# Отчёт по лабораторной работе №8 "Выявление аномалий"

In [81]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from scipy.io import loadmat

%matplotlib inline
```

## 1. Загрузите данные ex8data1.mat из файла.

In [82]:

```
mat = loadmat('data/ex8data1.mat')
X = mat['X']
X_val = mat['Xval']
y_val = mat['yval']
y_val = y_val.reshape(y_val.shape[0])
```

In [83]:

```
X.shape
```

Out[83]:

```
(307, 2)
```

In [84]:

```
X_val.shape
```

Out[84]:

```
(307, 2)
```

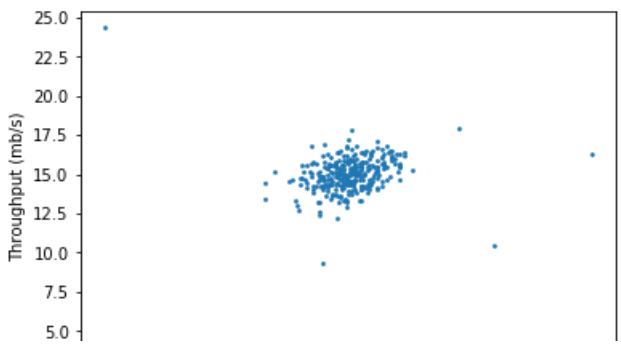
## 2. Постройте график загруженных данных в виде диаграммы рассеяния.

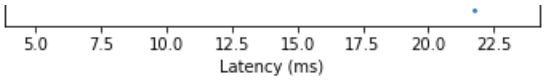
In [85]:

```
plt.scatter(X[:,0],X[:,1], s=3)
plt.xlabel("Latency (ms)")
plt.ylabel("Throughput (mb/s)")
```

Out[85]:

```
Text(0, 0.5, 'Throughput (mb/s)')
```






## 3. Представьте данные в виде двух независимых нормально распределенных случайных величин.

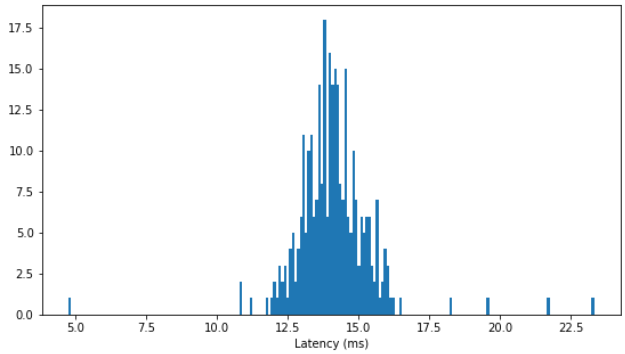
In [86]:

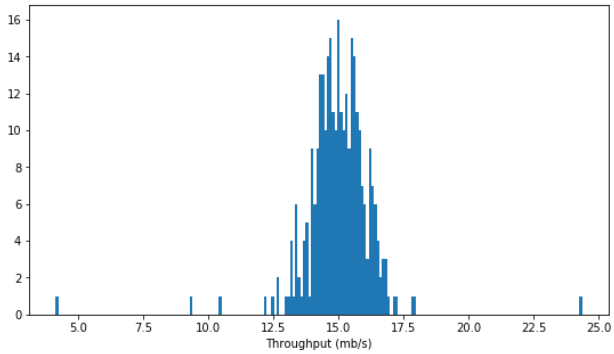
```
x1, x2 = X[:, 0], X[:, 1]

fig, axs = plt.subplots(1, 2, figsize=(20, 5))
axs[0].hist(x1, bins=200)
axs[0].set_xlabel("Latency (ms)")

axs[1].hist(x2, bins=200)
axs[1].set_xlabel("Throughput (mb/s)")

plt.show()
```




## 4. Оцените параметры распределений случайных величин.

Оба признака являются нормально распределенными случайными величинами.

Gaussian Distribution $p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{\frac{(x-\mu)^2}{2\sigma^2}}$

$\mu_i = \frac{1}{m}\sum^m_{j=1}x^{(j)}$

$\sigma^2_i = \frac{1}{m}\sum^m_{j=1}(x^{(j)} - \mu_j)^2$

In [87]:

```
def estimate_gaussian(X):
    return X.mean(axis=0), X.std(axis=0)
```

In [88]:

```
mu, sigma = estimate_gaussian(X)
```

## 5. Постройте график плотности распределения получившейся случайной величины в виде изолиний, совместив его с графиком из пункта 2.

In [89]:

```
import scipy.stats as stats

def p(X):
    axis = int(len(X.shape) > 1)
    mu, sigma = estimate_gaussian(X)
    return stats.norm.pdf(X, mu, sigma).prod(axis=axis)
```

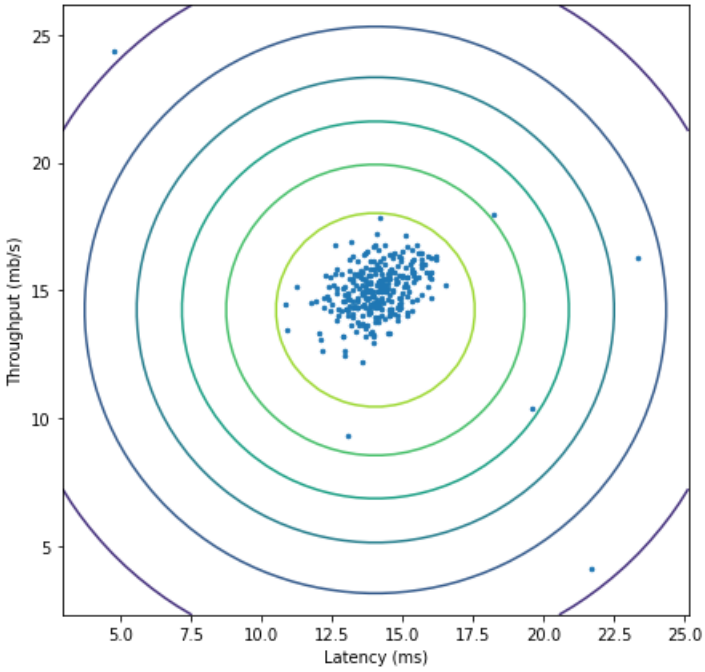
In [90]:

```
x, y = X[:, 0], X[:, 1]

h = 1.8
u = np.linspace(x.min() - h, x.max() + h, 50)
v = np.linspace(y.min() - h, y.max() + h, 50)
u_grid, v_grid = np.meshgrid(u, v)
Xnew = np.column_stack((u_grid.flatten(), v_grid.flatten()))
z = p(Xnew).reshape((len(u), len(v)))

fig, ax = plt.subplots(figsize=(7, 7))
ax.contour(u, v, z)
ax.scatter(x, y, s=6)

plt.xlabel("Latency (ms)")
plt.ylabel("Throughput (mb/s)")
plt.show()
```




## 6. Подберите значение порога для обнаружения аномалий на основе валидационной выборки. В качестве метрики используйте F1-меру.

In [91]:

```
def predict_anomalies(X, mu, sigma, eps):
    axis = int(len(X.shape) > 1)
    p = stats.norm.pdf(X, mu, sigma).prod(axis=axis)
    res = p < eps
    return res.astype(int) if axis else int(res)

def calc_eps(y_val, p_y_val, X_val, mu, sigma):
    best_eps = 0
    best_F1 = 0

    stepsize = (max(p_y_val) - min(p_y_val))/1000
    eps_range = np.arange(p_y_val.min(), p_y_val.max(), stepsize)
    for eps in eps_range:
        predictions = predict_anomalies(X_val, mu, sigma, eps)
        tp = np.sum(predictions[y_val==1]==1)
        fp = np.sum(predictions[y_val==0]==1)
        fn = np.sum(predictions[y_val==1]==0)

        # compute precision, recall and F1
        prec = tp/(tp+fp)
        rec = tp/(tp+fn)

        F1 = (2*prec*rec)/(prec+rec)
```

```
        if F1 > best_F1:
            best_F1 = F1
            best_eps = eps

    return best_eps, best_F1
```

In [92]:

```
p_y_val = p(X_val)
mu, sigma = estimate_gaussian(X_val)
eps, f1_score = calc_eps(y_val, p_y_val, X_val, mu, sigma)
```

```
/Users/anton/Documents/Maga/ml_venv/lib/python3.7/site-packages/ipykernel_launcher.py:20:
RuntimeWarning: invalid value encountered in long_scalars
```

In [93]:

```
eps
```

Out[93]:

```
0.0001572946256973518
```

## 7. Выделите аномальные наблюдения на графике из пункта 5 с учетом выбранного порогового значения.

In [94]:

```
mu, sigma = estimate_gaussian(X)
y_pred = predict_anomalies(X, mu, sigma, eps)
```

In [95]:

```
y_pred.shape
```

Out[95]:

```
(307,)
```

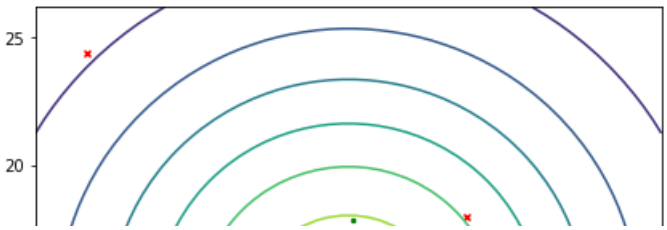
In [96]:

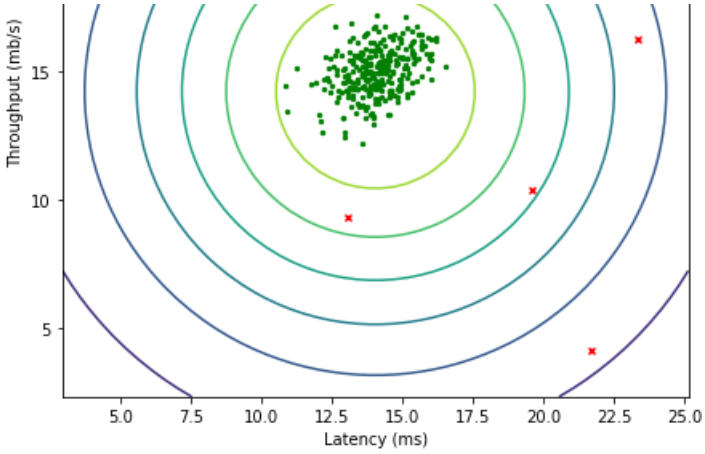
```
x, y = X[:, 0], X[:, 1]

h = 1.8
u = np.linspace(x.min() - h, x.max() + h, 50)
v = np.linspace(y.min() - h, y.max() + h, 50)
u_grid, v_grid = np.meshgrid(u, v)
Xnew = np.column_stack((u_grid.flatten(), v_grid.flatten()))
z = p(Xnew).reshape((len(u), len(v)))

fig, ax = plt.subplots(figsize=(7, 7))
ax.contour(u, v, z)
ax.scatter(x[y_pred == 0], y[y_pred == 0], s=6, color='green')
ax.scatter(x[y_pred == 1], y[y_pred == 1], s=16, color='red', marker='x')

plt.xlabel("Latency (ms)")
plt.ylabel("Throughput (mb/s)")
plt.show()
```






## 8. Загрузите данные ex8data2.mat из файла.

In [97]:

```
mat = loadmat('data/ex8data2.mat')
X = mat['X']
X_val = mat['Xval']
y_val = mat['yval']
y_val = y_val.reshape(y_val.shape[0])
```

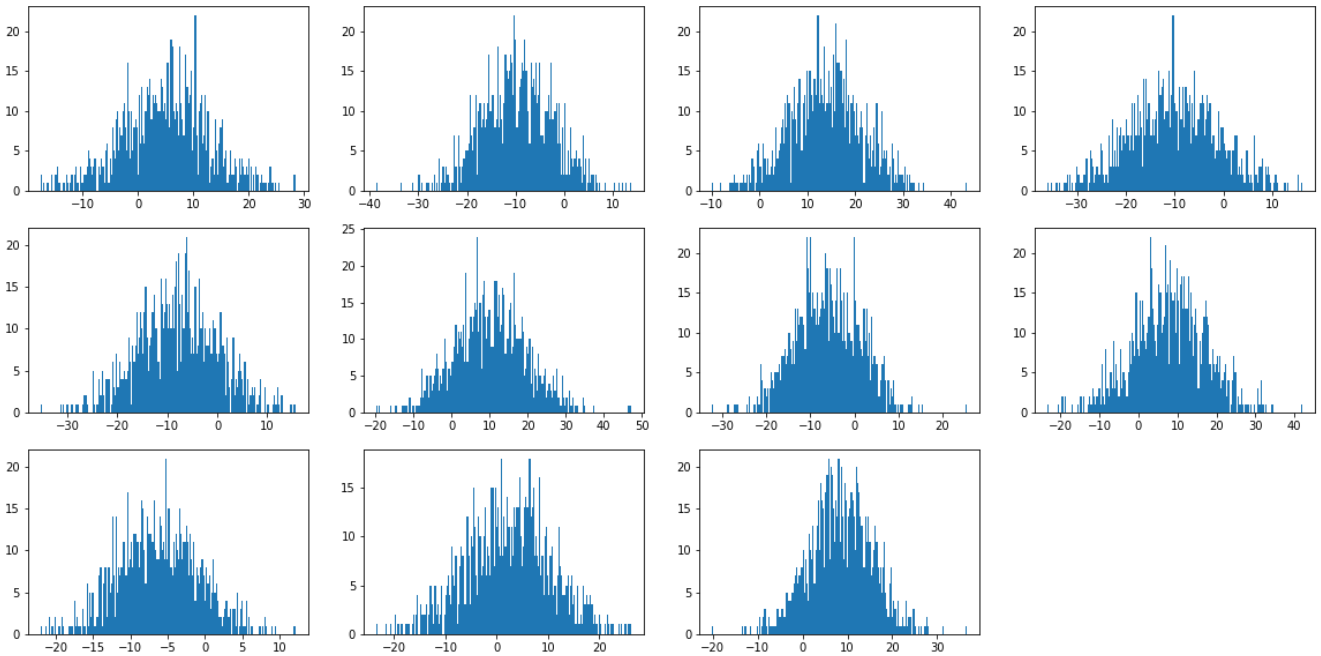
## 9. Представьте данные в виде 11-мерной нормально распределенной случайной величины.

In [98]:

```
plt.figure(figsize=(20, 10))

for i in range(0, 11):
    ax = plt.subplot(3, 4, i + 1)
    ax.hist(X[:, i], bins=200)

plt.show()
```



## 10. Оцените параметры распределения случайной величины.

In [99]:

```
mu, sigma = estimate_gaussian(X)
```

## 11. Подберите значение порога для обнаружения аномалий на основе валидационной выборки. В качестве метрики используйте F1-меру.

In [100]:

```
p_y_val = p(X_val)
mu, sigma = estimate_gaussian(X_val)
eps, f1_score = calc_eps(y_val, p_y_val, X_val, mu, sigma)
eps, f1_score
```

```
/Users/anton/Documents/Maga/ml_venv/lib/python3.7/site-packages/ipykernel_launcher.py:20:
RuntimeWarning: invalid value encountered in long_scalars
```

Out[100]:

```
(1.6620587111948565e-18, 0.6153846153846154)
```

## 12. Выделите аномальные наблюдения в обучающей выборке. Сколько их было обнаружено? Какой был подобран порог?

In [102]:

```
mu, sigma = estimate_gaussian(X)
y_pred = predict_anomalies(X, mu, sigma, eps)
anomaly_count = np.count_nonzero(y_pred)
anomaly_count, eps
```

Out[102]:

```
(134, 1.6620587111948565e-18)
```