Universidade Federal de Goiás
Instituto de informática
Profa Nádia Félix Felipe da Silva

Relatório da 1ª Competição de Inteligência Computacional

Aluno: Otávio Pontes de Oliveira Salomão
Disciplina: Inteligência Computacional

Dezembro
2022

Conteúdo

1 Resumo

O problema proposto na 1ª Competição de Inteligência Artificial diz respeito a gama de problemas denominados de problemas de classificação. No caso, a ideia é classificar se um requisição para o plano de saúde deve ser negado ou autorizado pelo operador do plano.

É um problema muito relevante atualmente, principalmente para grupos de risco, como idosos ou pessoas com doenças crônicas e/ou graves; mas também para grande parte da população. A proposta de se construir um modelo que consiga prever esse tipo de dado seria de grande valia, podendo inclusive ser o carro chefe de uma nova startup.

Para realizar essa tarefa, foi proposto que se utiliza-se dos conhecimentos adquiridos até então durante as aulas de Inteligência Computacional, podendo extrapolar no que fizer sentido para o problema.

2 Descrição do Conjunto de dados

O Conjunto de Dados é composto por dados preenchidos por um colaborador da prestadora (hospital, clínica, laboratório ou consultório) requisitando cobertura das despesas de produtos e serviços prestados ao cliente (beneficiário do plano). Os dados disponibilizados foram anonimizados e fornecidos por uma operadora de plano de saúde (dados reais).

Foram disponibilizados dois conjuntos de dados em formato CSV, um para treino e outro para teste do modelo. O conjunto de Treino possui 32 colunas e aproximadamente 227.122 entradas. Já o de teste conta com as 31 primeiras colunas iguais a do conjunto de teste, excluindo-se portanto a última coluna, de resultado (Autorizado ou Negado); e possui cerca de 186.145 entradas.

Quanto aos tipos de dados, estão separados em dados numéricos, como quantidade de dias e datas, e dados textuais, como descrições e códigos de doenças. Além disso, na primeira coluna há os IDs correspondentes para cada entrada.

Os dados numéricos escolhidos para o modelo foram: NR_SEQ_REQUISICAO (Ordem de prioridade da requisição), NR_SEQ_ITEM (Ordem de prioridade de item), DT_REQUISICAO (Data que a requisição foi feita), DT_NASCIMENTO (Data de nascimento do solicitante), QT_SOLICITADA (Quantidade Solicitada), QT_TEMPO_DOENCA (Tempo que o solicitante sofre com a doença), QT_DIA_SOLICITADO (Previsão de dias de internação), CD_ITEM (Código de item na tabela do plano), CD_GUIA_REFERENCIA (Código de referência do guia na tabela do plano).

Já os dados textuais, ou categóricos, escolhidos foram: DS_REGIME_INTERNACAO (Tipo de internação), DS_CLASSE (Tipo de classe da solicitação), DS_TIPO_ACOMODACAO (Tipo de acomodação para internação), DS_TIPO_INTERNACAO (Tipo de internação), DS_UNIDADE_TEMPO_DOENCA (Dias, Semanas, Meses ou Anos), DS_CARATER_ATENDIMENTO (Procedimento Urgente ou Eletivo), DS_TIPO_ATENDIMENTO (Tipo de atendimento necessário), DS_INDICACAO_ACIDENTE (Houve acidente ou não), DS_GRUPO (Grupo em que o procedimento está classificado), DS_CBO (Ocupação do profissional que fará o atendimento), DS_SUBGRUPO (Subgrupo em que o procedimento está classificado).

3 Descrição de atividades

3.1 Análise e Compreensão do Domínio do Problema

A primeira etapa da atividade foi buscar compreender o domínio do problema, os dados, colunas e a relação disso tudo para gerar o resultado, que no caso é autorizar ou não autorizar a requisição. Para fazer essa análise é necessário tentar se colocar no lugar do analista ou médico que faria essa análise na vida real a fim de garantir uma maior assertividade e coerência, o que não é uma tarefa simples.

