Trabalho Inteligência artificial

Tiago Lopes , 8180341

17 de junho de 2024

Conteúdo

Capítulo 1

1.1 Resumo

Este projeto tem como tema principal o treino de modelos com Machine Learning abordados na disciplina de inteligência artificial dos quais os modelos escolhidos foram RandomForest [3], DecisionTree [1], GradientBoosting [2], todos os modelos são treinados com *scikit learn*. Para o treino do modelo foi utilizado um dataset sobre o preço de casas, foi feita uma comparação ente os diferentes modelos na sua eficácia nas previsões. Foi construido uma aplicação web onde os utilizadores podem interagir, escolher diferentes modelos e receber uma previsão sobre os dados inseridos.A aplicação web foi construido atráves das frameworks fastapi(backend) e angular(frontend).

1.2 Introdução

Ao longo deste documento desenvolvido no âmbito da disciplina de inteligência artificial é descrito os processos de treino de 3 modelos diferentes e do desenvolvimento da aplicação web para os utilizadores interagirem com os mesmos já treinados. Numa primeira fase é descrito o tratamento dos dados para o treino, de seguida os modelos são treinados com as várias configurações e por ultimo são comparados os resultados dos vários modelos.

1.2.1 Arquitetura da aplicação

Para o utilizador conseguir interagir com os modelos treinados foi criado uma aplicação web, para o seu desenvolvimento foi utilizado as frameworks FastApi e o Angular.

Para a o backend esta a ser utilizado o FastApi que é a api que fornece os pedidos para o frontend desenvolvido com a framework angular. Para guardar as previsões feitas pelo utilizar é usado a base de dados *mongoDB*.

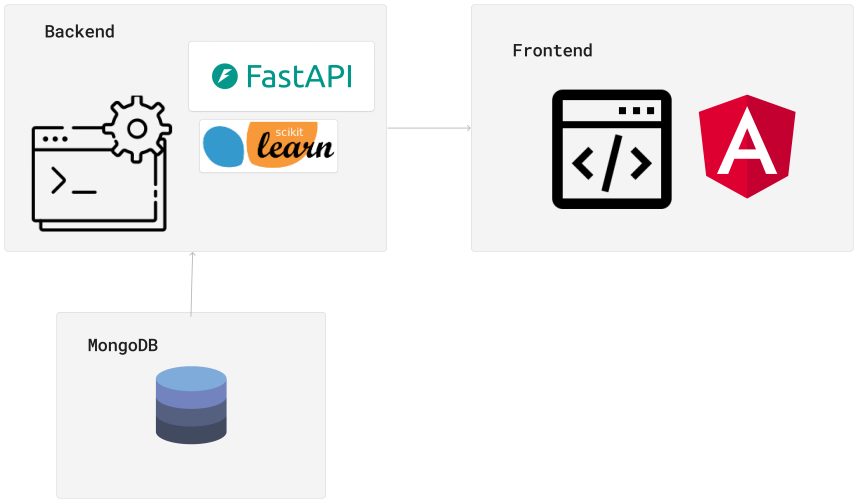


Figura 1.1: Arquitetura

1.3 Dataset

Para treinar os modelos foi escolhido um dataset no *website Kagle* sobre o preço de varias casas.

O conjunto de dados inclui preços de venda de casas nos EUA. As casas foram vendidas no período de maio de 2014 a maio de 2015.

Colunas:

- **id**: id para uma casa
- **date**: Data de venda da casa
- **price**: O preço é o alvo da previsão
- **bedrooms**: Número de quartos na casa
- **bathrooms**: Número de casas de banho
- **sqft_living**: area da casa
- **sqft_lot**: area do terreno
- **floors**: Nº de andares
- **waterfront**: Vista para o mar
- **view**: Se foi vista
- **condition**: Qualidade da casa geral
- **grade**: Grau geral dado a cada, com base no sistema de classificação dos EUA
- **sqft_above**: area da casa sem porão
- **sqft_basement**: area da porão
- **yr_built**: Ano de construção
- **yr_renovated**: Ano em que a casa foi renovada
- **zipcode**: código postal
- **lat**: latitude
- **long**: longitude
- **sqft_living15**: Área em 2015 (implica algumas renovações)
- **sqft_lot15**: Área do terreno em 2015 (implica algumas renovações)

`https://www.kaggle.com/datasets/soylevbeytullah/house-prices-dataset`

1.4 Tratamento dos dados

Antes de começar a se treinar os modelos primeiro é necessário fazer o tratamento do *dataset*, o que inclui remover dados desnecessário, adicionar novas variáveis e converter alguma variável para ser um valor numérico caso seja necessário.

Após remover todas as colunas do *dataset* menos relevantes converti as areas para metros quadrados.

```
[21613 rows x 8 columns]
         price  bedrooms  bathrooms  living_m2      lot_m2  grade      lat     long
0     221900.0         3       1.00  109.62554  524.901950      7  47.5112 -122.257
1     538000.0         3       2.25  238.76071  672.803526      7  47.7210 -122.319
2     180000.0         2       1.00   71.53531  929.030000      6  47.7379 -122.233
3     604000.0         4       3.00  182.08988  464.515000      7  47.5208 -122.393
4     510000.0         3       2.00  156.07704  750.656240      8  47.6168 -122.045
...        ...       ...        ...        ...         ...    ...      ...      ...
21608 360000.0         3       2.50  142.14159  105.073293      8  47.6993 -122.346
21609 400000.0         4       2.50  214.60593  540.045139      8  47.5107 -122.362
21610 402101.0         2       0.75   94.76106  125.419050      7  47.5944 -122.299
21611 400000.0         3       2.50  148.64480  221.852364      8  47.5345 -122.069
21612 325000.0         2       0.75   94.76106   99.963628      7  47.5941 -122.299
```

Figura 1.2: Dados brutos

Frame para o treino do modelo:

```
[21613 rows x 8 columns]
--------------------------------------------------
       bedrooms  bathrooms  living_m2      lot_m2  grade      lat     long
0             3       1.00  109.62554  524.901950      7  47.5112 -122.257
1             3       2.25  238.76071  672.803526      7  47.7210 -122.319
2             2       1.00   71.53531  929.030000      6  47.7379 -122.233
3             4       3.00  182.08988  464.515000      7  47.5208 -122.393
4             3       2.00  156.07704  750.656240      8  47.6168 -122.045
...         ...        ...        ...         ...    ...      ...      ...
21608         3       2.50  142.14159  105.073293      8  47.6993 -122.346
21609         4       2.50  214.60593  540.045139      8  47.5107 -122.362
21610         2       0.75   94.76106  125.419050      7  47.5944 -122.299
21611         3       2.50  148.64480  221.852364      8  47.5345 -122.069
21612         2       0.75   94.76106   99.963628      7  47.5941 -122.299
```

Figura 1.3: Dados de treino

Depois de os dados estarem prontos para o treino do modelo o frame dos dados foi separado em dois frames, frame de treino que representa 80 por cento

dos dados e o frame de teste que representa os restantes 20 por cento.

O frame de treino é utilizado para treinar o modelo e o frame de teste é para validar e comparar o valor previsto com o valor real de cada previsão.O próximo passo é configurar os modelos e enviar os frames para realizar o treino.

1.5 Treino dos modelos

Para o treino dos modelos foi utilizado modelos de regressão ideias para prever dados de valores contínuos e identificar relações entre variáveis como por exemplo, preços ,vendas, numero de clientes entre outros. Podemos encontrar esta informação na documentação do [4]

Modelos utilizados:

- **Random Forest Regressor**
- **Decision Tree Regressor**
- **Gradient Boosting Regressor**

Estes modelos enquadram-se com o dataset escolhido já que estamos a prever o preço das casas.

1.5.1 Random Forest Regressor

O modelo Random Forest Regressor utiliza várias árvores de decisão para fazer as suas previsões,dividindo as árvores em diferentes sub amostras do *dataset* e utiliza a média para melhorar a precisão da previsão e controlar o overfitting.

Numa primeira fase o modelo foi treinado como os parâmetro pré-definidos do scikit learn, que obteve previsões bastante razoáveis.O score,que é o coeficiente de determinação foi **0.8433995172992348** que é um resultado razoável.

A melhor pontuação possível para o coeficiente de determinação é **1**.E pode ser negativo (porque o modelo pode ser pior). Um modelo constante que sempre prevê o valor esperado de y, independentemente das características de entrada, obteria uma pontuação de **0**.

