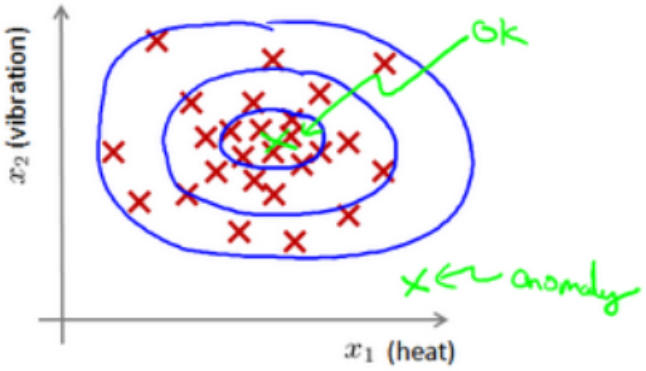
异常检测(Anomaly Detection)

问题的动机

异常检测(**Anomaly detection**)是机器学习中常见的问题，可以应用于各种实际的生产应用。比如检验飞机引擎的质量问题，某个网站的用户行为检测等等。

以飞机的引擎质量检测为例：

给定数据集$x^{(1)}, x^{(2)}, . . , x^{(m)}$，其中$x^{(i)}$包含两个特征值。我们希望知道新的数据$x_{test}$是不是异常D的。我们所构建的模型应该能根据该测试数据的位置计算可能性 $p(x)$。 $p(x)$的值可以帮我们判断该引擎是属于**Anomaly**还是**Normal**类的。



上图中，在蓝色圈内的数据属于该组数据的可能性较高，而越是偏远的数据，其属于该组数据的可能性就越低。这种方法称为密度估计，表达如下：

$$if \quad p(x) \begin{cases} < \varepsilon & anomaly \\ >= \varepsilon & normal \end{cases}$$

高斯分布与异常检测算法

如果变量 x 符合高斯分布 $x \sim N(\mu, \sigma^2)$则其概率密度函数为：

$$p(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

我们可以利用已有的数据来预测总体中的μ和σ^2，计算方法如下：

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2$$

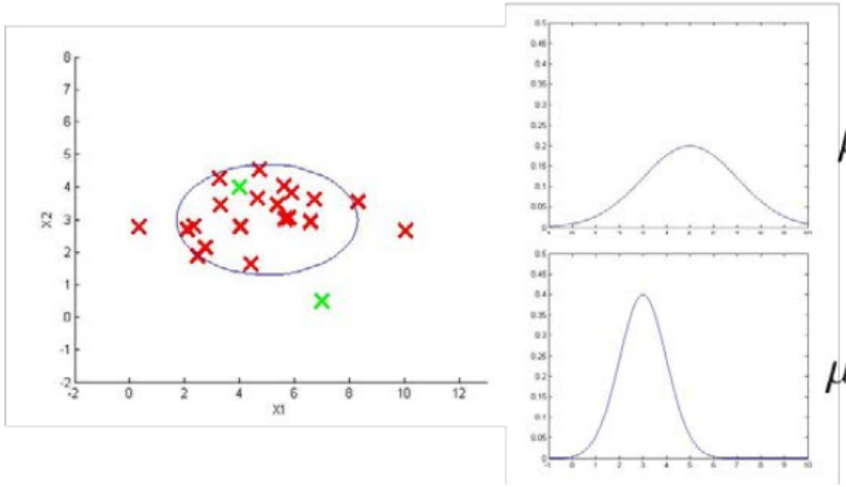
接下来，我们可以应用高斯分布开发异常检测算法：

对于给定的数据集 $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$，我们要针对每一个特征计算 μ 和 σ^2 的估计值。一旦我们获得了平均值和方差的估计值，给定新的一个训练实例x_{test}，根据模型计算 $p(x)$：

