Multivariate Gaussian for Anomaly Detection

- The multivariate normal distribution is a generalization of the univariate normal to two or more variables. In our case we have 28 variables.
- It is a distribution for random vectors of correlated variables, each element of which has a univariate normal distribution.
- If there is no correlation among variables, then the elements of the vectors are independent univariate normal random variables.
- Can be applied to those distributions that follow Gaussian.

Aim:

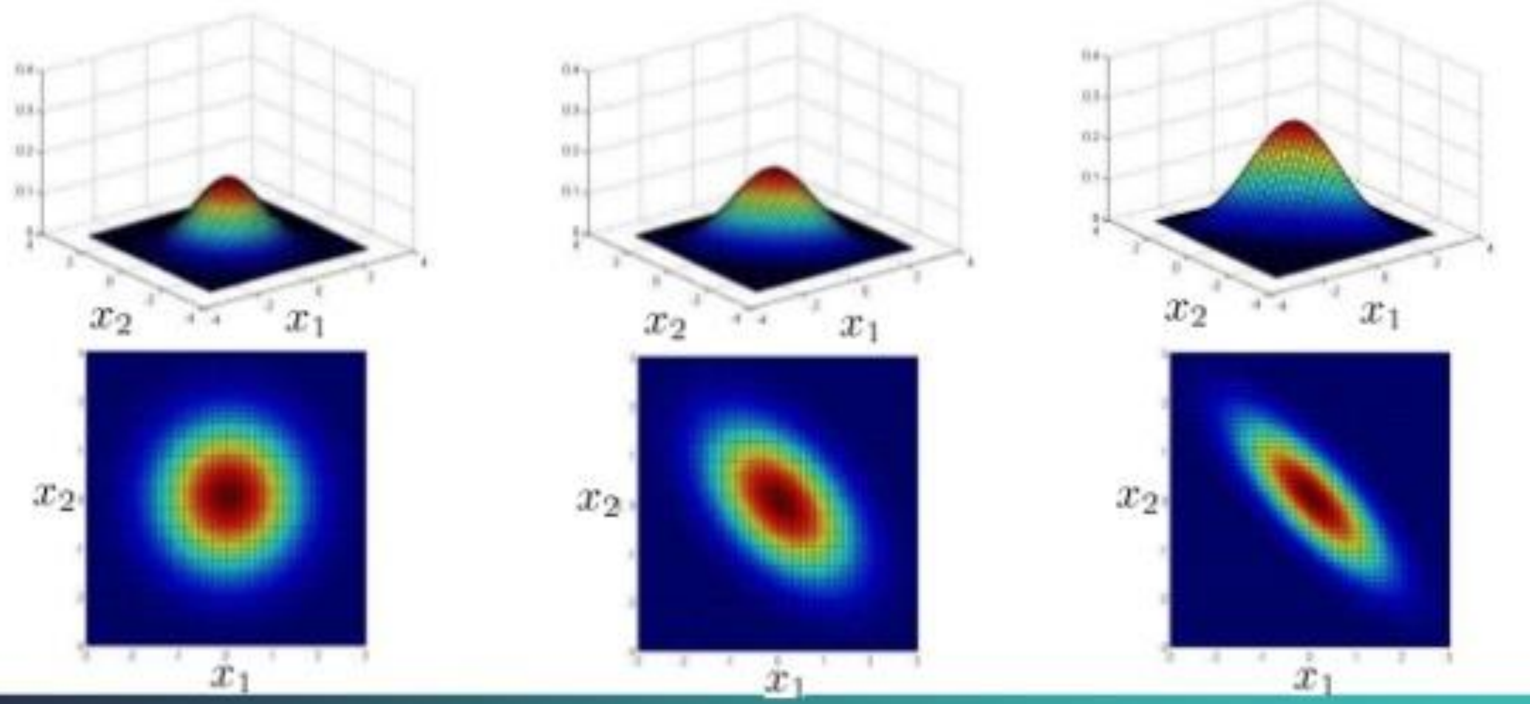
- Identify if the underlying distribution for ‘Credit Card fraud’ has Gaussian distribution?
- If so, identify the correlation matrix.
- After correlation matrix, apply the algorithm to identify anomalies.
- Determine how Gaussian distribution behaved using metrics.

