LOW LEVEL DESIGN

(version 1.0)

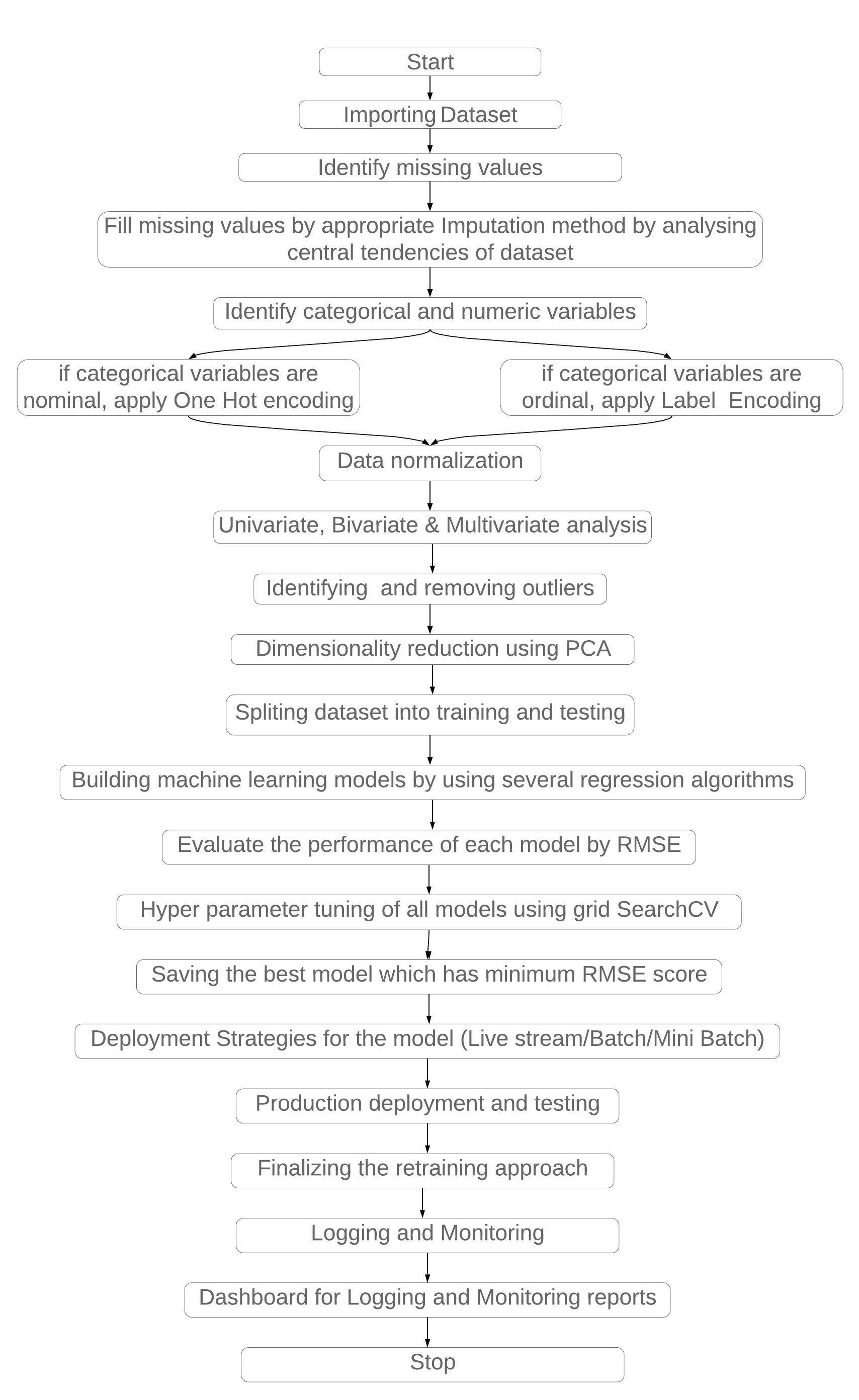


Team Leader: Devleena Banerjee

Created By: - Ajith, Viraj, Satyadeep

# **Business Problem**

* In all sort of business, it is been proven about 80-20 rule, which tells us 80% of the Income would be generated by 20% of our potential customer.
* So, our goal is to predict the revenue that is going to be generated by those potential customers in the near feature. So that marketing teams will invest appropriate money on promotional strategies to attract potential customers.
* In simple words we are given with the user’s past data and transactions (when they logged into G-store), so by using this data we need to predict the future revenue will be created by those customers.



Activity Diagram

# Data Selection

The data is from the below Kaggle link

<https://www.kaggle.com/c/ga-customer-revenue-prediction/data>

We need to download train.csv and test.csv. Some of the features are in json format so we need to parse those json columns., regarding this we will see in brief at the time of data reading.

# Data Description

***fullVisitorId****:* A unique identifier for each user of the Google Merchandise Store.

***channelGrouping****:*The channel via which the user came to the Store.

***date****:*The date on which the user visited the Store.

***device****:*The specifications for the device used to access the Store.

***geoNetwork****:* This section contains information about the geography of the user.

***sessionId****:*A unique identifier for this visit to the store.

***socialEngagementType****:*Engagement type, either “Socially Engaged” or “Not Socially Engaged”.

***totals****:*This section contains aggregate values across the session.

***trafficSource****:*This section contains information about the Traffic Source from which the session originated.

***visitId****:*An identifier for this session. This is part of the value usually stored as the utmb cookie. This is only unique to the user. For a completely unique ID, you should use a combination of fullVisitorId and visitId.

