**SVC Model in Python**

Για να δημιουργήσετε ένα μοντέλο Support Vector Classifier (SVC) στην Python, μπορείτε να χρησιμοποιήσετε τη βιβλιοθήκη scikit-learn, η οποία παρέχει μια απλή και αποτελεσματική υλοποίηση:

from sklearn.svm import SVC  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
  
# Assuming you have your data in X (features) and y (labels) arrays  
  
# Split the data into training and testing sets  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Create an SVC model  
model = SVC()  
  
# Train the model  
model.fit(X\_train, y\_train)  
  
# Make predictions on the test set  
y\_pred = model.predict(X\_test)  
  
# Evaluate the model's accuracy  
accuracy = accuracy\_score(y\_test, y\_pred)  
print("Accuracy:", accuracy)

***Aνάλυση των βημάτων:***

1. Εισάγετε τις απαραίτητες βιβλιοθήκες: SVC από το sklearn.svm, train\_test\_split από το sklearn.model\_selection και accuracy\_score από το sklearn.metrics.
2. Χωρίστε τα δεδομένα σας σε σύνολα εκπαίδευσης(train) και δοκιμής(test) χρησιμοποιώντας τη συνάρτηση train\_test\_split. Αυτό σας επιτρέπει να αξιολογήσετε την απόδοση του μοντέλου σε αθέατα δεδομένα.
3. Δημιουργήστε ένα στιγμιότυπο του μοντέλου SVC καλώντας την SVC().
4. Εκπαιδεύστε το μοντέλο με τα δεδομένα εκπαίδευσης χρησιμοποιώντας τη μέθοδο fit, η οποία λαμβάνει τα χαρακτηριστικά (x\_train) και τις αντίστοιχες ετικέτες (y\_train).
5. Χρησιμοποιήστε το εκπαιδευμένο μοντέλο για να κάνετε προβλέψεις στο σύνολο δοκιμής(test set) καλώντας τη μέθοδο predict με το x\_test.
6. Αξιολογήστε την ακρίβεια του μοντέλου συγκρίνοντας τις προβλεπόμενες ετικέτες (y\_pred) με τις πραγματικές ετικέτες (y\_test) χρησιμοποιώντας τη συνάρτηση accuracy\_score.

Σημειώστε ότι αυτό είναι ένα βασικό παράδειγμα και μπορείτε να βελτιστοποιήσετε και να συντονίσετε περαιτέρω το μοντέλο προσαρμόζοντας διάφορες παραμέτρους, όπως τον τύπο του πυρήνα(kernel type), την παράμετρο κανονικοποίησης (regularization parameter) και άλλα.

**How to make a KNN model in Python**

***Η δημιουργία ενός μοντέλου K-Nearest Neighbors (KNN) με διαχωρισμένα δεδομένα περιλαμβάνει τα ακόλουθα βήματα:***

1. Εισαγωγή των απαραίτητων βιβλιοθηκών.

2. Φόρτωση και προεπεξεργασία του συνόλου δεδομένων.

3. Χωρίστε το σύνολο δεδομένων σε σύνολα εκπαίδευσης και δοκιμής.

4. Δημιουργήστε και εκπαιδεύστε το μοντέλο KNN χρησιμοποιώντας τα δεδομένα εκπαίδευσης.

5. Αξιολογήστε την απόδοση του μοντέλου χρησιμοποιώντας τα δεδομένα δοκιμής.

***Ας δούμε αναλυτικά κάθε βήμα σε Python:***

import numpy as np  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix  
  
# Step 2: Load and preprocess the dataset  
  
# Load your dataset here, replace 'X' with your feature matrix and 'y' with your target variable.  
  
# Make sure 'X' contains only numeric features and 'y' contains the corresponding labels.  
  
# Example: X, y = load\_your\_dataset()  
  
# Step 3: Split the dataset into training and testing sets  
  
# Use the train\_test\_split function from scikit-learn to split the dataset.  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Step 4: Create and train the KNN model using the training data  
  
# Instantiate the KNeighborsClassifier and specify the number of neighbors (k).  
  
knn\_model = KNeighborsClassifier(n\_neighbors=5) # You can change the value of 'k' as needed.  
  
knn\_model.fit(X\_train, y\_train)

# Step 5: Evaluate the model's performance using the testing data  
  
# Make predictions on the test data.  
  
y\_pred = knn\_model.predict(X\_test)  
  
# Calculate the accuracy of the model.  
  
accuracy = accuracy\_score(y\_test, y\_pred)  
  
print("Accuracy: {:.2f}%".format(accuracy \* 100))  
  
# Display the classification report and confusion matrix.  
  
print("Classification Report:")  
  
print(classification\_report(y\_test, y\_pred))  
  
print("Confusion Matrix:")  
  
print(confusion\_matrix(y\_test, y\_pred))

Σε αυτό το παράδειγμα, χρησιμοποιήσαμε τη συνάρτηση `train\_test\_split` της scikit-learn για να χωρίσουμε το σύνολο δεδομένων σε σύνολα εκπαίδευσης και δοκιμής. Η παράμετρος `test\_size` καθορίζει το ποσοστό των δεδομένων που θα χρησιμοποιηθεί για δοκιμή, ενώ η παράμετρος `random\_state` εξασφαλίζει την αναπαραγωγιμότητα των αποτελεσμάτων. Στη συνέχεια, δημιουργήσαμε ένα μοντέλο KNN με το `KNeighborsClassifier`, καθορίζοντας το `n\_neighbors` ως τον αριθμό των γειτόνων που πρέπει να ληφθούν υπόψη.

Αφού εκπαιδεύσαμε το μοντέλο με τα δεδομένα εκπαίδευσης, το χρησιμοποιήσαμε για να κάνουμε προβλέψεις στα δεδομένα δοκιμής (`X\_test`). Στη συνέχεια υπολογίσαμε την ακρίβεια του μοντέλου συγκρίνοντας τις προβλεπόμενες ετικέτες (`y\_pred`) με τις πραγματικές ετικέτες (`y\_test`). Επιπλέον, χρησιμοποιήσαμε τις συναρτήσεις `classification\_report` και `confusion\_matrix` για να αξιολογήσουμε περαιτέρω την απόδοση του μοντέλου.

Θυμηθείτε να αντικαταστήσετε τα `'X'` και `'y'` με τον πραγματικό πίνακα χαρακτηριστικών και τη μεταβλητή-στόχο, αντίστοιχα. Επιπλέον, μη διστάσετε να προσαρμόσετε την τιμή της `n\_neighbors` για να βρείτε τη βέλτιστη τιμή για το συγκεκριμένο σύνολο δεδομένων σας.