Algorithm:

1) Import libraries:
   import pandas for data handling
   Import variance & stdev functions from statistic model

2) Read data:
   Read data from a csv file into a pandas dataframe ('df')
   create new data frame ('data') with selected column

3) Calculate mean, median, mode:
   calculate mean, median & mode using function .mean
   .median () & .mode () & print them

4) Range calculation:
   calculate Range of column using formula (max-min)

5) Variance & standard deviation:
   calculate variance & standard variation

6) Skewness calculation:
   calculate skewness using formula 3*(mean - median)
   print result.

Algorithm :

1) Import libraries :

Import pandas for data handling

Import the math file for mathematical calculations.

2) Read Data :

Read csv file into pandas dataframe.

3) calculate Means :

calculate mean for 'trip_distance' ($\bar{x}$) & total distance ($\bar{y}$)

4) Calculate deviations :

calculate deviation $x-\bar{x}$ & $y-\bar{y}$.

5) calculat squeue & product :

calculate product ~~of~~ & squeue of deviations $(x-\bar{x})^2$, $(y-\bar{y})^2$, $(x-\bar{x})(y-\bar{y})$.

6) calculate correlation coefficient (r) :

$$r = \frac{\Sigma (x-\bar{x})(y-\bar{y})}{\sqrt{\Sigma (x-\bar{x})^2} + \sqrt{\Sigma (y-\bar{y})^2}}$$

7) print ran pearsons coefficient of correlation r :

print value of r.

for grouped data

1) import libraries

2) Create dataframe :

create dataframe with two grouped column "pass-count" & "trip-distance" & frequency.

3) Use .corr() function to calculate correlation between data

4) print r value

Algorithm :

1) Import libraries.
   Import pandas for data handling
   Import KNN imputer from scikit learn for K Nearest Neighbors imputation
2) Read Data
   Read uber data CSV file into pandas Dataframe ('df')
3) Fill NaN values with constant
   create new data frame df1 by filling NaN value with a constant value of 1.
4) Fill NaN values with previous value
   fill the null value with previous values using method = pad
5) Fill NaN with Next value:
   fill null values with next non-Nan value using method = bfill
6) Fill NaN with Mean, Media & Mode:
   fill null values with "passenger_count" column mean, median & Mode
7) Fill NaN with Max & ~~mean~~ Min value
   fill null values with .max() & .min() function on passenger_count column
8) Fill NaN value with frequently occur value
   fill Nan or null value using mode function & taking max value
9) Fill NaN using KNN Imputation
   Use KNN imputer with 2 Neighbors to impute NaN values in "passenger_count".

Date:

formula for calculating T-score.:

$$T_{score} = \frac{\bar{x} - \mu}{\sigma / \sqrt{n}}$$

where,

$\bar{x} \Rightarrow$ mean of sample data

$\mu \Rightarrow$ mean of population

$\sigma \Rightarrow$ standard deviation

$n \Rightarrow$ The sample size.

Algorithm:

Steps in conducting a T- Test :-

1) formulate hypothesis :

Ho : There is no significant difference

H1 : There is a significant differences.

2) Select significance level (α) :

Select α (common choices are 0.05, 0.01

3) calculate T-score :

calculate t score by putting values in formula

4) Determine critical region :

calculate critical value using of alpha using t-table

5) Make a decision :

If calculated statistics falls in critical region reject the null hypothesis.

6) Draw conclusion :

Based on result, make inferences about the population

Date:

outlier, & more specifically they are considered global outlier.

Primarily KNN rely on distance based outlier detection. In KNN the number of nearest neighbors K is a parameter that need to determine. There are other variants of the KNN algorithm supported by pyOD.

Unlike supervised learning approach unsupervised anomaly detection does not require labeled data indicating which instances are anomalies & which are not instead it identify outliers solely based on distribution of data points in the feature space. An appropriate threshold is choosen to differentiate between normal data points & anomalies while those below it are considered normal.

Overall unsupervised KNN for anomaly detection provides a flexible & ~~[illegible]~~ interpretable approach for identifying anomalies.

Algorithm:

1. Load data & libraries:
   - Import numpy pandas & matplotlib.
   - Read given csv file into a Dataframe
   - Extract trip_distance as numpy array

2. Define function:
   - Create function to calculate distance to nearest neighbor to each point
   - Identify those with mean distance to K nearest neighbor exceeding the threshold as anomalies.

Date:

(3) Detect anomalies:

call function & with tmp_distance, chosen r & threshold

(4) Separate anomalies:

Separate the points in tmp distance into anomalies & normal points.

(5) plot Result:

visualize anomalies & normal points.

Conclusion:

Distance base method offer a parameter driven approach for anomaly detection in data. Unsupervised KNN effectively identifies anomalies in distance based data by leveraging the concept of nearest neighbor. Over all KNN provide a valuable unsupervised tool for anomaly detection in various application

14/3/24

SMOTE is carried to create synthetic examples using lines connecting existing minority class instance ensuring that the synthetic data remain within boundaries of the existing distribution, thus prevents introduction of noise & outliers that might adversely affect model performance.

SMOTE is overall valuable tool for addressing class imbalance issue & improving the performance of model on imbalance dataset.

Algorithm :

Step 1 : import necessary libraries.

Step 2 : Read data from 'csv file' into a dataframe called Data

Step 3 : Create subplot with two panels
plot the original data in the first panel

Step 4 : Separate features (X) & target variable (y) from dataframe

Step 5 : Initialize SMOTE object.

Step 6 : Apply SMOTE oversampling using fit_sample method on X & y
Generate synthetic samples for the minority class to balance the dataset

Step 7 : Concatenet resampled features & target variables into a new Dataframe 'resampled_data'

Step 8 : plot resample data in the second panel

Step 9 : Display plot

Date:

- Integrated (I): represents the differencing of raw observations to allow the time series to become stationary
- Moving average (MA): Incorporate dependency between an observation & a residual error from a moving average model applied to lagged observations.

ARIMA Parameters:

Each component in ARIMA functions as a parameter with a standard notation. For ARIMA models, a standard notation would be ARIMA with p, d & q where integer values substitute for the parameters to indicate the type of model used. The parameters can be defined as.

- p : The number of lag observations in the model also known as the lag order.
- d : The number of times the raw observations are differenced also known as the differencing
- q : The size of moving average window also known as the order of moving average

Algorithm:

Step 1:

import necessary libraries such as pandas, numpy matplotlib.pyplot & ARIMA model from statsmodel

Step 2:

Load the time series data from a CSV file into pandas dataframe.

Date:

Step3 :

visualize the time series data using matplotlib.

Step4 :

perform augmented Dickey-fuller test (ADF) test to check the stationarity of time series data

Step5 :

convert the data column to date time format & set it as the index of dataframe.

Step 6 :

fit ARIMA model to the time series data with specific order (p,d,q)

Step7 :

Generate forecasts for future time periods using the fitted ARIMA model.

Conclusion :

In conclusion, this experiment utilized ARIMA modelling to analyze & forecast time series dataset representing total amount over time. This experiment demonstrate ARIMA modelling for time series analysis & forecasting

10/4/24.