Отчёт по лабораторной работе №8 "Выявление аномалий"

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import scipy.stats as stats

%matplotlib inline
```

In [2]:
```
from scipy.io import loadmat

def load_file(filename, keys=None):
    if keys is None:
        keys = ['X', 'y']
    mat = loadmat(filename)
    ret = tuple([mat[k].reshape(mat[k].shape[0]) if k.startswith('y') else mat[k] for k in keys])
    return ret
```

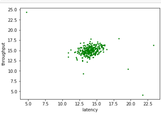
1. Загрузите данные ex8data1.mat из файла.

In [3]:
```
X, Xval, yval = load_file('ex8data1.mat', keys=['X', 'Xval', 'yval'])
print(f'X shape: {X.shape}')
print(f'Xval shape: {Xval.shape}')
print(f'yval shape: {yval.shape}')
```

```
X shape: (307, 2)
Xval shape: (307, 2)
yval shape: (307,)
```

2. Постройте график загруженных данных в виде диаграммы рассеяния.

In [4]:
```
fig, ax = plt.subplots()
ax.scatter(X[:, 0], X[:, 1], s=4, color='green')
ax.set_xlabel('latency')
ax.set_ylabel('throughput')
plt.show()
```



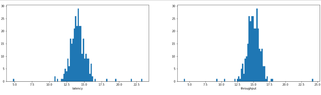
3. Представьте данные в виде двух независимых нормально распределенных случайных величин.

In [5]:
```
x, y = X[:, 0], X[:, 1]

fig, axs = plt.subplots(1, 2, figsize=(20, 5))
axs[0].hist(x, bins=100)
axs[0].set_xlabel('latency')

axs[1].hist(y, bins=100)
axs[1].set_xlabel('throughput')

plt.show()
```



Как видим из гистограм, каждый из признаков представлен в виде случайной величины с нормальным распределением.

4. Оцените параметры распределений случайных величин.

Данная функция возвращает математическое ожидание и стандартное отклонение случайных величин.

In [6]:
```
def fit(X):
    return X.mean(axis=0), X.std(axis=0)
```

In [7]:
```
mu, sigma = fit(X)
```

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

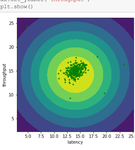
In [8]:
```
def p(X):
    axis = int(len(X.shape) > 1)
    mu, sigma = fit(X)
    return stats.norm.pdf(X, mu, sigma).prod(axis=axis)
```

In [9]:
```
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=(5, 5))
ax.contourf(u, v, z)
ax.scatter(x, y, s=4, color='green')

ax.set_xlabel('latency')
ax.set_ylabel('throughput')
plt.show()
```



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

In [10]:
```
def f1_score(y_true, y_pred):
    try:
        true_positives = np.count_nonzero(y_pred & y_true)
        precision = true_positives / np.count_nonzero(y_pred)
        recall = true_positives / np.count_nonzero(y_true)
        return 2 * (precision * recall) / (precision + recall)
    except ZeroDivisionError:
        return 0
```

In [11]:
```
def predict(X, dist_params):
    mu, sigma, eps = dist_params
    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)
```

In [12]:
```
def search_eps(X, Xval, yval, fit_func, predict_func, eps_list):

    eps_with_max_score = None
    max_score = -np.inf

    mu, sigma = fit_func(X)
    for eps_test in eps_list:
        y_pred = predict_func(Xval, (mu, sigma, eps_test))
        score = f1_score(yval, y_pred)
        if score >= max_score:
            max_score = score
            eps_with_max_score = eps_test

    return eps_with_max_score
```

In [13]:
```
eps_list = np.linspace(0.0001, 0.001, 1000)
eps = search_eps(X, Xval, yval, fit, predict, eps_list)
print(f'Epsilon with max F1 score: {eps}')
```

```
Epsilon with max F1 score: 0.0004981981981981982
```

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

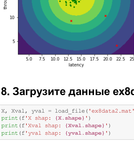
In [14]:
```
y_pred = predict(Xval, (mu, sigma, eps))
```

In [15]:
```
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=(5, 5))
ax.contourf(u, v, z)
ax.scatter(x[y_pred == 0], y[y_pred == 0], s=4, color='green')
ax.scatter(x[y_pred == 1], y[y_pred == 1], s=16, color='red', marker='x')

ax.set_xlabel('latency')
ax.set_ylabel('throughput')
plt.show()
```



8. Загрузите данные ex8data2.mat из файла.

In [16]:
```
X, Xval, yval = load_file('ex8data2.mat', keys=['X', 'Xval', 'yval'])
print(f'X shape: {X.shape}')
print(f'Xval shape: {Xval.shape}')
print(f'yval shape: {yval.shape}')
```

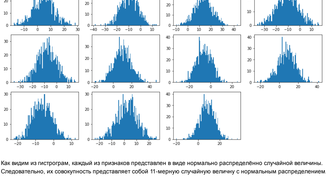
```
X shape: (1000, 11)
Xval shape: (100, 11)
yval shape: (100,)
```

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

In [17]:
```
SIZE = 11
plt.figure(figsize=(15, 8))

for i in range(SIZE):
    ax = plt.subplot(3, 4, i + 1)
    ax.hist(X[:, i], bins=100)

plt.show()
```



Как видим из гистограм, каждый из признаков представлен в виде нормально распределённо случайной величины. Следовательно, их совокупность представляет собой 11-мерную случайную величину с нормальным распределением.

10. Оцените параметры распределения случайной величины.

Данная функция возвращает математическое ожидание случайных величин и их ковариационную матрицу

In [18]:
```
def fit_multivariance(X):
    mu = X.mean(axis=0)
    X_norm = X - mu
    Sigma = np.dot(X_norm.T, X_norm) / X_norm.shape[0]
    return mu, Sigma
```

In [19]:
```
mu, Sigma = fit_multivariance(X)
```

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

In [20]:
```
def predict_multivariance(X, dist_params):
    mu, Sigma, eps = dist_params
    p = stats.multivariate_normal.pdf(X, mu, Sigma)
    res = p < eps
    return res.astype(int) if len(X.shape) > 1 else int(res)
```

In [21]:
```
eps_list = np.linspace(1e-25, 1e-15, 10000)
eps_mult = search_eps(X, Xval, yval, fit_multivariance, predict_multivariance, eps_list)
```

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

In [24]:
```
predictions = predict_multivariance(X, (mu, Sigma, eps_mult))
anomaly_count = np.count_nonzero(predictions)
print(f'Number of anomaly: {anomaly_count}')
print(f'Epsilon with max F1 score for 11-dimensional data: {eps_mult}')
```

```
Number of anomaly: 48
Epsilon with max F1 score for 11-dimensional data: 3.000301029702970e-19
```