Relatório Grupo 33

Engenharia do Conhecimento

Membros:

- Tomás Ezequiel - 55177 - 16h
- Rodrigo Silva - 54416 - 16h
- Pedro Correia - 54570 - 16h
- João Leitão - 50929 - 16h

Introdução e Objetivos:

O objetivo deste relatório é desenvolver o melhor 'Classification Model' possível para a tarefa utilizando um conjunto de dados fornecidos. No nosso caso, o conjunto de dados em questão é uma versão editada de um dataset *'QSAR biodegradation Data Set'* , especificamente com o nome "*biodegradable_a.csv*".

Para a realização desta tarefa, exploramos diversos métodos de modo a podermos tirar conclusões. O foco principal será selecionar o melhor modelo com base no seu desempenho e na sua simplicidade. Neste caso, a variável a ser classificada é o atributo "*Biodegradable*", indicando se um composto é biodegradável ou não.

É importante observar que nem todos os dados estão normalizados, o que significa que algumas variáveis podem ter diferentes escalas ou intervalos. De mencionar que no conjunto de dados fornecido, existem vários valores ausentes, que eventualmente necessitam de ser tratados adequadamente (durante a etapa de processamento de dados).

De modo a atingirmos os nossos objetivos com esta análise, aplicaremos diferentes configurações de '*hyperparameter*' aos '*Classification Models*' de modo a encontrarmos a melhor combinação para uma classificação exata.

Resumindo, este relatório tem como principal objetivo desenvolver o melhor modelo de classificação para prever a biodegradabilidade de compostos usando o conjunto de dados fornecidos. Abordaremos diferentes métodos, realizaremos o tratamento de dados, experimentaremos diferentes configurações de '*hyperparameter*' e identificaremos assim as características mais significativas que contribuem para a tarefa de classificação. O objetivo final é fornecer insights valiosos para o avanço da pesquisa em biodegradação e desenvolvimento de materiais sustentáveis.

Data processing:

Neste passo importante, foram realizadas diversas etapas de processamento de dados para preparar o conjunto de dados fornecidos para a construção de modelos de classificação. As etapas de processamento de dados são essenciais para lidar com as características específicas do conjunto de dados e garantir que os modelos são capazes de extrair informações relevantes para a tarefa de classificação.

De seguida, está um resumo das nossas etapas de processamento:

1. **<u>Separação dos dados dos nomes das colunas</u>**:

```
X = data.iloc[1:,:-1]
y = data.iloc[1:, -1]
```

No caso da variável **X**:

- o primeiro parâmetro dentro do iloc (`data.iloc[1:,:-1]`) permite-nos filtrar e não incluir a primeira linha neste caso;
- o segundo parâmetro dentro do iloc (`data.iloc[1:,:-1]`) permite-nos filtrar e incluir tudo menos a última coluna ;

No caso da variável **Y**:

- o primeiro parâmetro dentro do iloc (`data.iloc[1:, -1]`) permite-nos filtrar e não incluir a primeira linha neste caso;
- o segundo parâmetro dentro do iloc (`data.iloc[1:, -1]`) permite-nos filtrar e incluir somente a última coluna ;

2. **<u>Preenchimento dos espaços vazios:</u>**

O conjunto de dados fornecido, como já foi mencionado, contém valores ausentes que devem ser tratados de maneira apropriada antes de qualquer análise.

```
imputer = SimpleImputer(strategy='median')
X = pd.DataFrame(imputer.fit_transform(X))
```

No nosso caso, optamos por usar a classe '***SimpleImputer()***' (classe da biblioteca *sklearn.impute*), que nos permite substituir valores em falta em um conjunto de dados. Neste caso, a estratégia que optamos é '*median*' (mediana). A mediana é o valor do meio de uma lista de números ordenada.

Quando aplicada a uma variável (coluna) com valores ausentes, o imputer calcula o valor da mediana com base nos valores não ausentes dessa variável e substitui os valores ausentes pela mediana calculada.

Na segunda linha, o metodo '***fit_transform()***' ajusta o *imputer* aos dados de treinamento (calculando a mediana), e de seguida, realiza a substituição dos valores ausentes substituindo-os pela mediana. O resultado é uma matriz numpy transformada já com os valores todos inseridos. É então criado um novo objeto DataFrame '**X**' para armazenar a matriz modificada.

3. Normalização:

Após a realização dos passos mencionados acima, notamos que algumas variáveis podem ter diferentes escalas ou intervalos, é então necessário normalizar para efetuar esse ajuste de escalas, garantido assim que tem uma distribuição comparável.

```
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X))
```

Para tal, optamos por usar, o "***StandartScaller()***" (classe da biblioteca *sklearn.impute*), na qual é armazenado na variável '*scaler*'. Esta classe é utilizada para normalizar os dados, o que significa que ele ajusta os dados para que tenham uma média de zero e um desvio padrão de um.

Novamente, na segunda linha, o método '***fit_transform()***' do objecto *scaller* é aplicado ao conjunto de dados '**X**'. Este método ajusta a normalização dos dados (calculando a média e o desvio padrão) e, em seguida, é aplicada a alteração. O resultado é uma matriz numpy transformada que contém os dados normalizados. É então criado um novo objeto DataFrame '**X**' para armazenar a matriz normalizada.

Resumindo, estas linhas de código normalizam, utilizando a média e o desvio padrão dos dados existentes nas colunas do conjunto de dados '**X**' e, em seguida, aplica a normalização, transformando os dados para ter uma média zero e um desvio padrão de um.

Variable Selection:

Usamos o SequentialFeatureSelector com Logistic Regression que avalia as interações entre as colunas e escolhe que colunas seleciona com base no seu impacto no desempenho do modelo. São selecionadas as 15 colunas mais importantes, 15 para não serem nem demasiadas colunas, nem demasiado poucas. Demasiadas colunas, principalmente usando o Grid Search pode causar tempos de execução muito longos, enquanto poucas podem por em causa os resultados obtidos.

O SequentialFeatureSelector oferece métodos para obter informações sobre o processo de seleção de características, como as pontuações de cada característica, a ordem em que as características foram selecionadas e o conjunto final de características selecionadas.

O SequentialFeatureSelector pode ser manipulado por vários parâmetros como:

- **estimator**: O estimador ou modelo usado para avaliar a importância das características.

- **n_features_to_select**: O número de características a serem seleccionadas (no nosso caso optamos por selecionar somente 15)

- **direction**: A direção do processo de seleção, que pode ser "forward" ou “backward”

No nosso caso, utilizamos o SequentialFeatureSelector para selecionar as características mais informativas com base no modelo de regressão logística e transformar o conjunto de dados de acordo, resultando em selected_features e X contendo apenas as características escolhidas pelo Selector.

Model results:

Biodegradable = RB:

Avaliação Logistic Regression:

```
Logistic Regression Metrics:
Accuracy:  0.9452354874041621
Precision:  0.954954954954955
Recall:  0.9801849405548216
The Matthews correlation coefficient is:  0.7990
F1-score:  0.9674054758800522
```

Avaliação Decision Tree:

```
Decision Tree Metrics:
Accuracy:  0.9485213581599123
Precision:  0.9622395833333334
Recall:  0.9762219286657859
The Matthews correlation coefficient is:  0.8139
F1-score:  0.9691803278688524
```

Avaliação Random Forest:

```
Random Forest Metrics:
Accuracy:  0.963855421686747
Precision:  0.9763157894736842
Recall:  0.9801849405548216
The Matthews correlation coefficient is:  0.8715
F1-score:  0.9782465392221491
```

Avaliação SVM:

```
SVM Metrics:
Accuracy:  0.9649507119386638
Precision:  0.9713914174252276
Recall:  0.9867899603698811
The Matthews correlation coefficient is:  0.8734
F1-score:  0.9790301441677589
```

Biodegradable = NRB:

Avaliação Logistic Regression:

```
Logistic Regression Metrics:
Accuracy:  0.9452354874041621
Precision:  0.8897058823529411
Recall:  0.7756410256410257
The Matthews correlation coefficient is:  0.7990
F1-score:  0.8287671232876713
```

Avaliação Decision Tree:

```
Decision Tree Metrics:
Accuracy:  0.9496166484118291
Precision:  0.8819444444444444
Recall:  0.8141025641025641
The Matthews correlation coefficient is:  0.8175
F1-score:  0.8466666666666667
```

Avaliação Random Forest:

```
Random Forest Metrics:
Accuracy:  0.963855421686747
Precision:  0.9019607843137255
Recall:  0.8846153846153846
The Matthews correlation coefficient is:  0.8715
F1-score:  0.8932038834951457
```

Avaliação SVM:

```
SVM Metrics:
Accuracy:  0.9649507119386638
Precision:  0.9305555555555556
Recall:  0.8589743589743589
The Matthews correlation coefficient is:  0.8734
F1-score:  0.8933333333333334
```

Os campos usados para realizar esta avaliação destes modelos, são campos normalmente usados para avaliar o desempenho de modelos de classificação.
Ao considerar mos '*Accuracy*', '*Precision*', '*Recall*', '*The Mathews correlation*', '*F1-score*', estamos a avaliar vários aspectos de desempenho do modelo, como acurácia geral,

capacidade de identificar corretamente casos positivos, capacidade de minimizar falsos positivos e falsos negativos e o equilíbrio geral entre precisão e recall.

Hyperparameter tuning:

Utilizámos o GridSearch para encontrar a combinação ideal de hiperparâmetros para o modelo respetivo. Resumidamente, o GridSearch realiza uma busca exaustiva por todas as combinações possíveis de valores especificados para os hiperparâmetros do modelo; é usado também para otimizar o desempenho do modelo, testando diferentes combinações e selecionando aquela que produz os melhores resultados; durante o processo de GridSearch, o modelo é treinado e avaliado várias vezes usando validação cruzada, o que proporciona uma estimativa mais confiável do desempenho do modelo.

Discussão e conclusão:

Na avaliação podemos verificar que os modelos que melhor pontuam em Accuracy são o Random Forest e o SVM as restantes métricas bastante semelhantes entre si. Para uma melhor comparação entre os dois, colocamos todas as métricas lado a lado para tirar conclusões.

Com base nas métricas de avaliação, bem como na análise visual da Confusion Matrix, tanto o Random Forest quanto o SVM demonstram ser modelos robustos e eficientes para problemas de classificação no QSAR Biodegradation Data Set. Embora ambos os modelos apresentam resultados promissores, o Random Forest tende a alcançar um desempenho ligeiramente superior em termos de Accuracy e Matthews Correlation Coefficient. A Accuracy, escolhida como métrica principal para a otimização de hiperparâmetros, indica a proporção de classificações corretas em relação ao total de amostras. Além disso, o MCC avalia a qualidade das classificações binárias, considerando tanto verdadeiros positivos quanto verdadeiros negativos. Com base nessas métricas, conclui-se que o modelo Random Forest é o melhor classificador produzido neste projeto, demonstrando seu potencial para prever a biodegradabilidade no QSAR Biodegradation Data Set.

É importante notar que os dois primeiros modelos embora não vençam em pontuação em Accuracy nem Matthew Correlation Coefficient, não têm pontuações muito inferiores dos restantes e são mais simples e mais rápidos de implementar e executar.