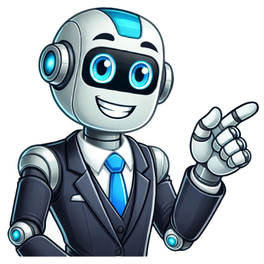
Continue



Scikit-learn user guide

You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. By introducing the process, it can motivate more sophisticated extensions of this type of imputation such as using a statistical model to replace missing values. Let’s consider the iris data for our Python implementation of a logistic regression model. Let’s consider the regression task of predicting housing prices using the Boston housing data. To demonstrate how to perform data imputation using Scikit-learn, we’ll work with the University of California–Irvine’s data set on housing electric power consumption, which is available here. The library also enables data processing tasks such as imputation, data standardization and data normalization. Creating Polynomial Features: Generates additional features that represent polynomial combinations of the original ones, capturing non-linear patterns. For example, one tree may ask about age and credit score on a fraction of the train data. First, let’s import the StandardScalar() method from Scikit-learn: scaler = StandardScaler() scaler.fit(np.array(df_new[['Global_intensity']])) df_new['Global_intensity'] = scaler.transform(np.array(df_new[['Global_intensity']])) Now we see that the min and max are 7.6 and -1.0: print(df_new.head()) print("Max: ", df_new['Global_intensity'].max()) print("Min: ", df_new['Global_intensity'].min()) Image: Screenshot Normalization Data normalization scales a numerical column such that its values are between 0 and 1. All of these data sets are easy to load using a few simple lines of Python code. Access the ... What is scikit-learn? Logistic Regression Logistic regression is a simple classification model that predicts binary or even multiclass output. These sets allow beginners to quickly get their feet wet with different types of data and use cases such as regression, classification and image recognition. In each of these trees, we ask statistical questions on random chunks and different features of the data. Python from sklearn.preprocessing import StandardScaler scaler = StandardScaler().fit(X_train) X_train_standardized = scaler.transform(X_train) X_test_standardized = scaler.transform(X_test) b. Let’s consider standardizing the Global_intensity in the power consumption data set. These techniques standardize, normalize, or otherwise prepare data.a. Standardization: Ensures that features have zero mean and unit variance, which improves model performance. No warranties are given. Although I’ve only covered three in this post, the logic for building other widely used models such as support vector machines and K-nearest neighbors is very similar. All we need to do is import the random forest regressor module, initiate the regressor object, fit, test and evaluate our model: from sklearn.ensemble import RandomForestRegressor rf_reg = RandomForestRegressor() rf_reg.fit(X_train, y_train) y_pred = rf_reg.predict(X_test) from sklearn.metrics import mean_squared_error from sklearn.metrics import r2_score rms = mean_squared_error(y_test, y_pred) r2 = r2_score(y_test, y_pred) print("RF MSE:", rms) print("RF R^2:", r2) Image: Screenshot We see a slight improvement in performance compared to linear regression. By learning how to use it, you'll be ready to apply machine learning to many different challenges. The logic for training and testing is similar to linear regression. This tutorial will serve as an introduction to some of its functions. This column has values ranging from 0.2 to 36. The data that falls into the “yes” bucket will have more customers who default than the data that falls into the “no” bucket. Linear regression is used for regression tasks. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. Random Forests Random forests, also called random decision trees, is a statistical model for both classification and regression tasks. Since the data set is quite large, we’ll take a random sample of 40,000 records for simplicity and store the down-sampled data in a separate csv file called “hpc.csv”: df = pd.read_csv('household_power_consumption.txt', sep=';') df = df.sample(40000) df.to_csv('hpc.csv') Next, let’s read in our newly created data set and print the first five rows: df = pd.read_csv('hpc.csv') print(df.head()) Image: Screenshot As we can see, the third row (second index) contains missing values specified by ? Instead, we can display the first five digits in the data using the visualization library matplotlib: from sklearn.datasets import load_iris, load_boston, load_digits import matplotlib.pyplot as plt def get_data(dataset): try: data = dataset df = pd.DataFrame(data.data,columns=data.feature_names) df['target'] = pd.Series(data.target) print(df.head()) except(AttributeError): data = dataset plt.gray() for i in range(0,5): plt.matshow(data.images[i]) plt.show() And if we call our function with load_digits(), we get the following displayed images: Image: Screenshot I can’t overstate the ease with which a beginner in the field can access these toy data sets. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Python from sklearn.preprocessing import LabelEncoder encoder = LabelEncoder() y_encoded = encoder.fit_transform(y) e. Random forest then performs consensus voting across these decision trees and uses the majority vote for the final prediction. Scikit-learn provides many metrics for this purpose.a. Metrics for Classification ModelsAccuracy Score: Measures the proportion of correctly predicted labels. Python from sklearn.linear_model import LinearRegression model = LinearRegression() model.fit(X_train, y_train) y_pred = model.predict(X_test) Naive Bayes: A fast algorithm based on Bayes’ theorem, often used for text classification. Decision trees continue asking statistical questions about the data until achieving maximal separation between the data corresponding to those who default and those who don’t. They’re also useful in cases where we can assume the data is normally distributed and for interpreting coefficients in linear models to be of variable importance. This enables fast prototyping and experimentation of models, which leads to accurate results faster. For example, it has a set called iris data, which contains information corresponding to different types of iris plants. In general, you should standardize data if you can safely assume it’s normally distributed. This is often necessary in order to achieve satisfactory performance with more complicated models like support vector machines and neural networks. values with NaN values. Scikit-learn is a powerful machine learning library that provides a wide variety of modules for data access, data preparation and statistical model building. Additionally, it gives you the skills to prepare data, select the optimal model, and assess performance. These tasks can often lead to significant improvements in model performance. Normaliziation: Scales individual rows of data so that their norm equals 1, which is useful for distance-based models like KNN. Whether you're working on a small task or a big project, Scikit-learn offers the flexibility and support you need. Similar to linear regression, logistic regression depends on a linear sum of inputs used to predict each class. Python from sklearn.cluster import KMeans kmeans = KMeans(n_clusters=3, random_state=42) kmeans.fit(X_train) y_pred = kmeans.predict(X_test) Evaluating Model PerformanceEvaluation metrics are used to judge a model's performance. We’ll use sepal length (cm), sepal width (cm), petal length (cm) and petal width (cm) to predict the type of iris plant: df_iris= get_data(load_iris()) X = df_iris[['sepal length (cm)', 'sepal width (cm) ', 'petal length (cm)', 'petal width (cm)']] y = df_iris[['target']] X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42) logistic_model = LogisticRegression() logistic_model.fit(X_train, y_train) y_pred = linear_model.predict(X_test) We can evaluate and visualize the model performance using a confusion matrix: Image: Screenshot We see that the model correctly captures all of the true positives across the three iris plant classes. Python from sklearn.metrics import mean_squared_error print("MSE:", mean_squared_error(y_test, y_pred)) R² Score: Indicates how well the model explains the variance in the target variable. Python from sklearn.metrics import accuracy_score print("Accuracy:", accuracy_score(y_test, y_pred)) Classification Report (Precision, Recall, F1): Provides detailed metrics for classification tasks. Linear Regression Linear regression is a statistical modeling approach in which a linear function represents the relationship between input variables and a scalar response variable. It has a good selection of clean toy data sets that are great for people just getting started with data analysis and machine learning. We’ll choose 100 estimators, a max depth of 10 and a random state of 42: rf_reg = RandomForestRegressor(n_estimators= 100, max_depth=10, random_state =42) Image: Screenshot We see that we get a slight improvement in both MSE and R^2. Finally, the handwritten digits data set is an image data set that is great for building image classification models. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . All of these tasks require relatively few lines of code, making the barrier to entry for beginners in data science and machine learning research quite low. Python from sklearn.metrics import classification_report print(classification_report(y_test, y_pred)) Confusion Matrix Insights: Shows the counts of true positives, true negatives, false positives, and false negatives. It involves measuring its accuracy or error rate on test data. Whether the task is model benching marking with toy data, preparing/cleaning data, or evaluating model performance Scikit-learn is a fantastic tool for building machine learning models for a wide variety of use cases. It’s also useful as a benchmark against more sophisticated methods like random forests and support vector machines. Let’s build a linear regression model with age (AGE), average number of rooms (RM), and pupil-to-teacher ratio (PTRATION). The parameter n_estimators is simply the number of decision trees that the random forest is made up of. This will result in slight variations in output and performance. Further, specifying random_state makes our results reproducible since it ensures the same random chunks of data are used to construct the decision trees. Python from sklearn.model_selection import train_test_split X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) Data Transformation TechniquesPreprocessing transforms raw data into a suitable format for machine learning models. Further, new data scientists unfamiliar with data imputation can quickly pick up how to use the SimpleImputer package in Scikit-learn and implement some standard methods for replacing missing or bad values in data. The licensor cannot revoke these freedoms as long as you follow the license terms. Since we didn’t specify any values for these parameters, the random forest module automatically selects a default value for each parameter. Handling Missing Values: Handle missing data with a strategy like replacing with the mean, median, or mode. Max_depth measures the longest path from the first question to a question at the base of the tree. A question that we can ask using historical lending data is whether or not the customer’s credit score is below 700. Random forests are basically a set of questions and answers about the data organized in a tree-like structure. Upon becoming familiar with different use cases, the user can then easily port over what they’ve learned to more real-life applications. Lab Objective: The scikit-learn package is the one of the fundamental tools in Python for machine learning. It provides simple and powerful tools to help you turn data into useful predictions. All we need to do is redefine X (input) as follows: X = df_housing[['AGE', 'PTRATIO', 'RM']] This gives the following improvement in performance: Image: Screenshot Linear regression is a great method to use if you’re confident that there is a linear relationship between input and output. We first need to import the pandas and numpy packages: import pandas as pd import numpy as np Next, we relax the display limits on the columns and rows: pd.set_option('display.max_columns', None) pd.set_option('display.max_rows', None) We then load the iris data from Scikit-learn and store it in a pandas data frame: from sklearn.datasets import load_iris data = load_iris() df = pd.DataFrame(data.data,columns=data.feature_names) df['target'] = pd.Series(data.target) Finally, we print the first five rows of data using the head() method: print(df.head()) Image: Screenshot We can repeat this process for the Boston housing data set. Metrics for Regression ModelsMean Absolute Error (MAE): Measures the average absolute difference between predicted and actual values. Although Scikit-learn’s SimpleImputer isn’t the most sophisticated imputation method, it removes much of the hassle around building a custom imputer. Python from sklearn.preprocessing import Binarizer binarizer = Binarizer(threshold=1.0).fit(X) X_binarized = binarizer.transform(X) d. Scikit-learn also has a Boston housing data set, which contains information on housing prices in Boston. Python from sklearn.metrics import r2_score print("R² Score:", r2_score(y_test, y_pred)) c. Random_state is how the algorithm randomly chooses chunks of the data for question-asking. Finally, Scikit-learn makes building a wide variety of machine learning models very easy. Overall, Scikit-learn provides many easy-to-use modules and methods for accessing and processing data and building machine learning models in Python. Jump Into Machine LearningThe Top 10 Machine Learning Algorithms Every Beginner Should Know By allowing systems to learn from data and make judgments without explicit programming, machine learning is revolutionizing a number of sectors. It is also very suitable for beginners who have limited knowledge of how these algorithms work under the hood, given that each model object comes with default parameters that give baseline performance. In the context of Scikit-learn, they’re extremely easy to implement and modify for improvements in performance. This means, upon each model run, different chunks of data will be randomly selected and used to construct the decision trees in the random forests. Since this is an image data set, it’s neither necessary nor useful to store it in a data frame. Both of these are useful in machine learning methods that involve calculating a distance metric like K-nearest neighbors and support vector machines. Even better, easy access to these data sets removes the hassle of searching for and downloading files from an external data source. Python from sklearn.naive_bayes import GaussianNB model = GaussianNB() model.fit(X_train, y_train) y_pred = model.predict(X_test) Python from sklearn.neighbors import KNeighborsClassifier model = KNeighborsClassifier(n_neighbors=3) model.fit(X_train, y_train) y_pred = model.predict(X_test) Unsupervised Learning AlgorithmsUnsupervised learning is used when the data has no labels or target variable, often for clustering or dimensionality reduction.PCA: Reduces high-dimensional data into fewer dimensions while preserving variance. Beginners in machine learning will also find the library useful since each model object is equipped with default parameters that provide baseline performance. Random forests can be used for both regression and classification. We’ll walk through how to implement