INDUSTRIAL TRAINING REPORT

On

Forecast hourly bike rental demand

Submitted by

JYOTI VARSHNEY
Roll No:181500302

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University
Mathura- 281406, INDIA
2020

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,
Mathura – 281406

Declaration

I hereby declare that the work which is being presented in the Industrial Training **"Forecast hourly bike rental demand",** in partial fulfillment of the requirements for Industrial Training viva voce, is an authentic record of my own work carried under the supervision of "organization name"**Internshala Training.**

Signature of Candidate: JYOTI VARSHNEY

Name of Candidate: JYOTI VARSHNEY

Roll. No. : 181500302

Course: B tech (CS)

Year: 2018-2022

Semester: 5

Student Information:

Name: Jyoti Varshney	University Roll. No:181500302
Mobile: 6398707690	Email:jyoti.varshney_cs18@gla.ac.in

Information about Industry/Organization:

Industry/Organization Name with full Address	Internshala Training (Online Mode)
Contact Person	Name & Designation: Servesh Agrawal Mobile/email:

Project Information:

Title Of Project/Training/Task	forecast hourly bike rental demand
Role & Responsibility	Full project
Technical Details	Hardware Requirements: i5 processor-based computer 2GB RAM (minimum) Software Requirements: Anaconda 3 (Jupyter Notebook) Technology Used: Machine Leaning(Linear Regression Model,Decision Tree) Data Science(Basics) Python modules(Numpy,pandas,matplotlib.pyplot, seaborn)
Training Implementation Details	Fully Implemented
Training Period	Start Date:1st June, 2020 End Date:13th July, 2020 Duration Of Training (In Weeks): 6 weeks

Summary of the Training Work:

About project:

It is Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able to rent a bike from one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying

mobility in a city

Datasets:

train.csv : Use this dataset to train the model. This file contains all the weather related features as well as the target variable "count". Train dataset is comprised of first 18 months.

test.csv : Use the trained model to predict the count of total rentals for each hour during the next 6 months.

In this project, we combine historical usage patterns with weather data in order to forecast hourly bike rental demand.

Training Certificate

INTERNSHALA TRAININGS

Certificate of Training

Jyoti Varshney

from **GLA University** has successfully completed a six weeks online training on **Data Science** from 1st June, 2020 to 13th July, 2020. The training consisted of Introduction to Data Science, Python for Data Science, Understanding the Statistics for Data Science and Predictive Modeling and Basics of Machine Learning modules. In the final assessment, Jyoti scored 87% marks. We wish Jyoti all the best for the future.

Sarvesh Agrawal
Founder & CEO, Internshala

Date of certification: 2020-07-11 Certificate no. : C2361A57-69F7-2944-4871-43FF808F0037

For certificate authentication, please visit https://trainings.internshala.com/verify_certificate

Acknowledgment

I owe to God who continues to look after me hearties of my flaws.

It is my pleasure to have the opportunity to extend my heartiest thank to everybody that has given a valuable contribution towards the successful completion of my project.

First of all, I want to express my deep sense of gratitude to SARVESH AGRAWAL SIR, who has encouraged me in the successful completion of my project.

I am highly obliged to my parents who supported me both financially and morally and were the source of great inspiration , without their help I was unable to complete this project.

JYOTI VARSHNEY

B. TECH (C.S.E)

CONTENTS

List of Figure :

Forecast hourly bike rental demand

1.1 INTRODUCTION

The present document is aimed to study the demand of Bike sharing systems through weather variables. The main objective has been to create a data based predictive model for the demand of Capital Bikeshare, one of the U.S.A.'s larges bicycle sharing systems. After having done some research on the problem and the methodologies that have been used to try to solve it before, the first step was to acquire the data on which the model was later built. This required downloading data from the Capital Bikeshare website, Washington D.C.'s local government website and designing a web scrapping tool to acquire weather data from an internet weather service website. This data was later aggregated into a more manageable single table.The present document is aimed to study the demand of Bike sharing systems through weather variables. The main objective has been to create a data based predictive model for the demand of Capital Bikeshare, one of the U.S.A.'s larges Bike sharing systems. After having done some research on the problem and the methodologies that have been used to try to solve it before, the first step was to acquire the data on which the model was later built.

In this project a demand forecast model was developed, which can be easily trained and deployed. It can be used by bike sharing services to try to solve one of their major causes of customer loss (the lack of Bike or parking spots at a given station) by either doing dynamic repositioning or designing an incentive based system.

It is Bike sharing systems are a means of renting bicycles where the
process of obtaining membership, rental, and bike return is automated via
a network of kiosk locations throughout a city. Using these systems, people are able to rent a bike from one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.
The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded.

Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city.

As a first approach to solving this problem, a data based forecast method is going to be developed to predict the total demand of bicycles for a bicycle sharing system for each hour, based on meteorological data. The model could be later scaled to fit evert single station. For this project Washington D.C.'s Capital Bikeshare Systems data is going to be used. A machine learning method has been chosen to solve the problem, namely the random forest algorithm which is one of the most successful techniques on the field.

Motivation

Heavy street traffic in busy cities and a desire for an environmentally friendly form of transportation make biking an attractive alternative to traveling by car. A limited supply of bikes, increasing demand for bikes, and cost of storage and relocation of bikes serve as motivation for forecasting the demand of bikes.

1.2 Problem Statement

In this project, you are asked to combine historical usage patterns with

weather data in order to forecast hourly bike rental demand.

1.3 Objective

The aim of this project is to develop a demand forecast model for the Capital Bikeshare system in Washington D.C. This is a model that for every day and hour, given some meteorological variables and whether the day is a holiday or not, attempts to predict the total number of Bike that will be rented that hour.

In order to build the model, these steps have to be followed:

- Research methodologies that have been used to analyze problems of a similar nature.
- Acquire the required data to build the model.
- Analyze the data and find out which variables drive the demand.
- Design a predictive model to forecast the demand.
- Implement and tune the model to better fit the data.
- Draw conclusions from the previous work.

1.4 Definition, Acronyms and Abbreviations:

Python: Python can be used on a server to create web applications. Python can be used alongside software to create workflows .Python can connect to database systems. It can also read and modify files. Python can be used to handle big data and perform complex mathematics. Python can be used for rapid prototyping, or for production-ready software development. Python can be used on a server to create web applications. Python can be used alongside software to create workflows. Python can connect to database systems. It can also read and modify files. Python can be used to handle big data and perform complex mathematics. Python can be used for rapid Prototyping, or for production-ready software development

Anaconda: Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012.

Numpy: It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several

other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

Pandas: pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

Matplotlib: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Literature Survey

As per various literature surveys it is found that for implementing this project four basic steps are required to be performed.

Bikesharing systems demand has been modeled in the past as part of systems that optimize fleet repositioning and also as a way to study the way bikesharing systems' users behave. One of the first attempts to model a bikesharing system's demand was made by Borgnat, Abry, & Rouquier (2009) as a part of their study of Lyon's Vélo'V. Their forecast model combined statistical models to predict de cyclical fluctuations with linear regression. Barcelona's Bicing has been studied at least twice even though it only offers real time data and its data platform has to be continually scrapped to obtain a certain volume of data. It was first studied by Froehlich, Neumann, & Oliver (2009) who used Bicing data to study the city's mobility. They used predicting methods of increasing complexity: last value, historic mean, historic trend and Bayesian networks. Kaltenbrunner, Meza, Grivolla, Codina, & Banchs (2010) also explored Bicing's patterns, to predict the number of bicycles on a station they used some simple models as well as some time series analysis tools like the Auto-Regressive Moving Average (ARMA). Another case of a city being studied through its bikeshare systems is Vogel, Greiser, & Mattfeld's (2011) study of Vienna's Citybike Wien's patterns through cluster analysis. K-means clustering, expectation maximization and sequential Information-Bottleneck were the algorithms of choice. Finally, Caggiani & Ottomanelli (2013) designed a model for fleet repositioning based on simulations. To forecast the demand they used Neural Networks.. Lastly, some previous research with a similar project scope is discussed, focusing on the used methodology and the results achieved.

2.1 Development of a station-level demand prediction and visualization tool to support bike-sharing systems' operators

Bike-sharing systems operate in a number of cities around the world, aiming to promote sustainable urban mobility. Successful management of these systems is to a large extent linked to the optimal distribution of bicycles, which implies the accurate prediction of demand for rentals and returns at each station within the day. For this purpose, a tool for predicting bike demand for rentals and returns and visualizing the

results has been developed and is presented in the present paper. Different predictive models based on machine learning regression algorithms are trained and evaluated. The tool is tested using data from the bike-sharing system that operates in the city of Thessaloniki, Greece for which the results indicate that the tested system's utilization is highly correlated to the location and spatial characteristics of a station, as well as the season of the year and time of day. The proposed machine learning algorithms use custom engineered features to learn those correlations and achieve the highest possible performance.

2.2 Methods and results of similar projects

Bike-sharing companies have the collective goal of meeting the needs of the final customer, by supplying bicycles at the right place, time and price (Helms, Ettkin, and Chapman 2000). Accurate and effective forecasting of future demand is essential for making supply chain decisions. Yet, the complexity and uncertainty existing around demand makes forecasting a challenge. Since customer demand is influenced

by a variety of factors, understanding relationships between these factors and bike sharing demand would enable companies to improve the forecasts (Chopra and Meindl 2014). Due to its importance for companies' success, other case studies related to bike-sharing demand forecasting already exist. We have summarized the most relevant studies in Table 2 with focus on the modeling methodologies, the valuation method and the results.

