K nearest neighbours

September 29, 2023

1 Making a dataframe and primary initializing

[11]:
```
import pandas as pd
from sklearn import datasets

import numpy as np
np.random.seed(254)

X, y = datasets.make_classification(1000, n_features=2, n_informative=2,␣
 ↪n_redundant=0, n_repeated=0, n_classes=3,
                                    n_clusters_per_class=1)

df = pd.DataFrame({"feature_1": X[:,0], "feature_2": X[:,1], "category":y},
                  columns=["feature_1", "feature_2", "category"])
df.shape
```

[11]:
```
(1000, 3)
```

[12]:
```
df.head()
```

[12]:

	feature_1	feature_2	category
0	1.316455	1.906756	0
1	0.499191	-1.668219	2
2	0.937946	0.834301	0
3	0.930180	0.648558	0
4	0.309252	-1.365109	2

[13]:
```
df.describe()
```

[13]:

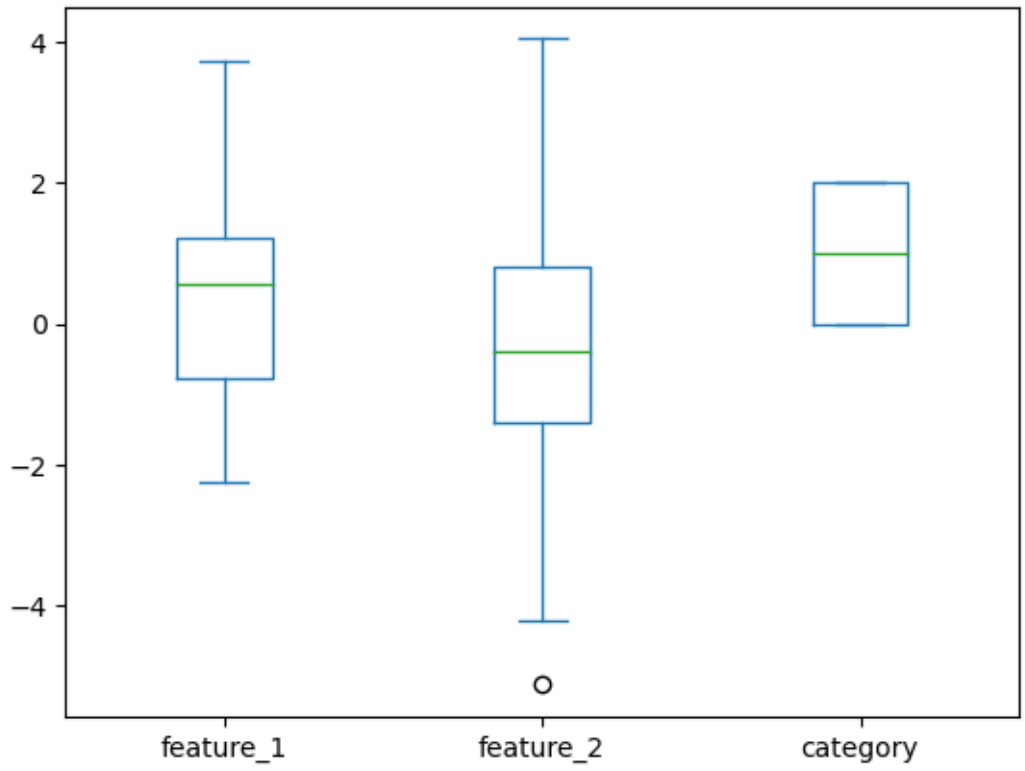
	feature_1	feature_2	category
count	1000.000000	1000.000000	1000.000000
mean	0.332238	-0.299943	0.998000
std	1.142013	1.516528	0.818942
min	-2.234235	-5.113364	0.000000
25%	-0.779555	-1.399917	0.000000
50%	0.568081	-0.391351	1.000000
75%	1.221679	0.795395	2.000000
max	3.724172	4.034715	2.000000

[14]:
```
df.category.value_counts()
```

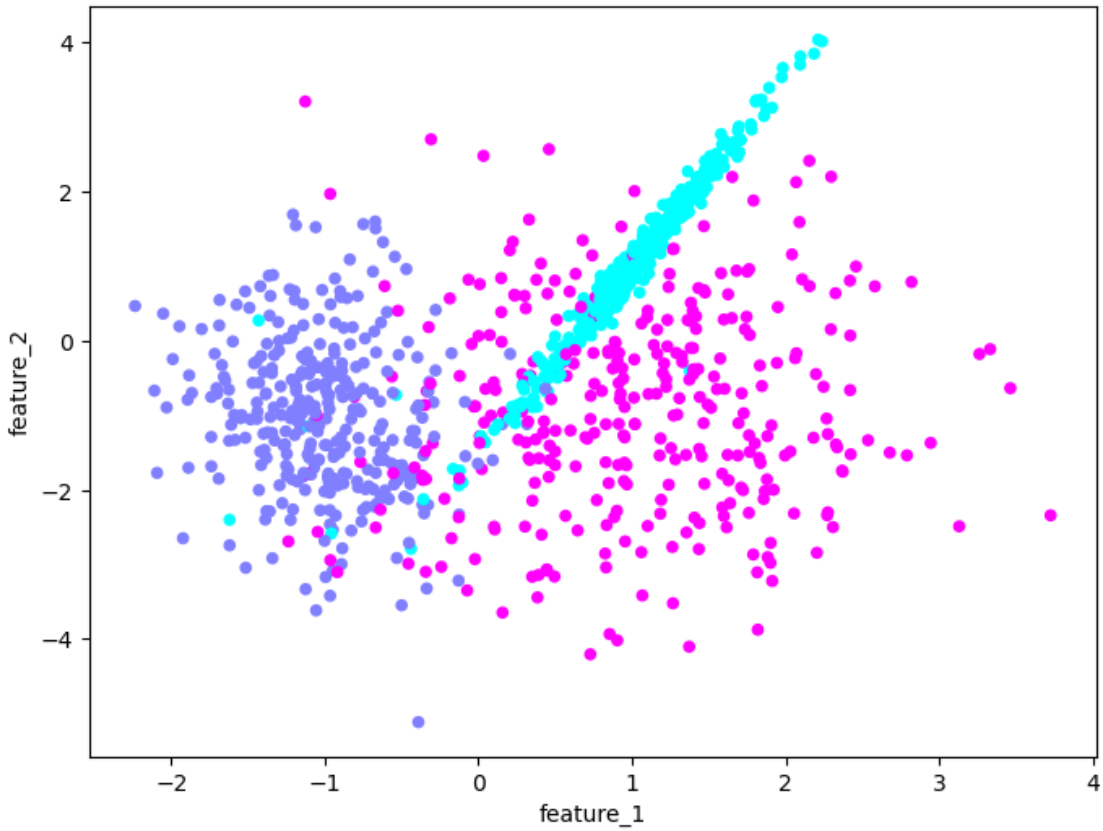
[14]:
```
0    336
2    334
1    330
Name: category, dtype: int64
```

[18]:
```
import matplotlib.pyplot as plt
from matplotlib import pylab

df.plot.box()
plt.show()
```



[20]:
```
df.plot.scatter("feature_1","feature_2",c="category",cmap=pylab.cm.
  ↪cool,figsize=(8,6),colorbar=False)
plt.show()
```



2 Scaling

[24]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df[["feature_1","feature_2"]])
```

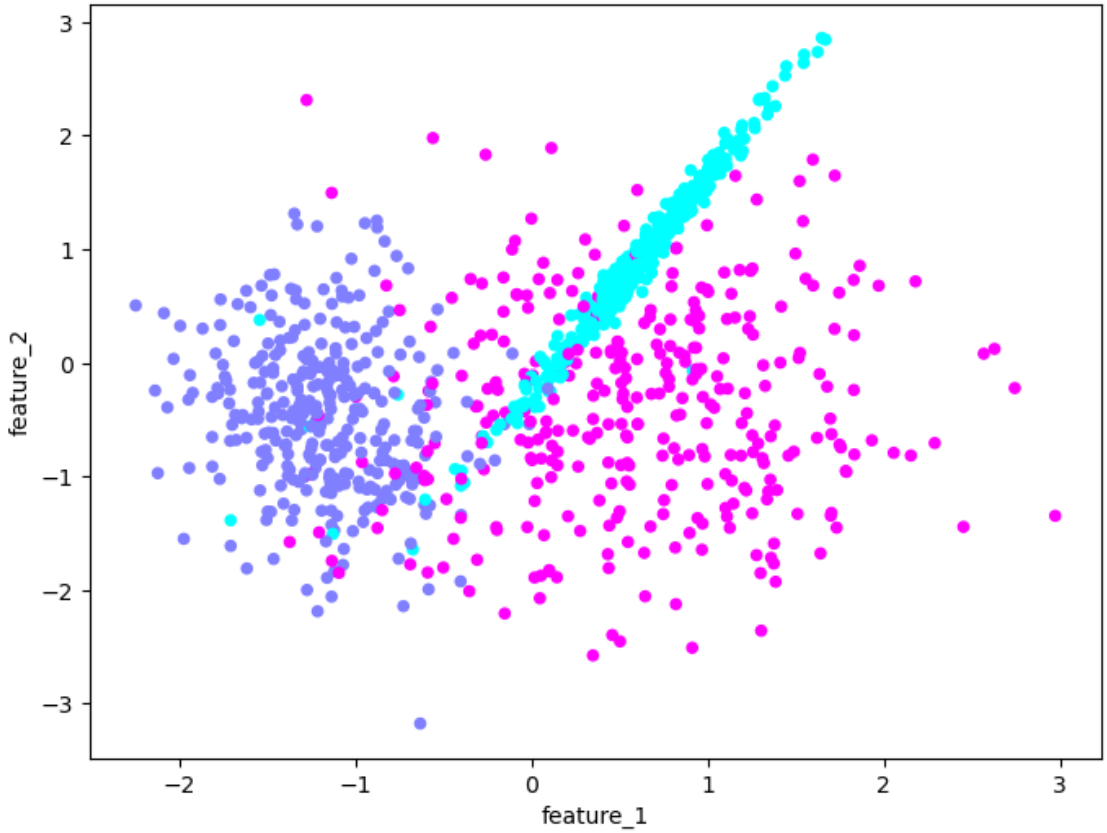
[24]:
```
StandardScaler()
```

[26]:
```
standardized = scaler.transform(df[["feature_1","feature_2"]])
standardized
```

[26]:
```
array([[ 0.86225717,  1.45582779],
       [ 0.14626453, -0.90269403],
       [ 0.53065139,  0.74829594],
       …,
       [-1.11216242, -1.02386509],
       [ 1.82951322,  0.24260611],
       [ 0.12922246,  0.15564846]])
```

[28]:
```
df_scaled = df.copy()
df_scaled.loc[:,["feature_1","feature_2"]] = standardized
```

[29]:
```
df_scaled.plot.scatter("feature_1","feature_2",c="category",cmap=pylab.cm.
↪cool,figsize=(8,6),colorbar=False)
plt.show()
```



3 K nearest neighbors

[5]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[6]:
```
knn = KNeighborsClassifier(n_neighbors=10)
```

[30]:
```
knn.fit(X=df[["feature_1","feature_2"]],y=df.category)
```

[30]:
```
KNeighborsClassifier(n_neighbors=10)
```

[31]:
```
knn.predict([(0,0)])
```

```
C:\Users\malir\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but KNeighborsClassifier was fitted with
feature names
  warnings.warn(
C:\Users\malir\anaconda3\lib\site-
packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other
reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode`
typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will
change: the default value of `keepdims` will become False, the `axis` over which
the statistic is taken will be eliminated, and the value None will no longer be
accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

[31]:
```
array([2])
```

[35]:
```
df["knn_prediction"] = knn.predict(df[["feature_1","feature_2"]])
df.head()
```

```
C:\Users\malir\anaconda3\lib\site-
packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other
reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode`
typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will
change: the default value of `keepdims` will become False, the `axis` over which
the statistic is taken will be eliminated, and the value None will no longer be
accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

[35]:

	feature_1	feature_2	category	knn_prediction
0	1.316455	1.906756	0	0
1	0.499191	-1.668219	2	2
2	0.937946	0.834301	0	0
3	0.930180	0.648558	0	0
4	0.309252	-1.365109	2	2

[39]:
```
from sklearn import metrics
knn_metrics = metrics.precision_recall_fscore_support(df.category,df.
 ↪knn_prediction)
knn_metrics
```

[39]:
```
(array([0.88108108, 0.898017  , 0.94945848]),
 array([0.9702381 , 0.96060606, 0.78742515]),
 array([0.92351275, 0.92825769, 0.8608838 ]),
 array([336, 330, 334], dtype=int64))
```

[40]:
```
metrics.precision_score(df.category, df.knn_prediction, average="weighted")
```

[40]:
```
0.9095079858824062
```

[41]:
```
metrics.recall_score(df.category, df.knn_prediction, average="weighted")
```

[41]: 0.906

[42]:
```
knn_f1 = metrics.f1_score(df.category, df.knn_prediction, average="weighted")
knn_f1
```

[42]: 0.9041605081053793