Parâmetro	Valor	Explicação
max_depth	None	Profundidade máxima das árvores.
max_features	1.0	Número de recursos a serem considerados para a melhor divisão.
max_leaf_nodes	None	Número máximo de nós folha.
max_samples	None	Número máximo de amostras
min_samples_leaf	1	Número mínimo de amostras que um nó folha deve ter.
min_samples_split	2	Número mínimo de amostras necessárias para dividir um nó.
n_estimators	100	Número de árvores.

Tabela 1.1: Parâmetros do RandomForestRegressor

- Descrições dos parâmetros:
  - **max_depth=None**: Permite que as árvores cresçam até que todas as folhas sejam puras ou tenham menos de `min_samples_split` amostras.
  - **max_features=1.0**: Utiliza todos os recursos para considerar a melhor divisão em cada nó.
  - **max_leaf_nodes=None**: Não limita o número de nós folha.
  - **max_samples=None**: Usa todas as amostras disponíveis para treinar cada árvore.
  - **min_samples_leaf=1**: Cada nó folha deve ter pelo menos 1 amostra.
  - **min_samples_split=2**: Cada divisão de nó requer pelo menos 2 amostras.
  - **n_estimators=100**: Constrói 100 árvores.

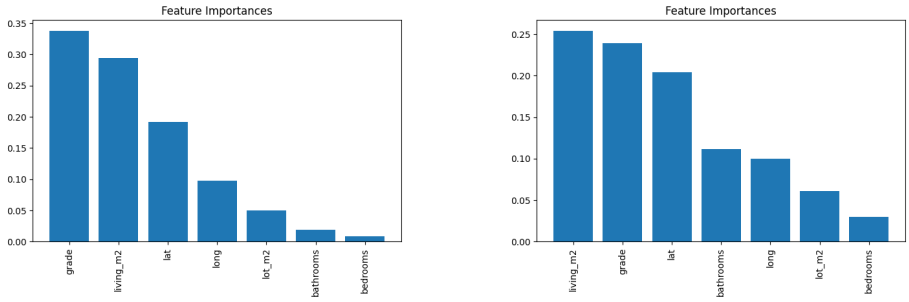
Numa segunda fase no treino do modelo as configurações do mesmo foram alteradas para tentar obter previsões mais precisas para tal foi utilizado os seguintes parâmetro:

Parâmetro	Valor
max depth	20
max features	log2
min samples leaf	1
min samples split	5
n estimators	500
bootstrap	False

Tabela 1.2: Legenda da tabela

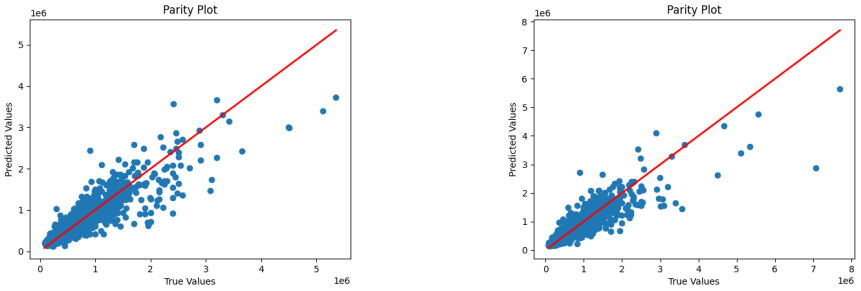
Foi aumentado o numero de de árvores para tornar o modelo mais robusto,limitou-se a profundidade de cada árvore para evitar overfitting.

Comparação



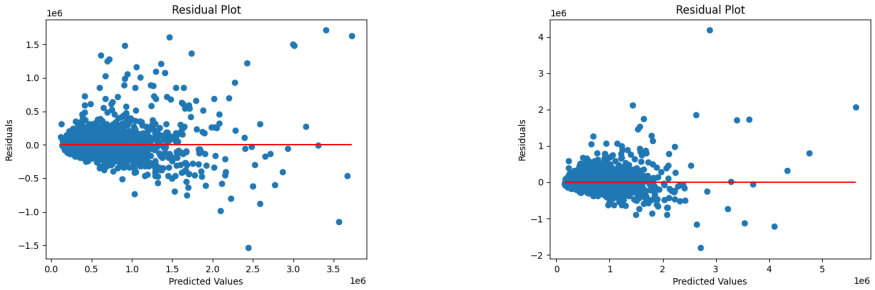
Default Tuning

Figura 1.4: Variáveis mais importantes



Default Tuning

Figura 1.5: Comparação entre valores reais e previstos



Default Tuning

Figura 1.6: Resíduos nos valores previsto

Modelo	Valor previsto	Valor real
RandomForestRegressor(Default)	232,979.5€	221900.0€
RandomForestRegressor(Tuning)	225,555.59€	221900.0€

Ao analisar os resultados deste modelo obtivemos uma previsão bastante próxima do valor real.Dos 3 modelos este é o que tem melhor desempenho.

