Homework 2: Unsupervised Learning: Dimensional Reduction: Principal Component Analysis (PCA)

- Explain the problem
- Explain the AI Model & Data
- Explain/Analysis the results

Explain the problem:

Using PCA to reduce the dimension of wine data after that using K_Means to seperate the wine base on its characteristic. This will be using for new wine that come to the shop without the label and user do not know how many group that they will divide the wine into. Base on its characteristic, my AI model will caculate the best group that we should divide the wine into. Then plot it. I also run a test to validate the wine with its original label and the result come out very close to 39% (which is good for just unsupervised learning on multi dimension data set). Please note we are not design an AI/ML model to classify each wine botle in this project (that should be the job of supervised learning algorithm with labels like decision tree). My goal is to find the best k possible in a group. This AI model can be use for any applications (sperate water, liquids, softdrink) or any data with number attributes only

Data Source:

The data will be retrieved from sklearn simple data sets (data is clean and tidy) under the name datasets.load_wine() (from sklearn import datasets)

1) Exploratory Data Analyst:

In [32]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA # Principle component Analyst
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.metrics import silhouette_score
```

In [5]:
```
# Load the data
# Read the wine dataset and translate to pandas dataframe
wine_sk = datasets.load_wine()

# Make sure data is in the same range
wine_sk.data = MinMaxScaler().fit_transform(wine_sk.data)
```

```
