ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Trabalho prático ML/NFML

Previsão de preços de casas através de modelos de Machine Learning

Machine Learning / Noções Fundamentais de Machine Learning

Mestrado em Engenharia Informática

2022/2023

Trabalho realizador por:

João Bragança – 8190555

José Fernandes – 8190239

Olavo Alves – 8180231

Pedro Afonso – 8090457

Professor Orientador

João Ricardo Martins Ramos

Conteúdo

Índice de figuras

1. Introdução

1.1. Contextualização

De forma a colocar em prática todos os conhecimentos teóricos adquiridos na Unidade Curricular de Machine Learning foi-nos proposto o seguinte trabalho, explorar um *dataset* para uma análise exploratória dos dados e proceder ao respetivo trabalho de limpeza (verificar valor omissos, *outliers*, converter dados no caso de necessidade, entre outros). É pretendida a aplicação de pelo menos um algoritmo de aprendizagem supervisionada e um algoritmo de aprendizagem não supervisionada com todas as suas implicações e justificações. Por fim, desenvolver uma interface de visualização, que permite ao utilizar inserir dados. Para isso, o modelo treinado recebe esses inputs, processa-os, e retorna o output.

Tendo em conta esta proposta, o nosso grupo procedeu à escolha do tema, sendo este, baseado no dataset fornecido pelo docente sobre uma lista de preços de venda de casas. Esta escolha é justificada, visto que o tema é claro para todos os elementos do grupo e é possível identificar os processos inerentes ao negócio para iniciar a análise exploratória.

Trabalhamos este *dataset* de forma a conseguir prever o preço de uma casa com base nas suas características (número de camas, número de casas de banho, hectares, estado e tamanho da casa).

1.2. Ferramentas utilizadas

Foram utilizadas as ferramentas, PyCharm e DataSpell para editar e correr os scripts criados através de linguagem python.

Utilizamos a framework *Angular*, para criação do frontend da aplicação.

A framework *Flask* foi utilizada para criação do backend, que por sua vez, utiliza o modelo para prever o resultado.

O package *joblib* foi utilizado para guardar o modelo treinado em formato de ficheiro (formato.pkl) para que pudesse ser utilizado pelo backend.

Outros packages como, *pandas*, *numpy*, *matplotlib* e *sklearn* são considerados essenciais para a criação e treino do modelo e por isso foram utilizados ao longo do projeto.

2. Estado da arte

2.1. Machine Learning

Atualmente as aplicações de *Machine Learning* são cada vez mais uma solução utilizada tanto no meio empresarial, como no nosso dia a dia. Estas soluções permitem melhorar e aperfeiçoar tarefas que antes eram realizadas de forma manual. Com isto conseguimos ganhar tempo e rapidez nessas tarefas, uma vez que elas são cada vez mais automatizadas e eficientes.

Alguns exemplos disso são:

- Reconhecimento de imagem e vídeo;
- Analisar grandes volumes de dados em tempo real e identificar anomalias ou padrões incomuns;
- Utilizados para prever resultados futuros com base em dados históricos, classificar dados em categorias e identificar padrões em grandes conjuntos de dados;
- Automatizar tarefas repetitivas, como triagem de e-mails, classificação de documentos e deteção de fraudes;
- Área da Saúde, podemos utilizar para diagnóstico de doenças, previsão de riscos e tratamentos personalizados;
- Ameaças de segurança, detetar comportamentos suspeitos e prevenir fraude
- Na área do Marketing, pode ser utilizado para personalizar campanhas de marketing, prever o comportamento do consumidor e segmentar públicos-alvo com base em dados históricos.

Os algoritmos de *Machine Learning* podem ser classificados de acordo com a quantidade e o tipo de supervisão, neste contexto, existem quatro categorias principais de aprendizagem, sendo elas as seguintes: supervisionada, não supervisionada, semisupervisionada e aprendizagem por reforço. [1]

2.1.1. Aprendizagem Supervisionada

A aprendizagem supervisionada é uma técnica de *machine learning* em que podemos treinar um modelo com dados rotulados. Ou seja, iremos ter um conjunto de dados identificados com uma "etiqueta" que identifica a que classe eles pertencem. Neste tipo de aprendizagem, os modelos são treinados com o objetivo de adivinhar um determinado resultado. [2]

Existem vários métodos que podem ser aplicados, mas a maioria deles segue sempre as mesmas etapas, sendo estas:

- Preparação de dados: Os dados são divididos em conjunto de treino e conjunto de testes. O conjunto de treino serve para treinar o modelo e o de teste para o validar e avaliar.
- Seleção do modelo: Conforme o problema pode ser escolhido um modelo diferente como: árvores de decisão, redes neuronais, regressão linear e regressão logística.
- Treino de modelo: Com o conjunto de dados de treino o modelo é criado.
- Avaliação do modelo: O modelo é avaliado com o conjunto de dados de teste de forma a determinar a sua precisão e desempenho.