Table 2: Overview of similar demand forecasting case studies

Title	Modeling Methodologies	Valuation	Conclusion
Forecasting Utilization in City Bike-Share Program (a)	(1) Neural Network (2) Poisson Regression (3) Markov Model (4) Mean Value Benchmark	RMSLE	Neural Network is found to perform the best which is 0.49.
Predicting Capital Bikeshare Demand in R (b)	(1) Regression (2) Generalized Boosted Models	RMSLE	The GBM model clearly performed better than our linear regression model
Bike Share Demand Prediction using RandomForests (c)	(1) Random Forest (2) Enhancing RM model using TuneRF (3) Conditional Inference Trees (4) Generalized Boosted Models	RMSLE	Random Forest with TuneRF performance is 0.49
Research on Antecedents and Consequences of Factors Affecting the Bike Sharing System (d)	(1) Multiple linear regression (2) Poisson and Topology method	P-Values	Temperature that people feel is most important determinant
Forecasting Bike Rental Demand (e)	(1) Basic Linear Regression (2) Generalized Linear Models with Elastic Net Regularization (3) Generalized Boosted Models (4) Principal Component Regression (5) Support Vector Regression (6) Random Forest (7) Conditional Inference Trees	RMSLE	Two of the Tree based models are found to perform the best: • CTree: 0.46 • Random Forest: 0.50
Demand Prediction of Bicycle Sharing Systems (f)	(1) Ridge Regression (2) Support Vector Regression (3) Random Forest (4) Gradient Boosted Tree	RMSLE	Random forest is found to perform best
Bike Sharing Demand: Forecast use of a city bikeshare system (g)	(1) Gradient Boosted Decision Trees (GBDT)	RMSLE	RMSLE value of GBDT Loop is 0.5683
Forecasting Bike Rental Demand Using New York Citi Bike Data (h)	(1) Linear Regression (2) Neural Network (3) Decision Tree (4) Random Forest	RMSLE	Random forest model has shown relative best performance
Prediction of Bike Sharing Demand (i)	(1) SVM (2) Neural Network (3) Poisson Regression (4) Random Forest (5) Extra Trees Regressor (6) GBM (7) Linear Combination Model (8) Discriminating Linear Combination Model	RMSE	Linear Combination model and Discriminating Linear Combination model are good models for predicting bike sharing demand with RMSE being close to 0.36

Software Requirement Analysis

System Analysis is a detailed study of the various operations performed by a system and their relationship within and outside the system .It is a systematic technique that defines goals and objectives the goal of the development is to deliver the system in the line with the user's requirements, and analysis is this process.

3.1 Specific Requirement

As the project is developed in python, we have used Anaconda as IDE and some csv file used to data modeling .

1. train.csv : Use this dataset to train the model. This file contains all the

weather related features as well as the target variable "count". Train

dataset is comprised of first 18 months.

2. test.csv : Use the trained model to predict the count of total rentals for

each hour during the next 6 months.

These CSV files contain these kind of data which uses for a purpose.

3.1.1- Trip history data: Whenever a rental occurs in the Capital Bikeshare system, their software records basic data about the trip in order to monitor the system. The data is stored, and can be downloaded from the Capital Bikeshare official website. Respecting data privacy policies, private data including member names is not included in the files. The data available when the research was conducted belonged to the 2010Q4-2014Q4 period. The data came in a .csv format and separated in files, each containing data for one quarter of a year.
Those files consisted of a list of trips featuring the following variables:

- Duration: Duration of trip
- Start date: Start date and time
- End date: End date and time
- Start station: Starting station name and number
- End station: Ending station name and number
- Bike #: ID number of bike used for the trip
- Member Type: Whether the user was a Registered or Casual member.

The data was manually downloaded from the website and manipulated using the PostgreSQL database management system to summarize the individual trip data into hourly counts, using the start date and hour as reference. The formats of the different files differed a bit in some cases and were treated separately. At the end all data was aggregated into one table containing data from years 2011-2014. Data from 2010Q4 was not used because it was suspected of containing irregularities, since it was the first period for which data was made available.

3.1.2- Holiday data: The website of Washington D.C.'s local government hosts the holiday schedule for the city . Schedules for years 2011-2014 were manually copied and pasted on a text file, which was later processed using PostgreSQL. The data contained the date and the reason for the holiday, which was disregarded. This data translated into a binary variable, being 1 in case the day was a holiday and 0 otherwise.

3.1.3- Weather data: Weather Underground is an internet weather service that offers historical data as well as weather forecasts for locations all around the world. Hourly weather data for individual days can be easily consulted on the website, however, data for 1461 days was needed so manual download was not an option.

Demand forecast model for a bicycle sharing service 14 The data was downloaded using the web scrapping technique described in the methodologies chapter. Historical hourly weather data from every day in years 2011-2014 was downloaded and saved into a text file. The data was imported to PostgreSQL and properly structured. From the downloaded data, the following variables were kept:

- Hour: Time at which the data was recorded.
- Temperature: Temperature measured in Celsius degrees.
- Dew Point: temperature at which the water vapor in a sample of air at constant pressure condenses into liquid water at the same rate at which it evaporates, measured in Celsius degrees.
- Humidity: amount of water vapor in the air relative to the maximum for that temperature, expressed as a percent.
- Mean sea level pressure: Atmospheric pressure at sea level. Measured in hPa.
- Visibility: Distance at which an object or light can be clearly discerned, measured in Km.
- Wind Direction: Direction from which the wind originates. This information is given as a class, featuring 16 different directions (N, NNE, NE, ENE, E, ESE …), plus two other categories (calm and variable).
- Wind speed: Flow velocity of the wind, measured in Km/h. Precipitation: Amount of water that rained. Measured in mm. Conditions: Description of the weather state (i.e. foggy, overcast, etc.), divided into 25 different classes.

3.1.4- Resulting dataset: The above data was joined by date and hour, and a table was created with the following columns:

Data Dictionary

Here is the description of all the variables :

Variable	Definition
datetime	hourly date + timestamp
season	Type of season (1 = spring, 2 = summer, 3 = fall, 4 = winter)
holiday	whether the day is considered a holiday
workingday	whether the day is neither a weekend nor holiday
weather	weather
temp	temperature in Celsius
atemp	"feels like" temperature in Celsius
humidity	relative humidity
windspeed	wind speed
casual	number of non-registered user rentals initiated
registered	number of registered user rentals initiated
count	number of total rentals

Total Count Out of a total of 35,064 observations, weather data was missing for 16 of them. Since it represented such a low percentage of the dataset, those hours were excluded from the study. Demand forecast model for a bicycle sharing service 15 SQL and Python codes.

Hardware Interfaces

1. Processor : Intel Pentium processor with minimum 2.9 GHz speed.
2. RAM : Minimum 2 GB.
3. Hard Disk : Minimum 500 GB

Software Interfaces

1. Anaconda
2. Database Storage : XML , CSV files
3. Operating System : Windows 8

3.2 Non Functional Requirement:-

1 Performance:-

The System should have enough memory and Software to run this application so XML and CSV files will process easily

2 Portability:-

the System work across the anaconda application and that should

Be compatible with desktop & tablets

3.3 Model Description:-

There are two model used for it so first we use linear Regression and then Decision tree.

1 Linear Regression:- In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.[3] Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

2 Decision Tree Model:- In computational complexity the **decision tree model** is the model of computation in which an algorithm is considered to be basically a decision tree, i.e., a sequence of branching operations based on comparisons of some quantities, the comparisons being assigned unit computational cost.

The branching operations are called "tests" or "queries". In this setting the algorithm in question may be viewed as a computation of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$

where the input is a series of queries and the output is the final decision. Each query may be dependent on previous queries.

Several variants of decision tree models have been introduced, depending on the complexity of the operations allowed in the computation of a single comparison and the way of branching.

Decision trees models are instrumental in establishing lower bounds for complexity theory for certain classes of computational problems and algorithms. The computational complexity of a problem or an algorithm expressed in terms of the decision tree model is called its **decision tree complexity** or **query complexity**.

Proposed Work & Implementation

1. We will first explore the dataset provided
2. We will create models to predict the hourly bike rental demand.
3. We will also make predictions for hourly demand in the test set

importing the libraries:

```
import numpy as np

import pandas as pd

from datetime import datetime

from datetime import date

import calendar

import matplotlib.pyplot as plt

import seaborn as sn

%matplotlib inline
```

Loading the dataset

```
train = pd.read_csv('train.csv')

test = pd.read_csv('test.csv')
```

There are 12 columns in train dataset, whereas 11 in the test dataset. The missing column in the test dataset is the target variable and we will train our model to predict that variable

```
train.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 0:00:00	1	0	0	1	9.84	14.395	81.0	0.0	3	13	16
1	2011-01-01 1:00:00	1	0	0	1	9.02	13.635	80.0	0.0	8	32	40
2	2011-01-01 2:00:00	1	0	0	1	9.02	13.635	80.0	0.0	5	27	32
3	2011-01-01 3:00:00	1	0	0	1	9.84	14.395	75.0	0.0	3	10	13
4	2011-01-01 4:00:00	1	0	0	1	9.84	14.395	75.0	0.0	0	1	1

test.head()

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered
0	2012-06-30 1:00:00	3	0	0	3	26.24	28.790	89.0	15.0013	3	55
1	2012-06-30 2:00:00	3	0	0	2	26.24	28.790	89.0	0.0000	7	54
2	2012-06-30 3:00:00	3	0	0	2	26.24	28.790	89.0	0.0000	3	20
3	2012-06-30 4:00:00	3	0	0	2	25.42	27.275	94.0	0.0000	3	15
4	2012-06-30 5:00:00	3	0	0	1	26.24	28.790	89.0	11.0014	3	7

We can infer that "count" is our target variable as it is missing from the test dataset..

Univariate Analysis

```
# distribution of count variable
sn.distplot(train["count"])
```

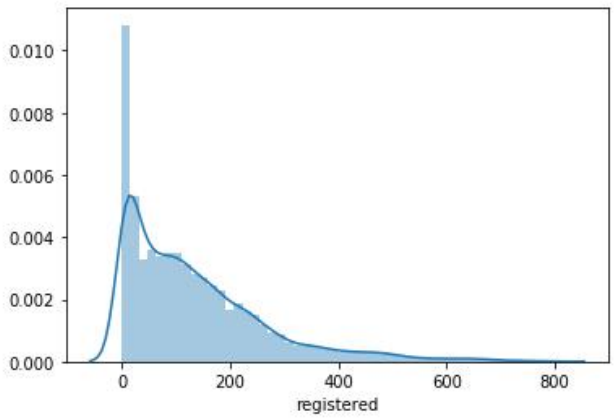

Fig 4.1- distribution of count variable

The distribution is skewed towards right and hence we can take log of the variable and see if the distribution becomes normal.

sn.distplot(np.log(train["count"]))

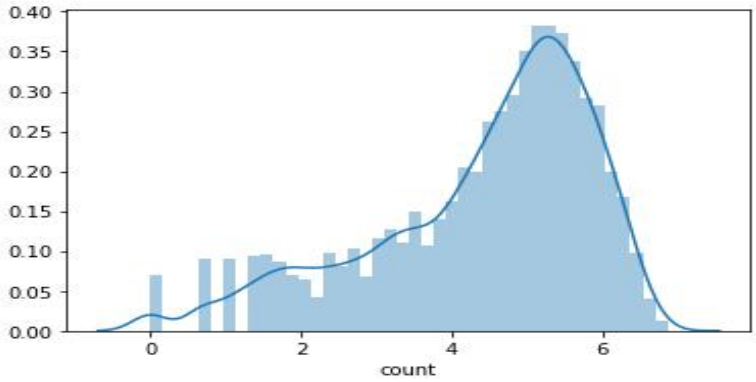

Fig 4.2- distribution of count variable take log

Now the distribution looks less skewed. Let's now explore the variables to have a better understanding of the dataset. We will first explore the variables individually using univariate analysis, then we will look at the relation between various independent variables and the target variable. We will also look at the correlation plot to see which variables affects the target variable most.

check the number of registered user rentals initiated

sn.distplot(train["registered"]).

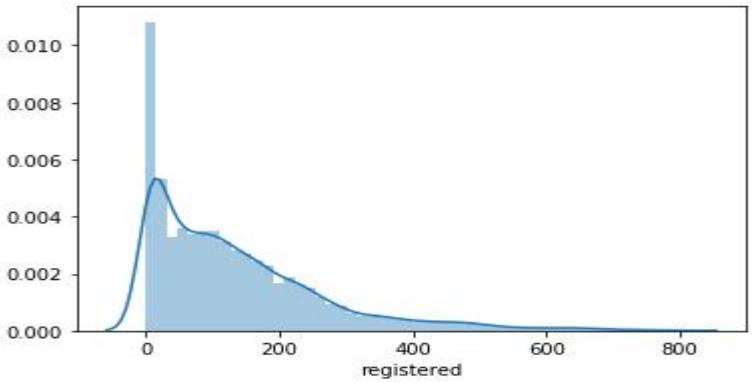

Fig 4.3- distribution of registration variable

We can see that most of the registered rentals lies in the range of 0 to 200. The registered users at a particular time step will always be less than or equal to the demand (count) of that particular timestep.

Bivariate Analysis

```
# looking at the correlation between numerical variables
corr = train[["temp","atemp","casual","registered","humidity","windspeed","count"]].corr()
mask = np.array(corr)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sn.heatmap(corr, mask=mask,vmax=.9, square=True,annot=True, cmap="YlGnBu")
```

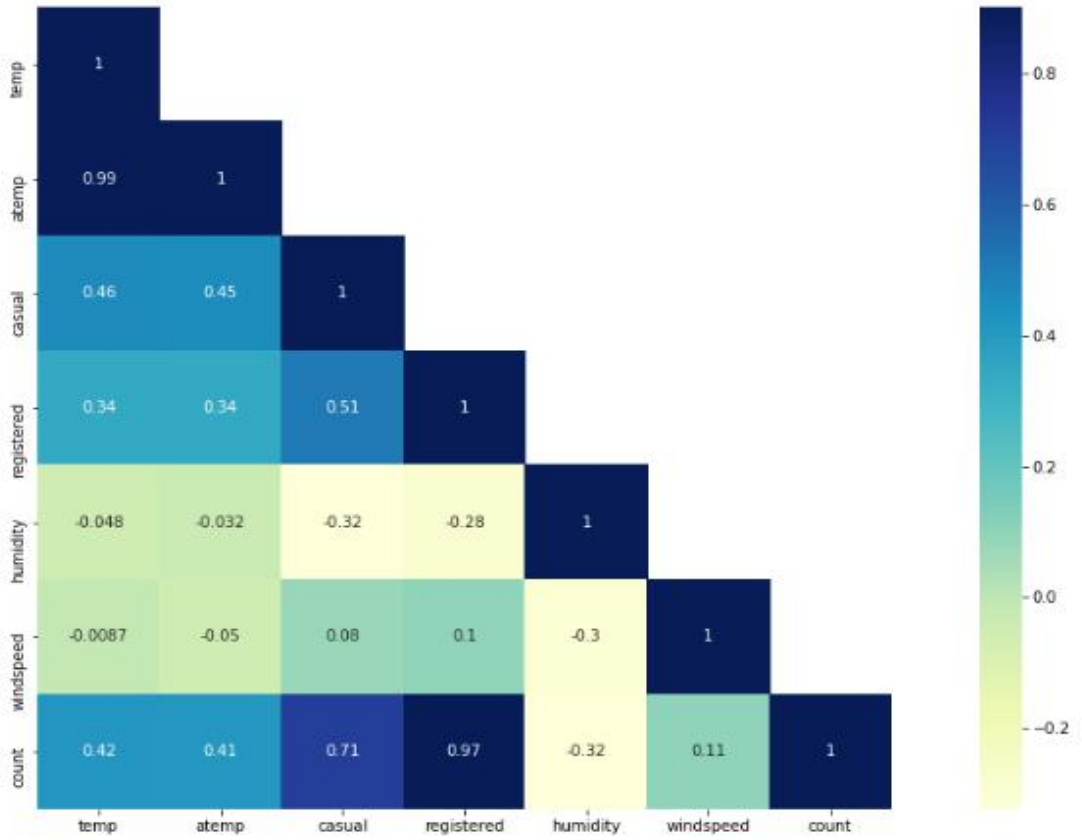

Fig 4.4 - Correlation between numerical variables

Some of the inferences from the above correlation map are:

1- temp and humidity features has got positive and negative correlation with count respectively.Although the correlation between them are not very prominent still the count variable has got little dependency on "temp" and "humidity".

2- windspeed will not be really useful numerical feature and it is visible from it correlation value with "count"

3- Since "atemp" and "temp" has got strong correlation with each other, during model building any one of the variable has to be dropped since they will exhibit multi-collinearity in the data.

Before building the model, let's check if there are any missing values in the dataset

```
# looking for missing values in the datasaet
train.isnull().sum().
```

```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

There are no missing values in the train dataset.
We can extract the date, hour, month.from train and test datasets.

```
# extracting date, hour and month from the datetime
train["date"] = train.datetime.apply(lambda x : x.split()[0])
train["hour"] = train.datetime.apply(lambda x : x.split()[1].split(":")[0])
train["month"] = train.date.apply(lambda dateString : datetime.strptime(dateString,"%Y-%m-%d").month)

test["date"] = test.datetime.apply(lambda x : x.split()[0])
test["hour"] = test.datetime.apply(lambda x : x.split()[1].split(":")[0])
test["month"] = test.date.apply(lambda dateString : datetime.strptime(dateString,"%Y-%m-%d").month)
```

- We will drop the datetime, date variable as we have already extracted features from these variables.
- We will also drop the atemp variable as we saw that it is highly correlated with the temp variable.

```
train = train.drop(['datetime','date', 'atemp'],axis=1)
test = test.drop(['datetime','date', 'atemp'], axis=1)
training = training.drop(['datetime','date', 'atemp'],axis=1)
validation = validation.drop(['datetime','date', 'atemp'],axis=1)
```

Model Building

Linear Regression Model

```
# initialize the linear regression model

lModel = LinearRegression()
```

We will remove the target variable from both the training and validation set and keep it in a separate variable. We saw in the visualization part that the target variable is right skewed, so we will take its log as well before feeding it to the model.

```
X_train = training.drop('count', 1)
y_train = np.log(training['count'])
X_val = validation.drop('count', 1)
y_val = np.log(validation['count'])
```

```
# checking the shape of X_train, y_train, X_val and y_val
X_train.shape, y_train.shape, X_val.shape, y_val.shape
```

```
((10774, 11), (10774,), (2206, 11), (2206,))
```

```
# fitting the model on X_train and y_train
lModel.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Now we have a trained linear regression model with us. We will now make prediction on the X_val set and check the performance of our model. Since the evaluation metric for this problem is RMSLE, we will define a model which will return the RMSLE score.

```
# making prediction on validation set
prediction = lModel.predict(X_val)
```

```
# defining a function which will return the rmsle score
def rmsle(y, y_):
    y = np.exp(y),   # taking the exponential as we took the log of target variable
    y_ = np.exp(y_)
    log1 = np.nan_to_num(np.array([np.log(v + 1) for v in y]))
    log2 = np.nan_to_num(np.array([np.log(v + 1) for v in y_]))
    calc = (log1 - log2) ** 2
    return np.sqrt(np.mean(calc))
```

Let's now calculate the rmsle value of the predictions

```
rmsle(y_val,prediction)
```

```
0.8875379204281795
```

We got a rmsle value of 0.8875 on the validation set.

Decision Tree

defining a decision tree model with a depth of 5. You can further tune the hyperparameters to improve the score

dt_reg = DecisionTreeRegressor(max_depth=5)

fit the decision tree model .

```
dt_reg.fit(X_train, y_train)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=5, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=None, splitter='best')
```

Its time to make prediction on the validation set using the trained decision tree model.

predict = dt_reg.predict(X_val)

```
# calculating rmsle of the predicted values
rmsle(y_val, predict)
```

```
0.1710226739944699
```

The rmsle value has decreased to 0.171. This is a decent score.

now make predictions for the test dataset which you can submit in the excel sheet provided to you to generate your score.

```
test_prediction = dt_reg.predict(test)
```

These are the log values and we have to convert them back to the original scale.

```
final_prediction = np.exp(test_prediction)
```

Finally, we will save these predictions into a csv file. You can then open this csv file and
copy paste the predictions on the provided excel file to generate score.

submission = pd.DataFrame()

```
# creating a count column and saving the predictions in it
submission['count'] = final_prediction
```

```
submission.to_csv('submission.csv', header=True, index=False)
```

Predictions on test data set

	A	B
1	datetime	count
2	2012-06-30 1:00:00	67.3375
3	2012-06-30 2:00:00	67.3375
4	2012-06-30 3:00:00	22.2185
5	2012-06-30 4:00:00	14.5184
6	2012-06-30 5:00:00	8.65276
7	2012-06-30 6:00:00	46.0346
8	2012-06-30 7:00:00	94.2149
9	2012-06-30 8:00:00	206.334
10	2012-06-30 9:00:00	289.82
11	2012-06-30 10:00:0(	386.357
12	2012-06-30 11:00:0(	386.357
13	2012-06-30 12:00:0(	386.357
14	2012-06-30 13:00:0(	386.357
15	2012-06-30 14:00:0(	386.357
16	2012-06-30 15:00:0(	386.357
17	2012-06-30 16:00:0(	386.357
18	2012-06-30 17:00:0(	386.357

Fig 4.5- Predication on test data sets

Software Design

5.1 Flow Chart

A **flowchart** is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields

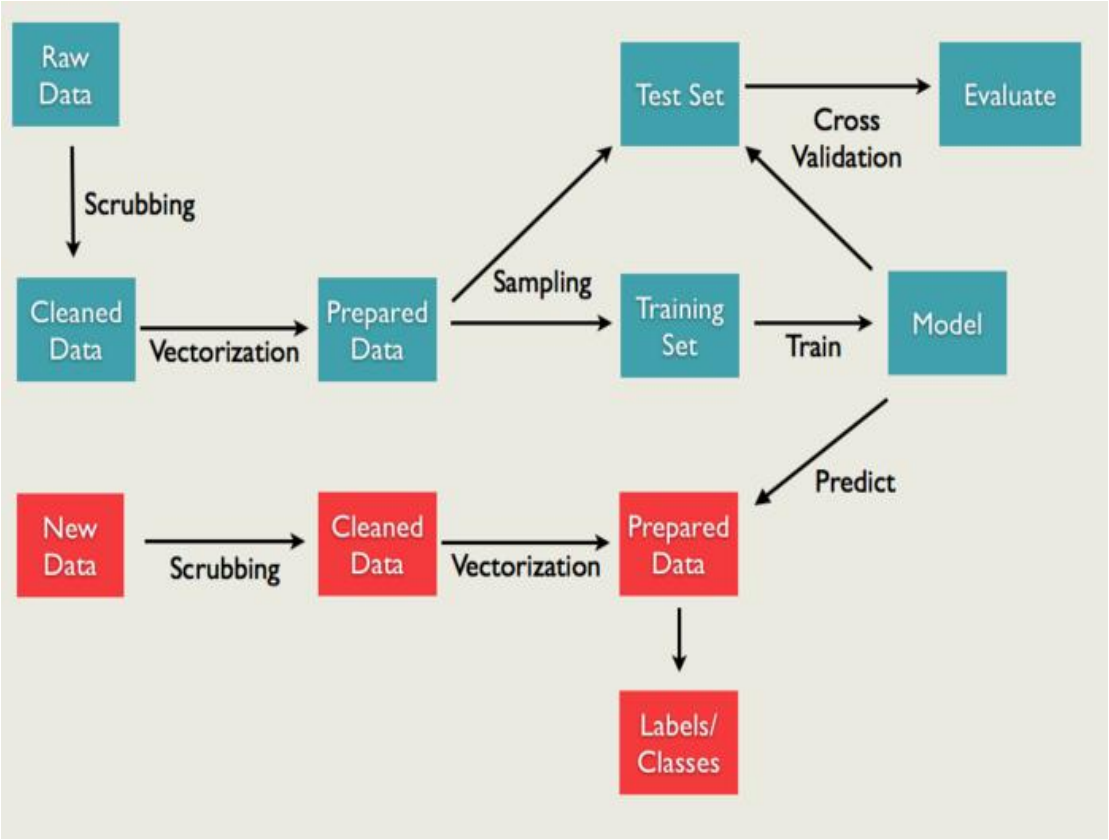

Figure 5.1: Flow Chart

5.2 Use Case Diagram

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

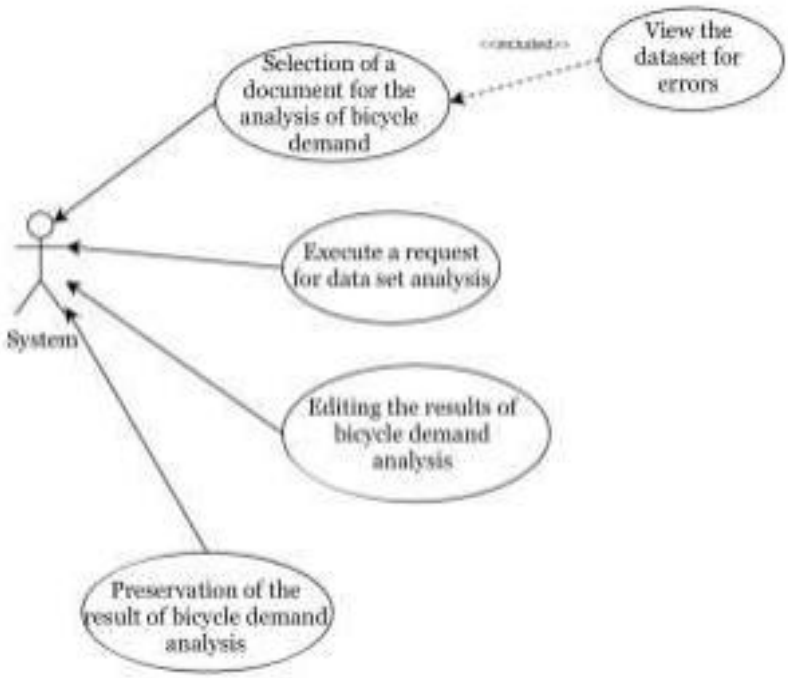

Fig. 5.2 Use Case Diagram

5.3 Sequence diagram

The main purpose of the activity diagram is to obtain the dynamics of system behaviour. Activity is part of the functioning of the system. The only limitation of the activity chart is that it does not display messages that are created and received from one part of a functioning system to another

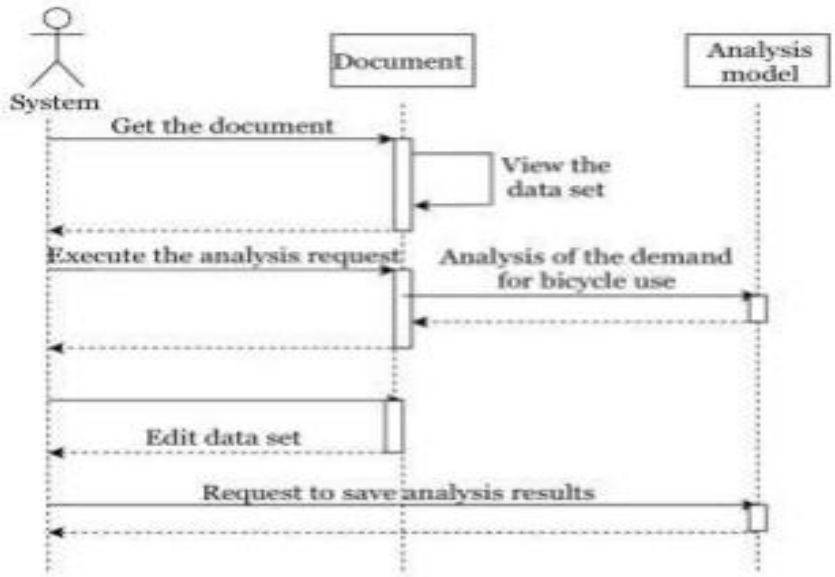

Fig. 5.3 Sequence diagram

CONCLUSION

In order to forecast the demand for bicycle sharing in Smart City, the following tasks have been considered and solved in this paper. As an example, the well-known Bike Sharing dataset was selected. This dataset is available from the UCI Machine Learning Repository. On the example of this dataset, a set of problems could be solved. To forecast the demand for bicycle sharing in Lviv Smart City it was recommended to use regression models of data analysis. To demonstrate possible cases and the work of the main processes of information systems, UML diagrams were created. To determine the peaks in demand for bicycles in a certain period of time, it was proposed to use regression models of data analysis. The dataset transformations have been made, such as attributes renaming, types changing, and categories definition. The proposed decision trees were recommended for modeling new datasets in such a Smart City like Lviv. The Regressor Decision Tree better predicts the demand for bicycles compared to linear regression

REFERENCES

1-www.tutorialspoint.com

2- www.kaggle.com

3- Google