2- datu library

Module 1.1: machine learning	Duration <1	Type concept	Category: Introduction
Machine learning is a subset of AI that enables computers to learn and improve from data without explicit programming. It uses algorithms to analyze data, identify patterns, and make predictions. This iterative process empowers machines to adapt and improve their performance over time.			
Machine learning is a field of study and application within artificial intelligence (AI) that focuses on developing computer systems capable of automatically learning and improving from experience without being explicitly programmed. It involves the development of algorithms and models that enable machines to analyze and interpret complex data, identify patterns, and make predictions or decisions based on the observed patterns. Machine learning algorithms are designed to iteratively learn from data, allowing them to adapt and improve their performance over time. **At its core, machine learning involves the utilization of statistical techniques and mathematical models to enable machines to learn and extract meaningful information from data. The learning process typically involves three key components: data, models, and optimization. First, large volumes of relevant and diverse data are collected, which serve as the input for the learning process. Next, a machine learning model is created, which represents the relationships and patterns within the data. This model is then trained using various algorithms and techniques to optimize its parameters and improve its ability to generalize from the provided data. Once the model is trained, it can be deployed to make predictions or decisions on new, unseen data, effectively leveraging the knowledge and patterns learned from the training phase.** **Machine learning finds applications in various domains, including image and speech recognition, natural language processing, recommender systems, fraud detection, and autonomous vehicles, among others.**			

Module 1.2: ML python packages	Duration: 5	Type code	Category: Introduction

Machine Learning is the science of programming computers to learn from data. It eliminates the need for explicit programming by utilizing algorithms and mathematical formulas. Python is a popular language for this task, with libraries like NumPy, SciPy, scikit-learn, TensorFlow, Keras, and PyTorch providing efficient solutions for data processing and analysis. These libraries enable tasks such as matrix processing, optimization, image manipulation, classical ML algorithms, deep learning, and neural network design, making Machine Learning more accessible and efficient than ever before.

Machine Learning, as the name suggests, is the science of programming a computer by which they are able to learn from different kinds of data. A more general definition given by Arthur Samuel is – “Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.” They are typically used to solve various types of life problems.

In the older days, people used to perform Machine Learning tasks by manually coding all the algorithms and mathematical and statistical formulas. This made the processing time-consuming, tedious, and inefficient. But in the modern days, it is become very much easy and more efficient compared to the olden days with various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reasons is its vast collection of libraries. Python libraries that are used in Machine Learning are:

Numpy
Scipy
Scikit-learn
TensorFlow
Keras
PyTorch
Pandas
Matplotlib
NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

Scikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with ML.

TensorFlow is a very popular open-source library for high performance numerical computation developed by the Google Brain team in Google. As the name suggests, Tensorflow is a framework that involves defining and running computations involving tensors. It can train and run deep neural networks that can be used to develop several AI applications. TensorFlow is widely used in the field of deep learning research and application.

Keras is a very popular Machine Learning library for Python. It is a high-level neural networks API capable of running on top of TensorFlow, CNTK, or Theano. It can run seamlessly on both CPU and GPU. Keras makes it really for ML beginners to build and design a Neural Network. One of the best thing about Keras is that it allows for easy and fast prototyping.

PyTorch is a popular open-source Machine Learning library for Python based on Torch, which is an open-source Machine Learning library that is implemented in C with a wrapper in Lua. It has an extensive choice of tools and libraries that support Computer Vision, Natural Language Processing(NLP), and many more ML programs. It allows developers to perform computations on Tensors with GPU acceleration and also helps in creating computational graphs.

Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for grouping, combining and filtering data.

Matplotlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc,

Module 1.3: Type of ML	Duration =1	Type concept	Category: Introduction

Machine learning encompasses various algorithms, including supervised learning (using labeled data for prediction) and unsupervised learning (finding patterns in unlabeled data). These techniques enable tasks like classification, regression, clustering, and anomaly detection.

There are several types of machine learning algorithms that are commonly used in the field. Supervised learning is one type where the algorithm is trained using labeled examples, meaning the input data is paired with the corresponding desired output. The algorithm learns to map inputs to outputs by finding patterns in the labeled data. This type of learning is often used for tasks like classification, regression, and prediction.

Another type is unsupervised learning, where the algorithm is given unlabeled data and is tasked with finding patterns or structures within the data. Unlike supervised learning, there are no predefined outputs to learn from. Unsupervised learning algorithms focus on discovering inherent relationships, clustering similar data points, or reducing the dimensionality of the data. This type of learning is useful for tasks such as anomaly detection, data exploration, and recommendation systems.

Module 1.4: import Library	Duration =3	Type code	Category: Introduction

To utilize popular machine learning packages in Python, import scikit-learn (sklearn), Matplotlib, pandas, NumPy, and Seaborn. These libraries provide essential tools for data analysis, visualization, and machine learning algorithms.

To import popular machine learning packages such as scikit-learn (sklearn), Matplotlib, pandas, NumPy, and Seaborn, you can use the following code:

```
# importing libraries
import sklearn
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
```

The above code imports the packages and assigns them convenient aliases (sklearn, plt, pd, np, sns), which can be used to access the functionality provided by

each package. These packages offer a wide range of tools and functions for data manipulation, visualization, statistical analysis, and machine learning algorithms.

Module 2.1: introduction to data	Duration 2	Type concept	Category: data
DATA: It can be any unprocessed fact, value, text, sound, or picture that is not being interpreted and analyzed. Data is the most important part of all Data Analytics, Machine Learning, Artificial Intelligence. Without data, we can’t train any model and all modern research and automation will go in vain. Big Enterprises are spending lots of money just to gather as much certain data as possible. **How we split data in Machine Learning?** **Training Data: The part of data we use to train our model. This is the data that your model actually sees(both input and output) and learns from.** **Validation Data: The part of data that is used to do a frequent evaluation of the model, fit on the training dataset along with improving involved hyperparameters (initially set parameters before the model begins learning). This data plays its part when the model is actually training.** **Testing Data: Once our model is completely trained, testing data provides an unbiased evaluation. When we feed in the inputs of Testing data, our model will predict some values(without seeing actual output). After prediction, we evaluate our model by comparing it with the actual output present in the testing data. This is how we evaluate and see how much our model has learned from the experiences feed in as training data, set at the time of training.**			

Module 2.2: data preprocessing	Duration 7	Type code	Category: data
Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. **This article contains 3 different data preprocessing techniques for machine learning.**			

1. Rescale Data

When our data is comprised of attributes with varying scales, many machine learning algorithms can benefit from rescaling the attributes to all have the same scale.
This is useful for optimization algorithms used in the core of machine learning algorithms like gradient descent.
It is also useful for algorithms that weight inputs like regression and neural networks and algorithms that use distance measures like K-Nearest Neighbors.
We can rescale your data using scikit-learn using the MinMaxScaler class.

```
# importing libraries
import pandas
import scipy
import numpy
from sklearn.preprocessing import MinMaxScaler

# data set link
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/pima
-indians-diabetes/pima-indians-diabetes.data"
# data parameters
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass',
'pedi', 'age', 'class']

# preparating of dataframe using the data at given link and
defined columns list
dataframe = pandas.read_csv(url, names = names)
array = dataframe.values

# separate array into input and output components
X = array[:,0:8]
Y = array[:,8]

# initialising the MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
# learning the statistical parameters for each of the data and
transforming
rescaledX = scaler.fit_transform(X)

# summarize transformed data
numpy.set_printoptions(precision=3)
print(rescaledX[0:5,:])
```

After rescaling see that all of the values are in the range between 0 and 1.

2. Binarize Data (Make Binary)

We can transform our data using a binary threshold. All values above the threshold are marked 1 and all equal to or below are marked as 0.
This is called binarizing your data or threshold your data. It can be useful when you have probabilities that you want to make crisp values. It is also useful when feature engineering and you want to add new features that indicate something meaningful.
We can create new binary attributes in Python using scikit-learn with the Binarizer class.

```
# import libraries
from sklearn.preprocessing import Binarizer
import pandas
import numpy

# data set link
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/pima
-indians-diabetes/pima-indians-diabetes.data"
# data parameters
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass',
'pedi', 'age', 'class']

# preparating of dataframe using the data at given link and
defined columns list
dataframe = pandas.read_csv(url, names = names)
array = dataframe.values

# separate array into input and output components
X = array[:, 0:8]
Y = array[:, 8]
binarizer = Binarizer(threshold = 0.0).fit(X)
binaryX = binarizer.transform(X)

# summarize transformed data
numpy.set_printoptions(precision = 3)
print(binaryX[0:5,:])
```

We can see that all values equal or less than 0 are marked 0 and all of those above 0 are marked 1.

3. Standardize Data

Standardization is a useful technique to transform attributes with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.
We can standardize data using scikit-learn with the StandardScaler class.
Code: Python code to Standardize data (0 mean, 1 stdev)

```
# importing libraries
from sklearn.preprocessing import StandardScaler
import pandas
import numpy

# data set link
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/pima
-indians-diabetes/pima-indians-diabetes.data"
# data parameters
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass',
'pedi', 'age', 'class']

# preparating of dataframe using the data at given link and
defined columns list
dataframe = pandas.read_csv(url, names = names)
array = dataframe.values

# separate array into input and output components
X = array[:, 0:8]
Y = array[:, 8]
scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)

# summarize transformed data
numpy.set_printoptions(precision = 3)
print(rescaledX[0:5,:])
```

The values for each attribute now have a mean value of 0 and a standard deviation of 1.

Module 2.3: Cleaning data	Duration 2	Type concept	Category: data

Data cleaning is a crucial step in machine learning to ensure reliable and accurate results. It involves removing duplicate and irrelevant observations, fixing structural errors, managing outliers, and handling missing data. Proper data cleaning improves the efficiency and quality of models. Various tools like Openrefine and Trifacta Wrangler facilitate the data cleaning process.

Data cleaning is one of the important parts of machine learning. It plays a significant part in building a model. It surely isn't the fanciest part of machine learning and at the same time, there aren't any hidden tricks or secrets to uncover. However, the success or failure of a project relies on proper data cleaning. Professional data scientists usually invest a very large portion of their time in this step because of the belief that "Better data beats fancier algorithms".
If we have a well-cleaned dataset, there are chances that we can get achieve good results with simple algorithms also, which can prove very beneficial at times especially in terms of computation when the dataset size is large.

Obviously, different types of data will require different types of cleaning. However, this systematic approach can always serve as a good starting point.

1. **Removal of unwanted observations**

This includes deleting duplicate/ redundant or irrelevant values from your dataset. Duplicate observations most frequently arise during data collection and Irrelevant observations are those that don't actually fit the specific problem that you're trying to solve.
Redundant observations alter the efficiency by a great extent as the data repeats and may add towards the correct side or towards the incorrect side, thereby producing unfaithful results.
Irrelevant observations are any type of data that is of no use to us and can be removed directly.

2. **Fixing Structural errors**

The errors that arise during measurement, transfer of data, or other similar situations are called structural errors. Structural errors include typos in the name of features, the same attribute with a different name, mislabeled classes, i.e. separate classes that should really be the same, or inconsistent capitalization.
For example, the model will treat America and America as different classes or values, though they represent the same value or red, yellow, and red-yellow as

different classes or attributes, though one class can be included in the other two classes. So, these are some structural errors that make our model inefficient and give poor quality results.

3. **Managing Unwanted outliers**

Outliers can cause problems with certain types of models. For example, linear regression models are less robust to outliers than decision tree models. Generally, we should not remove outliers until we have a legitimate reason to remove them. Sometimes, removing them improves performance, sometimes not. So, one must have a good reason to remove the outlier, such as suspicious measurements that are unlikely to be part of real data.

4. **Handling missing data**

Missing data is a deceptively tricky issue in machine learning. We cannot just ignore or remove the missing observation. They must be handled carefully as they can be an indication of something important. The two most common ways to deal with missing data are:
Dropping observations with missing values.
The fact that the value was missing may be informative in itself.
Plus, in the real world, you often need to make predictions on new data even if some of the features are missing!
Imputing the missing values from past observations.
Again, “missingness” is almost always informative in itself, and you should tell your algorithm if a value was missing.
Even if you build a model to impute your values, you’re not adding any real information. You’re just reinforcing the patterns already provided by other features.
Missing data is like missing a puzzle piece. If you drop it, that’s like pretending the puzzle slot isn’t there. If you impute it, that’s like trying to squeeze in a piece from somewhere else in the puzzle.
So, missing data is always an informative and an indication of something important. And we must be aware of our algorithm of missing data by flagging it. By using this technique of flagging and filling, you are essentially allowing the algorithm to estimate the optimal constant for missingness, instead of just filling it in with the mean.

Some data cleansing tools

Openrefine
Trifacta Wrangler
TIBCO Clarity
Cloudingo
IBM Infosphere Quality Stage
Conclusion:
So, we have discussed four different steps in data cleaning to make the data more reliable and to produce good results. After properly completing the Data Cleaning steps, we’ll have a robust dataset that avoids many of the most common pitfalls. This step should not be rushed as it proves very beneficial in