- Ajuste do modelo: Se o modelo não estiver de acordo com o pretendido é possível ajustar os parâmetros e repetir o processo de treino até que alcance um desempenho satisfatório.

Podem ser aplicados em várias situações, como por exemplo:

- Previsão de preços: A regressão linear pode ser usada para prever o preço de um bem de acordo com as suas características, como o preço de uma casa.
- Análise de sentimento: As redes neuronais podem ser usadas para classificar comentários positivos e negativos com base no comentário em si.
- Deteção de fraudes: As árvores de decisão podem ser usadas para detetar transações fraudulentas com base em dados históricos.

2.1.2. Aprendizagem Não Supervisionada

A aprendizagem não supervisionada é uma técnica de *machine learning* que tem como objetivo encontrar padrões de dados sem rótulos, ou seja, não temos nenhuma identificação dos dados. Estes padrões são relações entre os dados que os agrupam de forma a dar algumas introspeções úteis.[3]

Existem várias técnicas de aprendizagem não supervisionada, algumas delas são:

- Clustering: esta técnica é utilizada para agrupar dados com grupos com base nas suas semelhanças. Existem algumas abordagens para esta técnica como k-means, clustering hierárquico e DBSCAN. Estas técnicas podem ser usadas para segmentação de clientes, análise de mercado e deteção de anomalias.
- Dimension Reduction: consiste em reduzir a quantidade de variáveis num conjunto de dados mantendo a maior quantidade de informação possível. Pode ser útil em situações em que os dados são muito complexos, permitindo uma melhor visualização.
- Associação: esta técnica consiste em encontrar relações frequentes entre itens de um conjunto de dados. É normalmente utilizado em mineração de dados.

Algumas indústrias onde se pode aplicar este tipo de aprendizagem são as seguintes:

- Análise de texto: a dimension reduction e o clustering podem ser utilizados para agrupar documentos de texto em tópicos semelhantes, para poderem ser pesquisados.
- Deteção de erros: O clustering pode, por exemplo, ser utilizado para a deteção de anomalias de segurança informática.
- Identificar padrões de compras: A associação pode ser utilizada para identificar padrões de compras de clientes e recomendar produtos relacionados.

2.1.3. Aprendizagem por reforço

A aprendizagem por reforço é uma técnica de *machine learning* onde um agente aprende a realizar uma ou várias tarefas num ambiente através da interação com o mesmo. Conforme o seu comportamento ele recebe recompensas ou punições de forma que, em cada iteração ele se torne melhor a realizar a determinada tarefa.

Alguns exemplos da aprendizagem por reforço são:

- Robôs autônomos: a aprendizagem por reforço tem sido aplicada no desenvolvimento de robôs que são capazes de navegar por diversos ambientes, como edifícios ou florestas e completar uma série de tarefas.[4]
- Jogos arcada: a aprendizagem por reforço pode ser utilizada para ensinar um agente a jogar algum jogo, como por exemplo o Super Mario Bros.[5]

2.1.4. Aprendizagem semi supervisionada

A aprendizagem semi supervisionada consiste em combinar a aprendizagem supervisionada e a aprendizagem não supervisionada. Normalmente existem grandes quantidades de dados, mas apenas alguns estão rotulados. A ideia principal é que os dados rotulados providenciem alguma informação sobre a estrutura dos dados. Neste tipo de aprendizagem é normal dar rótulos aos dados não rotulados tendo em conta os dados rotulados e depois treinar um modelo sobre os dados resultantes.[6]

Alguns exemplos deste tipo de aprendizagem são:

- Identificação de imagens: Podemos ter um conjunto de imagens não rotuladas, então, rotulamos algumas delas e utilizados essas imagens para inferir o rótulo das outras imagens.
- Processamento de linguagem natural: Quando temos uma grande quantidade de documentos, muitas vezes poucos deles estão categorizados.

3. Limpeza de Dados

3.1. Análise Exploratória

Inicialmente, abrimos o *dataset* fornecido pelo orientador, em formato csv e observamos, de forma exploratória, as suas características. Possui 12 colunas:

- Status (string), indica se a casa se encontra à venda ou pronta a construir.
- Price (string), indica o preço da casa, em USD.
- bed (string), indica o número de camas.
- bath (float), indica o número de casas de banho.
- acre_lot (float), indica o espaço exterior da casa, em acres.
- full_address (string), indica o endereço completo.
- street (string), indica a rua.
- city (string), indica a cidade.
- state (string), indica o estado.
- zip_code (float), indica o código-postal
- house_size (string), indica o tamanho da casa, em pés quadrados (sqft)
- sold_date (string), indica a data em que a casa foi vendida.

