# Institute of Technology of Cambodia (ITC)

## Master of Data Science

Subject: Programming for Data Science

Outliers Detection of Programming for Data Science

Name: ENG KHUN

ID: M080101

# Table of Contents



In [2]:

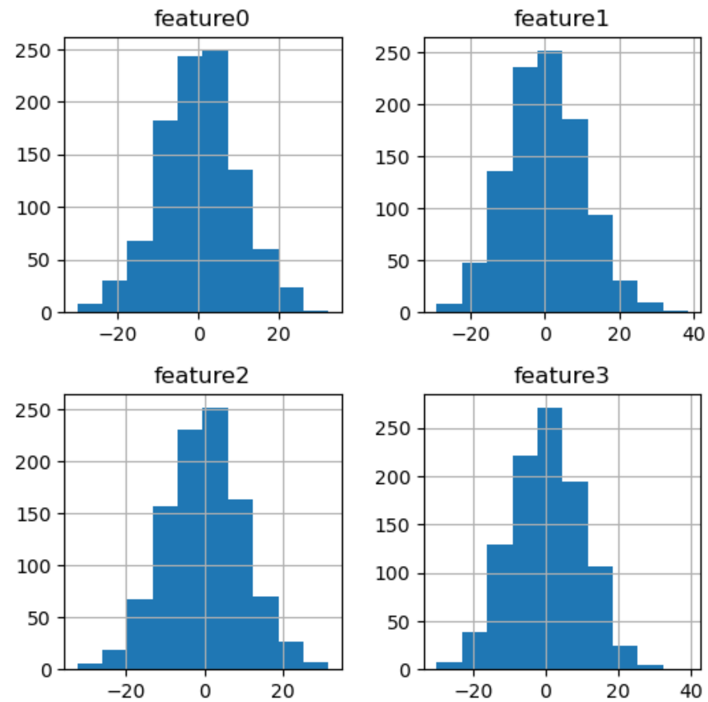
```python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import IsolationForest
from sklearn.covariance import EllipticEnvelope
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.svm import OneClassSVM
from sklearn.neighbors import LocalOutlierFactor
```

# # Parametric methods: Univariate

Create a dummy dataframe where features have **normal distributions** to practice parametric methods.

In [3]:

```python
# seed for reproducibility
np.random.seed(42)
n_feats=4
dummydf = pd.DataFrame(np.random.normal(scale=10.0, size=(1000, n_feats)),
                           columns=['feature{}'.format(i) for i in range(n_feats)])
# all normally distributed
dummydf.hist(figsize=(6,6));
```




In [4]:

```python
# enough variation between features to show outliers
dummydf.describe()
```

Out[4]:

| | feature0 | feature1 | feature2 | feature3 |
|---|---|---|---|---|
| **count** | 1000.000000 | 1000.000000 | 1000.000000 | 1000.000000 |
| **mean** | 0.306239 | 0.248285 | -0.082552 | 0.300861 |
| **std** | 9.639191 | 10.118843 | 10.060754 | 10.069640 |
| **min** | -30.195122 | -28.962554 | -32.412673 | -29.911360 |
| **25%** | -6.129422 | -6.770368 | -6.752986 | -6.708710 |
| **50%** | 0.561874 | 0.202097 | -0.075088 | 0.211577 |
| **75%** | 6.648813 | 6.938808 | 6.422820 | 6.958780 |
| **max** | 32.430930 | 38.527315 | 31.520567 | 39.262377 |

Define two functions that statistically identify outliers in a pandas Series using a standard deviation and interquartile range method.

In [5]:

```python
def out_std(s, nstd=3.0, return_thresholds=False):
    """
    Return a boolean mask of outliers for a series
    using standard deviation, works column-wise.
    param nstd:
        Set number of standard deviations from the mean
        to consider an outlier
    :type nstd: ``float``
    param return_thresholds:
        True returns the lower and upper bounds, good for plotting.
        False returns the masked array
    :type return_thresholds: ``bool``
    """
    data_mean, data_std = s.mean(), s.std()
    cut_off = data_std * nstd
    lower, upper = data_mean - cut_off, data_mean + cut_off
    if return_thresholds:
        return lower, upper
    else:
        return [True if x < lower or x > upper else False for x in s]

def out_iqr(s, k=1.5, return_thresholds=False):
    """
    Return a boolean mask of outliers for a series
    using interquartile range, works column-wise.
    param k:
        some cutoff to multiply by the iqr
    :type k: ``float``
    param return_thresholds:
        True returns the lower and upper bounds, good for plotting.
        False returns the masked array
    :type return_thresholds: ``bool``
    """
    # calculate interquartile range
    q25, q75 = np.percentile(s, 25), np.percentile(s, 75)
    iqr = q75 - q25
    # calculate the outlier cutoff
    cut_off = iqr * k
    lower, upper = q25 - cut_off, q75 + cut_off
    if return_thresholds:
        return lower, upper
    else: # identify outliers
        return [True if x < lower or x > upper else False for x in s]
```

In [6]:

```python
# outlier_mask is a boolean list identifies the indices of the outliers
outlier_mask = out_std(dummydf['feature0'], nstd=3.0)
# first 10 elements
outlier_mask[:10]
```

Out[6]:

```
[False, False, False, False, False, False, False, False, False, False]
```

Identify the outliers, notice these values are on both low and high.

In [7]:

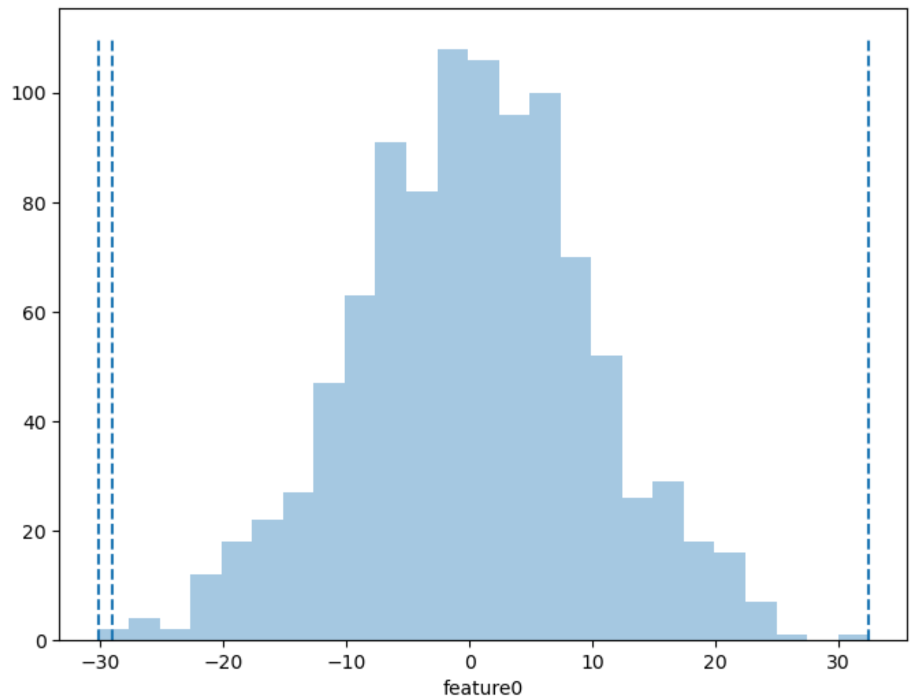
```python
dummydf['feature0'][outlier_mask]
```

Out[7]:

```
506   -30.195122
788   -29.069882
929    32.430930
Name: feature0, dtype: float64
```

