UNIVERSIDADE PRESBITERIANA MACKENZIE
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
PROGRAMA DE PÓS GRADUAÇÃO EM COMPUTAÇÃO APLICADA

Aprendizagem de Máquina

Projeto Final da Disciplina

Magna Fernandes - RA: 10722096
Renato Godoi - RA: 10406532

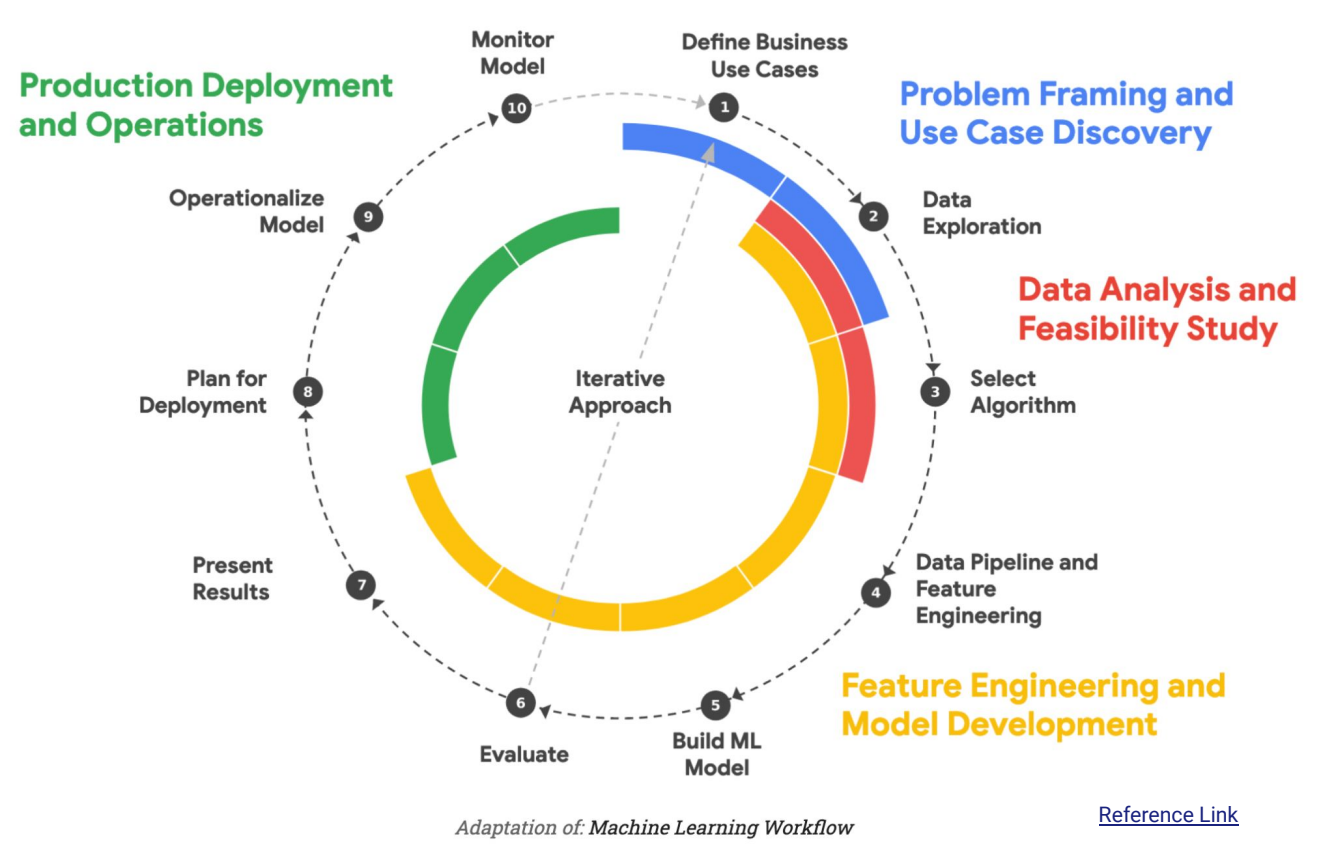
Dezembro, 2024

Contexto sobre o Projeto e Caso de Uso

Análise preditiva de Internações Hospitalares - Um estudo comparativo de algoritmos de Machine Learning no contexto da Polifarmácia

Este projeto trata-se de um estudo **experimental** realizado com o objetivo de **prever hospitalizações associadas a combinações específicas de medicamentos**, utilizando algoritmos de aprendizado de máquina. Investigamos a eficácia de diferentes abordagens, incluindo **Regressão Logística, Árvore de Decisão, Random Forest, Gradient Boosting e Support Vector Machine (SVM)**, para identificar padrões preditivos em um grande conjunto de dados.

ML Lifecycle



Adaptation of: ***Machine Learning Workflow***

Reference Link

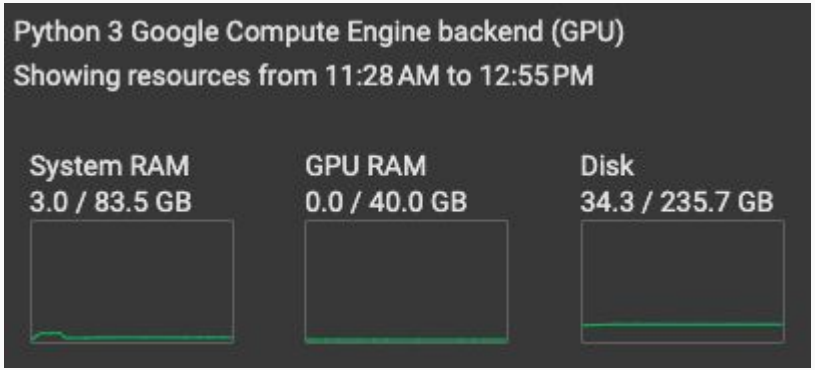
Seleção de Algoritmos

Modelo	Conceito	Prós	Contras	Complexidade de Tempo (Treinamento)	Complexidade de Espaço	Complexidade	Interpretabilidade	Sensibilidade ao Desbalanceamento
Logistic Regression	Modelo linear que estima a probabilidade de uma amostra pertencer a uma classe.	Simples, rápido, fácil de interpretar, poucos hiperparâmetros.	Assume relação linear entre features e target, sensível a outliers e multicolinearidade.	O(n*d) (n= amostras, d= features)	O(d)	Baixa	Alta	Alta
DecisionTreeClassifier	Árvore de decisão que divide recursivamente o espaço de features baseado em critérios de pureza.	Fácil de entender e visualizar, não requer normalização dos dados.	Propensão a overfitting, instável (pequenas mudanças nos dados causam grandes mudanças na árvore).	O(ndlog n)	O(d*n)	Baixa a Média	Alta	Moderada
RandomForestClassifier	Conjunto de árvores de decisão, cada uma treinada em um subconjunto aleatório dos dados.	Robusto, bom desempenho em datasets de alta dimensionalidade, lida bem com dados faltantes.	Difícil de interpretar individualmente as árvores, pode ser computacionalmente caro para datasets muito grandes.	O(ndlog n)	O(d*n)	Média a Alta	Baixa	Baixa
HistGradientBoostingClassifier	Modelo que constrói árvores de decisão sequencialmente, minimizando o erro residual das árvores anteriores.	Alto desempenho, robusto, lida bem com dados faltantes e não-lineares.	Pode ser computacionalmente caro, requer ajuste cuidadoso de hiperparâmetros.	O(ndlog n)	O(d*n)	Média a Alta	Baixa	Baixa
SVC (Support Vector Classifier)	Encontra o hiperplano que melhor separa as classes no espaço de features. Utiliza kernels para lidar com dados não-lineares.	Robusto, bom desempenho em datasets de alta dimensionalidade, funciona bem com dados de alta dimensionalidade	Pode ser computacionalmente caro para datasets muito grandes, escolha do kernel é crucial.	Depende do kernel (pode variar de $O(n^2)$ a $O(n^3)$)	O(n*d)	Média a Alta	Baixa	Moderada

Ambiente

Google Colab Pro

Recursos



Arquitetura do Experimento - Alto Nível

Data Extraction

Biblioteca: `pandas`

Ações:
Carrega o conjunto de dados do arquivo CSV.

Remove linhas duplicadas para garantir a qualidade dos dados.

Amostra 10.000 linhas aleatoriamente do conjunto de dados para acelerar o processamento e teste.

Data Exploration

Bibliotecas: `pandas`, `matplotlib.pyplot` e `seaborn`.

Ações:
Verifica os tipos de dados de cada coluna usando `df.info()`.
Conta o número de IDs de pacientes únicos.
Verifica os valores únicos nas colunas relacionadas a medicamentos (`drug_`).
Analisa a distribuição da variável alvo (`hospit`) usando `value_counts()` e a visualiza com um gráfico de contagem.

Data Transformation

Bibliotecas: `pandas`, `sklearn.preprocessing`,`imblearn.over_sampling`
Ações:
Tratamento de Valores Ausentes
Feature Engineering
Escalonamento de Recursos
Tratamento de Desequilíbrio de Classe

Model Training

Bibliotecas:
`sklearn.model_selection`,
`sklearn.linear_model`,
`sklearn.tree`,
`sklearn.ensemble`,
`sklearn.svm`,
`sklearn.metrics`, `time`.

Ações:
Divide os dados em conjuntos de treinamento, validação e teste.
Define uma lista de modelos de classificação para testar
Define distribuições de parâmetros para cada modelo para usar na otimização de hiperparâmetros.
Usa `RandomizedSearchCV` com validação cruzada
Treina cada modelo com os melhores hiperparâmetros encontrados.

Model evaluation / validation

Bibliotecas:
`sklearn.metrics`
`matplotlib.pyplot`,
`seaborn`.

Ações:
Avalia o desempenho de cada modelo nos conjuntos de treinamento e teste.
Gera um relatório de classificação para cada modelo.
Calcula e plota a matriz de confusão para cada modelo.
Plota curvas ROC para cada modelo para visualizar seu desempenho.
Seleciona o melhor modelo com base na pontuação F1 no conjunto de teste.
Plota gráficos relevantes para o melhor modelo, incluindo matriz de confusão e importância das features.

Abordagem de Desenvolvimento do Modelo

Modelos: `LogisticRegression`, `DecisionTreeClassifier`, `RandomForestClassifier`, `HistGradientBoostingClassifier` e `SVC`.

Tunning de Hiperparâmetros: Um `RandomizedSearchCV` foi usado para otimizar os hiperparâmetros de cada modelo. A principal razão para usar `RandomizedSearchCV` ao invés de `GridSearchCV` é a eficiência computacional. `GridSearchCV` testa todas as combinações possíveis de hiperparâmetros especificadas, o que pode ser muito demorado, especialmente com muitos hiperparâmetros e valores a serem testados. `RandomizedSearchCV`, por outro lado, amostra aleatoriamente um número especificado de combinações, fornecendo uma boa aproximação da melhor configuração de hiperparâmetros em muito menos tempo. Ele é particularmente útil quando o espaço de busca de hiperparâmetros é grande.

Validação Cruzada: A validação cruzada de 5 folds (`cv=5`) foi usada para avaliar o desempenho dos modelos de forma robusta, evitando o overfitting.

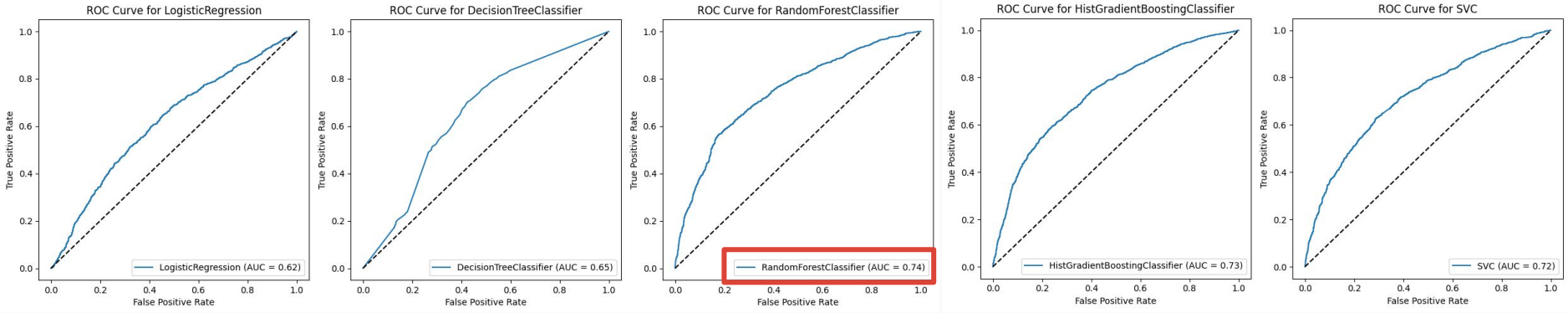
Métricas: As métricas de avaliação utilizadas são `accuracy_score`, `precision_score`, `recall_score`, `f1_score`, `roc_auc_score`, `confusion_matrix` e `classification_report`. O F1-score ponderado (`average='weighted'`) é usado como a métrica principal para a busca de hiperparâmetros. A escolha do F1-score ponderado se justifica no contexto de desbalanço de classes, já que é mais robusto ao comparar modelos em datasets com distribuições desiguais de classes, diferentemente da acurácia, que pode ser enganosa.

Avaliação - Iteração #1

Amostra: 10.000 registros aleatórios

	model	training_time_minutes	train_accuracy	val_accuracy	test_accuracy	train_precision	test_precision	train_recall	test_recall	train_f1	test_f1	val_f1	train_roc_auc	test_roc_auc
0	LogisticRegression	0.045090	0.590475	0.590395	0.588854	0.593471	0.590520	0.590475	0.588854	0.587166	0.586953	0.587071	0.617220	0.614883
1	DecisionTreeClassifier	0.009619	0.618031	0.616915	0.646178	0.620299	0.648104	0.618031	0.646178	0.616222	0.645025	0.615587	0.623443	0.648186
2	RandomForestClassifier	0.106736	0.681985	0.681826	0.687261	0.682715	0.687678	0.681985	0.687261	0.681666	0.687087	0.679976	0.744409	0.746282
3	HistGradientBoostingClassifier	0.062615	0.678321	0.674897	0.679936	0.678705	0.680208	0.678321	0.679936	0.678148	0.679815	0.676130	0.732867	0.738928
4	SVC	7.245940	0.653313	0.653313	0.649045	0.656494	0.652313	0.653313	0.649045	0.651543	0.647151	0.651516	0.711745	0.710732

O modelo RandomForestClassifier alcançou uma precisão de 68.73% e uma pontuação F1 de 0.69 no conjunto de teste.

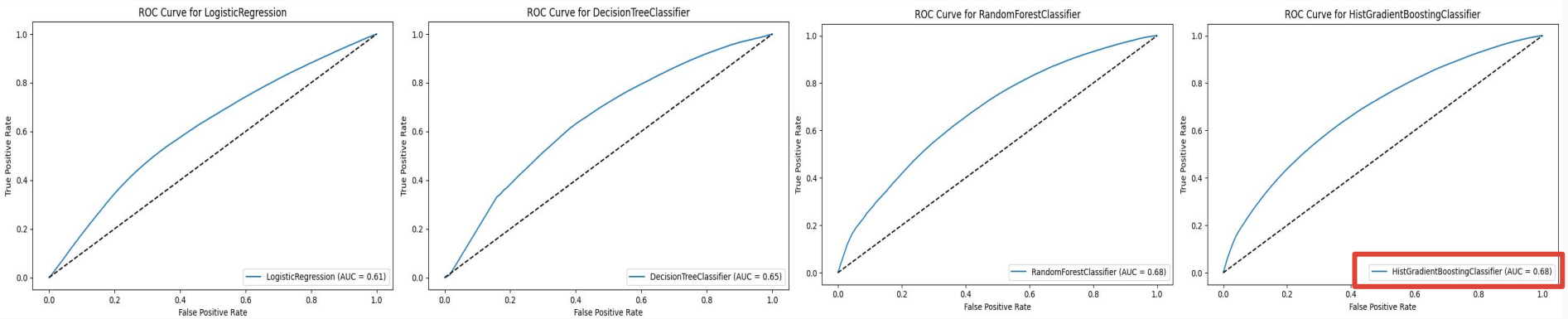


Avaliação - Iteração #2

Amostra: 1.000.000 registros aleatórios

	model	training_time_minutes	train_accuracy	val_accuracy	test_accuracy	train_precision	test_precision	train_recall	test_recall	train_f1	test_f1	val_f1	train_roc_auc	test_roc_auc
0	LogisticRegression	2.456694	0.579696	0.579695	0.579756	0.582593	0.582670	0.579696	0.579756	0.575978	0.576020	0.575975	0.611489	0.612789
1	DecisionTreeClassifier	0.901784	0.611735	0.611753	0.614451	0.612454	0.615106	0.611735	0.614451	0.611114	0.613902	0.611124	0.647132	0.649648
2	RandomForestClassifier	14.119962	0.627283	0.627164	0.628682	0.627314	0.628701	0.627283	0.628682	0.627260	0.628668	0.627140	0.677629	0.679561
3	HistGradientBoostingClassifier	3.051518	0.629356	0.629606	0.630928	0.629881	0.631695	0.629356	0.630928	0.628980	0.630389	0.628897	0.679742	0.681329

Métricas relevantes para a avaliação:
- `test_accuracy`, `test_precision`, `test_recall`, `test_f1`: Métricas de desempenho no conjunto de teste.
- `val_f1`: Métrica de validação cruzada, mostrando o desempenho médio em diferentes partições dos dados.
- `training_time`: Tempo gasto para treinar o modelo com os parâmetros escolhidos.



Futuras Iterações

Iteração #3: Aprimoramento das técnicas de tratamento de dados e Feature Engineering

- Criar novas features representando interações entre as diferentes drogas
- Aplicar transformações não-lineares (explorar transformações como polinômios ou splines para capturar relações não-lineares entre as variáveis e a variável target).
- Explorar técnicas mais avançadas para tratamento de valores ausentes:
    - Imputação com modelos: Use um modelo de aprendizado de máquina para prever os valores ausentes com base nas outras variáveis.
    - Imputação por KNN: Utilize o método dos k-vizinhos mais próximos para imputar valores ausentes baseados nos dados mais próximos.
    - Análise de Dados Ausentes: Investigue os padrões de dados ausentes. Eles são aleatórios ou há algum motivo subjacente? Isso pode informar a melhor estratégia de imputação.
- Considerar aplicar outras técnicas de escalonamento de recursos:
    - MinMaxScaler: Escala os dados para um intervalo específico (por exemplo, 0 a 1).
    - RobustScaler: Robusto a outliers, uma boa opção se os seus dados tiverem outliers significativos.
    - Avaliação Comparativa: Compare o desempenho dos modelos com diferentes métodos de escalonamento para encontrar a melhor opção.

Iteração #4: Melhoria da Seleção e Avaliação de Modelos

- Validação Cruzada Mais Rigorosa:
    - Aumente o número de folds: experimentar com um número maior de folds.
    - Métodos de validação cruzada: Explorar outros métodos como validação cruzada leave-one-out ou validação cruzada repetida.
- Tunning de Hiperparâmetros:
    - `GridSearchCV`: Avaliar novamente o uso `GridSearchCV` para uma busca exaustiva de hiperparâmetros, para acelerar o processo, executar a busca de hiperparâmetros em paralelo usando `n_jobs=-1`.
    - Otimização Bayesiana: Para espaços de busca maiores e mais complexos, explore algoritmos de otimização bayesiana para uma busca mais eficiente.
- Além das métricas já usadas explorar AUC (área sob a curva) e curva de precisão-recall, útil para avaliar o desempenho em diferentes limiares de classificação

Desafios enfrentados

- Falta de informações mais detalhadas sobre o dataset, já que o mesmo era simulado
  - Não tinha informações sobre prescrição ou não
  - Poderia conter mais informações demográficas sobre o paciente
- Foi necessário buscar conhecimento sobre `RandomizedSearchCV` no lugar de `GridSearchCV` para validação cruzada.
- Só após realizar N iterações, pensamos em armazenar o resultado de todas as iterações para compará-las.
- Capacidade computacional.

GitHub e Medium Post

Download Dataset Link

Draft Link