

Previsão de Sucesso de Venda de Startups a Partir de Modelos de Regressão

Clara Nicolini

June 2025

1 Considerações Iniciais

A ideia principal do projeto é encontrar uma forma de prever se uma empresa da área da tecnologia está propensa a ser vendida e, se estiver, qual a probabilidade dessa venda ser bem-sucedida. Esse tema foi escolhido por ser um fator considerável no setor de investimentos, tanto para investidores que querem adicionar percentuais de outras empresas na sua cartela de investimentos, quanto para aqueles que possuem uma startup e gostariam de saber, com base em dados, quais as chances da venda da sua empresa ser feita no momento certo — evitando vender antes da hora, não atingindo um lucro satisfatório, ou vendendo depois da hora por um valor menor que o potencial.

Para o desenvolvimento dessa análise, entendeu-se que seria necessário o uso de informações mais técnicas da área de investimento, de forma a conseguir definir quais variáveis seriam importantes no modelo e deveriam estar presentes no banco de dados.

Dentre as variáveis importantes que foram identificadas como úteis para o desenvolvimento do projeto, estão:

- **Investment value:** quanto as empresas já investiram na startup;
- **Rounds:** quantidades de rodadas de investimento que a empresa já participou;
- **Valuation:** estimativa de valor de venda da empresa inteira ou partes baseada em potencial de lucro;
- **Number of employees:** quantos funcionários a empresa tem;
- **Years of activity:** quantos anos a empresa está funcionando ou em que data ela foi criada.

Primeiro foram pesquisados bancos de dados que pudessem ser úteis para a análise. O primeiro identificado foi o *Startup Growth & Funding Trends*, que possuía os seguintes tipos de dados:

1. Startup Name – The name of the startup (e.g., Startup_1, Startup_2);
2. Industry – The sector in which the startup operates (e.g., AI, FinTech, HealthTech);
3. Funding Rounds – The total number of funding rounds raised by the startup (1–5);
4. Funding Amount (M USD) – The total amount of funding received in millions of USD;
5. Valuation (M USD) – The startup's post-money valuation in millions of USD;
6. Revenue (M USD) – The estimated annual revenue in millions of USD;
7. Employees – The number of employees working in the startup (ranging from 5 to 5000);
8. Market Share (%) – The percentage of the market the startup has captured;
9. Profitable – A binary indicator (1 = Profitable, 0 = Not Profitable).

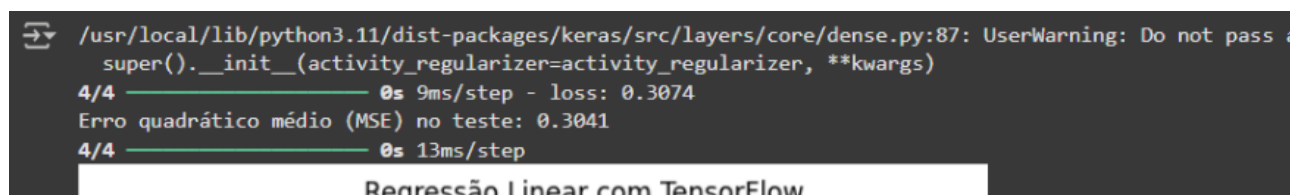
Decidiu-se pelo uso desse banco, cujo objetivo principal de sua criação é machine learning.

A partir desse banco de dados, a ideia inicial foi construir três regressões: uma linear, uma logística e uma multilinear. Cada uma traria um tipo de resultado diferente ao analisar os dados — se a empresa deveria ou não ser vendida, e se sim, qual a probabilidade dessa venda ser feita no timing correto.

Para isso, seria necessário desenvolver:

- **Uma regressão linear**, que adquirisse as variáveis Funding, Employees, Market Share, Time of Operation e retornasse previsões de Revenue e Valuation;
- **Uma regressão logística**, que transformaria categorias adicionais próprias do banco de dados — sendo esses dados qualitativos — em valores binários que contribuiriam para o resultado final da análise. Esses dados qualitativos eram *region* e *industry*, que foram convertidos em colunas binárias;
- **Uma regressão multilinear**, que utilizaria todas as variáveis numéricas para validar Revenue e Valuation, fazendo uma comparação ao resultado da regressão linear.

Em geral, com o primeiro banco de dados utilizado, foi possível desenvolver a regressão linear, que apresentou boas métricas: 0,71 de *accuracy* e 0,3074 de *loss*. Também foram testados gráficos para verificar se o modelo estava de fato aprendendo e considerando todos os valores.



```

/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass i
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
4/4 — 0s 9ms/step - loss: 0.3074
Erro quadrático médio (MSE) no teste: 0.3041
4/4 — 0s 13ms/step
Regressão Linear com TensorFlow

```

Figure 1: Resultados da Primeira Regressão

Após o desenvolvimento da linear, iniciou-se a construção da regressão binária, mas surgiram problemas. Em relação à regressão logística, ainda que tivesse sido feito um tratamento de *one-hot encoding* para os dados de *Industry* e *Region*, qualitativos, o modelo estava persistindo em não ler os valores 1 — apenas reconhecia os 0. Isso indicava um desbalanceamento significativo entre as classes. Foram feitas tentativas de balanceamento da análise, ajustes no *learning rate* para diversos valores, alterações na quantidade de rodadas de teste, e finalmente uma transformação para o modelo Perceptron (rede neural). Nenhuma das tentativas obteve resultados consideráveis.

2 Substituição do Banco de Dados

Depois de quase dois dias de testes, descobriu-se a partir da falta de opções quanto ao tratamento dos dados que o problema poderia ser proveniente do tamanho reduzido do banco escolhido. E, de fato, a hipótese foi confirmada futuramente. Como o banco possuía somente valores referenciais de 500 empresas, houve sucesso no desenvolvimento da regressão linear, mas a regressão binária não possuía valores suficientes para treinar o modelo de forma satisfatória. Como os dados foram divididos entre 80% para treinamento e 20% para teste, o modelo estava com pouca referência do que seriam bons valores de observação. Ainda, com o *one-hot encoding*, muitos novos dados foram introduzidos, cuja proporção de 0 para 1 era, também, excessivamente desbalanceada.

Diante disso, optou-se pela troca do banco de dados. Foi encontrado um conjunto semelhante, *Startup Growth & Investment Data*, com variáveis similares e uma quantidade consideravelmente maior de dados, com informações sobre 5.000 empresas. As variáveis do banco de dados final são:

- **Startup Name:** Name of the startup;
- **Industry:** The industry sector (e.g., AI, Fintech, HealthTech, etc.);

- **Funding Rounds:** Number of funding rounds received by the startup;
- **Investment Amount (USD):** Total investment received in USD;
- **Valuation (USD):** Estimated company valuation in USD;
- **Number of Investors:** Total number of investors backing the startup;
- **Country:** Country where the startup is based;
- **Year Founded:** Year when the startup was founded;
- **Growth Rate (%):** Annual growth rate percentage.

Com informações semelhantes o suficiente para descartar a necessidade de criação de uma ideia nova para o projeto, este banco de dados foi o escolhido para o desenvolvimento do projeto final.

Devido ao novo tamanho de amostra do banco e à escolha por aprofundar o estudo de desenvolvimento de regressões e rede neural, a estrutura geral do projeto também foi alterada. Para esse banco de dados, foi descartada a regressão linear e adicionadas em seu lugar duas redes neurais — uma associada à regressão multilinear e outra associada à regressão logística.

Ainda que os modelos tenham sido divididos em duas redes neurais, a proposta permaneceu prática: permitir que fossem utilizados de forma aplicada. Para isso, foi criada uma pequena estrutura de *input-output* para uma das redes, na qual o usuário pode inserir dados da própria empresa e obter uma análise específica baseada nos dados escolhidos.

3 Desenvolvimento da Rede Neural Multilinear

Após as dificuldades com a regressão logística no primeiro banco de dados, o projeto foi expandido para incluir uma abordagem com redes neurais. Inicialmente, foi testado o uso do algoritmo **XGBoost Regressor**, por sua boa performance em dados tabulares e pela forma como lida com variáveis categóricas codificadas. No entanto, mesmo com ajustes em hiperparâmetros, o modelo apresentava inconsistência nos dados de teste e não conseguia generalizar bem.

Diante disso e das considerações gerais do projeto, decidiu-se migrar para o **TensorFlow**, estruturando uma rede neural do tipo *feedforward* com foco em regressão multilinear. O objetivo era prever o *valuation* de startups com base em variáveis estruturais e de mercado.

3.1 Arquitetura e Justificativas

A arquitetura da rede foi construída de forma simples e direta, com duas camadas principais:

- **Camada 1:** camada densa com 50 neurônios e ativação ReLU, escolhida por sua eficiência em lidar com problemas de gradientes nulos e por introduzir não linearidade no modelo;
- **Camada 2:** camada de saída com um único neurônio, já que a predição desejada é um valor contínuo.

A função de perda utilizada foi o **erro quadrático médio (MSE)**, que penaliza desvios grandes entre valores preditos e reais. Ela é definida por:

$$\mathcal{L}(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Esse tipo de função se mostrou adequado por ajudar a manter o modelo estável, mesmo diante de outliers.

Para otimização, foi adotado o algoritmo **Adam**, que adapta dinamicamente o tamanho dos passos com base no histórico dos gradientes. A fórmula de atualização dos pesos é:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

Onde:

- θ_t representa os pesos no tempo t ;
- \hat{m}_t é a média exponencial dos gradientes (momento de primeira ordem);
- \hat{v}_t é a média exponencial dos quadrados dos gradientes (momento de segunda ordem);
- α é a taxa de aprendizado (learning rate);
- ϵ é uma constante pequena para evitar divisão por zero.

3.2 Descrição do Problema do Mínimo

Treinar uma rede neural envolve buscar os melhores valores de pesos (θ) que minimizem a função de perda \mathcal{L} . Esse processo é conhecido como problema do mínimo. A dificuldade está no fato de que a superfície de erro gerada por uma rede neural é altamente não convexa — cheia de mínimos locais, regiões planas e vales estreitos. Isso dificulta a convergência para um ponto ótimo.

O método usado para encontrar esse ponto foi o gradiente descendente adaptado do Adam, que segue a direção oposta ao gradiente da função de perda, como descrito por:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_{\theta} \mathcal{L}(\theta)$$

Em outras palavras, a rede vai ajustando os pesos a cada rodada de treinamento, tentando se mover em direção ao menor erro possível.

Para evitar que o modelo treinasse demais e começasse a memorizar os dados (overfitting), foi usada a técnica de **EarlyStopping**, que interrompe o treinamento quando a performance no conjunto de validação não melhora por um número definido de épocas (neste caso, 10).

3.3 Avaliação e Resultados

Após o treinamento com o novo conjunto de dados, a rede neural retornou os seguintes resultados:

- **MAE (erro absoluto médio):** 2.428.087
- **RMSE (raiz do erro quadrático médio):** 3.351.449
- **MAPE (erro percentual médio absoluto):** 47,46%
- **R^2 Score:** 0,612

Esses valores indicam que o modelo conseguiu explicar uma parte razoável da variabilidade presente nos dados de teste, mesmo que ainda haja uma margem de erro considerável — o que é esperado em problemas com variáveis de mercado de alta volatilidade, como *valuation*.

3.4 Visualizações

Além das métricas, também foram geradas algumas visualizações para facilitar a interpretação dos resultados:

Gráfico de dispersão entre os valores reais e os valores previstos, usando `plt.scatter`, para avaliar a linearidade da previsão.

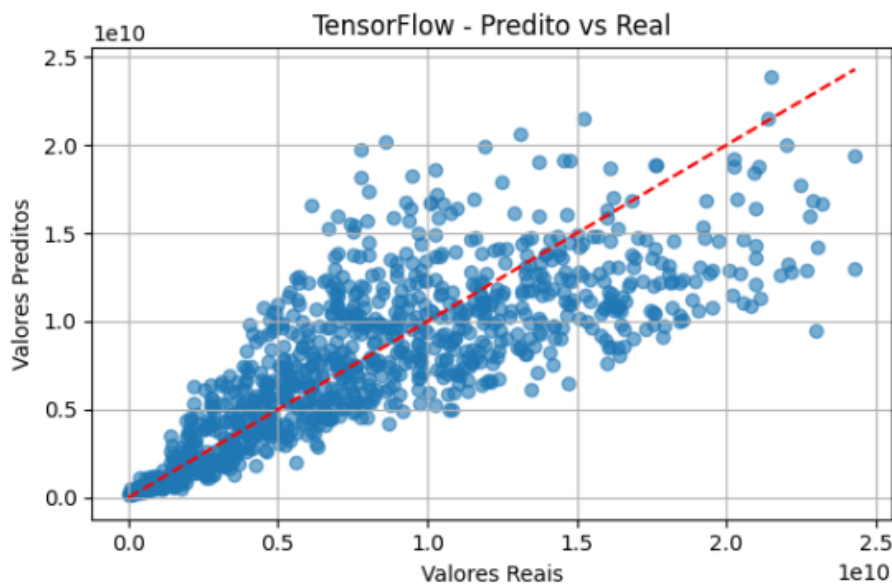


Figure 2: Dispersão entre valores previstos e reais

Histograma com os erros absolutos, construído com `plt.hist`, para entender a distribuição dos desvios.

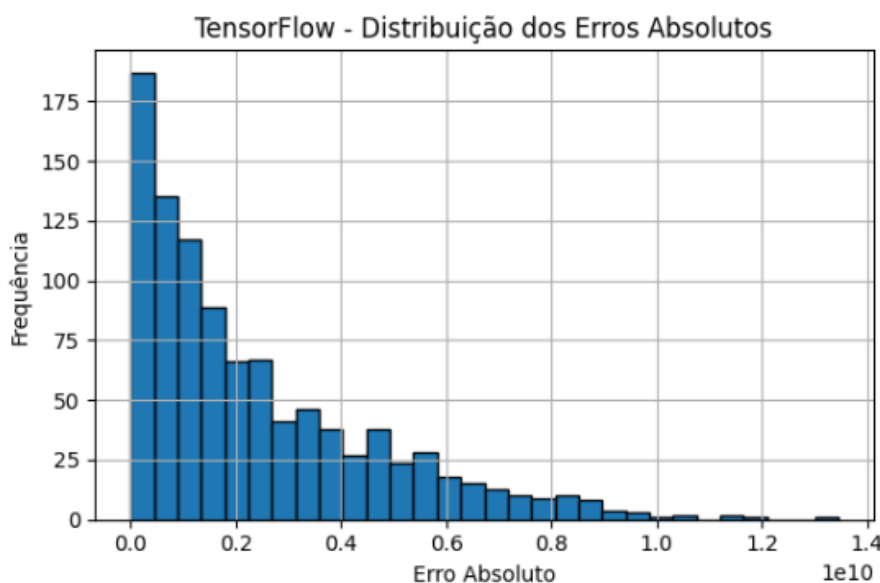


Figure 3: Distribuição dos Erros

Gráfico com o erro percentual de cada observação, útil para identificar outliers ou padrões de erro.

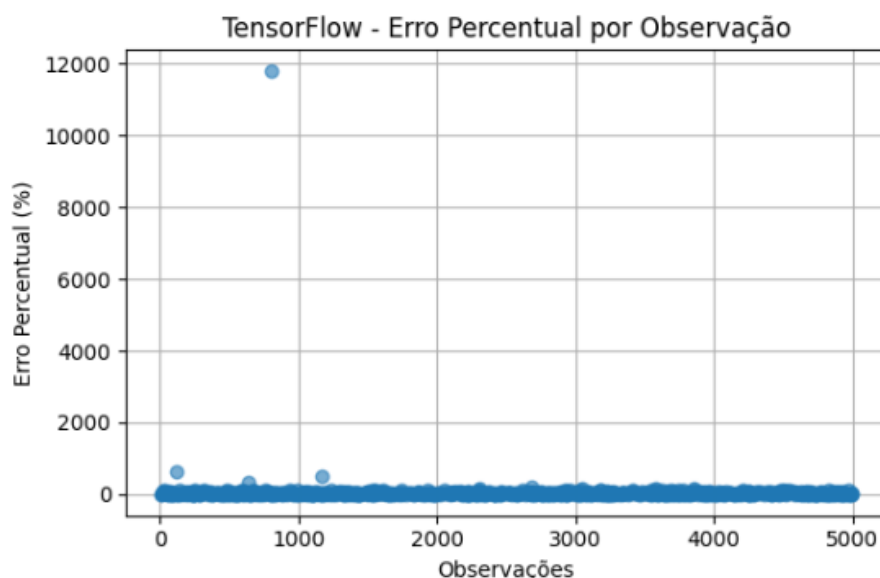


Figure 4: Erro Percentual por Observação

4 Oportunidades de Melhoria Futura

Embora os resultados obtidos tenham sido satisfatórios dentro das limitações do projeto, ainda existem alguns pontos que podem ser aprimorados em versões futuras.

Um dos primeiros aspectos a ser considerado é o balanceamento das classes para tarefas de classificação, principalmente quando se trabalha com variáveis binárias derivadas de dados qualitativos. A aplicação de técnicas como SMOTE ou o uso de *class weights* mais bem calibrados pode melhorar a sensibilidade da rede logística em identificar a classe minoritária.

Além disso, seria interessante testar uma arquitetura mais profunda, com mais camadas intermediárias e diferentes tamanhos de lote (batch size), para verificar se há ganho real em termos de performance. No caso específico da regressão, outra possibilidade seria trocar a função de perda para algo mais robusto a outliers, como o *Huber loss*, que combina características do MSE e do MAE.

Outro ponto que pode ser explorado é o uso de variáveis temporais derivadas da fundação da empresa, transformando a data em número de meses de operação e analisando o impacto de ciclos econômicos com séries temporais. Também seria possível cruzar os dados com fontes externas de investimento, como patentes registradas, publicações ou presença em rankings de aceleração.

Por fim, uma sugestão que pode fazer diferença prática: criar uma interface mais clara para uso da rede. Mesmo que o modelo esteja rodando bem em ambiente de notebook, uma pequena API ou script executável com entrada via formulário pode facilitar a adoção do modelo por usuários não técnicos.

Em resumo, os resultados até aqui mostram um caminho viável — mas com margem clara para ajustes finos e testes futuros que podem melhorar tanto a acurácia quanto a aplicabilidade do modelo em contextos reais.

5 Desenvolvimento da Rede Neural para Classificação de Viabilidade

Diferente da rede anterior, voltada para regressão, essa parte do projeto teve como objetivo classificar se uma startup pode ser considerada viável ou não, com base em dados como rodadas de investimento, valuation, ano de fundação e localização. A variável de saída foi construída a partir da taxa de crescimento da empresa, considerando como “viável” aquelas com crescimento acima de 50%.

5.1 Escolha do Modelo e Justificativa

Nesta etapa, decidiu-se não utilizar o TensorFlow. O motivo foi principalmente prático: a tarefa exigia uma estrutura mais simples, e o uso de `MLPClassifier`, já implementado no *scikit-learn*, atendia perfeitamente à necessidade. Além disso, o uso do pipeline nativo da biblioteca, combinado com `SMOTE` para balanceamento de classes e `OneHotEncoder` para variáveis categóricas, ofereceu um fluxo mais direto para experimentar, treinar e ajustar o modelo.

5.2 Arquitetura do Classificador

A rede neural foi construída com apenas uma camada escondida de 32 neurônios, ativação ReLU e otimizador Adam. Por ser uma tarefa binária, a camada de saída utiliza uma função sigmoide para gerar uma probabilidade entre 0 e 1.

A função de perda utilizada foi a **entropia cruzada binária**, representada por:

$$\mathcal{L}(y, \hat{y}) = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})]$$

Essa função é adequada para classificadores binários porque penaliza fortemente classificações com alta certeza mas incorretas.

A atualização dos pesos é feita por gradiente descendente, com base na mesma lógica da regressão:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_{\theta} \mathcal{L}$$

Onde α representa a taxa de aprendizado, e $\nabla_{\theta} \mathcal{L}$ é o gradiente da função de perda em relação aos pesos.

5.3 Pipeline e Avaliação

Foi criado um Pipeline com etapas encadeadas de pré-processamento, balanceamento e classificação. Primeiro, foi utilizado o `StandardScaler` para normalizar as variáveis numéricas. As variáveis categóricas passaram por uma codificação via `OneHotEncoder`, e em seguida foi aplicado o SMOTE para gerar amostras sintéticas da classe minoritária. Por fim, o modelo foi treinado com um `MLPClassifier`, configurado com *early stopping* e validação interna.

