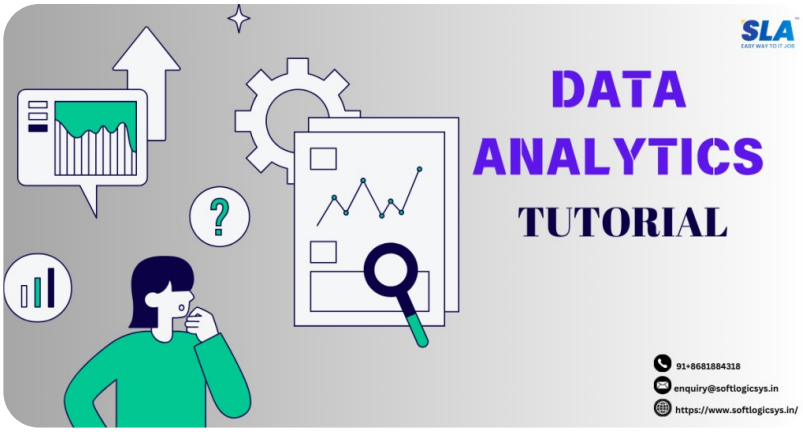

Share on your Social Media

Data Analytics Tutorial

Published On: August 8, 2024

Data Analytics Tutorial

Organizations can use data analytics to mine raw data for valuable insights and trends. Numerous data visualization concepts are covered in this data analytics tutorial.

Download Data Analytics Tutorial PDF

Introduction to Data Analytics Tutorial

Data analytics analyzes, cleans, predicts, and manipulates data to extract insights, provide recommendations, and facilitate decision-making. This Data Analytics Tutorial will cover the following:

- Overview of Data Analytics
- Data Preprocessing
- Exploratory Data Analysis
- Time Series Data Analysis

Featured Articles

Want to know more about becoming an expert in IT?

Click Here to Get Started

100% Placement Assurance

AUTHORISED CERTIFICATION PARTNER IBM

Quick Enquiry

Related Courses at SLA

- Data Analytics Online Training
- Data Analytics Training in OMR
- Data Analytics Training in Chennai

Related Posts

- Data Analysis Tools
- Data Analysis Libraries

Overview of Data Analytics

Large-scale, unstructured data collection from various sources is called data analysis, and it involves the following steps to turn the data into information:

- Data requirements specification
- Data Collection
- Data Processing
- Data Cleaning
- Data Analysis
- Communication

Data Analytics Interview Questions

Importance of Data Analytics

Data analytics is important for the performance optimization of businesses. Additionally, data analytics can help a company analyze customer trends and fulfillment, which can lead to the discovery of new and improved products and services.

It can also help an organization make better business decisions. Integrating it into the company plan shows how companies can minimize expenses by identifying more productive ways to operate.

Applications of Data Analytics

The following are applications for data analytics:

- Improved decision-making
- Identifying the possible risks
- Boost work efficiency
- Providing useful products
- Monitor shifts in customer behavior

Data Analytics Project Ideas

Published On: November 2, 2024

Data Analytics Project Ideas are a great way for students to get hands-on experience and...

MERN Stack Tutorial for Web Development Aspirants

Published On: October 14, 2024

MERN Stack Tutorial for Web Development Aspirants There is a growing need for competent MERN...

Tableau Developer Salary in Chennai

Published On: October 12, 2024

Introduction A Tableau Developer designs, develops, and maintains dashboards and visualizations using Tableau software. Key...

VMware Tutorial for

- Effective marketing
- Better customer service
- Efficient operations.

Data Preprocessing

Any project involving data analysis or machine learning must start with data preparation. It entails a range of activities designed to convert unprocessed data into an organized and functional format.

Well-prepared data guarantees more dependable and accurate analysis outcomes, which improves decision-making and increases the efficacy of prediction models.

