Engenharia do Conhecimento

(Relatório do 2º Trabalho)

Docente: Sofia Teixeira

Alunos: Frederico Prazeres (fc56269), Ricardo Sobral (fc56332), Nina Tinga (fc53531), Edson Aníbal (fc57046)

Introdução

Neste relatório vamos explicar todas as abordagens e escolhas que tomámos durante a realização do trabalho.

Começamos pelo pré-processamento de dados (1º passo no jupyter notebook), onde tornámos os valores da coluna alvo (Biodegradable) em valores binários (0 e 1) e separámos esta do resto do dataset (conjunto Y para a coluna Biodegradable e X para o resto das colunas). Criamos duas listas uma com as colunas que não deve ser escaladas e as que devem ser escaladas (explicação na Fase de Scaling), e damos split ao dataset (80/20), 80% do dataset é usado para testar e encontrar os melhores modelos de imputing, scaling, feature selection e classificação. Os restantes 20% mantêm-se inalterados para no fim, para que assim seja possível verificar a eficácia que o melhor modelo selecionado tem de classificar um com uma data set no qual ele nunca foi treinado.

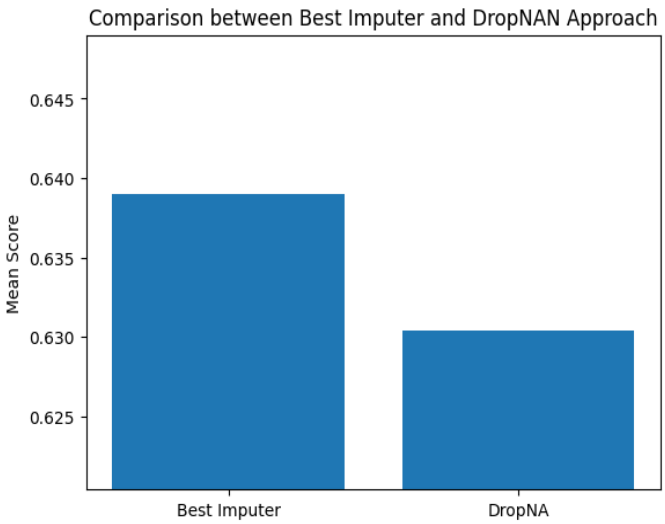
Para o cross-validation, usamos sempre a mesma função, o combined_score, que consiste numa média ponderada das métricas lecionadas em aula. Deu-se mais importância ao f1-score, que têm em conta a parte positiva do dataset (o dataset contêm muitos mais valores positivos do que negativos) e o Mathews Correlation Coefficient que conjuga a eficiência do modelo identificar valores positivos e negativos. É de notar que as métricas são escaladas para que por exemplo o Mathews Correlation Coefficient que tem uma escala de -1 a 1, ou seja diferente das outras, tenha mais impacto do que o lhe é devido.

O modelo que usámos para o Cross-Validation foi o StratifiedKFold com 5 splits, visto que para cada fold é mantida a relação entre o número de positivos e negativos (RB e NRB), por exemplo se tivermos uma relação de 1 para 10 na data set inteiro, para cada fold vai se manter essa relação de 1 para 10, o que é importante ter em conta quando a data set não é equilibrada como é o caso. Para além disso, este modelo ofereceu um tempo de computação relativamente baixo. Os scores médios do cross-validation para cada modelo e em cada fase encontram-se no jupyter notebook enviado.

Para encontrar os melhores modelos, usámos o K-Nearest Neighbours, visto que é um modelo relativamente rápido. O código inicialmente incorporava mais modelos para teste, mas optámos por usar apenas um, pois usar dois ou mais modelos aumentava significativamente o tempo de computação, tornando assim difícil realizar a testagem e a avaliação dos resultados. Vale a pena referir que o código está preparado para testar mais do que um modelo de classificação (existem ciclos).

Fase de imputing

Nesta fase, fomos à procura do melhor modelo de imputing para o dataset. Como já referimos em cima, usámos o KNN como modelo de classificação e cross validation com o combined_score para encontrar o melhor modelo. Os imputers que testámos foram: o Mean, Median, Most Frequent e KNN imputer. O melhor foi o Most Frequent por uma margem curta.



O gráfico de barras que mostra a diferença de performance entre os imputers testados encontra-se no ipynb.

Adicionalmente, testámos a performance do dataset com a abordagem de NaN's drop, ou seja, de ignorar os valores que não são números do dataset. Este foi o gráfico que obtemos:

Concluiu-se que a estratégia de dropar NaN's não melhora a performance, sendo até pior que todos os outros imputers, o que é expectável dado o elevado número de elementos NaN no dataset (este número é exibido antes de testar DropNaN Aproach).

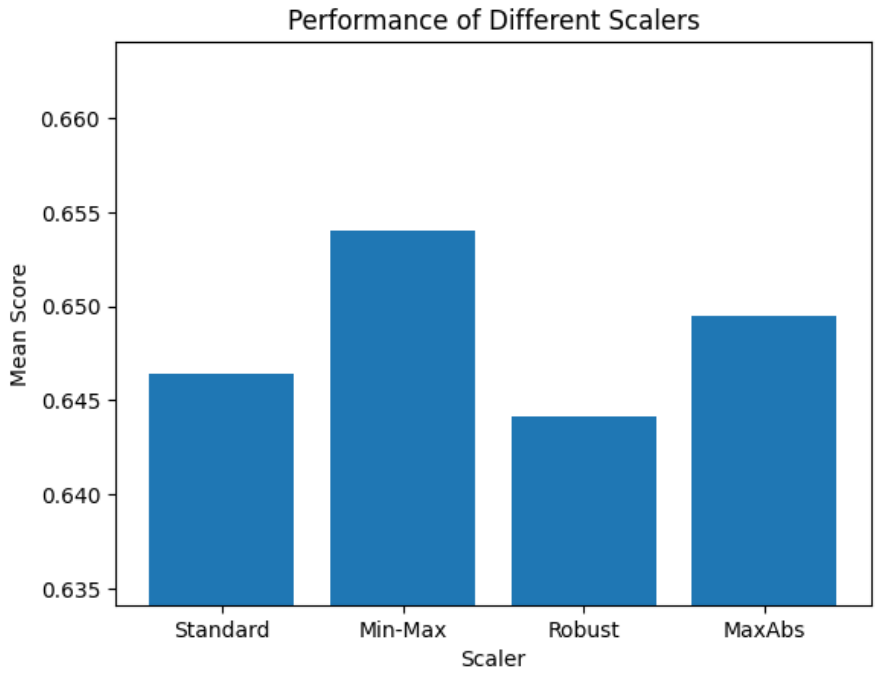
Fase de scaling

Para o scaling escolhemos os seguintes scalers: Standard, Min-Max, Robust e MaxAbs scaler. O melhor modelo encontrado foi o MinMax.

Num tom de nota, utilizámos o DecisionTreeClassifier no ínicio do trabalho para encontrar os melhores imputers, scalers etc. Porém, ao descobrirmos que o DecisionTree não é sensível ao scaling, decidimos retirá-lo do array dos modelos para teste.

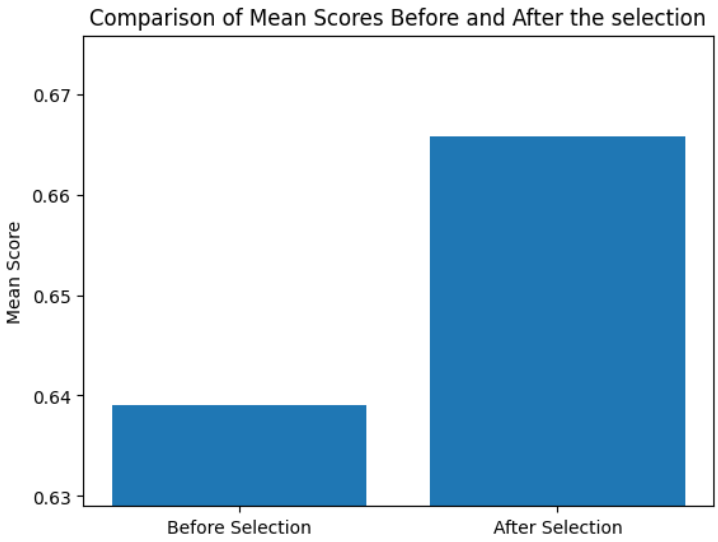
É nesta fase que ignoramos as colunas com valores que não devem ser scaled, visto que apesar de serem colunas numéricas na verdade a correspondem a valores categóricos e não devem ser escalados como os outros valores numéricos. Para isso, utilizou-se o ColumnTranformer do Scikit-Learn. Após o scaling com o melhor modelo obteu-se uma melhoria de cerca de 2 %.

Este é o gráfico que demonstra as diferenças de performance entre os scalers testados:



Feature Selection

Para o feature selection, utilizámos os seguintes modelos: SelectKBest, Recursive Feature Elimination, Stepwise backward e forward. Para o estimator dos últimos 3 modelos usou-se o decision tree. A certa altura usámos o LogisticRegression mas como número de iterações era muito elevado decidimos usar o DecisionTree como selecionado nas aulas. O modelo com melhor performance foi o Stepwise Foward com uma melhoria de cerca de 4% em relação ao dataset sem feature selection. O gráfico que mostra as performances dos selectors testados encontra-se no jupyter notebook.



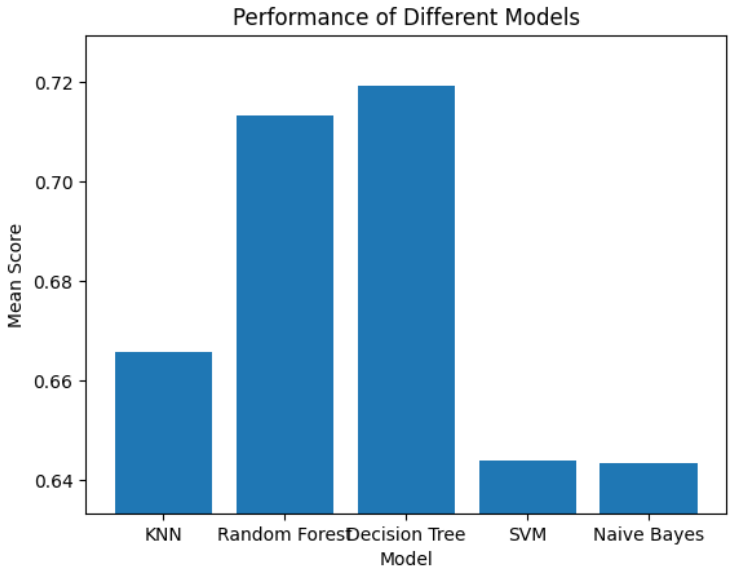
A diferença entre o dataset com selection e sem selection é evidenciada por este gráfico:

O melhor feature selector selecionou as seguintes features como as que melhor definem o dataset: J_Dz(e), nCb, C, SM6_L, Me, nArNO2, nCrt, F02_CN, SM6_B, nArCOOR.

Melhor modelo de classificação

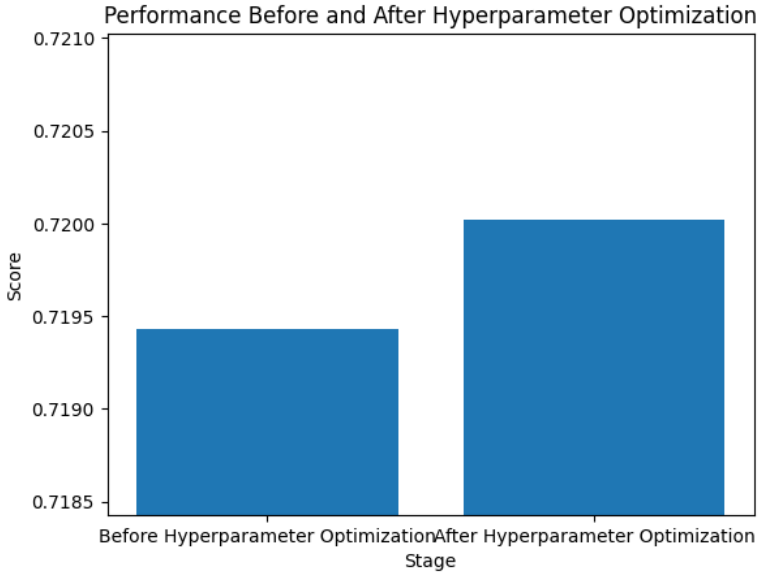
O melhor modelo é o Decision Tree. Para além deste foram testados os seguintes modelos: KNN, Random Forest, SVM e Naive Bayes. O Decision Tree desde muito cedo no trabalho demonstrou ser o modelo com o melhor score de Cross-Validation.

Este é o plot que evidencia a diferença (Cross-Validation) entre o Decision Tree e os outros modelos:



Hiper-parametrização

Após alguma testagem encontrou-se um grupo de pârametros no qual o grid_search encontra um hiper-parametro que melhora o melhor modelo. Este é o parâmetro: {'criterion': 'entropy', 'max_depth': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 6} Este é o gráfico de barras que evidencia a melhoria após a hiperparametrização do modelo.



Conclusões finais

Como foi referido no ínicio do relatório, houve uma divisão do dataset. Um dataset permaneceu inalterado, não tendo sido alvo de nenhum procedimento de scaling, feature selection e hiperparametrização até agora de maneira a deixar a dataset a mais pura possível. Posto isto os 20% levam imput porque o cross-validation não aceita que algumas colunas tenham NaN's, são escalados e são selecionadas as features pois o melhor modelo esta à espera de receber 10 features e não 41 como o como a dataset tem originalmente. Após estas transformações já é possível, calcular o combined_score do melhor modelo com os melhores hiperparametros encontrados sobre o os 20% do split incial.

Os resultados obtidos foram bastante satisfatórios visto que o resulto do combined_score foi bastante semelhante ao resultado do melhor modelo com os devidos hiperparametros do bloco de código acima, pelo que a diferença entre os dois valores é de cerca de 1%. Sendo assim é possível afirmar que o modelo resultante dos testes realizados pelo grupo, é generalizável, ou seja, é espectável que para uma data set no qual nunca viu, tenha uma performance semelhante. Contudo é importante frisar que estes resultados são específicos para a data set biodegradable_a, pelo que se fosse fornecido outra data set, era provável que os valores fossem diferentes, mas bastante semelhantes a estes.