WORKING WITH NANOPARTICLE TOXICITY DATASET IN PYTHON

PROGRAMMING FOR CHEMISTS

KIRILL NIKIFIROV
INSAF GAINANOV
GALINA BELYAEVA
VLADIMIR GABREIAN

OUTLINE

GitHub

Process of work

Attempts

Program

Conclusions

GITHUB

WE WON IT.

BUT NOT FOR THE FIRST ATTEMPT

Belvglin Создано в Colab — a3b4da9 · 1 minute ago — **18 Commits**

Nanoparticle-Toxicity-Dataset	alo	2 weeks ago
Kirill_Nanopart_tox_dataset.ipynb	alo	51 minutes ago
Nanoparticles_Galya.ipynb	Создано в Colab	1 minute ago
README.md	first commit	2 weeks ago
aboba.txt	alo	2 weeks ago
attention.txt	alo	2 weeks ago
belv.docx	belv	last week
insaf.txt	Insaf	2 weeks ago
nanotox_dataset.csv	alo	2 weeks ago
proba.txt	alo	2 weeks ago
prov2.txt	alo	2 weeks ago
text.txt	alo	2 weeks ago
textovic.txt	alo	2 weeks ago
Проба.ipynb	Создано в Colab	last week

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 3

StuffWeDid Kirill
InsafGain
Belvglin

Languages

PROCESS OF WORK

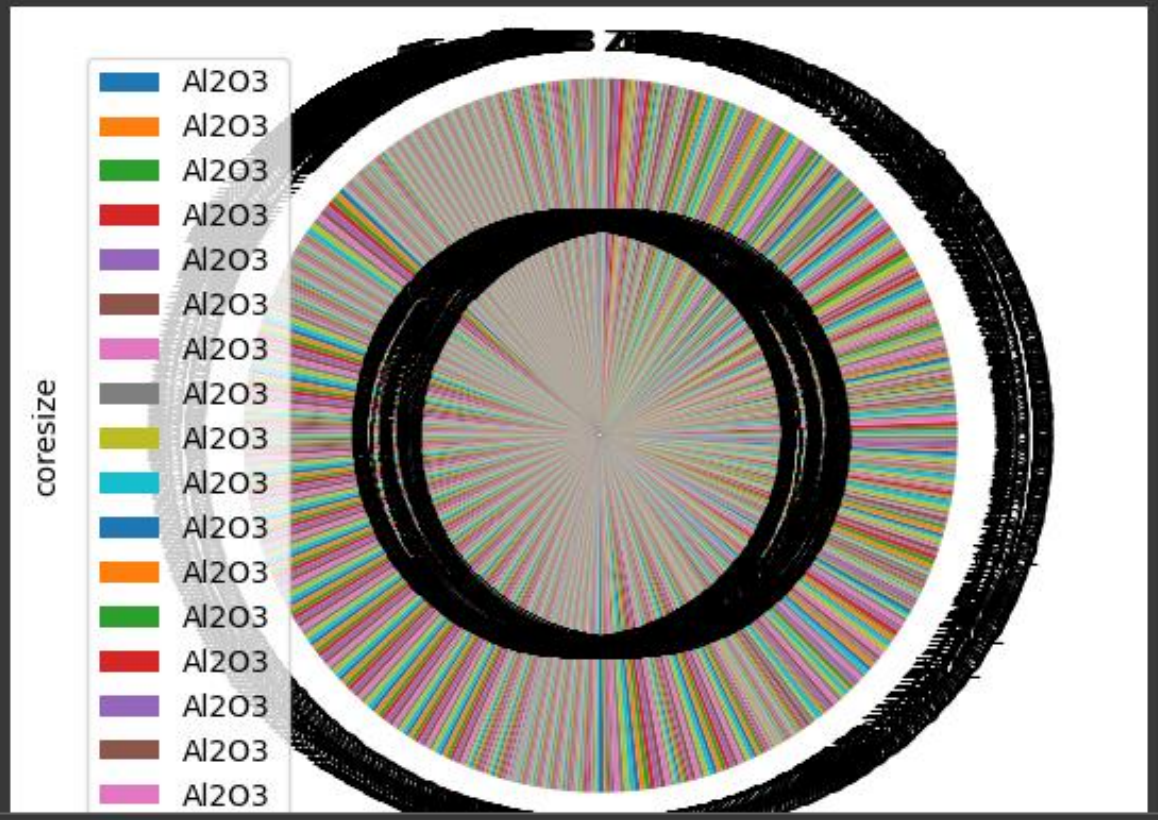
- Choosing a database
- Fighting with GitHub and GitBush: discovered tokens
- Structuring the Dataset
- Spending a lot of hours on finding the best relations of parameters and matching them to the conscious result
- Looking through a lot of courses and papers
- Finally got it.

BUT NOT WITHOUT FAILS

YES, WE ARE CHEMIST, BE TOLERANT

WE GOT THIS...

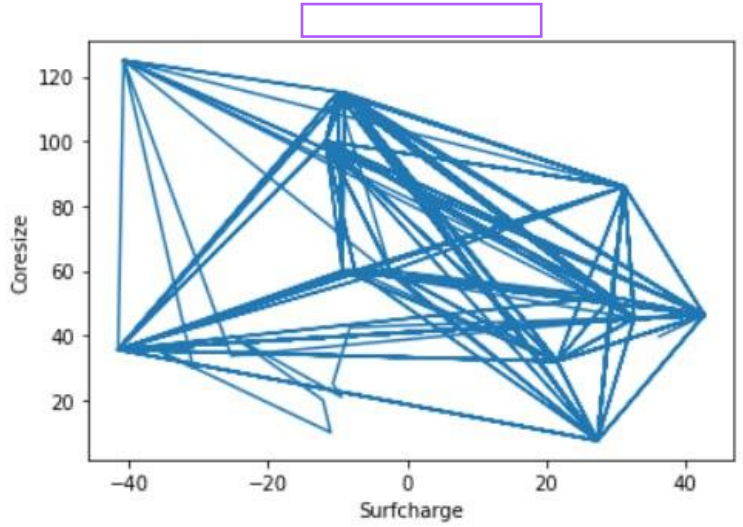
```
import pandas as pd
cvsData = pd.read_csv("nanotox_dataset.csv")
cvsData.plot.pie(y='coresize', labels=cvsData['NPs'], autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.show()
```

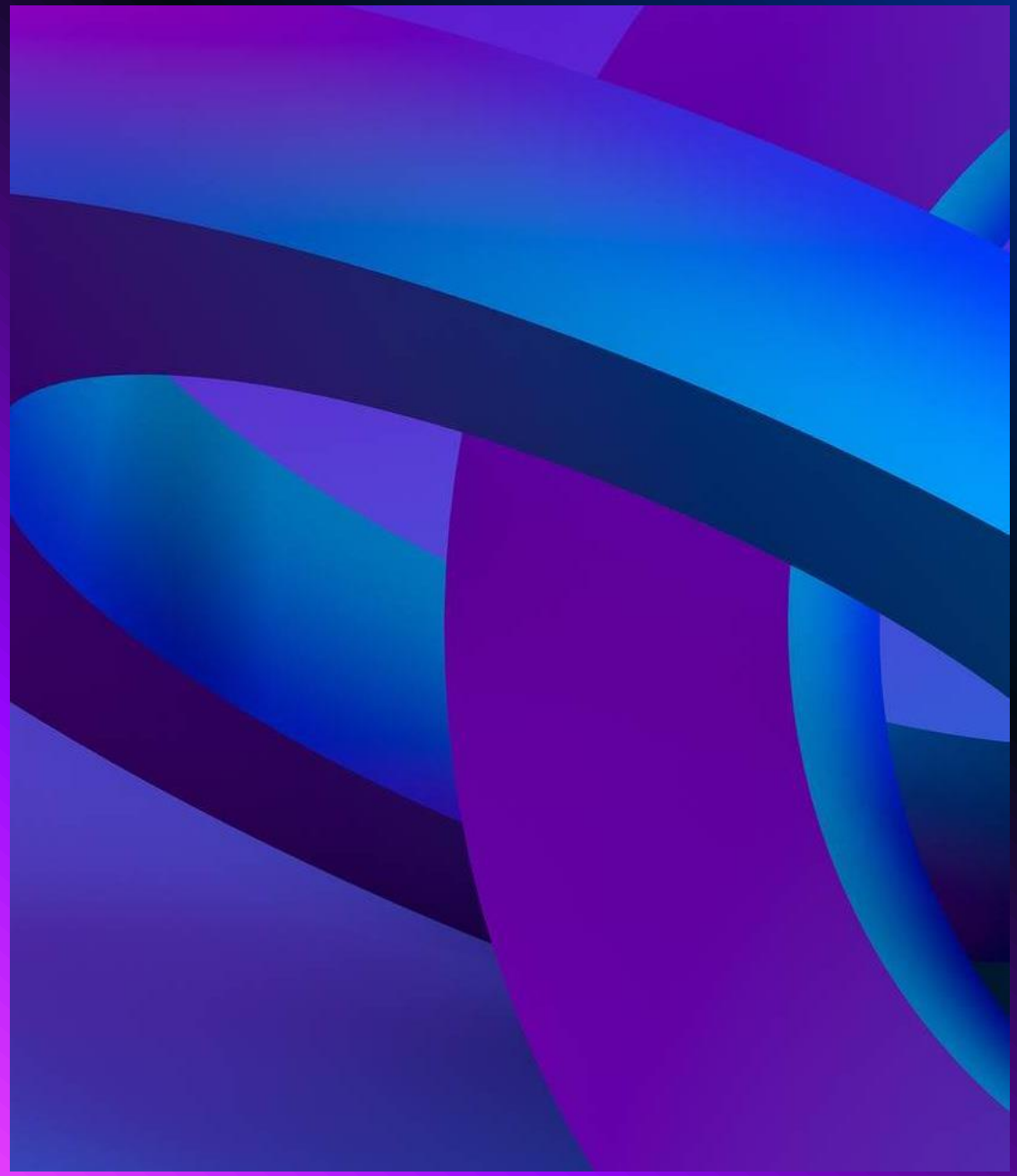


AND THIS...

Ввод [3]:

```
import matplotlib.pyplot as plt
import numpy as np
import csv

X = []
Y = []

with open('nanotox_dataset.csv', 'r', encoding='utf-8') as datafile:
    plotting = csv.reader(datafile, delimiter=',')
    for ROWS in plotting:
        X.append(float(ROWS[3]))
        Y.append(float(ROWS[1]))
plt.plot(X, Y)
plt.title(
plt.xlabel('Surfcharge')
plt.ylabel('Coresize')
plt.show()
```



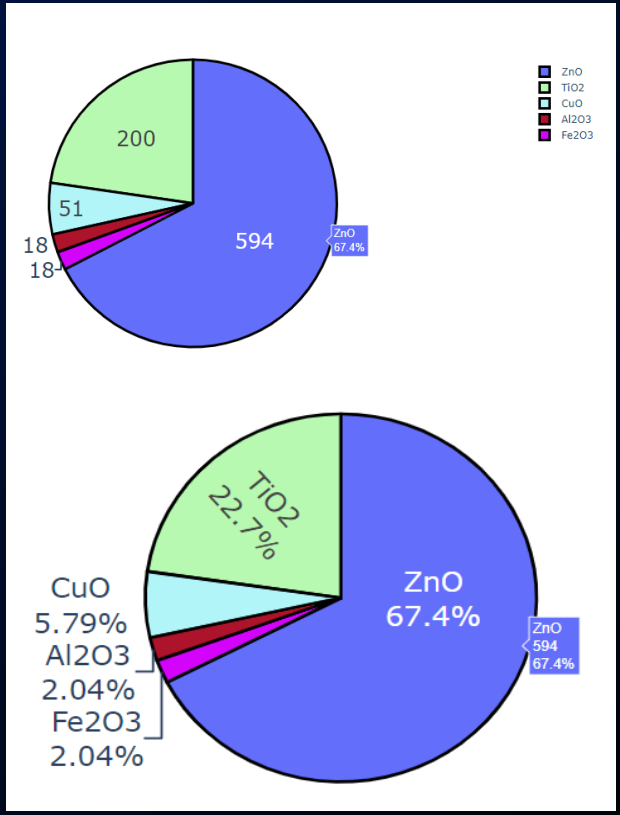


FIRST OF ALL WE FIND OUT ALL RELATIONS WE COULD

It was challenging due to:

- Experimental type of data
- Abundance of source data and only one column with results
- In addition this one parameter was uncountable and has data like ‘yes/no’

WE STRUCTURED OUR DATASET TO GET RID OF PARAMETERS WE DON'T NEED




```
# Sorting by column 'Name of column'
cvsData.sort_values(by=['hydrosize','class'],ascending=[False, False])
cvsData.drop("Ec", axis = 1) # Удалить переменную из фрейма данных
#Параметр «axis» в Pandas определяет направление выполнения операции:
#Движение вниз по строкам (по умолчанию) соответствует axis=0, что эквивалентно операциям агрега
#Движение вбок по столбцам соответствует axis=1, что эквивалентно операциям агрегации данных по
```

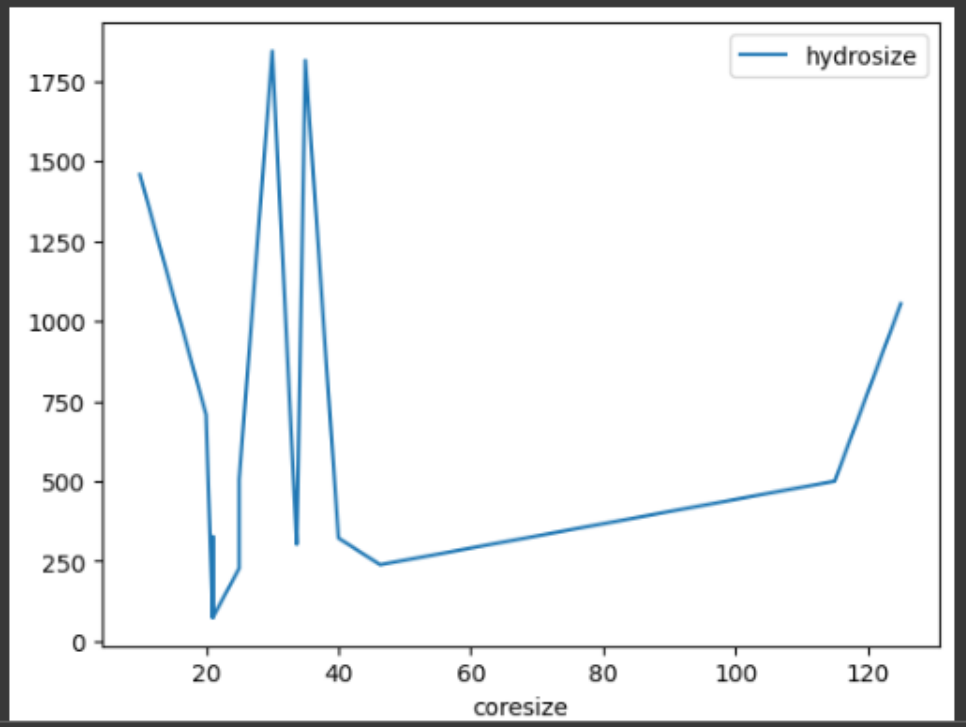
	NPs	coresize	hydrosize	surfcharge	surfarea	Expotime	dosage	e	NOxygen	class
0	Al2O3	39.7	267.0	36.3	64.7	24	0.001	1.61	3	nonToxic
1	Al2O3	39.7	267.0	36.3	64.7	24	0.010	1.61	3	nonToxic
2	Al2O3	39.7	267.0	36.3	64.7	24	0.100	1.61	3	nonToxic
3	Al2O3	39.7	267.0	36.3	64.7	24	1.000	1.61	3	nonToxic
4	Al2O3	39.7	267.0	36.3	64.7	24	5.000	1.61	3	nonToxic
...	...	...	...	...	...	...	...	...	...	...
876	ZnO	45.3	310.0	32.7	21.3	24	20.000	1.65	1	Toxic
877	ZnO	32.0	1093.0	21.6	37.0	24	25.000	1.65	1	Toxic
878	ZnO	46.3	239.0	42.8	24.1	12	100.000	1.90	1	Toxic

```
import plotly as py
import plotly.graph_objs as go
import numpy as np
groups = ['Al2O3', 'Fe2O3', 'TiO2','CuO','ZnO']
amount = [18,18,200,51,594]
colors = ['#ad132a', '#d303fc', '#b7f9b1', '#b1f5f9']

trace = go.Pie(labels=groups, values=amount,
               hoverinfo='label+percent', textinfo='value',
               textfont=dict(size=25),
               marker=dict(colors=colors,
                           line=dict(color='#000000', width=3)))
from plotly.offline import iplot
iplot([trace])
```

SO AT FIRST WE JUST DRAW SOME GRAPHS

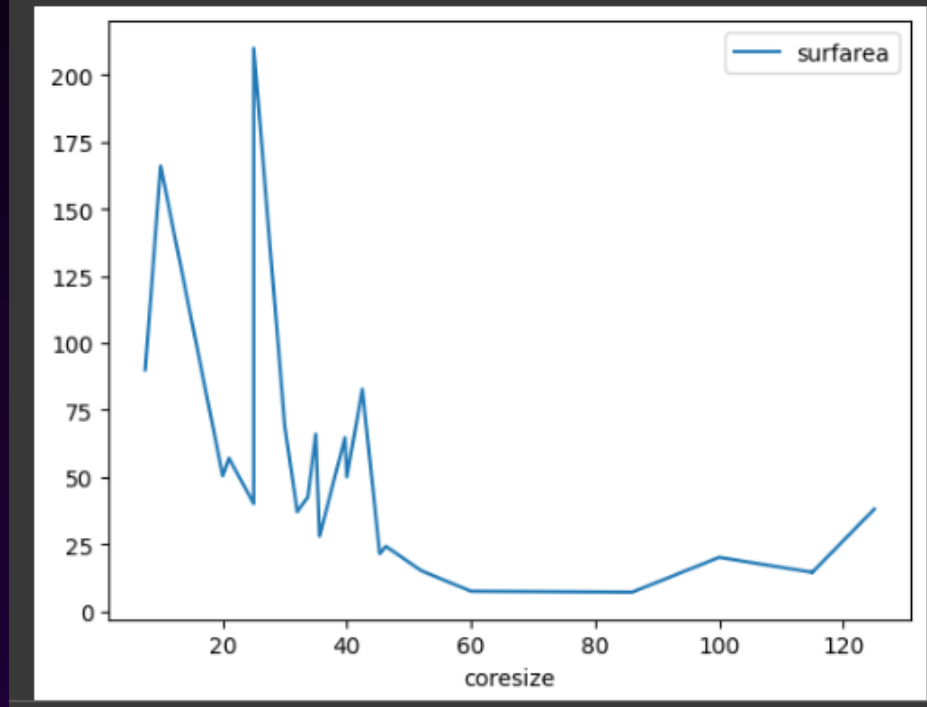
```
data.sort_values(data.columns[1],inplace=True)
filtr_data = data[data['NPs'] == 'TiO2']
filtr_data.plot(x='coresize', y='hydrosize')
```

```
<Axes: xlabel='coresize'>
```



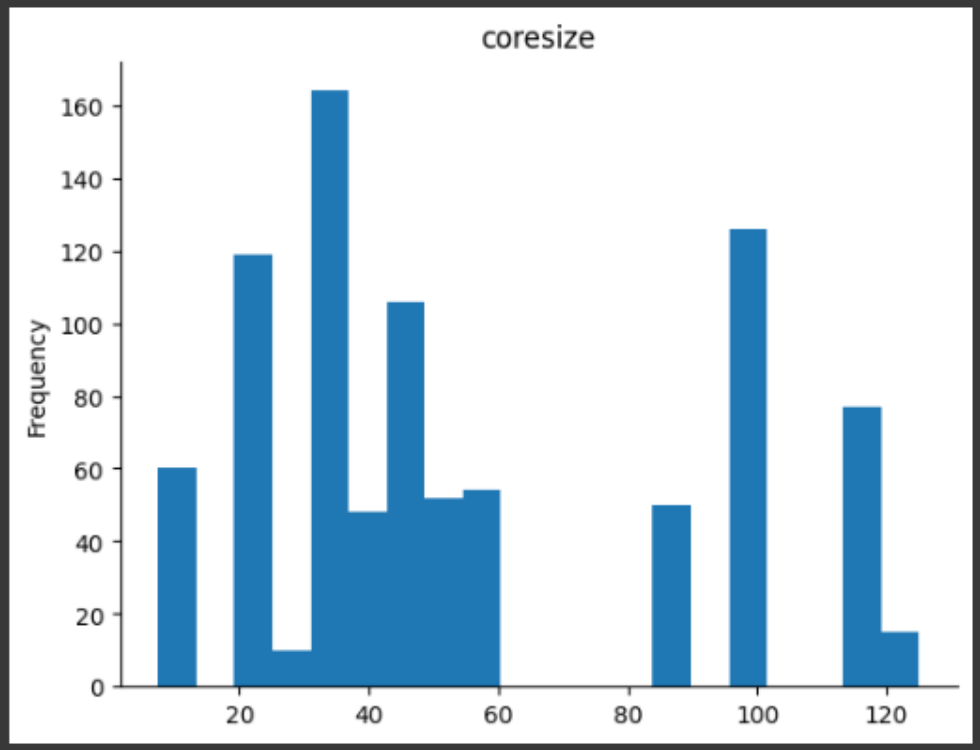

```
import pandas as pd
cvsData = pd.read_csv("nanotox_dataset.csv")
cvsData.sort_values(cvsData.columns[1],inplace=True)
cvsData.plot(x='coresize',y='surfarea')
```

```
<Axes: xlabel='coresize'>
```



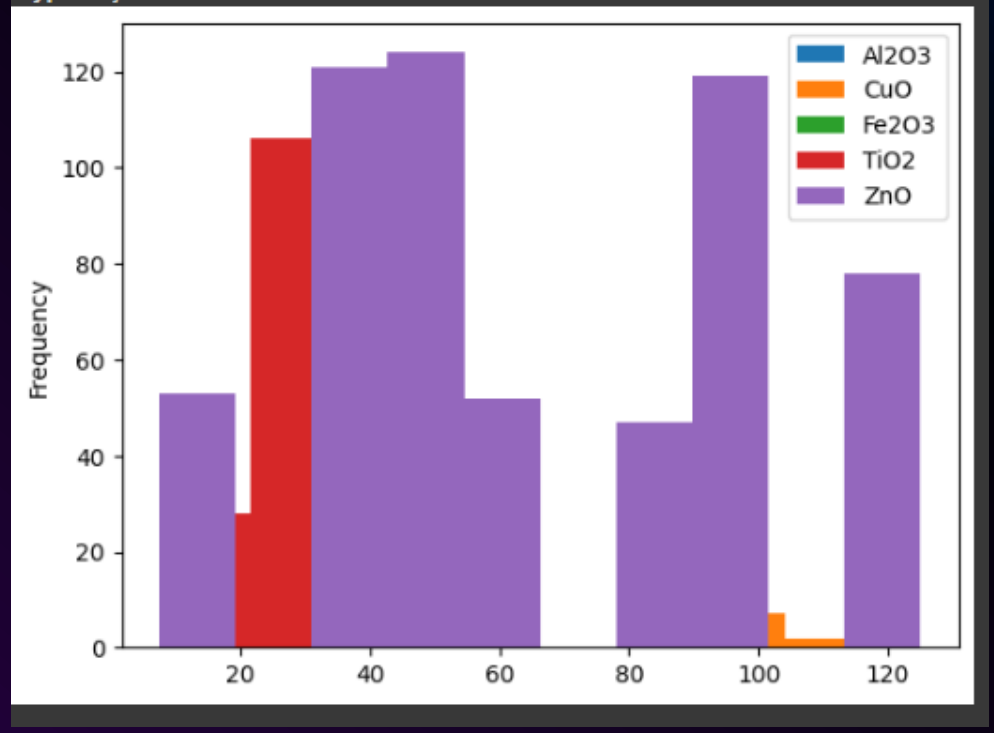
AND MORE

```
from matplotlib import pyplot as plt
df_12['coresize'].plot(kind='hist', bins=20, title='coresize')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

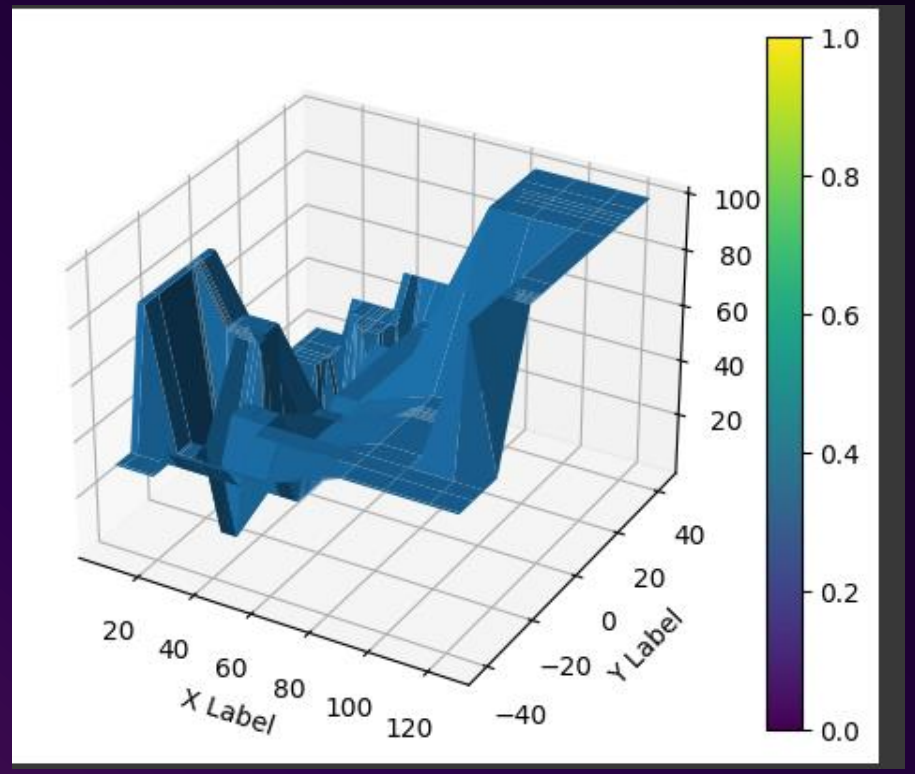



```
df.groupby('NPs')['coresize'].plot(kind='hist', legend = True)
```

```
dtype: object
```

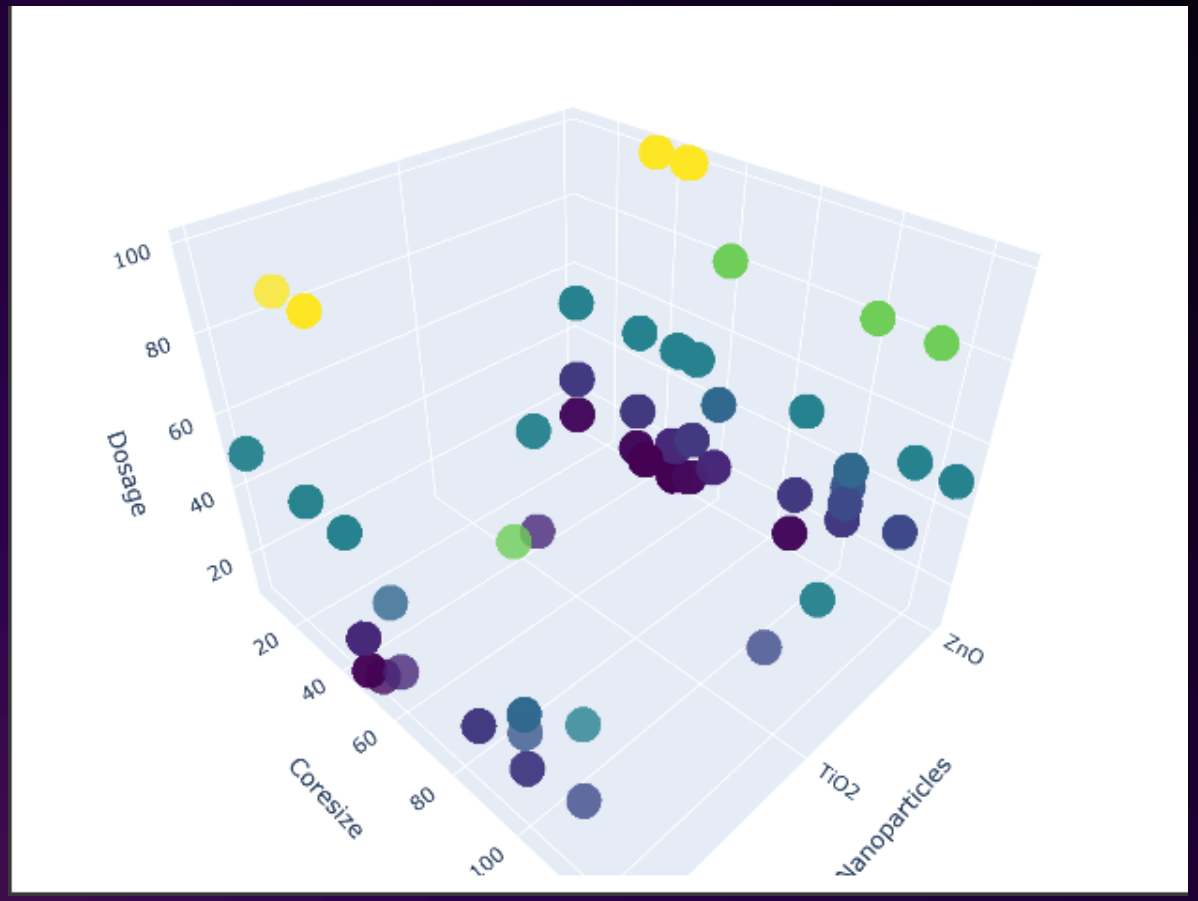


BUT WE UPGRADE OUR SKILL AND GOT THIS



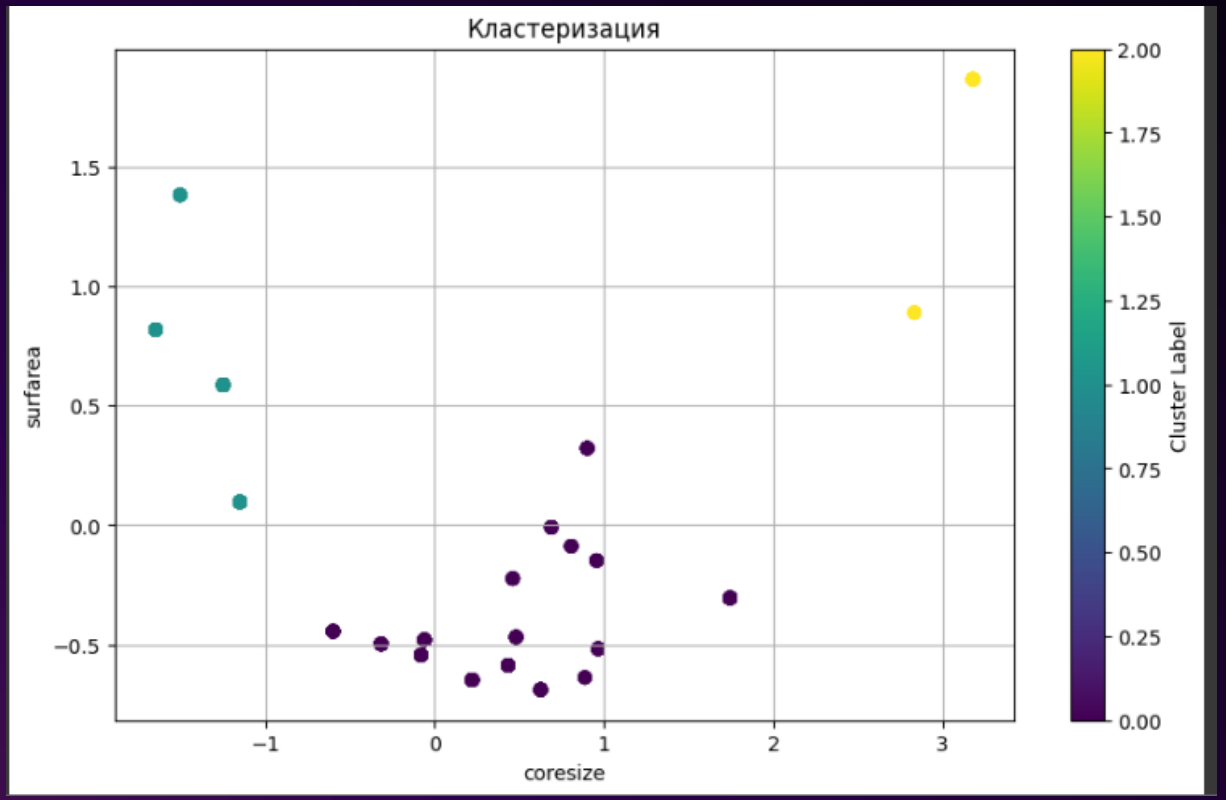

```python
x = data['coresize'].values
y = data['surfcharge'].values
z = data['dosage'].values

# Создание сетки
X = np.unique(xsurf)
Y = np.unique(ysurf)
X, Y = np.meshgrid(X, Y)

# Интерполяция Z значений
Z = griddata((xsurf, ysurf), z, (X, Y), method='nearest')

# Создание 3D графика
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Построение поверхности
surf = ax.plot_surface(X, Y, Z)

# Добавление цветовой шкалы
fig.colorbar(surf)

# Добавление подписей
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')

# Отображение графика
plt.show()
```

AND ALSO THIS



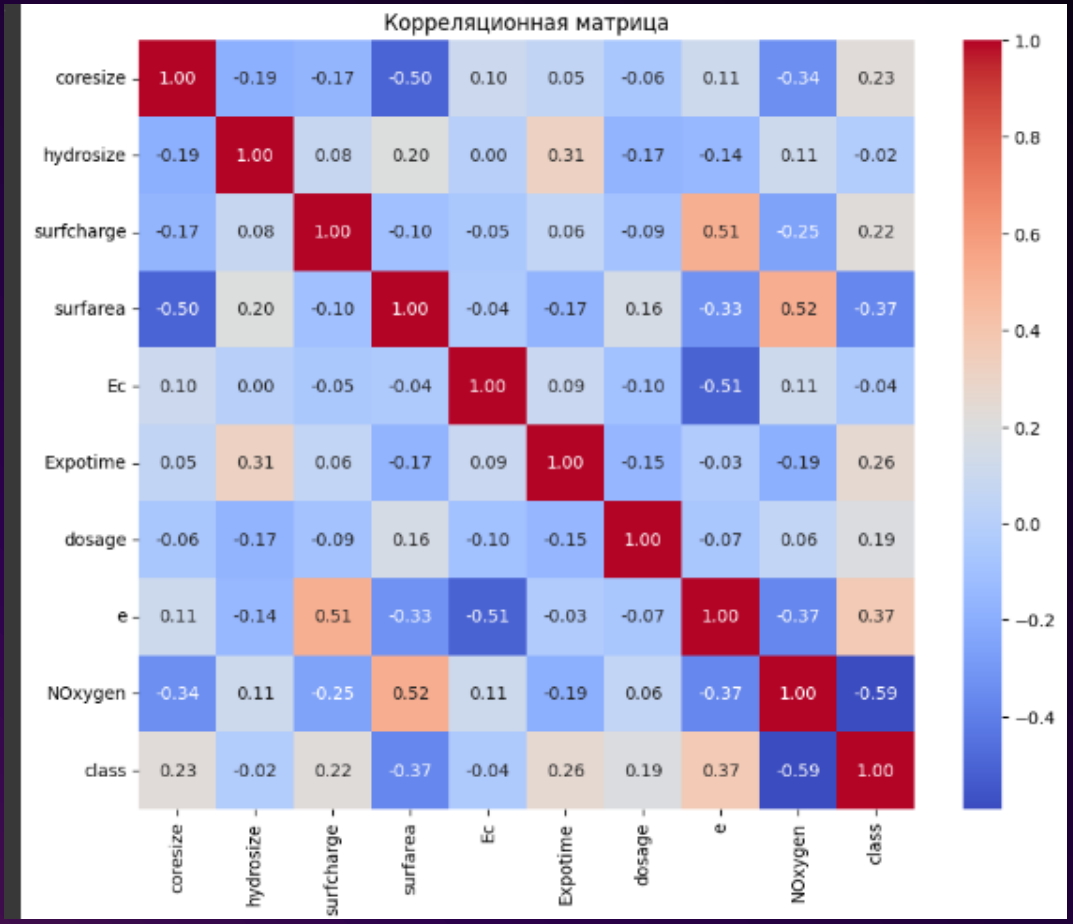

```
#Это 3д модель распределения токсичных частиц по размеру и до
x = df3['NPs']
y = df3['coresize']
z = df3['dosage']

trace2 = go.Scatter3d(
    x=x,
    y=y,
    z=z,
    mode='markers',
    marker=dict(
        size=12,
        color=z,
        colorscale='Viridis',
        opacity = 0.8))

data2 = [trace2]
fig2 = go.Figure(data=data2, layout=layout)
fig2.update_layout(scene = dict(
                    xaxis_title='Nanoparticles',
                    yaxis_title='Coresize',
                    zaxis_title='Dosage'),
                    width=700,
                    margin=dict(r=20, b=10, l=10, t=10))
iplot(fig2, filename='3d-plot')
```

WE ALSO MADE A FEW MORE THINGS



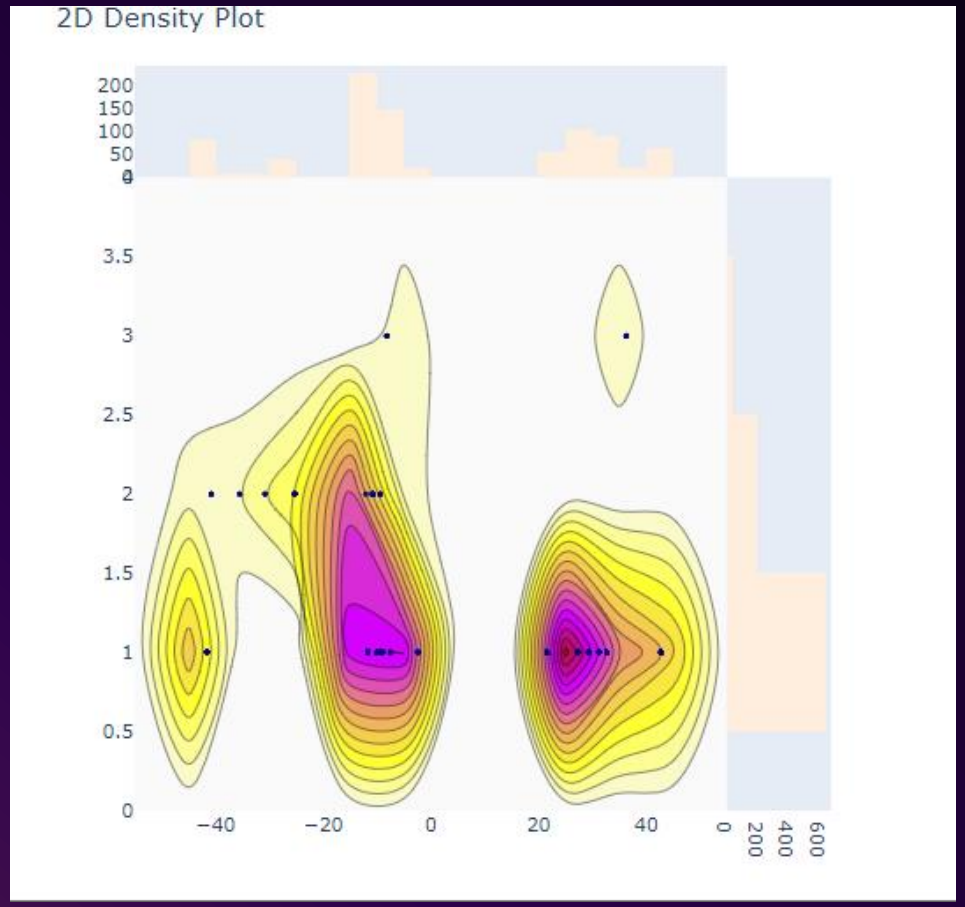

