1.

```python
from cProfile import label
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

'''Given the dataset: recipes_muffins_cupcakes_scones.csv,
    1. print the variance ratio and plot the cumulative sum of the variance ratio for all 8 features.                    /*DONE*/
    2. In addition, using as a reference the plots in slides 110-115, create a scatter plot between PC1 and PC2.        /*DONE*/
    3. In addition, create a histogram of features and plot a heatmap with the features with the largest variation in PC1 and PC2.  /*DONE*/
    4. Furthermore, find and print the features with the highest (max) and lowest (min) variation both in PC1 and PC2
        (positively and negatively correlated).                                                                          /*DONE*/
    5. Finally, plot a correlation heatmap.                                                                              /*DONE*/

Note: You will need to standardize your data points.
'''

df = pd.read_csv('recipes_muffins_cupcakes_scones.csv')

df.loc[df['Type'] == 'Muffin' , 'Type'] = 0
df.loc[df['Type'] == 'Cupcake' , 'Type'] = 1
df.loc[df['Type'] == 'Scone' , 'Type'] = 2

X = np.array(df.loc[:, 'Flour':'Salt' ])
y = np.array(df.loc[:, 'Type'])


Xscaler = StandardScaler()
X = Xscaler.fit_transform(X)
```

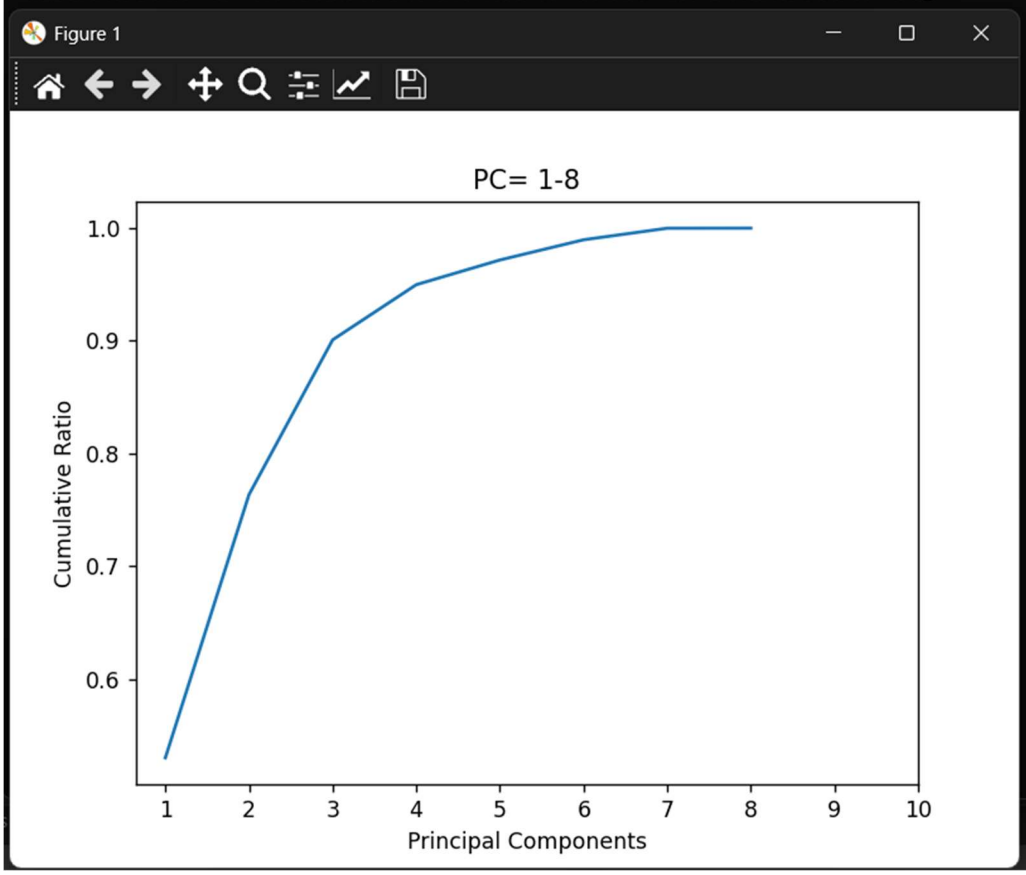
```python
pca = PCA(n_components=8)
PC = pca.fit_transform(X)
explained_variance = pca.explained_variance_ratio_
print(f'\nVariance Ratio: {explained_variance}')
print()

#cumulative sum
x= np.arange(1,9)
plt.plot(x, np.cumsum(explained_variance))
plt.xlabel('Principal Components')
plt.ylabel('Cumulative Ratio')
plt.title('PC= 1-8')
plt.xticks(range(1,11))
plt.show()

# scatter plot
plt.scatter(PC[:, 0], PC[:, 1], c=y , cmap='plasma')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title(f'PC= 2, Variance: {explained_variance[0:1]}')
plt.show()

# histogram
Muffin = X[y==0]
Cupcake = X[y==1]
Scone = X[y==2]

fig,axes =plt.subplots(2,4, figsize=(12, 9)) # 3 columns each containing 10 figures, total 30 features
ax=axes.ravel()# flat axes with numpy ravel
feature_names = [*df]
feature_names.pop(0)

```

```
for i in range(8):
    _,bins=np.histogram(X[:,i],bins=25)
    ax[i].hist(Muffin[:,i],bins=bins,color='r',alpha=.5)# red color for malignant class
    ax[i].hist(Cupcake[:,i],bins=bins,color='g',alpha=0.3)# alpha is           for transparency in the overlapped region
    ax[i].hist(Scone[:,i],bins=bins,color='b',alpha=0.2)# alpha is           for transparency in the overlapped region
    ax[i].set_title(feature_names[i],fontsize=9)
    ax[i].axes.get_xaxis().set_visible(False) # the x-axis co-ordinates are not so useful, as we just want to look how well separated the histograms are
    ax[i].set_yticks(())

ax[0].legend(['Muffin','Cupcake', 'Scone'],loc='best',fontsize=8)
plt.tight_layout()# let's make good plots
plt.show()

#highest (max) and lowest (min) variation

PCAcomp = pca.components_
PCAcomp = PCAcomp[0:2,:]


print(f"PC1 highest variation feature {feature_names[int(np.where(PCAcomp == max(PCAcomp[0,:]))[1][0]  )]}: {max(PCAcomp[0,:])} ")
print(f"PC1 lowest variation feature {feature_names[int(np.where(PCAcomp == min(PCAcomp[0,:]))[1][0])]}: {min(PCAcomp[0,:])} ")

print(f"PC2 highest variation feature {feature_names[int(np.where(PCAcomp == max(PCAcomp[1,:]))[1][0]  )]}: {max(PCAcomp[1,:])} ")
print(f"PC2 lowest variation feature {feature_names[int(np.where(PCAcomp == min(PCAcomp[1,:]))[1][0])]}: {min(PCAcomp[1,:])} ")
```

```
#heatmap
df2 = df.drop('Type',axis='columns')
s=sns.heatmap(df2.corr(),cmap='coolwarm')
s.set_yticklabels(s.get_yticklabels(),rotation=30,fontsize=7)
s.set_xticklabels(s.get_xticklabels(),rotation=30,fontsize=7)
plt.show()

#KNN
pca = PCA(n_components=2)
X_t = pca.fit_transform(X)
y=y.astype('int')

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_t, y)

data = np.array([38, 18, 23, 20, 9, 3, 1, 0]).reshape(1,-1)
data = Xscaler.transform(data)
data = pca.transform(data)



pred = knn.predict(data)

prediction = ''
if pred == 0:
    prediction = 'Muffin'
elif pred == 1:
    prediction = 'Cupcake'
else:
    prediction = 'Scone'

print()
print('the data is predicted to be a ', prediction)
```

```
plt.scatter(X_t[:, 0], X_t[:, 1], c=y , cmap='plasma')
plt.scatter(data[0,0], data[0,1], c='r' , cmap='plasma')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title(f'PC= 2, Variance: {explained_variance[0:1]}')

# Displaying the legend
plt.show()
```

```
Variance Ratio: [5.30713818e-01 2.32926199e-01 1.37271639e-01 4.89399359e-02
 2.18813363e-02 1.79333825e-02 1.02410755e-02 9.26139804e-05]
```



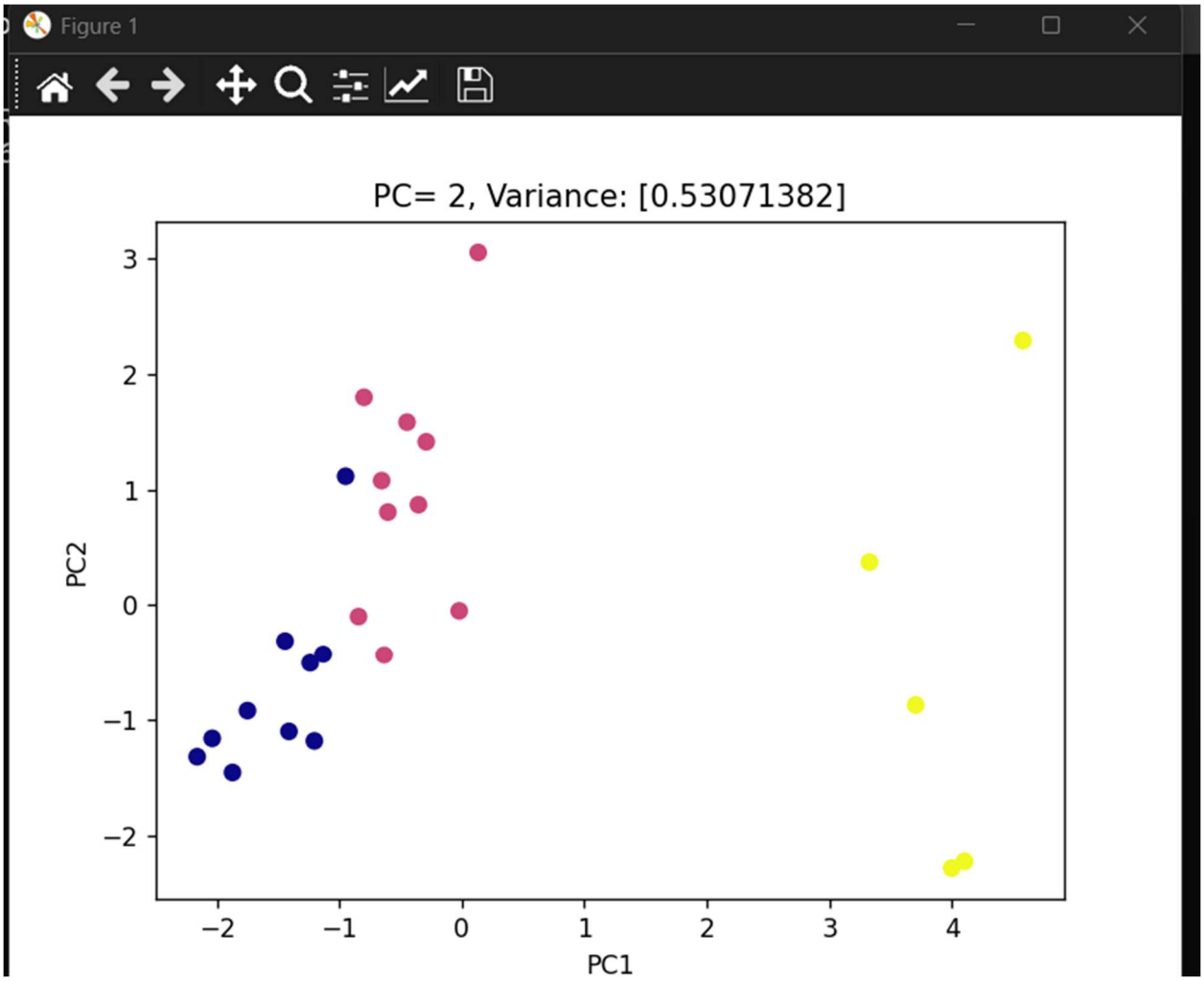


Figure 1
PC= 2, Variance: [0.53071382]
PC2
PC1
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4

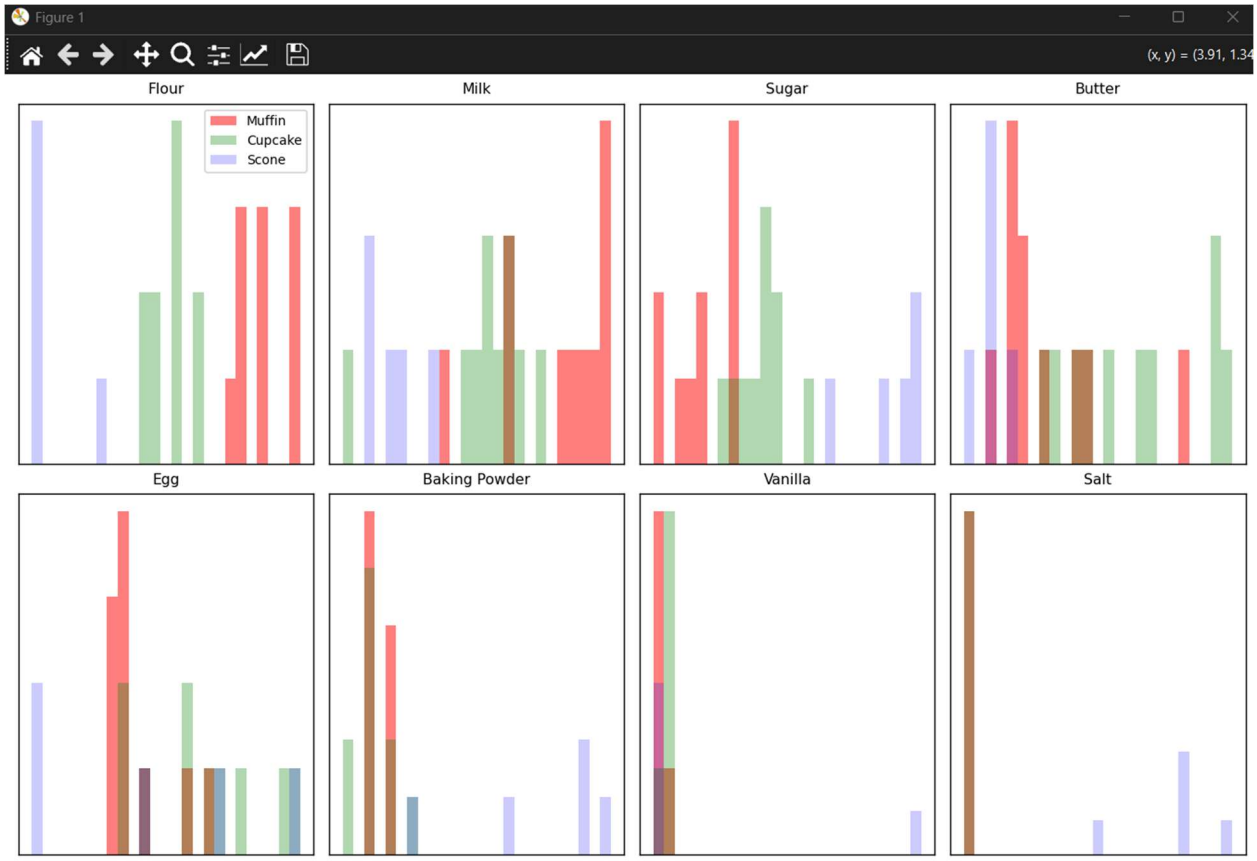


Figure 1
(x, y) = (3.91, 1.34
Flour
Milk
Sugar
Butter
Muffin
Cupcake
Scone
Egg
Baking Powder
Vanilla
Salt

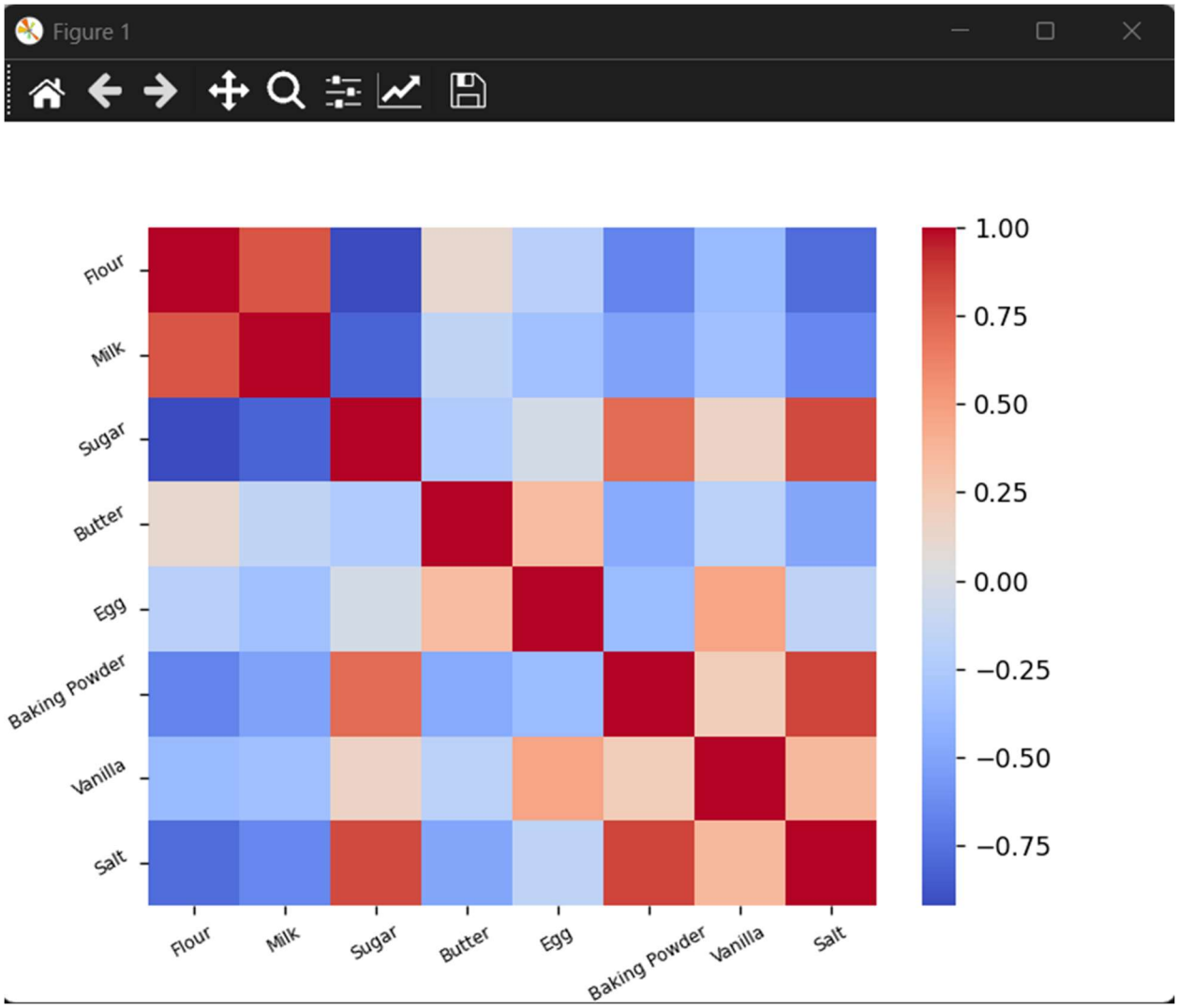
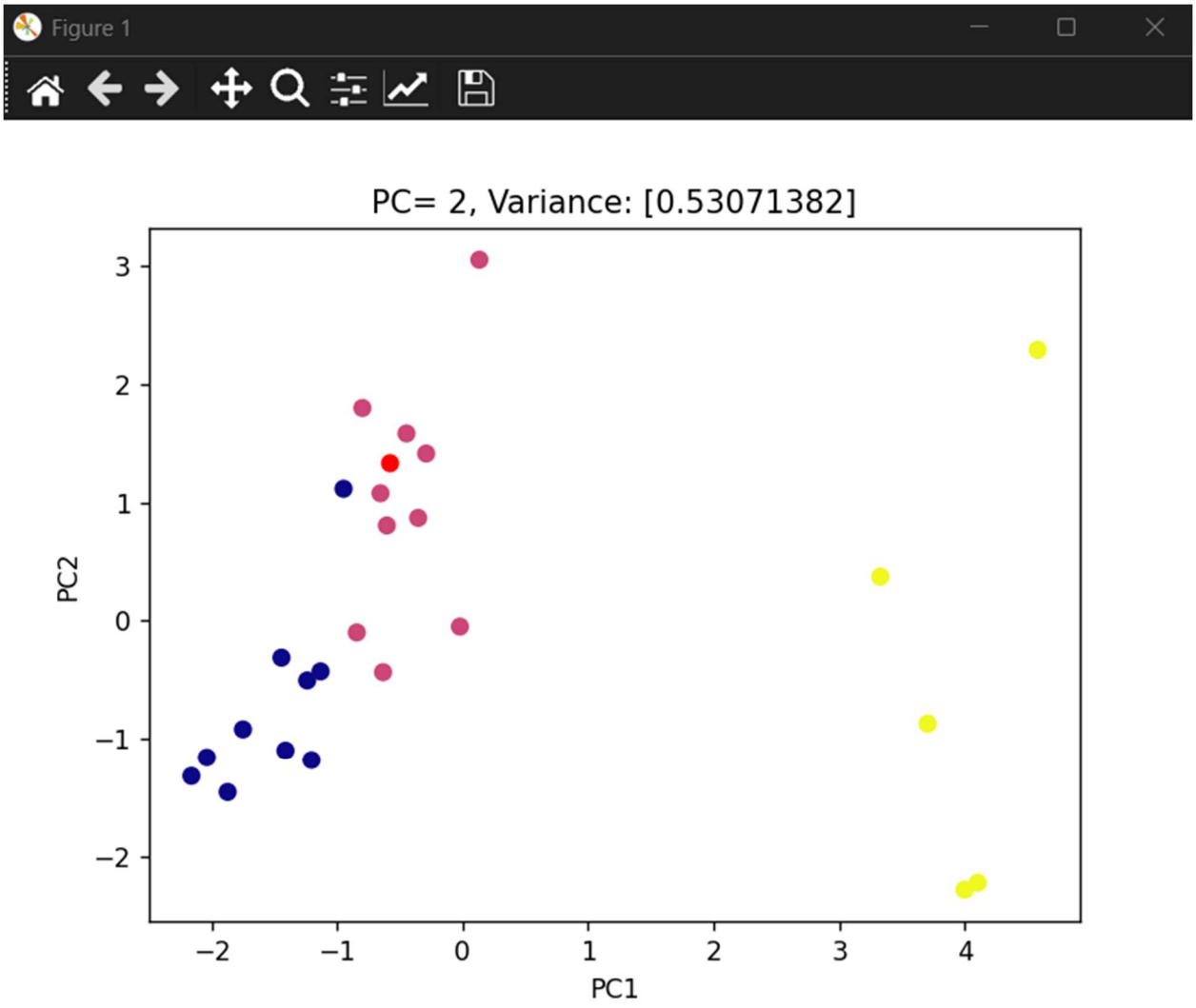


Figure 1
Flour
Milk
Sugar
Butter
Egg
Baking Powder
Vanilla
Salt
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75




```
Variance Ratio: [5.30713818e-01 2.32926199e-01 1.37271639e-01 4.89399359e-02
 2.18813363e-02 1.79333825e-02 1.02410755e-02 9.26139804e-05]

PC1 highest variation feature Salt: 0.45560967418642023
PC1 lowest variation feature Flour: -0.4452797934996038
PC2 highest variation feature Egg: 0.6628137748507001
PC2 lowest variation feature Milk: -0.3233860340000608

the data is predicted to be a  Cupcake
```