Theory for DATA ANALYTICS-I:

1. Problem Statement: The objective of this assignment is to perform various data analytics operations using Python on an open source dataset. The tasks involve importing the required Python libraries, selecting and describing the dataset, loading it into a pandas DataFrame, and performing data preprocessing, data formatting, and data normalization.
2. Importing Python Libraries: Python libraries such as pandas and numpy are essential for data manipulation and analysis. These libraries provide functions and tools for handling tabular data, performing mathematical operations, and working with arrays. We will import these libraries to utilize their functionality in this assignment.
3. Dataset Selection and Description: For this assignment, an open source dataset will be selected from a reliable source such as Kaggle. The dataset will be described in terms of its source, providing the URL of the website where it can be accessed. The description will include relevant details about the dataset's content, structure, and purpose.
4. Loading Dataset into Pandas DataFrame: The pandas library offers a powerful data structure called a DataFrame, which is suitable for loading and manipulating tabular data. We will use the pandas library to load the selected dataset into a DataFrame. This will enable us to access and analyze the data more efficiently using pandas' built-in functions and methods.
5. Data Preprocessing: Data preprocessing is a crucial step in data analytics. We will perform several tasks, including checking for missing values in the dataset using the isnull() function provided by pandas. Additionally, we will utilize the describe() function to obtain initial statistics, such as summary statistics for numeric variables. Variable descriptions will be provided, including information about the types of variables present in the dataset (numeric, character, integer, factor, logical). Furthermore, we will examine the dimensions of the DataFrame using appropriate pandas functions to determine the number of rows and columns.
6. Data Formatting and Normalization: To ensure accurate analysis, it is important to format and normalize the data properly. We will summarize the types of variables in the dataset by checking their data types using pandas. If any variables are not in the correct data type for analysis, we will apply suitable type conversions. In particular, categorical variables will be transformed into quantitative variables using specific techniques available in Python.

Academic dataset

|  | **RollNo** | **Name** | **DSBDA** | **AI** | **CC** | **WT** | **TotalMarks** | **Percentage** | **Result** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | A | 50.0 | 67 | 88 | 54.0 | 259 | 64.75 | Pass |
| **1** | 2 | B | 68.0 | 34 | 28 | 28.0 | 158 | 39.50 | Fail |
| **2** | 3 | C | 98.0 | 59 | 67 | 67.0 | 291 | 72.75 | Pass |
| **3** | 4 | D | 50.0 | 79 | 91 | 72.0 | 292 | 73.00 | Pass |
| **4** | 5 | E | 56.0 | 67 | 35 | 78.0 | 236 | 59.00 | Pass |
| **5** | 6 | F | 79.0 | 72 | 43 | 72.0 | 266 | 66.50 | Pass |
| **6** | 7 | G | 71.0 | 69 | 72 | 83.0 | 295 | 73.75 | Pass |
| **7** | 8 | H | 75.0 | 93 | 87 | 200.0 | 455 | 113.75 | Pass |
| **8** | 9 | I | 30.0 | 89 | 93 | 60.0 | 272 | 68.00 | Fail |
| **9** | 10 | J | 57.0 | 56 | 70 | 32.0 | 215 | 53.75 | Fail |
| **10** | 11 | K | 68.0 | 45 | 63 | 65.0 | 241 | 60.25 | Pass |
| **11** | 12 | L | 64.0 | 88 | 59 | NaN | 211 | 52.75 | False |
| **12** | 13 | M | 63.0 | 41 | 59 | 66.0 | 229 | 57.25 | Pass |
| **13** | 14 | N | 45.0 | 36 | 80 | 58.0 | 219 | 54.75 | Pass |
| **14** | 15 | O | 75.0 | 78 | 36 | 84.0 | 273 | 68.25 | Pass |
| **15** | 16 | P | NaN | 82 | 36 | 33.0 | 151 | 37.75 | Pass |
| **16** | 17 | Q | 58.0 | 78 | 66 | 80.0 | 282 | 70.50 | Pass |
| **17** | 18 | R | 72.0 | 82 | 33 | 83.0 | 270 | 67.50 | Fail |
| **18** | 19 | S | 41.0 | 42 | 31 | 52.0 | 166 | 41.50 | Fail |
| **19** | 20 | T | 55.0 | 59 | 59 | 88.0 | 261 | 65.25 | Pass |
| **20** | 21 | U | 37.0 | 37 | 33 | 66.0 | 173 | 43.25 | Fail |
| **21** | 22 | V | 78.0 | 52 | 65 | 71.0 | 266 | 66.50 | Pass |
| **22** | 23 | W | 75.0 | 64 | 62 | 38.0 | 239 | 59.75 | Pass |
| **23** | 24 | X | 56.0 | 84 | 44 | 90.0 | 274 | 68.50 | Pass |
| **24** | 25 | Y | 85.0 | 78 | 96 | 81.0 | 340 | 85.00 | Pass |
| **25** | 26 | Z | 45.0 | 85 | 45 | 45.0 | 220 | 55.00 | Pass |

Theory for the practical: "Create an Academic Performance dataset and perform data operations using Python."

