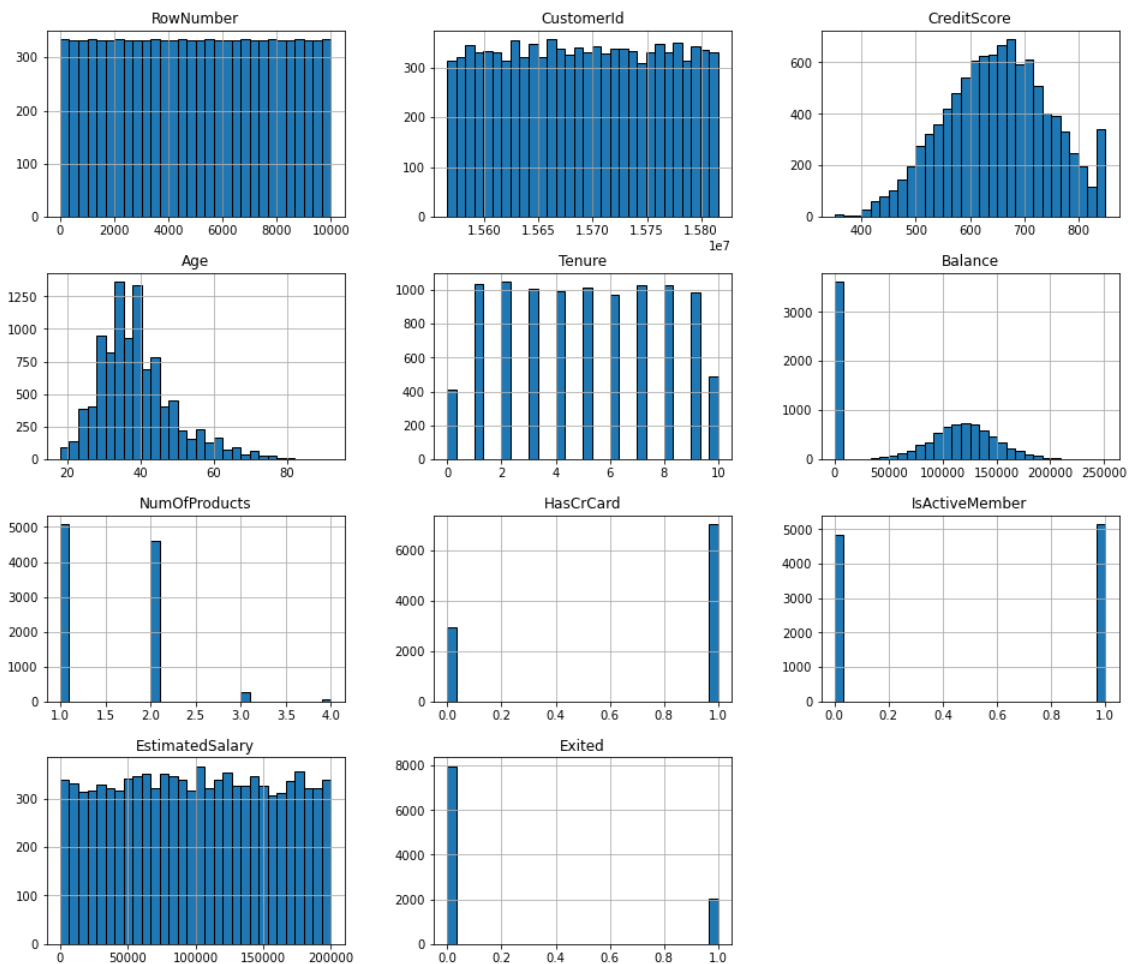


1. Atividades

1. Descreva graficamente os dados disponíveis, apresentando as principais estatísticas descritivas. Comente o por quê da escolha dessas estatísticas.

1. Para essa atividade foi usado o histograma para as *features* numéricas, como forma de entender suas distribuições.

Distribuição das Variáveis Numéricas



Para essa primeira análise, podemos desconsiderar as colunas “RowNumber” e “CustomerId” por se tratarem de identificação.

Os campos “CreditScore” e “Age” possuem uma tendência à distribuição Normal, com deslocamentos da média e mediana para direita e

esquerda, respectivamente. A “balance” também segue essas características, mas com uma dispersão maior, dada pelo seu “achatamento” e possuindo possivelmente *outliers*.

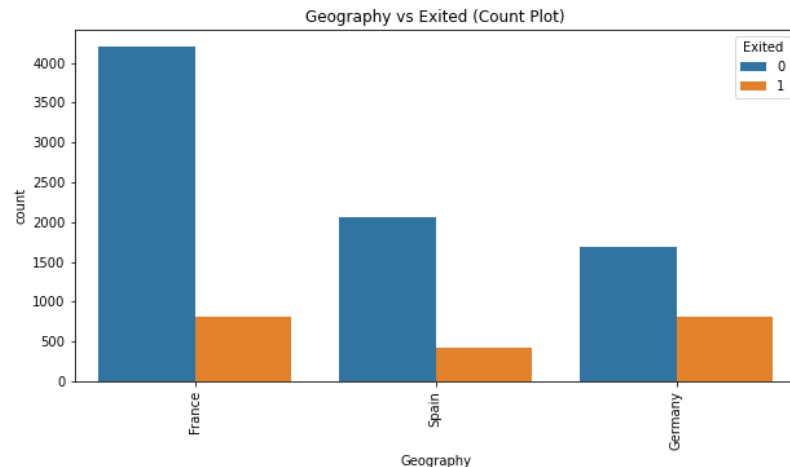
“Tenure” e “EstimatedSalary” seguem uma tendência de distribuição Uniforme, caracterizada por um pico mais linear.

“NumOfProducts” é possível identificar um maior apressio por apenas 1 e 2 produtos. Para 3 e 4 a relação é quase desprezível.

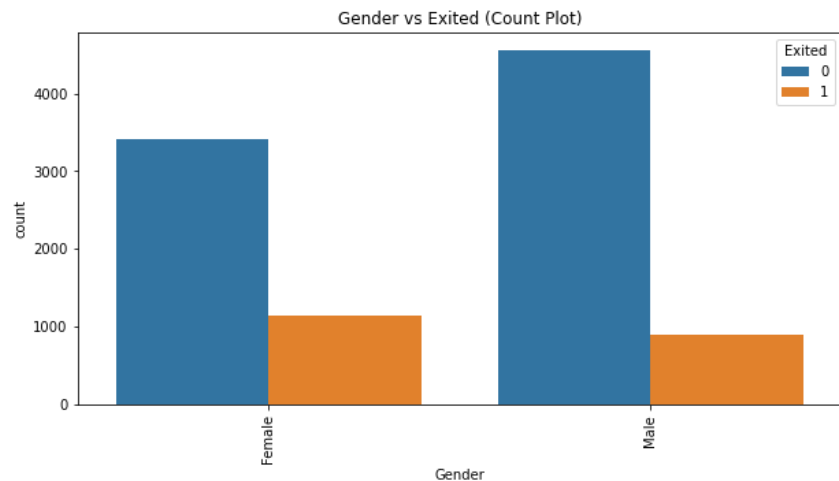
“HasCrCard” e “Exited”, variáveis binárias, destaca-se o desbalanceamento das classes. Sendo a classe 1 maior em “HasCrCard”, enquanto que a classe 0 é maior em “Exited”.

Todos esses pontos podem influenciar na decisão final de métodos estatísticos, assim como nas transformações que serão empregadas a eles.

Para as *features* categóricas foi gerado gráfico das classes e suas respectivas representatividades no Churn.



Para “Geography” a quantidade para 0 é maior em todos os países, possuindo um volume maior para França, sendo ela e Espanha os países que causam maior desbalanceamento.



A *feature* “Gender” possui um volume quase uniforme para ambas as classes tanto para 0 quanto para 1, mas com destaque para os homens com maior número no *dataset*.

2. Explique como você faria a previsão do Churn a partir dos dados. Quais variáveis e/ou suas transformações você utilizou e por quê? Qual tipo de problema estamos resolvendo (regressão, classificação)? Qual modelo melhor se aproxima dos dados e quais seus prós e contras? Qual medida de performance do modelo foi escolhida e por quê?

1. Para um projeto de Ciência de Dados, considero o CRISP-DS (*Cross Industry Standard Process for Data Science*). Esse método se resume em criar um ciclo de melhoria da entrega.

No case, para um *Baseline*, foi aplicado transformação nas variáveis categóricas, sendo elas as: “Geography” e “Gender”. Para as duas foram gerados *Dummies*, que faz a criação de classes para os valores encontrados nessas *features*. Para “Geography” são “Germany”, “France” e “Spain”, enquanto que “Gender” são “Male” e “Female”.

Única transformação em *feature* numérica foi desconsiderar as “RowNumber” e “CustomerId”.

Para as *features* “CreditScore”, “Age”, “Balance” e “EstimatedSalary” pode ser aplicado transformação de suas escalá-las, de forma a

padroniza-las. Isso deve auxiliar em melhoria do desempenho dos algoritmos.

Para as variáveis binárias, o desbalanceamento das *feature target* foi priorizada. Ela foi dividida de forma que fosse possível aproveitar ao máximo para o treinamento, mas também que não houvesse *overfitting* para a classe de maior número. Houve então a separação por classes 70-30.

Para o problema de Classificação que estamos lidando, foram usadas as seguintes métricas:

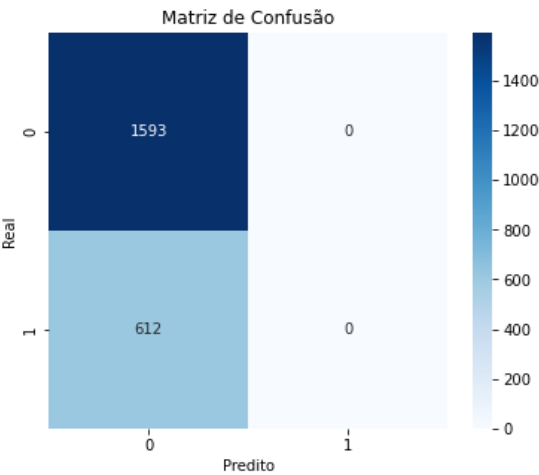
- *Precision* – Indica quanto dos exemplos previstos como Churn realmente são;
- *Recall* – Indica quanto dos exemplos de Churn foram previstos de forma correta;
- *F1-Score* – Média Harmônica entre *Precision* \times *Recall*, balanceando as duas métricas;

De modelos, foram usados 4 algoritmos:

- Regressão Logística;
Apresentando uma média geral de *Precision* no valor de 36%, enquanto que *Recall* uma média de 50%, mas para ambas a classe 1 não obteve previsão, consequentemente em um *F1-Score* igual a 0%. O desbalanceamento dos dados pode explicar esse fato, assim como tipos diferentes de *features*.

Modelo: LogisticRegression
Relatório de Classificação:

	precision	recall	f1-score	support
0	0.72	1.00	0.84	1593
1	0.00	0.00	0.00	612
accuracy			0.72	2205
macro avg	0.36	0.50	0.42	2205
weighted avg	0.52	0.72	0.61	2205

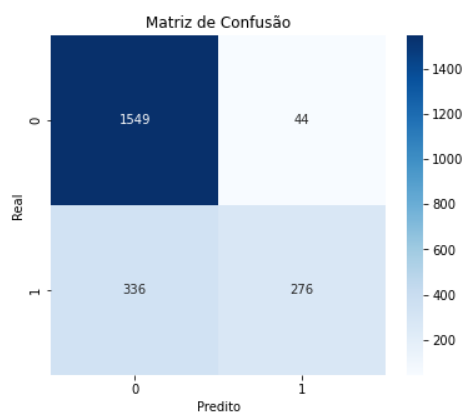


- *Random Forest Classifier*;

Apresentando uma média geral de *Precision* no valor de 84%, enquanto que *Recall* uma média geral de 71%. Destacando que dos exemplos de Churn, 45% foram classificados de forma correta. No fim, temos um *F1-Score* de 59% para Churn.

Modelo: RandomForestClassifier
Relatório de Classificação:

	precision	recall	f1-score	support
0	0.82	0.97	0.89	1593
1	0.86	0.45	0.59	612
accuracy			0.83	2205
macro avg	0.84	0.71	0.74	2205
weighted avg	0.83	0.83	0.81	2205

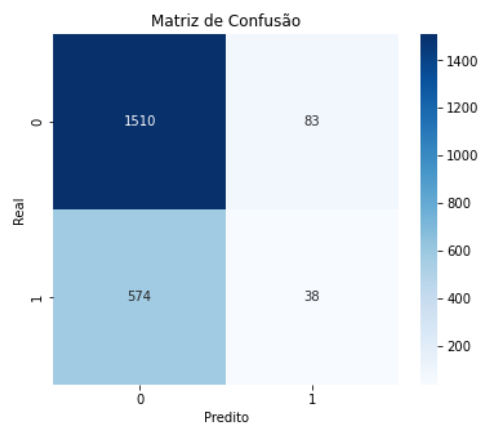


- K-Nearest Neighbors (KNN);

Apresentando uma média geral de *Precision* no valor de 52%, enquanto que *Recall* uma média geral de 50%. Mas um *Recall* para classe 1 (churn) de 6% indica que apenas 6% dos exemplos de churn foram classificados de maneira correta.

Modelo: K-Nearest Neighbors (KNN)
Relatório de Classificação:

	precision	recall	f1-score	support
0	0.72	0.95	0.82	1593
1	0.31	0.06	0.10	612
accuracy			0.70	2205
macro avg	0.52	0.50	0.46	2205
weighted avg	0.61	0.70	0.62	2205



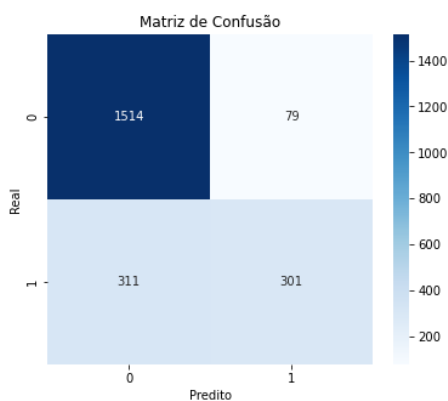
- XGBoost Classifier.

Apresentando uma média geral de *Precision* no valor de 81%, enquanto que *Recall* uma média geral de 72%. Destacando que dos exemplos de Churn, 49% foram classificados de forma correta. No fim, temos um *F1-Score* de 61% para Churn.

```
Modelo: XGBClassifier
Relatório de Classificação:
precision    recall  f1-score   support

   0         0.83    0.95    0.89     1593
   1         0.79    0.49    0.61      612

 accuracy          0.82     2205
 macro avg         0.81    0.72    0.75     2205
weighted avg         0.82    0.82    0.81     2205
```



Resumindo:

		Precision	Recall	F1-Score
Regressão Logística	0	72%	100%	84%
	1	0%	0%	0%
K-Nearest Neighbors	0	72%	95%	82%
	1	31%	6%	10%
Random Forest	0	82%	97%	89%
	1	86%	45%	59%
XGBoost	0	83%	95%	89%
	1	79%	49%	61%

Dessa forma, o algoritmo selecionado para subir como primeira versão foi o XGBoost.

Na sequência, o XGBoost sofreu um *Tunning* de seus parâmetros. Inclusive, esse é um ponto contra esse algoritmo, por ser robusto, a otimização de seus parâmetros se torna mais difícil e pode não ser indicado para base de dados pequena.

Pró dele, como já indicado, é robusto com e com ótimo desempenho, indicado e usado em diversos desafios do Kaggle, assim como evita *overfitting*.

3. Construa um pipeline de machine learning que realize a previsão de churn a partir de um dataset CSV. Esse pipeline deve ser reproduzível e permitir realizar uma previsão a partir de qualquer arquivo CSV com a mesma estrutura de dados. O modelo utilizado pelo pipeline deve ser treinado com o dataset `Abandono_clientes.csv`. Você deve nos enviar o repositório de código com o pipeline.

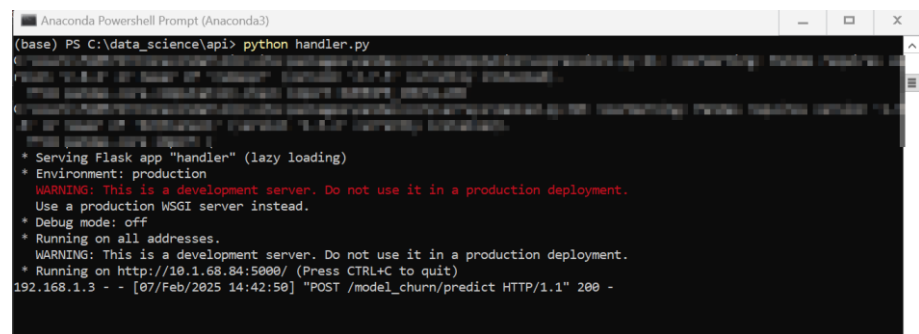
1. Para produção foi criado uma API POST, que recebe a planilha no *layout* e retorna um CSV com a predição correspondente.

A aplicação está em formato LOCAL, mas foi desenvolvida para está na nuvem ou em servidor específico, onde pode ser processada, fornecendo os dados requisitados pelo Cliente. Isso possível com um executável disponibilizado (ou App) para que ele rode e pegue o respectivo CSV.

Foi gerado o “handler.py”, que possui o corpo da API e a *pipeline* que será seguida pelos dados. Para nuvem ou servidor local serão necessárias adaptações de caminho e link de conexão.

Há também o arquivo “model_churn.py”, que apresenta as funções que serão responsáveis pelas transformações e pela predição de Churn.

Na imagem a seguir é apresentado o servidor acionado



```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\data_science\api> python handler.py
...
* Serving Flask app "handler" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.1.68.84:5000/ (Press CTRL+C to quit)
192.168.1.3 - - [07/Feb/2025 14:42:50] "POST /model_churn/predict HTTP/1.1" 200 -
```

E a seguir a requisição sendo feita com base de teste

1	# Aplicando base de teste
2	df_api_t = df_teste
1	# Convertendo para json
2	data = json.dumps(df_api_t.to_dict(orient='records'))
1	# API call
2	# url = 'http://0.0.0.0:5000/model_churn/predict'
3	url = 'http://192.168.1.3:5000/model_churn/predict'
4	header = {'Content-type': 'application/json'}
5	data = data
6	
7	r = requests.post(url,data=data,headers = header)
8	
9	print('Status Code {}'.format(r.status_code))
Status Code 200	
1	res = pd.DataFrame(r.json(),columns=r.json()[0].keys())
2	res

CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	predictedValues
15798485	Copley	565	France	Male	31	1	0.00	1	0	1	20443.08	0
15588959	Tang	569	France	Male	34	4	0.00	1	0	1	4045.90	0
15624896	Ku	669	France	Female	20	7	0.00	2	1	0	128838.67	0
15639629	McConnan	694	France	Male	39	4	173255.48	1	1	1	81293.10	0
15638852	Ts'ui	504	Spain	Male	28	10	109291.36	1	1	1	187593.15	0
...
15730373	Starks	531	France	Female	34	10	118306.79	1	1	0	28493.05	0
15716191	Dixon	575	Germany	Male	49	2	136822.70	1	1	0	2487.74	1
15673900	Wilkinson	520	France	Female	74	4	0.00	1	0	0	26742.92	0
15581432	Oatley	675	Spain	Male	23	8	0.00	2	0	0	162342.21	0
15669645	Tyler	684	France	Male	46	10	0.00	2	1	0	188772.98	0

4. Envie o resultado final do modelo em um arquivo com apenas duas colunas (rowNumber, predictedValues) gerado ao rodar o pipeline de 3 no dataset abandono_teste.csv em anexo.

1. Em anexo vai a planilha com a predição dos dados de teste enviados.