1.5.2 Decision Tree Regressor

O modelo de Decision Tree Regressor cria uma estrutura de árvore de decisão, onde cada nó interno representa uma característica do dataset, cada ramo representa uma decisão baseada nessa característica, e cada folha representa um valor numérico da previsão.

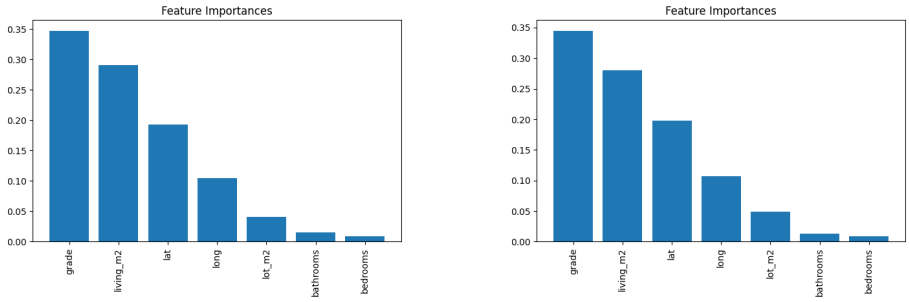
Com os parâmetros pré-definidos do scikit learn obtivemos um score de **0.6887864786944242**. Para tentar melhorar o modelo as configurações foram alteradas para os seguintes parâmetros:

Parâmetro	Valor
Criterion	squared_error
Splitter	best
Max_depth	None
Min_samples_split	2
Min_samples_leaf	1
Min_weight_fraction_leaf	0.0
Max_features	None
Random_state	42
Max_leaf_nodes	None
Min_impurity_decrease	0.0

Tabela 1.3: Parâmetros do modelo de árvore de decisão

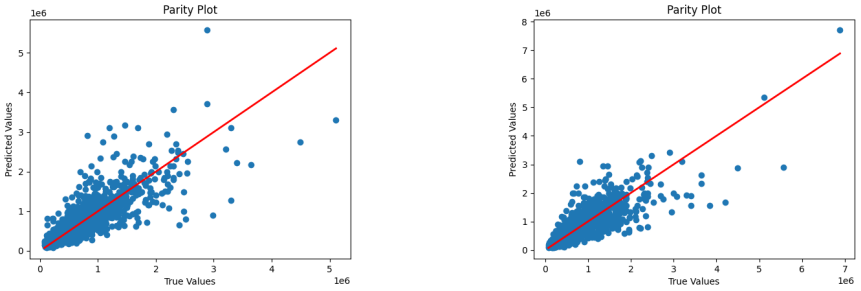
O score obtido teve uma pequena melhoria , **0.7111943250955093**

Comparação



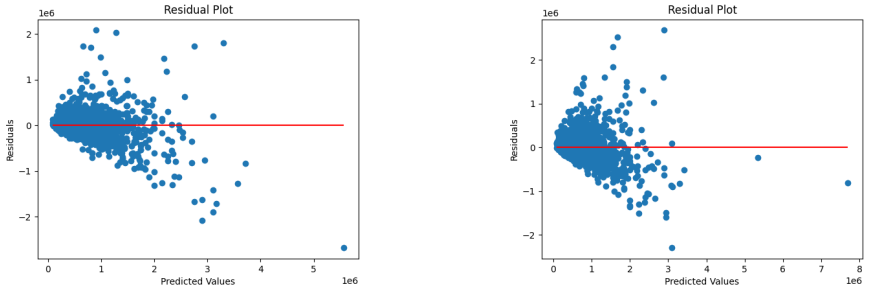
Default Tuning

Figura 1.7: Variáveis mais importantes



Default Tuning

Figura 1.8: Comparação entre valores reais e previstos



Default Tuning

Figura 1.9: Resíduos nos valores previsto

Modelo	Valor previsto	Valor real
DecisionTreeRegressor(Default)	221900.0€	221900.0€
DecisionTreeRegressor(Tuning)	221900.0€	221900.0€

Neste tipo de modelos os dados usados no treino ficam "gravados"e caso o input de dados seja o mesmo o valor da previsão vai ser igual ao valor real.Não fica muito claro se o modelo esta a prever bem ou não.

