KURUNJI VENKATRAMANA GOWDA POLYTECHNIC SULLIA-574327

5TH SEMESTER

AI/ML WEEK-5

Week: -05

Importance of data pre-processing

Data preprocessing is a required first step before any machine learning machinery can be applied, because the algorithms learn from the data and the learning outcome for problem solving heavily depends on the proper data needed to solve a particular problem – which are called features.These features are key for learning and understanding, and therefore, machine learning is often considered as feature engineering. Data preprocessing, however, inflicts a heavy danger; for example, during the preprocessing, data can be inadvertently modified; for example, "interesting" data may be removed. Consequently, for discovery purposes, it would be wise to have a look at the original raw data first and maybe do a comparison between nonprocessed and preprocessed data. Data integration is a hot topic generally and in health informatics specifically, and solutions can bridge the gap between clinical and biomedical research (bench vs. bed). This is becoming even more important due to the increasing amounts of heterogeneous, complex patient-related data sets, resulting from various sources including picture archiving and communication systems (PACS) and radiological information systems (RIS), hospital information systems (HIS), laboratory information systems (LIS), physiological and clinical data repositories, and all sorts of *omics data from laboratories, using samples from biobanks. The latter include large collections of DNA sequence data, proteomic and metabolic data, resulting from sophisticated highthroughput analytic technologies. Along with classical patient records containing large amounts of unstructured information (N.B., avoid the term unstructured data) and semistructured information, integration efforts incorporate enormous problems but at the same time offers new possibilities for translational research. Data cleaning What is Data Cleaning? Data cleaning is the process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted. But, as we mentioned above, it isn't as simple as organizing some rows or erasing information to make space for new data.

Data cleaning

Data cleaning is a lot of muscle work. There's a reason data cleaning is the most important step if you want to create a data-culture, let alone make airtight predictions. It involves:

O Fixing spelling and syntax errors

O Standardizing data sets

O Correcting mistakes such as empty fields

O identifying duplicate data points

Assess Data Quality

Measuring data quality: The data quality assessment

The term "data quality" refers to the suitability of data to serve its intended purpose. So, measuring data quality involves performing data quality assessments to determine the degree to which your data adequately supports the business needs of the company.

A data quality assessment is done by measuring particular features of the data to see if they meet defined standards. Each such feature is called a "data quality dimension," and is rated according to a relevant metric that provides an objective assessment of quality.

The Industry hasn't yet settled on a standard set of data quality dimensions, but the following is a representative group:

Four metrics of data quality

Let's take a brief look at each of these and at the metrics used in assessing them.

1. Completeness

Completeness relates to whether all required information is present in the dataset. For example, if the customer information in a database is required to include both first and last names, any record in which the first name or last name field is not populated is marked as incomplete. The metric used in assessing this dimension is the percentage of records that are complete.

2. Validity

Data is characterized as valid if it matches the rules specified for it. Those rules typically include specifications such as format (number of digits, etc.), allowable types (integer, floating-point, string, etc.), and range (minimum and maximum values). For example, a telephone number field that contains the string '1809 Oak Street' is not valid. The metric for this dimension is the percentage of records in which all values are valid.

3. Timeliness

Timeliness relates to whether the information is up to date for the intended use. In other words, is the correct information available when needed?

For example, if a customer has notified the company of an address change, but the new address is not in the database at the time billing statements are processed, that entry fails the timeliness test. The metric used to measure timeliness is the time difference between when data is needed and when it is available.

4. Consistency

A data item is consistent if all representations of that item across data stores match.

If, for example, a birth date is entered in one system using the U.S. format (mm/dd/yyyy), but it is imported into another system where the date is entered using the European standard (dd/mm/yyyy), that data lacks consistency.

Data anomalies

Anomaly Detection is the technique of identifying rare events or observations which can raise suspicions by being statistically different from the rest of the observations. Such "anomalous" behaviour typically translates to some kind of a problem like a credit card fraud, failing machine in a server, a cyber attack, etc.

An anomaly can be broadly categorized into three categories -

1. Point Anomaly: A tuple in a dataset is said to be a Point Anomaly if it is far off from the rest of the data.

2. Contextual Anomaly: An observation is a Contextual Anomaly if it is an

anomaly because of the context of the observation.

