Previsão de Intervalos dos Valores de Venda de Casas

César Augusto Julio da Silva

Instituto de Computação
Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro, RJ – Brasil

cesarsilva06@dcc.ufrj.br

Abstract. *This report contains an analysis of three methods linked to the code created in accord with the guidelines related to the replacement exam of the first exam of the machine learning course, lectured by the teacher João Carlos.*

Resumo. *Este relatório contém uma análise da aplicação de três métodos aprendidos durante o curso de aprendizado de máquina aplicado a um conjunto de dados compostos por características e valores de vendas de casas.*

1. Introdução

O objetivo deste relatório é comentar sobre os experimentos realizados no conjunto de dados *House Prices - Advanced Regression Techniques* e seus resultados. Essa base foi escolhida por já ter sido utilizada em competições e ser uma base razoavelmente boa para aprendizado de máquina.

2. Descrição da base

O conjunto de dados é composto por aproximadamente 1400 linhas e 81 colunas com diversas características de imóveis (valores categóricos) e seus preços de venda (valores numéricos), onde o foco é predizer por meio do aprendizado de máquina, e dado as características da casa, qual é o valor final de venda.

3. Pré Processamento dos dados

3.1. Análise de distribuição e correlação.

Inicialmente, foi executada uma análise do conjunto para averiguar a distribuição dos valores alvos, quantidade de dados faltantes e correlações entre os mesmos.

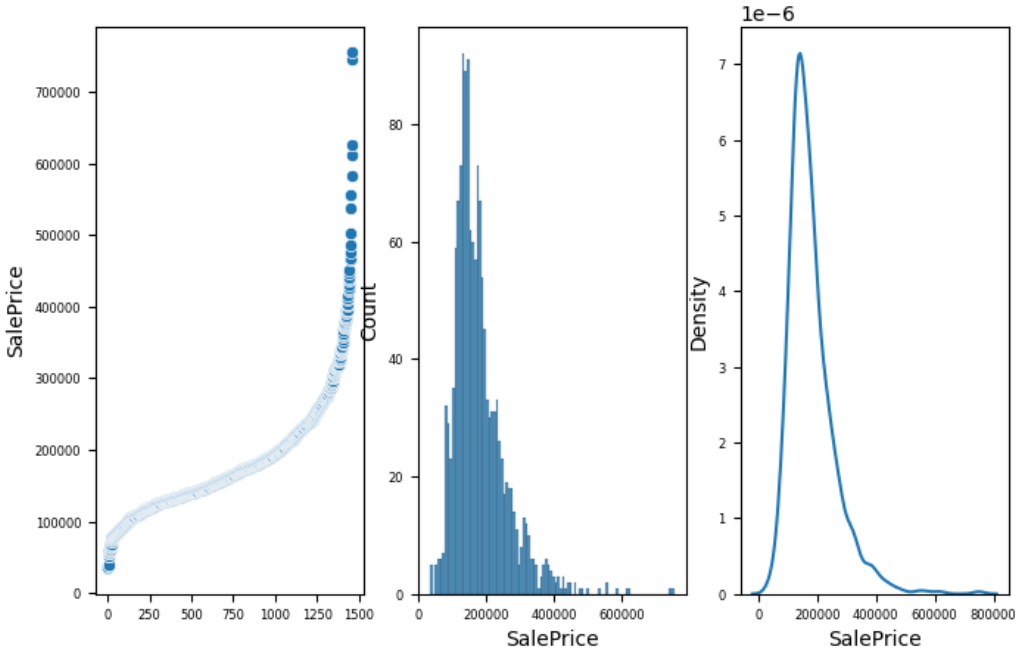

Figura 1. Distribuição dos valores alvos.

É possível observar uma grande concentração de valores na faixa de preço de $200000, o que irá influenciar um decréscimo no desempenho de classificação dos valores mais distantes a direita (valores próximos de $700000).

Além disso, também foi averiguado o mapa de calor com a correlação de valores faltantes.

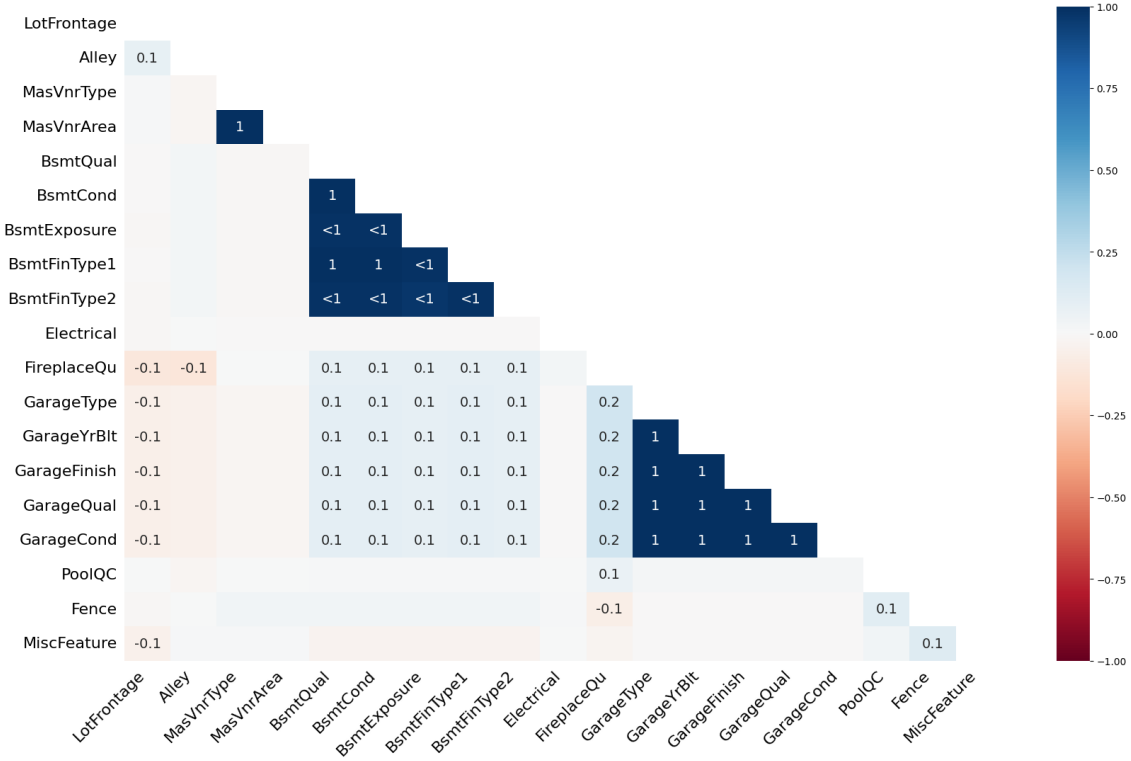

Figura 2. Mapa de calor com correlação dos valores faltantes.

Aqui, é possível observar que existe um razoável número de colunas com correlação alta, o que é óbvio devido ao fato dessas colunas descreverem característica de uma mesma propriedade do imóvel contido no conjunto (por exemplo: garagem está descrito em *GarageType*, *GarageYrBlt*, *GarageFinish*, etc). Isso demonstra que existe a possibilidade de métodos de imputação não terem um desempenho muito bom (pois eles não vão levar em consideração essa correlação).

3.2. Remoção de valores nulos.

Os valores nulos foram tratados de duas formas: em um conjunto todas as colunas com tais valores foram removidas, em outro conjunto todas as colunas com mais de 10% de valores nulos foram removidas e o restantes foram imputados a moda (valor com maior frequência de ocorrências), já que não existe uma ordem entre os valores categóricos. Foram denominados os nomes *df_rmv_all* e *df_mode* respectivamente.

3.3. Criação das categorias.

Como a base de dados é focada em previsão dos valores e o experimento realizado era focado em métodos de classificação, foi criado uma nova coluna (*PriceRange*) baseado em intervalos de valores de preços dos imóveis. Em um caso, dividimos em dez intervalos focando em densidade (mesma quantidade de variáveis) em outra focamos em preço (mesmo intervalo de preço) para fins de comparação. Foram denominados os nomes *dfs_quantity* e *dfs_range* para os conjuntos, respectivamente.

...	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold	SaleType	SaleCondition	PriceRange
...	115	0	0	0	0	11	2009	WD	Abnorml	34900:106250
...	0	0	0	0	0	10	2006	WD	Abnorml	34900:106250
...	0	0	0	0	0	5	2009	WD	Abnorml	34900:106250
...	0	0	0	0	0	1	2007	WD	Normal	34900:106250
...	172	0	0	0	0	7	2008	WD	Normal	34900:106250
...	...	...	...	...	...	...	...	...	...	...
...	0	0	192	0	0	1	2009	New	Partial	278000:755000
...	0	0	0	0	0	3	2010	New	Partial	278000:755000
...	0	0	0	0	0	7	2006	WD	Normal	278000:755000
...	0	0	0	555	0	7	2007	WD	Abnorml	278000:755000
...	0	0	0	0	0	1	2007	WD	Normal	278000:755000

Figura 3. Parte do conjunto de dados com divisão por quantidade.

4. Métodos utilizados

Os métodos utilizados foram: rede neural, árvore de decisão e SVM. Esses métodos foram utilizados basicamente pela sua facilidade de implementação e averiguação dos resultados.

4.1. Árvore de Decisão.

A aplicação da árvore de decisão foi feita pelo método *TreeDecisionClassifier* do scikit que lida com variáveis categóricas ao invés de regressão. Somente os parâmetros padrão foram aplicados por falta de tempo e foram utilizadas 5 divisões para o *StratifiedKFold*. No *dfs_quantity*, a acurácia média foi de aproximadamente 0.3587 para o *df_rm_all* e 0.3698 para o *df_mode*.

Já no *dfs_range*, ocorreu um problema, dado que certos intervalos tinham um número de instância menor do que o número de *folds* e emitia um alerta. A acurácia foi muito mais alta mas é provável que seja devido ao fato de como o grupo de dados está distribuído por valor algumas classes têm muito mais instâncias do que outras e a árvore aprendeu muito bem essas classes mais populosas e menos as outras. De todas as formas a acurácia média foi 0.6524 para o *df_rmv_all* e 0.6515 para o *df_mode*.

