Associação Instituto de Inteligência Artificial Aplicada - I2A2

As regras do jogo:

A planilha contém 17 sensores e 8 equipamentos. Um dos equipamentos está com defeito. Descobrir qual é o equipamento defeituoso. Importante, o equipamento, e não o sensor que está alarmando o defeito.

Carregamento de bibliotecas e conjunto de dados

In [1]:

```
# Carregamento de bibliotecas
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Utilizar códigos abaixo somente se necessário para ignorar avisos 'warnings'
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
# Carrega os dados de um arquivo excel ra o conjunto de dados: df
df = pd.read_excel(os.getcwd() + '\\Defective_Equipment(revised).xlsx')
```

Data Wrangling

In [3]:

```
# Visualizar os dados carregados
df
```

Out[3]:

	Seq	V1	V2	V3	V4	V5	V6	V7	V8
0	1	375	135	458	475	509	336	469	492
1	2	57	47	53	73	63	62	63	58
2	3	245	267	242	227	271	219	268	286
3	4	1472	1494	1462	1582	1613	1323	1490	1493
4	5	105	66	103	103	118	98	101	118
5	6	54	41	62	64	55	59	63	59
6	7	193	209	184	235	207	172	223	156
7	8	147	93	122	160	139	130	152	101
8	9	1102	674	957	1137	1058	990	1098	878
9	10	720	1033	566	874	628	646	706	320
10	11	253	143	171	265	193	226	247	99
11	12	685	586	750	803	830	615	699	777
12	13	488	355	418	570	465	437	467	313
13	14	198	187	220	203	247	176	209	204
14	15	360	334	337	365	376	322	363	348
15	16	1374	1506	1572	1256	1734	1235	1597	1684
16	17	156	139	147	175	167	138	164	170

In [4]:

```
# Conferir o corpo do conunto de dados carregados
df.shape
```

Out[4]:

```
(17, 9)
```

In [5]:

```
# Analise de valores estatísticos padrão
df.describe()
```

Out[5]:

	Seq	V1	V2	V3	V4	V5	V6	V7	V8
count	17.000000	17.000000	17.000000	17.000000	17.000000	17.000000	17.000000	17.000000	17.000000
mean	9.000000	469.647059	429.941176	460.235294	503.941176	510.176471	422.588235	492.882353	444.470588
std	5.049752	452.701466	479.896014	469.595907	463.859282	517.552562	405.843267	480.721448	491.302748
min	1.000000	54.000000	41.000000	53.000000	64.000000	55.000000	59.000000	63.000000	58.000000
25%	5.000000	156.000000	135.000000	147.000000	175.000000	167.000000	138.000000	164.000000	118.000000
50%	9.000000	253.000000	209.000000	242.000000	265.000000	271.000000	226.000000	268.000000	286.000000
75%	13.000000	685.000000	586.000000	566.000000	803.000000	628.000000	615.000000	699.000000	492.000000
max	17.000000	1472.000000	1506.000000	1572.000000	1582.000000	1734.000000	1323.000000	1597.000000	1684.000000

In [6]:

```
# Verificação quantos valores nulos esistem no conjunto de dados
df.isnull().sum()
```

Out[6]:

```
Seq    0
V1     0
V2     0
V3     0
V4     0
V5     0
V6     0
V7     0
V8     0
dtype: int64
```

In [7]:

```
df_todos = df.drop(['Seq'], axis=1)  # Drop a coluna 'Seq' retirar elementos potualmente desnecessários
df_todos.shape  # Conferir o corpo do conunto de dados carregados
```

Out[7]:

```
(17, 8)
```

In [8]:

```
# Gerar tabela de correlações
df_todos.corr()
```

Out[8]:

	V1	V2	V3	V4	V5	V6	V7	V8
V1	1.000000	0.947619	0.984917	0.990654	0.985102	0.999954	0.994283	0.950215
V2	0.947619	1.000000	0.940742	0.937061	0.940745	0.947792	0.948130	0.900400
V3	0.984917	0.940742	1.000000	0.964122	0.999965	0.985104	0.995020	0.987685
V4	0.990654	0.937061	0.964122	1.000000	0.964438	0.990426	0.975923	0.917521
V5	0.985102	0.940745	0.999965	0.964438	1.000000	0.985238	0.995116	0.987497
V6	0.999954	0.947792	0.985104	0.990426	0.985238	1.000000	0.994309	0.950647
V7	0.994283	0.948130	0.995020	0.975923	0.995116	0.994309	1.000000	0.970414
V8	0.950215	0.900400	0.987685	0.917521	0.987497	0.950647	0.970414	1.000000

In [9]:

```
# Analise de valores estatísticos padrão
df_todos.corr().describe()
```

Out[9]:

	V1	V2	V3	V4	V5	V6	V7	V8
count	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000
mean	0.981593	0.945311	0.982194	0.967518	0.982263	0.981684	0.984149	0.958047
std	0.020980	0.027090	0.020313	0.028370	0.020280	0.020844	0.017926	0.035336
min	0.947619	0.900400	0.940742	0.917521	0.940745	0.947792	0.948130	0.900400
25%	0.976242	0.939822	0.979718	0.957356	0.979936	0.976490	0.974546	0.942042
50%	0.987878	0.944182	0.986395	0.970181	0.986368	0.987832	0.994296	0.960531
75%	0.995700	0.947876	0.996256	0.990483	0.996328	0.995720	0.995044	0.987544
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [10]:

```
# Analise de valores estatísticos padrão: média
df_todos.corr().mean()
```

Out[10]:

```
V1    0.981593
V2    0.945311
V3    0.982194
V4    0.967518
V5    0.982263
V6    0.981684
V7    0.984149
V8    0.958047
dtype: float64
```

In [11]:

```
# Analise de valores estatísticos padrão: Menor valor médio
print(f'O menor valor médio entre as máquinas é: {df_todos.corr().mean().min()*100:.2f} %')
```

```
O menor valor médio entre as máquinas é: 94.53 %
```

In [12]:

```
# Analise de valores estatísticos padrão: Maior valor médio
print(f'O maior valor médio entre as máquinas é: {df_todos.corr().mean().max()*100:.2f} %')
```

```
O maior valor médio entre as máquinas é: 98.41 %
```

Impressões do Data Wrangling

A maquina V2 apresenta um valor médio baixo com variação média aproximadamente 5,5 %, o que torna um ponto a ser verificado no analise posterior.

As caracteristica do conjunto de dados demonstram uma maquina térmica com sensores de temperatura diversos. Exemplo: Alto forno, forno de tratamento térmico ou coluna de separação.

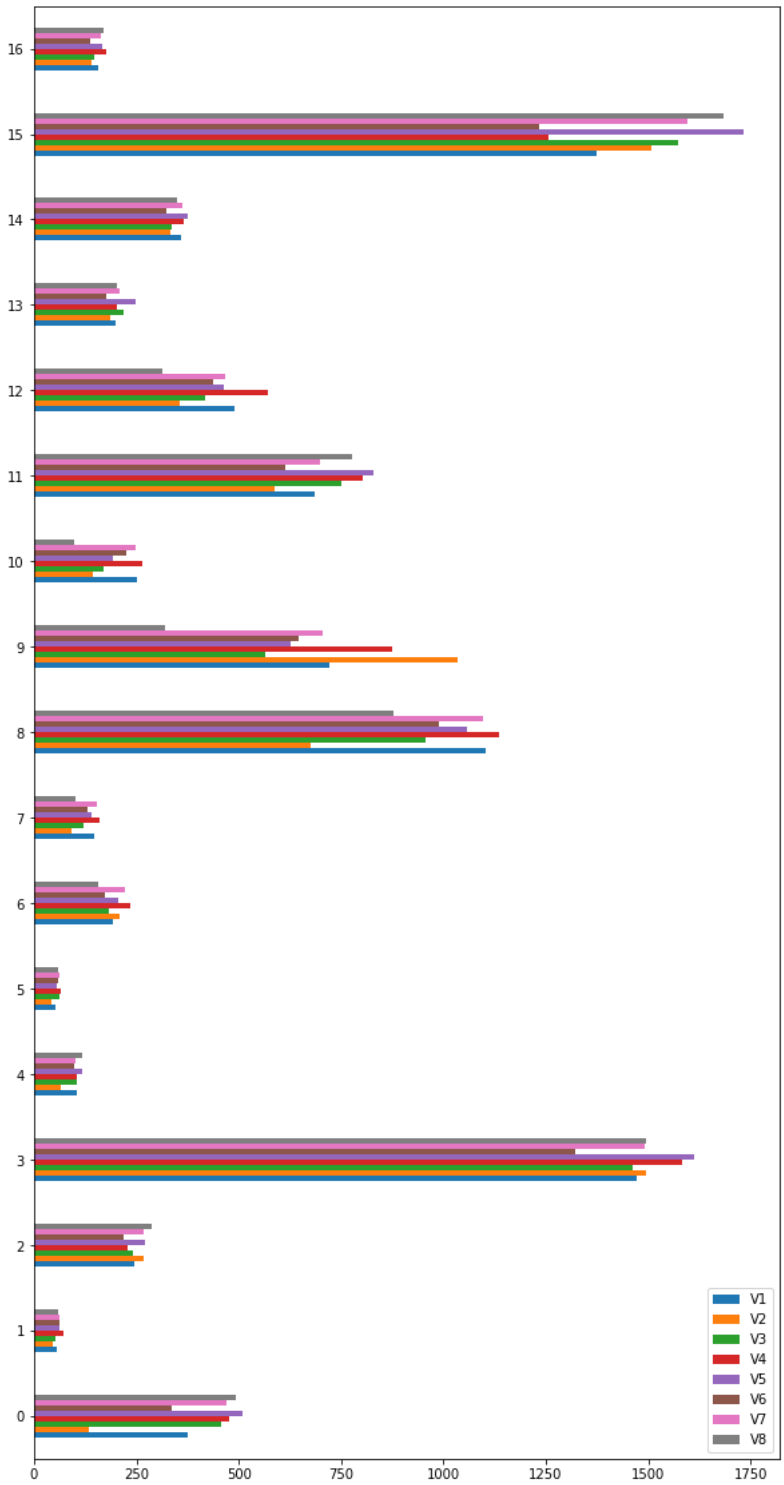
Análise Visual (Gráfica)

