Comentários do autor do código

Aproveito para informar que o dataset utilizado na implementação do Truco com CBR foi o "dbtrucoimitacao_maos.csv" (este arquivo também está contido no repositório do Github junto ao código do jogo).

Além disso, achei essas informações que podem ser relevantes:

Valores iguais a -130 eram para casos incompletos que o cluster ignorava;

Os valores das colunas "quemContraFlorResto, quemNegouFlor, pontosFlorRobo, pontosFlorHumano, quemGanhouFlor, tentosFlor, quemEscondeuPontosEnvido e quemEscondeuPontosFlor" estavam como null antes do agente aprender automaticamente, os valores zerados foram após isso.

Uma sugestão para os valores negativos, caso necessário dá pra reavaliar a codificação e criar funções para alterar cada um desses valores individualmente.

	Pontuacao
Oponente vai ao baralho	-46
Carta Virada	-1
Agente foi ao baralho	-10
Agente Perdeu	-15

Já que aparentemente todo o dataset contém alguns valores nulos em cada uma das linhas/colunas, talvez seja interessante fazer subsets com o conhecimento no momento de cada jogada. Assim, não tem perigo de realizar uma função dropna() para limpar os subsets, evitando limpar o dataset original inteiro com tal função.

O que foi feito com os valores negativos na última atualização da aplicação:

Não foram tratados no código, o valor negativo na mão do bot eram desconsiderados pela própria codificação do jogo e método de avaliação dos valores. Por estarem muito distantes dos valores onde o bot ganha a rodada, sequer eram considerados.

Sugestões do que há para fazer no jogo (que eu ainda lembro, existem outras diversas possibilidades em aberto):

Ajustar a exportação de dados para um csv, que está dentro da Classe Dados. Nesse caso, alguns dos pontos do jogo na __main__ e nas outras classes não estão atualizando esse acompanhamento de dados da partida;

Corrigir bugs na exibição das chamadas do Envido e do recebimento dos códigos referentes a aceito, real, falta, etc. Foram as últimas condições que trabalhei antes de entregar o trabalho, não ficaram tão polidas quanto as outras e ainda há espaço para melhoria;

Passar os prints que não são de debug ("jogador aceitou envido", "jogador negou truco") para a classe interface;

Implementar lógica de jogar carta virada ao estar com a rodada perdida;

GRANDES possibilidades de ajustes nas condições da classe CBR, pois na maioria dos if-else considerei o vencedor geral e a pontuação de cartas, ou seja, o código não considera explicitamente quando:

O bot foi para o baralho,

Jogou-se carta virada

Quando o jogador foi ao baralho e o bot estava com determinada carta;

A principal melhoria no jogo como um todo, seria remover o df[colunas_string].fillna(-66) na linha 21 da classe de Dados e começar a atualizar a classe CBR para considerar os vizinhos mais próximos por subsets (compostos pelas colunas do dataframe) a cada rodada do jogo (exemplo: na rodada 2 já sabemos a carta que o bot jogou na primeira rodada e podemos usar isso no conhecimento do bot), e não pelo "dataset inteiro + conhecimento da rodada" como está sendo feito atualmente.

Essa linha está preenchendo os NaN relacionados ao clustering que são vazios em sua grande parte, valores vazios por conta da desistência do bot/humano e outros vazios que não conseguimos identificar o motivo;

Não lembro o que exatamente acontece se só remover essa linha, mas imagino que o bot vá ao baralho em todas as partidas, ou quando há o dropna no cálculo de vizinho mais próximos ficamos com o dataset zerado (o que causa erro na função).

Por fim, também envio em anexo a cópia de codificação que serviu como base para o entendimento do dataset usado. As únicas coisas realmente utilizadas dessa tabela foram:

A coluna codificação para definir a pontuação de cada carta;

As Combinações de cartas/pontos para o envido.