Untitled

March 27, 2023

[15]:
```
import sklearn
print(sklearn.__version__)
```

```
1.0.2
```

[30]:
```
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=5000, n_features=20, n_informative=10,␣
 ↪n_redundant=10, random_state=7)

print(X.shape, y.shape)
```

```
(5000, 20) (5000,)
```

LOGISTIC REGRESSION

[31]:
```
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=5000, n_features=20, n_informative=10,␣
 ↪n_redundant=10, random_state=7)

model = LogisticRegression()

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Accuracy: 0.889 (0.012)
```

PCA

[32]:
```
from numpy import mean
from numpy import std
```

```
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=5000, n_features=20, n_informative=10,␣
 ↪n_redundant=10, random_state=7)

steps = [('pca', PCA(n_components=10)), ('m', LogisticRegression())]
model = Pipeline(steps=steps)

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Accuracy: 0.889 (0.012)
```

Singular Value Decomposition

[33]:
```
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.decomposition import TruncatedSVD
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=5000, n_features=20, n_informative=10,␣
 ↪n_redundant=10, random_state=7)

steps = [('svd', TruncatedSVD(n_components=10)), ('m', LogisticRegression())]
model = Pipeline(steps=steps)

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Accuracy: 0.889 (0.012)
```

lİNEAR DISCRIMINAT

[34]:
```
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
```

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=5000, n_features=20, n_informative=10,␣
 ↪n_redundant=10, random_state=7)

steps = [('lda', LinearDiscriminantAnalysis(n_components=1)), ('m',␣
 ↪LogisticRegression())]
model = Pipeline(steps=steps)

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Accuracy: 0.886 (0.012)
```

ISOMAP EMBEDDED

[35]:
```
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.manifold import Isomap
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=5000, n_features=20, n_informative=10,␣
 ↪n_redundant=10, random_state=7)

steps = [('iso', Isomap(n_components=10)), ('m', LogisticRegression())]
model = Pipeline(steps=steps)

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Accuracy: 0.946 (0.008)
```

Locally Linear Embedding

[36]:
```
from numpy import mean
from numpy import std
```

```
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.manifold import LocallyLinearEmbedding
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=5000, n_features=20, n_informative=10,␣
 ↪n_redundant=10, random_state=7)

steps = [('lle', LocallyLinearEmbedding(n_components=10)), ('m',␣
 ↪LogisticRegression())]
model = Pipeline(steps=steps)

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Accuracy: 0.928 (0.013)
```

Modified Locally Linear Embedding

[37]:
```
from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.pipeline import Pipeline
from sklearn.manifold import LocallyLinearEmbedding
from sklearn.linear_model import LogisticRegression

X, y = make_classification(n_samples=5000, n_features=20, n_informative=10,␣
 ↪n_redundant=10, random_state=7)

steps = [('lle', LocallyLinearEmbedding(n_components=5, method='modified',␣
 ↪n_neighbors=10)), ('m', LogisticRegression())]
model = Pipeline(steps=steps)

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv, n_jobs=-1)

print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

```
Accuracy: 0.873 (0.013)
```

[ ]: