UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PESQUISA

PROGRAMA DE INICIAÇÃO CIENTÍFICA VOLUNTÁRIA – PICVOL

Técnicas de interpretabilidade de modelos na classificação de elementos arquitetônico em modelos BIM

Relatório Final

Período da bolsa: de Setembro/2022 a Agosto/2023

Este projeto é desenvolvido com bolsa de iniciação científica
PICVOL

SUMÁRIO

1. Introdução

É notório que o uso de inteligência artificial está em exponente ascensão nos mais diversos campos de mercado e vista a inevitável abrangência de seu uso, se torna crucial cada vez mais pesquisadores do tema para os mais diversos fins. Além disso, todos eles têm uma característica em comum, todas precisam de grande bases de dados e bom fundamento teórico para evoluir.

Levando em conta essa abrangência citada anteriormente, sabemos que a IA se divide em diversos ramos de pesquisa e um deles é a aprendizagem de máquina. Nesse projeto iremos tratar unicamente do aprendizado Supervisionado para problemas de Classificação, ou seja, ensinar o modelo a, por meio de padrões, classificar componentes, no nosso caso, elementos arquitetônicos no modelo BIM. O uso de aprendizado supervisionado em BIM é interessante pois é possível extrair padrões de altura, volume, entre outros padrões, que permitem o uso para detectar elementos fora do padrão ou que apresentem atributos faltantes.

No contexto do uso de AM em projetos BIM, tem-se como base (Bonsang Koo , Byungjin Shin, 2018), que traz uma pesquisa sobre a identificação de elementos BIM, e um trabalho de conclusão de curso, baseado no projeto citado anteriormente, que traz as bases de dados utilizadas nesta pesquisa e aplica diversas técnicas de aprendizagem de máquina e as compara, utilizando a técnica de K-FOLD (Eike Natan Sousa Brito,2020).

A partir dessa base de trabalhos e de dados em mãos, foi pensado em fortalecer a parte do enriquecimento semântico, que consiste em, além de treinar o modelo e observar os resultados, seria possível entender o porquê daquele modelo dar tal resultado e entender quais os parâmetros que levam o modelo a por exemplo, dizer que uma porta é uma porta a partir de dados numéricos e que caminho ele levou para dar essa resposta

2. Objetivos

Objetivo geral

"Explorar e aplicar modelos simbólicos na previsão de elementos arquitetônicos em projetos BIM"

Os objetivos específicos do trabalho são:

- Estudar sobre a área de Aprendizagem de máquina
- Desenvolver modelos de Aprendizagem supervisionada
- Realizar experimentos analisando resultados de modelos
- Aprender a utilizar bibliotecas específicas de inteligência artificial em python
- Consultar portais de artigos científicos para embasamento teórico

3. Metodologia

Para a realização desse projeto, foram necessárias diversas atividades que, juntas, compõem o escopo do projeto e o que foi apresentado. Inicialmente, foi bastante importante e necessária a formação de uma base de conhecimento sobre o assunto tratado, desde a parte de BIM até toda a fundamentação teórica e prática sobre Aprendizagem de máquina por meio de cursos online e leitura de artigos científicos no tema.

Após essa etapa, partimos para a parte de explorar as bibliotecas na linguagem Python que dão suporte a o que foi estudado e o que foi planejado a ser aplicado, também analisando o que foi realizado por (Eike Natan,2020). A partir disso foi possível estudar e selecionar as ferramentas a serem utilizadas no decorrer de todo o projeto. Em consequência desse processo, houve uma fase focada em aplicar esse modelos e bibliotecas para fixar o conhecimento e analisar a viabilidade na base de dados original. Para isso, foram utilizados os jupyter notebooks para fazer a organização do projeto e facilitar a compreensão.

Posteriormente, utilizamos os modelos prontos e aplicamos na base de dados real para dar prosseguimento no projeto como um todo. Com isso, foi possível extrair diversas informações relevantes, principalmente métricas de desempenho do modelo de AM. A partir dessas informações e dos modelos estáveis e funcionais, houve um foco total na parte dos experimentos. Avaliando em condições controladas o desempenho dos modelos construídos a partir das bases de dados apresentadas.

3.1 Fundamentação teórica

Como citado anteriormente, para realizar os experimentos, utilizamos aprendizagem de máquina, que consiste em utilizar imensas bases de dados para fazer com que o modelo "aprenda" a interpretar novos exemplos e identificar tais elementos com base em padrões criados pelo processo de aprendizagem. Esse aprendizado é dividido essencialmente em 3 tipos: Supervisionado, não-supervisionado e semi-supervisionado. E como citado, vamos utilizar o aprendizado supervisionado para o tratamento dos dados e treinamento dos modelos. As bases de dados são divididas em treino e teste, nesse caso, 70% de treino e 30% de teste. E a partir da base de treino, aplicamos o algoritmo de treinamento, nesse caso, na maior parte do tempo foi utilizado o algoritmo de Árvore de decisão.

Além desse método, é de interesse testar todas as partes da base de dados como teste e treino, a fim de obter o melhor resultado possível, por isso, além do método padrão de divisão 70/30 comentado anteriormente, a divisão da base de dados em 10 partes iguais, usando uma delas como teste e o resto como treino, é conhecida como K-Fold, que é um método de validação cruzada dos dados, esse método, mesmo que mais lento, por rodar o código n vezes, sendo n o número de divisões, ele geralmente nos traz resultados mais sólidos e com menos sobreajuste de dados.

O algoritmo de árvore de decisão utiliza um modelo de decisões binárias que impõe, a cada nó, uma condição em relação a algum elemento da base de dados, gerando uma representação gráfica e lógica que envolve diversos pontos de decisão e que geram no fim, uma classificação que são representadas pelas folhas dessa árvore. A partir do resultado gerado por esse algoritmo na base de dados em questão, é possível salvar o estado do modelo treinado, armazenando a lógica em um arquivo que pode ser reutilizado em outros trechos de código, evitando assim a execução constante.

Contudo, ao expandir a gama de elementos que podem ser classificados, foi encontrado um empecilho, especificamente na parte de atributos faltantes, ou seja, linhas da base de dados em que algum dado não existia ou não convinha no contexto. Para isso, fez-se necessária a padronização de valores para o preenchimento dos campos, possibilitando o prosseguimento do modelo.

Se tratando do formato dos dados, contamos com uma base relativamente completa, com mais de 5000 exemplos de elementos para treino e teste. Esses dados foram inicialmente retirados do aplicativo Revit no padrão IFC e seguindo o modelo BIM ,dividido em 9

projetos e foi aplicado um algoritmo para retirar os dados necessários para a base final, gerando um arquivo CSV, que facilita a compreensão pelo Python.

Sobre as métricas de interesse nesse projeto, são consideradas as mais relevantes para analisar os modelos a acurácia e o F1-Score, que medem, respectivamente, a quantidade de acertos em relação à quantidade de testes realizados e a análise combinada da acurácia e o recall, que por sua vez, mede a quantidade de elementos classificados em cada classe, em relação a quantidade real de elementos daquela classe na base de teste.

3.1.2 Trabalhos relacionados

- BIM and machine learning in seismic damage prediction for non-structural exterior infill walls, que trabalho no contexto de previsão de danos de terremotos ou acidentes sísmicos em paredes com infiltrações utilizando técnicas de Classificação com árvores de decisão, SVM e Random Forest

- Integration of Building Information Modeling and Machine Learning for Railway Defect Localization, esse que utiliza técnicas de Regressão para prever a localização de defeitos em ferrovias utilizando Redes neurais

- Machine learning for optimized buildings morphosis. Breve artigo sobre o uso de ML para classificar formas de construções, ainda assim utilizando ML e BIM

- Classification of Building Information Model (BIM) Structures with Deep Learning, um artigo mais geral que traz conceitos importantes e atualizados em formato de resumo de como utilizar Deep learning na classificação BIM

- Automatic classification of wall and door BIM element subtypes using 3D geometric deep neural networks. Esse trabalho traz conceitos mais complexos sobre classificação e redes neurais aplicados em modelos 3D, além do que já era visto anteriormente, esse artigo trouxe novas visões sobre o tema, principalmente ao aplicar geometria.