1. Dataset Creation:
   * Create an "Academic Performance" dataset that includes various variables related to student performance.
   * The dataset should consist of relevant variables such as student ID, exam scores, attendance, study hours, etc.
   * Ensure that the dataset has a sufficient number of observations to perform data operations effectively.
2. Scan for Missing Values and Inconsistencies:
   * Use Python and pandas library functions to scan the dataset for missing values and inconsistencies.
   * Missing values can be identified using functions like isnull() or isna() in pandas.
   * Inconsistencies may involve invalid values or data entry errors that need to be identified and resolved.
3. Dealing with Missing Values and Inconsistencies:
   * Apply suitable techniques to handle missing values and inconsistencies in the dataset.
   * For missing values, options include removing rows with missing values, imputing missing values with mean/median values, or using more advanced techniques like multiple imputation.
   * Inconsistencies can be addressed by correcting erroneous values, removing outliers, or applying domain-specific rules.
4. Scan for Outliers in Numeric Variables:
   * Focus on numeric variables in the dataset and scan for outliers using appropriate techniques.
   * Common methods to identify outliers include calculating z-scores, using boxplots, or applying statistical tests.
   * Outliers are observations that significantly deviate from the normal pattern and may require special attention in the analysis.
5. Dealing with Outliers:
   * Apply suitable techniques to handle outliers in the dataset.
   * Options for handling outliers include removing the outliers, winsorizing (replacing outliers with a threshold value), or transforming the data to reduce the impact of outliers on the analysis.
6. Data Transformations:
   * Select at least one variable from the dataset to apply data transformations.
   * The purpose of the transformation should be one of the following reasons:
     + Change the scale of the variable to improve understanding (e.g., scaling exam scores from a range of 0-100 to a range of 0-10).
     + Convert a non-linear relationship to a linear one (e.g., taking the logarithm of a variable with exponential growth).
     + Decrease skewness and achieve a normal distribution (e.g., applying a square root transformation to a positively skewed variable).
7. Applying Data Transformations:
   * Use appropriate Python functions or libraries to apply the chosen data transformations on the selected variable.
   * Ensure that the transformation is properly documented and clearly explains the reason behind choosing a specific transformation.

A3:

1. Summary Statistics: Summary statistics provide a concise summary of the main characteristics of a dataset. They include measures such as mean, median, minimum, maximum, and standard deviation. These statistics help in understanding the central tendency, variability, and distribution of the data.
2. Grouping by Categorical Variable: Grouping the dataset by a categorical variable allows us to analyze the numeric variables separately for each category. By grouping the data, we can compare the summary statistics across different categories and identify any variations or patterns.
3. Categorical and Numeric Variables: Categorical variables are qualitative variables that represent groups or categories, such as age groups or species. Numeric variables, on the other hand, are quantitative variables that represent numerical values, such as age or income.
4. Summary Statistics for Numeric Variables Grouped by Categorical Variable: To obtain summary statistics for numeric variables grouped by a categorical variable, we perform the following steps: a. Select the categorical variable and the numeric variable of interest. b. Group the dataset by the categorical variable using Python libraries like pandas. c. Calculate the summary statistics (mean, median, minimum, maximum, standard deviation) for the numeric variable within each group. d. Create a list that contains a numeric value for each response to the categorical variable, representing the summary statistics for each group.

Theory for "Displaying Basic Statistical Details for Iris Dataset":

1. Iris Dataset: The Iris dataset is a popular dataset used in machine learning and statistics. It consists of measurements of sepal length, sepal width, petal length, and petal width for three species of iris flowers: Iris-setosa, Iris-versicolor, and Iris-virginica.
2. Basic Statistical Details: Basic statistical details provide information about the distribution and summary statistics of a dataset. These details include percentiles, mean, standard deviation, etc., which help understand the central tendency, spread, and variation in the data.
3. Python Program for Displaying Basic Statistical Details: To display basic statistical details for specific species in the Iris dataset, we can use the following steps: a. Load the Iris dataset using Python libraries like pandas. b. Filter the dataset to include only the records for the desired species (e.g., Iris-setosa, Iris-versicolor, Iris-virginica). c. Calculate and display the desired statistical details (percentiles, mean, standard deviation) for the selected species.

By following these steps, we can obtain and display basic statistical details for specific species in the Iris dataset, including the desired percentiles, mean, standard deviation, and other relevant summary statistics.

1. Linear Regression Model: Linear regression is a statistical modeling technique used to analyze the relationship between a dependent variable (target variable) and one or more independent variables (predictor variables). It assumes a linear relationship between the predictor variables and the target variable and aims to find the best-fit line that minimizes the sum of squared errors.
2. Boston Housing Dataset: The Boston Housing dataset is a widely used dataset in machine learning and regression analysis. It contains information about various houses in Boston and includes 506 samples (observations) with 14 feature variables (predictors) and a target variable (home prices).
3. Problem Statement: The problem is to create a linear regression model using Python or R to predict home prices based on the given Boston Housing dataset. The goal is to analyze the relationship between the predictor variables (such as number of rooms, crime rate, etc.) and the target variable (home prices) and develop a model that can accurately predict the home prices for new data.
4. Steps to Create a Linear Regression Model: a. Load the Boston Housing dataset into a suitable data structure (such as a DataFrame in Python's pandas or a data frame in R). b. Preprocess the dataset by handling missing values, outliers, and scaling if necessary. c. Split the dataset into training and testing sets to evaluate the model's performance. d. Select the appropriate features (predictor variables) and the target variable (home prices) for the model. e. Fit a linear regression model to the training data using suitable libraries like scikit-learn in Python or lm() function in R. f. Evaluate the model's performance using suitable evaluation metrics such as mean squared error (MSE), root mean squared error (RMSE), or R-squared. g. Use the trained model to make predictions on the testing data and evaluate its accuracy. h. Analyze the coefficients of the model to interpret the impact of each predictor variable on the home prices.

By following these steps, you can create a linear regression model using Python or R to predict home prices based on the Boston Housing dataset. The model aims to find the relationship between the predictor variables and the target variable, allowing for accurate predictions of home prices for new data.



thoery for : DATA ANALYTICS-II 1. Problem Statement Implement logistic regression using Python/R to perform classification on Social\_Network\_Ads.csv dataset. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

ChatGPT

Theory for "Implementing Logistic Regression for Classification and Computing Confusion Matrix":

1. Logistic Regression: Logistic regression is a popular classification algorithm used to model the relationship between a binary dependent variable and one or more independent variables. It estimates the probability of the dependent variable belonging to a particular class by fitting a logistic function to the data.
2. Social\_Network\_Ads.csv Dataset: The Social\_Network\_Ads.csv dataset is a dataset that contains information about users in a social network, including their age, estimated salary, and whether they purchased a particular product (the dependent variable). The goal is to use logistic regression to classify whether a user will purchase the product based on their age and estimated salary.
3. Problem Statement: The problem is to implement logistic regression using Python or R on the Social\_Network\_Ads.csv dataset. The objective is to classify users into two groups: those who purchased the product (positive class) and those who did not (negative class). Additionally, the confusion matrix will be computed to evaluate the performance of the logistic regression model.
4. Steps to Implement Logistic Regression and Compute Confusion Matrix: a. Load the Social\_Network\_Ads.csv dataset into a suitable data structure (e.g., DataFrame in Python's pandas or data frame in R). b. Preprocess the dataset by handling missing values, scaling if necessary, and encoding categorical variables (if any). c. Split the dataset into training and testing sets to evaluate the model's performance. d. Select the independent variables (age and estimated salary) and the dependent variable (purchased) for the logistic regression model. e. Fit a logistic regression model to the training data using suitable libraries like scikit-learn in Python or glm() function in R. f. Use the trained model to make predictions on the testing data and obtain the predicted classes. g. Compute the confusion matrix using the predicted classes and the actual classes from the testing data. h. Calculate various performance metrics from the confusion matrix, such as true positive (TP), false positive (FP), true negative (TN), false negative (FN), accuracy, error rate, precision, and recall.

