PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina

Lucas Guimarães Cavalheiro

Inteligência Artificial na Previsão de Preços de Aluguel Imobiliário: Abordagens voltadas ao consumidor

Belo Horizonte
Outubro de 2023

Lucas Guimarães Cavalheiro

Inteligência Artificial na Previsão de Preços de Aluguel Imobiliário: Abordagens voltadas ao consumidor

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Belo Horizonte
Outubro de 2023

SUMÁRIO

1. Preparação dos Dados para os Modelos de Aprendizado de Máquina

Como parte do processo de preparação, teste e validação, foi constatado que as colunas *hoa*, *rent amount*, *fire insurance* e *property tax* removidas durante o relatório anterior possuem impacto significativo na acurácia dos resultados dos modelos de *machine learning,* considerando isso, os *jupyter notebooks* anteriores foram alterados para incluirem estas colunas novamente, recomenda-se que eles sejam executados novamente para atualizar o arquivo .csv que será utilizado durante este relatório, afim de alcançar resultados similares aos que serão demonstrados a seguir.

Não foi necessário nenhum preparo em especial para a utilização dos dados nos modelos de *machine learning,* bastando a divisão dos dados entre bases de teste e de treinamento, 20% e 80%, respectivamente. A biblioteca *scikit learn* disponibiliza a função *train_test_split* para facilitar a divisão dos dados entre bases de teste e treinamento:

```
data = data.sample(frac=1.0, random_state=0).reset_index(drop=True)
X = data.drop('total', axis = 1)
y = data['total']

X = pd.DataFrame(X, index=X.index, columns=X.columns)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)
```

Opcionalmente, podemos usar um *StandardScaler* para padronizar os dados, removendo a média e escalando a variância em uma unidade, isso tornará os dados mais manejáveis para os modelos:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Foi preparado um dicionário com todos os modelos candidatos para a solução do problema, todas as implementações destes modelos também foram fornecidos pela biblioteca *scikit learn,* este dicionário de será usado na próxima parte deste relatório para facilitar os testes dos modelos:

```
models = {
    "Ridge":{"model":Ridge() },
    "SVR": {"model": SVR()},
    "DecisionTreeRegressor":{"model":DecisionTreeRegressor() },
    "LinearRegression":{"model":LinearRegression() },
    "Lasso":{"model":Lasso() },
    "RandomForestRegressor":{"model":RandomForestRegressor() },
    "AdaBoostRegressor":{"model":AdaBoostRegressor() },
    "MLPRegressor":{"model":MLPRegressor() },
    "GradientBoostingRegressor":{"model":GradientBoostingRegressor() },
    "KNeighborsRegressor": {"model": KNeighborsRegressor()}
}
```

2. Aplicação de Modelos de Aprendizado de Máquina

Para testar os modelos candidatos, serão usadas três métricas: tempo de treinamento, RMSE e R2 score. Levamos em conta o tempo de treinamento pois não só queremos resultados precisos, mas queremos eles dentro de um tempo de treinamento aceitável.

Usamos o RMSE *(root mean square error)* para mensurar a raiz quadrática média dos erros entre valores de teste e predições, com esta métrica podemos medir a acurácia dos modelos, quanto menor o RMSE, mais acurácia o modelo apresenta.

Aproveitaremos a facilidade que o *scikit learn* nos oferece e iremos medir o *R2 score* dos modelos também. Com as medidas *R2 score* (R-quadrado), poderemos ver o quanto cada modelo explica a variabilidade das predições ao redor da média, sendo assim, quanto maior o R-quadrado de um modelo, melhor ele performou.

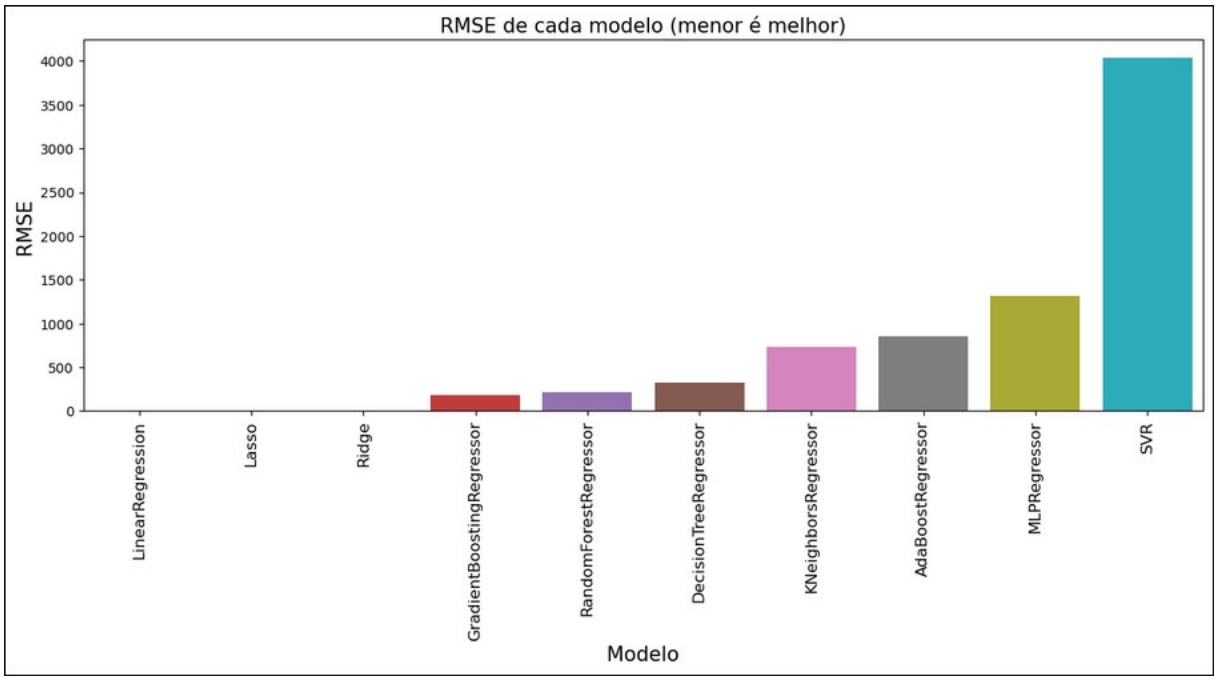
Os modelos poderão ser testados executando o código a seguir, com os resultados dos testes já sendo guardados em um dataframe:

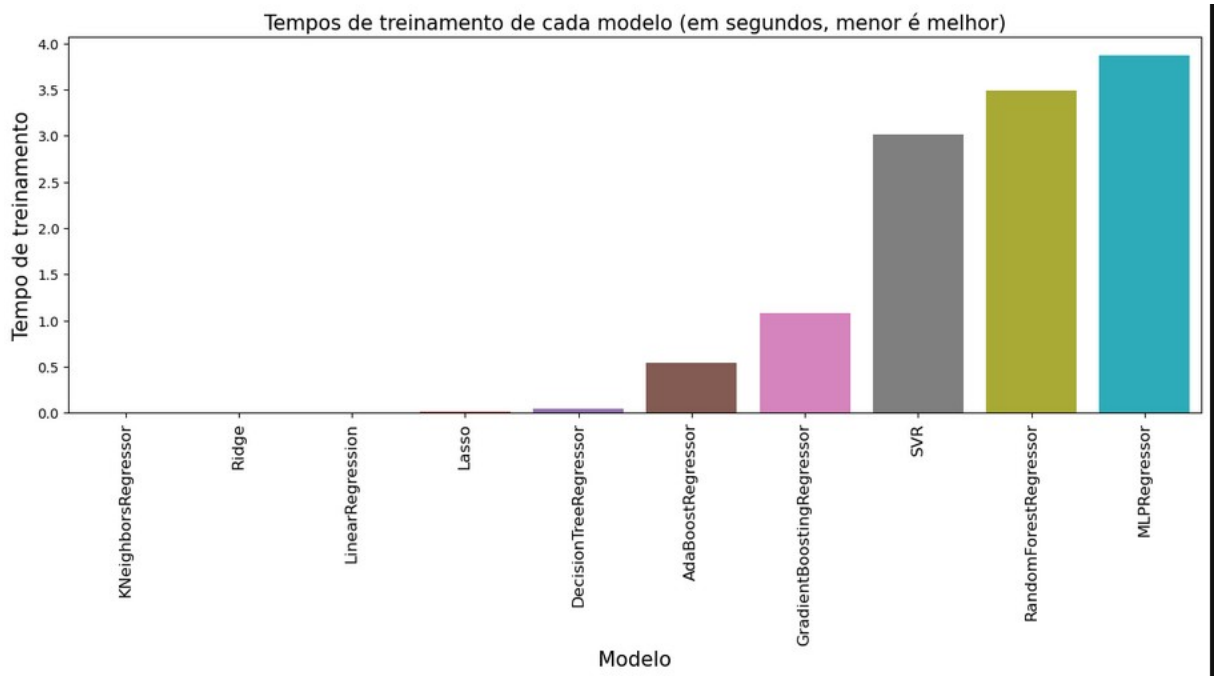
```
result_list = []

for name, model_item in models.items():
    model = model_item['model']
    model.fit(X_train,y_train)

    # Cross validation
    result = cross_validate(model, X_train,y_train, cv = 5, scoring='neg_mean_squared_error')

    # RMSE
    RMSE = [(-x)**0.5 for x in result['test_score']]
    RMSE = int(sum(RMSE)/len(RMSE))

    # Training time
    training_time = round(sum(result['fit_time']) / len(result['fit_time']), 4)

    # R2 score
    score = r2_score(y_test,model.predict(X_test))

    result_list.append([name, RMSE, score, training_time])

df_results = pd.DataFrame(result_list, columns = ['model','RMSE', 'r2_score', 'training_time'])
df_results.sort_values(by=['RMSE', 'r2_score'], ascending=True, inplace=True)
df_results.reset_index(inplace=True,drop=True)
df_results
```

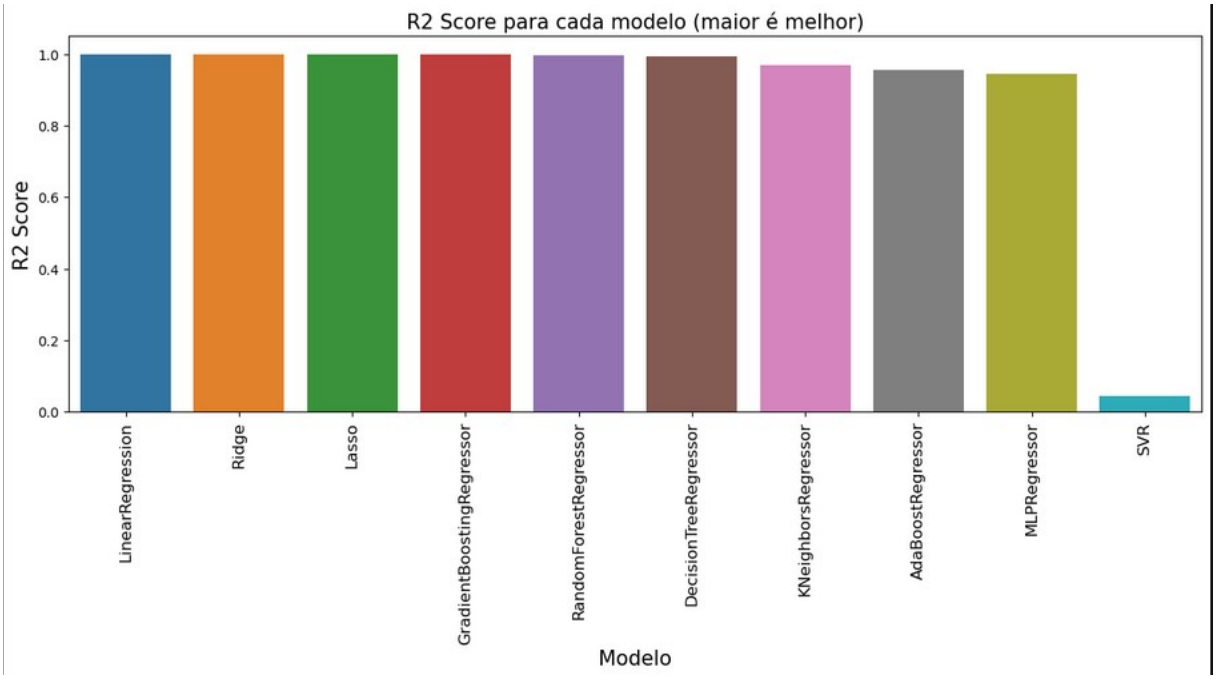
Após a execução do código, serão exibidos os seguintes resultados para os testes dos modelos, já ordenados por melhor *RMSE* e *R2 score*:

