

ANA RAQUEL

POSTECH

DATA ANALYTICS

MACHINE LEARNING AVANÇADO

AULA 01

SUMÁRIO

O QUE VEM POR AÍ?	3
CONHEÇA SOBRE O ASSUNTO	4
HANDS ON	13
O QUE VOCÊ VIU NESTA AULA?	14
REFERÊNCIAS.....	15

EMSE

O QUE VEM POR AÍ?

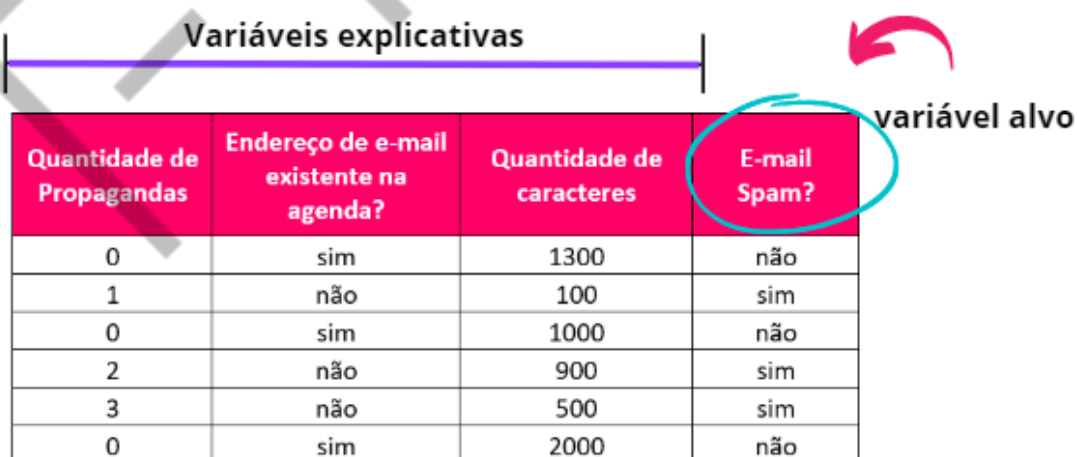
Você aprendeu como prever valores numéricos com o modelo regressão, mas já imaginou como poderíamos criar um algoritmo para prever classes? Será que utilizamos as mesmas técnicas? Ou os dados possuem um comportamento diferente? Nesta aula, você vai aprender o que definimos como um “modelo supervisionado” em Machine Learning e quais são os algoritmos e as principais técnicas para resolver esse tipo de problema.

EMAND

CONHEÇA SOBRE O ASSUNTO

Como podemos definir um modelo supervisionado? Bem, para você poder assimilar melhor o contexto, vamos utilizar um modelo bem simples da vida real: como poderíamos classificar se um e-mail é spam ou não spam? Não queremos perder um e-mail importante do(a) chefe, certo? Nesse caso, podemos classificar se um e-mail é spam pelo seu conteúdo duvidoso, propaganda de bens e serviços, entre outros atributos que qualificam um e-mail spam. Em contrapartida, para classificarmos um e-mail comum, podemos considerar outros pontos, como remetente conhecido, assunto de trabalho etc. Perceba que aqui listamos alguns fatores (características) que qualificam se o e-mail pode ser um spam ou não.

Com esse exemplo, podemos fazer uma analogia com **modelos supervisionados**, onde os **fatores que classificam** se meu e-mail é spam ou não **são as minhas variáveis explicativas** (também chamados de características), enquanto a nossa **classe pode ser definida como o tipo de e-mail** (spam ou não spam). Modelos supervisionados possuem esse tipo de comportamento nos dados, temos **algumas variáveis que podem qualificar a minha classe alvo**. Observe a figura 1 – “Primeiro modelo de classificação”, para melhor compreensão:



Variáveis explicativas			
Quantidade de Propagandas	Endereço de e-mail existente na agenda?	Quantidade de caracteres	E-mail Spam?
0	sim	1300	não
1	não	100	sim
0	sim	1000	não
2	não	900	sim
3	não	500	sim
0	sim	2000	não

Figura 1 – Primeiro modelo de classificação
Fonte: Elaborado pela autora (2023)

Observe que, nesse exemplo do spam, utilizando essas três variáveis explicativas, podemos classificar se um e-mail é spam ou não. Mas, como conseguimos de fato identificar? Será que podemos **encontrar algum padrão** nos dados? Perceba que todos os e-mails considerados “não spam” possuem um certo padrão:

- Não possuem propagandas em seu corpo de e-mail.
- Todos os e-mails recebidos são de contatos que já existem na agenda.
- Possuem um total maior ou igual a 1000 caracteres.

É assim que funciona a engrenagem de um algoritmo classificador. Claro que depende muito da técnica e a matemática/lógica por trás do algoritmo, mas é com base em alguns padrões ou similaridade entre os dados que é decidida a classe. No modelo de classificação **podemos classificar uma ou mais classes**, e diferente da regressão, onde tentamos encontrar a linearidade entre os dados, na classificação queremos **encontrar quais são as melhores features (colunas) que descrevem uma classe**.

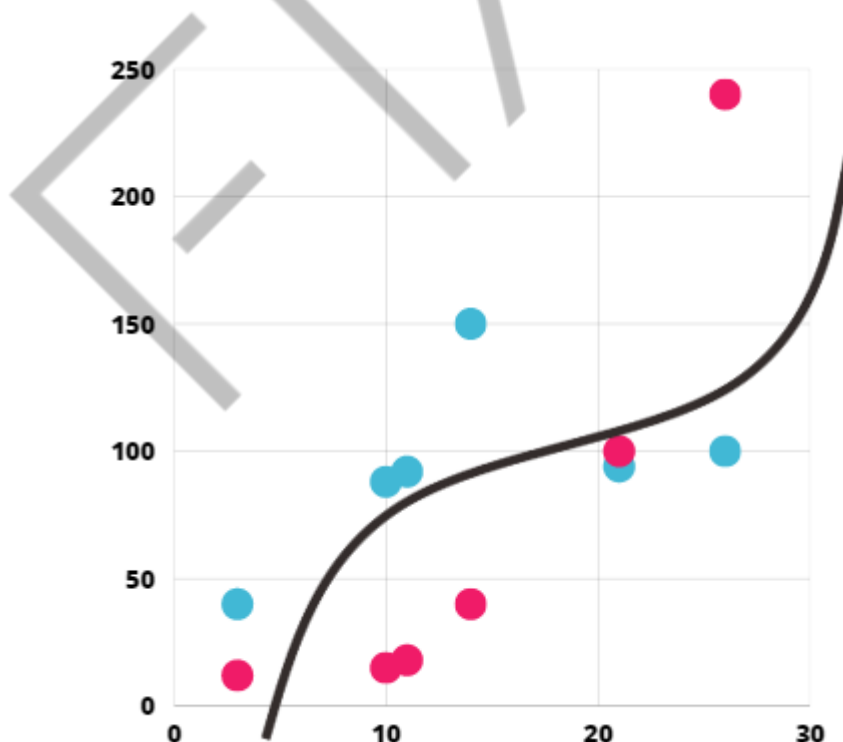


Figura 2 – Segundo modelo de classificação
Fonte: Elaborado pela autora (2023)

Para treinarmos alguns modelos de classificação, vamos utilizar a famosa biblioteca do Scikit-learn. O Scikit-learn é uma biblioteca de aprendizado de máquina de código aberto, que suporta aprendizado supervisionado e não supervisionado. Ele também fornece várias ferramentas para ajuste de modelo, pré-processamento de dados, seleção de modelo, avaliação de modelo e muitos outros utilitários.