3. Collective Anomaly: A set of data instances help in finding an anomaly. Anomaly detection can be done using the concepts of Machine Learning. It can be done in the following ways-

1. Supervised Anomaly Detection: This method requires a labeled dataset containing both normal and anomalous samples to construct a predictive model to classify future data points. The most commonly used algorithms for this purpose are supervised Neural Networks, Support Vector Machine learning, KNearest Neighbors Classifier, etc.

2. Unsupervised Anomaly Detection: This method does require any training data and instead assumes two things about the data ie Only a small percentage of data is anomalous and Any anomaly is statistically different from the normal samples. Based on the above assumptions, the data is then clustered using a similarity measure and the data points which are far off from the cluster are considered to be anomalies.

We now demonstrate the process of anomaly detection on a synthetic dataset using the K-Nearest Neighbors algorithm which is included in the pyod module.

Step 1: Importing the required libraries

Python 3

```
Import numpy as np

From scipy import stats

Import matplotlib.pyplot as plt

Import matplotlib.font_manager From pyod.models.knn import KNN

From pyod.utils.data import generate_data, get_outliers_inliers
```

Step 2: Creating the synthetic data

Python 3

```
# generating a random dataset with two features

X_train, y_train = generate_data(n_train = 300, train_only = True,

N_features = 2)

# Setting the percentage of outliers

Outlier_fraction = 0.1

# Storing the outliers and inliners in different numpy arrays
```

X_outliers, X_inliers = get_outliers_inliers(X_train, y_train)

N_inliers = len(X_inliers)

N_outliers = len(X_outliers)

#Separating the two features

F1=X_train[:, [0]].reshape(-1, 1) F2 =X_train[:,[1]].reshape(-1, 1)

Step 3: Visualising the data

Python 3

Visualising the dataset

create a meshgrid

Xx, yy = np.meshgrid(np.linspace(-10, 10, 200) Np.linspace(-10, 10,200))

scatter plot

Plt.scatter(f1, f2)

Plt.xlabel('Feature 1')

Polysyllable('Feature 2')

Step 4: Training and evaluating the model

Python 3

#Training the classifier

Clf KNN(contamination outlier_fraction)

Cif.fit(X_train, y_train)

You can print this to see all the prediction scores

Scores_pred = clf.decision_function(X_train)*-1

Y_pred = clf.predict(X_train)

N_errors=(y_pred != y_train).sum()

Counting the number of errors

Print("The number of prediction errors are ' + str(n_errors))

Step 5: Visualising the predictions

Python 3

#threshold value to consider a

#datapoint inlier or outlier

Threshold = stats.scoreatpercentile(scores_pred, 100 outlier_fraction)

decision function calculates the raw

anomaly score for every point

Z= clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) ● -1

```
Z = Z.reshape(xx.shape)

# fill blue colormap from minimum anomaly

# score to threshold value

Subplot = plt.subplot(1, 2, 1)

=

163/442
Subplot.contourf(xx, yy, Z, levels = np.linspace(Z.min(), Threshold, 10), cmap = plt.cm.Blues_r)
# draw red contour line where anomaly
# score is equal to threshold
A = subplot.contour(xx, yy, Z, levels =[threshold],

Linewidths = 2, colors='red')

# fill orange contour lines where range of anomaly

# score is from threshold to maximum anomaly score

Subplot.contourf(xx, yy, Z, levels =[threshold, Z.max()], colors='orange')

#scatter plot of inliers with white dots

B = subplot.scatter(X_train[:-n_outliers, 0], X_train[:-n_outliers, 1],
C='white', s 20, edgecolor='k') =

#scatter plot of outliers with black dots
```

C = subplot.scatter(X_train[-n_outliers:, 0], X_train[-n_outliers:, 1),

C='black', s = 20, edgecolor='k') Subplot.axis('tight')

Subplot.legend(

[a.collections[0], b, c),

['learned decision function', 'true inliers', 'true outliers'],

Prop = matplotlib.font_manage(size = 10),

Loc='lower right')

Subplot.set_title('K-Nearest Neighbours')

Subplot.set_xlim((-10, 10))

Subplot.set_ylim((-10, 10))

Plt.show()

Detect missing values with pandas dataframe functions: .info() and .isa()

Pandas.DataFrame.info()

buf=None,

max_cols=None,

DataFrame.info(verbose=None,

memory_usage=None, show_counts=None, null_counts=None)[source]

Print a concise summary of a DataFrame. This method prints information about a DataFrame including the index

dtype and columns, non-null values and memory usage.

