GUILHERME SANTOS NUNES – 558989

KAIQUE ZAFFARANI – 556677

KAIRO DA SILVA SILVESTRE DE CARVALHO – 558288

PEDRO JOSUÉ PEREIRA ALMEIDA – 554913

CHALLENGE – SPRINT 4

DATA SCIENCE & STATISTICAL COMPUTING

SÃO PAULO

2025

RESUMO

Esta documentação apresenta diferentes experimentações com algoritmos em modelos de machine learning (aprendizado de máquina) capazes de resolver problemas relacionados ao desafio principal do challenge – realizado com a empresa parceira DASA – descrevendo sobre os processos de implementação dos algoritmos em modelos e indicando melhores abordagens em pról da eficiência produtiva com qualidade probatória.

A sprint 4 contempla um novo problema baseado no desafio da empresa parceira, e um outro que fora previamente desenvolvido na sprint 3, mas trazendo outras técnicas visando aperfeiçoamento do prévio modelo. Ambos os projetos trabalham com pequenos datasets (bases de dados) gerados para testes (nenhum conteúdo apresentado nos mesmos é sigiloso).

Enquanto o problema passado enfatizou o tema central do challenge: baixa visibilidade de insumos em estoques para laboratórios, o novo prioriza a escolha do fornecedor de insumos mais confiável entre os demais, considerando características específicas de aprovação da empresa e visão mercadológica.

Ao sanar o problema, que inclusive exige uma predição binária, a empresa reavaliará fornecedores para que futuramente receba mais encomendas à prazo com produtos homologados e os envie para exames laboratoriais, resultando em prontidão e reposição dos estoques eficazmente. Por fim, Variadas métricas foram exploradas de acordo com o progresso dos modelos trabalhados.

SUMÁRIO

1. **Problema de Predição** 4
2. **Modelo KNN** 5
3. **Avaliação do KNN** 6
4. **Regressão Logística** 7
5. **Comparação de Resultados** 8
6. **Regressão de Ridge (L2) | Regressão de Lasso (L1)** 9
7. **Regressão Polinomial de 2º e 3º Grau** 11
8. **Árvore de Decisão e Random Forest** 14
9. **Recomendações Finais** 18
10. **Referências** 19

1. Problema de Predição

O problema de predição trata-se da insegurança durante a escolha de fornecedores para a compra de insumos e reestoque geral. Como solução, foi considerada a compra de insumos baseada no fornecedor mais compatível a determinadas caracteríticas (features), sendo cada critério avaliado de 0 a 10 pontos. A definição de fornecedor confiável ou não tão confiável provém de um resultado binário, considerando essas labels acima da média 8 e abaixo da média 8, respectivamente.

O dataset possui 500 fornecedores fictícios. Todos foram ranqueados com as seguintes features:

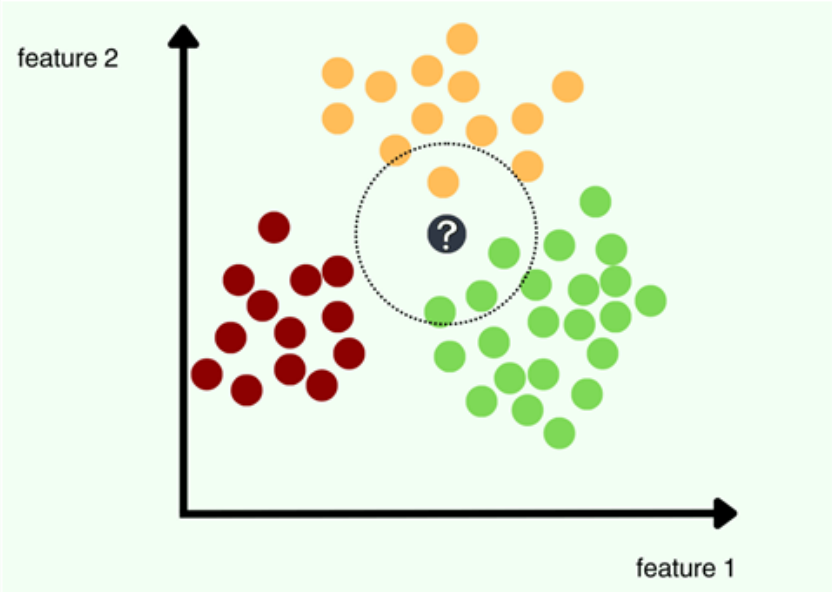
- **certificado** | O fornecedor tem homologações e possui normas da Anvisa?
- **rastreavel** | O mesmo apresenta números de lotes, validades, fabricantes e instruções de uso?
- **suporte** | Disponibiliza treinamentos, manuais claros e equipe de apoio capacitada?
- **logistico** | Conta com centros de distribuição organizados, controle de inventário e canais de atendimento prontos para responder rapidamente?
- **qualidade** | Os produtos fornecidos, além de eficientes, são bem construídos, duráveis e reconhecidos pelo mercado?

2. Modelo KNN

KNN, ou K-Nearest Neighbors (k-vizinhos mais próximos), é um algoritmo de aprendizado de máquina supervisionado que classifica ou prevê um novo ponto de dados com base na maioria das classes de seus "k" vizinhos mais próximos em um conjunto de dados. É uma abordagem simples e popular usada para tarefas de classificação e regressão. O algoritmo funciona comparando um ponto de dados com um conjunto de dados previamente treinado e memorizado para encontrar os "k" vizinhos mais similares.