```python
# Шаг 1: Загрузка данных
df = pd.DataFrame(data, columns=['coresize', 'surfarea'])
# Шаг 2: Предобработка данных
# Удаляем строки с пропущенными значениями и выбираем только числовые колонки
data_cleaned = df.dropna()
X = data_cleaned.select_dtypes(include=[np.number])

# Нормализация данных
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Шаг 3: Кластеризация K-means
kmeans = KMeans(n_clusters=3, random_state=10)  # Выберите количество кластеров
kmeans.fit(X_scaled)
labels = kmeans.labels_

# Шаг 4: Визуализация результатов
# Снижение размерности до 2D для визуализации
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(10, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis', marker='o')
plt.title('Кластеризация')
plt.xlabel('coresize')
plt.ylabel('surfarea')
plt.colorbar(label='Cluster Label')
plt.grid()
plt.show()
```

WE ALSO MADE A FEW MORE THINGS



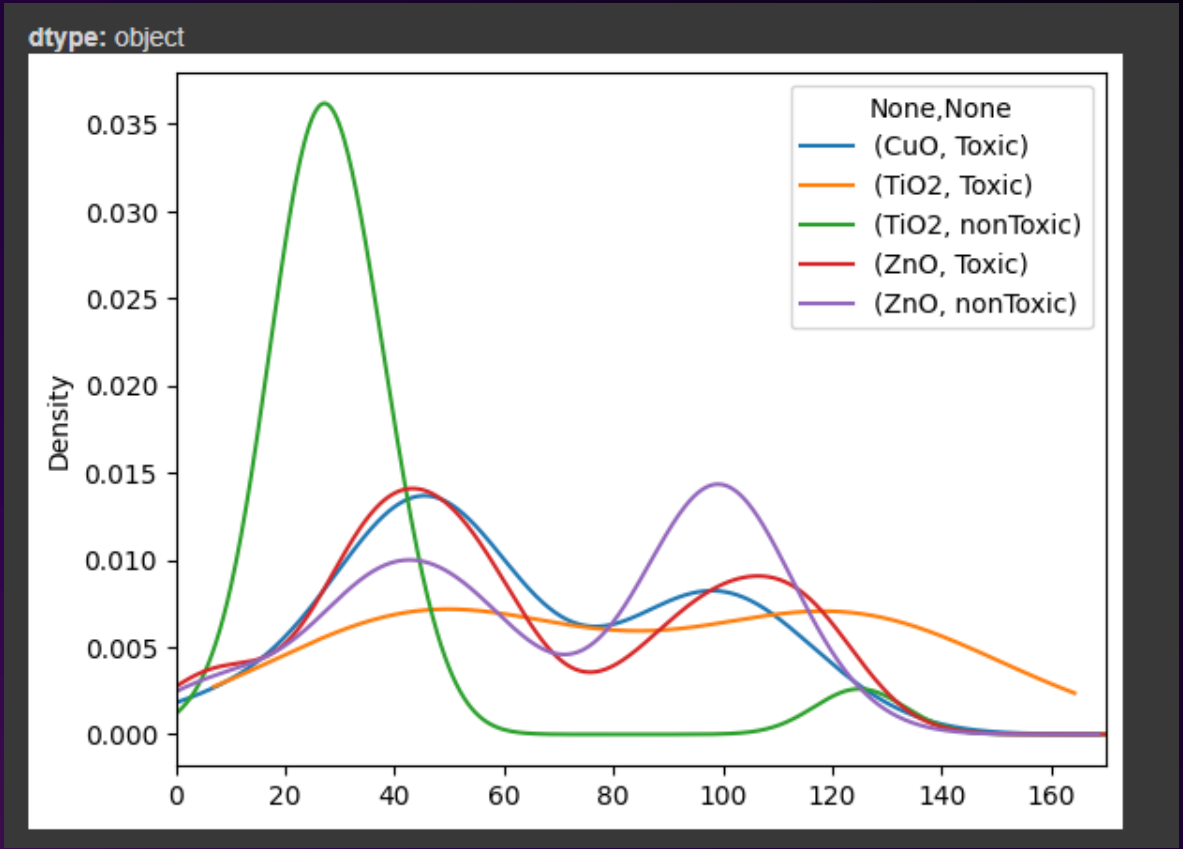

```python
# Шаг 2: Извлечение только числовых данных
numeric_data = data.select_dtypes(include=['number'])

# Шаг 3: Построение корреляционной матрицы
correlation_matrix = numeric_data.corr()

# Шаг 4: Визуализация корреляционной матрицы
plt.figure(figsize=(10, 8))  # Задаем размер графика
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',
            xticklabels=correlation_matrix.columns,
            yticklabels=correlation_matrix.index)
plt.title('Корреляционная матрица')
plt.show()
```

WHAT ABOUT MORE SPECIFIC INFORMATION?




```
#здесь представлена плотность распределения по заряду поверхности и количеству атомов кислорода
import plotly.figure_factory as ff
x = df['surfcharge']
y = df['NOxygen']
colorscale = ['#ad132a', '#d303fc', 'rgb(236,158,105)', (1,1,0.2), (0.98,0.98,0.98)]
fig1 = ff.create_2d_density(
    x,y,colorscale=colorscale,
    hist_color='rgb(255,237,222)', point_size=3)

iplot(fig1, filename='histogram_subplots')
```

TOXICITY



- `df2.groupby(['NPs','class'])['coresize'].plot(kind='kde', xlim=[0,170], legend = True)`

THANKS FOR YOUR ATTENTION

(We really did our best)