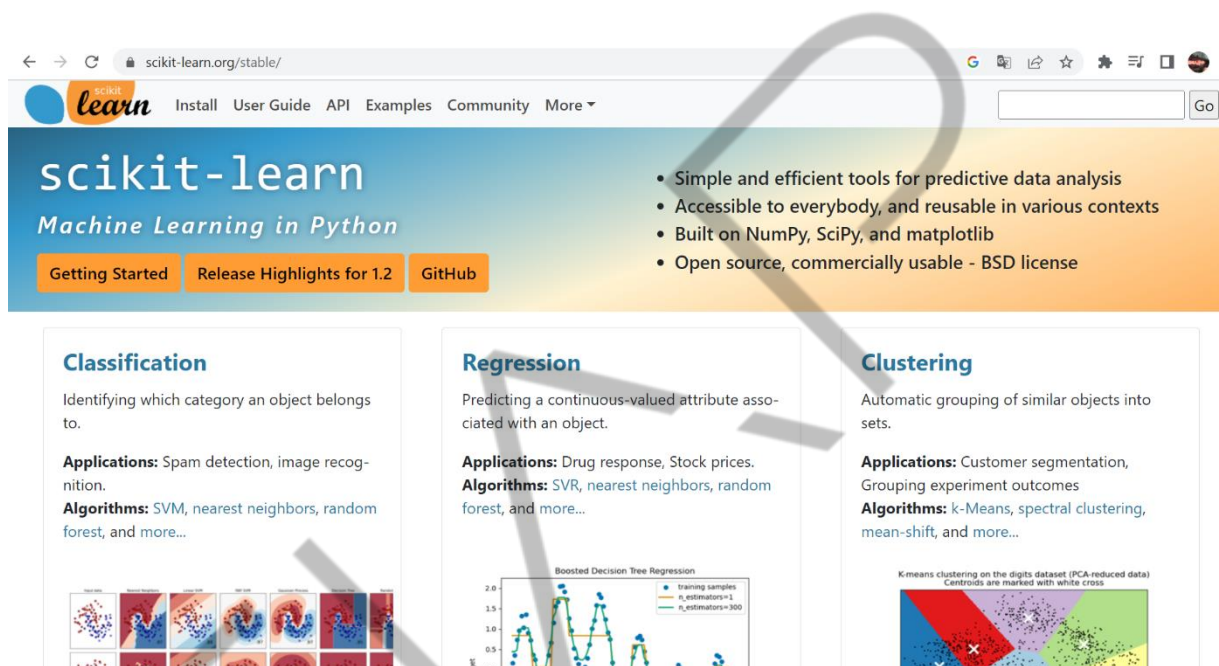


Figura 3 - Scikit-Learn
Fonte: Scikit-learn (2023)

Pré-processamento de dados em modelos de classificação

Observe que, no problema de classificação, queremos prever uma classe, ou seja, uma variável categórica no formato de string (texto). É muito importante lembrarmos que o computador não entende texto por si só, sendo assim, é preciso analisar as variáveis do nosso modelo e identificar a necessidade de transformá-las em formato numérico. Mas como eu posso fazer essa transformação?

Primeiramente, é importante destacar que alguns classificadores trabalham melhor com características binárias, pois ao converter dados categóricos em números, o classificador pode aprender erroneamente que um “dado é maior que o outro”. Veja no exemplo abaixo a técnica **Label Encoding**:

Label Encoding		
Fruta	Calorias	Categoria
Morango	45	1
Abacaxi	50	2
Maçã	52	3

Figura 4 - Exemplo de Label Encoding
Fonte: Elaborado pela autora (2023)

A técnica de Label Encoding é útil quando as categorias não têm qualquer ordem intrínseca. É possível observar no exemplo, que as frutas estão ordenadas pela ordem da categoria, mas se analisarmos a questão das calorias, não necessariamente temos a ordem da categoria representando qual fruta possui maior caloria. Para resolver esse caso, podemos optar pela técnica **One Hot Encoding**.

A técnica de One Hot Encoding consiste na criação de novas colunas a partir das categorias. Cada uma delas se torna uma nova coluna e o valor na linha correspondente será 1, caso tenha a presença da característica. Do contrário, será 0. Veja no exemplo da Figura 5 – “Exemplo de One Hot Encoding”:

One Hot Encoding			
Morango	Abacaxi	Maçã	Calorias
1	0	0	45
0	1	0	50
0	0	1	52

Figura 5 - Exemplo de One Hot Encoding
Fonte: Elaborado pela autora (2023)

Essa técnica precisa ser utilizada com sabedoria, pois, observe que, ao criar uma nova coluna para cada categoria, a dimensão dos dados aumenta. Isso pode ser um desafio em modelos de Machine Learning e pode levar ao problema da **maldição da dimensionalidade** dos dados. Modelos com muitas features (colunas) podem tornar o treinamento complexo, e, nesse caso, teríamos que optar por **técnicas de redução de dimensionalidade** para reduzir o volume de variáveis de forma significativa.