	model	RMSE	r2_score	training_time
0	LinearRegression	4	1.000000	0.0038
1	Lasso	6	0.999999	0.0114
2	Ridge	6	0.999999	0.0034
3	GradientBoostingRegressor	180	0.998570	1.0851
4	RandomForestRegressor	209	0.996436	3.4938
5	DecisionTreeRegressor	323	0.993198	0.0515
6	KNeighborsRegressor	727	0.969245	0.0012
7	AdaBoostRegressor	851	0.955774	0.5410
8	MLPRegressor	1319	0.943875	3.8726
9	SVR	4039	0.046190	3.0131

Podemos notar que o melhor candidato para a solução do problema será o modelo de regressão linear, mas podemos examinar os resultados dos testes mais a fundo usando gráficos:


RMSE de cada modelo (menor é melhor)
RMSE
4000
3500
3000
2500
2000
1500
1000
500
0
LinearRegression
Lasso
Ridge
GradientBoostingRegressor
RandomForestRegressor
DecisionTreeRegressor
KNeighborsRegressor
AdaBoostRegressor
MLPRegressor
SVR
Modelo


Tempos de treinamento de cada modelo (em segundos, menor é melhor)
Tempo de treinamento
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
KNeighborsRegressor
Ridge
LinearRegression
Lasso
DecisionTreeRegressor
AdaBoostRegressor
GradientBoostingRegressor
SVR
RandomForestRegressor
MLPRegressor
Modelo



3. Discussão dos Resultados

Agora que temos um modelo final escolhido, iremos treiná-lo e averiguar seus resultados:

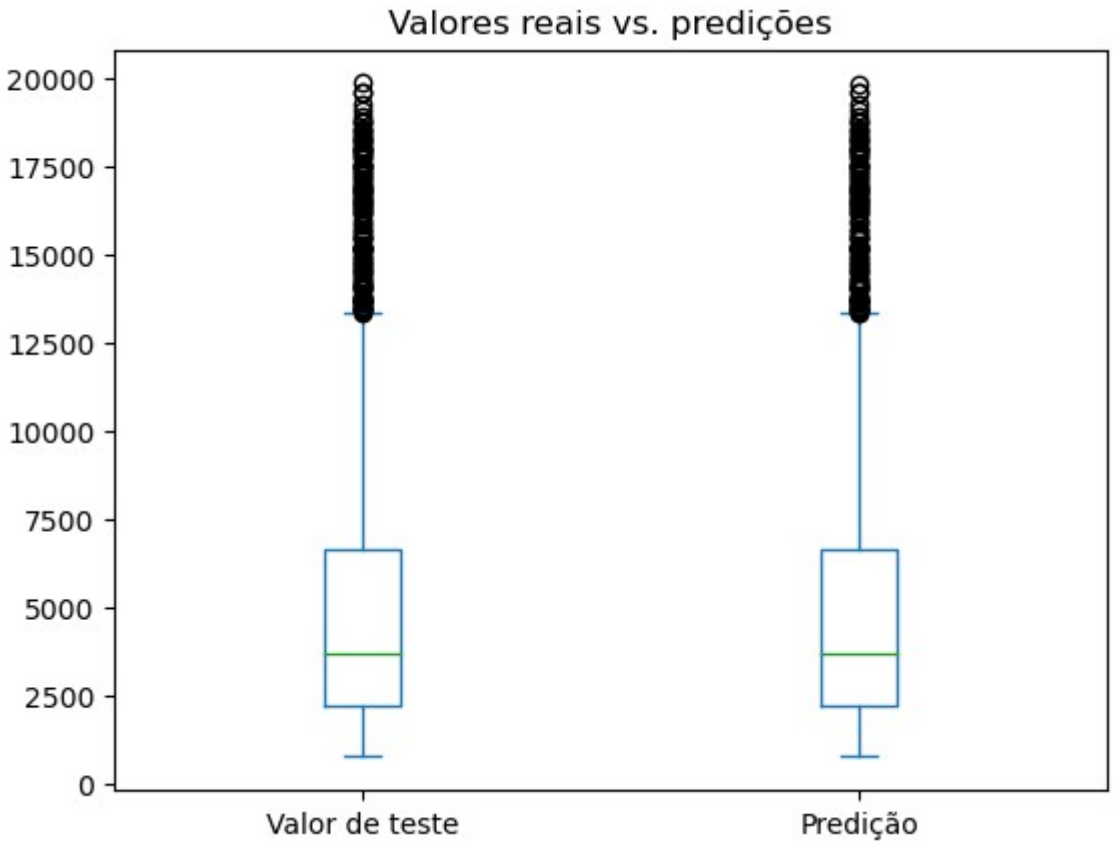
```
best_model = df_results.iloc[0]

model = models[best_model[0]]['model']
model.fit(X_train,y_train)

pred = model.predict(X_test)

pred_s = pd.Series(pred)
y_test_s = y_test.reset_index(drop=True)

df_result = pd.concat([y_test_s,round(pred_s,0)], axis = 1)
df_result.columns = ['Valor de teste', 'Predição']
df_result = df_result.astype({'Predição': 'int64'})

diff = df_result[df_result['Valor de teste'] != df_result['Predição']].size
print(f'Valores com diferença: {diff}/{df_result.size} ({((diff/df_result.size) * 100).round(2)}%)')

df_result
```

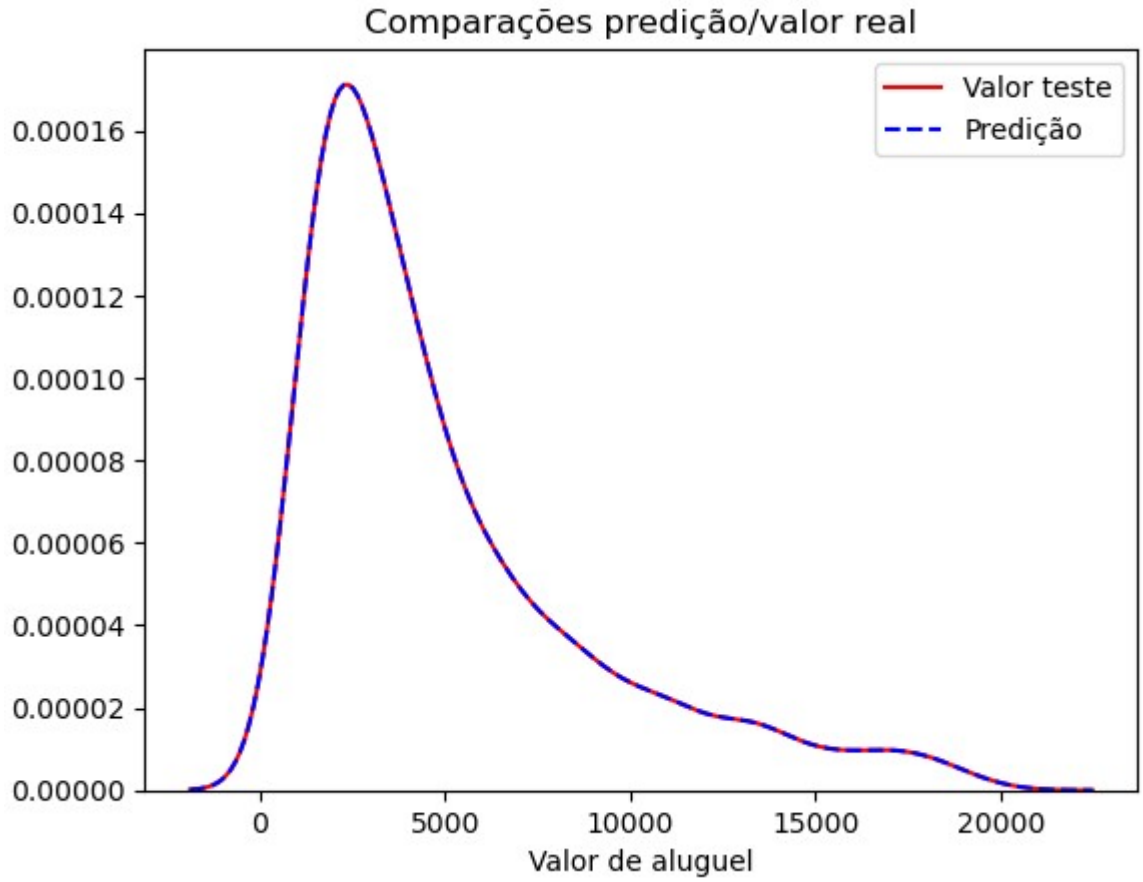
Valores com diferença: 730/4050 (18.02%)

	Valor de teste	Predição
0	2553	2553
1	7774	7775
2	1174	1174
3	2557	2557
4	4684	4684
...	...	...
2020	5372	5372
2021	1602	1602
2022	5370	5370
2023	6144	6144
2024	2979	2979

2025 rows × 2 columns

Podemos ver que temos menos de mil predições que não foram iguais aos valores esperados na base de teste, esse resultado a princípio parece satisfatório, mas podemos ter uma melhor visualização utilizando gráficos:





Como podemos observar nos gráficos, temos box plots muito similares e linhas no gráfico de linhas sobrepondo-se quase que inteiramente, podemos concluir que chegamos em um modelo de alta performance e acurácia (por volta de 99%), e que nos servirá bem para a resolução do problema proposto no primeiro relatório.

4. Conclusão

Abordamos a aplicação de técnicas de Inteligência Artificial na previsão de preços de aluguel imobiliário. Durante a pesquisa, destacamos a importância da preparação adequada dos dados, incluindo a reintrodução de colunas previamente omitidas para melhorar a acurácia dos modelos de aprendizado de máquina.

Ao aplicar diversos modelos de aprendizado de máquina e avaliar seu desempenho através das métricas escolhidas (tempo de treinamento, *RMSE* e *R2 score*), identificamos o modelo de regressão linear como o mais promissor para a resolução do problema proposto, demonstrando alta performance e acurácia.

Este estudo ressalta a importância da Inteligência Artificial na área imobiliária, oferecendo uma ferramenta valiosa para prever preços de aluguel com precisão. Os

resultados obtidos fornecem insights significativos para profissionais do setor imobiliário e pesquisadores interessados no tema, mas, mais importante, os resultados contém informações importantes para o consumidor final. Este estudo também demonstra que as ferramentas necessárias para se chegar neste tipo resultado são democráticas, estão ao alcance de todos e úteis para todas as faixas da sociedade.

Em resumo, este estudo representa um passo importante na utilização da Inteligência Artificial no setor imobiliário e ressalta seu potencial para oferecer soluções inovadoras e precisas.

5. Links

- Repositório deste relatório: https://github.com/lgcavalheiro/pucminas-projeto-integrado
- Fonte do dataset: https://www.kaggle.com/datasets/0fc2c2957155f98e380a0e5e7db219aff29962b37b13e6ac5a569389cfe26e83?select=houses_to_rent_v2.csv