```
In [86]:
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.decomposition import PCA
    from sklearn import svm
    from sklearn.pipeline import Pipeline
    from sklearn.model_selection import cross_val_score,train_test_split, GridSear chCV
    from sklearn.feature_selection import RFE
    from sklearn.naive_bayes import GaussianNB
    import warnings
    warnings.simplefilter('ignore', category=FutureWarning)
```

The data used here is for AIDS detection. we have 4 files with different sizes so we will clean the data by the wrangle function where we will check for *null* and replace them with the mean of the values. Then, we will combine them into one single file.

```
In [93]:
          df1 = wrangle('AIDS_Classification.csv')
          the null values =
           time
                       0
          trt
                       0
          age
                       0
          wtkg
                       0
          hemo
          homo
                       0
          drugs
          karnof
                       0
                       0
          oprior
          z30
                       0
                       0
          preanti
          race
                       0
          gender
                       0
          str2
                       0
                       0
          strat
                       0
          symptom
                       0
          treat
          offtrt
                       0
          cd40
          cd420
                       0
          cd80
                       0
          cd820
                       0
          infected
                       0
          dtype: int64
In [94]: df2 = wrangle('AIDS_Classification_5000.csv')
          the null values =
           time
                        0
          trt
                       0
                       0
          age
                       0
          wtkg
          hemo
                       0
          homo
                       0
          drugs
                       0
          karnof
                       0
          oprior
                       0
          z30
                       0
                       0
          preanti
                       0
          race
                       0
          gender
          str2
                       0
          strat
                       0
          symptom
                       0
          treat
                       0
          offtrt
                       0
          cd40
                       0
                       0
          cd420
          cd80
                       0
          cd820
                       0
          infected
          dtype: int64
```

```
In [95]:
          df3 = wrangle('AIDS_Classification_15000.csv')
          the null values =
           time
                       0
          trt
                       0
          age
                       0
          wtkg
                       0
          hemo
          homo
                       0
          drugs
          karnof
                       0
                       0
          oprior
          z30
                       0
                       0
          preanti
                       0
          race
          gender
                       0
          str2
                       0
                       0
          strat
                       0
          symptom
                       0
          treat
          offtrt
                       0
          cd40
          cd420
                       0
          cd80
                       0
          cd820
                       0
          infected
                       0
          dtype: int64
In [96]: df4 = wrangle('AIDS_Classification_50000.csv')
          the null values =
           time
                        0
          trt
                       0
                       0
          age
                       0
          wtkg
          hemo
                       0
          homo
                       0
          drugs
                       0
          karnof
                       0
          oprior
                       0
          z30
                       0
                       0
          preanti
                       0
          race
                       0
          gender
          str2
                       0
          strat
                       0
          symptom
                       0
          treat
                       0
          offtrt
                       0
          cd40
                       0
                       0
          cd420
          cd80
                       0
          cd820
                       0
          infected
          dtype: int64
```

```
In [10]: # Merge all dataframes into one
    df = pd.concat([df1, df2, df3, df4])

# Export merged dataframe to a file
    df.to_csv('merged_data.csv', index=False)
```

using info function we notice that our data consists of integers and float values.

```
In [11]: | df.info()
         <class 'pandas.core.frame.DataFrame'>
         Index: 72139 entries, 0 to 49999
         Data columns (total 23 columns):
          #
              Column
                        Non-Null Count Dtype
              time
                                        int64
          0
                        72139 non-null
          1
              trt
                        72139 non-null
                                        int64
          2
              age
                        72139 non-null
                                        int64
          3
                                        float64
              wtkg
                        72139 non-null
          4
              hemo
                        72139 non-null
                                        int64
          5
              homo
                        72139 non-null
                                        int64
              drugs
          6
                        72139 non-null
                                        int64
          7
              karnof
                        72139 non-null
                                        int64
          8
              oprior
                        72139 non-null
                                        int64
          9
              z30
                        72139 non-null
                                        int64
          10
              preanti
                        72139 non-null
                                        int64
          11
              race
                        72139 non-null
                                        int64
          12 gender
                        72139 non-null
                                        int64
          13
              str2
                        72139 non-null
                                        int64
             strat
                        72139 non-null
          14
                                        int64
          15
              symptom
                        72139 non-null
                                        int64
          16 treat
                        72139 non-null
                                        int64
          17 offtrt
                        72139 non-null
                                        int64
          18 cd40
                        72139 non-null
                                        int64
          19 cd420
                        72139 non-null
                                        int64
          20 cd80
                        72139 non-null
                                        int64
          21 cd820
                        72139 non-null
                                        int64
          22 infected 72139 non-null
                                        int64
         dtypes: float64(1), int64(22)
         memory usage: 13.2 MB
```

