Inteligência Computacional - Relatório Final

Filipe Santos Pacheco Prates - 1160113311

June 2021

1 Introdução

O código que acompanha o trabalho pode ser encontrado em
https://github.com/FilipePrates/COC361-Intelig-ncia-Computacional/

1.1 Descrição do Problema

Uma rede de hotéis possui informações de reservas passadas. Nesse projeto usarei diferentes métodos de Inteligência Computacional para tentar predizer se um cliente irá cumprir ou não com uma reserva futura, dado certas características da reserva. A rede de hotels possui milhares de clientes e reservas todos os meses, e seria de extrema importância financeira para a empresa conseguir determinar o quão seguro é uma reserva, e como melhorar esse fato.

2 Dataset e Tecnologia

2.1 Descrição dos Dados

O dataset com os dados são originários de
https://www.sciencedirect.com/science/article/pii/S2352340918315191
escrito por Nuno Antonio, Ana Almeida, and Luis Nunes para "Data in Brief, Volume 22, Fevereiro 2019". E disponibilizado no Kaggle.
https://www.kaggle.com/jessemostipak/hotel-booking-demand
São 32 colunas contendo as informações referentes à 119390 reservas de uma rede de hotéis.

Temos diversas informações, de diversos tipos, dados Geográficos, sobre os clientes, sobre os hotéis e temporais sobre a reserva em si. Todas as colunas podem ser sub-divididas entre as Categóricas, que serão tratadas com alguma forma de codificação, as Numéricas, que serão normalizadas, e aquelas que serão removidas do dataset para uma melhor experiência para os modelos ou por conter informações que não seriam disponíveis caso usadas para prever o resultado de uma reserva futura.

Podemos já perceber que algumas dessas colunas contém informações diretas sobre o resultado da reserva, se foi cancela ou não. A primeira é a própria coluna

```
In [64]: list(df.columns)

Out[64]: ['hotel',
          'is_canceled',
          'lead_time',
          'arrival_date_year',
          'arrival_date_month',
          'arrival_date_week_number',
          'arrival_date_day_of_month',
          'stays_in_weekend_nights',
          'stays_in_week_nights',
          'adults',
          'children',
          'babies',
          'meal',
          'country',
          'market_segment',
          'distribution_channel',
          'is_repeated_guest',
          'previous_cancellations',
          'previous_bookings_not_canceled',
          'reserved_room_type',
          'assigned_room_type',
          'booking_changes',
          'deposit_type',
          'agent',
          'company',
          'days_in_waiting_list',
          'customer_type',
          'adr',
          'required_car_parking_spaces',
          'total_of_special_requests',
          'reservation_status',
          'reservation_status_date']
```

Figure 1: Colunas

"is canceled", que é removida do dataset e será utilizada como Y, o resultado que queremos prever. Além dela, as colunas 'reservation status' e 'reservation status date' são perfeitamente correlatas com a coluna Y, e se não removidas do dataset, os modelos conseguem perfeita classificação.

Os dados em forma geral estavam bem tratados já de início, com poucos valores nulos, e nenhum erro de digitação nas variáveis categóricas.

Conseguimos observar que a grande maioria das colunas categóricas nos passam informações úteis para a predição e serão aceitas para os dataset que será entregado aos modelos, porém algumas, por possuir muitas categorias (incluindo muitas pouco representadas), causariam um dataset muito esparso caso codificada de maneira ingênue.

Esse foi o caso da categoria 'country', que contém a informação referente ao

Figure 2: Relação das colunas de 'Hotel', 'Mês de Chegada', 'Comida', e 'País' na taxa de cancelamento

país do cliente. Temos em todo o dataset quase 200 diferentes países, porém a maioria esmagadora das reservas foram feitas por moradores de um número muito mais reduzido de países. Determinei que irei codificar apenas os primeiros 20 países, e todos as outras reservas seriam agrupadas em uma única categoria 'Outros'.

Por último normalizamos todos os valores numéricos para serem centralizados em zero e com desvio padrão um. Dessa maneira nenhuma dimensão possui prioridade sobre outra, e os modelos podem receber as informações contidas no dataset da melhor maneira.