Theory for "Implementing Simple Naïve Bayes Classification Algorithm and Computing Confusion Matrix":

1. Simple Naïve Bayes Classification Algorithm: The Simple Naïve Bayes classification algorithm is a probabilistic algorithm used for classification tasks. It is based on Bayes' theorem and assumes that the features are conditionally independent given the class variable. Despite its simplicity, Naïve Bayes has been proven to be effective in many real-world applications.
2. iris.csv Dataset: The iris.csv dataset is a popular dataset used in machine learning and classification tasks. It consists of measurements of sepal length, sepal width, petal length, and petal width for three species of iris flowers: Iris-setosa, Iris-versicolor, and Iris-virginica. The goal is to use the Simple Naïve Bayes algorithm to classify the species based on the provided measurements.
3. Problem Statement: The problem is to implement the Simple Naïve Bayes classification algorithm using Python or R on the iris.csv dataset. The objective is to classify the iris flowers into their respective species based on the measurements of sepal length, sepal width, petal length, and petal width. Additionally, the confusion matrix will be computed to evaluate the performance of the Naïve Bayes model.
4. Steps to Implement Simple Naïve Bayes and Compute Confusion Matrix: a. Load the iris.csv dataset into a suitable data structure (e.g., DataFrame in Python's pandas or data frame in R). b. Preprocess the dataset by handling missing values, scaling if necessary, and encoding categorical variables (if any). c. Split the dataset into training and testing sets to evaluate the model's performance. d. Select the independent variables (sepal length, sepal width, petal length, petal width) and the dependent variable (species) for the Naïve Bayes model. e. Fit a Simple Naïve Bayes model to the training data using suitable libraries like scikit-learn in Python or naiveBayes() function in R. f. Use the trained model to make predictions on the testing data and obtain the predicted classes. g. Compute the confusion matrix using the predicted classes and the actual classes from the testing data. h. Calculate various performance metrics from the confusion matrix, such as true positive (TP), false positive (FP), true negative (TN), false negative (FN), accuracy, error rate, precision, and recall.

Theory for "Document Preprocessing and Creating Document Representation":

1. Document Preprocessing: Document preprocessing is a crucial step in natural language processing (NLP) tasks to transform raw text data into a format suitable for analysis. It involves several techniques to clean, normalize, and transform the text data.
2. Tokenization: Tokenization is the process of splitting a document or a sentence into individual words or tokens. It helps in extracting the basic units of meaning from the text. Tokenization can be performed by splitting on whitespace or using more advanced techniques like regular expressions.
3. POS Tagging: Part-of-speech (POS) tagging is the process of assigning grammatical tags (such as noun, verb, adjective, etc.) to each word in a sentence. It helps in understanding the role and syntactic context of each word in the document. POS tagging can be performed using pre-trained models or rule-based approaches.
4. Stop Words Removal: Stop words are commonly occurring words (such as articles, prepositions, etc.) that do not add much meaning to the text. Removing stop words helps in reducing noise and improving the efficiency of text analysis. Stop words can be obtained from predefined lists or custom-defined based on the specific task.
5. Stemming and Lemmatization: Stemming and lemmatization are techniques used to reduce words to their base or root forms. Stemming involves removing suffixes from words to obtain the stem, while lemmatization aims to find the lemma (base form) of a word by considering its context and part of speech. These techniques help in reducing the dimensionality of the text data and capturing the core meaning of words.
6. Term Frequency (TF) and Inverse Document Frequency (IDF): Term Frequency (TF) measures the frequency of a term (word) within a document. It provides information about the importance of a term within a document. Inverse Document Frequency (IDF), on the other hand, measures the rarity or uniqueness of a term across the entire document collection. It helps in identifying terms that are more informative and discriminative. TF-IDF is a product of TF and IDF, which is commonly used to represent documents in information retrieval and text mining tasks.
7. Document Representation: To create a representation of a document, TF-IDF values can be calculated for each term in the document. TF-IDF assigns higher weights to terms that are frequent within the document but rare in the overall document collection. The resulting TF-IDF vector represents the document in a numerical form, capturing the importance of different terms in the document.

Theory for:

1. Titanic Dataset: The Titanic dataset contains information about the passengers who boarded the Titanic ship, including their demographic details and survival status. The dataset is commonly used for exploratory data analysis and predictive modeling tasks.
2. Exploratory Data Analysis (EDA): Exploratory Data Analysis is a crucial step in understanding the data and uncovering patterns or insights. It involves summarizing the main characteristics of the dataset, visualizing the data, and identifying relationships or trends.
3. Seaborn Library: Seaborn is a popular Python library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of creating visualizations and offers various plotting functions to explore relationships in data.
4. Problem Statement: The problem is to use the Titanic dataset and the Seaborn library to explore patterns in the data and specifically analyze the distribution of ticket prices ('fare') for each passenger. The goal is to plot a histogram to visualize how the ticket prices are distributed among the passengers.
5. Steps to Explore and Plot Ticket Price Distribution: a. Load the Titanic dataset into a suitable data structure (e.g., DataFrame in Python's pandas). b. Use the Seaborn library to visualize the dataset and uncover patterns. Seaborn provides functions like scatter plots, bar plots, box plots, etc., to explore the relationships between variables and identify patterns. c. Focus on the 'fare' column to analyze the ticket prices for each passenger. d. Create a histogram using the Seaborn library to visualize the distribution of ticket prices. A histogram displays the frequency or count of values within certain bins or intervals. e. Set appropriate parameters for the histogram, such as the number of bins, axis labels, title, etc., to make the visualization informative and visually appealing. f. Plot the histogram using Seaborn's plotting function, and display the resulting visualization.

