In [261…
```
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
df=pd.read_csv("e11.csv")
```

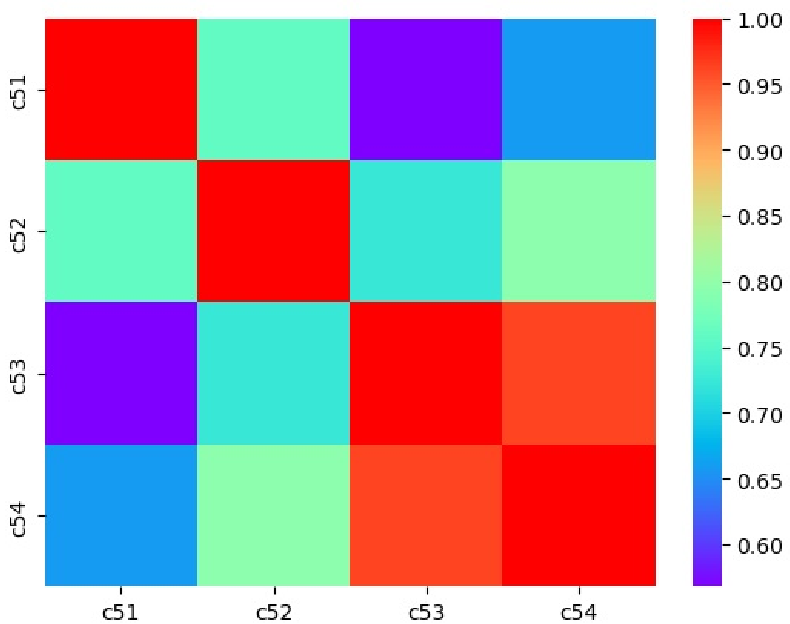
In [262…
```
fig=px.line(df,x="c1",y=["c168","c169","c170","c171"])
fig.show()
```

In [263…
```
vibdf=df[["c1","c51","c52","c53","c54"]]
```

In [264…
```
correlation=vibdf.corr()
correlation
import seaborn as sns
sns.heatmap(correlation,cmap="rainbow")
### c53 and c54 are closely related
```

```
C:\Users\amish\AppData\Local\Temp\ipykernel_388\144836469.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fals
e. Select only valid columns or specify the value of numeric_only to silence this warning.
```

Out[264]:
```
<Axes: >
```



In [265…
```
fig2=px.box(df,y=["c26", "c27", "c28", "c29", "c30", "c31", "c32",
"c33", "c39", "c139", "c142", "c143", "c155", "c156", "c157", "c158", 'c160', "c161", "c162", "c163"])
```

```
fig2.show()
```

In [266…
```
fig4=px.box(df,y="c4")
fig4.show()
```

In [267…
```
len(df['c1'])
```

Out[267]: 1025

In [268…
```
df["c200"].empty
```

Out[268]: False

In [269…
```
dropcol=[]
j=0
for i in df.columns:
    if pd.isnull(df[i]).all()==True:
        dropcol.append(i)
        j=j+1
```

```
        df.drop(i,axis=1,inplace=True)
```

In [270…
```
dropcol
```

Out[270]:
```
['c199', 'c202', 'c204', 'c226', 'c229']
```

In [271…
```
df
```

Out[271]:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	...	c231	c232	c233
0	01-09-2018	2	164.154800	155.467568	0.707847	1.227071	2.264965	18.978886	7.561194	0.719497	...	NaN	NaN	NaN
1	02-09-2018	2	162.115737	153.645223	0.454140	0.671909	2.260011	18.639050	7.844962	0.702348	...	NaN	NaN	NaN
2	03-09-2018	2	153.718388	142.880092	0.634565	0.683161	2.085850	16.671939	6.708392	0.656799	...	NaN	NaN	NaN
3	04-09-2018	2	169.214952	157.342985	0.646557	0.722743	2.258069	17.871757	7.239110	0.662394	...	NaN	NaN	NaN
4	05-09-2018	2	176.335010	163.307519	0.683941	0.681609	2.252392	18.591802	7.257898	0.689549	...	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	18-06-2021	2	170.995057	165.998017	0.610354	2.492540	2.150980	19.448113	8.506249	0.577761	...	27.423187	27.447340	26.236872
1021	19-06-2021	2	171.841123	166.840518	0.335817	2.580334	2.145125	19.449504	8.916708	0.585410	...	27.515604	27.473610	26.253992
1022	20-06-2021	2	173.074392	168.084249	0.368131	2.711153	2.149443	19.411187	9.235679	0.586067	...	27.514629	27.480869	26.255411
1023	21-06-2021	2	173.254503	168.244718	0.461550	2.724708	2.144927	19.426058	9.236836	0.583313	...	27.554869	27.514990	26.290037
1024	22-06-2021	2	173.474188	168.481954	0.359504	2.706028	2.144609	19.457788	9.249983	0.587881	...	27.570024	27.513462	26.298931

1025 rows × 235 columns

In [272…
```
contro_para=["c26", "c27", "c28", "c29", "c30", "c31", "c32",
"c33", "c39", "c139", "c142", "c143", "c155", "c156", "c157", "c158", 'c160', "c161", "c162", "c163"]
```

In [273…
```
pd.isna(df['c231']).all()
```

Out[273]:
```
False
```

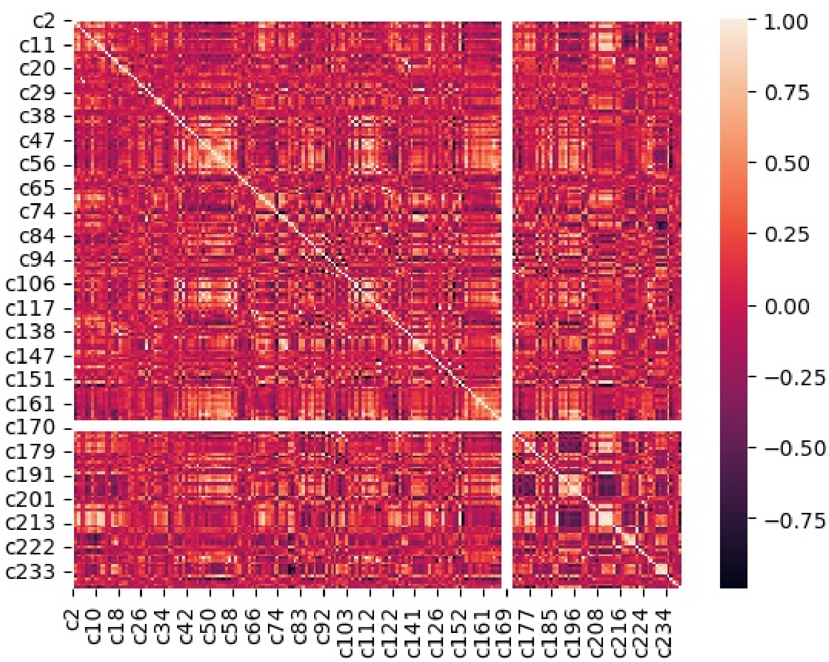
In [274…
```
sns.heatmap(df.corr())
df.corr()
```

```
C:\Users\amish\AppData\Local\Temp\ipykernel_388\1263487528.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fals
e. Select only valid columns or specify the value of numeric_only to silence this warning.

C:\Users\amish\AppData\Local\Temp\ipykernel_388\1263487528.py:2: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fals
e. Select only valid columns or specify the value of numeric_only to silence this warning.
```

Out[274]:

	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	...	c230	c231	c2
c2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	N
c3	NaN	1.000000	0.485878	0.033676	0.121921	0.468627	0.394117	-0.117518	0.284288	0.375895	...	-0.036415	0.388217	0.1985
c4	NaN	0.485878	1.000000	-0.025150	0.190722	0.394992	0.687367	0.489179	0.464843	0.796756	...	0.131032	0.017885	0.0081
c5	NaN	0.033676	-0.025150	1.000000	0.063986	-0.024854	0.055179	-0.121926	0.027878	-0.010756	...	0.028892	-0.004866	0.0101
c6	NaN	0.121921	0.190722	0.063986	1.000000	-0.006538	0.162946	0.172658	0.170553	0.113656	...	0.164482	-0.009578	-0.0150
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
c236	NaN	-0.613803	0.058927	-0.087376	-0.033738	-0.363522	-0.145193	0.300451	-0.124603	-0.047125	...	-0.136944	-0.394412	-0.3542
c237	NaN	0.666057	0.122517	0.058810	0.002468	0.496533	0.323096	-0.344793	0.329517	0.162875	...	0.128044	0.045863	0.2910
c238	NaN	0.145531	0.053238	-0.091961	-0.113725	0.086076	0.004329	-0.168504	-0.006342	0.065654	...	-0.174252	-0.048404	-0.1857
c239	NaN	0.079187	0.279739	0.004979	0.115043	0.040694	0.052747	0.258965	0.146370	0.015911	...	-0.116000	-0.089542	-0.1798
c241	NaN	-0.147792	-0.354786	-0.038801	-0.070600	-0.425758	-0.646640	-0.252910	-0.315409	-0.681410	...	-0.013046	-0.270307	-0.1353

214 rows × 214 columns



In [275…

```
fig3=px.imshow(df.corr(),text_auto=True)
fig3.show()
```

```
C:\Users\amish\AppData\Local\Temp\ipykernel_388\2202880543.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fals
e. Select only valid columns or specify the value of numeric_only to silence this warning.
```

In [276…
```
columns=df.columns
columns=columns.delete(0)
columns
```

Out[276]:
```
Index(['c2', 'c3', 'c4', 'c5', 'c6', 'c7', 'c8', 'c9', 'c10', 'c11',
       ...
       'c231', 'c232', 'c233', 'c234', 'c235', 'c236', 'c237', 'c238', 'c239',
       'c241'],
      dtype='object', length=234)
```

In [277…
```
df
```

Out[277]:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	...	c231	c232	c233
0	01-09-2018	2	164.154800	155.467568	0.707847	1.227071	2.264965	18.978886	7.561194	0.719497	...	NaN	NaN	NaN
1	02-09-2018	2	162.115737	153.645223	0.454140	0.671909	2.260011	18.639050	7.844962	0.702348	...	NaN	NaN	NaN
2	03-09-2018	2	153.718388	142.880092	0.634565	0.683161	2.085850	16.671939	6.708392	0.656799	...	NaN	NaN	NaN
3	04-09-2018	2	169.214952	157.342985	0.646557	0.722743	2.258069	17.871757	7.239110	0.662394	...	NaN	NaN	NaN
4	05-09-2018	2	176.335010	163.307519	0.683941	0.681609	2.252392	18.591802	7.257898	0.689549	...	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	18-06-2021	2	170.995057	165.998017	0.610354	2.492540	2.150980	19.448113	8.506249	0.577761	...	27.423187	27.447340	26.236872
1021	19-06-2021	2	171.841123	166.840518	0.335817	2.580334	2.145125	19.449504	8.916708	0.585410	...	27.515604	27.473610	26.253992
1022	20-06-2021	2	173.074392	168.084249	0.368131	2.711153	2.149443	19.411187	9.235679	0.586067	...	27.514629	27.480869	26.255411
1023	21-06-2021	2	173.254503	168.244718	0.461550	2.724708	2.144927	19.426058	9.236836	0.583313	...	27.554869	27.514990	26.290037
1024	22-06-2021	2	173.474188	168.481954	0.359504	2.706028	2.144609	19.457788	9.249983	0.587881	...	27.570024	27.513462	26.298931

1025 rows × 235 columns

In [278…
```
df
```

Out[278]:

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	...	c231	c232	c233
0	01-09-2018	2	164.154800	155.467568	0.707847	1.227071	2.264965	18.978886	7.561194	0.719497	...	NaN	NaN	NaN
1	02-09-2018	2	162.115737	153.645223	0.454140	0.671909	2.260011	18.639050	7.844962	0.702348	...	NaN	NaN	NaN
2	03-09-2018	2	153.718388	142.880092	0.634565	0.683161	2.085850	16.671939	6.708392	0.656799	...	NaN	NaN	NaN
3	04-09-2018	2	169.214952	157.342985	0.646557	0.722743	2.258069	17.871757	7.239110	0.662394	...	NaN	NaN	NaN
4	05-09-2018	2	176.335010	163.307519	0.683941	0.681609	2.252392	18.591802	7.257898	0.689549	...	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	18-06-2021	2	170.995057	165.998017	0.610354	2.492540	2.150980	19.448113	8.506249	0.577761	...	27.423187	27.447340	26.236872
1021	19-06-2021	2	171.841123	166.840518	0.335817	2.580334	2.145125	19.449504	8.916708	0.585410	...	27.515604	27.473610	26.253992
1022	20-06-2021	2	173.074392	168.084249	0.368131	2.711153	2.149443	19.411187	9.235679	0.586067	...	27.514629	27.480869	26.255411
1023	21-06-2021	2	173.254503	168.244718	0.461550	2.724708	2.144927	19.426058	9.236836	0.583313	...	27.554869	27.514990	26.290037
1024	22-06-2021	2	173.474188	168.481954	0.359504	2.706028	2.144609	19.457788	9.249983	0.587881	...	27.570024	27.513462	26.298931

1025 rows × 235 columns

In [ ]:

```
import numpy as np
df.replace(["#REF!","#VALUE!","#DIV/0!"],np.nan,inplace=True)
for i in df.columns:
    if (df.isnull().sum()[i])>0:
        print(df.isnull().sum()[i])
```

In [280…

```
df['c231']
```

Out[280]:

```
0             NaN
1             NaN
2             NaN
3             NaN
4             NaN
          ...
1020    27.423187
1021    27.515604
1022    27.514629
1023    27.554869
1024    27.570024
Name: c231, Length: 1025, dtype: float64
```

In [ ]:

```
for i in df.columns:
    if (df.isna().sum()[i])>0:
        print(i)
```

In [282…

```
for i in columns:
    df[i]=pd.to_numeric(df[i],errors='coerce')
nan_cols=[]
for i in df.columns:
    if (df.isna().sum()[i])>0:
        nan_cols.append(i)
for i in nan_cols:
    df[i].fillna(df[i].median(),inplace=True)
df['c231']
```

Out[282]:
```
0       27.211634
1       27.211634
2       27.211634
3       27.211634
4       27.211634
          ...
1020    27.423187
1021    27.515604
1022    27.514629
1023    27.554869
1024    27.570024
Name: c231, Length: 1025, dtype: float64
```

In [283…
```
df['c231'][0]=df['c231'].median()
df['c231'].median()
```

```
C:\Users\amish\AppData\Local\Temp\ipykernel_388\1663566827.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[283]:
```
27.21163368
```

In [284…
```
for i in df.columns:
    if (df.isnull().sum()[i])>0:
        print(df.isnull().sum()[i])
```

In [285…
```
px.line(df,x='c1',y='c141')
```

In [286…
```
px.line(df,x="c1",y="c141")
```

In [ ]:

```
new_df=df.copy()
for col in columns:
        q1=df[col].quantile(0.25)
        q2=df[col].quantile(0.50)
        q3=df[col].quantile(0.75)
        low=q1-1.5*(q3-q1)
        high=q3+1.5*(q3-q1)
        rolling_median = df[col].rolling(window=10, min_periods=1,center=True).median()
        for j in range(0,len(new_df)):
            if(new_df[col][j]>high or new_df[col][j]<low):
                new_df[col][j]=rolling_median[j]
```

In [288…

```
px.line(new_df,x="c1",y=["c51","c52","c53","c54"])
```

In [289…

```
dfroll=new_df.rolling(15, min_periods=1, center=True,).mean()
```

```
C:\Users\amish\AppData\Local\Temp\ipykernel_388\1418382976.py:1: FutureWarning:

Dropping of nuisance columns in rolling operations is deprecated; in a future version this will raise TypeError
. Select only valid columns before calling the operation. Dropped columns were Index(['c1'], dtype='object')
```

In [290…

```
px.line(df,x="c1",y=["c51","c52","c53","c54"])
```

In [291…

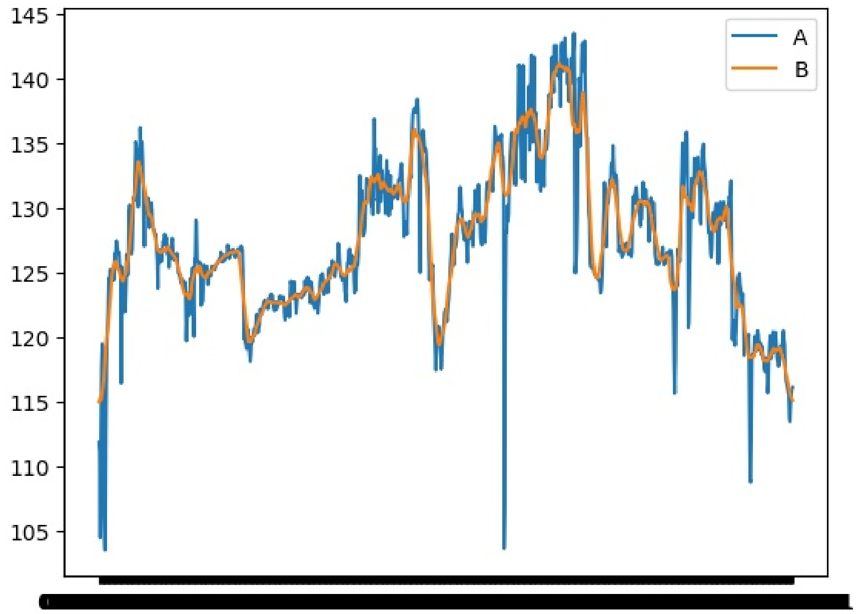
```
px.line(dfroll,y="c232")
#px.line(df,y="c235")
```

In [292…

```
px.line(vibdf,x='c1',y=["c51","c52","c53","c54"])
```

In [293…
```python
import statsmodels.api as sm
```

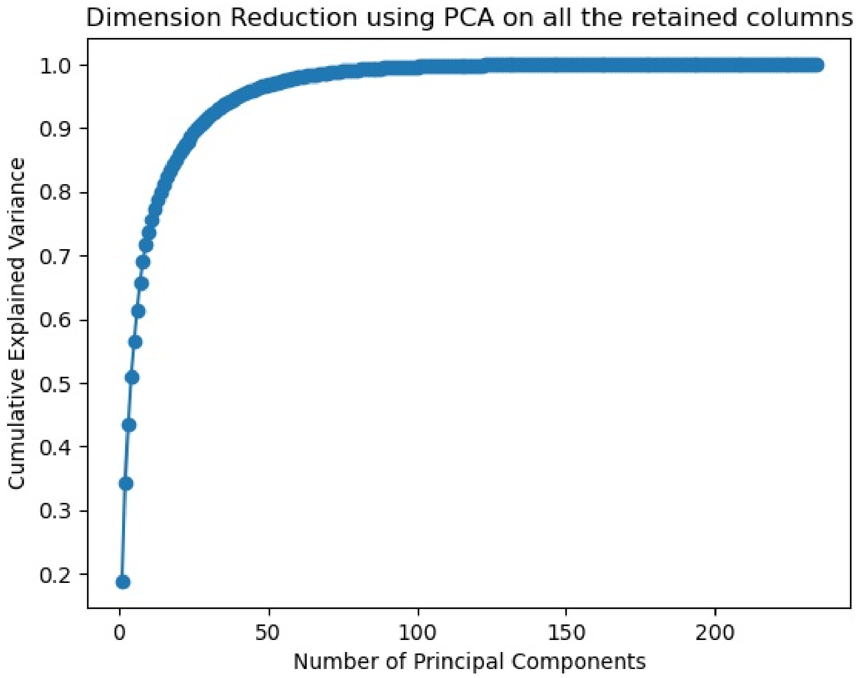
In [294…
```python
plt.plot(df['c1'],df['c141'])
plt.plot(df['c1'],dfroll['c141'])
#plt.plot(df['c1'],new_df['c111'])
plt.legend(["A","B","C"])
plt.show()
```



In [295…
```python
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
std_scaler = StandardScaler()
scaledX = std_scaler.fit_transform(new_df.drop('c1',axis=1))
pca = PCA()
XPCA3 = pca.fit_transform(scaledX)
```

In [ ]:
```python

```

In [296…
```python
cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
import matplotlib.pyplot as plt

plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance,marker='o')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance')
plt.title("Dimension Reduction using PCA on all the retained columns")
plt.show()
```



In [297…

```
dfroll.corr()
```

Out[297]:

	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	...	c231	c232	c2
c2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	Na
c3	NaN	1.000000	0.398238	0.043649	0.143847	0.508162	0.469668	-0.295607	0.225339	0.449793	...	0.397575	0.219655	0.3306
c4	NaN	0.398238	1.000000	-0.082387	0.193211	0.162920	0.636827	0.384562	0.477439	0.784189	...	-0.003779	-0.069100	-0.0007
c5	NaN	0.043649	-0.082387	1.000000	0.117115	-0.075335	0.053457	-0.219698	0.018203	-0.147430	...	-0.012877	-0.001187	-0.0165
c6	NaN	0.143847	0.193211	0.117115	1.000000	-0.083150	0.214008	0.187399	0.220199	0.141709	...	0.002932	0.036607	0.0343
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
c236	NaN	-0.690897	0.099734	-0.121365	-0.059789	-0.423756	-0.147968	0.381635	-0.087703	-0.024661	...	-0.438892	-0.401596	-0.4456
c237	NaN	0.714422	0.074925	0.088033	0.005642	0.592521	0.442277	-0.468451	0.344582	0.203934	...	0.161428	0.208511	0.1901
c238	NaN	0.213232	0.006453	-0.170563	-0.189679	0.163715	0.059517	-0.293532	0.046989	0.220875	...	-0.070992	-0.154310	-0.0548
c239	NaN	0.025189	0.398820	0.077771	0.178232	0.022630	0.050918	0.300711	0.213918	-0.016434	...	-0.069384	-0.129182	-0.0495
c241	NaN	-0.380834	0.060030	-0.003387	-0.196295	-0.053709	-0.267411	0.110509	-0.110077	-0.403652	...	-0.193475	-0.140799	-0.2312

234 rows × 234 columns

In [298…

```
constant=[]
for col in columns:
    if(dfroll[col].std()==0):
        constant.append(col)
        dfroll.drop(col,axis=1,inplace=True)
print(constant)
```

['c2', 'c82', 'c110']

In [299…

```
fig3=px.imshow(dfroll.corr(),text_auto=True)
fig3.show()
```

In [300…

```
dfroll
```

Out[300]:

	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	...	c231	c232
0	167.113977	159.855661	0.657942	0.745669	2.241367	19.150170	7.131911	0.692249	54.210518	9.631048	...	27.211634	27.100533
1	165.889586	159.882557	0.657619	0.768435	2.242767	19.164981	7.001924	0.684414	54.319049	9.645601	...	27.211634	27.100533
2	164.679567	160.202300	0.667353	0.879457	2.206972	19.213539	6.884669	0.679889	54.473979	9.677103	...	27.211634	27.100533
3	164.847401	160.613136	0.669618	0.975480	2.211388	19.311562	6.737859	0.684983	54.628355	9.702877	...	27.211634	27.100533
4	165.609130	160.799746	0.674939	1.047894	2.227731	19.408114	6.663686	0.689954	54.856234	9.762435	...	27.211634	27.100533
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	173.120811	168.123184	0.517241	2.621845	2.144946	19.489063	9.116470	0.600314	58.317157	11.282926	...	27.353846	27.419108
1021	172.915765	167.918900	0.529139	2.612095	2.145441	19.484904	9.097530	0.599794	58.165027	11.259874	...	27.354558	27.418297
1022	172.755395	167.758762	0.535356	2.600491	2.147612	19.468672	9.099571	0.599576	58.057397	11.244555	...	27.365608	27.423436
1023	172.541256	167.545346	0.527250	2.589496	2.149691	19.452580	9.095960	0.599715	57.952107	11.228816	...	27.379114	27.429718
1024	172.353605	167.357269	0.514411	2.581333	2.152201	19.448073	9.078929	0.599231	57.864016	11.215549	...	27.395995	27.437570

1025 rows × 231 columns

In [301…

```
#Dropping c168,c169,c170,c171 as they have Nan correlation.
dfprep=dfroll.drop(['c168','c169','c170','c171'],axis=1)
```

In [302…

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [ ]:

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

def perform_pca(data_frame, variance_retained=0.97):
    standardized_data = StandardScaler().fit_transform(data_frame)
    pca = PCA(n_components=variance_retained)
    principal_components = pca.fit_transform(standardized_data)
    principal_df = pd.DataFrame(data=principal_components, columns=[f'PC{i}' for i in range(1, pca.n_components
    retained_variance = sum(pca.explained_variance_ratio_)
    print(f"Variance retained: {retained_variance * 100:.2f}%")
    return principal_df
result_df = perform_pca(new_df.drop(['c1','c51','c52','c53','c54','c82','c2','c110'],axis=1))
print(result_df)
```

In [ ]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant

def calculate_vif(data_frame):
    data_frame_with_constant = add_constant(data_frame)
    vif_data = pd.DataFrame()
    vif_data["Variable"] = data_frame_with_constant.columns
```

```
    vif_data["VIF"] = [variance_inflation_factor(data_frame_with_constant.values, i) for i in range(data_frame_

    return vif_data
result = pd.DataFrame(calculate_vif(new_df.drop(['c1','c51','c52','c53','c54','c241'],axis=1)))
print(result)
result2 = pd.DataFrame(calculate_vif(result_df))
print(result2)
```

In [305…
```
VIF_rem_col=[]
for i in range(len(result)):
    if result['VIF'][i]<20:
        VIF_rem_col.append(result['Variable'][i])
print(len(VIF_rem_col))
print(VIF_rem_col)
```

```
42
['c2', 'c12', 'c14', 'c20', 'c21', 'c22', 'c23', 'c27', 'c30', 'c34', 'c35', 'c36', 'c37', 'c42', 'c44', 'c45',
'c60', 'c62', 'c63', 'c73', 'c82', 'c110', 'c133', 'c147', 'c156', 'c160', 'c161', 'c162', 'c163', 'c168', 'c16
9', 'c170', 'c171', 'c177', 'c179', 'c190', 'c206', 'c207', 'c218', 'c223', 'c238', 'c239']
```

In [306…
```
X=dfprep[['c3', 'c4', 'c5', 'c6', 'c7', 'c8', 'c9', 'c10', 'c11', 'c12', 'c13', 'c14', 'c15', 'c16', 'c17', 'c1
```

In [307…
```
y=dfprep['c51']
```

In [308…
```
X= sm.add_constant(X)
model=sm.OLS(y,X).fit()
```

In [309…
```
X=result_df[['PC1', 'PC2', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC10',
       'PC12', 'PC13', 'PC14', 'PC15', 'PC16', 'PC17', 'PC18',
       'PC20', 'PC21', 'PC22', 'PC23', 'PC25', 'PC26', 'PC27', 'PC28',
       'PC29', 'PC30', 'PC31', 'PC33', 'PC34', 'PC35', 'PC36', 'PC37',
         'PC40', 'PC41', 'PC44', 'PC45', 'PC46', 'PC48', 'PC52']]
y=new_df['c51']
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
X=sm.add_constant(X)
model1=sm.OLS(y_train,X_train).fit()
print(model1.summary())
y_pred_ols =model1.predict(X_test)
r2_ols = r2_score(y_test, y_pred_ols)
print(f'R-squared (OLS): {r2_ols:.2f}')
c51pred=model1.predict(X)
r3_ols = r2_score(new_df['c51'],c51pred)
print(f'R-squared whole (OLS): {r3_ols:.2f}')
# X=sm.add_constant(X)
# model1=sm.OLS(y,X).fit()
# print(model1.summary())
# y_pred_ols =model1.predict(X)
# r2_ols = r2_score(y, y_pred_ols)
# print(f'R-squared (OLS): {r2_ols:.2f}')
```

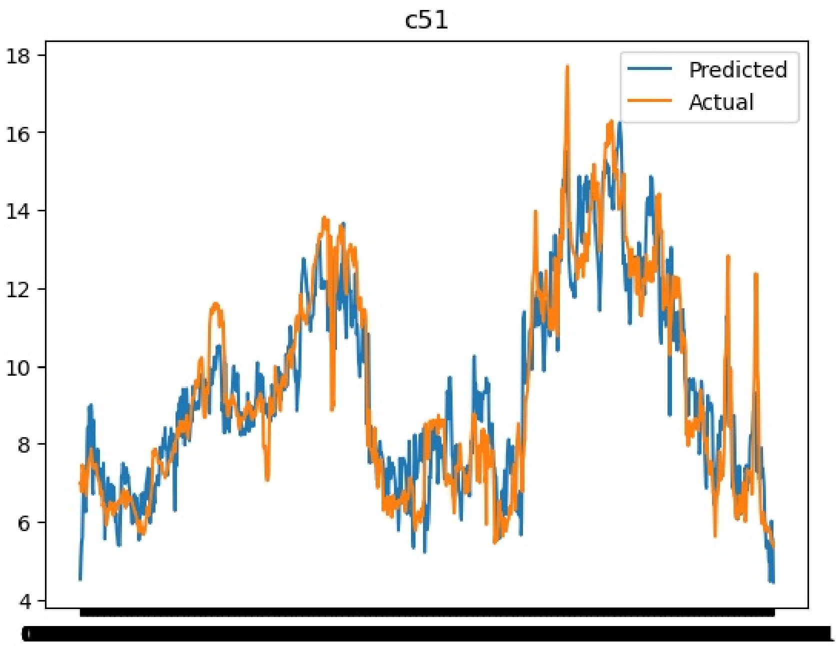
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c51   R-squared:                       0.838
Model:                            OLS   Adj. R-squared:                  0.831
Method:                 Least Squares   F-statistic:                     106.7
Date:                Sun, 12 Nov 2023   Prob (F-statistic):          2.84e-280
Time:                        19:58:16   Log-Likelihood:                -1235.2
No. Observations:                 820   AIC:                             2548.
Df Residuals:                     781   BIC:                             2732.
Df Model:                          38
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.4251      0.039    239.851      0.000       9.348       9.502
PC1            0.2105      0.006     34.837      0.000       0.199       0.222
PC2            0.1132      0.007     17.102      0.000       0.100       0.126
PC4            0.0693      0.009      7.374      0.000       0.051       0.088
PC5           -0.2934      0.011    -25.778      0.000      -0.316      -0.271
PC6           -0.0922      0.012     -7.887      0.000      -0.115      -0.069
PC7           -0.0406      0.012     -3.414      0.001      -0.064      -0.017
PC8            0.1798      0.015     12.109      0.000       0.151       0.209
PC10          -0.3190      0.018    -17.339      0.000      -0.355      -0.283
PC12           0.2858      0.020     14.225      0.000       0.246       0.325
PC13           0.0512      0.021      2.422      0.016       0.010       0.093
PC14          -0.2225      0.023     -9.546      0.000      -0.268      -0.177
PC15          -0.0782      0.024     -3.250      0.001      -0.125      -0.031
PC16           0.1917      0.025      7.592      0.000       0.142       0.241
PC17           0.1066      0.026      4.041      0.000       0.055       0.158
PC18           0.1744      0.026      6.665      0.000       0.123       0.226
PC20          -0.1127      0.030     -3.816      0.000      -0.171      -0.055
PC21           0.2856      0.031      9.082      0.000       0.224       0.347
PC22           0.1318      0.030      4.350      0.000       0.072       0.191
PC23           0.0889      0.032      2.805      0.005       0.027       0.151
PC25          -0.0700      0.034     -2.035      0.042      -0.138      -0.002
PC26           0.3495      0.036      9.577      0.000       0.278       0.421
PC27          -0.2143      0.036     -6.023      0.000      -0.284      -0.144
PC28           0.1787      0.038      4.688      0.000       0.104       0.254
PC29          -0.6333      0.039    -16.424      0.000      -0.709      -0.558
PC30           0.4723      0.040     11.791      0.000       0.394       0.551
PC31           0.1541      0.041      3.768      0.000       0.074       0.234
PC33           0.4115      0.044      9.280      0.000       0.324       0.499
PC34          -0.2620      0.044     -5.982      0.000      -0.348      -0.176
PC35          -0.2177      0.046     -4.706      0.000      -0.309      -0.127
PC36          -0.1589      0.046     -3.427      0.001      -0.250      -0.068
PC37          -0.1325      0.049     -2.721      0.007      -0.228      -0.037
PC40          -0.2969      0.053     -5.606      0.000      -0.401      -0.193
PC41           0.2634      0.057      4.650      0.000       0.152       0.375
PC44           0.2470      0.059      4.167      0.000       0.131       0.363
PC45          -0.1838      0.062     -2.963      0.003      -0.306      -0.062
PC46           0.1937      0.063      3.073      0.002       0.070       0.317
PC48          -0.1261      0.064     -1.980      0.048      -0.251      -0.001
PC52          -0.4365      0.070     -6.192      0.000      -0.575      -0.298
==============================================================================
Omnibus:                        7.358   Durbin-Watson:                   2.024
Prob(Omnibus):                  0.025   Jarque-Bera (JB):               10.266
Skew:                           0.021   Prob(JB):                      0.00590
Kurtosis:                       3.547   Cond. No.                         11.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
R-squared (OLS): 0.84
R-squared whole (OLS): 0.84
```

In [310…

```
plt.plot(df['c1'],c51pred)
plt.plot(df['c1'],new_df['c51'])
plt.legend(['Predicted','Actual'])
plt.title("c51")
```

Out[310]:

```
Text(0.5, 1.0, 'c51')
```



In [ ]:

```
vibdf['class51']=''
vibdf['c51pred']=c51pred
for i in range(len(c51pred)):
    if c51pred[i]>20:
        vibdf['class51'][i]=4
    elif (c51pred[i]<=20 and c51pred[i]>10):
        vibdf['class51'][i]=3
    elif c51pred[i]<=10 and c51pred[i]>5:
        vibdf['class51'][i]=2
    elif c51pred[i]<=5:
        vibdf['class51'][i]=1
```

In [312…

```
x=np.arange(0,1025,1)
fig1 = px.scatter(new_df, x, y='c51')
fig1.show()
```

In [313…

```
X=result_df[['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10',
       'PC11', 'PC12', 'PC13', 'PC14', 'PC16', 'PC17',
       'PC20', 'PC21', 'PC22', 'PC23', 'PC24', 'PC25', 'PC26', 'PC27',
       'PC29', 'PC30', 'PC31', 'PC33', 'PC34', 'PC35', 'PC36', 'PC37',
        'PC39', 'PC40', 'PC41', 'PC45',
       'PC47', 'PC48', 'PC52']]
y=new_df['c52']
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
X=sm.add_constant(X)
model1=sm.OLS(y_train,X_train).fit()
print(model1.summary())
y_pred_ols =model1.predict(X_test)
r2_ols = r2_score(y_test, y_pred_ols)
print(f'R-squared (OLS): {r2_ols:.2f}')
c52pred=model1.predict(X)
r3_ols = r2_score(new_df['c52'],c51pred)
print(f'R-squared whole (OLS): {r3_ols:.2f}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c52   R-squared:                       0.921
Model:                            OLS   Adj. R-squared:                  0.917
Method:                 Least Squares   F-statistic:                     233.2
Date:                Sun, 12 Nov 2023   Prob (F-statistic):               0.00
Time:                        19:58:23   Log-Likelihood:                -775.87
No. Observations:                 820   AIC:                             1632.
Df Residuals:                     780   BIC:                             1820.
Df Model:                          39
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.1183      0.022    405.860      0.000       9.074       9.162
PC1            0.1693      0.003     49.021      0.000       0.163       0.176
PC2            0.1088      0.004     28.782      0.000       0.101       0.116
PC3           -0.0826      0.005    -16.677      0.000      -0.092      -0.073
PC4            0.1642      0.005     30.594      0.000       0.154       0.175
PC5           -0.3330      0.007    -51.213      0.000      -0.346      -0.320
PC6           -0.0496      0.007     -7.417      0.000      -0.063      -0.036
PC7            0.1264      0.007     18.581      0.000       0.113       0.140
PC8            0.0672      0.008      7.926      0.000       0.051       0.084
PC9           -0.0297      0.009     -3.326      0.001      -0.047      -0.012
PC10          -0.1662      0.011    -15.815      0.000      -0.187      -0.146
PC11           0.0721      0.011      6.746      0.000       0.051       0.093
PC12           0.2581      0.011     22.464      0.000       0.236       0.281
PC13          -0.1305      0.012    -10.789      0.000      -0.154      -0.107
PC14          -0.0699      0.013     -5.252      0.000      -0.096      -0.044
PC16           0.0848      0.014      5.874      0.000       0.056       0.113
PC17           0.2442      0.015     16.178      0.000       0.215       0.274
PC20          -0.0720      0.017     -4.267      0.000      -0.105      -0.039
PC21           0.1288      0.018      7.168      0.000       0.094       0.164
PC22          -0.0583      0.017     -3.366      0.001      -0.092      -0.024
PC23           0.0533      0.018      2.941      0.003       0.018       0.089
PC24           0.1026      0.019      5.370      0.000       0.065       0.140
PC25           0.0527      0.020      2.683      0.007       0.014       0.091
PC26           0.0892      0.021      4.277      0.000       0.048       0.130
PC27           0.0593      0.020      2.914      0.004       0.019       0.099
PC29          -0.1368      0.022     -6.202      0.000      -0.180      -0.093
PC30           0.0602      0.023      2.629      0.009       0.015       0.105
PC31           0.1636      0.023      7.001      0.000       0.118       0.209
PC33           0.1353      0.025      5.337      0.000       0.086       0.185
PC34          -0.1227      0.025     -4.899      0.000      -0.172      -0.074
PC35          -0.1333      0.026     -5.040      0.000      -0.185      -0.081
PC36           0.0555      0.026      2.097      0.036       0.004       0.108
PC37          -0.0826      0.028     -2.971      0.003      -0.137      -0.028
PC39           0.1678      0.029      5.723      0.000       0.110       0.225
PC40          -0.0785      0.030     -2.590      0.010      -0.138      -0.019
PC41           0.1345      0.032      4.158      0.000       0.071       0.198
PC45          -0.2975      0.035     -8.382      0.000      -0.367      -0.228
PC47          -0.2211      0.036     -6.192      0.000      -0.291      -0.151
PC48          -0.0952      0.036     -2.616      0.009      -0.167      -0.024
PC52          -0.1844      0.040     -4.580      0.000      -0.263      -0.105
==============================================================================
Omnibus:                       18.310   Durbin-Watson:                   2.085
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               35.955
Skew:                           0.040   Prob(JB):                     1.56e-08
Kurtosis:                       4.023   Cond. No.                         11.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
R-squared (OLS): 0.90
R-squared whole (OLS): 0.44
```

In [ ]:

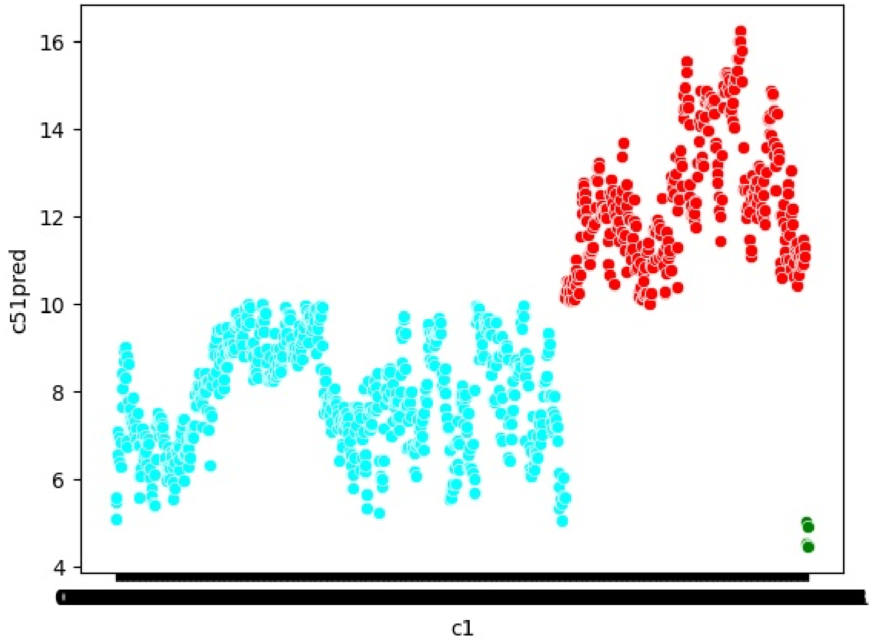
```
vibdf['class52']=''
vibdf['c52pred']=c52pred
for i in range(len(c52pred)):
    if c52pred[i]>20:
        vibdf['class52'][i]=4
    elif (c52pred[i]<=20 and c52pred[i]>10):
        vibdf['class52'][i]=3
    elif c52pred[i]<=10 and c52pred[i]>5:
        vibdf['class52'][i]=2
    elif c52pred[i]<=5:
        vibdf['class52'][i]=1

x=np.arange(0,1025,1)
fig1 = px.scatter(vibdf, x, y='c52pred', color='class52')
fig1.show()
```

In [315…

```
x=np.arange(0,1025,1)
fig1 = px.scatter(new_df, x, y='c54')
fig1.show()
```

In [316…
```
sns.scatterplot(data=vibdf,x=vibdf[vibdf['class51']==2]['c1'],y=vibdf[vibdf['class51']==2]['c51pred'],color='aq
sns.scatterplot(data=vibdf,x=vibdf[vibdf['class51']==3]['c1'],y=vibdf[vibdf['class51']==3]['c51pred'],color='re
sns.scatterplot(data=vibdf,x=vibdf[vibdf['class51']==1]['c1'],y=vibdf[vibdf['class51']==1]['c51pred'],color='gr
plt.figure(figsize=(20, 12))
```

Out[316]:
```
<Figure size 2000x1200 with 0 Axes>
```




```
<Figure size 2000x1200 with 0 Axes>
```

In [317…
```
vibdf['class51'].value_counts()
```

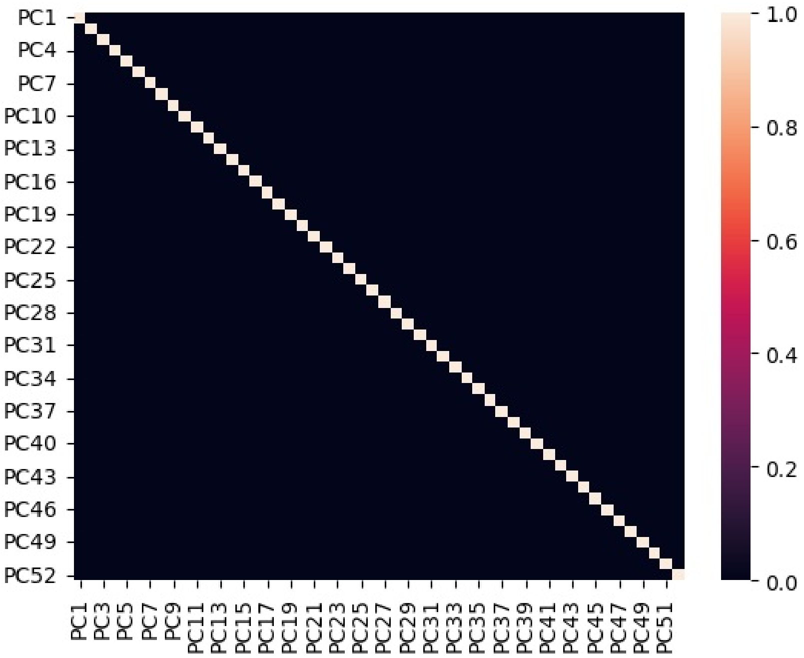
Out[317]:
```
2    664
3    356
1      5
Name: class51, dtype: int64
```

In [318…
```
df['c51'].min()
```

Out[318]:
```
5.349019912
```

In [319…
```
sns.heatmap(result_df.corr())
```

Out[319]:
```
<Axes: >
```



In [320…

```
result_df.columns
```

Out[320]:

```
Index(['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10',
       'PC11', 'PC12', 'PC13', 'PC14', 'PC15', 'PC16', 'PC17', 'PC18', 'PC19',
       'PC20', 'PC21', 'PC22', 'PC23', 'PC24', 'PC25', 'PC26', 'PC27', 'PC28',
       'PC29', 'PC30', 'PC31', 'PC32', 'PC33', 'PC34', 'PC35', 'PC36', 'PC37',
       'PC38', 'PC39', 'PC40', 'PC41', 'PC42', 'PC43', 'PC44', 'PC45', 'PC46',
       'PC47', 'PC48', 'PC49', 'PC50', 'PC51', 'PC52'],
      dtype='object')
```

In [321…

```
X=result_df[['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10',
       'PC11', 'PC12', 'PC13', 'PC14', 'PC15', 'PC16', 'PC17', 'PC19',
       'PC20', 'PC22', 'PC23', 'PC24', 'PC26', 'PC27', 'PC28',
       'PC29', 'PC30', 'PC31', 'PC32', 'PC34', 'PC36', 'PC37',
       'PC38', 'PC39', 'PC43', 'PC44', 'PC45', 'PC46',
       'PC47', 'PC48', 'PC50']]
y=new_df['c53']
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
X=sm.add_constant(X)
model1=sm.OLS(y_train,X_train).fit()
print(model1.summary())
y_pred_ols =model1.predict(X_test)
r2_ols = r2_score(y_test, y_pred_ols)
print(f'R-squared (OLS): {r2_ols:.2f}')
c53pred=model1.predict(X)
c54pred=model1.predict(X)
r3_ols = r2_score(new_df['c53'],c53pred)
r4_ols = r2_score(new_df['c54'],c54pred)
print(f'R-squared whole (OLS): {r3_ols:.2f}')
print(f'R-squared whole (OLS): {r4_ols:.2f}')
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                    c53   R-squared:                       0.954
Model:                            OLS   Adj. R-squared:                  0.951
Method:                 Least Squares   F-statistic:                     390.7
Date:                Sun, 12 Nov 2023   Prob (F-statistic):               0.00
Time:                        19:58:30   Log-Likelihood:                -1430.5
No. Observations:                 820   AIC:                             2945.
Df Residuals:                     778   BIC:                             3143.
Df Model:                          41                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.8235      0.050    196.639      0.000       9.725       9.922
PC1            0.7623      0.008     99.337      0.000       0.747       0.777
PC2            0.3414      0.008     40.524      0.000       0.325       0.358
PC3           -0.2093      0.011    -18.986      0.000      -0.231      -0.188
PC4            0.4685      0.012     39.257      0.000       0.445       0.492
PC5           -0.1143      0.014     -7.901      0.000      -0.143      -0.086
PC6           -0.3585      0.015    -24.117      0.000      -0.388      -0.329
PC7            0.1228      0.015      8.113      0.000       0.093       0.153
PC8            0.1141      0.019      6.042      0.000       0.077       0.151
PC9            0.3332      0.020     16.752      0.000       0.294       0.372
PC10          -0.2039      0.023     -8.716      0.000      -0.250      -0.158
PC11           0.0761      0.024      3.193      0.001       0.029       0.123
PC12           0.1230      0.026      4.817      0.000       0.073       0.173
PC13          -0.0709      0.027     -2.629      0.009      -0.124      -0.018
PC14          -0.3085      0.030    -10.412      0.000      -0.367      -0.250
PC15           0.3301      0.031     10.789      0.000       0.270       0.390
PC16           0.4026      0.032     12.532      0.000       0.339       0.466
PC17           0.3482      0.034     10.370      0.000       0.282       0.414
PC19           0.1700      0.036      4.695      0.000       0.099       0.241
PC20          -0.2386      0.038     -6.345      0.000      -0.312      -0.165
PC22           0.1002      0.039      2.599      0.010       0.025       0.176
PC23          -0.2972      0.040     -7.367      0.000      -0.376      -0.218
PC24           0.2258      0.043      5.306      0.000       0.142       0.309
PC26          -0.1635      0.046     -3.527      0.000      -0.254      -0.072
PC27           0.1732      0.045      3.828      0.000       0.084       0.262
PC28          -0.6000      0.048    -12.385      0.000      -0.695      -0.505
PC29           0.3328      0.049      6.781      0.000       0.236       0.429
PC30           0.3744      0.051      7.352      0.000       0.274       0.474
PC31           0.6597      0.052     12.725      0.000       0.558       0.762
PC32           0.2097      0.054      3.891      0.000       0.104       0.315
PC34          -0.3888      0.056     -6.981      0.000      -0.498      -0.280
PC36           0.2575      0.059      4.371      0.000       0.142       0.373
PC37          -0.4253      0.062     -6.877      0.000      -0.547      -0.304
PC38           0.1867      0.065      2.879      0.004       0.059       0.314
PC39           0.3631      0.065      5.574      0.000       0.235       0.491
PC43           0.3315      0.071      4.652      0.000       0.192       0.471
PC44          -0.4854      0.075     -6.440      0.000      -0.633      -0.337
PC45          -0.5222      0.079     -6.605      0.000      -0.677      -0.367
PC46          -0.3520      0.080     -4.384      0.000      -0.510      -0.194
PC47          -0.7336      0.079     -9.230      0.000      -0.890      -0.578
PC48          -0.4935      0.081     -6.089      0.000      -0.653      -0.334
PC50          -0.3993      0.088     -4.543      0.000      -0.572      -0.227
==============================================================================
Omnibus:                       65.985   Durbin-Watson:                   2.045
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              141.460
Skew:                           0.484   Prob(JB):                     1.92e-31
Kurtosis:                       4.789   Cond. No.                         11.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
R-squared (OLS): 0.94
R-squared whole (OLS): 0.95
R-squared whole (OLS): 0.86
```

In [ ]:

```
vibdf['class53']=''
vibdf['c53pred']=c53pred
for i in range(len(c53pred)):
    if c53pred[i]>20:
        vibdf['class53'][i]=4
    elif (c53pred[i]<=20 and c53pred[i]>10):
        vibdf['class53'][i]=3
    elif c53pred[i]<=10 and c53pred[i]>5:
        vibdf['class53'][i]=2
    elif c53pred[i]<=5:
        vibdf['class53'][i]=1

x=np.arange(0,1025,1)
fig1 = px.scatter(vibdf, x, y='c53pred', color='class53',color_discrete_sequence=['blue','green','orange','red'
fig1.show()
```

In [ ]:

```
X=vibdf['c53pred']
y=vibdf['c54']
#X=sm.add_constant(X)
```

```
model=sm.OLS(y,X).fit()
c54pred=model.predict(X)
model.summary()

vibdf['class54']=''
vibdf['c54pred']=c54pred
for i in range(len(c54pred)):
    if c54pred[i]>20:
        vibdf['class54'][i]=4
    elif (c54pred[i]<=20 and c54pred[i]>10):
        vibdf['class54'][i]=3
    elif c54pred[i]<=10 and c54pred[i]>5:
        vibdf['class54'][i]=2
    elif c54pred[i]<=5:
        vibdf['class54'][i]=1

x=np.arange(0,1025,1)
fig1 = px.scatter(vibdf, x, y='c54pred', color='class54',color_discrete_sequence=['blue','green','orange','red'
fig1.show()
```

In [324…

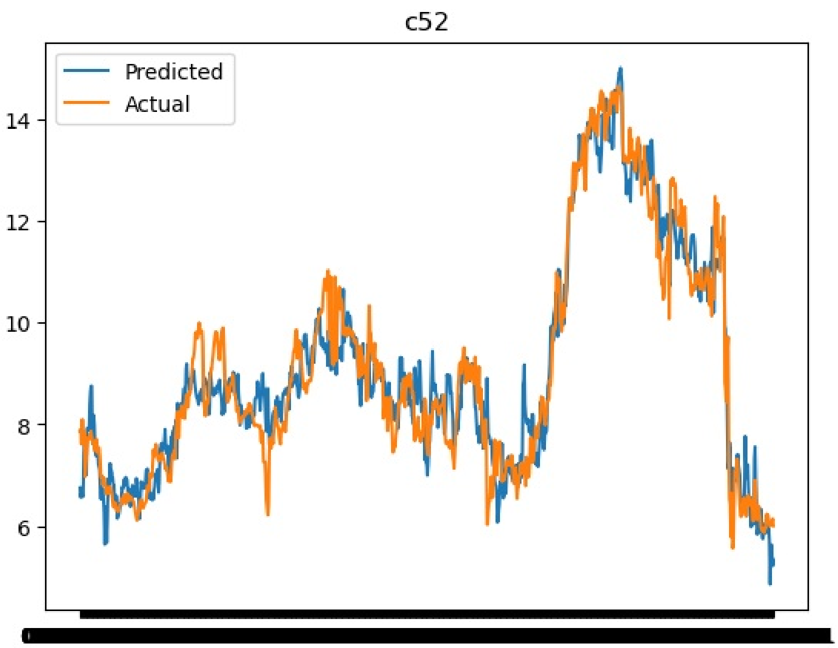
```
x=np.arange(0,1025,1)
fig1 = px.scatter(new_df, x, y='c53')
fig1.show()
```

In [325…

```
plt.plot(df['c1'],c52pred)
plt.plot(df['c1'],new_df['c52'])
plt.legend(['Predicted','Actual'])
plt.title("c52")
```

Out[325]:

```
Text(0.5, 1.0, 'c52')
```

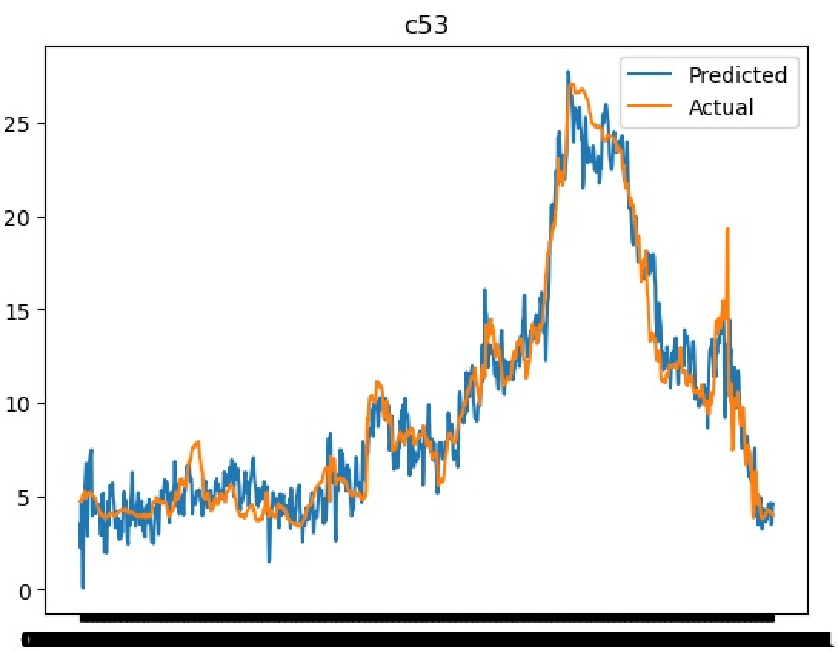


In [326…

```
plt.plot(df['c1'],c53pred)
plt.plot(df['c1'],new_df['c53'])
plt.legend(['Predicted','Actual'])
plt.title("c53")
```

Out[326]:

```
Text(0.5, 1.0, 'c53')
```

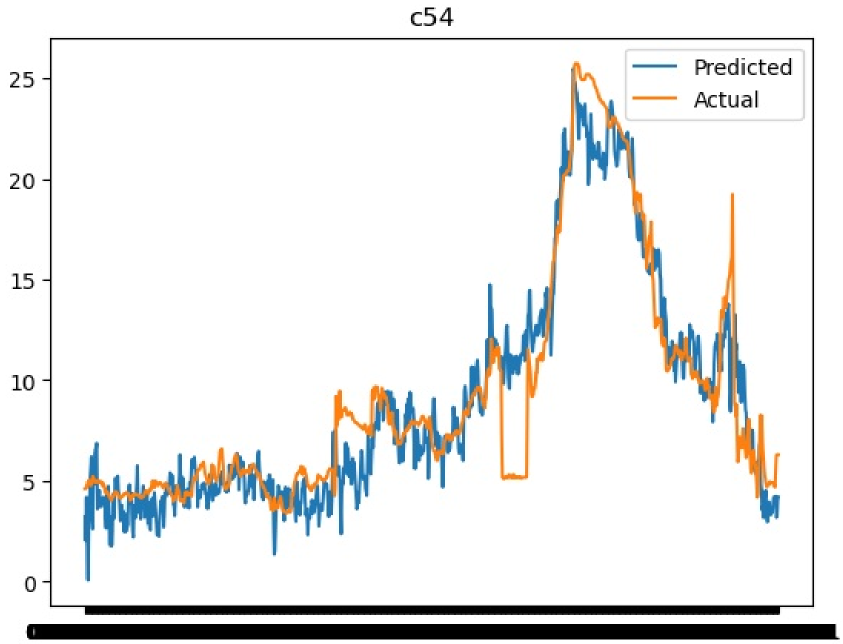


In [327…

```
plt.plot(df['c1'],c54pred)
```

```
plt.plot(df['c1'],new_df['c54'])
plt.legend(['Predicted','Actual'])
plt.title("c54")
```

Out[327]: Text(0.5, 1.0, 'c54')



In [328…

```
X=result_df[result_df.columns]
y=new_df['c54']
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
model1=sm.OLS(y_train,X_train).fit()
print(model1.summary())
y_pred_ols =model1.predict(X_test)
r2_ols = r2_score(y_test, y_pred_ols)
print(f'R-squared (OLS): {r2_ols:.2f}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c54   R-squared:                       0.954
Model:                            OLS   Adj. R-squared:                  0.951
Method:                 Least Squares   F-statistic:                     309.0
Date:                Sun, 12 Nov 2023   Prob (F-statistic):               0.00
Time:                        19:58:51   Log-Likelihood:                -1343.0
No. Observations:                 820   AIC:                             2792.
Df Residuals:                     767   BIC:                             3042.
Df Model:                          52
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.1961      0.045    203.123      0.000       9.107       9.285
PC1            0.6262      0.007     89.827      0.000       0.613       0.640
PC2            0.3147      0.008     41.234      0.000       0.300       0.330
PC3           -0.1798      0.010    -17.991      0.000      -0.199      -0.160
PC4            0.4471      0.011     41.222      0.000       0.426       0.468
PC5           -0.2279      0.013    -17.361      0.000      -0.254      -0.202
PC6           -0.3940      0.013    -29.224      0.000      -0.420      -0.368
PC7            0.2223      0.014     16.201      0.000       0.195       0.249
PC8            0.2015      0.017     11.772      0.000       0.168       0.235
PC9            0.1351      0.018      7.496      0.000       0.100       0.170
PC10          -0.4447      0.021    -20.970      0.000      -0.486      -0.403
PC11           0.0943      0.022      4.373      0.000       0.052       0.137
PC12           0.1515      0.023      6.546      0.000       0.106       0.197
PC13          -0.0896      0.024     -3.669      0.000      -0.138      -0.042
PC14          -0.1796      0.027     -6.692      0.000      -0.232      -0.127
PC15           0.2314      0.028      8.346      0.000       0.177       0.286
PC16           0.4552      0.029     15.630      0.000       0.398       0.512
PC17           0.3754      0.030     12.329      0.000       0.316       0.435
PC18          -0.2216      0.030     -7.347      0.000      -0.281      -0.162
PC19           0.1699      0.033      5.179      0.000       0.106       0.234
PC20          -0.4821      0.034    -14.149      0.000      -0.549      -0.415
PC21           0.2522      0.036      6.956      0.000       0.181       0.323
PC22           0.1685      0.035      4.821      0.000       0.100       0.237
PC23          -0.1962      0.037     -5.364      0.000      -0.268      -0.124
PC24           0.2652      0.039      6.874      0.000       0.189       0.341
PC25          -0.2723      0.040     -6.872      0.000      -0.350      -0.195
PC26           0.0952      0.042      2.261      0.024       0.013       0.178
PC27          -0.0259      0.041     -0.631      0.528      -0.106       0.055
PC28          -0.5523      0.044    -12.557      0.000      -0.639      -0.466
PC29           0.0342      0.045      0.769      0.442      -0.053       0.122
PC30           0.5102      0.046     11.047      0.000       0.420       0.601
PC31           0.5485      0.047     11.632      0.000       0.456       0.641
PC32           0.2994      0.049      6.108      0.000       0.203       0.396
PC33           0.0735      0.051      1.438      0.151      -0.027       0.174
PC34          -0.4480      0.051     -8.862      0.000      -0.547      -0.349
PC35          -0.0944      0.053     -1.767      0.078      -0.199       0.010
PC36           0.1643      0.053      3.073      0.002       0.059       0.269
PC37          -0.5369      0.056     -9.554      0.000      -0.647      -0.427
PC38           0.2058      0.059      3.501      0.000       0.090       0.321
PC39           0.5618      0.059      9.502      0.000       0.446       0.678
PC40          -0.2448      0.061     -3.997      0.000      -0.365      -0.125
PC41          -0.0999      0.065     -1.530      0.126      -0.228       0.028
PC42           0.0637      0.065      0.974      0.330      -0.065       0.192
PC43           0.2536      0.065      3.922      0.000       0.127       0.381
PC44          -0.2391      0.068     -3.497      0.000      -0.373      -0.105
PC45          -0.3205      0.072     -4.470      0.000      -0.461      -0.180
PC46          -0.1776      0.073     -2.440      0.015      -0.320      -0.035
PC47          -0.9725      0.072    -13.503      0.000      -1.114      -0.831
PC48          -0.3644      0.074     -4.955      0.000      -0.509      -0.220
PC49           0.0557      0.076      0.732      0.464      -0.094       0.205
PC50          -0.2501      0.080     -3.142      0.002      -0.406      -0.094
PC51           0.1746      0.079      2.207      0.028       0.019       0.330
PC52          -0.2457      0.081     -3.015      0.003      -0.406      -0.086
==============================================================================
Omnibus:                      104.620   Durbin-Watson:                   2.082
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              374.952
Skew:                           0.570   Prob(JB):                     3.80e-82
Kurtosis:                       6.110   Cond. No.                         12.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
R-squared (OLS): 0.95
```

In [329…

```
X=result_df[['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10',
       'PC11', 'PC12', 'PC13', 'PC14', 'PC15', 'PC16', 'PC17', 'PC18', 'PC19',
       'PC20', 'PC21', 'PC22', 'PC23', 'PC24', 'PC25', 'PC26', 'PC27',
       'PC29', 'PC30', 'PC31', 'PC32', 'PC33', 'PC34', 'PC35', 'PC36', 'PC37',
       'PC38', 'PC39', 'PC40', 'PC41', 'PC42', 'PC43', 'PC44']]
y=dfprep['c51']
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
X_train = sm.add_constant(X_train)
X_test = sm.add_constant(X_test)
```

```
model1=sm.OLS(y_train,X_train).fit()
print(model1.summary())
y_pred_ols =model1.predict(X_test)
r2_ols = r2_score(y_test, y_pred_ols)
print(f'R-squared (OLS): {r2_ols:.2f}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c51   R-squared:                       0.867
Model:                            OLS   Adj. R-squared:                  0.860
Method:                 Least Squares   F-statistic:                     117.8
Date:                Sun, 12 Nov 2023   Prob (F-statistic):          7.53e-308
Time:                        19:58:51   Log-Likelihood:                -1125.9
No. Observations:                 820   AIC:                             2340.
Df Residuals:                     776   BIC:                             2547.
Df Model:                          43
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.4143      0.034    273.113      0.000       9.347       9.482
PC1            0.2071      0.005     39.034      0.000       0.197       0.217
PC2            0.1162      0.006     19.988      0.000       0.105       0.128
PC3            0.0165      0.008      2.162      0.031       0.002       0.031
PC4            0.0801      0.008      9.687      0.000       0.064       0.096
PC5           -0.2964      0.010    -29.628      0.000      -0.316      -0.277
PC6           -0.0822      0.010     -8.003      0.000      -0.102      -0.062
PC7           -0.0425      0.010     -4.067      0.000      -0.063      -0.022
PC8            0.1786      0.013     13.688      0.000       0.153       0.204
PC9           -0.0094      0.014     -0.687      0.492      -0.036       0.018
PC10          -0.3163      0.016    -19.561      0.000      -0.348      -0.285
PC11           0.0105      0.016      0.637      0.524      -0.022       0.043
PC12           0.2979      0.018     16.881      0.000       0.263       0.333
PC13           0.0341      0.019      1.834      0.067      -0.002       0.071
PC14          -0.1957      0.020     -9.565      0.000      -0.236      -0.156
PC15          -0.0742      0.021     -3.515      0.000      -0.116      -0.033
PC16           0.1727      0.022      7.784      0.000       0.129       0.216
PC17           0.1211      0.023      5.217      0.000       0.076       0.167
PC18           0.1795      0.023      7.817      0.000       0.134       0.225
PC19           0.0325      0.025      1.299      0.194      -0.017       0.082
PC20          -0.1380      0.026     -5.312      0.000      -0.189      -0.087
PC21           0.2470      0.028      8.936      0.000       0.193       0.301
PC22           0.1440      0.027      5.404      0.000       0.092       0.196
PC23           0.0956      0.028      3.433      0.001       0.041       0.150
PC24           0.0352      0.029      1.198      0.231      -0.022       0.093
PC25          -0.0897      0.030     -2.969      0.003      -0.149      -0.030
PC26           0.3117      0.032      9.717      0.000       0.249       0.375
PC27          -0.2127      0.031     -6.805      0.000      -0.274      -0.151
PC29          -0.6136      0.034    -18.089      0.000      -0.680      -0.547
PC30           0.4566      0.035     12.982      0.000       0.388       0.526
PC31           0.1556      0.036      4.331      0.000       0.085       0.226
PC32           0.1049      0.037      2.817      0.005       0.032       0.178
PC33           0.3748      0.039      9.613      0.000       0.298       0.451
PC34          -0.1751      0.038     -4.550      0.000      -0.251      -0.100
PC35          -0.1400      0.041     -3.443      0.001      -0.220      -0.060
PC36          -0.0978      0.041     -2.401      0.017      -0.178      -0.018
PC37          -0.1791      0.043     -4.185      0.000      -0.263      -0.095
PC38          -0.0785      0.045     -1.754      0.080      -0.166       0.009
PC39          -0.0518      0.045     -1.151      0.250      -0.140       0.037
PC40          -0.3298      0.047     -7.069      0.000      -0.421      -0.238
PC41           0.1889      0.050      3.799      0.000       0.091       0.287
PC42           0.0939      0.050      1.888      0.059      -0.004       0.192
PC43          -0.0529      0.049     -1.074      0.283      -0.150       0.044
PC44           0.2373      0.052      4.556      0.000       0.135       0.340
==============================================================================
Omnibus:                        0.328   Durbin-Watson:                   2.041
Prob(Omnibus):                  0.849   Jarque-Bera (JB):                0.423
Skew:                           0.022   Prob(JB):                        0.809
Kurtosis:                       2.898   Cond. No.                         10.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
R-squared (OLS): 0.85
```

In [330…

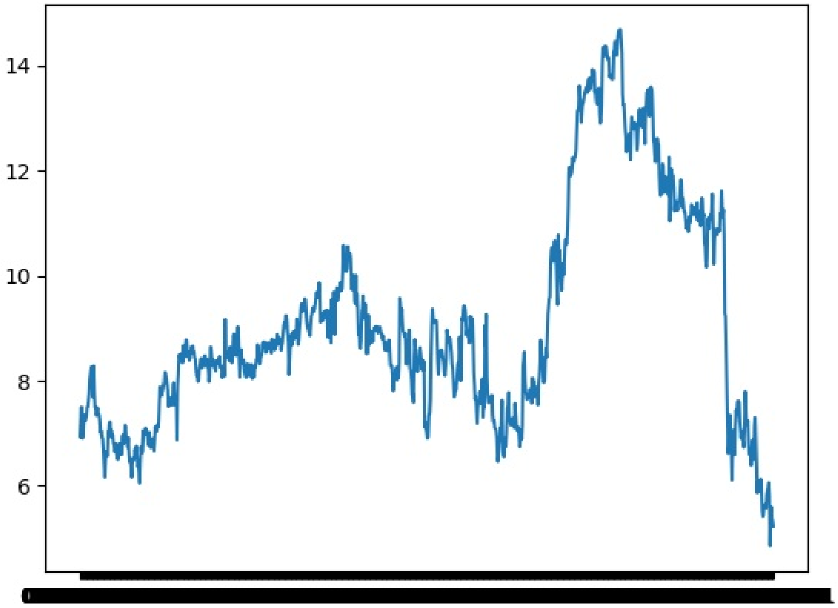
```
y=dfprep['c52']
X=result_df[['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9', 'PC10',
       'PC11', 'PC12', 'PC13', 'PC14', 'PC15', 'PC16', 'PC17', 'PC18', 'PC19',
       'PC20', 'PC21', 'PC22', 'PC23', 'PC24', 'PC26', 'PC27', 'PC28',
       'PC29', 'PC30', 'PC31', 'PC32', 'PC33', 'PC34', 'PC35', 'PC36', 'PC37',
       'PC38', 'PC39', 'PC41', 'PC42', 'PC43', 'PC44']]
X=sm.add_constant(X)
model2=sm.OLS(y,X).fit()
print(model2.summary())
model2.predict()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c52   R-squared:                       0.926
Model:                            OLS   Adj. R-squared:                  0.923
Method:                 Least Squares   F-statistic:                     292.5
Date:                Sun, 12 Nov 2023   Prob (F-statistic):               0.00
Time:                        19:58:51   Log-Likelihood:                -916.46
No. Observations:                1025   AIC:                             1919.
Df Residuals:                     982   BIC:                             2131.
Df Model:                          42
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.1058      0.019    482.289      0.000       9.069       9.143
PC1            0.1691      0.003     57.384      0.000       0.163       0.175
PC2            0.1051      0.003     32.949      0.000       0.099       0.111
PC3           -0.0808      0.004    -19.321      0.000      -0.089      -0.073
PC4            0.1573      0.005     34.396      0.000       0.148       0.166
PC5           -0.3244      0.005    -59.032      0.000      -0.335      -0.314
PC6           -0.0468      0.006     -8.236      0.000      -0.058      -0.036
PC7            0.1189      0.006     19.956      0.000       0.107       0.131
PC8            0.0594      0.007      8.503      0.000       0.046       0.073
PC9           -0.0288      0.008     -3.750      0.000      -0.044      -0.014
PC10          -0.1707      0.009    -19.209      0.000      -0.188      -0.153
PC11           0.0761      0.009      8.276      0.000       0.058       0.094
PC12           0.2551      0.010     25.843      0.000       0.236       0.274
PC13          -0.1348      0.010    -13.254      0.000      -0.155      -0.115
PC14          -0.0722      0.011     -6.473      0.000      -0.094      -0.050
PC15          -0.0137      0.012     -1.180      0.238      -0.036       0.009
PC16           0.1044      0.012      8.527      0.000       0.080       0.128
PC17           0.2410      0.013     19.276      0.000       0.216       0.265
PC18          -0.0056      0.013     -0.439      0.661      -0.031       0.019
PC19           0.0176      0.014      1.283      0.200      -0.009       0.045
PC20          -0.0680      0.014     -4.822      0.000      -0.096      -0.040
PC21           0.1438      0.015      9.720      0.000       0.115       0.173
PC22          -0.0446      0.015     -2.974      0.003      -0.074      -0.015
PC23           0.0381      0.015      2.507      0.012       0.008       0.068
PC24           0.1212      0.016      7.658      0.000       0.090       0.152
PC26           0.0660      0.017      3.900      0.000       0.033       0.099
PC27           0.0475      0.017      2.758      0.006       0.014       0.081
PC28           0.0198      0.018      1.095      0.274      -0.016       0.055
PC29          -0.1326      0.019     -7.141      0.000      -0.169      -0.096
PC30           0.0992      0.019      5.115      0.000       0.061       0.137
PC31           0.1439      0.020      7.251      0.000       0.105       0.183
PC32           0.0549      0.020      2.729      0.006       0.015       0.094
PC33           0.1348      0.021      6.300      0.000       0.093       0.177
PC34          -0.1356      0.021     -6.318      0.000      -0.178      -0.093
PC35          -0.1151      0.022     -5.278      0.000      -0.158      -0.072
PC36           0.0359      0.022      1.597      0.111      -0.008       0.080
PC37          -0.0441      0.024     -1.875      0.061      -0.090       0.002
PC38          -0.0269      0.025     -1.092      0.275      -0.075       0.021
PC39           0.1642      0.025      6.589      0.000       0.115       0.213
PC41           0.1367      0.027      5.132      0.000       0.084       0.189
PC42           0.0377      0.027      1.393      0.164      -0.015       0.091
PC43          -0.0553      0.027     -2.023      0.043      -0.109      -0.002
PC44           0.0256      0.028      0.899      0.369      -0.030       0.082
==============================================================================
Omnibus:                        0.819   Durbin-Watson:                   0.207
Prob(Omnibus):                  0.664   Jarque-Bera (JB):                0.712
Skew:                           0.055   Prob(JB):                        0.700
Kurtosis:                       3.066   Cond. No.                         9.67
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Out[330]:
```
array([6.93730502, 7.21922084, 7.49545764, ..., 5.39005204, 5.33192822,
       5.22793534])
```

In [331…
```
plt.plot(df['c1'],model2.predict())
```

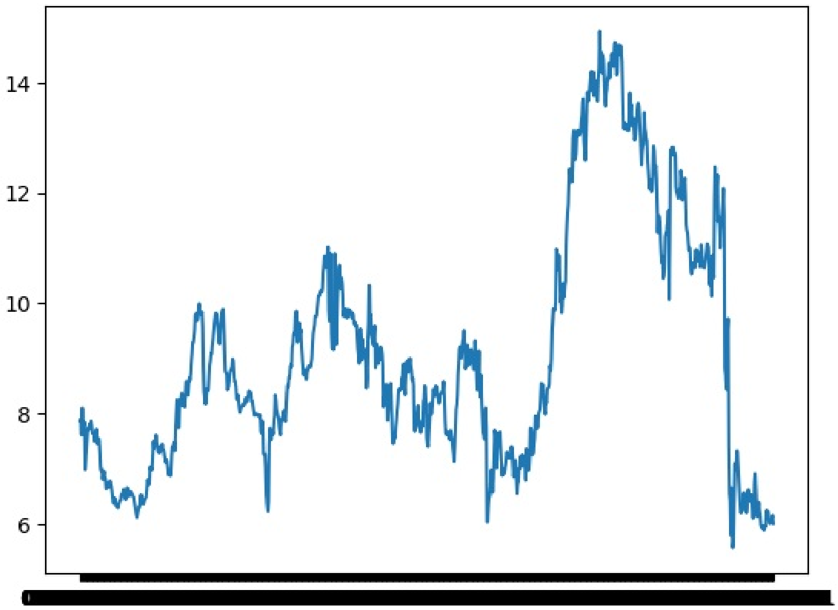
Out[331]:
```
[<matplotlib.lines.Line2D at 0x2739a7b4b50>]
```



In [332…

```
plt.plot(df['c1'],df['c52'])
```

Out[332]: [<matplotlib.lines.Line2D at 0x27396f99930>]



In [333…

```
vibdf.corr()
```

```
C:\Users\amish\AppData\Local\Temp\ipykernel_388\595330355.py:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to Fals
e. Select only valid columns or specify the value of numeric_only to silence this warning.
```

Out[333]:

	c51	c52	c53	c54	c51pred	c52pred	c53pred	c54pred
c51	1.000000	0.759471	0.568096	0.658248	0.916199	0.730097	0.571478	0.571478
c52	0.759471	1.000000	0.723674	0.794431	0.770851	0.957851	0.718423	0.718423
c53	0.568096	0.723674	1.000000	0.962310	0.622454	0.731255	0.975388	0.975388
c54	0.658248	0.794431	0.962310	1.000000	0.703702	0.799071	0.942281	0.942281
c51pred	0.916199	0.770851	0.622454	0.703702	1.000000	0.804668	0.638033	0.638033
c52pred	0.730097	0.957851	0.731255	0.799071	0.804668	1.000000	0.749379	0.749379
c53pred	0.571478	0.718423	0.975388	0.942281	0.638033	0.749379	1.000000	1.000000
c54pred	0.571478	0.718423	0.975388	0.942281	0.638033	0.749379	1.000000	1.000000

In [334…

```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

def perform_pca(data_frame, variance_retained=0.99):
    standardized_data = StandardScaler().fit_transform(data_frame)
    pca = PCA(n_components=variance_retained)
    principal_components = pca.fit_transform(standardized_data)

    # Convert to DataFrame for better visualization
    principal_df = pd.DataFrame(data=principal_components, columns=[f'PC{i}' for i in range(1, pca.n_components_

    # Percentage of variance retained
    retained_variance = sum(pca.explained_variance_ratio_)
    print(f"Variance retained: {retained_variance * 100:.2f}%")

    return principal_df
result_vibdf = perform_pca(vibdf.drop('c1',axis=1))
print(result_vibdf)
```

```
Variance retained: 99.33%
           PC1       PC2       PC3       PC4       PC5       PC6       PC7  \
0    -3.887056 -1.149170 -1.291516  1.155980  0.291000  0.606667  0.626017
1    -3.479862 -0.079185 -0.484267  0.202015 -0.649961 -0.159601  0.739326
2    -3.249369 -0.260376 -0.344603  0.243006 -0.714224 -0.068387  0.456532
3    -3.362155  0.099010 -0.454747  0.273735 -0.550565 -0.089691  0.736554
4    -3.485641  0.349533 -0.387235  0.207433 -0.447730  0.022009  0.495451
...        ...       ...       ...       ...       ...       ...       ...
1020 -4.249603 -1.652636 -0.728669  0.947458 -0.122769  1.023355  0.173100
1021 -3.705541 -0.517140  0.039934  0.048337 -1.032923  0.220480  0.046719
1022 -3.773578 -0.620959  0.042727  0.038837 -1.107105  0.199024  0.172808
1023 -3.710835 -0.699788  0.104894  0.065373 -1.145056  0.241260  0.200074
1024 -4.325129 -1.801644 -0.763701  0.956883 -0.254704  0.946511  0.236946

           PC8
0    -0.249137
1    -0.183268
2    -0.090104
3    -0.195473
4    -0.203939
...        ...
1020 -0.123197
1021 -0.031335
1022 -0.040180
1023 -0.020536
1024 -0.130718

[1025 rows x 8 columns]
```

In [335…

```
X=dfprep[['c51','c52','c53']]
y=dfprep['c54']
X=sm.add_constant(X)
modelvib=sm.OLS(y,X).fit()
print(modelvib.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c54   R-squared:                       0.959
Model:                            OLS   Adj. R-squared:                  0.959
Method:                 Least Squares   F-statistic:                     8054.
Date:                Sun, 12 Nov 2023   Prob (F-statistic):               0.00
Time:                        19:59:03   Log-Likelihood:                -1601.8
No. Observations:                1025   AIC:                             3212.
Df Residuals:                    1021   BIC:                             3231.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -3.3400      0.176    -18.966      0.000      -3.686      -2.994
c51            0.2028      0.023      8.988      0.000       0.159       0.247
c52            0.3782      0.032     11.675      0.000       0.315       0.442
c53            0.7296      0.008     87.042      0.000       0.713       0.746
==============================================================================
Omnibus:                      286.451   Durbin-Watson:                   0.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              804.230
Skew:                          -1.416   Prob(JB):                    2.31e-175
Kurtosis:                       6.289   Cond. No.                         85.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [336…

```
X=new_df[[ 'c21', 'c22', 'c27', 'c30', 'c35', 'c36', 'c37', 'c44', 'c45', 'c60', 'c62', 'c63', 'c73', 'c82', 'c
```

In [337…

```
y=new_df['c51']
```

In [338…

```
X=sm.add_constant(X)
modeln1=sm.OLS(y,X).fit()
print(modeln1.summary())
modeln1.predict()
```

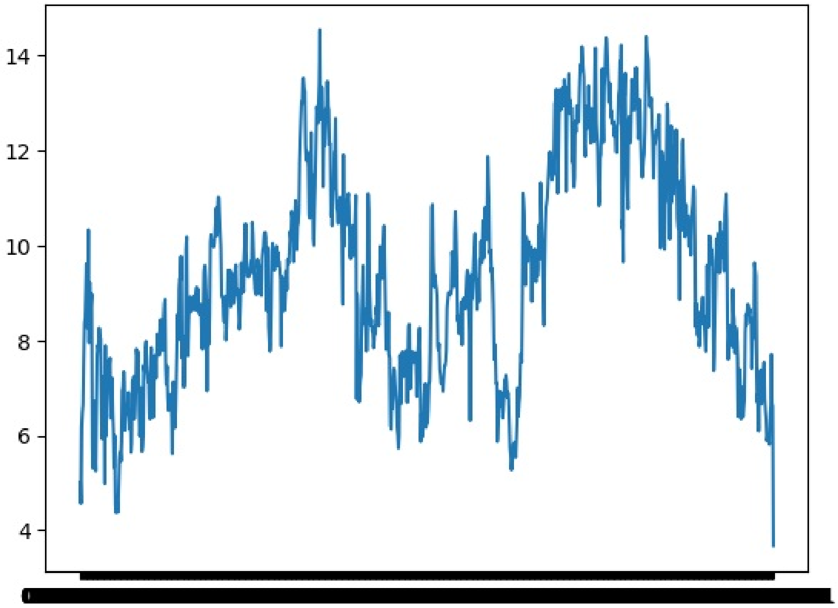
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c51   R-squared:                       0.660
Model:                            OLS   Adj. R-squared:                  0.651
Method:                 Least Squares   F-statistic:                     80.76
Date:                Sun, 12 Nov 2023   Prob (F-statistic):          1.30e-214
Time:                        19:59:03   Log-Likelihood:                -1929.9
No. Observations:                1025   AIC:                             3910.
Df Residuals:                    1000   BIC:                             4033.
Df Model:                          24
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c21           -0.6537      0.051    -12.874      0.000      -0.753      -0.554
c22            0.2913      0.045      6.546      0.000       0.204       0.379
c27           -1.6824      0.339     -4.961      0.000      -2.348      -1.017
c30            2.0594      0.493      4.176      0.000       1.092       3.027
c35           -4.0208      1.469     -2.737      0.006      -6.903      -1.138
c36          -13.5777      4.737     -2.867      0.004     -22.872      -4.283
c37           -1.1622      0.455     -2.556      0.011      -2.055      -0.270
c44            0.6078      0.091      6.704      0.000       0.430       0.786
c45            0.5065      0.071      7.107      0.000       0.367       0.646
c60           -0.4682      0.056     -8.338      0.000      -0.578      -0.358
c62           -1.0012      0.211     -4.741      0.000      -1.416      -0.587
c63            1.8575      0.294      6.317      0.000       1.280       2.435
c73           -0.9445      0.223     -4.237      0.000      -1.382      -0.507
c82            0.1042      0.020      5.153      0.000       0.064       0.144
c110           1.2416      0.241      5.153      0.000       0.769       1.714
c133          89.2495     12.707      7.023      0.000      64.313     114.186
c147          -0.0188      0.002     -9.239      0.000      -0.023      -0.015
c156          -0.5825      0.106     -5.478      0.000      -0.791      -0.374
c160          -0.0100      0.001     -7.591      0.000      -0.013      -0.007
c161           0.0148      0.002      9.571      0.000       0.012       0.018
c163          -0.0089      0.003     -2.873      0.004      -0.015      -0.003
c179          -2.5056      0.711     -3.522      0.000      -3.902      -1.109
c190           0.8212      0.134      6.115      0.000       0.558       1.085
c207          -0.0008   8.92e-05     -8.644      0.000      -0.001      -0.001
c218          -0.8699      0.232     -3.755      0.000      -1.324      -0.415
c238           0.2507      0.025      9.914      0.000       0.201       0.300
==============================================================================
Omnibus:                       29.057   Durbin-Watson:                   0.269
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               31.312
Skew:                           0.390   Prob(JB):                     1.59e-07
Kurtosis:                       3.352   Cond. No.                     1.01e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 6.39e-21. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Out[338]: array([5.00957926, 4.56849632, 6.16287139, ..., 6.67630032, 6.61352269,
       3.67595062])

In [339…

```
plt.plot(df['c1'],modeln1.predict())
```

Out[339]: [<matplotlib.lines.Line2D at 0x2738940f7f0>]



In [340…

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

X_train_ols = sm.add_constant(X_train)
X_test_ols = sm.add_constant(X_test)

ols_model = sm.OLS(y_train, X_train_ols).fit()


print(ols_model.summary())
y_pred_ols = ols_model.predict(X_test_ols)
mse_ols = mean_squared_error(y_test, y_pred_ols)
r2_ols = r2_score(y_test, y_pred_ols)

print(f'\nMean Squared Error (OLS): {mse_ols:.2f}')
print(f'R-squared (OLS): {r2_ols:.2f}')
```

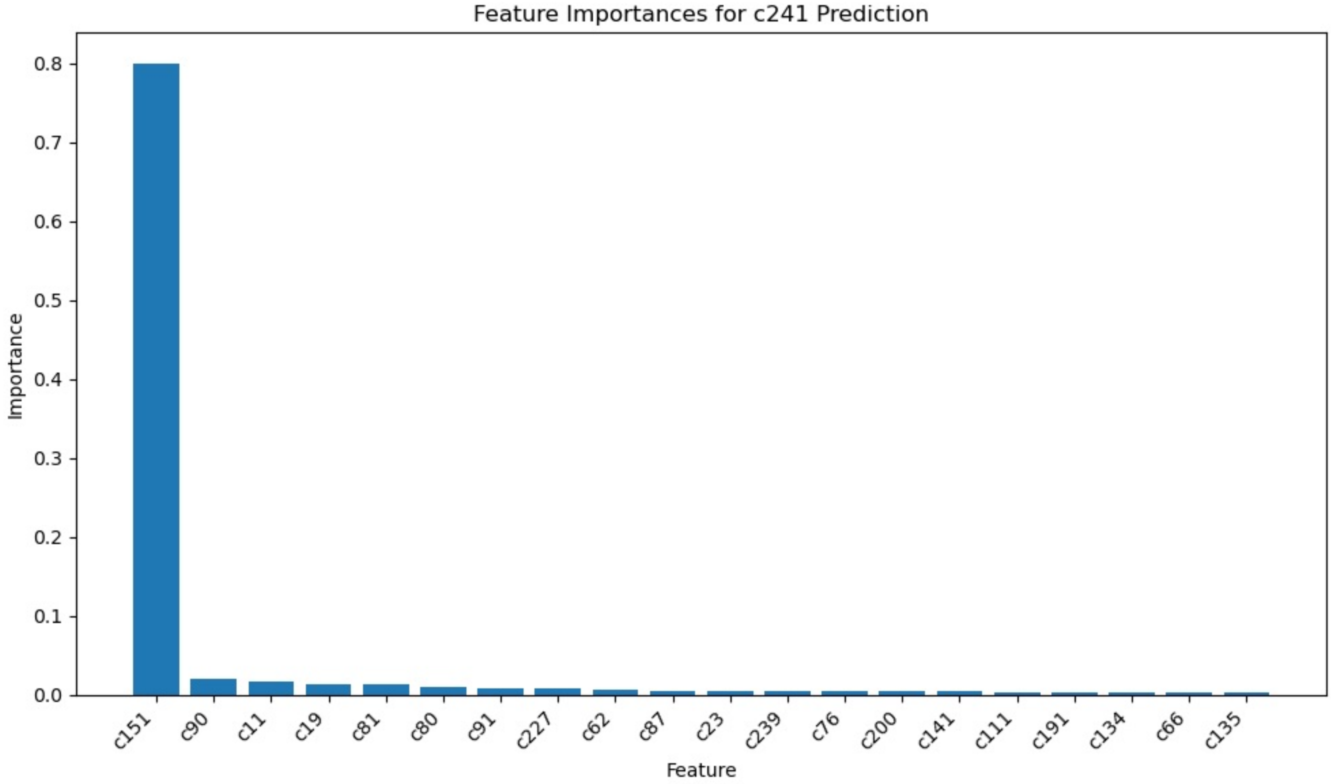
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c51   R-squared:                       0.657
Model:                            OLS   Adj. R-squared:                  0.647
Method:                 Least Squares   F-statistic:                     63.48
Date:                Sun, 12 Nov 2023   Prob (F-statistic):         1.94e-166
Time:                        19:59:09   Log-Likelihood:                -1543.9
No. Observations:                 820   AIC:                             3138.
Df Residuals:                     795   BIC:                             3255.
Df Model:                          24
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c21           -0.6418      0.057    -11.285      0.000      -0.753      -0.530
c22            0.3126      0.050      6.309      0.000       0.215       0.410
c27           -1.8894      0.380     -4.975      0.000      -2.635      -1.144
c30            2.2453      0.549      4.090      0.000       1.168       3.323
c35           -4.1364      1.620     -2.553      0.011      -7.317      -0.956
c36          -13.4338      5.638     -2.383      0.017     -24.501      -2.366
c37           -1.1969      0.512     -2.338      0.020      -2.202      -0.192
c44            0.6626      0.101      6.561      0.000       0.464       0.861
c45            0.6046      0.081      7.443      0.000       0.445       0.764
c60           -0.4945      0.062     -7.945      0.000      -0.617      -0.372
c62           -1.1662      0.237     -4.916      0.000      -1.632      -0.701
c63            2.0955      0.331      6.325      0.000       1.445       2.746
c73           -0.9091      0.256     -3.545      0.000      -1.413      -0.406
c82            0.1273      0.022      5.684      0.000       0.083       0.171
c110           1.5171      0.267      5.684      0.000       0.993       2.041
c133          78.3838     14.358      5.459      0.000      50.200     106.567
c147          -0.0167      0.002     -7.153      0.000      -0.021      -0.012
c156          -0.6310      0.124     -5.101      0.000      -0.874      -0.388
c160          -0.0098      0.001     -6.946      0.000      -0.013      -0.007
c161           0.0136      0.002      8.151      0.000       0.010       0.017
c163          -0.0094      0.003     -2.763      0.006      -0.016      -0.003
c179          -3.3303      0.790     -4.215      0.000      -4.881      -1.779
c190           0.7916      0.150      5.266      0.000       0.497       1.087
c207          -0.0007   9.92e-05     -7.476      0.000      -0.001      -0.001
c218          -0.7182      0.262     -2.746      0.006      -1.232      -0.205
c238           0.2485      0.028      8.991      0.000       0.194       0.303
==============================================================================
Omnibus:                       26.307   Durbin-Watson:                   1.949
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               28.553
Skew:                           0.413   Prob(JB):                     6.31e-07
Kurtosis:                       3.394   Cond. No.                     1.01e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 5.11e-21. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

Mean Squared Error (OLS): 2.63
R-squared (OLS): 0.66
```

In [341…

```python
####3

from sklearn.ensemble import RandomForestRegressor
import pandas as pd
import matplotlib.pyplot as plt
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
X=new_df[['c3','c4','c5','c6', 'c7', 'c8','c9','c10','c11', 'c12', 'c13', 'c14', 'c15',
'c16', 'c17', 'c18', 'c19', 'c20', 'c21', 'c22', 'c23', 'c24', 'c25', 'c34',
'c35', 'c36', 'c37', 'c38', 'c40', 'c41', 'c42', 'c43', 'c44', 'c45', 'c46',
'c47', 'c48', 'c49', 'c50', 'c51', 'c52', 'c53', 'c54', 'c55', 'c56', 'c57',
'c58', 'c59', 'c60', 'c61', 'c62', 'c63', 'c64', 'c65', 'c66', 'c67', 'c68',
'c69', 'c70', 'c71', 'c72', 'c73', 'c74', 'c75', 'c76', 'c77', 'c78', 'c79',
'c80', 'c81', 'c83', 'c84', 'c85', 'c86', 'c87', 'c88', 'c89', 'c90', 'c91',
'c92', 'c93', 'c94', 'c95', 'c96', 'c97', 'c98', 'c99', 'c100', 'c101', 'c102',
'c103', 'c104', 'c105', 'c106', 'c107', 'c108', 'c109', 'c111', 'c112', 'c113',
'c114', 'c115', 'c116', 'c117', 'c118', 'c119', 'c120', 'c121', 'c122', 'c123',
'c124', 'c133', 'c134', 'c135', 'c136', 'c137', 'c138', 'c140', 'c141', 'c144',
'c145', 'c146', 'c147', 'c148', 'c149', 'c125', 'c126', 'c127', 'c128', 'c129',
'c130', 'c131', 'c132', 'c150', 'c151', 'c152', 'c153', 'c154', 'c159', 'c164',
'c165', 'c166', 'c167', 'c168', 'c169', 'c170', 'c171', 'c172', 'c173', 'c174',
'c175', 'c176', 'c177', 'c178', 'c179', 'c180', 'c181', 'c182', 'c183', 'c184',
'c185', 'c186', 'c187', 'c191', 'c192', 'c193', 'c194', 'c195', 'c196', 'c197',
'c198', 'c200', 'c201', 'c203', 'c205', 'c207', 'c208', 'c209', 'c210', 'c211',
'c212', 'c213', 'c214', 'c215', 'c216', 'c217',
'c218','c219','c220','c221','c222','c223','c224','c225','c227','c228','c230','c231','c232','c233','c234','c235'
'c238','c239']]
y=new_df['c241']
rf_model.fit(X, y)
feature_importances = rf_model.feature_importances_
summary_table = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importances})
summary_table = summary_table.sort_values(by='Importance', ascending=False)
print("Feature Importances:")

important_summary_table=summary_table.head(20)
```

```
print(important_summary_table)

plt.figure(figsize=(10, 6))
plt.bar(important_summary_table['Feature'], important_summary_table['Importance'])
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importances for c241 Prediction')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
Feature Importances:
    Feature  Importance
134    c151    0.798811
77      c90    0.019817
8       c11    0.017025
16      c19    0.013610
69      c81    0.012728
68      c80    0.010425
78      c91    0.008089
194    c227    0.007187
50      c62    0.005662
74      c87    0.004779
20      c23    0.004554
205    c239    0.004496
64      c76    0.004455
171    c200    0.004037
118    c141    0.003875
97     c111    0.002603
163    c191    0.002427
112    c134    0.002052
54      c66    0.002013
113    c135    0.001961
```



In [ ]:
```
from sklearn.ensemble import RandomForestRegressor
import pandas as pd
import matplotlib.pyplot as plt

rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
# making ML model from Operating and Controllable Parameters
X=result_df[result_df.columns]
y=new_df['c241']


rf_model.fit(X, y)


feature_importances = rf_model.feature_importances_


summary_table = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importances})


summary_table = summary_table.sort_values(by='Importance', ascending=False)
print("Feature Importances:")
print(summary_table)
```

```
important_summary_table=summary_table.head(20)
print(important_summary_table)
plt.figure(figsize=(10, 6))
plt.bar(important_summary_table['Feature'], important_summary_table['Importance'])
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importances for c241 Prediction')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Question 2 Control Parameters ML Model

In [343…

```
control=new_df[contro_para]
```

In [344…

```
control
```

Out[344]:

	c26	c27	c28	c29	c30	c31	c32	c33	c39	c139	c142	c14
0	493.796764	104.553871	41.187601	290.965340	14.379552	70.678458	48.679005	-69.203403	0.410096	13.599070	48.457544	37.00000
1	493.661889	104.513206	41.580752	290.621190	14.315323	71.707542	48.057417	-69.414081	0.409465	13.167193	48.794365	37.00000
2	495.644947	104.502457	40.744572	290.621190	14.566180	72.736626	47.320586	-69.645378	0.410025	12.611031	49.131185	37.00000
3	494.354041	104.452871	40.288181	292.676229	14.605181	72.736626	47.980460	-69.452794	0.410889	14.832367	49.131185	37.00000
4	492.051373	104.488584	41.266692	289.017462	14.548926	76.621067	48.217299	-69.344057	0.411034	15.943873	51.185354	37.00000
...	...	...	...	...	...	...	...	...	...	...	...	.
1020	497.999661	104.960764	35.711850	293.988342	14.450879	78.082852	51.380085	-68.004586	0.550094	12.339225	50.383117	32.00016
1021	497.139686	104.960249	35.658364	293.975309	14.452064	79.062697	51.606934	-67.893767	0.550332	13.007149	51.008752	32.00016
1022	497.557435	104.963291	35.666902	294.001376	14.437922	79.930899	52.030272	-67.727372	0.550160	13.366582	51.452608	32.00016
1023	497.669483	104.958298	35.685112	294.049924	14.449497	80.262698	52.246631	-67.620510	0.550423	13.588131	51.645568	32.00016
1024	498.180745	104.955014	35.738588	294.275404	14.453477	80.376672	52.382273	-67.546656	0.550027	13.969828	51.737968	32.00016

1025 rows × 20 columns

In [345…

```
fig4=px.imshow(control.corr(),text_auto=True)
fig4.show()
```

In [346…

```
result3 = pd.DataFrame(calculate_vif(control))
print(result3)
```

```
   Variable           VIF
0     const  1.725440e+06
1       c26  1.194677e+01
2       c27  2.299547e+00
3       c28  2.092221e+01
4       c29  1.042599e+01
5       c30  3.734582e+00
6       c31  1.126588e+01
7       c32  5.695735e+01
8       c33  6.044384e+01
9       c39  2.058152e+00
10     c139  1.948884e+00
11     c142  2.073712e+01
12     c143  1.395678e+01
13     c155  5.923843e+00
14     c156  1.322119e+00
15     c157  1.616818e+00
16     c158  2.668356e+00
17     c160  2.462057e+00
18     c161  3.439316e+00
19     c162  2.215065e+00
20     c163  1.744254e+00
```

In [347…

```
VIF_rem_col=[]
for i in range(len(result3)):
    if result['VIF'][i]<20:
        VIF_rem_col.append(result3['Variable'][i])
print(len(VIF_rem_col))
print(VIF_rem_col)
```

```
6
['const', 'c139', 'c143', 'c161', 'c162', 'c163']
```

In [348…

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X = control
y = vibdf['c51']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
feature_importances = pd.Series(rf_model.feature_importances_, index=X.columns)
sorted_features = feature_importances.sort_values(ascending=False)
print("Feature Importance (Descending Order):")
print(sorted_features.head(5))
y_train_pred = rf_model.predict(X_train)
r2_train = r2_score(y_train, y_train_pred)
print(f'R-squared (Train): {r2_train:.2f}')
y_test_pred = rf_model.predict(X_test)
r2_test = r2_score(y_test, y_test_pred)
print(f'R-squared (Test): {r2_test:.2f}')
c51pred_c=rf_model.predict(X)
```

```
Feature Importance (Descending Order):
c155    0.482266
c161    0.103223
c39     0.102509
c158    0.056922
c28     0.043443
dtype: float64
R-squared (Train): 0.99
R-squared (Test): 0.92
```

In [349…

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X = control
y = vibdf['c52']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
feature_importances = pd.Series(rf_model.feature_importances_, index=X.columns)
sorted_features = feature_importances.sort_values(ascending=False)
print("Feature Importance (Descending Order):")
print(sorted_features.head(5))
y_train_pred = rf_model.predict(X_train)
r2_train = r2_score(y_train, y_train_pred)
print(f'R-squared (Train): {r2_train:.2f}')
y_test_pred = rf_model.predict(X_test)
r2_test = r2_score(y_test, y_test_pred)
print(f'R-squared (Test): {r2_test:.2f}')
```

```
Feature Importance (Descending Order):
c155    0.457650
c161    0.161503
c158    0.119824
c39     0.058013
c143    0.034318
dtype: float64
R-squared (Train): 0.99
R-squared (Test): 0.97
```

In [350…

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X = control
y = vibdf['c53']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
feature_importances = pd.Series(rf_model.feature_importances_, index=X.columns)
sorted_features = feature_importances.sort_values(ascending=False)
print("Feature Importance (Descending Order):")
print(sorted_features.head(5))
y_train_pred = rf_model.predict(X_train)
r2_train = r2_score(y_train, y_train_pred)
print(f'R-squared (Train): {r2_train:.2f}')
y_test_pred = rf_model.predict(X_test)
r2_test = r2_score(y_test, y_test_pred)
print(f'R-squared (Test): {r2_test:.2f}')
```

```
Feature Importance (Descending Order):
c155    0.745427
c157    0.124698
c143    0.029893
c31     0.013184
c27     0.011819
dtype: float64
R-squared (Train): 1.00
R-squared (Test): 0.98
```

In [351…

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
X = control
y = vibdf['c54']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
feature_importances = pd.Series(rf_model.feature_importances_, index=X.columns)
sorted_features = feature_importances.sort_values(ascending=False)
print("Feature Importance (Descending Order):")
print(sorted_features.head(5))
y_train_pred = rf_model.predict(X_train)
r2_train = r2_score(y_train, y_train_pred)
print(f'R-squared (Train): {r2_train:.2f}')
y_test_pred = rf_model.predict(X_test)
r2_test = r2_score(y_test, y_test_pred)
print(f'R-squared (Test): {r2_test:.2f}')
```

```
Feature Importance (Descending Order):
c155    0.773661
c161    0.079449
c143    0.055850
c39     0.014335
c27     0.011694
dtype: float64
R-squared (Train): 1.00
R-squared (Test): 0.98
```

In [352…

```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
X = control
y = vibdf['c51']
X = sm.add_constant(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
ols_model = sm.OLS(y_train, X_train)
ols_result = ols_model.fit()
print(ols_result.summary())
y_train_pred = ols_result.predict(X_train)
r2_train = r2_score(y_train, y_train_pred)
print(f'R-squared (Train): {r2_train:.2f}')
y_test_pred = ols_result.predict(X_test)
r2_test = r2_score(y_test, y_test_pred)
print(f'R-squared (Test): {r2_test:.2f}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    c51   R-squared:                       0.540
Model:                            OLS   Adj. R-squared:                  0.529
Method:                 Least Squares   F-statistic:                     46.93
Date:                Sun, 12 Nov 2023   Prob (F-statistic):          5.13e-120
Time:                        19:59:40   Log-Likelihood:                -1664.2
No. Observations:                 820   AIC:                             3370.
Df Residuals:                     799   BIC:                             3469.
Df Model:                          20
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       -227.5775    100.899     -2.255      0.024    -425.637     -29.518
c26            0.1706      0.089      1.925      0.055      -0.003       0.345
c27           -1.5018      0.459     -3.274      0.001      -2.402      -0.601
c28           -0.0349      0.062     -0.567      0.571      -0.156       0.086
c29           -0.1911      0.089     -2.147      0.032      -0.366      -0.016
c30            1.2827      0.863      1.486      0.138      -0.412       2.978
c31            0.0088      0.056      0.158      0.875      -0.101       0.119
c32            1.7948      0.404      4.438      0.000       1.001       2.589
c33           -3.6000      0.941     -3.824      0.000      -5.448      -1.752
c39           14.4852      1.911      7.580      0.000      10.734      18.236
c139          -0.1501      0.059     -2.542      0.011      -0.266      -0.034
c142           0.0941      0.109      0.862      0.389      -0.120       0.309
c143           0.0671      0.049      1.362      0.174      -0.030       0.164
c155           0.0786      0.019      4.082      0.000       0.041       0.116
c156          -0.4247      0.146     -2.911      0.004      -0.711      -0.138
c157          -0.1271      0.011    -11.953      0.000      -0.148      -0.106
c158           0.1599      0.027      5.947      0.000       0.107       0.213
c160          -0.0136      0.002     -8.234      0.000      -0.017      -0.010
c161           0.0173      0.002      9.193      0.000       0.014       0.021
c162          -0.0004      0.002     -0.187      0.852      -0.005       0.004
c163          -0.0016      0.004     -0.401      0.689      -0.009       0.006
==============================================================================
Omnibus:                       12.282   Durbin-Watson:                   1.947
Prob(Omnibus):                  0.002   Jarque-Bera (JB):               12.430
Skew:                           0.282   Prob(JB):                      0.00200
Kurtosis:                       3.213   Cond. No.                     1.66e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.66e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
R-squared (Train): 0.54
R-squared (Test): 0.50
```

In [353…

```
px.imshow(new_df[['c155','c157','c143','c39','c27']].corr(),text_auto=True)
```

In [ ]:

```
vibdf
```

In [355…

```
X=new_df[['c155','c157','c143','c31']]
```

```
y=vibdf['c53']
X = sm.add_constant(X)
model=sm.OLS(y,X).fit()
print(model.summary())
coefficients = model.params
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                    c53   R-squared:                       0.771
Model:                            OLS   Adj. R-squared:                  0.770
Method:                 Least Squares   F-statistic:                     857.2
Date:                Sun, 12 Nov 2023   Prob (F-statistic):               0.00
Time:                        19:59:40   Log-Likelihood:                -2602.5
No. Observations:                1025   AIC:                             5215.
Df Residuals:                    1020   BIC:                             5240.
Df Model:                           4                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -9.7474      2.189     -4.453      0.000     -14.043      -5.452
c155           0.6708      0.015     45.355      0.000       0.642       0.700
c157          -0.1456      0.015     -9.829      0.000      -0.175      -0.117
c143           0.2545      0.020     12.523      0.000       0.215       0.294
c31            0.0536      0.026      2.043      0.041       0.002       0.105
==============================================================================
Omnibus:                       34.815   Durbin-Watson:                   0.074
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               36.907
Skew:                           0.449   Prob(JB):                     9.68e-09
Kurtosis:                       2.759   Cond. No.                     2.23e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.23e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [356…

```
coefc53=coefficients

coefc53=pd.DataFrame(coefc53)
```

In [357…

```
coefc53=coefc53.transpose()
```

In [ ]:

```
vibdf['c53diff']=vibdf['c53pred']-10
for i in range(len(vibdf)):
    a=vibdf['c53diff'][i]
    if vibdf['c53diff'][i]>10 :
        print(vibdf['c53pred'][i])
        print(" CRITICAL ALERT!!! System Activated")
        delc155=(a-10)/coefc53['c155']
        delc157=(a-10)/coefc53['c157']
        delc143=(a-10)/coefc53['c143']
        delc31=(a-10)/coefc53['c31']
        print('Reduce the parameters')
#         print('c155 by',delc155)
#         print('c157 by',delc157)
#         print('c143 by',delc143)
#         print('c31 by', delc31)
        Tunedf=pd.DataFrame([delc155,delc157,delc143,delc31])
        print(Tunedf)
    elif vibdf['c53diff'][i]>0 :
        print(vibdf['c53pred'][i])
        print(" High ALERT!!! System Activated")
        delc155=(a)/coefc53['c155']
        delc157=(a)/coefc53['c157']
        delc143=(a)/coefc53['c143']
        delc31=(a)/coefc53['c31']
        print('Reduce the parameters')
#         print('c155 by',delc155)
#         print('c157 by',delc157)
#         print('c143 by',delc143)
#         print('c31 by', delc31)
        Tunedf=pd.DataFrame([delc155,delc157,delc143,delc31])
        print(Tunedf)
```

In [ ]:

```
df.info()
```

In [ ]:

```
new_df.info()
```