Machine Learning Engineer Nanodegree

Trabalho de Conclusão de Curso

Robô Agrônomo

Juan Manoel Marinho Nascimento

30 de Maio de 2018

I. Definição

Visão Geral do Projeto

Este trabalho trata-se do desenvolvimento de um "robô" que analisa sensores de umidade de solo de jardins ou hortas em pequena e média escala, com esses dados a IA prediz se necessita de rega. Os dados são fornecidos pelo microcontrolador nodemcu (esp8266-12) que através do sensor higrômetro e por meio de uma uma rede wireless, comunica-se com um servidor que armazena os dados coletados em um banco de dados (csv) de monitoramento da planta, registrando os dados de níveis de umidade de solo das plantas, em porcentagens. O sistema é utilizado para irrigação inteligente que permite saber qual momento do dia é melhor para regar as plantas. Portanto, a proposta é programar uma IA que classifique os níveis de umidade e prediga a situação da amostra quando coletada, obtendo as probabilidades de irrigação. Através desse sistema, pessoas desprovidas de muito capital de investimento ou recursos hídricos podem automatizar seus jardins e cuidar da sua horta de forma mais adaptativa para cada planta em questão. Resultando em um melhor plantio e cultivo.

Problema

Com base na realidade de Porto Velho, que enfrentou 2 paralisações da Companhia de Águas e Esgotos de Rondônia -CAERD- em um intervalo menor que 6 meses, causando impactos nos recursos hídricos que as residências possuem nos reservatórios de água para consumo local. Muitas residências que possuem plantios, os abastecem com a água que possuem no reservatório, e nos momentos de crises hídricas, isso torna-se um problema, já que todos os consumos locais precisam de acompanhamento da quantidade gasta para determinados usos. Além do fato de cada planta necessitar de uma quantidade de água para o cultivo, sendo que algumas necessitam de pulverização em suas folhas. Os cultivos caseiros sobrevivem enquanto tiverem disponibilidade de água e supervisão de regas,

contando que uma horta urbana pode não possuir a disponibilidade necessária para o cultivo do plantio, pontuando fatores determinantes como tempo, umidade, ambiente, entre outros.

Métricas

A Métrica utilizada foi a de precisão geral de accuracy. É basicamente o número de acertos (positivos) dividido pelo número total de exemplos. Devendo ser usada em datasets com a mesma proporção de exemplos para cada classe e, quando as penalidades de acerto e erro para cada classe forem as mesmas.

Em problemas com classes desproporcionais, ela causa uma falsa impressão de bom desempenho. Por exemplo, num dataset em que 80% dos exemplos pertençam a uma classe, só de classificar todos os exemplos naquela classe já se atinge uma precisão de 80%, mesmo que todos os exemplos da outra classe estejam classificados incorretamente.

$$Precisão\ Geral = \frac{P}{P + N}$$

Para definição de quais são os níveis de umidade, será utilizada a quantidade de vezes em que os dados passam pelos intervalos da tabela abaixo. Identificando cada ponto de monitoramento dos sensores que enviam os dados, desconsiderando a ordem de transmissão dos dados para o broker, nem a quantidade de cada sensor em relação aos grupos. Desta forma, as características de cada sensor terão a dimensão do total de pontos de monitoramento, sendo o valor de cada atributo a contagem das passagens do sensor por aquele ponto.

Tabela 01 – Descrição dos dados.

IoTs	Níveis	Descrição	Intervalo
Sensor 01	1	árido	0 - 25
Sensor 02	2	seco	25 - 50
Sensor 03	3	úmido	50 - 75
Sensor 04	4	molhado	75 - 100

Métrica de avaliação

Lembrando o objetivo de encontrar um grupo de sensores que circulam na área monitorada, no qual possui um comportamento mais parecido com o dos

sensores de solo, passamos a analisar os seguintes fatores: ID de cada solo, mês de coleta, dia, hora e o mais importante, a umidade.

