**Regression Analysis**

**WORKFLOW:**

Let us try to understand the workflow which we will work on. We have the data in local database in the form of a table.

A picture containing text

Description automatically generatedDiagram, schematic

Description automatically generated with medium confidenceChart, line chart

Description automatically generatedDiagram, shape

Description automatically generatedText

Description automatically generated

Circle

Description automatically generated with medium confidence

1. Data collection: Data collection is a systematic course of gathering the observations or measurements. Data collection permits you to acquire direct information and unique experiences into your exploration issue. Once we get our source dataset. We try to understand what the different labels are and features the datasets contains.
2. Once we have data, we have lot of processing to do like encoding the categorical features. Data preprocessing in Machine Learning is a critical step that helps upgrade the quality of data to promote the extraction of significant insights from the data. Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and sorting out) the raw data to make it suitable for a building and training Machine Learning models. In basic words, data preprocessing in Machine Learning is a data mining technique that changes raw data into a understandable and readable format.
3. We analyze the data called as exploratory data analysis where we put the data in some suitable plots and graphs and see what the distribution of data points is and what is count of various features. Data analysis is the method of cleaning, changing, and processing raw data, and extricating actionable, relevant information that helps businesses make informed decisions. The method diminishes the risks inherent in decision-making by providing helpful experiences and statistics, often presented in charts, images, tables, and graphs.
4. Once we complete the data analysis part, we split our data into training data and testing data. In machine learning, we train our machine learning model with train data, and we test/ evaluate it with test data. Train test split is a [model validation](https://builtin.com/data-science/model-validation-test) procedure that allows us to simulate how a model would perform on new/unseen data. This consists of [random sampling without replacement](https://towardsdatascience.com/understanding-sampling-with-and-without-replacement-python-7aff8f47ebe4) about 75 percent of the rows and putting them into training set. The remaining 25 percent is put into test set. The colors in “Features” and “Target” indicate where the data will go (“X\_train,” “X\_test,” “y\_train,” “y\_test”) for a particular train test split.
5. Once we split our data into training data and testing, we feed this to our machine learning model. Train the model on the training set. Here, we are using regression model called XGBoost Regressor.
6. Once we train the regressor with training data, we will evaluate it with the

test data. Test the model on the testing set and evaluate the performance.

**CODING**

Moving on to coding part, we are using python language to implement the above process.

First, we import the dependency packages.

Text

Description automatically generated

Numpy – used for making numpy arrays

Pandas – for creating dataframe

Matplotlib.pyplot – for making plots and graphs

Seaborn – another plotting library

Sklearn.preprocessing – machine learning libraries

LabelEncoder – to name the labels

train\_test\_split – for splitting data into training and test data

XGBRegressor from XGBoost – for classification, where we predict the categories

Metrics from sklearn – gives us metrics to find the performance of our model

Data collection and Analysis:

1. the dataset from snowflake to pandas dataframe

Text

Description automatically generated

1. Load data from snowflake table to pandas dataframe:

Graphical user interface, text

Description automatically generated

The data in the dataframe looks like this:

A picture containing graphical user interface

Description automatically generated

1. First, we see number of data points and number of features

***df.shape –*** gives rows and columns in the dataset

1. Get the information about the dataset:

Text

Description automatically generated

1. Now we identify the categorical features:

Date

ORG\_ID

ORDER\_ID

CART\_ID

1. Checking the missing values

Text

Description automatically generated

Total field has 171 nulls!

1. We need to handle the nulls now, because we cannot train the machine learning model with null values in it,

We handle nulls with either

Mean 🡪 average value

Mode 🡪 Most Repeated Value

Here we got NULLs in the numerical column, so we calculate mean of that particular column. The calculated mean value is: 3016.93

Text

Description automatically generated

1. Now we fill the nulls with the mean value and check the nulls again

Text

Description automatically generated

There no Nulls in any of the columns in the dataset.

**DATA ANALYSIS**

1. We plot the sales against months using matplot library. Below is a replication of the bar graph:

Chart, bar chart, histogram

Description automatically generated

1. Plotting sales based on org\_id. We were working only on only one brand now i.e, kfc:

Chart, bar chart

Description automatically generated

**DATA PRE-PROCESSING**

1. Count of each stores

Text

Description automatically generated

1. We cannot train the machine learning model on the non- numerical columns. An approach to encoding categorical values is to use a technique called label encoding. Label encoding is simply converting each value in a column data to a number or numerical type. After encoding, the looks like this:

Graphical user interface

Description automatically generated with medium confidence

1. After doing the categorical encoding, the graph plot for total stores for each store\_id looks like this:

Chart

Description automatically generated

**TRAIN AND TEST SPLIT**

1. We split the data into features and target. i.e., we separate total column from the remaining dataset:

The rest of the columns data is stored in X:

Graphical user interface, text

Description automatically generated

The total column data is stored in Y:

Text

Description automatically generated

The code for the above logic is as below:

Text

Description automatically generated

1. Splitting the data into training and testing data. Code for the logic is as below:

Text

Description automatically generated

The shape of the training and testing data is as follows:

A screenshot of a computer screen

Description automatically generated

**Machine learning Model Training**

1. We are using XGBoost Regressor to train our machine learning model. Hence, we call it using a variable. Training data is in X\_train and corresponding Y value is in Y\_train. The code is as below:

Text

Description automatically generated

**Evaluation**

1. **Prediction on training data**

We have trained our machine learning model with regressor, now we will predict the Y -value using the X\_train values. The code is as below:

A screenshot of a computer

Description automatically generated with medium confidence

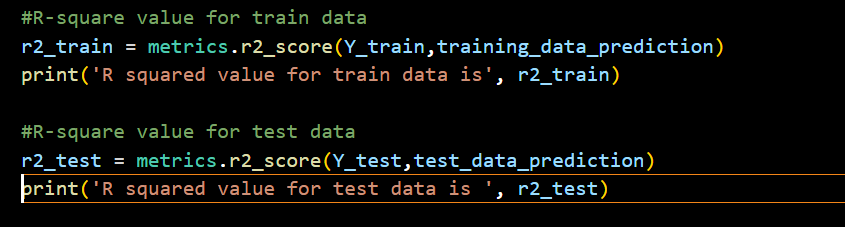
The predicted Y- values are as below for training and testing data:

A screenshot of a computer

Description automatically generated with medium confidence

1. Calculating the R-squared value to check the performance of our machine learning model

The code to calculate the R-squared value is as below:



The outcome of R-squared values are as below:

A screenshot of a computer

Description automatically generated with medium confidence

From the above outcomes, R2\_train and R\_test are very close to each other. Meaning, our machine-learning model is working perfectly!