## **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**

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.

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

Tabela 1:Descrição dos dados.

### **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:

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 ^ (1/2).

### **II. Análise**

### **Exploração dos Dados**

O conjunto principal a ser utilizado possui as características:

* ID – nivel de 1 a 4
* mes/dia/hora – dados do horario e estação do ano que o dado foi coletado
* úmidade – Dados do nivel de 0 - 100 de úmidade de solo
* count – contagem da quantidade de vezes que determinado sensor passou pela coleta

**Tabela 01 – Amostra das 6 primeiras linhas do conjunto de dados**

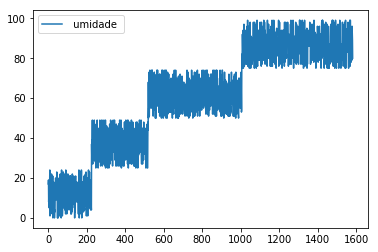
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | MES | DIA | HORA | UMIDADE |
| 1 | 1 | 6 | 18 | 17 |
| 1 | 1 | 10 | 12 | 19 |
| 1 | 1 | 19 | Q5 | 8 |
| 1 | 1 | 23 | 9 | 12 |
| 1 | 1 | 4 | 16 | 13 |

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 – nivel de umidade de solo
* nivel – varia de 1 a 4 para cada tipo de nivel de classificação

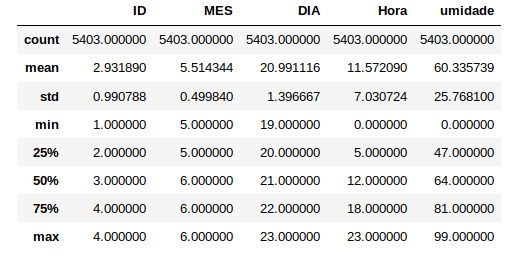
**Visualização exploratória**



**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 02 – Estatísticas de monitoramento**



A Tabela 2 mostra a distribuição de dados estatísticos onde podemos observar os níveis de média, min e máximo dos dados, totalizando os niveis de umidade de solo nas faixas de min e max um range de 0 a 99.000.

### **Algorítimos e Técnicas**

***K-NN – K Nearest Neighbors(***[Algoritmo do vizinho mais próximo](https://pt.wikipedia.org/wiki/Algoritmo_do_vizinho_mais_pr%C3%B3ximo)***)***

É um [algoritmo](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/algorithm/&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhisRQ5TeD6R5afdYD2_6av5i9eOmg) simples de [aprendizado de máquina](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/machine-learning/&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhiomcq7A9oWmNGff2dGhUWvRcLWbg) que categoriza uma [entrada](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/input-output/&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhji4nbR_VIVWUsVCkxmV6XfQh4F0Q) 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*](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/non-parametric/%3Fwiki_title%3Dnon-parametric&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhj4U6BNsyJzbltJSUGcOL5cdRXZbw) , o que significa que não faz nenhuma suposição sobre a [distribuição](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/probability-distribution/&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhjqY1stmhJqwbnEQobCSZyxD_JQqg) de [probabilidade](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/probability-distribution/&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhjqY1stmhJqwbnEQobCSZyxD_JQqg) da entrada. Isso é útil para aplicações com propriedades de entrada que são desconhecidas e, portanto, tornam o k-NN mais [robusto do](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/robust/%3Fwiki_title%3Drobust&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhgz3Z9IMxeh69VP4KeAAoi_RSatfg) que os algoritmos [*paramétricos*](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/parametric/%3Fwiki_title%3Dparametric&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhg3bHp5JOBZILRxb-g17iAlYLW1Sg). O contraste é que algoritmos paramétricos de aprendizado de máquina tendem a produzir menos erros do que os não-paramétricos, já que levar em conta as probabilidades de entrada pode influenciar a tomada de decisão.

Além disso, kNN é um tipo de [*aprendizagem preguiçosa*](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/lazy-learning/%3Fwiki_title%3Dlazy%2520learning&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhjGh-NX0KKB_gHy6EVyf3majDnK5A) , que é um método de aprendizagem que generaliza os dados na [fase de testes](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/testing/&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhgM4cnDACWVO7YJaF2F_6NfIkE6eA) , e não durante a [fase de treinamento](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/training/%3Fwiki_title%3Dtraining%2520phase&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhgIpE50ZIzg4Rgdyvh8YJ3tkpnJPA) . Isso é contrastado com o [*aprendizado rápido*](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/eager-learning/%3Fwiki_title%3Deager%2520learning&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhiTh_SPkfhUDqd2GW_lwZEultvjPA) , 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](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&prev=search&rurl=translate.google.com&sl=en&sp=nmt4&u=https://brilliant.org/wiki/computation/%3Fwiki_title%3Dcomputation&xid=17259,15700023,15700124,15700149,15700168,15700173,15700186,15700191,15700201&usg=ALkJrhhvNz5tM490-Vp9kBo6MpreloP_2w) ocorre durante o teste (uso real), em vez de pré-cálculo durante o treinamento.

### **II. 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 niveis de umidade de cada sensor por cada area de atuação;

Para cada *leitura* são seguidos os seguintes passos:

* É criado uma variavel X para pegar dados de treino
* É criada uma variavel y para cada dados de test
* É criada uma variável “Umidade” para visualização dos possíveis grupos de

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**

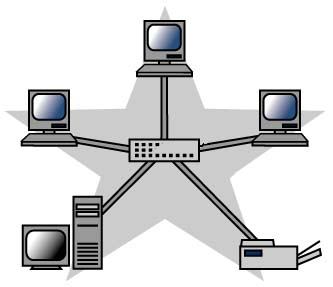
**KNN**

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 proxumos e passa a grupar os dados em 4 grupos. Inicialmente é aplicado aos dados um KNN-Classifer com 4 componentes principais, após o treino e test, é feita a predição dos dados.

### **Comunicação com os IoTs**

#### **Topologia**

##### **Topologia de estrela**

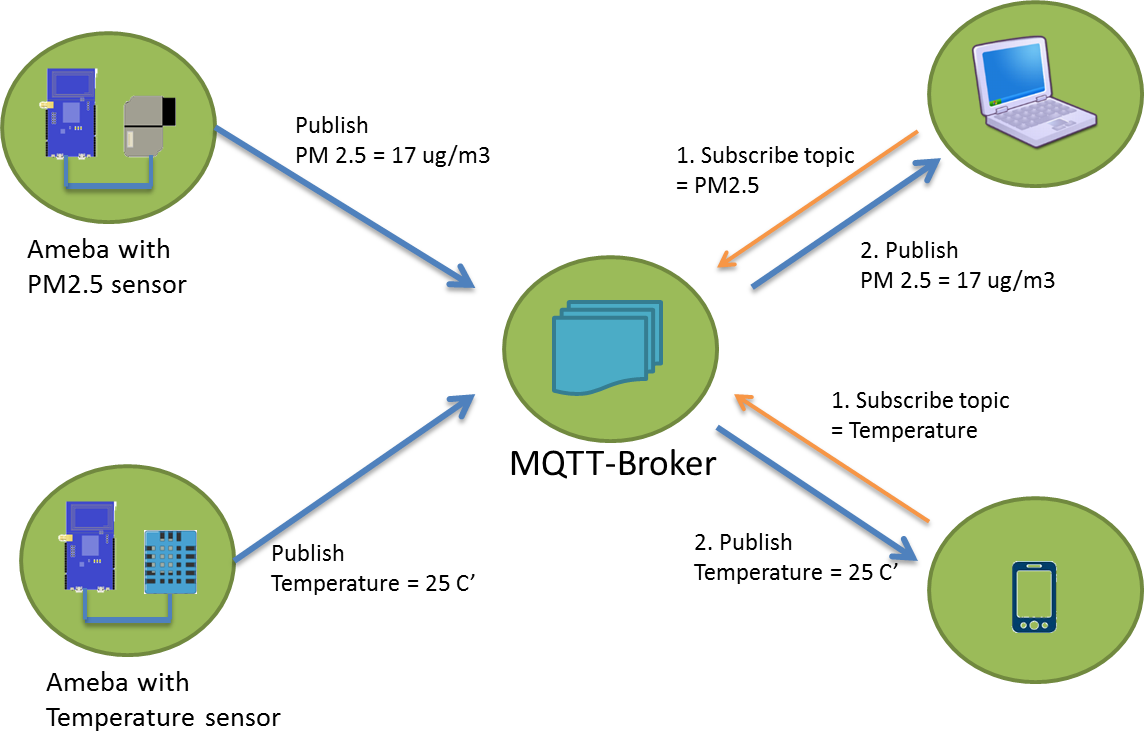


#### Topologia de Estrela, é um dos modelos computacionais de comunicação mais simples estudado, ele opera com um comutador no centro que fala com os demais e conectados em cada ponta da estrela. O comutador central ao receber entradas de determinado ponto ele sabe para quanto ponto vai encaminhar a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