- A Machine Learning-Based Approach for BIM Object Localization. Esse artigo traz uma abordagem próxima à utilizada neste trabalho, pois utiliza enriquecimento semântico

para localizar elementos BIM através de um modelo de ML

- Applications of machine learning to BIM: A systematic literature review, mais um artigo geral de passagem de conhecimento sobre as aplicações de ML em BIM, apresentando vários modelos e como cada algoritmo pode ajudar

3.2 Desenvolvimento

Para compreensão do desenvolvimento do projeto, é importante como o ambiente de execução é configurado e suas devidas dependências são instaladas. Antes de mais nada, todos os códigos são desenvolvidos em python (3.9), alterando apenas o formato para a utilização de python notebooks, para facilitar a compreensão e realizar a execução guiada dos códigos. Em python, precisamos das bibliotecas focadas em análise gráfica, aplicação de modelos de aprendizagem supervisionada e análise de dados. Para isso, são utilizadas basicamente as bibliotecas: Pandas, sci-kit learn, numpy e matplotLib. Juntos, já conseguimos executar basicamente todos os testes e treinos do projeto. Além disso, como citado anteriormente, é importante o conhecimento e utilização dos jupyter notebooks.

O desenvolvimento do código relativo a esse projeto pode ser dividido em 3 partes. A primeira é a parte de tratamento dos dados. A base de dados é constituída essencialmente de um arquivo csv com 11 colunas, cada coluna representando um atributo do elemento especificado, com exceção da última, que indica sua classe no Revit, essa base é tratada para que os dados faltantes, que eventualmente atrapalham no treino do modelo, pudessem ser ignorados e dando prosseguimento para a segunda parte. Para fazer o tratamento da base, é usado um código que extrai corretamente do IFC, caso necessário. Nesse caso, existem 9 arquivos IFC, que são convertidos para 9 csv e juntos em uma só base completa. Segue o modelo que mapeia os arquivos csv separados e os junta em um só.

```python
def junta_csv():

    folder_path = './bd/'

    dfs = []
    # Juntando o conteudo dos aquivos CSV
    for filename in os.listdir(folder_path):
        if filename.endswith('.csv'):
            filepath = os.path.join(folder_path, filename)
            df = pd.read_csv(filepath)
            dfs.append(df)

    # Concatene todos os dataframes na lista em um único dataframe
    merged_df = pd.concat(dfs, ignore_index=True)

    # Salve o dataframe combinado em um arquivo CSV
    merged_df.to_csv('./base_completa.csv', index=False)
```

Com os dados devidamente tratados e a base devidamente ajustada. A segunda parte consiste em utilizar essa base para treino do modelo. Inicialmente, esse treino era feito diretamente na base de dados, separando 70% para treino e 30% para teste, aleatoriamente. Esse método, ainda que efetivo, realizava o treino apenas uma vez, que trazia apenas um resultado, e dependendo da porção utilizada para treino, tanto a acurácia, quanto outras métricas importantes, eram significativamente afetadas. Para resolver esse problema, foi aplicado o método K-Fold. Que separa 10 bases de teste diferentes e testa com outras 10 bases de treino, realizando uma validação cruzada. A partir desses 10 casos, utilizamos o caso com maior acurácia e o fazemos nosso modelo oficial, gerando assim, um arquivo de formato joblib, que salva o modelo para importações e utilizações futuras. Segue o exemplo, em código, como é treinada a árvore de decisão, salvo o modelo Joblib e plotada a figura de árvore, que mostra os caminhos de interesse da análise simbólica.

```python
clf = tree.DecisionTreeClassifier()
clf.fit(X,y)
dump(clf,"model.joblib")
plt.figure(figsize=(30,30))
tree.plot_tree(clf, filled= True, fontsize=6, class_names=values)
```

A terceira parte do projeto consiste em extrair informações do modelo gerado na segunda etapa. Utilizando novamente o arquivo joblib, é possível agora realizar testes por meio de métodos em python, que já reconhecem o modelo e teste a partir do que foi desenvolvido.

Além disso, esse arquivo foi utilizado para dar a ideia de enriquecimento semântico a partir da árvore gerada a partir de uma função que tenta imprimir o caminho da árvore a partir das decisões tomadas e quais foram os critérios utilizados. Por exemplo:

```
[Node: 0] Espessura ⩽ 0.052 (Valor: 0.025)
[Node: 1] Espessura ⩽ 0.046 (Valor: 0.025)
[Node: 2] PosY > -2.444 (Valor: 0.677)
[Node: 8] PosY ⩽ 6.381 (Valor: 0.677)
[Node: 9] Espessura > -2.0 (Valor: 0.025)
```

Além disso, foram utilizados também, além dos testes únicos, algumas bases de teste para verificar a acurácia e F1-Score, por exemplo.

