PROJETOS EM CIÊNCIA DE DADOS:
Relatório Parcial (27/03)

Gianlucca Devigili e Maisa de O. Fraiz

Pré Processamento dos Dados

O primeiro passo do trabalho consistiu em uma preparação inicial dos dados de forma a facilitar posterior tratamento e processamento dos mesmos. Primeiramente realizamos uma exploração inicial e pré-processamento dos dados, visando a compreensão das colunas presentes no dataset, bem como sua padronização, tanto de nomenclatura de variáveis quanto de tipagem, em especial colunas de data que se encontravam em diversos *datatypes* e formatações diferentes. Após isso, foi realizada o *unpacking* dos dados do dataset principal, já que cada uma de suas colunas continha um dataset próprio, de modo que a granularidade dos dados ficasse separada por data e jogador. Então foram separados os dados em 12 diferentes datasets, criando uma base de dados semelhante a um *Data Warehouse,* tendo uma tabela principal com as variáveis *target* se conectando com as demais por meio de uma chave composta contendo Data e Jogador.

Outra medida tomada, considerando o tamanho do conjunto de dados, foi a transformação do mesmo em um arquivo de formato pickle (.pkl), reduzindo seu tempo de carregamento de 28.4s para 5.34s no dataset completo, e os datasets individuais demorando menos de 1s para a carga completa. Vale ressaltar também que a separação do dataset em diversos arquivos visava a possibilidade de versionamento dos mesmos, não necessitando realizar as transformações necessárias sempre que o *notebook* python fosse processado novamente e também a possibilidade de retornar versões caso venha a acontecer algum problema.

Por fim, foram calculadas novas colunas para servir como *features*, mais especificamente foram criados "*shifts*" dos valores das séries, considerando o valor $\mathbf{y^t}$ de cada um dos targets nos tempos $\mathbf{y^{t-i}}$ com **i** assumindo os valores [1, 2, 3, 4, 5, 6, 7, 14, 30]. Além disso, estão na lista de possíveis *features* a serem calculadas e incluídas posteriormente, caso se observe a necessidade, médias e medianas agrupadas por equipe e temporada, além de valores máximos do target, tempo do jogador na equipe, médias e medianas gerais de cada data, dentre outras possíveis covariáveis.

Modelagem de Baselines

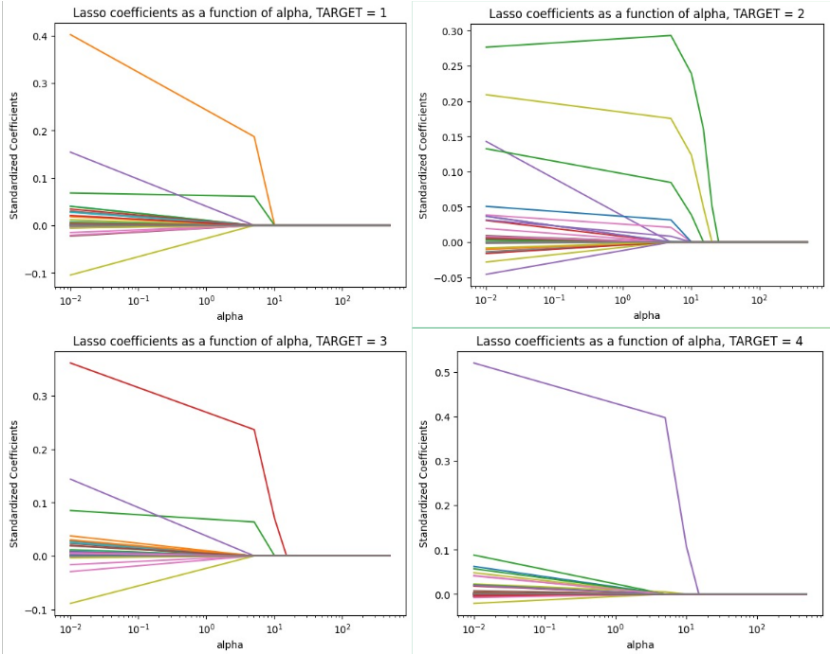
Como parâmetro inicial do projeto, utilizamos dois modelos clássicos da modelagem de séries temporais: o modelo *naive*, que realiza as predições repetindo o último valor observado na série, e o modelo de média que, como diz o nome, utiliza de tal métrica para considerar o próximo valor da série. Como a granularidade do dado não está apenas pela data, mas também para o jogador, não obtivemos sucesso ao utilizar modelos pré construídos em bibliotecas padrões da linguagem python, como *sklearn* e *sktime*, além da difícil adaptação dos mesmos para trabalhar com quatro *targets*. Sendo assim, os modelos foram programados manualmente de modo que retornassem os 4 *targets* com os valores da predição sendo calculados com base nas observações passadas de cada jogador.

Pela análise das estatísticas padrões dos targets, é esperado que o modelo de média apresente um erro elevado para alguns casos, já que as médias dos *targets* são baixas (em ordem: 0.5, 2.46, 0.69, 1.14), porém os mesmos apresentam um alto desvio padrão (4.17 , 6.23, 5.07, 4.23). Deve-se considerar que os valores dos targets podem assumir valores de até 100.

Modelagem LASSO

O primeiro modelo utilizado para predição dos dados, e também para a regularização e seleção de *features* foi o LASSO, tanto em sua versão padrão, quanto o Multitask LASSO, que utiliza norma mista L1/L2 para a regularização. Primeiramente treinamos os modelos com o parâmetro *alpha*

pré-fixado em 0.1. No quesito de seleção de *features*, as *features* que tiveram uma maior importância de acordo com ambos os modelos treinados, foram as de *shift* 1 de todos os 4 targets, e seguidos dos shifts 2 e 3, que intuitivamente são os valores mais importantes já que representam um tempo mais próximo.



Resultados

Para a metrificação da performance do modelo, foi utilizado o Erro Médio Absoluto (*Mean Average Error*). Vale ressaltar que a métrica irá avaliar melhor o modelo nos casos gerais, onde os targets possuem médias próximas de 1, e não sendo um estimador tão efetivo para os casos em que os *targets* "estouram" e apresentam valores extremos.

Os resultados obtidos nas predições encontram-se tabulados abaixo:

	model	target1	target2	target3	target4	mean
0	naive	0.794010	1.349561	0.626269	0.809013	0.894713
1	mean	0.943662	2.253810	0.954913	1.021249	1.293408
2	lasso	0.749040	1.402901	0.676788	0.760047	0.897194
3	multitask LASSO	0.748527	1.503057	0.672377	0.789836	0.928449
4	lasso_best	1.132658	2.746612	1.070628	1.475053	1.606238

Podemos observar na tabela de resultados que o modelo *naive* se mostrou o melhor predizendo os targets 2 e 3, o que explica claramente os modelos lasso terem dado uma maior importância para as features de shift 1. Já os modelos LASSO e Multitask LASSO se mostraram melhores nos targets 4 e 2, respectivamente, com uma diferença relativamente significativa de aproximadamente 0.03 no target 4 e a baixa diferença de 0.0005 no target 2.

Além disso, vale ressaltar que o modelo Multitask LASSO, nomeado pelo grupo como *lasso_best*, treinado com o hiperparâmetro *alpha=1991341338.80* calculado através de Cross Validation, acabou por performar pior que os modelos com o alpha fixado em 0.1. A questão de tal erro ainda necessita uma melhor análise, mas as hipóteses já levantadas encontram-se o elevado número máximo de iterações (10 mil) e os alphas iniciais (0.01, 500 e 100) utilizados.