APRESENTAÇÃO
Avaliação 02 – Parte 2

DATASET

Nome: Forest Fires Data Set

Descrição: objetivo é prever a área queimada de incêndios florestais, na região nordeste de Portugal, usando dados meteorológicos e outros

Volume de dados: Número de instâncias: **517**; Número de atributos: **13**

Link: https://archive.ics.uci.edu/ml/datasets/Forest+Fires

DATASET

Principais colunas e seus significados:

1. **X** - Coordenada espacial do eixo x no mapa do parque Montesinho: 1 a 9
2. **Y** - Coordenada espacial do eixo y no mapa do parque Montesinho: 2 a 9
3. **month** - mês do ano: "jan" a "dec"
4. **day** - dia da semana: "mon" a "sol"
5. **FFMC** - Índice FFMC do sistema FWI: 18,7 a 96,20
6. **DMC** - Índice DMC do sistema FWI: 1.1 a 291.3
7. **DC** - Índice DC do sistema FWI: 7,9 a 860,6
8. **ISI** - Índice ISI do sistema FWI: 0,0 a 56,10
9. **temp** - temperatura em graus Celsius: 2,2 a 33,30
10. **RH** - umidade relativa em%: 15,0 a 100
11. **wind** - Velocidade do vento em km / h: 0,40 a 9,40
12. **rain** - chuva externa em mm / m2: 0,0 a 6,4
13. **area** - A área queimada da floresta (em ha): 0,00 a 1090,84

DATASET

Principais colunas e seus significados:

FFMC (Fine Fuel Moisture Code) - um indicador do potencial de incêndios que podem ser espalhados na área atingida.
DMC (DUFF Moisture Code) - indica o consumo de combustível da camada orgânica da superfície com baixa densidade aparente.
DC (Drought Code) - indicador útil dos efeitos sazonais da seca sobre os combustíveis florestais.
ISI (Initial Spread Index) - índice de propagação inicial (ISI) é uma classificação numérica da taxa esperada de propagação do fogo.

DATASET

Objetivo do dataset, qual o resultado esperado após os algoritmos?

Objetivo é prever a área queimada de incêndios florestais, na região nordeste de Portugal, usando dados meteorológicos e outros usando ferramentas automáticas baseadas em sensores locais.

Espera-se que ao final dos testes com os algoritmos obtenha-se (a partir de uma detecção rápida) a estimativa de quantos porcentos de área queimada é possível no local.

ANÁLISE DOS DADOS

Tipos das colunas

1. **X** int64
2. **Y** int64
3. **month** object
4. **day** object
5. **FFMC** float64
6. **DMC** float64
7. **DC** float64
8. **ISI** float64
9. **temp** float64
10. **RH** int64
11. **wind** float64
12. **rain** float64
13. **area** float64

ANÁLISE DOS DADOS

No Jupyter Notebook

In [1]:
```
import numpy as np
import pandas as pd

dt = pd.read_csv('forestfires.csv')
dt.head()

#TIPOS DAS COLUNAS
print("\nVARIAVEL  -  TIPO\n", dt.dtypes)
```

```
VARIAVEL  -  TIPO
 X          int64
Y           int64
month      object
day        object
FFMC      float64
DMC       float64
DC        float64
ISI       float64
temp      float64
RH          int64
wind      float64
rain      float64
area      float64
dtype: object
```

ANÁLISE DOS DADOS

Campos Nulos:

Não existem

In [2]:
```
import numpy as np
import pandas as pd

dt = pd.read_csv('forestfires.csv')
dt.head()

#CAMPOS NULOS
print("\nNulos\n ", dt.isnull().sum())
```

```
Nulos
  X        0
Y          0
month      0
day        0
FFMC       0
DMC        0
DC         0
ISI        0
temp       0
RH         0
wind       0
rain       0
area       0
dtype: int64
```

ANÁLISE DOS DADOS

Variáveis Categóricas:

month, day

```
In [3]: import numpy as np
        import pandas as pd

        dt = pd.read_csv('forestfires.csv')
        dt.head()

        print(dt)
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.00
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.00
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.00
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.00
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.00
10	7	5	sep	sat	92.5	88.0	698.6	7.1	17.8	51	7.2	0.0	0.00
11	7	5	sep	sat	92.8	73.2	713.0	22.6	19.3	38	4.0	0.0	0.00
12	6	5	aug	fri	63.5	70.8	665.3	0.8	17.0	72	6.7	0.0	0.00
13	6	5	sep	mon	90.9	126.5	686.5	7.0	21.3	42	2.2	0.0	0.00
14	6	5	sep	wed	92.9	133.3	699.6	9.2	26.4	21	4.5	0.0	0.00
15	6	5	sep	fri	93.3	141.2	713.9	13.9	22.9	44	5.4	0.0	0.00
16	5	5	mar	sat	91.7	35.8	80.8	7.8	15.1	27	5.4	0.0	0.00
17	8	5	oct	mon	84.9	32.8	664.2	3.0	16.7	47	4.9	0.0	0.00
18	6	4	mar	wed	89.2	27.9	70.8	6.3	15.9	35	4.0	0.0	0.00
19	6	4	apr	sat	86.3	27.4	97.1	5.1	9.3	44	4.5	0.0	0.00

ANÁLISE DOS DADOS

Conversões entre tipos de colunas

In [10]:
```
import numpy as np
import pandas as pd

dt = pd.read_csv('forestfires.csv')
dt.head()

meses = dt[['month']]
conv_meses = pd.get_dummies(meses).astype(int)

print("\nConversao de dados:\n", conv_meses)
```

```
Conversao de dados:
      month_apr  month_aug  month_dec  month_feb  month_jan  month_jul  \
0             0          0          0          0          0          0
1             0          0          0          0          0          0
2             0          0          0          0          0          0
3             0          0          0          0          0          0
4             0          0          0          0          0          0
5             0          1          0          0          0          0
6             0          1          0          0          0          0
7             0          1          0          0          0          0
8             0          0          0          0          0          0
9             0          0          0          0          0          0
10            0          0          0          0          0          0
11            0          0          0          0          0          0
12            0          1          0          0          0          0
13            0          0          0          0          0          0
14            0          0          0          0          0          0
15            0          0          0          0          0          0
```

ANÁLISE DOS DADOS

Conversões entre tipos de colunas

In [10]:
```
import numpy as np
import pandas as pd

dt = pd.read_csv('forestfires.csv')
dt.head()

meses = dt[['month']]
conv_meses = pd.get_dummies(meses).astype(int)

print("\nConversao de dados:\n", conv_meses)
```

```
Conversao de dados:
      month_apr  month_aug  month_dec  month_feb  month_jan  month_jul  \
0             0          0          0          0          0          0
1             0          0          0          0          0          0
2             0          0          0          0          0          0
3             0          0          0          0          0          0
4             0          0          0          0          0          0
5             0          1          0          0          0          0
6             0          1          0          0          0          0
7             0          1          0          0          0          0
8             0          0          0          0          0          0
9             0          0          0          0          0          0
10            0          0          0          0          0          0
11            0          0          0          0          0          0
12            0          1          0          0          0          0
13            0          0          0          0          0          0
14            0          0          0          0          0          0
15            0          0          0          0          0          0
```

ANÁLISE DOS DADOS

Tecnica de análise: NLTK (Natural Language Toolkit)

In [2]:
```
#NOVA ANALISE DE DADOS
import numpy as np
import pandas as pd
import nltk

dt = pd.read_csv('forestfires.csv')
dt.head()

