.843130816013261 {'classifier_C': 0.0004498432668969444, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.6529382570436983 {'classifier_C': 0.0006551285568595509, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.7996786156837717 {'classifier_C': 0.0006551285568595509, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.45961863886603604 {'classifier_C': 0.0006551285568595509, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.843130816013261 {'classifier_C': 0.0006551285568595509, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.8482761666925528 {'classifier_C': 0.0009540954763499944, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.7997190818452214 {'classifier_C': 0.0009540954763499944, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.8482761666925528 {'classifier_C': 0.0009540954763499944, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.8866190756529931 {'classifier_C': 0.0009540954763499944, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.869403768847459 {'classifier_C': 0.0013894954943731374, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.8670152766916589 {'classifier_C': 0.0013894954943731374, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.8482761666925528 {'classifier_C': 0.0013894954943731374, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.8870388185753826 {'classifier_C': 0.0013894954943731374, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.885877586589284 {'classifier_C': 0.0020235896477251557, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.8778231239355862 {'classifier_C': 0.0020235896477251557, 'classifier_class_weight': None, 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
0.8696098522588466 {'classifier_C': 0.0020235896477251557, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'liblinear'}
0.8868250573282881 {'classifier_C': 0.0020235896477251557, 'classifier_class_weight': 'balanced', 'classifier_penalty': 'l1', 'classifier_solver': 'saga'}
```

Analisando os resultados para o modelo TPOT:

Classe 0 (não churn):

- **Precisão:** 96%. Isso significa que quando o modelo previu a classe 0, estava correto 96% das vezes.
- **Recall:** 92%. Isso significa que o modelo foi capaz de identificar 92% das instâncias da classe 0 corretamente.
- **F1-score:** 94%. Isso sugere que o equilíbrio entre precisão e recall para a classe 0 é excelente.

Classe 1 (churn):

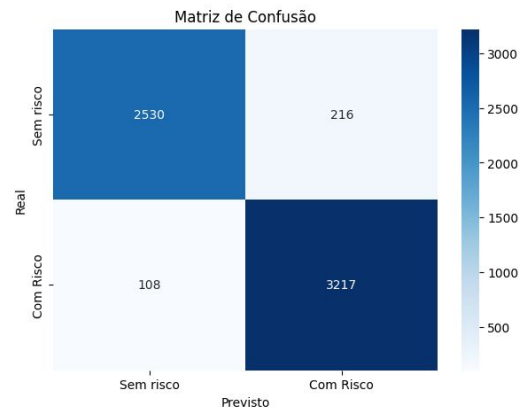
- **Precisão:** 94%. Isso significa que quando o modelo previu a classe 1, estava correto 94% das vezes.
- **Recall:** 97%. Isso significa que o modelo foi capaz de identificar 97% das instâncias da classe 1 corretamente.
- **F1-score:** 95%. Isso sugere que o equilíbrio entre precisão e recall para a classe 1 é excelente.

**Acurácia total:** 95%. Isso significa que, em geral, o modelo fez a previsão correta para 95% das instâncias.

Esses resultados sugerem que o modelo TPOT teve um desempenho ainda melhor do que o modelo aperfeiçoado. Ele apresentou maior acurácia, melhor precisão e recall para ambas as classes. O modelo parece ser particularmente bom na identificação da classe 1 (churn), como evidenciado pela alta pontuação de recall.

Com base nestes resultados, podemos concluir que o modelo TPOT foi mais eficaz na classificação dos dados do que o modelo aperfeiçoado.

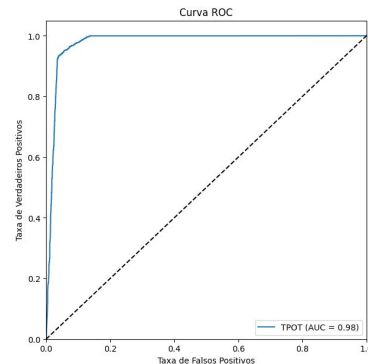
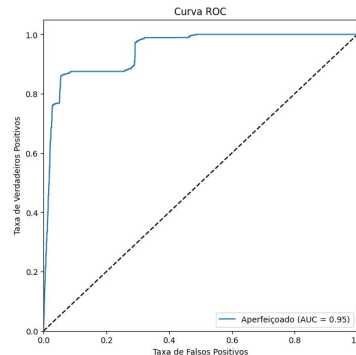
	precision	recall	f1-score	support
0	0.96	0.92	0.94	2746
1	0.94	0.97	0.95	3325
accuracy			0.95	6071
macro avg	0.95	0.94	0.95	6071
weighted avg	0.95	0.95	0.95	6071



A **AUC-ROC para o modelo aperfeiçoado é de 0.95**. Isso significa que, em média, o modelo classifica corretamente uma instância aleatória positiva da classe 1 acima de uma instância aleatória da classe 0 com **95% de probabilidade**.