In [13]:

```
# Sensores e equipamentos gráfico de barras
df_todos.plot(kind="barh", fontsize=10, figsize=(10,20))
```

Out[13]:

<AxesSubplot: >

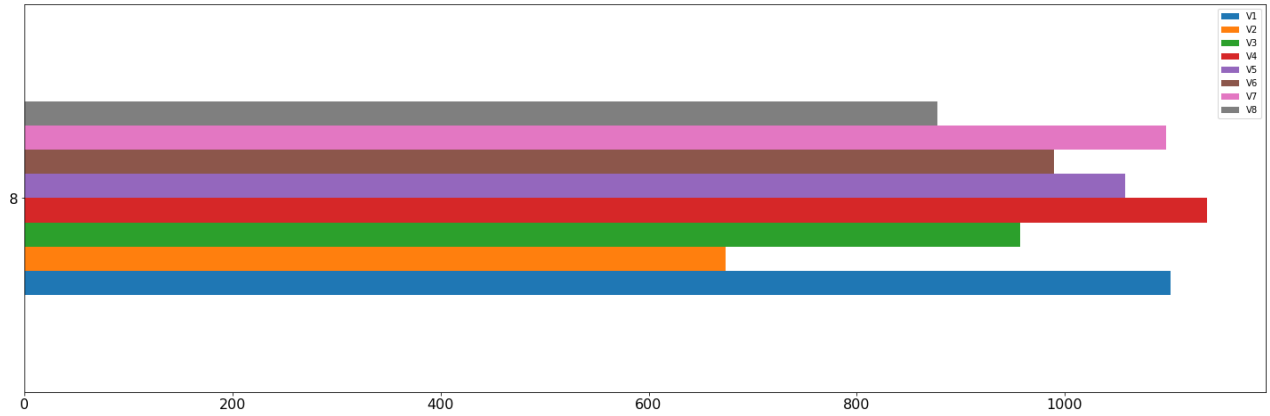


In [14]:

```
# Análise do maior ponto de discrepância
df_todos.iloc[[8]].plot(kind="barh", fontsize=16, figsize=(25,8))
```

Out[14]:

```
<AxesSubplot: >
```



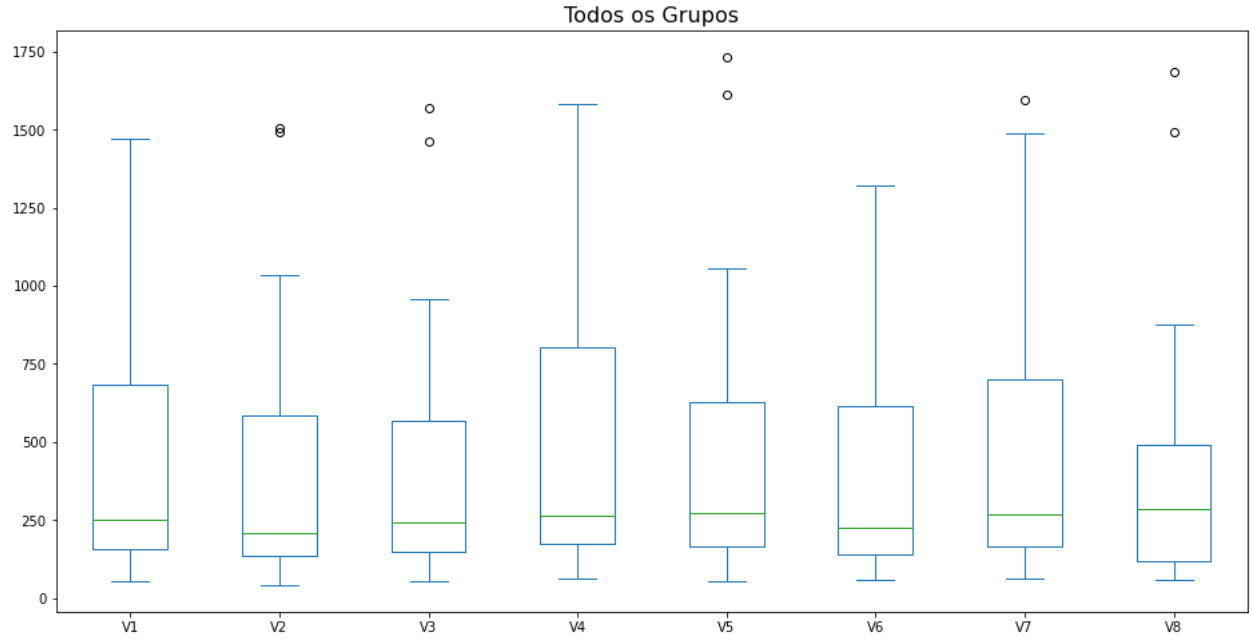
Observe: O gráfico acima monstra que a maquina 'V2' está com menor valor de medição em relação aos outros.

In [15]:

```
# Verificação gráfico de caixa 'Box' por quartis e pontos fora de padrão(outliers)
df_todos.plot(kind = 'box', figsize=(16,8)).set_title('Todos os Grupos', fontsize=16)
```

Out[15]:

```
Text(0.5, 1.0, 'Todos os Grupos')
```



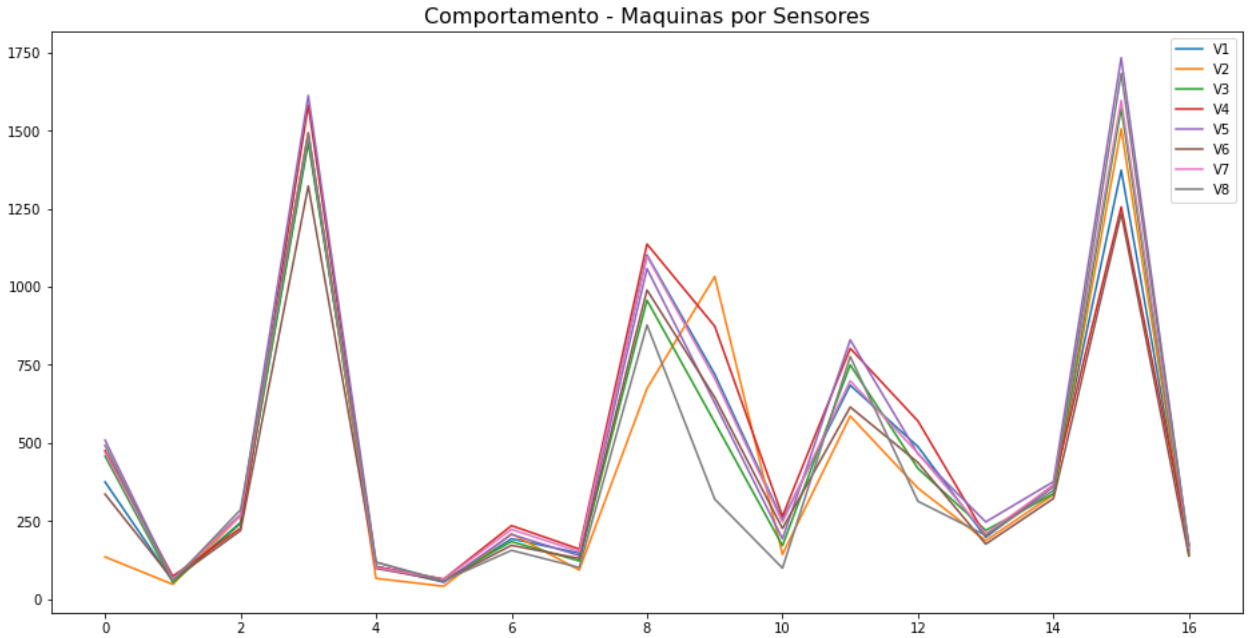
Observe a maquina V2 os Outliers estão sobre postos o que denota anomalia onde a calda superior nos maiores valores.

In [16]:

```
# Amostragem gráfica de linhas do comportamento de máquinas por sensores
df_todos.plot.line(figsize=(16,8)).set_title('Comportamento - Maquinas por Sensores', fontsize=16)
```

Out[16]:

```
Text(0.5, 1.0, 'Comportamento - Maquinas por Sensores')
```



Impressão da análise visual

Novamente a variação do sensor na maquina V2, no sensor 'Seq' = 9 no grafico de linha, e a amostragem no gráfico de caixa 'Box' que os outliers estão sobre postos o que montra falha na medição em fim de escala (sobre a calda superior).

1 - Observavél no gráfico 'Box' que existem 5 maquinas com outliers e 3 sem outliers.

2 - No gráfico de linha, entre o os sensores 11 e 13, existem variações como nos sensores 15 que devem ser observados.

