Project_P4

February 3, 2026

```
[1]: ## Step 1: Import tools and load data frame
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns

     from sklearn.preprocessing import StandardScaler
     from sklearn.decomposition import PCA
     from sklearn.cluster import KMeans
     from sklearn.linear_model import LogisticRegression
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score, confusion_matrix,␣
      ↪classification_report
     from sklearn.linear_model import LogisticRegression
     from sklearn.model_selection import train_test_split, cross_val_score
     from sklearn.metrics import accuracy_score, confusion_matrix,␣
      ↪classification_report
     from mpl_toolkits.mplot3d import Axes3D
     from sklearn.metrics import accuracy_score, f1_score

     df = pd.read_csv('project_data.csv')
     df
```

```
[1]:        Unnamed: 0                Tweet  Followers  Friends  Num_tweets  \
     0               0  1441497018807906305       1198    605.0        6166
     1               1  1431812786099613699        608   1259.0        1811
     2               2  1426644707313135617        173    167.0        4306
     3               3  1431814908425998337       2540    222.0        6008
     4               4  1432862687533441027       4439     11.0        9985
     …             …                    …        …        …         …
     28818       28815  1431029315274162181      16182   1150.0       58245
     28819       28816  1437970083087851520     688077    636.0       12308
     28820       28817  1428110541093052418      57068   1225.0       25131
     28821       28818  1430722665514377219         66    121.0         549
     28822       28819  1441165693974503442        169    276.0        2910

            Verified  Listed_count Location  Age  Length  Num_users  \
0         False            1     True    7       6          4
1         False            5     True    2       2          2
2         False            0     True    0       4          2
3         False            0     True    9       2          2
4         False           55     True    2       4          3
…         …           …      …  …     …        …
28818      True          685    False   12       2          2
28819      True          361    False   10     247        218
28820     False            0     True    5      22         21
28821     False            3     True    1       4          3
28822     False            7     True    0       2          2

       Num_author_replies  TOXICITY_x  Num_toxic_direct_replies  \
0                       2    0.235235                       0.0
1                       0    0.085582                       0.0
2                       2    0.076877                       0.0
3                       0    0.095684                       0.0
4                       1    0.165919                       0.0
…                     …         …                      …
28818                   0    0.042905                       0.0
28819                   3    0.039698                       8.0
28820                   0    0.018346                       2.0
28821                   1    0.924899                       0.0
28822                   0    0.035456                       0.0

       Num_toxic_nested_replies  Num_author_toxic_replies  Num_toxic_replies  \
0                           0.0                       0.0                0.0
1                           0.0                       0.0                0.0
2                           0.0                       0.0                0.0
3                           0.0                       0.0                0.0
4                           0.0                       0.0                0.0
…                         …                       …                …
28818                       0.0                       0.0                0.0
28819                       2.0                       0.0               10.0
28820                       0.0                       0.0                2.0
28821                       0.0                       0.0                0.0
28822                       0.0                       0.0                0.0

       Toxic
0          0
1          0
2          0
3          0
4          0
…        …
28818      0
28819      1
```

```
28820      1
28821      0
28822      0

[28823 rows x 18 columns]
```

[2]:
```
## Step 2: Clean data
df = df.dropna(axis=0) # removes rows containing missing values
df = df.drop_duplicates() # removes duplicate rows leaving just one
```

[3]:
```
## Create Total_toxic_replies and Toxic_conversation columns
df['Total_toxic_replies'] = df['Num_toxic_direct_replies'] +␣
↪df['Num_toxic_nested_replies']
df['Toxic_conversation'] = (df['Total_toxic_replies'] > 0).astype(int)
df
```

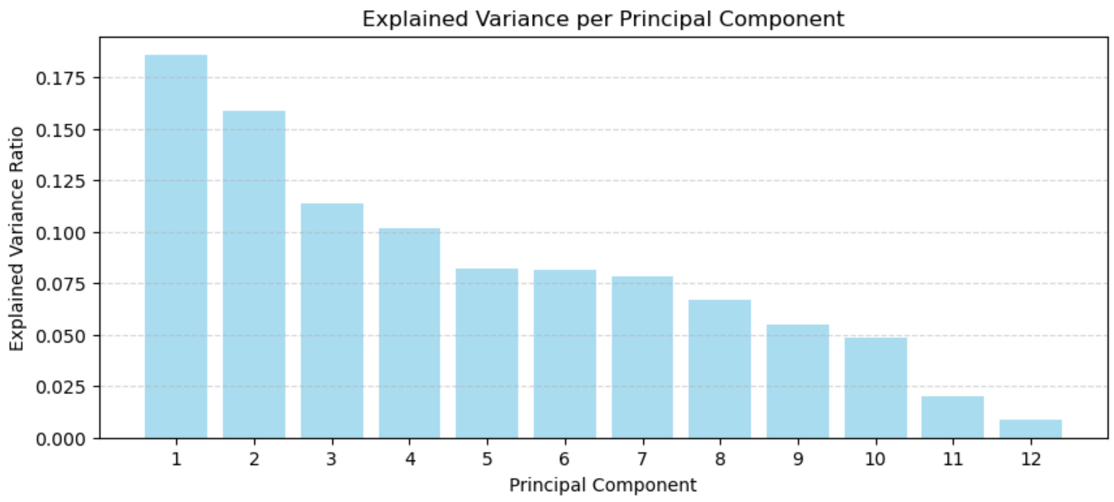
[3]:
```
       Unnamed: 0                Tweet  Followers  Friends  Num_tweets  \
0               0  1441497018807906305       1198    605.0        6166
1               1  1431812786099613699        608   1259.0        1811
2               2  1426644707313135617        173    167.0        4306
3               3  1431814908425998337       2540    222.0        6008
4               4  1432862687533441027       4439     11.0        9985
…             …                    …         …       …           …
28818       28815  1431029315274162181      16182   1150.0       58245
28819       28816  1437970083087851520     688077    636.0       12308
28820       28817  1428110541093052418      57068   1225.0       25131
28821       28818  1430722665514377219         66    121.0         549
28822       28819  1441165693974503442        169    276.0        2910

       Verified  Listed_count Location  Age  Length  Num_users  \
0         False             1     True    7       6          4
1         False             5     True    2       2          2
2         False             0     True    0       4          2
3         False             0     True    9       2          2
4         False            55     True    2       4          3
…          …            …       …  …     …        …
28818      True           685    False   12       2          2
28819      True           361    False   10     247        218
28820     False             0     True    5      22         21
28821     False             3     True    1       4          3
28822     False             7     True    0       2          2

       Num_author_replies  TOXICITY_x  Num_toxic_direct_replies  \
0                       2    0.235235                       0.0
1                       0    0.085582                       0.0
2                       2    0.076877                       0.0
3                       0    0.095684                       0.0
```

```
4                       1    0.165919                      0.0
…                   …         …                        …
28818                   0    0.042905                      0.0
28819                   3    0.039698                      8.0
28820                   0    0.018346                      2.0
28821                   1    0.924899                      0.0
28822                   0    0.035456                      0.0

       Num_toxic_nested_replies  Num_author_toxic_replies  Num_toxic_replies  \
0                           0.0                       0.0                0.0
1                           0.0                       0.0                0.0
2                           0.0                       0.0                0.0
3                           0.0                       0.0                0.0
4                           0.0                       0.0                0.0
…                         …                        …                …
28818                       0.0                       0.0                0.0
28819                       2.0                       0.0               10.0
28820                       0.0                       0.0                2.0
28821                       0.0                       0.0                0.0
28822                       0.0                       0.0                0.0

       Toxic  Total_toxic_replies  Toxic_conversation
0          0                  0.0                   0
1          0                  0.0                   0
2          0                  0.0                   0
3          0                  0.0                   0
4          0                  0.0                   0
…      …                  …                 …
28818      0                  0.0                   0
28819      1                 10.0                   1
28820      1                  2.0                   1
28821      0                  0.0                   0
28822      0                  0.0                   0

[28818 rows x 20 columns]
```

[4]:
```
## Standardize features using standard scaler
X_features = ['Followers','Friends','Num_tweets','Verified','Listed_count',
            ␣
 ↪'Location','Age','Length','Num_users','Num_author_replies','TOXICITY_x','Num_author_toxic_r
X = df[X_features]
y = df['Toxic_conversation']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

[4]:
```
array([[-0.05292775, -0.06192729, -0.38627708, …,  0.18440487,
         0.18690991, -0.18927377],
       [-0.05388842, -0.02343061, -0.44759049, …, -0.37976303,
        -0.50456491, -0.18927377],
       [-0.05459671, -0.08770947, -0.41246375, …,  0.18440487,
        -0.54478787, -0.18927377],
       …,
       [ 0.03804261, -0.02543197, -0.11927161, …, -0.37976303,
        -0.81522795, -0.18927377],
       [-0.05477093, -0.09041719, -0.465358  , …, -0.09767908,
         3.37351113, -0.18927377],
       [-0.05460322, -0.08129336, -0.43211783, …, -0.37976303,
        -0.73617363, -0.18927377]])
```

[5]:
```
## Step 7: Principal Component Analysis
# Fit PCA to the standardized data and determine the number of components by␣
 ↪plotting the explained variance ratio for each component and select the␣
 ↪number of components that explain at least 90% of the variance.
pca = PCA(n_components=12)
X_pca = pca.fit_transform(X_scaled)

# Explained variance per component
explained_variance_ratio = pca.explained_variance_ratio_

# Plot explained variance per component
plt.figure(figsize=(10, 4))
plt.bar(range(1, len(explained_variance_ratio) + 1), explained_variance_ratio,␣
 ↪alpha=0.7, color='skyblue')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance Ratio')
plt.title('Explained Variance per Principal Component')
plt.xticks(range(1, len(explained_variance_ratio) + 1))
plt.grid(True, axis='y', linestyle='--', alpha=0.5)
plt.show()
```

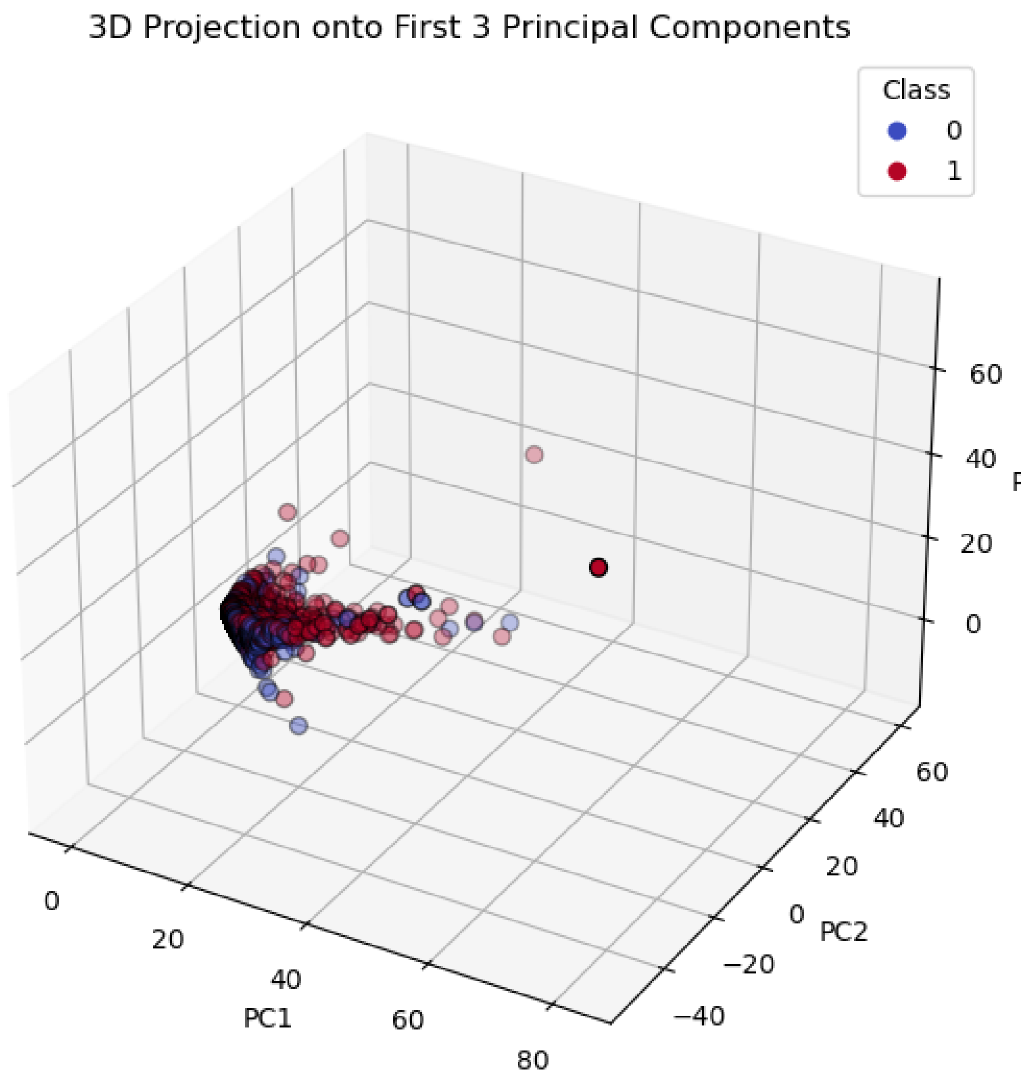


0.1 Explain choice for number of components chosen:

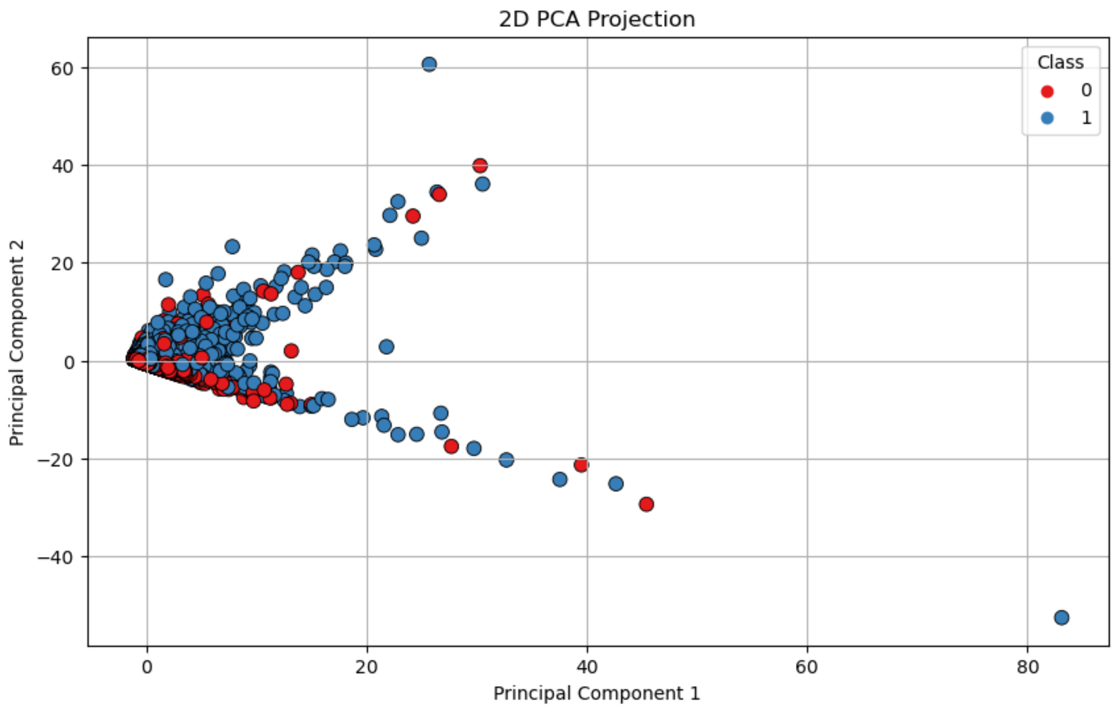
The first 9 components explain just about 90% of the variance so I will use those 9 components to transform the scaled features.

[6]:
```
## Use the selected number of components to transform the scaled features.
# Use the number of components needed for 90% variance
pca_9 = PCA(n_components=9)
X_pca_9 = pca_9.fit_transform(X_scaled)
```

[7]:
```
## Data visualized using first 3 components
pca_3d = PCA(n_components=3)
X_pca_3d = pca_3d.fit_transform(X_scaled)

# Create a 3D scatter plot
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

# You can color by known labels or by clusters if you've run KMeans
scatter = ax.scatter(
    X_pca_3d[:, 0], X_pca_3d[:, 1], X_pca_3d[:, 2],
    c=y,  # or use cluster_labels
    cmap='coolwarm', edgecolor='k', s=40
)

ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('3D Projection onto First 3 Principal Components')
plt.legend(*scatter.legend_elements(), title="Class")
plt.show()
```



[8]:
```
## Data visualized by first 2 components
pca_2d = PCA(n_components=2)
X_pca_2d = pca_2d.fit_transform(X_scaled)

plt.figure(figsize=(10, 6))
sns.scatterplot( x=X_pca_2d[:, 0], y=X_pca_2d[:, 1], hue=y,  palette='Set1',␣
  ↪s=60,edgecolor='k')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('2D PCA Projection')
plt.legend(title='Class')
plt.grid(True)
plt.show()
```



[9]:
```
## Steps 8, 9 & 10: Split the transformed dataset (after PCA was applied) into␣
↪training and testing. Train the logistic regression classifier by using the␣
↪training dataset by using 5-fold cross validation.
# Report f1 and accuracy scores across folds
X_train, X_test, y_train, y_test = train_test_split(X_pca_9, y, test_size=0.2)

log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)

y_pred = log_reg.predict(X_test)

f1_scores_logreg = cross_val_score(log_reg, X_train, y_train, cv=5,␣
↪scoring='f1_weighted')

# Print F1 scores across folds and mean F1 score
print("Logistic Regression F1 scores across folds:", f1_scores_logreg)
print("Logistic Regression mean F1 Score:", f1_scores_logreg.mean())

# Perform cross-validation and get Accuracy scores
accuracy_scores_logreg = cross_val_score(log_reg, X_train, y_train, cv=5,␣
↪scoring='accuracy')
```

```
# Print accuracy scores across folds and mean accuracy score
print("Logistic Regression accuracy scores across folds:",␣
 ↪accuracy_scores_logreg)
print("Logistic Regression mean accuracy score:", accuracy_scores_logreg.mean())
```

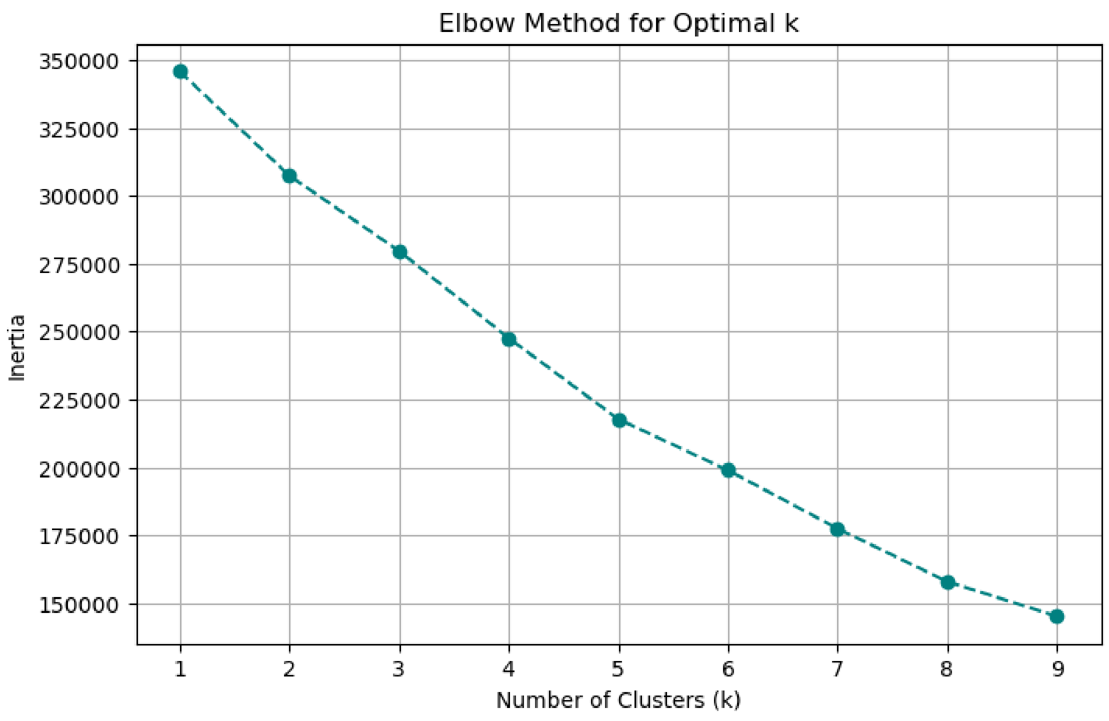
```
Logistic Regression F1 scores across folds: [0.72632613 0.72214491 0.72619668
0.72981471 0.73280022]
Logistic Regression mean F1 Score: 0.7274565306106957
Logistic Regression accuracy scores across folds: [0.77857298 0.77770549
0.78095858 0.77944047 0.78329718]
Logistic Regression mean accuracy score: 0.7799949380689675
```

[10]:
```
## Step 11
from sklearn.metrics import accuracy_score, f1_score
test_accuracy = accuracy_score(y_test, y_pred)
test_f1 = f1_score(y_test, y_pred, average='weighted')  # Weighted for␣
 ↪multi-class

print("Test Accuracy of logistic regression model:" + str(test_accuracy))
print("Test F1 Score of logistic regression model:"  + str(test_f1))
```

```
Test Accuracy of logistic regression model:0.7734212352532963
Test F1 Score of logistic regression model:0.722255556890736
```

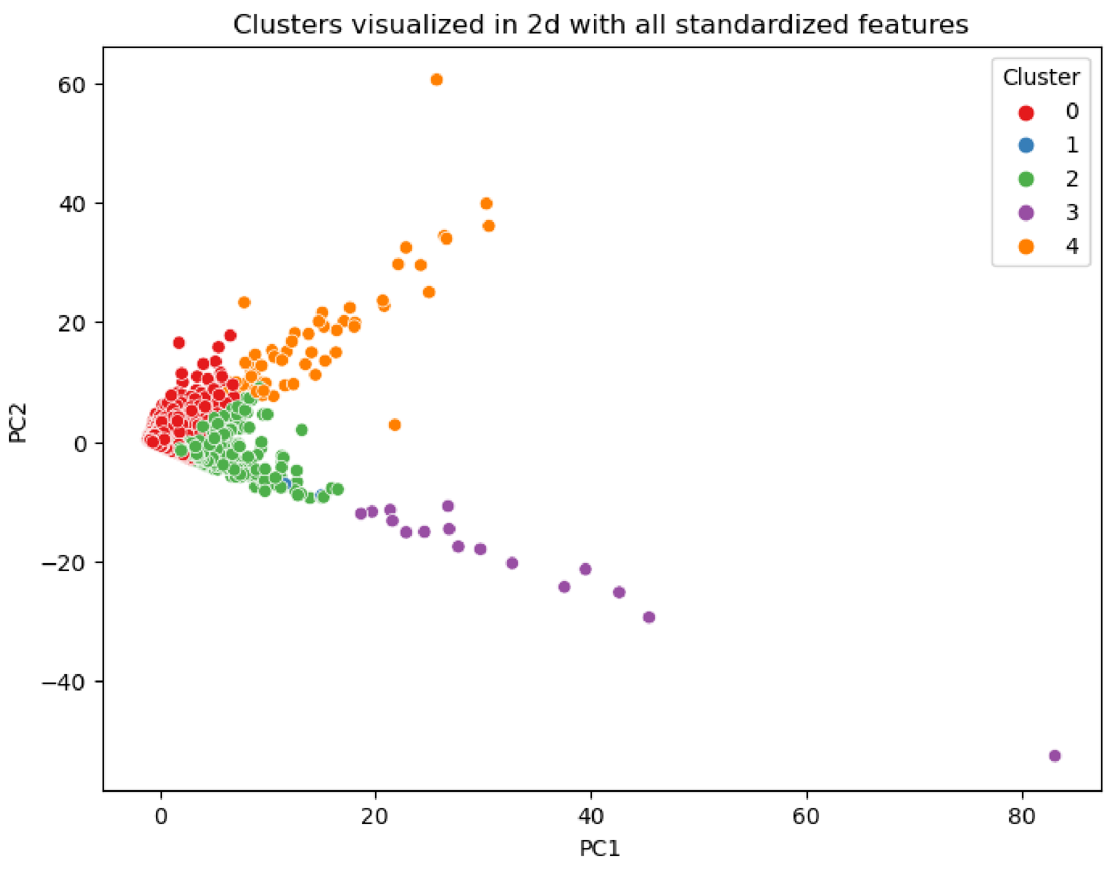
[15]:
```
## Step 12: K-means
# Perform K-Means clustering on the standardized dataset (all the features) and␣
 ↪plot inertia for k values 1-10.
inertias = []
k_range = range(1, 10)

for k in k_range:
    kmeans = KMeans(n_clusters=k, n_init=10, random_state=42)
    kmeans.fit(X_scaled)
    inertias.append(kmeans.inertia_)
```

[16]:
```
## Using Elbow method, determine the sufficient number of clusters
plt.figure(figsize=(8, 5))
plt.plot(k_range, inertias, marker='o', linestyle='--', color='teal')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.xticks(k_range)
plt.grid(True)
plt.show()
```



[17]:
```
## Fit K-means one more time using the selected K.
final_kmeans = KMeans(n_clusters=5)
cluster_labels = final_kmeans.fit_predict(X_scaled)

# 2D PCA for visualization
X_2D = X_pca[:, :2]

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_2D[:, 0], y=X_2D[:, 1], hue=cluster_labels, palette='Set1')
plt.title('Clusters visualized in 2d with all standardized features')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.legend(title='Cluster')
plt.show()
```

```
/opt/conda/envs/anaconda-ai-2024.04-py310/lib/python3.10/site-
packages/sklearn/cluster/_kmeans.py:1412: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



```
[18]: # Get cluster centers in standardized feature space
      cluster_centers = final_kmeans.cluster_centers_

      # Convert to a DataFrame for readability
      centers_df = pd.DataFrame(cluster_centers, columns=X.columns)
      centers_df
```

```
[18]:    Followers     Friends  Num_tweets  Verified  Listed_count  Location  \
      0  -0.045742   -0.016632   -0.050327 -0.226730     -0.056177 -0.013668
      1   3.109168  103.426838    1.212870  4.410533      0.722334  0.565462
      2   0.507678    0.169177    0.931918  4.333857      0.709330  0.275189
      3  31.541856    0.646218    4.778221  4.410533     32.608342 -0.163890
      4   0.555388    0.038089    0.019111  1.521746      0.434076 -0.123238

              Age     Length  Num_users  Num_author_replies  TOXICITY_x  \
      0 -0.069212  -0.053639  -0.059305            0.005137    0.020345
      1  1.581875   0.136902   0.191280            0.184405   -0.706645
      2  1.319778   0.335409   0.446217           -0.212865   -0.387563
      3  1.987090   0.987992   1.054574           -0.273982   -0.408459
      4  0.361485  16.135633  16.129802            2.616801   -0.153316
```

```
   Num_author_toxic_replies
0                  0.008292
1                 -0.189274
2                 -0.165337
3                 -0.189274
4                  0.108455
```

0.2 Describe observations taken from each cluster

Cluster 0: - Centered around average values across most features. - Above average initial tweet toxicity, but the conversations were short lived and did not see a ton of engagement compared to other clusters - Negative verified coefficient suggests most accounts in this cluster are not verified and do not get a ton of engagement in general most likely.

Cluster 1: - Higher values for Followers, Verified, and friends. - Low toxicity of the original tweet in these conversations - Could be some influencers, other popular figures who try to avoid public toxicity in this cluster as they have above average followers and friends and have low toxicity ratings for the original tweet

Cluster 2: - Also contains verified users similar to cluster 1 - Higher toxicity from the original tweet than cluster 1, which is likely what led to more engagement in the conversations for this cluster than the previous even with very similar coefficients for other features. - Probably verified users with not a lot of followers, relatively low toxicity ratings for original tweet also likely due to there being verified users in the cluster who try and stay away from public toxicity

Cluster 3: - Extreme outliers for Followers and Listed_count, showing there is likely a lot of famous people and influencers who have a lot of followers, original users also have a lot more total tweets posted on their accounts in this cluster showing how the users in this cluster are more active on twitter than other clusters - The status of the original users can likely help explain the above average engagement in the conversations in this cluster

Cluster 4: - Extremely long conversations with lots of users - Original user's do not have a ton of followers, these conversations were likely controversial topics or events that recieved a lot of engagement even without as many influencers or other popular people online who are verified and have more followers as they have very high lengths and number of users. - Slightly higher author toxic replies, the only cluster with a positive value here, showing the original user contributed more toxicity to the conversations than other clusters and there were more author replies in general, which is likely due to all of the engagement these conversations recieved