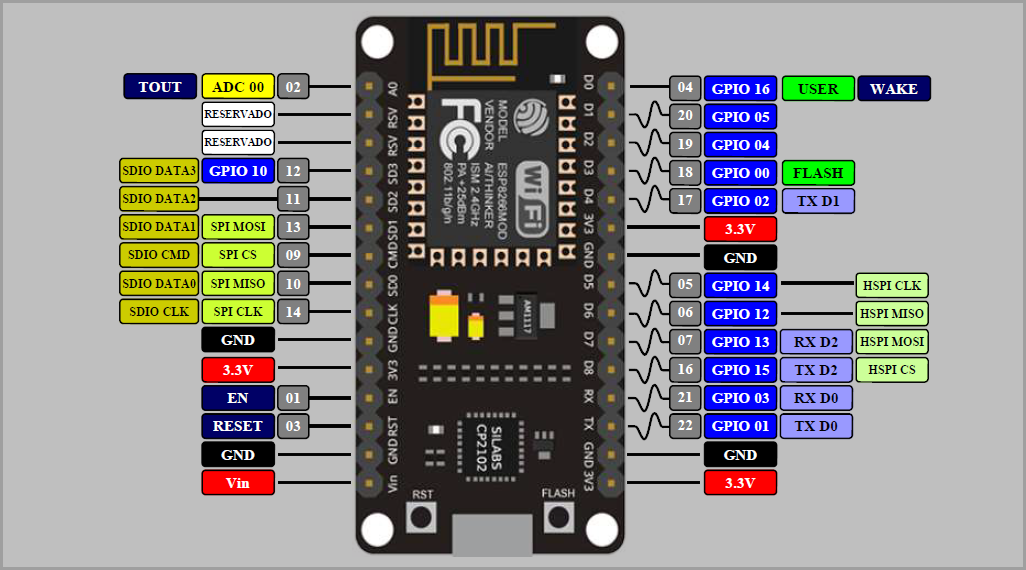
#### **Protocolo de Sistema Embarcado**

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**



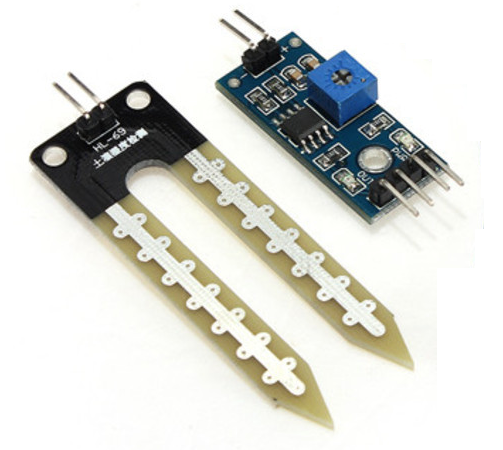
imageNode.png

O embarcado utilizado neste projeto foi o NODEMCU(esp8266-12e), ele se torna muito útil em coleta de dados de campo, foi utilizado 4 nodemcu durante o projeto ele possui portas de I/O e possui 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 foi feito para detectar as variações de umidade no solo, sendo 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.

### **IV. Resultados**

## **Avaliação e validação de modelos**

### **Justificativa**

A predição alcançou um score 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 análise 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 robo 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 usando 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 a água. Neste ponto a solução analisada para foi usar dispositivos IoTs para coletar dados e treinar um robô para dizer se uma novo IoT 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

### **VI. Referências**

[1][http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html](http://scikit-learn.org/stable/modules/clustering.html#k-means)

[3]

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

[4]

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

[5]

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

[6]

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

[7]

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