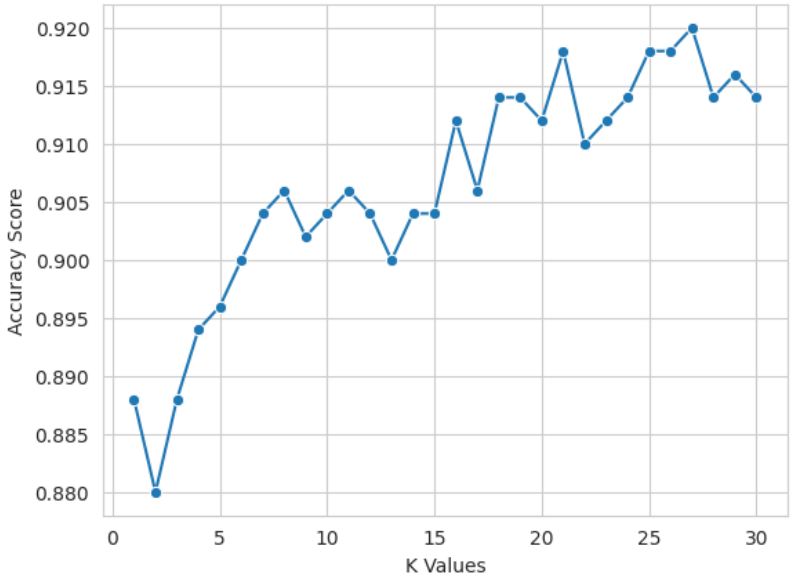
Através da análise exploratória de dados (EDA), a relação entre confiabilidade de fornecedor em relação às labelas de todas as features escolhidas foi nitidamente observável. Quanto menor a nota de certificação, rastreabilidade, suporte, logística ou qualidade, mais tangente à confiabilidade 0 o fornecedor se encontrará.

Após normalização dos dados, proporcionando mesma escala para valores de diferentes features, separação de dados e utilização de método StandardScaler para normalização, o dataset estava pronto para treinamento. Como primeira tentativa, 3 foi o número escolhido para k a fim de obter o resultado de proximidade entre outros vizinhos.



3. Avaliação do KNN

Durante a avaliação básica, uma acurácia de 89% foi atingida sendo k = 3. Entretanto, para encontrar o melhor valor de k, o método de validação cruzada foi aplicada: O número 27 se destacou como melhor no gráfico de pontuação de acurácia.




```
# Avaliação de acurácia, precisão, recall e f1 (resultados podem ser diferentes devido a randomização)
y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Acurácia:", accuracy) # % de acertos
print("Precisão:", precision) # Evita falsos positivos
print("Recall:", recall) # Evita falsos negativos
print("F1:", f1) # Equilíbrio entre precisão e recall
```

```
Acurácia: 0.944
Precisão: 0.9607843137254902
Recall: 0.9074074074074074
F1: 0.9333333333333333
```

Houve uma melhora significativa na acurácia ao aumentar o k após a validação cruzada: 94%. Os resultados demonstram boa precisão para inferência, considerando que o dataset é pouco abrangente – limitado em contextos e com nível de generalização (predispondo overfitting).

4. Regressão Logística

A regressão logística é um modelo estatístico usado para prever a probabilidade de um resultado categórico (como "sim" ou "não", "spam" ou "não spam") com base em uma ou mais variáveis independentes. Ela funciona estimando a probabilidade de uma variável dependente pertencer a uma determinada classe usando a função logística (ou função logit), que produz um valor entre 0 e 1.

Após a instância, treinamento e captura de inferência, um resultado de acurácia de 98% – mais alta em relação ao KNN, significando melhor adaptação do modelo às features fornecidas. Após a obtenção dos coeficientes e intercepto, vieram os seguintes resultados:

```
# Instância de modelo
logreg = LogisticRegression(random_state=42)

# Treinamento do modelo reaproveitando separação de dados
logreg.fit(X_train, y_train)

# Obtenção de inferência
y_pred = logreg.predict(X_test)

# Nível de acurácia
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

```
0.984
```

```
# Obtenção de coeficientes e intercepto
X = df.drop('confiabilidade', axis=1) # Reexecutando dataframe (não interfere nos resultados)

coeficientes = pd.DataFrame({
    'Variável': X.columns,
    'Coeficiente (β)': logreg.coef_[0],
    'e^β (Odds Ratio)': [round(pow(2.71828, b), 3) for b in logreg.coef_[0]]
})

intercepto = logreg.intercept_[0]

print("Intercepto (β0):", intercepto) # Indica se o modelo tem propensão inicial
# a classificar fornecedores como confiáveis,
# mesmo antes de considerar as variáveis (certificado, rastreável, etc.).

