**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**

**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

Alexandre Fábio de L.S Sales

**Modelo Preditivo para Aprovação de Cartão de Crédito**

**Utilizando Algoritmos Classificadores.**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Rio de Janeiro

2023

**SUMÁRIO**

1. [**Introdução ....................................................................................................... 2**](#Indrodução)
   1. [**Contextualização ................................................................................... 2**](#Indrodução)
   2. [**O problema proposto ............................................................................. 3**](#problemaproposto)
2. [**Coleta de Dados ............................................................................................ 4**](#coletadedados)
3. [**Processamento/Tratamento de Dados ...................................................... 5**](#processamento)
   1. [**Outliers .................................................................................................... 14**](#outlier)
4. [**Análise e Exploração dos Dados ............................................................... 17**](#analise)
5. [**Criação de Modelos de Machine Learning ............................................. 34**](#modelo)
6. [**Apresentação dos Resultados ................................................................ 49**](#apresentacao)
7. [**Links .......................................................................................................... 51**](#links)

[**APÊNDICE ...................................................................................................... 51**](#apendice)

**1. Introdução**

* 1. **Contextualização**

**Cartão de crédito** é um dos meios de pagamentos mais populares no Brasil e no Mundo. Segundo a ABECS (Associação Brasileira das Empresas de Cartões de Crédito e Serviços), no Brasil, **as transações em cartões foram em torno de R$ 3,31 trilhões no ano de** 2022**.**

Os primeiros cartões de crédito surgiram nos Estados Unidos na década de 1920 por donos de empreendimentos - hotéis, postos de gasolina, lojas de departamentos - que os davam aos clientes mais fiéis e abonados. Uma espécie de cartão de fidelidade.

Somente em 1949, o esquecimento da carteira para pagar a conta de um jantar de negócios, inspirou a ideia de cartão para restaurantes por um executivo de uma corporação de crédito - Frank Macnamara. Em 1950 foi lançado “The Diners Club” (Clube da Janta), considerado o primeiro cartão de crédito universal. Assim, surgiu um novo conceito em vez das empresas oferecerem crédito, haveria um intermediário, sendo aceito em vários estabelecimentos. No início o cartão era de papel, aceito apenas em restaurantes e pessoas influentes. As empresas que aceitavam aderir ao modelo pagavam um percentual por transação, e os assinantes uma taxa anual.

O desenvolvimento foi rápido, no 3º aniversário existiam 42 mil membros e 330 estabelecimentos no EUA cadastrados. Os associados pagavam US$3 ano, e os estabelecimentos 7%. No Brasil chegou em 1954.

Várias outras empresas adotaram o novo modelo, e assim foi criada a indústria de cartões de crédito. Algumas bandeiras mais conhecidas do mercado, se originaram a partir do final da década de 50, por exemplo:

1958 – American Express Card

1958 – BankAmericard que se tornou VISA (1976-77)

1966 – Intercard Bank que se tornou Mastercard (1979)

Com o tempo os cartões se popularizaram. Surgem novos formatos com a evolução tecnológica, por exemplo, incorporação de chips, senhas e aproximação. Interessante verificar que ainda tem muito espaço para o crescimento. Segundo o site Statista e Banco Mundial a penetração varia com o país, função da cultura ou necessidade de investimentos das organizações. A média mundial para a população acima de 15 anos é de apenas 10%.

* O Canadá foi um dos três países do mundo em 2021, onde a posse de cartão de crédito entre consumidores com 15 anos ou mais era superior a 70%.
* No Brasil a estimativa é que em torno de 40.43% da população com 15 anos ou mais, possuía um cartão de crédito.
  1. **O problema proposto**

No mundo de hoje, onde as transações online se tornaram a norma, a fraude com cartões de crédito tornou-se uma grande preocupação para bancos e instituições financeiras. É essencial que os bancos prevejam se um cliente apresenta um risco de crédito para minimizar as perdas financeiras por inadimplência ou fraude.

Um cartão de crédito é emitido por um banco ou empresa de serviços financeiros que permite aos titulares do cartão emprestar fundos para pagar bens e serviços com comerciantes que aceitam cartões como forma de pagamento. Os cartões de crédito impõem a condição de que os titulares do cartão devolvam o dinheiro emprestado, acrescido de quaisquer juros aplicáveis, bem como quaisquer encargos adicionais acordados, integralmente até a data de cobrança ou ao longo do tempo.

O problema proposto consiste na construção de um algoritmo para auxiliar na tomada de decisão do cálculo de novos clientes de cartão de crédito. O trabalho não deve ser visto como a construção de uma metodologia para definir a aquisição de novos clientes de cartão de crédito, mas sim uma ferramenta que apoia a decisão baseada em dados históricos.

Temos como objetivos dessa análise:

• Realizar a análise descritiva dos dados dos clientes de cartão de crédito;

• Verificar a correlação entre eles;

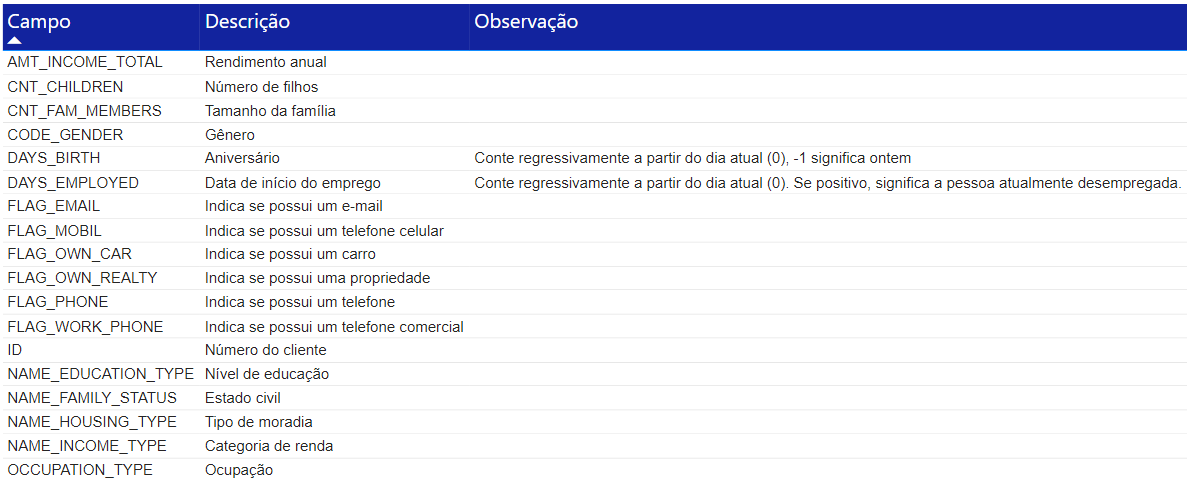
• Criar modelos preditivos para clientes que apresentem um risco de inadimplência ao adquirir um cartão de crédito, utilizando os algoritmos de classificação Árvore de Decisão, Regressão Logística, Naïve Bayes, Gradiente Descendente, KNN (K - Nearest Neighbors) e Randon Forest.

Os dados utilizados neste trabalho contém transações efetuadas com cartões de crédito, para análise exploratória, extraídos do site <https://www.kaggle.com/datasets/> .

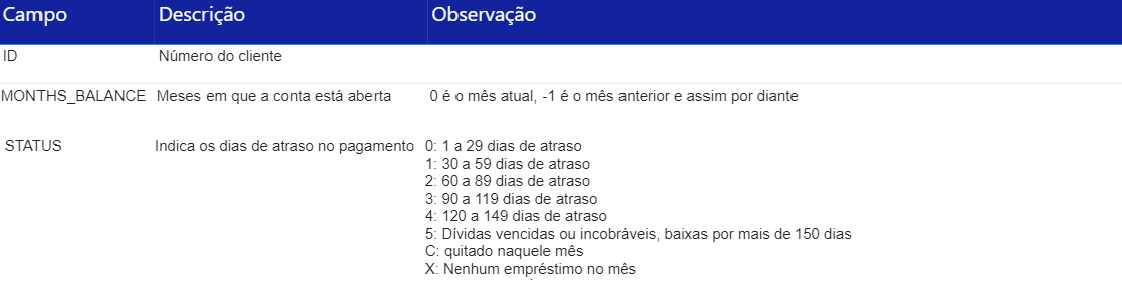
**2. Cole****ta de Dados**

Para o tratamento do problema proposto, foram utilizados dois *datasets* (conjunto de dados) relativos às transações de cartões de crédito, extraídos do site <https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction/data>.

O primeiro dataset, "application\_record.csv", traz as informações cadastrais dos clientes, que podem ser utilizadas como recursos para a previsão. Esse dataset possui os seguintes campos:



O segundo dataset, "credit\_record.csv", registra o comportamento dos clientes de cartão de crédito. Esse dataset possui os seguintes campos:



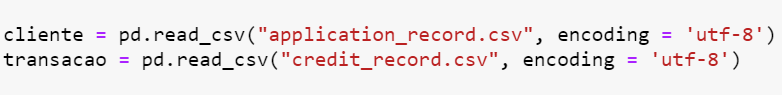
**3. Processamento/Tratamento de Dados**

O processamento e o tratamento dos dados foram feitos utilizando a linguagem Python, versão 3.8.17, no ambiente Jupyter Notebook, versão 6.5.4. Dentro da linguagem, utilizou-se a biblioteca “pandas” que é uma poderosa ferramenta para tratamento e análise de dados.

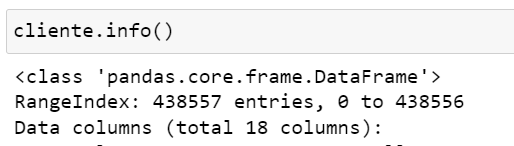
Inicialmente foi necessário importar a biblioteca “*Pandas*” (Figura 8), que é uma biblioteca de código aberto de licença BSD (em inglês, Berkeley Software Distribution) que fornece estruturas de dados de alto desempenho e fáceis de usar, além de ferramentas de análise de dados para a linguagem de programação Python.

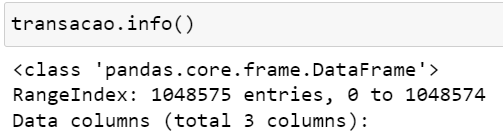


Em seguida, deve-se fazer a leitura e tratamento dos daframes, que, nesse caso, será feita individualmente para cada um deles. Os daframes “application\_record.csv” e “credit\_record.csv” serão processados e importados para as variáveis “cliente” e “transação” por meio do comando “pd.read.csv”, seguindo os parâmetros estabelecidos nos arquivos com as características dos *datasets*.



O uso do comando “pd.read.csv”, cria uma estrutura bidimensional de dados, como uma planilha, chamada daframe. A utilização da função “info ()”, apresenta informações sobre os dataframes criados.

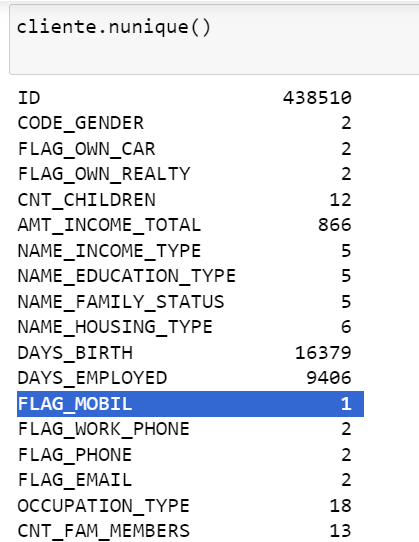


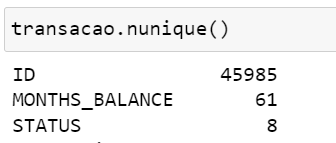


Os dataframes criados apresentam as seguintes dimensões:

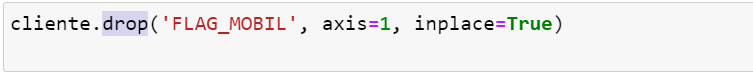
* Dataframe “cliente” - 438557 linhas, divididas em 18 colunas.
* Dataframe “transacao” - 1048575 linhas, divididas em 3 colunas.

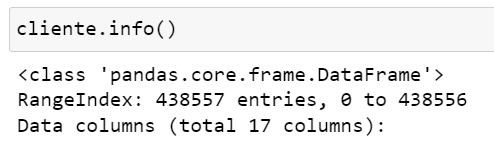
A coluna FLAG\_MOBIL não foi mantida nesse estudo, pelo fato de que essa coluna é uma constante e por isso foi desconsiderada. O dataframe “transacao” não apresenta colunas com valores constantes.





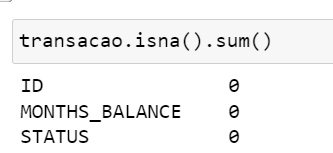
Para remover a coluna FLAG\_MOBIL, foi utilizado o seguinte código:





O dataframe “cliente” apresenta agora 17 colunas com as 438557 entradas filtradas anteriormente. Para verificar se há dados nulos nos dataframes, utilizou-se a função “isnull()”, que faz essa análise, em conjunto com a função “sum()”, que, nesse caso, soma os valores encontrados na função anterior.

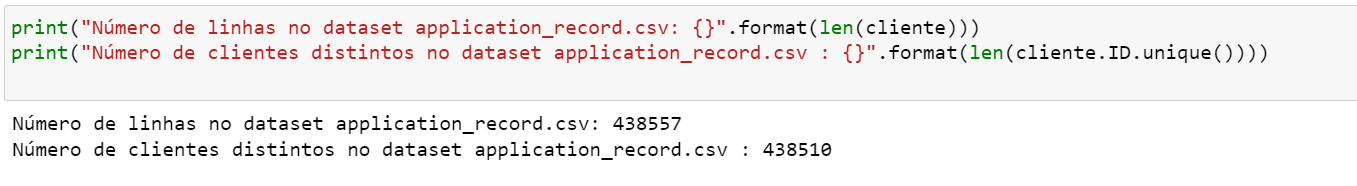




Após esse comando, verificou-se que o dataframe cliente, apresenta dados nulos na coluna OCCUPATION\_TYPE que necessitam de tratamento. Nesse caso, os dados nulos serão substituídos pelo valor 'Other'. Foi utilizada a função “fillna” para a substituição.



O número de clientes com ID’s ditintos no dataframe “cliente” deveria ser igual ao número de linhas neste dataframe . A verificação de dados duplicados foi confirmada pelos método “unique”.



Para remoção dos registros duplicados foi utilizado o método “drop\_duplicates”, mantendo o primeiro registro encontrado.



O dataframe “transação” não apresenta dados duplicados.



Para modelos de machine learning supervisionados é fundamental a etapa de criação de uma coluna target no dataset. Em problemas que isso é possível, faz com que os modelos e resultados obtidos possam ser explorados com maior facilidade.

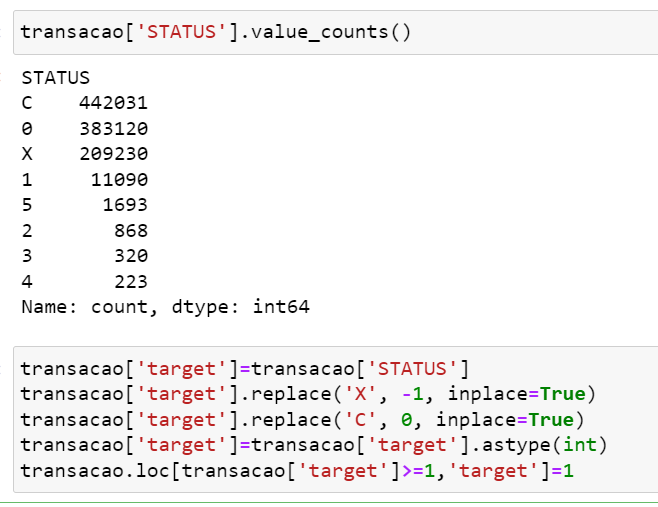
Na maioria das situações, um algoritmo de aprendizado de máquina supervisionado é usado para derivar a variável target. Esse algoritmo usa dados históricos para aprender padrões e descobrir relacionamentos entre outras partes do seu conjunto de dados e a variável target.

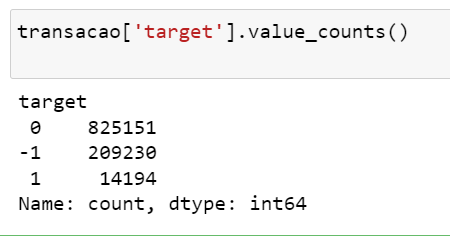
O caso em análise busca avaliar a previsibilidade se o usuário é um cliente 'bom' ou 'ruim' com base em alguns parâmetros. Dessa forma, a informação que utilizaremos como resultado é a coluna STATUS, que traz os dias em atraso no pagamento.

Os clientes que se enquadraram na categoria “Dados insuficientes” STATUS=X foram aqueles que não utilizaram o cartão de crédito, indicando falta de histórico de transações.

Os clientes considerados “maus pagadores” atrasaram os pagamentos por mais de 30 dias pelo menos uma vez em seu histórico de pagamentos. Isso incluiu clientes com códigos específicos (1, 2, 3, 4 ou 5) em seu histórico mensal.

Por fim, os clientes com histórico consistente de pagamentos pontuais (status “C”) ou atrasos de até 29 dias (status “0”) foram categorizados como de baixo risco e considerados perfis adequados para aprovação de cartão de crédito.



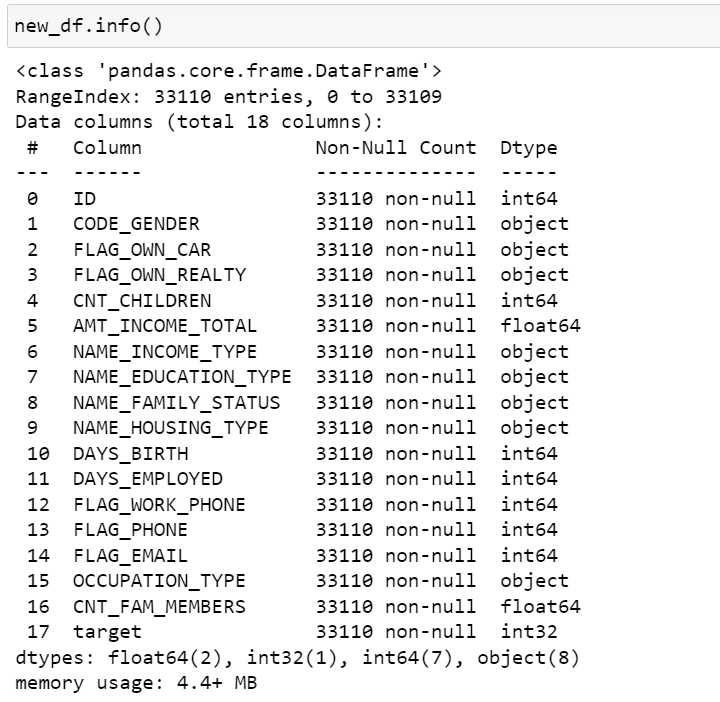


O valor de target = -1 será eliminado da construção do modelo porque não ocorreu a utilização do cartão.



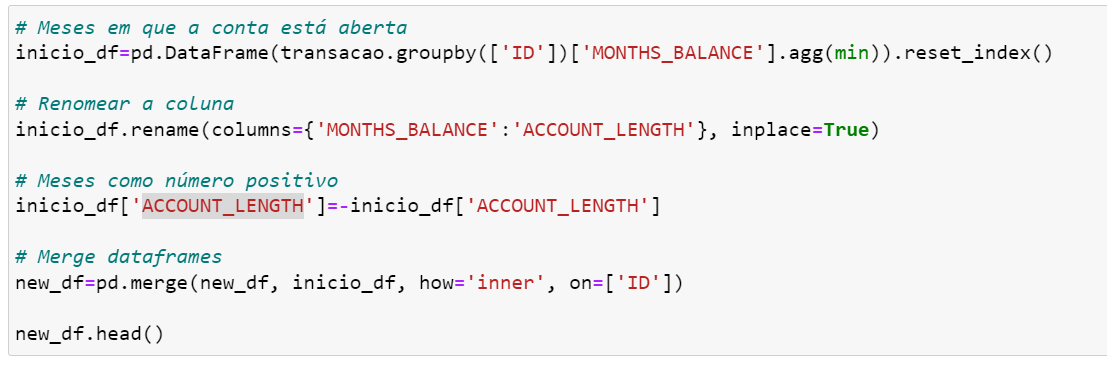
O objetivo agora é criar um único dataset consolidado (new\_df), gerado a partir da integração (join/merge) dos datasets gerados. Para isso, foi necessário agrupar o dataset “transacao”, por meio de seu ID de cliente, através dos comandos “groupby” e “agg(max)” que, em conjunto, selecionam o valor máximo de “target” e o ID para cada cliente.

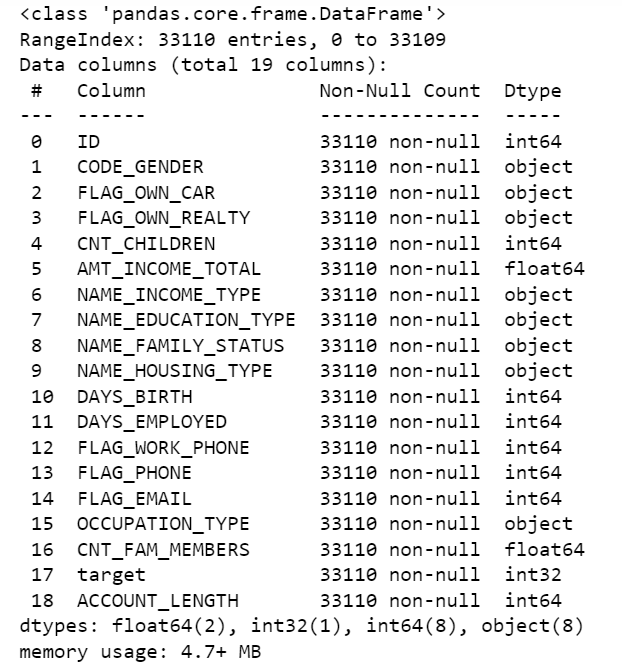




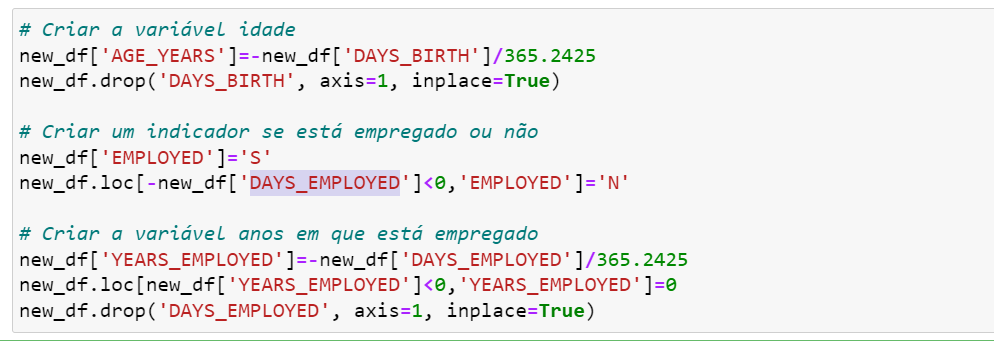
O próximo passo é incluir a coluna “MONTHS\_BALANCE”, que foi removida no merge anterior. Faz sentido que o número de meses em que a conta está aberta, seja correlacionado com o risco do cliente (uma vez que há mais oportunidades de perder pagamentos).

