27 Εισαγωγη στη Μηχανικη Μαθηση - SciKit-Learn

27.0.1 Προηγούμενες Ασκήσεις

1. Γράψτε ένα πρόγραμμα το οποίο να προσθέτει, αφαιρεί, πολλαπλασιάζει και διαιρεί 2 pandas series (γραμμές) Χρησιμοποιήστε τις σειρές: [2, 4, 6, 8, 10], [1, 3, 5, 7, 9]

Λύση

```
import pandas as pd
ds1 = pd.Series([2, 4, 6, 8, 10])
ds2 = pd.Series([1, 3, 5, 7, 9])
ds = ds1 + ds2
print("Πρόσθεση 2 σειρών:")
print(ds)
print("Αφαίρεση 2 σειρών :")
ds = ds1 - ds2
print(ds)
print("Πολλαπλασιασμός 2 σειρών:")
ds = ds1 * ds2
print(ds)
print("Διαίρεση της σειράς 1 από τη σειρά 2:")
ds = ds1 / ds2
print(ds)
```

και η εκτύπωση είναι (η πρώτη στήλη είναι index):

```
Πρόσθεση 2 σειρών:
0     3
1     7
2    11
3    15
4    19
dtype: int64
Αφαίρεση 2 σειρών :
0    1
1    1
2    1
3    1
4    1
dtype: int64
Πολλαπλασιασμός 2 σειρών:
0     2
1    12
2    30
3    56
4    90
dtype: int64
Διαίρεση της σειράς 1 από τη σειρά 2:
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
```

2. Γράψτε ένα πρόγραμμα το οποίο να συγκρίνει τις παρακάτω σειρές: [2, 4, 6, 8, 10], [1, 3, 5, 7, 10]. (ποιοι αριθμοί είναι ίσοι, μεγαλύτεροι ή μικρότεροι; Χρησιμοποιήστε τους τελεστές «==», «>» και «<»).

```python
import pandas as pd
ds1 = pd.Series([2, 4, 6, 8, 10])
ds2 = pd.Series([1, 3, 5, 7, 10])
print("Σειρά 1:")
print(ds1)
print("Σειρά 2:")
print(ds2)
print("Σύγκριση των στοιχείων των σειρών:")
print("Ίσα στοιχεία:")
print(ds1 == ds2)
```

```
print("Μεγαλύτερα από:")
print(ds1 > ds2)
print("Μικρότερα από:")
print(ds1 < ds2)
```

και η εκτύπωση είναι:

```
Σειρά 1:
0     2
1     4
2     6
3     8
4    10
dtype: int64
Σειρά 2:
0     1
1     3
2     5
3     7
4    10
dtype: int64
Σύγκριση των στοιχείων των σειρών:
Ίσα στοιχεία:
0    False
1    False
2    False
3    False
4     True
dtype: bool
Μεγαλύτερα από:
0     True
1     True
2     True
3     True
4    False
dtype: bool
Μικρότερα από:
0    False
1    False
2    False
3    False
4    False
dtype: bool
```

3. Γράψτε ένα πρόγραμμα το οποίο να τυπώνει τις πρώτες 3 σειρές του παρακάτω DataFrame:

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Μπορείτε να χρησιμοποιήσετε την iloc και αυτά που μάθαμε για τα slices ώστε να πάρουμε μόνο τα 3 στοιχεία.

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine',
'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin',
'Jonas'],
        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5,
np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes',
'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("First three rows of the data frame:")
print(df.iloc[:3])
```

και παίρνουμε την εκτύπωση:

```
First three rows of the data frame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
```

27.1.0 Τι είναι η Μηχανική Μάθηση

Πριν μιλήσουμε για το Scikit Learn, πρέπει να κατανοήσουμε την έννοια της μηχανικής μάθησης. Με τη μηχανική μάθηση, δεν χρειάζεται να συγκεντρώσουμε τις πληροφορίες μας με μη αυτόματο τρόπο. Χρειαζόμαστε απλώς έναν αλγόριθμο. Το Scikit Learn είναι μια ειδική βιβλιοθήκη της Python με την οποία μπορούμε να εφαρμόσουμε μηχανική μάθηση. Μια άλλη επίσης γνωστή και αρκετά διαδεδομένη βιβλιοθήκη είναι το Tensorflow. Θα μπορούσαμε να πούμε ότι είναι ακόμα πιο δυνατό, αλλά δεν μπορεί να εξυπηρετήσει ευρέως. Το Scikit Learn είναι μια δωρεάν βιβλιοθήκη μηχανικής μάθησης που περιέχει απλά και αποτελεσματικά εργαλεία για σκοπούς ανάλυσης και εξόρυξης δεδομένων.

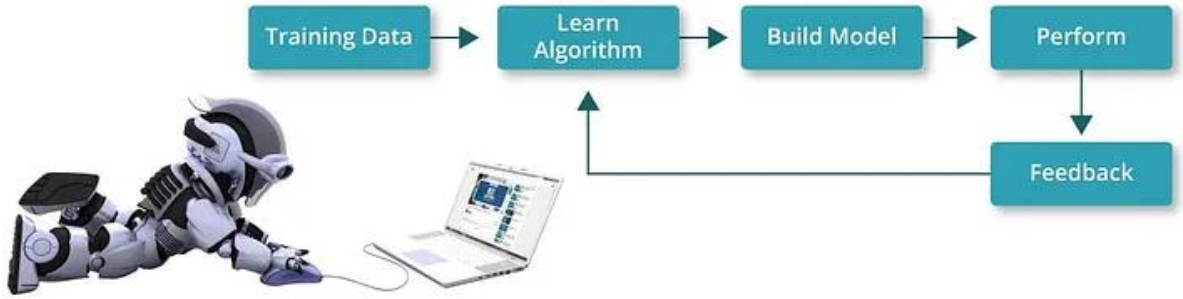
Ας γνωρίσουμε κάποιες απαραίτητες έννοιες

Τι είναι η μηχανική μάθηση;

Η μηχανική μάθηση είναι ένας τύπος τεχνητής νοημοσύνης που επιτρέπει στις εφαρμογές λογισμικού να μαθαίνουν από τα δεδομένα και να γίνονται πιο ακριβείς σε πρόβλεψη αποτελεσμάτων χωρίς ανθρώπινη παρέμβαση.

Πώς γίνεται όμως αυτό; Για να εξυπηρετηθεί αυτός ο σκοπός, το πρόγραμμα πρέπει να εκπαιδευτεί σε ορισμένα δεδομένα και με βάση αυτά, θα εντοπίσει ένα μοτίβο για τη δημιουργία ενός μοντέλου.

Αυτή η διαδικασία απόκτησης γνώσης από τα δεδομένα και παροχής ισχυρών πληροφοριών καλείται μηχανική μάθηση. Ας δούμε την παρακάτω εικόνα για να κατανοήσουμε καλύτερα τη λειτουργία του:



Χρησιμοποιώντας τα δεδομένα, το σύστημα (πρόγραμμα, υπολογιστής κ.λ.π.) μαθαίνει έναν αλγόριθμο και στη συνέχεια τον χρησιμοποιεί για να δημιουργήσει ένα μοντέλο πρόβλεψης. Αργότερα, προσαρμόζουμε το μοντέλο ή βελτιώνουμε την ακρίβεια του μοντέλου χρησιμοποιώντας τα δεδομένα ανατροφοδότησης (από το feedback). Χρησιμοποιώντας αυτά τα δεδομένα ανατροφοδότησης, συντονίζουμε το μοντέλο και επαναλαμβάνουμε την κίνηση, ξαναπροβλέποντας και χρησιμοποιώντας το νέο σύνολο δεδομένων.

Στη συνέχεια, είναι απαραίτητο να γνωρίσουμε επίσης πως υπάρχουν τρεις τύποι μηχανικής μάθησης:

27.1.1 Εποπτευόμενη μάθηση (Supervised Learning)

Αυτή είναι μια διαδικασία ενός αλγορίθμου που μαθαίνει από το σύνολο των δεδομένων εκπαίδευσης.

Στην εποπτευόμενη μάθηση δημιουργούμε μια συνάρτηση αντιστοίχισης μεταξύ της μεταβλητής εισόδου (X) και μιας μεταβλητής εξόδου (Y) και χρησιμοποιούμε έναν αλγόριθμο για να δημιουργήσουμε μια συνάρτηση μεταξύ τους.

Αυτή η μέθοδος είναι επίσης γνωστή ως προγνωστική μοντελοποίηση (predictive modeling) που αναφέρεται σε μια διαδικασία κατά την οποία γίνονται προβλέψεις χρησιμοποιώντας τα δεδομένα. Κάποιοι από τους αλγόριθμους είναι: Γραμμική παλινδρόμηση (linear regression), Λογιστική παλινδρόμηση (logistic regression), Δέντρο απόφασης (decision tree), Τυχαίο δάσος (random forest) και ταξινομητής Naive Bayes (*Naive Bayes classifier)*.

27.1.2 Εκμάθηση χωρίς επίβλεψη

Αυτή είναι μια διαδικασία όπου ένα μοντέλο εκπαιδεύεται χρησιμοποιώντας πληροφορίες που δεν φέρουν ετικέτα. Αυτή η διαδικασία μπορεί να χρησιμοποιηθεί για τη ομαδοποίηση των δεδομένων εισόδου σε κλάσεις με βάση τις στατιστικές τους ιδιότητες. Η μάθηση χωρίς επίβλεψη ονομάζεται επίσης ανάλυση ομαδοποίησης που σημαίνει την ομαδοποίηση αντικειμένων με βάση τις πληροφορίες που βρίσκονται στα δεδομένα που περιγράφουν τα

αντικείμενα ή τη σχέση τους. Ο στόχος είναι τα αντικείμενα σε μια ομάδα να είναι παρόμοια μεταξύ τους αλλά διαφορετικά από τα αντικείμενα μιας άλλης ομάδας. Μερικοί από τους αλγόριθμους περιλαμβάνουν ομαδοποίηση K-means (K-Means clustering), Ιεραρχική ομαδοποίηση (Hierarchical clustering) κ.λπ.

27.1.3 Ενισχυτική μάθηση

Η ενισχυτική μάθηση είναι η μάθηση μέσω της αλληλεπίδρασης με το χώρο ή ένα περιβάλλον. Ένας πράκτορας RL μαθαίνει από τις συνέπειες των πράξεών του, αντί να διδάσκεται ρητά. Επιλέγει τις ενέργειές του με βάση τις προηγούμενες εμπειρίες του (εκμετάλλευση) αλλά και με νέες επιλογές (εξερεύνηση).

27.1.4 Εισαγωγή στο SciKit-Learn

Η Scikit-learn, αναπτύχθηκε για πρώτη φορά σαν ένα έργο του Google Summer of Code το 2007, είναι η πλέον ευρέως γνωστή ως η πιο δημοφιλής βιβλιοθήκη της Python για μηχανική μάθηση.

Υπάρχουν διάφοροι λόγοι για τους οποίους αυτή η βιβλιοθήκη θεωρείται ως μία από τις καλύτερες επιλογές για έργα μηχανικής μάθησης, ειδικά σε συστήματα παραγωγής. Κάποιοι από τους λόγους είναι;

- Έχει υψηλό επίπεδο υποστήριξης και αυστηρή εποπτεία στην ανάπτυξή της που σημαίνει ότι είναι και γίνεται όλο και περισσότερο ένα ισχυρό εργαλείο.
- Υπάρχει ένα σαφές, συνεπές στυλ κώδικα που διασφαλίζει ότι ο κώδικας μηχανικής μάθησης είναι εύκολο να κατανοηθεί και να αναπαραχθεί, και επίσης μειώνει σημαντικά το εμπόδιο στην είσοδο για κωδικοποίηση μοντέλων μηχανικής εκμάθησης.
- Υποστηρίζεται ευρέως από εργαλεία τρίτων, επομένως είναι δυνατό να εμπλουτιστεί η λειτουργικότητα για να ταιριάζει σε μια σειρά περιπτώσεων χρήσης.
- Είναι η πιο κατάλληλη για την εισαγωγή στη μηχανική μάθηση. Η απλότητά της σημαίνει ότι είναι αρκετά εύκολη και μειώνει σημαντικά τον πήχη εισόδου στα μοντέλα μηχανικής μάθησης.

Η εγκατάστασή της γίνεται όπως και με όλες τις βιβλιοθήκες, στο command line:

```
pip install scikit-learn
```

Τα παρακάτω παραδείγματα κώδικα εκτελούνται στο κλασικό σύνολο δεδομένων κρασιού (wine data set) που μπορεί να εισαχθεί απευθείας από το Scikit-learn API. Το data set είναι ενσωματωμένο στην SciKit-Learn και κάποια κύρια χαρακτηριστικά του είναι:

Classes	3
Samples per class	[59,71,48]
Samples total	178
Dimensionality	13
Features	real, positive

Μπορούμε να βρούμε την πλήρη τεκμηρίωσή του, στην τεκμηρίωση του SciKit-Learn εδώ .

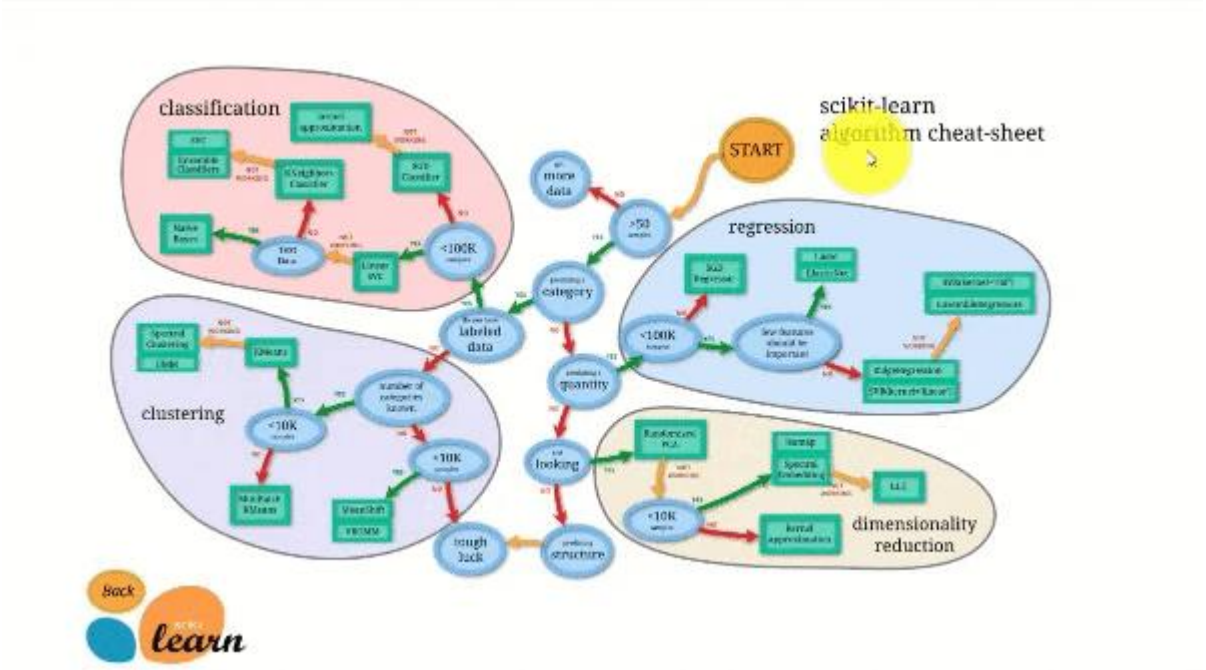
27.1.5 Estimators

Η βιβλιοθήκη Scitkit-learn παρέχει μια πολύ μεγάλη ποικιλία προκατασκευασμένων αλγορίθμων για την εκτέλεση τόσο εποπτευόμενης όσο και μη εποπτευόμενης μηχανικής εκμάθησης. Αυτοί οι αλγόριθμοι, γενικά αναφέρονται ως εκτιμητές (estimators).

Ο estimator που θα επιλέξουμε για το έργο μας θα εξαρτηθεί από το σύνολο δεδομένων που έχουμε και το πρόβλημα που προσπαθούμε να επιλύσουμε.

Η τεκμηρίωση του Scikit-learn δημοσιεύει ένα ευρέως διαδεδομένο διάγραμμα, που φαίνεται παρακάτω, για να μας βοηθήσει να προσδιορίσουμε ποιος αλγόριθμος είναι κατάλληλος για την εργασία μας.

Δεν έχουμε παρά να ακολουθήσουμε με το ποντίκι μας το διάγραμμα, δίνοντας απαντήσεις σχετικά με το πρόβλημά μας και οδηγούμαστε πολύ εύκολα στον κατάλληλο estimator με όλες τις πληροφορίες και λεπτομέρειες που χρειαζόμαστε για τη χρήση του.



Αυτό που κάνει το Scikit-learn τόσο απλό στη χρήση είναι ότι ανεξάρτητα από το μοντέλο ή τον αλγόριθμο που χρησιμοποιούμε, η δομή του κώδικα για την εκπαίδευση και την πρόβλεψη μοντέλων είναι η ίδια.

Για να το διευκρινίσουμε αυτό, ας δούμε ένα παράδειγμα.

Ας υποθέσουμε ότι εργαζόμαστε σε ένα πρόβλημα παλινδρόμησης και θέλουμε να εκπαιδεύσουμε έναν αλγόριθμο linear regression και να χρησιμοποιήσουμε το μοντέλο που προκύπτει για να κάνουμε προβλέψεις.

Χρησιμοποιώντας το «έξυπνο διάγραμμα» που μόλις περιγράψαμε, ιο πρώτο βήμα, με το Scikit-learn, είναι να καλέσουμε logistic regression estimator και να τον αποθηκεύσουμε ως αντικείμενο.

Το παρακάτω παράδειγμα καλεί τον αλγόριθμο και τον αποθηκεύει ως αντικείμενο που ονομάζουμε **lr** .

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
```

Το επόμενο βήμα είναι η προσαρμογή του μοντέλου σε ορισμένα δεδομένα εκπαίδευσης. Αυτό εκτελείται χρησιμοποιώντας τη μέθοδο **fit()** .

Καλούμε το αντικείμενο lr.fit() πάνω στα χαρακτηριστικά και τα δεδομένα του στόχου και αποθηκεύουμε το μοντέλο που προκύπτει ως αντικείμενο που ονομάζουμε **model** .

Στο παρακάτω παράδειγμα, χρησιμοποιούμε επίσης τη μέθοδο train_test_split() για να χωρίσουμε το σύνολο δεδομένων σε δεδομένα δοκιμής και εκπαίδευσης.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=0)

model = lr.fit(X_train, y_train)
```

Στη συνέχεια, χρησιμοποιούμε το μοντέλο και τη μέθοδο **predict()** για να προβλέψουμε πάνω σε δεδομένα που δεν έχουν χρησιμοποιηθεί.

```
predictions = model.predict(X_test)
```

Αν χρησιμοποιούσαμε τώρα το Scitkit-learn για να εκτελέσουμε μια διαφορετική εργασία, για παράδειγμα, θέλαμε να εκπαιδεύσουμε έναν random forest classifier, ο κώδικας θα έμοιαζε πολύ και θα είχε τον ίδιο αριθμό βημάτων.

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
rf_model = rf.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

Αυτή η συνεπής δομή κώδικα καθιστά την ανάπτυξη μοντέλων μηχανικής εκμάθησης εξαιρετικά απλή και παράγει επίσης κώδικα που είναι εξαιρετικά αναγνώσιμος και παράγεται εύκολα.

27.1.6 Προεπεξεργασία

Στα περισσότερα έργα μηχανικής εκμάθησης πραγματικού κόσμου, τα δεδομένα που χρησιμοποιούμε δεν θα είναι απαραίτητα έτοιμα για την εκπαίδευση ενός μοντέλου. Άλλωστε το είδαμε κι αυτό με τη χρήση της βιβλιοθήκης pandas, όπου όλα τα αρχεία που χρησιμοποιήσαμε, χρειάζονταν πρώτα ένα στάδιο καθαρισμού των δεδομένων. Έτσι, είναι και πάλι πολύ πιθανό ότι θα χρειαστεί πρώτα να εκτελέσουμε ορισμένα βήματα προεπεξεργασίας και μετασχηματισμού των δεδομένων, όπως χειρισμό τιμών που λείπουν, μετατροπή δεδομένων κατηγοριώνσε αριθμητικά ή εφαρμογή κλιμάκωσης χαρακτηριστικών.

Το Scikit-learn έχει ενσωματωμένες μεθόδους για την εκτέλεση αυτών των βημάτων προεπεξεργασίας.

Για παράδειγμα, η **SimpleImputer()** συμπληρώνει τις τιμές που λείπουν χρησιμοποιώντας μια μέθοδο της επιλογής μας.

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')
X_train_clean = imputer.fit(X_train)
```

Η τεκμηρίωση του Scikit-learn παραθέτει τις πλήρεις επιλογές για την προεπεξεργασία των δεδομένων εδώ .

27.1.7 Model evaluation - Αξιολόγηση Μοντέλου

Μόλις ένα μοντέλο έχει εκπαιδευτεί, πρέπει να μετρήσουμε πόσο καλό είναι στην πρόβλεψη για νέα δεδομένα.

Αυτό το βήμα είναι γνωστό ως αξιολόγηση μοντέλου και η μέτρηση που θα επιλέξουμε θα καθοριστεί από την εργασία που προσπαθούμε να επιλύσουμε.

Για παράδειγμα, συνήθως σε ένα πρόβλημα regression, μπορεί να επιλέξουμε την RMSE (Root Mean Squared Error) ενώ για classification μπορεί να επιλέξουμε την F1-score.

Όλοι οι εκτιμητές περιλαμβάνουν μια μέθοδο **score()** που επιστρέφει μια προεπιλεγμένη μέτρηση που είναι πιο σχετική με την εργασία μηχανικής εκμάθησης που εκτελούν.

Το Scikit-learn έχει επιπλέον ένα σύνολο **μετρικών συναρτήσεων** που παρέχουν μια πιο λεπτομερή αξιολόγηση για ένα μοντέλο. Για παράδειγμα, για εργασίες ταξινόμησης, η βιβλιοθήκη έχει μια αναφορά ταξινόμησης που παρέχει ακρίβεια, ανάκληση, βαθμολογία F1 και συνολική ακρίβεια.

```
from sklearn.metrics import classification_report

print(classification_report(rf_predictions, y_test))
```

27.1.8 Model optimisation - Βελτιστοποίηση μοντέλου

Όλοι οι estimators στη βιβλιοθήκη Scikit-learn περιέχουν μια σειρά παραμέτρων, για τις οποίες υπάρχουν πολλές επιλογές. Οι τιμές που επιλέγουμε για έναν συγκεκριμένο αλγόριθμο θα επηρεάσουν την απόδοση του τελικού μοντέλου. Για παράδειγμα, με τον RandomForestClassifier μπορούμε να ορίσουμε το max_depth του δέντρου σε δυνητικά οποιαδήποτε τιμή, ανάλογα με τα δεδομένα και την εργασία μας. Διαφορετικές τιμές για αυτήν την παράμετρο θα παράγουν διαφορετικά αποτελέσματα.

Αυτή η διαδικασία δοκιμής διαφορετικών συνδυασμών παραμέτρων για την εύρεση του βέλτιστου συνδυασμού είναι γνωστή ως **βελτιστοποίηση υπερπαραμέτρων (hyperparameter optimisation)** .

Το Scikit-learn παρέχει δύο εργαλεία για την αυτόματη εκτέλεση αυτής της εργασίας, **το GridSearchCV** που εφαρμόζει μια τεχνική γνωστή ως εξαντλητική αναζήτηση πλέγματος (exhaustive grid search) και **το**

RandomizedSearchCV που εκτελεί τυχαιοποιημένη βελτιστοποίηση παραμέτρων (randomized parameter optimisation) .

Το παρακάτω παράδειγμα χρησιμοποιεί το GridSearchCV για να βρει τις βέλτιστες παραμέτρους για τον RandomForestClassifier.

Η έξοδος φαίνεται κάτω από τον κώδικα.

```
{'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'n_estimators': 200}
0.9774928774928775
```

27.1.9 Μελέτη του dataset

Θα χρησιμοποιήσουμε το ενσωματωμένο dataset του SciKit-Learn, το Iris dataset.

Λίγα λόγια για το dataset (από τη Wikipedia)

Το Iris dataset ή το Fisher's Iris dataset είναι ένα σύνολο δεδομένων πολλαπλών μεταβλητών που χρησιμοποιήθηκε και έγινε διάσημο από τον Βρετανό στατιστικολόγο και βιολόγο Ronald Fisher στην εργασία του το 1936 “The use of multiple memeters in taxonomic Problems” ως παράδειγμα ανάλυσης γραμμικής διάκρισης.

Μερικές φορές ονομάζεται Anderson's Iris dataset επειδή ο Edgar Anderson συνέλεξε τα δεδομένα για να ποσοτικοποιήσει τη μορφολογική παραλλαγή των λουλουδιών Iris τριών συγγενών ειδών.

Αυτό το σύνολο δεδομένων αποτελείται από 3 διαφορετικούς τύπους λουλουδιών ίριδας (Setosa, Versicolour και Virginica) με μήκος πέταλου και σέπαλου, αποθηκευμένα σε ένα numpy.ndarray 150x4

Οι γραμμές είναι τα δείγματα και οι στήλες είναι: Μήκος σεπάλου, Πλάτος σεπάλου, Μήκος πετάλου και Πλάτος πετάλου.

Ένα δείγμα των επικεφαλίδων και των πρώτων γραμμών του dataset είναι:

Dataset order	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.1	3.5	1.4	0.2	*I. setosa*
2	4.9	3.0	1.4	0.2	*I. setosa*
3	4.7	3.2	1.3	0.2	*I. setosa*
4	4.6	3.1	1.5	0.2	*I. setosa*
5	5.0	3.6	1.4	0.3	*I. setosa*

Αφού έχουμε μάθει κι εξασκήσει και τη βιβλιοθήκη Pandas καθώς και τη βιβλιοθήκη οπτικοποίησης δεδομένων pyplot, ας δώσουμε λίγες γραμμές κώδικα για να πάρουμε εκτύπωση ενός διαγράμματος Scatter Plot που ταξινομεί τα λουλούδια σύμφωνα με τα χαρακτηριστικά τους:

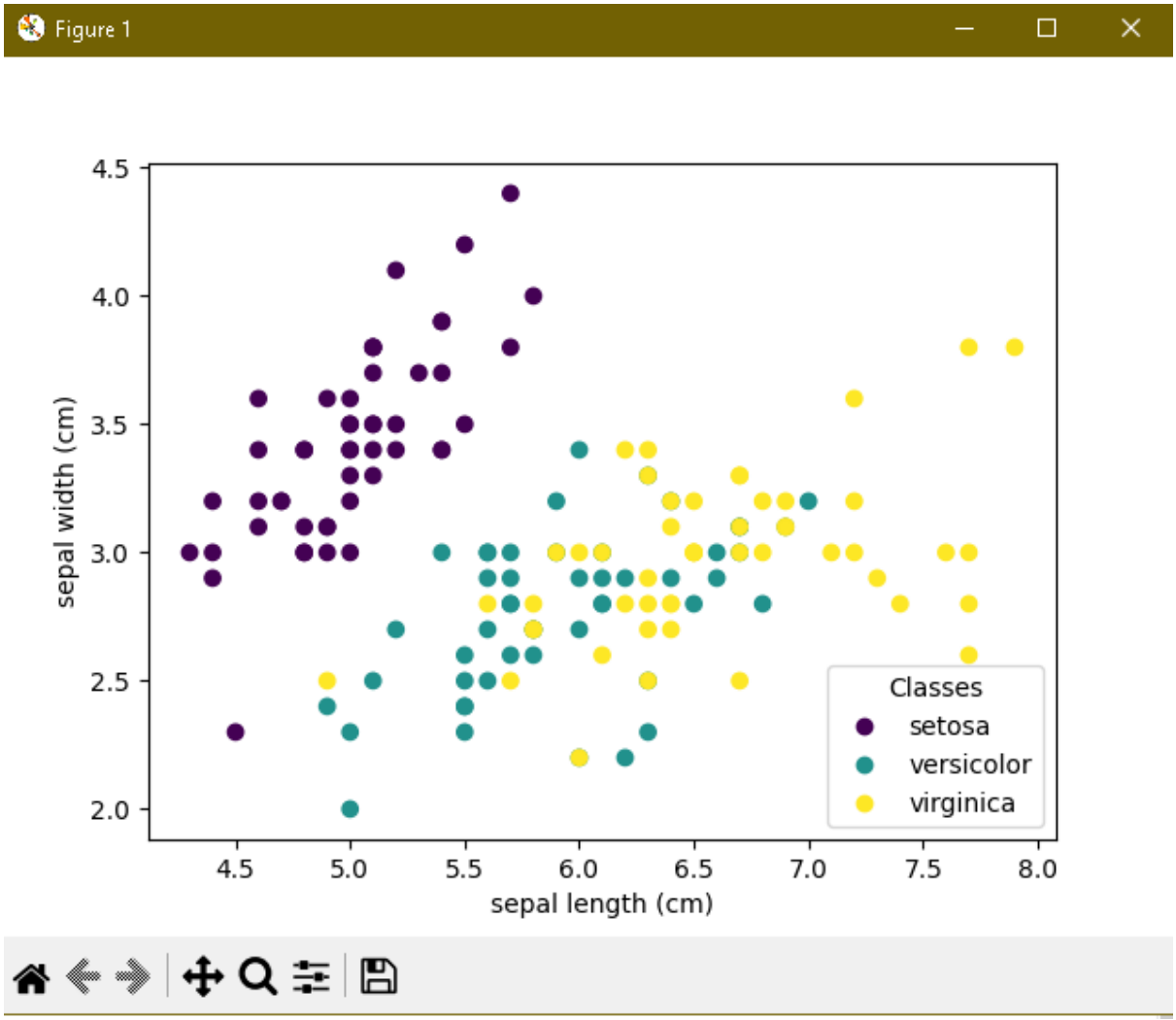
```
import matplotlib.pyplot as plt
from sklearn import datasets

iris = datasets.load_iris()

_, ax = plt.subplots()
scatter = ax.scatter(iris.data[:, 0], iris.data[:, 1],
c=iris.target)
ax.set(xlabel=iris.feature_names[0],
ylabel=iris.feature_names[1])
_ = ax.legend(
    *scatter.legend_elements(), loc="lower right",
title="Classes"
)
```

```
plt.show()
```

Και παίρνουμε την εκτύπωση:



27.1.10 Περαιτέρω εξήγηση του κώδικα (για το underscore)

Η υπογράμμιση "_" χρησιμοποιείται ως ένα συμβατικό όνομα μεταβλητής κράτησης θέσης για να υποδείξει ότι η τιμή που της έχει εκχωρηθεί δεν προορίζεται για χρήση ή αναφορά στον κώδικα. Συχνά χρησιμοποιείται όταν θέλουμε να αγνοήσουμε μια τιμή ή ένα αποτέλεσμα που δεν χρειαζόμαστε.

Στον συγκεκριμένο κώδικα:

```
_, ax = plt.subplots()
```

Καλούμε το plt.subplots(), το οποίο επιστρέφει μια πλειάδα που περιέχει δύο τιμές: ένα αντικείμενο figure (το οποίο δεν χρησιμοποιείται σε αυτόν τον κώδικα) και ένα αντικείμενο axes (που εκχωρείται στη μεταβλητή ax).

Εφόσον δεν μας ενδιαφέρει το αντικείμενο figure, χρησιμοποιούμε το «_» ως σύμβολο κράτησης θέσης για να υποδείξουμε ότι το αγνοούμε σκόπιμα.

Ομοίως, στη γραμμή:

```
_ = ax.legend(scatter.legend_elements()[0],
iris.target_names, loc="lower right", title="Classes")
```

Αγνοούμε την επιστρεφόμενη τιμή της συνάρτησης ax.legend εκχωρώντας την σε ένα «_», υποδεικνύοντας ότι δεν χρειάζεται να χρησιμοποιήσουμε την επιστρεφόμενη τιμή στον κώδικα.

Η χρήση του «_» ως σύμβολο κράτησης θέσης, είναι μια κοινή πρακτική στην Python για να υποδείξουμε σε άλλους προγραμματιστές (ή στον εαυτό μας) ότι μια συγκεκριμένη τιμή δεν είναι σημαντική για το τρέχον περιβάλλον.

27.1.11 Παράδειγμα κατανόησης

Ας μελετήσουμε τον παρακάτω κώδικα:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

# Φόρτωση του Iris dataset
data = load_iris()
X = data.data
y = data.target

# Split σε training και testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Δημιουργία ενός RandomForestClassifier
rf = RandomForestClassifier()

# Καθορισμός του grid
param_grid = {
    'n_estimators': [200, 500],
    'max_features': [None, 'sqrt', 'log2', 1, 0.5],
    'max_depth': [4, 5, 6],
    'criterion': ['gini', 'entropy']
}

# Εκτέλεση αναζήτησης στο grid
print("Έναρξη Grid Search...")
CV = GridSearchCV(rf, param_grid, n_jobs=1)
CV.fit(X_train, y_train)
print("Τέλος Grid Search.")

# Print best parameters and score
print("Καλύτερες Παράμετροι:", CV.best_params_)
print("Καλύτερο Score:", CV.best_score_)
```

Πριν να δούμε την εκτύπωση, ας εξηγήσουμε τον κώδικα:

```
'n_estimators': [200, 500]:
```

Αυτό το χαρακτηριστικό ελέγχει τον αριθμό των δέντρων αποφάσεων στο δάσος των δέντρων αποφάσεων. Στον κώδικά μας, ελέγχουμε δύο τιμές, 200 και 500. Αυτό σημαίνει ότι το δάσος των δέντρων θα δημιουργηθεί με 200 ή 500 δέντρα.

```
'max_features': [None, 'sqrt', 'log2', 1, 0.5]:
```

Αυτό το χαρακτηριστικό καθορίζει τον μέγιστο αριθμό των χαρακτηριστικών (ή μεταβλητών) που κάθε δέντρο αποφάσεων επιτρέπεται να λάβει υπόψη κατά τη λήψη μιας απόφασης. Οι επιλογές περιλαμβάνουν:

‘None’: Σημαίνει ότι όλα τα χαρακτηριστικά λαμβάνονται υπόψη.

'sqrt': Σημαίνει την τετραγωνική ρίζα του συνολικού αριθμού των χαρακτηριστικών.

'log2': Σημαίνει τον δυαδικό λογάριθμο του συνολικού αριθμού των χαρακτηριστικών.

1: Σημαίνει ότι λαμβάνεται υπόψη μόνο ένα χαρακτηριστικό.

0.5: Σημαίνει ότι λαμβάνεται υπόψη το μισό των χαρακτηριστικών.

```
'max_depth': [4, 5, 6]:
```

Αυτό το χαρακτηριστικό ελέγχει το μέγιστο βάθος ή τον αριθμό των επιπέδων σε κάθε δέντρο αποφάσεων. Ένα μικρότερο αριθμητικό όπως το 4 σημαίνει

ένα ρηχό δέντρο, ενώ ένα μεγαλύτερο αριθμητικό όπως το 6 σημαίνει ένα βαθύτερο δέντρο. Αυτό περιορίζει το πόσο πολύπλοκο μπορεί να είναι κάθε δέντρο.

```
'criterion': ['gini', 'entropy']:
```

Αυτό το χαρακτηριστικό καθορίζει το κριτήριο που χρησιμοποιείται για τη μέτρηση της ποιότητας μιας διαίρεσης σε κάθε δέντρο αποφάσεων. Δοκιμάζουμε δύο επιλογές:

'gini': Χρησιμοποιεί τον δείκτη Gini ως κριτήριο.

'entropy': Χρησιμοποιεί τον δείκτη πληροφορίας (entropy) ως κριτήριο.

Όσον αφορά τα άλλα χαρακτηριστικά:

```
test_size=0.2
```

Αυτό το χαρακτηριστικό χρησιμοποιείται όταν διαιρούμε το σύνολο δεδομένων μας σε ένα σύνολο εκπαίδευσης και ένα σύνολο ελέγχου. Στην περίπτωσή μας, αυτό σημαίνει ότι το 20% των δεδομένων μας θα χρησιμοποιηθεί ως το σύνολο ελέγχου, ενώ το υπόλοιπο 80% θα χρησιμοποιηθεί ως το σύνολο εκπαίδευσης. Αυτό βοηθάει στην αξιολόγηση της απόδοσης του μοντέλου μας σε δεδομένα που δεν έχει δει κατά την εκπαίδευση.

```
random_state=42:
```

Αυτό το χαρακτηριστικό χρησιμοποιείται για να ορίσουμε ένα σπόρο (seed) για τον random number generator. Με τον ορισμό ενός σταθερού τυχαίου σπόρου (σε αυτήν την περίπτωση, το 42), εξασφαλίζουμε ότι η διαίρεση των δεδομένων είναι αναπαράστασιμη. Αυτό σημαίνει ότι εάν εκτελέσουμε τον κώδικα ξανά με τον ίδιο τυχαίο σπόρο, θα λάβουμε την ίδια διαίρεση των δεδομένων. Αυτό είναι χρήσιμο για τη διασφάλιση συνεπών αποτελεσμάτων όταν εργαζόμαστε με τυχαίες διαδικασίες όπως η διαίρεση των δεδομένων.

Τρέχοντας τον κώδικα έχουμε σαν έξοδο:

```
= RESTART: C:\Users\NK\AppData\Local\Programs\Python\Python311\scikit-learn_samp
le1.py
Έναρξη Grid Search...
Τέλος Grid Search.
Καλύτερες Παράμετροι: {'criterion': 'gini', 'max_depth': 5, 'max_features': 0.5,
 'n_estimators': 500}
Καλύτερο Score: 0.9583333333333334
```

27.2.0 Ερωτήσεις κατανόησης

1. Τι είναι η βιβλιοθήκη scikit-learn;
2. Τι είναι η μηχανική μάθηση;
3. Τι είναι ένας ταξινομητής (classifier);
4. Ποια είναι η διαδικασία της σταθεροποίησης των τυχαίων αριθμών στο scikit-learn;
5. Τι είναι ο διαχωρισμός των δεδομένων σε σύνολο εκπαίδευσης και σύνολο ελέγχου;
6. Τι είναι ο ταξινομητής RandomForestClassifier;
7. Τι είναι μια υπερ-παράμετρος (hyperparameter);

8. Τι είναι η διαδικασία της διασταυρούμενης επικύρωσης (cross-validation);
9. Τι είναι το πλέγμα υπερ-παραμέτρων (hyperparameter grid);
10. Τι κάνει η συνάρτηση GridSearchCV;
11. Πώς μπορούμε να αποκτήσουμε τις καλύτερες υπερ-παραμέτρους μετά από μια αναζήτηση GridSearchCV;
12. Τι κριτήρια μπορούμε να χρησιμοποιήσουμε για την δημιουργία ενός δέντρου αποφάσεων στο scikit-learn;
13. Πώς μπορείτε να περιορίσουμε το βάθος ενός δέντρου αποφάσεων στο scikit-learn;
14. Τι υπερ-παραμέτρους μπορούμε να ρυθμίσουμε για τον ταξινομητή RandomForestClassifier στο scikit-learn;