print("\nCoeficientes interpretáveis:") # Cada ponto a mais em certificado aumenta (×2.58)
# as chances do fornecedor ser confiável.
print(coeficientes)
```

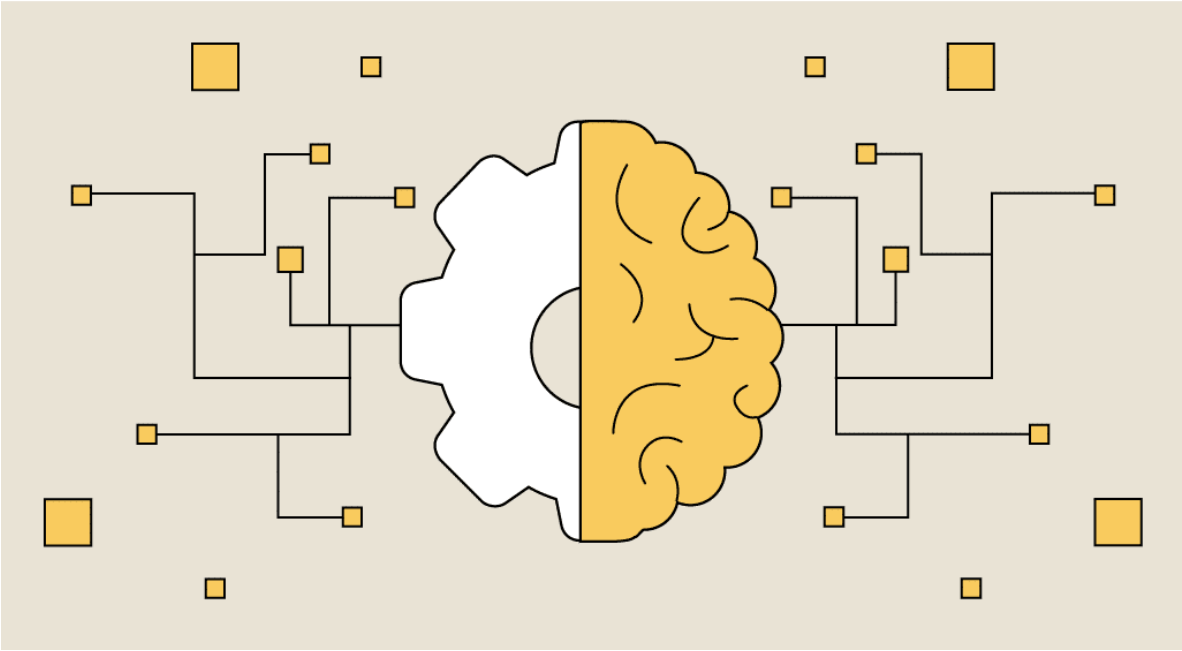
```
Intercepto (β0): -0.0035688595453757877

Coeficientes interpretáveis:
      Variável  Coeficiente (β)  e^β (Odds Ratio)
0  certificado         2.581459            13.216
1   rastreavel         2.890542            18.003
2      suporte         2.683382            14.634
3    logistico         2.606717            13.554
4    qualidade         2.659468            14.289
```

5. Comparação de Resultados

Considerando normalização de dados em ambos os processos, além do mesmo uso de dataset, apesar do algoritmo de KNN parecer convincente ao executar cálculo de distância euclidiana entre variáveis para evitar flutuação de resposta, permitir multidimensionalidade e encontrar um melhor valor (k) para acurácia com validação cruzada, a regressão logística trouxe maior precisão e interpretação dos coeficientes mais intercepto, prezando por uma escolha de fornecedor para compra de insumos mais confiável.

Em termos de perfomance, não houveram intervalos consideráveis que atrasassem a execução dos modelos. Todavia, o tempo de resposta pode variar em decorrência do estado aleatório de treinamento dos modelos e quantidade de dados presentes em dataset.

Uma das peculiaridades do KNN em relação à regressão logística é a sua rápida implementação em sistemas experimentais e intuitiva visualização das features em Exploratory Data Analysis (EDA), permitindo melhor interpretação dos resultados. No que se diz respeito à regressão, se destacam menor propensão à overfitting, perfomance, escalabilidade, e pouca demanda de poder computacional.



6. Regressão de Ridge (L2) | Regressão de Lasso (L1)

As técnicas a seguir estão sendo adicionadas ao projeto da sprint 3.

A regressão de Ridge é uma técnica estatística de regularização usada para corrigir o overfitting e a multicolinearidade em modelos de regressão, adicionando uma penalidade à soma dos quadrados dos coeficientes. Conhecida como regularização L2, ela encolhe os coeficientes para mais perto de zero, mas sem zerá-los, tornando o modelo mais robusto e estável, especialmente quando há muitas variáveis com pesos altos.

A regressão Lasso (Least Absolute Shrinkage and Selection Operator) é um tipo de regressão linear que adiciona uma penalização ao modelo para evitar overfitting e melhorar a seleção de variáveis. Diferente da regressão linear tradicional, o Lasso inclui um termo de regularização baseado na soma dos valores absolutos dos coeficientes. Isso faz com que alguns coeficientes sejam reduzidos exatamente a zero, eliminando variáveis menos relevantes e deixando o modelo mais simples e interpretável.

Após leitura do dataset, separação de features mais target, normalização e divisão dos dados para treinamento mais teste do modelo, houve validação cruzada para encontrar melhor valor de alfa tanto para regressão de Ridge quanto para a regressão de Lasso, e por fim o treinamento com melhor alfa.

```
# Validação cruzada para encontrar melhor valor de alfa para regressão Ridge
alphas = np.logspace(-3, 3, 100)  # testa de 0.001 até 1000
ridge_cv = RidgeCV(alphas=alphas, cv=5)
ridge_cv.fit(X_scaled, y)

best_alpha_ridge = ridge_cv.alpha_

print("Melhor alpha (Ridge):", best_alpha_ridge)
```

```
Melhor alpha (Ridge): 2.4770763559917115
```

```
# Validação cruzada para encontrar melhor valor de alfa para regressão Lasso
alphas = np.logspace(-3, 1, 100)
lasso_cv = LassoCV(alphas=alphas, cv=5, max_iter=10000)
lasso_cv.fit(X_scaled, y)

best_alpha_lasso = lasso_cv.alpha_

print("Melhor alpha (Lasso):", best_alpha_lasso)
```

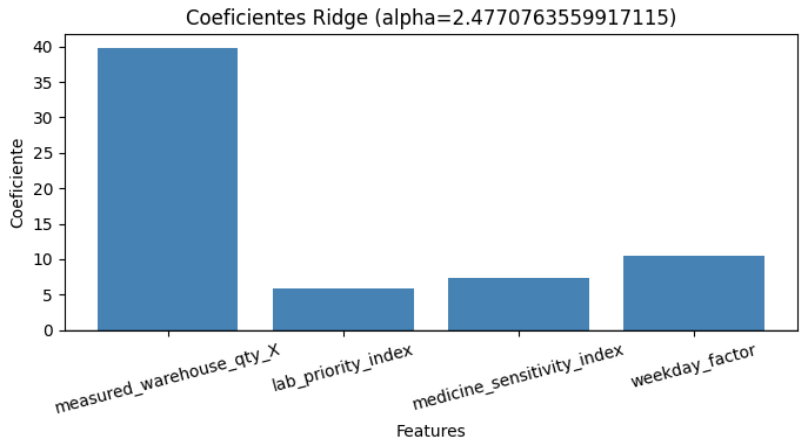
```
Melhor alpha (Lasso): 0.021544346900318846
```

```python
# Ridge com alpha ideal
ridge_final = Ridge(alpha=best_alpha_ridge)
ridge_final.fit(X_scaled, y)
ridge_coefs_final = ridge_final.coef_

# Lasso com alpha ideal
lasso_final = Lasso(alpha=best_alpha_lasso, max_iter=10000)
lasso_final.fit(X_scaled, y)
lasso_coefs_final = lasso_final.coef_
```

Na regressão de Ridge, todas as linhas (coeficientes) diminuem suavemente conforme α cresce. Nenhum coeficiente zera. O modelo fica mais “equilibrado” e estável.

Já na regressão de Lasso, várias linhas zeram de forma abrupta conforme α aumenta. O modelo “descarta” features irrelevantes automaticamente. Nenhum coeficiente foi excluído do modelo.

O alfa encontrado na validação é justamente o ponto que equilibra desempenho e regularização. É perceptível que a quantidade de insumos medida em cada almoxarifado é um fator decisivo para a inferência do modelo de regressão.

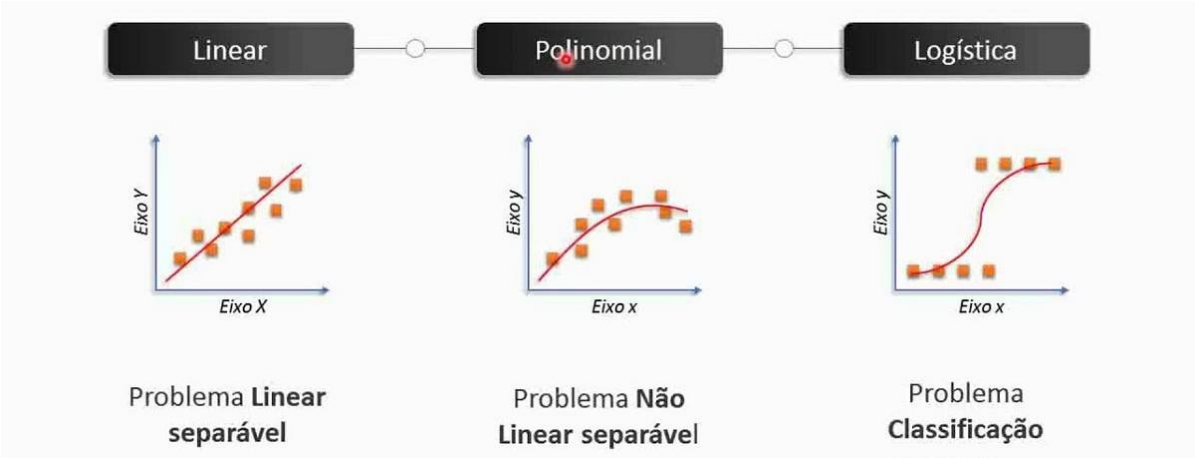


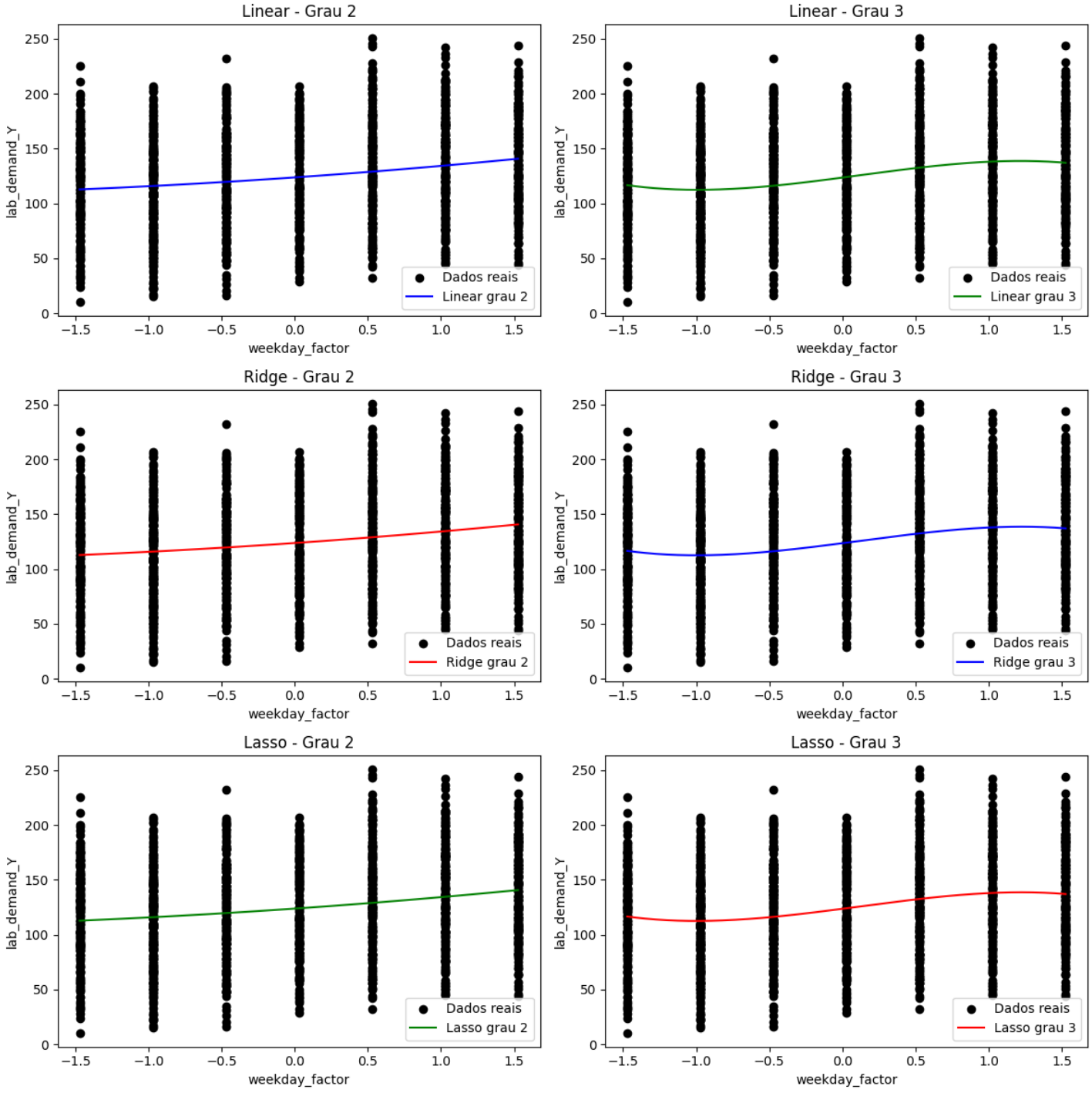
7. Regressão Polinomial de 2º e 3º Grau

A regressão polinomial é um método de análise de dados que modela a relação entre uma variável independente e uma variável dependente como um polinômio. Ela é usada quando a relação entre as variáveis não é linear, como em curvas em forma de U, e é uma forma de regressão linear, pois é linear em relação aos parâmetros a serem estimados.