Agora, vamos visualizar no Python como podemos aplicar essas técnicas. Para exemplificar Label Encoding e One Hot Encoding, vamos utilizar a biblioteca **preprocessing** do **Sklearn**, importando o **LabelEncoder** para o One Hot Encoding. Vamos utilizar da própria biblioteca do **pandas**, o **get_dummies**. Para realizar esse exemplo, vamos utilizar a base de dados *Frutas.xlsx*.

```
# importando bibliotecas necessárias
import pandas as pd
```

```
#Subindo a base dos dados:
df = pd.read_excel('Frutas.xlsx')
```

```
df.head() #visualizando os dados
```

	Fruta	Calorias
0	Abacaxi	48
1	Amora	43
2	Abacate	160
3	Banana-prata	89
4	Banana-nanica	92

Figura 6 – Primeiro exemplo de Label Encoding
Fonte: Elaborado pela autora (2023)

```
from sklearn.preprocessing import LabelEncoder #importando LabelEncoder
```

```
# Instanciando o LabelEncoder
labelencoder = LabelEncoder()
```

```
# Atribuindo valores numéricos e armazenando em outra coluna
df['CategoriasFrutas'] = labelencoder.fit_transform(df['Fruta'])
```

```
df.head() #visualizando o resultado com LabelEncoder
```

	Fruta	Calorias	CategoriasFrutas
0	Abacaxi	48	1
1	Amora	43	4
2	Abacate	160	0
3	Banana-prata	89	7
4	Banana-nanica	92	6

Figura 7 – Segundo exemplo de Label Encoding
Fonte: Elaborado pela autora (2023)

```
#Subindo a base dos dados:
df = pd.read_excel('Frutas.xlsx')
```

```
# criando a classificação binária por tipo de categoria
dum_df = pd.get_dummies(df, columns=["Fruta"])
```

```
dum_df
```

	Calorias	Fruta_Abacate	Fruta_Abacaxi	Fruta_Acerola	Fruta_Ameixa	Fruta_Amora	Fruta_Banana-maçã	Fruta_Banana-nanica	Fruta_Banana-prata	Fruta_Caju	Fruta_Caqui
0	48	0	1	0	0	0	0	0	0	0	0
1	43	0	0	0	0	1	0	0	0	0	0
2	160	1	0	0	0	0	0	0	0	0	0
3	89	0	0	0	0	0	0	0	1	0	0
4	92	0	0	0	0	0	0	1	0	0	0
5	89	0	0	0	0	0	1	0	0	0	0
6	46	0	0	0	0	0	0	0	0	0	0
7	70	0	0	0	0	0	0	0	0	0	1

Figura 8 - Exemplo de One Hot Encoding
 Fonte: Elaborado pela autora (2023)

Pré-processamento de dados em colunas numéricas

Já que entramos no assunto de tratar variáveis categóricas, vale relembrar sobre o processo de tratar variáveis numéricas, o chamado processo de **feature scaling**. Alguns modelos de Machine Learning exigem que os valores estejam em escalas similares para que eles não se tornem tendenciosos. Sendo assim, podemos destacar algumas principais técnicas de padronização e normalização dos dados: **MinMaxScaler** e **StandardScaler**, da biblioteca do Sklearn!

MinMax Scaler:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Figura 9 - MinMax Scaler
 Fonte: Elaborado pela autora (2023)

$$z = \frac{x - \mu}{\sigma}$$

Figura 10 - Standard Scaler
 Fonte: Elaborado pela autora (2023)

Vamos utilizar um exemplo para compreensão das escalas. Suponhamos que eu tenha uma base de dados sobre os meus clientes, que possui uma coluna de salário e outra de idade:

Cliente	Idade	Salário
123	22	2300
321	30	20000
445	25	18000
256	18	5000

Figura 11 - Exemplo de Feature Scaling
Fonte: Elaborado pela autora (2023)

Analisando as colunas, se atribuíssemos a um algoritmo classificador as variáveis salário e idade, sem normalizar ou padronizar os dados, poderíamos correr o risco de o classificador entender que a variável salário pode ser mais “importante” que a variável idade, sendo que isso pode não ser verdade. É nesse momento que optamos por utilizar as técnicas de Feature Scaling.

Cliente	Idade	Salário	Salário Padronizado	Idade Padronizada
123	22	2300	0,26	-0,35
321	30	20000	2,23	1,24
445	25	18000	2,01	0,25
256	18	5000	0,56	-1,14

Figura 12 - Exemplo de z-score (padronização)
Fonte: Elaborado pela autora (2023)

Cliente	Idade	Salário	Salário Normalizado	Idade Normalizada
123	22	2300	0,00	0,33
321	30	20000	1,00	1,00
445	25	18000	0,89	0,58
256	18	5000	0,15	0,00

Figura 13 - Exemplo de min-max (normalização)
Fonte: Elaborado pela autora (2023)

Como treinar um classificador

Entendemos como podemos identificar um problema de classificação, mas como eu posso criar um classificador? Antes de falarmos sobre quais são os modelos de classificação, vamos entender como treinar um modelo classificador.

Para ensinar um algoritmo a aprender a classificar dados, basicamente **devemos separar os dados em treino e teste**. Você deve estar se questionando: mas por que devo separar dessa forma? Vamos utilizar mais uma analogia da vida real para melhor compreensão. Quando queremos ensinar uma criança o que é uma maçã, atribuímos amostras de imagens ou até mesmo a própria fruta para a criança observar e correlacionar que essa imagem é uma maçã. Fazemos isso várias vezes até ela aprender. Para testar se a criança aprendeu de fato o que é uma maçã, testamos novamente mostrando a fruta e, quando ela acerta, logo é atribuído um “parabéns, você acertou!”. É dessa forma que o algoritmo aprende, damos a ele um **conjunto de amostras de treinamento** e posteriormente **testamos se de fato esse algoritmo está preciso**.

A separação da base consiste entre **70% ~80% para treinar** o algoritmo, e o resto da base entre **20% ~30% para teste**.

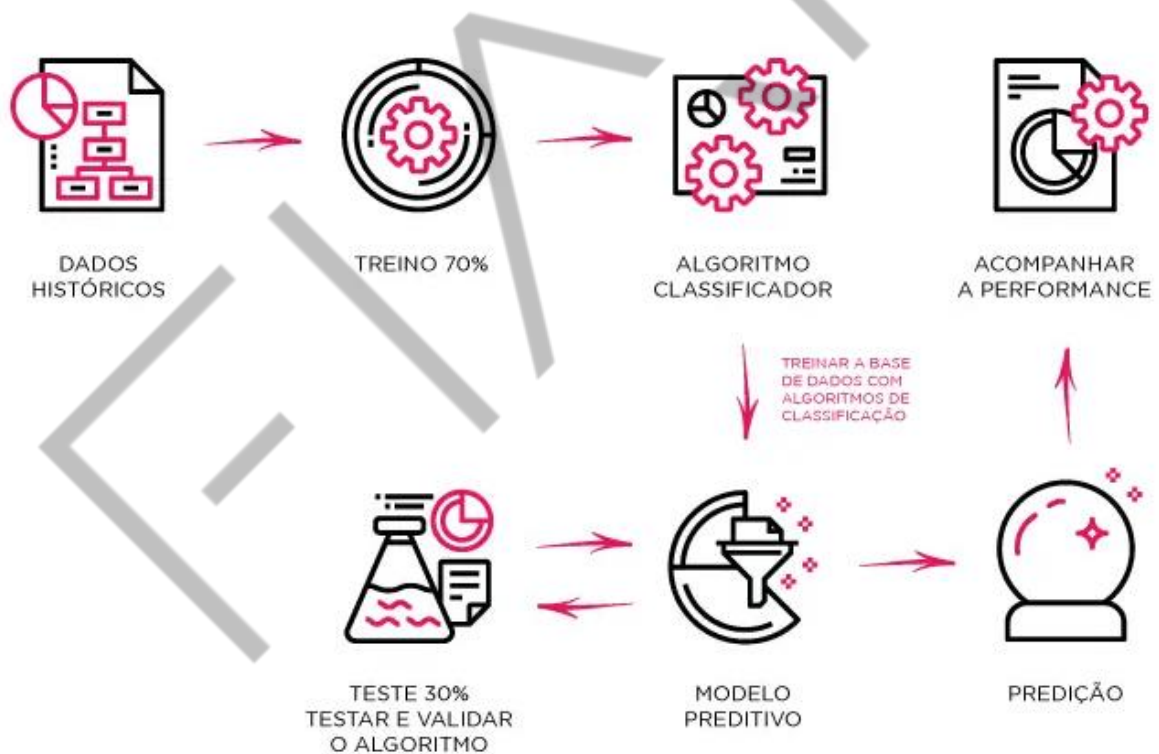


Figura 14 - Treinamento e teste do algoritmo
Fonte: Elaborado pela autora (2023)

