```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn import model_selection
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import train_test_split
from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import mean_squared_error
```

```python
data = pd.read_csv("Volumetric_features.csv")
```

```python
data.head()
```

```
   S.No  Left-Lateral-Ventricle  Left-Inf-Lat-Vent  \
0     1                 22916.9              982.7
1     2                 22953.2              984.5
2     3                 23320.4             1062.1
3     4                 24360.0             1000.5
4     5                 25769.4             1124.4

   Left-Cerebellum-White-Matter  Left-Cerebellum-Cortex  Left-Thalamus
\
0                       15196.7                 55796.4         6855.5

1                       15289.7                 55778.6         6835.1

2                       15382.1                 55551.2         7566.0

3                       14805.4                 54041.8         8004.6

4                       16331.1                 54108.6         6677.4


   Left-Caudate  Left-Putamen  Left-Pallidum  3rd-Ventricle  ...  \
0        2956.4        4240.7         2223.9         2034.4  ...
1        3064.2        4498.6         2354.1         1927.1  ...
2        3231.7        4456.2         1995.4         2064.7  ...
3        3137.3        4262.2         1983.4         2017.7  ...
4        2964.4        4204.6         2409.7         2251.8  ...

   rh_supramarginal_thickness  rh_frontalpole_thickness  \
0                       2.408                     2.629
1                       2.417                     2.640
2                       2.374                     2.601
3                       2.366                     2.639
4                       2.381                     2.555

   rh_temporalpole_thickness  rh_transversetemporal_thickness  \
0                      3.519                            2.009
```

```
1                       3.488                                 2.111
2                       3.342                                 2.146
3                       3.361                                 2.056
4                       3.450                                 2.052

   rh_insula_thickness  rh_MeanThickness_thickness
BrainSegVolNotVent.2  \
0                2.825                     2.33635
1093846
1                2.720                     2.34202
1099876
2                2.684                     2.31982
1097999
3                2.700                     2.29215
1070117
4                2.574                     2.30397
1075926

        eTIV.1  Age  dataset
0  1619602.965   85        1
1  1624755.130   85        1
2  1622609.518   86        1
3  1583854.236   87        1
4  1617375.362   89        1

[5 rows x 141 columns]
```

```
y = data['Left-Lateral-Ventricle'].values
X = data.values[:, 1:]
```

```
y.shape
```

```
(4226,)
```

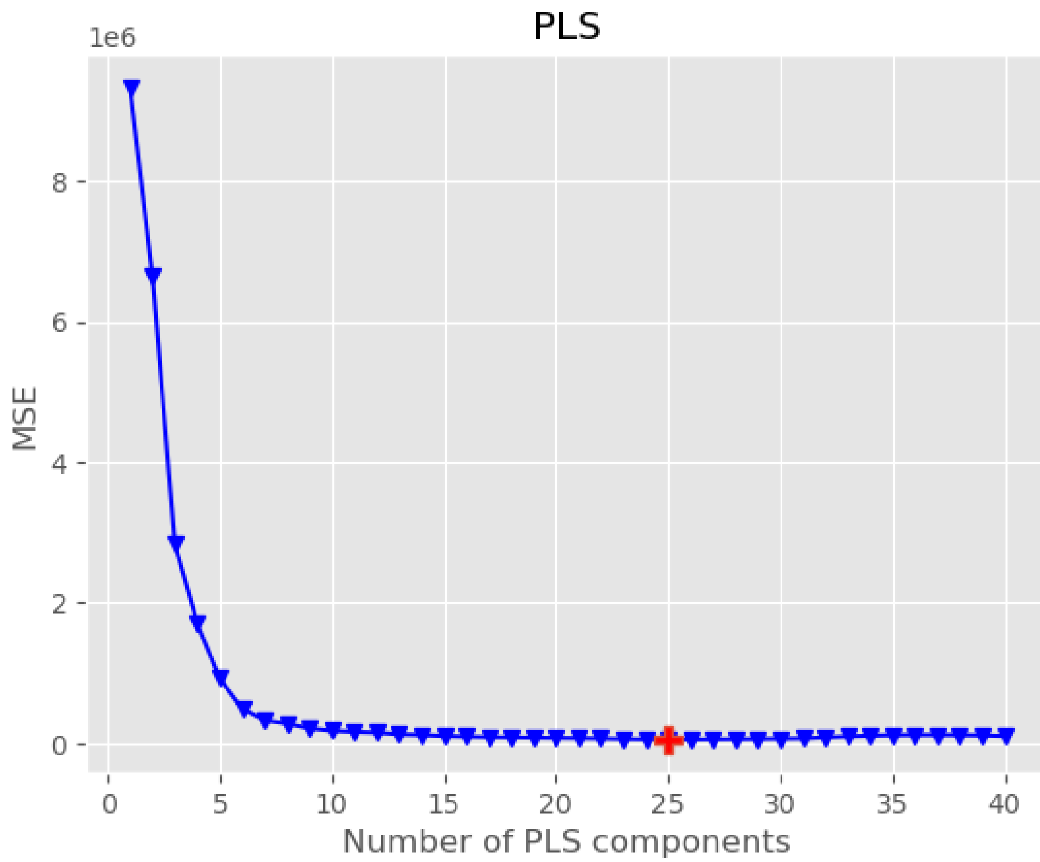
```
X.shape
```

```
(4226, 140)
```

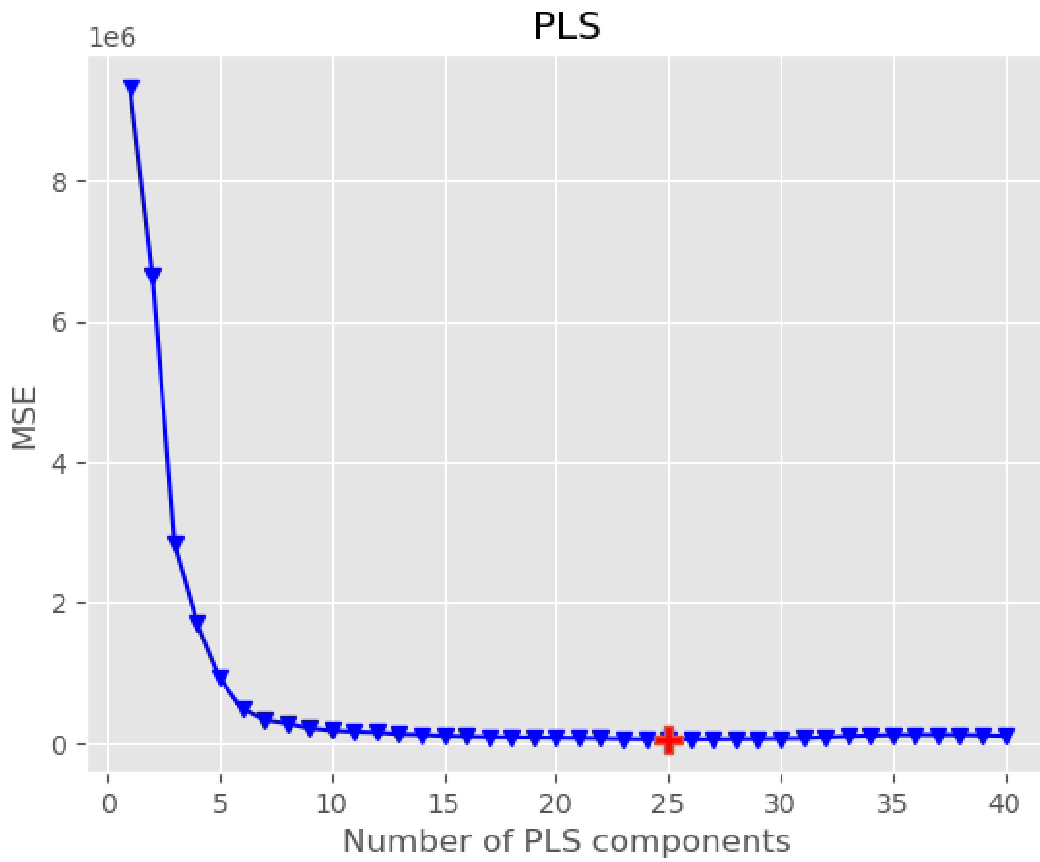
```
X2 = savgol_filter(X, 17, polyorder=2, deriv=2)
```

```
def optimise_pls_cv(X, y, n_comp):
    # Define PLS object
    pls = PLSRegression(n_components=n_comp)

    # Cross-validation
    y_cv = cross_val_predict(pls, X, y, cv=10)

    # Calculate scores
    r2 = r2_score(y, y_cv)
    mse = mean_squared_error(y, y_cv)
    rpd = y.std()/np.sqrt(mse)
```

```python
    return (y_cv, r2, mse, rpd)

r2s = []
mses = []
rpds = []
xticks = np.arange(1, 41)
for n_comp in xticks:
    y_cv, r2, mse, rpd = optimise_pls_cv(X2, y, n_comp)
    r2s.append(r2)
    mses.append(mse)
    rpds.append(rpd)

def plot_metrics(vals, ylabel, objective):
    with plt.style.context('ggplot'):
        plt.plot(xticks, np.array(vals), '-v', color='blue',
mfc='blue')
        if objective=='min':
            idx = np.argmin(vals)
        else:
            idx = np.argmax(vals)
        plt.plot(xticks[idx], np.array(vals)[idx], 'P', ms=10,
mfc='red')

        plt.xlabel('Number of PLS components')
        plt.xticks = xticks
        plt.ylabel(ylabel)
        plt.title('PLS')

    plt.show()

plot_metrics(mses, 'MSE', 'min')
```




```
plot_metrics(mses, 'MSE', 'min')
```

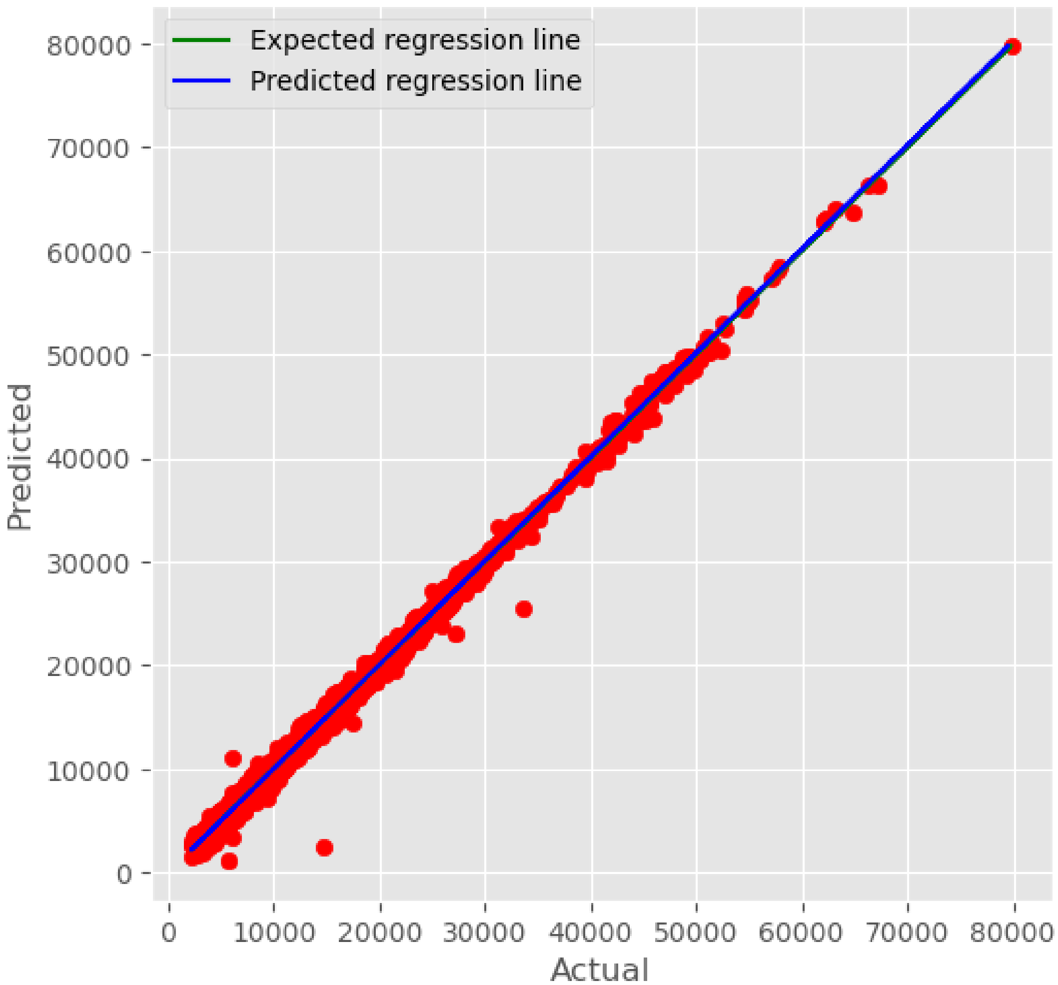



```
y_cv, r2, mse, rpd = optimise_pls_cv(X2, y, 7)
```

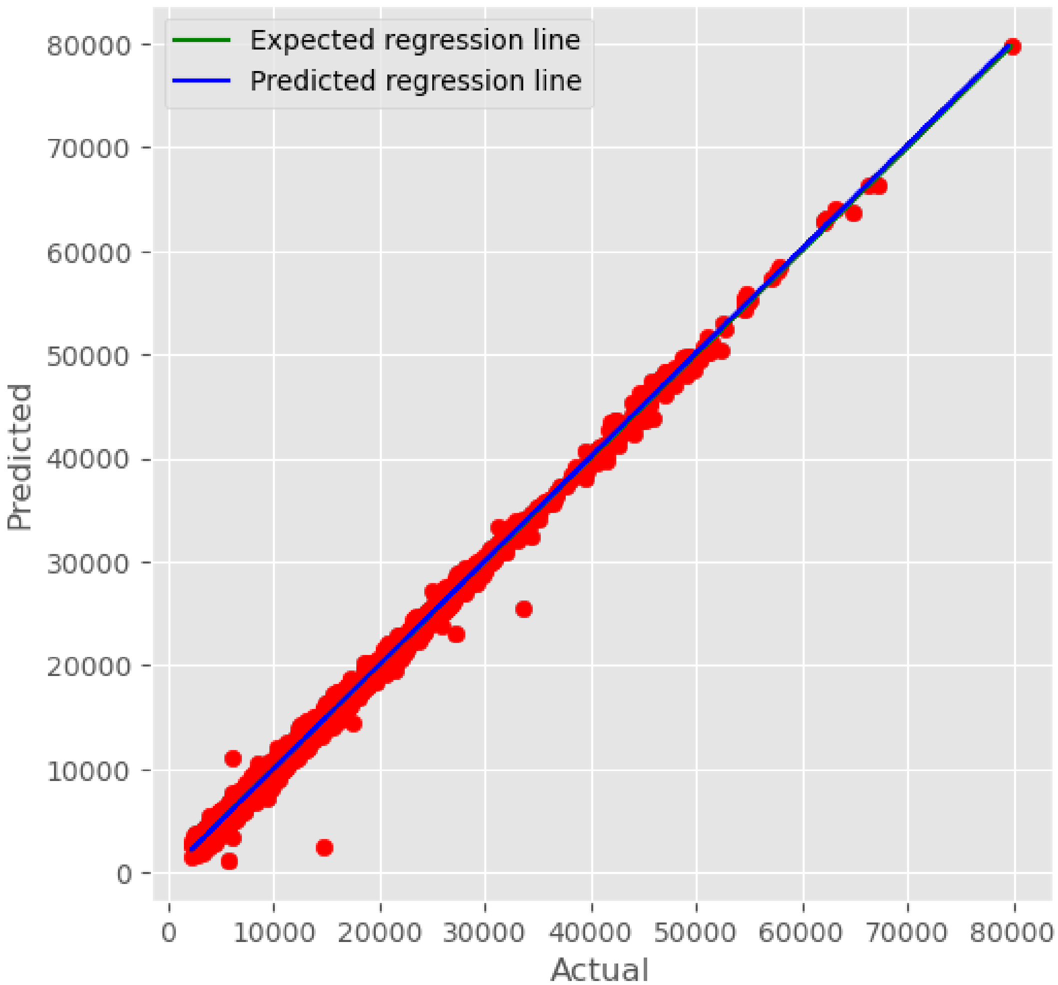
```
print('R2: %0.4f, MSE: %0.4f, RPD: %0.4f' %(r2, mse, rpd))
```

```
R2: 0.9961, MSE: 325886.1622, RPD: 16.1051
```

```
plt.figure(figsize=(6, 6))
with plt.style.context('ggplot'):
    plt.scatter(y, y_cv, color='red')
    plt.plot(y, y, '-g', label='Expected regression line')
    z = np.polyfit(y, y_cv, 1)
    plt.plot(np.polyval(z, y), y, color='blue', label='Predicted
regression line')
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.legend()
    plt.plot()
    plt.show()
```



Expected regression line
Predicted regression line
80000
70000
60000
50000
40000
30000
20000
10000
0
Predicted
0
10000
20000
30000
40000
50000
60000
70000
80000
Actual


Expected regression line
Predicted regression line
80000
70000
60000
50000
40000
30000
20000
10000
0
Predicted
0
10000
20000
30000
40000
50000
60000
70000
80000
Actual