By following these step

Theory for "Data Visualization II - Box Plot and Inferences":

1. Data Visualization: Data visualization is a powerful technique for representing data graphically to extract meaningful insights and patterns. It helps in understanding the data distribution, relationships between variables, and identifying outliers or anomalies.
2. Box Plot: A box plot, also known as a box-and-whisker plot, is a graphical representation of the distribution of a numerical variable. It displays the summary statistics of the data, such as the median, quartiles, and potential outliers.
3. Problem Statement: The problem is to use the Titanic dataset, specifically the 'titanic' dataset used in the previous problem, to visualize the distribution of age with respect to each gender and their survival status. The goal is to create a box plot that represents the age distribution for each gender and distinguishes the survived and not survived passengers.
4. Steps to Plot the Box Plot: a. Load the Titanic dataset, 'titanic', into a suitable data structure (e.g., DataFrame in Python's pandas). b. Extract the relevant columns for analysis, namely 'sex', 'age', and 'survived'. c. Group the data by gender and survival status, creating separate groups for each combination (e.g., female-survived, female-not survived, male-survived, male-not survived). d. Plot a box plot using a visualization library like Seaborn or Matplotlib, with 'age' as the y-axis and 'sex' as the x-axis. Each box represents the age distribution for a specific gender and survival status group. e. Add labels, title, and any other necessary information to the plot to make it informative and visually appealing.
5. Observations and Inferences:
   * By analyzing the box plot, we can observe the median (represented by the line inside the box) and the interquartile range (IQR) for each gender and survival status group.
   * The box represents the middle 50% of the data, with the bottom and top edges indicating the 25th and 75th percentiles, respectively.
   * The whiskers extend to show the range of the data, excluding any potential outliers.
   * Any data points outside the whiskers are considered outliers and are represented as individual points or circles.
   * By comparing the box plots for different gender and survival status groups, we can infer differences in the age distributions.
   * We can observe if there are any significant differences in the median age, spread, or presence of outliers between different gender and survival status groups

Theory for a10;

1. Iris Flower Dataset: The Iris flower dataset is a popular and widely used dataset in machine learning and statistics. It contains measurements of different features of three species of Iris flowers (Setosa, Versicolor, and Virginica). Each sample in the dataset consists of four numeric features (sepal length, sepal width, petal length, and petal width) and the corresponding species label.
2. DataFrame: A DataFrame is a data structure in Python's pandas library that organizes data in a tabular form, similar to a spreadsheet or SQL table. It provides functionalities for data manipulation, analysis, and visualization.
3. Problem Statement: The problem is to download the Iris flower dataset (or any other dataset) and perform exploratory data analysis on it. The goal is to analyze the dataset's features, their types, and distributions using histograms and boxplots. Additionally, we need to compare the feature distributions and identify any outliers present in the dataset.
4. Steps to Explore and Analyze the Dataset: a. Download the Iris flower dataset from a reliable source, such as the UCI Machine Learning Repository. b. Load the dataset into a DataFrame, using suitable functions provided by the pandas library. c. Scan the dataset to understand its structure and contents. d. List down the features present in the dataset. In the Iris flower dataset, the features include sepal length, sepal width, petal length, and petal width. e. Identify the types of each feature. In this case, the features are numeric since they represent measurements. f. Create a histogram for each feature to visualize its distribution. A histogram shows the frequency or count of values within certain bins or intervals. g. Create a boxplot for each feature to understand its central tendency, spread, and identify any potential outliers. A boxplot displays summary statistics such as the median, quartiles, and outliers. h. Compare the distributions of different features to identify any patterns or differences. i. Look for any outliers in the dataset, which are data points that significantly deviate from the rest of the distribution.
5. Inferences:
   * By analyzing the histograms, we can observe the shape and spread of each feature's distribution. This helps in understanding the range and frequency of values for each feature.
   * Boxplots provide a visual representation of the central tendency (median), spread (interquartile range), and any outliers present in the data.
   * Comparing the feature distributions allows us to identify any patterns or differences between them.
   * Outliers can be identified as data points that fall significantly outside the whiskers of the boxplot. These outliers may require further investigation and consideration during data analysis.

By following these step