# 

# Pontifícia Universidade Católica RJ

# Previsão de *Churn em* Telecom

Utilização de algoritmos de machine learning para classificação de Churn em produtos de Telecom

**Curso de Pós-graduação:** BI Master - Intelligent Business Decision Support Systems

**Disciplina:** Data Mining

**Aluno:** Daniel Werneck Rodrigues

Rio de Janeiro, 08 de janeiro de 2022

1. Introdução
2. Bibliotecas Importadas
3. Configurações e Funções Úteis
4. Análise Inicial da Base de Dados
5. Análise Exploratória
   1. Variáveis Contínuas
   2. Variáveis Categóricas
   3. Análise de Correlação
   4. Análise Bivariada
   5. Separação em X e y
   6. Treinando Árvore de Decisão
6. Pré Processamento
   1. Padronização dos Dados
   2. Utilizando Técnicas de Resample
7. Escolha de Métricas para Avaliação de Modelos
8. Treinando e Avaliando Modelos de Machine Learning
   1. Avaliação Inicial de Modelos
   2. Comparativo de Bases Utilizadas (Overfit)
9. Avaliação e Seleção de Modelos Promissores
   1. Gradient Boost
   2. Random Forest
   3. XGBoost
   4. LightGBM
   5. SVM
10. *Insights* do Modelo e Avaliação de Melhor Fronteira de Decisão
    1. Importância dos Atributos
    2. Relatório xgbfir
    3. Curva de Aprendizagem
    4. Curva ROC
    5. Curva Precisão-Recall
    6. Gráfico de Ganhos Cumulativos
    7. Gráfico de *Lift*
    8. Explicabilidade do modelo com *Lime*
    9. Avaliação de Melhor Fronteira de Decisão
11. Conclusão
12. Referências
13. **Introdução**

Com o rápido desenvolvimento da indústria de telecomunicações, os provedores de serviços estão cada vez mais interessados na expansão da base de assinantes. Para atender à necessidade desse ambiente competitivo, a retenção dos clientes existentes tornou-se um grande desafio. Afirma-se que o custo de adquirir um novo cliente é muito maior do que o de manter o existente. Portanto, é imperativo que as indústrias de telecomunicações usem análises avançadas para entender o comportamento do consumidor e, por sua vez, prever a afinidade dos clientes quanto ao cancelamento de serviço com a empresa.

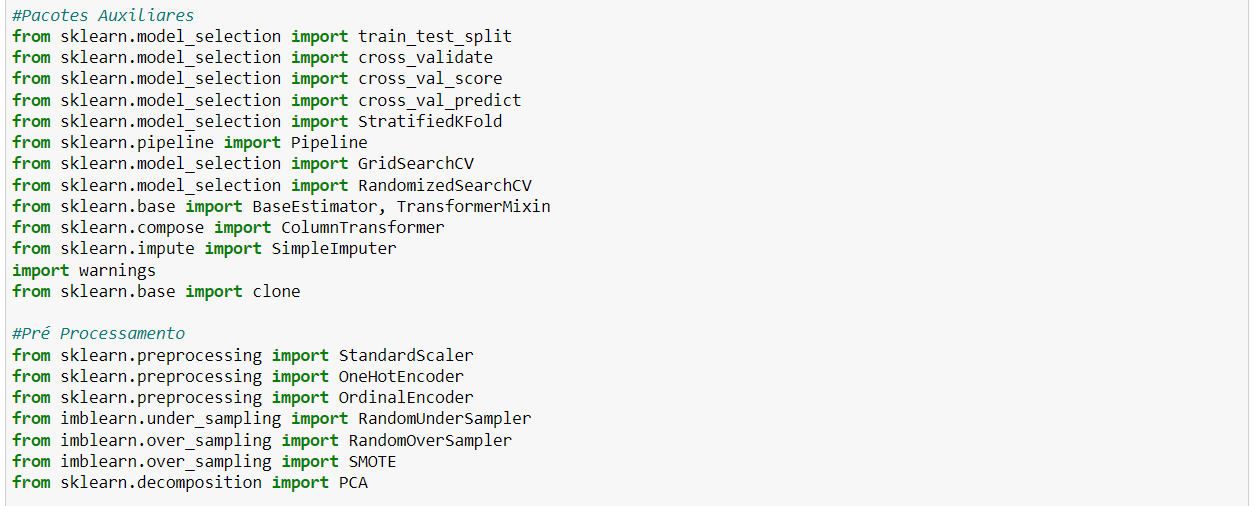
A base utilizada nesse projeto foi retirada do site de competições Kaggle, com link no capítulo de referências. Para a análise proposta, temos à disposição uma tabela com 3.333 linhas e 11 colunas, sendo essas:

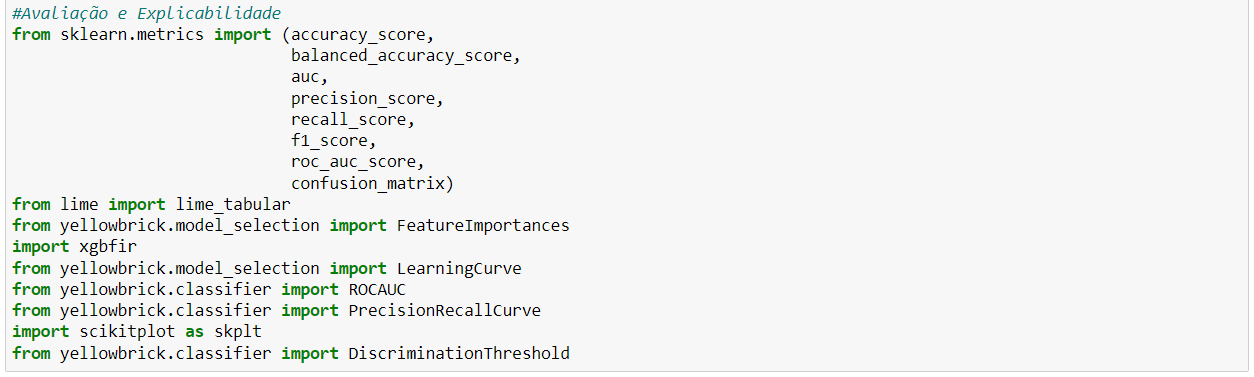
* **Churn**: *Flag* Indicando cancelamento de serviço (1 caso cliente tenha cancelado o serviço, 0 caso contrário).
* **AccountWeeks**: Número de semanas que cliente possui a sua conta ativa
* **ContractRenewal**: *Flag* indicando a renovação recente de contrato do cliente com a empresa (1 caso cliente tenha renovado seu contrato recentemente, 0 caso contrário)
* **DataPlan**: *Flag* referente a contratação de plano de dados pelo cliente (1 caso cliente tenha plano de dados, 0 caso contrário).
* **DataUsage**: *Gigabytes* de dados usados mensalmente.
* **CustSurvCalls**: Quantidade de chamadas para o *customer service*
* **DayMins**: Média de minutos em ligações por mês.
* **DayCalls**: Média de quantidade de ligações por dia
* **MonthlyCharge**: Média da fatura mensal
* **OverageFee**: Maior valor de pagamento acima do valor da fatura em 12 meses
* **RoamMins**: Média de minutos utilizando serviços de *roaming*

1. **Bibliotecas Importadas**

Foram importadas bibliotecas divididas em seções, conforme abaixo:







1. **Configurações e Funções Úteis**

Configuração básica para definição de estilo, paleta de cores e quantidades máximas de linhas e colunas exibidas:



Além disso, nessa etapa foram configuradas algumas funções, sendo essas:

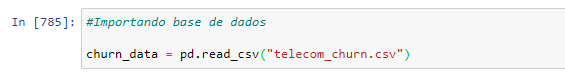
**analisa\_metricas\_cross\_val(clf, X\_train, y\_train, n\_folds):** Avalia modelo de classificação utilizando *cross\_validation* e exibe as médias das seguintes métricas por *n\_folds*: acurácia, acurácia balanceada, precisão, *recall*, F1-*Score* e Roc Auc *Score*. Além disso, função também exibe matriz de confusão em mapa de calor.

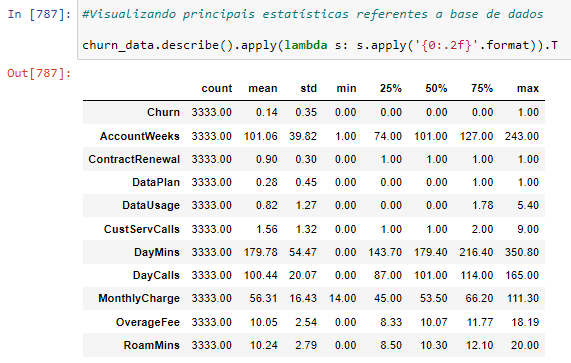
**analisa\_metricas\_treino\_teste(clf, X\_train, y\_train, X\_test, y\_test, n\_folds):** Avalia modelo de classificação utilizando *cross\_validation* e exibe as médias das seguintes métricas por *n\_folds*: acurácia, acurácia balanceada, precisão, *recall*, F1-*Score* e Roc Auc *Score*. Função também realiza previsões na base teste e avalia mesmas métricas que foram avaliadas no conjunto de treinamento. Por fim, exibe as matrizes de confusão de treino e teste em mapa de calor.

**treina\_avalia\_compara(clf\_dict, X\_train, y\_train, n\_folds):** Exibe a média de métricas por *n\_folds*, utilizando *cross\_validation*, em formato de *heatmap*, para cada modelo fornecido na variável clf\_dict.

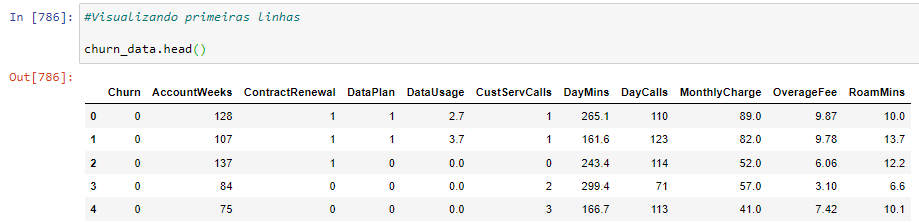
1. **Análise Inicial da Base de Dados**

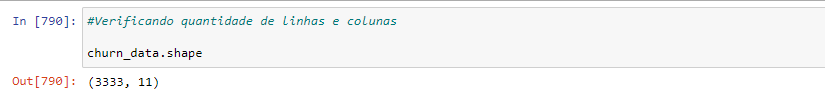
Nessa etapa, foi realizada a importação da base de dados através da função *read\_csv()*, e foram analisadas principais estatísticas referentes à base de dados através do método *.describe*.





Além disso, foram utilizados o método .*head()* e o atributo *.shape* para verificar a integridade da base.





Por fim, foi verificada a presença de atributos com valores faltantes através da função *msno.matrix()* e do método *.info(),* que também foi utilizado para identificar os tipos das variáveis analisadas.



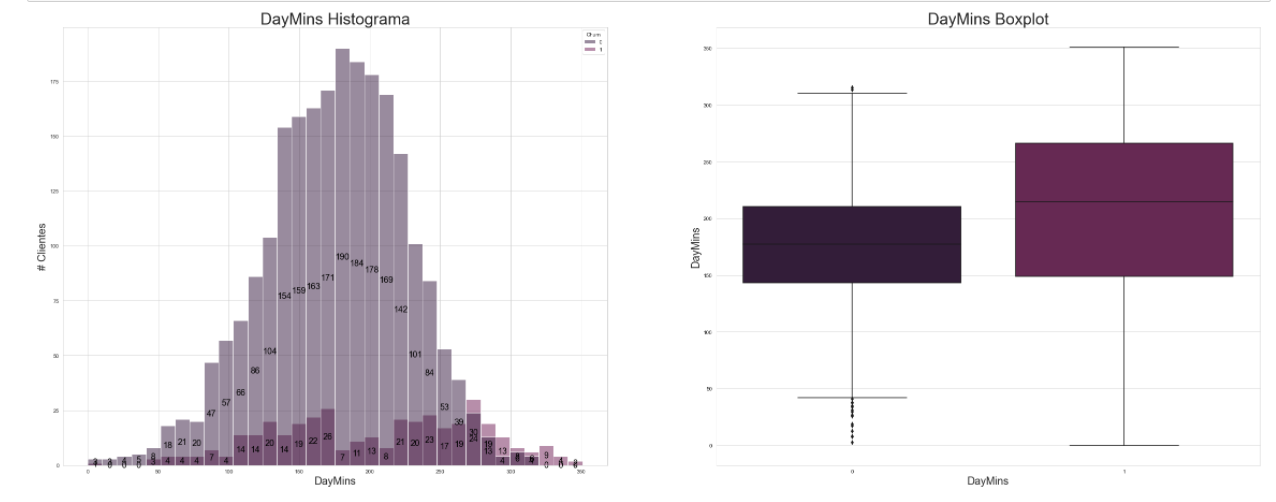


1. **Análise Exploratória**

Foi realizada análise gráfica com objetivo de estudar o comportamento dos atributos em relação a variável *target*. Para isso, inicialmente, foi feita a divisão em treino e teste com técnica de amostra estratificada pela variável *target*, e em seguida, foram realizados diversos gráficos, de acordo com cada tipo de variável (contínua, discreta ou categórica). Em seguida, foi analisada a correlação entre atributos e entre a variável *target* através da matriz de correlação. Por último, foi realizada análise bivariada dos principais atributos identificados e treinado modelo de árvore de decisão, pelo qual podemos identificar comportamento das principais variáveis no modelo e suas importâncias.

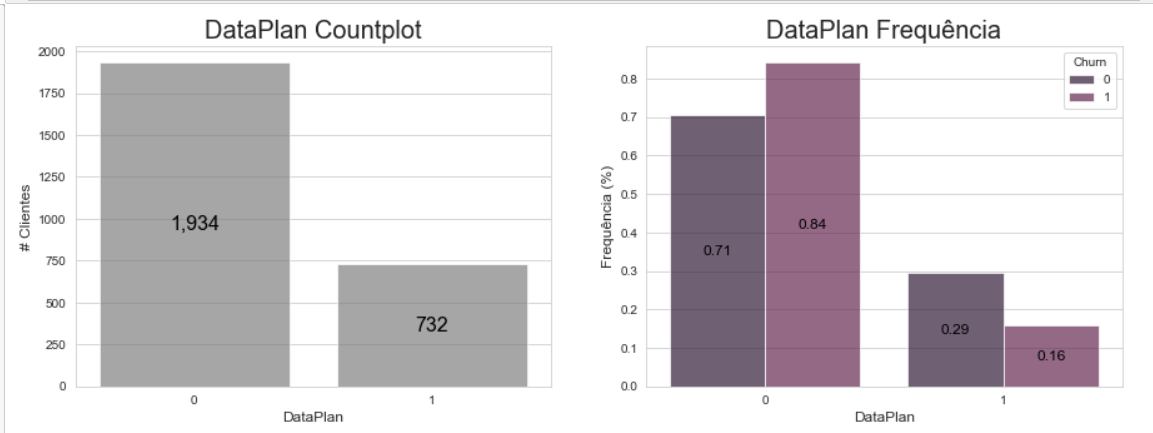
1. **Variáveis Contínuas**

Foi analisado comportamento de cada atributo através de gráficos de histograma e *boxplot*, com separação entre as classes da variável *target* (*Churn*=0 ou *Churn*=1). A seguir, segue *layout* adotado para todos os atributos contínuos, exemplificado através do atributo *DayMins*:



1. **Variáveis Categóricas**

Foi analisado o comportamento das variáveis através de histograma e gráfico de frequência, onde cada cor é representada por uma das classes da variável *target* (*Churn*=0 ou *Churn*=1). Abaixo, temos exemplo de layout adotado para variáveis categóricas através de atributo *DataPlan*.