É possível observar 923.159 registos no *dataset*. Existem registos com valores nulos, sendo a coluna "sold_date" a que mais valores nulos possui (466.763 registos nulos).

Através da análise de valores distintos, detetamos valores com erros na coluna "Status", onde vemos vamos "for_salee" e "for_ssale" em vez de "for_sale". Para corrigir este erro, aplicamos a função *replace* da biblioteca *pandas*.

De forma a diminuir o tamanho do *dataset*, optamos por eliminar as colunas, "sold_date", pois não pretendemos considerar esta variável; e as colunas "full_address" e "street" pois não pretendemos utilizar dados tão específicos para a localização.

Também foram eliminados registos duplicados, o que reduziu o tamanho de registos para 111,466.

3.2. Conversão de data types

Algumas colunas que se encontram no formato de texto (string) foram convertidas para formato numérico (float), através da função pd.to_numeric. Encontramos alguns registos que provocaram erro ao realizar esta conversão, consequentemente, optamos por eliminar esses registos, presentes nas linhas 732 e 1399.

A coluna "status" que se encontrava em formato de texto foi convertida em numérica, para reduzir o tamanho do *dataset*. Os valores "for_sale" e "ready_to_build" foram substituídos por "1" e "0", respectivamente. Após este passo, optamos por eliminar os registos que possuíam valor "0", ficando assim apenas com registos de casas já construídas no dataframe.

3.3. Remoção de Outliers (boxplot)

Através da análise de valores mínimos e máximos, e recorrendo a diagramas de caixa, podemos identificar a presença de outliers.