each of these models using the Scikit-learn machine learning library in Python. Users can employ this data for building, training and testing classification models that can classify types of iris plants based on their characteristics. This accuracy demonstrates the power of random forests and the ease with which the data science beginner can implement an accurate random forest model. This can serve as the foundation for learning more advanced methods of data imputation, such as using a statistical model for predicting missing values. Data Imputation Scikit-learn also provides a variety of methods for data processing tasks. The user guide covers various algorithms, models, parameters, and examples ... Learn how to use scikit-learn, a simple and efficient tool for predictive data analysis, with examples and algorithms for classification, regression, clustering, and more. As such, logistic regression models are referred to as generalized linear models. Finally, all the code in this post is available on GitHub. Standardization Standardization is the process of subtracting values in numerical columns by the mean and scaling to unit variance (through dividing by the standard deviation). These questions split the data into subgroups so that the data in each successive subgroup are most similar to each other. This ensures we can evaluate the model on unseen data.Loading Built-in Datasets: Scikit-learn provides datasets like Iris and Boston Housing for experimentation. Adapt — remix, transform, and build upon the material for any purpose, even commercially. Data Standardization and Normalization Data standardization and normalization are also easy with Scikit-learn. Being able to accurately impute missing values is a skill that both data scientists and industry domain experts should have in their toolbox. This cheat sheet will be a useful resource to effectively create machine learning models, covering everything from data pretreatment to model evaluation.What is Scikit-learn?Scikit-learn is an open-source, free Python library. Python from sklearn.decomposition import PCA pca = PCA(n_components=2) X_pca = pca.fit_transform(X_train) K-Means: Groups similar data points into clusters based on their features. Both beginners and experts wishing to efficiently create, improve, and assess machine learning models turn to it because of its ease of use and extensive feature set.Scikit-Learn Cheat-SheetIn this article, we provide a Scikit-learn Cheat Sheet that covers the main features, techniques, and tasks in the library. Let’s do this for the iris classification task: rf_cass = RandomForestClassifier(n_estimators= 100, max_depth=10, random_state =42) rf_cass.fit(X_train, y_train) y_pred = rf_cass.predict(X_test) And the corresponding confusion matrix is just as accurate: Image: Screenshot Random forests are a great choice for building a statistical model since they can be applied to a wide range of prediction use cases. Python from sklearn.preprocessing import Normalizer scaler = Normalizer().fit(X_train) X_train_normalized = scaler.transform(X_train) X_test_normalized = scaler.transform(X_test) c. Let’s demonstrate this with Global_active_power: df['Global_active_power'].replace('?', np.nan, inplace = True) print(df.head()) Image: Screenshot We can repeat this process for the rest of the columns: df['Global_reactive_power'].replace('?', np.nan, inplace = True) df['Voltage'].replace('?', np.nan, inplace = True) df['Global_intensity'].replace('?', np.nan, inplace = True) df['Sub_metering_1'].replace('?', np.nan, inplace = True) df['Sub_metering_2'].replace('?', np.nan, inplace = True) Image: Screenshot Now, to impute the missing values, we import the SimpleImputer method from Scikit-learn. Applying random forest models to classification tasks is very straightforward. Scikit-learn is a powerful machine learning library that provides a wide variety of modules for data access, data preparation and statistical model building. Share — copy and redistribute the material in any medium or format for any purpose, even commercially. We will define an imputer object that simply imputes the mean for missing values: from sklearn.impute import SimpleImputer imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean') And we can fit our imputer to our columns with missing values: X = df[['Global_active_power', 'Global_reactive_power', 'Voltage', 'Global_intensity', 'Sub_metering_1','Sub_metering_2']] imp_mean.fit(X) Store the result in a data frame: df_new = pd.DataFrame(imp_mean.transform(X), columns = ['Global_active_power', 'Global_reactive_power', 'Voltage', 'Global_intensity', 'Sub_metering_1','Sub_metering_2']) Add back the additional date and time columns: df_new['Date'] = df['Date'] df_new['Time'] = df['Time'] And print the first five rows of our new data frame: print(df_new.head()) Image: Screenshot As we can see, the missing values have been replaced. To start, let’s define our input and output variables: X = df_housing[['AGE']] y = df_housing[['target']] Next, let’s split our data for training and testing: from sklearn.model_selection import train_test_split X_train, y_train, X_test, y_test = train_test_split(X, y, random_state = 42) Now