Seria desejável encontrar um segmento dos sensores que transitam na área de coleta, no qual todos os dados coletados alvo se encaixam. Definimos então a taxa de sensores alvo que se encaixam em cada segmento como um modelo de leitura ideal.

A escolha de k é muito crítica - Um pequeno valor de k significa que o ruído terá uma influência maior no resultado. Um grande valor o torna computacionalmente caro e meio que derrota a filosofia básica por trás do KNN (os pontos próximos podem ter densidades ou classes similares). Uma abordagem simples para selecionar k é definida como k = n ^ (1/4).

II. Análise

Exploração dos Dados

O conjunto principal a ser utilizado possui as seguintes características:

- ID – nível de 1 a 4;
- Mês/dia/hora – dados do horário e registro da coleta;
- Umidade – dados do nível de 0 a 100 de umidade de solo;
- Count – contagem da quantidade de vezes que determinado sensor passou pela coleta.

Tabela 02 – Amostra das 5 primeiras linhas do conjunto de dados

ID	MÊS	DIA	HORA	UMIDADE
1	1	6	18	17
1	1	10	12	19
1	1	19	5	8
1	1	23	9	12
1	1	4	16	13

O conjunto de dados dos sensores em geral possui no total 1583.000000. Esses dados são referentes aos dias 1 e 24 de Janeiro de 2018, segunda a sexta-feira.

O conjunto de dados dos sensores com restrição possui as características:

- Umidade – nível de umidade de solo;
- Nível – varia de 1 a 4 para cada tipo de nível de classificação.

Visualização exploratória

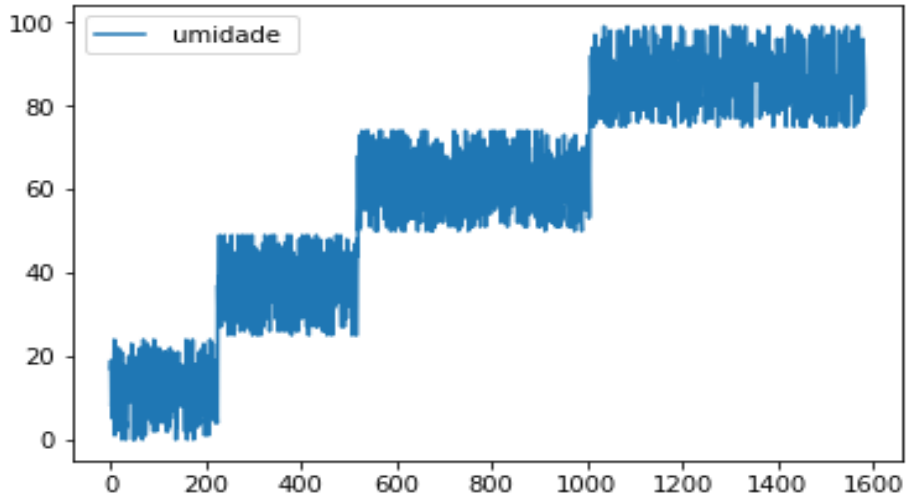


Figura 1 – Distribuição de frequência dos níveis nos pontos de monitoramento

A Figura 1 mostra a distribuição de frequência dos dados nos pontos de monitoramento. Para este projeto, estes pontos de monitoramento serão utilizados como dimensões, e cada sensor possui como características a quantidade de dados por cada ponto.

Tabela 03 – Estatísticas de monitoramento

	ID	MES	DIA	Hora	umidade
count	5403.000000	5403.000000	5403.000000	5403.000000	5403.000000
mean	2.931890	5.514344	20.991116	11.572090	60.335739
std	0.990788	0.499840	1.396667	7.030724	25.768100
min	1.000000	5.000000	19.000000	0.000000	0.000000
25%	2.000000	5.000000	20.000000	5.000000	47.000000
50%	3.000000	6.000000	21.000000	12.000000	64.000000
75%	4.000000	6.000000	22.000000	18.000000	81.000000
max	4.000000	6.000000	23.000000	23.000000	99.000000

A Tabela 3 mostra a distribuição de dados estatísticos onde podemos observar os níveis de média, mínimo e máximo dos dados, totalizando os níveis de umidade de solo nas faixas de min e max, um range de 0 a 99.000.

Algoritmos e Técnicas

K-NN – K Nearest Neighbors (Algoritmo do vizinho mais próximo***)***

É um algoritmo simples de aprendizado de máquina que categoriza uma entrada usando seus *k* vizinhos mais próximos.

Os vizinhos *k*-nearest, possuem várias propriedades que o diferenciam de outros algoritmos de aprendizado de máquina. Primeiro, kNN é *não-paramétrico* , o que significa que não faz nenhuma suposição sobre a distribuição de probabilidade da entrada. Isso é útil para aplicações com propriedades de entrada que são desconhecidas e, portanto, tornam o k-NN mais robusto do que os algoritmos *paramétricos*. O contraste é que, algoritmos paramétricos de aprendizado de máquina tendem a produzir menos erros do que os não-paramétricos, já que leva em conta as probabilidades de entrada que podem influenciar a tomada de decisão.

Além disso, kNN é um tipo de *aprendizagem preguiçosa*, que é um método de aprendizagem que generaliza os dados na fase de testes, e não durante a fase de treinamento. Isso é contrastado com o *aprendizado rápido*, que generaliza os dados na fase de treinamento, e não na fase de testes. Um benefício do aprendizado preguiçoso é que ele pode se adaptar rapidamente às mudanças, já que não está esperando um determinado conjunto de dados generalizado. No entanto, uma grande desvantagem é que uma quantidade enorme de computação ocorre durante o teste (uso real), em vez de pré-cálculo durante o treinamento.

K-MEANS – (K-Médias**)**

A idéia do algoritmo K-Means, é de fornecer uma classificação de informações de acordo com os próprios dados. Esta classificação, é baseada em análise e comparações entre valores numéricos dos dados. Desta maneira, o algoritmo automaticamente vai fornecer uma classificação automática sem necessidade de nenhuma supervisão humana, ou seja, sem nenhuma pré-classificação existente. Por causa dessas características, K-Means é considerado como um algoritmo de mineração de dados não supervisionado.

O algoritmo vai indicar uma classe (cluster) e vai dizer quais linhas pertencem a esta classe. O usuário deve fornecer ao algoritmo a quantidade de classes que ele deseja. Este número de classes que deve ser passada para o algoritmo é chamado de k e é daí que vem a primeira letra do algoritmo: K-Means.

Para gerar as classes e classificar as ocorrências, o algoritmo faz uma comparação entre cada valor de cada linha por meio da distância. Geralmente utiliza-se a distância euclidiana para calcular o quão 'longe' uma ocorrência está da outra. A maneira de calcular esta distância vai depender da quantidade de atributos da tabela fornecida. Após o cálculo das distâncias o algoritmo calcula centróides para cada uma das classes. Conforme o algoritmo vai iterando, o valor de cada centróide é refinado pela média dos valores de cada atributo de cada ocorrência que pertence a este centróide. Com isso, o algoritmo gera k centróides e coloca as ocorrências da tabela de acordo com sua distância dos centróides.

Um Gaussian mixture model (GMM)

O modelo de misturas de gaussianas é um modelo estocástico, que modela classes - que, para o reconhecimento de locutor, podem ser unidades acústicas ou mesmo um locutor - sem que se considerem informações temporais.

O objeto de interesse de um problema de classificação é o cálculo da chamada probabilidade a posteriori, que pode ser calculada através da fórmula de Bayes, como mostrado na equação.

$$P(C_i|\mathbf{o}) = \frac{p(\mathbf{o}|C_i)P(C_i)}{p(\mathbf{o})}$$

É uma forma de modelagem de séries temporais que consiste no agrupamento de densidades de probabilidade gaussianas com características individuais. Proporciona a vantagem de pouco volume de informação armazenado e de pouca carga computacional para efetuar as estimativas de densidade, principalmente quando se trabalha com conjunto de dados de n-dimensões. Os parâmetros da função de densidade de probabilidade (pdf) de uma Mistura Gaussiana são o número de Gaussianas, o coeficiente de ponderação da mistura, a média e a matriz de covariância de cada função de densidade Gaussiana. O objetivo é determinar o número de Gaussianas que melhor represente a densidade de probabilidade dos dados coletados.

PCA – Principal Component Analisys

A Análise de Componentes Principais (ACP) ou *Principal Component Analysis (PCA)* é um procedimento matemático que utiliza uma transformação ortogonal (ortogonalização de vetores) para converter um conjunto de observações de variáveis possivelmente correlacionadas num conjunto de valores de variáveis linearmente não correlacionadas chamadas de componentes principais. O número

de componentes principais é menor ou igual ao número de variáveis originais. Esta transformação é definida de forma que o primeiro componente principal tem a maior variância possível (ou seja, é responsável pelo máximo de variabilidade nos dados), e cada componente seguinte, por sua vez, tem a máxima variância sob a restrição de ser ortogonal a (i.e., não correlacionado com) os componentes anteriores. Os componentes principais são garantidamente independentes apenas se os dados forem normalmente distribuídos (conjuntamente). O PCA é sensível à escala relativa das variáveis originais. Dependendo da área de aplicação, o PCA é também conhecido como transformada de *Karhunen-Loève* (KLT) discreta, transformada de *Hotelling* ou decomposição ortogonal própria (POD). [1]

O PCA será utilizado para fornecer uma dimensão reduzida dos dados, pois suas componentes possuem a perspectiva mais informativa dos dados. Para um problema que se inicia com 4 dimensões, a utilização de uma técnica de redução de dimensionalidade como o PCA é essencial.

Para implementação do PCA será utilizada a biblioteca do *Scikit-learn sklearn.decomposition.PCA*. Será a princípio calculado o PCA com 2 componentes, e após a análise da variância será reduzida para 2 dimensões para de verificado se o PCA pode ser recalculado com menos componentes ou se necessitará de um número maior de componentes.

III. Metodologia

Pré-processamento de dados

A tabela de dados carregada no projeto estão em arquivos .csv, estes arquivos são carregados em *data frame*, como exemplificado anteriormente nas Tabelas 01, 02 e 03.

O *data frame* possuem em suas linhas a contagem dos níveis de umidade de cada sensor por cada área de atuação;

Para cada *leitura* são seguidos os seguintes passos:

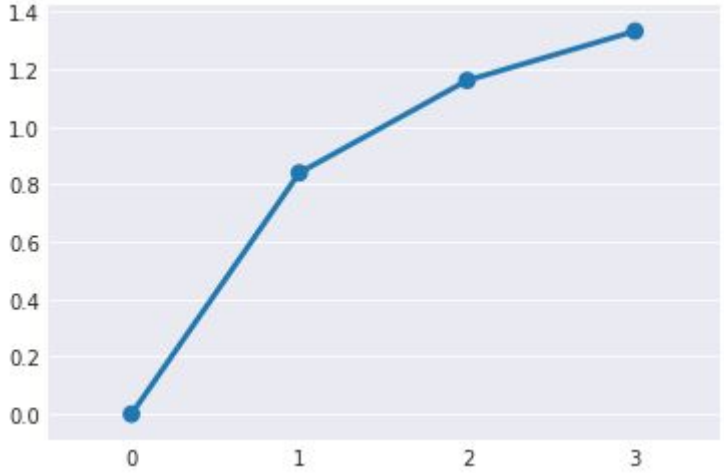
- É criado uma variável X para pegar dados de treino com 4 características;
- É criada uma variável y para cada dados de test com dados de ID ;
- É criada uma variável “Umidade” para visualização dos possíveis grupos classificados.

Não foram utilizadas técnicas para excluir *outliers*, pois esse projeto trata justamente de encontrar padrões que divergem do comportamento comum, portanto os *outliers* podem estar em os alvos do projeto.

Implementação

KNN

O data frame que contém os dados dos sensores em geral é denominado “df” de dataframe. Como “df” possui 5 características, será utilizado o agrupamento para descobrir qual vizinho se comunica com os vizinhos mais próximos e passa agrupar os dados em 4 grupos. Inicialmente é aplicado aos dados um KNN-Classifier com 4 componentes principais, após o treino e teste, é feita a predição dos dados.

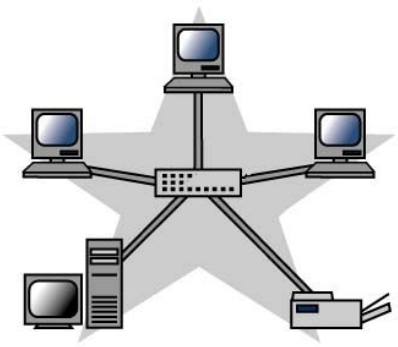


Nesse gráfico fica claro onde os K e seus respectivos centróides se divergem entre os valores de 0-3. Tendo como o K mais otimizado o terceiro K.

Comunicação com os IoTs

Topologia

Topologia de estrela

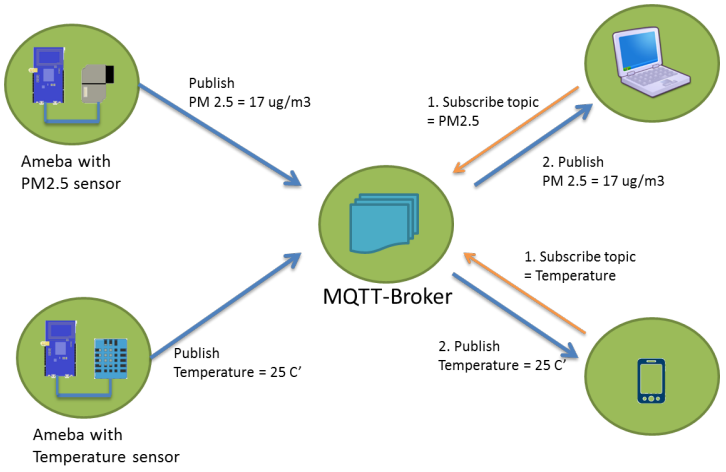


Topologia de Estrela, é um dos modelos computacionais de comunicação mais simples estudado, ele opera com um comutador no centro que fala com os demais e conectados em cada ponta da estrela. O comutador central ao receber entradas de determinado ponto ele sabe para quanto ponto vai encaminhar a comunicação a ser trafegada.

Observando pela perspectiva de internet das coisas podemos entender que cada nó da rede possa ser um IoT conectado, enviando e recebendo dados para ser processado e tratado pelo servidor local (broker), onde está hospedado nosso Robô que fará as análises e tomadas de decisões adequadas.

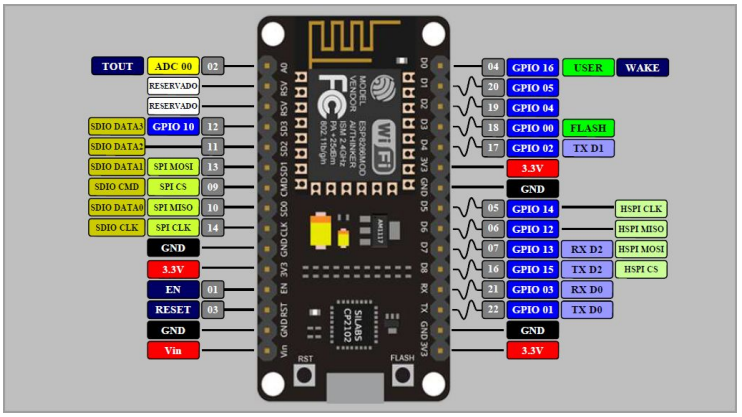
Protocolo de Sistema Embarcado

MQTT é um protocolo comumente usado na indústria de coleta de dados e aplicado dentro de sistemas como scala, é foi desenvolvido pela IBM, para comunicação entre mainframes, por ser de fácil implementação é um protocolo que é extremamente fácil de implementar e otimizar.



O MQTT opera usando conceito de tópicos para envio e recebimento de mensagens, usando um conceito de topologia de rede muito semelhante a rede estrela, onde temos um broker (server), e vários iot conectados e enviando e recebendo mensagens, sendo para published (enviar) e subscribe (receber).

Embarcado



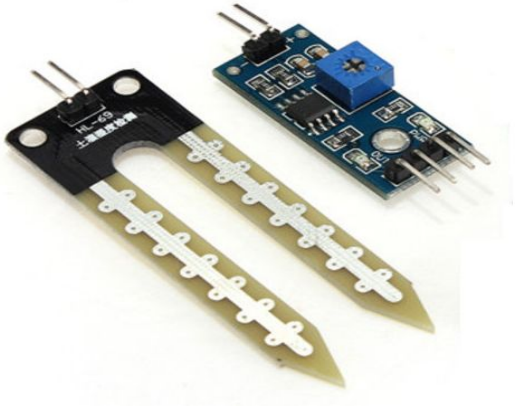
imageNode.png

O embarcado utilizado neste projeto foi o NODEMCU (esp8266-12e), ele torna-se muito útil em coleta de dados de campo, foram utilizados 4 nodemcu durante o projeto, possuindo portas de I/O e um chip esp específico para trabalhar com redes sem fio, por operar em modelo tcp/ip embarcado, o node tem bibliotecas que falam com mqtt.

Em uma rede local, é possível fazer coleta de dados do *node*, usando protocolo de baixa latência para otimização de coleta, por ser barato e fácil de encontrar no mercado, foi a melhor opção para esse tipo de problema.

Sensor de Umidade

Sensor de Umidade do Solo Higrômetro detecta as variações de umidade no solo, visto que quando o solo está seco a saída do sensor fica em estado alto e quando úmido em estado baixo.



O limite entre seco e úmido pode ser ajustado através do potenciômetro presente no sensor que regulará a saída digital D0. Contudo para ter uma resolução melhor é possível utilizar a saída analógica A0 e conectar a um conversor AD, como a presente no nodemcu por exemplo.

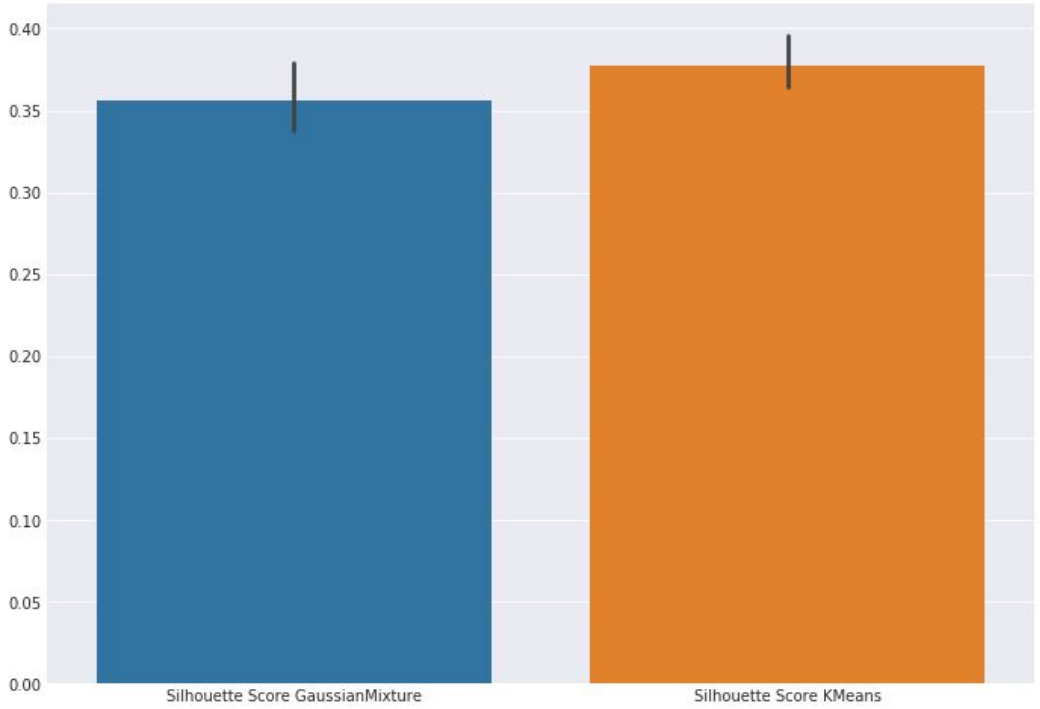
Benchmark

O modelo aqui apresentado será comparado com um modelo que define o melhor parâmetro “k” para o *k-means* e comparar com o algoritmo de GaussianMixture por meio do Coeficiente de silhueta.

O Coeficiente de silhueta é calculado utilizando a distância média intra-cluster (a) e a distância média do cluster mais próximo (b) para cada amostra. O Coeficiente de Silhueta para uma amostra é b-amax (a, b) . Para que fique claro, é a distância entre uma amostra e o aglomerado mais próximo que a amostra não pertence. Observando que o Coeficiente de Silhueta só pode ser definido se o número de *labels* (n_{labels}) for $2 \leq nlabels \leq nsamples-1$, onde $n_{samples}$ é o número de amostras.

A função *sklearn.metrics.silhouette_score* retorna o coeficiente médio de silhueta sobre todas as amostras. O melhor valor é 1 e o pior valor é -1. Valores próximos de 0 indicam grupos sobrepostos. Valores negativos indicam geralmente que uma amostra foi atribuída ao cluster errado, quando outro cluster seria mais similar.

Foi testado



No gráfico fica claro que a validação de Silhouette Score no Algoritmo de Gaussian Mixture fica mais próximo de 1 em relação ao K Means, tornando ele uma fator de suma importância para validação do modelo.

Neste ponto fiz uma train_test_split para medir niveil de Score do moledo KNN .O método irá por padrão dividir aleatoriamente seus dados em 25% para teste o restante para treino. Obs: O percentual de 25% é a opção padrão, caso queira altera esse valor, use o parâmetro test_size, por exemplo, se quiser que seu conjunto de teste seja 30% faça: test_size = 0.3 Vamos entender as variáveis que criei para receber esses dados (X_treino, X_teste, y_treino, y_teste) O nome já fala por si só, a variável X_treino recebe a porção de dados que iremos usar para treinar o modelo.

Accuracy com Score e train_test_split

```
y_pred_val = knn.predict(X)

Accurary_score =  float(accuracy_score(y, y_pred_val))* 100
print "ACCURACY: \t   [",Accurary_score,"    ] Score"

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=.5,
                                                    random_state=42)
classifier = KNeighborsClassifier(n_neighbors=4)
classifier.fit(X_train, y_train)
y_score = classifier.score(X_test,y_test)*100
print "Train_test_split:  [",y_score,"] Score"
```

```
ACCURACY:          [ 99.6853599852     ] Score
Train_test_split:  [ 98.63064396743152 ] Score
```

Na figura acima temos uma validação usada para comparar se o KNN pode ser usado com melhor treino e test em relação aos outros algoritmos visto que o score de train_test_split foi menor que o test de accuracy.

IV. Resultados

Avaliação e validação de modelos

Justificativa

A predição alcançou um score de accuracy de 99.68% . Esta base de dados possui um total de 5404 de dados coletados pelos sensores. Aplicando a clusterização do conjunto de dados de coleta, foi possível identificar os grupos com seus respectivos níveis de umidade. O sistema foi capaz de encontrar e predizer a um novo sensor que ele se encaixa em um determinado grupo já classificado, segundo esses dados é possível saber quando é o melhor momento para a rega do plantio, porém, ao analisar os sensores com relação a onde eles foram postos de forma geográfica, fica claro que para determinados tipos de solo é factível que seja mais úmido ou mais seco para determinadas plantas, porém este dado foi apenas

observado. Ainda há de considerar que é preciso mais análises com dados distintos para novas correlações. Contudo apenas com esse simples robô é possível otimizar quantidade de rega e saber quando é de fato necessário regar.

V. Conclusão

Levando em consideração esses aspectos, é possível analisar a aquosidade da gleba baseando-se em, construção de IoT, programação de cada módulo wi-fi com sensor de umidade, configuração na rede e comunicação local, estabelecida entre IoT e servidor, onde no servidor, o robô recebe a coleta, analisa, classifica e prediz o estado do solo, baseado em um modelo treinado. Apesar disso, existe a necessidade abranger novos parâmetros para comparação, com dados distintos, correlacionando com dados meteorológicos, geográficos e fluviais.

Reflexão

O problema em questão é bem simples, analisar níveis de umidade de solo consumindo dados dos sensores para saber quando é necessário para regar ou não na horta urbana, visto que na cidade de porto velho ouve a greve que interferiu e gerou um problema para todo uma cidade que depende da CAERD para receber água. Neste ponto, a solução analisada foi usar dispositivos IoTs para coletar dados e treinar um robô para dizer se um novo dado pertence a um determinado grupo já classificado. Um dos pontos mais difíceis do projeto foi entender quais dados seriam necessários para treino do modelo e aplicação dele, outro ponto difícil é que sensores de umidade de solo na região norte do país não são vendidos com tanta facilidade e demorou bastante pra chegar e por consequência atrasou o estudo e desenvolvimento do projeto. Uma das partes mais interessantes do projeto foi certamente a implementação do projeto em uma horta urbana de casa, onde podemos observar o funcionamento do projeto de forma clara e ver onde melhorar e o que não funciona. O modelo construído funciona muito bem para resolução deste tipo de problema, visto por uma perspectiva de agrotech, funcionaria muito bem a uma plantação, com mais dados coletados e uma predição mais assertiva, tornando o projeto expansivo e adaptativo a outros cenários

Possíveis Melhorias

Acredito na seguinte questão, temos apenas dados locais de leitura, com IoTs lendo dados diretamente na hora/plantação não temos dados de outras fontes apenas dados locais, observando por outra perspectiva vejo que preciso ter dados globais, como dados meteorológicos e geográficos, possibilitando a leitura de temperatura, pressão atmosférica e também como é a região em que a plantação está, se é relevo, pico, colina dentro uma diversidade de dados que a geografia do solo pode me passar, levando em consideração também dados da plantação, como se é flores ou hortaliças, acredito que com esses dados

aplicado a uma rede neural ou alguma rede baseada em fuzzy com markov decision, para análise mais específica de quando regar ou não. Acredito também que o aprimoramento e maior quantidade e variedades de dados poderia também aplicar um SVM.

VI. Referências

[1]
http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[2]

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[3]

https://www.filipeflop.com/blog/planta-iot-com-esp8266-nodemcu/

[4]

https://www.embarcados.com.br/mqtt-protocolos-para-iot/

[5]

https://www.amazon.com.br/Learning-Scikit-Learn-Machine-Python/dp/1783281936

[6]

https://techtutorialsx.com/2017/04/09/esp8266-connecting-to-mqtt-broker/

[7]

http://www.sbmac.org.br/cmac-se2011/trabalhos/PDF/194.pdf

[8]

https://www.gta.ufrj.br/grad/07_2/viviane/GaussianMixtureModels-GMMs.html