Corrim�o	1.00	1.00	1.00	2
Elemento	0.00	0.00	0.00	3
Janela	0.88	0.88	0.88	8
Laje	0.75	1.00	0.86	3
Parede	0.77	1.00	0.87	10
Porta	1.00	0.80	0.89	5
accuracy			0.84	31
macro avg	0.73	0.78	0.75	31
weighted avg	0.77	0.84	0.80	31

Para a parte final do projeto, os modelos foram melhores incorporados com novas funções e a aplicação do K-Fold e cross validation e trazendo novas funções para realizar os experimentos que serão apresentados posteriormente. Nesse momento, além da técnica de Árvores de decisão, o modelo de Random forest também foi estudado e implementado, porém, esse não foi estudado tão profundamente quanto as AD, ainda que sua implementação tenha sido igual ao outro modelo. Com a organização e a aplicação de ambos os modelos para os casos dos experimentos, a parte do desenvolvimento foi concluída.

```
def kfoldSolution(X,y):

    for i in range(1,30):

        cv = KFold(n_splits=10, shuffle=False, random_state=None)
        modelDT = tree.DecisionTreeClassifier()
        modelRF = ensemble.RandomForestClassifier()
        y_pred = cross_val_predict(modelDT, X, y, cv=10)
        conf_mat = confusion_matrix(y, y_pred)

        new_row = []

        cross_DT = cross_validate(modelDT, X, y.values.ravel(), cv=10, scoring=['accuracy','f1_macro'])
        cross_RF = cross_validate(modelRF, X, y.values.ravel(), cv=10, scoring=['accuracy','f1_macro'])
```

Por fim, sobre a quantidade e organização dos arquivos de códigos, houve uma boa modularização dos códigos, tendo 3 arquivos apenas de funções e mais 3 de teste e execução de experimentos, além da pasta de dados, onde estão o arquivo csv e o arquivo de integração.

4. Resultados e discussões

A partir das funcionalidades apresentadas na parte de desenvolvimento, foram realizados alguns experimentos específicos para simular os melhores resultados para de fato testar o modelo. Como mostrado também anteriormente, usamos a técnica de K-Fold para treinar o modelo usando validação cruzada. Porém, para garantir o melhor resultado, rodamos o código de K-Fold 30 vezes com a mesma seed (fator de aleatoriedade), isso foi feito para que os diferentes algoritmos sejam avaliados na mesma base de dados em todas as suas partições. A partir disso, teremos o melhor resultado de 30 execuções. Além disso, utilizamos não só o método de árvores de decisão, mas também o de Random Forest, que é igualmente utilizado para classificação.

Foram realizados dois experimentos. O primeiro realiza o método de validação cruzada com 10 divisões em uma base de dados com 5030 elementos em 30 execuções. A partir dessa execução serão analisados o F1-Score e a acurácia do modelo. Já no segundo, é usada uma base de 265 linhas de 6 projetos. Nesse caso é realizado o treino de uma árvore de decisão e realizado um teste com 40 elementos, analisando as mesmas métricas, e depois, analisando o comportamento dos gráficos de acurácia e f1-Score de 30 execuções e comparando os resultados.

Utilizando a validação cruzada, utilizamos as mesmas regras de execução e treino para DT e

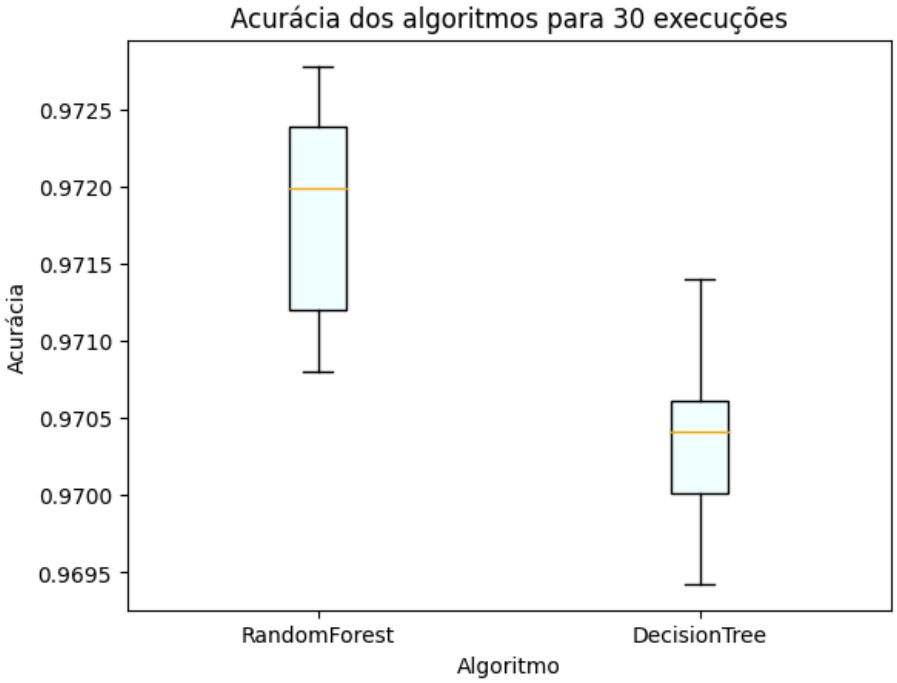
RF, analisamos cada resultado, caso a caso e armazenamos em um vetor todas as execuções, a fim de encontrar o resultado com melhor acurácia, o código abaixo faz a validação cruzada e nos retorna dois dados importantes: Além da acurácia, o F1-Score. Esses são armazenados em um vetor e no final, selecionamos o melhor modelo para comparar e utilizar. Na primeira imagem, temos o código que origina os dados, na segunda, a tabela com esses dados.