Por outro lado, a **AUC-ROC para o TPOT é de 0.98**. Isso indica que, em média, o modelo TPOT classifica corretamente uma instância aleatória positiva da classe 1 acima de uma instância aleatória da classe 0 com **98% de probabilidade**. Isso sugere que o modelo TPOT é mais eficaz em classificar corretamente as instâncias das duas classes.

Portanto, enquanto ambos os modelos parecem funcionar bem, porém o modelo **TPOT indica ter um desempenho ligeiramente superior ao modelo aperfeiçoado**.



Dagshub e MLflow demonstraram ser ferramentas poderosas para gerenciar, manter e visualizar os experimentos.

Como pode ser observado, vários experimentos foram executados, onde é possível visualizar cada hiperparâmetro utilizado, bem como a acurácia. O melhor modelo obteve 89%

O repositório público pode ser visualizado em:  
(Submeti os resultados do modelo aperfeiçoado e TPOT.)

<https://dagshub.com/mtsfreitas/ccf726tp2/experiments/#/>

	Code	Name	Created	Labels	Source	Group	C	penalty	solver	score	Accuracy
<input type="checkbox"/>	(W)	sneaky-stag-579	2 hours ago		mlflow	ChurnRisk				0.9466315269...	
<input type="checkbox"/>	(W)	calm-dolphin-971	7 hours ago		mlflow	ChurnRisk				0.9466315269...	
<input type="checkbox"/>	(W)	abracadabra-yak-239	5 hours ago		mlflow	ChurnRisk				0.9459726568...	
<input type="checkbox"/>	(W)	beautiful-midge-15	9 hours ago		mlflow	ChurnRisk				0.943018942...	
<input type="checkbox"/>	(W)	adaptable-carp-65	5 hours ago		mlflow	ChurnRisk				0.9410311316...	
<input type="checkbox"/>	(W)	flexible-crow-386	5 hours ago		mlflow	ChurnRisk	0.0020235896...	l1	saga	0.8870310390...	0.8952396639...
<input type="checkbox"/>	(W)	redident-cuck-972	8 hours ago		mlflow	ChurnRisk	0.0020235896...	l1	saga	0.8867015157...	0.8945807939...
<input type="checkbox"/>	(W)	bright-lamb-740	7 hours ago		mlflow	ChurnRisk	0.0009540954...	l1	saga	0.8874015109...	0.8944160764...
<input type="checkbox"/>	(W)	hilarious-midge-96	5 hours ago		mlflow	ChurnRisk	0.0020235896...	l1	saga	0.8876076113...	0.8944160764...
<input type="checkbox"/>	(W)	flawless-shad-734	7 hours ago		mlflow	ChurnRisk	0.0009540954...	l1	saga	0.8875250355...	0.8940866414...
<input type="checkbox"/>	(W)	linguist-ros-299	5 hours ago		mlflow	ChurnRisk	0.0009540954...	l1	saga	0.8875250355...	0.8940866414...
<input type="checkbox"/>	(W)	dazzling-vole-8	2 hours ago		mlflow	ChurnRisk				0.8870309203...	0.8932630538...
<input type="checkbox"/>	(W)	sincere-shark-656	9 minutes ago		mlflow	ChurnRisk				0.8576254762...	0.8657552297...
<input type="checkbox"/>	(W)	zealous-elk-733	17 minutes ago		mlflow	ChurnRisk				0.8485236156...	0.8580135068...

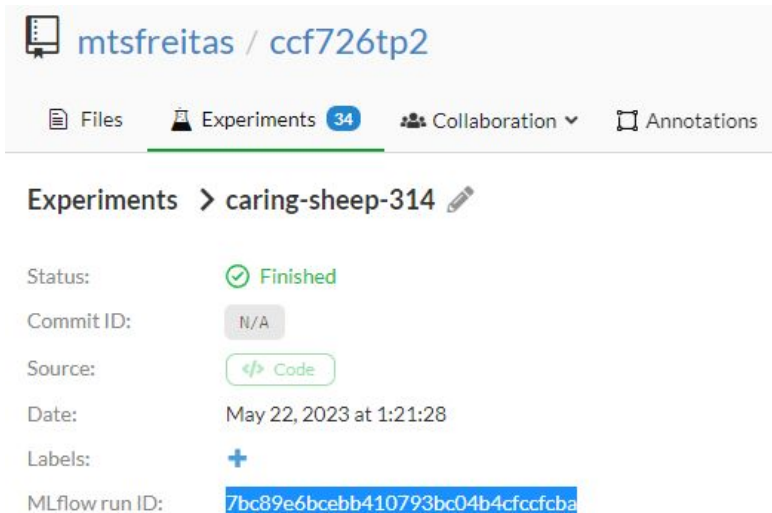
Experiments > Comparing 3 Experiments

Experiment ID:	zealous-elk-733	sincere-shark-656	dazzling-vole-8
Commit ID:	N/A	N/A	N/A
Created:	14 minutes ago	5 minutes ago	2 hours ago
Labels:			
Parameters			
classifier_penalty	l2	l2	l1
classifier_solver	newton-cg	newton-cg	saga
classifier_class_weight	undefined	None	balanced
classifier_C	undefined	0.0013894954943731374	0.0009540954763499944
Metrics			
Accuracy	0.858013506835777	0.865755229780926	0.893263053862626
score	0.848523615603967	0.857625476209693	0.887030920311475

Além disso, é possível visualizar outras informações de forma gráfica dos modelos.



Também é possível utilizar o modelo para realizar previsões, basta utilizar o **MLflow run ID**:



The screenshot shows the MLflow web interface for a user named 'mtsfreitas'. The breadcrumb navigation shows 'ccf726tp2' > 'Experiments'. The 'Experiments' tab is active, showing a list of experiments. The experiment 'caring-sheep-314' is selected, showing its details: Status is 'Finished' (green checkmark), Commit ID is 'N/A', Source is 'Code' (with a code icon), Date is 'May 22, 2023 at 1:21:28', and Labels are empty. The MLflow run ID is highlighted in blue: '7bc89e6bcebb410793bc04b4cfccfcb4'.

▼ Predict do modelo para realizar previsões