Após essa primeira análise, é preciso escolher quais dados vão entrar no modelo, o que já foi descrito na seção de Descrição do Conjunto de Dados.

3.2 Pré-Processamento dos Dados

Nesta etapa, é preciso decidir como prosseguir com algumas adversidades que aparecem em conjuntos de dados da vida real, como dados faltantes ou corrompidos. Há diversas abordagens disponíveis, em que as mais comuns são preencher esses valores problemáticos com um valor coringa (como '0' ou 0), fazer a média ou mediana dos valores e preencher os problemáticos com esse valor e apagar completamente a linha com os dados problemáticos.

A abordagem escolhida para este problema foi a de preencher com o valor coringa, devido a grande quantidade de colunas de valores categóricos de conteúdo muito diverso.

3.3 Configuração do Conjunto de Treino

A configuração do conjunto de treino é o momento de fazer o download do arquivo CSV disponibilizado, separar as colunas a serem usadas no modelo e setar o X (Dataframe com todas as colunas menos a de resultado) e o y (Dataframe com a coluna de resultado).

Ademais é preciso fazer o encoding de X. Há diversos tipos de encoders disponíveis, como o OneHotEncoder, o MinMaxScaler e o StandardScaler, sendo o OneHotEncoder e o StandardScaler de uso mais geral e o MinMaxScaler de uso normalizante de dados. Aqui será usado o OneHotEncoder

para as colunas de atributos categóricos e o MinMaxScaler para colunas de atributos numéricos.

3.4 Separação dos Dados de Treino

Neste passo, é preciso separar os dados de treino em um conjunto de treino e teste. Isso é importante para fazer um posterior avaliação dos Algoritmos e Classificadores e escolher o mais adequado para o problema.

Quanto a separação em si não há uma proporção considerada como regra, mas é sempre importante deixar uma quantidade maior para o treinamento do modelo. As duas proporções mais sugeridas pela literatura são o 70/30 e o 80/20. Neste problema será adotada a proporção 80/20.

3.5 Configuração e Execução do Classificador Naive Bayes

O algoritmo Naive Bayes é baseado no teorema de Bayes, bastante usado em problemas de aprendizagem supervisionada, como a classificação de textos. O "naive"(ou ingênuo) no nome se dá pelo fato do algoritmo assumir que as features possuem uma certa independência entre si, ou seja, todas tem o mesmo nível de importância dentro do modelo.

O classificador usado nesta etapa foi o Multinomial Naive Bayes, modelo disponível na biblioteca do Scikitlearn, uma versão que faz uso da distribuição multinomial para cada feature. Foi feito o treinamento do modelo com a porcentagem de dados de treino e posteriormente foi testado e avaliado, tendo como principal métrica a acurácia do f1-score.

3.6 Configuração e Execução do Classificador KNN

KNN(K — Nearest Neighbors) é um algoritmo de aprendizagem supervisionada usado no campo de data mining e machine learning. Ele é um classificador onde o aprendizado é baseado no quão similar é um dado do outro. O treinamento é formado por vetores de n dimensões.

A configuração usada para esse classificador foi uma instância do modelo disponível na biblioteca do Scikitlearn, com o K = 5, como sugerido pela

literatura. Foi feito o treinamento do modelo com a porcentagem de dados de treino e posteriormente foi testado e avaliado, tendo como principal métrica a acurácia do f1-score.

3.7 Configuração e Execução do Classificador Decision Tree

Árvore de Decisão é uma técnica de aprendizado de máquina supervisionado amplamente utilizada em problemas de classificação (embora também pode ser usada em problemas de regressão), sendo nesse caso denominadas de Classification and Regression Trees (CART). Com um nome sugestivo, estas são estruturas do tipo árvore, compostas por nós, ramos e folhas (ou nós terminais).

A configuração usada foi a padrão da biblioteca do Scikitlearn. Foi feito o treinamento do modelo com a porcentagem de dados de treino e posteriormente foi testado e avaliado, tendo como principal métrica a acurácia do f1-score.

3.8 Configuração e Execução do Classificador Random Forest

O algoritmo do Random Forest cria várias árvores de decisão, de maneira aleatória, formando o que podemos enxergar como uma floresta, onde cada árvore será utilizada na escolha do resultado final, em uma espécie de votação. Ele faz parte dos métodos Ensemble, os quais são construídos da mesma forma que algoritmos mais básicos, como regressão linear, árvore de decisão ou knn, mas com a combinação de diferentes modelos para se obter um único resultado.

A configuração usada foi a padrão da biblioteca do Scikitlearn, com 100 estimadores e estado aleatório sendo 0. Foi feito o treinamento do modelo com a porcentagem de dados de treino e posteriormente foi testado e avaliado, tendo como principal métrica a acurácia do f1-score.

Como este algoritmo tem uma natureza mais plural e com uma precisão bem próxima da árvore de decisão, será ele o modelo a ser utilizado em produção.

3.9 Validação Cruzada

Cross Validation (ou Validação Cruzada) é uma técnica muito utilizada para avaliação de desempenho de modelos de machine learning. Consiste em particionar os dados em conjuntos(partes), onde um conjunto é utilizado para treino e outro conjunto é utilizado para teste e avaliação do desempenho do modelo. A utilização do CV tem altas chances de detectar se o seu modelo está sobre-ajustado aos seus dados de treinamento, ou seja, sofrendo overfitting.

Um dos tipos de Validação Cruzada mais populares é o K-fold, que consiste em dividir a base de dados de forma aleatória em K subconjuntos com aproximadamente a mesma quantidade de amostras em cada um deles. A cada iteração, treino e teste, um conjunto formado por K-1 subconjuntos são utilizados para treinamento e o subconjunto restante será utilizado para teste gerando um resultado de métrica para avaliação (ex: acurácia). Esse processo garante que cada subconjunto será utilizado para teste em algum momento da avaliação do modelo.

No problema em questão, foi utilizado um K igual a 10 e, devido ao quantidade massiva de entradas e grande quantidade de tempo necessário, escolheu-se fazer essa validação apenas com o Classificador Random Forest, o qual será usado para teste em produção posteriormente. Utilizou-se o método cross_val_score da biblioteca do Scikitlearn para avaliar, gerar uma média e printar os resultados.

Em seguida, tentou-se utilizar o método cross_val_predict, também da biblioteca do Scikitlearn, para tentar prever os resultados, passando o algoritmo desejado (Random Forest), o conjunto de treino e teste e o K.

3.10 Configuração do Conjunto de Teste

A configuração do conjunto de teste é o momento de fazer o download do arquivo CSV disponibilizado, separar as colunas a serem usadas no modelo e setar o X (Dataframe com todas as colunas menos a de resultado).

Ademais é preciso fazer o encoding de X. Há diversos tipos de encoders disponíveis, como o OneHotEncoder, o MinMaxScaler e o StandardScaler, sendo o OneHotEncoder e o StandardScaler de uso mais geral e o MinMaxScaler de uso normalizante de dados. Aqui será usado o OneHotEncoder

para as colunas de atributos categóricos e o MinMaxScaler para colunas de atributos numéricos.

3.11 Configuração e Execução do Classificador Random Forest para Produção

Nesta etapa, uma nova instância do Classificador Random Forest foi criada com as mesma configurações da anterior. Contudo, agora o treinamento será realizado com todo o conjunto de treino, sem dividi-lo. O intuito desta fase é maximizar o aprendizado do modelo fornecendo mais dados.

3.12 Previsão com os Dados de Teste e Submissão

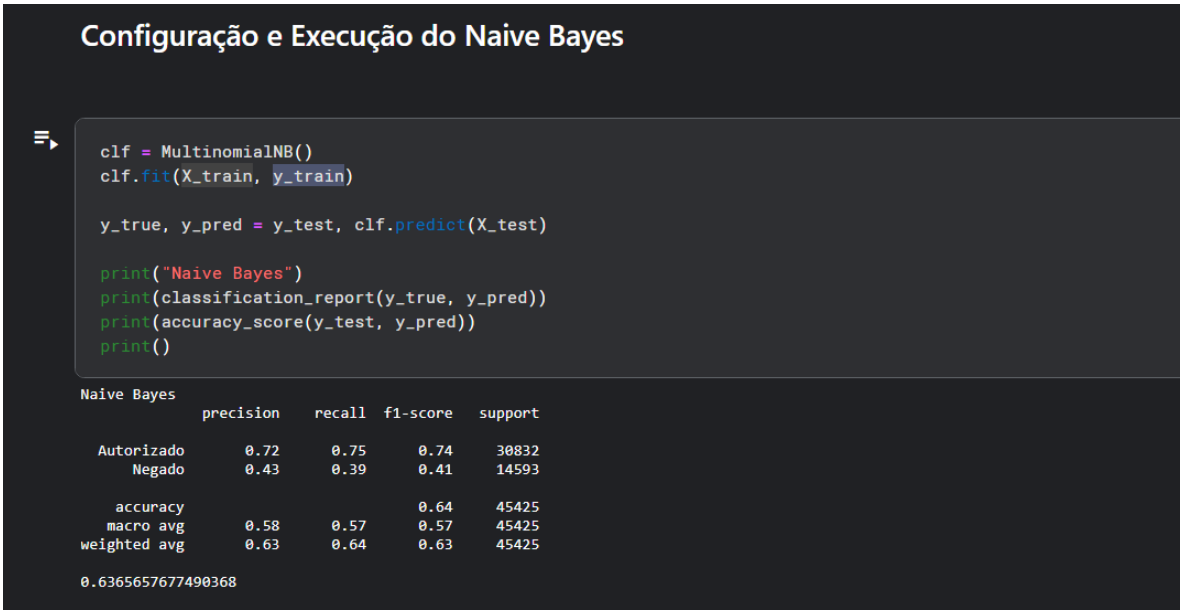
Com o classificador de produção configurado, chegou a hora dele fazer as previsões da classe desejada, ou seja, se a requisição foi Autorizada ou Negada.

Feita a previsão, é setado um conjunto de dados com essas previsões, o qual será usado para avaliar a pontuação do modelo. Primeiramente será liberado cerca de 30% do conjunto de teste para comparação e, próximo a data final, será liberado os outros 70% para consolidar a pontuação e classificação dos modelos.

4 Análise dos Resultados

4.1 Naive Bayes

O algoritmo do Naive Bayes obteve como resultado a acurácia de 64%, a mais baixa de todos os modelos. Esse resultado se explica pela simplicidade do algoritmo ao lidar com este problema, que é relativamente complexo para ele.



4.2 KNN

O algoritmo do KNN obteve como resultado a acurácia de 79%, um bom número porém ainda inferior aos modelos seguintes.

Configuração e Execução do KNN

[30]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)

y_true, y_pred = y_test, knn.predict(X_test)

print("KNN")
print(classification_report(y_true, y_pred))
print(accuracy_score(y_test, y_pred))
print()
```

```
KNN
              precision    recall  f1-score   support

  Autorizado       0.83      0.87      0.85     30832
      Negado       0.69      0.62      0.65     14593

    accuracy                           0.79     45425
   macro avg       0.76      0.74      0.75     45425
weighted avg       0.78      0.79      0.79     45425

0.7887947165657677
```

4.3 Decision Tree

O algoritmo do Decision Tree obteve como resultado a acurácia de 85%, a melhor de todos os modelos testados no problema. Todos as demais métricas como a precisão, o recall e o f1-score foram os mais elevados também.

Configuração e Execução do Decision Tree Classifier

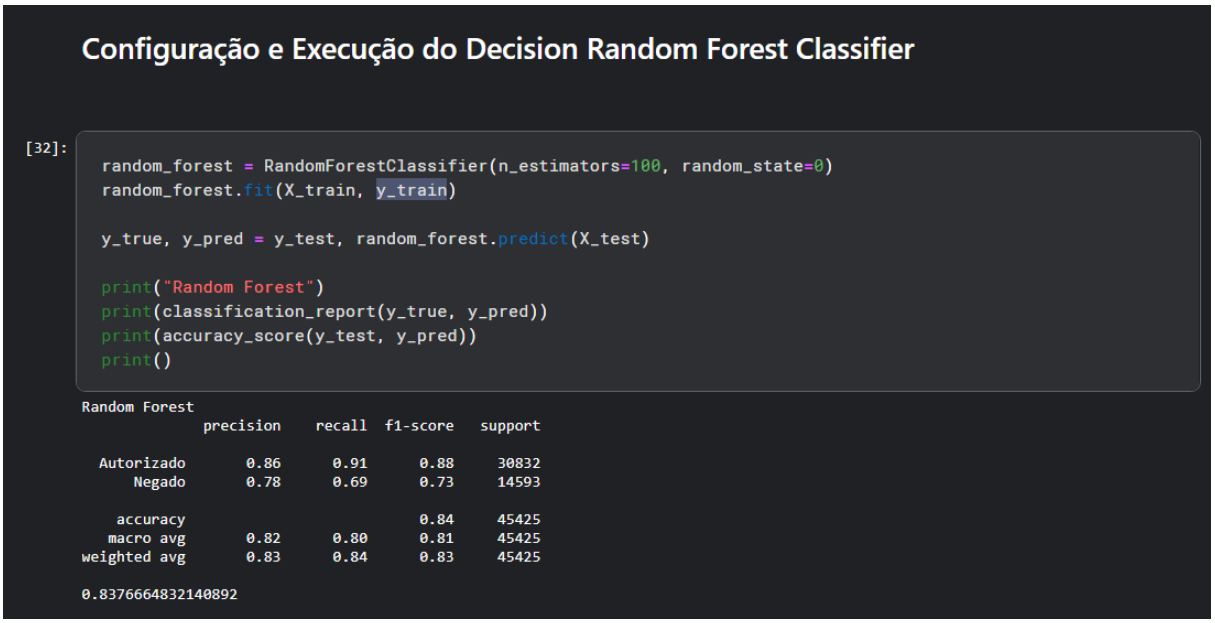
+ Code + Markdown

[31]:
```
tree = DecisionTreeClassifier()
tree.fit(X_train,y_train)

y_true, y_pred = y_test, tree.predict(X_test)

print("Decision Tree")
print(classification_report(y_true, y_pred))
print(accuracy_score(y_test, y_pred))
print()
```

```
Decision Tree
              precision    recall  f1-score   support

  Autorizado       0.89      0.89      0.89     30832
      Negado       0.76      0.77      0.77     14593

    accuracy                           0.85     45425
   macro avg       0.83      0.83      0.83     45425
weighted avg       0.85      0.85      0.85     45425

0.8484534947716016
```

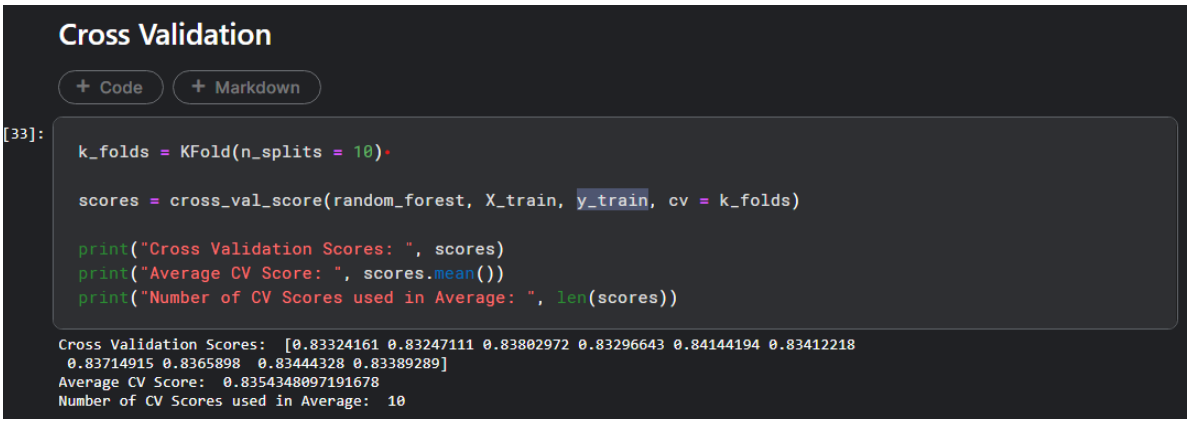
4.4 Random Forest

O algoritmo do Random Forest obteve como resultado a acurácia de 84%, chegando muito próximo do resultado do Árvore de Decisões. As demais métricas também ficaram um pouco abaixo, mas ainda assim bem próximas as da árvore.



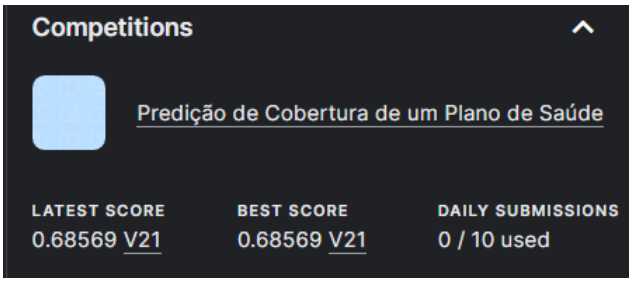
4.5 Cross Validation

O algoritmo de Validação Cruzada obteve uma média de 83,5% de acurácia, ou seja, na teoria o modelo com a Random Forest está aprendendo.



4.6 Resultado com 30% do Conjunto de teste

Com cerca de 30% de todo o conjunto de teste para produção disponível, o modelo foi capaz de obter a pontuação de 68,56%.



5 Trabalhos Futuros

Como trabalhos futuros, seria interessante o estudo para aprimoramento desse modelo desenvolvido até então. Isso poderia ser feito por meio de novos modelos, técnicas, refinamento da análise do domínio do problema e maior tempo de prática. É importante também avaliar os erros que foram cometidos, principalmente os mais graves, e evitar comete-los nas possíveis futuras versões desse modelo.

Ademais, seria interessante a publicação de artigos discorrendo sobre este problema, a fim de apresentar outros pontos de vista e buscar possíveis parceiros para continuidade do projeto, além de um possível financiamento.

Bibliografia

RABELLO, E. B. Cross Validation: Avaliando seu modelo de Machine Learning. Disponível em: https://medium.com/@edubrazrabello/cross-validation-avaliando-seu-modelo-de-machine-learning-1fb70df15b78. Acesso em: 03/12/2022.

K-Nearest Neighbor(KNN) Algorithm for Machine Learning. Disponível em: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning. Acesso em: 02/12/2022.

Scikit Learn User Guide. Disponível em: https://scikit-learn.org/stable/user_guide.html. Acesso em: 02/12/2022.

FAYYAD, U. From Data Mining to Knowledge Discovery in Databases. Disponível em: https://ojs.aaai.org//index.php/aimagazine/article/view/1230. Acesso em: 01/12/2022.