Low-Level Design Document
Bike Share Prediction

Revision Number: 1.1

Last Date of Revision: 17/3/2024

Sameer Singh

Contents

1. Abstract

Bike sharing systems are a new generation of traditional bike rentals where the whole process from membership, rental and return back has become automatic. Through these systems, users are able to easily rent a bike from a particular position and return at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousand bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues. Apart from interesting real-world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for the research.

2 Introduction

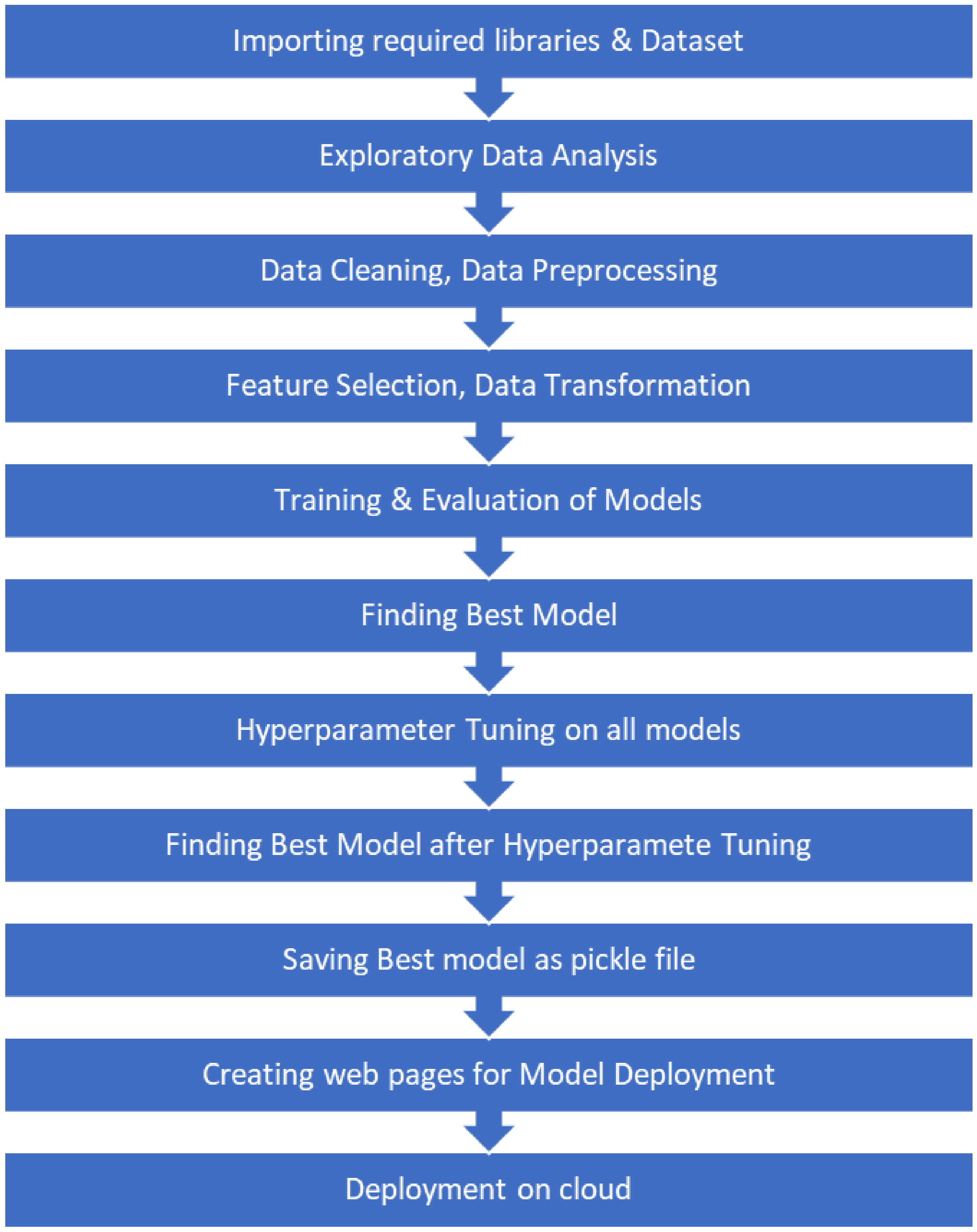
2.1 Why this Low-Level Design Document?

The goal of LLD or a low-level design document (LLD) is to give the internal logical design of the actual program code for Rental Bikeshare Demand Prediction. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

2.2 Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

3. Architecture


Importing required libraries & Dataset
Exploratory Data Analysis
Data Cleaning, Data Preprocessing
Feature Selection, Data Transformation
Training & Evaluation of Models
Finding Best Model
Hyperparameter Tuning on all models
Finding Best Model after Hyperparamete Tuning
Saving Best model as pickle file
Creating web pages for Model Deployment
Deployment on cloud

3.1 Architecture Design

This project is completely based on the life cycle of machine learning, where we will be predicting the count of rental bikes that would be needed based on certain conditions. The tools used in this project are Python, Pandas, NumPy, Matplotlib, Seaborn, Scikit learn, Flask for Backend, HTML, CSS & Java script was used for the Frontend part of the web application.

Using online platforms like GitHub to store projects to have an online backup of project work done so far. For deployment, railway.app (a platform like Heroku) was used.

3.2 Data requirement

Whenever we are working on any project the data is completely dependent on the requirement of the problem statement. For this project the problem statement was to create a Hyper tuned Regression machine learning model which can predict the total count of bikes rented at a particular day based on various parameters

3.3 Data Collection

The data (hour.csv file) which is used in this project was taken from UCI Machine Learning Repository.

Dataset link:
https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset

Bike-sharing rental process is highly correlated to the environmental and seasonal settings. For instance, weather conditions, humidity, day of week, season, hour of the day, etc. can affect the rental behaviors. The core data set is related to the two-year historical log corresponding to years 2011 and 2012 from Capital Bikeshare system, Washington D.C., USA which is publicly available at https://capitalbikeshare.com/system-data. We got the aggregated data (hour.csv from uci repository) having info about corresponding weather and seasonal information. The UCI repository mentions about this dataset that the Weather information was extracted from https://freemeteo.in/ .

3.4 About the Dataset

File Contain.

- Readme.txt
- hour.csv : bike sharing counts aggregated on hourly basis. Records of 17379 hours.

3.5 Data Description

The hour.csv file has the following fields

● instant: record index
● dteday : date
● season : season (1:springer, 2:summer, 3:fall, 4:winter)
● yr : year (0: 2011, 1:2012)
● mnth : month (1 to 12)
● hr : hour (0 to 23)
● holiday : weather day is holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)
● weekday : day of the week
● workingday : if day is neither weekend nor holiday is 1, otherwise it is 0.
● weathersit :
○ - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
○ - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
○ - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
○ - 4: Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog
● temp : Normalized temperature in Celsius. The values are divided to 41 (max)
● atemp: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
● hum: Normalized humidity. The values are divided to 100 (max)

● windspeed: Normalized wind speed. The values are divided to 67 (max) ● casual: count of casual users
● registered: count of registered users
● cnt: count of total rental bikes including both casual and registereds been used for this project.

3.6 Tools & Software Used

● Python has been used for this project.
● Python libraries such as NumPy, pandas, matplotlib, seaborn and scikit-learn (Used for implementation of machine learning algorithms)
● Jupyter for Exploratory Data Analysis and testing code, Visual studio code is used as an IDE for writing the code.
● HTML, CSS & Java Scripts are used for developing the front end of our web application.
● Flask is used for backend development.
● Github is used as the version control system.
● railway.app (a platform like Heroku) is used for deployment.

3.7 Loading Data

Local file system was used for loading the dataset (hour.csv) using read_csv function of Pandas Library.

3.8 Data Preprocessing

Have taken the hour.csv file as my dataset.
● All the necessary libraries were imported first such as Numpy, Pandas, Matplotlib, Seaborn.
● Checking the basic profile of the dataset. To get a better understanding of the dataset. ○ Using Info method ○ Using Describe method ○ Checking for unique values of each column.
● Checking for null values, There are no null values present in our dataset.
● The instant column is just acting as an index column for us, we can drop that column.

● We already have month, weekday, year data as separate columns hence dteday is not required. Hence dropping it.
● We will also be dropping casual and registered columns since the count column is already feature engineered with addition of casual and registered column in count column (casual+registered=cnt). Also it has been understood that increasing casual or registered users both will be profitable factor for the business. Hence 'cnt' column would be sufficient for now.
● Columns 'temp' and 'atemp' are highly correlated and their respective collinearities with 'cnt' column are also same. Hence dropping 'atemp' since feeling temperature can be relatively less accurate compared to temperature.
● Dropping 'holiday' column as it is highly correlated to 'workingday' column.
● Have converted the integer columns
['season', 'yr', 'mnth', 'hr' , 'holiday' , 'weekday' , 'workingday', 'weathersit'] into categorical columns.
● So, finally following are the predictor variables that will be used for prediction.
numerical_columns=['temp', 'hum', 'windspeed']
categorical_columns=['yr', 'mnth', 'hr', 'weekday', 'season', 'workingday', 'weathersit']
● Standard scaling has been done on numerical and categorical columns. Also, One Hot Encoding has been done on Categorical columns.

Now the dataset is ready and can be processed into the stage of modeling.

3.9 Modeling

● The data was split into 2 sets X and y. X contains all the columns except the target column in our case (Count), y contains only the Target column. ● Using train test split we first split the dataset into X_train,X_test, y_train, y_test .
● Following Regression Algorithms were used: CatBoost, Random Forest, Bagging, Decision Tree, Gradient Boost, KNeighbors, Linear Regression.

● After creating the above normal ones without any hyper tuned models we get CatBoost Regressor as the best model having the highest accuracy on testing data.
● We used Grid Search CV to find the best suiting parameters for all models and yet again the best performing model came out to be CatBoost Regression model.

3.10 UI Integration

As stated in the problem statement we have to create a user interface page where users can input their data and our machine learning model will give them the predicted result.

3.11 Data from the User

Data from the user is retrieved using the HTML & CSS template, and output is calculated using the app.py as our Flask application which is using our machine learning model to give the predicted result.

3.12 Data Validation

The data which is entered by the user is validated by the app.py file which is built using the python flask and then this data is transferred to our model.

3.13 Rendering the result

The result for our model is rendered at the HTML page. As we created an API for our predicted result page

4. Deployment

This model is deployed on railway.app (a platform like Heroku). The following are the steps to deploy this application:

Follow the steps to deploy on railway app using any of the instruction videos available on YouTube.
Instruction video: https://youtu.be/WzxEHzb2NzM?si=QIptg2bOKziR8L9