Download Data Analytics Syllabus PDF

Data Formating

Here are a few different methods for formatting numbers in Pandas in Python. Therefore, the following describes a few commonly used techniques and approaches for data formatting in Pandas.

The column values should be rounded to two decimal places:

In this example, a dictionary holding monthly expenses is converted into a Pandas DataFrame called “data frame” using the Pandas library. The original data frame is then printed. following the configuration of the Pandas option to show float data to two decimal places.

```
import pandas as pd

data = {'Month': ['January', 'February', 'March', 'April'],

    'Expense': [21525220.653, 31125840.875, 23135428.768,
56245263.942]}

dataframe = pd.DataFrame(data, columns=['Month', 'Expense'])

print("Given Dataframe :\n", dataframe)
```

```
pd.options.display.float_format = '{:.2f}'.format

print('\nResult :\n', dataframe)
```

Formatting Pandas DataFrames with Decimal Precision and Commas

The code in the example below creates a DataFrame called "*products_dataframe*" with the names of the products and their corresponding prices using Pandas.

After printing the first DataFrame, it rounds the numbers to two decimal places and adds commas to the "Price" column.

```
import pandas as pd

data = {'Product': ['Laptop', 'Phone', 'Tablet', 'Desktop'],

    'Price': [1200.50, 799.99, 349.75, 1500.25]}

products_dataframe = pd.DataFrame(data, columns=['Product', 'Price'])

print("Given Dataframe :\n", products_dataframe)

pd.options.display.float_format = '{:,.2f}'.format

formatted_products = products_dataframe.copy()

formatted_products['Price'] = formatted_products['Price'].apply(lambda x: '{:,.2f}'.format(x))

print('\nResult :\n', formatted_products)
```

Population Data Formatting and Scaling in a Pandas DataFrame

The Pandas library is used in this example code to generate a DataFrame called "city_dataframe" that contains the names of the cities and their corresponding populations.

It shows the DataFrame at first, then scales the numbers to millions and formats the population column with commas.

```
import pandas as pd

data = {'City': ['New York', 'Los Angeles', 'Chicago', 'Houston'],

    'Population': [8336817, 3980400, 2716000, 2328000]}

city_dataframe = pd.DataFrame(data, columns=['City', 'Population'])

print("Given DataFrame:\n", city_dataframe)
```

```
pd.options.display.float_format = '{:,.2f}'.format

city_dataframe['Population'] = city_dataframe['Population'] /
1000000  # Convert population to millions

print('\nResult:\n', city_dataframe)
```

How Do You Verify the Type of Data in a Pandas DataFrame?

A two-dimensional data structure with an adjustable size that holds heterogeneous tabular data is called a Pandas dataframe. Python comes with a variety of built-in data types. Pandas are used in two different ways to verify the datatypes.

Making a Dataframe in Pandas DataFrame to Verify DataType

Consider a dataset from a retail establishment that includes information on the customer's name, serial number, product ID of the item they bought, cost, and date of purchase.

```
import pandas as pd

df = pd.DataFrame({

'Cust_No': [1,2,3],

'Cust_Name': ['Alex', 'Bob', 'Sophie'],

'Product_id': [12458,48484,11311],

'Product_cost': [65.25, 25.95, 100.99],

'Purchase_Date': [pd.Timestamp('20180917'),

        pd.Timestamp('20190910'),

        pd.Timestamp('20200610')]

})

df
```

Use Pandas to verify the data type in *pandas.DataFrame.dtypes*

This function can be used by users to determine the data type of a specific dataset or dataset column. This method can return a single data type for a given column or a list of data types for each

column.

df.dtypes

Check the data type in Pandas using *pandas.DataFrame.select_dtypes*

In contrast to validating Data Type, the user can also check to see if data for a specific Datatype exists; if not, the user will receive an empty dataset. Based on the column dtypes, this method returns a subset of the columns in the DataFrame.

df.select_dtypes(include = 'int64')

How to modify the Python Pandas datetime format?

"YYYY-MM-DD" is the default format for dates and times. As a result, the date format for August 15, 2024 will be "2024-08-15". It is possible to alter the datetime format; by altering, we mean to change the format's order and style.

Python's strftime() can modify the date format.

Syntax

> strftime(format)
>
> In this case, the string format denotes the kind of necessary date format.
>
> For year %y
>
> For month %m
>
> For day %d

> **Example:**
>
> ```
> import pandas as pd
> date_sr = pd.Series(pd.date_range(
>    '2019-12-31', periods=3, freq='M', tz='Asia/Calcutta'))
> ind = ['Day 1', 'Day 2', 'Day 3']
> date_sr.index = ind
> change_format = date_sr.dt.strftime('%d,%m,%Y')
> print(change_format)
> ```

In a Pandas dataframe, change the column type from string to datetime format.

Time series data are frequently encountered when working with data in Pandas, and as we all know, Pandas is an excellent tool for handling time-series data in Python.

Replace the string dataframe column type with a datetime one

The techniques listed below allow us to change a type in a Pandas Dataframe from string to datetime format:

Using pd.to_datetime() function.

Using DataFrame.astype() function

Using pandas.to_datetime()

Pandas uses the pd.to_datetime() function to convert a column to date time.

The pd.to_datetime() function uses the Pandas library to construct a DataFrame named “df” with the columns “Date,” “Event,” and “Cost.” The DataFrame is then printed, and the info() method is used to provide details about the data types and non-null values of each column.

```
import pandas as pd

df = pd.DataFrame({'Date':['11/8/2011', '04/23/2008',
'10/2/2019'],

        'Event':['Music', 'Poetry', 'Theatre'],

        'Cost':[10000, 5000, 15000]})

print(df)

df.info()
```

Pandas uses the DataFrame.astype() function to convert a column to datetime.

In this example, the DataFrame.astype() function is used to create a Pandas DataFrame called “df” with

the columns “Date,” “Event,” and “Cost.” The DataFrame is then printed, and the info() method is used to output information about the data types and non-null values in each column, with a focus on the “Date” column.

```
import pandas as pd

df = pd.DataFrame({'Date':['11/8/2011', '04/23/2008',
'10/2/2019'],

        'Event':['Music', 'Poetry', 'Theatre'],

        'Cost':[10000, 5000, 15000]})

print(df)

df.info()
```

Pandas.to_datetime() can be used to convert the column type from string to the format “yyyymmdd.”

The Pandas library is used in this example’s *pandas.to_datetime()* function to generate a DataFrame called “df” from a nested list called “player_list” that contains information about medical treatments. The data types of each column are then shown using the dtypes property, and the DataFrame is printed.

Example

```
import pandas as pd

player_list = [['20200712',50000,'20200812'],

        ['20200714',51000,'20200814'],

        ['20200716',51500,'20200816'],

        ['20200719',53000,'20200819'],

        ['20200721',54000,'20200821'],

        ['20200724',55000,'20200824'],

        ['20200729',57000,'20200824']]

df = pd.DataFrame(

  player_list,columns = ['Treatment_start',

            'No.of Patients',
```

```
                'Treatment_end'])

print(df)

print()

print(df.dtypes)
```

Exploratory Data Analysis

Another critical phase in the data analysis process is exploratory data analysis (EDA), which entails highlighting a dataset's key features, frequently using visual aids.

- EDA aims to ascertain the fundamental structure of the data, identify trends and irregularities, verify assumptions, and test theories. Making wise choices about feature engineering, modeling, and data pretreatment requires the use of EDA.
- Identifying anomalies in the dataset and formulating suggestions for further research are the main objectives of exploratory data analysis (EDA), which ensures a full understanding of the nuances of the data.
- Analysts employ a range of EDA techniques, such as summary statistics, correlation analysis, and data visualization with tools like box plots, scatter plots, and histograms, to gain a thorough knowledge of the data.
- EDA improves understanding of data distribution, variable correlations, and anomalies, which aids in the generation of hypotheses and decision-making. All things considered, the ability of EDA to recognize patterns and abnormalities improves the effectiveness of data-driven projects.

Big Data Analytics Salary

Univariate Analysis

Data visualization is crucial because it can aid in our understanding of the data in univariate analysis by helping us plot the appropriate charts.

A Python 2D charting library called Matplotlib can be used to make simple charts.

Based on the Matplotlib architecture, Seaborn is a Python module that uses brief code segments to create and modify statistical charts from Pandas and Numpy.

Example: We will draw various plot types for univariate analysis, including countplots, *violinplots,* and *swarmplots*.

```
quality_counts = df['quality'].value_counts()

plt.figure(figsize=(8, 6))

plt.bar(quality_counts.index, quality_counts, color='darpink')

plt.title('Count Plot of Quality')

plt.xlabel('Quality')

plt.ylabel('Count')

plt.show()
```

Kernel Density Plots

```
sns.set_style("darkgrid")

numerical_columns = df.select_dtypes(include=["int64",
"float64"]).columns

plt.figure(figsize=(14, len(numerical_columns) * 3))

for idx, feature in enumerate(numerical_columns, 1):

    plt.subplot(len(numerical_columns), 2, idx)

    sns.histplot(df[feature], kde=True)

    plt.title(f"{feature} | Skewness: {round(df[feature].skew(), 2)}")

plt.tight_layout()

plt.show()
```

Bivariate Analysis

Two variables are analyzed simultaneously in a bivariate analysis to search for patterns, relationships, or interactions between them. This statistical strategy is necessary to understand how changes in one variable may correspond to

changes in another.

Bivariate analysis provides information on the kind and strength of connections, which enables a comprehensive understanding of the interdependence between two variables within a dataset.

```
sns.set_palette("Pastel1")

plt.figure(figsize=(10, 6))

sns.pairplot(df)

plt.suptitle('Pair Plot for DataFrame')

plt.show()
```

- The distribution of the various variables is displayed by the histograms of kernel density plots if the plot is diagonal.
- The association between the variable pairs is shown if the scatter plot is located in the lower triangle.
- Symmetry is shown if the scatter plots above and below the diagonal are mirror images.
- Peak positions are represented by more centrally placed histogram plots.
- The histogram's skewness can be determined by looking at how symmetrical or skewed it is to the left or right.

Violin Plot

```
df['quality'] = df['quality'].astype(str)

plt.figure(figsize=(10, 8))

sns.violinplot(x="quality", y="alcohol", data=df, palette={

        '3': 'lightcoral', '4': 'lightblue', '5': 'lightgreen', '6': 'gold', '7': 'lightskyblue', '8': 'lightpink'}, alpha=0.7)

plt.title('Violin Plot for Quality and Alcohol')

plt.xlabel('Quality')

plt.ylabel('Alcohol')

plt.show()
```

To understand the violin plot,

- A greater breadth denotes a higher density and potentially more data points.

- A balanced distribution is indicated by a symmetrical graphic.
- The most common value in distribution is shown by the peak or bulge in the violin plot.
- Greater variability is indicated by longer tails.
- The violin plot's middle line is known as the median line. It facilitates comprehension of central tendencies.

Multivariate Analysis

Multivariate analysis examines and interprets a dataset's interactions between three or more variables at the same time.

- It aims to uncover complex patterns, correlations, and interactions between various variables in order to offer a thorough knowledge of their collective behavior.
- Multivariate analysis uses advanced statistical techniques like factor analysis, principal component analysis, and multivariate regression to investigate connections and dependencies between several variables.
- Widely used in fields including biology, economics, and marketing, multivariate analysis provides deep insights and aids in decision-makers' ability to make well-informed decisions based on intricate linkages revealed in multidimensional datasets.