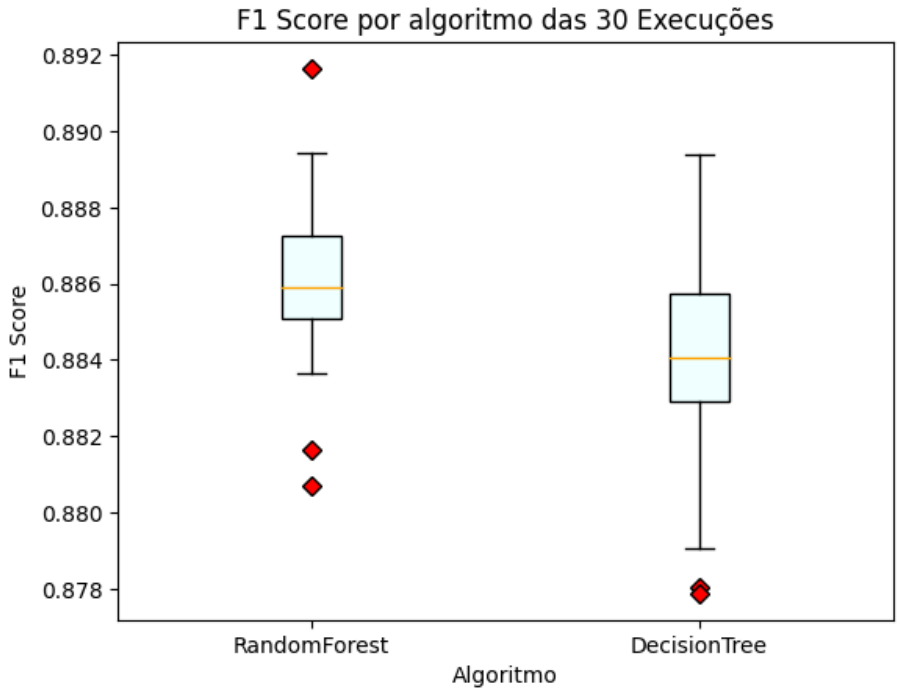
```
def kfoldSolution(X,y):

    for i in range(1,30):

        cv = KFold(n_splits=10, shuffle=False, random_state=None)
        modelDT = tree.DecisionTreeClassifier()
        modelRF = ensemble.RandomForestClassifier()
        y_pred = cross_val_predict(modelDT, X, y, cv=10)
        conf_mat = confusion_matrix(y, y_pred)

        new_row = []

        cross_DT = cross_validate(modelDT, X, y.values.ravel(), cv=10, scoring=['accuracy','f1_macro'])
        cross_RF = cross_validate(modelRF, X, y.values.ravel(), cv=10, scoring=['accuracy','f1_macro'])

        new_row = {'Execução': i, 'Acurácia_RF': cross_RF['test_accuracy'].mean(), 'Acurácia_DT': cross_DT['test_accuracy'].mean() , 'Desvio_RF': st.pstdev(cross_RF['test_a
        df_data.append(new_row)


    df_newData = pd.DataFrame(df_data)
    display(df_newData.head(10))
    return df_newData
```