```
# Carregando o modelo do MLflow
model = mlflow.sklearn.load_model("runs:/7bc89e6bcebb410793bc04b4cfccfcb4/model")

# Features e o alvo (target)
features = ['membership_category', 'feedback', 'points_in_wallet']
target = 'churn_risk_score'

# Preparando os dados usando a função prepare_data
(train_x, test_x, train_y, test_y) = prepare_data(df, target, features)

# Usando o modelo carregado para fazer previsões
predicted_qualities = model.predict(test_x)

# Métricas de desempenho
metrics = eval_metrics(test_y, predicted_qualities)
print(metrics)

{'accuracy': 0.8944160764289244, 'precision': 0.9000416348945727, 'recall': 0.8944160764289244}
```



## Modelo de Regressão Logística aperfeiçoado X TPOT:

- Vale destacar que a acurácia obtida com o TPOT foi de aproximadamente **94%**, o que é superior à acurácia do modelo aperfeiçoado de **89%**. Isso sugere que o **TPOT foi capaz de encontrar um pipeline de aprendizado de máquina que produz um melhor desempenho no seu conjunto de teste.**
- Entretanto, apesar de seus pontos fortes, o TPOT pode não ser a melhor solução em todos os casos. **Ele pode demorar muito para executar, especialmente em grandes conjuntos de dados ou com um grande número de gerações.**
- Ainda assim, o **TPOT** é uma ferramenta poderosa que **pode economizar muito tempo e esforço, pois ele automatiza o que normalmente seria um processo de ajuste de parâmetros manual e trabalhoso.**

# OBRIGADO!

matheus.f.martins@ufv.br