O objetivo de usar regressão polinomial com Ridge ou Lasso é visualizar como a modelagem não-linear captura padrões complexos:

- **Regressão linear** → Linha reta → Captura apenas tendência linear
- **Polinomial de 2º grau** → Curvatura simples (U ou inverso de U)
- **Polinomial de 3º grau** → Curvatura mais complexa (S ou mudança de inclinação)
- **Regressão de Ridge** → Coeficientes suavizados → Evita overfitting
- **Regressão de Lasso** → Alguns termos polinomiais podem zerar → Curva simplificada




Linear - Grau 2
Linear - Grau 3
Ridge - Grau 2
Ridge - Grau 3
Lasso - Grau 2
Lasso - Grau 3
lab_demand_Y
weekday_factor
Dados reais
Linear grau 2
Linear grau 3
Ridge grau 2
Ridge grau 3
Lasso grau 2
Lasso grau 3

Utilizando a feature de fator diário da semana, o efeito do target y (demanda do laboratório) sobre ela muda de intensidade ou sinal perceptivelmente ao utilizar regressão polinomial de 3º grau.

- **Curvatura acentuada** → feature tem efeito não-linear forte.
- **Curva achatada / quase reta** → efeito da feature é quase linear ou regularização forte está “achatando” os coeficientes.
- **Mudança de direção da curva** → ponto crítico onde a influência da feature sobre y muda.

Resumo (média ± desvio):

	Métrica	Treino (média ± std)	Validação (média ± std)
0	R2	0.809 ± 0.006	0.803 ± 0.023
1	MAE	16.34 ± 0.19	16.46 ± 0.78
2	RMSE	20.42 ± 0.21	20.58 ± 0.85

Não houveram médias significativamente altas para o modelo em relação a anterior quando a regressão de Ridge foi utilizada. Possivelmente o mesmo aconteceria aplicando Lasso, ou polinômios de graus diferentes.

Estratégias como estender, reduzir ruído e adicionar mais features relevantes ao dataset (minimizando quantidade de itens no almoxarifado), além de considerar polinômios tendo padrão não-linear, podem consequentemente contribuir para o aumento das métricas e do nível de explicabilidade do modelo.

8. Árvore de Decisão e Random Forest

Uma árvore de decisão é um modelo de aprendizado de máquina que usa uma estrutura semelhante a um fluxograma para tomar decisões, dividindo os dados em ramificações até chegar a um resultado.

Uma Random Forest é um algoritmo mais avançado que combina várias árvores de decisão (uma "floresta") para fazer previsões mais precisas e robustas, evitando o problema de superajuste (overfitting) que pode ocorrer em uma única árvore.

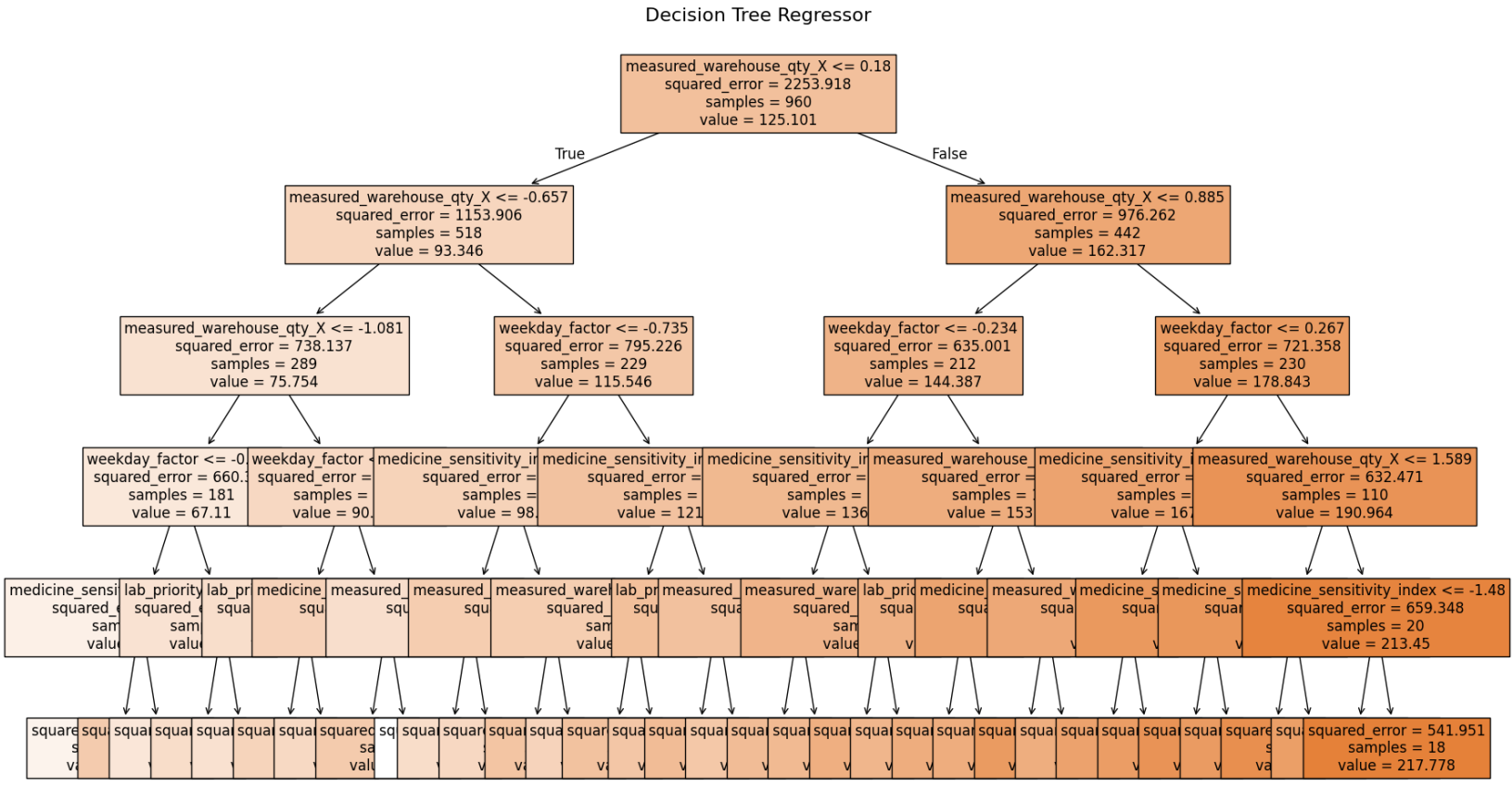
Projeto da sprint 3

=== Decision Tree ===

R² treino: 0.808, R² teste: 0.694

MAE treino: 16.71, MAE teste: 18.54

RMSE treino: 20.82, RMSE teste: 24.04

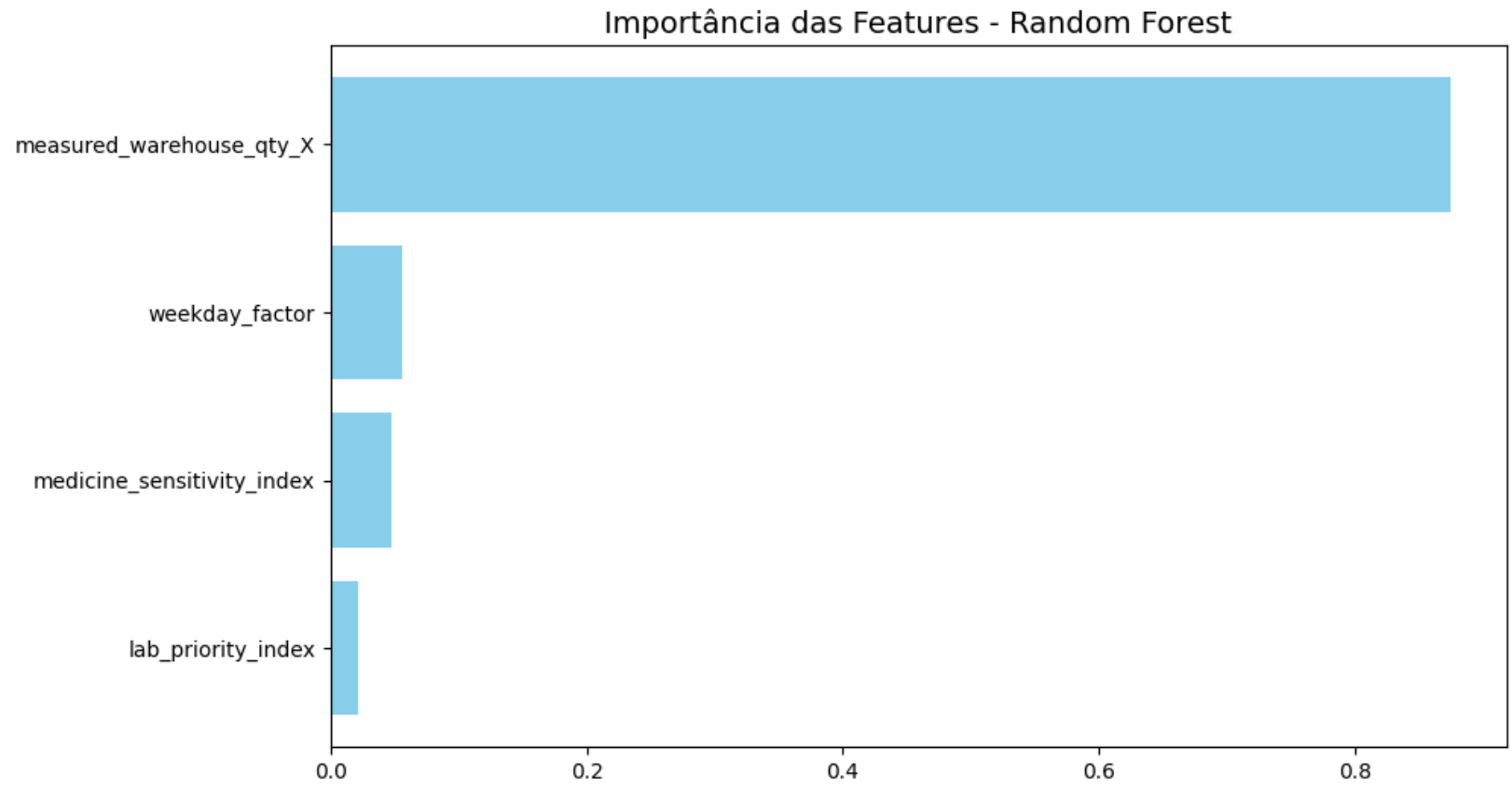


=== Random Forest ===

R^2 treino: 0.839, R^2 teste: 0.768

MAE treino: 15.48, MAE teste: 16.32

RMSE treino: 19.03, RMSE teste: 20.90



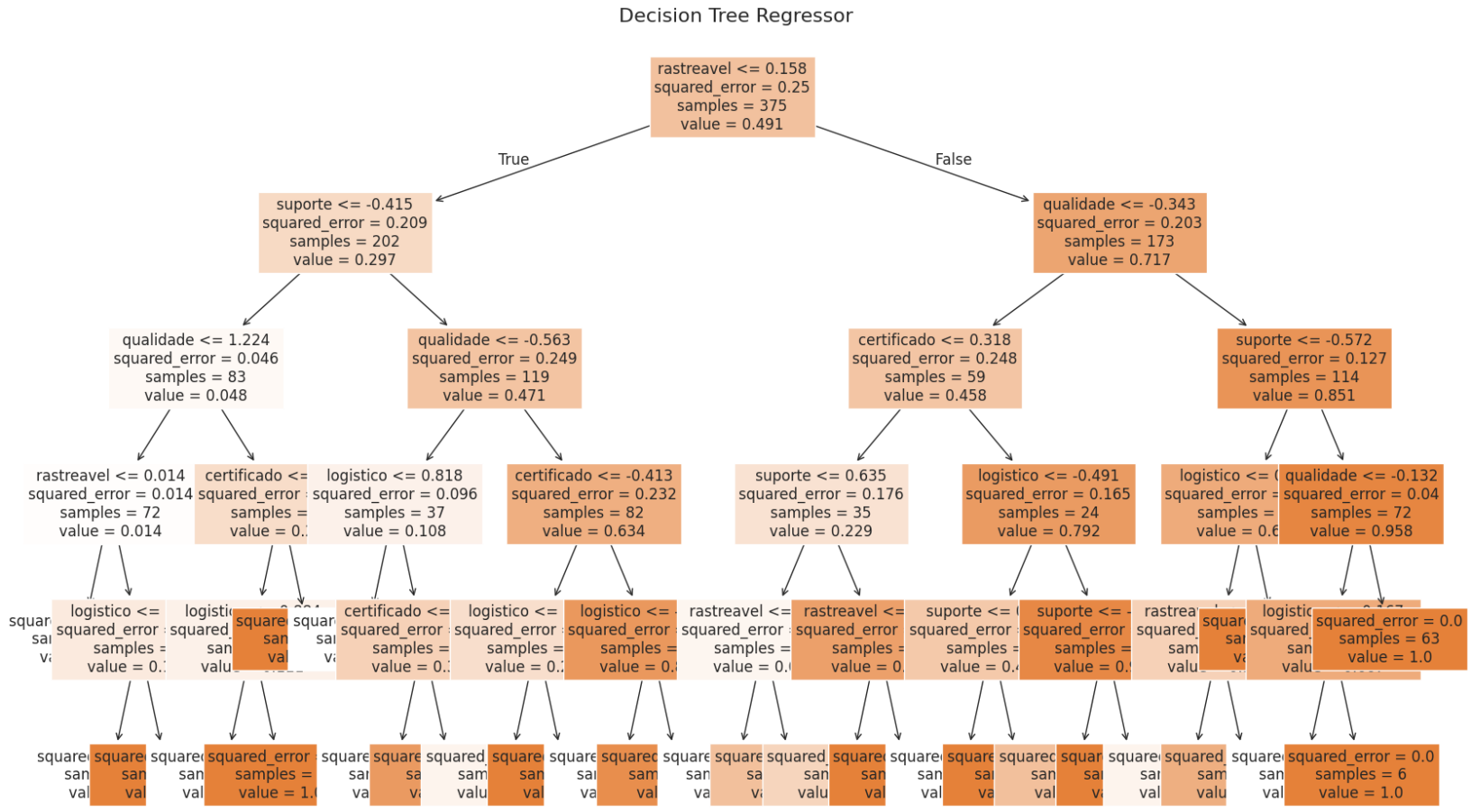
Projeto da sprint 4 (atual)

=== Decision Tree ===

R² treino: 0.864, R² teste: 0.344

MAE treino: 0.07, MAE teste: 0.19

RMSE treino: 0.18, RMSE teste: 0.40

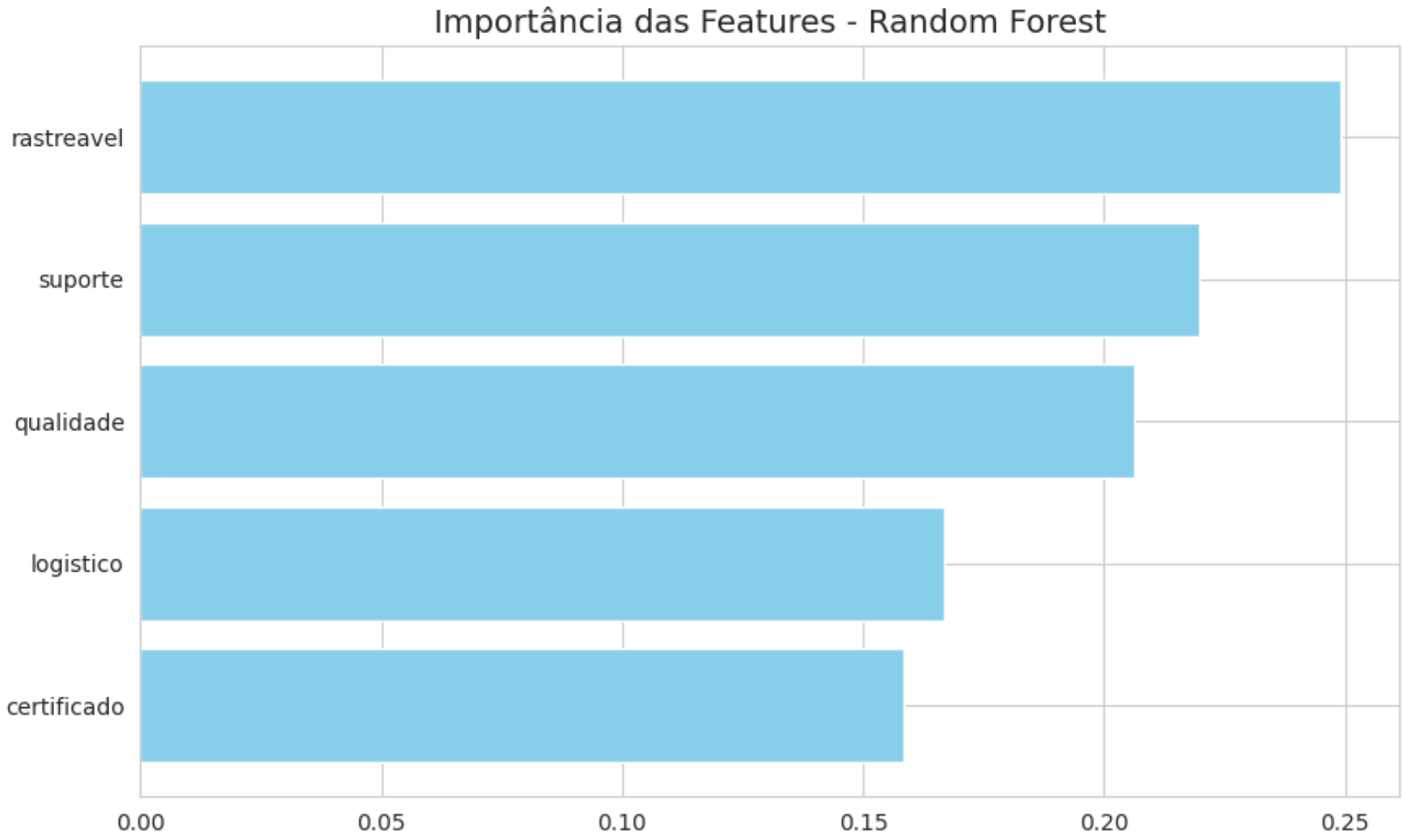


=== Random Forest ===

R^2 treino: 0.884, R^2 teste: 0.635

MAE treino: 0.13, MAE teste: 0.23

RMSE treino: 0.17, RMSE teste: 0.30



*Tanto a árvore de decisão quanto random forest precisaram de limitação na profundidade e tamanho das folhas, garantindo resultado sintético em dataset pequeno. As métricas de treino sairam perfeitamente altas. Além de indicar overfitting, não são estratégias eficazes para este caso.

9. Recomendações Finais

Tendo em vista a necessidade de praticidade, escalabilidade e análise detalhada dos coeficientes com justificativas claras para a escolha de fornecedores de insumos destinados aos estoques dos laboratórios da Dasa, a adoção de um modelo de regressão logística mostra-se plenamente adequada. Resultados ainda mais robustos poderiam ser obtidos mediante a aplicação de validação cruzada durante o processo de modelagem.

O aspecto essencial deste projeto reside na gestão, limpeza e diversidade das variáveis do dataset, considerando suas inter-relações e relevância para o modelo. Esse tipo de algoritmo pode ser integrado a sistemas corporativos, como inventários inteligentes, plataformas ERP (por exemplo, SAP) e até mesmo soluções robóticas.

Além disso, testes e desafios práticos fornecem aprendizado contínuo ao sistema, aprimorando a acurácia e a confiabilidade dos resultados, enquanto reduzem o risco de limitações estruturais ou interpretações equivocadas. Isso possibilita atender a diferentes tipos de demanda de forma descentralizada e eficiente.

Naturalmente, problemas mais complexos poderão exigir, no futuro, a migração de modelos, algoritmos, sistemas ou até mesmo hardware. Entretanto, graças ao conhecimento absorvido, documentado, armazenado e constantemente atualizado, essas futuras implementações tendem a ser mais simples, com uma curva de aprendizado reduzida e, sobretudo, com ganhos significativos em eficiência produtiva e qualidade analítica.

10. Referências

Insumos médicos: o que avaliar antes de comprar

MultiClass Classification Using K-Nearest Neighbours

K-Nearest Neighbors (KNN) Classification with scikit-learn

K-Nearest Neighbors (KNN) in Python

How to add more than 2 variables in K nearest neighbour regression

Understanding Logistic Regression in Python

Lasso and Ridge Regression in Python Tutorial