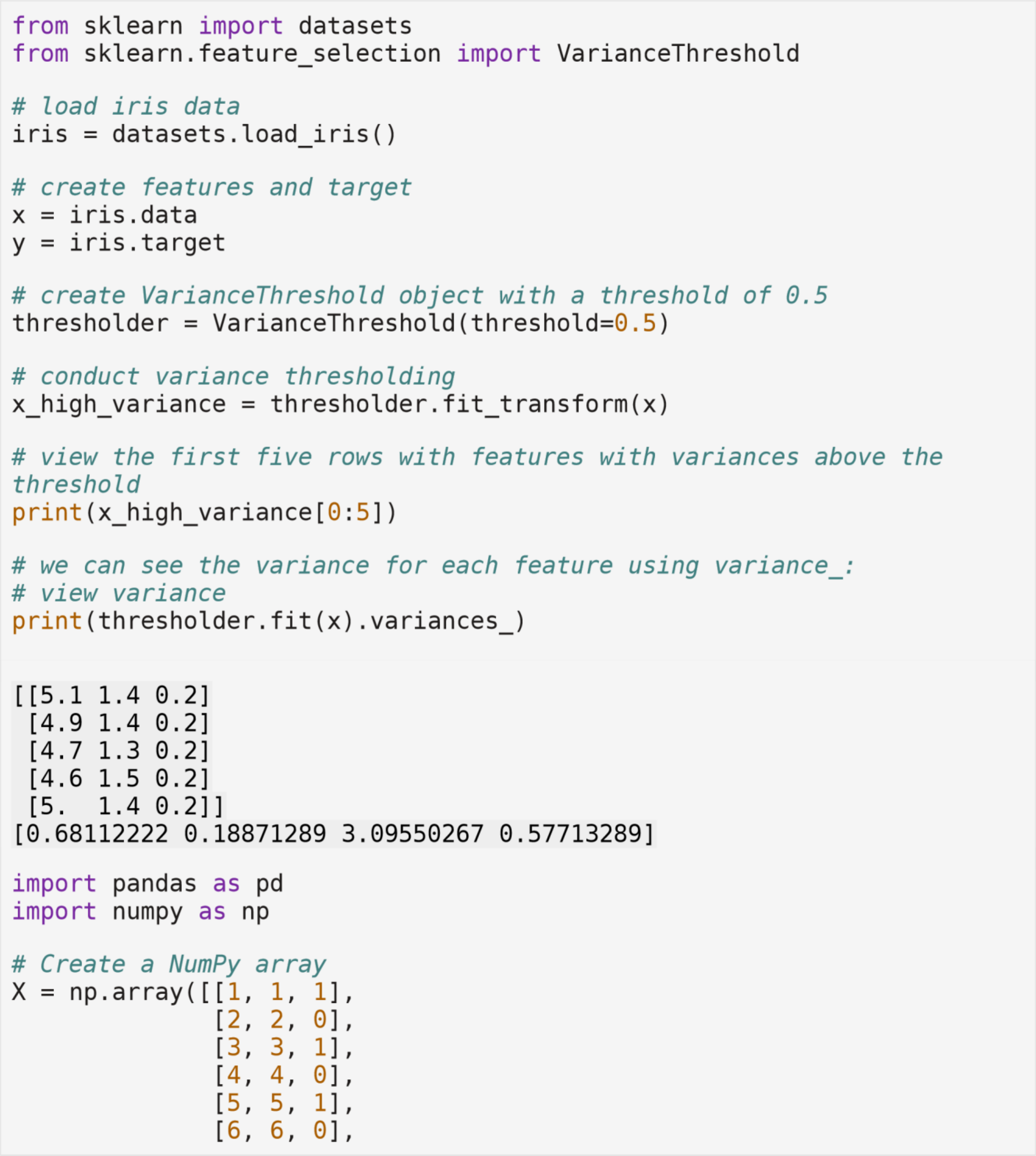
Dimensionality reduction using Feature selection



```
from sklearn import datasets
from sklearn.feature_selection import VarianceThreshold

# load iris data
iris = datasets.load_iris()

# create features and target
x = iris.data
y = iris.target

# create VarianceThreshold object with a threshold of 0.5
thresholder = VarianceThreshold(threshold=0.5)

# conduct variance thresholding
x_high_variance = thresholder.fit_transform(x)

# view the first five rows with features with variances above the
threshold
print(x_high_variance[0:5])

# we can see the variance for each feature using variance_:
# view variance
print(thresholder.fit(x).variances_)
```

```
[[5.1 1.4 0.2]
 [4.9 1.4 0.2]
 [4.7 1.3 0.2]
 [4.6 1.5 0.2]
 [5.  1.4 0.2]]
[0.68112222 0.18871289 3.09550267 0.57713289]
```

```
import pandas as pd
import numpy as np

# Create a NumPy array
X = np.array([[1, 1, 1],
              [2, 2, 0],
              [3, 3, 1],
              [4, 4, 0],
              [5, 5, 1],
              [6, 6, 0],
              [7, 7, 1],
              [8, 8, 0],
              [9, 9, 1]])

# Convert the features matrix into a DataFrame
df = pd.DataFrame(X)

# View the DataFrame
print(df)
   0  1  2
0  1  1  1
1  2  2  0
2  3  3  1
3  4  4  0
4  5  5  1
5  6  6  0
6  7  7  1
7  8  8  0
8  9  9  1
corr_matrix = df.corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))

# Find index of features columns with correlation greater than 0.95
to_drop = [column for column in upper.columns if any(upper[column] >
0.95)]

# Drop features
df = df.drop(df[to_drop], axis=1)

# correlation matrix
df.corr()
     0    2
0  1.0  0.0
2  0.0  1.0
# upper triangle of correlation matrix
upper
    0    2
0 NaN  0.0
2 NaN  NaN
# drop features
df.drop(df[to_drop], axis=1)
```

```
   0  2
0  1  1
1  2  0
2  3  1
3  4  0
4  5  1
5  6  0
6  7  1
7  8  0
8  9  1
```

```
# load libraries
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

# load iris data
irirs = load_iris()

# create features and  target
x = iris.data y = iris.target

# convert o fetaures data by converting data  to integers
X = x.astype(int)

# select two features with highest chi-squared statistics
chi2_selector =SelectKBest(chi2, k=2) x_kbest =
chi2_selector.fit_transform(x, y)

# show results
print('original number of features:', X.shape[1])
print('reduced number of features:', x_kbest.shape[1])
```

```
original number of features: 4
reduced number of features: 2
```

```
# load libraries
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

# load iris data
iris = load_iris()

#create features and target
x = iris.data y =
iris.target

#create an SelectKbBest object to select features with best anova
Fvalues
```

```
fvalue_selector = SelectKBest(f_classif, k=2)

# apply the SelectKBest object to the features and target
x_kbest = fvalue_selector.fit_transform(x, y)

# show results
print('original number of features:', X.shape[1])
print('Reduced number of features:', x_kbest.shape[1])
```

```
original number of features: 4
Reduced number of features: 2
```