Divisão dos conjuntos de dados em duas cateorias: Com Outliers e sem Outliers

In [17]:

```
# Criação de conjunto de dados sem pontos fora da amostragem (sem outliers)
#without_outliers_group = df.drop(['V2', 'V3', 'V5', 'V7', 'V8'], axis=1, inplace=True)

df_V2 = df_todos.drop(['V2'], axis=1)  # Remove a maquina V2
df_V3 = df_V2.drop(['V3'], axis=1)     # Remove a maquina V3
df_V5 = df_V3.drop(['V5'], axis=1)     # Remove a maquina V5
df_V7 = df_V5.drop(['V7'], axis=1)     # Remove a maquina V7
without_outliers_group = df_V7.drop(['V8'], axis=1)  # Remove a maquina V8
```

In [18]:

```
# Conjunto de dados sem pontos fora da curva (sem outliers)
without_outliers_group
```

Out[18]:

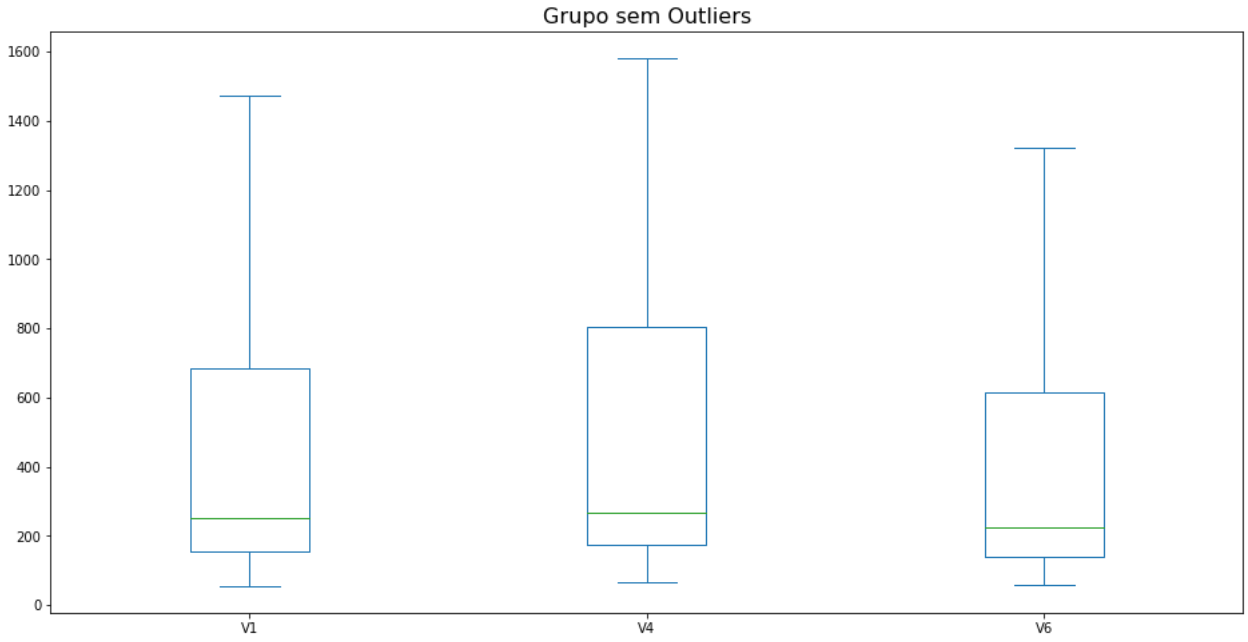
	V1	V4	V6
0	375	475	336
1	57	73	62
2	245	227	219
3	1472	1582	1323
4	105	103	98
5	54	64	59
6	193	235	172
7	147	160	130
8	1102	1137	990
9	720	874	646
10	253	265	226
11	685	803	615
12	488	570	437
13	198	203	176
14	360	365	322
15	1374	1256	1235
16	156	175	138

In [19]:

```
# Conjunto de dados sem pontos fora da amostragem (sem outliers) - BOX PLOT
without_outliers_group.plot(kind = 'box',figsize=(16,8)).set_title('Grupo sem Outliers', fontsize=16)
```

Out[19]:

```
Text(0.5, 1.0, 'Grupo sem Outliers')
```

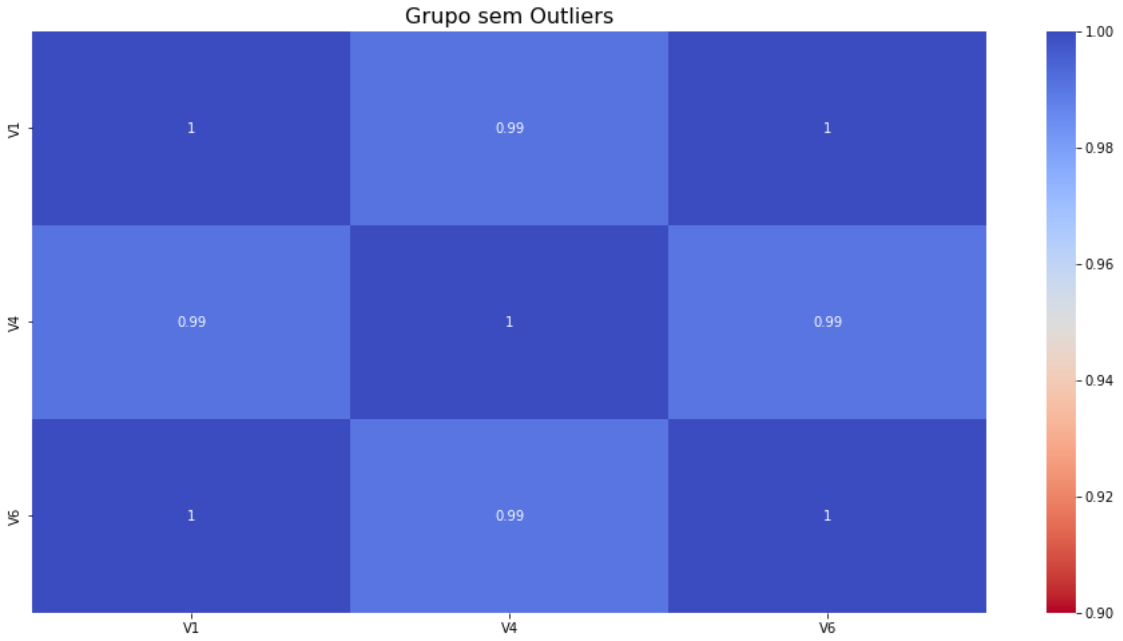


In [20]:

```
# Mapa de calor do conjuto de dados sem Outliers
plt.figure(figsize=(16,8))
sns.heatmap(without_outliers_group.corr(),annot=True, vmin=.9, vmax=1, cmap="coolwarm_r").set_title('Grupo sem Outliers', fontsize=16)
```

Out[20]:

```
Text(0.5, 1.0, 'Grupo sem Outliers')
```



In [21]:

```
# Criação de conjunto de dados com pontos fora da amostragem (com outliers)
df_V1 = df_todos.drop(['V1'], axis=1)  # Remove a maquina V1
df_V4 = df_V1.drop(['V4'], axis=1)     # Remove a maquina V4
only_outliers_group = df_V4.drop(['V6'], axis=1)  # Remove a maquina V6
```

In [22]:

```
# Visualizar os dados carregados Outliers
only_outliers_group
```

Out[22]:

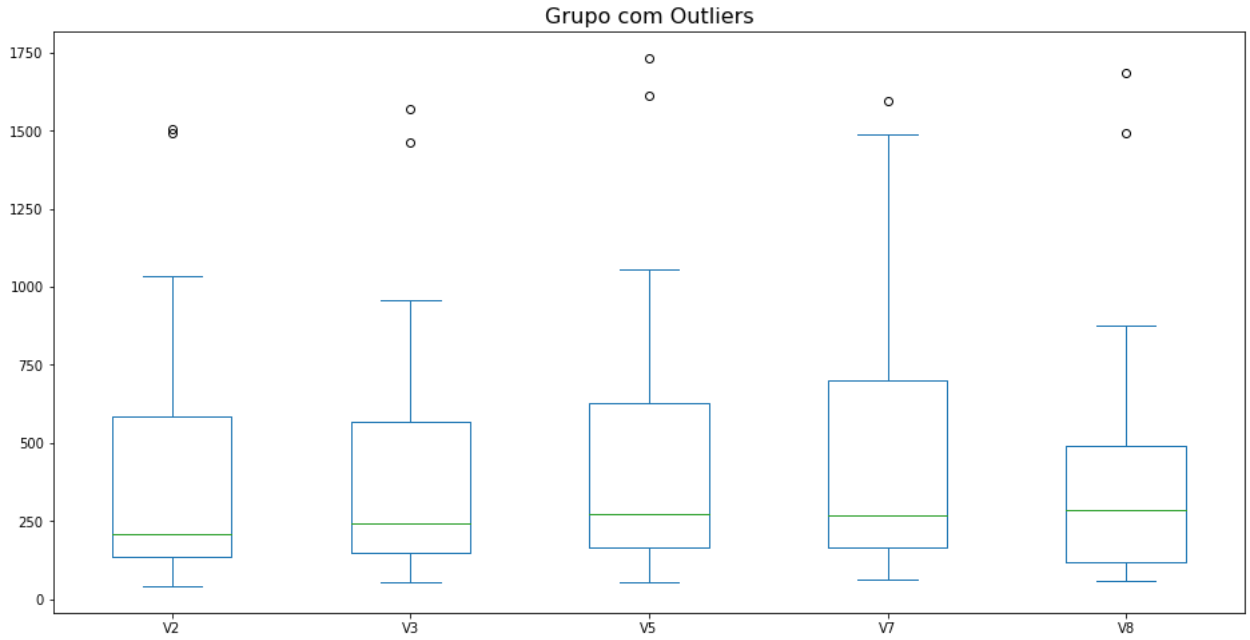
	V2	V3	V5	V7	V8
0	135	458	509	469	492
1	47	53	63	63	58
2	267	242	271	268	286
3	1494	1462	1613	1490	1493
4	66	103	118	101	118
5	41	62	55	63	59
6	209	184	207	223	156
7	93	122	139	152	101
8	674	957	1058	1098	878
9	1033	566	628	706	320
10	143	171	193	247	99
11	586	750	830	699	777
12	355	418	465	467	313
13	187	220	247	209	204
14	334	337	376	363	348
15	1506	1572	1734	1597	1684
16	139	147	167	164	170