***visitNumber****:*The session number for this user. If this is the first session, then this is set to 1.

***visitStartTime****:*The timestamp (expressed as POSIX time)

# Logging

## Phase 1

Options for Logging in DB

1. Separate Folder for logs
2. Logging of every step
3. Entry to the methods
4. Exit from the methods with success/ failure message
5. Error message Logging
6. Model comparisons
7. Training start and end
8. Prediction start and end
9. Achieve asynchronous logging

Options for Log Publish

# Data Preprocessing

After reading the data, following steps should be implemented

1. Importing required libraries
2. Loading dataset into data frame
3. Data cleaning
4. Number of categorical columns and their list
5. Number of numerical columns and their list
6. Encoding
7. Data scaling
8. Removing duplicate values

# Exploratory data analysis (EDA)

In statistics, exploratory data analysis is an approach to analysing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modelling or hypothesis testing task.

1. Univariate analysis
2. Bivariate analysis
3. Multivariate analysis
4. Outlier detection
5. Finding collinearity between features

# Dimensionality reduction using (PCA)

The principal components of a collection of points in a real p-space are a sequence of p direction vectors, where the eigen vector is the direction of a line that best fits the data while being orthogonal to the first i-1 vectors.

Finding best subset of feature using principle component analysis by using following steps

1. Compute the covariance matrix of the features from the dataset.
2. Create the covariance matrix
3. Compute eigen vectors and eigen values
4. Compute basic vectors

e) Represent each sample as a linear combination of basis vectors.

# Model building

Split the dataset into training and testing part 70% data is used for training and 30% is used for testing. Train several regression models by using regression machine learning algorithms. After training, test the ML models by using testing dataset.

Following algorithms used to build ML models -

* + - 1. Regularized regressor.
      2. Decision tree regressor.
      3. Random forest regressor.
      4. XG boost regressor.
      5. KNN regressor.
      6. Time series regression

# Hyper parameter tuning

In order to improve the performance of the model, the hyperparameter tuning should be done with the help of GridSearchCV or RandomizedSearchCV.

# Performance evaluation

Evaluate each machine learning model performance by calculating root mean square error value. The model which will have the minimum RMSE score, then it will be considered as best model.

# Save the model

Save the entire state of the model for later reference

Ex: model.save('models/model\_11.h5')

# Deployment

The deployment of machine learning models is the process of making models available in production where web applications, enterprise software and APIs can consume the trained model by providing new data points and generating predictions.

## Step 1 Model building

The machine learning model which is finalized is saved as a pickle file.

## Step2 Building Web Application

Now that our machine learning pipeline and model are ready we will start building a web application that can connect to them and generate predictions on new data in real-time. This application will support ‘Online’ as well as ‘Batch’ predictions through a csv file upload. There are two parts of this application:

Front-end (designed using Streamlit/Qt5) – Streamlit/Qt5 is an open-source Python library that makes it easy to build beautiful custom web-apps for machine learning and data science. Back-end (developed using Flask in Python)-

The back-end of a web application is developed using a Flask framework. It is a framework that allows you to build web applications. A web application can be a commercial website, a blog, e-commerce system, or an application that generates predictions from data provided in real-time using trained models. Before we publish the application on Google Cloud platforms to test the web app locally. Open Anaconda Prompt and navigate to the folder where ‘app.py’ is saved on your computer and run the python file python app.py. Once executed, copy the URL into a browser, and it should open a web application hosted on your local machine (127.0.0.1). Try entering test values to see if the predict function is working.

## Step 3 Create a Docker file

A container is a type of software that packages up an application and all its dependencies so the application runs reliably from one computing environment to another. Docker is a company that provides software (also called Docker) that allows users to build, run and manage containers while Docker’s containers are the most common to containerize our application for deployment we need a Docker image that becomes a container at runtime. A Docker image is created using a Docker file. A Docker file is just a file with a set of instructions. The Docker file is case-sensitive and must be in the project folder with the other project files.

## Step 4 Deploy a ML pipeline on GKE/Cloud (AWS/GCP/Azure)

Google Kubernetes is a powerful open-source system developed by Google back in 2014, for running and managing containerized applications across a cluster of applications.

GitLab CI/CD is configured by a file called (. gitlab-ci.yml) placed at the repository’s root. This file creates a [pipeline](https://docs.gitlab.com/ee/ci/pipelines/index.html), which runs for changes to the code in the repository. Pipelines consist of one or more stages that run-in order and can each contain one or more jobs that run in parallel. These jobs (or scripts) get executed by the [GitLab Runner](https://docs.gitlab.com/runner/) agent.

Sign-in to GCP console and go to Manage Resources and then click on Create New Project followed by importing the Project Code.

Set Environment Variable

Build the Docker image of the application followed by authenticating to the Container Registry.

Create Cluster

Deploy Application -To deploy and manage applications on a GKE cluster, you must communicate with the Kubernetes cluster management system

Expose the application to the internet.

Check Service.