Correlation Matrix

```
plt.figure(figsize=(15, 10))

sns.heatmap(df.corr(), annot=True, fmt='.2f', cmap='Pastel2', linewidths=2)

plt.title('Correlation Heatmap')

plt.show()
```

To understand a plot of a correlation matrix,

- Strong positive correlation is indicated by values around +1, strong negative correlation is shown by values near -1, and no linear correlation is suggested by values of 0.

- Lighter hues indicate weaker correlations, whereas darker shades indicate stronger correlations.
- Variables with positive correlation travel in the same directions. The other increases along with the first.
- Variables with negative correlation move against each other. A rise in one variable corresponds to a fall in the other.

Big Data analytics Project Ideas

Time Series Data Analysis

The data in this kind of sequence are numerical data that are regularly recorded. They are produced by economic processes such as medical observations and stock market analysis. When researching natural phenomena, they are helpful.

These days, piecewise data approximations for additional analysis are done using series. We locate a subsequence in this time series data that corresponds to our search query.

Time Series Forecasting

Forecasting is a technique for predicting future events based on historical and current data. One technique for forecasting time series is trend analysis.

- This function creates historical time series patterns that are utilized in both short- and long-term forecasts.
- Time series can yield a variety of patterns, such as cyclic, trend, and seasonal motions that are apparent in relation to time or season.
- Several well-liked techniques for this kind of analysis include extended memory time series modeling, SARIMA, and ARIMA.

Symbolic Data: This kind of structured collection of objects or occurrences can be documented with or

without a precise sense of time.

- Symbolic data can include certain sequences, including online clickstreams and customer shopping sequences.
- Symbolic sequences are the main application for sequential pattern mining.

Biological Data: Protein and DNA sequences make up their composition. Though lengthy and intricate, they contain a deeper meaning.

- The sequence of amino acids or nucleotides is determined using these kinds of data.
- These analyses are vital to bioinformatics and contemporary biology, serving as tools for indexing, aligning, and biological sequence analysis.

Types of Time Series Data

Two general categories can be used to categorize time series data:

- Continuous time series data that includes temperature data, stock market data, and sensor data.
- Discrete time series data that includes count data, categorical data, and binary data.

Data Analysis Tools and Libraries

There are libraries in Python that help data analytics. They are as follows:

Pandas: This important library provides flexible data structures such as DataFrames, which facilitate effective data analysis, visualization, and manipulation.

Gaining proficiency with Pandas can greatly improve your capacity to manage and glean insights from intricate datasets, making it an

essential competency for any scientist or data analyst.

NumPy: This core library makes data handling and calculation extremely efficient by supporting matrices, arrays, and high-level mathematical operations.

Comprehending NumPy is essential for carrying out sophisticated data analysis and scientific computing, and it forms the basis for numerous other data science libraries.

Matplotlib is a tool for creating 2D and 3D visualization plots. It can also produce a wide range of output formats, such as data graphs.

SciPy: A Python package called Scipy helps resolve a variety of mathematical problems and procedures. By utilizing its high-level functions, the code's complexity will be greatly reduced, improving data analysis.

TensorFlow: TensorFlow is an open-source machine learning library developed by Google. Because TensorFlow makes it easier to create computational networks and execute them efficiently across a range of hardware platforms, it is used to develop and train deep learning models.

Conclusion

All set to take on the future? This data analytics tutorial is the key to the modern world. Develop your expertise with our **data analytics training in Chennai.**

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Courses

- Python
- Software Testing
- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development

- About Us
- Blog Posts
- Careers
- Contact
- Placement Training
- Corporate Training
- Hire With Us
- Job Seekers
- SLA's Recently Placed Students
- Reviews
- Sitemap

Important Links

- Disclaimer
- Privacy Policy
- Terms and Conditions

Social Media Links

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed

DevOps

Software Suggest

Sitejabber