Parameters data: DataFrame

DataFrame to print information about.

Verbose: bool, optional

Whether to print the full summary. By default, the setting in pandas.options.display.max_info_columns is followed.

Buf:writable buffer, defaults to sys.stdout

Where to send the output. By default, the output is printed to sys.stdout. Pass a writable buffer if you need to further process the output. Max_cols: int, optional When to switch from the verbose to the truncated output. If the DataFrame has more than max_cols columns, the truncated output is used. By default, the setting in pandas.options.display.max_info_columns is used.

Examples:

>>>Int_values = [1, 2, 3, 4, 5]

>>>Text_values = ['alpha', 'beta', 'gamma', 'delta', 'epsilon'] >>>Float_values = [0.0, 0.25, 0.5, 0.75, 1.0]

[11:31 pm, 25/11/2022] +91 73488 20642: >>>df = pd.DataFrame({"int_col": int_values, "text_col": text_values,

"float_col": float_values})

>>>df

Int_col

text_col

float_col

0 alpha

1 0.00

2 beta

3 0.25

4 gamma

5 0.50

5

epsilon

Prints information of all columns:

>>>df.info(verbose=True)

<class 'pandas.core.frame.DataFrame">

Rangelndex: 5 entries, 0 to 4 Data columns (total 3 columns):

#

Column

Non-Null Count Dtype

0

int_col

5 non-null

int64

1

text_col 5 non-null

object

2

flatcar

5 non-null

float64

Dtypes: float64(1), int64(1), object(1)
Memory usage: 248.0+ bytes

Prints a summary of columns count and its dtypes but not per column information:

>>>df.info(verbose=False)

<class 'pandas.core.frame.DataFrame">

RangeIndex: 5 entries, 0 to 4

Columns: 3 entries, int_col to float_col Dtypes: float64(1), int64(1), object(1)

Memory usage: 248.0+ bytes

Pipe output of DataFrame.info to buffer instead of sys.stdout, get buffer content and writes to a text file:

>>>Import io

>>>Buffer = io.String10()

>>>df.info(buf-buffer)

>>>S = buffer.getvalue()

>>>With open("df_info.txt", "w",

Encoding="utf-8") as f

f.write(s)

260

The memory_usage parameter allows deep introspection mode, specially useful for big DataFrames and fine-tune memory optimization:

>>>Random_strings_array = np.random.choice(['a', 'b', 'c'], 10 ^ ** 6)

>>>df = pd.DataFrame({

'column_1': np.random.choice(['a', ^ prime b ^ prime , 'c'], 10 ^ ** 6) ,

'column_2': np.random.choice ([^ prime a ^ prime ,^ prime b^ prime ,^ prime c^ prime] , 10 ^ ** 6) ,

'column_3': np.random.choice([^ prime a ^ prime ,^ prime b^ prime ,^ prime c^ prime] 10 ^ ** 6^ prime

... })

>>>df.info()

<class 'pandas.core.frame.DataFrame">

RangeIndex: 1000000 entries, 0 to 999999 Data

columns (total 3 columns):

#

Column

Non-Null Count

Dtype

0

column_1

1000000 non-null

1 2 column_2 column_3

1000000 non-null

object

1000000 non-null

object

object

3 dtypes: object(3) memory usage: 22.9+ MB

>>>df.info(memory_usage='deep') <class 'pandas.core.frame.DataFrame">
RangeIndex: 1000000 entries, 0 to 999999 Data

columns (total 3 columns):

#

Non-Null Count

Column

Dtype

0

1

2

column_1

1000000 non-null

object

column_2

column_3

1000000 non-null

object 1000000 non-null object dtypes: object(3)

memory usage: 165.9 MB

pandas.DataFrame.isna()

DataFrame.isna()[source] Detect

missing values.

Return a boolean same-sized object indicating if the values are NA. NA values, such as None or numpy.NaN, gets mapped to True values. Everything else gets mapped to False values. Characters such as empty strings" or numpy.inf are not considered NA values (unless you set pandas.options.mode.use_inf_as_na = True).

Returns: DataFrame

Mask of bool values for each element in DataFrame that indicates whether an element is an NA value.

Examples:

NaN dtype: float64

>>>Ser.isna()

0

False

1

2

False

True dtype: bool

Diagnose type of missing values with visual and statistical methods (eg.

Chi-squared test of independence) Missing values and outliers are frequently encountered during the data collection phase of observational or experimental studies conducted in all fields of natural and social sciences.

Missing values can arise from information loss as well as dropouts and nonresponses of the study participants. The presence of missing values leads to a smaller sample size than intended and eventually compromises the reliability of the study results. It can also produce biased results when inferences about a population are drawn based on such a sample, undermining the reliability of the data. As a part of the pretreatment process, missing data are either ignored in favor of simplicity or replaced with substituted values estimated with a statistical method. In general, the analysis of missing values involves the consideration of efficiency, handling of missing data and the resulting complexity in analysis, and the bias between missing and observed values.

Chi-Square Test of Independence determines whether there is an association between categorical variables (i.e., whether the variables are independent or related). It is a nonparametric test.

This test is also known as:

• Chi-Square Test of Association

Test Statistic The test statistic for the Chi-Square Test of Independence is denoted X ^ 2 and is computed as:

R

C

x ^ 2 = sum

overline I = 1

j = 1

eij

Where

O ij is the observed cell count in the throw and jth column of the table

eij is the expected cell count in the I ^ (th) row and Jth column of the table,

computed as

e-row i total coli total

grand total

The quantity (Oij - eij) is sometimes referred to as the residual of cell

1 denoted The calculated X2 value is then compared to the critical value from the X^2 distribution table with degrees of freedom df = (R-1 -1) and chosen confidence level. If the calculated X2 value > critical X^2 value, then we reject the null hypothesis.

Approaches to deal with missing values What

is a Missing Value?

Missing data is defined as the values or data that is not stored (or not

present) for some variable/s in the given dataset. Below is a sample of the missing data from the Titanic dataset. You can see the columns 'Age' and 'Cabin' have some missing values.

+ Keep the missing value as is

+ Remove data objects with missing values

+ Remove the attributes with missing values +Estimate and impute missing values

Keep the missing value as is

The problem of missing value is quite common in many real-life datasets. Missing value can bias the results of the machine learning models and/or reduce the accuracy of the model. This article describes what is missing data, how it is represented, and the different reasons for the missing data. Along with the different categories of missing data, it also details out different ways of handling missing values with examples.

Outliers are those data points that are significantly different from the rest of the dataset. They are often abnormal observations that skew the data distribution, and arise due to inconsistent data entry, or erroneous observations.

How to Detect Outliers Using Standard Deviation When the data, or certain features in the dataset, follow a normal distribution.

you can use the standard deviation of the data, or the equivalent z-score to

detect outliers. In statistics, standard deviation measures the spread of data around the mean, and in essence, it captures how far away from the mean the data points are. For data that is normally distributed, around 68.2% of the data will lie within one standard deviation from the mean. Close to 95.4% and 99.7% of the data lie within two and three standard deviations from the mean, respectively.

Let's denote the standard deviation of the distribution by σ, and the mean by μ.

One approach to outlier detection is to set the lower limit to three standard deviations below the mean (u-3*0), and the upper limit to three standard deviations above the mean (u + 3°a). Any data point that falls outside this range is detected as an outlier. As 99.7% of the data typically lies within three standard deviations, the number of outliers will be close to 0.3% of the size of the dataset.

Code for Outlier Detection Using Standard Deviation

Now, let's create a normally-distributed dataset of student scores, and perform outlier detection on it.

As a first step, we'll import the necessary modules.

```
import numpy as np

import pandas as pd

import seaborn as sns
```

Next, let's define the function generate_scores () that returns a normally- distributed dataset of student scores containing 200 records. We'll make a call to

the function, and store the returned array in the variable scores_data.

```
def generate_scores (mean 60 std_dev-12.num_samples-200):

np.random.seed
```

Output

```
score 11.854434

dtype: float64
```

As discussed earlier, set the lower limit (lower limit) to be three standard deviations below the mean, and the upper limit (upper limit) to be three standard deviations above the mean.

```
lower_limit = df_scores mean) - 3"df_scores.std()

upper_limit = df_scores mean() + 3*df_scores std()

print(lower_limit) print(upper limit)
```

#Output

25.530716709142666

96.47928329085734

Now that you've defined the lower and upper limits, you may filter the dataframe of scores to only retain the data points in the

interval [lower limit, upper limit), as shown below. df_scores_filtered-df_scores [[df_scores['score']>lower_limit)&(df_sco

res 'score']<upper_limit)] print(df_scores_filtered)

Output

score

0

1

2

3

4

75.0

5…

How to Detect Outliers Using the Z-Score

Now let's explore the concept of the z-score. For a normal distribution with mean

μ and standard deviation a, the z-score for a value x in the dataset is given by:

z = (x -μ)/o

From the above equation, we have the following:

When x = p, the value of z-score is 0.

