PCA for anonymization

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Data engineer

Principal component analysis (PCA)

Dimensionality-reduction method that is often used to reduce the dimensionality of large datasets.

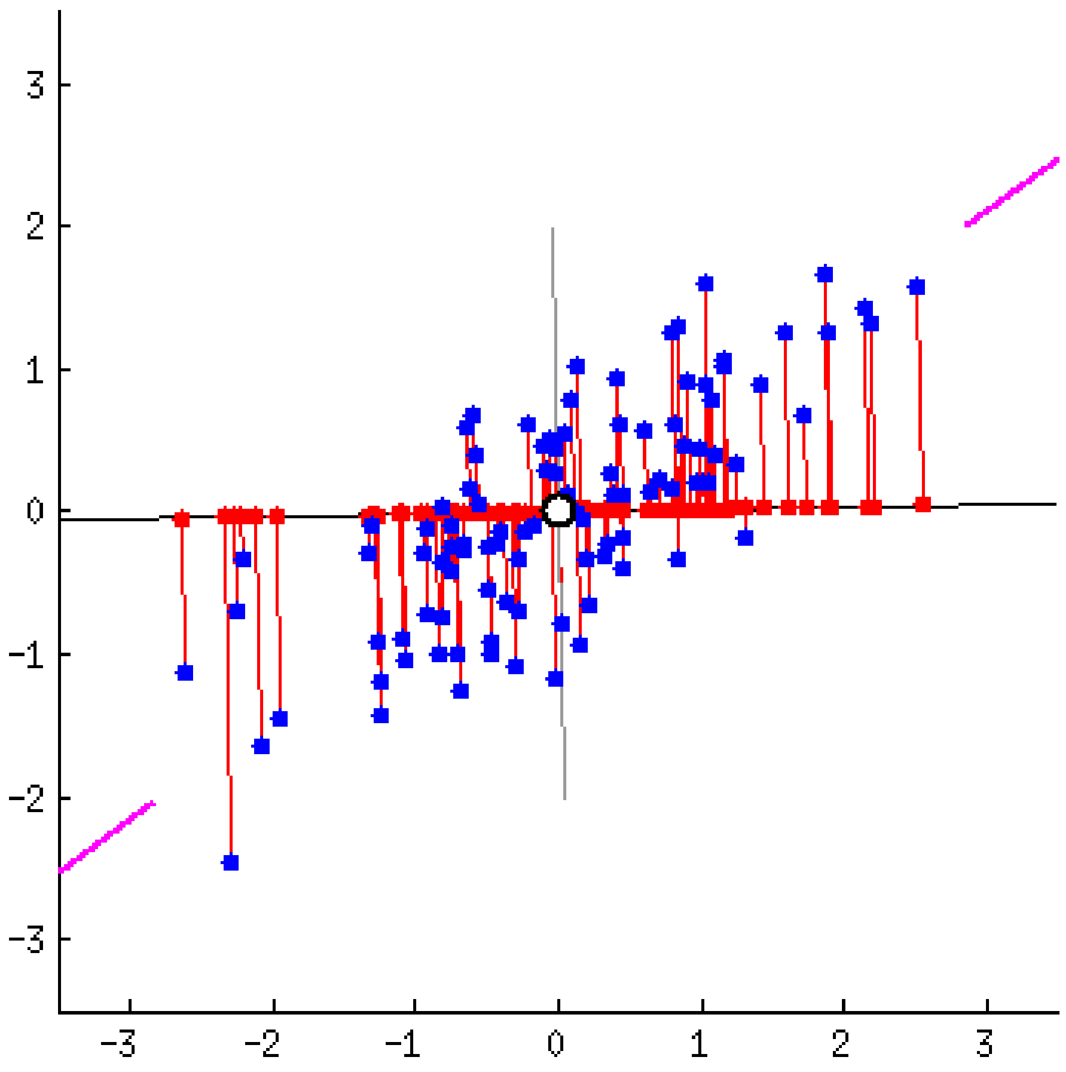
Data masking with PCA

- PCA creates new "principal components" from linear transformations of the original features.
- In a dataset of beers, PCA will constructs new features. For example:

$$2 \times AlcoholicVolume - BitternessLevel$$

Data masking with PCA

New projections of the data



Data masking with PCA

PCA without the dimensionality reduction

- It's only a rotation of the original space in the dataset.
- This means that the distances are preserved.
- An advantage for predictive tasks and algorithms.

Data masking with PCA

- If those resulting values are released without explanation, algorithms could still be trained with these and make accurate predictions.
- **Adversaries would not know how to interpret those masked values.**

Data masking with PCA

```
# Explore the dataset
heart_df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Data masking with PCA and Scikit-learn

```
# Obtain the data without the target column
x_data = df.drop(['target'], axis = 1)

# Target column as array of values
y = df.target.values
```

Data masking with PCA and Scikit-learn

```
# Import PCA from Scikit-learn
from sklearn.decomposition import PCA

# Initialize PCA with number of components to be the same as the number of columns
pca = PCA(n_components=len(x_data.columns))

# Apply PCA to the data
x_data_pca = pca.fit_transform(x_data)
```

Data masking with PCA and Scikit-learn

```python
# See the data
x_data_pca
```

```
array([[-1.22673448e+01,  2.87383781e+00,  1.49698788e+01, ...,
         7.31102828e-01, -2.90393586e-01,  5.12575925e-01],
       [ 2.69013712e+00, -3.98713736e+01,  8.77882303e-01, ...,
         4.04206943e-01, -4.25920179e-01, -1.48124511e-01],
       [-4.29502141e+01, -2.36368199e+01,  1.75944589e+00, ...,
        -9.15397287e-01,  2.17828257e-01,  7.97593843e-02],
       ...,
```

Data masking with PCA and Scikit-learn

```
# Create a DataFrame from the resulting PCA transformed data
df_x_data_pca = pd.DataFrame(x_data_pca)

# Inspect the shape of the dataset
df_x_data_pca.shape
```

```
(1213, 13)
```

Data utility after PCA data masking

- Perform classification with logistics regression and look for accuracy loss by comparing original and resulting data.

- **Logistic regression is a classification algorithm** used to predict a binary outcome based on a set of independent variables.

Data utility after PCA data masking

Perform classification with logistic regression and look for accuracy loss.

```
# Split the resulting dataset into training and test data
x_train, x_test, y_train, y_test = train_test_split(x_data_pca, y, test_size=0.2)

# Create the model
lr = LogisticRegression(max_iter=200)

# Fit train the model
lr.fit(x_train,y_train)

# Run the model and perform predictions to obtain accuracy score
acc = lr.score(x_test, y_test) * 100
print("Test Accuracy is ", acc)
```

```
Test Accuracy is 85.24590163934425
```

Data utility before PCA data masking

Perform classification with logistic regression and see the score with the original data

```
# Split the resulting dataset into training and test data
x_train, x_test, y_train, y_test = train_test_split(x_data.to_numpy(),y,test_size = 0.2)

# Create the model
lr = LogisticRegression(max_iter=200)

# Fit train the model
lr.fit(x_train,y_train)

# Run the model and perform predictions to obtain accuracy score
acc = lr.score(x_test,y_test) * 100
print("Test Accuracy is ", acc)
```

```
Test Accuracy is 85.24590163934425
```

Let's practice!

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Generating realistic datasets with Faker

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Data engineer

Generating data with Faker

```
fake_data.name()
```

```
'Kelly Clark'
```

```
fake_data.name_male()
```

```
'Antonio Henderson'
```

```
fake_data.name_female()
```

```
'Jennifer Ortega'
```

Clients DataFrame

```
clients_df
```

```
gender    active
0    Female    No
1    Male      Yes
2    Male      No
3    Female    Yes
4    Male      Yes
...    ...    ...
1465    Male    Yes
1466    Male    Yes
1467    Male    Yes
1468    Male    Yes
1469    Male    Yes
1470 rows × 2 columns
```

Generating a dataset with Faker

- Generate unique names that are consistent with gender
- Generating random cities
- Generating specified cities that follow a probability distribution
- Generating e-mails
- Generating dates in a time range

Making names match their gender

Generating unique names in the dataset

Avoiding duplicates

```python
# Import the Faker class
from faker import Faker

# Initialize a Faker class
fake_data = Faker()

# Generate a name according to the gender, that will be unique in the dataset
clients_df['name'] = [fake_data.unique.name_female() if x == "Female"
                      else fake_data.unique.name_male()
                      for x in clients_df['gender']]
```

Making names match their gender

```
# Explore the dataset
clients_df
```

```
    gender    active    name
0    Female    No        Michelle Lang
1    Male      Yes       Robert Norton
2    Male      No        Matthew Brown
3    Female    Yes       Sherry Jones
4    Male      Yes       Steven Vega
...    ...    ...    ...
1465    Male    Yes    Bradley Smith
1466    Male    Yes    Tyler Yu
1467    Male    Yes    Mr. Joshua Gallegos
1468    Male    Yes    Brian Aguilar
1469    Male    Yes    David Johnson
1470 rows × 3 columns
```

Generating a random city

```python
# Generating a random city
clients_df['city'] = [fake_data.city()
                      for x in range(len(clients_df))]

clients_df.head()
```

```
   Gender   Active   Name             City
0   Female  No       Stacy Hooper     Reedland
1   Male    Yes      Michael Rogers   North Michellestad
2   Male    No       James Sanchez    West Josephburgh
3   Female  Yes      Taylor Berger    Hermanton
4   Male    Yes      Joshua Coleman   South Amandaland
```

Generating emails

```python
# Generating emails with different domains
clients_df['contact email'] = [fake_data.company_email()
                               for x in range(len(clients_df))]


# Explore the dataset
clients_df.head()
```

```
    gender    active    name              city                  contact email
0    Female   No        Stacy Hooper      Reedland              mendozamegan@davis.com
1    Male     Yes       Michael Rogers    North Michellestad    julianweeks@parker-mcknight.com
2    Male     No        James Sanchez     West Josephburgh      brianbecker@rivera.org
3    Female   Yes       Taylor Berger     Hermanton             xwilkins@morgan-knapp.com
4    Male     Yes       Joshua Coleman    South Amandaland      lindsay11@tran-kennedy.net
```

Generating emails

```python
# Generating emails with company like domains and username similar to name
clients_df['Contact email'] = [x.replace(" ", "") + "@" +
                               fake_data.domain_name()
                               for x in clients_df['Name']]


# Explore the resulting DataFrame
clients_df.head()
```

```
    gender    active   name            city                contact email
0    Female   No       Stacy Hooper    Reedland            StacyHooper@peterson.com
1    Male     Yes      Michael Rogers  North Michellestad  MichaelRogers@clarke.com
2    Male     No       James Sanchez   West Josephburgh    JamesSanchez@yates-tanner.inf
3    Female   Yes      Taylor Berger   Hermanton           TaylorBerger@johnson.org
4    Male     Yes      Joshua Coleman  South Amandaland    JoshuaColeman@tucker.info
```

Generating dates

Dates between two times

```
# Generating dates between two times
clients_df['date'] = [fake_data.date_between(start_date="-10y", end_date="now")
                              for x in range(len(clients_df))]
# Explore the resulting DataFrame
clients_df.head()
```

	gender	active	name	city	contact email	date
0	Female	No	Stacy Hooper	Reedland	StacyHooper@peterson.com	2019-11-20
1	Male	Yes	Michael Rogers	North Michellestad	MichaelRogers@clarke.com	2015-02-22
2	Male	No	James Sanchez	West Josephburgh	JamesSanchez@yates-tanner.inf	2015-12-11
3	Female	Yes	Taylor Berger	Hermanton	TaylorBerger@johnson.org	2012-12-13
4	Male	Yes	Joshua Coleman	South Amandaland	JoshuaColeman@tucker.info	2014-05-22

Generating cities following a probabilistic distribution

When imitating a real dataset, we can avoid leaking the real names of values.

```
# Import numpy
import numpy as np

# Obtain or specify the probabilities
p = (0.58, 0.23, 0.16, 0.03)
cities = ["New York", "Chicago", "Seattle", "Dallas"]

# Generate the cities from the selected ones following a distribution
clients_df['city'] = np.random.choice(cities, size=len(clients_df), p=p)
```

Generating cities following a probabilistic distribution

```
# See the resulting dataset
clients_df.head()
```

```
    gender    active    name             city        contact email                      date
0    Female   No        Stacy Hooper     Chicago     StacyHooper@peterson.com           2019-11-20
1    Male     Yes       Michael Rogers   New York    MichaelRogers@clarke.com           2015-02-22
2    Male     No        James Sanchez    New York    JamesSanchez@yates-tanner.inf      2015-12-11
3    Female   Yes       Taylor Berger    Chicago     TaylorBerger@johnson.org           2012-12-13
4    Male     Yes       Joshua Coleman   New York    JoshuaColeman@tucker.info          2014-05-22
```

Let's generate datasets!

DATA PRIVACY AND ANONYMIZATION IN PYTHON

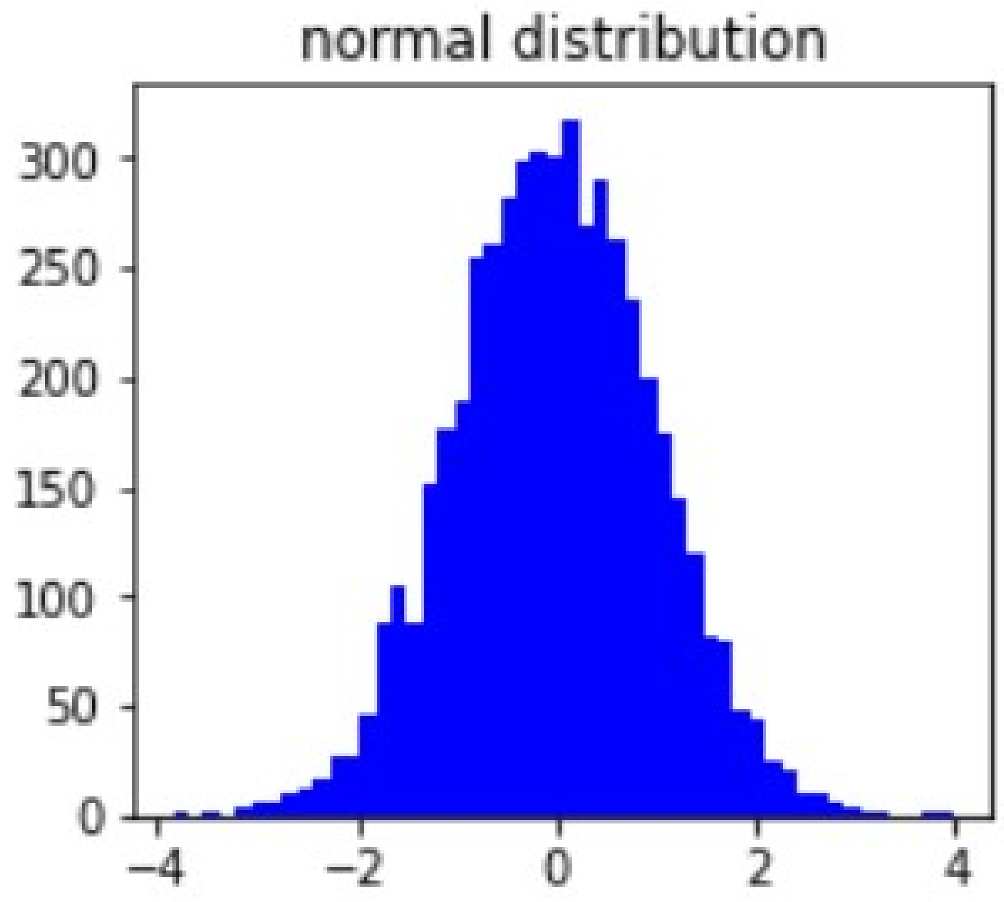
Creating synthetic datasets using scikit-learn

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Data engineer

Generating datasets with Scikit-learn

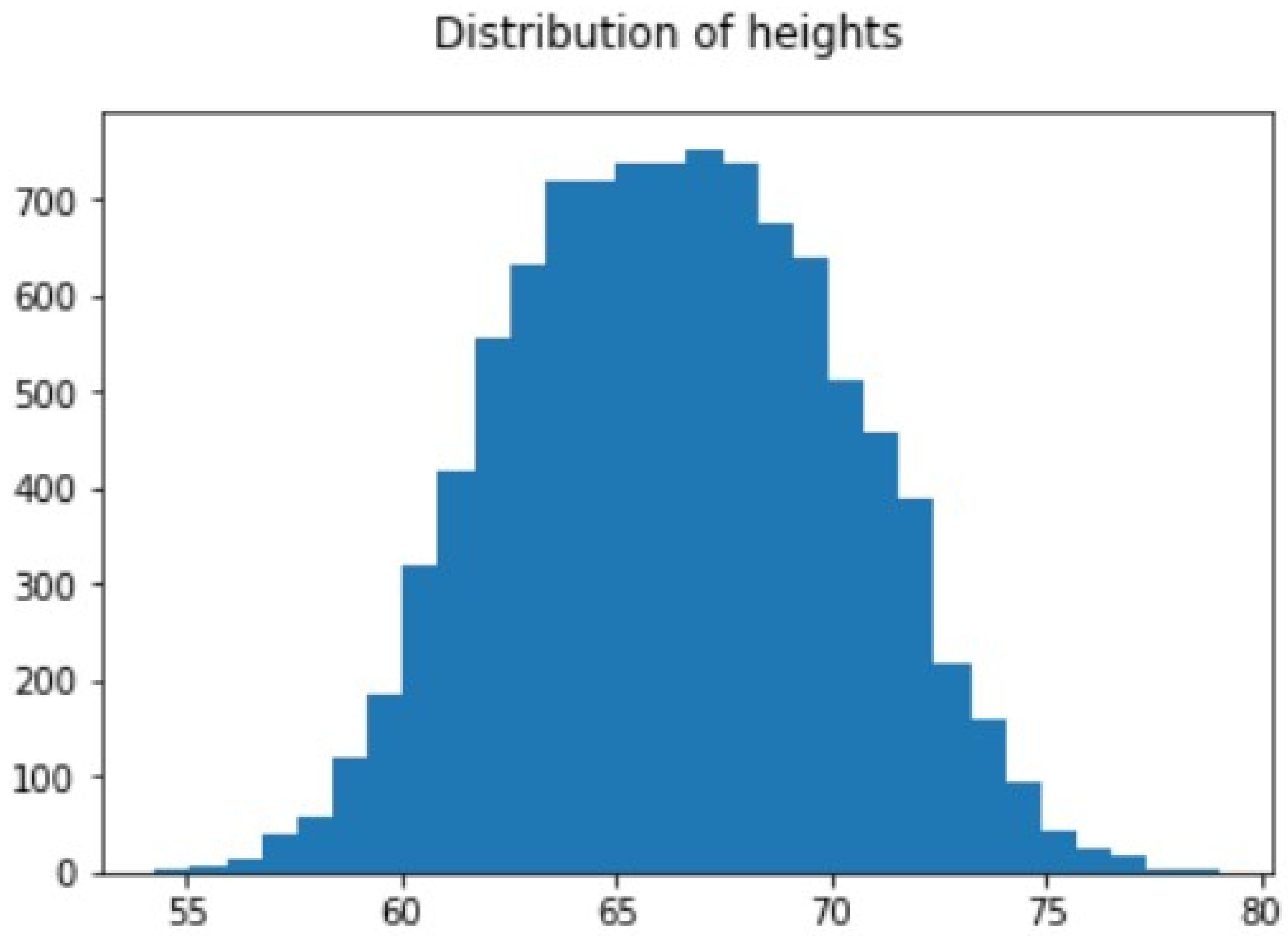
- We can create datasets that sample from probability distributions
- Such as the normal distribution



Normal distribution

Often occur in nature

- Heights
- Blood pressure
- IQ scores

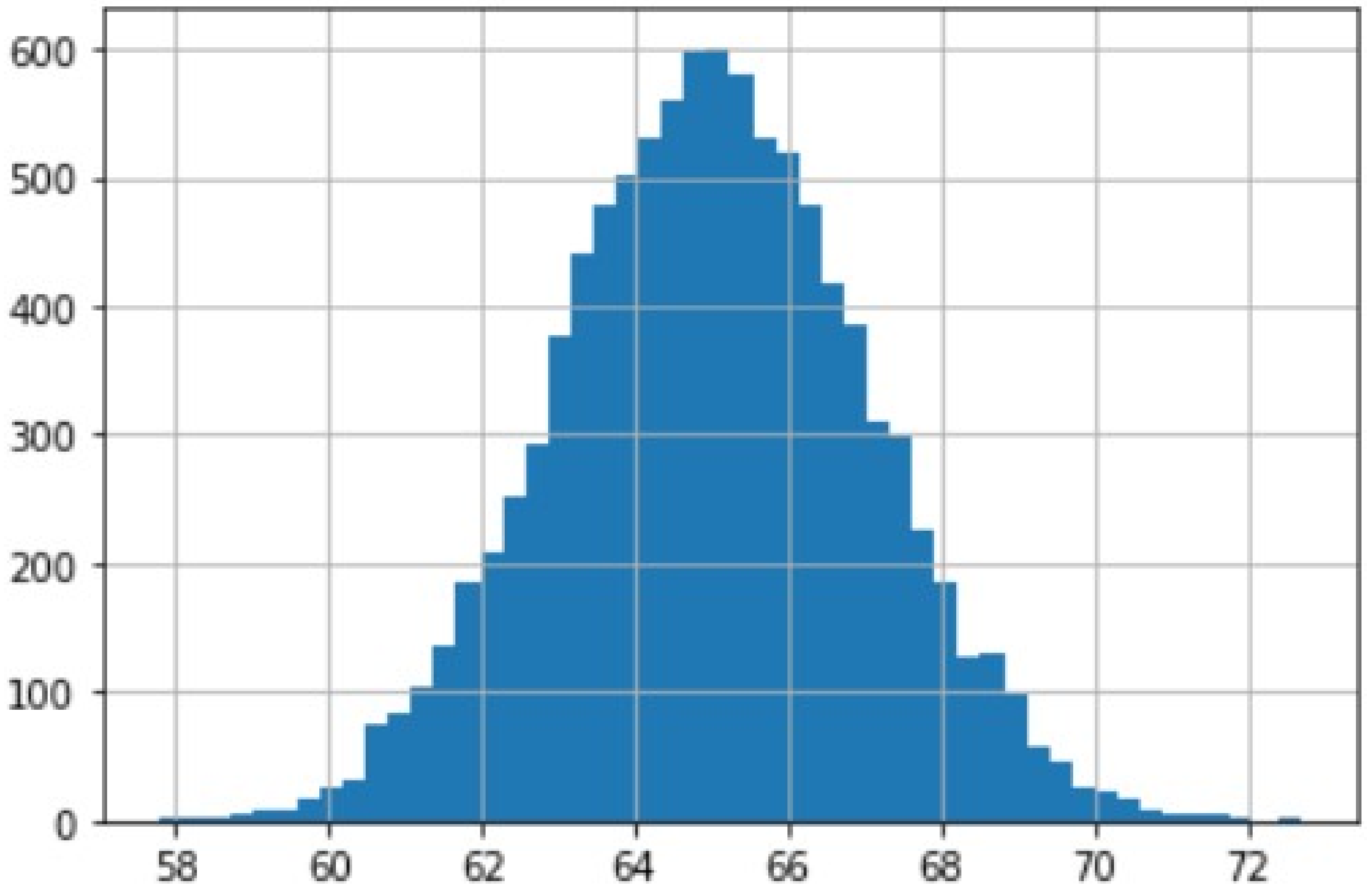


Sample from a normal distribution

```
import numpy as np

# Create new pandas DataFrame
new_measures = pd.DataFrame()

# Selecting the mean/center values and the standard deviation of the sample
mean = 65
standard_deviation = 2

# Generating the sample
new_measures['Height'] = np.random.normal(mean, standard_deviation, 10000)
```

Sample from a normal distribution

```
# Draw histogram to see the resulting heights distribution
new_measures['Height'].hist(bins=50)
```



Creating datasets using scikit-learn

Scikit-learn has simple and easy-to-use functions for generating datasets to perform:

- Classification
- Clustering
- Regression

Synthetic data for classification and clustering

`make_classification()`

- It allocates normally-distributed clusters of points
- Can create correlated and uninformative features

`make_blobs()`

- Greater control regarding the centers and standard deviations of clusters

Synthetic data for classification

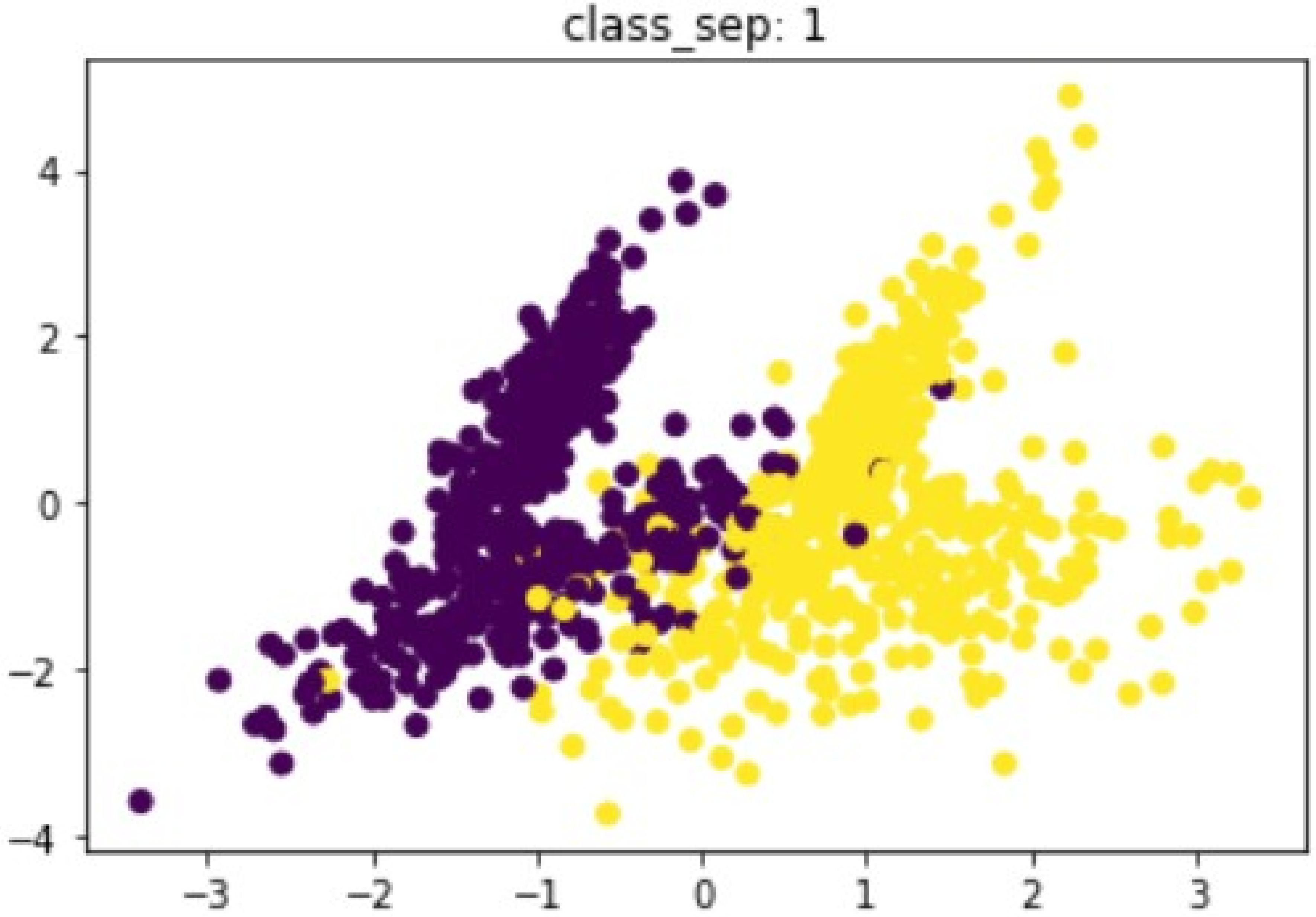
```python
# Import make_classification from sklearn datasets module
from sklearn.datasets import make_classification

# Generate the samples and their labels
x, y = make_classification(n_samples=1000,
                           n_classes=2,
                           n_informative=2,
                           n_features=4,
                           n_clusters_per_class=2,
                           class_sep=1)
```

Synthetic data for classification

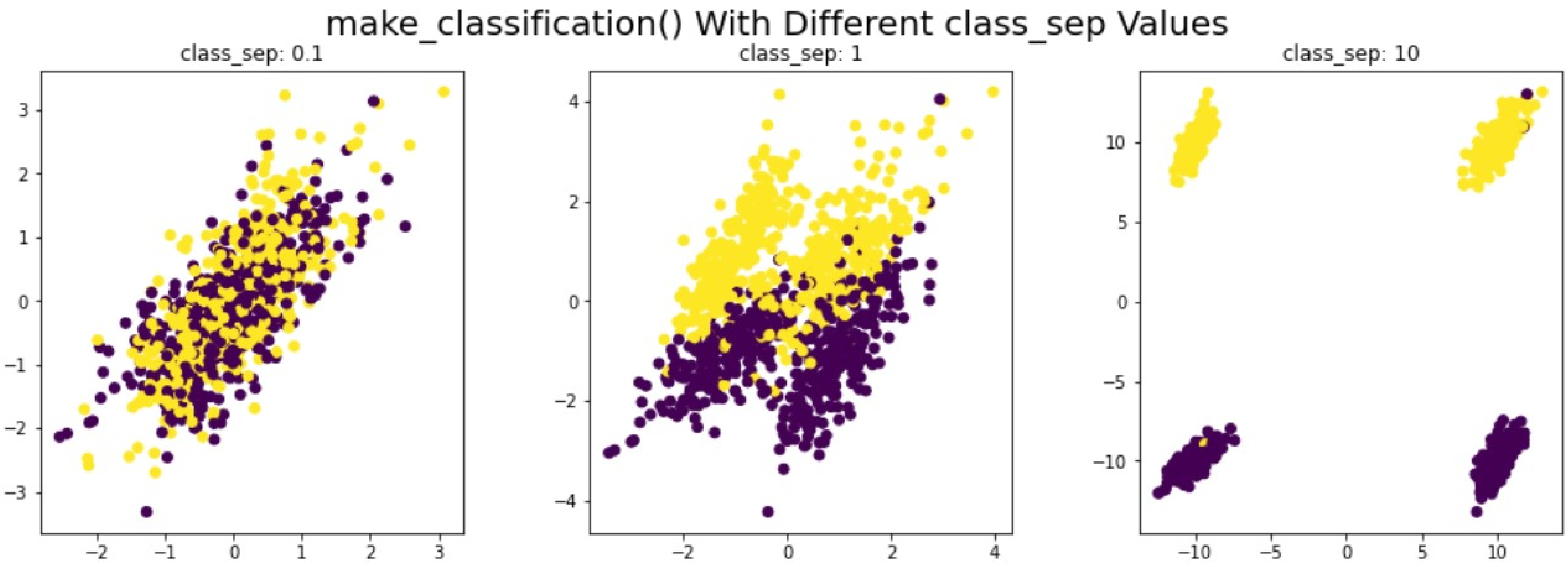
```
# See the generated data and labels
print(x.shape)
print(y.shape)
print(x)
```

```
(1000, 4)
(1000,)
[[ 1.22914870e+00 -2.62386795e+00  2.25878743e+00  2.55377055e+00]
 [-1.10279812e+00 -1.15816087e+00  1.55571279e+00  7.80565898e-02]
 [ 2.65581977e-03 -2.33278818e+00  2.37837858e+00  1.57533194e+00]
 ...
 [ 4.51006972e-01  7.53435745e-01 -9.21597108e-01 -2.20659747e-01]
 [ 5.31925876e-01  7.42210504e-01 -9.37625248e-01 -1.61488855e-01]
 [ 1.62862108e+00 -2.72435345e+00  2.22562940e+00  2.87628246e+00]]
```

Synthetic data for classification



Synthetic data for classification

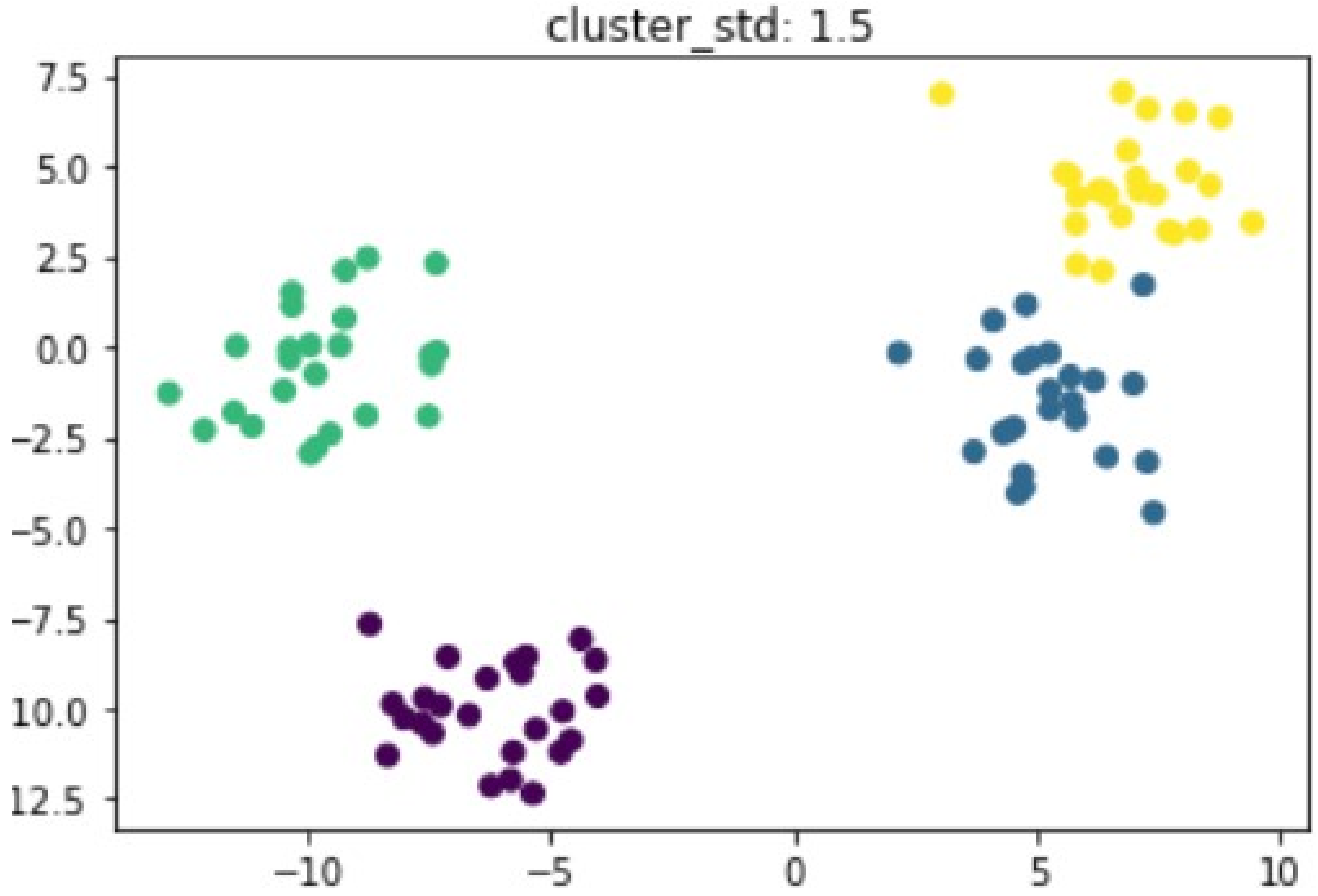


Synthetic data for clustering

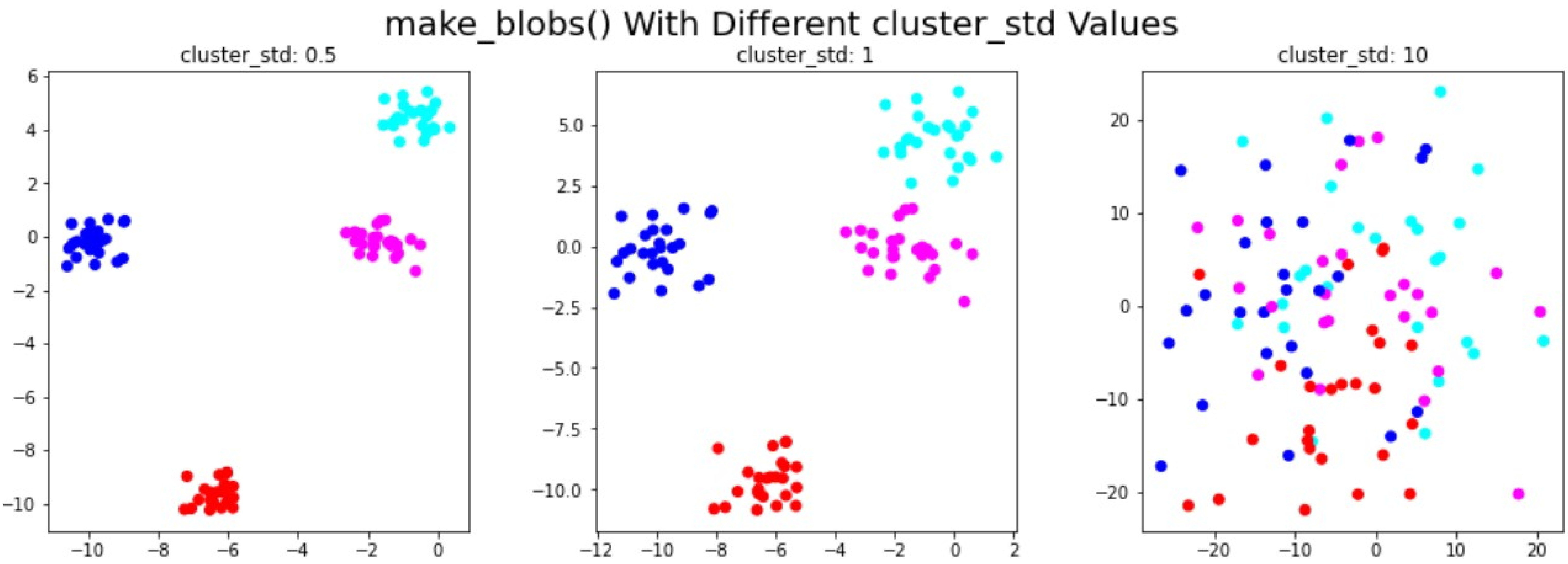
```python
# Import the datasets module for generating clustering datasets
from sklearn.datasets import make_blobs

# Specify a value for standard deviation
standard_deviation = 1.5

# Generate the data and labels of the dataset
x, labels = make_blobs(n_features=3,
                       centers=4,
                       cluster_std=standard_deviation)

# See the shape of the generated data
print(x.shape)
```

```
(100, 3)
```

Synthetic data for clustering



Synthetic data for clustering



Let's practice!

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Safely release datasets to the public

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Data engineer

Exploring datasets

For analyzing possible privacy concerns, first obtain some domain and statistical knowledge of them.

```
# Explore the dataset
cross_selling.head()
```

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vintage	Response
0	1	Male	44	1	28.0	0	> 2 Years	217	1
1	2	Male	76	1	3.0	0	1-2 Year	183	0
2	3	Male	47	1	28.0	0	> 2 Years	27	1
3	4	Male	21	1	11.0	1	< 1 Year	203	0
4	5	Female	29	1	41.0	1	< 1 Year	39	0

Exploring datasets

```
# Explore the dataset
cross_selling.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381109 entries, 0 to 381108
Data columns (total 9 columns):
 #   Column              Non-Null Count   Dtype
     ------              --------------   -----
 0   id                  381109 non-null  int64
 1   Gender              381109 non-null  object
 2   Age                 381109 non-null  int64
 3   Driving_License     381109 non-null  int64
 4   Region_Code         381109 non-null  float64
 5   Previously_Insured  381107 non-null  int64
 6   Vehicle_Age         381109 non-null  object
 7   Vintage             381109 non-null  int64
 8   Response            381109 non-null  int64
dtypes: float64(1), int64(6), object(2)
memory usage: 26.2+ MB
```

Exploring datasets

```
# Calculate the number of unique values in a DataFrame
cross_selling.nunique()
```

```
id                    381109
Gender                     2
Age                       66
Driving_License            2
Region_Code               53
Previously_Insured         2
Vehicle_Age                3
Vintage                  290
Response                   2
dtype: int64
```

Suppressing unique attributes

```
# Apply attribute suppression on the id column
suppressed_df = cross_selling.drop('id', axis="columns")

# Check the head of the resulting DataFrame
suppressed_df.head()
```

	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vintage	Response
0	Male	44	1	28.0	0	> 2 Years	217	1
1	Male	76	1	3.0	0	1-2 Year	183	0
2	Male	47	1	28.0	0	> 2 Years	27	1
3	Male	21	1	11.0	1	< 1 Year	203	0
4	Female	29	1	41.0	1	< 1 Year	39	0

Cleaning data

```
# Drop null and NaN rows
cleaned_df = suppressed_df.dropna(axis="index")

cleaned_df.head()
```

	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vintage	Response
0	Male	44	1	28.0	0	> 2 Years	217	1
1	Male	76	1	3.0	0	1-2 Year	183	0
2	Male	47	1	28.0	0	> 2 Years	27	1
3	Male	21	1	11.0	1	< 1 Year	203	0
4	Female	29	1	41.0	1	< 1 Year	39	0

Sampling from categorical values

```
# Compute the probability distribution
cleaned_df['Gender'].value_counts(normalize=True)
```

```
Male      0.540957
Female    0.459043
Name: Gender, dtype: float64
```

Sampling from categorical values

```
# Get the probability distribution values
distributions = cleaned_df['Gender'].value_counts(normalize=True)

# Sample from the calculated probability distributions
cleaned_df['Gender'] = np.random.choice(distributions.index,
                                        p=distributions,
                                        size=len(cleaned_df))
```

Sampling from categorical values

```
# See the resulting dataset
cleaned_df
```

```
    Gender   Age   Driving_License   Region_Code   Previously_Insured   Vehicle_Age   Vintage   Response
0    Male    44    1                 28.0          0                    > 2 Years     217        1
1    Male    76    1                 3.0           0                    1-2 Year      183        0
2    Male    47    1                 28.0          0                    > 2 Years     27         1
3    Female  21    1                 11.0          1                    < 1 Year      203        0
4    Male    29    1                 41.0          1                    < 1 Year      39         0
...    ...    ...    ...    ...    ...    ...    ...    ...
381107 rows × 8 columns
```

Sampling from categorical values

```
# Compute the probability distribution
cleaned_df['Gender'].value_counts(normalize=True)
```

```
Male      0.541973
Female    0.458027
Name: Gender, dtype: float64
```

Removing column names

```
# Replace column names with numbers
cleaned_df.columns = range(len(df.columns))
```

	0	1	2	3	4	5	6	7
0	Male	44	1	28.0	0	> 2 Years	217	1
1	Male	76	1	3.0	0	1-2 Year	183	0
2	Male	47	1	28.0	0	> 2 Years	27	1
3	Female	21	1	11.0	1	< 1 Year	203	0
4	Male	29	1	41.0	1	< 1 Year	39	0

Let's practice!

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Great work!

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Data engineer

You have completed the course

Recap: What you have learned

- Sensitive and non-sensitive personally identifiable information (PII)
- Quasi-identifiers
- Linkage attacks
- Data suppression
- Data masking
- Data generalization
- Synthetic data generating
- Sampling from probability distributions for different type of attributes

Privacy models: k-anonymity

- K-anonymous datasets
- Exploring possible combinations in the dataset
- Generalizing data using hierarchies and ranges
- Avoid re-identification attacks
- **Without falsifying or randomizing data!**

Privacy models: differential privacy

- Differential privacy systems can measure and quantify privacy in data releases
- One of the most important definitions of privacy in present time

Differentially private models and operations

- People are increasingly working with differentially private machine and deep learning models
- Trained and run different type of differentially private machine learning models!
- Practiced advanced concepts such as privacy budget and tracking

Other interesting libraries

- Google's differential privacy
- TensorFlow Privacy
- ARX Data Anonymization Tool

Congrats!

DATA PRIVACY AND ANONYMIZATION IN PYTHON