entidades = nltk.pos_tag(dt['month'])
print(entidades)
```

```
[('mar', 'NN'), ('oct', 'MD'), ('oct', 'VB'), ('mar', 'NN'), ('mar', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('se
p', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('aug', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('mar', 'N
N'), ('oct', 'NN'), ('mar', 'NN'), ('apr', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('jun', 'NN'), ('aug', 'NN'), ('aug', 'NN'),
('aug', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep',
'NN'), ('sep', 'NN'), ('sep', 'NN'), ('oct', 'NN'), ('oct', 'NN'), ('oct', 'NN'), ('mar', 'NN'), ('jul', 'NN'), ('aug', 'NN'),
('aug', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('jul', 'NN'), ('mar', 'NN'), ('mar', 'NN'), ('sep',
'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('oct', 'NN'), ('feb', 'NN'),
('feb', 'NN'), ('mar', 'NN'), ('mar', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('sep', 'NN'), ('sep',
'NN'), ('sep', 'NN'), ('mar', 'NN'), ('mar', 'NN'), ('sep', 'NN'), ('mar', 'NN'), ('aug', 'NN'), ('sep', 'NN'), ('feb', 'NN'),
('feb', 'NN'), ('mar', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug', 'NN'), ('aug',
'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('sep', 'NN'), ('mar', 'NN'), ('aug', 'NN'), ('mar', 'NN'), ('aug', 'NN'),
```

ANÁLISE DOS DADOS

Penn Part of Speech Tags

Note: these are the 'modified' tags used for Penn tree banking; these are the tags used in the Jet system. NP, NPS, PP, and PP$ from the original Penn part-of-speech tagging were changed to NNP, NNPS, PRP, and PRP$ to avoid clashes with standard syntactic categories.

1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	to
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

EXECUÇÃO DOS ALGORITMOS

Listagem dos algoritmos aplicáveis:

- MultinomialNB,
- AdaBoostClassifier,
- OneVsRest,
- OneVsOne
- **OutputCode**.

EXECUÇÃO DOS ALGORITMOS

Algoritmo ainda não visto na disciplina:

- OutputCode.

In [8]:
```
import numpy as np
import pandas as pd
from sklearn.svm import LinearSVC

dt = pd.read_csv('forestfires.csv')
meses = dt[['X', 'Y','month','FFMC','DMC','DC','ISI','temp','RH','wind','rain','area']]
dias = dt['day']

conv_meses = pd.get_dummies(meses).astype(int)

M = conv_meses.values
D = dias.values

porcentagem_treino = 0.8
porcentagem_teste = 0.1

tamanho_de_treino = int(porcentagem_treino * len(D))
tamanho_de_teste = int(porcentagem_teste * len(D))

treino_dados = M[:tamanho_de_treino]
treino_marcacoes = D[:tamanho_de_treino]

fim_de_treino = tamanho_de_treino + tamanho_de_teste

teste_dados = M[tamanho_de_teste:fim_de_treino]

teste_marcacoes = D[tamanho_de_teste:fim_de_treino]
```

EXECUÇÃO DOS ALGORITMOS

Algoritmo ainda não visto na disciplina:

- OutputCode.

Acerto OutputCode: 18.16%
Numero de acertos: 75

```
porcentagem_treino = 0.8
porcentagem_teste = 0.1

tamanho_de_treino = int(porcentagem_treino * len(D))
tamanho_de_teste = int(porcentagem_teste * len(D))

treino_dados = M[:tamanho_de_treino]
treino_marcacoes = D[:tamanho_de_treino]

fim_de_treino = tamanho_de_treino + tamanho_de_teste

teste_dados = M[tamanho_de_teste:fim_de_treino]

teste_marcacoes = D[tamanho_de_teste:fim_de_treino]


def fit_and_predict(nome, modelo, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes):
    modelo.fit(treino_dados, treino_marcacoes)
    resultado = modelo.predict(teste_dados)
    acertos = 0
    tamanho = len(teste_marcacoes)

    for i in range(tamanho):
        if teste_marcacoes[i] == resultado[i]:
            acertos = acertos + 1

    print('Acerto %s: %.2f%%' % (nome, (acertos* 100/ tamanho)))
    print("Numero de acertos: ", acertos)

from sklearn.multiclass import OutputCodeClassifier
modeloOutputCode = OutputCodeClassifier(LinearSVC(random_state = 0))
resultadoOutputCode = fit_and_predict("OutputCode", modeloOutputCode, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes)
```

