TRABALHO DE IAA006 – Arquitetura de Dados

Equipe 03

- Gustavo Costa de Souza
- Marcos Vinicius de Melo
- Marcus Eneas Silveira Galvao do Rio Apa II
- Patrícia Verdugo Pascoal
- Rodrigo de Araujo
- William de Souza Alencar

Atividade 02 - melhorar o desempenho de RP em conjunto de dados existentes

A atividade 02 visa trabalhar com um conjunto de dados pré-construído, onde as opções que o desenvolvedor tem, são de aplicar as técnicas de pré-processamento abaixo relacionadas:

- Seleção
- Limpeza
- Codificação
- Enriquecimento
- Normalização
- Construção de Atributos
- Correção de Prevalência
- Partição do Conjunto de Dados

Busque uma base de dados na UCI Machine Learning que seja indicada para problemas de classificação. (https://archive.ics.uci.edu/datasets)

Para esse exemplo, vou usar a base de segmentação de imagens (https://archive.ics.uci.edu/dataset/50/image+segmentation)

Baixando o dataset direto do site da UCI.

In [188…

```
# base de dados disponível na UCI Machine Learning - https://archive.ics.uci.edu/dataset/50/image+segmentation

from ucimlrepo import fetch_ucirepo
import pandas as pd

# fetch dataset
img_segmentation_repo = fetch_ucirepo(id=50)

# data (as pandas dataframes)
img_seg_features = img_segmentation_repo.data.features
img_seg_target = img_segmentation_repo.data.targets

img_seg_df = pd.concat([img_seg_features, img_seg_target], axis=1)
```

```python
# metadata
print(img_segmentation_repo.metadata)

# variable information
print(img_segmentation_repo.variables)
```

{'uci_id': 50, 'name': 'Image Segmentation', 'repository_url': 'https://archive.ics.uci.edu/dataset/50/image+segmentation', 'data_url': 'https://archive.ics.uci.edu/static/public/50/data.csv', 'abstract': 'Image data described by high-level numeric-valued attributes, 7 classes', 'area': 'Other', 'tasks': ['Classification'], 'characteristics': ['Multivariate'], 'num_instances': 2310, 'num_features': 19, 'feature_types': ['Real'], 'demographics': [], 'target_col': ['class'], 'index_col': None, 'has_missing_values': 'no', 'missing_values_symbol': None, 'year_of_dataset_creation': 1990, 'last_updated': 'Fri Oct 27 2023', 'dataset_doi': '10.24432/C5GP4N', 'creators': [], 'intro_paper': None, 'additional_info': {'summary': 'The instances were drawn randomly from a database of 7 outdoor images. The images were handsegmented to create a classification for every pixel. \r\n\r\n   Each instance is a 3x3 region.', 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': '   1. region-centroid-col: the column of the center pixel of the region.\r\n   2. region-centroid-row: the row of the center pixel of the region.\r\n   3. region-pixel-count: the number of pixels in a region = 9.\r\n   4. short-line-density-5: the results of a line extractoin algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region.\r\n   5. short-line-density-2: same as short-line-density-5 but counts lines of high contrast, greater than 5.\r\n   6. vedge-mean: measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.\r\n   7. vegde-sd: (see 6)\r\n   8. hedge-mean: measures the contrast of vertically adjacent pixels. Used for horizontal line detection. \r\n   9. hedge-sd: (see 8).\r\n   10. intensity-mean: the average over the region of (R + G + B)/3\r\n   11. rawred-mean: the average over the region of the R value.\r\n   12. rawblue-mean: the average over the region of the B value.\r\n   13. rawgreen-mean: the average over the region of the G value.\r\n   14. exred-mean: measure the excess red: (2R - (G + B))\r\n   15. exblue-mean: measure the excess blue: (2B - (G + R))\r\n   16. exgreen-mean: measure the excess green: (2G - (R + B))\r\n   17. value-mean: 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics)\r\n   18. saturatoin-mean: (see 17)\r\n   19. hue-mean: (see 17)', 'citation': None}}

	name	role	type	demographic \
0	class	Target	Categorical	None
1	region-centroid-col	Feature	Continuous	None
2	region-centroid-row	Feature	Continuous	None
3	region-pixel-count	Feature	Continuous	None
4	short-line-density-5	Feature	Continuous	None
5	short-line-density-2	Feature	Continuous	None
6	vedge-mean	Feature	Continuous	None
7	vedge-sd	Feature	Continuous	None
8	hedge-mean	Feature	Continuous	None
9	hedge-sd	Feature	Continuous	None
10	intensity-mean	Feature	Continuous	None
11	rawred-mean	Feature	Continuous	None
12	rawblue-mean	Feature	Continuous	None
13	rawgreen-mean	Feature	Continuous	None
14	exred-mean	Feature	Continuous	None
15	exblue-mean	Feature	Continuous	None
16	exgreen-mean	Feature	Continuous	None
17	value-mean	Feature	Continuous	None

```
18      saturation-mean  Feature   Continuous        None
19             hue-mean  Feature   Continuous        None

                                          description units missing_values
0                                                None  None             no
1        the column of the center pixel of the region  None             no
2           the row of the center pixel of the region  None             no
3                the number of pixels in a region = 9  None             no
4   the results of a line extractoin algorithm tha...  None             no
5   same as short-line-density-5 but counts lines ...  None             no
6   measure the contrast of horizontally adjacent ...  None             no
7                                               see 6  None             no
8   measures the contrast of vertically adjacent p...  None             no
9                                               see 8  None             no
10       the average over the region of (R + G + B)/3  None             no
11        the average over the region of the R value.  None             no
12        the average over the region of the B value.  None             no
13        the average over the region of the G value.  None             no
14             measure the excess red:  (2R - (G + B))  None             no
15            measure the excess blue:  (2B - (G + R))  None             no
16           measure the excess green:  (2G - (R + B))  None             no
17  3-d nonlinear transformation of RGB. (Algorith...  None             no
18                                             see 17  None             no
19                                             see 17  None             no
```

In [189…

```
img_seg_df.head()
```

Out[189…

	region-centroid-col	region-centroid-row	region-pixel-count	short-line-density-5	short-line-density-2	vedge-mean	vedge-sd	hedge-mean	hedge-sd	intensity-mean	rawred-mean	rawblue-mean	rawgreen-mean	e
0	140.0	125.0	9	0.0	0.0	0.277778	0.062963	0.666667	0.311111	6.185185	7.333334	7.666666	3.555556	3.4
1	188.0	133.0	9	0.0	0.0	0.333333	0.266667	0.500000	0.077778	6.666666	8.333334	7.777778	3.888889	5.0
2	105.0	139.0	9	0.0	0.0	0.277778	0.107407	0.833333	0.522222	6.111111	7.555555	7.222222	3.555556	4.3
3	34.0	137.0	9	0.0	0.0	0.500000	0.166667	1.111111	0.474074	5.851852	7.777778	6.444445	3.333333	5.7
4	39.0	111.0	9	0.0	0.0	0.722222	0.374074	0.888889	0.429629	6.037037	7.000000	7.666666	3.444444	2.8

In [190…

```
img_seg_df.describe()
```

Out[190…

	region-centroid-col	region-centroid-row	region-pixel-count	short-line-density-5	short-line-density-2	vedge-mean	vedge-sd	hedge-mean	hedge-sd	intensity-mean	rawred-mean
count	210.000000	210.000000	210.0	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000	210.000000
mean	124.647619	122.757143	9.0	0.008466	0.006349	1.925132	5.719529	2.604233	11.638377	37.091005	32.967725
std	74.104024	58.139686	0.0	0.029549	0.030077	3.158211	43.495942	4.798268	97.390023	38.677168	35.540563
min	1.000000	11.000000	9.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	60.500000	81.500000	9.0	0.000000	0.000000	0.666667	0.400921	0.777779	0.410816	6.453704	7.000000
50%	123.500000	121.500000	9.0	0.000000	0.000000	1.222222	0.828695	1.388889	0.913176	21.314816	18.611112
75%	189.750000	174.500000	9.0	0.000000	0.000000	1.888890	1.676634	2.597221	1.980485	52.629629	46.750000
max	252.000000	250.000000	9.0	0.111111	0.222222	25.500000	572.996400	44.722225	1386.329200	143.444440	136.888890

In [191…

```
# Verificação do balanceamento das classes.
```

```
img_seg_df['class'].value_counts()
```

Out[191…

```
class
BRICKFACE    30
SKY          30
FOLIAGE      30
CEMENT       30
WINDOW       30
PATH         30
GRASS        30
Name: count, dtype: int64
```

Hora de realizar os tratamentos

no exemplo, iremos normalizar as colunas, remover a coluna de identificação e separar a classe dos atributos.

In [192…

```
# Tipos das colunas
img_seg_df.dtypes
```

Out[192…

```
region-centroid-col     float64
region-centroid-row     float64
region-pixel-count        int64
short-line-density-5    float64
short-line-density-2    float64
vedge-mean              float64
vedge-sd                float64
hedge-mean              float64
hedge-sd                float64
intensity-mean          float64
rawred-mean             float64
rawblue-mean            float64
rawgreen-mean           float64
exred-mean              float64
exblue-mean             float64
exgreen-mean            float64
value-mean              float64
saturation-mean         float64
hue-mean                float64
class                    object
dtype: object
```

In [193…

```
# Verificação de dados ausentes
img_seg_df.isnull().sum()
```

Out[193…
```
region-centroid-col     0
region-centroid-row     0
region-pixel-count      0
short-line-density-5    0
short-line-density-2    0
vedge-mean              0
vedge-sd                0
hedge-mean              0
hedge-sd                0
intensity-mean          0
rawred-mean             0
rawblue-mean            0
rawgreen-mean           0
exred-mean              0
exblue-mean             0
exgreen-mean            0
value-mean              0
saturation-mean         0
hue-mean                0
class                   0
dtype: int64
```

In [194…
```
# Verificação de colunas com dados únicos
img_seg_df.nunique()
```

Out[194…
```
region-centroid-col     139
region-centroid-row     139
region-pixel-count        1
short-line-density-5      2
short-line-density-2      3
vedge-mean              160
vedge-sd                202
hedge-mean              164
hedge-sd                202
intensity-mean          196
rawred-mean             160
rawblue-mean            175
rawgreen-mean           154
exred-mean              156
exblue-mean             168
exgreen-mean            151
value-mean              175
saturation-mean         202
hue-mean                202
class                     7
dtype: int64
```

In [195…
```
# removendo a feature com dados únicos
img_seg_df = img_seg_df.drop('region-pixel-count', axis=1)
img_seg_df.nunique()
```

Out[195...

```
region-centroid-col     139
region-centroid-row     139
short-line-density-5      2
short-line-density-2      3
vedge-mean              160
vedge-sd                202
hedge-mean              164
hedge-sd                202
intensity-mean          196
rawred-mean             160
rawblue-mean            175
rawgreen-mean           154
exred-mean              156
exblue-mean             168
exgreen-mean            151
value-mean              175
saturation-mean         202
hue-mean                202
class                     7
dtype: int64
```

In [196...

```
# verificação de valores com baixa representação ou ocorrência
num_linhas = img_seg_df.shape[0]
cols = []
for c in img_seg_df.columns:
    num_unicos = len( img_seg_df[c].unique() )
    percentage = float(num_unicos) / num_linhas * 100
    if percentage < 1:
        print('%s, %d, %.1f%%' % (c, num_unicos, percentage))
        cols.append(c)
```

```
short-line-density-5, 2, 1.0%
```

In [197...

```
# removendo feature com baixa representatividade
img_seg_df = img_seg_df.drop('short-line-density-5', axis=1)
```

In [198...

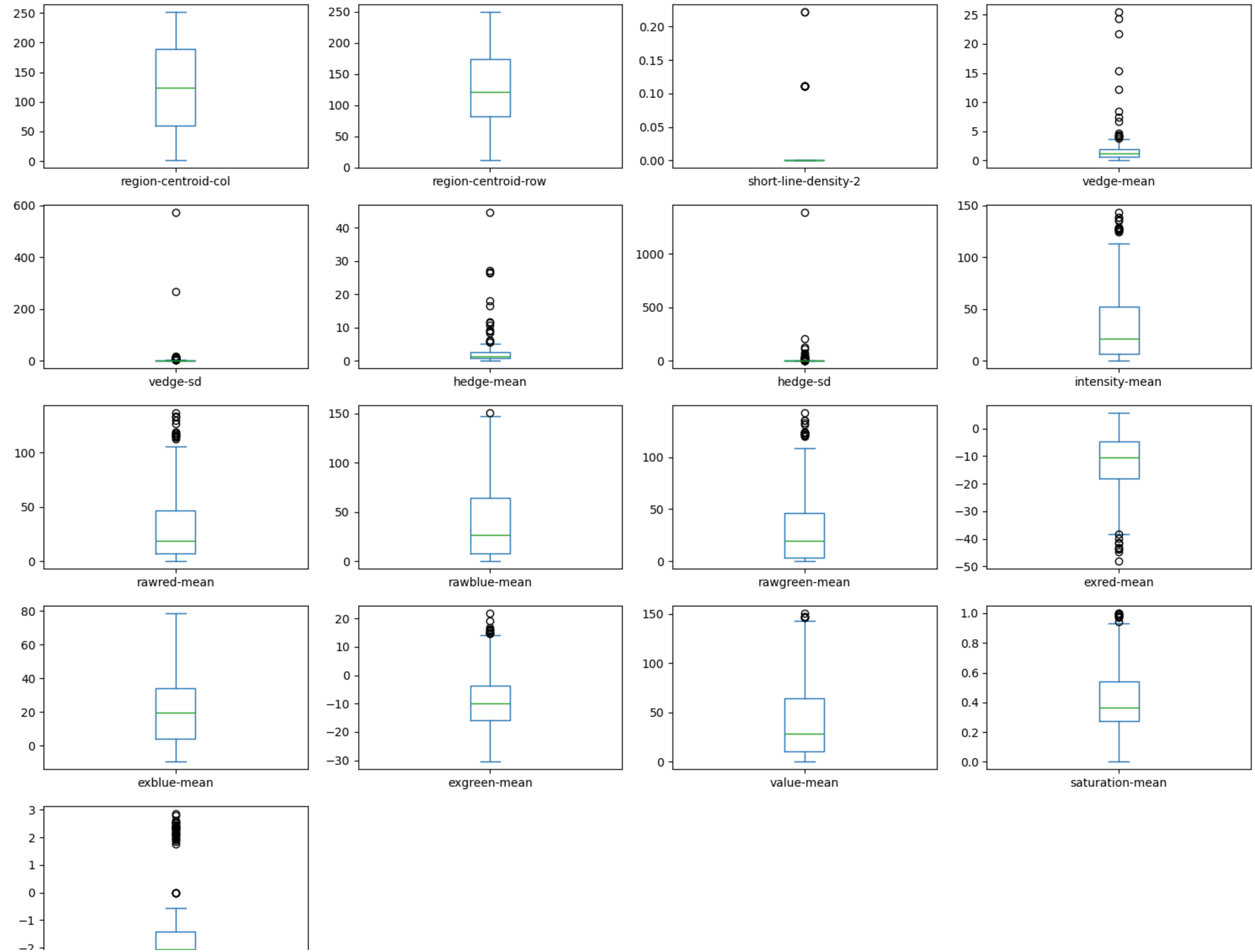
```
# Verificação e remoção de duplicados
print("Número de linhas duplicadas: ", img_seg_df.duplicated().sum())

img_seg_df_df_no_dups = img_seg_df.drop_duplicates()
```

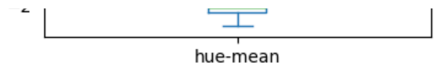
```
print("Total de padrões: ", img_seg_df.shape[0])
print("Total de padrões após remoção de duplicados: ", img_seg_df_df_no_dups.shape[0])
```

```
Número de linhas duplicadas:  0
Total de padrões:  210
Total de padrões após remoção de duplicados:  210
```

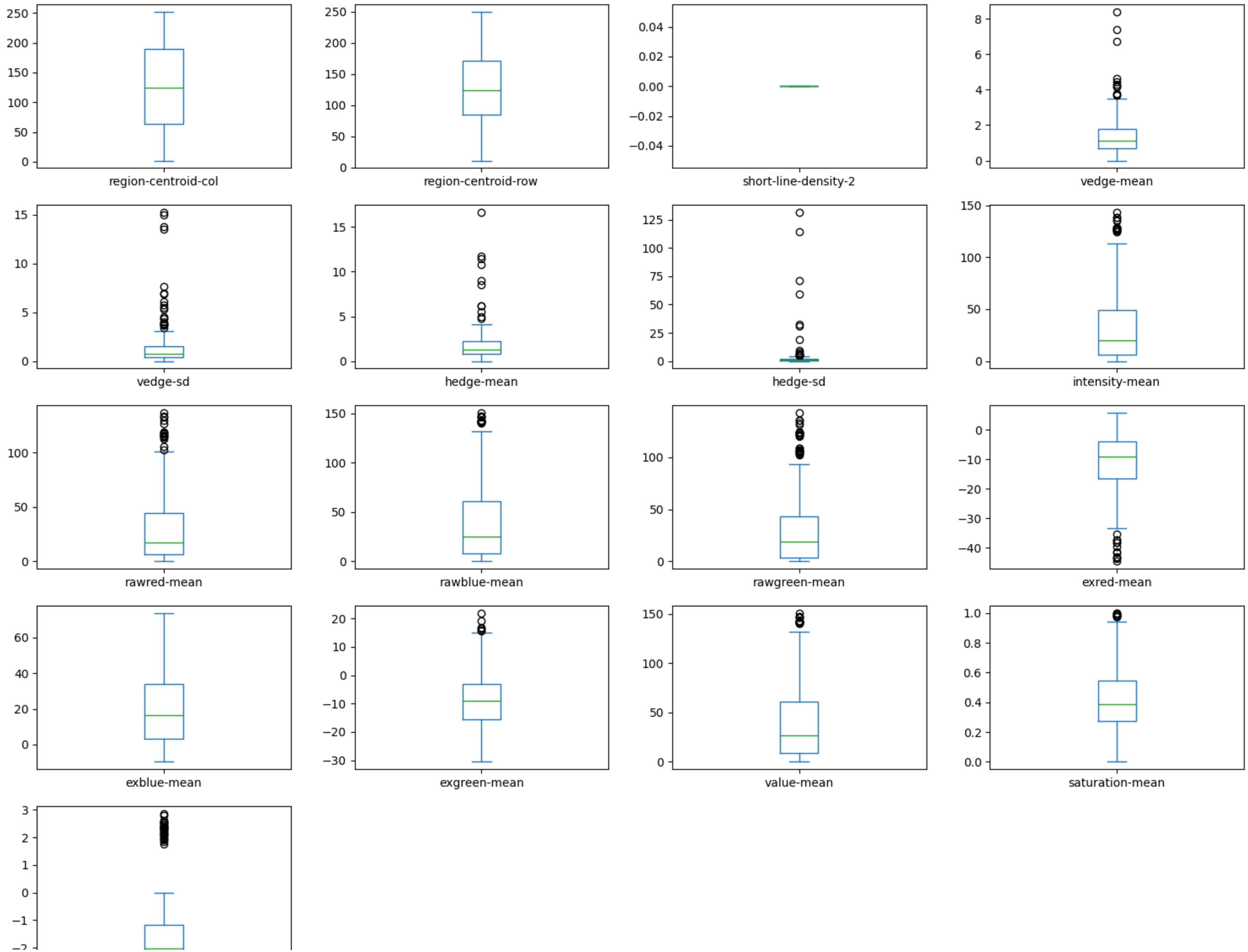
In [199…

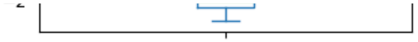
```
# Identificação e remoção de outliers

from scipy.stats import zscore
import matplotlib.pyplot as plt
import numpy as np

# Busca por outliers
img_seg_df_df_no_dups.plot(kind='box', subplots=True, layout=(5, 4), figsize=(15, 12), sharex=False, sharey=False)
plt.tight_layout()
plt.show()

z_scores = img_seg_df_df_no_dups.select_dtypes(include='number').apply(zscore)
outliers = (abs(z_scores) > 3)  # Z-score > 3 considered outlier

print("Total de linhas que contem pelo menos um outlier:", np.sum(np.any(outliers, axis=1)))

img_seg_df_df_no_dups_no_outliers = img_seg_df_df_no_dups[(~outliers).all(axis=1)]
print("Total de padrões com outilers: ", img_seg_df_df_no_dups.shape[0])
print("Total de padrões após remoção de outilers: ", img_seg_df_df_no_dups_no_outliers.shape[0])

img_seg_df_df_no_dups_no_outliers.plot(kind='box', subplots=True, layout=(5, 4), figsize=(15, 12), sharex=False, sharey=False)
plt.tight_layout()
plt.show()
```


region-centroid-col
region-centroid-row
short-line-density-2
vedge-mean
vedge-sd
hedge-mean
hedge-sd
intensity-mean
rawred-mean
rawblue-mean
rawgreen-mean
exred-mean
exblue-mean
exgreen-mean
value-mean
saturation-mean




```
Total de linhas que contem pelo menos um outlier: 15
Total de padrões com outilers:  210
Total de padrões após remoção de outilers:  195
```


region-centroid-col
region-centroid-row
short-line-density-2
vedge-mean
vedge-sd
hedge-mean
hedge-sd
intensity-mean
rawred-mean
rawblue-mean
rawgreen-mean
exred-mean
exblue-mean
exgreen-mean
value-mean
saturation-mean



In [200…

```python
# Verificação dos dados

X = img_seg_df_df_no_dups_no_outliers.drop('class', axis=1)
print(X.head())
Y = img_seg_df_df_no_dups_no_outliers['class']
print(Y.unique())
```

```
   region-centroid-col  region-centroid-row  short-line-density-2  vedge-mean  \
0                140.0                125.0                   0.0    0.277778   
1                188.0                133.0                   0.0    0.333333   
2                105.0                139.0                   0.0    0.277778   
3                 34.0                137.0                   0.0    0.500000   
4                 39.0                111.0                   0.0    0.722222   

   vedge-sd  hedge-mean  hedge-sd  intensity-mean  rawred-mean  rawblue-mean  \
0  0.062963    0.666667  0.311111        6.185185     7.333334      7.666666   
1  0.266667    0.500000  0.077778        6.666666     8.333334      7.777778   
2  0.107407    0.833333  0.522222        6.111111     7.555555      7.222222   
3  0.166667    1.111111  0.474074        5.851852     7.777778      6.444445   
4  0.374074    0.888889  0.429629        6.037037     7.000000      7.666666   

   rawgreen-mean  exred-mean  exblue-mean  exgreen-mean  value-mean  \
0       3.555556    3.444444     4.444445     -7.888889    7.777778   
1       3.888889    5.000000     3.333333     -8.333333    8.444445   
2       3.555556    4.333334     3.333333     -7.666666    7.555555   
3       3.333333    5.777778     1.777778     -7.555555    7.777778   
4       3.444444    2.888889     4.888889     -7.777778    7.888889   

   saturation-mean  hue-mean  
0         0.545635 -1.121818  
1         0.538580 -0.924817  
2         0.532628 -0.965946  
3         0.573633 -0.744272  
4         0.562919 -1.175773  
['BRICKFACE' 'SKY' 'FOLIAGE' 'CEMENT' 'WINDOW' 'PATH' 'GRASS']
```

In [201…

```python
# Feature selection
```

```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LogisticRegression

sfs = SequentialFeatureSelector(LogisticRegression(solver='liblinear', max_iter=1000), n_features_to_select=10, direction='for
sfs.fit(X, Y)

selected_features = X.columns[sfs.get_support()]
X = X[selected_features]
print(selected_features)
```

```
Index(['region-centroid-col', 'region-centroid-row', 'short-line-density-2',
       'vedge-sd', 'intensity-mean', 'rawred-mean', 'rawgreen-mean',
       'exred-mean', 'value-mean', 'saturation-mean'],
      dtype='object')
```

Na próxima seção que deverão ser realizada as tentativas de tratamento de dados, visando a melhoria no desempenho do classificador (SVM).

In [202…

```
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
import pandas as pd

X_orig = img_seg_df.drop('class', axis=1)
Y_orig = img_seg_df['class']
print(X_orig.head())
print(Y_orig.unique() )

# normalização min-max
X = pd.DataFrame( minmax_scale(X) )

print(X_orig.head())
print(X.head())
```

```
   region-centroid-col  region-centroid-row  short-line-density-2  vedge-mean  \
0                140.0                125.0                   0.0    0.277778   
1                188.0                133.0                   0.0    0.333333   
2                105.0                139.0                   0.0    0.277778   
3                 34.0                137.0                   0.0    0.500000   
4                 39.0                111.0                   0.0    0.722222   

   vedge-sd  hedge-mean  hedge-sd  intensity-mean  rawred-mean  rawblue-mean  \
0  0.062963    0.666667  0.311111        6.185185     7.333334      7.666666   
1  0.266667    0.500000  0.077778        6.666666     8.333334      7.777778   
2  0.107407    0.833333  0.522222        6.111111     7.555555      7.222222   
3  0.166667    1.111111  0.474074        5.851852     7.777778      6.444445   
4  0.374074    0.888889  0.429629        6.037037     7.000000      7.666666   

   rawgreen-mean  exred-mean  exblue-mean  exgreen-mean  value-mean  \
0       3.555556    3.444444     4.444445     -7.888889    7.777778   
1       3.888889    5.000000     3.333333     -8.333333    8.444445   
2       3.555556    4.333334     3.333333     -7.666666    7.555555   
3       3.333333    5.777778     1.777778     -7.555555    7.777778   
4       3.444444    2.888889     4.888889     -7.777778    7.888889   

   saturation-mean  hue-mean  
0         0.545635 -1.121818  
1         0.538580 -0.924817  
2         0.532628 -0.965946  
3         0.573633 -0.744272  
4         0.562919 -1.175773  
['BRICKFACE' 'SKY' 'FOLIAGE' 'CEMENT' 'WINDOW' 'PATH' 'GRASS']
   region-centroid-col  region-centroid-row  short-line-density-2  vedge-mean  \
0                140.0                125.0                   0.0    0.277778   
1                188.0                133.0                   0.0    0.333333   
2                105.0                139.0                   0.0    0.277778   
3                 34.0                137.0                   0.0    0.500000   
4                 39.0                111.0                   0.0    0.722222   

   vedge-sd  hedge-mean  hedge-sd  intensity-mean  rawred-mean  rawblue-mean  \
0  0.062963    0.666667  0.311111        6.185185     7.333334      7.666666   
1  0.266667    0.500000  0.077778        6.666666     8.333334      7.777778   
2  0.107407    0.833333  0.522222        6.111111     7.555555      7.222222   
3  0.166667    1.111111  0.474074        5.851852     7.777778      6.444445   
4  0.374074    0.888889  0.429629        6.037037     7.000000      7.666666   
```

```
   rawgreen-mean  exred-mean  exblue-mean  exgreen-mean  value-mean  \
0       3.555556    3.444444     4.444445     -7.888889    7.777778   
1       3.888889    5.000000     3.333333     -8.333333    8.444445   
2       3.555556    4.333334     3.333333     -7.666666    7.555555   
3       3.333333    5.777778     1.777778     -7.555555    7.777778   
4       3.444444    2.888889     4.888889     -7.777778    7.888889   

   saturation-mean  hue-mean  
0         0.545635 -1.121818  
1         0.538580 -0.924817  
2         0.532628 -0.965946  
3         0.573633 -0.744272  
4         0.562919 -1.175773  
       0         1    2         3         4         5         6         7  \
0  0.552  0.476987  0.0  0.004125  0.043119  0.053571  0.024942  0.953744   
1  0.744  0.510460  0.0  0.017471  0.046476  0.060877  0.027280  0.984581   
2  0.412  0.535565  0.0  0.007037  0.042603  0.055195  0.024942  0.971366   
3  0.128  0.527197  0.0  0.010920  0.040795  0.056818  0.023383  1.000000   
4  0.148  0.418410  0.0  0.024509  0.042086  0.051136  0.024162  0.942731   

          8         9  
0  0.051546  0.545635  
1  0.055965  0.538580  
2  0.050074  0.532628  
3  0.051546  0.573633  
4  0.052283  0.562919  
```

A próxima seção trata da construção do modelo, dos testes e das métricas da matriz de confusão.

In [203…

```python
from sklearn.model_selection import train_test_split
import numpy as np

# com os dados originais
X_oring_train, X_orig_test, y_orig_train, y_orig_test = train_test_split(X_orig,
                      Y_orig, test_size=0.25, stratify=Y_orig,random_state=10)

# com os dados tratados
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25,
                                                     stratify=Y,random_state=10)
```

Treina o modelo com base nos dados originais (SVM).

In [204…

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

treinador = svm.SVC()  #algoritmo escolhido

modelo_orig = treinador.fit(X_oring_train, y_orig_train)

# predição com os mesmos dados usados para treinar
y_orig_pred = modelo_orig.predict(X_oring_train)
cm_orig_train = confusion_matrix(y_orig_train, y_orig_pred)
print('Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_orig_train, y_orig_pred, zero_division=0))

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_orig_pred = modelo_orig.predict(X_orig_test)
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
print(cm_orig_test)
print(classification_report(y_orig_test, y2_orig_pred, zero_division=0))
```

```
Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO
[[18  0  0  0  0  0  5]
 [ 3 18  0  0  0  0  1]
 [15  1  2  0  0  0  5]
 [ 0  0  0 22  0  0  0]
 [ 0  0  0  0 22  0  0]
 [ 0  0  0  0  0 22  0]
 [ 6  1  0  0  0  0 16]]
              precision    recall  f1-score   support

   BRICKFACE       0.43      0.78      0.55        23
      CEMENT       0.90      0.82      0.86        22
     FOLIAGE       1.00      0.09      0.16        23
       GRASS       1.00      1.00      1.00        22
        PATH       1.00      1.00      1.00        22
         SKY       1.00      1.00      1.00        22
      WINDOW       0.59      0.70      0.64        23

    accuracy                           0.76       157
   macro avg       0.85      0.77      0.74       157
weighted avg       0.84      0.76      0.74       157

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[5 0 0 0 0 0 2]
 [1 6 0 1 0 0 0]
 [6 1 0 0 0 0 0]
 [0 0 0 8 0 0 0]
 [0 0 0 0 8 0 0]
 [0 0 0 0 0 8 0]
 [2 1 0 0 0 0 4]]
              precision    recall  f1-score   support

   BRICKFACE       0.36      0.71      0.48         7
      CEMENT       0.75      0.75      0.75         8
     FOLIAGE       0.00      0.00      0.00         7
       GRASS       0.89      1.00      0.94         8
        PATH       1.00      1.00      1.00         8
         SKY       1.00      1.00      1.00         8
      WINDOW       0.67      0.57      0.62         7

    accuracy                           0.74        53
```

```
   macro avg       0.67      0.72      0.68        53
weighted avg       0.68      0.74      0.70        53
```

Como os dados ficam após os processos de tratamento dos dados?

In [205…

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

treinador = svm.SVC()  #algoritmo escolhido

modelo = treinador.fit(X_train, y_train)

# predição com os mesmos dados usados para treinar
y_pred = modelo.predict(X_train)
cm_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO')
print(cm_train)
print(classification_report(y_train, y_pred, zero_division=0))

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_pred = modelo.predict(X_test)
cm_test = confusion_matrix(y_test, y2_pred)
print(cm_test)
print(classification_report(y_test, y2_pred, zero_division=0))
```

```
Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO
[[21  0  0  0  0  0  2]
 [ 1 17  0  2  0  0  0]
 [ 2  0 17  0  0  0  0]
 [ 0  0  0 22  0  0  0]
 [ 0  0  0  1 17  0  0]
 [ 0  0  0  0  0 22  0]
 [ 3  2  3  0  0  0 14]]
              precision    recall  f1-score   support

   BRICKFACE       0.78      0.91      0.84        23
      CEMENT       0.89      0.85      0.87        20
     FOLIAGE       0.85      0.89      0.87        19
       GRASS       0.88      1.00      0.94        22
        PATH       1.00      0.94      0.97        18
         SKY       1.00      1.00      1.00        22
      WINDOW       0.88      0.64      0.74        22

    accuracy                           0.89       146
   macro avg       0.90      0.89      0.89       146
weighted avg       0.89      0.89      0.89       146

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[7 0 0 0 0 0 0]
 [0 5 0 0 0 0 2]
 [1 0 5 0 0 0 0]
 [0 0 0 8 0 0 0]
 [0 0 0 0 6 0 0]
 [0 0 0 0 0 7 0]
 [1 0 0 0 0 0 7]]
              precision    recall  f1-score   support

   BRICKFACE       0.78      1.00      0.88         7
      CEMENT       1.00      0.71      0.83         7
     FOLIAGE       1.00      0.83      0.91         6
       GRASS       1.00      1.00      1.00         8
        PATH       1.00      1.00      1.00         6
         SKY       1.00      1.00      1.00         7
      WINDOW       0.78      0.88      0.82         8

    accuracy                           0.92        49
```

```
   macro avg       0.94      0.92      0.92        49
weighted avg       0.93      0.92      0.92        49
```

In [ ]:

```

```