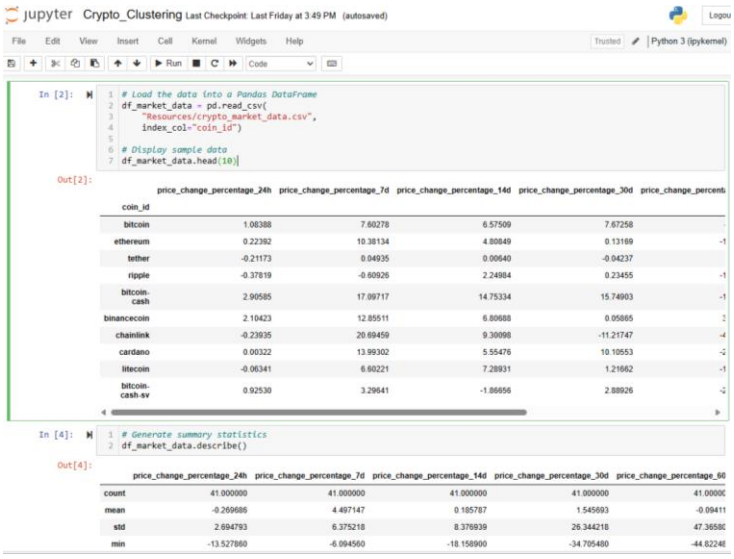

jupyter Crypto_Clustering Last Checkpoint: Last Friday at 3:49 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3 (ipykernel)

Run Code

In [2]:

```
# Load the data into a Pandas DataFrame
df_market_data = pd.read_csv(
    "Resources/crypto_market_data.csv",
    index_col="coin_id")

# Display sample data
df_market_data.head(10)
```

Out[2]:

coin_id	price_change_percentage_24h	price_change_percentage_7d	price_change_percentage_14d	price_change_percentage_30d	price_change_percentage_60d
bitcoin	1.08388	7.60278	6.57509	7.67258	
ethereum	0.22392	10.38134	4.80849	0.13169	-1
tether	-0.21173	0.04935	0.00640	-0.04237	
ripple	-0.37819	-0.60926	2.24984	0.23455	-1
bitcoin-cash	2.90585	17.09717	14.75334	15.74903	-1
binancecoin	2.10423	12.85511	6.80688	0.05865	
chainlink	-0.23935	20.69459	9.30098	-11.21747	-4
cardano	0.00322	13.99302	5.55476	10.10553	-2
litecoin	-0.06341	6.60221	7.28931	1.21662	-1
bitcoin-cash-sv	0.92530	3.29641	-1.86656	2.88926	-2

In [4]:

```
# Generate summary statistics
df_market_data.describe()
```

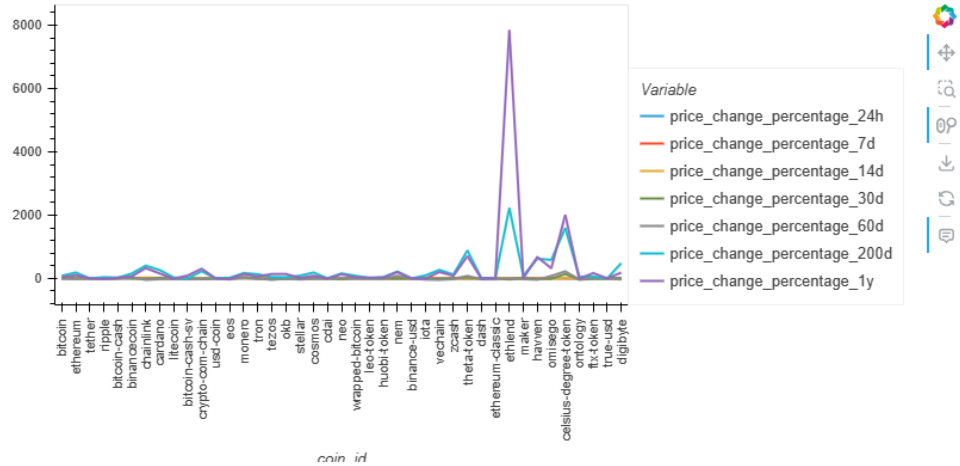
Out[4]:

	price_change_percentage_24h	price_change_percentage_7d	price_change_percentage_14d	price_change_percentage_30d	price_change_percentage_60
count	41.000000	41.000000	41.000000	41.000000	41.00000
mean	-0.269686	4.497147	0.185787	1.545693	-0.09411
std	2.694793	6.375218	8.376939	26.344218	47.36580
min	-13.527860	-6.094560	-18.158900	-34.705480	-44.82248

In [7]:

```
# Plot your data to see what's in your DataFrame
df_market_data.hvplot.line(
    width=800,
    height=400,
    rot=90
)
```

Out[7]:



In [9]:

```
df_market_data_scaled = pd.DataFrame(
    market_data_scaled,
    columns=['price_change_percentage_24h', 'price_change_percentage_7d',
       'price_change_percentage_14d', 'price_change_percentage_30d',
       'price_change_percentage_60d', 'price_change_percentage_200d',
       'price_change_percentage_1y']
)

# Copy the crypto names from the original data
df_market_data_scaled['coin_id'] = df_market_data.index

# Set the coinid column as index
df_market_data_scaled = df_market_data_scaled.set_index('coin_id')

# Display sample data
df_market_data_scaled.head()
```

Out[9]:

coin_id	price_change_percentage_24h	price_change_percentage_7d	price_change_percentage_14d	price_change_percentage_30d	price_change_percentage
bitcoin	0.508529	0.493193	0.772200	0.235460	-0.06
ethereum	0.185446	0.934445	0.558692	-0.054341	-0.27
tether	0.021774	-0.706337	-0.021680	-0.061030	0.00
ripple	-0.040764	-0.810928	0.249458	-0.050388	-0.37
bitcoin-cash	1.193036	2.000959	1.760610	0.545842	-0.29

```
df_elbow = pd.DataFrame(elbow_data)

# Review the DataFrame
df_elbow.head()
```

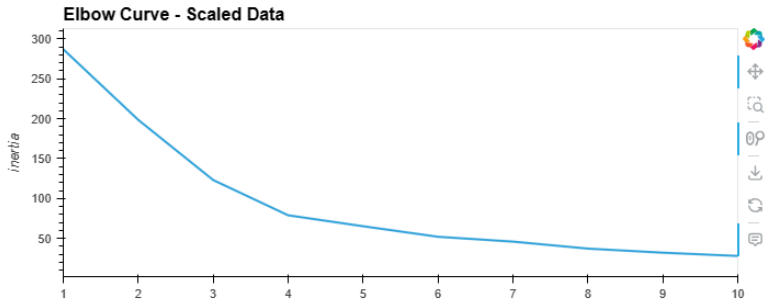
Out[13]:

	k	inertia
0	1	287.000000
1	2	198.571818
2	3	123.190482
3	4	79.022435
4	5	65.302379

In [14]:

```
# Plot a line chart with all the inertia values computed with
# the different values of k to visually identify the optimal value for k.
k_elbow = df_elbow.hvplot.line(
    x="k",
    y="inertia",
    title="Elbow Curve - Scaled Data",
    xticks=k
)
k_elbow
```

Out[14]:



In [17]:

```
# Answer the following question:
# Question: What is the best value for k?

# Answer:
# K value is the line of best fit and 4 would be the answer base on the line graph
```

Cluster Cryptocurrencies with K-means Using the Original Data

In [18]:

```
# Initialize the K-Means model using the best value for k
# clusters should be 4 based on the data from the line graph
model = KMeans(n_clusters=4, random_state=0)
```

In [19]:

```
# model.fit the K-Means model using the scaled data
model.fit(df_market_data_scaled)
```

Out[19]:

KMeans

KMeans(n_clusters=4, random_state=0)

In [54]:

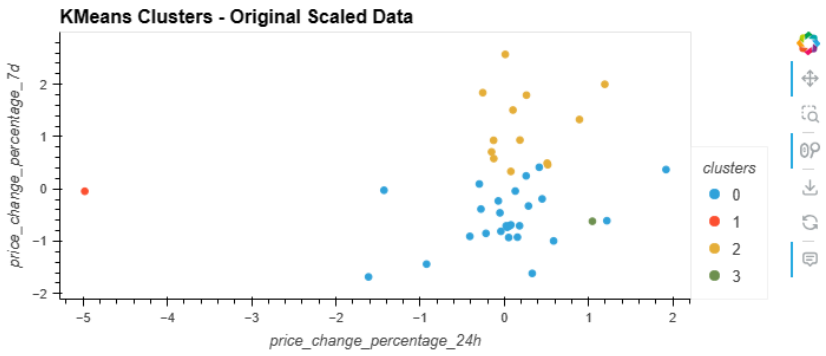
```
# Kmeanes = model.predict the clusters to group the cryptocurrencies using the scaled data
km = model.predict(df_market_data_scaled)

# Print the resulting array of cluster values.
print(km)
```

```
[2 2 0 0 2 2 2 2 2 0 0 0 0 2 0 2 0 0 2 0 0 2 0 0 0 0 0 0 0 2 0 0 0 1 2 0 0 3
 0 0 0 0]
```

In [23]:

```
# Create a scatter plot using hvPlot by setting
# `x="price_change_percentage_24h"` and `y="price_change_percentage_7d"`.
# Color the graph points with the labels found using K-Means and
# add the crypto name in the `hover_cols` parameter to identify
# the cryptocurrency represented by each data point.
k_plot = df_crypto_predictions.hvplot.scatter(
    x="price_change_percentage_24h",
    y="price_change_percentage_7d",
    title="KMeans Clusters - Original Scaled Data",
    by='clusters',
    hover_cols='coin_id'
)
k_plot
```

Out[23]:



In [25]:
```
# Use the PCA model with `fit_transform` to reduce to
# three principal components.
crypto_pca_data = pca.fit_transform(df_market_data_scaled)
# View the first five rows of the DataFrame.
crypto_pca_data[0:5]
```

Out[25]:
```
array([[-0.60066733,  0.84276006,  0.46159457],
       [-0.45826071,  0.45846566,  0.95287678],
       [-0.43306981, -0.16812638, -0.64175193],
       [-0.47183495, -0.22266008, -0.47905316],
       [-1.15779997,  2.04120919,  1.85971527]])
```

In [26]:
```
# Retrieve the explained variance to determine how much information
# can be attributed to each principal component.
pca.explained_variance_ratio_
```

Out[26]:
```
array([0.3719856 , 0.34700813, 0.17603793])
```

Answer the following question:

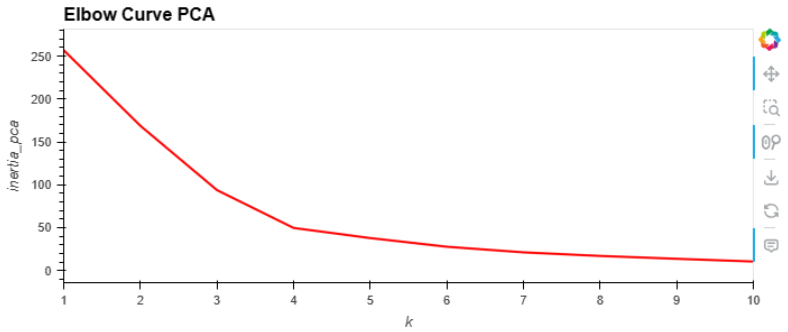
Question: What is the total explained variance of the three principal components?

Answer: Total Explained Variance for the model is 89.5%

2	3	93.774626
3	4	49.665497
4	5	37.839466

In [31]:
```
# Plot a line chart with all the inertia values computed with
# the different values of k to visually identify the optimal value for k.
pca_elbow = df_elbow_pca.hvplot.line(
    x="k",
    y="inertia_pca",
    title="Elbow Curve PCA",
    color='red',
    xticks=k
)
pca_elbow
```

Out[31]:



Answer the following questions:

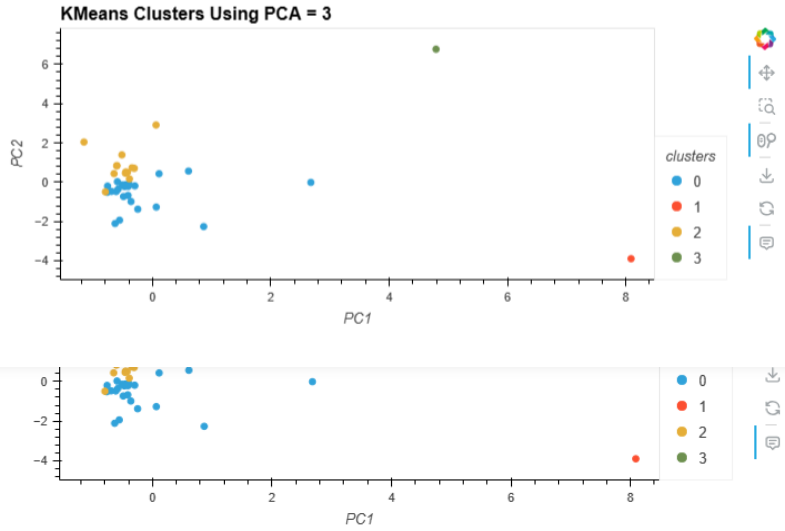
- **Question:** What is the best value for `k` when using the PCA data?
  - **Answer:**he best k value using PCA is 4.
- **Question:** Does it differ from the best k value found using the original data?
  - **Answer:** No, the original data also showed 4

coin_id				
bitcoin	-0.600667	0.842760	0.461595	2
ethereum	-0.458261	0.458466	0.952877	2
tether	-0.433070	-0.168126	-0.641752	0
ripple	-0.471835	-0.222660	-0.479053	0
bitcoin-cash	-1.157800	2.041209	1.859715	2

In [36]:

```
# Create a scatter plot using hvPlot by setting
# `x="PC1"` and `y="PC2"`.
# Color the graph points with the labels found using K-Means and
# add the crypto name in the `hover_cols` parameter to identify
# the cryptocurrency represented by each data point.
pca_plot = df_crypto_pca_predictions.hvplot.scatter(
    x="PC1",
    y="PC2",
    title="KMeans Clusters Using PCA = 3",
    by='clusters',
    hover_cols='coin_id'
)
pca_plot
```

Out[36]:



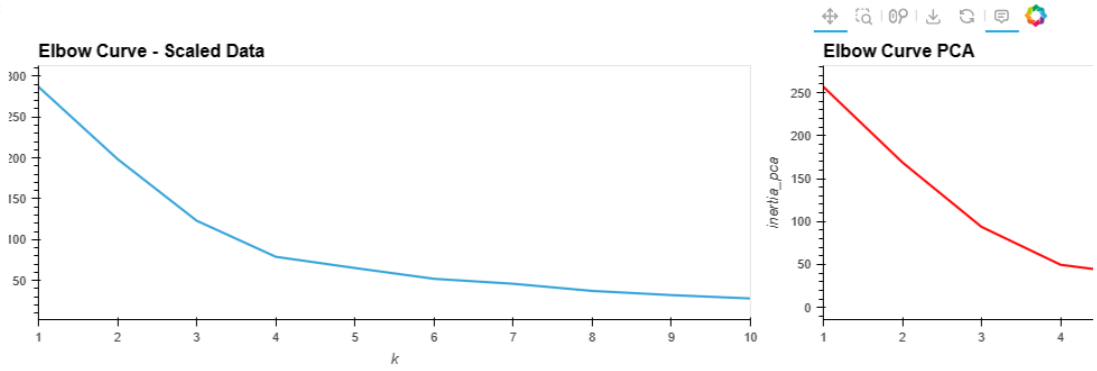
Visualize and Compare the Results

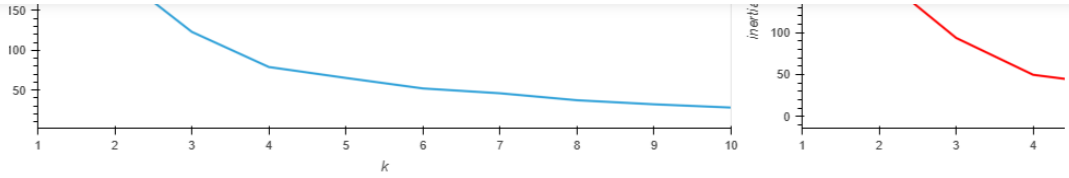
In this section, you will visually analyze the cluster analysis results by contrasting the outcome with and without using the optimization techniques.

In [37]:

```
# Composite plot to contrast the Elbow curves
composite_scatter_distinct = k_elbow + pca_elbow
composite_scatter_distinct
```

Out[37]:

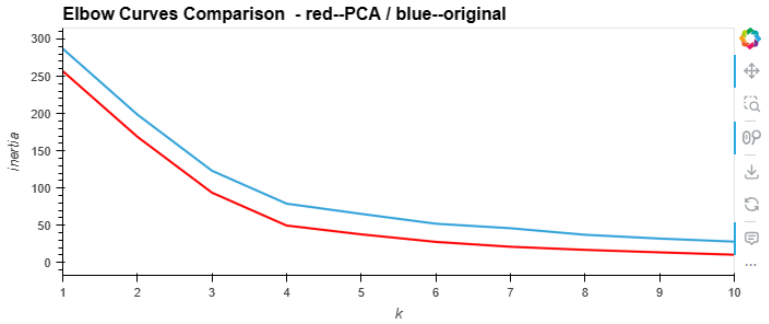




In [38]:

```
# Composite plot to contrast the Elbow curves - Combined graphs
composite_scatter_combined = k_elbow * pca_elbow
composite_scatter_combined.opts(title='Elbow Curves Comparison  - red--PCA / blue--original'

)

```

Out[38]:



In [73]:

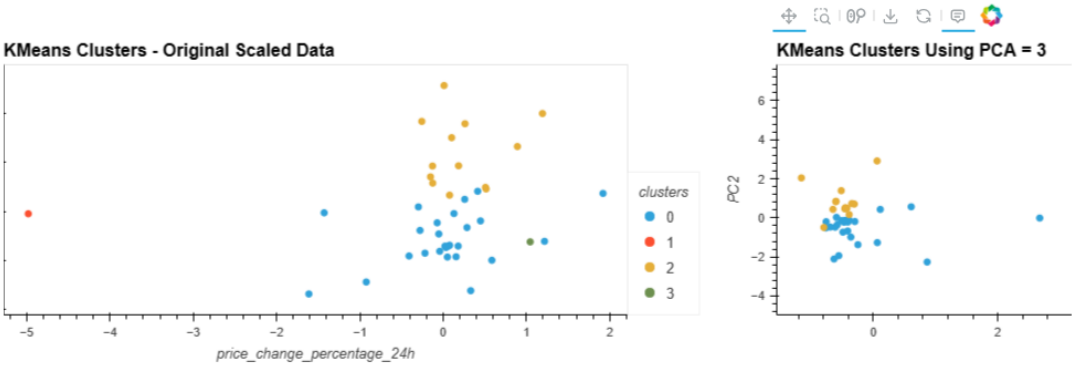
```
# Composite plot to contrast the clusters
# Side by side plot. A combined plot would be too messy to be helpful.
k_plot + pca_plot
```

Out[73]:

In [40]:

```
# Composite plot to contrast the clusters
# Side by side plot. A combined plot would be too messy to be helpful.
k_plot + pca_plot
```

Out[40]:



Answer the following question:

- **Question:** After visually analyzing the cluster analysis results, what is the impact of using fewer features to cluster the data using K-Means?
- **Answer:** Using fewer features to cluster the data using KMeans allows the removal of some variblitiy and a cleaner view of the clustering. PCA removes some of the overlaying of clusters.