let’s import the linear regression module from Scikit-learn: from sklearn.linear_models import LinearRegression Finally, let’s train, test and evaluate the performance of our model using R^2 and RMSE: linear_model = LinearRegression() linear_model.fit(X_train, y_train) y_pred = linear_model.predict(X_test) from sklearn.metrics import mean_squared_error from sklearn.metrics import r2_score rms = mean_squared_error(y_test, y_pred) r2 = r2_score(y_test, y_pred) print("MSE:", rms) print("R^2:", r2) Image: Screenshot Since we use one variable to predict a response, this is a simple linear regression. This data is useful for regression tasks like predicting the dollar value of a house. Within the yes bucket, we can further ask if the borrower’s income is below $30,000. Logistic regression is used for classification tasks in which the model predicts binary output or multiclass like predicting iris plant type based on characteristics. To demonstrate its implementation in Python, let’s consider the Boston housing data set. Implementing a random forests model for both regression and classification is straightforward and very similar to the steps we went through for linear regression and logistic regression. For example, say we’d like to predict whether or not a borrower will default on a loan. In this appendix we highlight and give examples of some of the more popular scikit ... Learn how to use scikit-learn for supervised and unsupervised learning, with examples of estimators, transformers, pipelines, and model evaluation. Python from sklearn.metrics import confusion_matrix print(confusion_matrix(y_test, y_pred)) b. Statistical Modeling With Scikit-Learn Scikit-learn also has methods for building a wide array of statistical models, including linear regression, logistic regression and random forests. Python from sklearn.metrics import homogeneity_score print("Homogeneity:", homogeneity_score(y_test, y_pred)) V-Measure: Measures the balance between homogeneity and completeness in clustering results. It covers important topics like creating models, testing their performance, working with different types of data, and using machine learning techniques like classification, regression, and clustering. This simplicity is useful for beginners who are dealing with missing data for the first time. The default value for n_estimators is 10, which corresponds to 10 decision trees. Standardization is necessary in cases where a wide range of numerical values may artificially dominate prediction outcomes. This can be roughly understood as the limit on the number of questions we ask about the data. Python from sklearn.model_selection import RandomizedSearchCV random = RandomizedSearchCV(SVC(), param_distributions={ 'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}, n_iter=10) random.fit(X_train, y_train) ConclusionIn conclusion, Scikit-learn makes machine learning easier and more accessible. Given that these transformations can be applied to numerical data with just a few lines of code, the StandardScaler() and Normalizer() methods are great options for beginners dealing with data fields that have widely varying values or data that isn’t normally distributed. Random forests extend decision trees by constructing a multitude of them. Another may ask about income and gender on a separate fraction of the training data, and so forth. Let’s see how to specify n_estimators, max_depth and random_state. Experiment With Scikit-Learn Overall, Scikit-learn provides many easy-to-use tools for accessing benchmark data, performing data processing, and training, testing and evaluating machine learning models. This can result in misleading results and poor model performance. Normalizing data using Scikit-learn follows similar logic to standardization. One of the most well-known and easily available libraries for machine learning in Python is Scikit-learn. Python from sklearn.impute import SimpleImputer imputer = SimpleImputer(strategy='mean') X_imputed = imputer.fit_transform(X) f. But we can also use more than one variable in a multiple linear regression. It features an easy-to-use interface for each model object type, which facilitates fast prototyping and experimentation with models. Further, it serves as a good demonstration of how imputation works. The first thing we can do is replace the ? Presumably, the yes bucket here will have an even greater percentage of customers who default. Scikit-learn, which is built on top of existing Python libraries like NumPy and SciPy, is easy to use, popular, and perfect for both novices and machine learning specialists.Scikit-learn Cheat-SheetThis Scikit-learn Cheat Sheet will help you learn how to use Scikit-learn for machine learning. To do so, let’s wrap our existing code in a function that takes a Scikit-learn data set as input: def get_data(dataset): data = dataset df = pd.DataFrame(data.data,columns=data.feature_names) df['target'] = pd.Series(data.target) print(df.head()) We can call this function with the iris data and get the same output as before: get_data(load_iris()) Image: Screenshot Now that we see that our function works, let’s import the Boston housing data and call our function with the data: from sklearn.datasets import load_iris, load_boston get_data(load_boston()) Image: Screenshot Finally, let’s load the handwritten digits data set, which contains images of handwritten digits from zero through nine. With its wide range of algorithms and features, you can quickly build, test, and improve your models. Python from sklearn.datasets import load_iris data = load_iris() X, y = data.data, data.target Splitting Data into Training and Testing: Split the dataset into training data for model learning and testing data for evaluation. The license may not give you all of the permissions necessary for your intended use. To start, let’s walk through loading the iris data. It facilitates activities such as classifying data, clustering similar data, forecasting values, and simplifying data for tasks like dimensionality reduction. Specifically, it works for the prediction of continuous output like housing price, for example. Let’s apply the normalizer method to the Sub_metering_2 column: from sklearn.preprocessing import Normalizer normalizer = Normalizer() normalizer.fit(np.array(df_new[['Sub_metering_2']])) df_new['Sub_metering_2'] = normalizer.transform(np.array(df_new[['Sub_metering_2']])) print(df_new.head()) print("Max: ", df_new['Sub_metering_2'].max()) print("Min: ", df_new['Sub_metering_2'].min()) Image: Screenshot Now we see that the min and max are 1.0 and 0. Users can quickly access toy data sets and familiarize themselves with different machine learning use cases (classification, regression, clustering) without the hassle of finding a data source, downloading and then cleaning the data. The default value for random_state is None. The default value for max_depth is None, which means there is no cut-off for the length of the path from the first question to the last question at the base of the decision tree. The ones available in Scikit-learn can be applied to supervised learning tasks such as regression and classification. We can build a linear regression model that uses age as an input for predicting the housing value. Given that logistic regression models a linear relationship between input and output, they’re best employed when you know that there is a linear relationship between input and class membership. And having the appropriate tools is crucial in this quickly changing sector. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. Encoding Non-Numerical Data: Converts categorical features into numeric ones using label encoder. Python from sklearn.metrics import mean_absolute_error print("MAE:", mean_absolute_error(y_test, y_pred)) Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values. Scikit-learn also provides a variety of packages for building linear models, tree-based models, clustering models and much more. Scikit-learn Data Sets Scikit-learn provides a wide variety of toy data sets, which are simple, clean, sometimes fictitious data sets that can be used for exploratory data analysis and building simple prediction models. The three I’ll point out here are n_estimators, max_depth and random_state. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . Metrics for ClusteringAdjusted Rand Index: Evaluates the similarity between two clusterings by considering all pairs of points. Despite using default values, we achieve pretty good performance. Conversely, if you can safely assume that your data isn’t normally distributed, then normalization is a good method for scaling it. The random forest object takes several parameters that can be modified to improve performance. Find out how to install, choose, and ... Learn how to use scikit-learn, a Python machine learning library, for supervised and unsupervised learning tasks. Python from sklearn.metrics import v_measure_score print("V-Measure:", v_measure_score(y_test, y_pred)) Optimizing ModelsModel optimization involves fine-tuning hyperparameters to improve performance.a. Exhaustive Search with GridSearchCV: Tests all combinations of hyperparameters to find the best set. Python from sklearn.model_selection import GridSearchCV param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']} grid = GridSearchCV(SVC(), param_grid, cv=5) grid.fit(X_train, y_train) print("Best Parameters:", grid.best_params_) b. Randomized Search for Hyperparameters: Randomly samples hyperparameters for a faster search. Additionally, the standard scaler and normalizer methods make data preparation for advanced models like neural networks and support vector machines very straightforward. It is changing how companies function and innovate in a variety of industries, including healthcare and entertainment, opening up new avenues for automation and clever solutions. It’s a fantastic tool that every data scientist should have in their back pocket. First, let’s take a look at data imputation, which is the process of replacing missing data and is important because oftentimes real data contains either inaccurate or missing elements. Python from sklearn.preprocessing import PolynomialFeatures poly = PolynomialFeatures(degree=2) X_poly = poly.fit_transform(X) Building Machine Learning ModelsSupervised Learning AlgorithmsSupervised learning involves training models on labeled data, where the target variable is known.Linear Regression: Used to predict continuous values by fitting a linear relationship between input features and target variables. I Simple and efficient tools for predictive data analysis I Machine Learning methods I Data processing I Visualization I Accessible to everybody, and reusable in various ... You can check out the documentation for a full description of all random forest parameters. Binarization: Converts numeric features into binary values based on a threshold. It’s a great guide to help you get hands-on experience and explore machine learning more easily.Download the Cheat-Sheet here: Scikit-learn Cheat-SheetInstalling Scikit-learnOnce you have Python installed, you can use the following command to install the scikit-learn library on Windows:pip install scikit-learnData PreprocessingFunctionDescriptionStandardScalerStandardize features by removing the mean and scaling to unit variance.MinMaxScalerScale features to a specific range (e.g., 0 to 1).BinarizerTransform features into binary values (thresholding).LabelEncoderEncode target labels with values between 0 and n_classes-1.OneHotEncoderPerform one-hot encoding of categorical features.PolynomialFeaturesGenerate polynomial and interaction features.SimpleImputerImpute missing values using a strategy (mean, median, most frequent).KNNImputerImpute missing values using k-nearest neighbors.Model Selection and EvaluationFunctionDescriptiontrain_test_splitSplit data into training and testing setscross_val_scorePerform cross-validation on the model.cross_val_predictCross-validation generator for predictions.accuracy_scoreEvaluate classification accuracy.confusion_matrixGenerate confusion matrix for classification.classification_reportDetailed classification report (precision, recall, F1-score).mean_squared_errorEvaluate regression performance with mean squared error.r2_scoreEvaluate regression performance with R² score.roc_auc_scoreCompute area under the ROC curve for binary classification.f1_scoreCompute the F1 score for classification models.precision_scoreCompute precision score for classification models.recall_scoreCompute recall score for classification models.Classification ModelsFunctionDescriptionLogisticRegressionA linear model used for binary or multi-class classification.SVCSupport Vector Classifier, used for both linear and non-linear classification.RandomForestClassifierAn ensemble method that builds multiple decision trees for robust classification.GradientBoostingClassifierAn ensemble method that builds trees sequentially to correct errors of previous trees.GaussianNBNaive Bayes classifier based on Gaussian distribution of data.KNeighborsClassifierClassifier that assigns labels based on nearest neighbors' majority class.DecisionTreeClassifierA tree-based classifier that splits data into branches to make decisions.Regression ModelsFunctionDescriptionLinearRegressionA linear model used to predict continuous numerical values.RidgeA linear regression model with L2 regularization to prevent overfitting.LassoA linear regression model with L1 regularization to enhance sparsity.DecisionTreeRegressorA tree-based model that predicts continuous values by learning splits on the data.RandomForestRegressorAn ensemble method that averages the predictions of multiple decision trees for better accuracy.SVRSupport Vector Regressor, used for predicting continuous values with support vector machines.Clustering ModelsFunctionDescriptionKMeansA popular clustering algorithm that partitions data into k distinct clusters based on similarity.DBSCANA density-based clustering algorithm that groups data points based on density, allowing for irregular shapes.AgglomerativeClusteringA hierarchical clustering method that builds clusters iteratively by merging or splitting clusters.Dimentionality ReductionFunctionDescriptionPCAPrincipal Component Analysis (PCA) reduces the number of features by finding new dimensions that maximize variance.TruncatedSVDA dimensionality reduction method suited for sparse matrices, especially in text mining.t-SNEA technique for visualizing high-dimensional data by mapping it to a lower-dimensional space.FeatureAgglomerationA method for feature reduction that merges features based on their similarity.Model Training and PredictionFunctionDescriptionfit()Trains the model using the provided data (X_train, y_train).predict()Makes predictions based on the trained model for unseen data (X_test).fit_predict()Combines training and prediction into a single method, commonly used in clustering.predict_proba()Returns probability estimates for classification models, indicating class likelihoods.score()Evaluates the model's performance using a scoring metric, typically accuracy for classification or R² for regression.Hands-on Practice with Scikit-learnImporting and Preparing DataBefore building models, we need to load our dataset and split it into training and testing subsets. Python from sklearn.metrics import adjusted_rand_score print("ARI:", adjusted_rand_score(y_test, y_pred)) Homogeneity: Checks if clusters contain only data points that belong to a single class. This includes classification, regression and even unsupervised clustering tasks. and NaN.