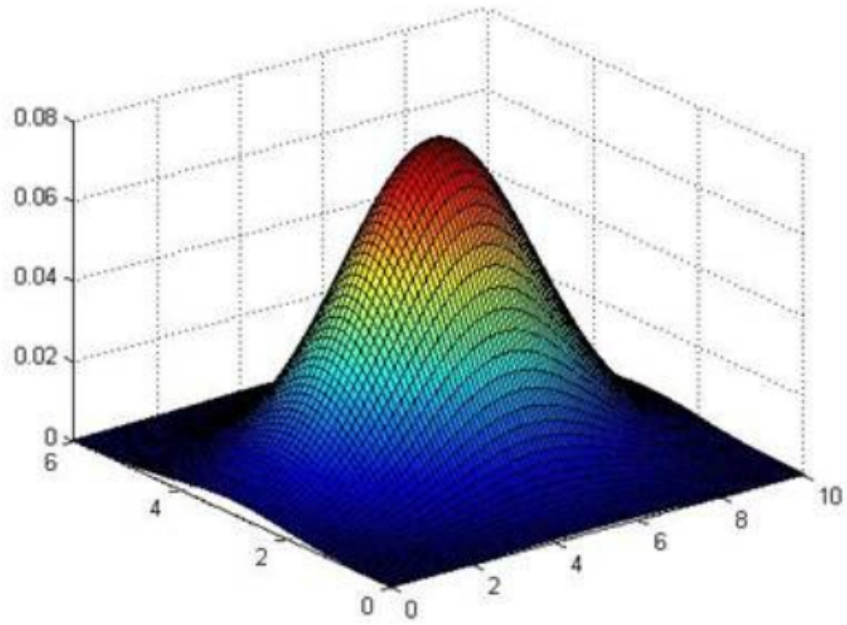
$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2})$$

当$p(x) < \varepsilon$时，判断训练实例x_{test}为异常。

下图是一个由两个特征的训练集，以及特征的分布情况：



下面的三维图表表示的是密度估计函数，z轴为根据两个特征的值所估计$p(x)$值：



我们选择一个ε，将$p(x) = \varepsilon$作为我们的判定边界，当$p(x) > \varepsilon$时预测数据为正常数据，否则为异常。

异常检测算法是一个非监督学习算法，意味着我们无法根据结果变量 y 的值来告诉我们数据是否真的是异常的。我们需要另一种方法来帮助检验算法是否有效。当我们开发一个异常检测系统时，我们从带标记（异常或正常）的数据着手，选择一部分正常数据用于构建训练集，然后用剩下的正常数据和异常数据混合的数据构成交叉检验集和测试集。

例如：我们有10000台正常引擎的数据，有20台异常引擎的数据。 我们可以这样分配数据：

6000台正常引擎的数据作为训练集

2000台正常引擎和10台异常引擎的数据作为交叉检验集

2000台正常引擎和10台异常引擎的数据作为测试集

具体的评价方法如下：

1. 根据测试集数据，我们估计特征的平均值和方差并构建$p(x)$函数

2. 对交叉检验集，我们尝试使用不同的ε值作为阀值，并预测数据是否异常，根据$F1$值或者**precision**与**recall**的比例来选择 ε
3. 选出 ε 后，针对测试集进行预测，计算异常检验系统的$F1$值，或者查准**precision**与**recall**之比。

异常检测与监督学习对比

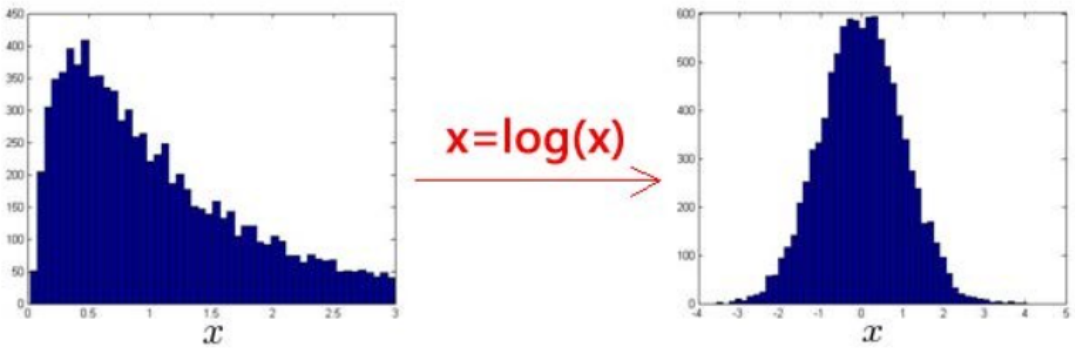
两者比较：

异常检测	**监督学习**
非常少量的正向类（异常数据 $y=1$）, 大量的负向类（$y=0$）	同时有大量的正向类和负向类
许多不同种类的异常，非常难整理出所有的异常情况。因此根据非常少量的正向类数据来训练算法往往可以得到更好的结果。	有足够多的正向类实例，足够用于训练算法，未来遇到的正向类实例可能与训练集中的非常近似。
未来遇到的异常可能与已掌握的异常并不类似，甚至可能有较大的差异。	
应用领域：欺诈行为检测生产（例如飞机引擎）检测，数据中心的计算机运行状况	应用领域：邮件过滤器，天气预报，肿瘤分类

选择特征

异常检测假设特征符合高斯分布，如果数据的分布不是高斯分布，异常检测算法也能够工作，但是最好还是将数据转换成高斯分布，例如使用对数函数：$x=log(x+c)$，其中 c 为非负常数。

如图所示为使用对数函数后将数据转换为高斯分布的例子：



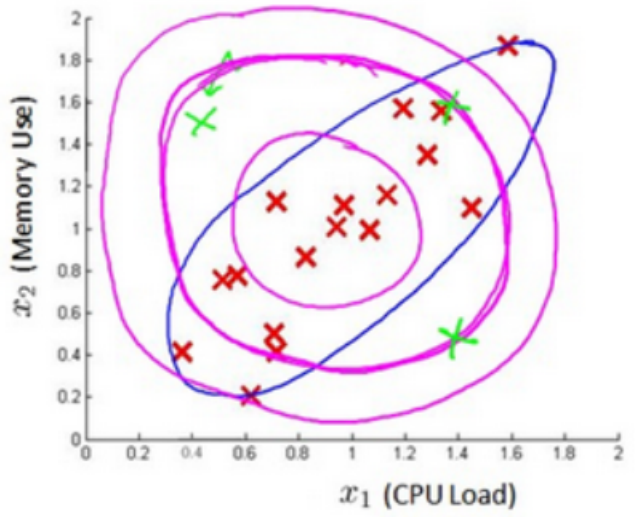
误差分析：

一些异常的数据可能也会有较高的$p(x)$值，因而被算法认为是正常的。我们可以分析那些被算法错误预测为正常的数据，观察判断是否需要增加一些新的**feature**，增加这些新的**feature**后获得的新算法能够帮助我们更好地进行异常检测。

有时我们也可以通过将一些相关的特征进行组合，来获得一些新的更好的特征（异常数据的该特征值异常地大或小）。例如在检测数据中心的计算机状况的例子中，我们可以用**CPU**负载与网络通信量的比值作为一个新的特征，如果该值异常地大，便有可能意味着该服务器的**CPU**高速运转，但网络通信的数据量很少，意味着该服务器可能陷入了死循环的异常情况。

多元高斯分布

假使我们有两个相关的特征，而且这两个特征的值域范围比较宽，这种情况下，一般的高斯分布模型可能不能很好地识别异常数据。下图中是两个相关特征，粉红色的线是一般的高斯分布模型获得的判定边界。由图中数据点的分布可以看出来，很明显绿色的**X**所代表的数据点很可能是异常值，但是其$p(x)$值却仍然在正常范围内。多元高斯分布将创建像图中蓝色曲线所示的判定边界。



在一般的高斯分布模型中，我们计算 $p(x)$ 的方法是：
通过分别计算每个特征对应的几率然后将其累乘起来，在多元高斯分布模型中，我们将构建特征的协方差矩阵，用所有的特征一起来计算 $p(x)$。

我们首先计算所有特征的平均值，然后再计算协方差矩阵：

$$p(x)=\prod_{j=1}^{n}p(x_j;\mu,\sigma_j^2)=\prod_{j=1}^{n}\frac{1}{\sqrt{2\pi}\sigma_j}exp(-\frac{(x_j-\mu_j)^2}{2\sigma_j^2})$$