In [8]:
```python
plt.figure(figsize=(8,6))
sns.distplot(dummydf['feature0'], kde=False);
plt.vlines(dummydf['feature0'][outlier_mask], ymin=0, ymax=110, linestyles='dashed');
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated func
tion and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function wi
th similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

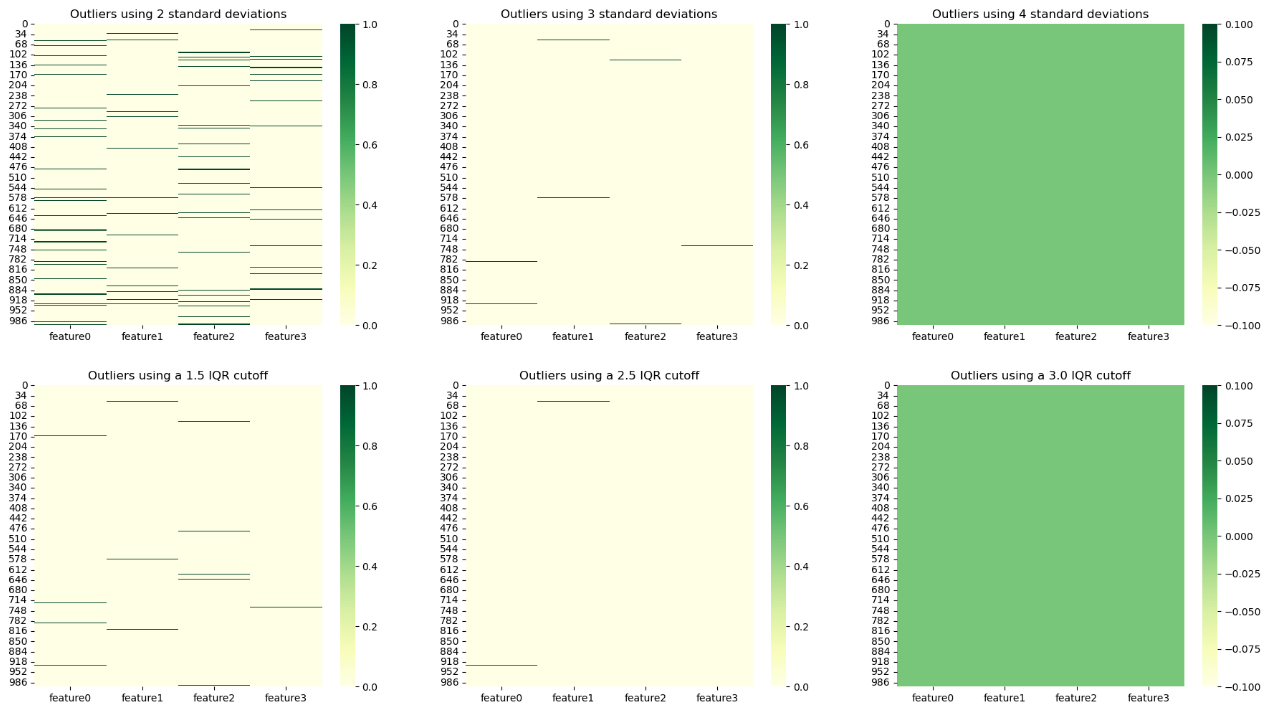



Visualize the outliers in the context of the feature's distribution.

### Compare Standard Deviation and IQR

In [9]:
```python
# For comparison, make one array each using standard deviations of 2.0, 3.0 and 4.0.
std2 = dummydf.apply(out_std, nstd=2.0)
std3 = dummydf.apply(out_std, nstd=3.0)
std4 = dummydf.apply(out_std, nstd=4.0)

# For comparison, make one array each at varying values of k.
iqr1 = dummydf.apply(out_iqr, k=1.5)
iqr2 = dummydf.apply(out_iqr, k=2.0)
iqr3 = dummydf.apply(out_iqr, k=3.0)
```

In [10]:
```python
f, ((ax1, ax2, ax3), (ax4, ax5, ax6)) = plt.subplots(ncols=3, nrows=2, figsize=(22, 12));
ax1.set_title('Outliers using 2 standard deviations');
ax2.set_title('Outliers using 3 standard deviations');
ax3.set_title('Outliers using 4 standard deviations');
ax4.set_title('Outliers using a 1.5 IQR cutoff');
ax5.set_title('Outliers using a 2.5 IQR cutoff');
ax6.set_title('Outliers using a 3.0 IQR cutoff');

sns.heatmap(std2, cmap='YlGn', ax=ax1);
sns.heatmap(std3, cmap='YlGn', ax=ax2);
sns.heatmap(std4, cmap='YlGn', ax=ax3);
sns.heatmap(iqr1, cmap='YlGn', ax=ax4);
sns.heatmap(iqr2, cmap='YlGn', ax=ax5);
sns.heatmap(iqr3, cmap='YlGn', ax=ax6);