Configuramos a maior parte da base de treinamento do algoritmo porque precisamos garantir que ele conseguirá aprender os padrões dos dados. A menor parte é atribuída na parte de teste, com o objetivo de testar se o algoritmo tem um

nível de assertividade eficiente em uma base de dados “nunca vista antes”, ou seja, novos dados para teste.

No Scikit-learn podemos importar da biblioteca **model_selection** a ferramenta **train_test_split** para realizar essa separação dos dados de forma muito simples, conforme é possível visualizar na Figura 15 – Separação dos dados em treino e teste.

```
[11] x = dados[['Comprimento do Abdômen', 'Comprimento das Antenas']]  
      y = dados['Espécie']  
  
[12] x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y,  
                                                         random_state=42)
```

Figura 15 - Separação dos dados em treino e teste
Fonte: Elaborado pela autora (2023)

Bem, e o que seria essa divisão do treino e teste utilizando “x” e “y”? Essas variáveis são utilizadas de forma representativa para armazenar os dados, sendo variáveis independentes armazenadas em x e variável dependente em y.

A ferramenta **train_test_split** distribuirá os dados de x e y entre **x_train**, **y_train**, **x_test** e **y_test**, de acordo com o tamanho da base configurado em **test_size**. Perceba que nosso **test_size** está configurado para obter 20% (menor parte) do total da base, ou seja, deixando assim a base de treino com 80% dos dados (maior parte).

O hiperparâmetro **random_state** configura a aleatoriedade dos dados no momento da distribuição entre as bases de treino e teste. O **stratify** é responsável por equilibrar as categorias entre as bases de treino e teste.

Pareceu meio confuso? Calma que iremos te ajudar! Nas próximas videoaulas, vamos aprender na prática como utilizar essa separação de base e entender passo a passo o uso de cada hiperparâmetro. Vamos lá?!

HANDS ON

Agora, chegou o momento de ver, na prática, como podemos identificar um problema de classificação e quais técnicas podemos aplicar na base antes de realizar o treinamento de um algoritmo.

Para essa aula, temos alguns notebooks para você. Acesse abaixo:

[Notebook 1](#)

[Notebook 2](#)

Além disso, também disponibilizamos as bases de dados, para te ajudar com os estudos e exercícios:

[Base de dados 1](#)

[Base de dados 2](#)

O QUE VOCÊ VIU NESTA AULA?

Como identificar problemas de classificação em Machine Learning; como realizar o pré-processamento de dados antes de construir um algoritmo; como um algoritmo supervisionado funciona; como separar os dados em treino e teste.

Daqui para a frente, é importante que você replique os conhecimentos adquiridos para fortalecer mais suas bases e conhecimentos.

IMPORTANTE: não esqueça de praticar com o desafio da disciplina, para que assim você possa aprimorar os seus conhecimentos!

Você não está só nesta jornada! Te esperamos no Discord e nas lives com os nossos especialistas, onde você poderá tirar dúvidas, compartilhar conhecimentos e estabelecer conexões!

REFERÊNCIAS

DOCUMENTAÇÃO SCIKIT-LEARN. Disponível em: <<https://scikit-learn.org/stable/>>. Acesso em: 10 mar 2023.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. 2nd Edition. [s.l.]: O'Reilly Media, Inc., 2019.

EMEND

PALAVRAS-CHAVE

Modelo Supervisionado de Classificação. Sklearn. Train Test Split.

EMEND

The background is a dark blue field filled with numerous small, light blue dots. Overlaid on this are several large, flowing, wavy lines in shades of teal and yellow. A vertical line on the left side has a small 'x' at its base. A circle containing the number '7' is positioned in the upper center, with a thin line extending downwards from it. In the bottom right corner, there is a faint, light blue hexagonal shape.

POSTECH