In [23]:

```
# Visualizar os dados carregados Outliers = BOX PLOT
only_outliers_group.plot(kind = 'box', figsize=(16,8)).set_title('Grupo com Outliers', fontsize=16)
```

Out[23]:

```
Text(0.5, 1.0, 'Grupo com Outliers')
```

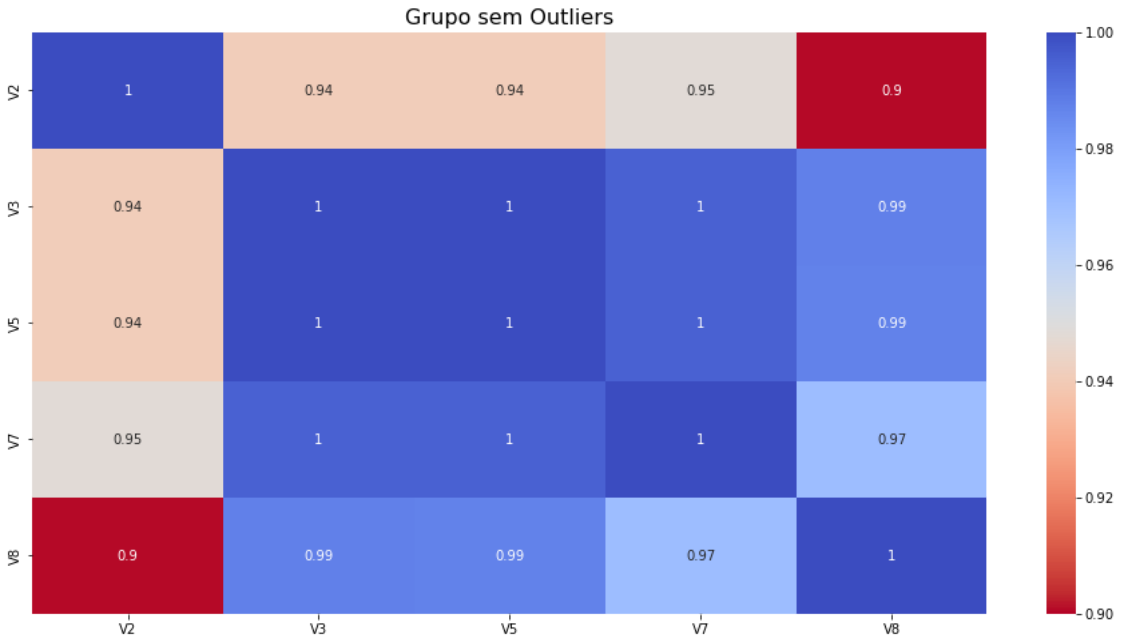


In [24]:

```
# Heatmap do grupo com Outliers
plt.figure(figsize=(16,8))
sns.heatmap(only_outliers_group.corr(),annot=True, vmin=.9, vmax=1, cmap="coolwarm_r").set_title('Grupo sem Outliers', fontsize=16)
```

Out[24]:

```
Text(0.5, 1.0, 'Grupo sem Outliers')
```



In [25]:

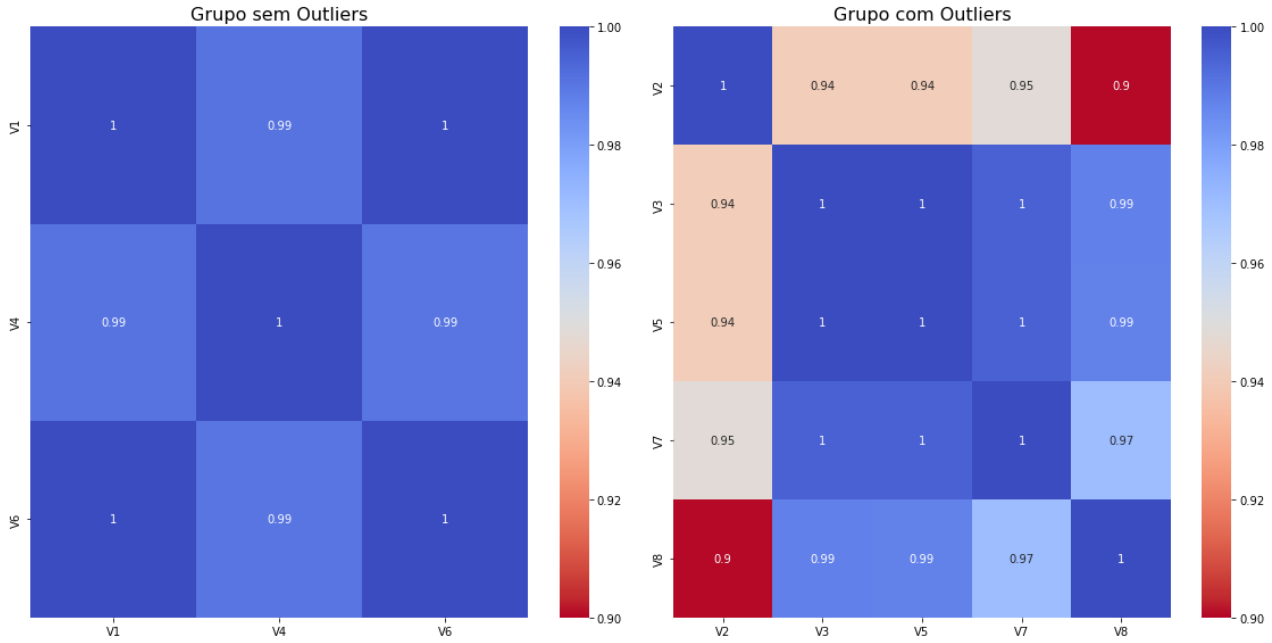
```
# Heatmap dos grupos sem e com Outliers

f, ax = plt.subplots(nrows=1, ncols=2, figsize=(16, 8))

sns.heatmap(without_outliers_group.corr(),annot=True, vmin=.9, vmax=1, cmap="coolwarm_r", ax=ax[0])
ax[0].set_title('Grupo sem Outliers', fontsize=16)

sns.heatmap(only_outliers_group.corr(),annot=True, vmin=.9, vmax=1, cmap="coolwarm_r", ax=ax[1])
ax[1].set_title('Grupo com Outliers', fontsize=16)

plt.tight_layout()
plt.show()
```



In [26]:

```
# Heatmap todo o conjunto de dados
plt.figure(figsize=(16,8))
sns.heatmap(df_todos.corr(),annot=True, vmin=.9, vmax=1, cmap="coolwarm_r").set_title('Todos os Grupos', fontsize=16)
```

Out[26]:

```
Text(0.5, 1.0, 'Todos os Grupos')
```

Utilizando o mapa de calor é possível verificar os menores valore para o range de comparação na máquina 'V2'

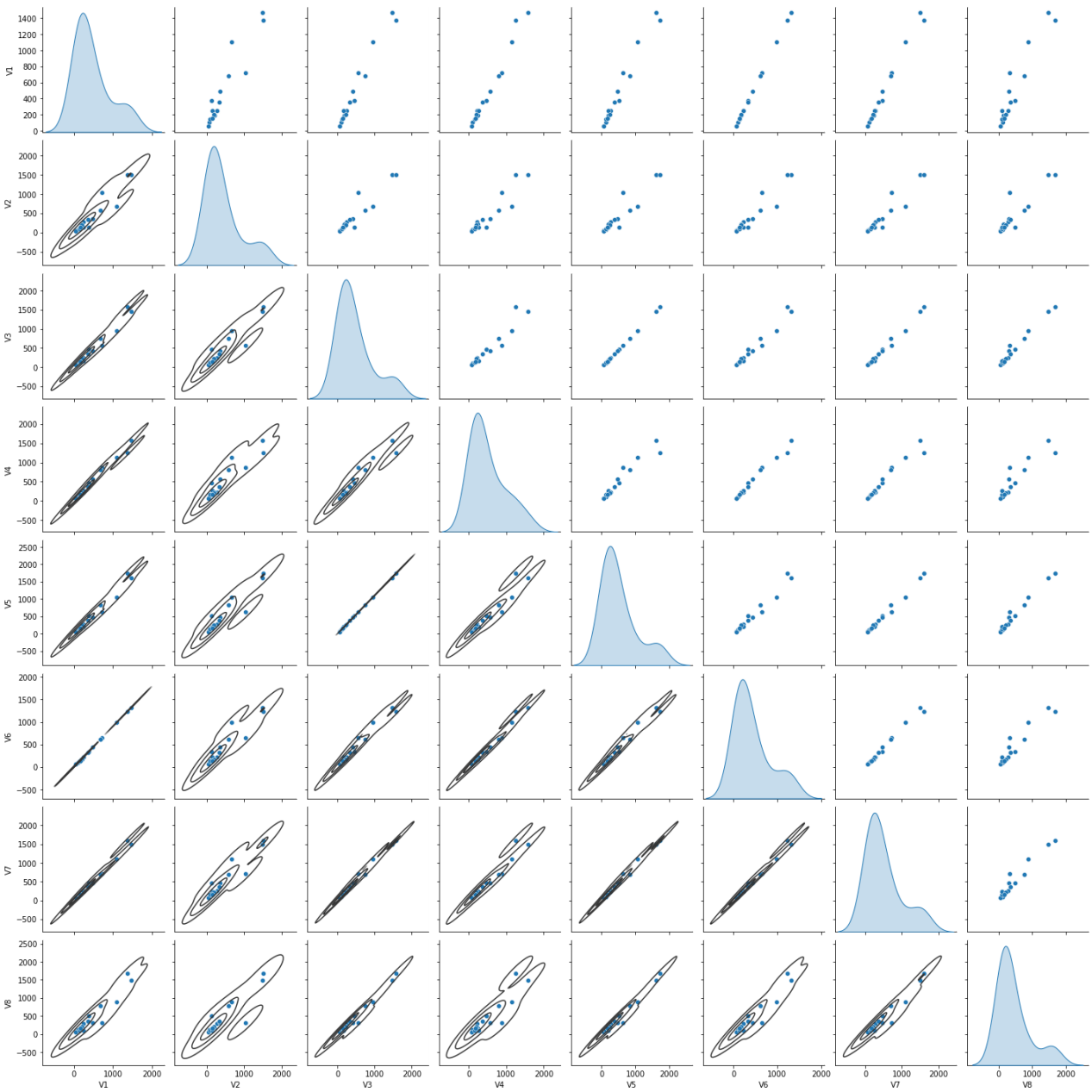
Matriz em pares de relacionamento

In [27]:

```
# Par de relacionamentos
g = sns.pairplot(df_todos, diag_kind="kde")
g.map_lower(sns.kdeplot, levels=4, color=".2")
```

