Lista de Exercícios 09

(prazo final para entrega: 09/05/2019 - quinta)

1) **Implemente a classe KNNModel e suas subclasses KNNClassifier e KNNRegressor. Estas duas últimas classes devem ter interfaces semelhantes às suas respectivas implementações do scikit learn (KNeighborsClassifier e KNeighborsRegressor). Para simplificar, você somente precisa implementar o algoritmo de força bruta, como mostramos em laboratório (ver notebook no github).**

- **Dica: adapte a solução que fizemos no laboratório para implementar as suas classes.**

2) **Crie e avalie modelos preditivos usando o kNN com o dataset iris. Compare os resultados da sua implementação com a implementação de kNN (KNeighborsClassifier) do scikit learn. Use k=[1, 3 e 5]**
   a) **Divida de forma estratificada o Dataset Iris em apenas 10% para treino e 90% para teste. Essa divisão não é usual, mas vamos usar para tornar o problema mais desafiador.**
   b) **Faça a standardização dos dados.**

3) **Implemente as medidas de distância a seguir (ver Figura 1):**
   a) **minkowski_distance(X, row, p)**
   b) **euclidean_distance(X, row). Dica: usar minkowski_distance com p=2.**
   c) **manhattan_distance(X, row) Dica: usar minkowski_distance com p=1.**
   d) **chebyshev_distance(X, row)**

Onde X é uma matriz e row é uma linha para calculo da distância entre X e esta linha (row). Veja exemplo de implementação da distância euclidiana neste jupyter notebook.

Metrics intended for real-valued vector spaces:

identifier	class name	args	distance function
“euclidean”	EuclideanDistance	•	`sqrt(sum((x - y)^2))`
“manhattan”	ManhattanDistance	•	`sum(|x - y|)`
“chebyshev”	ChebyshevDistance	•	`max(|x - y|)`
“minkowski”	MinkowskiDistance	p	`sum(|x - y|^p)^(1/p)`

Figura 01