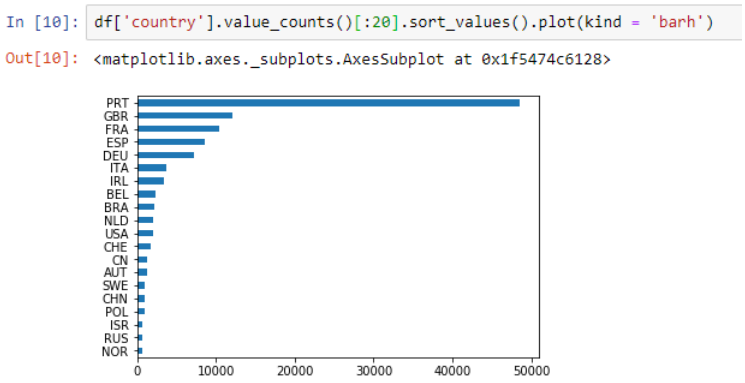

Figure 3: Países mais representados nas reservas

```
In [70]: sns.heatmap(df.corr())
Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x26b0c78fb38>
```

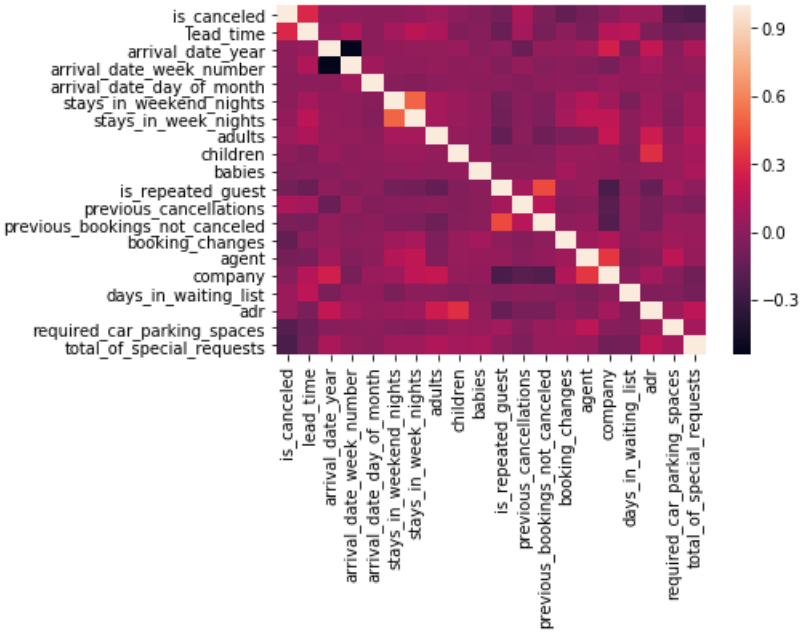

Figure 4: Mapa de calor de correlação entre variáveis.

2.2 Apresentação da tecnologia

3 Metodologia

3.1 Descrição Teórica

Após o EDA e a preparação dos dados, eles estão prontos para os modelos. Primeiramente precisamos separar nosso X e y em para usarmos para o treinamento e teste dos modelos. E também para garantir a generalização deles, usamos a validação cruzada de k-fold. (no caso k=5).

Primeiramente tentamos simples modelos lineares de classificação. Os usados para esse problema de foram a Regressão Logística e o Naive Bayes.

3.2 Regressão Logística - Teoria

A intuição por trás da regressão logística é bastante simples: em vez de acharmos a reta que melhor se ajusta aos dados, vamos achar uma curva em formato de 'S' que melhor se ajusta aos dados. E ela separará o espaço entre as categorias.

Na prática a diferença na Regressão Logística é uma Regressão linear que tenta simplesmente achar a melhor reta que encaixa nos dados é que a Logística segue:

$$y = \sigma(Xw + \epsilon)$$

onde

$$\sigma()$$

é uma função sigmóide. Assim temos uma maneira simples de gerar a probabilidade de uma amostra estar em um grupo, ou em outro, determinando algum limiar entre elas, y = 0,5 por exemplo.

3.3 Naive Bayes - Teoria

Utiliza o famoso teorema de Bayes sobre probabilidades, é um modelo iterativo que vai atualizando a probabilidade de cada classificação. É "Naive" ou "ingênuo" pois não utiliza a correlação entre variáveis. A ideia do modelo é calcular uma probabilidade a posteriori de um resultado dado uma variável, multiplicando a probabilidade a priori de tal resultado com a probabilidade de tal variável dado o resultado. Assim atualizamos a probabilidade de tal resultado.

Depois de estudarmos os resultados com os modelos lineares, aplicaremos os modelos não-lineares, capazes de maior generalização e modelar categorizações mais complexas. Em termos de modelos não-lineares trataremos com a Árvore de Decisão, Floresta Aleatória (Random Forest), Gradient Boosting, e por ultimo algumas arquiteturas de Redes Neurais.

3.4 Árvore de Decisão - Teoria

O modelo de árvore de decisão consistem em achar as separações (splits) mais eficientes respondendo condicionais nas variáveis, visando gerar uma árvore que indicará qual categoria cada entrada pertence.

A ideia é começar com um nó Raíz que indica todo o Set de entrada, então observa de todas as colunas, qual a que contém maior 'quantidade de informação', e cria uma separação em tal coluna. Assim repetindo considerando sempre atributos nunca utilizados para uma separação antes.

O resultado é uma árvore que indica, assumindo os atributos da entrada, um caminho que leva à classificação.

3.5 Random Forest - Teoria

É um modelo do estilo ensemble, ou seja, uma aglomeração de outros modelos, no caso Árvores de Decisões. Existem diferentes maneiras de implementar e usar algoritmos da família da Random Forest, como por exemplo simplesmente tomar uma média de cada árvore da floresta, e tomar como resultado, ou algo mais complexo.

Em geral árvores muito profundas tendem à ter um overfit aos dados os conjuntos de treino, ou seja, tem resultados específicos demais para a base de treinamento que não é generalizável ao tentar prever amostras diferentes das que o modelo recebeu. RandomForest é uma maneira de diluir os viéses de cada árvore de decisão, com o objetivo de reduzir a variância e obter um resultado mais confiável.

3.6 Gradient Boosting - Teoria

Também é um modelo do estilo ensemble, uma maneira iterativa de combinar e melhorar outros modelos mais fracos em um mais poderoso (Boosting), Tomamos primeiro um modelo simples, e medimos como ele performou comparando com o valor real do resultado (uma maneira simples de ver a performance é utilizando o erro médio quadrático do previsto com o real. Então é criado um outro modelo que tenta gerar um h() tal que:

$$F_{m+1}(x) = F_m(x) + h_m(x) = y$$

Assim a cada iteração o erro é minimizado e temos um modelo mais poderoso que o anterior.

3.7 Support Vector Machines (SVMs) - Teoria

Uma máquina de vetores de suporte é um conjunto de algoritmos que, quando usado para classificação, separam o espaço das amostras, caracterizando-as no processo., O SVM padrão toma como entrada um conjunto de Dados, e cria bordas entre as amostras de cada categoria, de maneira que elas sejam divididas por um espaço claro que seja tão amplo quanto possível. A máquina acha o melhor hiperplano para dividir as categorias, de maneira que as amostras de cada categoria estão o mais longe dessa borda possível, criando uma larga zona tampão entre as categorias.

3.8 Redes Neurais - Teoria

Utilizarei Redes Neurais Feed-Foward. Tal modelo consiste em nós (neurônios) conectados entre si, cada nó recebe uma série de valores de input de conexões com outros nós (a série de atributos da amostra na primeira camada), possui uma função interna que dependendo dos inputs retorna um valor que será seu output, e esse output é multiplicado por um valor de peso da conexão que é propagado para os neurônios nas camadas posteriores.

Assumindo uma rede neural treinada, ao receber um vetor com os atributos na primeira camada, o simples ato de propagar para frente os valores até a camada final, que possui só um nó, irá gerar o resultado. Por exemplo se o output final ¿ 0,5, então assumiriamos aquela reserva com uma alta probabilidade de ser cancelada.

Para alcançar tal estado usamos do método de Backpropagation. Simplesmente, para nossa base de dados de treino, fazemos o processo de propagar para

frente cada entrada numa rede com os pesos aleatórios, e vemos o erro médio quadrático da saida com o seu real valor. Então utilizado de tal erro podemos 'caminhar para trás' na rede atualizando os pesos das conexões de maneira para minimizar o erro (descendo pelo gradiente).

Para cada uma das entradas, os pesos da rede são atualizados e temos uma rede melhor capaz de prever os resultados. Ficamos então de olho durante o desenvolvimento do treinamento para observar indícios de quando que a rede está começando a ter um overfit nos dados de treinamento (usando o subset de validação), e paramos de treinar na hora ótima.

4 Resultados

4.1 Descrição Validação Cruzada

A validação cruzada é uma técnica para garantir que o modelo consegue generalizar para aplicações práticas. Com ela podemos avaliar a capacidade de generalização de um modelo, a partir de um conjunto de dados. Busca-se então estimar o quão preciso é este modelo e o seu desempenho para um novo conjunto de dados. O conceito central da técnica de validação cruzada é o particionamento do conjunto de dados em subconjuntos mutuamente exclusivos, e posteriormente, o uso de alguns destes subconjuntos para a estimação dos parâmetros do modelo (dados de treinamento), sendo os subconjuntos restantes (dados de validação ou de teste) empregados na validação do modelo. Assim diminuimos a possibilidade de overfit e podemos tomar decisões mais informadas sobre os hiperparâmetros ótimos dos modelos.

4.2 Resultado dos modelos lineares

4.2.1 Regressão Logística - Resultados

AUC	f1 score
0.907	0.821

4.2.2 Naive Bayes - Resultados

AUC	f1 score
0.582	0.403

4.3 Resultado dos modelos não-lineares

4.3.1 Árvore de Decisão - Resultados

AUC	f1 score
0.853	0.860

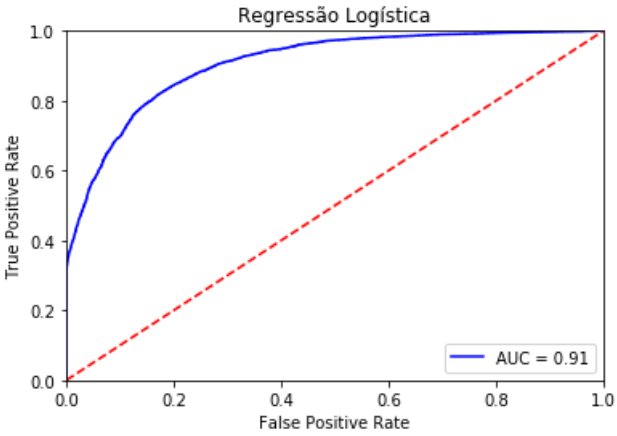

Figure 5: Regressão Logística.

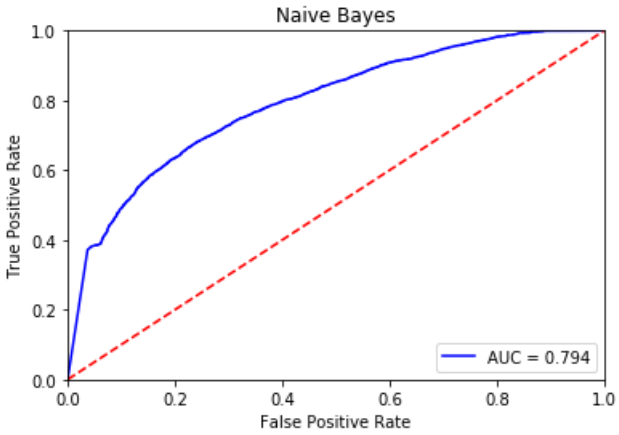

Figure 6: Naive Bayes - Podemos observar que tive dificuldades com esse modelo, que performou abaixo do esperado, não consegui determinar o motivo

4.3.2 Random Forest - Resultados

AUC	f1 score
0.945	0.879

4.3.3 Gradient Boosting - Resultados

AUC	f1 score
0.929	0.851

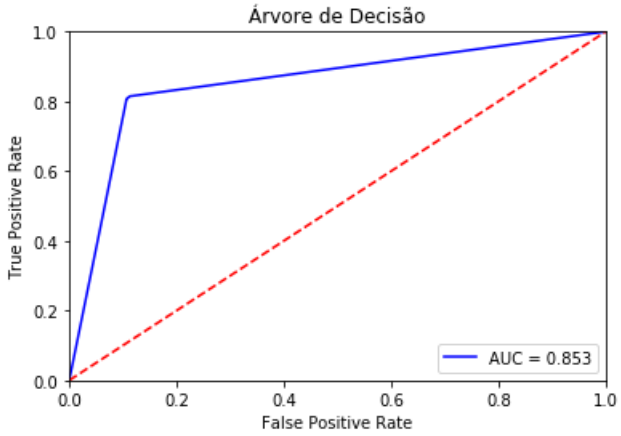

Figure 7: Árvore de Decisão

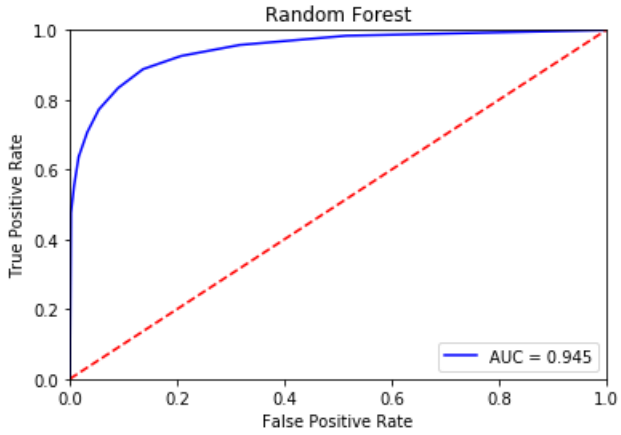

Figure 8: Random Forest

4.3.4 SVM - Resultados

AUC	f1 score
0.558	0.533

4.3.5 svm1

AUC	f1 score
0.940	0.870

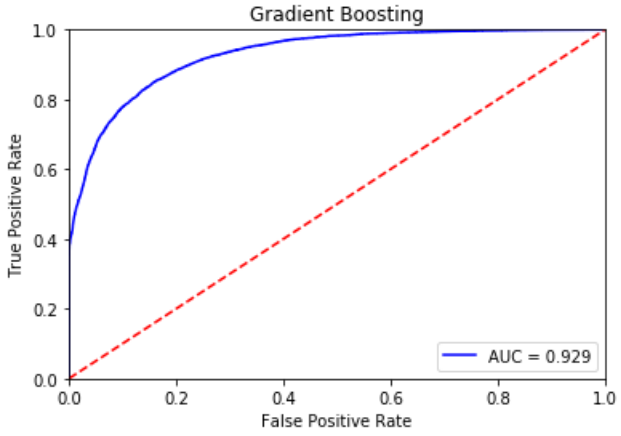

Figure 9: Gradient Boosting

Figure 10: SVM, com maxiter=10000, gamma='auto', probability='False'

Figure 11: svm2

4.3.6 Rede Neural FF 133-60-13-1 - Resultados

AUC	f1 score
0.940	0.872

$\text{cross}_v al_s core 0.940$	0.872	0.872	0.872	0.872

4.3.7 Rede Neural FF 80-60-1 - Resultados

AUC	f1 score
0.941	0.872

4.3.8 SVM

Tive problemas na criação de modelos de SVM. Por algum motivos todos as implementações que tentei estavam resultando em loops infinitos, e mesmo colocando um limite forçado de iterações máximas, o resultado era muito insatisfatório e demorava muito. Tentei diversos hiperparametros, mas não consegui nada que fosse melhor que um modelo aleatório. Acho que pode ter sido pela quantidade grande de colunas no dataset, que cria um hiperespaço grande que deixa a tarefa de encontrar a melhor divisão entre as categorias mais difícil. Teoricamente o escalar dos dados para [-1,1] deveria ajudar o algoritmo à convergir mais rapidamente, porém não consegui resultados.

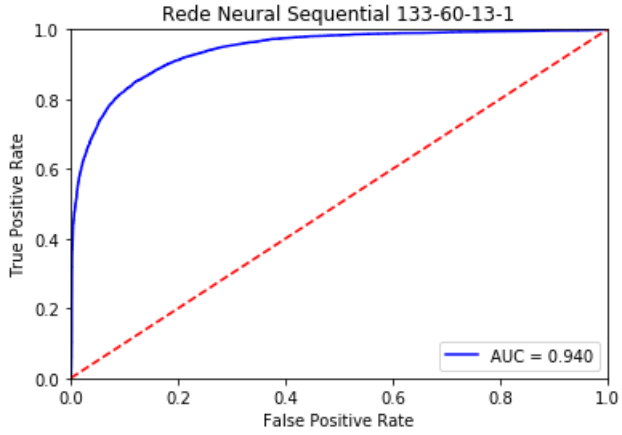

Figure 12: Rede Neural FF com camadas sequenciais de 133-60-13-1 neurônios

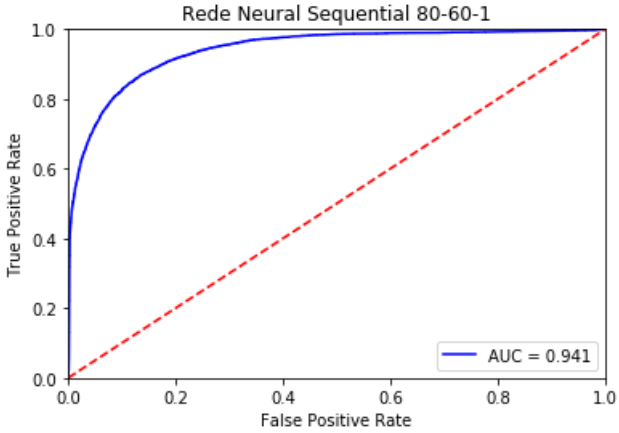

Figure 13: Rede Neural FF, terceira arquitetura.

4.4 Comparação de Resultados

Modelo	logReg	NaiBayes	ArvDecisão	RandForest	GBOOST	RN1	RN2	RN3
AOC	0.907	0.794	0.853	0.945	0.929	0.940	0.940	0.941
F1	0.821	0.703	0.860	0.879	0.851	0.870	0.872	0.872

*usando média dos sunconjuntos de dados pela validação cruzada, com cv=5

Conseguimos observar que apesar de ser um modelo linear e mais 'simples', a Regressão Logística teve um resultado bem satisfatório na tarefa. O mesmo, porém, não pode ser dito ao Naive Bayes, que foi de longe o que teve piores

resultados.

Todos os modelos não-lineares tiveram resultados satisfatórios, categorizando corretamente a grande maioria dos casos. Destaque para a RandomForest, que teve os melhores resultados nas métricas tomadas.

Também é interessante ver que a alteração da arquitetura dos modelos de redes neurais sequenciais quase não interferiram no resultado, isso mostra que provavelmente o tamanho utilizado está longe de ser o ótimo para a tarefa, se estivesse perto do limiar, esperaríamos uma mudança mais expressiva nos resultados.

5 Conclusões

Os modelos performaram de maneira satisfatória, o que era esperado, já que o dataset era de ótima qualidade, com dados reais e bem tratado para os modelos fazerem o que foram projetados. Os hotéis (que provavelmente são em Portugal, ou pelo majoritariamente frequentado por Portugueses), poderiam utilizar de modelos semelhantes para determinar qual seria um valor que faria sentido como depósito pela reserva, por exemplo, ou para prever o quão cheio estará um hotel em determinada época, tendo retorno financeiro e um sistema de reservas mais eficiente.

References

https://www.sciencedirect.com/science/article/pii/S2352340918315191