Out[27]:

```
<seaborn.axisgrid.PairGrid at 0x1da6eed1a30>
```

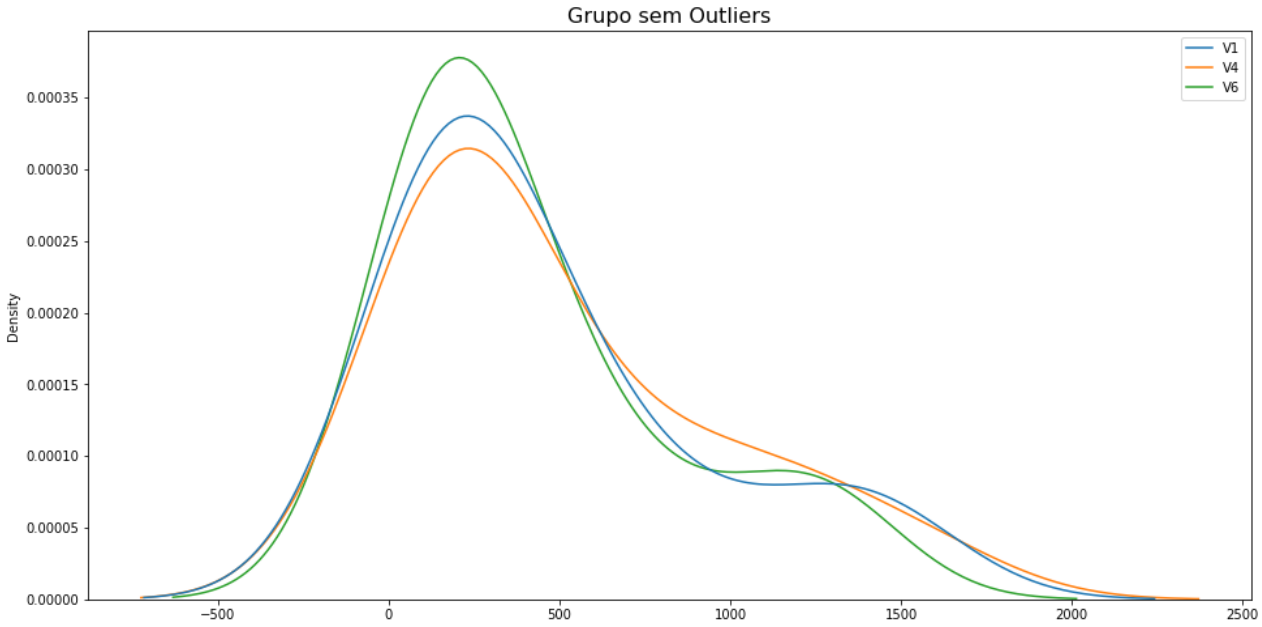


In [28]:

```
# Grupo sem Outliers
plt.figure(figsize=(16,8))
sns.kdeplot(without_outliers_group).set_title('Grupo sem Outliers', fontsize=16)
```

Out[28]:

```
Text(0.5, 1.0, 'Grupo sem Outliers')
```

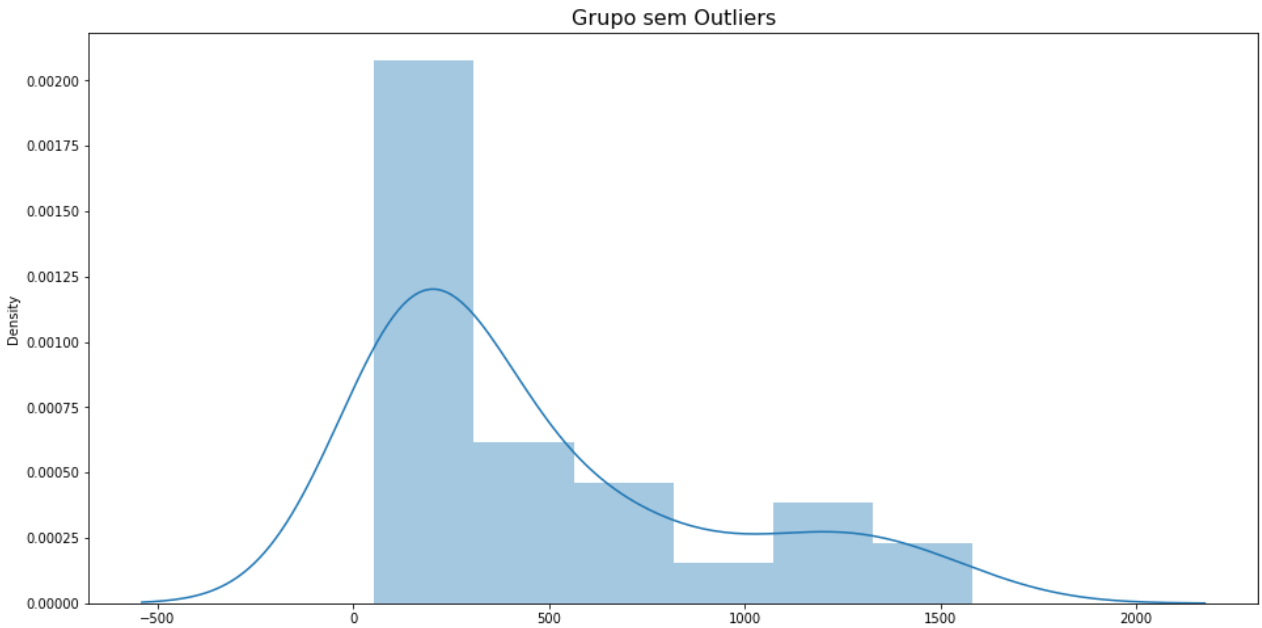


In [29]:

```
# Histograma sem Outliers
plt.figure(figsize=(16,8))
sns.distplot(without_outliers_group).set_title('Grupo sem Outliers', fontsize=16)
```

Out[29]:

```
Text(0.5, 1.0, 'Grupo sem Outliers')
```

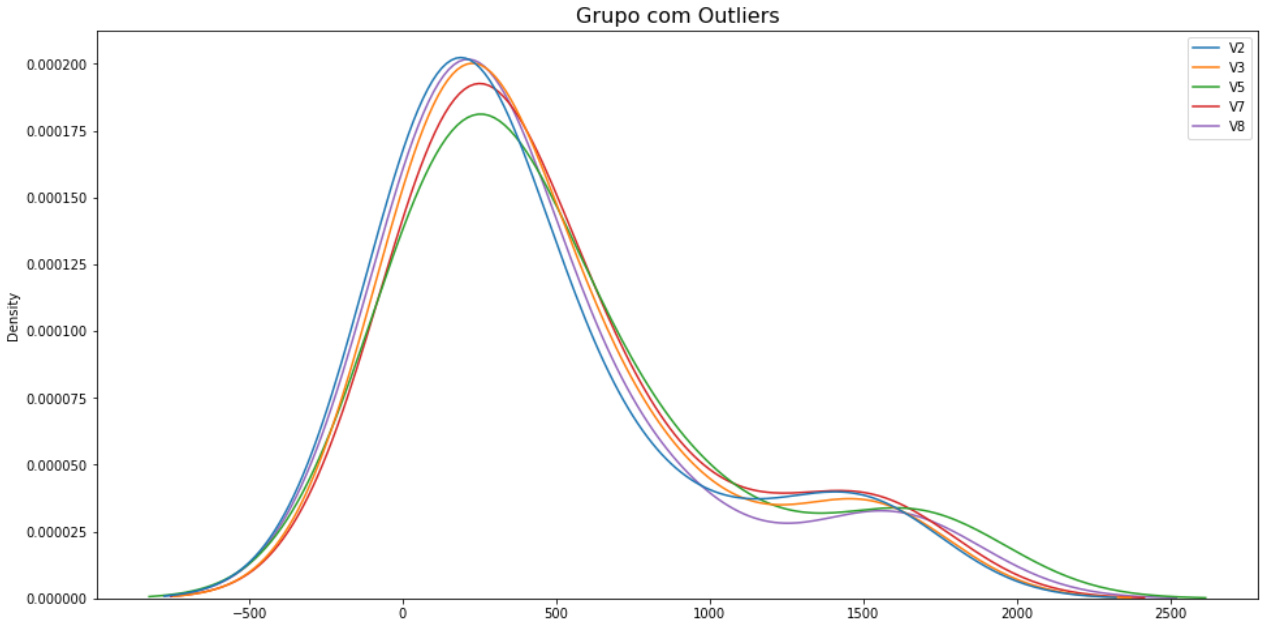


In [30]:

```
# Grupo com Outliers
plt.figure(figsize=(16,8))
sns.kdeplot(only_outliers_group, color='green').set_title('Grupo com Outliers', fontsize=16)
```