Como a coluna “MONTHS\_BALANCE” apresenta valores negativos, informando o número de meses em que a conta está aberta, foi selecionado o valor mínimo da coluna para cada ID. Após a seleção, a coluna será renomeada para “ACCOUNT\_LENGTH”, os valores transformados para positivo e incluídos no dataframe “new\_df”.

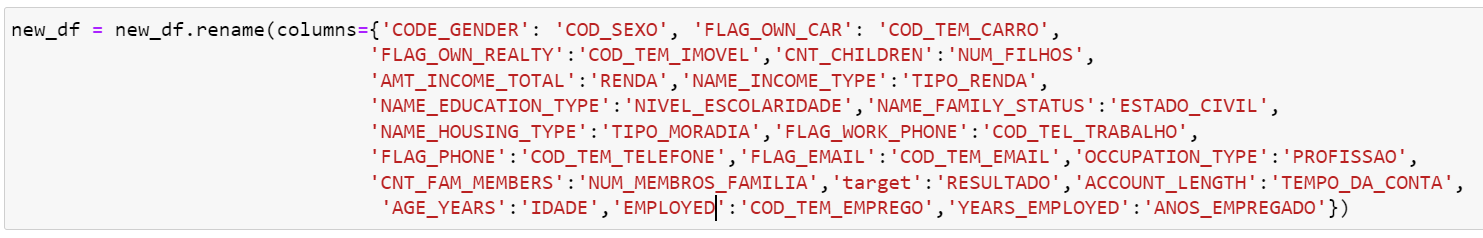




Percebe-se que o novo dataframe possui 33110 linhas. Agora iremos transformar as variáveis “DAYS\_BIRTH” e “DAYS\_EMPLOYED”, em variáveis que indiquem a idade, os anos de emprego e se o cliente está empregado. As antigas variáveis transformadas serão eliminadas do dataset.



Como o dataset apresenta os nomes das colunas na língua inglesa, a tradução para a língua portuguesa foi realizada para um melhor entendimento.



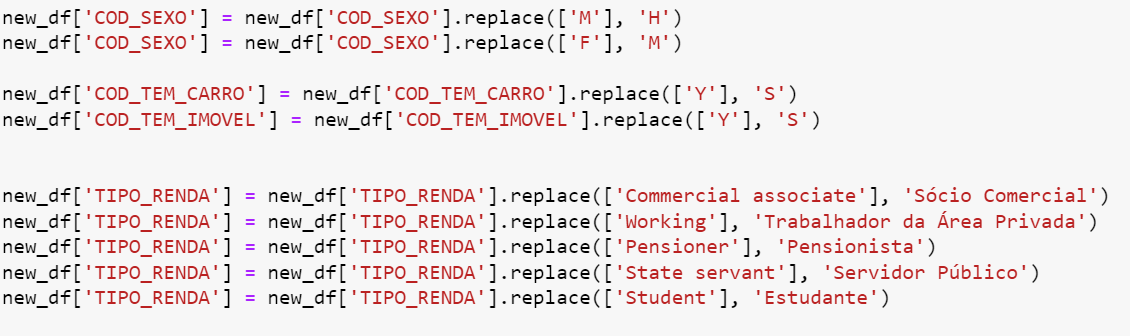
A transformação dos valores das colunas COD\_SEXO, COD\_TEM\_CARRO, COD\_TEM\_IMOVEL, TIPO\_RENDA, TIPO\_MORADIA, NIVEL\_ESCOLARIDA DE, ESTADO\_CIVIL e PROFISSAO do tipo “object” para a língua portu

guesa também foi realizada. A coluna NUM\_MEMBROS\_FAMILIA foi transfor

mada para o tipo int, correspondendo aos valores apresentados. O uso da fun

ção “unique()” apresenta os valores distintos das colunas que serão traduzidas para língua portuguesa.





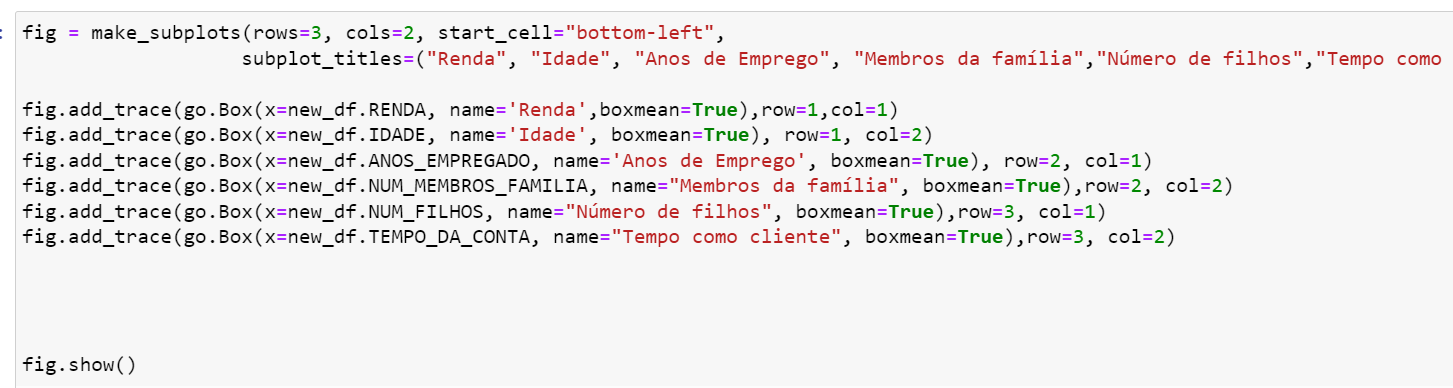


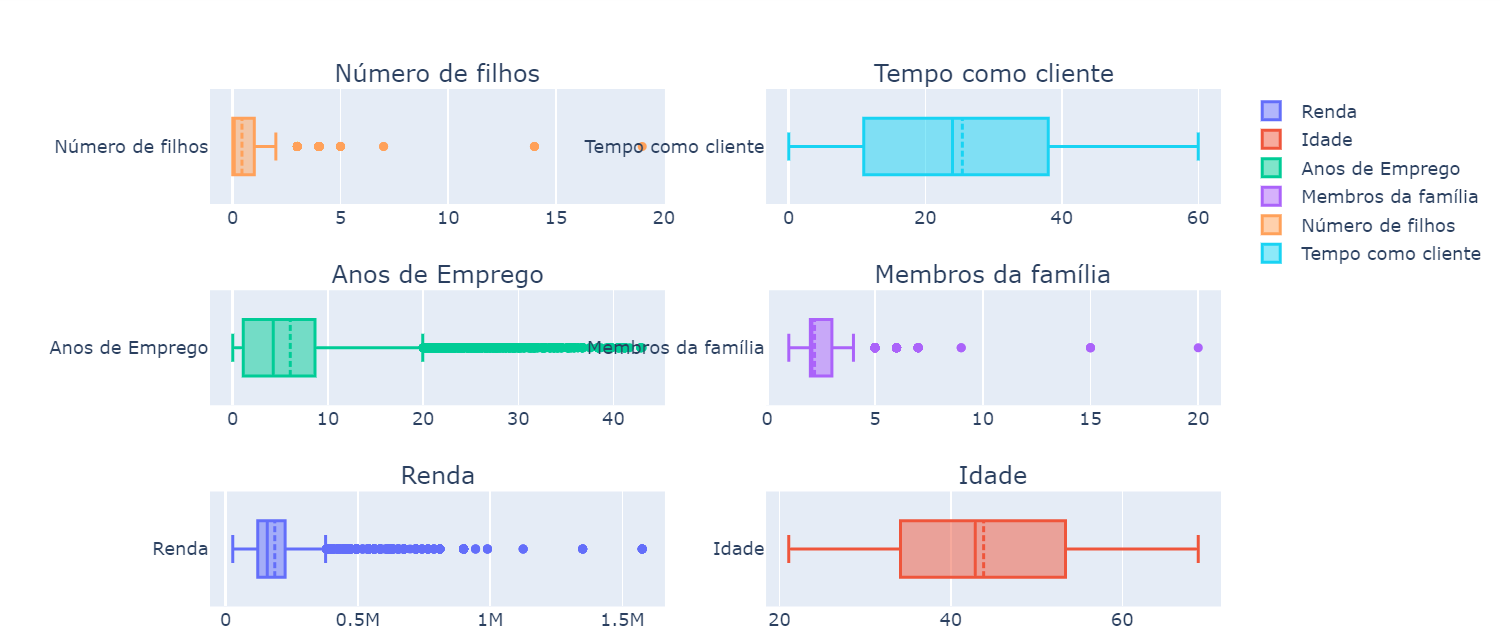
**3.1 Outliers**

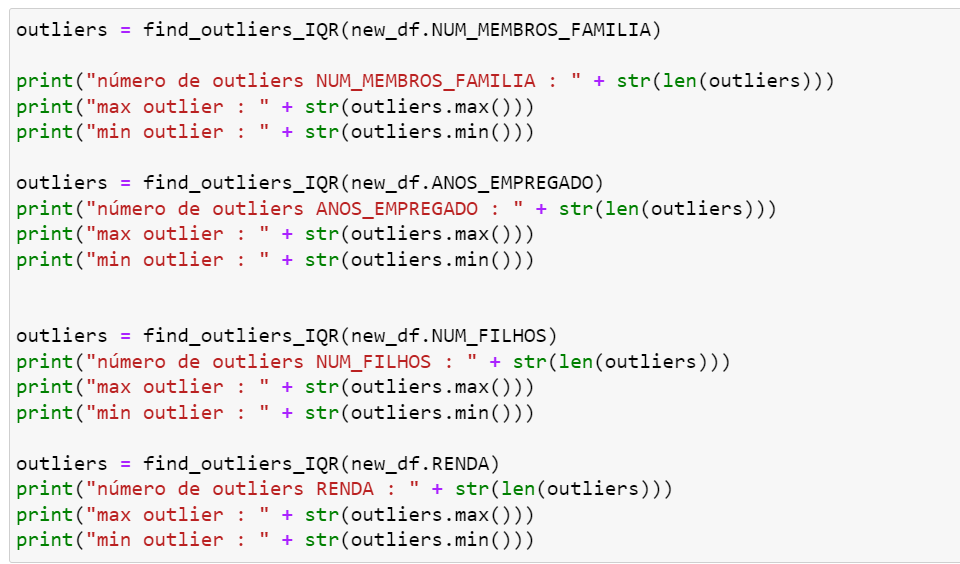
Um outlier é uma observação que está a uma distância anormal de outros valores em uma amostra aleatória de uma população. Eles podem aparecer devido a erros na entrada de dados ou medição, ou apenas porque há variação na população que você está observando. Seja qual for o motivo pelo qual aparecem, saber como identificar e lidar com valores discrepantes é uma parte importante da limpeza de dados.

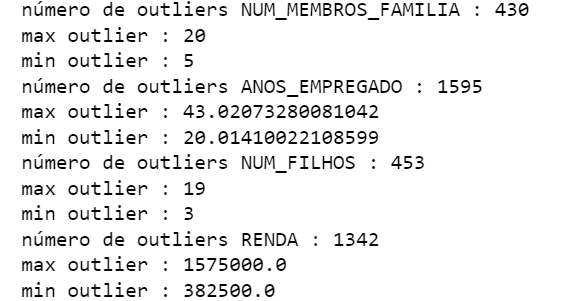
Os valores discrepantes serão removidos pelo método IQR, também chamado de "intervalo interquartil". Primeiro classificando os elementos em ordem crescente e depois encontrando Q1, Q3 e iqr pela diferença de Q3 e Q1. Finalmente, definindo o limite superior e inferior e removendo-os.

Uma das maneiras de ver se o conjunto de dados tem valores discrepantes é traçar os dados em gráfico. A função “go.Box” será utilizada para apresentar um gráfico de caixa que é uma representação estatística da distribuição de uma variável através de seus quartis. As extremidades da caixa representam os quartis inferior e superior, enquanto a mediana (segundo quartil) é marcada por uma linha dentro da caixa. As colunas que informam a renda, idade, anos de emprego, membros da família, número de filhos e tempo como cliente serão analisadas.

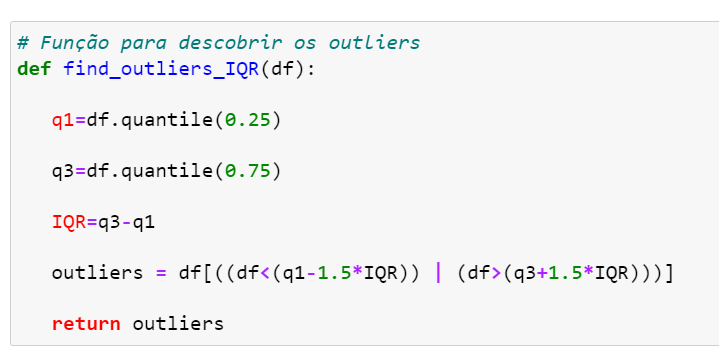


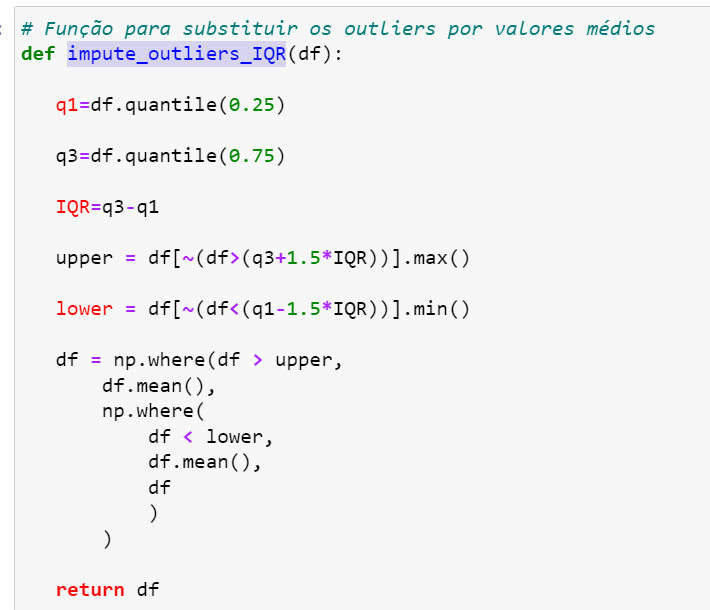


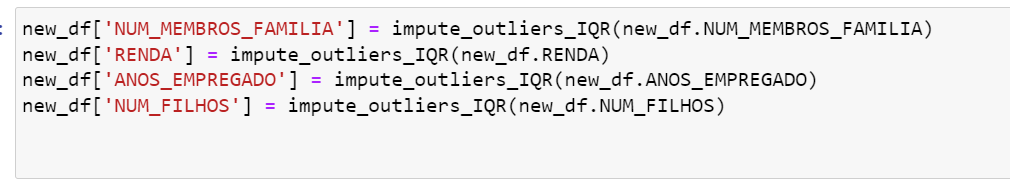




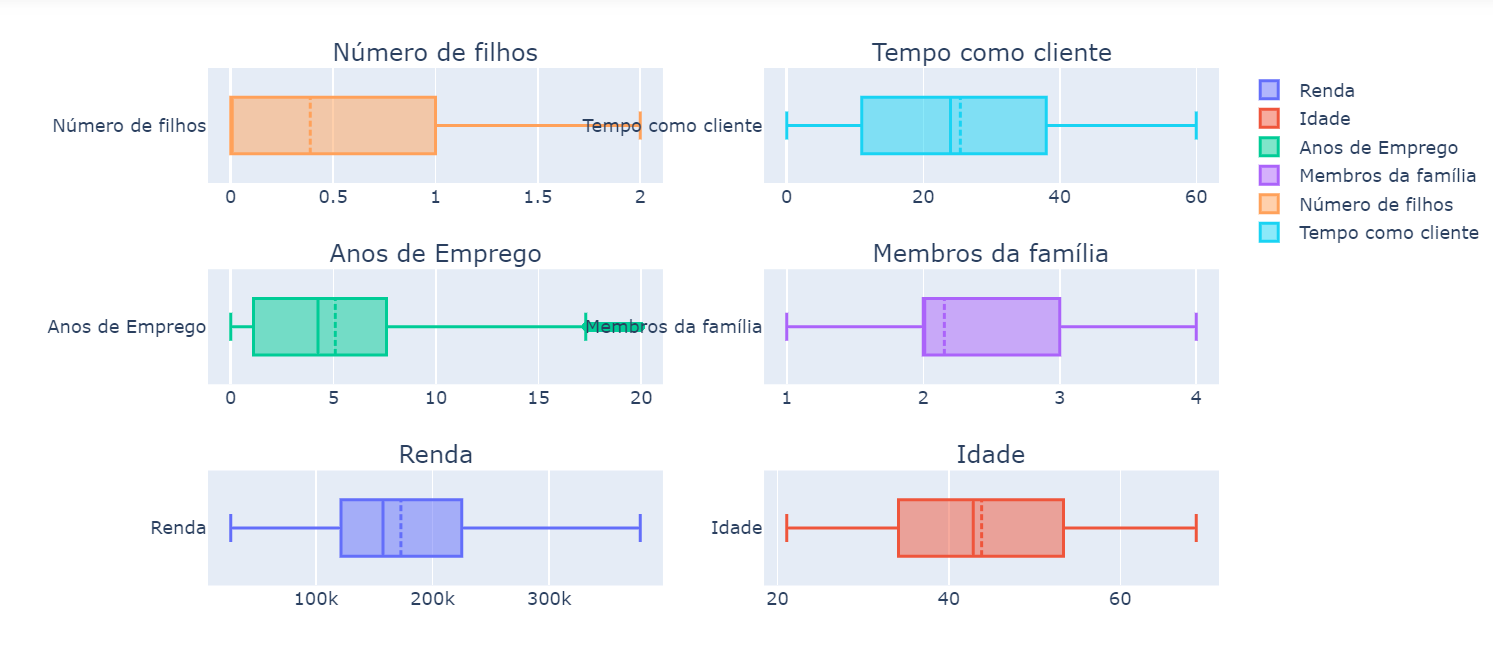
Pode-se observar que existem outliers para o número de filhos, anos de emprego, membros da família e renda. A função “find\_outliers\_IQR” foi utilizada para descobrir os outliers e a substituição dos outliers por valores médios, será feita pela função “impute\_outliers\_IQR”.





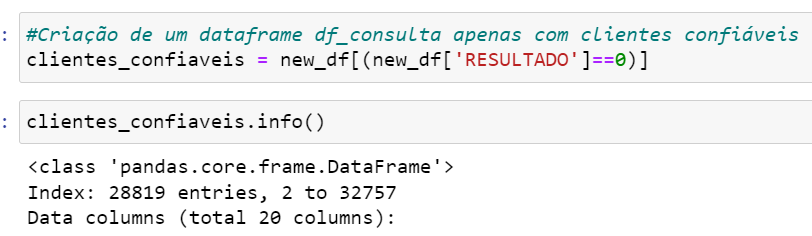


Após a substituição dos outliers, o gráfico de caixa não apresenta valores discrepantes.



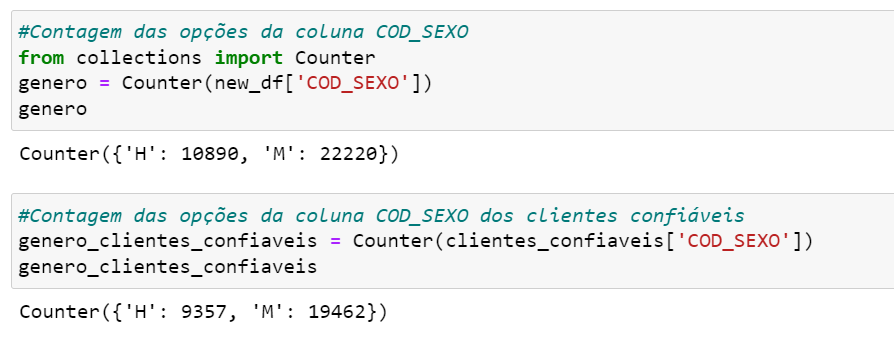
**4. Análise** **e Exploração de Dados**

O objetivo da análise é verificar a discrepância de característica entre os os clientes que possuem cartão de crédito e os que efetivamente foram considerados como confiáveis. Sempre que possível, as informações serão separadas nesses critérios e, para isso, a partir do dataframe “new\_df” será criado o dataframe “clientes\_confiaveis” apenas com os clientes confiáveis.



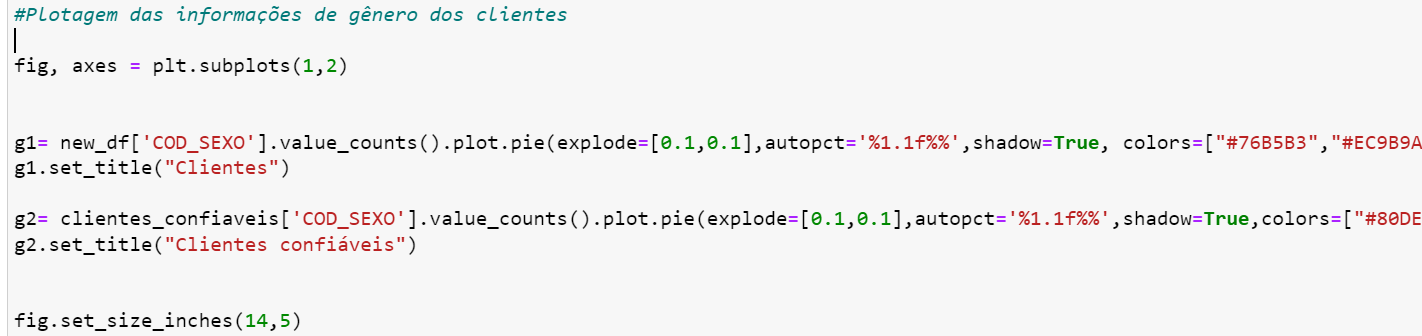
Observa-se que o dataframe “clientes\_confiaveis” possui 28819 linhas, o que corresponde a um grande número em relação ao total de linhas no dataframe original. Inicialmente a análise será feita pelo gênero dos clientes, fazendo a contagem do total de clientes por gênero, como também, a contagem por gênero dos clientes confiáveis.

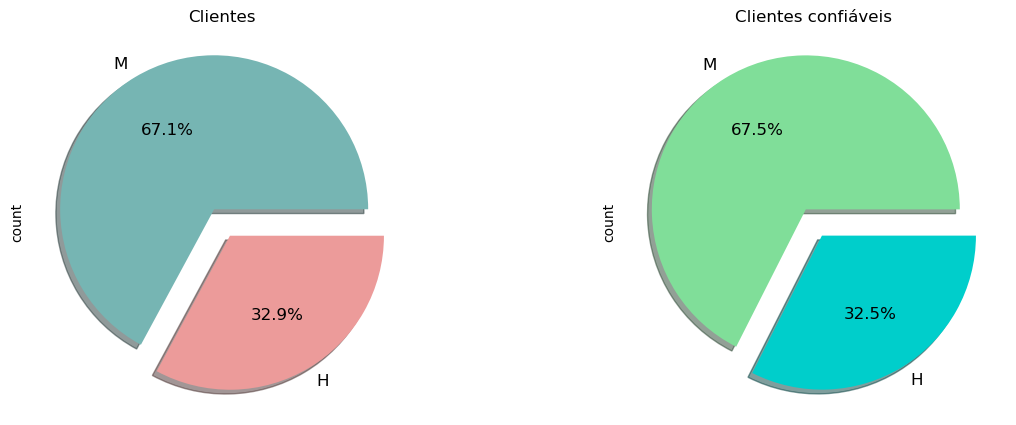
Faremos a importação da biblioteca “Collections”, com o uso da função "Counter", que conta quantas vezes uma determinada opção aparece em uma série de dados a armazena esses valores em um dicionário.



Para facilitar a visualização das informações sobre cada gênero, através biblioteca “Matplotlib” para a criação de gráficos, serão criados os gráficos de pizza com os percentuais e gêneros.

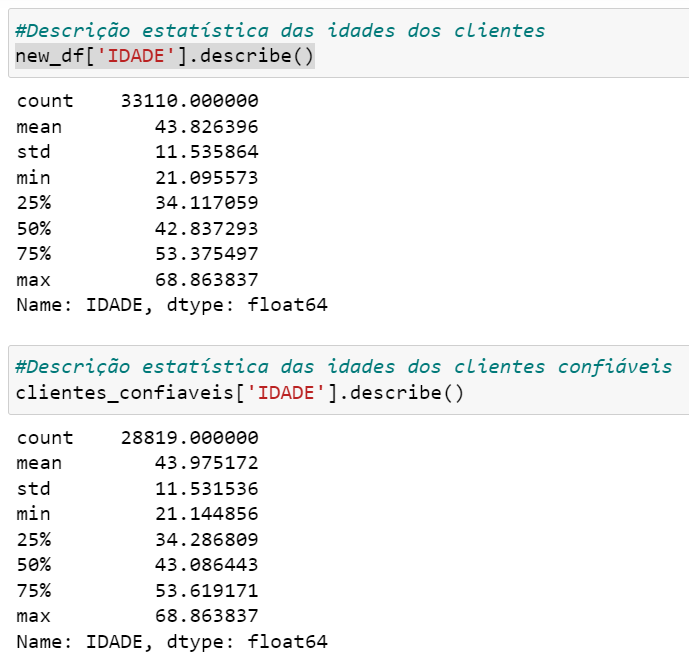


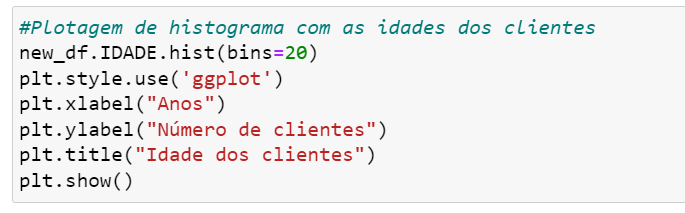


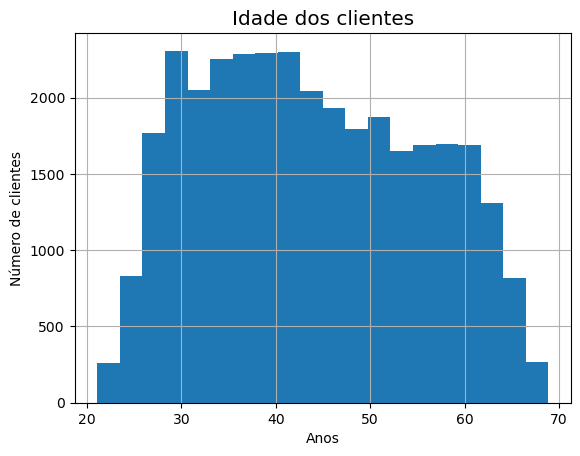


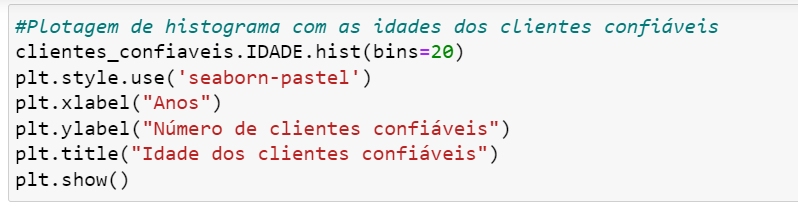
Analisando os gráficos pode-se perceber que o percentual de cada gênero é muito semelhante entre o total dos clientes e os clientes confiáveis. Em ambos os gráficos, a maioria do percentual é do sexo feminino com valores de 67.1% e 67.5%.

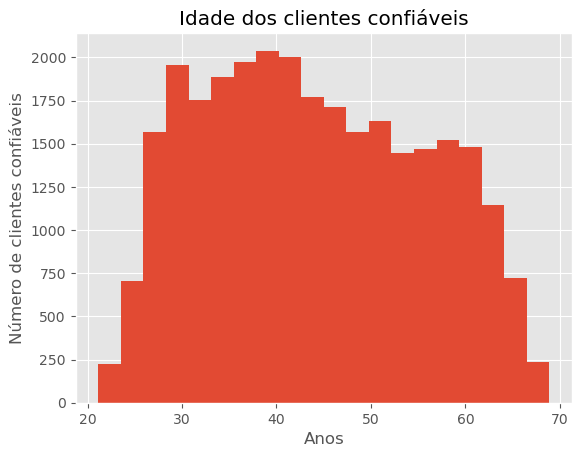
O próximo item a ser analisado é a idade dos clientes. Para essa análise será utilizada a função “describe” que apresenta os principais indicadores estatísticos de uma série de dados. Na visualização da distribuição dessas idades serão utilizados histogramas, seguindo os mesmos estilos descritos anteriormente.





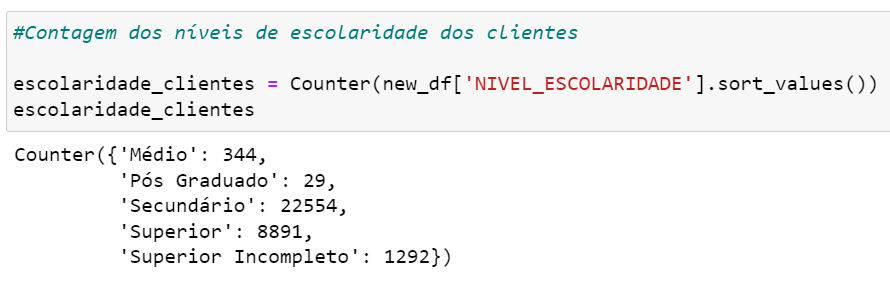


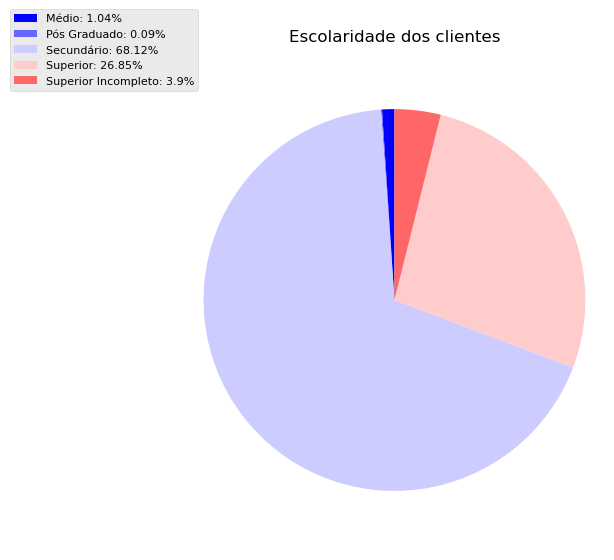


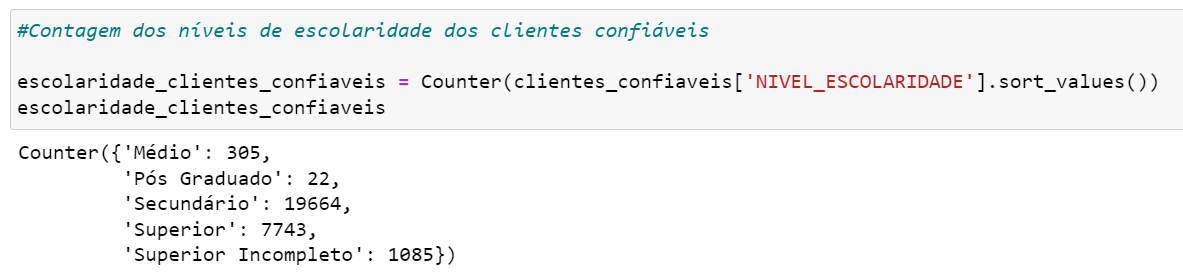


Analisando os histogramas, não se verifica nenhuma mudança significativa. Essa constatação pode ser confirmada por meio dos indicadores estatísticos das duas séries de dados que são muito similares, com variações mínimas na média de idade, por exemplo: 43,82 anos para todos os clientes e 43,97 anos para os clientes confiáveis.

A próxima análise será o nível de escolaridade dos clientes. Utilizaremos novamente a função “Counter”, para determinar quantas vezes um nível de escolaridade aparece no dataframe, e o gráfico de pizza com os valores e percentuais dos níveis de escolaridade.



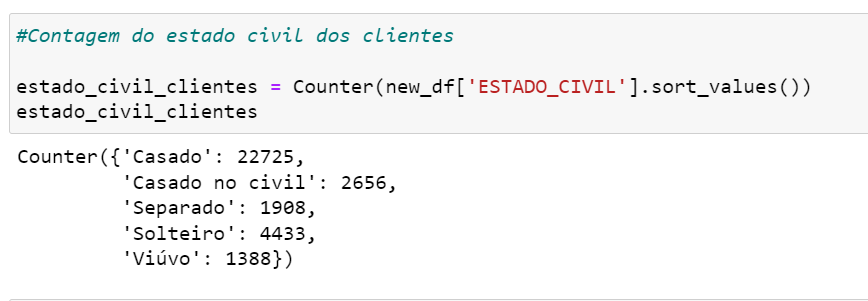


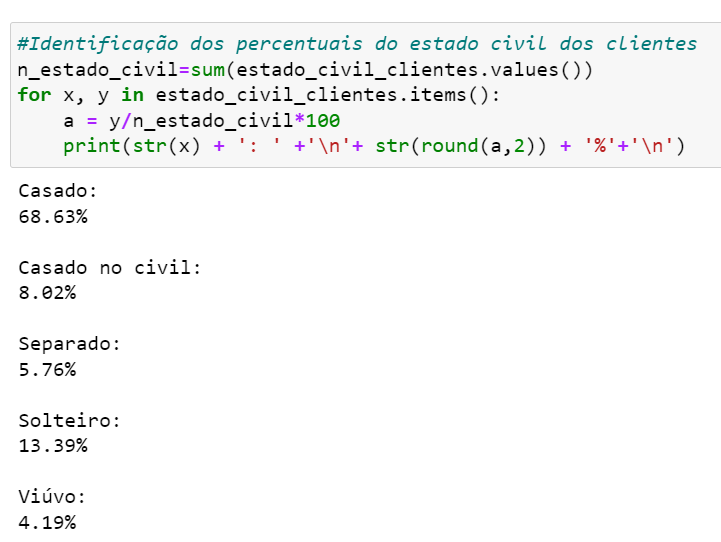


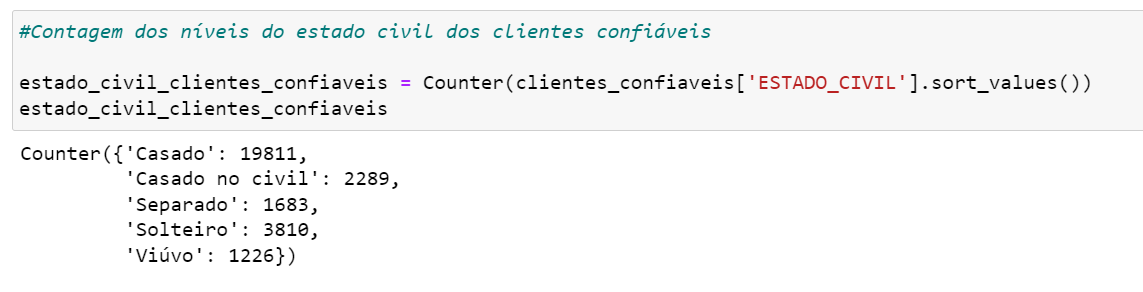


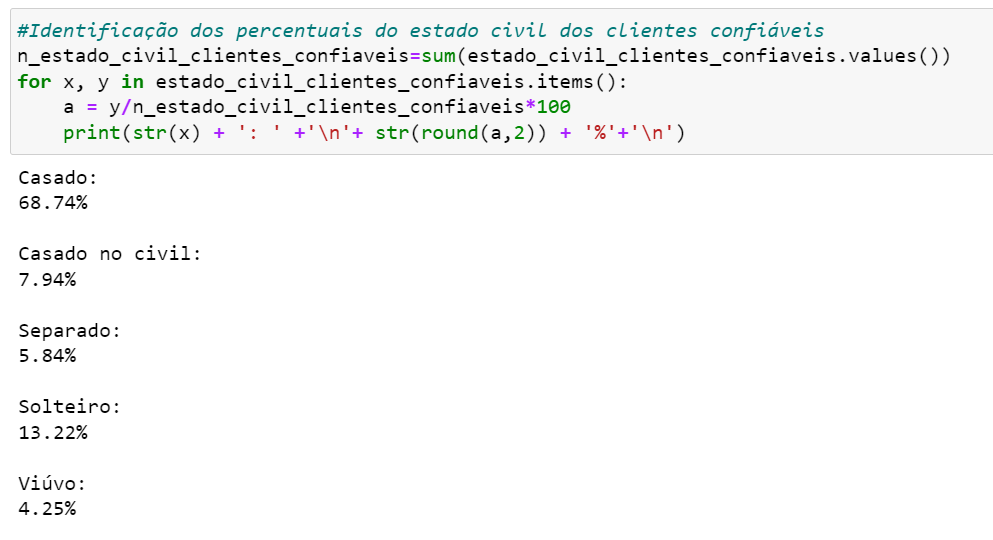
Percebe-se uma grande semelhança entre os níveis de escolaridade de todos os clientes e os clientes confiáveis. É importante destacar que a grande maioria dos clientes são do nível secundário.

A próxima análise será o estado civil. Os passos utilizados para a análise dos níveis de escolaridade serão semelhantes para o estado civil dos clientes, ou seja, a contagem dos valores nos dataframes e a apresentação dos gráficos de pizza. Porém, foi criado um laço de repetição “for” para apresentação dos percentuais de cada estado civil.

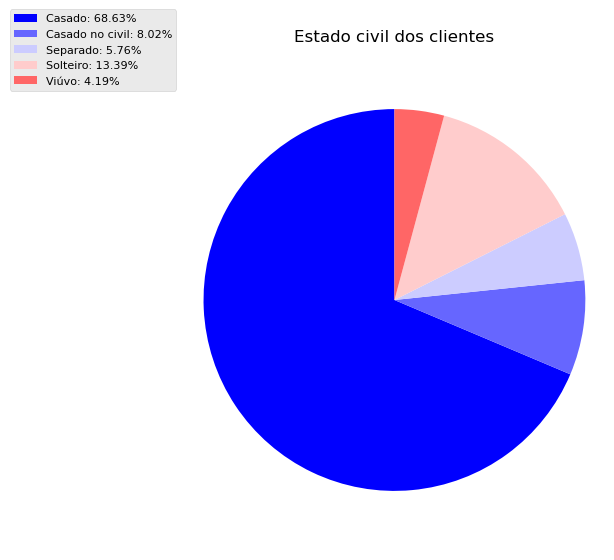


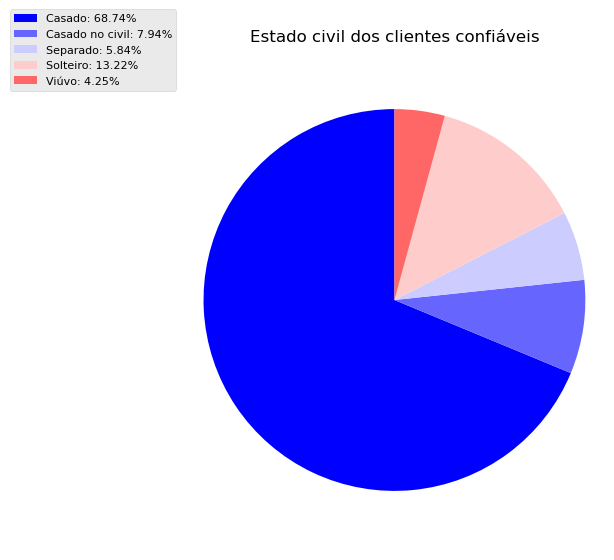






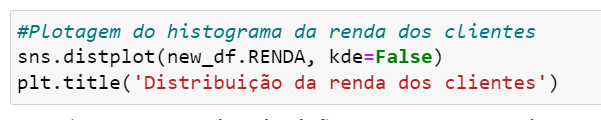
Os resultados da análise do estado civil também serão apesentados em gráficos.

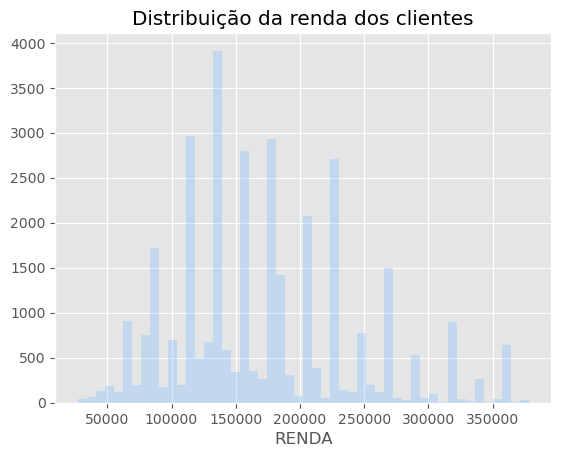


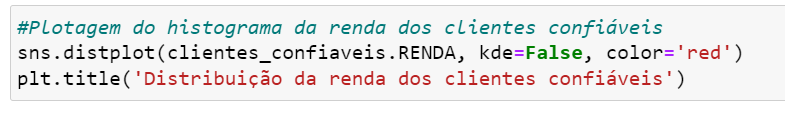


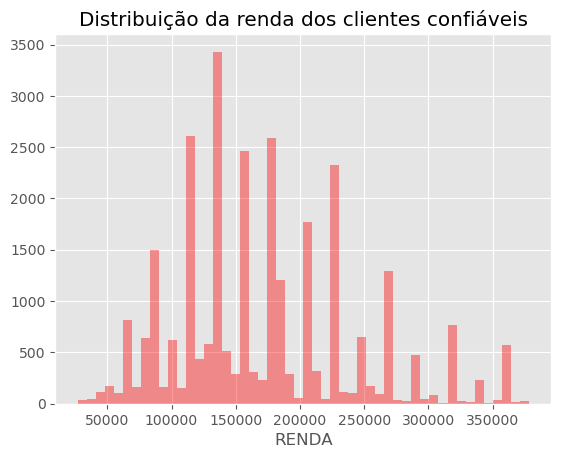
Percebe-se por meio dos gráficos e percentuais que a maioria dos clientes é casado, com 68,63%, número semelhante quando avaliamos os clientes confiáveis, com 68,74%.

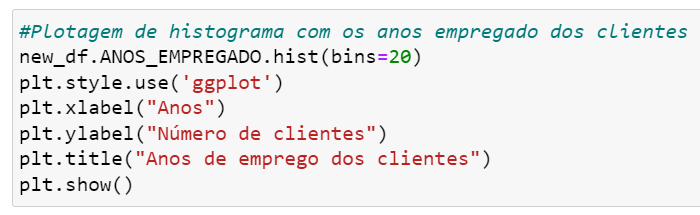
Para a distribuição de renda, anos de emprego e a posse de imóvel, carro e emprego, os percentuais entre o total de clientes e os clientes confiáveis estão também semelhantes, conforme os gráficos apresentados.

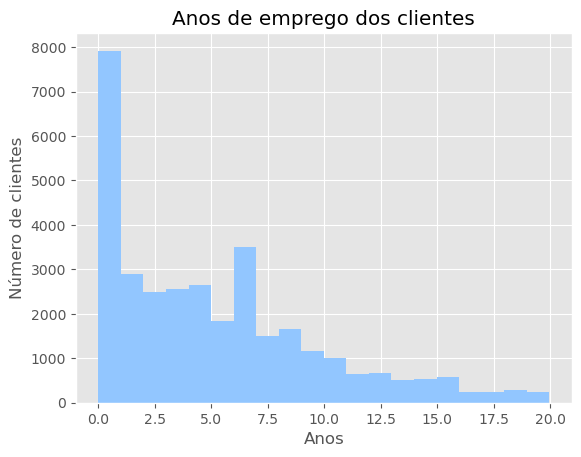




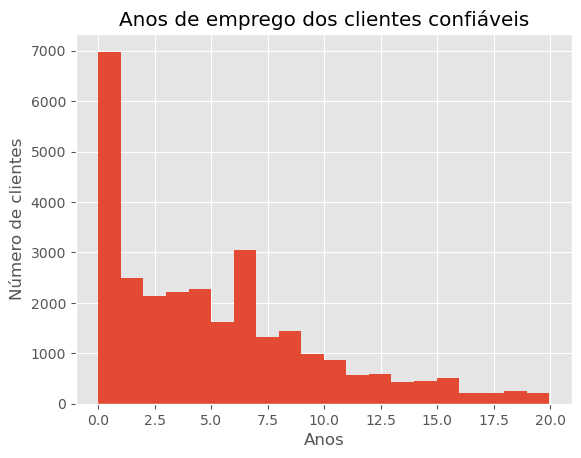


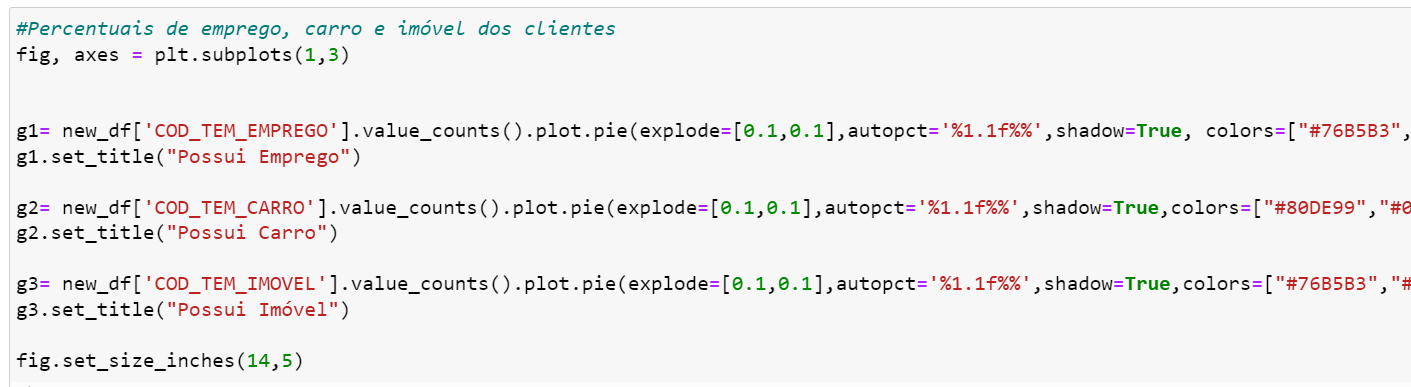


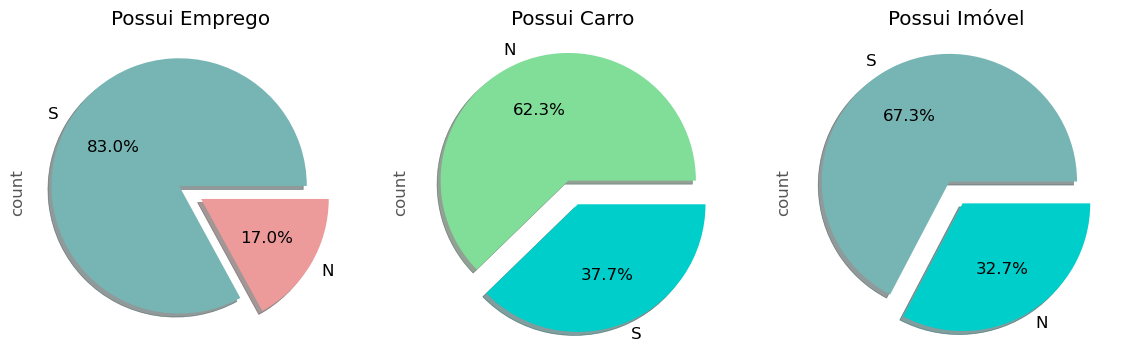


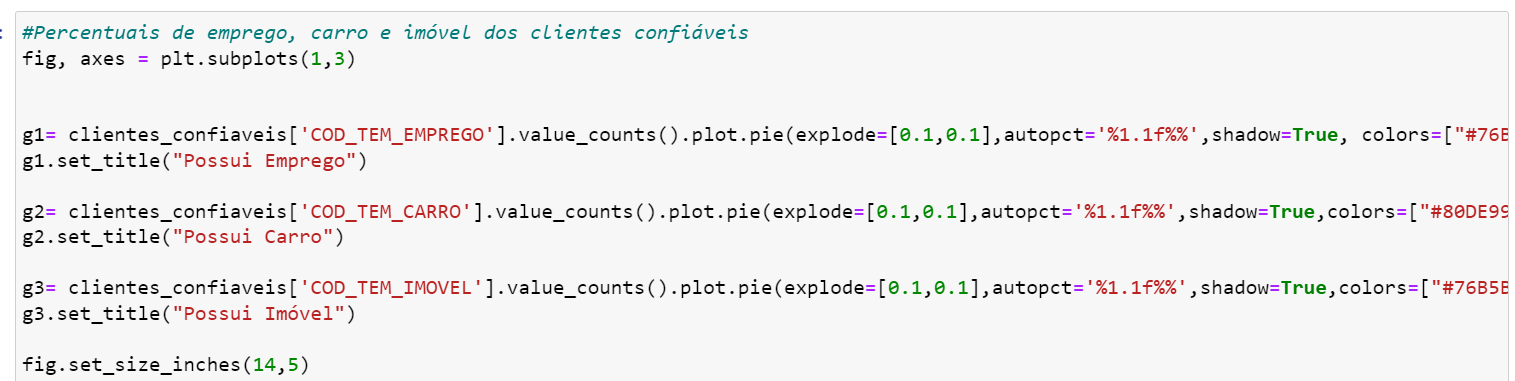


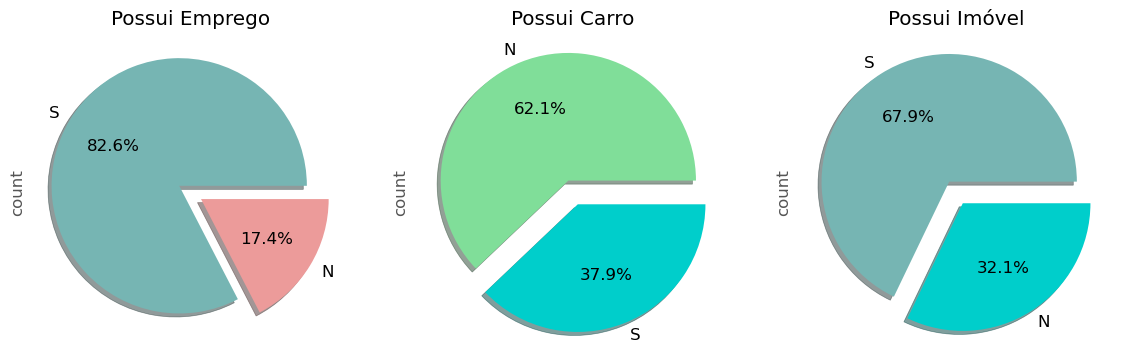






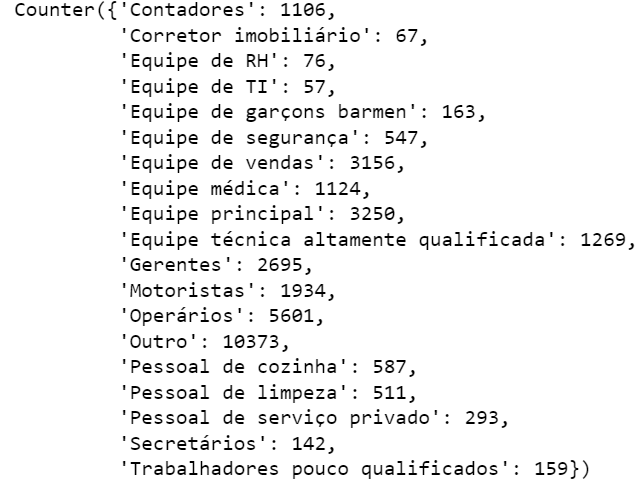






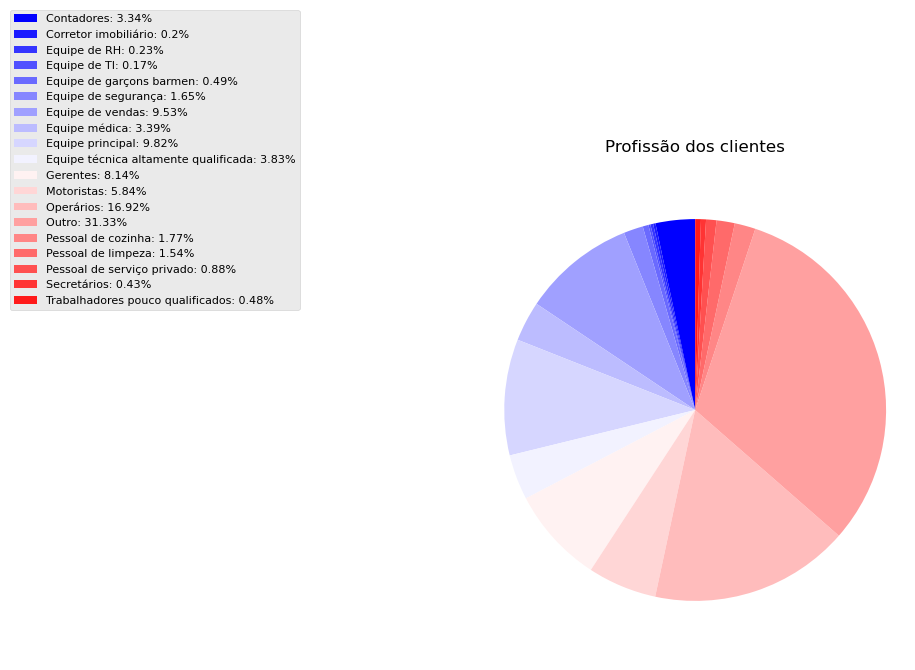
Ao analisar a profissão, podemos perceber que há um grande percentual para as profissões que não foram categorizadas, aproximadamente 31%, para todos os clientes e os clientes confiáveis.



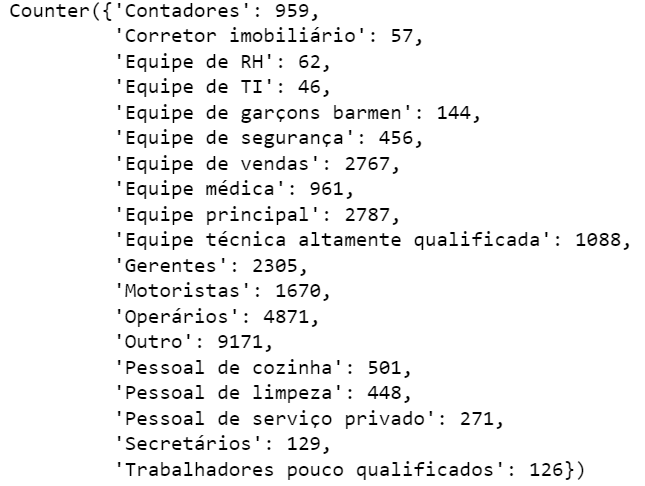


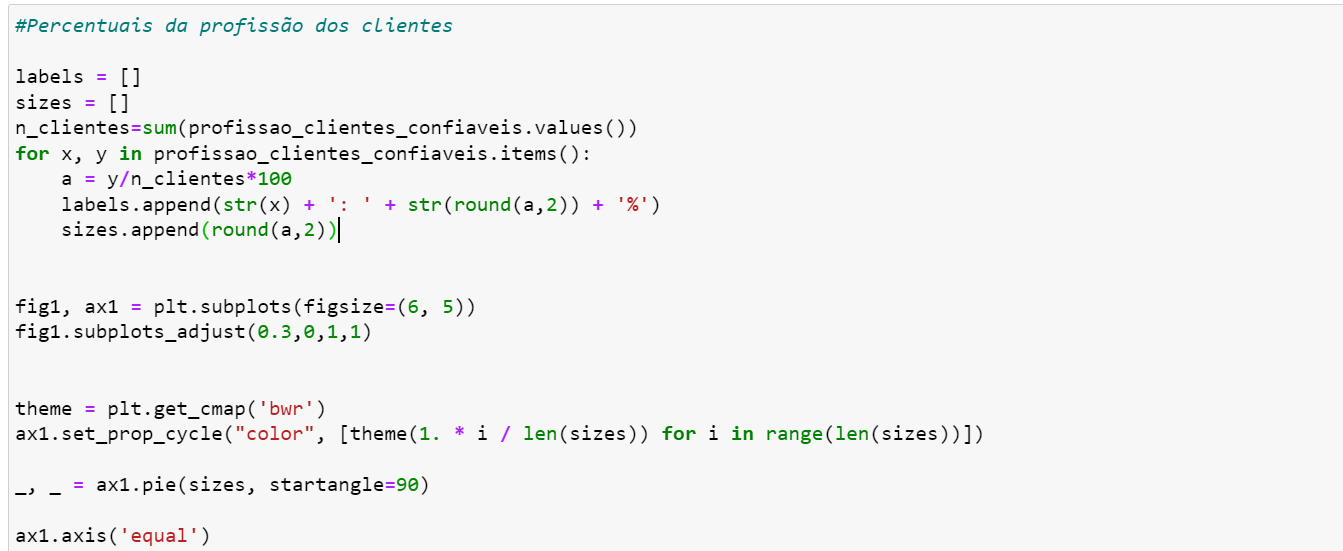


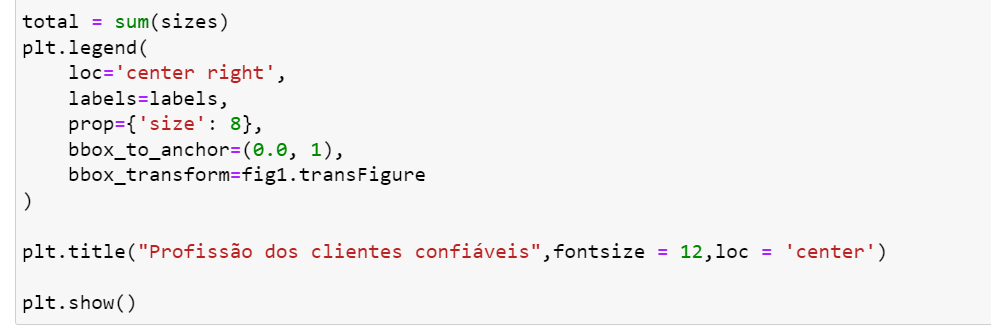






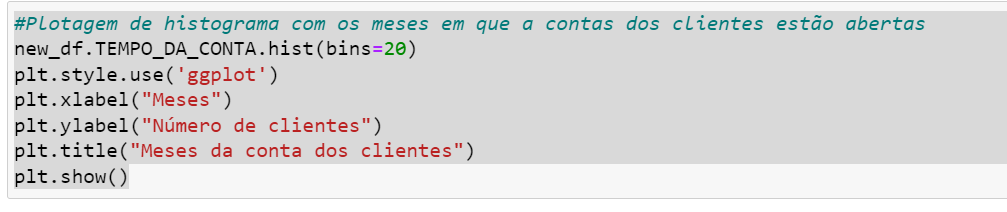


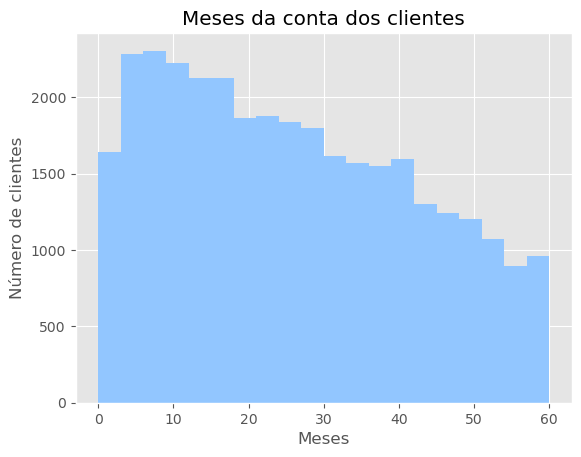


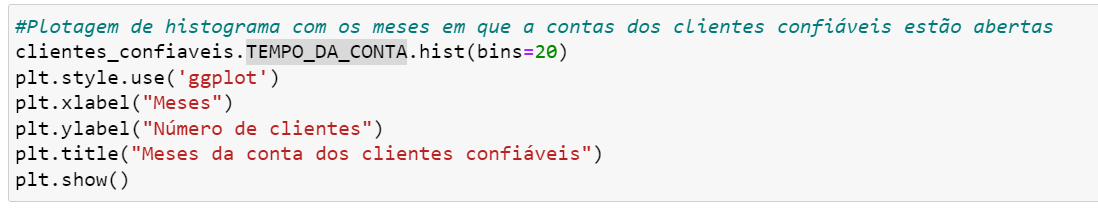


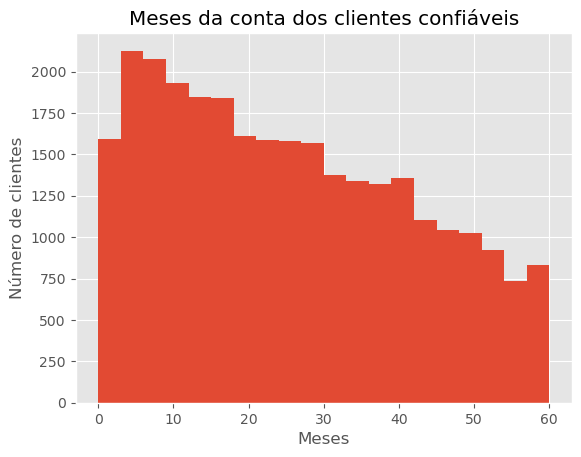


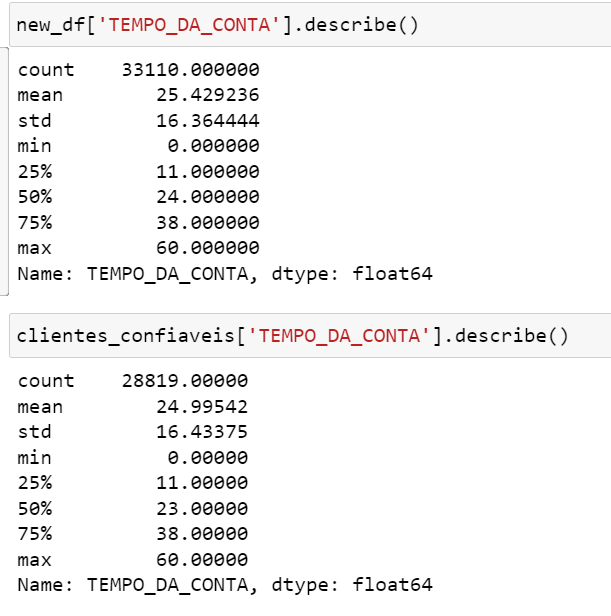
A quantidade de meses em que as contas estão abertas, são analisadas através do histograma e demonstram que há semelhanças entre os clientes confiáveis e o total de clientes.











Da análise de todas as informações desse dataframe, percebe-se que os dados dos clientes, confiáveis ou não, são semelhantes, o que indica um comportamento padrão nos dados do dataframe.

**5. C****riação de Modelos de Machine Learning**

O aprendizado de máquina (Machine Learning) é o estudo de algoritmos de computador que melhoram automaticamente através da experiência e do uso de dados. Pode ser classificado em: a) aprendizado supervisionado, em que o computador é apresentado com exemplos de entradas e suas saídas desejadas, rotuladas, e o objetivo é aprender uma regra geral que mapeia entradas em saídas; b) aprendizado não supervisionado, em que nenhum rótulo é dado ao computador, deixando-o sozinho para encontrar a estrutura em sua entrada. O aprendizado não supervisionado pode ser um objetivo em si (descobrir padrões ocultos nos dados) ou um meio para um fim (aprendizado de recursos); e c) aprendizado por reforço, em que o computador interage com um ambiente dinâmico no qual deve realizar uma determinada tarefa. À medida que navega no espaço do problema, o programa recebe um feedback, que ele tenta maximizar.

O objetivo do estudo é identificar quais clientes são confiáveis dado um conjunto de características e, para isso, optou-se por utilizar algoritmos de classificação, que são cálculos preditivos usados para atribuir dados a categorias predefinidas, analisando um conjunto de dados de treinamento.

Nesse estudo serão utilizados 6 tipos de algoritmos de classificação, são eles: Árvore de Decisão, Regressão Logística, Naïve Bayes, Gradiente Descendente, KNN (K - Nearest Neighbors) e Randon Forest.

Árvores de Decisão é uma tabela de decisão sob a forma de árvore com nós e folhas sequenciais e interligados que classificam as instâncias, ordenando-as com base nos valores dos recursos. Cada nó em uma árvore de decisão representa uma característica em uma instância a ser classificada, e cada ramo representa um valor que o nó pode assumir. As instâncias são classificadas começando no nó raiz com base em seus valores de recursos. Trata-se de um dos modelos mais práticos e mais utilizados em inferência por indução.

A regressão logística está entre as técnicas mais utilizadas quando se deseja entender as correlações entre variáveis de entrada e saída de um modelo, normalmente são as primeiras a serem testadas em modelos de machinelearning. É uma técnica recomendada para situações em que a variável dependente é de natureza dicotômica ou binária. Quanto às independentes, tanto podem ser categóricas ou não. Ela é mais útil no entendimento da influência de diversas variáveis independentes em uma saída única variável. A desvantagem é que funciona apenas quando a variável prevista é binária, assume que todos os preditores são independentes uns dos outros e assume que os dados estão livres de valores ausentes.

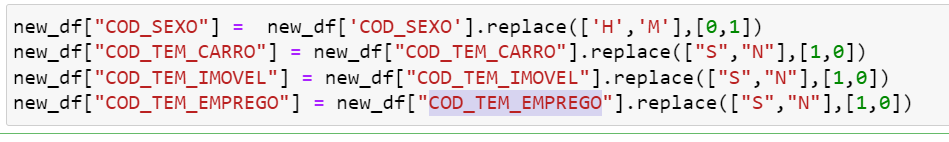
O algoritmo Naive Bayes é uma técnica de classificação com uma suposição de independência entre as variáveis preditoras. O classificador Naive(do inglês ingênuo) Bayesassume que a presença de uma característica particular em uma classe não está relacionada com a presença de qualquer outra. As principais vantagens deste algoritmo é que ele requer uma pequena quantidade de dados de treinamento para estimar os parâmetros necessários e é extremamente rápido em comparação com métodos mais sofisticados. A maior desvantagem é que ele é conhecido por não ser um bom avaliador.

O algoritmo gradiente descendente é um dos algoritmos de maior sucesso em problemas de Machine Learning. O método consiste em encontrar, de forma iterativa, os valores dos parâmetros que minimizam determinada função de interesse. As vantagens desse algoritmo são sua eficiência e facilidade de implementação, mas ele requer vários hiperparâmetros e é sensível ao dimensionamento de recursos.

O algoritmo k-nearest neighbor, muitas vezes abreviado k-nn, é uma abordagem para classificação de dados que estima a probabilidade de um ponto de dados ser membro de um grupo ou de outro, dependendo de qual grupo os pontos de dados mais próximos a ele estão. Esse algoritmo é de fácil implementação, robusto a variações nos dados de treinamento e efetivo quando temos muitos dados de treinamento. O ponto negativo desse algoritmo é que exige muito recurso computacional.

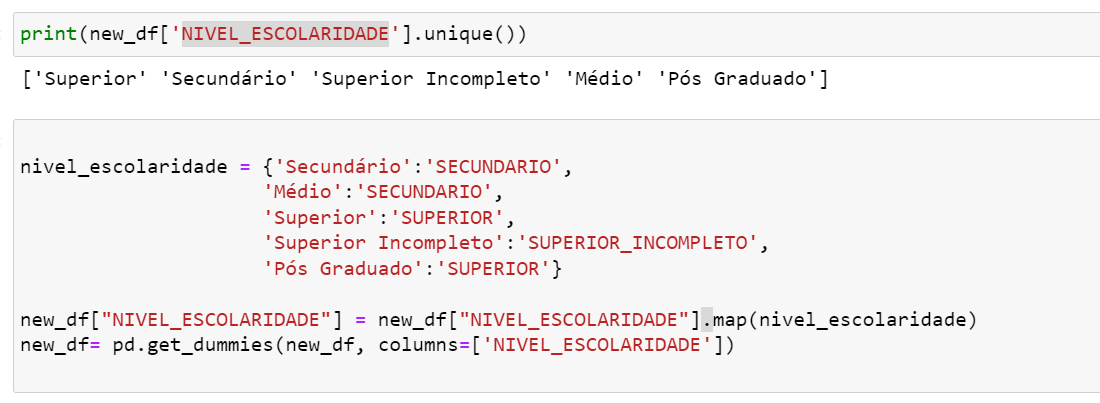
Por fim, o algoritmo Random Forest que é um algoritmo de aprendizagem supervisionada que cria uma “floresta” de um modo aleatório. A “floresta” é uma combinação (ensemble) de árvores de decisão, na maioria dos casos treinados com o método de bagging. A ideia principal do método de bagging é que a combinação dos modelos de aprendizado o que aumenta o resultado geral. Dizendo de outro modo, o algoritmo cria várias árvores de decisão e as combina para obter uma predição com maior acurácia e mais estável. As dificuldades desse modelo são sua dificuldade de implementação e complexidade do algoritmo.

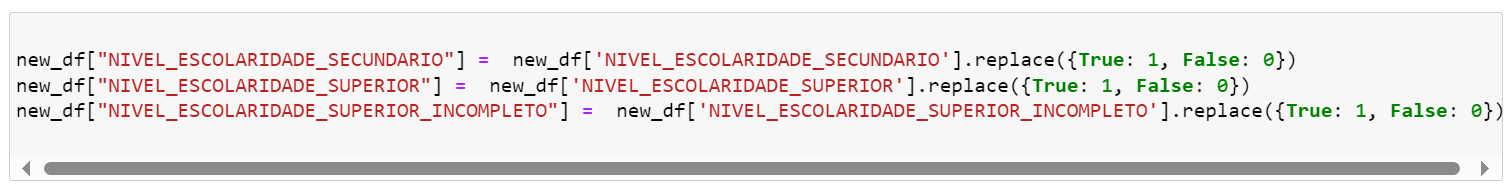
Para o trabalho com os algoritmos mencionados, os dados categóricos foram transformados para valores numéricos. Os valores das colunas COD\_SEXO, COD\_TEM\_CARRO, COD\_TEM\_IMOVEL e COD\_TEM\_EMPREGO serão substituídos por 0 ou 1.



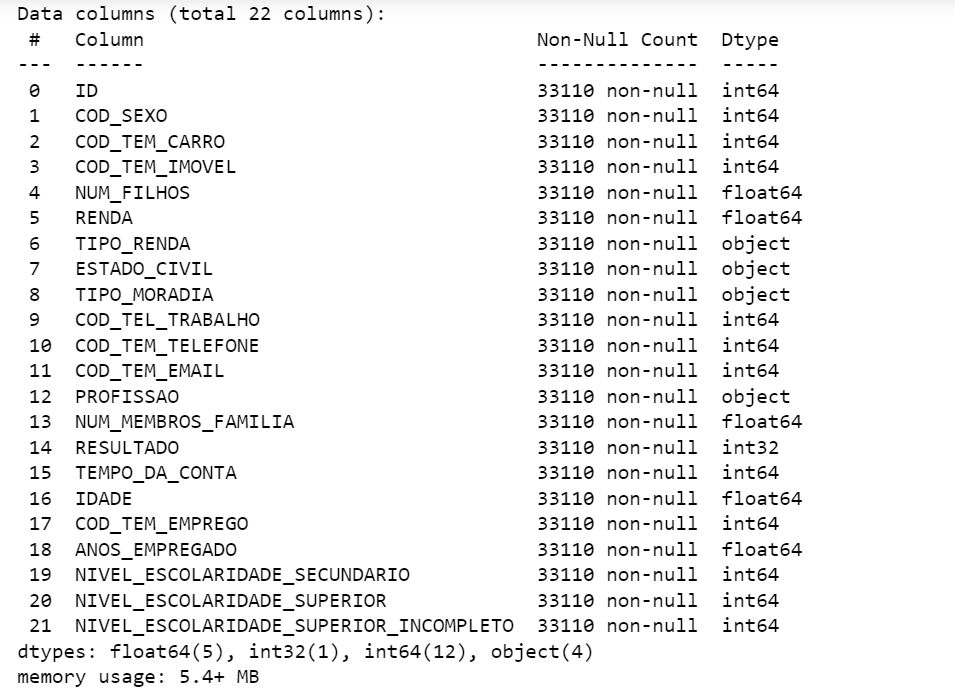
Para as demais colunas que não permitem uma classificação binária, a escolha foi utilizar a função “get\_dummies” que cria novas colunas no dataframe que terão como nome os valores de cada coluna com informações categóricas e preenche cada linha com “0” e “1” dependendo do valor que a entrada possuía.

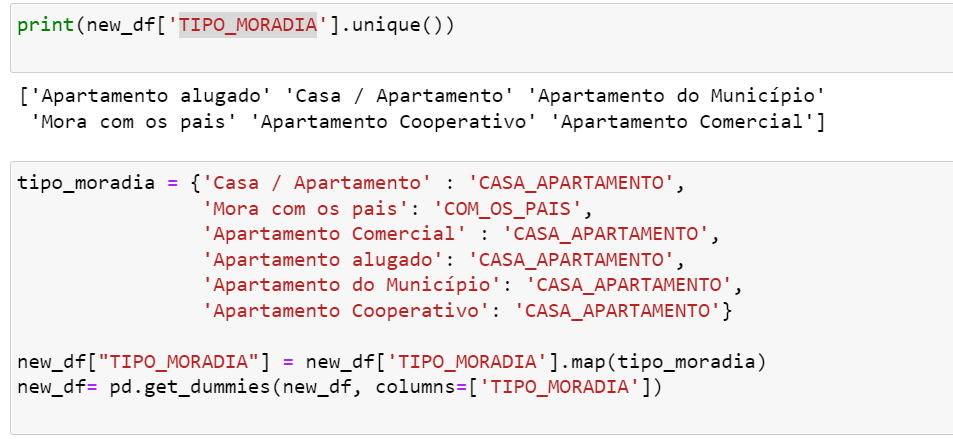
As colunas NIVEL\_ESCOLARIDADE, TIPO\_MORADIA, ESTADO\_CIVIL , TIPO\_RENDA e PROFISSAO serão sumarizadas em categorias de acordo com seus valores e transformadas para numéricas.





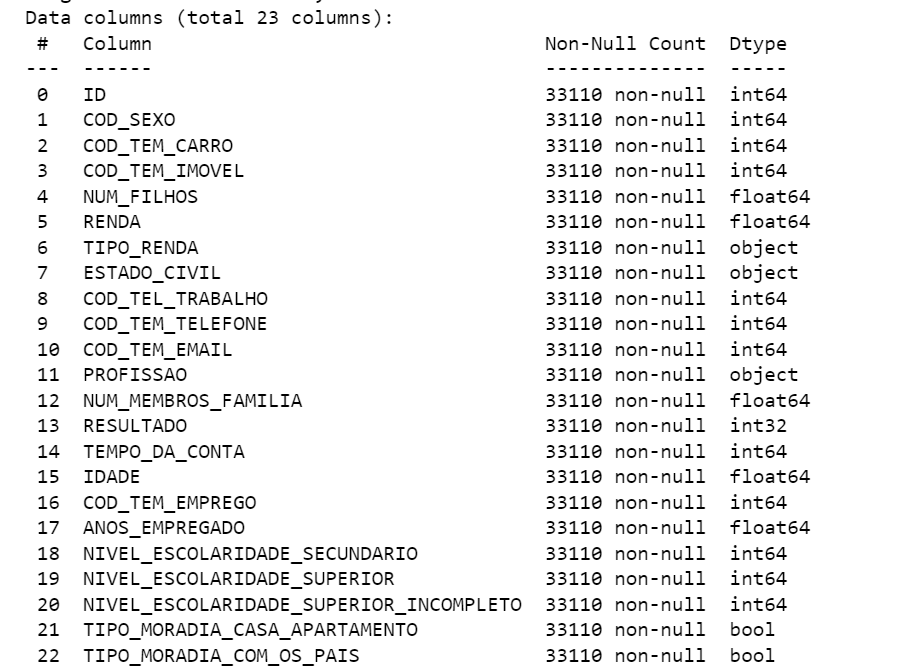






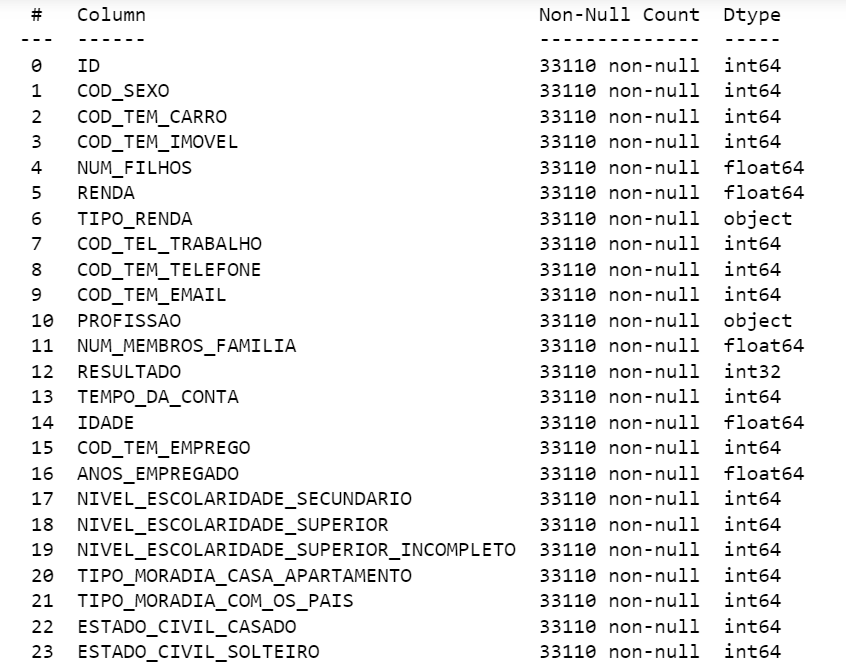






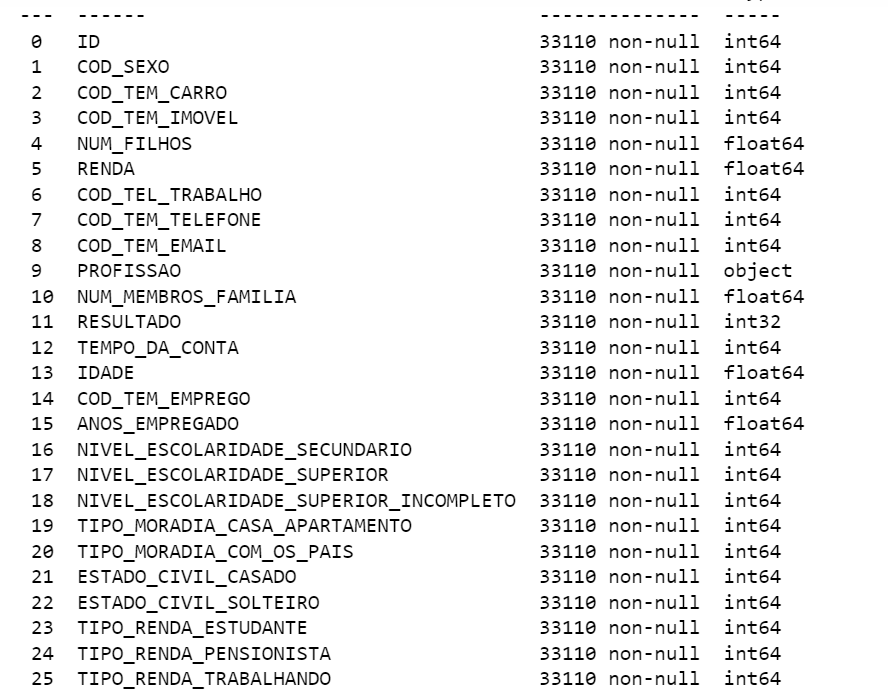


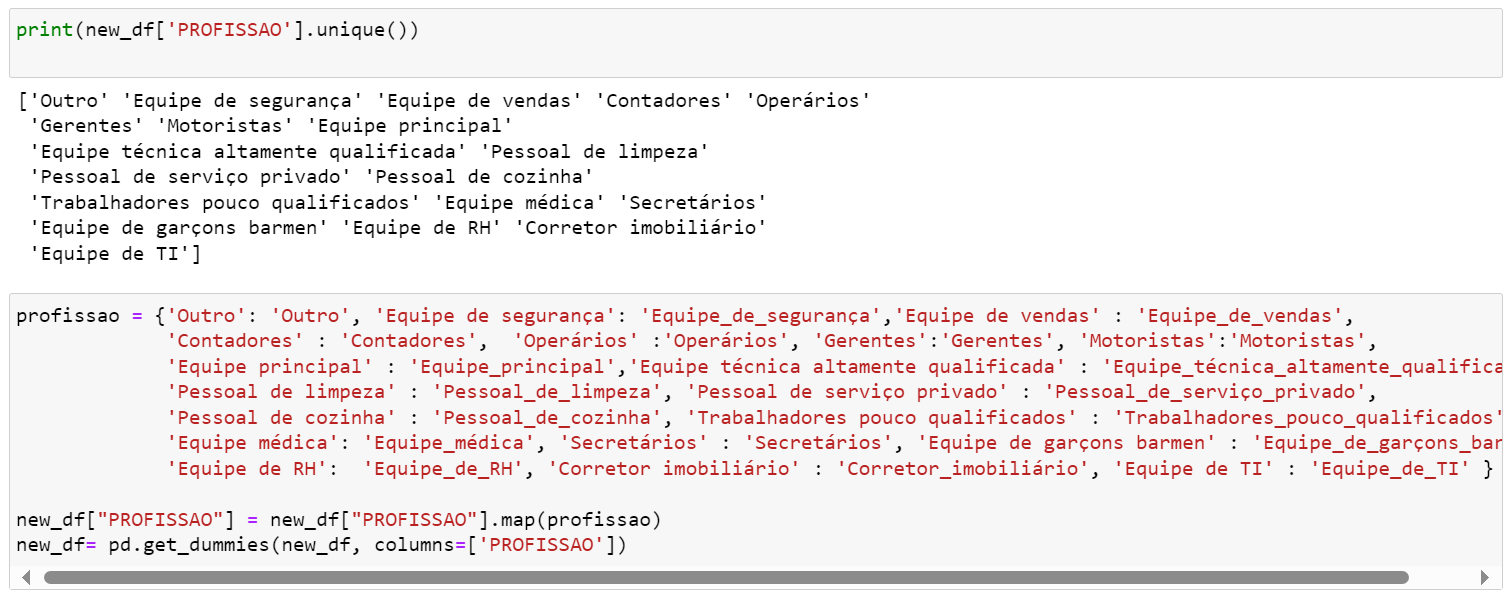






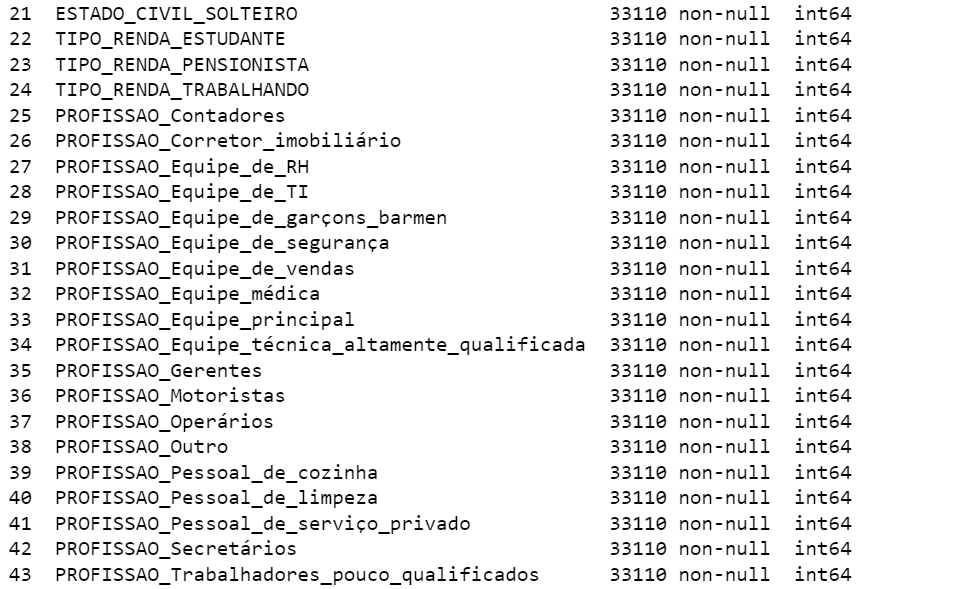






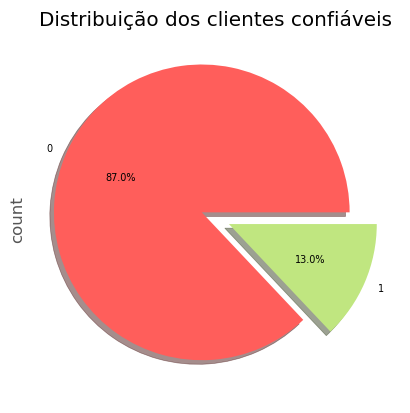


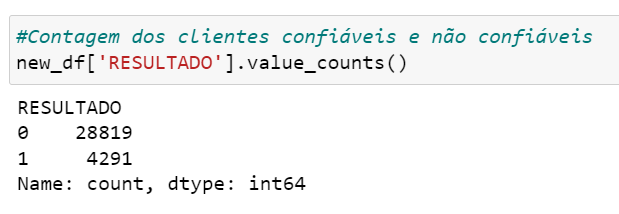




É possível perceber que todas as colunas agora são numéricas.

A análise do trabalho avalia a previsibilidade de um cliente ser confiável ou não com base em alguns parâmetros. Dessa forma, a informação que utilizaremos como resultado é a coluna RESULTADO, que traz se o cliente é confiável. Porém, há muito mais clientes confiáveis no dataframe conforme o gráfico, 88.2 % de clientes confiáveis.

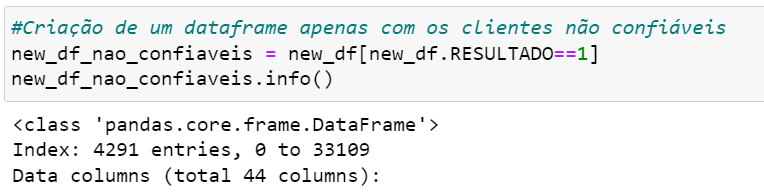




Essa proporção prejudica a construção do algoritmo porque uma simples afirmação de que o cliente será confiável acertaria em 87% das vezes, nesse caso concreto. Para contornar esse problema, será utilizada a biblioteca Scikit-learn, mais especificamente sua função resample, que, após alguns passos, igualará a quantidade de amostras de clientes confiáveis ou não.

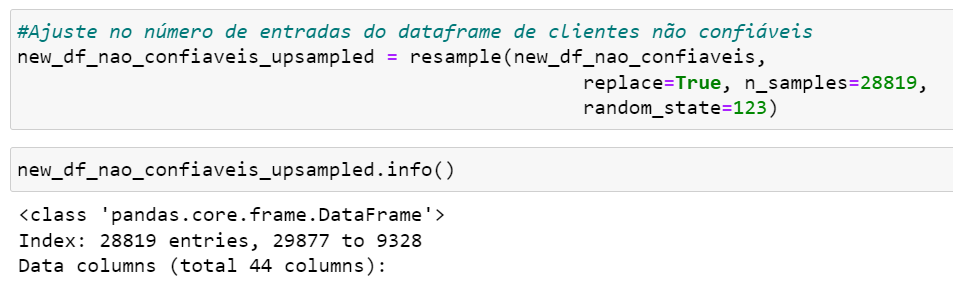


Logo após, o dataframe será dividido de acordo o valor da coluna RESULTADO, que possui valor 0 para clientes confiáveis e 1 para não confiáveis.

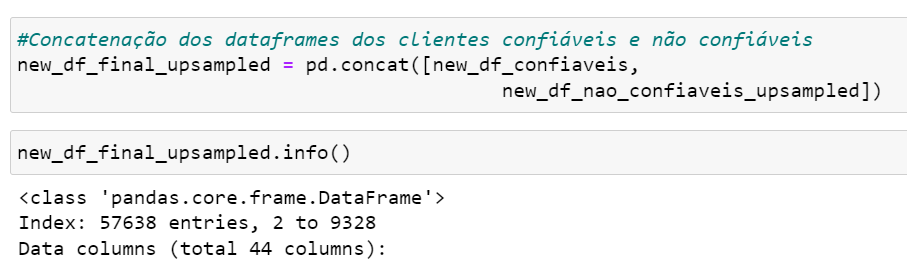


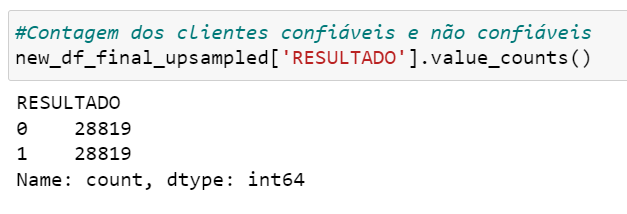


Em seguida, esses novos dataframes terão as mesmas quantidades de registros conforme o comando:



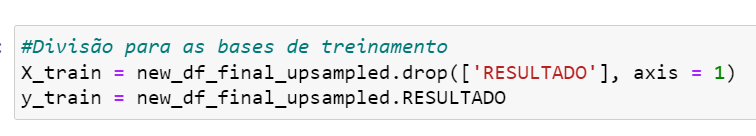
Utilizando a função pd.concat, os dois dataframes com os mesmo números de registros serão concatenados para uma única fonte de dados.