Checking for any duplicates values

```
In [91]: print(f"Total number of duplicate rows: {df.duplicated().sum()}")

Total number of duplicate rows: 0
```

```
df.head()
In [57]:
Out[57]:
                                 wtkg hemo homo drugs karnof oprior z30 ... str2 strat symptom tr
               time
                     trt age
                      2
                                                                              0
            0
                948
                          48
                              89.8128
                                           0
                                                   0
                                                          0
                                                                100
                                                                                             1
                                                                                                        0
                                                   0
            1
               1002
                      3
                              49.4424
                                           0
                                                          0
                                                                90
                                                                         0
                                                                                             3
                                                                                                        0
                          61
                                                                              1
                                                                                       1
                961
                      3
                              88.4520
                                                          1
                                                                90
                                                                                             3
                                                                                                        0
                                                   1
```

5 rows × 23 columns

3

0

47

43

85.2768

66.6792

0

3

symptom -

correlated.

1166

1090

Using matplotlib's heatmap to understand the correlation between the values. We notice that the data is not that

1

1

0

0

100

100

0

0

1 -0.036 -0.0055 0.037 -0.0046 0.025

1 0.015 -0.029

0.04 0.037 -0.029 0.16

3

3

0 0

```
In [53]:
                   plt.figure(figsize=(20, 10))
                    sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
                   plt.title('Correlation Heatmap')
                   plt.show()
                                                                                     Correlation Heatmap
                             0.048 0.0082 0.039 0.013 0.056 0.011 0.02 0.019 0.075 0.053 0.059 0.051 0.076 0.067 0.021 0.067 0.11 0.046 0.097 0.018 0.024 0.12
                                   0.042 0.0087 0.0016 0.066 0.0054 0.039 0.0039 0.0032 0.0051 0.062 0.041 0.0035 0.0078 0.018 0.25 0.029 0.013 0.028 0.0031 0.016 0.047
                                         0.01 -0.027 0.11 0.0045 0.049 0.014 0.12 0.098 -0.075 0.041 0.12 0.11 0.0045 0.043 -0.05 -0.052 -0.069 0.00610.00098 0.03
                        age - 0.0082 0.042
                                               1 -0.036 0.092 -0.016 0.013 0.00067 -0.098 -0.063 -0.046 0.074 -0.1 -0.087 0.0022 -0.0018 -0.03 0.045 0.062 0.021 -0.0028 -0.048
                       wtkg - 0.039 0.0087 0.01
                                                                                                                                                                               0.8
                                                    1 -0.082 -0.002-0.00042 0.011 0.065 0.049 0.0046 -0.013 0.065 0.068 -0.0062 0.0066 0.014 -0.031 -0.039 -0.011 -0.0093 0.024
                              0.013 0.0016 -0.027
                             0.056 0.066 0.11 0.092 0.082 1 0.013 0.011 0.095 0.017 0.019 0.21 0.00061 0.015 0.024 0.074 0.1 0.0017 0.013 0.029 0.0015 0.0014
                             0.011 0.0054 0.0045 0.016 0.002 0.058 1 0.0089 0.0096 0.033 0.036 0.049 0.033 0.039 0.048 0.0049 0.012 0.039 0.023 0.025 0.0042 0.0086 0.027
                             0.02 -0.039 -0.049 0.013 -0.00042 -0.013 0.0088 1 -0.0014 -0.12 -0.087 0.036 0.00083 -0.13 -0.13 -0.00097 -0.055 -0.028 0.046 0.063 0.021 0.012 -0.025
                             0.019 -0.0039 0.014 0.00067 0.011 0.011 -0.0096-0.0014 1 0.056 0.044 -0.02 0.0037 0.065 0.069 0.02 -0.013 -0.011 -0.04 -0.042 0.0042 -0.011 0.043
                        230 - 0.075 0.0032 0.12 -0.098 0.065 -0.0095 -0.033 -0.12 0.056 1 0.43 -0.093 -0.023 0.62 0.58 0.02 0.017 -0.0085 0.21 0.28 -0.019 0.0021 0.24
                      preanti - 0.053 0.0051 0.098 0.063 0.049 0.017 0.036 0.087 0.044 0.43 1 0.091 9.3e-05 0.46 0.48 0.024 0.024 0.013 0.17 0.23
                                                                                                                                                                               0.4
                        race - 0.039 0.062 0.075 0.046 0.0046 0.049 0.039 0.036 0.02 0.093 0.091 1 0.14 0.11 0.13 0.015 0.077 0.069 0.026 0.045 0.025 0.00074 0.032
                      gender - 0.051 0.041 0.041 0.044 0.013 0.024 0.013 0.021 0.033 0.00083 0.0037 0.023 9.3e-05 0.14 1 0.022 0.0099 0.0052 0.051 0.081 0.015 0.015 0.016 0.024 0.0094 0.002
                        str2 - 0.076 0.0035 0.12 -0.1 0.065-0.00061-0.039 -0.13 0.065 0.62 0.46 0.11 -0.022 1 0.63 0.035 0.027 -0.0013 -0.23 -0.3
                                                                                                                                                                               0.2
                                                                                             -0.13 -0.0099 0.63 1 0.023 0.021 -0.023 -0.23 -0.31
                       strat - -0.067 0.0078 0.11 -0.087 0.068 0.015 -0.048 -0.13 0.069 0.58 0.48
                                                                                                                                                  0.021-0.00015 0.25
                             0.021 0.018 0.0045 0.0022 0.0062 0.024 0.0049 0.00097 0.02 0.024 0.015 0.0052 0.035 0.023 1 0.033 0.0026 0.021 0.04 0.0049 0.022 0.029
```

0.067 0.25 0.043 -0.0018 0.0066 0.074 -0.012 -0.055 -0.013 0.017 0.024 -0.077 0.051 0.027 0.021 -0.033 -0.11 -0.029 -0.05 -0.03 0.014 -0.1 0.039 -0.028 -0.011 -0.0085 -0.013 0.069 -0.081 -0.0013 -0.023 0.0026 -0.036

cd80 - 0.018 -0.0031 0.0061 0.021 -0.011 0.029 -0.0042 0.021 0.0042 -0.019 -0.022 -0.025 0.024 -0.026 -0.021 0.0049 -0.0046 0.0055 0.015 0.012

0.12 0.047 0.03 0.048 0.024 0.0014 0.027 0.029 0.043 0.24 0.07 0.029 0.043 0.24 0.07 0.029 0.053 0.029 0.053 0.025 0.04 0.018 0.018

age wtkg hemo homo drugs karnof oprior 230 preanti race gender str2 strat symptom treat offtrt cd40 cd420 cd80 cd820 infected

cd40 - 0.046 0.013 -0.052 0.045 -0.031 -0.0017 0.023 0.046 -0.04 -0.21 -0.17 0.026 0.015 -0.23 -0.23

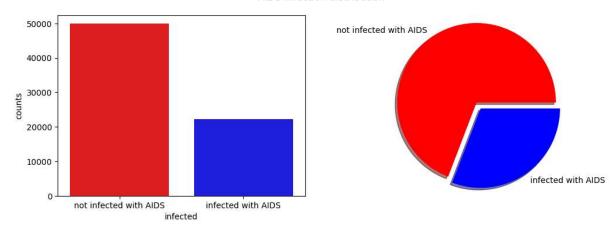
cd420 - 0.097 0.028 -0.069 0.062 -0.039 -0.013 0.025 0.063 -0.042 -0.28 -0.23 0.045 0.016

Using a barplot and pie chart to better understand our target variable

0.0

```
grouped_infect = df.groupby("infected")["infected"].count().reset_index(name
In [68]:
         ="counts")
         plt.figure(figsize=(12,4))
         plt.subplot(1,2,1)
         sns.barplot(x=grouped infect.infected, y=grouped infect.counts, palette=["re
         d", "blue"])
         plt.xticks(ticks=(0,1), labels=["not infected with AIDS", "infected with AID
         S"])
         plt.subplot(1,2,2)
         plt.pie(x=grouped_infect.counts, shadow=True,
                 labels=["not infected with AIDS", "infected with AIDS"], colors=["re
         d", "blue"],
                 radius=1.4-0.3, explode= [0.05,0.05])
         plt.suptitle("AIDS infection distribution")
         plt.show()
```

AIDS infection distribution



Corr function helps understand correlation value between variables and the target variable. sorting them shows the highest correlation is 2.47557 which is a low value.

```
df.corr()['infected'].sort_values(ascending=False)
In [69]:
Out[69]: infected
                      1.000000
         strat
                      0.247557
         z30
                      0.235942
         str2
                      0.234536
                      0.174728
         preanti
         oprior
                      0.043151
                      0.030119
         age
         symptom
                      0.028621
         hemo
                      0.023914
         homo
                     -0.001416
         cd80
                     -0.008919
         cd820
                     -0.017689
                     -0.020199
         gender
         offtrt
                     -0.025446
         drugs
                     -0.027247
         karnof
                     -0.029347
         race
                     -0.032233
         trt
                     -0.046882
         wtkg
                     -0.048424
         treat
                     -0.052980
         time
                     -0.115005
         cd40
                     -0.142910
         cd420
                     -0.186166
         Name: infected, dtype: float64
```

building our ML model using two ways: directly and with a pipeline

Directly:

set target to be infected

```
In [40]: y = df['infected']
X = df.drop('infected', axis=1)
```

we are using PCA fpr dimensionality reduction, using the variance threshold method with variance 0.95 and higher.

This equates to lambda value of 3

```
In [35]: # Create an instance of PCA with the desired threshold
    pca = PCA(n_components=0.95)

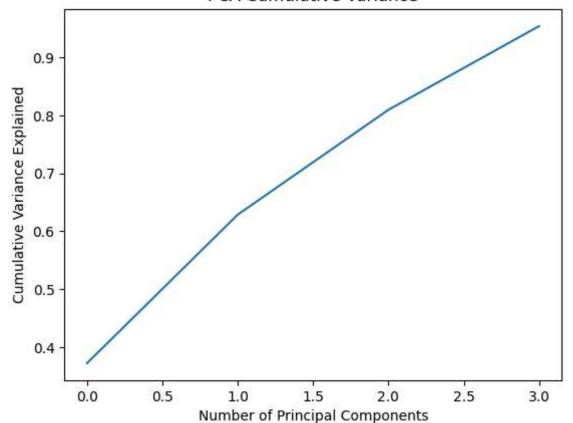
# Fit the PCA model to the data
    pca.fit(X)

# Transform the data to the reduced-dimensional representation
    X = pca.transform(X)

cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
    import matplotlib.pyplot as plt

plt.plot(cumulative_variance)
    plt.xlabel('Number of Principal Components')
    plt.ylabel('Cumulative Variance Explained')
    plt.title('PCA Cumulative Variance')
    plt.show()
```

PCA Cumulative Variance



Test size is 20% of the data and the rest for training. SVC (short for Support Vector Classifier) works by mapping data points to a high-dimensional space and then finding the optimal hyperplane that divides the data into two classes.

```
In [38]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=42)

clf = svm.SVC()

# Fit the classifier to the training data
clf.fit(X_train, y_train)

# Predict the target variable for the testing data
y_pred = clf.predict(X_test)

# Evaluate the performance of the classifier
accuracy = clf.score(X_test, y_test)
```

Final test accuracy is 69.2%

Pipeline:

```
In [19]: y = df['infected']
X = df.drop('infected', axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando m_state=42)

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)

X_train shape: (50497, 22)
y_train shape: (50497,)
X_test shape: (21642, 22)
y_test shape: (21642,)
```

We create a pipeline containing the PCA and SVC then we pass the pipeline to the cross_val_score function with cv set to 5 meaning that the training data is spit into 5 subsets, and the model is trained and evaluated 5 times, each time training on 4 subsets and evaluating on the remaining subset.

the accuracy of each step = $[0.69138614 \ 0.69217822 \ 0.69234578 \ 0.69422715 \ 0.69254382]$

Out[24]: <Axes: >