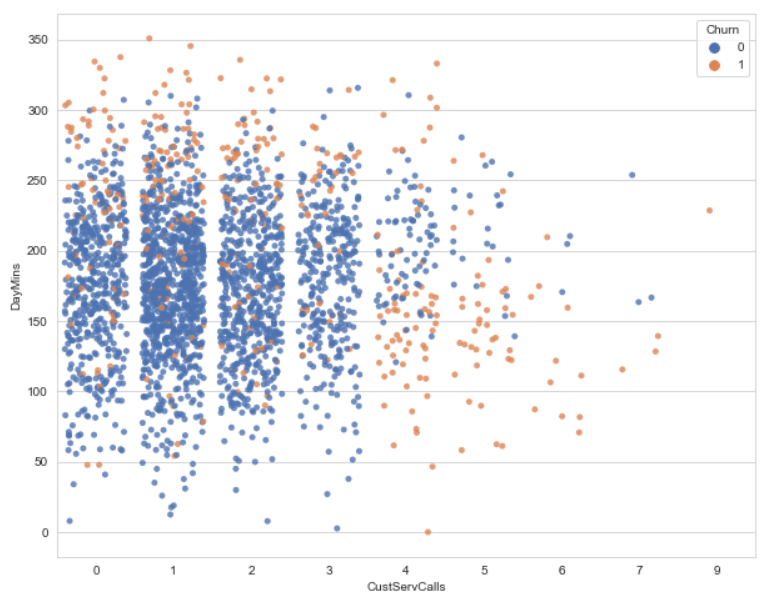
1. **Análise de Correlação**

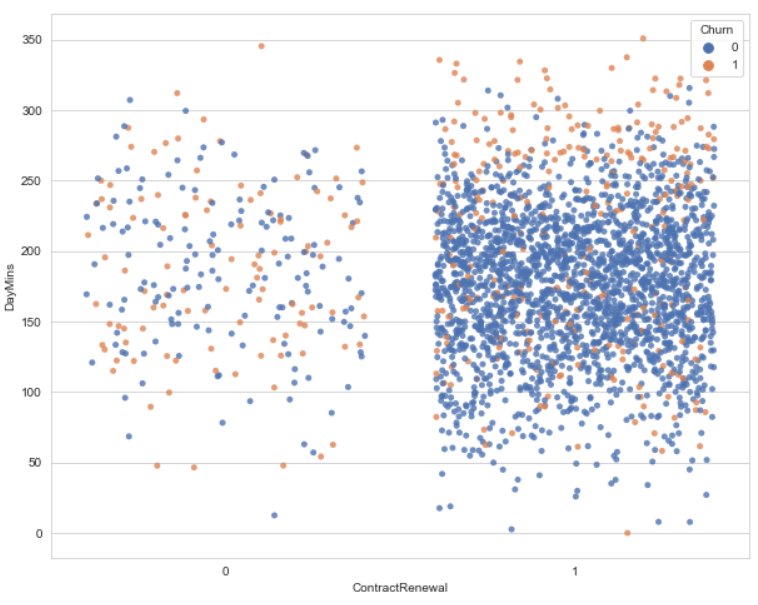
Foi analisada matriz de correlação utilizando método *Spearman* e foram identificados os atributos mais correlacionadas com variável *target*, dentre os quais podemos destacar: *ContractRenewal, CustServCalls, DayMins e DataPlan*. Além disso, também foram identificadas variáveis altamente correlacionadas entre si como é o caso dos atributos *MonthlyCharge-DataPlan e DataUsage-MonthlyCharge*.



1. **Análise Bivariada**

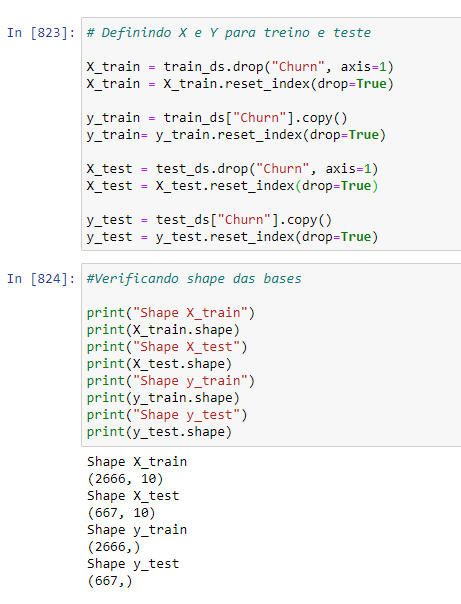
Na análise bivariada, foi utilizada a função .*stripplot* da biblioteca *seaborn* para plotar, nos eixos, as variáveis mais correlacionadas com o *Churn* com diferenciação de cor para cada classe da variável *target*. Podemos observar algumas possíveis formas de separar os dados, onde temos, aparentemente maiores concentrações de *Churn*=1 e menores concentrações de *Churn*=0, como é o caso para *DayMins* menores que 150 e *CustServCalls* maiores que 3, no primeiro gráfico abaixo:





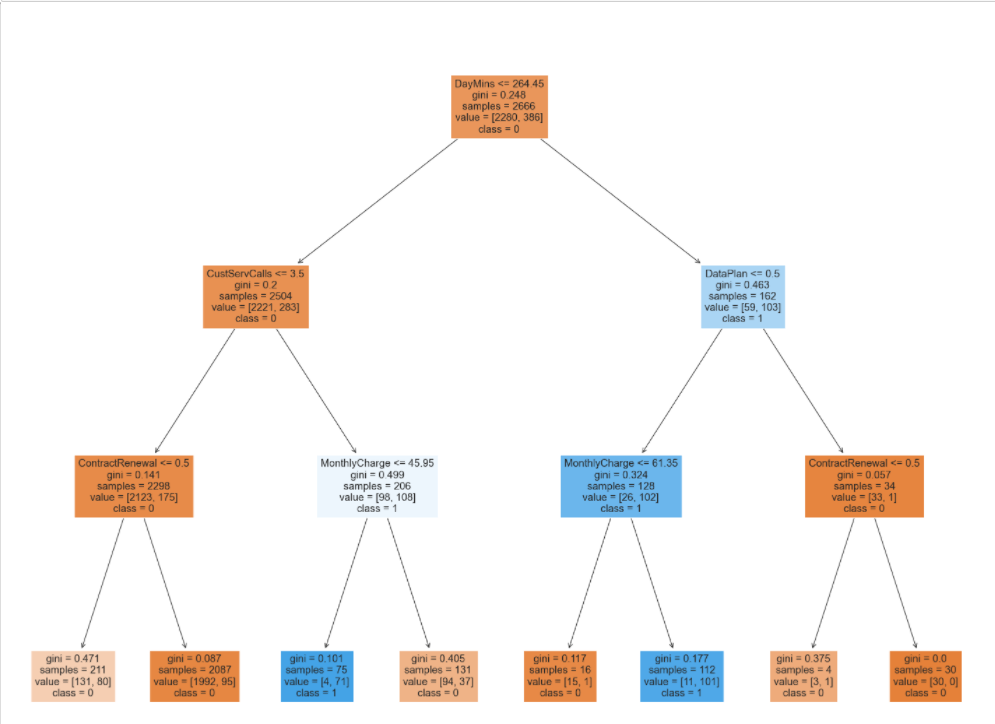
1. **Separação em X e y**

Foi realizada a separação entre variável *target* e atributos para treino e teste, para seguir com pré-processamento e treinamento/avaliação de modelos.

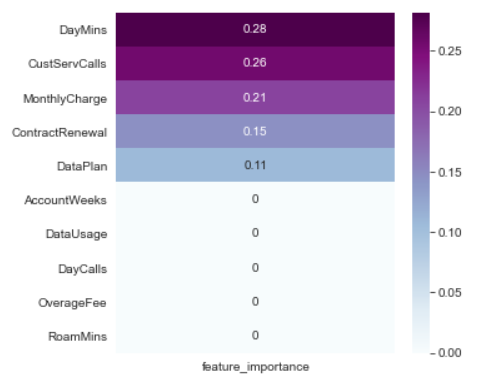


1. **Treinando Árvore de Decisão**

Foi treinado modelo de árvore de decisão com *max\_depth*=3, como forma de identificar os principais atributos e regras definidas pelo algoritmo. Depois de treinado o algoritmo, modelo foi exibido através da função *tree.plot\_tree.* É possível observar algumas análises interessantes, como é o caso da relação previamente destacada entre as variáveis *DayMins* e *CustServCalls*:



Em seguida, foi verificada a importância de cada variável através do atributo *.feature\_importances\_*. Previamente, todas as variáveis foram observadas entre as maiores taxas de correlação com o *target*, exceto *MonthlyCharge*, que apareceu como terceira variável mais importante no modelo de árvore de decisão *:*



Finalmente, avaliamos o modelo em treino e teste, utilizando a função criada analisa\_metricas\_treino\_teste. É importante destacar, que neste momento, não estamos preocupados com maximização das medidas de avaliação e sim com a obtenção de insights a partir do treinamento de modelo de árvore de decisão com baixa profundidade (*max\_depth*=3).

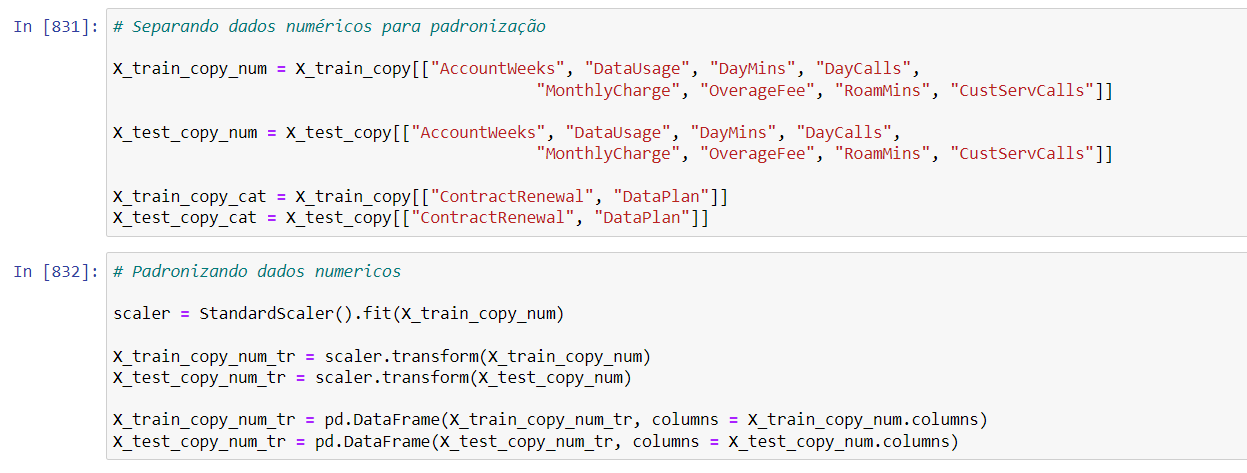


1. **Pré-Processamento**

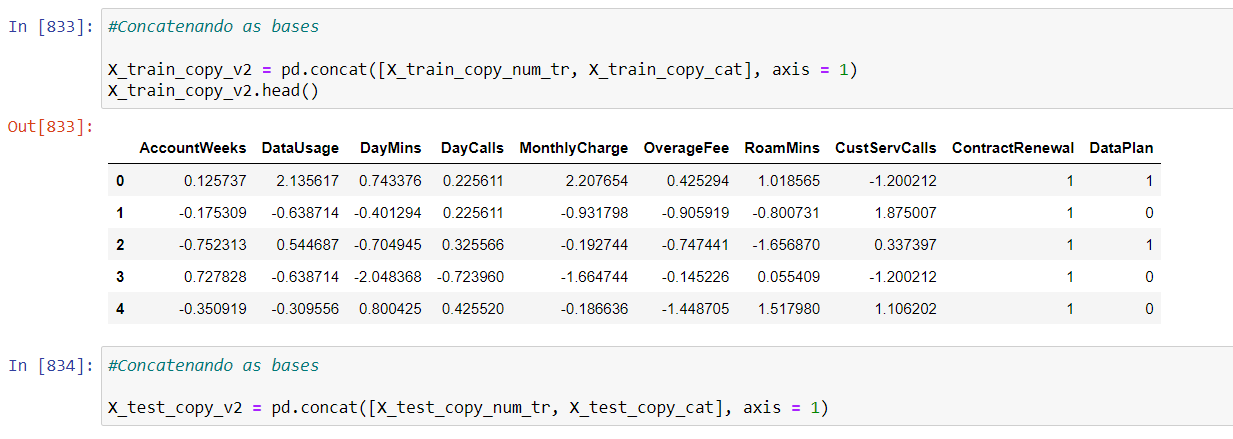
Para o pré-processamento, foi realizada a padronização de variáveis e técnicas de *resample* como forma de melhorar o desempenho de modelos treinados posteriormente. Não houve necessidade de transformação de variáveis textuais ou definição de estratégia para valores faltantes.

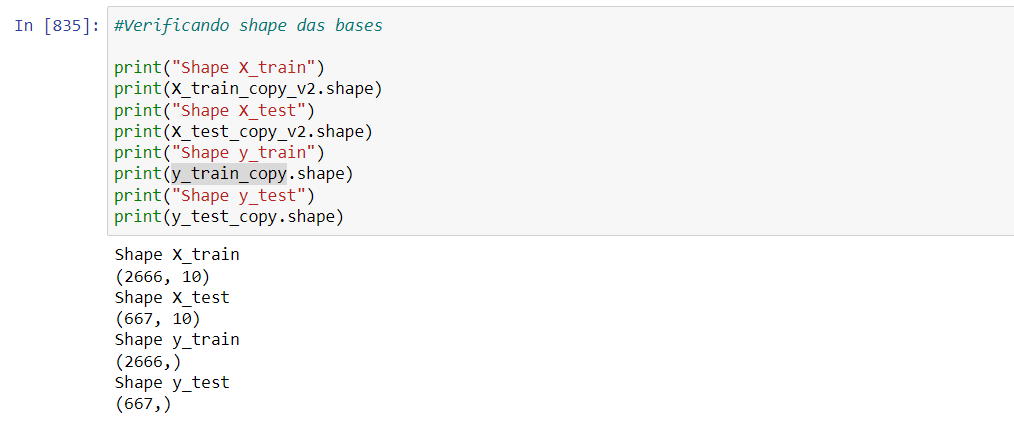
* 1. **Padronização dos Dados**

Inicialmente, foi feita a separação dos atributos a serem padronizados para, em seguida, efetuar a padronização.



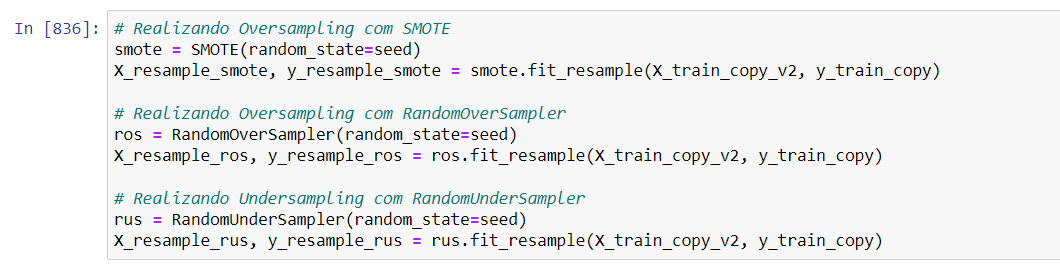
Enfim, foi realizada a concatenação das bases de treino e teste e verificado .*shape* das bases:



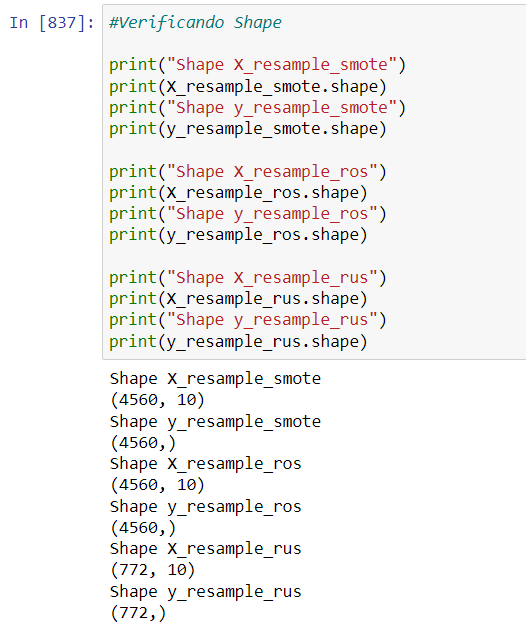


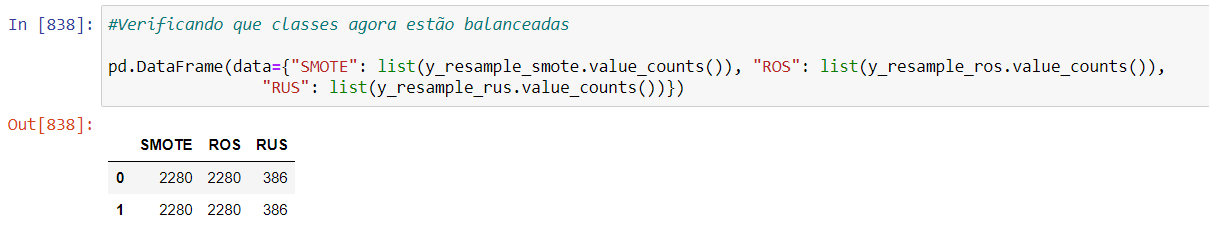
* 1. **Utilizando Técnicas de *Resample***

Nessa etapa, foram utilizadas três técnicas de *resample* para verificar qual delas é mais adequada para problema atual. Dessa forma, foram criadas bases de treino para X e y, utilizando a técnica de *resample* *SMOTE*, *RandomOverSampler* e *RandomUnderSampler*.

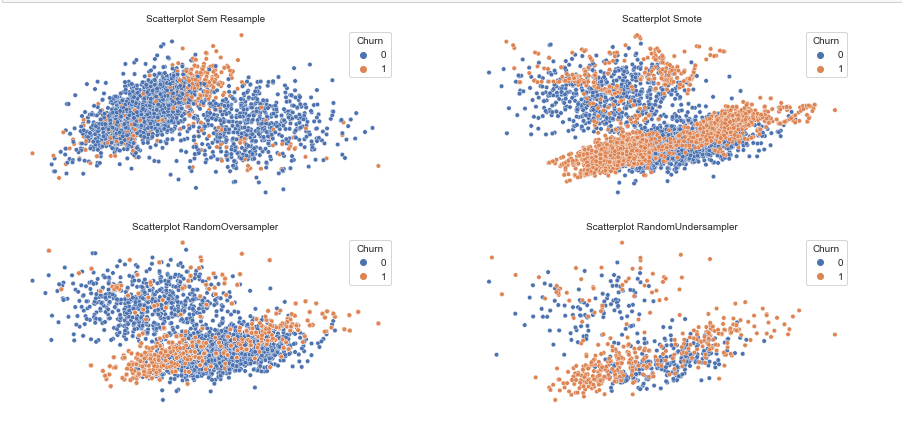


Em seguida, foi verificado o .*shape* de cada uma das bases e as contagens de cada classe da variável *target*, utilizando o método .*value\_counts*





Finalmente, visualizamos os dados em duas dimensões, utilizando *PCA*=2, para verificarmos cada base com suas classes balanceadas:



1. **Escolha de Métricas para Avaliação de Modelos**

Para avaliação dos modelos adotados partiremos das seguintes ações e premissas:

**Possíveis Ações:**

* **Ação A -** **(Cliente previsto como Churn=1 pelo modelo):** Oferecer pacote de benefícios e consultoria especial, dando maior atenção ao cliente, como forma de estratégia para que cliente cancele o plano com a empresa.
* **Ação B - (Cliente previsto como Churn=0 pelo modelo):** Não entrar em contato com cliente.

**Premissas:**

* **Premissa 1:** Caso o cliente cancele o produto, a empresa deixará de ter a receita pelos n meses subsequentes
* **Premissa 2:** Custo de saída da empresa>>Custo de Pacote de benefícios + Consultoria especial.

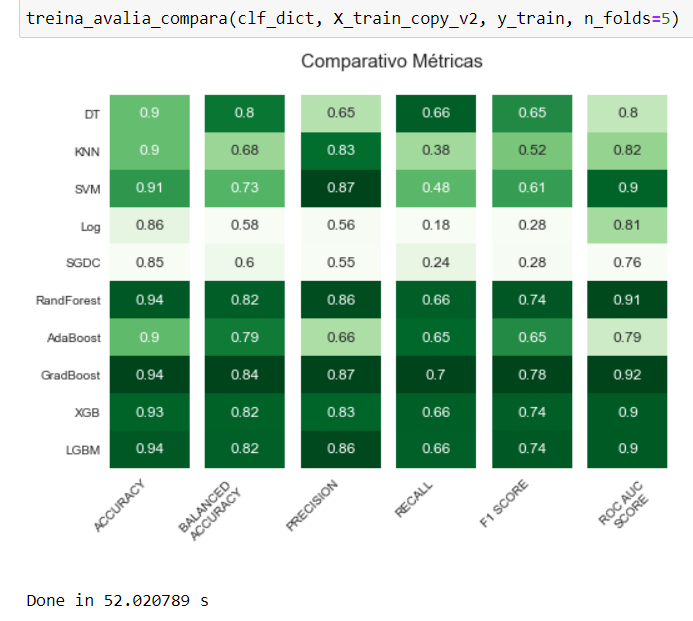
Dessa forma, de acordo com as premissas e possíveis ações estabelecidas, adotaremos o *recall* como a principal métrica a ser avaliada, priorizando assim, a redução de falsos negativos em detrimento do aumento de falsos positivos. Apesar disso, é importante a análise de outras métricas em conjunto com o *recall*, como a acuraria e o F1 *score*, de forma a ter maior qualidade no modelo.

1. **Treinando e Avaliando Modelos de *Machine Learning***

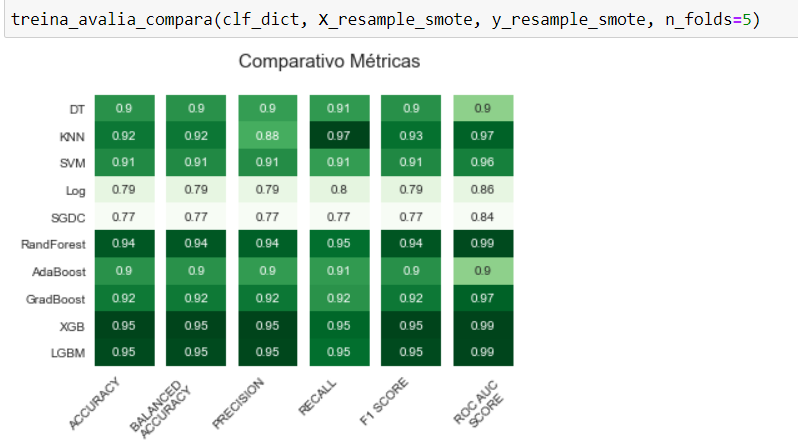
Foram avaliados os modelos destacados na imagem abaixo para cada uma das três bases, definidas na etapa de pré-processamento. O objetivo dessa etapa é, não apenas definir a base a ser utilizada, como também definir modelos promissores, que mais tarde serão avaliados também na base de teste. Para essa etapa, foi utilizada a função criada treina\_avalia\_compara com *n\_folds*=5.

1. **Avaliação Inicial de Modelos**

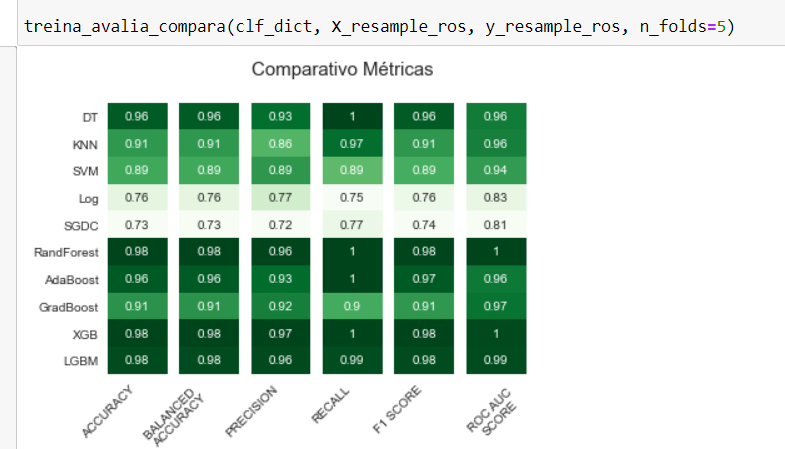
Base sem *Resample*:



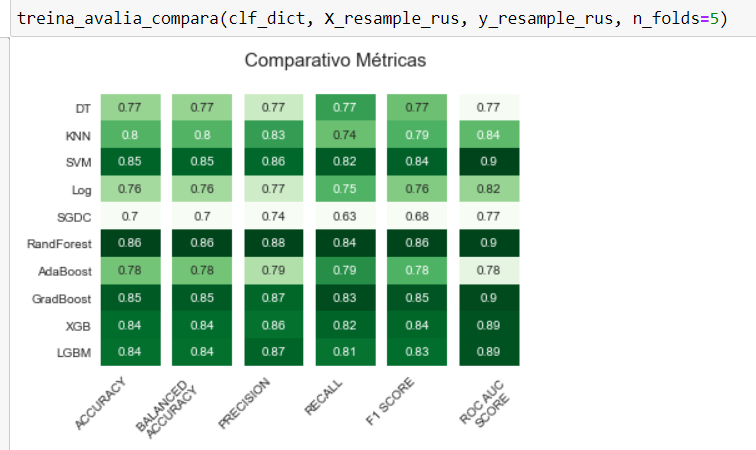
Base com *Resample SMOTE*



Base com *Resample RandomOverSampler*



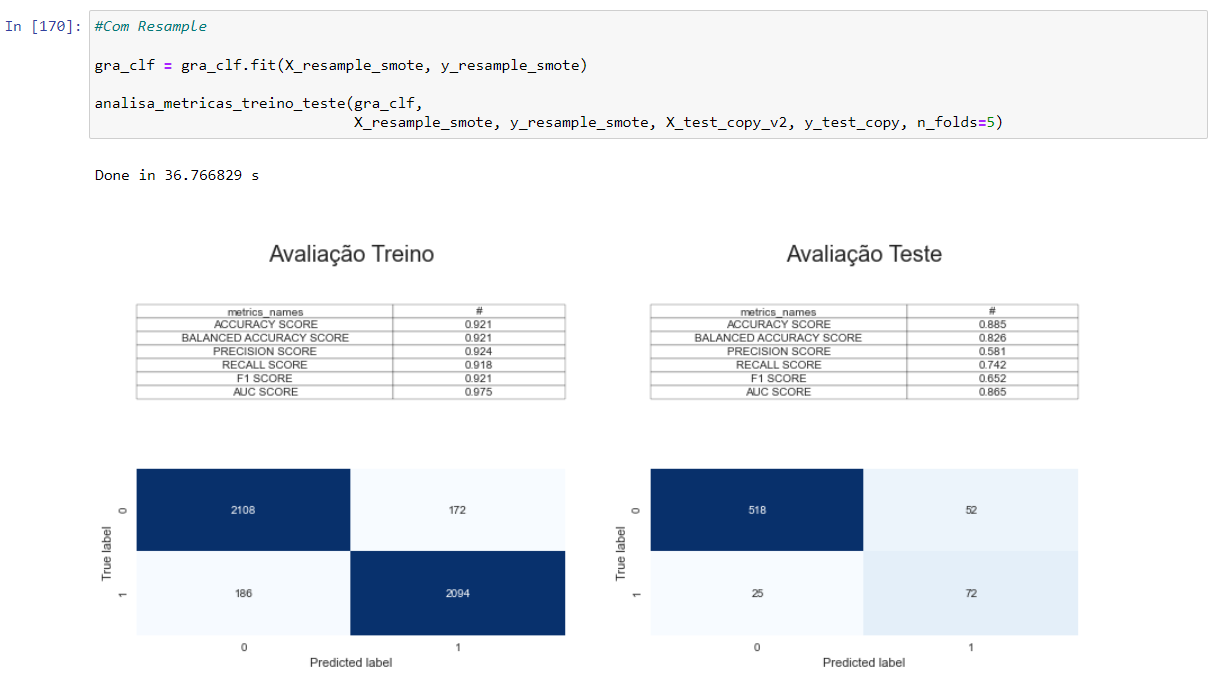
Base com RandomUnderSampler:



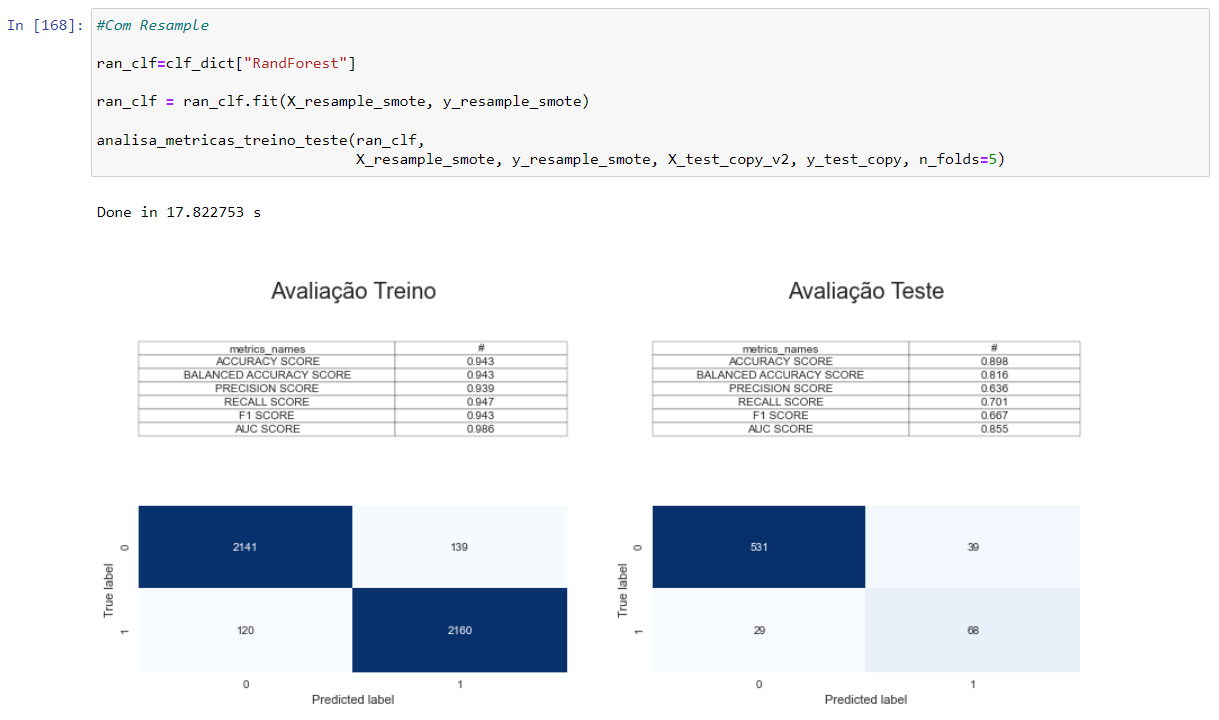
1. **Comparativo de Bases Utilizadas (*Overfit*)**

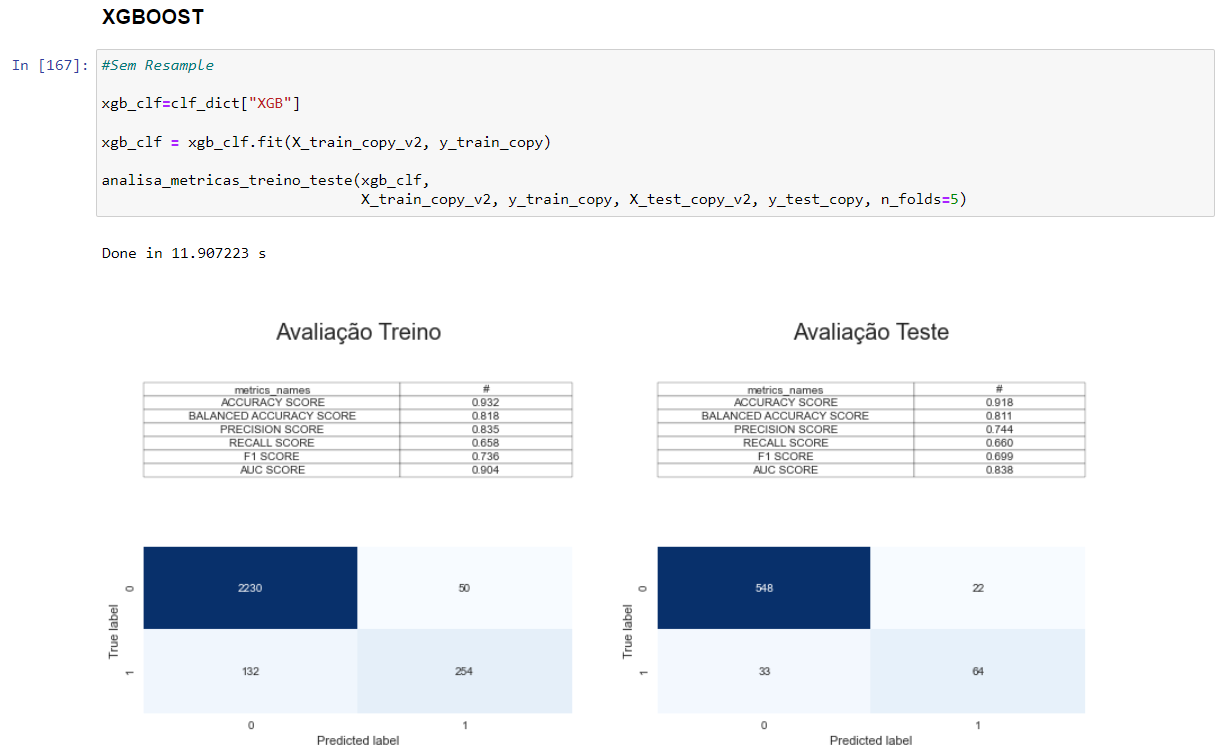
Posteriormente, para os modelos mais bem avaliados, será avaliado o comportamento no treino e teste para verificar se temos *overfit* e, finalmente, decidirmos a respeito sobre qual base iremos utilizar para treinar os modelos.



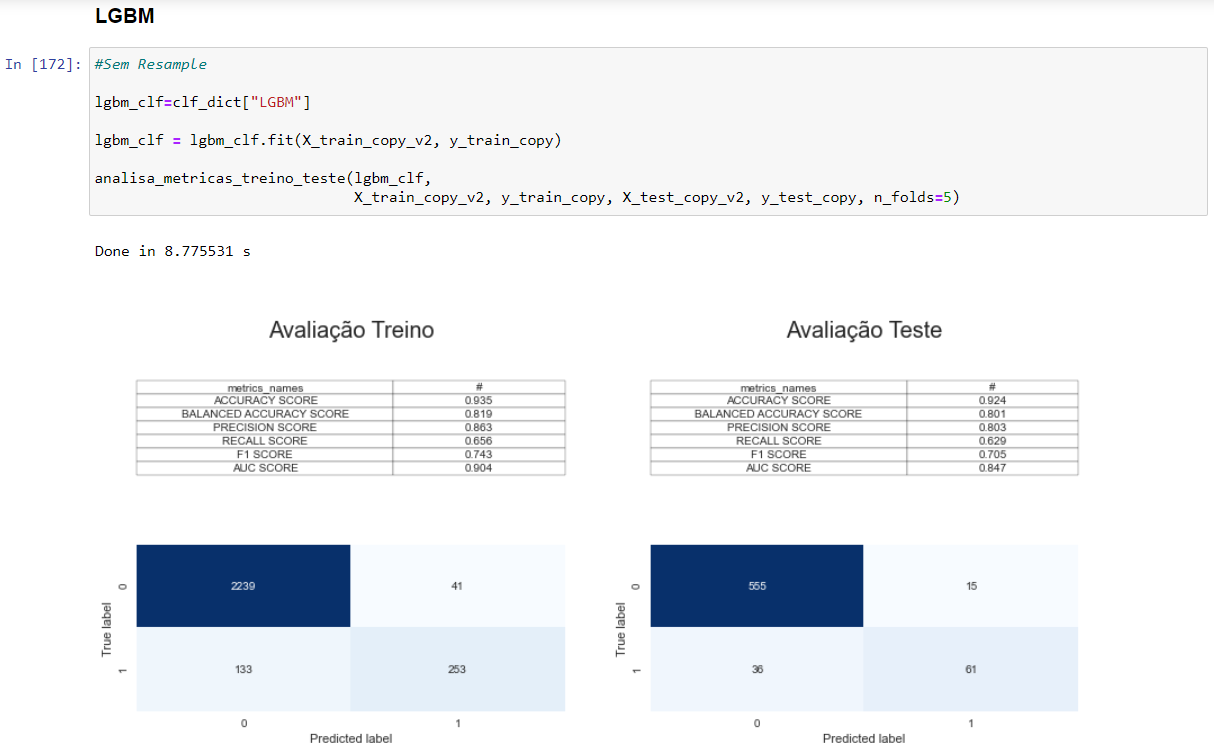


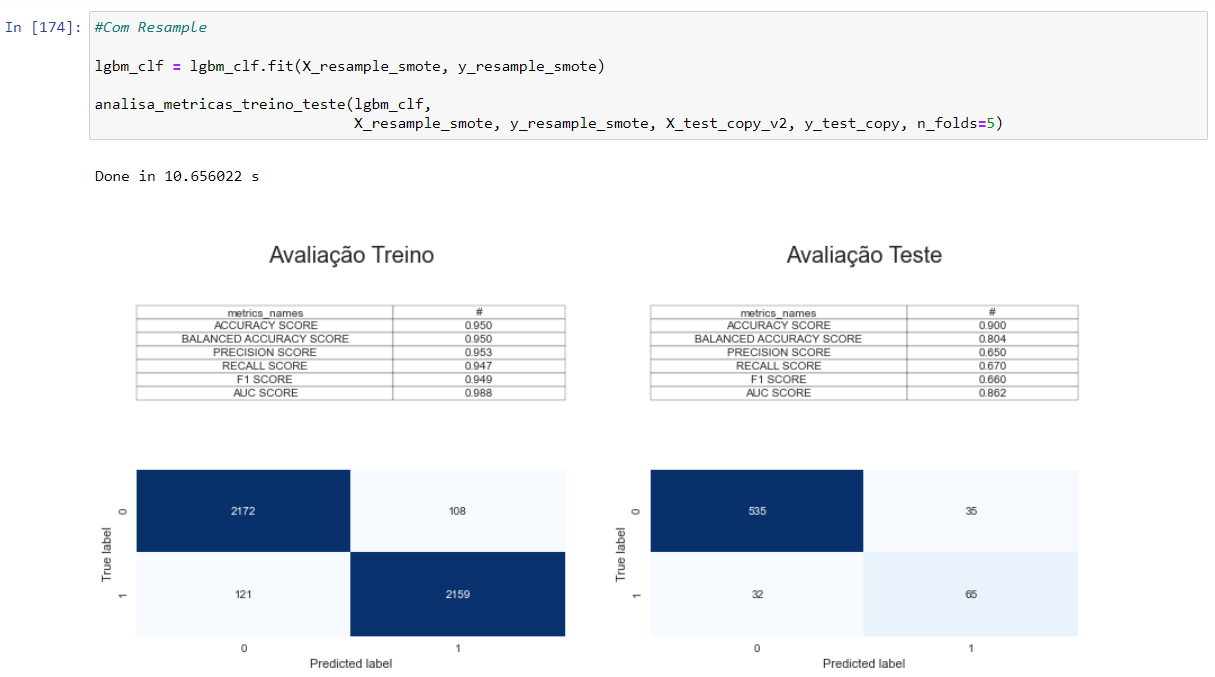










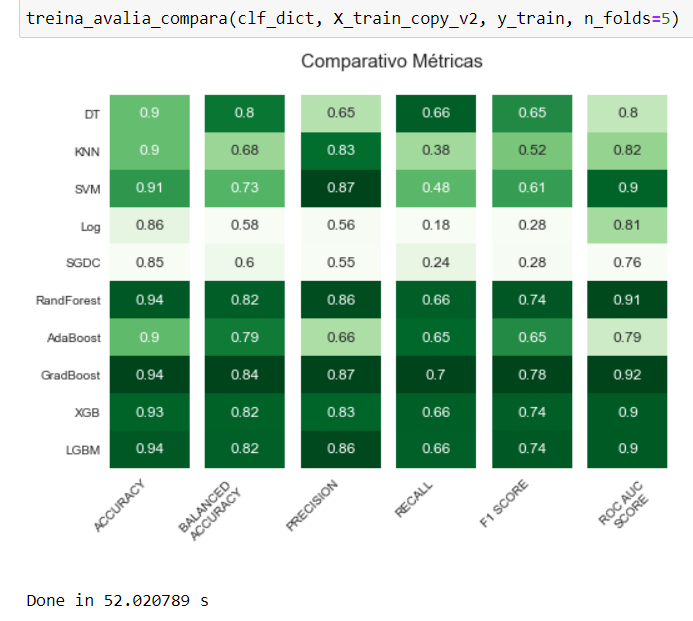


Para os passos seguintes, foi definido que não será utilizado *SMOTE*, visto que tivemos na base sem *resample*, para um recall semelhante, maior acurácia e F1-*Score*.

1. **Avaliação e Seleção de Modelos Promissores**

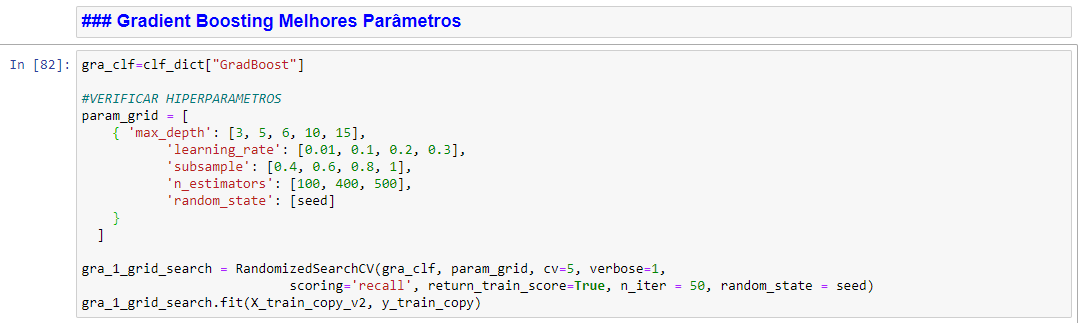
Analisando a tabela abaixo, é possível avaliar que os melhores resultados foram percebidos nos modelos: Gradient Boost, *Random Forest*, *XGBoost* e *LightGBM*. A seguir será realizado: o *tuning* de hiperparâmetros, análise dos dados na base de teste para verificar possíveis overfitting e, caso necessário, regularização de modelos.

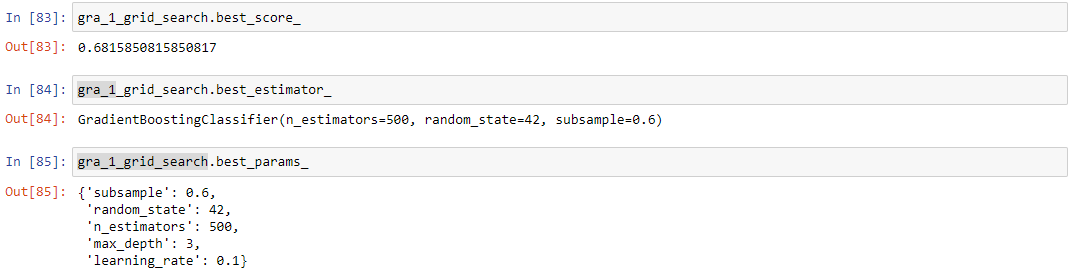
Também foi avaliado o modelo de *SVM*, como tentativa de avaliar o desempenho utilizando um *kernel* diferente.

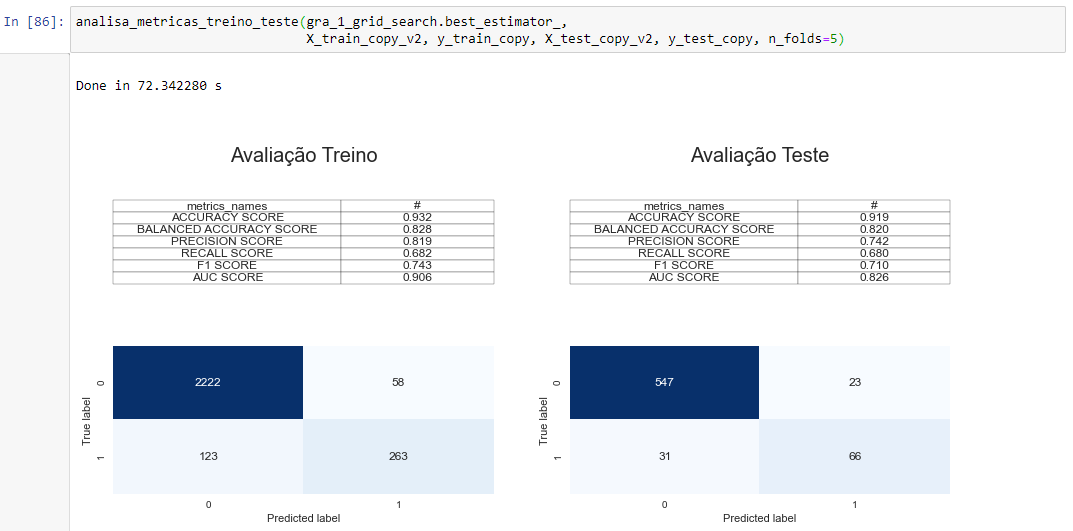


1. **Gradient Boost**

Inicialmente, foram analisados diversos hiperparâmetros, através da função *RandomizedSearchCV* com *n\_folds*=5 e *scoring*=’*recall’*. Para avaliação de cada modelo, utilizamos a função criada analisa\_metricas\_treino\_teste



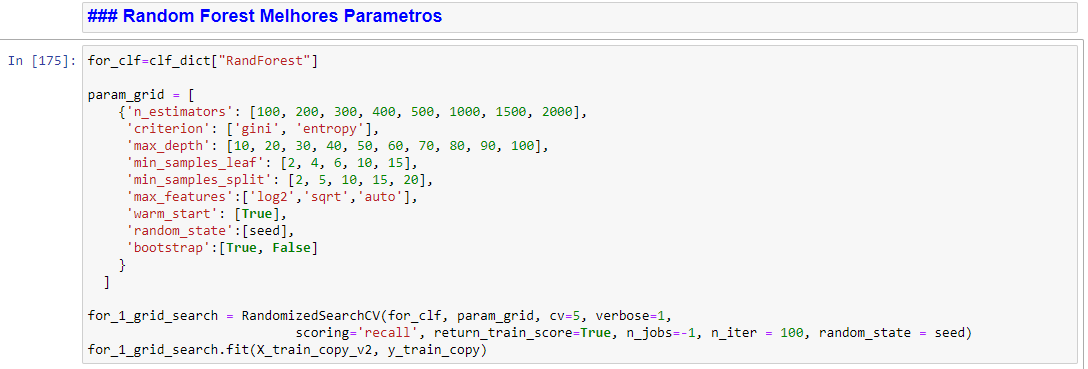


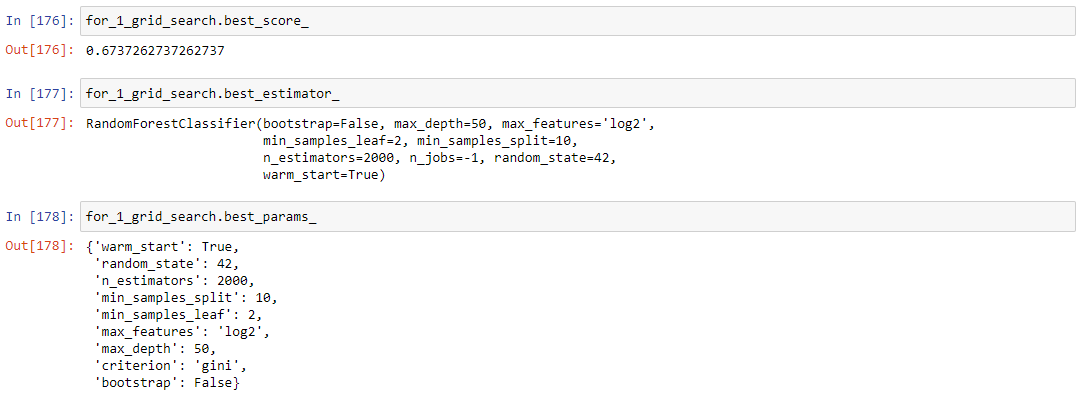


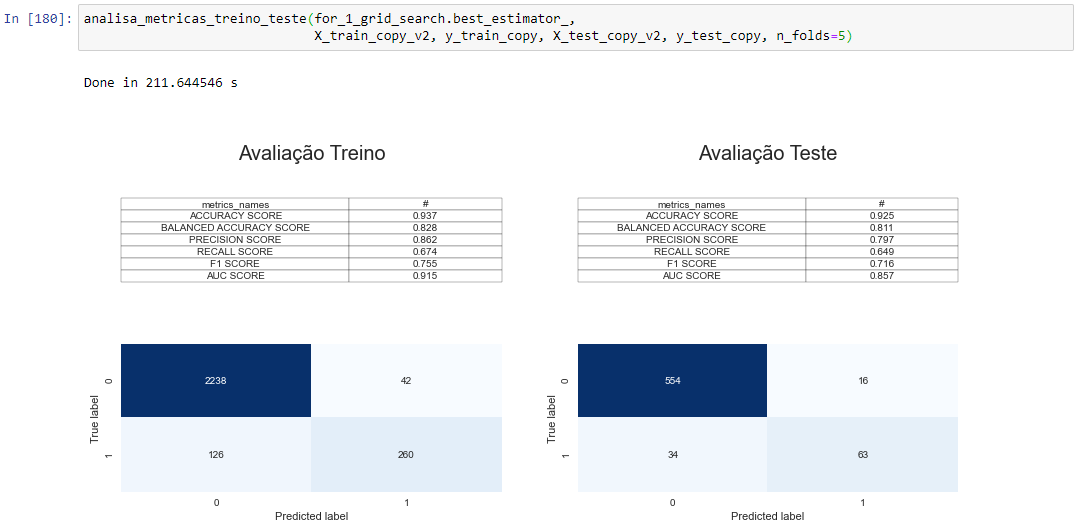
1. **Random Forest**

Foram analisados diversos hiperparâmetros, através da função *RandomizedSearchCV* com *n\_folds*=5 e *scoring*=’*recall’*. Para avaliação de cada modelo, utilizamos a função criada analisa\_metricas\_treino\_teste.

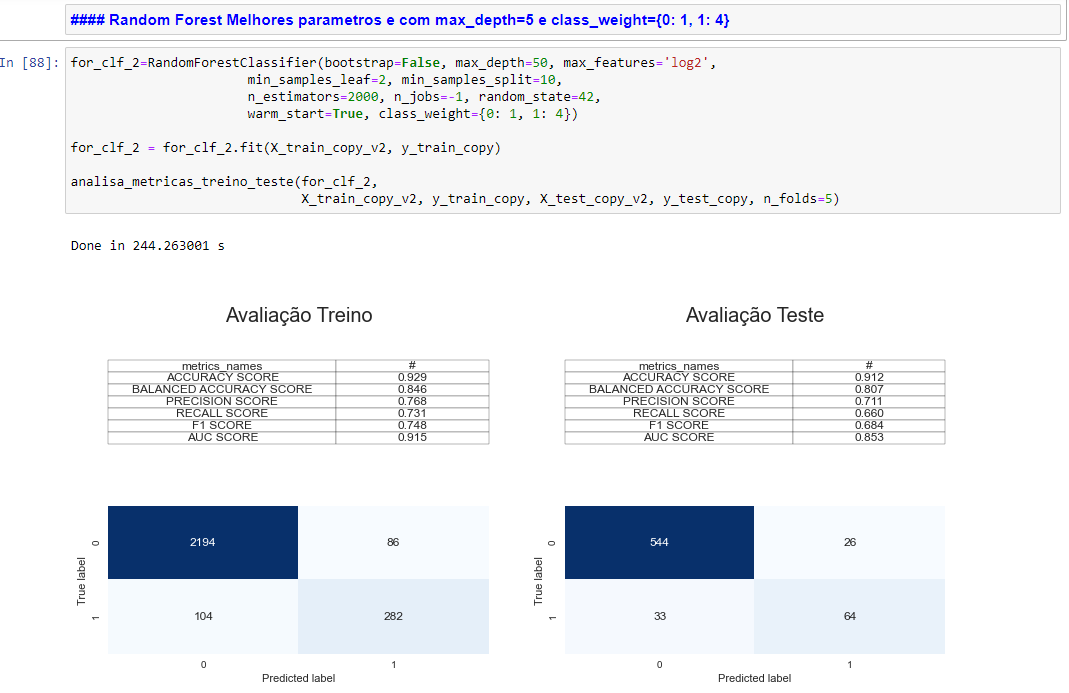
Além disso, foi treinado modelo utilizando diferentes pesos para a classe positiva da variável *target*, como forma de melhorar o *recall*. Através disso, foi percebido aumento discreto no *recall* e redução na acurácia e F1-*Score.*



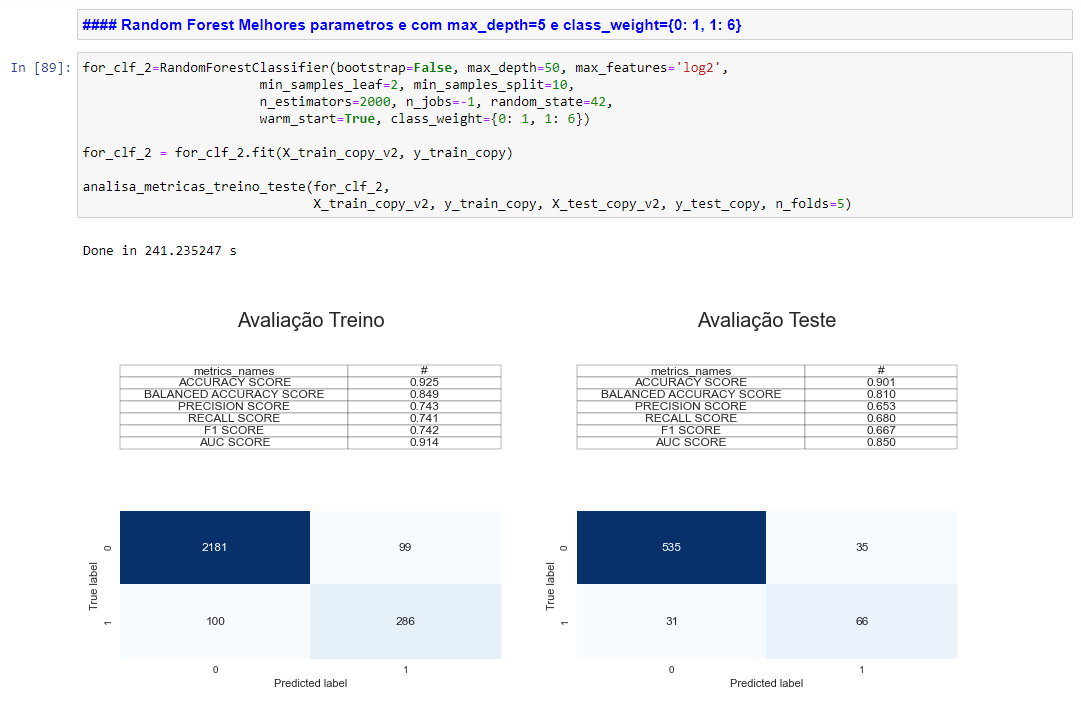




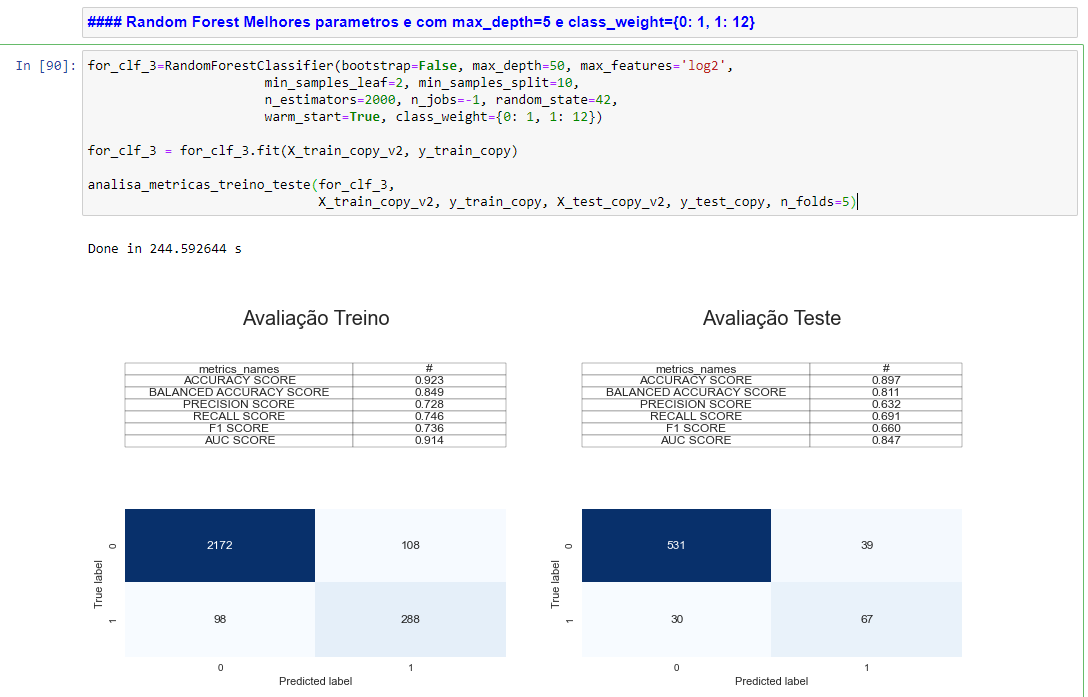
Random Forest com melhores parâmetros e *class\_weight*={0:1, 1:4}



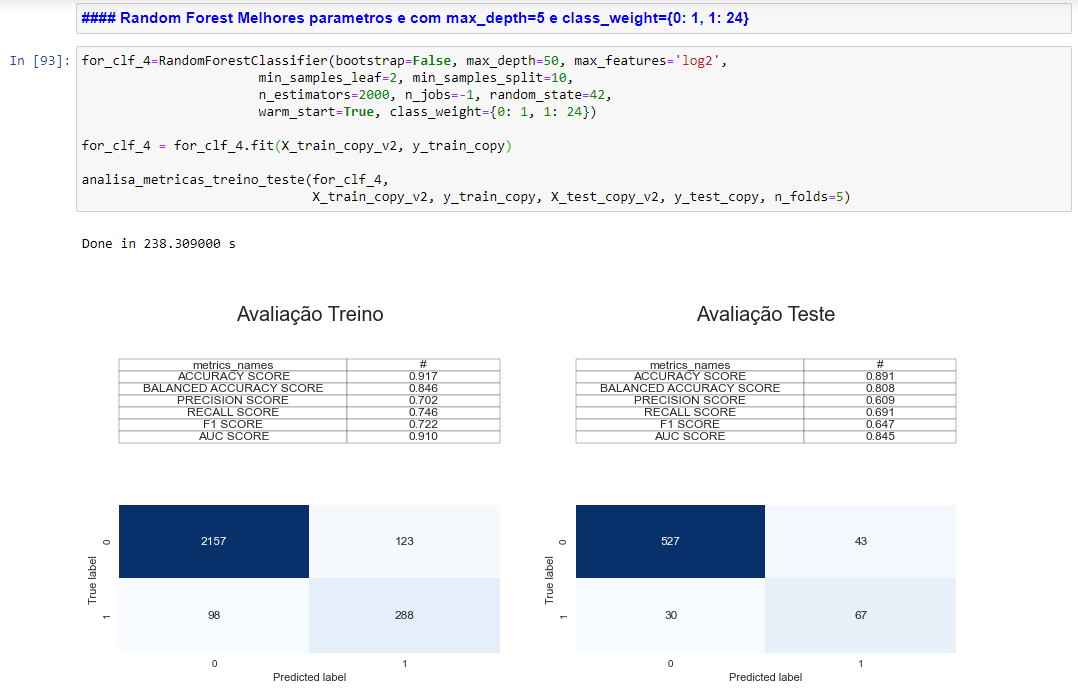
Random Forest com melhores parâmetros e *class\_weight*={0:1, 1:6}



Random Forest com melhores parâmetros e *class\_weight*={0:1, 1:12}

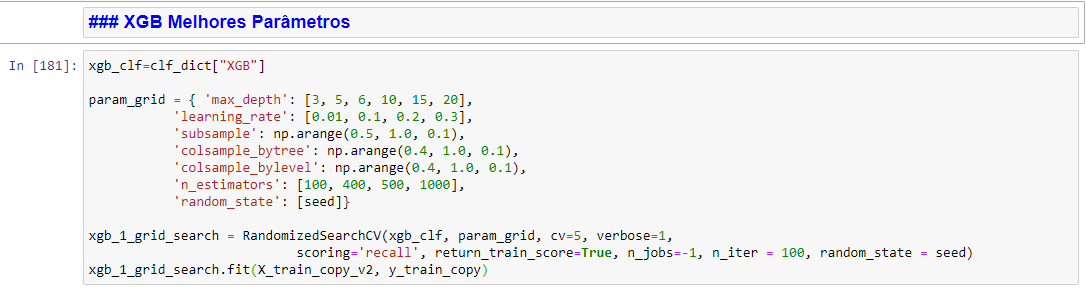


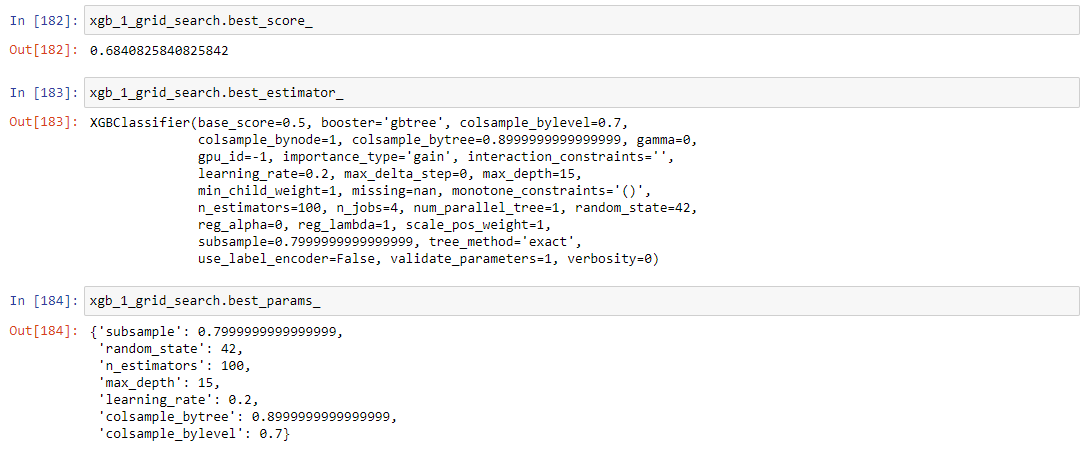
Random Forest com melhores parâmetros e *class\_weight*={0:1, 1:24}

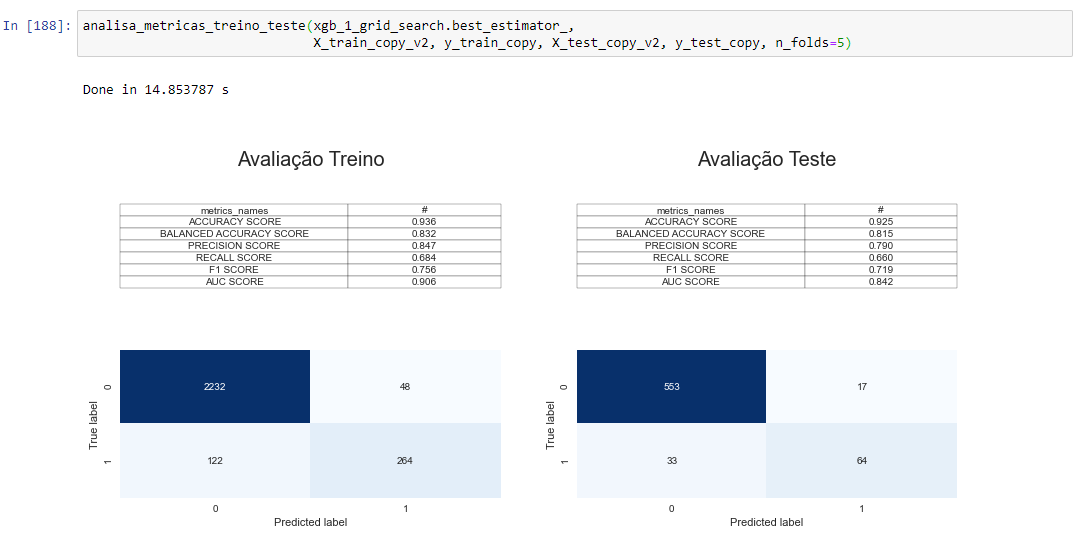


1. **XGBoost**

Foram analisados diversos hiperparâmetros, através da função *RandomizedSearchCV* com *n\_folds*=5 e *scoring*=’*recall’*. Para avaliação de cada modelo, utilizamos a função criada analisa\_metricas\_treino\_teste.

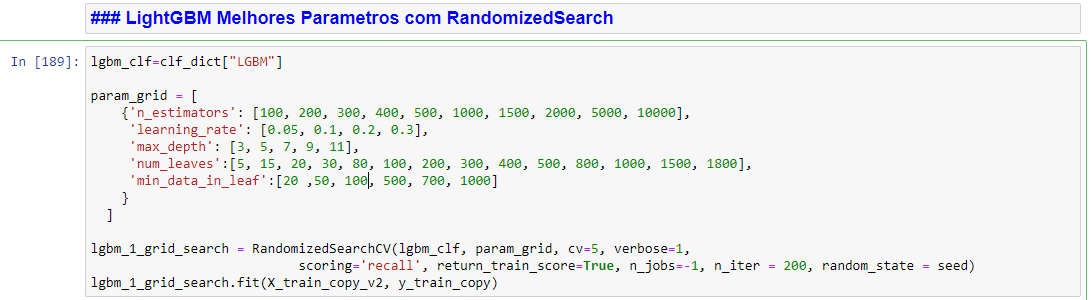




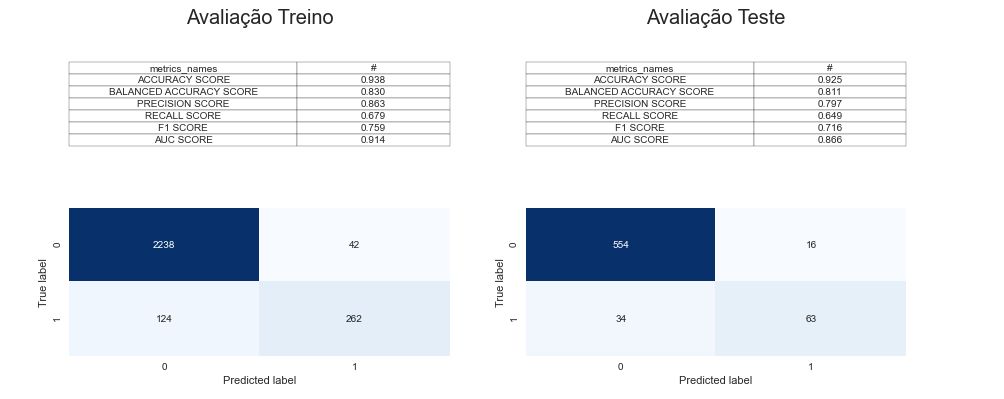


1. **LightGBM**

Foram analisados diversos hiperparâmetros, através da função *RandomizedSearchCV* com *n\_folds*=5 e *scoring*=’*recall’*. Para avaliação de cada modelo, utilizamos a função criada analisa\_metricas\_treino\_teste.

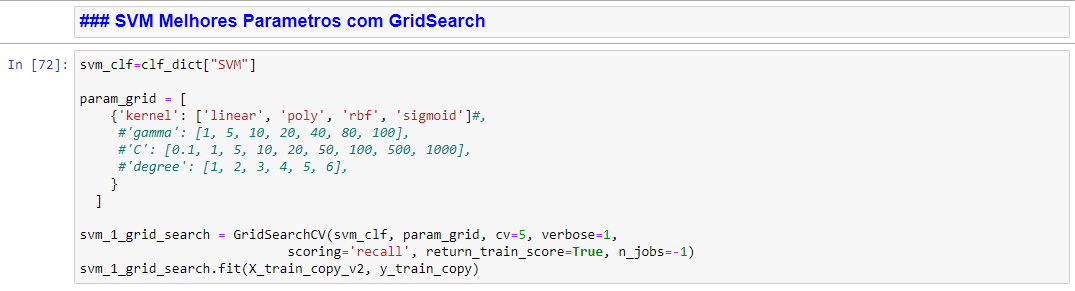


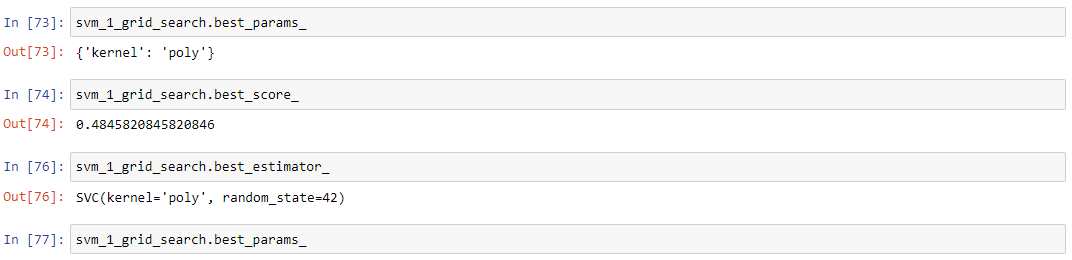


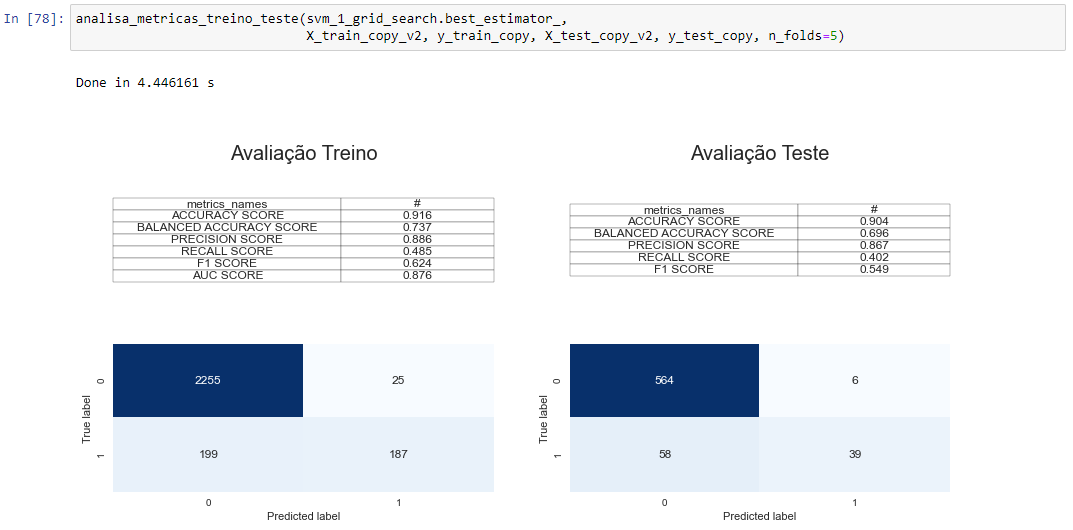


1. **SVM**

Foram analisados diversos hiperparâmetros, através da função *GridSearchCV* com *n\_folds*=5 e *scoring*=’*recall’*. Para avaliação de cada modelo, utilizamos a função criada analisa\_metricas\_treino\_teste.

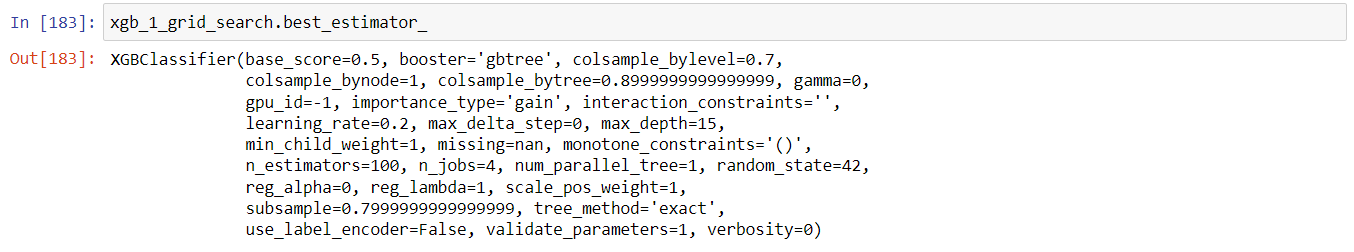


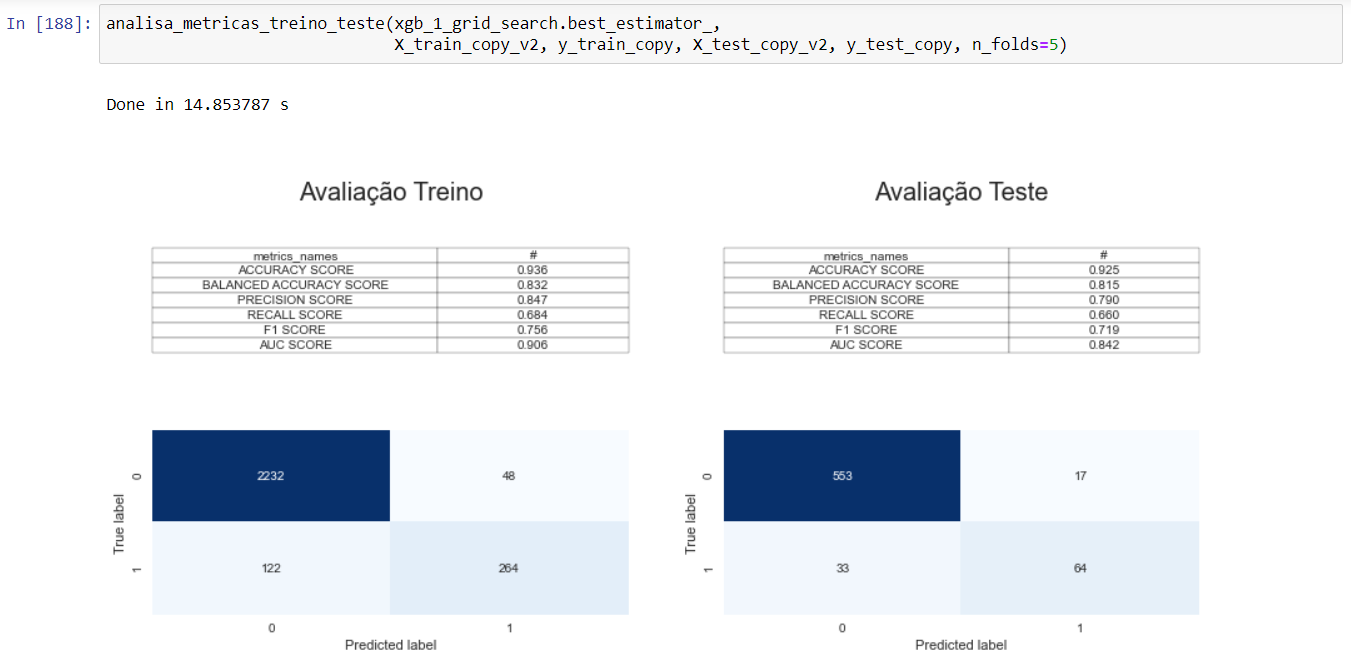




1. ***Insights* do Modelo e Avaliação de Melhor Fronteira de Decisão**

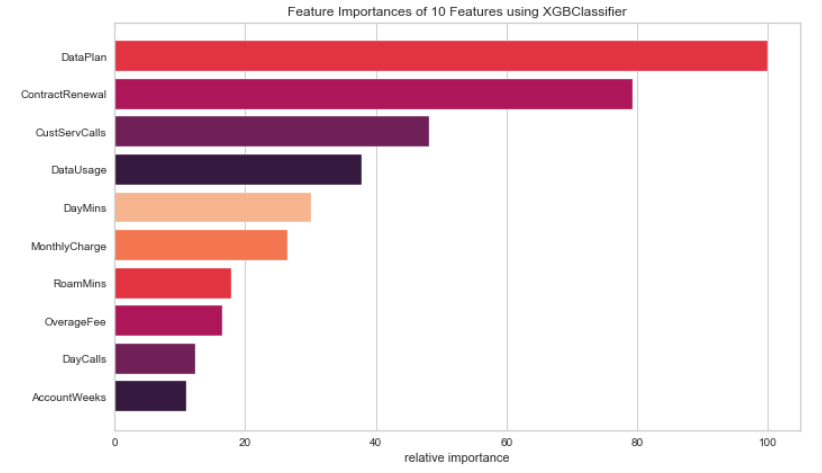
O modelo campeão, escolhido com base nas ações e premissas adotadas foi o ‘xgb\_1\_grid\_search.best\_estimator\_’, pois além possuir uma alta taxa de *recall*, também possui acurácia e F1 Score altos, comparado com o desempenho de outros modelos. Abaixo, segue a visão das métricas e matriz de confusão do modelo escolhido através da função analisa\_metricas\_treino\_teste. Mais tarde, iremos avaliar possível alteração da fronteira de decisão (*threshold*) como forma de melhorar o recall, visto que é a principal medida de avaliação para o problema observado.





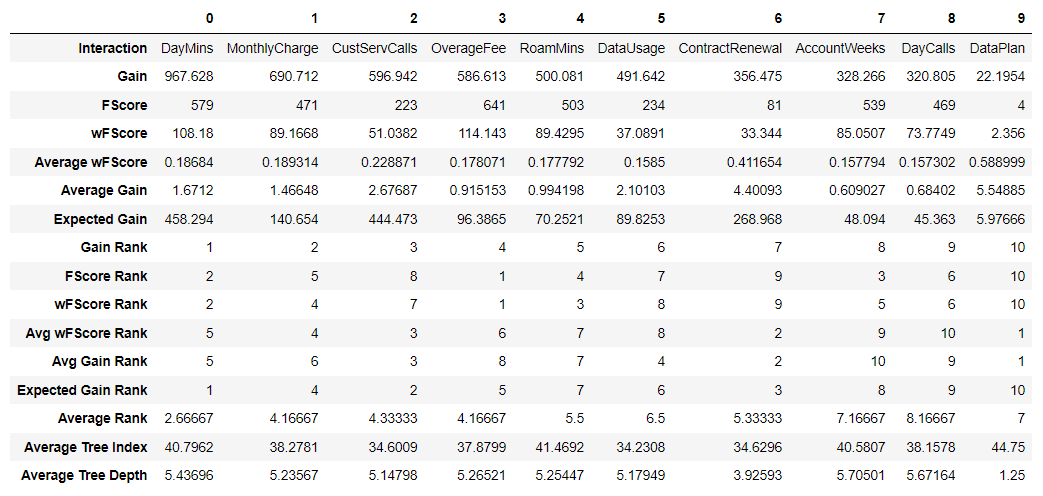
1. **Importância dos Atributos**

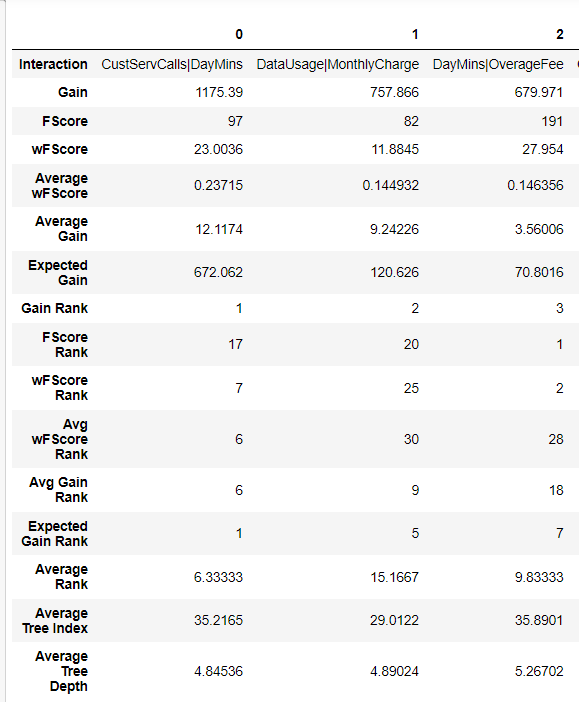
Foi utilizada a função *FeatureImportance* para exibir gráfico de importância normalizado de cada uma das variáveis. Vale destacar a diferença entre as importâncias das duas primeiras variáveis, *DataPlan* e *ContractRenewal*, em comparação com as demais.

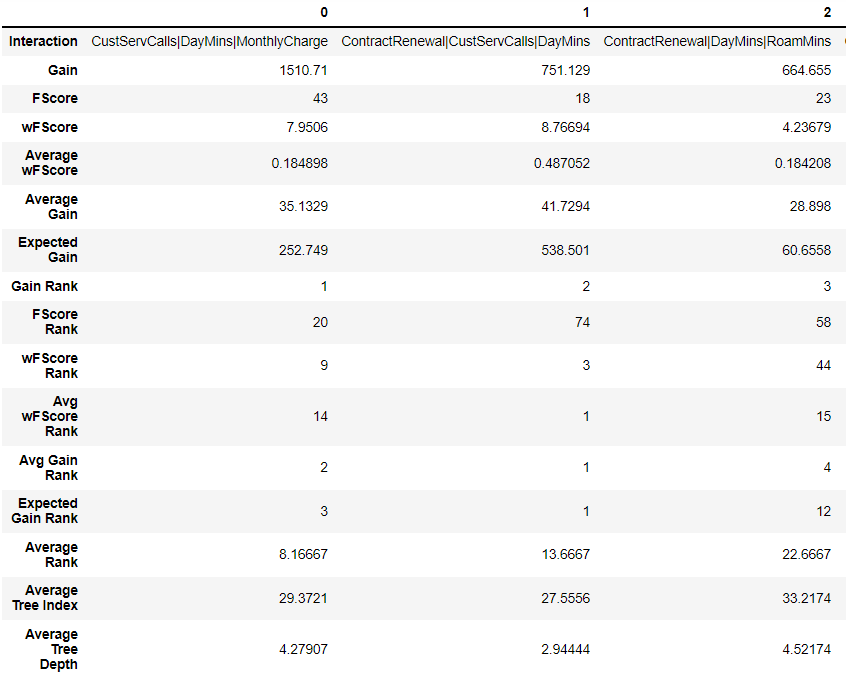


1. **Relatório xgbfir**

Através do relatório xgbfir, criado através da função saveXgbFI, podemos observar o ganho (*gain*) do modelo para cada variável, para cada dupla de variáveis e trinca de variáveis. É importante observar a presença de *DayMins* entre os maiores ganhos na relação observada por variável, por dupla e por trinca. Além disso, podemos analisar o comportamento da variável *ContractRenewal* que, na primeira tabela apresenta o sétimo maior ganho porém , na analise por trinca, está presente nos 3 primeiros ganhos, mostrando que a variável é mais relevante quando combinadas com outras do que analisada de forma isolada.

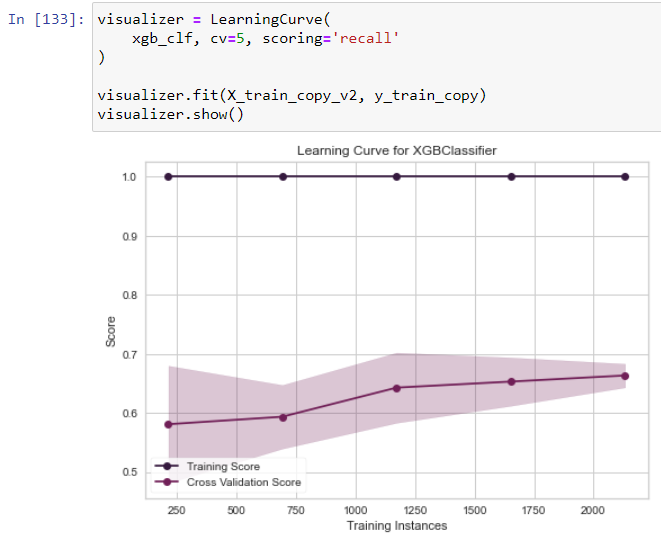


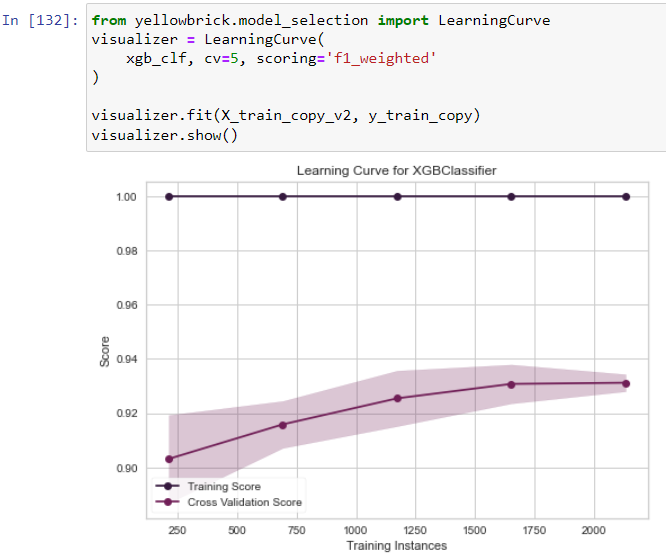




1. **Curva de Aprendizagem**

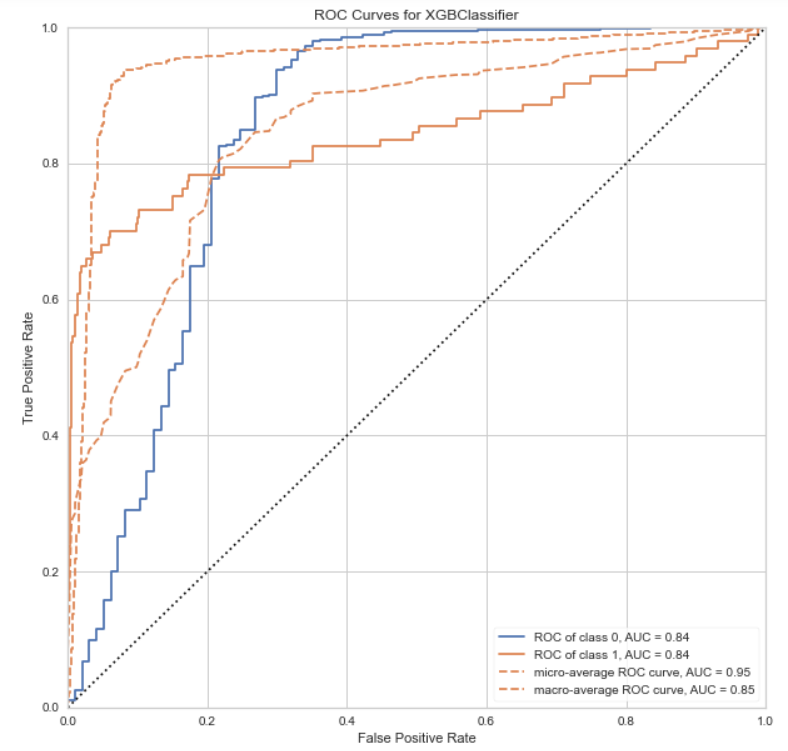
A curva de aprendizado, exibida através da função LearningCurve*,* nos permite analisar a relação entre a qualidade do modelo e a quantidade de dados de treinamento disponíveis. Além disso, por meio desse gráfico, também podemos perceber *overfitting* ou *underfitting* nos dados. Caso tenhamos alta variabilidade nos dados de treinamento, podemos estar sujeitos a *underfitting* e caso tenhamos alta variabilidade nos dados de validação, podemos estar sujeitos a *overfitting*. Para nosso modelo, observamos abaixo que não temos convergência entre o score de treino e validação, mostrando que o modelo se beneficiaria de mais amostras de dados. Ainda, podemos observar que temos a presença de *overfit*, dado a variabilidade no *score* de validação e a diferença entre o *score* de treinamento e validação.





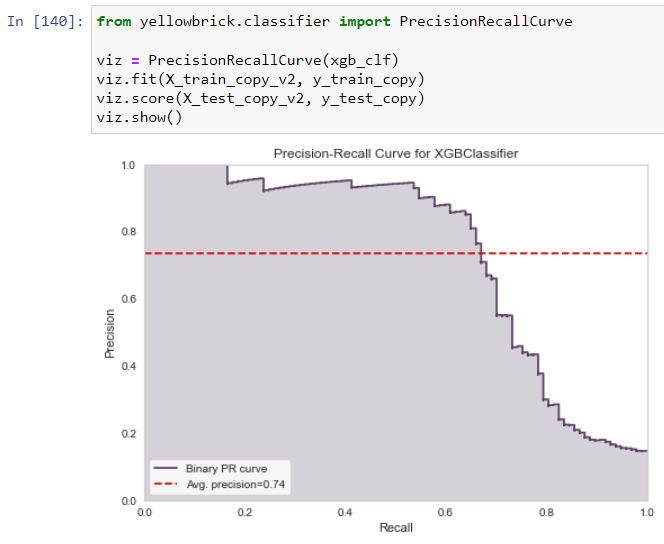
1. **Curva ROC**

A curva ROC, exibida através da função ROCAUC, nos permite observar o desempenho do classificador em relação à taxa de verdadeiros positivos e falsos positivos de ambas as classes. A linha tracejada represente como se comportaria um modelo puramente aleatório.



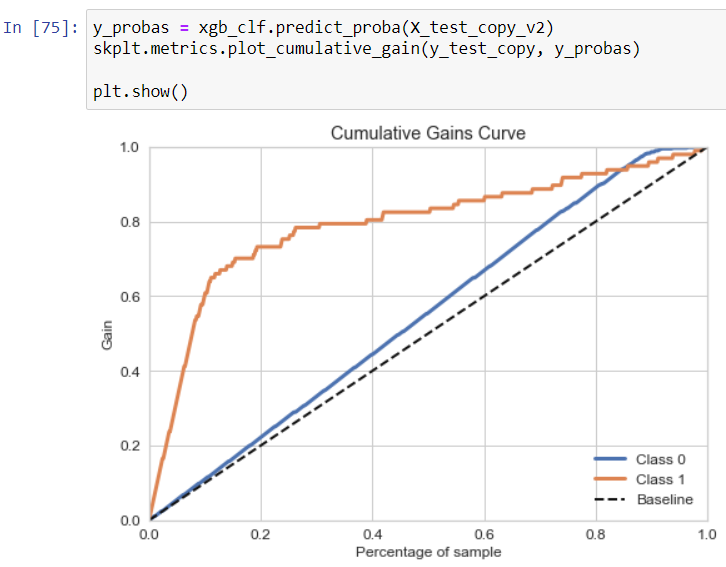
1. **Curva Precisão-Recall**

A curva de precisão-recall, exibida através da função PrecisionRecallCurve, nos permite visualizar a variação da precisão em função do recall. Podemos observar abaixo, quedas mais bruscas na precisão quando variamos o recall para aproximadamente 0,7.



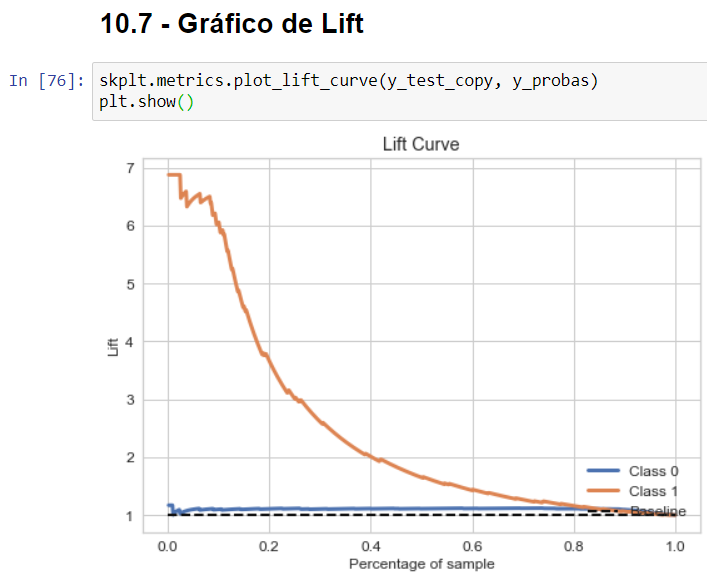
1. **Gráfico de Ganhos Cumulativos**

Através dessa visualização, exibida através da função plot\_cumulative\_gain, podemos verificar o ganho (*recall*) acumulado referente a cada classe para as amostras ordenadas de acordo com as probabilidades de pertencer à classe positiva. Podemos notar que para os 20% com maior probabilidade de pertencer a classe 1, temos cerca de 70% dos casos de *Churn* da base.



1. **Gráfico de *Lift***

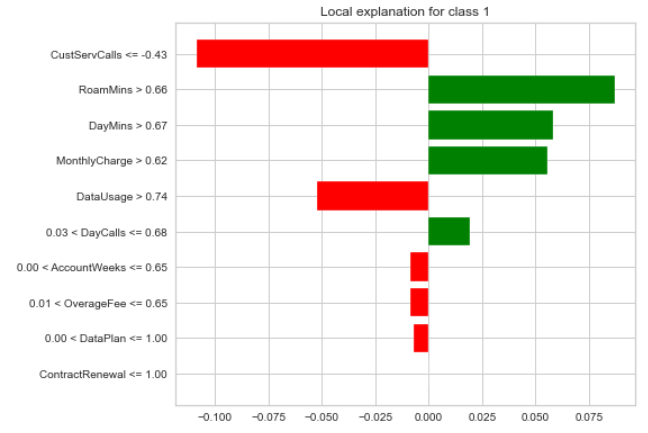
O gráfico de *Lift*, exibido atráves da função plot\_lift\_curve, mostra o comparativo de ganho associado da inferência do nosso modelo a escolher amostras de forma aleatória. O ganho associado, para as 20% das amostras com maior probabilidade de serem Churn=1, é aproximadamente 3,5 vezes maior do que escolher amostras aleatoriamente.



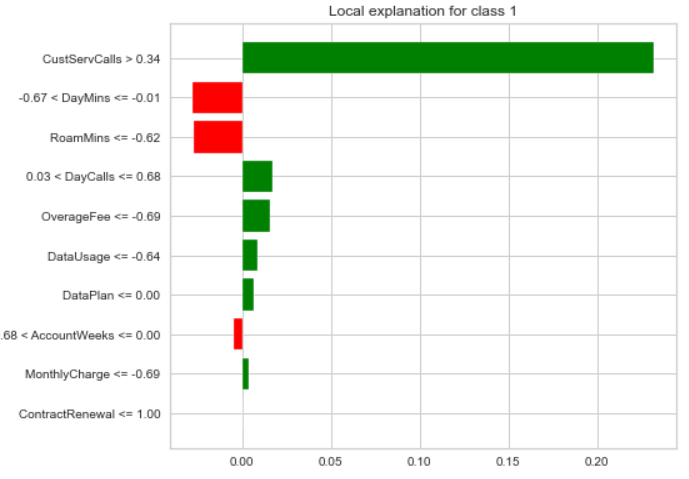
1. **Explicabilidade do Modelo com *Lime***

Abaixo, podemos verificar através da função LimeTabularExplainer, a explicação para algumas variáveis do conjunto de treino pertencer a classe 1 ou 0.

Cliente Propenso a Churn=0

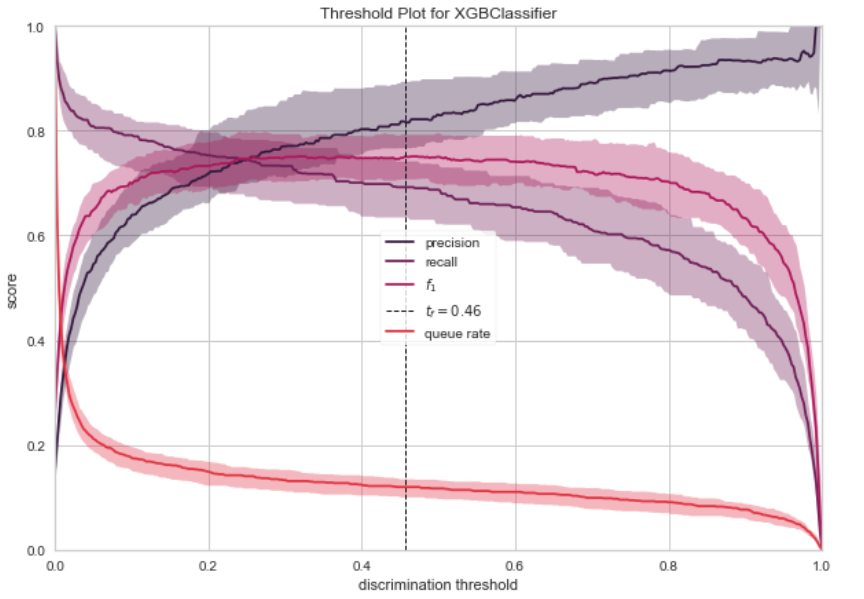


Cliente Propenso a Churn=1



1. **Avaliação de Melhor Fronteira de Decisão**

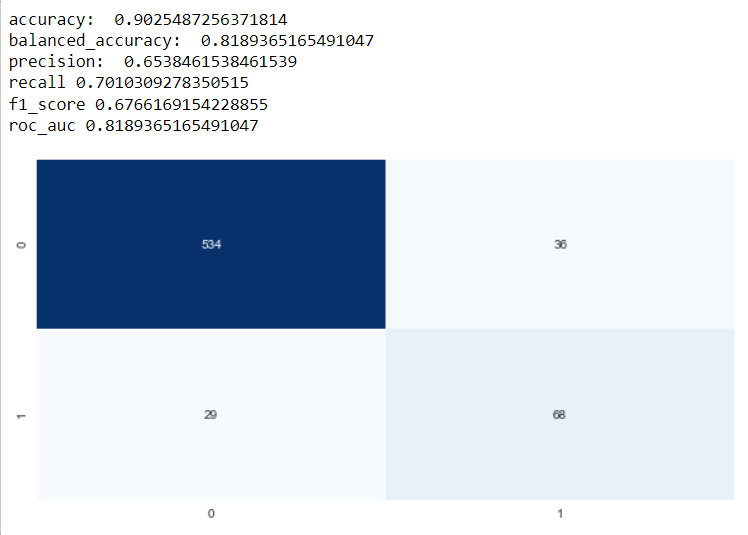
Abaixo, foi exibida a curva limiar de discriminação, através da função DiscriminationThreshold. Após análise, foi verificada a possibilidade de alteração do limiar, como forma de aprimorar o *recall* sem impacto negativo significativo nas outras métricas, como acurácia e F1-*Score*. Feito isso, foi alterado o limiar para 20%, melhorando resultado do recall de 0,68 e 0,66 para 0,72 e 0,70 em treino (*cross validation*) e teste, respectivamente. Em contrapartida, a acurácia foi reduzida de 0,94 e 0,95 para 0,92 e 0,90 e o F1-Score de 0,76 e 0,72 para 0,72 e 0,68, também avaliando respectivamente treino (*cross validation*) e teste.



Cross validation com 5 folds e threshold=0.2



Validação na base de teste com thresholds=0.2



1. **Conclusão**

Modelo Xgboost teve boa performance na base de dados atual e obtivemos bons resultados em treino e teste. O *recall* aprimorado através de mudança de limiar do modelo e obtivemos melhor aderência para problema abordado.

Apesar da escolha e construção de modelo com o *dataset* fornecido, há a necessidade de coletar mais dados para treinamento de modelo com melhor desempenho, como avaliado na sessão 10.3. Além disso, pode ser interessante identificação de mais variáveis que possam impactar no *Churn* do cliente.

Puderam ser percebidas algumas melhorias para o modelo atual, como a utilização de técnicas de seleção de atributos, visto que temos atributos com altas correlações. O tratamento de *outliers* também pode ser realizado para verificar possíveis melhorias no desempenho. Além disso, podem ser combinadas técnicas de *undersampling* e *oversampling* de forma a melhorar a qualidade do modelo final. Para futuras abordagens, também é valido, testar outros tipos de algoritmos e para os mesmos algoritmos, hiperparâmetros diferentes dos que foram avaliados.

1. **Referências**

Link do Kaggle com base para download: <https://www.kaggle.com/barun2104/telecom-churn>

LightGBM: <https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html#deal-with-over-fitting>

RandomForest: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

XGBoost: <https://xgboost.readthedocs.io/en/latest/parameter.html>

SVM: <https://scikit-learn.org/stable/modules/svm.html>

GradientBoosting: <https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>

Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow. Autor: Aurélien Géron

Machine Learning Guia de Referências Rápidas Trabalhando com dados estruturados em Python. Autor: Matt Harrison