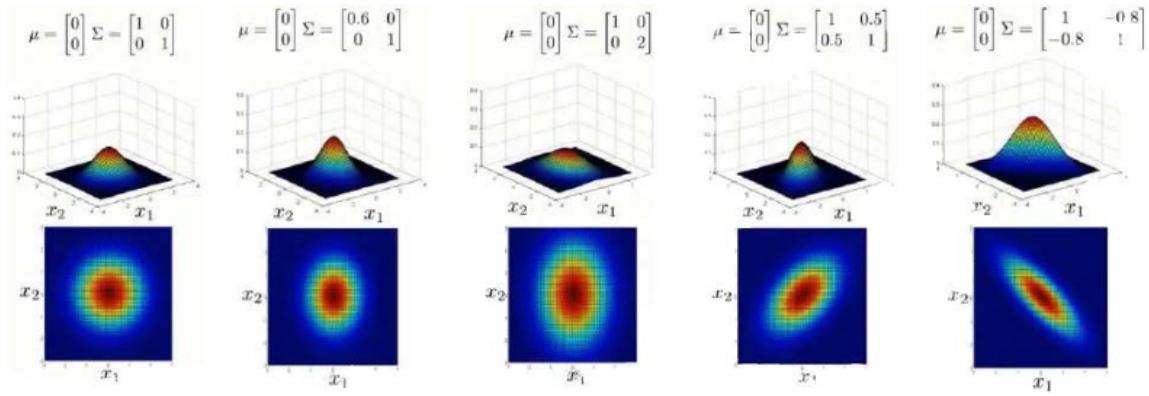
$$\mu=\frac{1}{m}\sum_{i=1}^{m}x^{(i)}$$

$$\Sigma=\frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\mu)(x^{(i)}-\mu)^T=\frac{1}{m}(X-\mu)^T(X-\mu)$$

其中μ 是一个向量，其每一个单元都是原特征矩阵中一行数据的均值。最后我们计算多元高斯分布的 $p\left(x\right)$:

$$p(x)=\frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}}exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

其中$|\Sigma|$是一个矩阵，在 **Octave** 中用 `det(sigma)` 计算。Σ^{-1} 是逆矩阵，如图是协方差对模型的影响：



上图是5个不同的模型，从左往右依次分析：

1. 是一个一般的高斯分布模型
2. 通过协方差矩阵，令特征1拥有较小的偏差，同时保持特征2的偏差
3. 通过协方差矩阵，令特征2拥有较大的偏差，同时保持特征1的偏差
4. 通过协方差矩阵，在不改变两个特征的原有偏差的基础上，增加两者之间的正相关性
5. 通过协方差矩阵，在不改变两个特征的原有偏差的基础上，增加两者之间的负相关性

原高斯分布模型和多元高斯分布模型的比较：

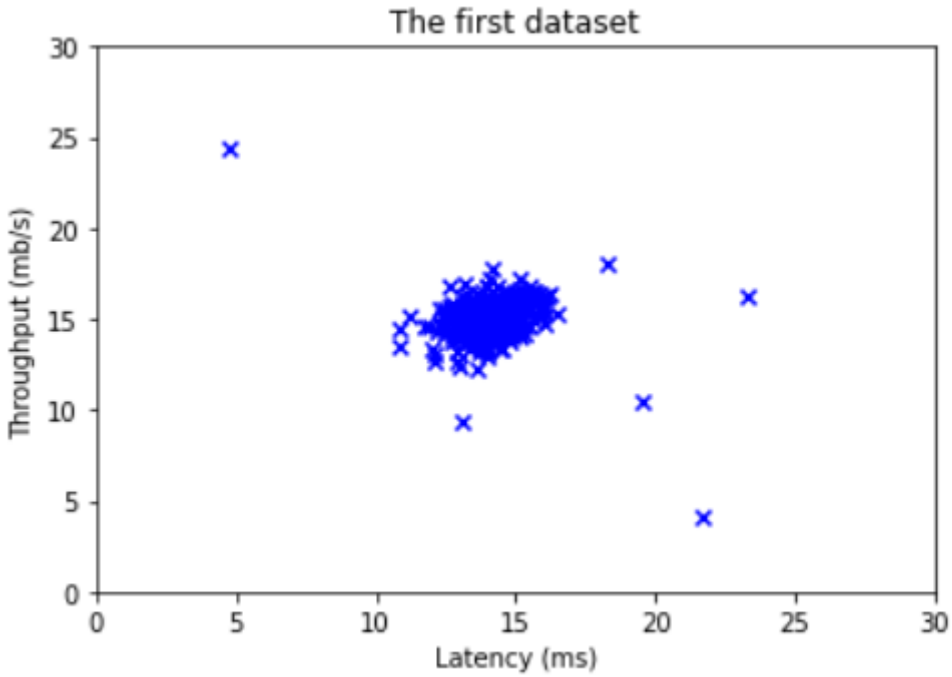
原高斯分布模型	**多元高斯分布模型**
不能捕捉特征之间的相关性，但可以通过将特征进行组合的方法来解决	自动捕捉特征之间的相关性
计算代价低，能适应大规模的特征	计算代价较高 训练集较小时也同样适用
	必须要有 $m > n$，不然的话协方差矩阵Σ不可逆的