# Note that the "target" attribute is species, represented as an integer
wine_data = pd.DataFrame(data= np.c_[wine_sk['data']],columns= wine_sk['feature_names']
```

In [3]:
```
# Validate to make sure this is the data doesn't contain any classification attributes
wine_data
```

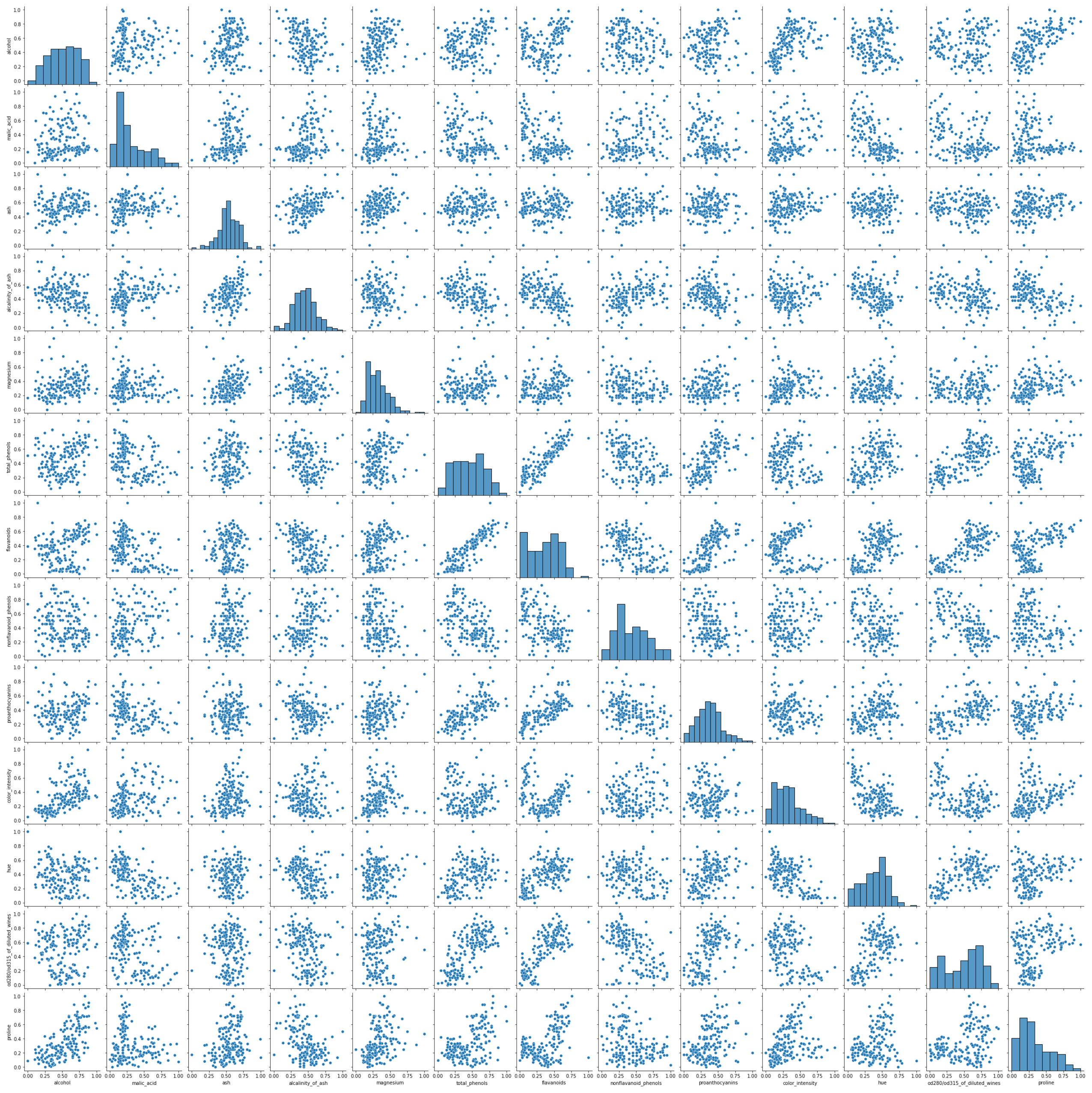
Out[3]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavano
0	0.842105	0.191700	0.572193	0.257732	0.619565	0.627586	0.573840	
1	0.571053	0.205534	0.417112	0.030928	0.326087	0.575862	0.510549	
2	0.560526	0.320158	0.700535	0.412371	0.336957	0.627586	0.611814	
3	0.878947	0.239130	0.609626	0.319588	0.467391	0.989655	0.664557	
4	0.581579	0.365613	0.807487	0.536082	0.521739	0.627586	0.495781	
...	...	...	...	...	...	...	...	
173	0.705263	0.970356	0.582888	0.510309	0.271739	0.241379	0.056962	
174	0.623684	0.626482	0.598930	0.639175	0.347826	0.282759	0.086498	
175	0.589474	0.699605	0.481283	0.484536	0.543478	0.210345	0.073840	
176	0.563158	0.365613	0.540107	0.484536	0.543478	0.231034	0.071730	
177	0.815789	0.664032	0.737968	0.716495	0.282609	0.368966	0.088608	

178 rows × 13 columns

In [7]:
```
# visualize the correlation between these data
sns.color_palette("bright")
sns.pairplot(wine_data)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x24a2fdcaf10>
```



Result: Does not look like any obvious grouping

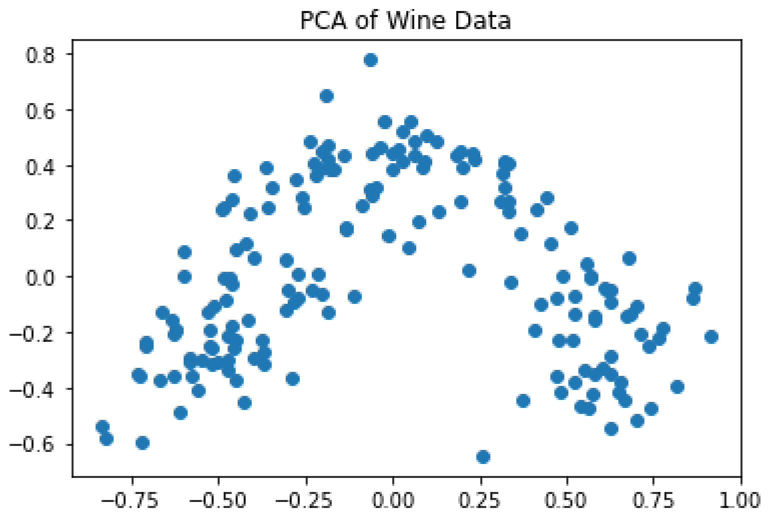
2) AI Model: PCA and K-Means

a) PCA:

- Principal component analysis is a machine learning, category as unsupervised learning which focus on dimensional reduction. It usually use as a prep for other machine learning methods (in my experience)
- Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.(Source: https://en.wikipedia.org/wiki/Principal_component_analysis)

In [15]:
```
# Keep track of our data
X = wine_sk.data

# Choose number of components
pca = PCA(n_components=2)

# Calculate PCA
pca.fit(X)

# Get PCA version of fitted data
transformed_X = pca.transform(X)

# Plot the results
plt.scatter(transformed_X[:, 0], transformed_X[:, 1])
plt.title("PCA of Wine Data")
```

Out[15]:
```
Text(0.5, 1.0, 'PCA of Wine Data')
```



Result: Does not look like any obvious grouping either, let try some clustering

b) Kmeans:

- K Mean Clustering is a machine learning methods, category as unsupervised learning which focus on clusturing unlabel data. k -means clustering is apply to large data sets, particularly when using heuristics functions (my experience)
- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and so on (Source: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning)

Self define function to calculate the best K (clusters) number

In [26]:
```
def calculate_cluster_sse(cluster_data):
    """
    Goal: calculate the SSE for the dataset 'data' given cluster assignments.

    INPUT:
          data: the dataset of size (n_samples, n_attributes) where n_samples is the nu
                and n_attributes is the number of attributes for each data point.
    OUTPUT:
          A floating point value containing the SSE of that cluster
    """
    cluster_centers = np.mean(cluster_data, axis=0)
    within_sse = np.sum(np.square(cluster_data - cluster_centers))

    return within_sse
```

In [27]:
```
def calculate_sse(data, cluster_assignments):
    """
    INPUT:
          data: the dataset of size (n_samples, n_attributes) where n_samples is the nu
                and n_attributes is the number of attributes for each data point.
          cluster_assignments: a numpy integer vector containing the cluster assignment
                           It is of shape (n_attributes, ).

    OUTPUT:
          A floating point value containing the SSE
    """
    clusters = np.unique(cluster_assignments) # make this unique so only 0, 1 is return
    final_SSE = 0

    for c in clusters:
        # c will return 1 cluster contatin inside the other clusters
        cluster_data = data[cluster_assignments == c, :]
        final_SSE += calculate_cluster_sse(cluster_data) # Call final SSE to return all
        # print(f'c: {c}, data: {cluster_data}, final: {final_SSE}')

    return final_SSE
```

In [28]:
```
def get_elbow_data(data, k_max):
    """
    Goal: generate a list of SSEs for each corresponding K-means clustering of size k.

    INPUT:
        data: the dataset of size (n_samples, n_attributes) where n_samples is the numb
                and n_attributes is the number of attributes for each data point.
        k_max: an integer representing the maximum number of clusterings to perform in

    OUTPUT:
        A list of SSE scores of length k: where SSE[k] is the SSE score of a K-means cl

    """
    return [calculate_sse(data,KMeans(k).fit(data).labels_) for k in range(2, k_max)]
```

In [33]:
```
def get_silhouettes(data, k_max):
    silhouettes = []
    """
    INPUT:
        data: the dataset of size (n_samples, n_attributes) where n_samples is the numb
                and n_attributes is the number of attributes for each data point.
        k_max: an integer representing the maximum number of clusterings to perform in

    OUTPUT:
        A list of silhouette scores of length k: where SSE[k] is the SSE score of a K-m
    """
    K = range(2, k_max)
    for k in K:
        kmeans = KMeans(n_clusters=k)
        results = kmeans.fit(data)
        # Now add the Silhouette score!
        silhouettes.append(silhouette_score(data, results.labels_))

    return silhouettes
```

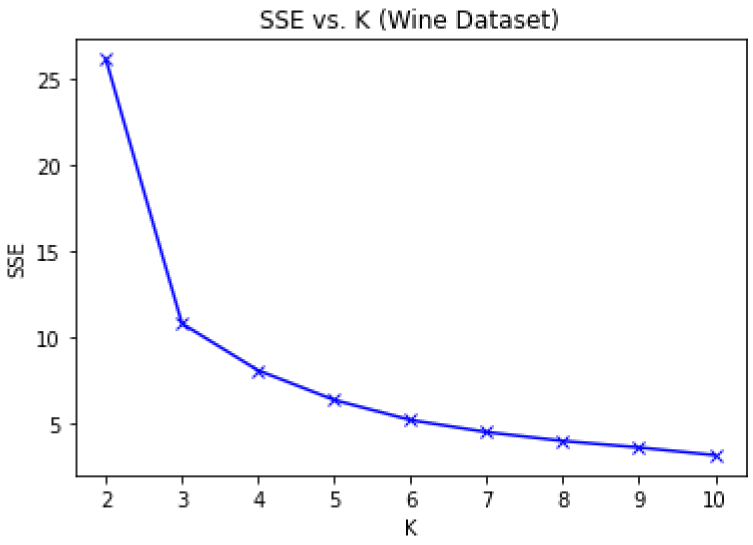
Use the elbow method and silhoutte score to find a good value of K for K-means.

In [25]:

```
sse_values = get_elbow_data(transformed_X, 11)

plt.plot(range(2, 11), sse_values, 'bx-')
plt.title("SSE vs. K (Wine Dataset)")
plt.xlabel("K")
plt.ylabel("SSE")
```

Out[25]:

```
Text(0, 0.5, 'SSE')
```

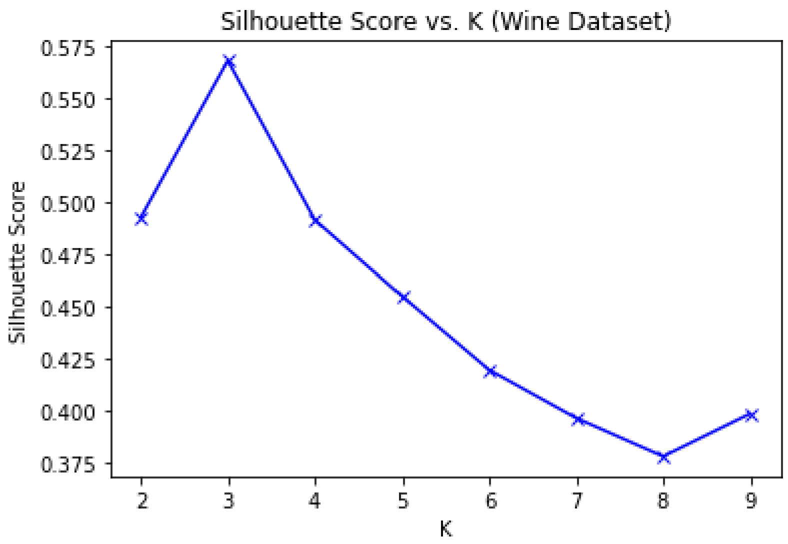


In [34]:

```
silhouette_scores = get_silhouettes(transformed_X, 10)

plt.plot(range(2, 10), silhouette_scores, 'bx-')
plt.title("Silhouette Score vs. K (Wine Dataset)")
plt.xlabel("K")
plt.ylabel("Silhouette Score")
```

Out[34]:

```
Text(0, 0.5, 'Silhouette Score')
```



Result: As we can clearly see in the chart SSE vs K and Silhoute Score vs. K the elbow is clearly at 3 (3 make dramatical change on the 2 graph). Therefore, we choose as the number of cluster: k=3

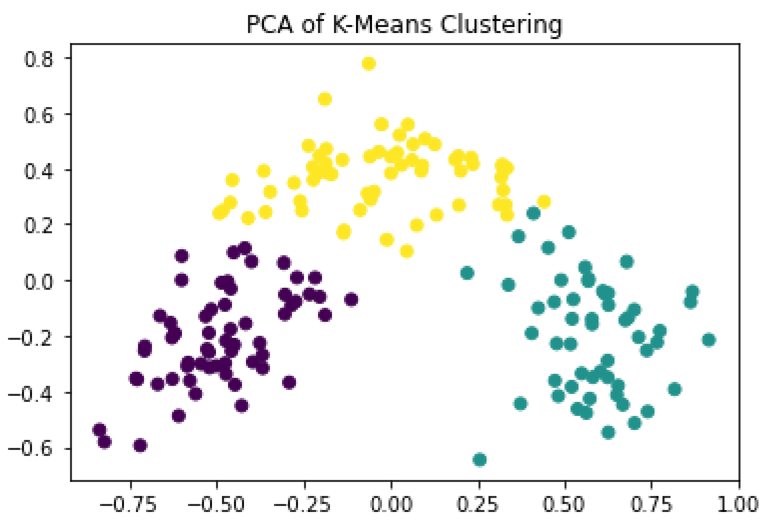
In [36]:
```
# CLustering using k = 3

k = 3
kmeans = KMeans(n_clusters=k)
results = kmeans.fit(wine_sk.data)
results.labels_
```

Out[36]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 2, 2, 2, 2,
       2, 2, 1, 2, 1, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

Plot the PCA with the color generate by Kmeans

In [37]:
```
# Plot the PCA with the color generate by Kmeans
plt.scatter(transformed_X[:, 0], transformed_X[:, 1], c = results.labels_)
plt.title("PCA of K-Means Clustering")
```

Out[37]:
```
Text(0.5, 1.0, 'PCA of K-Means Clustering')
```

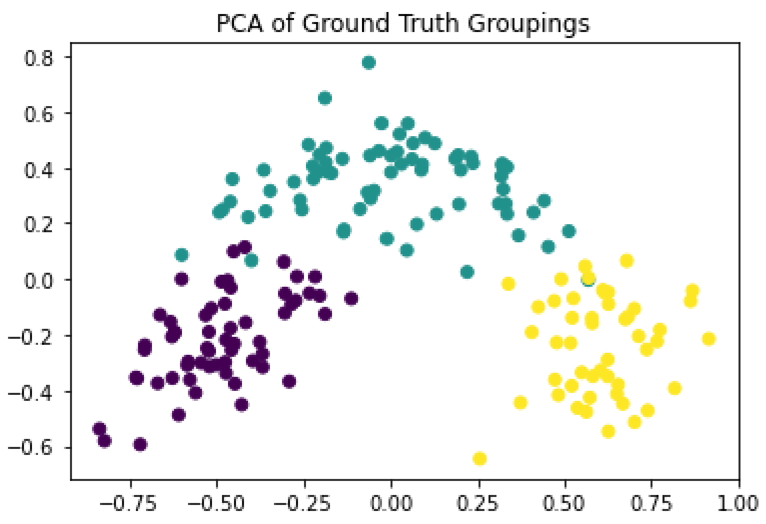


Test the data with the actual label from wine SK

In [38]:
```
# Plot the results
Y = wine_sk.target
print("There were "+str(len(np.unique(wine_sk.target)))+" cultivars")

plt.scatter(transformed_X[:, 0], transformed_X[:, 1], c = Y)
plt.title("PCA of Ground Truth Groupings")
```

```
There were 3 cultivars
```

Out[38]:
```
Text(0.5, 1.0, 'PCA of Ground Truth Groupings')
```



In [40]:
```
# Calculate the original result and the found-result (clustering)
(sum(Y == results.labels_)/sum(Y))
```

Out[40]:
```
0.38922155688622756
```

3) Explain the Analyst

Result: Result show data that is group in 3 group, and when compare with the original classification (lebel) in the chart it is very similar. Please notice I did not choose the k = 3 randomly but using custom algorithm to plot and find it. Therefore our AII model is work fairly well with this combination. **Explain:** Since the data is clean, we do not need to tidy it. The scale between the value is big so, I use mean max scaler to normalize the data. After that, we need to validate if there are any classification attributes in the data. Then we are going to use principal component analysis (PCA) to reduce the dimension of the data (transforming the data into two-dimension data). After that, we define some custom algorithms to calculate the best K-number which means the best clusters. This custom algorithm includes calculating the SSA number for a given dataset and generating a list of SSEs for each corresponding K-means clustering of size k. Then using data visualization to decide the number of clusters (k number). Finally, plot the found k results into the PCA data to see the results.

My AI model include all of the above step (to find the k, plot the k). However real world data set we may not have a predefined label to validate our results. Other methods should be enforce for validation.

The results of just apply this methods to group to the original label is only 39 %. However please notice this is unsupervised learning and our goal is to find the best number of cluster. How to evaluate each cluster to define can be further explain and work based on the clients need and command.

```
In [ ]:
```