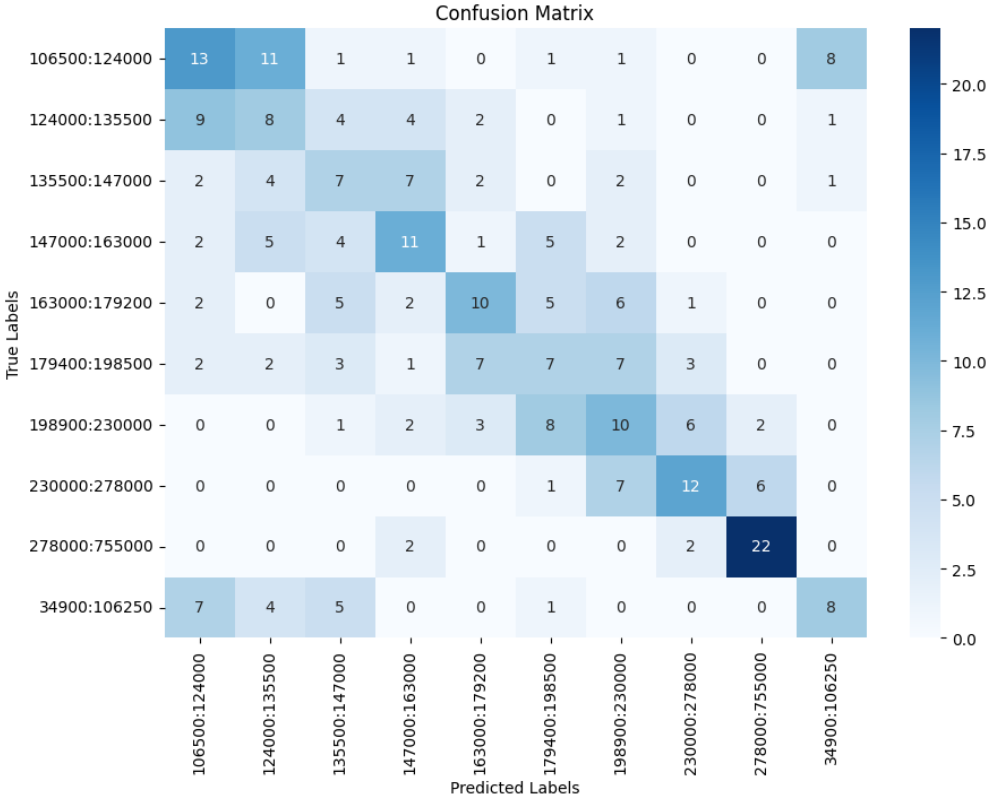

Figura 4. Matriz de confusão do *df_mode* em *dfs_quantity*.

É possível observar claramente que existe uma distribuição muito melhor da quantidade de variáveis no *dfs_quantity* dado que esse foi o foco, apesar de isso trazer a acurácia do conjunto em específico para baixo.

4.2. SVM

O fenômeno que ocorreu na árvore de decisão ocorreu também no SVM. A implementação foi bem parecida, utilizando o SVC do scikit para classificação e os

parâmetros padrão novamente. A acurácia do *dfs_quantity* ficou por volta de 0.2260 (*df_rmv_all*) e 0.2234 (*df_mode*) com uma acurácia de teste idêntica em 0.1986.

Novamente, no dfs_range, a acurácia sem mantém bem mais alta com uma média de 0.5162 e 0.5128 respectivamente e uma acurácia de teste de 0.5171 e 0.5136, também respectivamente, muito próximo ao de treinamento.

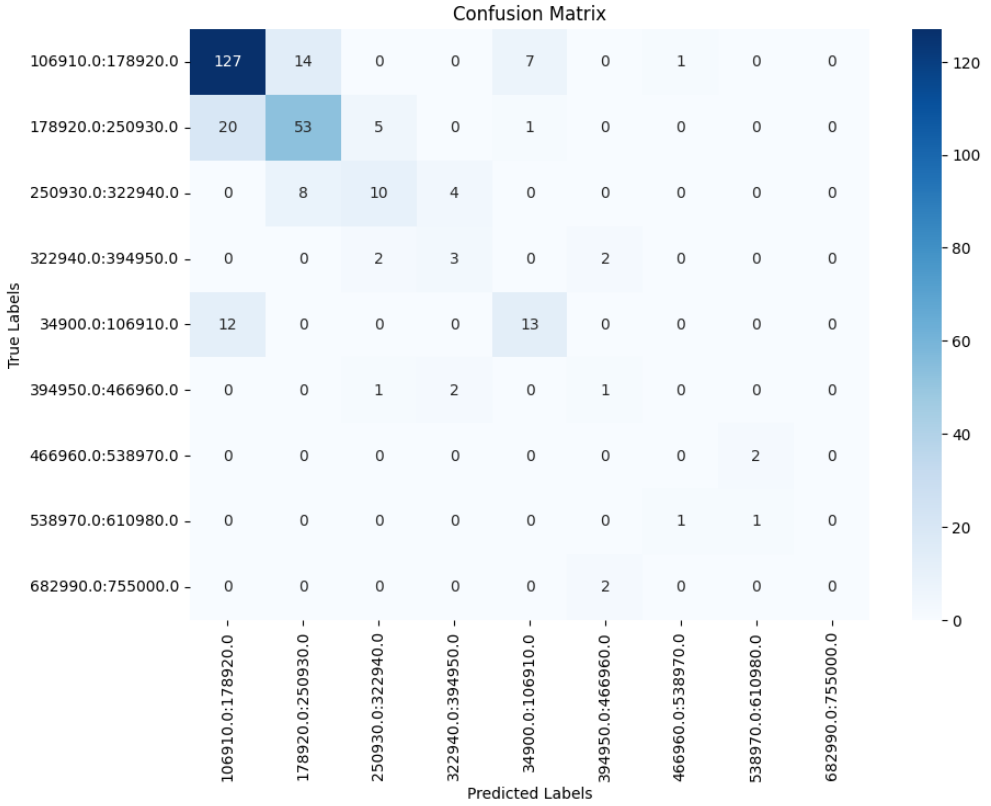

Figura 5. Matriz de confusão de um dos conjuntos de *df_range*.

Observe que nessa matriz, vemos que a distribuição dos valores está muito ruim, se tivermos um conjunto de testes que não seja focado na densidade do *dataframe* original e sem em uma distribuição diferente é muito provável que a acurácia de teste desça e muito.

4.3. Rede Neural

Para a aplicação da rede neural, foi criado um algoritmo que itera por diversos parâmetros que compõem o *MLPClassifier*, a função de classificação utilizado para esse método. Esse algoritmo guardava os parâmetros que tinham o melhor resultado de classificação, levando em conta os seguintes parâmetros: *learning_rate*, *activation*, *hidden_layers_size*, e *n_sizes* (esse último pertencente ao *StratifiedKFold*). Chegou ao seguinte resultado, respectivamente: *constant*, *identity*, (2, 2, 8), e 5. Era possível também alterar o parâmetro *solver* mas isso necessitaria uma modificação da base dados

para funcionar então foi mantido o valor padrão *adam*.

O primeiro teste foi executado no conjunto dividido por quantidade (*dfs_quantity*). O rendimento foi ruim em ambos os conjuntos, tanto o que removeu todos os valores nulos quanto o que teve imputação da moda (esse sendo o que teve melhor pontuação), com pontuação de aproximadamente 0.1575 e 0.2157 e perda razoavelmente constante em todos os *folds*.