Out[30]:

```
Text(0.5, 1.0, 'Grupo com Outliers')
```

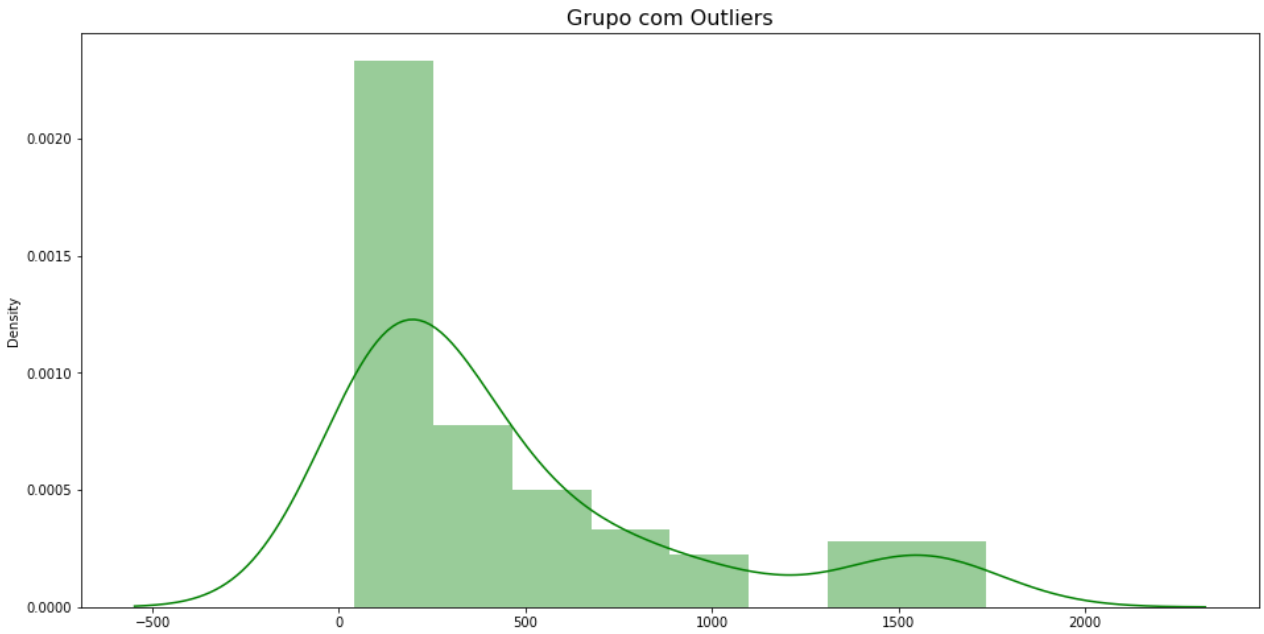


In [31]:

```
# Histograma com Outliers
plt.figure(figsize=(16,8))
sns.distplot(only_outliers_group, color='green').set_title('Grupo com Outliers', fontsize=16)
```

Out[31]:

```
Text(0.5, 1.0, 'Grupo com Outliers')
```

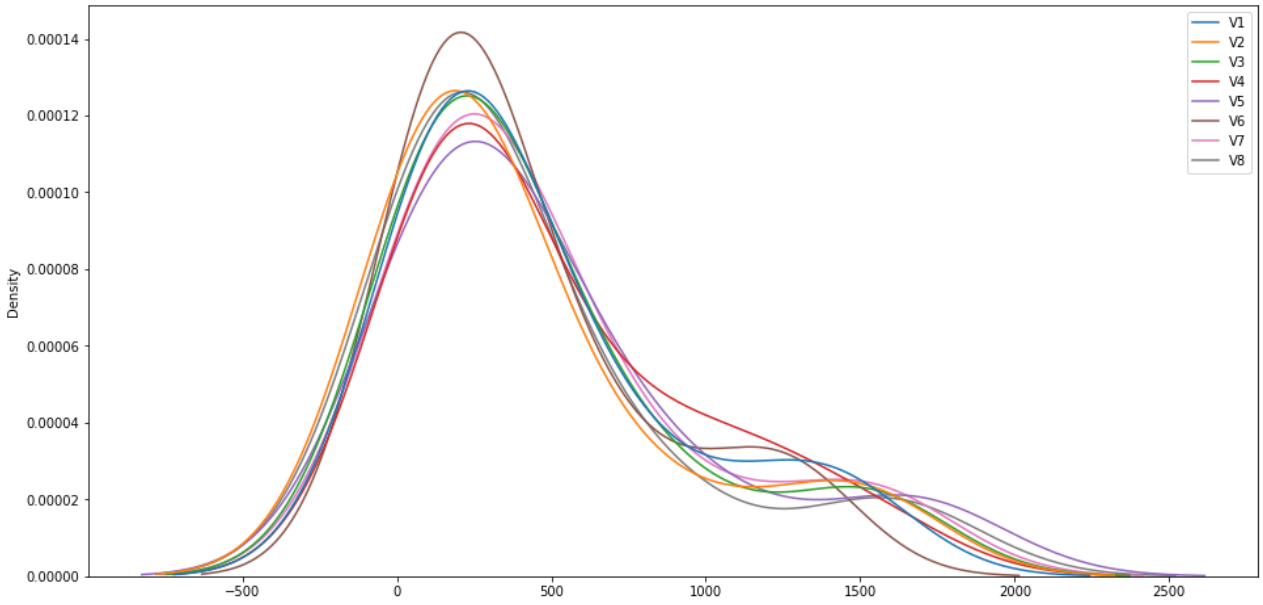


In [32]:

```
# Histograma conjunto de dados completo
plt.figure(figsize=(16,8))
sns.kdeplot(df_todos, color='black')
```

Out[32]:

```
<AxesSubplot: ylabel='Density'>
```

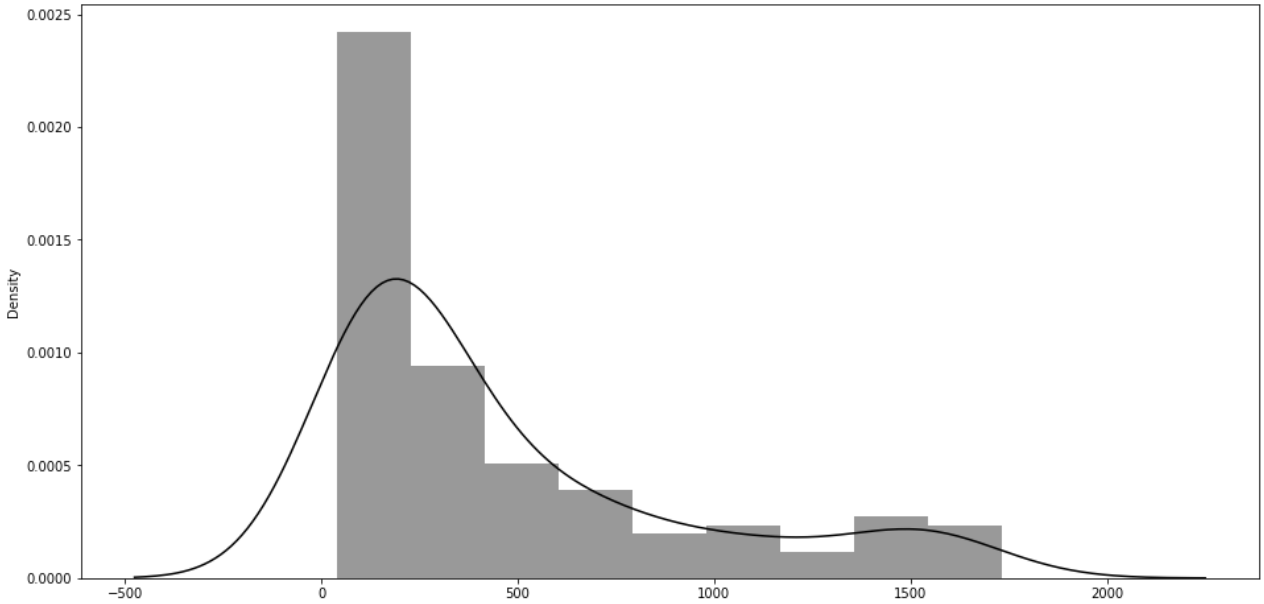


In [33]:

```
# Histograma conjunto de dados completo
plt.figure(figsize=(16,8))
sns.distplot(df_todos, color='black')
```

Out[33]:

```
<AxesSubplot: ylabel='Density'>
```

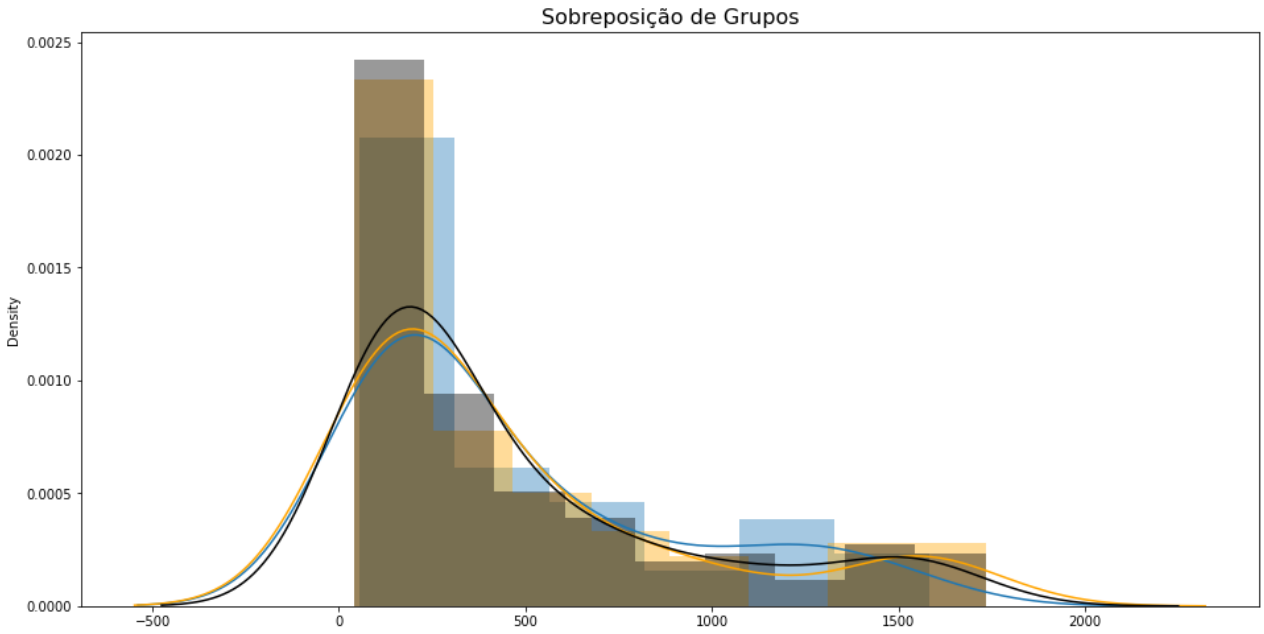


In [34]:

```
# Histograma - Sobre posição
plt.figure(figsize=(16,8))
sns.distplot(without_outliers_group)
sns.distplot(only_outliers_group, color='orange').set_title('Sobreposição de Grupos', fontsize=16)
sns.distplot(df_todos, color='black')
```

Out[34]:

```
<AxesSubplot: title={'center': 'Sobreposição de Grupos'}, ylabel='Density'>
```

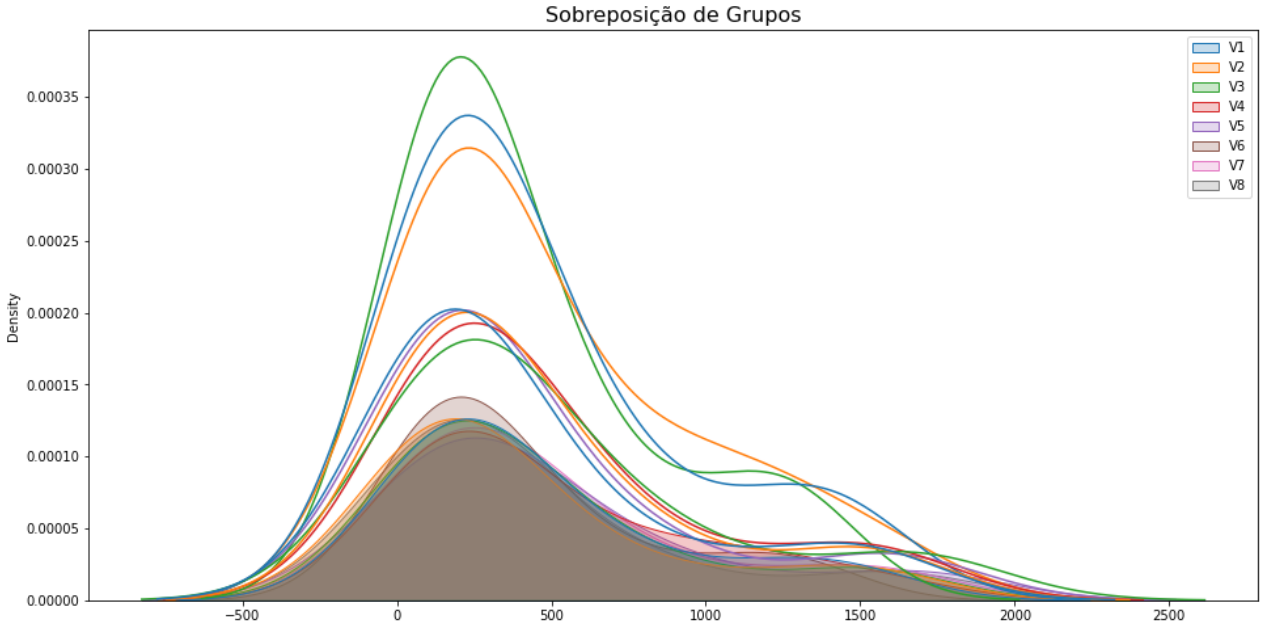


In [35]:

```
# Histograma - Sobre posição
plt.figure(figsize=(16,8))
sns.kdeplot(without_outliers_group)
sns.kdeplot(only_outliers_group).set_title('Sobreposição de Grupos', fontsize=16)
sns.kdeplot(df_todos, shade=True)
```

Out[35]:

```
<AxesSubplot: title={'center': 'Sobreposição de Grupos'}, ylabel='Density'>
```

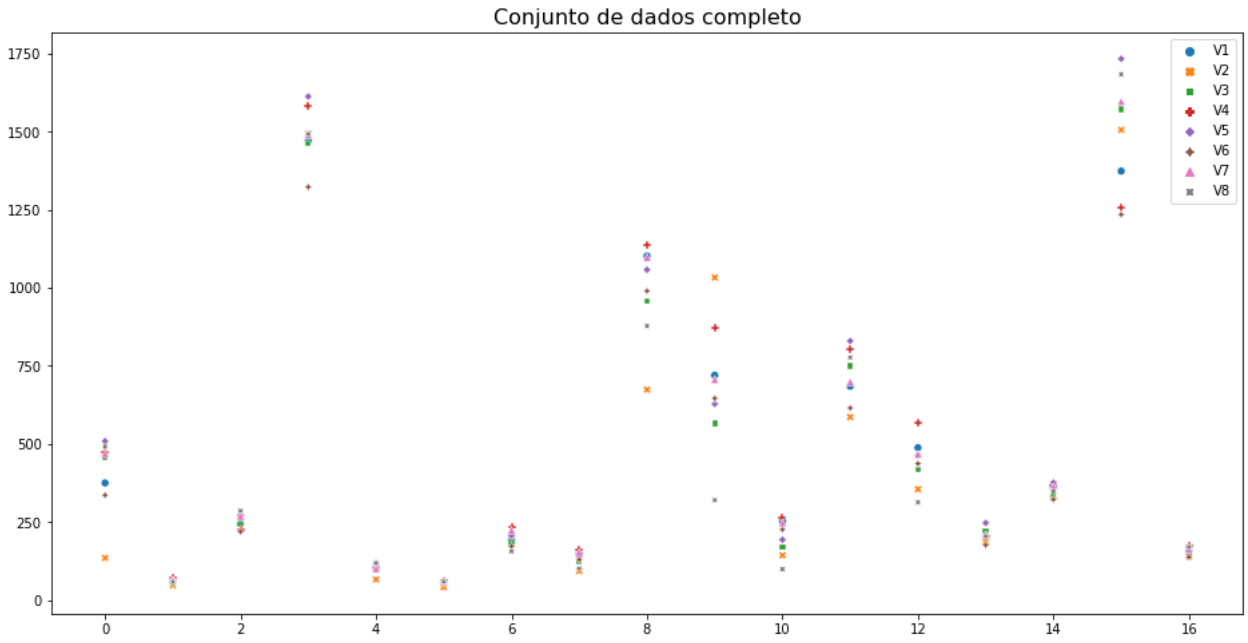


In [36]:

```
# Histograma - Sobre posição
plt.figure(figsize=(16,8))
# sns.set_style(style='whitegrid')
sns.scatterplot(df_todos).set_title('Conjunto de dados completo', fontsize=16)
```

Out[36]:

```
Text(0.5, 1.0, 'Conjunto de dados completo')
```



In [37]:

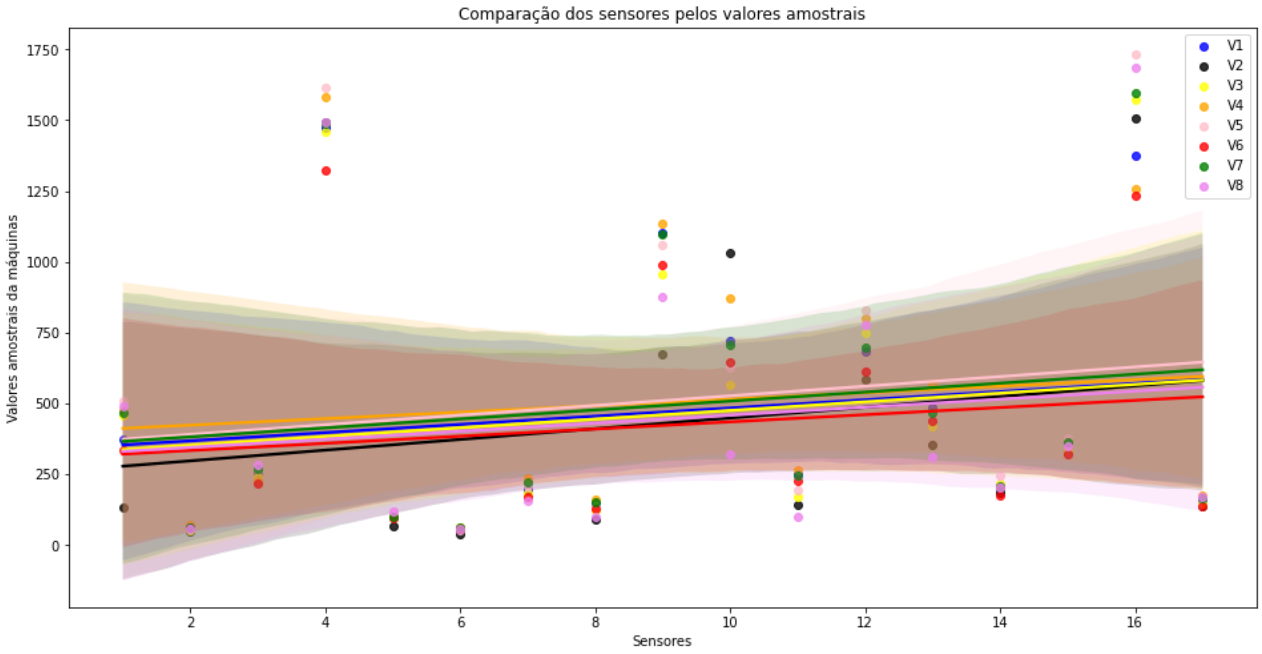
```
# PCA - Transformação
plt.figure(figsize=(16,8))

sns.regplot(data=df, x='Seq', y='V1', color='blue',   label='V1')
sns.regplot(data=df, x='Seq', y='V2', color='black',  label='V2')
sns.regplot(data=df, x='Seq', y='V3', color='yellow', label='V3')
sns.regplot(data=df, x='Seq', y='V4', color='orange', label='V4')
sns.regplot(data=df, x='Seq', y='V5', color='pink',   label='V5')
sns.regplot(data=df, x='Seq', y='V6', color='red',    label='V6')
sns.regplot(data=df, x='Seq', y='V7', color='green',  label='V7')
sns.regplot(data=df, x='Seq', y='V8', color='violet', label='V8')

plt.gca().set_xlabel('Sensores')
plt.gca().set_ylabel('Valores amostrais da máquinas')
plt.title('Comparação dos sensores pelos valores amostrais')
plt.legend()
```

Out[37]:

```
<matplotlib.legend.Legend at 0x1da76ef14c0>
```



Data Wrangling - Criar valor de referência.

In [38]:

```
# Imprimir conjunto de dados de referência de máquinas e sensores
df_todos
```

Out[38]:

	V1	V2	V3	V4	V5	V6	V7	V8
0	375	135	458	475	509	336	469	492
1	57	47	53	73	63	62	63	58
2	245	267	242	227	271	219	268	286
3	1472	1494	1462	1582	1613	1323	1490	1493
4	105	66	103	103	118	98	101	118
5	54	41	62	64	55	59	63	59
6	193	209	184	235	207	172	223	156
7	147	93	122	160	139	130	152	101
8	1102	674	957	1137	1058	990	1098	878
9	720	1033	566	874	628	646	706	320
10	253	143	171	265	193	226	247	99
11	685	586	750	803	830	615	699	777
12	488	355	418	570	465	437	467	313
13	198	187	220	203	247	176	209	204
14	360	334	337	365	376	322	363	348
15	1374	1506	1572	1256	1734	1235	1597	1684
16	156	139	147	175	167	138	164	170

In [39]:

```
# Gerar conjunto de dados de referência - Média
df_mean = df_todos.mean(axis=1)
```

In [40]:

```
# # Imprimir conjunto de dados de referência - Média
df_mean
```

Out[40]:

```
0      406.125
1       59.500
2      253.125
3     1491.125
4      101.500
5       57.125
6      197.375
7      130.500
8      986.750
9      686.625
10     199.625
11     718.125
12     439.125
13     205.500
14     350.625
15    1494.750
16     157.000
dtype: float64
```

In [41]:

```
# Verifica nome de coluna da tabela de referência média
print(df_mean.index.name)
```

```
None
```

In [42]:

```
# Concatena os conjuntos de dados de máquinas e sensores com o de referência média
df_ref =pd.concat([df_todos, df_mean], axis=1)
```

In [43]:

```
# Verifica os nomes de colunas
df_ref.columns
```

Out[43]:

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 0], dtype='object')
```

In [44]:

```
# Nomeia a coluna de referência
df_Ref = df_ref.rename(columns={df_ref.columns[-1]:'Ref'})
```

In [45]:

```
df_Ref.columns
```

Out[45]:

```
Index(['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'Ref'], dtype='object')
```

In [46]:

```
# Verifica os nomes de colunas
df_Ref
```

Out[46]:

	V1	V2	V3	V4	V5	V6	V7	V8	Ref
0	375	135	458	475	509	336	469	492	406.125
1	57	47	53	73	63	62	63	58	59.500
2	245	267	242	227	271	219	268	286	253.125
3	1472	1494	1462	1582	1613	1323	1490	1493	1491.125
4	105	66	103	103	118	98	101	118	101.500
5	54	41	62	64	55	59	63	59	57.125
6	193	209	184	235	207	172	223	156	197.375
7	147	93	122	160	139	130	152	101	130.500
8	1102	674	957	1137	1058	990	1098	878	986.750
9	720	1033	566	874	628	646	706	320	686.625
10	253	143	171	265	193	226	247	99	199.625
11	685	586	750	803	830	615	699	777	718.125
12	488	355	418	570	465	437	467	313	439.125
13	198	187	220	203	247	176	209	204	205.500
14	360	334	337	365	376	322	363	348	350.625
15	1374	1506	1572	1256	1734	1235	1597	1684	1494.750
16	156	139	147	175	167	138	164	170	157.000

In [47]:

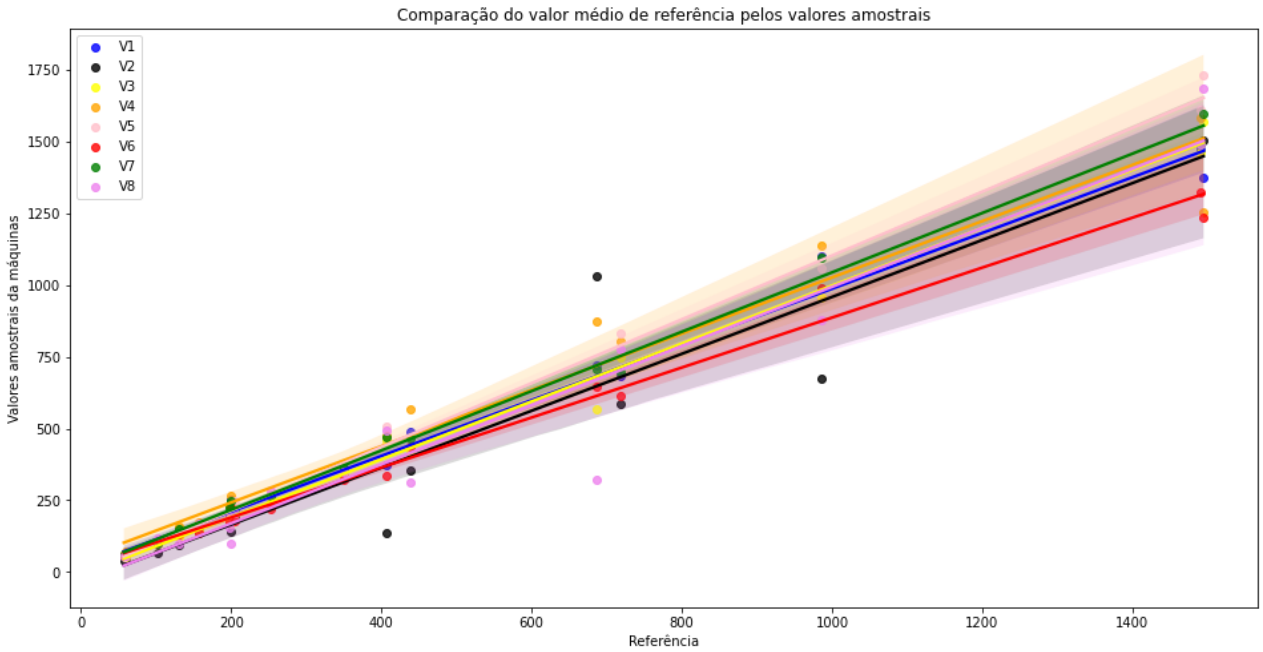
```
# PCA - Transformação com valor de referência

plt.figure(figsize=(16,8))
sns.regplot(data=df_Ref, x='Ref', y='V1', color='blue',   label='V1')
sns.regplot(data=df_Ref, x='Ref', y='V2', color='black',  label='V2')
sns.regplot(data=df_Ref, x='Ref', y='V3', color='yellow', label='V3')
sns.regplot(data=df_Ref, x='Ref', y='V4', color='orange', label='V4')
sns.regplot(data=df_Ref, x='Ref', y='V5', color='pink',   label='V5')
sns.regplot(data=df_Ref, x='Ref', y='V6', color='red',    label='V6')
sns.regplot(data=df_Ref, x='Ref', y='V7', color='green',  label='V7')
sns.regplot(data=df_Ref, x='Ref', y='V8', color='violet', label='V8')

plt.gca().set_xlabel('Referência')
plt.gca().set_ylabel('Valores amostrais da máquinas')
plt.title('Comparação do valor médio de referência pelos valores amostrais')
plt.legend()
```

Out[47]:

```
<matplotlib.legend.Legend at 0x1da76ff1cd0>
```



Plota distribuições univariadas ou bivariadas usando a estimativa de densidade de kernel.*

*** Um gráfico de estimativa de densidade de kernel (KDE) é um método para visualizar a distribuição de observações em um conjunto de dados, análogo a um histograma. O KDE representa os dados usando uma curva de densidade de probabilidade contínua em uma ou mais dimensões.**

Conclusão

Com base nos dados e informações númericas e gráficas pode se observar que a máquina com maior desvio nos elementos sensores e, comparação as outras máquinas é a ***máquina V2***.

Observações: Também é necessário corrigir os processos as máquinas: V3 e V8

In [ ]: