Sci-kit Learn materials for Machine Learning

Your appreciation and support matters a lot 🙏 .
Follow for more on **LinkedIn** [LINK]

Important Note :
`Green text -> Actual Code`

---

1. Dataset Splitting (train_test_split)

Import:

```
from sklearn.model_selection import
train_test_split
```

-

Use Case:
Used to split the dataset into training and testing sets. It helps to evaluate the model on unseen data.

```
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)
```

-

---

2. Pipelines

Import:

```
from sklearn.pipeline import Pipeline
```

-

Use Case:
Pipelines are used to streamline preprocessing and model building steps. You can chain multiple steps (e.g., scaling, PCA, classification) in one pipeline to ensure consistency.

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', LogisticRegression())
])
pipeline.fit(X_train, y_train)
```

-

---

3. StandardScaler

Import:

```
from sklearn.preprocessing import StandardScaler
```

-

Use Case:
StandardScaler standardizes features by removing the mean and scaling to unit variance. This is important for algorithms like SVM, KNN, and logistic regression.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

-

---

4. MinMaxScaler

Import:

```
from sklearn.preprocessing import MinMaxScaler
```

-

Use Case:
MinMaxScaler transforms features by scaling each feature to a given range (usually 0 to 1). This is useful for algorithms like neural networks where input data needs to be normalized.

```
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

-

---

5. LabelEncoder

Import:

```
from sklearn.preprocessing import LabelEncoder
```

-

Use Case:
LabelEncoder is used to convert categorical labels into numeric labels. This is required for machine learning algorithms to work with categorical target variables.

```
le = LabelEncoder()
```

```
y_encoded = le.fit_transform(y)
```

-

---

6. One-Hot Encoding

Import:

```
from sklearn.preprocessing import OneHotEncoder
```

-

Use Case:
OneHotEncoder is used for converting categorical features into a format that can be provided to ML algorithms, converting each category into a one-hot numeric array.

```
encoder = OneHotEncoder()
X_encoded = encoder.fit_transform(X)
```

-

---

7. SimpleImputer

Import:

```
from sklearn.impute import SimpleImputer
```

-

Use Case:
SimpleImputer fills in missing values using a specified strategy (mean, median, or most frequent). It is often used in data preprocessing.

```
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)
```

-

---

8. Cross-Validation (cross_val_score)

Import:

```
from sklearn.model_selection import
cross_val_score
```

-

Use Case:
`cross_val_score` is used to evaluate the model performance with cross-validation. It splits data into `k` folds and evaluates the model on each fold.

```
scores = cross_val_score(model, X, y, cv=5)
```

-

---

9. GridSearchCV

Import:

```
from sklearn.model_selection import GridSearchCV
```

-

Use Case:
GridSearchCV is used for hyperparameter tuning by searching over a specified parameter grid to find the best model parameters.

```
grid = GridSearchCV(model, param_grid, cv=5)
grid.fit(X_train, y_train)
```

-

---

10. RandomizedSearchCV

Import:

```
from sklearn.model_selection import
RandomizedSearchCV
```

-

Use Case:
Similar to `GridSearchCV`, but searches over random combinations of hyperparameters to find the best one, often faster for larger datasets.

```
random_search = RandomizedSearchCV(model,
param_distributions, n_iter=100, cv=5)
random_search.fit(X_train, y_train)
```

-

---

11. Feature Selection (SelectKBest)

Import:

```
from sklearn.feature_selection import SelectKBest,
chi2
```

-

Use Case:
`SelectKBest` selects the `k` best features based on a statistical test. This is used in feature selection for reducing dimensionality.

```
selector = SelectKBest(score_func=chi2, k=10)
X_new = selector.fit_transform(X, y)
```

-

---

12. Principal Component Analysis (PCA)

Import:

```
from sklearn.decomposition import PCA
```

-

Use Case:
PCA is a dimensionality reduction technique that reduces the number of features while retaining most of the variance in the data.

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

-

---

13. Logistic Regression

Import:

```
from sklearn.linear_model import
LogisticRegression
```

-

Use Case:
Logistic regression is used for binary classification problems. It models the probability of a target variable belonging to a class.

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

-

---

14. Support Vector Machines (SVM)

Import:

```
from sklearn.svm import SVC
```

-

Use Case:
SVM is used for classification and regression tasks. It works by finding the hyperplane that best separates the classes.

```
svm = SVC()
svm.fit(X_train, y_train)
```

-

---

15. Linear Regression (Ordinary Least Squares - OLS)

Import:

```
from sklearn.linear_model import LinearRegression
```

-

Use Case:
Linear regression models the relationship between input variables and a continuous target variable using OLS. It’s used in regression problems.

```
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

-

---

16. Lasso and Ridge Regression

Import:

```
from sklearn.linear_model import Lasso, Ridge
```

-

Use Case:
`Lasso` and `Ridge` regression are forms of regularised linear regression, used to prevent overfitting. Lasso performs feature selection by shrinking coefficients to zero, while Ridge penalises large coefficients but doesn’t shrink them to zero.