	Execução	Acurácia_RF	Acurácia_DT	Desvio_RF	Desvio_DT	F1_RF	F1_DT
0	1	0.972580	0.970805	0.017383	0.016161	0.888184	0.884902
1	2	0.971200	0.970411	0.017644	0.016965	0.885778	0.882750
2	3	0.971989	0.970411	0.016613	0.016965	0.887436	0.887091
3	4	0.971397	0.970608	0.017401	0.016767	0.887292	0.884828
4	5	0.971989	0.970411	0.017143	0.016021	0.885984	0.886447
5	6	0.971200	0.970608	0.017863	0.016081	0.886435	0.886720
6	7	0.971200	0.970805	0.017489	0.016089	0.888593	0.881800
7	8	0.972580	0.969819	0.017248	0.017284	0.888670	0.880233
8	9	0.972383	0.969622	0.016688	0.017174	0.887703	0.878662
9	10	0.971200	0.970608	0.016002	0.016721	0.881806	0.884888

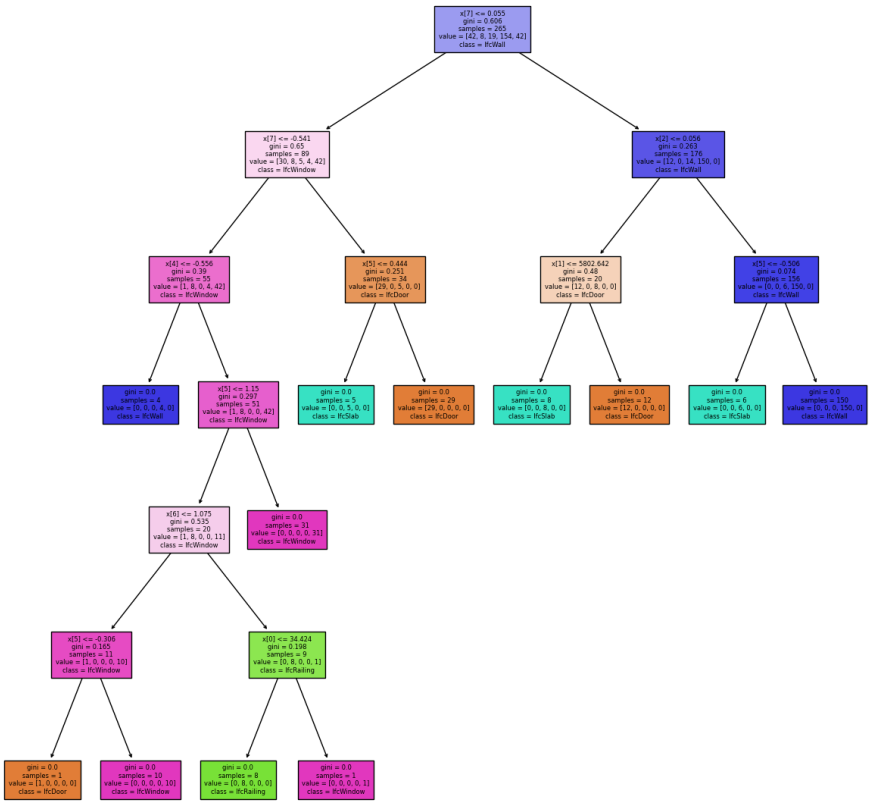
Utilizando esses dados, podemos aplicar o primeiro experimento, que é, utilizando a base de dados inicial, analisar a média dos resultados de acurácia e F1-Score baseado na tabela de resultados. A obtenção desses valores é muito importante para analisar a viabilidade da aplicação em contextos reais e perceber onde o modelo está acertando e errando, sempre abrindo a possibilidade de optimização e melhora. Isso pode servir, em pequena e grande escala, para analisar a eficácia de um modelo em relação a outro em um caso específico.

Nesse caso, ao realizar os experimentos e trazendo um boxplot, é possível fazer uma análise mais detalhada. Então, a partir do que é apresentado nos gráficos, é notável uma vantagem do método de Random Forest em relação ao de árvores de decisão, ainda que em alguns casos o desempenho seja equivalente, e que a partir desses altos valores apresentados, é possível aplicar os modelos criados em um ambiente real, considerando, a partir dos gráficos, a partir de uma base de 100 elementos, seria possível classificar 97 com precisão. Sobre o F1-Score, ele traz menores porcentagens, mas isso se deve a que como há objetos com dados faltantes, a classificação de algumas classes ficou comprometida, o que influencia no Recall e, consequentemente, no F1-Score.





Com o apoio dos pesquisadores de Engenharia Civil envolvidos no projeto, foi possível realizar um segundo experimento com uma nova base de dados com os dados de 6 novos pequenos projetos. Para a análise, foram unidas todas as bases e efetuado o mesmo procedimento de treino, tendo assim uma árvore de decisão, que com essa base é possível ver melhor e mais detalhadamente.

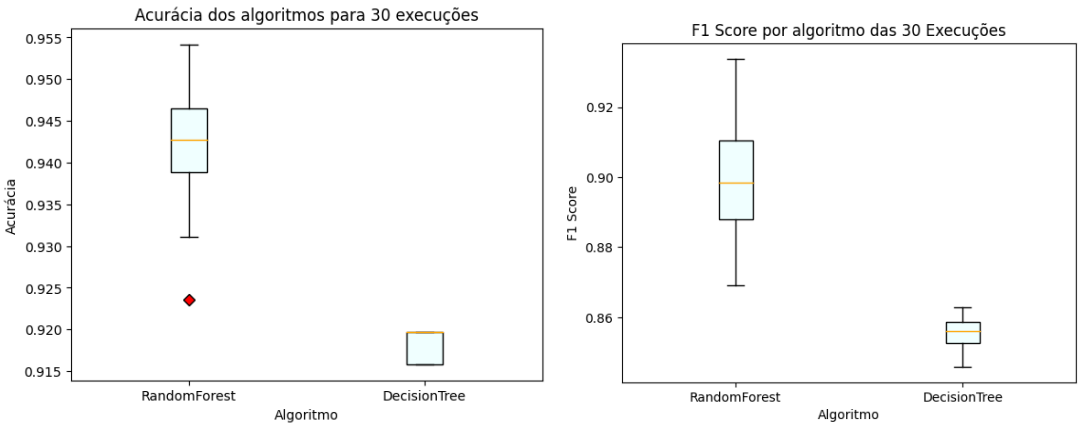


Cada nó da árvore representa um ponto de decisão e cada cor representa um elemento da classificação, onde:

- Laranja: porta
- Roxo: parede
- Rosa escuro: Janela
- Verde: Corrimão
- Azul: Laje

Além dessa base de treino, foi utilizada uma base de testes personalizada, produzida pelos pesquisadores do DEC, que, analisando a base de dados inicial, a analisaram e propuseram uma base que reflete melhor a realidade da área. Esse experimento tem como objetivo verificar o desempenho do modelo com novas bases de dados, gerando novos gráficos e trazendo novas métricas. Para esse novo caso, foi feita uma comparação de resultado entre o K-Fold na base de dados original e o uso da base de teste pronta.

	precision	recall	f1-score	support
Corrim□o	1.00	1.00	1.00	2
Elemento	0.00	0.00	0.00	3
Janela	0.88	0.88	0.88	8
Laje	0.75	1.00	0.86	3
Parede	0.77	1.00	0.87	10
Porta	1.00	0.80	0.89	5
accuracy			0.84	31
macro avg	0.73	0.78	0.75	31
weighted avg	0.77	0.84	0.80	31



A partir desse segundo experimento é possível notar que a técnica de validação cruzada, por encontrar diversas respostas, naturalmente consegue atingir melhores resultados. Ainda é possível perceber, em comparação ao experimento 1, que como a variação de árvores de decisão geradas é menor, a variação entre as 30 execuções também é muito baixa.

Além disso, é notável que como os dados da segunda base de dados são mais variados e com valores um pouco mais destoantes, é normal a queda de desempenho, que juntamente com a baixa quantidade de dados, pode gerar um sub-ajuste de dados.

A partir do que foi analisado no primeiro experimento, podemos extrair que, para a base de dados inicial, o algoritmo de Random Forest se sobressai em ambas métricas representadas, ainda que em alguns casos se iguale ao de árvores de decisão em relação ao F1-Score. Em casos como esses seria interessante também analisar o desempenho em função da acurácia, pois, com pouca diferença como essa, talvez um código mais rápido possa ser um diferencial para um algoritmo de classificação em tempo real, por exemplo.

5. Conclusões

Por fim, a partir do que foi apresentado e produzido, desde a separação das bases de dados, estudo dos atributos e das tecnologias de aprendizagem de máquina até realizar os experimentos e análise e comparação desses, é possível concluir que o uso dessa tecnologia é bastante apropriada e útil para a área de análise de modelos arquitetônicos, principalmente considerando os parâmetros que são utilizados para a análise e sua facilidade de integração com o python, e consequentemente, com os modelos de IA.