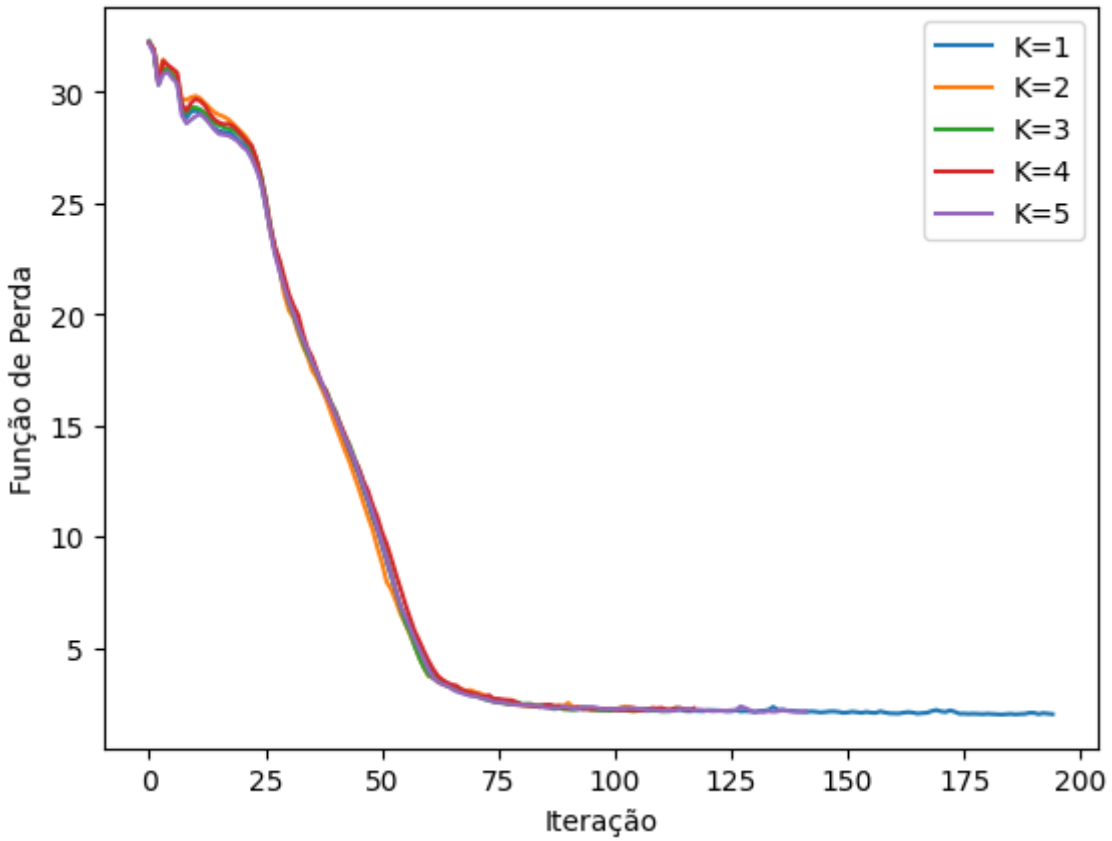

Figura 6. Curva de perda

Podemos observar que a curva de perda tem uma queda constante até chegar perto da iteração 60 onde ela estabiliza. As variações finais e iniciais parecem serem devido ao fato de pegar instâncias mais incomuns, como aquelas que estão nos intervalos de preço mais alto ou mais baixo.

O segundo teste foi executado no conjunto dividido por faixa de preço (*dfs_range*) e teve um rendimento muito melhor apesar de ter problemas dado que certos intervalos tinham um número de instância menor do que o número de *folds*. Ainda assim houve uma pontuação de aproximadamente 0.2534 para o primeiro conjunto (*df_rmv_all*) e 0.6917 (*df_mode*). O aumento substancial da pontuação do segundo *dataset* mostrou que a imputação talvez não tenha sido algo tão ruim, pelo menos não nessa distribuição de variáveis.

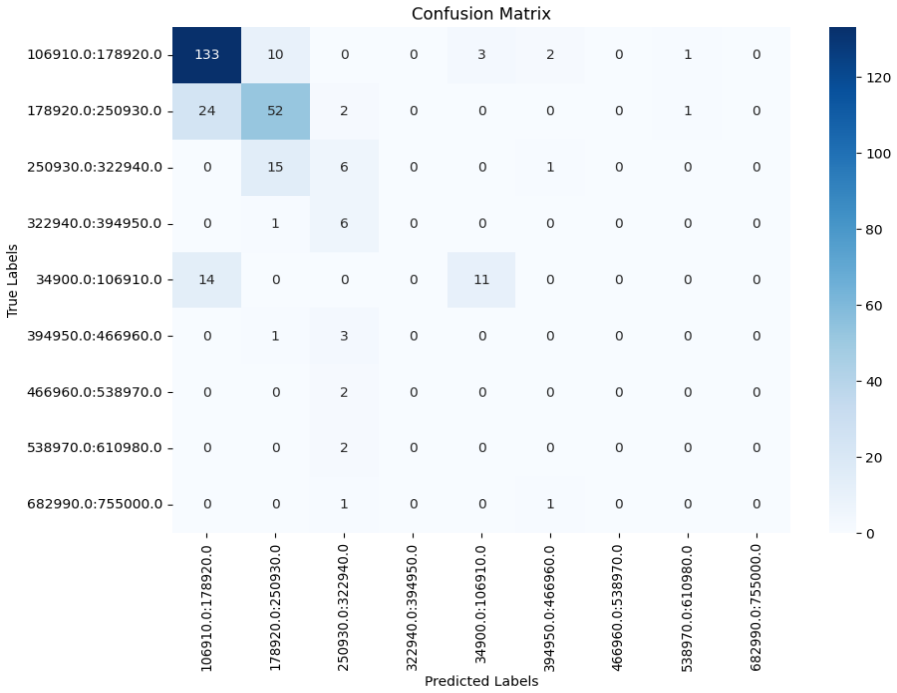

Figura 7. Matriz de confusão do último experimento em Rede Neural.

Contudo, podemos observar na matriz de confusão que grande parte das variáveis foram classificadas corretamente devido ao fato de que grande parte delas ficam no mesmo intervalo de preço. Isso mostra na realidade que não necessariamente esse modelo é melhor mas somente que ele está mais apto em adivinhar as classes mais populosas.

5. Trabalhos relacionados

Infelizmente, como o objetivo desse conjunto é realizar previsão de vendas, não existem trabalhos relacionados que possam ser comparados aos meus resultados. Contudo, encontrei um github onde o autor publicou algumas métricas de experimentos que ele fez para comparação. Essas métricas são *RMSE* e *k-Fold Cross Validation*.

- Linear Regression: 0.42793480397157035,0.4752199075347933
- Ridge: 0.3957886167433282,0.4167270749625921
- Lasso: 0.4059493256188701, 0.4297643272515321
- K- Nearest Neighbour: 0.41351487769327555
- Decision Tree: 0.4583579345988703 , 0.45825272349446555
- Random Forest: 0.38616747296757176, 0.403804172243945
- Support Vector Regressor: 0.3900469727418305, 0.40963206887105647
- Gradient Boosting Regressor: 0.4118219430457788, 0.4292489907640939
- Stacked Regressor: 0.3769718491202983, 0.4093903027876036

Figura 8. Métricas utilizadas pelo autor

Pode-se observar que grande parte dos valores no CV ficaram abaixo de 0.5 mostrando a dificuldade que é conseguir criar um aprendizado de máquina confiável para esse conjunto de dados.

6. Resultados

Os resultados foram surpreendentemente piores do que o esperado. Levando em consideração a divisão do dataset *dfs_quantity*, eu esperava uma acurácia média de 0.5 e um score similar, contudo a pontuação ficou bem mais baixa.

Acredito que seja devido ao fato de termos uma grande quantidade de variáveis categóricas e talvez não termos usado os melhores parâmetros para esse tipo de dataset, talvez nem mesmo o melhor tipo de métodos.

7. Conclusão

Em geral, alguns dos propósitos desse trabalho era averiguar se a diferença entre imputação e remoção afetaria a base e outra qual dispersão de intervalo seria melhor. Podemos observar que, apesar de depender do método, a diferença entre remoção e imputação não afetou severamente o desempenho dos conjuntos, podendo dar uma leve vantagem positiva ao método de imputação de moda. Já com relação a dispersão de intervalo, fica difícil dizer qual seria melhor.

Se o autor quer um aprendizado mais "*overfitted*" para a dispersão de instâncias da base dados, dividir focado no intervalo de preços seria melhor. Contudo, se quer algo um pouco mais igualitário, mesmo que se sacrifique precisão de valor alvo, o melhor é dividir focado na dispersão por grupos, mantendo números iguais de instâncias em cada classificação.

Para o futuro, seria melhor aplicar um ensemble para ver se existe uma melhoria no desempenho, utilizar outros métodos testados em outros trabalhos como o K-Nearest Neighbor ou Random Forest, e, aplicar outros métodos para lidar com valores nulos como regressão logística ou algo que leve em consideração a correlação mais a fundo.

8. Referências

da Silva, J. C. P. (2023) "ICP363 - Introdução ao Aprendizado de Máquina. Aula 10 - Redes Neurais". Disponível em: https://drive.google.com/drive/folders/16Fb9L-WNqZYHTlkv-4R_wpyZc5saNp0F

sklearn.neural_network.MLPClassifier. Scikit Learn. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier.predict

Creating multiple subplots using plot.subplots. **Matplotlib.** Disponível em: https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html

How is it possible that validation loss is increasing while validation accuracy is increasing as well. **Stack Exchange.** Disponível em: https://stats.stackexchange.com/questions/282160/how-is-it-possible-that-validation-l

oss-is-increasing-while-validation-accuracy

ankita1112/House-Prices-Advanced-Regression. **Github.** Disponível em: https://github.com/ankita1112/House-Prices-Advanced-Regression

Using Missingno to Diagnose Data Sparsity. **Kaggle.** Disponível em:

https://www.kaggle.com/code/residentmario/using-missingno-to-diagnose-data-sparsity/notebook