```
lasso = Lasso(alpha=0.1)
ridge = Ridge(alpha=0.1)
lasso.fit(X_train, y_train)
ridge.fit(X_train, y_train)
```

-

---

17. Decision Trees

Import:

```
from sklearn.tree import DecisionTreeClassifier,
DecisionTreeRegressor
```

-

Use Case:
`DecisionTreeClassifier` is used for classification problems, while `DecisionTreeRegressor` is used for regression problems. Both use a tree-like structure to make predictions.

```
tree = DecisionTreeClassifier()
tree.fit(X_train, y_train)
```

-

---

18. Random Forest

Import:

```
from sklearn.ensemble import
RandomForestClassifier, RandomForestRegressor
```

-

Use Case:
Random Forest is an ensemble method that combines multiple decision trees to improve performance. It can be used for both classification (`RandomForestClassifier`) and regression (`RandomForestRegressor`).

```
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
```

-

---

19. XGBoost (Extreme Gradient Boosting)

Import:

```
from xgboost import XGBClassifier, XGBRegressor
```

-

Use Case:
XGBoost is a highly efficient and powerful implementation of gradient boosting. It is widely used for both classification and regression tasks due to its performance in structured data.

```
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
```

-

---

20. Gradient Boosting

Import:

```
from sklearn.ensemble import
GradientBoostingClassifier,
GradientBoostingRegressor
```

-

Use Case:
Gradient Boosting builds models sequentially, optimizing each model to correct the errors made by previous models. It's effective for classification and regression.

```
gbc = GradientBoostingClassifier()
gbc.fit(X_train, y_train)
```

-

---

21. K-Means Clustering

Import:

```
from sklearn.cluster import KMeans
```

-

Use Case:
K-Means is used for unsupervised clustering problems. It partitions the dataset into `k` clusters.
python
Copy code

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
```

-

---

22. DBSCAN (Density-Based Clustering)

Import:

```
from sklearn.cluster import DBSCAN
```

-

Use Case:
DBSCAN is a clustering algorithm that forms clusters based on density, useful for datasets with noise and clusters of arbitrary

shape.

```
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan.fit(X)
```

-

---

23. K-Nearest Neighbors (KNN)

Import:

```
from sklearn.neighbors import
KNeighborsClassifier, KNeighborsRegressor
```

-

Use Case:
KNN is used for both classification and regression by looking at the `k` nearest neighbors of a point and assigning the majority class (classification) or averaging values (regression).

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

-

---

24. Naive Bayes

Import:

```
from sklearn.naive_bayes import GaussianNB
```

-

Use Case:
Naive Bayes is a classification algorithm based on Bayes' Theorem. It's particularly useful for text classification problems like spam detection.

```
nb = GaussianNB()
nb.fit(X_train, y_train)
```

-

---

25. Bagging Classifier

Import:

```
from sklearn.ensemble import BaggingClassifier
```

-

Use Case:
Bagging combines the predictions of multiple base estimators to improve generalization. Often used with decision trees.

```
bagging =
BaggingClassifier(base_estimator=DecisionTreeClass
ifier(), n_estimators=50)
bagging.fit(X_train, y_train)
```

-

---

26. AdaBoost (Adaptive Boosting)

Import:

```
from sklearn.ensemble import AdaBoostClassifier
```

-

Use Case:
AdaBoost works by combining multiple weak classifiers to create a strong classifier. Each classifier focuses on the errors of the previous one.

```
ada = AdaBoostClassifier(n_estimators=100)
ada.fit(X_train, y_train)
```

-

---

27. Voting Classifier

Import:

```
from sklearn.ensemble import VotingClassifier
```

-

Use Case:
Combines multiple models (e.g., logistic regression, SVM, etc.) and makes a prediction based on the majority vote (for classification tasks).

```
voting = VotingClassifier(estimators=[('lr',
logreg), ('rf', rf), ('svc', svm)], voting='hard')
voting.fit(X_train, y_train)
```

-

---

28. Mean Squared Error (MSE)

Import:

```
from sklearn.metrics import mean_squared_error
```

-

Use Case:
MSE is used to evaluate regression models by calculating the average of squared errors between predicted and actual values.

```
mse = mean_squared_error(y_test, y_pred)
```

-

---

29. Confusion Matrix

Import:

```
from sklearn.metrics import confusion_matrix
```

-

Use Case:
Confusion matrix is used to evaluate classification models by summarizing the correct and incorrect predictions made by the model.

```
cm = confusion_matrix(y_test, y_pred)
```

-

---

30. ElasticNet

Import:

```
from sklearn.linear_model import ElasticNet
```

-

Use Case:
ElasticNet is a linear regression model that combines both L1 and L2 regularization techniques (Lasso and Ridge). It helps with feature selection and model performance.

```
enet = ElasticNet(alpha=0.1, l1_ratio=0.7)
enet.fit(X_train, y_train)
```

-