. When x =μ±1,μ±2, or u± 3, the z-score is ± 1, ±2, or ± 3, respectively.

Notice how this technique is equivalent to the scores based on standard deviation we had earlier. Under this transformation, all data points that lie below the lower limit, p-3'a, now map to points that are less than -3 on the z-score scale.

Similarly, all points that lie above the upper limit, u + 3*o map to a value above 3

on the z-score scale. So [lower limit, upper limit] b…

How to Detect Outliers Using the Interquartile Range (IQR)

In statistics, interquartile range or IQR is a quantity that measures the difference between the first and the third quartiles in a given dataset..

If q25 is the first quartile, it means 25% of the points in the dataset have values lessthan q25.

. The third quartile is also called the three-fourth, or the 75% quartile.

- If q75 is the three-fourth quartile, 75% of the points have values less than q75.
- Using the above notations, IQR = q75 - q25.

Code for Outlier Detection Using Interquartile Range (IQR)

You can use the box plot, or the box and whisker plot, to explore the dataset and visualize the presence of outliers. The points that lie beyond the whiskers are detected as outliers.

You can generate box plots in Seaborn using the boxplot function. sns boxplot(data=scores_data) set(title="Box Plot of Scores")

Box Plot of Scores

As seen in the output, this method labels eight points as outliers, and the filtered dataframe is 192 records long.

You don't always have to call the describe method to identify the quartiles. You may instead use the percentile () function in NumPy. It takes in two arguments, a: an array or a dataframe and q: a list of quartiles. The code cell below shows how you can calculate the first and the third quartiles using the percentile function.

q25.q75 np percentile (a= df_scores.q=[25,75])

IQR = 975 925 print(IQR)

Output 13.0

How to Detect Outliers Using Percentile

In the previous section, we explored the concept of interquartile range, and its application to outlier detection. You can think of percentile as an extension to the interquartile range.

As discussed earlier, the interquartile range works by dropping all points that are

outside the range (q25 1.5 IQR, 975 + 1.5*1QR] as outliers. But removing outliers this way may not be the most optimal choice when your observations have a wide distribution. And you may be discarding more points- than you actually should-as outliers. Depending on the domain, you may want to widen the range of permissible values to estimate the outliers better. Next, let's revisit the scores dataset, and use percentile to detect outliers.

Code for Outlier Detection Using Percentile

Let's define a custom range that accommodates all data points that lie anywhere between 0.5 and 99.5 percentile of the dataset. To do this, set q = [0.5, 99.5] in the percentile function, as shown below.

```
lower_limit, upper_limit = np.percentile a=df_scores.q=[0.5,99.5])
```

```
print upper limit)
```

```
print(lower_limit)
```

Data Integration

So far, we've made sure to remove the impurities in data and make it clean. Now, the next step is to combine data from different sources to get a unified structure with more meaningful and valuable information. This is mostly used if the data is segregated into different sources. To make it simple, let's assume we have data in CSV format in different places, all talking about the same scenario. Say we have some data about an

employee in a database. We can't expect all the data about the employee to reside in the same table. It's possible that the employee's personal data will be located in one table, the employee's project history will be in a second table, the employee's time-in and time-out details will be in another table, and so on. So, if we want to do some analysis about the employee, we need to get all the employee data in one common place. This process of bringing data together in one place is called data integration. To do data integration, we can merge multiple pandas DataFrames using the merge function.

Let's solve an exercise based on data integration to get a clear understanding of it.

Exercise 5: Integrating Data

In this exercise, we'll merge the details of students from two datasets, namely student.csv and marks.csv. The student dataset contains columns such as Age, Gender, Grade, and Employed. The marks.csv dataset contains columns such as Mark and City. The Student_id column is common between the two datasets. Follow these steps to complete this exercise:

Open a Jupiter notebook and add a new cell. Write the following code to import pandas and load the student.csv and marks.csv datasets into the df1 and df2 pandas DataFrames:

```
import pandas as pd
```

```
dataset="https://github.com/TrainingByPackt/Data-Science-with-
Python/blob/master/Chapter01/Data/student.csv"
```

dataset2 = "https://github.com/TrainingByPack/Data-Science-with-Python/blob/master/Chapter01/Data/mark.csv"

deli = pd.read_csv (dataset1, header = 0)

df2 = pd.read_csv(dataset2, header = 0)

• To print the first five rows of the first DataFrame, add the following code:

dfl.head()

To be able to actually use all the data you get from different sources, you need to connect them, and a no-code integration platform like ZigiOps can help immensely with that. The right integration platform allows each team to correlate data from different sources, which has several benefits:

Faster error detection & resolution

- Data is automatically enriched, making it more useful
- Data becomes more accessible for all business units
- The decision-making process is improved, as it relies on a more comprehensive overview of the

available information.

2. Each Team is Using Different Systems

In most organizations each team is using systems that they're familiar with, and are hesitant to

use the apps of peers, even if they provide highly valuable info…

For this reason, we've designed ZigiOps with high availability (HA) in mind: our HA feature helps our clients maintain system availability and uptime at all times, regardless of the volume of integrated data, and of potential server failures or system downtimes. In addition to that, each of our system integration templates is infinitely scalable. They are fully customizable and can adapt to any scenario.

5. You Need Bi-Directional Integrations Bi-directional integrations are a must since they transfer and update data between the source and target systems. However they aren't always easy to set up - and you would definitely need

an integration platform for this.

For example, if you're only using your apps' native integration features, once you need to set …

Depending on the service, it can decrease expenditures.

Manual integration is slow, inefficient, and expensive both in people power and chargebacks for data errors.

2. Plug and Play Data Integration Approach:

Plug and play integration is like an app. Either a stand-alone download that you can manually configure or an extension of an application you already use into other common applications. You cannot change parameters or mold it to fit your exact needs, it is not flexible. In general, these solutions are one-size-fits-all and are low end in their functionality.

Pros

It is the best-tailored solution if you have a small to medium size business that has few complexities in their sales and/or supply chain and low sales volume in general.

Cons

Plug-and-p…

The Do It Yourself solution involves hardcoded integrations. Every change or new integration would have to be manually hardcoded for each project every time. Making this solution structurally rigid.

Pros

It can decrease expenditures, as it might be cheaper to seek and hire a developer.

This solution is ideal for larger enterprise- level businesses that can hire and maintain a multi-person IT team, as hardcoded integrations require maintenance and extensive documentation.

Cons

This approach is not always effective because the information is being built in a static point of time and in a programming language that cannot evolve with the market without manual intervention.

The code is not always documented. If the integrations are being managed by a single… It

is effective for small businesses with low sales and transaction volumes.

Manual file downloads and uploads limit businesses from scaling effectively as business rules cannot be applied directly to the data as it moves. For instance, there is a need for manual tasks like field matching or data clean-up post-upload to happen.

5. Data Integration as a Service (iPaaS) Approach:

Like VL OMNI's integration platform, iPaaS solutions have the unique ability to modify and clone real-time integrations as your business evolves without having to hit 'pause' on normal operations. If a new channel, trading partner, or application is added to the integration system, these integrations can also be duplicated from existing workflows, meaning less time and money to get…

The main part of objects is their attributes. Attributes define what the properties of a certain object are.

While working with objects, there may be many situations where we need to add a new attribute to an object in the middle of the program.

Python provides a function state() that can easily set the new attribute of an object. This function can even replace the value of the attribute.

It is a function with the help of which we can assign the value of attributes of the object. This method will provide us with many ways to allocate values to variables by certain constructors and object functions. By using this function, we will also be able to have other

substitutive ways to assign value.

Now, let's discuss the structure of this statt function. The st…

print("New Student Name:", new student.name

print ("New Student Roll #:", new student_roll_no print ("New Student Capai", new student.cgpa

Output:

As you can see from the above example, it was quite easy to set the object's attributes that we created. Now, let's discuss a different scenario.

Let's suppose we have a new object and want to set an attribute missing from the class. In some cases, there are no attributes, or all attributes are not created in a class. When this

happens, we assign a new attribute and can set a value for it. But to make it happen, the object should implement the_dict__(method. Let's go through an example and try to assign values to an attribute that doesn't exist

We will use the above example and try to assign a new attribute, … print("New

Student Degree:", new student.degree

Output:

As you can see from the above example, this function can also create new attributes that don't exist and assign values to them.

Data redution

Data reduction is the process of reducing the amount of capacity required to store data. Data reduction can increase storage efficiency and reduce costs. Storage vendors will often describe storage capacity

in terms of raw capacity and effective capacity, which refers to data after the reduction.

Data reduction can be achieved several ways. The main types are data deduplication, compression and single-instance storage. Data deduplication, also known as data dedupe, eliminates redundant segments of data on storage systems. It only stores redundant segments once and uses that one copy whenever a request is made to access that piece of data. Data dedupe is more granular than single-instance storage. Single-instance storage finds files such as email attachments sent to multiple people and only stores one copy of that file. As with dedupe, single-instance storage replaces duplicates with pointers to the one saved copy.

Distinction between data reduction & data redundancy

Data mining is applied to the selected data in a large amount database. When data analysis and mining is done on a huge amount of data, then it takes a very long time to process, making it impractical and infeasible.

Data reduction techniques ensure the integrity of data while reducing the data. Data reduction is a process that reduces the volume of original data and represents it in a much smaller volume. Data

reduction techniques are used to obtain a reduced representation of the dataset that is much smaller in volume by maintaining the integrity of the original data. By reducing the data, the efficiency of the data mining process is improved, which produces the same analytical results.Data reduction does not affect the result obtained from data mining. …

Objectives & ethods

- Objective methods do not rely on written or verbal responses from the individual under study but instead record phenomena from which the dimensions of physical activity can be inferred.

- The phenomena can be physiological, kinematic, biochemical, or environmental in nature.

- Technology is often used to capture these variables directly.

- Direct observation can also capture some of these phenomena. This method sits on the boundary between subjective and objective methods, since although observers make subjective records; they are typically independent from the person under study. However, when the observer is a proxy-reporter (e.g. teacher or parent), observations may not be fully independent from the person under study.

- The initial raw measurement by the Objective methods are robust to issues relating to respondent bias, such as recall errors and social desirability bias.

- As a result, these methods can provide more accurate estimates of diet, physical activity and anthropometry with a less complex error structure; they are often used as criterion methods to demonstrate the validity of subjective methods, or other objective methods.

- Objective methods can be costly, intrusive, plus burdensome in terms of time and effort for the participant and researcher, sometimes rendering them more difficult to apply to large epidemiological settings.

- Objective methods may require specialised training.

- Participant's consent is essential, as always. Depending on the mode of assemente

Numerosity data reduction

Data reduction process reduces the size of data and makes it suitable and feasible for analysis. In the reduction process, integrity of the data must be preserved and data volume is reduced. There are many techniques that can be used for data reduction. Numerosity reduction is one of them.

Numerosity Reduction is a data reduction technique which replaces the original data by smaller form of data representation. There are two techniques for numerosity reduction- Parametric and NonParametric methods.

Data tranceform

Data transformation can be simple or complex based on the required changes to the data between the source (initial) data and the target (final) data. Data transformation is typically performed via a mixture of manual and automated steps.[2] Tools and technologies used for data transformation can vary widely based on the format, structure, complexity, and volume of the data being transformed.

A master data recast is another form of data transformation where the entire database of data values is transformed or recast without extracting the data from the database. All data in a well designed database is directly or indirectly related to a limited set of master database tables by a network of foreign key constraints. Each foreign key constraint is dependent upon…ation **Need for data transfortion**

These days, understanding the steps involved in the data transformation process is important, even if data transformation is not a primary part of your job.

Because we live in a world where data is collected, stored, and analyzed in so many different formats, being able to perform the basic steps required to transform data from one form to another is a common requirement for many of us.

This article explains what those steps are by outlining a typical data transformation process.ation

Normalization

includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

1. First normal for Eliminate repeating groups in individual tables.

ceate a separate table for each set of related data.

Identify each set of related data with a primary

2. Second normal form

Create separate tables for sets of values that apply to multiple records.

Relate these tables with a foreign

3. Third normal form

Eliminate fields that do not depend on the key.

Values in a record that are not part of that record's key do not belong in the table. In general, anytime the contents of a group of fields may apply to more than a single record in the table, consider placing those fields in a separate table

Other normalization forms

Fourth normal form, also called Boyce Codd Normal Form (BCNF), and fifth normal form do exist, but are rarely considered in practical design. Disregarding these rules may result in less than perfect database design, but should not affect functionality.

Standardization

Standardization is the process of developing, promoting and possibly mandating standards-based and compatible technologies and processes within a given industry.

Standards for technologies can mandate the quality and consistency of technologies and ensure their compatibility, interoperability and safety. Standards organizations such as ANSI (American National Standards Institute), IEEE (Institute of Electrical and Electronics Engineers) and IETF (Internet Engineering Task Force) exist to promote standardization and endorse official standards (also known as de jure standards) for given applications.

A lack of standardization often manifests in large numbers of incompatible proprietary formats for a given technology and for technologies that must interoperate.… **Binary coding**

which we want to transform our data from numerical representation to categorical representation, or vice versa. To do these transformations, we will have to use one of three tools: binary coding, ranking transformation, and discretization.

As the following figure shows, to switch from Categories to Numbers, we either have to use Binary Coding or Ranking Transformation, and to switch from numbers to categories, we need to use Discretization:

Figure 14.3 – Direction of application for binary coding, ranking transformation, and discretization

One question that the preceding figure might bring to mind is, how do we know which one we choose when we want to move from categories to numbers: binary coding or ranking transformation? The answer is simple.

If the

Binary coding desing in the figure & exaple

USASCII code chart

b_4	b_3	b_2	b_1	Row ↓ / Column →	000	001	010	011	100	101	110	111
					0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

TRANSFORMATIONS

Ranking transformation

The Rank transformation selects the top or bottom range of data. Use the Rank transformation to return the largest or smallest numeric values in a group. You can also use the Rank transformation to return strings at the top or bottom of the mapping sort order.

For example, you can use a Rank transformation to select the top 10 customers by region. Or, you might identify the three departments with the lowest expenses in salaries and overhead.

The Rank transformation differs from the transformation functions MAX and MIN because the Rank transformation returns a group of values, not just one value. While the SQL language provides many functions designed to handle groups of data, identifying top or bottom strata within a s…

Discretization

Discretization is also related to discrete mathematics, and is an important component of granular computing. In this context, discretization may also refer to modification of variable or category

granularity, as when multiple discrete variables are aggregated or multiple discrete categories fused.

Whenever continuous data is discretized, there is always some amount of discretization error. The goal is to reduce the amount to a level considered negligible for the modeling purposes at hand.

The terms discretization and quantization often have the same denotation but not always identical connotations. (Specifically, the two terms share a semantic field.) The same is true of discretization error and quantization error.

Mathematical methods relating to discretization include the Euler–Maruyama method and the zeroorder hold.

Data transformation with ranking transformation discretization

The data are transformed in ways that are ideal for mining the data. The data transformation involves steps that are:

1. Smoothing:

It is a process that is used to remove noise from the dataset using some algorithms It allows for highlighting important features present in the dataset. It helps in predicting the patterns. When collecting data, it can be manipulated to eliminate or reduce any variance or any other noise form.

The concept behind data smoothing is that it will be able to identify simple changes to help predict different trends and patterns. This serves as a help to analysts or traders who need to look at a lot of data which can often be difficult to digest for finding patterns that they wouldn't see otherwise.

2. Aggregation:

Data collection or aggregation is the method of storing and presenting data in a summary format. The data may be obtained from multiple data sources to integrate these data sources into a data analysis description. This is a crucial step since the accuracy of data analysis insights is highly dependent on the quantity and quality of the data used. Gathering accurate data of high quality and a large enough quantity is necessary to produce relevant results.

The collection of data is useful for everything from decisions concerning financing or business strategy of the product, pricing, operations, and marketing strategies.

For example, Sales, data may be aggregated to compute monthly& annual total amounts.

3. Discretization:

It is a process of transforming continuous data into set of small intervals. Most Data Mining activities in the real world require continuous attributes. Yet many of the existing data mining frameworks are unable to handle these attributes.

Also, even if a data mining task can manage a continuous attribute, it can significantly improve its efficiency by replacing a constant quality attribute with its discrete values.

For example, (1-10, 11-20) (age:- young, middle age, senior).

4. Attribute Construction:

Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.

5. Generalization:

It converts low-level data attributes to high-level data attributes using concept hierarchy. For Example Age initially in N…