**Project name: Architectural Support for Cloud Computing, Programming Project**

**Mohan Krishna Sunkara**

**UIN: 01206224**

**Objective:**

The goal of your programming project is to simulate a Vehicular Cloud built on top of vehicles on a highway. The challenge facing the implementation of the Vehicular Cloud is to minimize job completion time in the face of the dynamically changing resources.

**Introduction:**

Vehicular Networks (VANETs) offer a range of opportunities for urban monitoring and data sharing on various aspects of the traffic. Vehicular networks do not have common constraints of Wireless Sensor Network ([WSN)s](https://www.sciencedirect.com/topics/computer-science/wireless-sensor-network), such as energy, bandwidth, and memory constraints, which allows for more accurate sensing and a larger amount of data to be collected. Furthermore, vehicles can contain sensors that are not commonly available in portable devices used in PSNs.

Another important aspect of VANETs is the coverage. Vehicles move through the whole city using streets and avenues. Because of this spread of mobility, vehicular networks can capture the details for several cities. All these features make VANETs an important data source that can complement data gained from PSNs, in order to better understand the urban phenomena.

[Vehicular applications](https://www.sciencedirect.com/topics/computer-science/vehicular-application) can be used in numerous scenarios. For instance, in VANETs there are diverse situations to be monitored, such as potholes, traffic jams, car accidents, and the presence of animals on the road. Thus, in this section, we present studies that focus on three main issues: monitoring general traffic events; the use of data of VANETs to study people’s routines; and the study of traffic jams. We also discuss various challenges associated with these issues.

Assumptions:

1. Job size is 10 MB
2. Job download time is 1 second
3. Job upload time is 1 second
4. Maximum highway speed 60 kilometers per hour or 60000 meters per hour
5. Highway patrol police make sure that cars entered are not jammed in traffic on highway
6. Length of the highway is 11 kilometers.
7. Access points are placed for every 2000 meters
8. Coverage area of each access point is 100 meters

Strategy 1:

First strategy was to drop the jobs which has more job completion time than the residency time of incoming vehicles. For example, if there are 2 incoming vehicles which has residency time of about 200 seconds, 300 seconds but job created at that moment has job completion time of 400 then job is dropped.

Assumptions for strategy 1 are:

1. Vehicle arrival time is selected randomly from 0 to 3600 seconds
2. Incoming speed of the vehicle is normally distributed with mean of 30 and standard deviation of 20
3. Job Processing time is also normally distributed with mean of 400 and standard deviation of 40
4. Job size is assumed to be 10 MB
5. Vehicle number is assumed to be size of 8 with digits and characters

Graph of residency time and total Job completion time

Chart, scatter chart

Description automatically generated

Strategy 2:

Instead of dropping jobs push jobs into stack which has high job completion time. Whenever vehicle comes calculate residency time of the vehicle and check whether is there any jobs that has total job completion time less than residency time of vehicle and if yes then assign that job on that stack to that vehicle If no such job is found then assign new job to that vehicle if total job completion time is lesser than residency time of the vehicle.

Assumptions for strategy 2 are:

1. Vehicle arrival time is selected randomly from 0 to 3600 seconds
2. Incoming speed of the vehicle is normally distributed with mean of 30 and standard deviation of 20
3. Job Processing time is also normally distributed with mean of 400 and standard deviation of 40
4. Job size is assumed to be 10 MB
5. Vehicle number is assumed to be size of 8 with digits and characters

Graph of residency time and total Job completion time

A picture containing chart

Description automatically generated

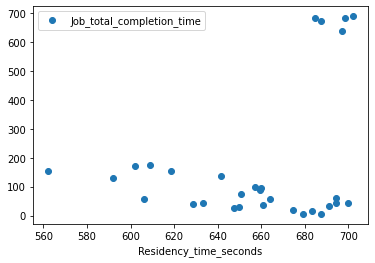
Strategy 3:

Strategy 3 is same as strategy 2 but you are dividing bigger jobs into 2 jobs and pushing it on stack to make sure job completion time is less than incoming vehicle residency time. Whenever vehicle comes calculate residency time of the vehicle and check whether is there any jobs that has total job completion time less than residency time of vehicle and if yes then assign that job on that stack to that vehicle If no such job is found then assign new job to that vehicle if total job completion time is lesser than residency time of the vehicle

Assumptions for strategy 3 are:

1. Vehicle arrival time is selected randomly from 0 to 3600 seconds
2. Incoming speed of the vehicle is normally distributed with mean of 30 and standard deviation of 20
3. Job Processing time is also normally distributed with mean of 400 and standard deviation of 40
4. Job size is assumed to be 10 MB
5. Vehicle number is assumed to be size of 8 with digits and characters

Graph of residency time and total Job completion time



You can see that for strategy 3 job completion time has decreased a lot compared to strategy 1 and strategy 2.

Strategy 4:

Using AI to choose which car to migrate. Suppose if many cars come at same time or we have to migrate VM to different cars. Again lets suppose all of them have same speed and same residency time. Then we can use AI to choose best car based on it specifications because few cars also break down during highway due to poor enginer performance. Some cars also will meet with accident when break failure happens due to its poor maintanance or if it car was highly used by its previous owners. So when ever break down or accident or enginer failure or break failure happens not only car but also job assigned to it is also lost. So using AI to choose best car to migrate VM or while assigning jobs to the car is the best thing.

For this sake, I have collected dataset and built model which gives performance score of the car. 0 for poor performance car and as the score increases implies performance also increases. We have used several methods while training this dataset and choose the method for the model which gives best accuracy.

The dataset has following columns

* "Name": Name of car for example: Maruti Wagon R, Honda etc
* "Location": Location of the car
* "Year": Purchased year of the car
* "Kilometers": Kilometers of the car used
* "Fuel\_Type": Type of the fuel of the car such as "petrol", "Diesel" etc
* "Transmission": Transmission type of the car whether it is manual or automatic "Owner\_Type": Owner type of the car whether it is first ownered car or second ownered car
* "Mileage": Mileage of the car
* "Engine": Engine power of the car for example 950 CC, 1000 CC etc
* "power":Power of the car for example 60 bhp, 100 bhp
* "Seats": Number of seats in a car
* "New\_Price": New price of the car

EDA process:

* Data Cleaning & Data Correction
* Feature Engineering & Visualization
* Model Training and Model Evaluation
* Cross Validation

Model evaluation:

For linear regression:

Accuracy on Training set: 0.6779073313110087

Accuracy on Testing set: 0.6861011272608684

For RandomForestRegressor:

Accuracy on Training set: 0.9837027978796756

Accuracy on Testing set: 0.9105006903903593

Based on above values we found that RandomForestRegressor is better performing that LinearRegression. So we use RandomForestRegressor for our model.

For RandomForestRegressor:

Error Table:

Mean Absolute Error : 1.528673556795046

Mean Squared Error : 10.366621093918717

Root Mean Squared Error : 3.2197237604985176

R Squared Error : 0.9076863897102971

On Conclusion, we can say that we can use this model built to choose best performing car to migrate VM or while assigning jobs to the car. So that whenever break down or accident or engine failure or break failure happens we can avoid not only car damage but also we can avoid losing of the job assigned to it.