What is Multivariate Gaussian/Normal

- It parameterized with a mean vector, μ, and a covariance matrix, Σ.
- Analogous to the mean μ and variance σ^2 parameters of a univariate normal distribution.
- The diagonal elements of Σ contain the variances for each variable, while the off-diagonal elements of Σ contain the covariances between variables.

Multivariate Gaussian (Normal) examples

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \qquad \mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}$$

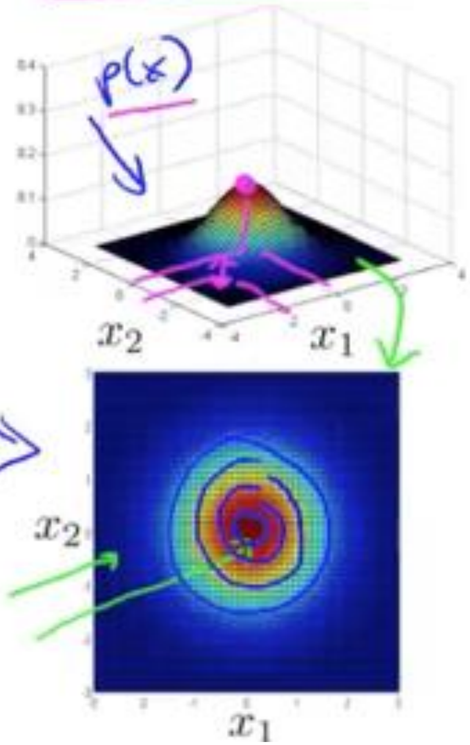


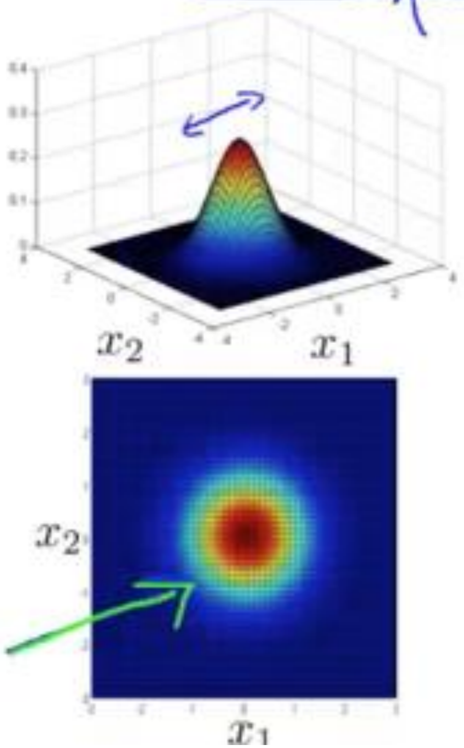
Multivariate Gaussian (Normal) examples

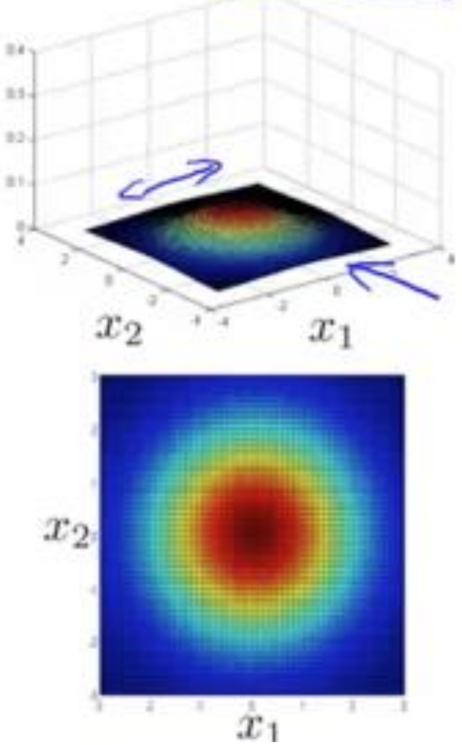
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\boxed{\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}$

$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\boxed{\Sigma = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}}$