plt.savefig('outliers.png') # testing control of newsfeed figure: https://www.kaggle.com/questions-and-answers/57099#post:
plt.show()
```




In [11]:
```python
def plot_cutoff(dataframe, col, nstd=2.0, color='red'):
    """
    Version 8 update: more generic function.
    Function added in Version 6, more readable code than previous versions.
    From version 4 update:
    This code block will plot lower and upper thresholds.
    I'm still thinking about how to best visualize this, suggestions welcome!
    """
    lower, upper = out_std(dataframe[col], nstd=nstd, return_thresholds=True)
    plt.axvspan(min(dataframe[col][dataframe[col] < lower], default=dataframe[col].min()), lower, alpha=0.2, color=color)
    plt.axvspan(upper, max(dataframe[col][dataframe[col] > upper], default=dataframe[col].max()), alpha=0.2, color=color)
```

### Nonparametric methods: Univariate

**The features in the Melbourne Housing dataset are skewed, they should serve as good distributions to test nonparametric outlier detection method.**

In [31]:
```python
df = pd.read_csv('Melbourne_housing_FULL.csv')
```

In [32]:
```python
df.fillna(df.median(), inplace = True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_24228\3097012583.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  df.fillna(df.median(), inplace = True)
```

In [33]:
```
df_num = df.select_dtypes (include = ["float64", "int64"])