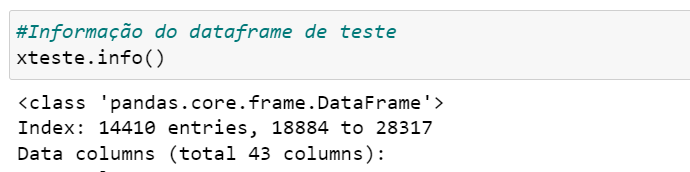
O novo dataframe possui 57638 linhas, dividido igualmente entre clientes confiáveis e não confiáveis.

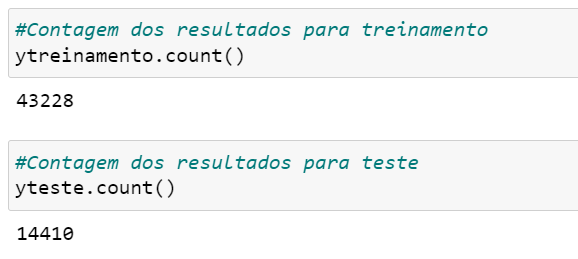
Para a aplicação dos algoritmos de classificação, o dataframe será dividido em dois, onde um datraframe consta apenas os atributos que serão analisados e outro que mostra se o cliente é confiável. O dataframe que contém todos os atributos será chamado do de X\_train e o que contém o resultado será chamado de y\_train.



O próximo passo para a aplicação dos algoritmos de classificação é dividir o dataframe em bases de treinamento e de teste. Para esse procedimento será utilizado a função “train\_test\_split”, também da biblioteca Scikit-learn. Para a divisão de percentual de dados para treinamento e para teste, será utilizado o padrão da função, que é de 75% para treinamento e 25% para teste.

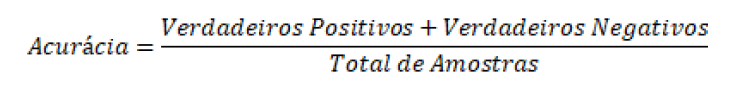




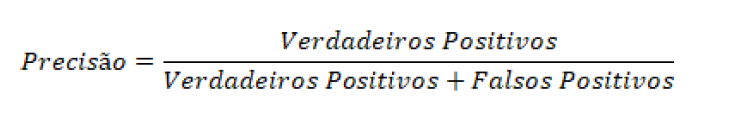


Utilizando as funções “accuracy\_score” e “classification\_report” da biblioteca Scikit-learn, será definida qual medida de avaliação será analisada para verificar a eficiência do modelo. Essas funções mostram as medidas de acurácia, precisão, revocação (recall) e f1-score.

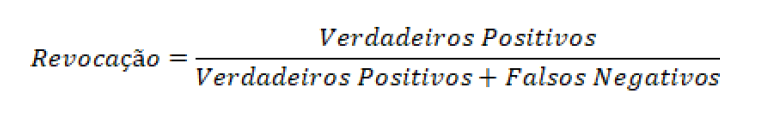
A acurácia é a quantidade de acertos do modelo dividido pelo total da amostra e indica uma performance geral do modelo:A acurácia é a quantidade de acertos do modelo dividido pelo total da amostra e indica uma performance geral do modelo:



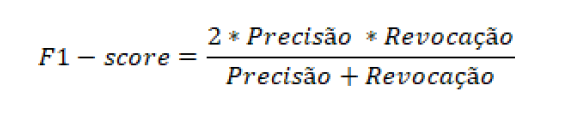
A precisão define os chamados positivos verdadeiros, ou seja, dentre os exemplos classificados como verdadeiros, quantos eram realmente verdadeiros.



A revocação (recall) indica qual a porcentagem de dados classificados como verdadeiros comparado com a quantidade real de resultados verdadeiros que existem na amostra.



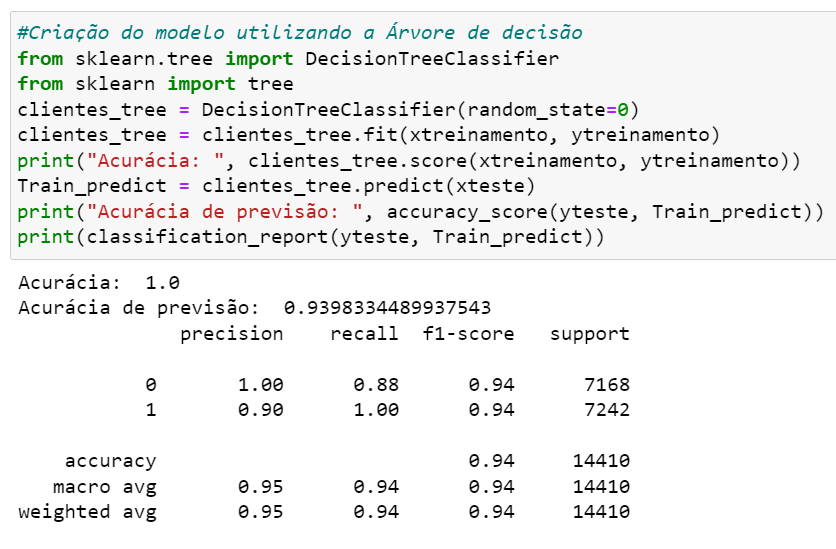
A F1-score traz a média ponderada de precisão e revocação e traz um número único que determina a qualidade geral do modelo.

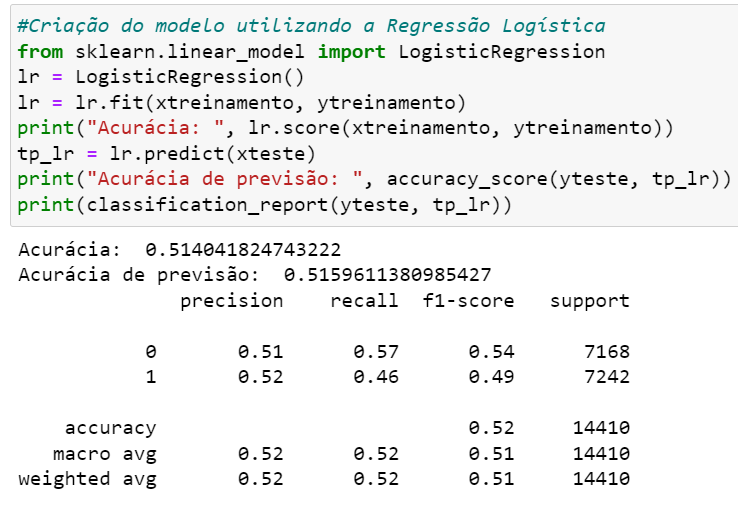


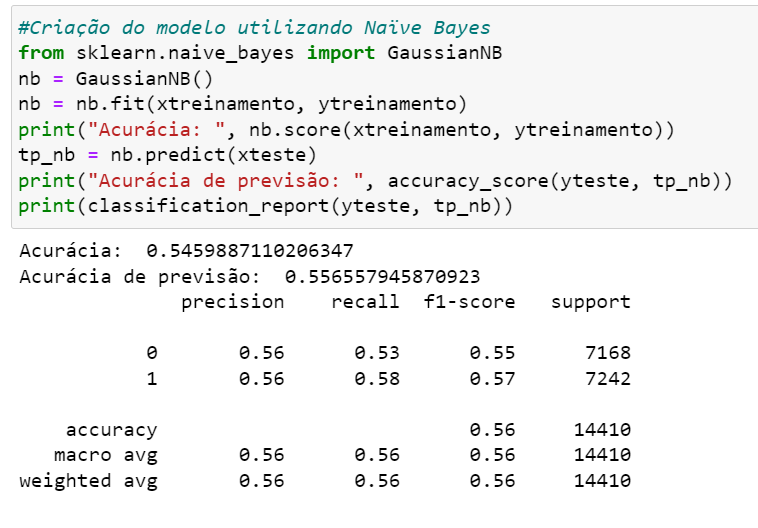
A medida de acurácia, será a principal medida utilizada nesta análise, porém todas outras medidas serão consideradas na avaliação. Essas medidas serão utilizadas através da importação das funções “accuracy\_score” e “classification\_report”.

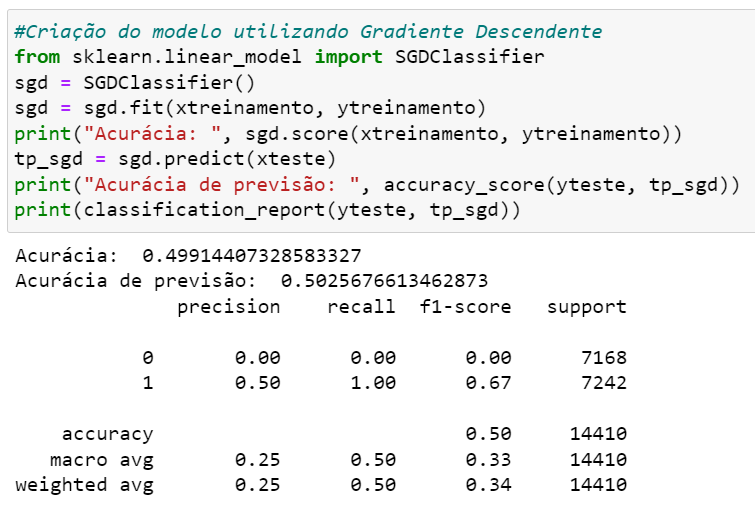


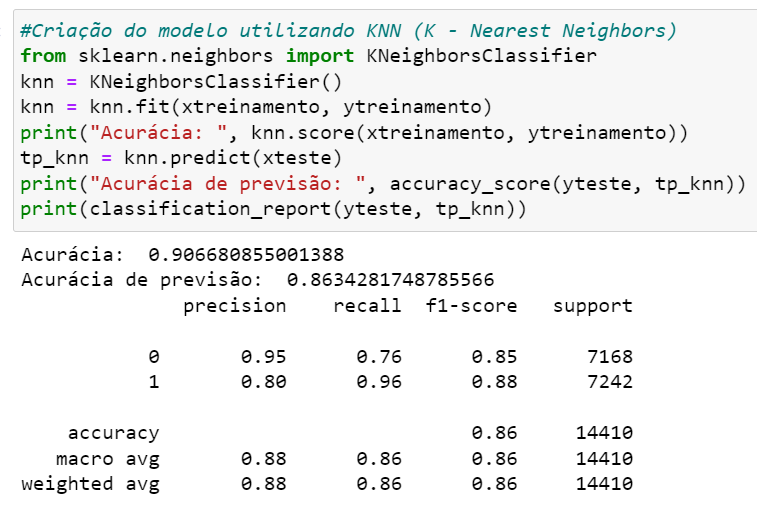
O próximo passo é a utilização dos algoritmos listados anteriormente, através dos dataframes de treinamento e de teste. Escolhidas as medidas de avaliação, o processo para todos dos algoritmos será o mesmo, começando pela importação do respectivo algoritmo, realizando o treinamento por meio da função “fit” nas duas bases de treinamento e registrando sua acurácia por meio da função score. Em seguida, será utilizada a função “predict” na base teste xteste e sua saída será comparada com a série yteste, tendo suas medidas de avaliação sendo geradas por meio das funções “accuracy\_score” e “classification\_report”.

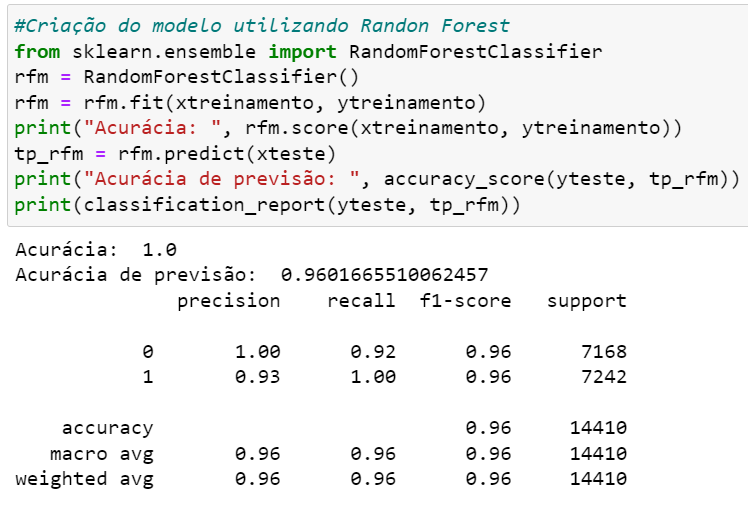






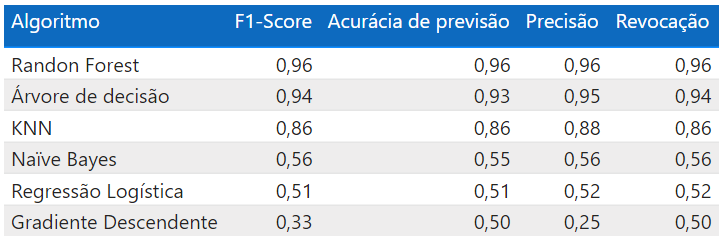


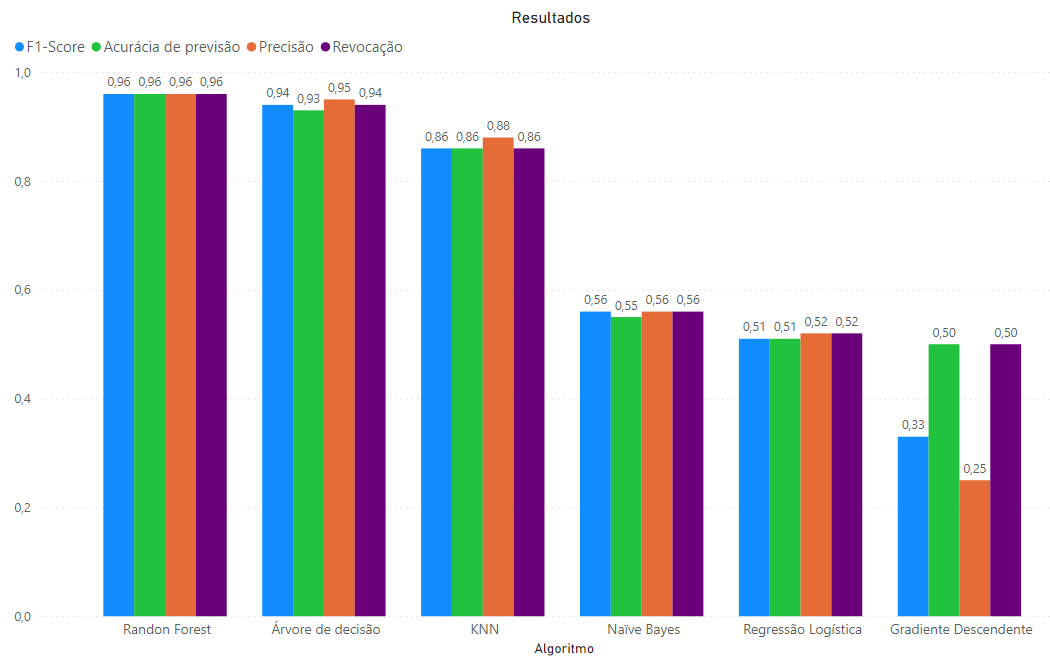




**6. Apresentação dos Resultados**

A tabela mostra os resultados médios dos algoritmos utilizados na análise:





A partir dos resultados apresentados é possível perceber que, comparativamente, os modelos Randon Forest e Árvore de decisão tiveram um melhor desempenho, com uma vantagem para o algoritmo Randon Forest.

Ao analisar o modelo Randon Forest, a precisão obtida para os clientes confiáveis apresenta um ótimo resultado com o valor 1.00, ou seja, algoritmo não identificou nenhum falso positivo no momento de identificar clientes confiáveis. O valor de 0.93 para os clientes não confiáveis também apresenta um bom resultando, demonstrando que para cada 100 previsões de um cliente não ser confiável, em apenas 7 previsões se teria um falso positivo. Porém, para os resultados de revocação o valor 1.00 é para um cliente não ser confiável, não identificando falso negativo. Para os clientes confiáveis, a cada 100 previsões, 8 seriam de falso negativo, indicando um valor de 0.92. Analisando a acurácia de previsão, temos o valor de 0,96, ou seja, de todas as amostras, o algoritmo acerta em 96% das vezes, afirmando que um cliente será confiável ou não.

O algoritmo Árvore de decisão demonstra uma precisão de 1.00 para os clientes confiáveis, sem falso positivo, e uma precisão de 0.90 para clientes não confiáveis, onde a cada 100 previsões, 10 seriam de falso negativo. Um maior valor da revocação para clientes não confiáveis, é semelhante ao que ocorreu no modelo Randon Forest. A cada 100 previsões de um cliente ser confiável, em 12 previsões se teria um falso positivo e nenhum falso positivo para um cliente não ser confiável. O valor de 0.93 para acurácia de previsão, demonstra que há um acerto de 93% em afirmar se um cliente é confiável ou não.

Por fim, concluiu-se que os modelos Randon Forest e Árvore de decisão apresentaram melhor capacidade de classificação, obtendo o melhor resultado nas métricas. Cabe agora testar os modelos em conjuntos de dados reais para verificar se o modelo está corretamente ajustado sendo capaz de atender aos objetivos do negócio.

**7. Links**

**Link para o vídeo :**

<https://youtu.be/Y7iVP1tuo0M>

**Link para o repositório Github :**

<https://github.com/henrique-m-oliveira/TCC---Big-Data>

**Link para o repositório Google Drive :**

https://drive.google.com/drive/folders/1VBsUZESYTF7MO9Kwi21cAWHSMzALmbIM?usp=sharing

**APÊNDICE**

**Programação**

import warnings  
warnings.filterwarnings('ignore')  
  
import numpy as np   
import pandas as pd   
  
import matplotlib  
import matplotlib.pyplot as plt  
import matplotlib.style as style  
from matplotlib import colors  
from mpl\_toolkits.mplot3d import Axes3D  
from matplotlib.colors import ListedColormap  
from IPython.display import Image  
import plotly.express as px  
import plotly.graph\_objs as go  
import plotly.figure\_factory as ff  
from plotly.subplots import make\_subplots  
import plotly.offline as pyo  
from plotly import tools  
import seaborn as sns  
  
import missingno as msno   
  
from imblearn.over\_sampling import SMOTE  
import itertools  
  
  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, confusion\_matrix,precision\_score,recall\_score,roc\_auc\_score,f1\_score,roc\_curve  
  
from sklearn.linear\_model import LogisticRegression  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.svm import SVC  
  
from xgboost import XGBClassifier  
from lightgbm import LGBMClassifier  
from catboost import CatBoostClassifier  
from sklearn import svm  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.neighbors import KNeighborsClassifier  
  
from sklearn.preprocessing import LabelEncoder   
  
pyo.init\_notebook\_mode()

# Processamento/Tratamento de Dados

O processamento e o tratamento dos dados foram feitos utilizando a linguagem Python, versão 3.7.0, no ambiente Jupyter Notebook, versão 5.6.0. Dentro da linguagem, utilizou-se a biblioteca “pandas” que é uma poderosa ferramenta para tratamento e análise de dados.

Em seguida, deve-se fazer a leitura e tratamento dos datasets, que, nesse caso, será feita individualmente para cada um deles

cliente = pd.read\_csv("application\_record.csv", encoding = 'utf-8') # é o arquivo que contém informações sobre todos os clientes quanto à sua situação socioeconômica   
transacao = pd.read\_csv("credit\_record.csv", encoding = 'utf-8') #é o arquivo que contém todos os registros de pagamento/inadimplência de um determinado cliente

Para se obter informações do dataframe, pode-se utilizar a função “info()” que, no caso específico, mostrou que o dataframe possui 438557 entradas, divididas nas 18 colunas apresentadas anteriormente. falar tambem do outro dataframe

cliente.info()

transacao.info()

Interessante perceber que uma informação potencialmente útil para o estudo não foi considerada, a informação da coluna FLAG\_MOBIL. Isso se deu pelo fato de que essa coluna é uma constante e por isso foi desconsiderada.

cliente.nunique()

transacao.nunique()

cliente.drop('FLAG\_MOBIL', axis=1, inplace=True)

cliente.info()

Percebe-se que agora há apenas as 16 colunas escolhidas, mas ainda com as 438557 entradas filtradas anteriormente

Para verificar se há dados nulos no dataframe, utilizou-se a função “isnull()”, que faz essa análise, em conjunto com a função “sum()”, que, nesse caso, soma os valores encontrados na função anterior.

cliente.isna().sum()

transacao.isna().sum()

Após esse comando, verificou-se que o dataframe cliente, apresenta dados nulos na coluna OCCUPATION\_TYPE que necessitam de tratamento. Nesse caso, os dados nulos serão substituídos pelo valor 'Other'.

cliente['OCCUPATION\_TYPE'].fillna(value='Other', inplace=True)

cliente.isna().sum()

Cada linha no dataframe cliente, deve corresponder a um único ID.

print("Número de linhas no dataset application\_record.csv: {}".format(len(cliente)))  
print("Número de clientes distintos no dataset application\_record.csv : {}".format(len(cliente.ID.unique())))

Para remoção dos registros duplicados foi utilizado o método “drop\_duplicates”.

cliente=cliente.drop\_duplicates(subset=cliente.columns[0], keep='first')

print("Número de linhas no dataset application\_record.csv: {}".format(len(cliente)))  
print("Número de clientes distintos no dataset application\_record.csv : {}".format(len(cliente.ID.unique())))

print("Número de linhas duplicadas no dataset credit\_record.csv : {}".format(transacao.duplicated().sum()))

Para modelos de machine learning supervisionados é fundamental a etapa de criação de uma coluna target no dataset. Em problemas que isso é possível, faz com que os modelos e resultados obtidos possam ser explorados com maior facilidade.

O caso em análise busca avaliar a previsibilidade se um candidato é um cliente 'bom' ou 'ruim' com base em alguns parâmetros. Dessa forma, a informação que utilizaremos como resultado é a coluna STATUS, que traz os dias em atraso no pagamento.

0: 1 a 29 dias de atraso 1: 30 a 59 dias de atraso 2: 60 a 89 dias de atraso 3: 90 a 119 dias de atraso 4: 120 a 149 dias de atraso 5: Mais de 150 dias de atraso ou dívidas inadimplentes, baixa C : Quitados no mês X: Não há empréstimos durante o mês

As linhas que possuem target=1 representam os clientes considerados “maus pagadores” QUE atrasaram os pagamentos por mais de 30 dias pelo menos uma vez em seu histórico de pagamentos. Isso incluiu clientes com códigos específicos (1, 2, 3, 4 ou 5) em seu histórico mensal.

As linhas que possuem target=0 representam os clientes com histórico consistente de pagamentos pontuais (status “C”) ou atrasos de até 29 dias (status “0”) foram categorizados como de baixo risco e considerados perfis adequados para aprovação de cartão de crédito .

As linhas que possuem target=-1 representam os clientes que não utilizaram o cartão de crédito, indicando falta de histórico de transações.

transacao['STATUS'].value\_counts()

transacao['target']=transacao['STATUS']  
transacao['target'].replace('X', -1, inplace=True)  
transacao['target'].replace('C', 0, inplace=True)  
transacao['target']=transacao['target'].astype(int)  
transacao.loc[transacao['target']>=1,'target']=1

transacao['target'].value\_counts()

# O valor de -1 deve ser eliminado da construção do modelo

transacao = transacao[transacao['target']!=-1]

transacao['target'].value\_counts()

Integração (merge/join) dos datasets. O objetivo agora é criar um único dataset consolidado, gerado a partir da integração (join/merge) dos datasets gerados. Para isso, foi necessário agrupar o dataset transacao, por meio de seu ID de cliente, através dos comandos “groupby” e “agg(max)” que, em conjunto, selecionam o valor máximo de target para cada cliente.

transacao\_df=pd.DataFrame(transacao.groupby(['ID'])['target'].agg(max)).reset\_index()

transacao\_df.info()

transacao\_df['target'].value\_counts()

# Merge dataframes  
new\_df=pd.merge(cliente, transacao\_df, how='inner', on=['ID'])

new\_df.info()

Faz sentido que o número de meses em que a conta está aberta, seja correlacionado com o risco do cliente (uma vez que há mais oportunidades de perder pagamentos).

# Meses em que a conta está aberta  
inicio\_df=pd.DataFrame(transacao.groupby(['ID'])['MONTHS\_BALANCE'].agg(min)).reset\_index()  
  
# Renomear a coluna  
inicio\_df.rename(columns={'MONTHS\_BALANCE':'ACCOUNT\_LENGTH'}, inplace=True)  
  
# Meses como número positivo  
inicio\_df['ACCOUNT\_LENGTH']=-inicio\_df['ACCOUNT\_LENGTH']  
  
# Merge dataframes   
new\_df=pd.merge(new\_df, inicio\_df, how='inner', on=['ID'])  
  
new\_df.info()

# Criar a variável idade  
new\_df['AGE\_YEARS']=-new\_df['DAYS\_BIRTH']/365.2425  
new\_df.drop('DAYS\_BIRTH', axis=1, inplace=True)  
  
# Criar um indicador se está empregado ou não  
new\_df['EMPLOYED']='S'  
new\_df.loc[-new\_df['DAYS\_EMPLOYED']<0,'EMPLOYED']='N'  
  
# Criar a variável anos em que está empregado  
new\_df['YEARS\_EMPLOYED']=-new\_df['DAYS\_EMPLOYED']/365.2425  
new\_df.loc[new\_df['YEARS\_EMPLOYED']<0,'YEARS\_EMPLOYED']=0  
new\_df.drop('DAYS\_EMPLOYED', axis=1, inplace=True)

new\_df.head()

Renomear as colunas

new\_df = new\_df.rename(columns={'CODE\_GENDER': 'COD\_SEXO', 'FLAG\_OWN\_CAR': 'COD\_TEM\_CARRO',   
 'FLAG\_OWN\_REALTY':'COD\_TEM\_IMOVEL','CNT\_CHILDREN':'NUM\_FILHOS',  
 'AMT\_INCOME\_TOTAL':'RENDA','NAME\_INCOME\_TYPE':'TIPO\_RENDA',  
 'NAME\_EDUCATION\_TYPE':'NIVEL\_ESCOLARIDADE','NAME\_FAMILY\_STATUS':'ESTADO\_CIVIL',  
 'NAME\_HOUSING\_TYPE':'TIPO\_MORADIA','FLAG\_WORK\_PHONE':'COD\_TEL\_TRABALHO',  
 'FLAG\_PHONE':'COD\_TEM\_TELEFONE','FLAG\_EMAIL':'COD\_TEM\_EMAIL','OCCUPATION\_TYPE':'PROFISSAO',  
 'CNT\_FAM\_MEMBERS':'NUM\_MEMBROS\_FAMILIA','target':'RESULTADO','ACCOUNT\_LENGTH':'TEMPO\_DA\_CONTA',  
 'AGE\_YEARS':'IDADE','EMPLOYED':'COD\_TEM\_EMPREGO','YEARS\_EMPLOYED':'ANOS\_EMPREGADO'})

new\_df.info()

new\_df.head()

#new\_df.dtypes

Vamos transformar os valores de algumas colunas para português

print(new\_df['PROFISSAO'].unique())

print(new\_df['TIPO\_RENDA'].unique())

print(new\_df['TIPO\_MORADIA'].unique())

print(new\_df['NIVEL\_ESCOLARIDADE'].unique())

print(new\_df['ESTADO\_CIVIL'].unique())

new\_df['COD\_SEXO'] = new\_df['COD\_SEXO'].replace(['M'], 'H')  
new\_df['COD\_SEXO'] = new\_df['COD\_SEXO'].replace(['F'], 'M')  
  
new\_df['COD\_TEM\_CARRO'] = new\_df['COD\_TEM\_CARRO'].replace(['Y'], 'S')  
new\_df['COD\_TEM\_IMOVEL'] = new\_df['COD\_TEM\_IMOVEL'].replace(['Y'], 'S')  
  
  
new\_df['TIPO\_RENDA'] = new\_df['TIPO\_RENDA'].replace(['Commercial associate'], 'Sócio Comercial')  
new\_df['TIPO\_RENDA'] = new\_df['TIPO\_RENDA'].replace(['Working'], 'Trabalhador da Área Privada')  
new\_df['TIPO\_RENDA'] = new\_df['TIPO\_RENDA'].replace(['Pensioner'], 'Pensionista')  
new\_df['TIPO\_RENDA'] = new\_df['TIPO\_RENDA'].replace(['State servant'], 'Servidor Público')  
new\_df['TIPO\_RENDA'] = new\_df['TIPO\_RENDA'].replace(['Student'], 'Estudante')  
  
  
new\_df['TIPO\_MORADIA'] = new\_df['TIPO\_MORADIA'].replace(['House / apartment'], 'Casa / Apartamento')  
new\_df['TIPO\_MORADIA'] = new\_df['TIPO\_MORADIA'].replace(['Rented apartment'], 'Apartamento alugado')  
new\_df['TIPO\_MORADIA'] = new\_df['TIPO\_MORADIA'].replace(['With parents'], 'Mora com os pais')  
new\_df['TIPO\_MORADIA'] = new\_df['TIPO\_MORADIA'].replace(['Municipal apartment'], 'Apartamento do Município')  
new\_df['TIPO\_MORADIA'] = new\_df['TIPO\_MORADIA'].replace(['Office apartment'], 'Apartamento Comercial')  
new\_df['TIPO\_MORADIA'] = new\_df['TIPO\_MORADIA'].replace(['Co-op apartment'], 'Apartamento Cooperativo')  
  
new\_df['NIVEL\_ESCOLARIDADE'] = new\_df['NIVEL\_ESCOLARIDADE'].replace(['Secondary / secondary special'], 'Secundário')  
new\_df['NIVEL\_ESCOLARIDADE'] = new\_df['NIVEL\_ESCOLARIDADE'].replace(['Lower secondary'], 'Médio')  
new\_df['NIVEL\_ESCOLARIDADE'] = new\_df['NIVEL\_ESCOLARIDADE'].replace(['Higher education'], 'Superior')  
new\_df['NIVEL\_ESCOLARIDADE'] = new\_df['NIVEL\_ESCOLARIDADE'].replace(['Incomplete higher'], 'Superior Incompleto')  
new\_df['NIVEL\_ESCOLARIDADE'] = new\_df['NIVEL\_ESCOLARIDADE'].replace(['Academic degree'], 'Pós Graduado')  
  
  
new\_df['ESTADO\_CIVIL'] = new\_df['ESTADO\_CIVIL'].replace(['Single / not married'], 'Solteiro')  
new\_df['ESTADO\_CIVIL'] = new\_df['ESTADO\_CIVIL'].replace(['Separated'], 'Separado')  
new\_df['ESTADO\_CIVIL'] = new\_df['ESTADO\_CIVIL'].replace(['Widow'], 'Viúvo')  
new\_df['ESTADO\_CIVIL'] = new\_df['ESTADO\_CIVIL'].replace(['Civil marriage'], 'Casado no civil')  
new\_df['ESTADO\_CIVIL'] = new\_df['ESTADO\_CIVIL'].replace(['Married'], 'Casado')  
  
  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Security staff' ], 'Equipe de segurança')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Sales staff' ],'Equipe de vendas')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Accountants' ],'Contadores')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Laborers' ],'Operários')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Managers' ],'Gerentes')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Drivers' ],'Motoristas')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Core staff' ], 'Equipe principal')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['High skill tech staff' ],'Equipe técnica altamente qualificada')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Cleaning staff'],'Pessoal de limpeza')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Private service staff' ],'Pessoal de serviço privado')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Cooking staff'],'Pessoal de cozinha')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Low-skill Laborers'],'Trabalhadores pouco qualificados')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Secretaries' ], 'Secretários')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Medicine staff' ], 'Equipe médica')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Waiters/barmen staff' ], 'Equipe de garçons barmen')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['HR staff'], 'Equipe de RH')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Realty agents' ], 'Corretor imobiliário')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['IT staff'],'Equipe de TI')  
new\_df['PROFISSAO'] = new\_df['PROFISSAO'].replace(['Other'],'Outro')  
  
new\_df['NUM\_MEMBROS\_FAMILIA']=new\_df['NUM\_MEMBROS\_FAMILIA'].astype(int)

new\_df.info()

# Remover outliers

fig = make\_subplots(rows=3, cols=2, start\_cell="bottom-left",  
 subplot\_titles=("Renda", "Idade", "Anos de Emprego", "Membros da família","Número de filhos","Tempo como cliente"))  
  
fig.add\_trace(go.Box(x=new\_df.RENDA, name='Renda',boxmean=True),row=1,col=1)  
fig.add\_trace(go.Box(x=new\_df.IDADE, name='Idade', boxmean=True), row=1, col=2)  
fig.add\_trace(go.Box(x=new\_df.ANOS\_EMPREGADO, name='Anos de Emprego', boxmean=True), row=2, col=1)  
fig.add\_trace(go.Box(x=new\_df.NUM\_MEMBROS\_FAMILIA, name="Membros da família", boxmean=True),row=2, col=2)  
fig.add\_trace(go.Box(x=new\_df.NUM\_FILHOS, name="Número de filhos", boxmean=True),row=3, col=1)  
fig.add\_trace(go.Box(x=new\_df.TEMPO\_DA\_CONTA, name="Tempo como cliente", boxmean=True),row=3, col=2)  
  
  
  
  
fig.show()

# Função para descobrir os outliers   
def find\_outliers\_IQR(df):  
  
 q1=df.quantile(0.25)  
  
 q3=df.quantile(0.75)  
  
 IQR=q3-q1  
  
 outliers = df[((df<(q1-1.5\*IQR)) | (df>(q3+1.5\*IQR)))]  
  
 return outliers

# Função para substituir os outliers por valores médios  
def impute\_outliers\_IQR(df):  
  
 q1=df.quantile(0.25)  
  
 q3=df.quantile(0.75)  
  
 IQR=q3-q1  
  
 upper = df[~(df>(q3+1.5\*IQR))].max()  
  
 lower = df[~(df<(q1-1.5\*IQR))].min()  
  
 df = np.where(df > upper,  
 df.mean(),  
 np.where(  
 df < lower,  
 df.mean(),  
 df  
 )  
 )  
  
 return df

outliers = find\_outliers\_IQR(new\_df.NUM\_MEMBROS\_FAMILIA)  
  
print("número de outliers NUM\_MEMBROS\_FAMILIA : " + str(len(outliers)))  
print("max outlier : " + str(outliers.max()))  
print("min outlier : " + str(outliers.min()))  
  
outliers = find\_outliers\_IQR(new\_df.ANOS\_EMPREGADO)  
print("número de outliers ANOS\_EMPREGADO : " + str(len(outliers)))  
print("max outlier : " + str(outliers.max()))  
print("min outlier : " + str(outliers.min()))  
  
  
outliers = find\_outliers\_IQR(new\_df.NUM\_FILHOS)  
print("número de outliers NUM\_FILHOS : " + str(len(outliers)))  
print("max outlier : " + str(outliers.max()))  
print("min outlier : " + str(outliers.min()))  
  
outliers = find\_outliers\_IQR(new\_df.RENDA)  
print("número de outliers RENDA : " + str(len(outliers)))  
print("max outlier : " + str(outliers.max()))  
print("min outlier : " + str(outliers.min()))

new\_df['NUM\_MEMBROS\_FAMILIA'] = impute\_outliers\_IQR(new\_df.NUM\_MEMBROS\_FAMILIA)  
new\_df['RENDA'] = impute\_outliers\_IQR(new\_df.RENDA)  
new\_df['ANOS\_EMPREGADO'] = impute\_outliers\_IQR(new\_df.ANOS\_EMPREGADO)  
new\_df['NUM\_FILHOS'] = impute\_outliers\_IQR(new\_df.NUM\_FILHOS)

fig = make\_subplots(rows=3, cols=2, start\_cell="bottom-left",  
 subplot\_titles=("Renda", "Idade", "Anos de Emprego", "Membros da família","Número de filhos","Tempo como cliente"))  
  
fig.add\_trace(go.Box(x=new\_df.RENDA, name='Renda',boxmean=True),row=1,col=1)  
fig.add\_trace(go.Box(x=new\_df.IDADE, name='Idade', boxmean=True), row=1, col=2)  
fig.add\_trace(go.Box(x=new\_df.ANOS\_EMPREGADO, name='Anos de Emprego', boxmean=True), row=2, col=1)  
fig.add\_trace(go.Box(x=new\_df.NUM\_MEMBROS\_FAMILIA, name="Membros da família", boxmean=True),row=2, col=2)  
fig.add\_trace(go.Box(x=new\_df.NUM\_FILHOS, name="Número de filhos", boxmean=True),row=3, col=1)  
fig.add\_trace(go.Box(x=new\_df.TEMPO\_DA\_CONTA, name="Tempo como cliente", boxmean=True),row=3, col=2)  
  
  
fig.show()

# Análise e Exploração dos Dados

#Criação de um dataframe df\_consulta apenas com clientes confiáveis  
clientes\_confiaveis = new\_df[(new\_df['RESULTADO']==0)]

clientes\_confiaveis.info()

#Contagem das opções da coluna COD\_SEXO  
from collections import Counter  
genero = Counter(new\_df['COD\_SEXO'])  
genero

#Contagem das opções da coluna COD\_SEXO dos clientes confiáveis  
genero\_clientes\_confiaveis = Counter(clientes\_confiaveis['COD\_SEXO'])  
genero\_clientes\_confiaveis

#Plotagem das informações de gênero dos clientes   
  
fig, axes = plt.subplots(1,2)  
  
  
g1= new\_df['COD\_SEXO'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True, colors=["#76B5B3","#EC9B9A"],textprops = {'fontsize':12}, ax=axes[0])  
g1.set\_title("Clientes")  
  
g2= clientes\_confiaveis['COD\_SEXO'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True,colors=["#80DE99","#00CECB"],textprops = {'fontsize':12}, ax=axes[1])  
g2.set\_title("Clientes confiáveis")  
  
  
fig.set\_size\_inches(14,5)

#Descrição estatística das idades dos clientes  
new\_df['IDADE'].describe()

#Descrição estatística das idades dos clientes confiáveis  
clientes\_confiaveis['IDADE'].describe()

#Plotagem de histograma com as idades dos clientes  
new\_df.IDADE.hist(bins=20)  
plt.style.use('ggplot')  
plt.xlabel("Anos")  
plt.ylabel("Número de clientes")  
plt.title("Idade dos clientes")  
plt.show()

#Plotagem de histograma com as idades dos clientes confiáveis  
clientes\_confiaveis.IDADE.hist(bins=20)  
plt.style.use('seaborn-pastel')  
plt.xlabel("Anos")  
plt.ylabel("Número de clientes confiáveis")  
plt.title("Idade dos clientes confiáveis")  
plt.show()

#Contagem dos níveis de escolaridade dos clientes  
  
escolaridade\_clientes = Counter(new\_df['NIVEL\_ESCOLARIDADE'].sort\_values())  
escolaridade\_clientes

#Percentuais de escolaridade dos clientes  
  
labels = []  
sizes = []  
n\_clientes=sum(escolaridade\_clientes.values())  
for x, y in escolaridade\_clientes.items():  
 a = y/n\_clientes\*100  
 labels.append(str(x) + ': ' + str(round(a,2)) + '%')  
 sizes.append(round(a,2))  
  
  
fig1, ax1 = plt.subplots(figsize=(6, 5))  
fig1.subplots\_adjust(0.3,0,1,1)  
  
  
theme = plt.get\_cmap('bwr')  
ax1.set\_prop\_cycle("color", [theme(1. \* i / len(sizes)) for i in range(len(sizes))])  
  
\_, \_ = ax1.pie(sizes, startangle=90)  
  
ax1.axis('equal')  
  
total = sum(sizes)  
plt.legend(  
 loc='center left',  
 labels=labels,  
 prop={'size': 8},  
 bbox\_to\_anchor=(0.0, 1),  
 bbox\_transform=fig1.transFigure  
)  
  
plt.title("Escolaridade dos clientes",fontsize = 12,loc = 'center')  
  
plt.show()

#Contagem dos níveis de escolaridade dos clientes confiáveis  
  
escolaridade\_clientes\_confiaveis = Counter(clientes\_confiaveis['NIVEL\_ESCOLARIDADE'].sort\_values())  
escolaridade\_clientes\_confiaveis

#Percentuais de escolaridade dos clientes confiáveis  
  
labels = []  
sizes = []  
n\_clientes\_confiaveis=sum(escolaridade\_clientes\_confiaveis.values())  
for x, y in escolaridade\_clientes\_confiaveis.items():  
 a = y/n\_clientes\_confiaveis\*100  
 labels.append(str(x) + ': ' + str(round(a,2)) + '%')  
 sizes.append(round(a,2))  
  
  
fig1, ax1 = plt.subplots(figsize=(6, 5))  
fig1.subplots\_adjust(0.3,0,1,1)  
  
  
theme = plt.get\_cmap('bwr')  
ax1.set\_prop\_cycle("color", [theme(1. \* i / len(sizes)) for i in range(len(sizes))])  
  
\_, \_ = ax1.pie(sizes, startangle=90)  
  
ax1.axis('equal')  
  
total = sum(sizes)  
plt.legend(  
 loc='center left',  
 labels=labels,  
 prop={'size': 8},  
 bbox\_to\_anchor=(0.0, 1),  
 bbox\_transform=fig1.transFigure  
)  
  
plt.title("Escolaridade dos clientes confiáveis",fontsize = 12,loc = 'right')  
  
plt.show()

#Contagem do estado civil dos clientes  
  
estado\_civil\_clientes = Counter(new\_df['ESTADO\_CIVIL'].sort\_values())  
estado\_civil\_clientes

#Identificação dos percentuais do estado civil dos clientes  
n\_estado\_civil=sum(estado\_civil\_clientes.values())  
for x, y in estado\_civil\_clientes.items():  
 a = y/n\_estado\_civil\*100  
 print(str(x) + ': ' +'\n'+ str(round(a,2)) + '%'+'\n')

#Contagem dos níveis do estado civil dos clientes confiáveis  
  
estado\_civil\_clientes\_confiaveis = Counter(clientes\_confiaveis['ESTADO\_CIVIL'].sort\_values())  
estado\_civil\_clientes\_confiaveis

#Identificação dos percentuais do estado civil dos clientes confiáveis  
n\_estado\_civil\_clientes\_confiaveis=sum(estado\_civil\_clientes\_confiaveis.values())  
for x, y in estado\_civil\_clientes\_confiaveis.items():  
 a = y/n\_estado\_civil\_clientes\_confiaveis\*100  
 print(str(x) + ': ' +'\n'+ str(round(a,2)) + '%'+'\n')

#Percentuais do estado civil dos clientes  
  
labels = []  
sizes = []  
n\_clientes=sum(estado\_civil\_clientes.values())  
for x, y in estado\_civil\_clientes.items():  
 a = y/n\_clientes\*100  
 labels.append(str(x) + ': ' + str(round(a,2)) + '%')  
 sizes.append(round(a,2))  
  
  
fig1, ax1 = plt.subplots(figsize=(6, 5))  
fig1.subplots\_adjust(0.3,0,1,1)  
  
  
theme = plt.get\_cmap('bwr')  
ax1.set\_prop\_cycle("color", [theme(1. \* i / len(sizes)) for i in range(len(sizes))])  
  
\_, \_ = ax1.pie(sizes, startangle=90)  
  
ax1.axis('equal')  
  
total = sum(sizes)  
plt.legend(  
 loc='center left',  
 labels=labels,  
 prop={'size': 8},  
 bbox\_to\_anchor=(0.0, 1),  
 bbox\_transform=fig1.transFigure  
)  
  
plt.title("Estado civil dos clientes",fontsize = 12,loc = 'center')  
  
plt.show()

#Percentuais do estado civil dos clientes confiáveis  
  
labels = []  
sizes = []  
n\_clientes=sum(estado\_civil\_clientes\_confiaveis.values())  
for x, y in estado\_civil\_clientes\_confiaveis.items():  
 a = y/n\_clientes\*100  
 labels.append(str(x) + ': ' + str(round(a,2)) + '%')  
 sizes.append(round(a,2))  
  
  
fig1, ax1 = plt.subplots(figsize=(6, 5))  
fig1.subplots\_adjust(0.3,0,1,1)  
  
  
theme = plt.get\_cmap('bwr')  
ax1.set\_prop\_cycle("color", [theme(1. \* i / len(sizes)) for i in range(len(sizes))])  
  
\_, \_ = ax1.pie(sizes, startangle=90)  
  
ax1.axis('equal')  
  
total = sum(sizes)  
plt.legend(  
 loc='center left',  
 labels=labels,  
 prop={'size': 8},  
 bbox\_to\_anchor=(0.0, 1),  
 bbox\_transform=fig1.transFigure  
)  
  
plt.title("Estado civil dos clientes confiáveis",fontsize = 12,loc = 'center')  
  
plt.show()

#descrição estatística dos valores da renda dos clientes  
new\_df['RENDA'].describe().astype('int')

#Plotagem do histograma da renda dos clientes  
sns.distplot(new\_df.RENDA, kde=False)  
plt.title('Distribuição da renda dos clientes')

#descrição estatística dos valores da renda dos clientes confiáveis  
clientes\_confiaveis['RENDA'].describe().astype('int')

#Plotagem do histograma da renda dos clientes confiáveis  
sns.distplot(clientes\_confiaveis.RENDA, kde=False, color='red')  
plt.title('Distribuição da renda dos clientes confiáveis')

#Plotagem de histograma com os anos empregado dos clientes  
new\_df.ANOS\_EMPREGADO.hist(bins=20)  
plt.style.use('ggplot')  
plt.xlabel("Anos")  
plt.ylabel("Número de clientes")  
plt.title("Anos de emprego dos clientes")  
plt.show()

#Plotagem de histograma com os anos empregado dos clientes confiáveis  
clientes\_confiaveis.ANOS\_EMPREGADO.hist(bins=20)  
plt.style.use('seaborn-pastel')  
plt.xlabel("Anos")  
plt.ylabel("Número de clientes")  
plt.title("Anos de emprego dos clientes confiáveis")  
plt.show()

#Percentuais de emprego, carro e imóvel dos clientes   
fig, axes = plt.subplots(1,3)  
  
  
g1= new\_df['COD\_TEM\_EMPREGO'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True, colors=["#76B5B3","#EC9B9A"],textprops = {'fontsize':12}, ax=axes[0])  
g1.set\_title("Possui Emprego")  
  
g2= new\_df['COD\_TEM\_CARRO'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True,colors=["#80DE99","#00CECB"],textprops = {'fontsize':12}, ax=axes[1])  
g2.set\_title("Possui Carro")  
  
g3= new\_df['COD\_TEM\_IMOVEL'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True,colors=["#76B5B3","#00CECB"],textprops = {'fontsize':12}, ax=axes[2])  
g3.set\_title("Possui Imóvel")  
  
fig.set\_size\_inches(14,5)

#Percentuais de emprego, carro e imóvel dos clientes confiáveis  
fig, axes = plt.subplots(1,3)  
  
  
g1= clientes\_confiaveis['COD\_TEM\_EMPREGO'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True, colors=["#76B5B3","#EC9B9A"],textprops = {'fontsize':12}, ax=axes[0])  
g1.set\_title("Possui Emprego")  
  
g2= clientes\_confiaveis['COD\_TEM\_CARRO'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True,colors=["#80DE99","#00CECB"],textprops = {'fontsize':12}, ax=axes[1])  
g2.set\_title("Possui Carro")  
  
g3= clientes\_confiaveis['COD\_TEM\_IMOVEL'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True,colors=["#76B5B3","#00CECB"],textprops = {'fontsize':12}, ax=axes[2])  
g3.set\_title("Possui Imóvel")  
  
fig.set\_size\_inches(14,5)

#Contagem da profissão dos clientes  
  
profissao\_clientes = Counter(new\_df['PROFISSAO'].sort\_values())  
profissao\_clientes

#Percentuais da profissão dos clientes  
  
labels = []  
sizes = []  
n\_clientes=sum(profissao\_clientes.values())  
for x, y in profissao\_clientes.items():  
 a = y/n\_clientes\*100  
 labels.append(str(x) + ': ' + str(round(a,2)) + '%')  
 sizes.append(round(a,2))  
  
  
fig1, ax1 = plt.subplots(figsize=(6, 5))  
fig1.subplots\_adjust(0.3,0,1,1)  
  
  
theme = plt.get\_cmap('bwr')  
ax1.set\_prop\_cycle("color", [theme(1. \* i / len(sizes)) for i in range(len(sizes))])  
  
\_, \_ = ax1.pie(sizes, startangle=90)  
  
ax1.axis('equal')  
  
total = sum(sizes)  
plt.legend(  
 loc='center right',  
 labels=labels,  
 prop={'size': 8},  
 bbox\_to\_anchor=(0.0, 1),  
 bbox\_transform=fig1.transFigure  
)  
  
plt.title("Profissão dos clientes",fontsize = 12,loc = 'center')  
  
plt.show()

#Contagem da profissão dos clientes confiáveis  
  
profissao\_clientes\_confiaveis = Counter(clientes\_confiaveis['PROFISSAO'].sort\_values())  
profissao\_clientes\_confiaveis

#Percentuais da profissão dos clientes  
  
labels = []  
sizes = []  
n\_clientes=sum(profissao\_clientes\_confiaveis.values())  
for x, y in profissao\_clientes\_confiaveis.items():  
 a = y/n\_clientes\*100  
 labels.append(str(x) + ': ' + str(round(a,2)) + '%')  
 sizes.append(round(a,2))  
  
  
fig1, ax1 = plt.subplots(figsize=(6, 5))  
fig1.subplots\_adjust(0.3,0,1,1)  
  
  
theme = plt.get\_cmap('bwr')  
ax1.set\_prop\_cycle("color", [theme(1. \* i / len(sizes)) for i in range(len(sizes))])  
  
\_, \_ = ax1.pie(sizes, startangle=90)  
  
ax1.axis('equal')  
  
total = sum(sizes)  
plt.legend(  
 loc='center right',  
 labels=labels,  
 prop={'size': 8},  
 bbox\_to\_anchor=(0.0, 1),  
 bbox\_transform=fig1.transFigure  
)  
  
plt.title("Profissão dos clientes confiáveis",fontsize = 12,loc = 'center')  
  
plt.show()

#Plotagem de histograma com os meses em que a contas dos clientes estão abertas  
new\_df.TEMPO\_DA\_CONTA.hist(bins=20)  
plt.style.use('ggplot')  
plt.xlabel("Meses")  
plt.ylabel("Número de clientes")  
plt.title("Meses da conta dos clientes")  
plt.show()

#Plotagem de histograma com os meses em que a contas dos clientes confiáveis estão abertas  
clientes\_confiaveis.TEMPO\_DA\_CONTA.hist(bins=20)  
plt.style.use('ggplot')  
plt.xlabel("Meses")  
plt.ylabel("Número de clientes")  
plt.title("Meses da conta dos clientes confiáveis")  
plt.show()

new\_df['TEMPO\_DA\_CONTA'].describe()

clientes\_confiaveis['TEMPO\_DA\_CONTA'].describe()

# Criação de Modelos de Machine Learning

#Converter as varáveis categóricas para numéricas

new\_df["COD\_SEXO"] = new\_df['COD\_SEXO'].replace(['H','M'],[0,1])  
new\_df["COD\_TEM\_CARRO"] = new\_df["COD\_TEM\_CARRO"].replace(["S","N"],[1,0])  
new\_df["COD\_TEM\_IMOVEL"] = new\_df["COD\_TEM\_IMOVEL"].replace(["S","N"],[1,0])  
new\_df["COD\_TEM\_EMPREGO"] = new\_df["COD\_TEM\_EMPREGO"].replace(["S","N"],[1,0])

Para que se possa avaliar os algoritmos mencionados, optou-se por realizar alguns ajustes no dataframe new\_df. Como boa prática, os dados categóricos serão transformados em valores inteiros, mais especificamente as colunas NIVEL\_ESCOLARIDADE, ESTADO\_CIVIL, TIPO\_MORADIA e PROFISSAO.

print(new\_df['NIVEL\_ESCOLARIDADE'].unique())

nivel\_escolaridade = {'Secundário':'SECUNDARIO',  
 'Médio':'SECUNDARIO',  
 'Superior':'SUPERIOR',  
 'Superior Incompleto':'SUPERIOR\_INCOMPLETO',  
 'Pós Graduado':'SUPERIOR'}  
  
new\_df["NIVEL\_ESCOLARIDADE"] = new\_df["NIVEL\_ESCOLARIDADE"].map(nivel\_escolaridade)  
new\_df= pd.get\_dummies(new\_df, columns=['NIVEL\_ESCOLARIDADE'])

new\_df.info()

new\_df["NIVEL\_ESCOLARIDADE\_SECUNDARIO"] = new\_df['NIVEL\_ESCOLARIDADE\_SECUNDARIO'].replace({True: 1, False: 0})  
new\_df["NIVEL\_ESCOLARIDADE\_SUPERIOR"] = new\_df['NIVEL\_ESCOLARIDADE\_SUPERIOR'].replace({True: 1, False: 0})  
new\_df["NIVEL\_ESCOLARIDADE\_SUPERIOR\_INCOMPLETO"] = new\_df['NIVEL\_ESCOLARIDADE\_SUPERIOR\_INCOMPLETO'].replace({True: 1, False: 0})

new\_df.info()

print(new\_df['TIPO\_MORADIA'].unique())

tipo\_moradia = {'Casa / Apartamento' : 'CASA\_APARTAMENTO',  
 'Mora com os pais': 'COM\_OS\_PAIS',  
 'Apartamento Comercial' : 'CASA\_APARTAMENTO',  
 'Apartamento alugado': 'CASA\_APARTAMENTO',  
 'Apartamento do Município': 'CASA\_APARTAMENTO',  
 'Apartamento Cooperativo': 'CASA\_APARTAMENTO'}  
  
new\_df["TIPO\_MORADIA"] = new\_df['TIPO\_MORADIA'].map(tipo\_moradia)  
new\_df= pd.get\_dummies(new\_df, columns=['TIPO\_MORADIA'])

new\_df.info()

new\_df["TIPO\_MORADIA\_CASA\_APARTAMENTO"] = new\_df['TIPO\_MORADIA\_CASA\_APARTAMENTO'].replace({True: 1, False: 0})  
new\_df["TIPO\_MORADIA\_COM\_OS\_PAIS"] = new\_df['TIPO\_MORADIA\_COM\_OS\_PAIS'].replace({True: 1, False: 0})

new\_df.info()

print(new\_df['ESTADO\_CIVIL'].unique())

estado\_civil = {'Solteiro':'SOLTEIRO',  
 'Separado':'SOLTEIRO',  
 'Viúvo':'SOLTEIRO',  
 'Casado no civil':'CASADO',  
 'Casado':'CASADO'}  
  
new\_df["ESTADO\_CIVIL"] = new\_df["ESTADO\_CIVIL"].map(estado\_civil)  
new\_df= pd.get\_dummies(new\_df, columns=['ESTADO\_CIVIL'])

new\_df["ESTADO\_CIVIL\_CASADO"] = new\_df['ESTADO\_CIVIL\_CASADO'].replace({True: 1, False: 0})  
new\_df["ESTADO\_CIVIL\_SOLTEIRO"] = new\_df['ESTADO\_CIVIL\_SOLTEIRO'].replace({True: 1, False: 0})

new\_df.info()

print(new\_df['TIPO\_RENDA'].unique())

tipo\_renda = {'Sócio Comercial':'TRABALHANDO',  
 'Servidor Público':'TRABALHANDO',  
 'Trabalhador da Área Privada':'TRABALHANDO',  
 'Pensionista':'PENSIONISTA',  
 'Estudante':'ESTUDANTE'}  
  
new\_df["TIPO\_RENDA"] = new\_df["TIPO\_RENDA"].map(tipo\_renda)  
new\_df= pd.get\_dummies(new\_df, columns=['TIPO\_RENDA'])

new\_df["TIPO\_RENDA\_ESTUDANTE"] = new\_df['TIPO\_RENDA\_ESTUDANTE'].replace({True: 1, False: 0})  
new\_df["TIPO\_RENDA\_PENSIONISTA"] = new\_df['TIPO\_RENDA\_PENSIONISTA'].replace({True: 1, False: 0})  
new\_df["TIPO\_RENDA\_TRABALHANDO"] = new\_df['TIPO\_RENDA\_TRABALHANDO'].replace({True: 1, False: 0})

new\_df.info()

print(new\_df['PROFISSAO'].unique())

profissao = {'Outro': 'Outro', 'Equipe de segurança': 'Equipe\_de\_segurança','Equipe de vendas' : 'Equipe\_de\_vendas',  
 'Contadores' : 'Contadores', 'Operários' :'Operários', 'Gerentes':'Gerentes', 'Motoristas':'Motoristas',  
 'Equipe principal' : 'Equipe\_principal','Equipe técnica altamente qualificada' : 'Equipe\_técnica\_altamente\_qualificada',   
 'Pessoal de limpeza' : 'Pessoal\_de\_limpeza', 'Pessoal de serviço privado' : 'Pessoal\_de\_serviço\_privado',  
 'Pessoal de cozinha' : 'Pessoal\_de\_cozinha', 'Trabalhadores pouco qualificados' : 'Trabalhadores\_pouco\_qualificados',  
 'Equipe médica': 'Equipe\_médica', 'Secretários' : 'Secretários', 'Equipe de garçons barmen' : 'Equipe\_de\_garçons\_barmen',  
 'Equipe de RH': 'Equipe\_de\_RH', 'Corretor imobiliário' : 'Corretor\_imobiliário', 'Equipe de TI' : 'Equipe\_de\_TI' }  
  
new\_df["PROFISSAO"] = new\_df["PROFISSAO"].map(profissao)  
new\_df= pd.get\_dummies(new\_df, columns=['PROFISSAO'])

new\_df.info()

new\_df["PROFISSAO\_Contadores"] = new\_df['PROFISSAO\_Contadores'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Corretor\_imobiliário"] = new\_df['PROFISSAO\_Corretor\_imobiliário'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Equipe\_de\_RH"] = new\_df['PROFISSAO\_Equipe\_de\_RH'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Equipe\_de\_TI"] = new\_df['PROFISSAO\_Equipe\_de\_TI'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Equipe\_de\_garçons\_barmen"] = new\_df['PROFISSAO\_Equipe\_de\_garçons\_barmen'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Equipe\_de\_segurança"] = new\_df['PROFISSAO\_Equipe\_de\_segurança'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Equipe\_de\_vendas"] = new\_df['PROFISSAO\_Equipe\_de\_vendas'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Equipe\_médica"] = new\_df['PROFISSAO\_Equipe\_médica'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Equipe\_principal"] = new\_df['PROFISSAO\_Equipe\_principal'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Equipe\_técnica\_altamente\_qualificada"] = new\_df['PROFISSAO\_Equipe\_técnica\_altamente\_qualificada'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Gerentes"] = new\_df['PROFISSAO\_Gerentes'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Motoristas"] = new\_df['PROFISSAO\_Motoristas'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Operários"] = new\_df['PROFISSAO\_Operários'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Outro"] = new\_df['PROFISSAO\_Outro'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Pessoal\_de\_cozinha"] = new\_df['PROFISSAO\_Pessoal\_de\_cozinha'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Pessoal\_de\_limpeza"] = new\_df['PROFISSAO\_Pessoal\_de\_limpeza'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Pessoal\_de\_serviço\_privado"] = new\_df['PROFISSAO\_Pessoal\_de\_serviço\_privado'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Secretários"] = new\_df['PROFISSAO\_Secretários'].replace({True: 1, False: 0})  
new\_df["PROFISSAO\_Trabalhadores\_pouco\_qualificados"] = new\_df['PROFISSAO\_Trabalhadores\_pouco\_qualificados'].replace({True: 1, False: 0})

new\_df.info()

new\_df['RESULTADO'].value\_counts().plot.pie(explode=[0.1,0.1],autopct='%1.1f%%',shadow=True, colors=['#FF5E5B', '#C0E680'],textprops = {'fontsize':7}).set\_title("Distribuição dos clientes confiáveis")  
  
plt.show()

#Contagem dos clientes confiáveis e não confiáveis  
new\_df['RESULTADO'].value\_counts()

#Importação da função resample da biblioteca sklearn  
from sklearn.utils import resample

#Criação de um dataframe apenas com os clientes não confiáveis  
new\_df\_nao\_confiaveis = new\_df[new\_df.RESULTADO==1]  
new\_df\_nao\_confiaveis.info()

#Criação de um dataframe apenas com os clientes confiáveis  
new\_df\_confiaveis = new\_df[new\_df.RESULTADO==0]  
new\_df\_confiaveis.info()

#Ajuste no número de entradas do dataframe de clientes não confiáveis  
new\_df\_nao\_confiaveis\_upsampled = resample(new\_df\_nao\_confiaveis,   
 replace=True, n\_samples=28819,   
 random\_state=123)

new\_df\_nao\_confiaveis\_upsampled.info()

#Concatenação dos dataframes dos clientes confiáveis e não confiáveis  
new\_df\_final\_upsampled = pd.concat([new\_df\_confiaveis,   
 new\_df\_nao\_confiaveis\_upsampled])

new\_df\_final\_upsampled.info()

#Contagem dos clientes confiáveis e não confiáveis  
new\_df\_final\_upsampled['RESULTADO'].value\_counts()

#Divisão para as bases de treinamento  
X\_train = new\_df\_final\_upsampled.drop(['RESULTADO'], axis = 1)  
y\_train = new\_df\_final\_upsampled.RESULTADO

#Importação da função train\_test\_split  
from sklearn.model\_selection import train\_test\_split

#Criação das bases de teste e treinamento  
xtreinamento, xteste, ytreinamento, yteste = train\_test\_split(X\_train, y\_train, random\_state = 0)

#Informação do dataframe de treinamento  
xtreinamento.info()

#Informação do dataframe de teste  
xteste.info()

#Contagem dos resultados para treinamento  
ytreinamento.count()

#Contagem dos resultados para teste  
yteste.count()

#Dataframe com os resultados  
df\_resultados = pd.DataFrame(columns=('Algoritmo','Acurácia de previsão', 'Precisão', 'Revocação','F1-Score'))

#Importação das funções para as medidas de avaliação dos algoritmos  
from sklearn.metrics import accuracy\_score, classification\_report

#Criação do modelo utilizando a Árvore de decisão  
from sklearn.tree import DecisionTreeClassifier  
from sklearn import tree  
clientes\_tree = DecisionTreeClassifier(random\_state=0)  
clientes\_tree = clientes\_tree.fit(xtreinamento, ytreinamento)  
print("Acurácia: ", clientes\_tree.score(xtreinamento, ytreinamento))  
Train\_predict = clientes\_tree.predict(xteste)  
print("Acurácia de previsão: ", accuracy\_score(yteste, Train\_predict))  
print(classification\_report(yteste, Train\_predict))

new\_row = {'Algoritmo': 'Árvore de decisão', 'Acurácia de previsão': 0.93, 'Precisão': 0.95,'Revocação':0.94,'F1-Score':0.94}   
df\_resultados.loc[len(df\_resultados)] = new\_row  
print(df\_resultados)

#Criação do modelo utilizando a Regressão Logística  
from sklearn.linear\_model import LogisticRegression  
lr = LogisticRegression()  
lr = lr.fit(xtreinamento, ytreinamento)  
print("Acurácia: ", lr.score(xtreinamento, ytreinamento))  
tp\_lr = lr.predict(xteste)  
print("Acurácia de previsão: ", accuracy\_score(yteste, tp\_lr))  
print(classification\_report(yteste, tp\_lr))

new\_row = {'Algoritmo': 'Regressão Logística', 'Acurácia de previsão': 0.51, 'Precisão': 0.52,'Revocação':0.52,'F1-Score':0.51}   
df\_resultados.loc[len(df\_resultados)] = new\_row  
print(df\_resultados)

#Criação do modelo utilizando Naïve Bayes  
from sklearn.naive\_bayes import GaussianNB  
nb = GaussianNB()  
nb = nb.fit(xtreinamento, ytreinamento)  
print("Acurácia: ", nb.score(xtreinamento, ytreinamento))  
tp\_nb = nb.predict(xteste)  
print("Acurácia de previsão: ", accuracy\_score(yteste, tp\_nb))  
print(classification\_report(yteste, tp\_nb))

new\_row = {'Algoritmo': 'Naïve Bayes', 'Acurácia de previsão': 0.55, 'Precisão': 0.56,'Revocação':0.56,'F1-Score':0.56}   
df\_resultados.loc[len(df\_resultados)] = new\_row  
print(df\_resultados)

#Criação do modelo utilizando Gradiente Descendente  
from sklearn.linear\_model import SGDClassifier  
sgd = SGDClassifier()  
sgd = sgd.fit(xtreinamento, ytreinamento)  
print("Acurácia: ", sgd.score(xtreinamento, ytreinamento))  
tp\_sgd = sgd.predict(xteste)  
print("Acurácia de previsão: ", accuracy\_score(yteste, tp\_sgd))  
print(classification\_report(yteste, tp\_sgd))

new\_row = {'Algoritmo': 'Gradiente Descendente', 'Acurácia de previsão': 0.5, 'Precisão': 0.25,'Revocação':0.5,'F1-Score':0.33}   
df\_resultados.loc[len(df\_resultados)] = new\_row  
print(df\_resultados)

#Criação do modelo utilizando KNN (K - Nearest Neighbors)  
from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier()  
knn = knn.fit(xtreinamento, ytreinamento)  
print("Acurácia: ", knn.score(xtreinamento, ytreinamento))  
tp\_knn = knn.predict(xteste)  
print("Acurácia de previsão: ", accuracy\_score(yteste, tp\_knn))  
print(classification\_report(yteste, tp\_knn))

new\_row = {'Algoritmo': 'KNN', 'Acurácia de previsão': 0.86, 'Precisão': 0.88,'Revocação':0.86,'F1-Score':0.86}   
df\_resultados.loc[len(df\_resultados)] = new\_row  
print(df\_resultados)

#Criação do modelo utilizando Randon Forest   
from sklearn.ensemble import RandomForestClassifier  
rfm = RandomForestClassifier()  
rfm = rfm.fit(xtreinamento, ytreinamento)  
print("Acurácia: ", rfm.score(xtreinamento, ytreinamento))  
tp\_rfm = rfm.predict(xteste)  
print("Acurácia de previsão: ", accuracy\_score(yteste, tp\_rfm))  
print(classification\_report(yteste, tp\_rfm))

new\_row = {'Algoritmo': 'Randon Forest', 'Acurácia de previsão': 0.96, 'Precisão': 0.96,'Revocação':0.96,'F1-Score':0.96}   
df\_resultados.loc[len(df\_resultados)] = new\_row  
print(df\_resultados)

df\_resultados.to\_excel('output.xlsx', index=False)