1.5.3 Gradient Boosting Regressor

O *Gradient Boosting Regressor* constrói varias árvores de decisão de forma sequencial, ajustando-as para minimizar os resíduos do modelo anterior. Ele é amplamente utilizado devido à sua capacidade de lidar com uma variedade de problemas de regressão, é indicado para dados complexos.

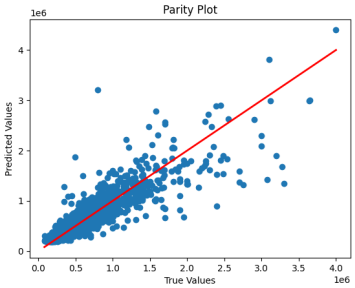
Na primeira fase de treino do modelo *Gradient Boosting Regressor* com os dados pré-definidos obtivemos um score de **0.8245403342159418**.

Parâmetro	Valor
Learning_rate	0.05
Max_depth	None
Min_samples_leaf	5
L2_regularization	0.1
Max_leaf_nodes	31
Max_bins	255

Tabela 1.4: Parâmetros

Com estes parâmetros obteve-se um score de **0.8701319137320694**

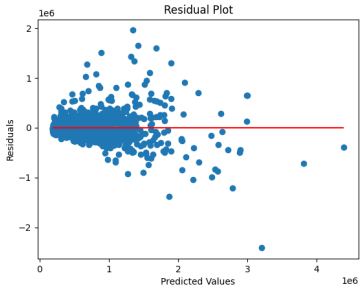
Comparação



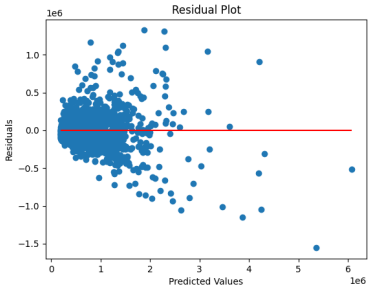
Default

Tuning

Figura 1.10: Comparação entre valores reais e previstos



Default



Tuning

Figura 1.11: Resíduos nos valores previsto

Modelo	**Valor previsto**	**Valor real**
GradientBoostingRegressor(Default)	282,515.47€	221900.0€
GradientBoostingRegressor(Tuning)	284,267.17€	221900.0€

Apesar deste modelo ter um score relativamente alto as previsões tem bastante diferença com o valor real.Se analisarmos os gráficos de comparação e de resíduos vemos que ambos os modelos tem muito *outliers* o que influencia na previsão do modelo.

1.6 Aplicação web

Para o utilizador interagir com os modelos treinados foi desenvolvido uma pequena aplicação web dividida em dois componentes principais, *backend* e*frontend*.

1.6.1 Backend(Api)

O *backend* oferece os seguintes pedidos:

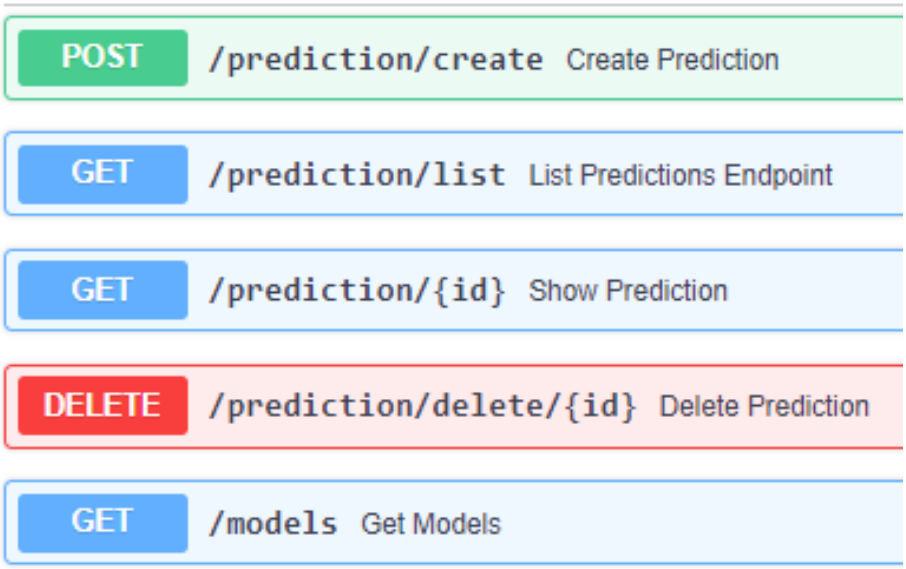


Figura 1.12: Api

1.6.2 Frontend

A parte interativa para o utilizador esta dividida em duas paginas, pagina principal e modelos