实验

Exercise1

exercise1是编写代码完成高斯分布的相关内容，利用公式计算μ和σ^2。同时exercise1中我们可以将高斯分布可视化，观察**Anomaly**和**Normal**数据点的分布情况。

我们先将数据集中的数据加载出来。数据集中数据包括两个**feature**，分别是**throughput (mb/s)** 和 **latency (ms)** 。



大部分的代码已经完成，我们只需要在红框中完成计算μ和σ^2的部分，注意要指定正确的轴。

In [16]:
```
# UNQ_C1
# GRADED FUNCTION: estimate_gaussian

def estimate_gaussian(X):
    """
    Calculates mean and variance of all features
    in the dataset

    Args:
        X (ndarray): (m, n) Data matrix

    Returns:
        mu (ndarray): (n,) Mean of all features
        var (ndarray): (n,) Variance of all features
    """

    m, n = X.shape

    ### START CODE HERE ###
    mu = np.mean(X, axis = 0)#别忘了要指定轴
    var = np.mean((X - mu)**2, axis = 0) ##注意**2是在sum里面的
    ### END CODE HERE ###

    return mu, var
```

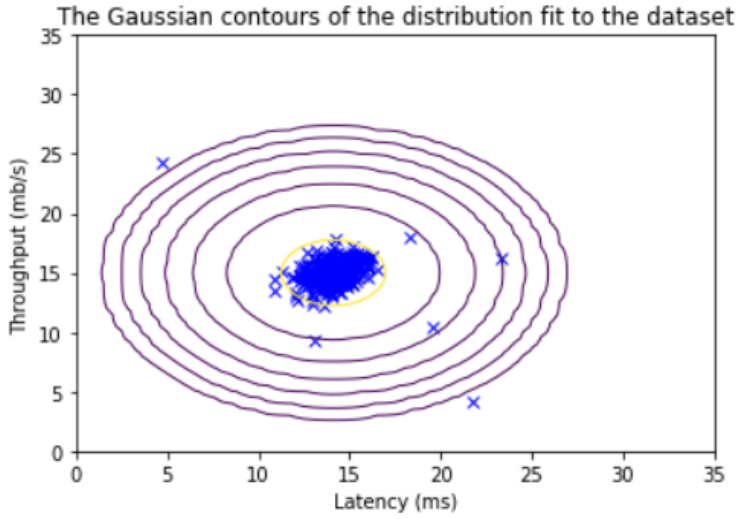
接下来运行代码，打印出μ和σ^2。红框中为打印出的结果。

In [17]:
```
# Estimate mean and variance of each feature
mu, var = estimate_gaussian(X_train)

print("Mean of each feature:", mu)
print("Variance of each feature:", var)

# UNIT TEST
from public_tests import *
estimate_gaussian_test(estimate_gaussian)
```

```
Mean of each feature: [14.11222578 14.99771051]
Variance of each feature: [1.83263141 1.70974533]
All tests passed!
```

在exercise1中利用多元高斯分布来观察数据集中的数据情况，可以发现大部分数据都处于**Normal**的圈中，少数几个异常点属于**Anomaly**的情况。

In [18]:
```
# Returns the density of the multivariate normal
# at each data point (row) of X_train
p = multivariate_gaussian(X_train, mu, var)

#Plotting code
visualize_fit(X_train, mu, var)
```



Exercise2

exercise2中我们要选择一个最好的ε，ε的选择基于$F1$值。

如图所示的红框为计算$F1$值的代码。需要先计算出**precision**和**recall**再代入计算$F1$值的公式。如果计算出的$F1$值比之前的最大值$F1$值还要大，就更新$F1$值和ε。

```
best_epsilon = 0
best_F1 = 0
F1 = 0

step_size = (max(p_val) - min(p_val)) / 1000

for epsilon in np.arange(min(p_val), max(p_val), step_size):

    ### START CODE HERE ###
    predictions = (p_val < epsilon)

    tp = np.sum((predictions == 1) & (y_val == 1))
    fp = np.sum((predictions == 1) & (y_val == 0))# Your code here to calculate
    fn = np.sum((predictions == 0) & (y_val == 1))# Your code here to calculate

    prec = tp / (tp + fp)
    rec = tp / (tp + fn)

    F1 = 2 * prec * rec / (prec + rec)# Your code here to calculate F1
    ### END CODE HERE ###


    if F1 > best_F1:
        best_F1 = F1
        best_epsilon = epsilon

return best_epsilon, best_F1
```

利用cross-validation找到最好的ε，并计算对应的$F1$值。

In [27]:
```
p_val = multivariate_gaussian(X_val, mu, var)
epsilon, F1 = select_threshold(y_val, p_val)

print('Best epsilon found using cross-validation: %e' % epsilon)
print('Best F1 on Cross Validation Set: %f' % F1)

# UNIT TEST
select_threshold_test(select_threshold)
```

```
Best epsilon found using cross-validation: 8.990853e-05
Best F1 on Cross Validation Set: 0.875000
All tests passed!
```

运行异常检测代码，可以发现异常点，并将其用红圈标注在图上。

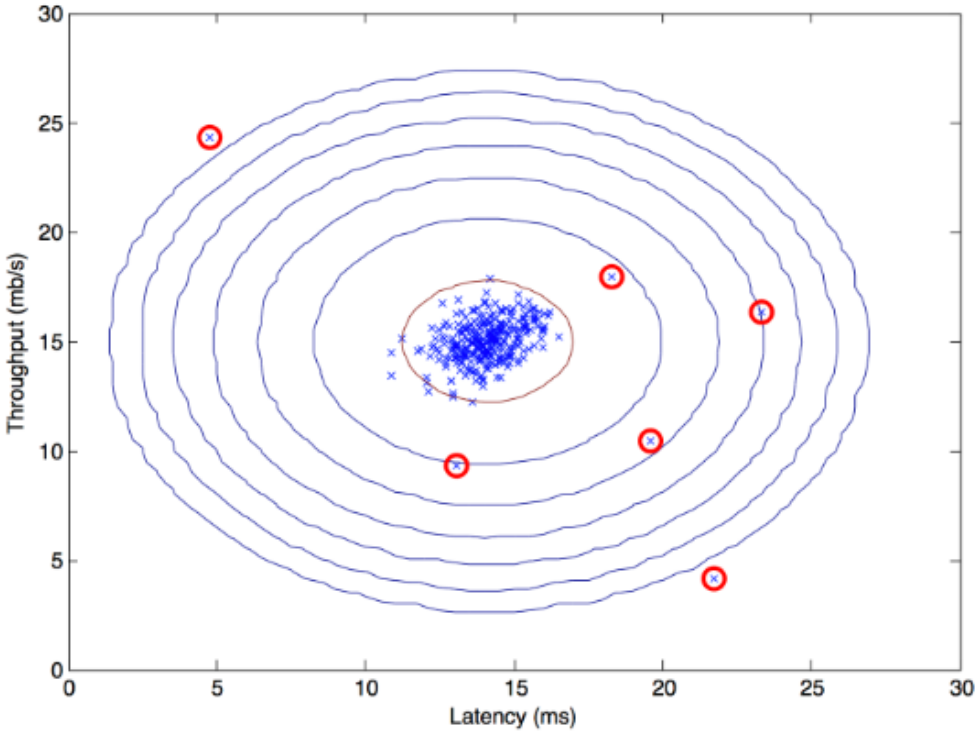


Figure 3: The classified anomalies.