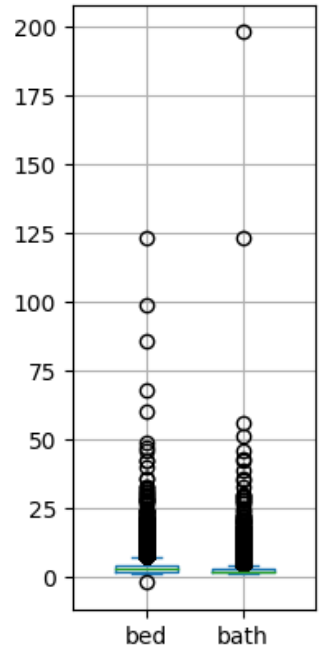


Figura 1 - Camas e quartos antes da remoção de outliers

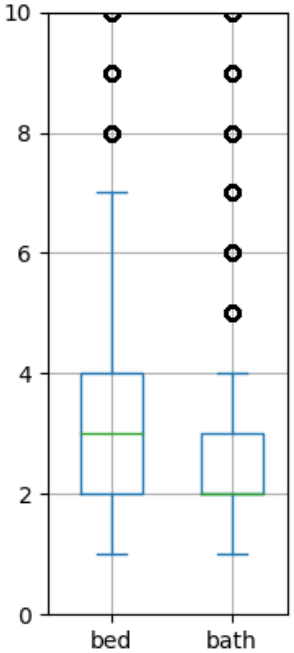


Figura 2 - Camas e quartos depois da remoção de outliers

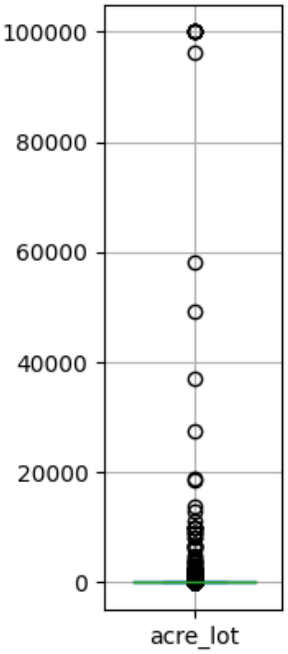


Figura 3 - hectares antes da remoção de outliers

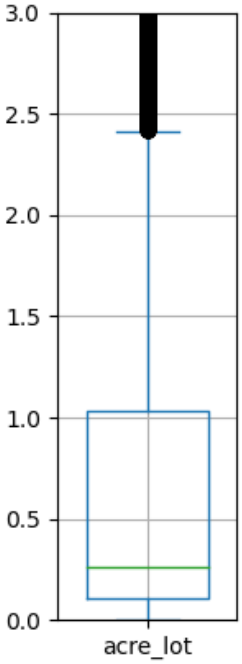


Figura 4 - hectares depois da remoção de outliers

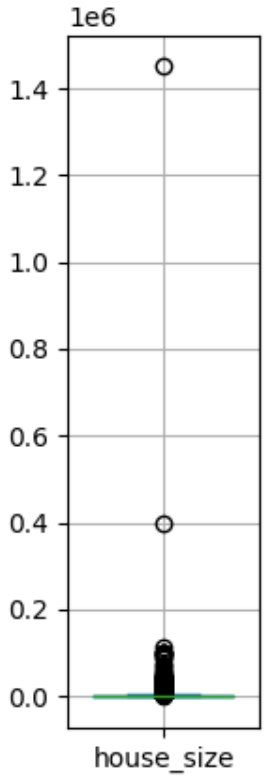


Figura 5 - Preço da casa antes da remoção de outliers

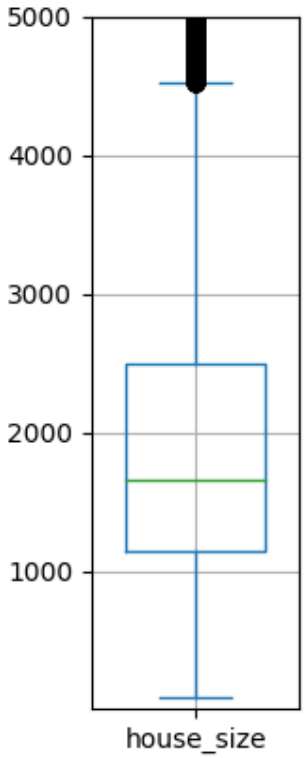


Figura 6 - Preço da casa depois da remoção de outliers

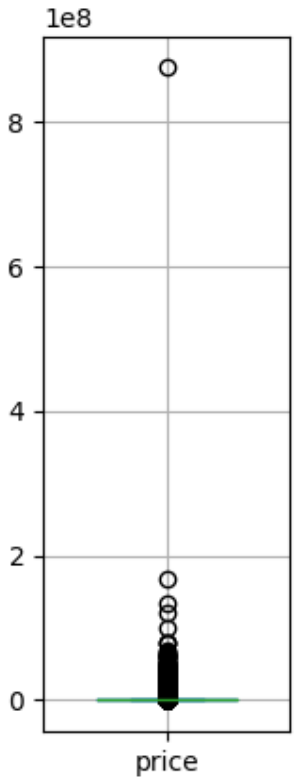


Figura 7 - Preço da casa antes da remoção de outliers

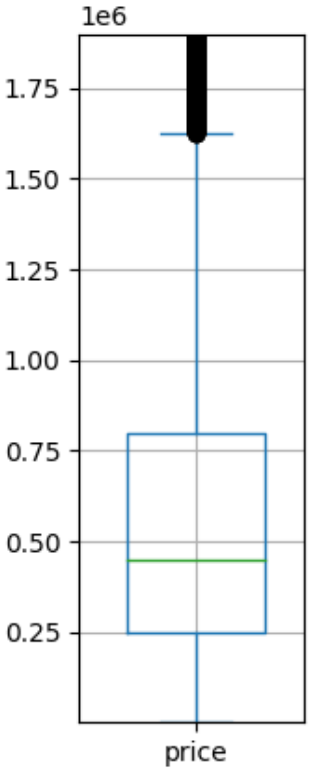


Figura 8 - Preço da casa depois da remoção de outliers

Através dos diagramas de caixa e de valores estatísticos, podemos observar alguns outliers. Foram implementadas as seguintes alterações para remover estes registos do dataframe:

- bed
- bath
- house_size
- price
- acre_lot

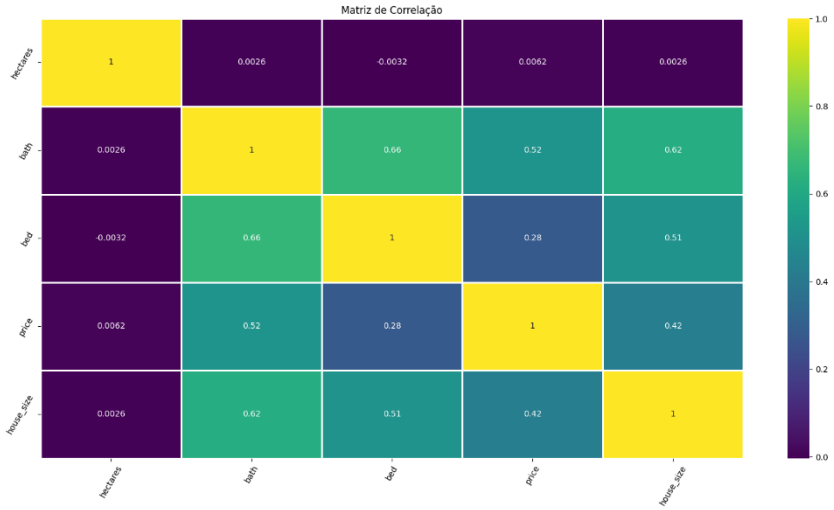
Apos estas alterações, o dataframe ficou com 111.182 registos espalhados por 9 colunas.

Observamos que ainda possuíamos demasiados valores nulos em house_size (35.218 registos). De forma a gerar dados para reduzir a quantidade de nulos nesta coluna, optamos pela criação de um modelo de regressão linear, e aplicamos (quando possível) os valores gerados pelo modelo no dataframe original, reduzindo o número de valores nulos. Este tópico é detalhado de forma aprofunda em 4.1.

3.4. Análise de colunas

De forma a melhorar os dados que serão posteriormente treinados, as colunas “status”, “zip_code” e “city” foram removidas, pois não acresentam valor aos dados para o nosso objetivo, estimar o valor de uma casa.

Observamos assim a matriz de correlação.



Através desta figura, podemos observar que o tamanho do terreno envolvente (acre_lot), pouco ou nada impacto as outras variáveis. Onde encontramos mais correlação foi entre as variáveis “bed” (número de quartos) e “bath” (número de casas de banho), 66%.

Como os valores do dataset se encontravam com notações americanas, convertemos os dados para notações europeias.

- Os registos da coluna “price”, foram multiplicados por 0,93, valor do dólar face ao euro a 13 de feveiro de 2023.

- Os registos da coluna “acre_lot”, foram multiplicados por 0,404, valor de conversão de acres para hectares.

- Os registos da coluna “house_size”, foram multiplicados por 0,09290304, valor de conversão de pés quadrados (square feet) para metros quadrados.

3.5. Outras anotações

O valor da coluna “acre_lot” foi renomeado para “hectares”.

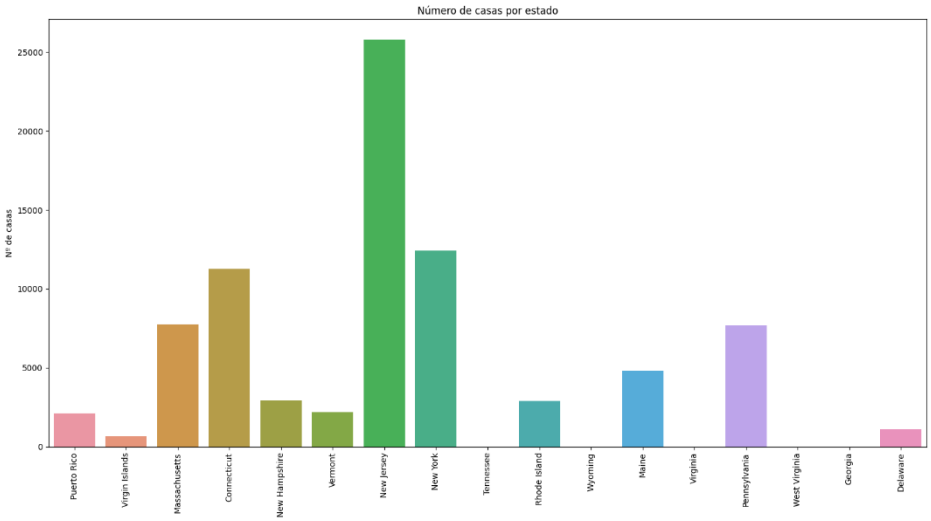


Figura 9 – Gráfico de barras (Número de casas por estado)

Através deste gráfico podemos observar a quantidade de casas presentes no dataset por estado. É notório a fraca presença dos estados “Tennessee”, “Wyoming”, “Virginia”, “West Virginia” e Georgia. Por este motivo, foram removidos do dataframe.

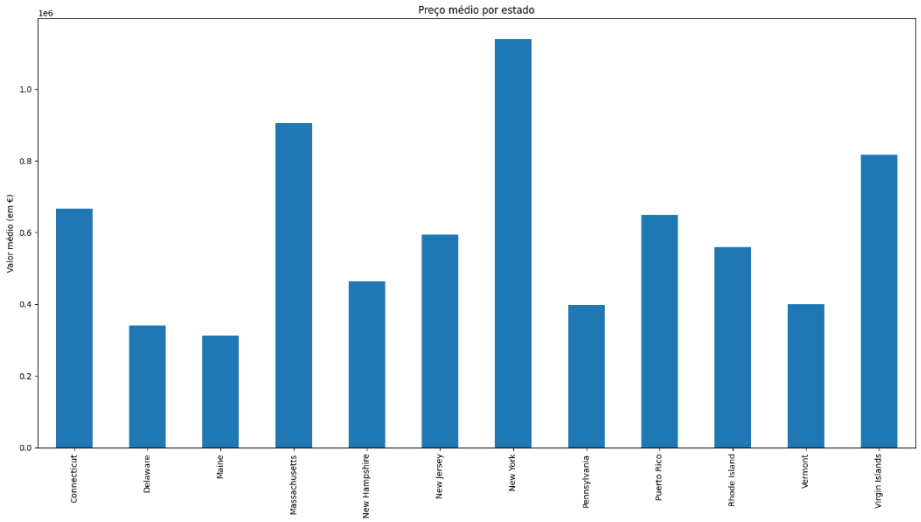


Figura 10 - Gráfico de barras (Preço médio por estado)

Através deste gráfico é visível o preço médio por estado. O valor médio mais elevado está presente no estado de New York. Já o valor médio mais baixo encontra-se no estado de Maine.

4. Criação de Modelos Supervisionados e Não Supervisionados

4.1. Criação de valores para house_size recorrendo a regressão linear

O modelo de regressão linear é usado para prever o valor de uma variável com base no valor de outras variáveis.

Para aplicar o modelo de regressão linear, é necessário remover todos os valores nulos do dataframe. Reduzimos assim o dataset para 53.190 registos.

Dividimos assim o dataset por colunas. Para as variáveis independentes (X_train) selecionamos as colunas “price“, “bed”, “bath”, “acre_lot”. Como variável dependente(y_train), selecionamos a coluna “house_size”, que é a variável que pretendemos prever mais tarde.

Chamamos o modelo de regressão linear, lr = LinearRegression(), Aplicamos a função fit, que é utilizada para aceitar inputs (X_train, y_train) e construir o modelo (output), com base nos dados fornecidos.

Não aplicamos dados de testes para criar métricas de analise (como *accuracy* ou *recall*), visto que treinamos o modelo com todos os valores possíveis do *subset*.

Criamos um *subset* com as colunas das variáveis independentes e onde os registos de “house_size” (várivavel dependente) são nulos. Aplicamos este subset ao modelo (através da função predict) e guardamos esse resultado. Depois aplicamos os valores previstos ao dataframe ,utilizado antes de proceder à criação do modelo. Utilizamos a função: “apply(lambda row: row.house_size if pd.notnull(row.house_size)

else row.y_pred, axis = 1)”, para percorrer a coluna “house_size”, caso o registo seja válido (não nulo), o registo é mantido. Caso seja um valor nulo na coluna “house_size” e a coluna “y_pred” possuir um registo, o valor é adicionado a coluna “house_size”.

4.2. Divisão do dataframe para treino e teste

Através da função train_test_split, que pertence a framework sklearn. Esta função divide o dataframe de forma aleatória, na percentagem definida no híper parâmetro ”test_size”, que no nosso caso, optamos por 75% dos dados para treino e 25% para teste. Achamos que esta divisão é acertada, considerando o extenso volume de dados. Desta forma, foram utilizados 49.698 registos para treinar os modelos e 16.566 registos para testar o modelo.

4.3. KNN (Supervisionada)

O modelo KNN (K-Nearest Neighbors), é um algoritmo utilizado em Machine Learning para aprendizagem supervisionada. O modelo calcula a distância entre os pontos e atribui um rótulo aos dados. Baseia-se no conceito de "semelhança", em que os pontos de dados mais próximos têm maior probabilidade de ter o mesmo rótulo.

Com este modelo obtemos um score de 35,15%. O erro quadrático médio (MSE) é de 1.858.593.764.044.

Mesmo após modificar o random_state na separação dos dados de treino e teste e modificar o valor n_neighbors da criação do modelo, não conseguimos obter melhores métricas de performance.

4.4. XGB Regressor (Supervisionada)

O modelo XGBoost Regressor é um algoritmo de Machine Learning, utilizado para modelagem preditiva e problemas de classificação. XGBoost significa eXtreme Gradient Boosting e é uma implementação de máquinas de aumento de gradiente. O XGBoost é um algoritmo de aumento eficiente e preciso usado em aprendizagem supervisionada, sendo que é baseado na noção de melhorar os dados que mais se afastaram do esperado.

Com este modelo obtemos um score de 74,46%. O erro quadrático médio (MSE) é de 1.271.570.301.020 (inferior ao do modelo anterior).

Decidimos avançar com este modelo para utilizar com os inputs do futuro GUI.

4.4.1 Exportação do modelo

Para exportação do modelo foi utilizada a biblioteca *joblib* uma vez que é mais eficiente no carregamento e tratamento de *arrays numpy* com grande volume de dados do que a biblioteca *pickle*. Para guardar o modelo basta executar o código que se encontra na linha 4 (Figura 11). Desta forma conseguimos armazenar o modelo treinado num único ficheiro e utilizá-lo para a efetuar previsão com base nos *inputs* fornecidos.

```
1   from sklearn.externals import joblib
2
3   # Save the model as a pickle in a file
4   joblib.dump(modelToSave, 'filename.pkl')
5
6   # Load the model from the file
7   knn_from_joblib = joblib.load('filename.pkl')
8
9   # Use the loaded model to make predictions
10  knn_from_joblib.predict(X_test)
```

Figura 11 - Código para exportação e importação de modelo

4.5. K-Means (Não Supervisionada)

De forma a proceder a implementação de um modelo de aprendizagem não supervisionada (*clustering*), utilizamos o modelo *K-Means*, que é um algoritmo que divide os dados em K *clusters*. É um processo iterativo que atribui aleatoriamente cada ponto a um dos K *clusters*. Em seguida, calcula-se a média dos pontos atribuídos, e cada ponto é atribuído ao cluster cujo centro (valor médio) está mais próximo. Este processo é repetido até que as atribuições parem de mudar. Finalmente, os clusters são rotulados de acordo com a média dos pontos.

Para o nosso caso implementamos 10 clusters, que surtiu na seguinte divisão:

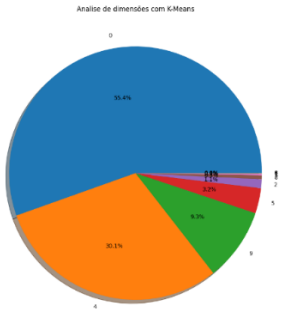


Figura 12 - percentagem de dados de cada cluster

5. Aplicação (Frontend e Backend)

Foi criada uma interface gráfica para o utilizador poder interagir com o modelo e fazer previsões de preços de cassa. Essa interface foi desenvolvida utilizando a *framework* Angular, que é uma plataforma de desenvolvimento para a criação de aplicações web com typscript, HTML e CSS.

A escolha desta *framework* prendeu-se pelo facto do Angular ser uma ferramenta poderosa e robusta, que oferece recursos avançados para a criação de interfaces gráficas interativas, incluindo a capacidade de trabalhar com componentes, diretivas e serviços, bem como gerir o estado da aplicação e se comunicar com APIs externas.

Com o Angular, foi possível criar umas interfaces gráficas responsivas, para isso também foi usada a framework bootstrap. Na imagem abaixo podemos ver a interface criada.

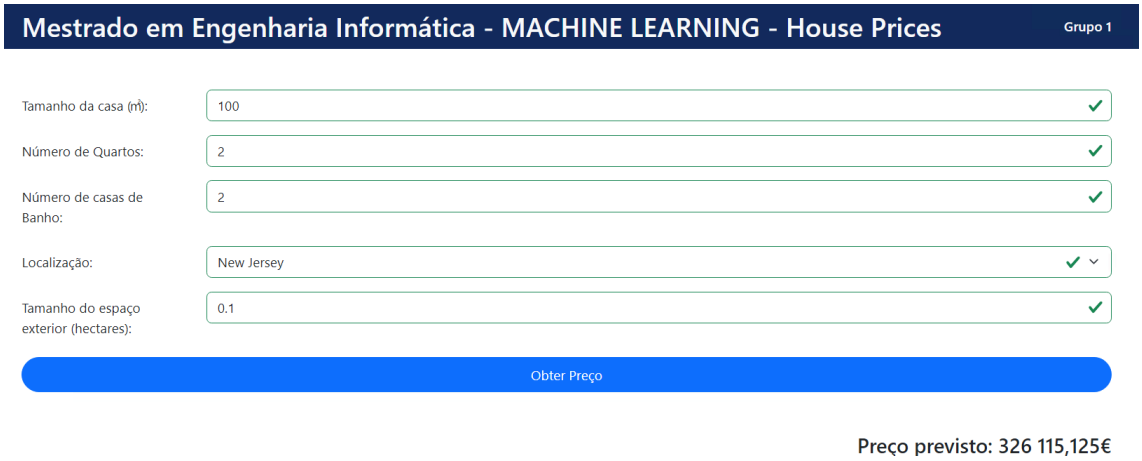


Figura 13 - Interface gráfica

O *backend* foi feito em Flask, que é uma *framework* que permite escrever em python uma API REST de forma simples, possibilitando a interação com o modelo criado. Apenas contém um endpoint que recebe os dados da casa representados na imagem acima e utiliza o modelo treinado para calcular o preço estimado da casa, devolvendo-o como resposta.

Na figura abaixo é possível ver como o Angular interage com o Flask de forma a pedir uma previsão para um preço de uma casa.

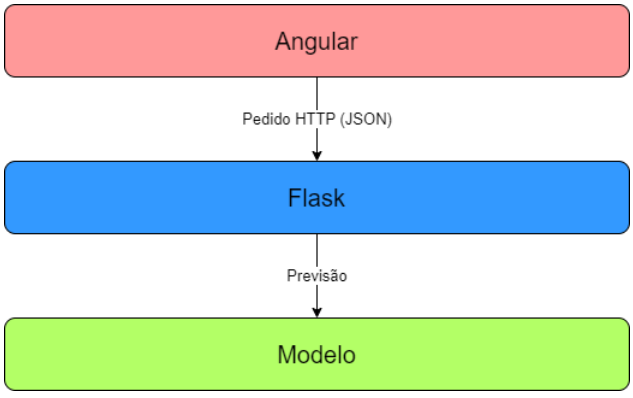


Figura 14 - Arquitetura Conceptual

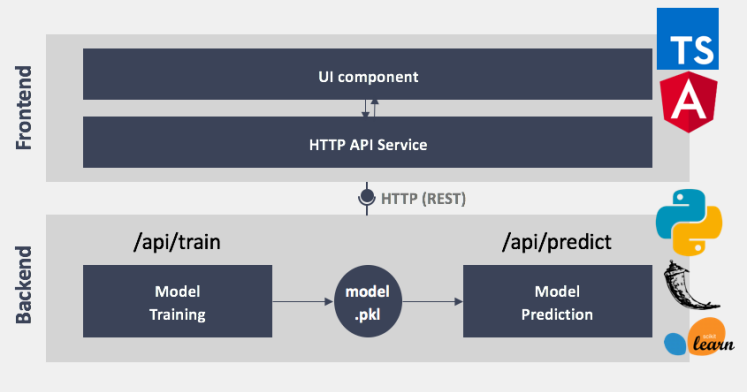


Figura 15 - Exemplo de arquitetura ideal

6. Resultados

Decidimos utilizar o algoritmo de aprendizagem não supervisionada, utilizamos para isso o XGB Regressor. Este Algoritmo teve um score de 74,46%. O erro quadrático médio (MSE) é de 1.271.570.301.020 muito inferior ao KNN.

7. Dificuldades e evolução do trabalho

Inicialmente foi difícil identificar quais os melhores algoritmos a utilizar para o projeto, visto que tivemos de fazer uma análise a um dataset desconhecido e existem vários algoritmos que podem ser aplicados cada um com resultados diferentes.

Também nos deparamos com um dataset com uma necessidade de limpeza, o que levou algum tempo a perceber como esta iria ser realizada.

Depois tivemos de perceber o impacto que cada atributo iria ter no nosso modelo. Tivemos algumas dificuldades também na conversão de datatypes e na passagem da coluna estado para numérico.

8. Conclusão

O trabalho descrito neste relatório baseou-se na análise de um conjunto de dados sobre os preços de imóveis.

Inicialmente procedeu-se a uma análise profunda do dataset, aqui verificou-se essencialmente o comportamento das variáveis, isto é, de que forma as variáveis influenciam na previsão do preço das habitações. Nesta análise deu-se especial atenção aos outliers existentes, uma vez que estes valores normalmente são os responsáveis pelos maus resultados dos modelos de previsão. A forma escolhida para contornar esta problemática passou por através da análise gráfica conseguir identificar esses valores, assim como os valores nulos e proceder à sua eliminação com a exeção do house_size, aqui foi utilizada uma modelo de regressão linear para substituir nulos.

Durante o projeto foram analisados e testados alguns algoritmos, como forma de perceber o que melhor se ajusta ao nosso caso. A escolha foi baseada no algoritmo que melhor resultado apresentou.

Posteriormente o último objetivo consistiu em permitir a acessibilidade ao modelo treinado, para isso, foi utilizado uma aplicação web em angular que permitiu fazer testes de imput e verificar o resultado do preço das habitações mediante determinadas características.

9. Bibliografia

[1] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. 2017.

[2] V. Nasteski, “An overview of the supervised machine learning methods,” *HORIZONS.B*, vol. 4, pp. 51–62, Dec. 2017, doi: 10.20544/horizons.b.04.1.17.p05.

[3] S. Abou El-Seoud, N. Farag, and G. McKee, “A review on non-supervised approaches for cyberbullying detection,” *International Journal of Engineering Pedagogy*, vol. 10, no. 4, pp. 25–34, 2020, doi: 10.3991/ijep.v10i4.14219.

[4] H. Nguyen and H. La, “Review of Deep Reinforcement Learning for Robot Manipulation,” in *Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019*, Mar. 2019, pp. 590–595. doi: 10.1109/IRC.2019.00120.

[5] T. Shu, J. Liu, and G. N. Yannakakis, “Experience-Driven PCG via Reinforcement Learning: A Super Mario Bros Study,” in *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2021, vol. 2021-August. doi: 10.1109/CoG52621.2021.9619124.

[6] Y. C A Padmanabha Reddy, P. Viswanath, and B. Eswara Reddy, “Semi-supervised learning: a brief review,” *International Journal of Engineering & Technology*, vol. 7, no. 1.8, p. 81, Feb. 2018, doi: 10.14419/ijet.v7i1.8.9977.