Página principal:

- Cria novas previsões
- Visualizar as previsões já feitas
- Eliminar previsões
- Selecionar vários modelos diferentes

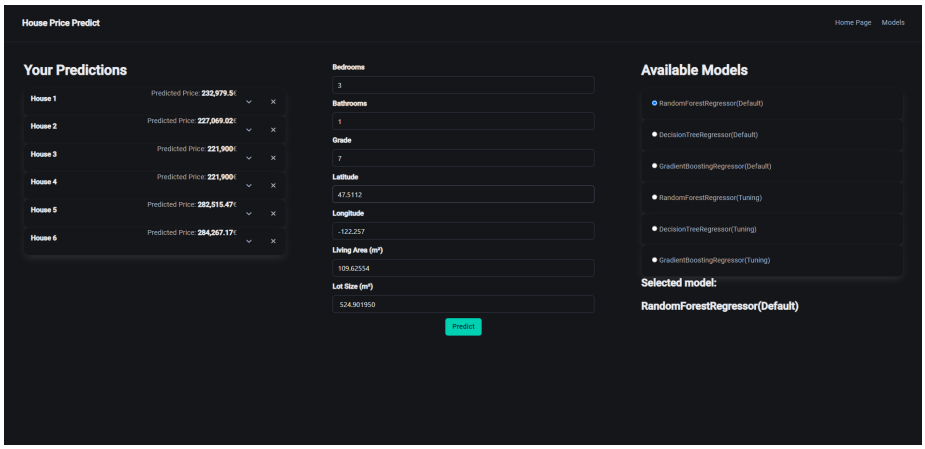


Figura 1.13: Página Principal

Modelos:

- Visualizar os gráficos dos modelos *default*
- Visualizar os gráficos com as diferentes configurações

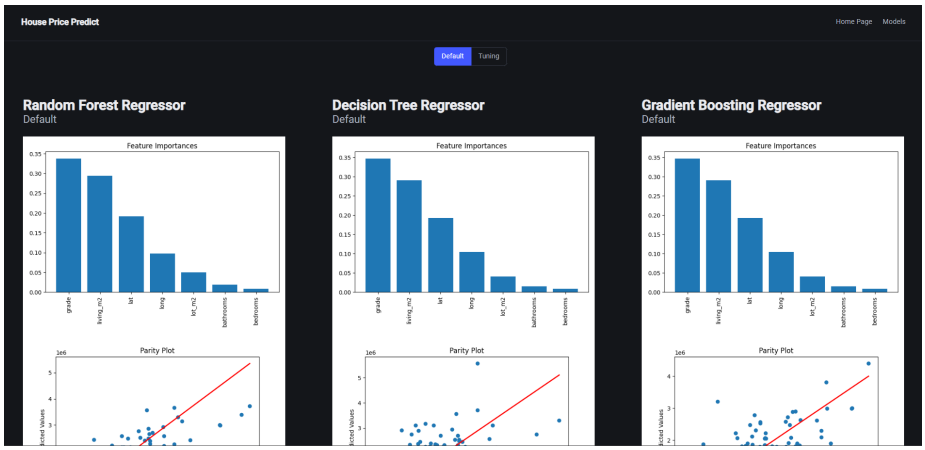


Figura 1.14: Gráficos

1.7 Conclusões e Trabalho Futuro

Com o desenvolvimento deste tipo de aplicações web e os treinos dos modelos foi possível criar uma ferramenta com utilidade para prever preços de casa.Após os resultado obtidos e os teste elaborados o modelo como melhor desempenho foi o **Random Forest Regressor**, o modelo apresenta previsões muito próximas aos valores reais.

Bibliografia

[1] Scikit-learn. Decision tree regressor, 2023.

[2] Scikit-learn. Gradient boosting regressor, 2023.

[3] Scikit-learn. Random forest regressor, 2023.

[4] Scikit-learn. Regresion, 2023.