Apesar da facilidade da integração, é importante citar a dificuldade em tratar dados faltantes e ver como esses podem influenciar no sucesso do modelo, necessitando consequentemente, de uma base maior ou valores para preencher tais campos.

Enfim, por meio desse trabalho, foi possível, dados os resultados obtidos, entender o funcionamento de algoritmos de classificação, em especial, o de árvore de decisão. Compreendendo além do que é retornado na linha de comando, analisando simbolicamente as decisões tomadas e tendo a possibilidade de comparar com outros métodos e criar modelos de classificação altamente reutilizáveis e com boa precisão. Além disso, utilizar conceitos de enriquecimento semântico para tornar os dados ainda mais compreensíveis a qualquer usuário que deseje entender o modelo.

6. Perspectivas de futuros trabalhos

Mesmo com todo o trabalho realizado, ainda é possível encontrar novas utilidades e expandir a utilização dos resultados obtidos. Em primeiro lugar, esse projeto ainda está limitado a 5 tipos de elementos arquitetônicos (Laje, porta, parede, janela e corrimão), número que pode ser aumentado com os devidos tratamentos dos dados em IFC e a variedade na base de dados de treino, possibilitando o aumento na gama de resultados e, consequentemente a de projetos aplicáveis.

Juntamente ao citado anteriormente, a aplicação do que foi feito em uma plataforma de modelagem arquitetônica é de suma importância e pode agregar imensamente na utilidade do modelo. Pensando nisso, o aplicativo Revit, esse de onde foram tirados os dados da base de dados, é uma opção interessante a realizar integração, tendo como maior objetivo realizar a

identificação de elementos em tempo real e analisar a viabilidade e validade do que está sendo produzido.

7. Referências bibliográficas

[1] Shashank Reddy Vadyala, Sai Nethra Betgeri, John C. Matthews, Elizabeth Matthews, A review of physics-based machine learning in civil engineering, Results in Engineering, Volume 13,2022,100316,ISSN 2590-1230, https://doi.org/10.1016/j.rineng.2021.100316. Keywords: Physics-based machine learning; Machine learning; Deep neural network; Civil engineering

[2] Milad Mousavi, Ali TohidiFar, Amin Alvanchi, BIM and machine learning in seismic damage prediction for non-structural exterior infill walls, Automation in Construction, Volume 139, 2022, 104288, ISSN 0926-5805, https://doi.org/10.1016/j.autcon.2022.104288.

[3] J. Sresakoolchai and S. Kaewunruen, "Integration of Building Information Modeling and Machine Learning for Railway Defect Localization," in IEEE Access, vol. 9, pp. 166039-166047, 2021, doi: 10.1109/ACCESS.2021.3135451.

[4] Wang, Jing, et al. "A Machine Learning-Based Approach for BIM Object Localization." *International Symposium on Advancement of Construction Management and Real Estate*. Singapore: Springer Singapore, 2019.

[5] Khaoula Raboudi and Abdelkader Ben Saci. 2020. Machine learning for optimized buildings morphosis. In Proceedings of the 2nd International Conference on Digital Tools & Uses Congress (DTUC '20). Association for Computing Machinery, New York, NY, USA, Article 20, 1–5. https://doi.org/10.1145/3423603.3424057

[6] Bonsang Koo, Byungjin Shin, Applying novelty detection to identify model element to IFC class misclassifications on architectural and infrastructure Building Information Models, Journal of Computational Design and Engineering, Volume 5, Issue 4, 2018, Pages 391-400, ISSN 2288-4300, https://doi.org/10.1016/j.jcde.2018.03.002.

[7] Eike Natan, Análise de Desempenho de Modelos de Classificação de Elementos Arquitetônicos em Modelos BIM . 2020

8. Outras atividades

Apesar de grande parte do trabalho ter sido realizada unicamente utilizando a parte de computação nas bases de dados de elementos, para acompanhar esse projeto, foram necessárias algumas reuniões para alinhamento de conhecimento com pesquisadores da Engenharia Civil, havendo assim uma boa troca de conhecimento sobre as áreas interessadas. Principalmente sobre a temática de treinamento de modelos e como os dados são retirados do Revit. A partir desses encontros, tornou-se mais fluida a execução do projeto, visto que, com base teórica e o entendimento dos conceitos, facilitou-se a conexão entre os respectivos projetos.