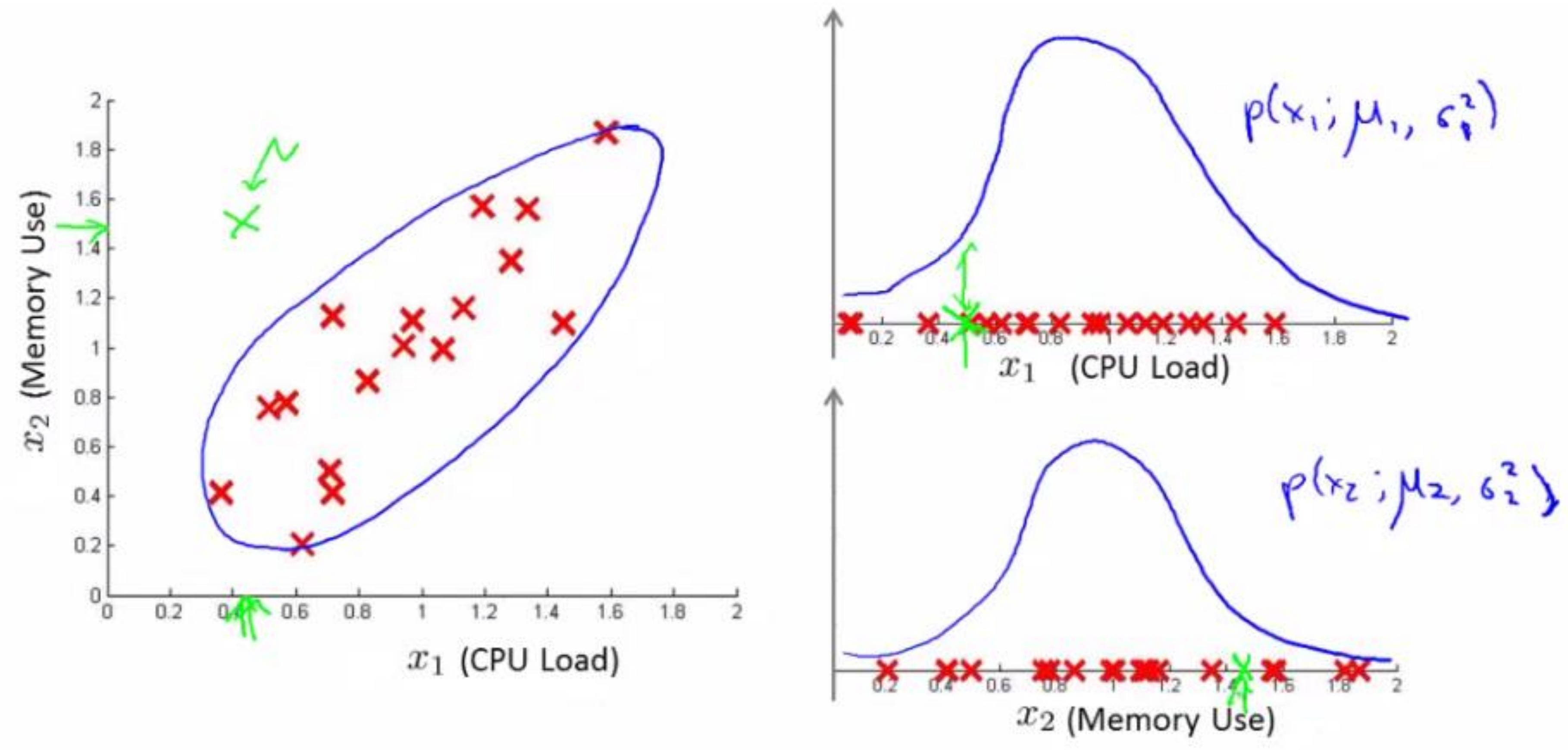
$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $\boxed{\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}}$





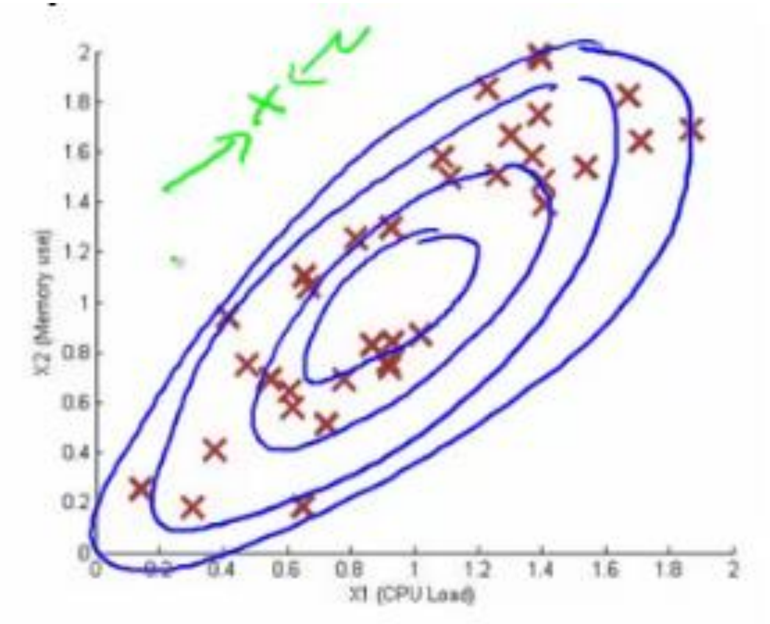


Why Multivariate Gaussian?



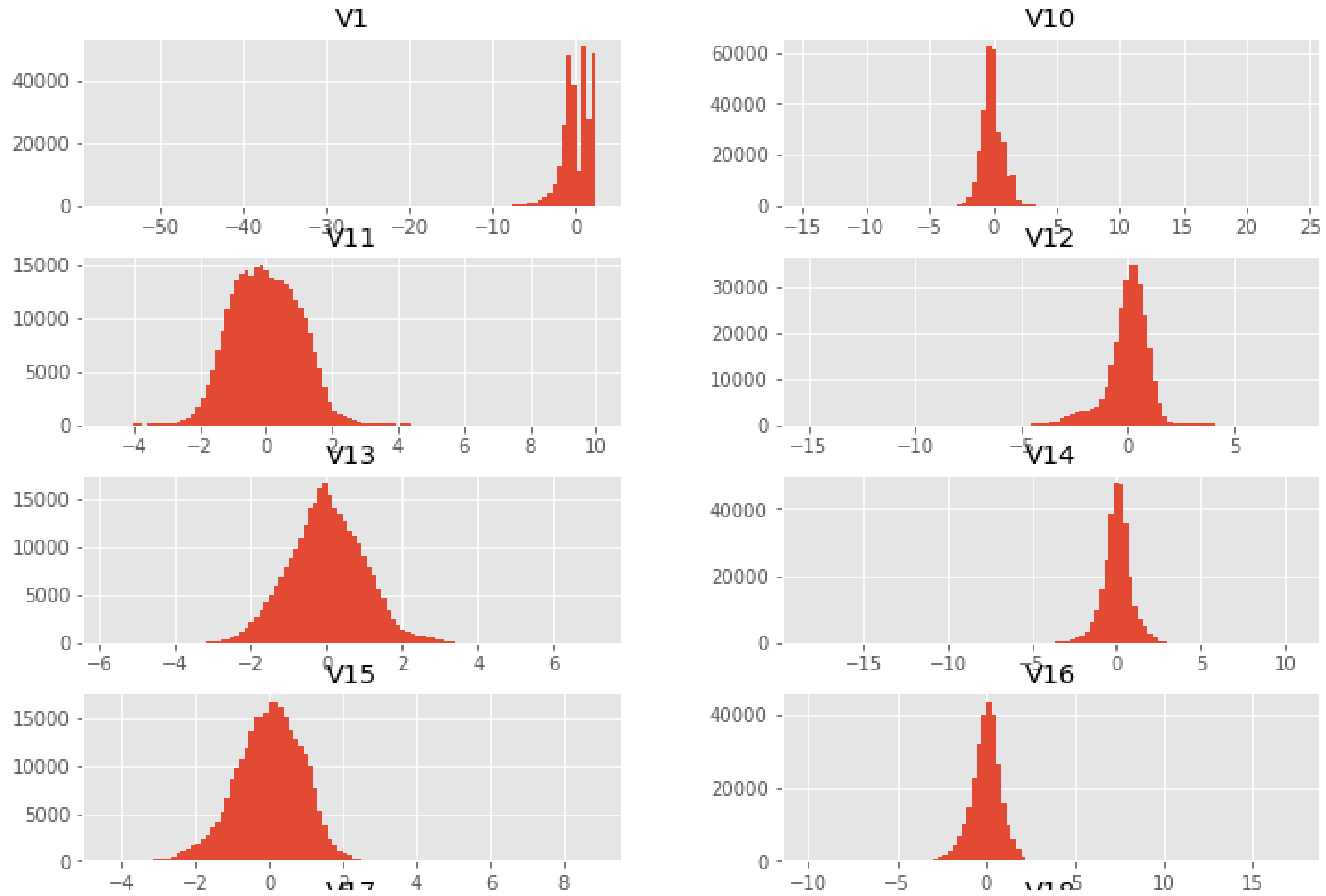
Why Multivariate Gaussian?

