Relatório do Projeto:

Modelos de Previsão de Decisão de Compra de Propriedades

Introdução:

O projeto de ciência de dados que desenvolvi tem como objetivo criar algoritmos de aprendizado de máquina supervisionados que previssem se um cliente comprou ou não uma determinada residência. Esses modelos de “Machine Learning” recebem variáveis contendo informações sobre as propriedades e os indivíduos de uma base de dados, retirada da plataforma Kaggle, para realizar tais previsões. Para acessar a página de onde essa foi obtida, clique nesse link.

Utilizando da linguagem Python e as suas principais bibliotecas para análise e manipulação de dados e modelos de aprendizado de máquina (Pandas, Matplotlib, Seaborn, Numpy, Plotly e Scikit-learn), esse projeto é divido em 4 etapas principais: a análise exploratória dos dados, onde as informações são apresentadas e compreendidas e, suas incoerências, corrigidas, o pré-processamento, que prepara os dados para que estejam adequados para os algoritmos de “Machine Learning”, a modelagem, em que os modelos são criados, treinados e, por fim, testados para alcançarem o objetivo proposto e, finalmente, a etapa de avaliação, que consiste em utilizar métricas e medidas para validar a eficiência e capacidade do modelo de realizar as previsões corretamente.

Análise Exploratória dos Dados:

A primeira parte da Análise Exploratória dos Dados consiste em realizar o tratamento inicial das informações presentes na base. Para fazer isso, verifiquei se existiam valores nulos e se existiam casos preenchidos incorretamente ou de maneira incoerente, de acordo com a variável que representavam. Após essa checagem, percebi que nenhuma inconsistência estava presente, então o único tratamento inicial realizado foi a exclusão de uma das variáveis, a ‘property_id’, ou seja, a coluna de ID único de cada propriedade registrada na base de dados.

A remoção dessa variável se deve ao fato de que, para o objetivo do projeto de prever quais os clientes que compraram ou não um imóvel, essa informação não apresenta nenhuma característica útil para o algoritmo e possivelmente atrapalharia seu processo, servindo apenas como uma forma de identificação de cada residência. Com isso, a base de dados é reduzida para 24 colunas distintas, sendo elas:

- País onde se encontra a residência (13 países diferentes);
- Cidade onde se encontra a residência (40 cidades diferentes);
- Tipo de residência (6 tipos diferentes);

- Se está mobiliada, semi-mobiliada ou sem mobília;
- Tamanho em pés quadrados da residência;
- Preço da residência;
- Ano de construção da residência;
- Quantidade de donos anteriores da residência;
- Quantidade de quartos da residência;
- Quantidade de banheiros da residência;
- Se tem garagem (0: Não, 1: Sim);
- Se tem jardim (0: Não, 1: Sim);
- Quantidade de casos criminosos reportados perto da residência;
- Se há conflito jurídico ocorrendo na residência (0: Não, 1: Sim);
- Salário do cliente/comprador;
- Quantidade de empréstimo planejado a ser feito;
- Prazo para pagamento do empréstimo em anos;
- Custos mensais do comprador;
- Pagamento adiantado feito;
- Porcentagem da renda mensal destinada a pagar parcelas de empréstimos, financiamentos e outros débitos;
- Pontuação de satisfação do cliente/comprador (1-10);
- Avaliação da vizinhança (Segurança, conveniência, localizada perto de escolas) (1-10);
- Avaliação da conectividade (Acesso a transporte público, internet de boa qualidade etc.) (1-10);
- **Variável Alvo:** Se o cliente comprou a residência ou não (0: Não, 1: Sim).

Com essas variáveis estabelecidas, segui para a próxima parte da etapa de exploração dos dados: a sua análise mais detalhada através de gráficos. Utilizando essa forma de visualizar as informações, consegui compreender melhor o comportamento e distribuição das variáveis, a forma como se relacionam entre si e evidenciam características e “insights” sobre os casos que estão sendo trabalhados.

Buscando entender quais fatores tendem a ser mais influentes de forma direta na decisão final do cliente, utilizei de gráficos BoxPlot e histogramas que mostravam a relação entre a variável alvo com outra informação da base. Isso permitiu que eu percebesse que 3 variáveis tinham um comportamento diferente entre os clientes compradores e os não compradores:

- **Score de Satisfação**: Notei que entre os clientes compradores, a nota de satisfação com o atendimento estava sempre entre os mais altos, indo de 7 a 10, enquanto para os que não adquiriram o imóvel variavam desde notas mais baixas até mais altas, mas se concentrando principalmente entre 2 e 6.

- **Quantidade de Crimes Reportados**: Apesar de muitos imóveis que não foram adquiridos que possuíam poucos ou nenhum caso criminal reportado, todas as residências que tiveram mais de 2 crimes reportados perto de sua localização não foram compradas, enquanto as que foram possuíam 0 ou 1 em sua maioria.
- **Propriedade em Casos Jurídicos**: Todas as residências que se encontravam em alguma disputa legal, cerca de $1/4$ de todos os casos, não foram adquiridas, representando $1/3$, enquanto apenas propriedades livres dessa situação foram adquiridas.

Com esses conhecimentos, ficou evidente que um bom atendimento ao cliente e residências em locais seguros, livres de problemas jurídicos são fatores importantes para que o cliente realize a compra e adquira o imóvel.

Além disso, para entender melhor os perfis dos clientes e das residências, utilizei de gráficos de dispersão que mostravam a relação entre as variáveis de preço, tamanho, valor do empréstimo e valor do pagamento adiantado. Entendendo melhor o comportamento dessas características, notei que não havia um padrão simples e linear entre os casos registados, mas sim uma correlação mais complexa, que era constituída por diversas informações. Com isso, é evidente que existem diversos grupos de compradores e propriedades, que compartilham semelhanças entre os integrantes, como localização ou tipo de residência, e explicam o porquê de a distribuição divergir em outros aspectos.

Tratamento de Outliers:

Para garantir que os modelos de previsão proveram os melhores resultados, utilizei de medidas estatísticas para encontrar possíveis “outliers” nas variáveis, valores fora do padrão que podem prejudicar o processamento de informações e enviesarem o algoritmo. Os métodos utilizados para isso foram o cálculo de média, mediana, primeiro quartil, terceiro quartil, desvio padrão, valor mínimo e máximo de cada uma, o que permitiu que observasse 4 variáveis tinham um comportamento inesperado: o preço do imóvel, o valor do empréstimo, o pagamento adiantado feito e a proporção de EMI para renda.

Criei gráficos BoxPlot para cada uma dessas colunas para entender mais a fundo a distribuição de valores presente e entender melhor a presença desses “outliers”. No caso das variáveis de preço, empréstimo e pagamento adiantado, notei que todos os valores extremos estavam acima do limite esperado, mas com uma variação reduzida dos valores que, em sua maioria, estavam próximos do limite superior. Com isso em mente, considerei que esses casos eram uma minoria dos casos e não estavam incorretos, já que as características de uma propriedade ou perfis de compradores são sempre distintos e únicos, sem seguir um padrão e variam por inúmeros fatores; dessa maneira, optei por não os tratar e mantê-los da maneira que estão.

A variável de proporção de Parcela Mensal Estimada (EMI) para renda mensal mostrou um comportamento diferente. Apesar de todos os outliers também estarem acima do padrão esperado, sua quantidade foi maior e a variação foi mais dispersa, o que me levou a concluir que um tratamento seria necessário, mas retirá-los ou alterá-los seria incoerente pois se trata de uma informação com alta imprevisibilidade em seu comportamento. Optei, então, por aplicar uma função logarítmica, que não alteraria ou retiraria os “outliers”, mas mudaria a escala de todos os valores e reduziria o impacto desses “outliers” no processamento dos modelos.

Pré-Processamento:

O pré-processamento é a etapa do projeto em que os dados são estruturados e reorganizados para que estejam de acordo com os parâmetros e padrões exigidos pelos modelos de aprendizagem. É aqui que diferentes ferramentas são utilizadas para garantir que o processamento do modelo ocorra sem grandes obstáculos e seu desempenho seja o mais favorável e realista possível. Com isso, apliquei os seguintes métodos para alcançar determinado objetivo:

Codificação de Variáveis Categóricas: Para que todas as informações sejam devidamente compreendidas, o algoritmo deve receber apenas valores numéricos em suas etapas de treinamento e teste. Dessa forma, apliquei 2 distintos métodos de codificação para transformar as 4 variáveis categóricas em numéricas. O primeiro foi o “Label Encoder”, que atribui um número para cada classificação de uma variável, começando do 0; essa função foi usada na variável de status de mobília de uma residência, pois se trata de uma informação categórica ordinal, ou seja, existe uma hierarquia por trás das categorias. O segundo método foi o “One-Hot Encoding”, que cria novas colunas binárias na base para cada valor presente na variável categórica, usada nas restantes informações desse tipo, já que são nominais, sem ordem hierárquica, resultando num aumento considerável no total de colunas na base.

Correlação das Variáveis: A análise da correlação das variáveis é útil para entender quais informações na base de dados possuem alguma tendência ou influência entre si e, principalmente, com a variável alvo, quantificada de –1 a 1, em que valores mais próximos de 0 são relações mais fracas. Ao perceber que muitas variáveis apresentaram apenas correlações neutras com outras informações, optei por manter somente os dados que tiveram alguma influência de fato significativa, retirando variáveis de pouco valor e reduzindo a quantidade de informações desnecessárias a serem fornecidas aos modelos, tornando-os mais rápidos e práticos.

Com isso, os modelos receberão os seguintes dados: **tamanho da propriedade**, **preço**, **quantidade de crimes reportados próximos**, se há **disputas legais ocorrendo no imóvel**, **salário do cliente**, **valor do empréstimo**, **pagamento adiantado**, **proporção de EMI para renda**, **nota de satisfação**, colunas binárias

para cada **país**, **cidade** e **tipo de residência** presente na base e a variável alvo de **decisão**.

Separação em Bases de Treino e Teste: Os modelos devem receber dados de treinamento para aprenderem sobre as informações, que serão fornecidas para realizar as previsões, e dados de teste, que servem para verificar a eficiência da aprendizagem. Por isso, separei as variáveis independentes da variável alvo em X e Y, respectivamente e, posteriormente, as separei novamente em 4 bases, duas de treino e duas de teste, em que cada dupla tem uma apenas com o que quer ser descoberto e outra com as informações que ajudaram a descobri-la.

Com essa separação feita, utilizei o “Standard Scaler” para **padronizar** as variáveis independentes. Essa ferramenta transforma todas as suas distintas escalas para uma igual, em que a média é 0 e o desvio padrão é 1, e é frequentemente utilizada em projetos de dados por sua eficácia. Realizei a **padronização**, pois a variação de valores das colunas da base era muito discrepante, variando entre unidades e dezenas até milhares e milhões, o que poderia resultar em comparações incoerentes por parte do algoritmo e ineficiência dos resultados, caso mantidas dessa maneira.

Balanceamento da Variável Alvo: Por fim, na última etapa do pré-processamento de dados, verifiquei como estava o balanceamento das opções possíveis da variável alvo de treino. Na etapa de treinamento, os modelos recebem uma parte maior dos dados da base (75%) para poderem entender ao máximo os padrões presentes e realizar previsões corretas, o que significa que não só é necessário que o algoritmo receba uma quantidade geral suficiente de informações, como também deve ser capaz de identificar qualquer classificação possível da variável alvo de maneira equivalente.

No caso desse projeto, a base de treino mostrou ter 77% de casos de clientes não compradores e 23% de compradores. Se mantido dessa maneira, os modelos aprenderiam muito bem como identificar os casos em que a venda não ocorreu, mas seriam incapazes de entender os que a compra foi feita, justamente por ter uma classe predominante enquanto a outra não possuí as informações necessárias para ser compreendida. Por isso, para que quando o algoritmo esteja igualmente preparado para ambos os casos ao se deparar com dados inéditos, apliquei a ferramenta “SMOTE”, que cria dados sintéticos da classe minoritária até que sua quantidade seja igual a majoritária. Assim, os modelos aprendem os padrões dos 2 casos e tem chances de enviesamento reduzidas.

Modelagem:

A etapa da modelagem é o momento do projeto em que são instanciados os algoritmos para previsão da variável alvo, em que primeiro os dados de treino são fornecidos para aprendizagem dos padrões e, posteriormente, a base de teste é utilizada para garantir a eficácia desse treinamento. Para esse projeto, utilizei 2

modelos distintos para alcançar o objetivo e poder comparar os seus desempenhos entre si, sendo eles:

- **Regressão Logística**: Adequado para o objetivo desse projeto por ser especialmente eficaz em previsões binárias, ou seja, para quando existem duas possíveis categorias presentes na variável alvo, que nesse caso seriam se a residência foi ou não comprada. Esse modelo de aprendizado de máquina supervisionado de classificação também foi escolhido por sua simplicidade e fácil interpretação de resultados, com um processamento mais rápido e dinâmico que permite uma entrega muito prática de constantes novas previsões.

  Como funciona: Através do cálculo da Função Sigmoide, que calcula a probabilidade de a variável alvo pertencer a certa categoria, nesse caso, '0' ou '1', em que o resultado será um valor entre 0 e 1, em que os resultados maiores que o limite são categorizados como '1' e os menores, como '0'.

- **XGBoost**: Método de "Machine Learning" conhecido como "ensemble", onde um grupo de modelos são utilizados para retornar o resultado mais preciso possível. Por conta de ser composto dessa maneira, o XGBoost é mais custoso computacionalmente, mas é mais otimizado e consegue captar padrões mais discretos que podem passar despercebidos por modelos mais simples, gerando previsões melhores em grande parte dos casos. Além disso, também é mais robusto, o que significa que ter maior capacidade em manter um bom desempenho mesmo quando deparado com dados de menor qualidade, cenário frequente em aplicações de mundo real, e impacto reduzido com variações extremas de valores, os "outliers", que estão presentes em algumas variáveis da base de dados. Por fim, é importante ressaltar que essa otimização e robustez contribuí em combater o "overfitting", obstáculo comum em muitos modelos de aprendizado, que consiste na compreensão excessiva dos dados de treinamento a ponto de classificá-los perfeitamente, mas em detrimento de ser incapaz de prever dados novos e inéditos, tornando-o uma escolha ainda mais indicada.

  Como funciona: O processo por trás das previsões feitas pelo XGBoost parte do conceito da "Árvore de Decisão". Essa última consiste em fazer questionamentos usando as variáveis de características e, a partir de cada resposta possível se ramifica em um caminho que pode levar a uma nova pergunta ou a uma previsão de classificação. A primeira pergunta, conhecida como "raiz", é escolhida como o melhor ponto de partida das características da base pelo algoritmo para dividir as informações e gera os primeiros "ramos", as possíveis respostas de cada pergunta, que resultam em "nós", novas pergunta fundada nas variáveis independentes que causam novos ramos, ou

em "folhas", que são o resultado de cada possível caminho e contém as previsões, o que causa uma ramificação similar a uma árvore. Contudo, como já explicado, o XGBoost é um método de "ensemble", então a forma como funciona é mais complexa e segue um método chamado de "gradient boosting" (boosting de gradiente): uma primeira árvore de decisão realiza suas previsões, que são apresentadas a uma nova árvore que irá buscar entender e corrigir os erros que ela teve, gerando novas previsões mais confiáveis, que passaram por esse mesmo procedimento até que seja alcançado um modelo com a melhor capacidade de previsão possível.

Avaliação dos Resultados:

A última etapa do projeto é a avaliação do desempenho de cada modelo. Para isso, utilizei das seguintes métricas confiáveis:

- **Precision**: A precisão indica a porcentagem de previsões do modelo de uma categoria que são, de fato, daquela classe. Em outras palavras, é a taxa de acerto das previsões feitas em relação a todas as previsões que o modelo realizou para aquela categoria;
- **Recall**: Indica a porcentagem das previsões corretas feitas de uma categoria em relação a todos os casos verdadeiramente pertencentes aquela categoria. Ou seja, é a quantidade de previsões que foram corretamente identificadas em comparação com todos os casos pertencentes a determinada classe;
- **F1-Score**: É a média harmônica entre a precisão e o "recall", ou seja, é a junção de ambas as métricas em apenas um valor, permitindo uma visão mais geral da capacidade do modelo em relação a cada classe;
- **Accuracy**: É a taxa de acerto das previsões do modelo em relação a todos os casos possíveis, isso é, a porcentagem total de previsões corretas, a acurácia;
- **Cross Validation**: Conjunto de valores que indicam o desempenho de modelos iguais aos que estão sendo avaliados, porém ao receber conjuntos de dados de treino e teste da mesma base, mas diferentes entre si, ou seja, recebem as mesmas informações, porém alterando sua ordem e posição. Aqui, foram feitos o padrão de 5 "folds" para a avaliação cruzada e uma média dos valores foi calculada e retornada. Essa métrica garante que o modelo oficial não tenha sofrido de problemas, como enviesamento, por causa da forma como os dados foram estruturados e organizados ao serem passados ao modelo;
- **Curva ROC**: Gráfico que visualiza a performance do modelo em todos os possíveis limiares de decisão, onde o eixo Y é a Taxa de Verdadeiros Positivos e o eixo X é a Taxa de Falsos Positivos, ambos de 0 a 1. Quanto mais rapidamente a curva sobe no eixo Y, se aproximando de 1, e mais

perto de 0 no eixo X ela fica, melhor foi a performance do modelo. Esse gráfico permite garantir que as previsões sendo feitas são mais eficazes que um chute ao certificar que a curva não seja uma diagonal perfeita ou, ainda pior, menos eficaz que um chute, sendo mais próxima ao 1 do eixo X e 0 do eixo Y;

- **AUC Score**: É a área sob a curva ROC, ou seja, o valor único que resume o desempenho da curva ROC e indica e explica o seu comportamento. Indica o quão bem o modelo consegue distinguir entre as classes.

Avaliação do modelo de Regressão Logística: Ao receber dados inéditos e desbalanceados, o modelo de regressão logística mostrou maior capacidade de identificar os casos de clientes não compradores, com um 'F1-Score' de 0.96, mas apenas com uma leve dificuldade maior para os compradores (que são uma quantidade significativamente menor dos casos da base de teste), em que o 'F1-Score' foi de 0.87. Contudo, notei que essa métrica foi diferente para os dados de treino, que foram de 0.95 para ambas as situações, me levando a concluir que um leve "overfitting" ocorreu para os clientes compradores, já que houve uma queda na quantidade de acertos para esses casos na etapa de teste, indicando uma compreendimento mais exagerado das informações usadas para a aprendizagem do modelo. Ainda assim, os resultados finais foram excelentes, com um acurácia geral alta, um "cross validation" de 0.95 e o gráfico da curva ROC evidenciando um AUC de 0.94, indicando que o algoritmo consegue distinguir muito bem os casos.

Avaliação do modelo de XGBoost: Os resultados desse modelo foram iguais tanto para os dados de treino quanto para dados de teste: todas as previsões foram corretas. Todas as principais métricas tiveram os melhores resultados, o que indica que o modelo não teve dificuldades em distinguir entre as possíveis categorias e classificá-las corretamente. Normalmente, casos assim são extremamente difíceis de ocorrer e muitas vezes indicam problemas no modelo, como "overfitting". Contudo, vendo que os resultados de um modelo mais simples, a Regressão Logística, foram excelentes, o XGBoost, que é mais otimizado e robusto, feito para contornar obstáculos comuns em "Machine Learning" para resultar nas melhores previsões possíveis, ter esse desempenho não indica uma anomalia, mas uma boa configuração do algoritmo, base de dados completa e com informações relevantes, além de um eficaz pré-processamento e limpeza dos dados.

Conclusão:

Os resultados finais de ambos os modelos foram muito gratificantes. O algoritmo de Regressão Logística apresentou uma maior diferença entre a fase de treinamento e a de teste, o que indica maiores dificuldades, mas considerando a simplicidade desse tipo aprendizado supervisionado, sua capacidade de previsões foi excelente. O XGBoost, por sua vez, conseguiu classificar todos os casos corretamente, independentemente da etapa da modelagem que se encontrou, o que

mostra sua eficácia para o objetivo do projeto e como sua otimização natural e robustez foram necessárias para esse resultado perfeito, ainda que mais demandante computacionalmente.

Buscando compreender como o processamento de cada modelo se sucedeu, utilizei um método que quantificasse a importância de cada variável para a efetuação das previsões de cada modelo: enquanto o XGBoost deu mais importância para 4 variáveis (se há disputas legais na propriedade, o score de satisfação do cliente, a quantidade de casos criminais reportados próximos ao imóvel e a proporção de EMI para renda), o modelo de Regressão Logística também deu importância para mais 4 outras colunas além dessas (salário do cliente, tamanho da residência, valor do empréstimo e preço). Isso indica que, por ser mais simples, esse último sofreu de um excesso de informações e teve maior dificuldade de distinguir quais tinham maior relevância para o objetivo do projeto.

Por fim, a conclusão final é que para identificar clientes compradores e não compradores com maior facilidade, o modelo de XGBoost é a escolha ideal, já que foi capaz de distinguir melhor os padrões e entender quais informações possuíam maior significância. Ainda que ele exija maior poder computacional e possa ter um processo mais lento ao receber constantes dados novos, sua resistência a valores extremos em algumas das variáveis da base de dados permitiriam o seu alto desempenho e acurácia, compensando por esses fatores. Além disso, não foram necessários realizar configurações mais avançadas, como o ajuste de hiper parâmetros, para obter esse resultado, o que evidencia ainda mais a capacidade do modelo de lidar com novos dados com praticidade.