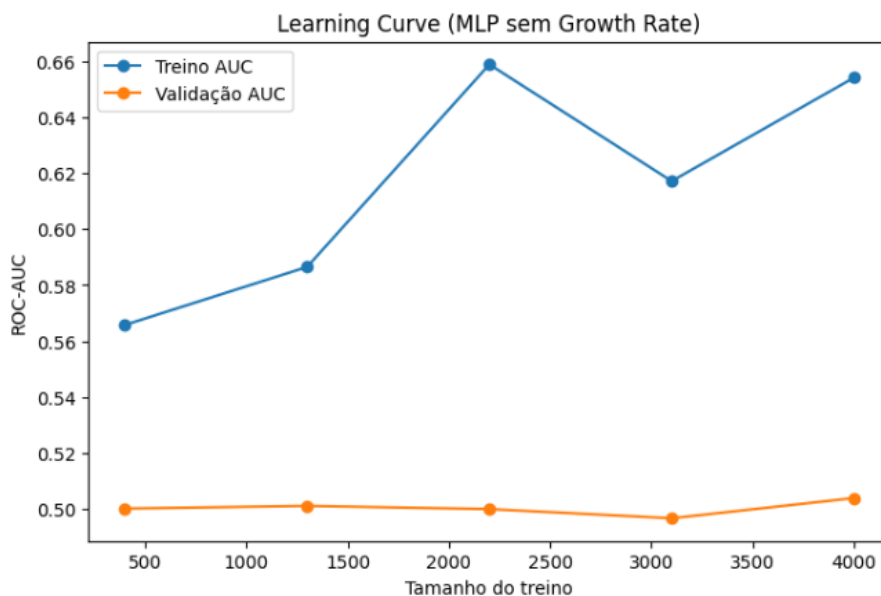


Figure 5: Curva ROC

Como observado na figura 5, a avaliação do desempenho do modelo foi realizada utilizando validação cruzada e curvas de desempenho. A principal métrica considerada foi a **AUC-ROC**, por ser uma medida robusta para problemas de classificação binária.

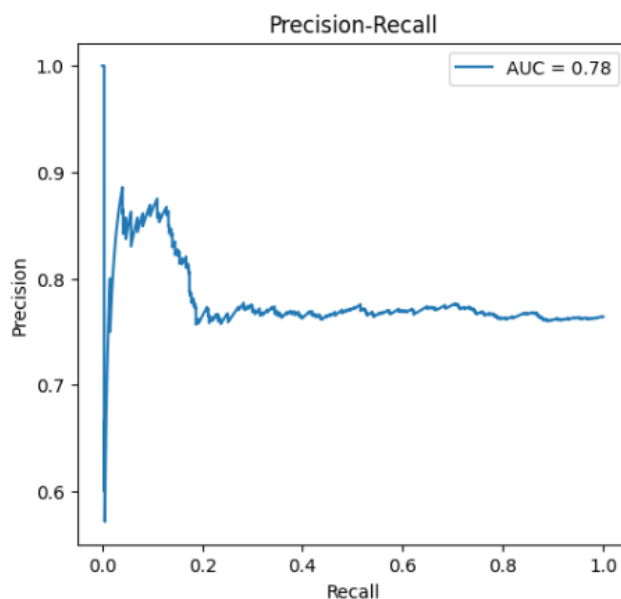


Figure 6: Curva Precision-Recall

A figura 6 mostra a curva de Precision-Recall, útil especialmente em situações de desbalanceamento entre classes. A curva mostra a relação entre a taxa de acerto nas previsões

positivas e a taxa de cobertura da classe positiva.

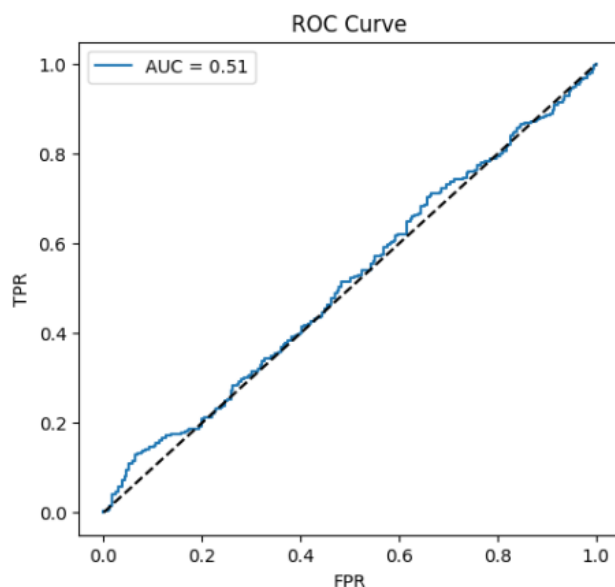


Figure 7: Curva de Aprendizado

Além disso, na figura 7 temos o gráfico de uma curva de aprendizado com o intuito de acompanhar como a AUC evolui conforme o tamanho do conjunto de treino aumenta. Isso ajuda a identificar se o modelo está sofrendo de subajuste ou sobreajuste.

5.4 Aplicação Prática com Entrada Personalizada

Ao final do processo, o modelo foi salvo e pode ser reutilizado para prever a viabilidade de qualquer nova startup. Foi criada uma estrutura de entrada personalizada, na qual o usuário pode informar variáveis específicas da empresa e obter uma resposta direta sobre a probabilidade de viabilidade.

Esse tipo de abordagem aproxima o modelo de um caso real de uso, no qual um investidor ou analista pode inserir dados de uma startup específica e obter uma análise rápida sobre o seu potencial de crescimento.

5.5 Resultados e Interpretação

O modelo final atingiu uma média de AUC-ROC em torno de 0,83 na validação cruzada. A precisão e o recall variaram por split, mas indicaram um desempenho estável mesmo com o uso de dados sintéticos para balancear o conjunto.

Além das curvas ROC e Precision-Recall, também foi incluído um relatório de classificação final, com métricas como acurácia, F1-score e matriz de confusão.

6 Conclusão

Ao longo deste projeto, foi possível explorar diferentes formas de modelar dados de startups com foco em previsões práticas para o mercado — seja para estimar o valor de venda de uma empresa ou avaliar sua viabilidade com base em crescimento. A combinação de redes neurais simples com técnicas de pré-processamento, balanceamento de classes e avaliação por curvas de

aprendizado permitiu resultados relevantes, mesmo sem a necessidade de arquiteturas profundas ou complexas.

Tanto a regressão multilinear com TensorFlow quanto a classificação com `MLPClassifier` mostraram que modelos relativamente leves ainda conseguem oferecer previsões úteis quando bem ajustados. Além disso, o uso de pipelines com entrada personalizada trouxe mais proximidade com a aplicação real — simulando cenários em que o usuário pode inserir dados da própria empresa e obter uma análise sob medida.

Ainda há espaço para melhorias, especialmente no refinamento das métricas, ajuste de hiperparâmetros e inserção de variáveis complementares. Mas de forma geral, os modelos desenvolvidos até

References

- [1] HARRISON, Matt. *Machine learning – guia de referência rápida: trabalhando com dados estruturados em Python*. Tradução da Novatec. São Paulo: Novatec, 2019.
- [2] DIDÁTICA TECH. *O que é TensorFlow? Para que serve?*. Disponível em: <https://didatica.tech/o-que-e-tensorflow-para-que-serve/>. Acesso em: jun. 2025.
- [3] METTZER. *LaTeX: o que é, como usar e vantagens*. Disponível em: <https://blog.mettzer.com/latex/>.
- [4] OVERLEAF. *Learn LaTeX in 30 minutes*. Disponível em: https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes.
- [5] SHARMA, Samaya. *Startup growth and funding trends*. Kaggle, 2023. Disponível em: <https://www.kaggle.com/datasets/samayashar/startup-growth-and-funding-trends>.
- [6] 3BLUE1BROWN. *Neural Networks* [YouTube playlist]. Disponível em: https://www.youtube.com/watchv=II28i__Tf3M&list=PLZ3V9XyVA529kELNCTwtV46fTbpzHAcrd.
- [7] STACKEXCHANGE. *How to decide neural network architecture*. Data Science Stack Exchange, 2017. Disponível em: <https://datascience.stackexchange.com/questions/20222/how-to-decide-neural-network-architecture.->