```
Acerto OutputCode: 18.16%
Numero de acertos:  75
```

EXECUÇÃO DOS ALGORITMOS

Uso de k-folding:

In [13]:

```
import numpy as np
import pandas as pd
from sklearn.cross_validation import cross_val_score

dt = pd.read_csv('forestfires.csv')
meses = dt[['X', 'Y','month']]
dias = dt['day']

conv_meses = pd.get_dummies(meses).astype(int)
M = conv_meses.values
D = dias.values

porcentagem_treino = 1
tamanho_treino = int(porcentagem_treino * len(D))
tamanho_validacao = len(D) - tamanho_treino

dados_treino = M[:tamanho_treino]
marcacoes_treino = D[:tamanho_treino]

dados_teste = M[-tamanho_validacao:]
marcacoes_teste = D[-tamanho_validacao:]

from sklearn.naive_bayes import MultinomialNB
modelo = MultinomialNB()
k = 6
scores = cross_val_score(modelo,dados_treino,marcacoes_treino,cv = k)

print(scores)
taxa_de_acerto = np.mean(scores)
print("Taxa de acerto: ", round(taxa_de_acerto,2))
```

```
[ 0.21348315  0.1954023   0.12790698  0.1627907   0.2         0.16666667]
Taxa de acerto:  0.18
```

EXECUÇÃO DOS ALGORITMOS

Uso de k-folding:

Taxa de acerto (MultinomialNB): 17.77 %

Taxa de acerto (AdaBoost): 13.59 %

Taxa de acerto (OneVsRest): 16.8 %

Taxa de acerto (OneVsOne): 17.2 %

Taxa de acerto (OutputCode): 11.01 %

APRESENTAÇÃO DOS RESULTADOS

Gráfico de comparação da performance dos algoritmos:

```
dt = pd.read_csv('forestfires.csv')
dt.head()

dt = pd.read_csv('forestfires.csv')
meses = dt[['X', 'Y','month','FFMC','DMC','DC','ISI','temp','RH','wind','rain','area']]
dias = dt['day']

conv_meses = pd.get_dummies(meses).astype(int)

M = conv_meses.values
D = dias.values

porcentagem_treino = 0.8
porcentagem_teste = 0.1

tamanho_de_treino = int(porcentagem_treino * len(D))
tamanho_de_teste = int(porcentagem_teste * len(D))


treino_dados = M[:tamanho_de_treino]
treino_marcacoes = D[:tamanho_de_treino]

fim_de_treino = tamanho_de_treino + tamanho_de_teste

teste_dados = M[tamanho_de_teste:fim_de_treino]
teste_marcacoes = D[tamanho_de_teste:fim_de_treino]


def fit_and_predict(nome, modelo, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes):
    modelo.fit(treino_dados, treino_marcacoes)
    resultado = modelo.predict(teste_dados)
    acertos = 0
    tamanho = len(teste_marcacoes)

    for i in range(tamanho):
        if teste_marcacoes[i] == resultado[i]:
            acertos = acertos + 1

    print('Acerto %s: %.2f%%' % (nome, (acertos* 100/ tamanho)))
    print("Total de acertos: ", acertos)
```

APRESENTAÇÃO DOS RESULTADOS

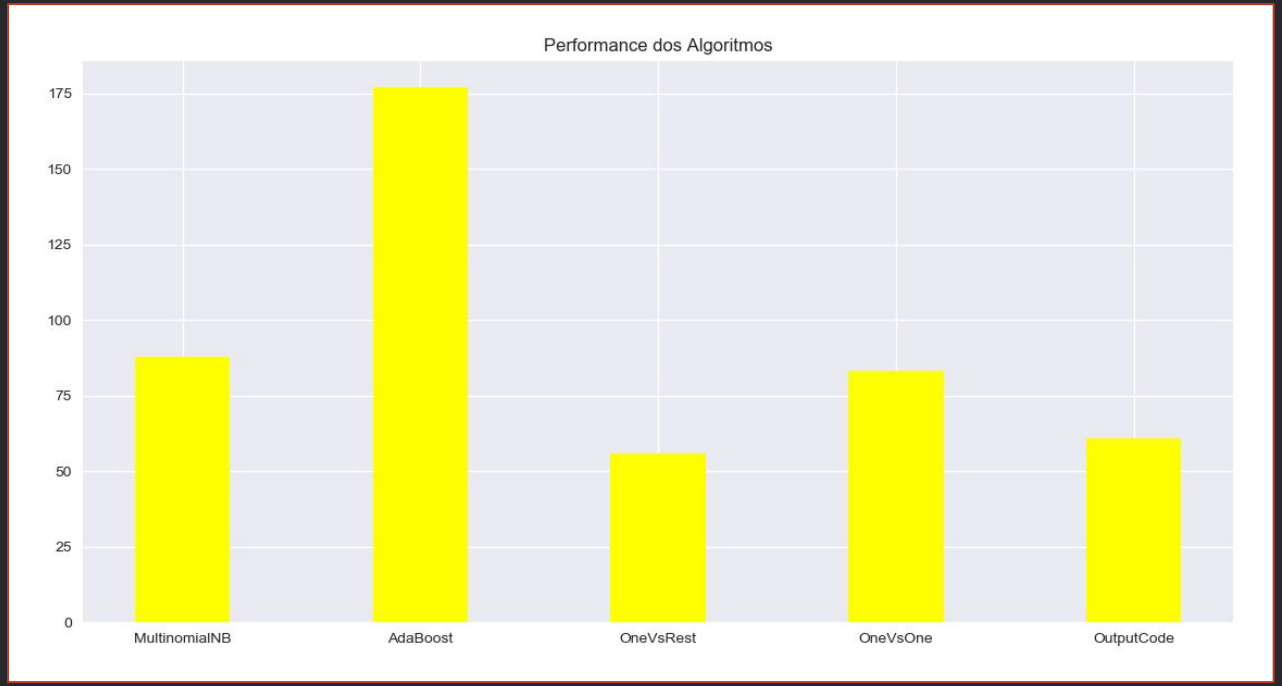
Gráfico de comparação da performance dos algoritmos:

```
from sklearn.naive_bayes import MultinomialNB
modeloMultinomial = MultinomialNB()
resultadoMultinomialNB = fit_and_predict("MultinomialNB", modeloMultinomial, treino_dados, treino_marcacoes, teste_dados, teste_
marcacoes)

from sklearn.ensemble import AdaBoostClassifier
modeloAdaBoost = AdaBoostClassifier()
resultadoAdaBoost = fit_and_predict("AdaBoostClassifier", modeloAdaBoost, treino_dados, treino_marcacoes, teste_dados, teste_mar
cacoes)

from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
modeloOneVsRest = OneVsRestClassifier(LinearSVC(random_state = 0))
resultadoOneVsRest = fit_and_predict("OneVsRest", modeloOneVsRest, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes)

from sklearn.multiclass import OneVsOneClassifier
modeloOneVsOne = OneVsOneClassifier(LinearSVC(random_state = 0))
resultadoOneVsOne = fit_and_predict("OneVsOne", modeloOneVsOne, treino_dados, treino_marcacoes, teste_dados, teste_marcacoes)

from sklearn.multiclass import OutputCodeClassifier
modeloOutputCode = OutputCodeClassifier(LinearSVC(random_state = 0))
resultadoOutputCode = fit_and_predict("OutputCode", modeloOutputCode, treino_dados, treino_marcacoes, teste_dados, teste_marcaco
es)
```

```
Acerto MultinomialNB: 21.31%
Total de acertos:  88
Acerto AdaBoostClassifier: 42.86%
Total de acertos:  177
Acerto OneVsRest: 13.56%
Total de acertos:  56
Acerto OneVsOne: 20.10%
Total de acertos:  83
Acerto OutputCode: 16.46%
Total de acertos:  68
```

APRESENTAÇÃO DOS RESULTADOS

Gráfico de comparação da performance dos algoritmos:



OBRIGADA!