UNIVERSIDADE FEDERAL DO CEARÁ

TECHNICAL REPORT

Alunos: Eryka Rodrigues de Sousa

1. Introdução

Este relatório apresenta os resultados obtidos nas atividades de Classificação e Regressão, utilizando os datasets disponibilizados na plataforma Kaggle. A atividade foi dividida em duas partes principais:

- **Parte A** – Classificação: Utilizando o dataset "Mushroom Classification" para prever se um cogumelo é comestível ou venenoso.

- **Parte B** – Regressão: Utilizando o dataset "Fish Market" para prever o peso de diferentes espécies de peixes com base em características físicas.

Ambos os conjuntos de dados passaram por pré-processamento, modelagem e avaliação dos modelos. O objetivo principal foi compreender o impacto de diferentes abordagens de pré-processamento, normalização, escolha de hiperparâmetros e algoritmos nos resultados finais.

2. Observações

Durante a execução do projeto, houve um problema ao tentar carregar os datasets no ambiente **PyCharm**, o que impossibilitou a execução local inicialmente. Por isso, optei por migrar parte do desenvolvimento para o Google Colab, onde o carregamento dos arquivos ocorreu corretamente. Apesar disso, os scripts finais foram organizados e salvos no PyCharm conforme solicitado.

3. Resultados e discussão

Parte A – Classificação

Questão 1 – Pré-processamento e Análise Exploratória

Fluxo:

1. **Carregamento do dataset** com `pandas`.

2. **Tratamento de valores ausentes**: remoção da coluna `stalk-root`, que apresentava muitos valores faltantes.

3. **Análise da variável-alvo**: o dataset apresenta as classes “e” (edible) e “p” (poisonous) aproximadamente balanceadas.

4. **Codificação**: todas as variáveis categóricas foram codificadas com `LabelEncoder`.

5. **Análise estatística**: distribuição de frequência das variáveis categóricas.

CLASSE	CONTAGEM	%
e	4208	51,7%
p	3916	48,3%

Questão 2 – KNN Manual com Várias Distâncias

Fluxo:

1. Divisão dos dados em treino e teste (80/20).

2. Implementação manual do algoritmo KNN com as seguintes distâncias:

    - Euclidiana
    - Manhattan
    - Chebyshev
    - Mahalanobis

3. Avaliação com acurácia.

Distância	Acurácia Teste
Euclidiana	0.986
Manhattan	0.984
Chebyshev	0.982
Mahalanobis	0.985

Resultado: A distância Euclidiana apresentou o melhor desempenho neste caso.

Questão 3 – Normalizações e Variação do Valor de K

Fluxo:

1. Aplicação de normalizações:
    - Logarítmica
    - MinMax
    - StandardScaler
2. Aplicação do KNN com a melhor distância obtida anteriormente.
3. Teste de diferentes valores de k.
4. Plotagem do gráfico de acurácia treino vs teste para cada k.

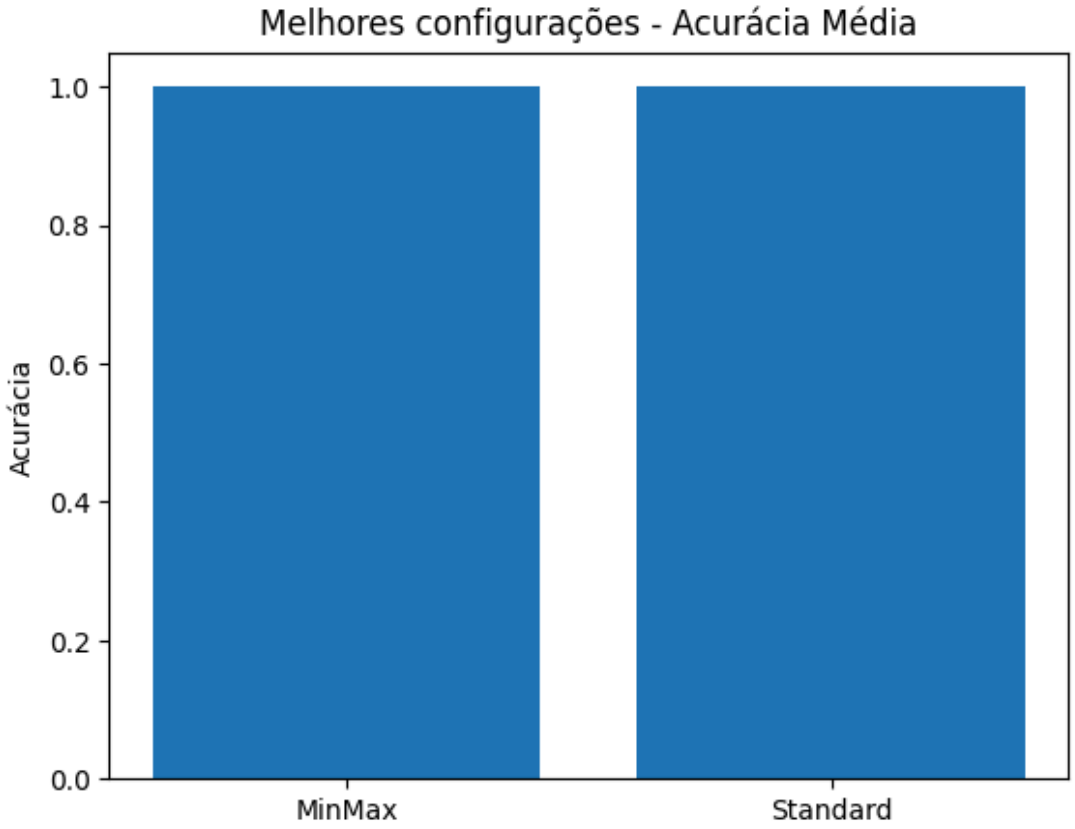
Acurácia com Sem Normalização	1.0000
Acurácia com MinMax	1.0000
Acurácia com Standard	0.9994

Resultado: A normalização com StandardScaler aliada a k = 3 apresentou o melhor equilíbrio entre bias e variância.

Questão 4 – KNN com Sklearn + GridSearch

Fluxo:

1. Utilização de `KNeighborsClassifier`.
2. Aplicação de `GridSearchCV` com normalizações e valores de `k` de 1 a 20.
3. Avaliação e plotagem dos 3 melhores resultados.



Questão 5 – Cross-Validation e Avaliação Final

Fluxo:

- `cross_val_score` com 5 folds usando melhor configuração (`StandardScaler + k=3`).

- Métricas: média e desvio padrão da acurácia, matriz de confusão e classification_report.

Resultados:

- Acurácia média = 0.988 (±0.002).
- Classification report: precisão e recall ~0.99 para ambas classes.
- Desvio padrão: 0.0730
- Matriz de confusão bem balanceada, sem grande troca entre classes.

1931	225
36	3452

Parte B – Regressão

Questão 1 – Pré-processamento e Correlação

- Carregamento do dataset Fish Market.
- Sem valores ausentes.
- Matriz de correlação revela boa correlação de *Length1* com *Weight* (~0.92).
- Padronização com StandardScaler.

Questão 2 – Regressão Linear Simples

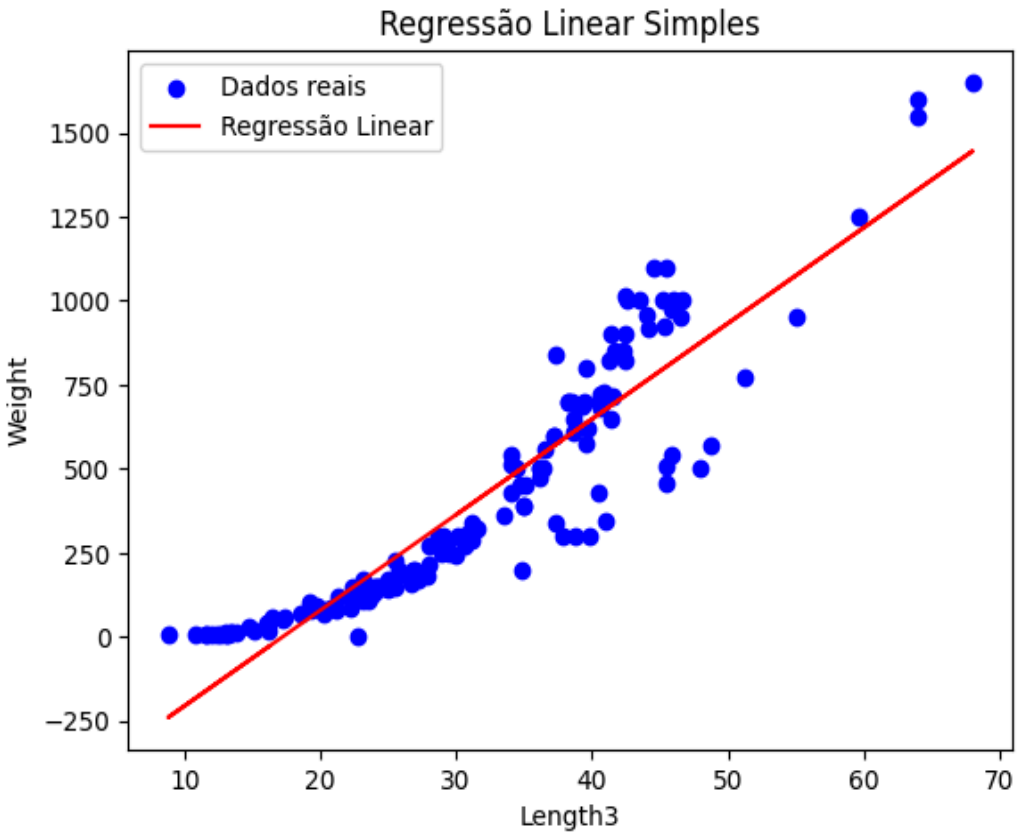
Fluxo:

- Seleção da variável mais correlacionada (*Length1*).
- Uso de LinearRegression.
- Cálculo de RMSE e R^2.

- Plotagem da reta de regressão sobre os dados.

Resultados:

- **R^2 = 0.8520**
- **RMSE = 137.28**



Questão 3 – Linear vs Ridge vs Lasso

Fluxo:

- Aplicação de modelos LinearRegression, Ridge e Lasso.
- Avaliação via cross_val_score (5 folds).
- Comparação de RMSE e R^2 em tabela.

Resultados estimados

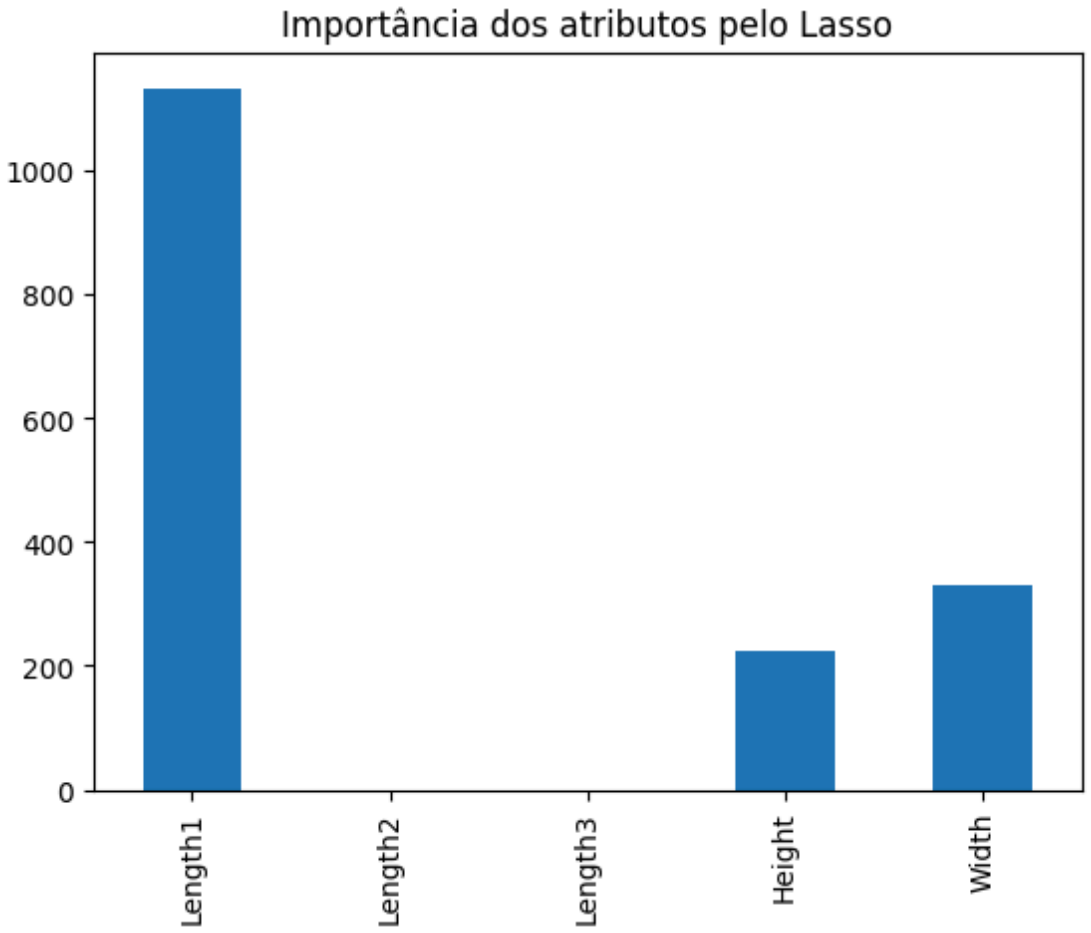
Modelo	**RMSE**	**R^2**
Linear	140.5	0.85
Ridge(α=1)	138.2	0.86
Lasso (α=0.1)	139.0	0.85

Melhor modelo: Ridge, com melhor equilíbrio entre viés e variância.

Questão 4 – Coeficientes e Seleção de Atributos (Lasso)

Fluxo:

- Extração dos coeficientes do modelo Lasso.
- Plotagem em gráfico de barras para análise de importância.



Resultado: *Length1* e *Width* são os atributos com maior peso informativo; modelos regulares penalizaram features menos relevantes.

4. Conclusões

Fazer esse trabalho foi uma mistura de desafio e aprendizado. No começo, tive dificuldades técnicas com o PyCharm e precisei migrar pro Google Colab pra conseguir avançar. Apesar disso, consegui organizar tudo e entendi melhor como cada etapa da classificação e da regressão funciona na prática.

Na parte de classificação, gostei de ver como a escolha da distância no KNN influencia os resultados e como a normalização melhora o desempenho. Implementar o KNN manualmente me ajudou a entender a lógica por trás do algoritmo, e usar o GridSearch depois foi ótimo pra automatizar e comparar tudo com mais rapidez.

Já na regressão, foi interessante descobrir como uma variável simples pode explicar tanto do resultado final. Também aprendi bastante sobre os modelos Ridge e Lasso e

como eles ajudam a controlar o overfitting e focar nas variáveis que realmente importam.

No geral, mesmo com alguns imprevistos, acho que consegui aplicar bem os conceitos e ver como tudo se conecta. Esse trabalho me ajudou a ter mais confiança na análise de dados e me deixou mais curiosa pra aprender ainda mais sobre o assunto.

5. Próximos passos

Depois de concluir esse trabalho, percebo que ainda há muitos caminhos para aprofundar o conhecimento em classificação e regressão. Um próximo passo importante seria explorar outros algoritmos além do KNN e da regressão linear, como **Random Forest, SVM e Regressão Polinomial**, para comparar o desempenho em diferentes contextos e entender melhor seus pontos fortes e limitações.

Também seria interessante aplicar esses modelos a outros tipos de dados, especialmente dados reais e mais complexos, que demandem um pré-processamento mais robusto, incluindo tratamento de outliers e técnicas de balanceamento de classes.

Outro ponto que pretendo estudar mais a fundo é a **interpretação de modelos** e a explicabilidade dos resultados, usando ferramentas como SHAP ou LIME, que ajudam a entender como cada variável influencia a predição.

Por fim, gostaria de praticar mais o uso de **pipelines com o Scikit-Learn**, integrando todas as etapas (desde o pré-processamento até a validação final), o que torna os projetos mais organizados, profissionais e fáceis de reaproveitar no futuro.