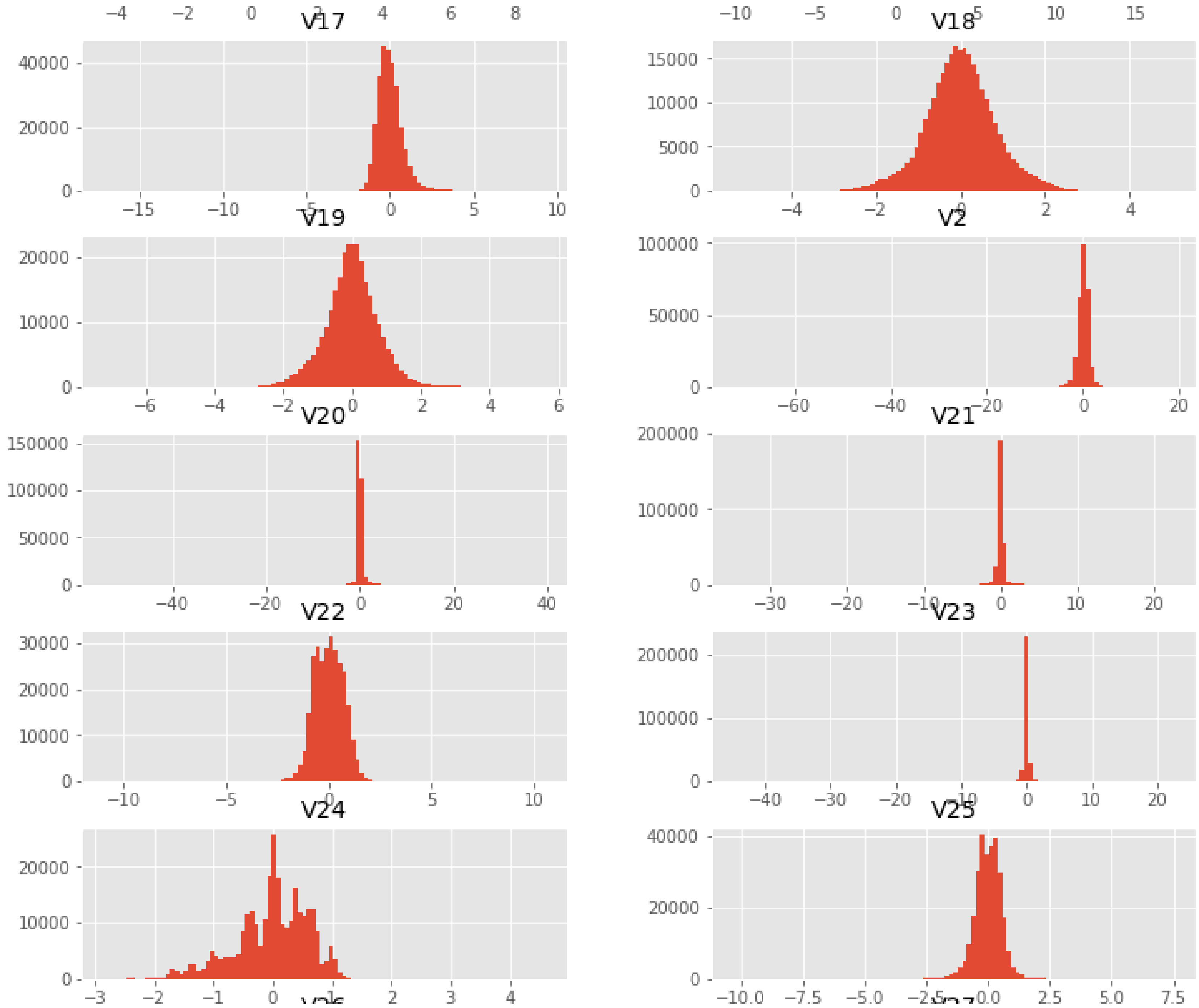
- We can change the correlation and mean matrix to identify our distribution correctly as seen before.
- Gaussians are convenient computationally
- Mixtures of Gaussians(Multivariate, non Multivariate) are sufficient to approximate a wide range of distributions



Is our distribution Gaussian?

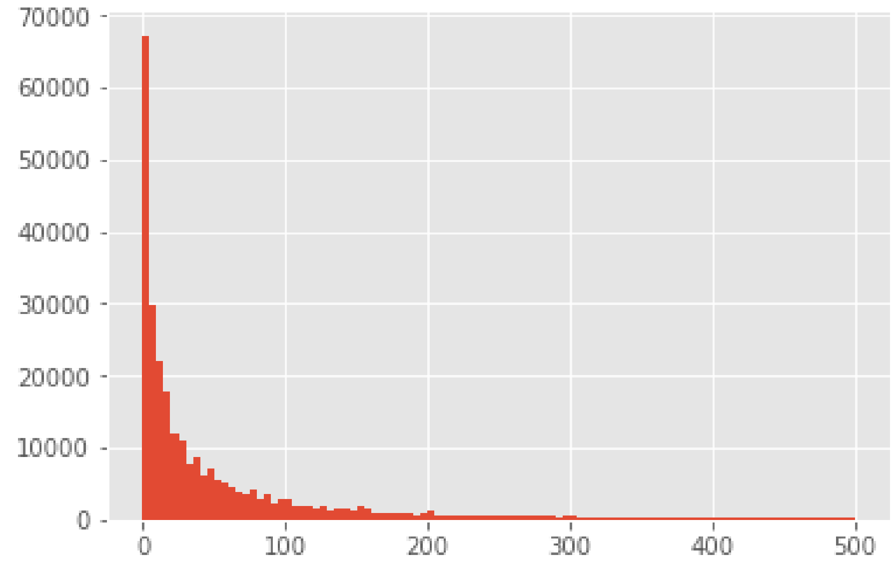
```
matplotlib.style.use('ggplot')
pca_columns = list(data)[1:-2]
normal_data[pca_columns].hist(stacked=False, bins=100, figsize=(12,30), layout=(14,2));
```




V17
V18
V19
V2
V20
V21
V22
V23
V24
V25

- Our distribution is fairly Gaussian.
- Feature V1 seems to behave a little differently and has the most deviance from normal distribution
- Visualization of the normal data:
-

```
normal_data["Amount"].loc[normal_data["Amount"] < 500].hist(bins=100);
```



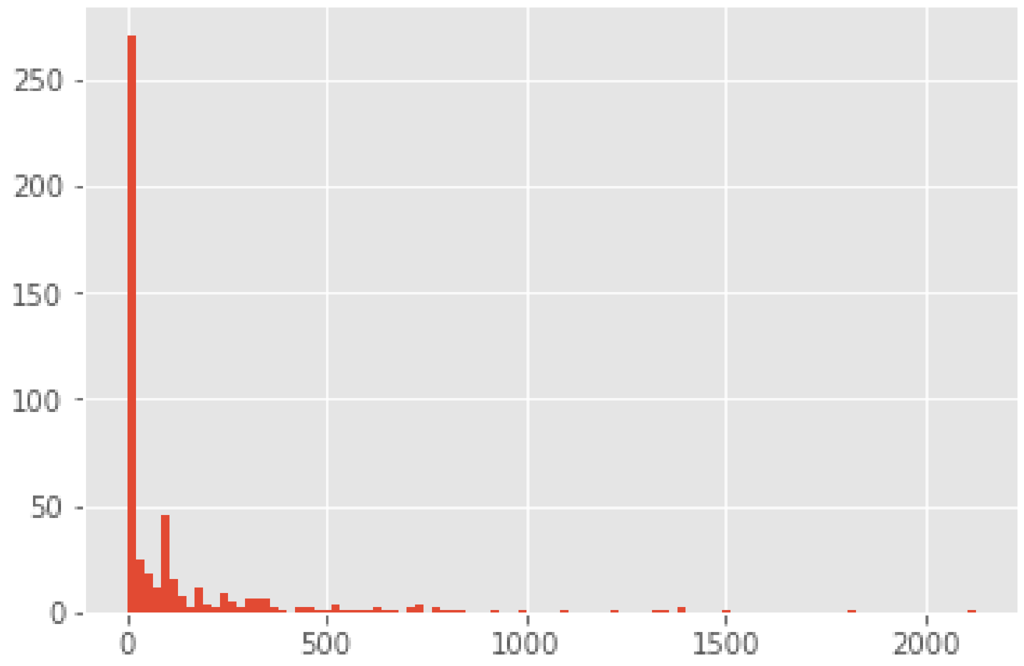

```
print("Mean", normal_data["Amount"].mean())
print("Median", normal_data["Amount"].median())
```

```
Mean 88.29102242225574
Median 22.0
```

A vast majority of normal withdrawals are btn 10-100$.

- Visualization of fraudulent data:

```
fraud_data["Amount"].hist(bins=100);
```



- The median is lower but the mean is higher for the fraudulent cases.
- This suggests there are some high value oriented criminals and some that focus on withdrawals below a limit to avoid detection.

```
print("Mean Fraudulent", fraud_data["Amount"].mean())
print("Median Fraudulent", fraud_data["Amount"].median())
```

```
Mean Fraudulent 122.21132113821133
Median Fraudulent 9.25
```

- Part 1 of our aim is done. Ie, Gaussian distribution can be useful in our case for fraud detection.
- Part 2: Determine the covariance matrix.
- A pair of random variables X and Y , their covariance is defined as: $Cov[X, Y] = E[(X - E[X])(Y - E[Y])] = E[XY] - E[X]E[Y]$.
- The covariance matrix,which we usually denote as Σ, is the $n \times n$ matrix whose (i, j)th entry is $Cov[X_i, X_j]$.
- We need a covariance matrix function in a hypersphere in higher dimensions which will then be used for the Multivariate Gaussian function.

General Algorithm

- A nx1 mean vector and a nxn covariance matrix.
- Generate a bunch of uniform random numbers and convert them into a Gaussian random number with a known mean and standard deviation.
- Do the previous step n times to generate an n-dimensional Gaussian vector with a known mean and covariance matrix.
- Transform this random Gaussian vector so that it lines up with the mean and covariance provided by the user.

- For our case, we do not have any inbuilt function for covariance matrix, so we need to write the function explicitly.

```
def covariance_matrix(X):
    m, n = X.shape
    tmp_mat = np.zeros((n, n))
    mu = X.mean(axis=0)
    for i in range(m):
        tmp_mat += np.outer(X[i] - mu, X[i] - mu)
    return tmp_mat / m
```

```
cov_mat = covariance_matrix(X_train)
```

```
cov_mat_inv = np.linalg.pinv(cov_mat)
cov_mat_det = np.linalg.det(cov_mat)
def multi_gauss(x):
    n = len(cov_mat)
    return (np.exp(-0.5 * np.dot(x, np.dot(cov_mat_inv, x.T)))
            / (2. * np.pi)**(n/2.)
            / np.sqrt(cov_mat_det))
```

This python code is used to represent the equation:

$$\frac{1}{\sqrt{|\Sigma|(2\pi)^d}} \exp\left(-\frac{1}{2}(x\text{-}\mu)\,\Sigma^{-1}(x\text{-}\mu)'\right)$$

where *x* and *μ* are 1-by-*n* vectors and Σ is a *n*-by-*n* covariance matrix.

Next week

- How did our code work in determining the frauds?
- Creation of metrics for comparison.