```

In [34]:
```
cols = df_num.columns.tolist()
cols
```

Out[34]:
```
['Rooms',
 'Price',
 'Distance',
 'Postcode',
 'Bedroom2',
 'Bathroom',
 'Car',
 'Landsize',
 'BuildingArea',
 'YearBuilt',
 'Lattitude',
 'Longtitude',
 'Propertycount']
```

In [35]:
```
df[cols].dtypes
```

Out[35]:
```
Rooms              int64
Price            float64
Distance         float64
Postcode         float64
Bedroom2         float64
Bathroom         float64
Car              float64
Landsize         float64
BuildingArea     float64
YearBuilt        float64
Lattitude        float64
Longtitude       float64
Propertycount    float64
dtype: object
```

In [36]:
```
from sklearn.ensemble import IsolationForest
```

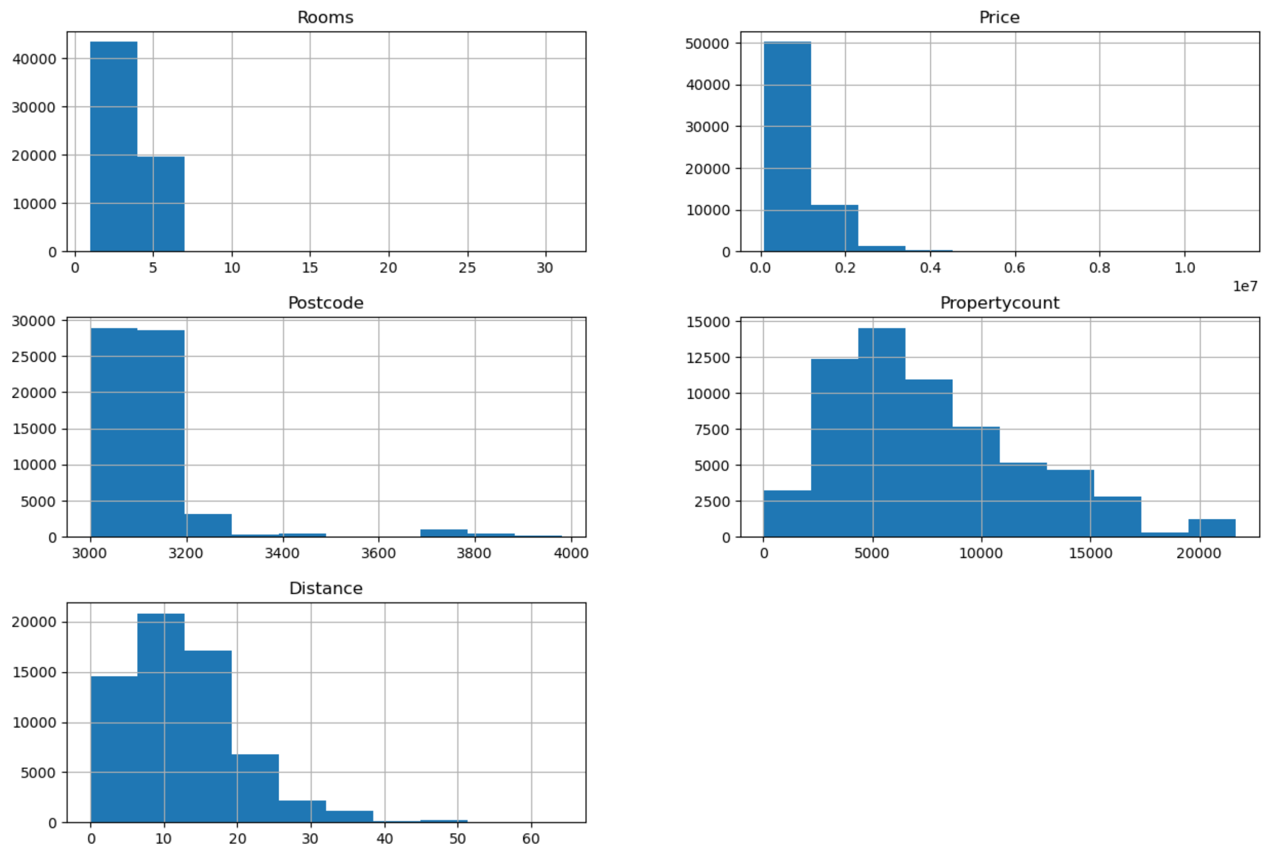
In [39]:
```
for i, column in enumerate(cols):
#     print (i, column)
    isolation_forest = IsolationForest(contamination='auto')
    isolation_forest.fit(df[column].values.reshape(-1,1))
```

In [ ]:
```
# for this exercise, just fill missing values with the median value for a column.
# Using median ensures filled values will be whole numbers.
df.fillna(df.median(), inplace=True)
```

In [ ]:
```
cols = ['Rooms', 'Price', 'Distance', 'Bedroom2', 'Bathroom', 'Car', 'Landsize', 'Propertycount']
```

In [28]:

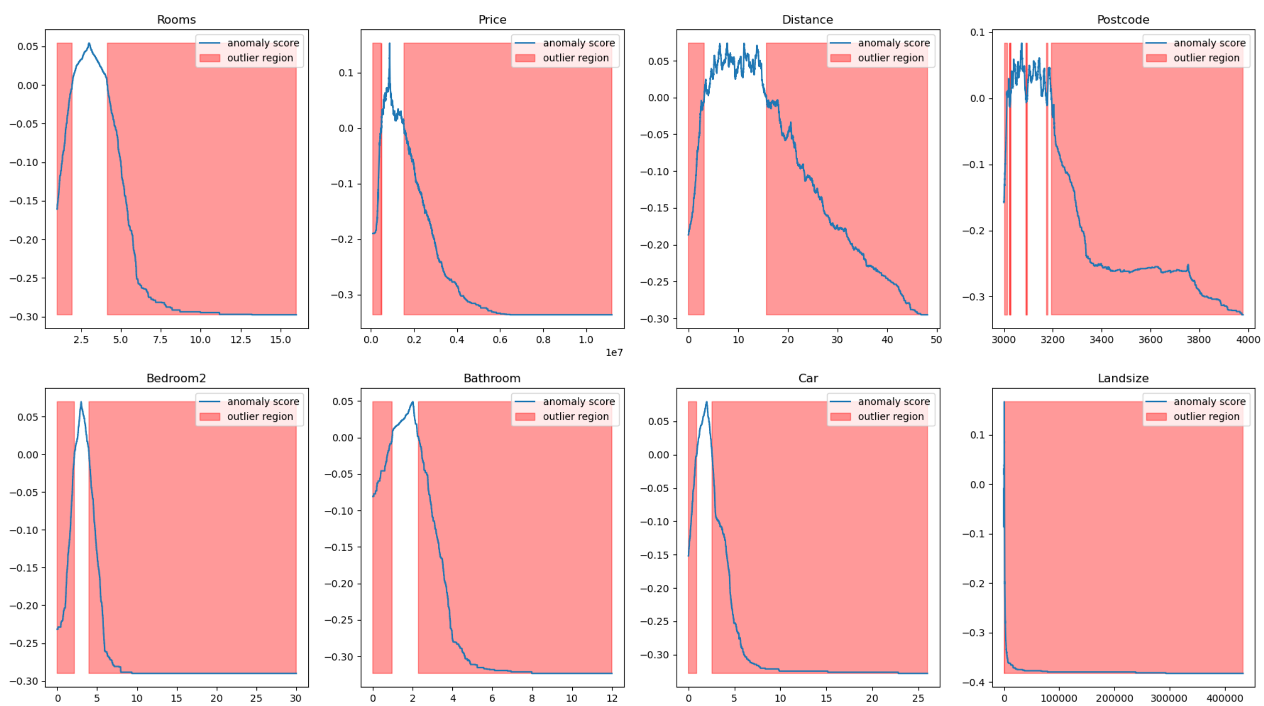
```
df[cols].hist(figsize=(15,10));
```



In [41]:

```
fig, axs = plt.subplots(2, 4, figsize=(22, 12), facecolor='w', edgecolor='k')
axs = axs.ravel()

for i, column in enumerate(cols):
    isolation_forest = IsolationForest(contamination='auto')
    isolation_forest.fit(df[column].values.reshape(-1,1))

    xx = np.linspace(df[column].min(), df[column].max(), len(df)).reshape(-1,1)
    anomaly_score = isolation_forest.decision_function(xx)
    outlier = isolation_forest.predict(xx)

    axs[i].plot(xx, anomaly_score, label='anomaly score')
    axs[i].fill_between(xx.T[0], np.min(anomaly_score), np.max(anomaly_score),
                     where=outlier==-1, color='r',
                     alpha=.4, label='outlier region')
    axs[i].legend()
    axs[i].set_title(column)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24228\146581728.py in <module>
     10     outlier = isolation_forest.predict(xx)
     11 
---> 12     axs[i].plot(xx, anomaly_score, label='anomaly score')
     13     axs[i].fill_between(xx.T[0], np.min(anomaly_score), np.max(anomaly_score),
     14                      where=outlier==-1, color='r',

IndexError: index 8 is out of bounds for axis 0 with size 8
```




# Parametric methods: Multivariate

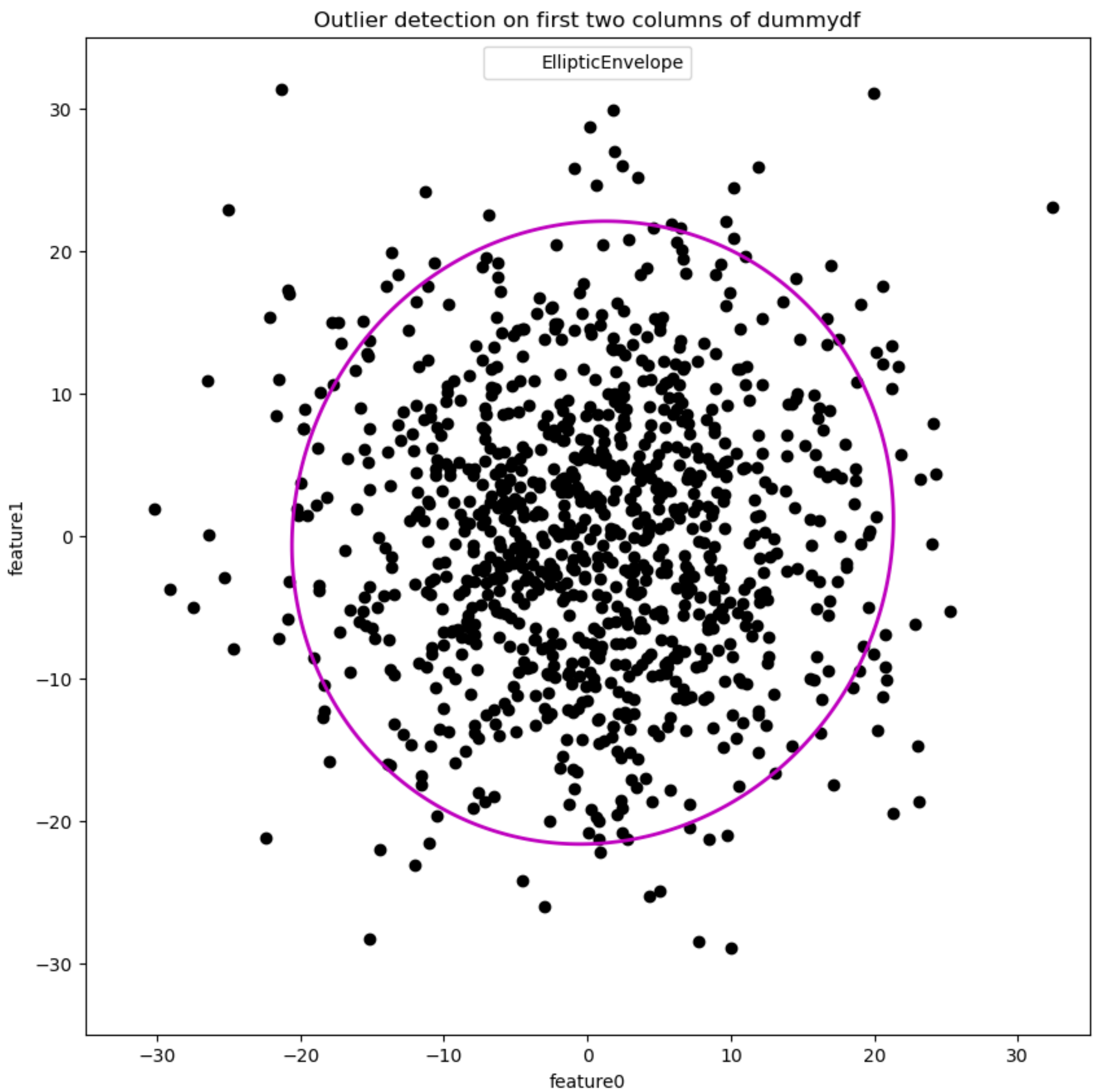
I will show multivariate outlier detection using two scikit-learn methods: For normally distributed data (ex. `dummydf` ), use [EllipticEnvelope (http://scikit-learn.org/stable/modules/generated/sklearn.covariance.EllipticEnvelope.html#sklearn.covariance.EllipticEnvelope)](http://scikit-learn.org/stable/modules/generated/sklearn.covariance.EllipticEnvelope.html#sklearn.covariance.EllipticEnvelope).

## EllipticEnvelope

For visualization purposes, use the first two features

In [43]:

```python
clf = EllipticEnvelope()

legend = {}
xx, yy = np.meshgrid(np.linspace(-35, 35, 500), np.linspace(-35, 35, 500))
plt.figure(1, figsize=(10,10))
clf.fit(dummydf.values[:,:2])
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
legend['EllipticEnvelope'] = plt.contour(
    xx, yy, Z, levels=[0], linewidths=2, colors=['m'])

legend_values_list = list(legend.values())
legend_keys_list = list(legend.keys())

plt.figure(1, figsize=(10,10))# two clusters
plt.title("Outlier detection on first two columns of dummydf")
plt.scatter(dummydf.values[:, 0], dummydf.values[:, 1], color='black')
bbox_args = dict(boxstyle="round", fc="0.8")
arrow_args = dict(arrowstyle="->")

plt.xlim((xx.min(), xx.max()))
plt.ylim((yy.min(), yy.max()))
plt.legend(legend_values_list[0].collections, legend_keys_list,
           loc="upper center");
plt.ylabel("feature1");
plt.xlabel("feature0");
```




## Nonparametric methods: Multivariate

## DBSCAN

DBSCAN on Melbourne Housing Data

For skewed distributions a quick and dirty method called [DBSCAN (http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html)](http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html). [Here (http://scikit-learn.org/stable/auto_examples/applications/plot_outlier_detection_housing.html)](http://scikit-learn.org/stable/auto_examples/applications/plot_outlier_detection_housing.html) is a good tutorial for other methods in scikit-learn. This is a quick **nonparametric** method that can be used in **multivariate** analyses. Parameters that will significantly affect clusters and worth tuning are `eps` and `min_samples`.

In [44]:

```
# scale data first
X = StandardScaler().fit_transform(df[cols].values)

db = DBSCAN(eps=3.0, min_samples=10).fit(X)
labels = db.labels_
```

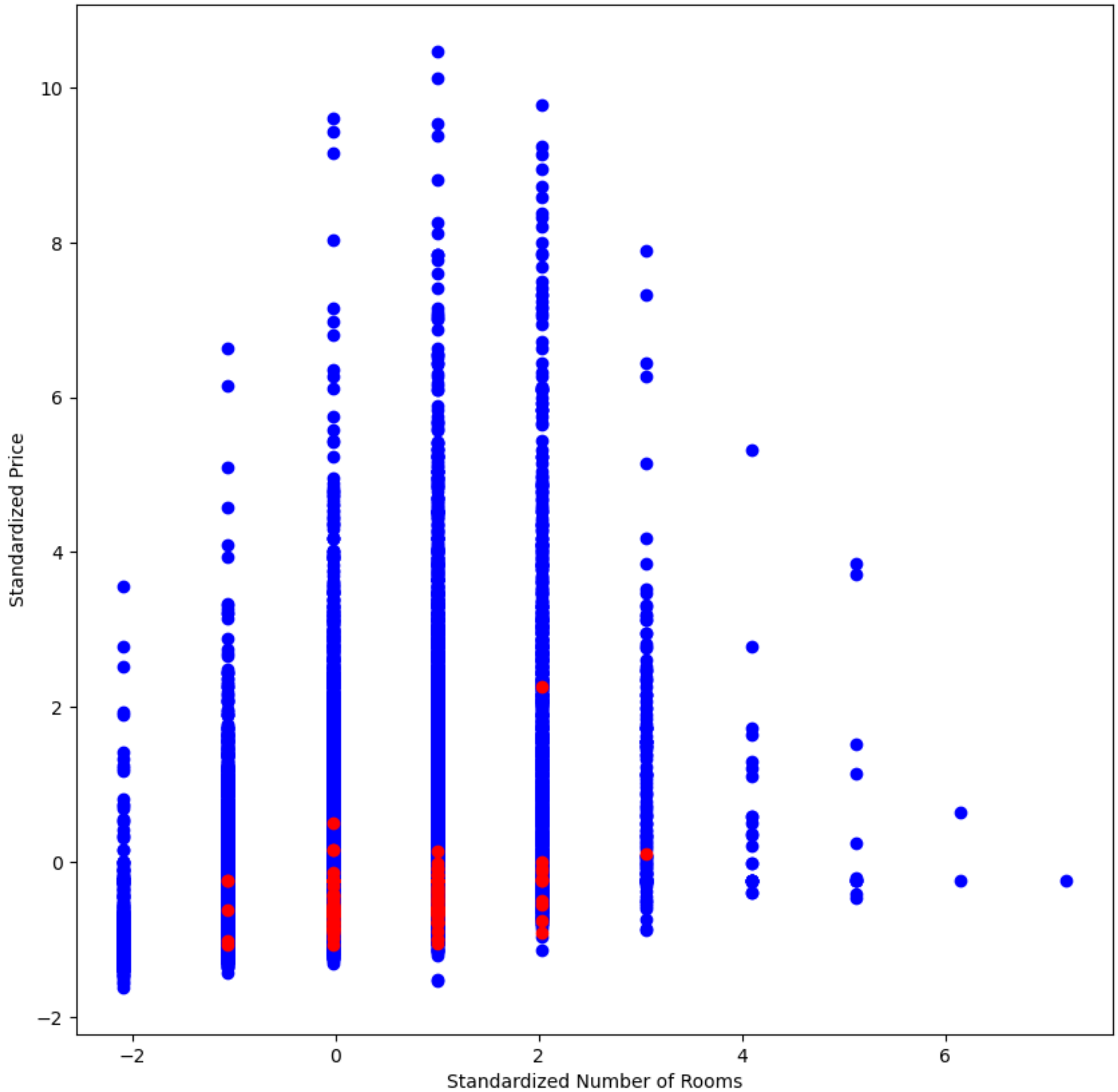
In [49]:

```
pd.Series(labels).value_counts()
```

Out[49]:

```
 0    34523
 1      182
-1      126
 2       26
dtype: int64
```

In [50]:
```
plt.figure(figsize=(10,10))

unique_labels = set(labels)
colors = ['blue', 'red']

for color,label in zip(colors, unique_labels):
    sample_mask = [True if l == label else False for l in labels]
    plt.plot(X[:,0][sample_mask], X[:, 1][sample_mask], 'o', color=color);
plt.xlabel('Standardized Number of Rooms');
plt.ylabel('Standardized Price');
```




## LocalOutlierFactor

I tried detecting outliers in the first two features again, but [LocalOutlierFactor (http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html#sklearn.neighbors.LocalOutlierFactor)](http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html#sklearn.neighbors.LocalOutlierFactor) may require fine-tuning.

In [56]:
```
clf = LocalOutlierFactor(n_neighbors=50, contamination='auto')
y_pred = clf.fit_predict(X[:,:2])
```

In [58]:

```python
plt.figure(figsize=(10,10))
# plot the level sets of the decision function
xx, yy = np.meshgrid(np.linspace(-3, 14, num=200), np.linspace(-3, 18, num=200))
Z = clf._decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

in_mask = [True if l == 1 else False for l in y_pred]
out_mask = [True if l == -1 else False for l in y_pred]

plt.title("Local Outlier Factor (LOF)")
plt.contourf(xx, yy, Z, cmap=plt.cm.Blues_r)
# inliers
a = plt.scatter(X[in_mask, 0], X[in_mask, 1], c='white',
                edgecolor='k', s=20)
# outliers
b = plt.scatter(X[out_mask, 0], X[out_mask, 1], c='red',
                edgecolor='k', s=20)
plt.axis('tight')
plt.xlabel('Standardized Number of Rooms');
plt.ylabel('Standardized Price');
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_24228\3329248804.py in <module>
      2 # plot the level sets of the decision function
      3 xx, yy = np.meshgrid(np.linspace(-3, 14, num=200), np.linspace(-3, 18, num=200))
----> 4 Z = clf._decision_function(np.c_[xx.ravel(), yy.ravel()])
      5 Z = Z.reshape(xx.shape)
      6 

AttributeError: 'LocalOutlierFactor' object has no attribute '_decision_function'
```

```
<Figure size 1000x1000 with 0 Axes>
```

In [ ]: