1. Loading the data and preparing for analysis:

- We first install the necessary packages- Numpy, Pandas, Matplotlib and Seaborn.
- Next, we load the data set The first 5 rows of the data can be printed out using the **df.head()** function.
- We can also use **df.shape** and **df.columns** to find out the shape of the data frame as well as its column names to help prepare for analysis

2. Analyzing the types of data:

- The **df.info()** function will return information about the data frame
- **df.dtypes** and **df.describe()** are also used to further analyze the data

3. Finding the missing features:

- We run a loop through the data frame to find any null values in the data frame using the **isnull()** function. On doing this, we can find that the **'MW2'** column has missing data.
- We replace the missing data in this using the **median** of the column. The median is preferred over the mean, since the mean can be skewed if there are a lot of outliers present in the data.

Converting categorical columns into numeric columns:

- To do this, we use the **Label Encoder** to convert the columns with categorical data into numeric data.

4. Finding the outliers:

- We first plot a **box plot** of the first few columns to visualize the outliers. On doing this, we can see that there are quite a few outliers in each column, so we will need to remove these
- To remove the outliers, we use the Interquartile Range. We first calculate **Q1** and **Q3** in the data and use that to find the **IQR**. The IQR is the range in which the **central 50%** of the data lies. We then make an **upper** and **lower bound** and filter the data accordingly.
- The outliers are replaced with the **median**.
- We can plot another box plot to verify that the outliers have been handled effectively.

5. Finding the correlated variables:

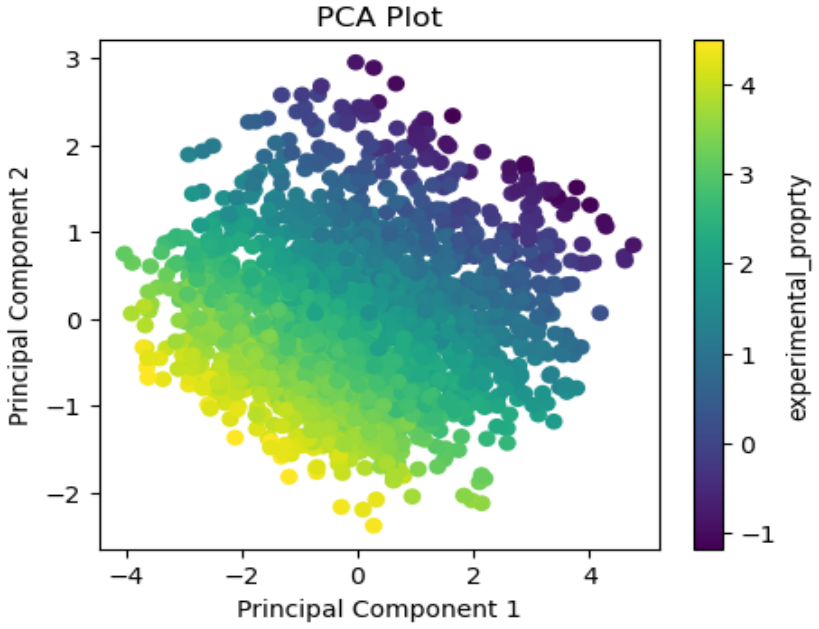
- We calculate the correlation matrix for the data frame, by identifying the pairs of values with a correlation higher than **0.994**.
- This threshold was found through **trial and error**. We set a high threshold to filter out the less significant values to visualize the data more effectively. This is due to the data set being very large.

6. Finding the highly correlated variables with the target:

- Here, the target variable is **'experimental_proprty'**. We set a threshold value of **3.15** which was found through **trial and error** to find the values that are highly correlated with the target.
- On running this, we can see that there are three columns- **FilterItLogS**, **nAromAtom**, and **SLogP** that are highly correlated with **'experimental_proprty'**.
- We can visualize their correlation by plotting a **Heat Map**.

7. Performing PCA in 2D:

- Before we perform PCA, we need to **standardize** the data. This can be done using the **StandardScaler**. The standard scalar works by removing the mean and scaling the data to unit variance.
- We can compute the PCA mathematically by finding the **eigenvalues** and **eigenvectors** and using that to find the top 2 principal components. This can then be used to plot the PCA.
- This can also be done using the PCA package in Python. We need to import the **PCA** package from **sklearn.decomposition** and then we can perform PCA.
- On observing the PCA plot, we can see that the **data is spread out with some clustering in the center**. We can observe a relatively **even distribution** among the principal components. This suggests that many of the data points are **similar**.
- We can also print out the **explained variance ratio** for some more insights. Observing the result of the explained variance ratio, we can see that the first principal component is approximately **62%** of the variance, and the second component is around **18.8%**. Together, these two components are approximately **80.8%** of the total variance in the dataset.
  This indicates that the dimensionality reduction by PCA **retains a substantial amount** of the original data's variability. So, it is reasonable to believe that the **PCA has effectively summarized the data with a reduced number of components**.



Analyzing the results from the PCA plot:

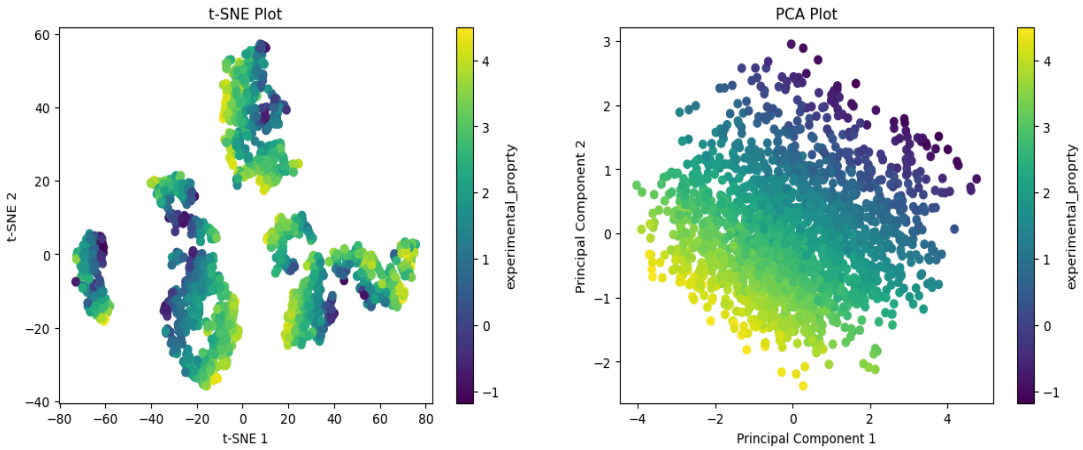
The PCA plot shows a gradual correlation between the principal components and the experimental property. Principal Component 1 likely captures the variance driven by **SLogP** (positively correlated) and **FilterItLogS** (negatively correlated), with higher experimental property values (yellow points) corresponding to higher **SLogP** and lower **FilterItLogS**. Principal Component 2 might reflect **nAromAtom**, as points higher on the y-axis are associated with a moderate positive correlation with experimental property.

Overall, the plot reveals a **continuous relationship** rather than distinct clusters.

Bonus Questions:

1. **Using t-SNE to plot the non-linear dimensionality reduction:**

- We can use the **TSNE** package to plot the t-SNE plot and compare the results with PCA.
- On observing the results, we can see that the t-SNE plot captures more of the **distinct clusters**. This is ideal for visualizing non-linear relationships.
- In the PCA plot, we can see that the data points are more **uniformly spread** out without any clear groupings. This is because PCA prioritizes **maximizing variance among the principal components**.
- The t-SNE plot on the other hand is useful for identifying **clusters** or **patterns** in the data.

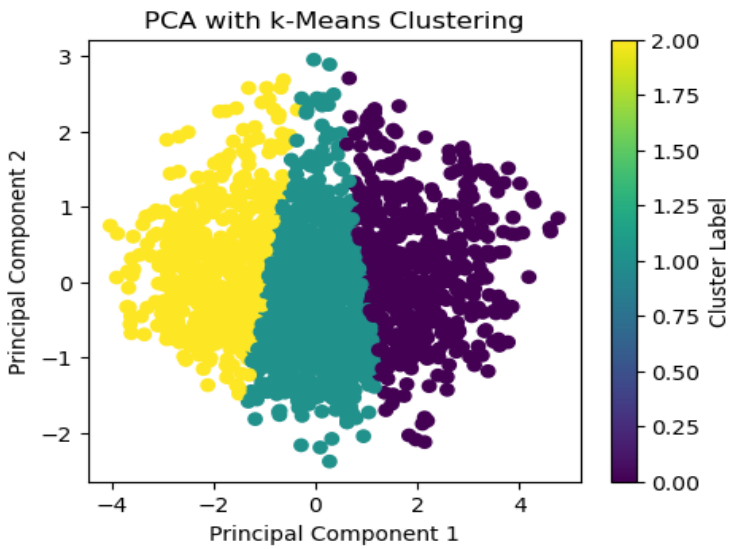


In the t-SNE plot, we see a few distinct clusters of data points. These clusters indicate that certain combinations of the features- **FilterItLogS**, **SLogP**, and **nAromAtom** lead to distinct experimental property behaviors.

The color gradient from purple to yellow reflects the values of the experimental property, and we can observe that certain clusters are **predominantly one color**. This implies that these clusters correspond to **similar experimental property values**.

2. **Surprising relations in the dataset:**

- The PCA plot with k-means clustering reveals three distinct clusters, **sharply separated** along the first principal component (PC1). This suggests that PC1, influenced by features like **SLogP** and **FilterItLogS**, plays a key role in differentiating the groups. PC2 adds some variation within the clusters.
- The clear boundaries imply that there are **hidden structures** in the data that k-means can effectively capture, despite the continuous nature observed in the previous PCA plot.


PCA with k-Means Clustering
Principal Component 2
Principal Component 1
Cluster Label
3
2
1
0
−1
−2
−4
−2
0
2
4
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00