A Project Report
on

Traffic Flow Prediction Using Machine Learning Methods

submitted in partial fulfilment of the requirements for the award of the

Degree
of

Bachelor of Technology

in
Computer Science and Engineering
by

P. Hemanth Naik (1506085)
J. Saibaba (1506099)
S. Venkatesh (1506100)
P. Naveen Kumar (1506101)

Under the supervision of

Prof. A. K. Dudyala

National Institute of Technology Patna
Department of Computer Science and Engineering

DECLARATION

We hereby declare that this project work entitled,“ **Traffic Flow Prediction Using Machine Learning Methods “** has been carried out in the department of **Computer Science and Engineering** of **National Institute of Technology, Patna** for the **7 th semester** minor project under the guidance of **Prof. A. k. Dudyala**, Department of Computer Science and Engineering, NIT Patna. No part of this work has been copied from any other project work/research paper/journal, etc or submitted for the award of degree or diploma to any institute.

P. Hemanth Naik	**1506085**
J.Saibaba	**1506099**
S. Venkatesh	**1506100**
P. Naveen Kumar	**1506101**

Place Patna
Date 11th December 2018

CERTIFICATE

Department of Computer Science and Engineering
National Institute of Technology Patna

This is to certify that **Hemanth Naik (1506085), Saibaba (1506099)**, **Venkatesh (1506100)** and **Naveen Kumar (1506101)** have carried out the project entitled, "**Traffic Flow Prediction Using Machine Learning Methods"** as their 7th Semester Minor Project under the guidance of **Prof. A. K. Dudyala**, Department of Computer Science and Engineering, NIT Patna. This project is bonafide work done by them for the fulfilment of the requirements for the award of degree of **B.Tech** in ***Computer Science and Engineering***.

Prof. A. K. Dudyala **Dr. Prabhat Kumar**

(supervisor) **(H.O.D)**

ACKNOWLEDGEMENT

This Minor Project of 7th semester was a great opportunity for learning latest technologies and self-development of practical knowledge. We do consider ourselves very privileged to have so many wonderful people lead us through in completion of this project.

We would like to express our deepest appreciation to all those who have provided us the possibility to complete this Project. A special gratitude, we give to our project mentor, **Prof. A. K. Dudyala** whose contribution in simulating suggestions and encouragement helped us to coordinate this project especially in writing the report.

A sincere thanks of gratitude to those who extended their valuable knowledge.

P. Hemanth Naik	**1506085**
J.Saibaba	**1506099**
S. Venkatesh	**1506100**
P. Naveen Kumar	**1506101**

SYNOPSIS

Due to increasing demand and growing cities, traffic prediction has been a topic of interest for many researchers for the past few decades. The availability of large amounts of traffic-related data and the emerging field of machine learning algorithms has led to a significant leap towards data-driven methods.

In this paper, loop counter data are used to develop models that can predict traffic flow for several different prediction intervals into the future. In depth exploratory data analysis and statistical testing is performed to obtain good quality informative features.

Several feature sets were compared by using different machine learning methods: Ridge Regression, SVR and Random Forests. The results show that in order to obtain good prediction results thorough feature extraction is just as or even more important than learning method selection.

Good features enables us to use less complex methods, which run faster, are more reliable and easier to maintain. In conclusion, we address ideas regarding how predictions could be improved even further.

Table of Contents

Chapters

CHAPTER 1

Introduction

Traffic congestion can have substantial effects on quality of life, especially in bigger cities. It is estimated that traffic congestion in United States causes two billion gallons of fuel to be wasted every year; 135 million US drivers spend two billion hours stuck in traffic every year. Altogether, 100 billion USD are spent because of fuel in the US alone. For an average American driver, this costs 800 USD per year (Liu et al., 2006). In addition to economic aspect (wasting money and time), there is also an ecological one. Pollution could be reduced significantly by reducing travel time and thus emissions.

The above mentioned facts are the main reasons that governments are investing in Intelligent Transportation Systems (ITS) technologies that would lead to more efficient use of transportation networks. Traffic prediction models have become a main component of most ITS. Accurate real time information and traffic flow prediction are crucial components of such systems. ITS vary in technologies applied in; from advanced travel information systems, variable message signs, traffic signal control systems, to special user-friendly applications, such as travel advisors. The aim of all of these technologies is the same, to ease traffic flow, reduce traffic congestion and decrease travel time by advising drivers about their routes, time of departure, or even type of transportation.

The availability of large amounts of traffic related data, collected from a variety of sources and emerging field of sophisticated machine learning algorithms, has led to significant leap from analytical modelling to data driven modelling approach. The main concept of this paper is to investigate different machine learning algorithms and engineer features that would enable us predicting traffic state several prediction intervals into the future.

CHAPTER 2

Objectives

In general, traffic prediction studies can be categorized into three major categories: naïve methods, parametric methods and non-parametric methods. Naïve methods are usually simple non-model baseline predictors, which can sometimes return good results. Parametric models are based on traffic flow theory and are researched separately and in parallel to non-parametric, more data-driven machine learning methods. A strong movement towards non-parametric methods can be observed in the recent years, probably because of increased data availability, the progress of computational power and the development of more sophisticated algorithms.

Non-parametric does not mean models are without parameters; but refers to model's parameters, which are flexible and not fixed in advance. The model's structure as well as model parameters are derived from data. One significant advantage of this approach is that less domain knowledge is required in comparison to parametric methods, but also more data is required to determine a model. This also implies that successful implementation of data-driven models is highly correlated to the quality of available data.

In contrast, traffic in urban areas can be much more dynamic and non-linear, mainly because of the presence of many intersections and traffic signs. In such environments, data-driven machine-learning approaches, such as neural Networks, Random Forests and kNN, can be more appropriate, due to their ability to model highly nonlinear relationships and dynamic processes.

CHAPTER 3

METHODS OF PREDICTION

we have tested three different data driven methods, from a well-known machine earning library scikit-learn (version 0.20) :

k Nearest Neighbors :

In pattern recognition, the k-nearest neighbor algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space.

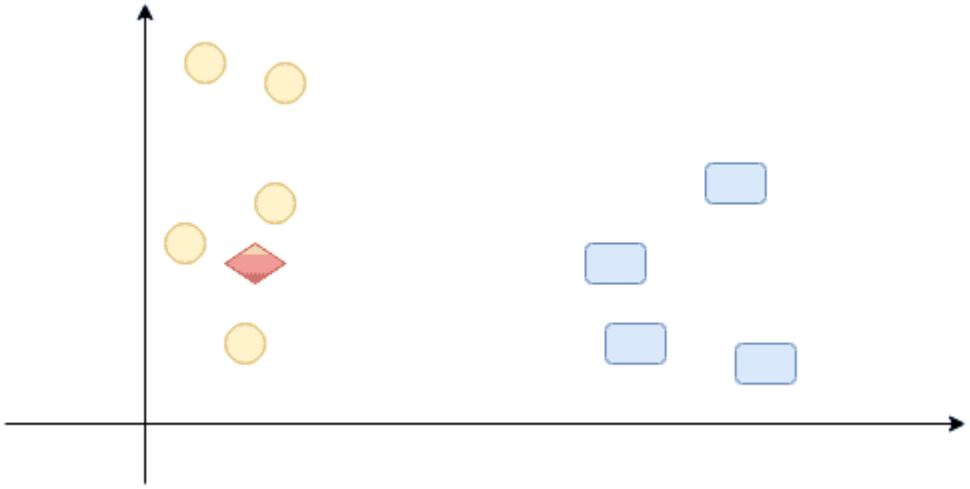
Random Forests:

An ensemble method that operates by constructing multitude of decision trees. Usually needs some parameter tuning to avoid over-fitting to their training data.

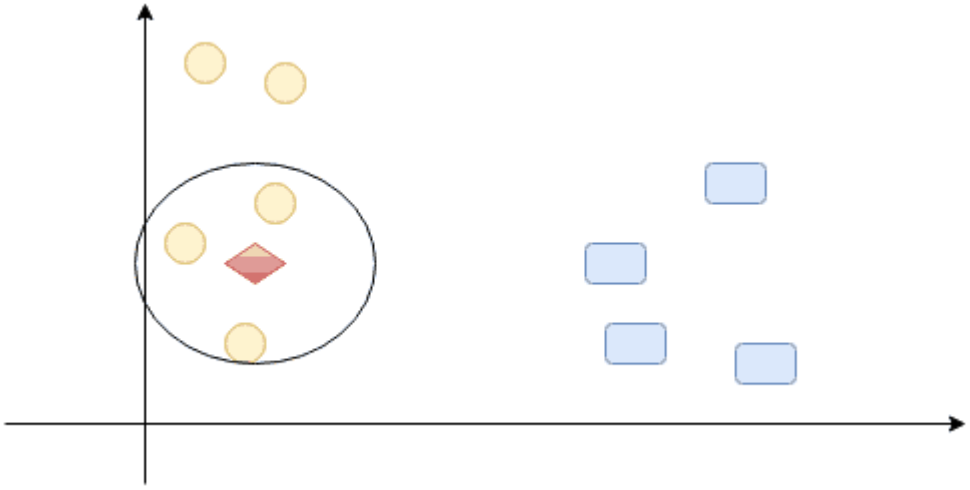
In order to train models for comparison purposes, we also need a target value, for which we want to predict values. Since traffic flow is the most informative attribute, in this research we used traffic flow as a target attribute. We also need to specify the prediction interval. Target values were obtained, by copying target attribute time series data, and lagging it according to selected prediction interval. However, if there is a need to predict any other attribute from the database, or change the prediction interval, this is a trivial task.

How does the KNN algorithm work ?

The below image shows the circles and rounded rectangles spread all over the graph. These represent two dominant classifications of data records in a dataset. For the sake of simplicity, let us assume that these are the only two classifications available in the dataset. Now, what if you have a test data record, the red diamond, and you want to find out which of these classes does this test sample belong to?



The “k” in k-NN algorithm is the count of nearest neighbors we wish to take a vote from. Let’s say k = 3. Hence, we make a circle with the diamond as the centre and just big enough to enclose only three data points on the plane.

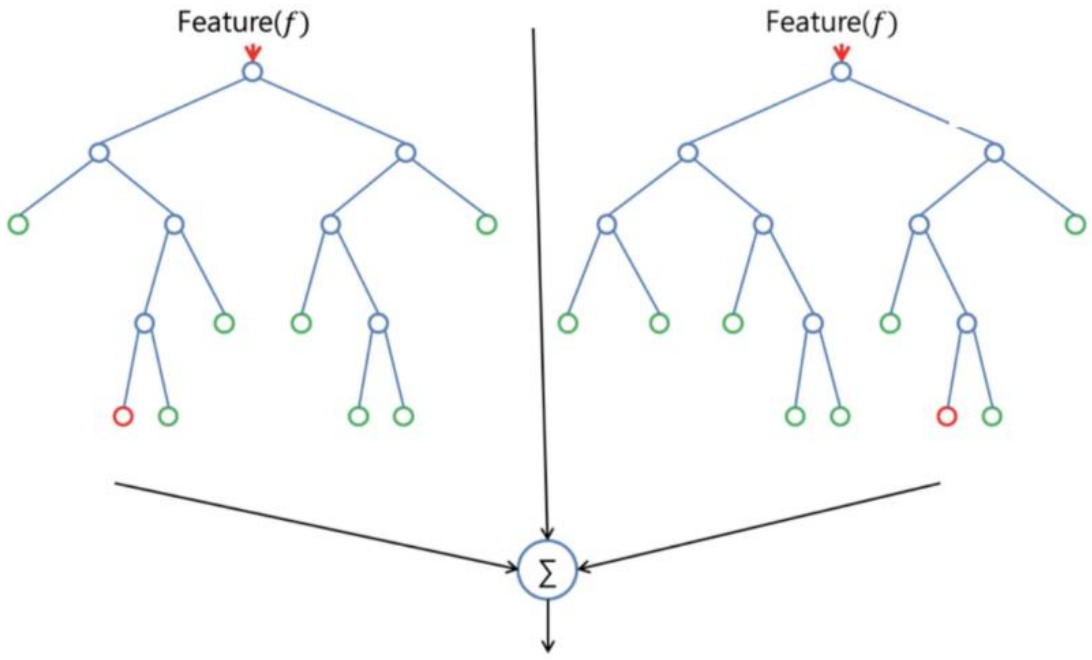


The three closest points to diamond are all circles, Hence with a good level of confidence, we can say that the diamond representing the test sample falls under circles class.

How does the Random Forest algorithm work ?

Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random. The forest it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Below you can see how a random forest would look like with two trees:



Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

CHAPTER 4

DATASET

One such event, which impacts our daily lives, is the act of commuting. As typical city dwellers, most of us commute every day for work, and often we run against the strict schedule. As an example, you need to head to the office tomorrow to attend an important early morning meeting. So you must start early and also ensure that the duration of commute, or the travel time, to your office is within that safe limit that ensures that you reach in time, neither late, nor too early. What you need is a recommendation system that can suggest you the most favourable moment to leave from home so that you are assured of reaching the office, just in time.

How to predict the traffic flow between two locations within a city?

Within a city, the traffic flow from point A to B depends on many dynamic factors along with the demographics. Even within each hour of the day, there can be variations in travel time based on busy and non-busy hour traffic. Also, the day of the week plays a role, as the weekends will witness faster travel times due to low traffic density. Apart from this, there are also some environmental conditions, such as weather conditions and temperature that indirectly affect the travel time. So let's consider these four data points that influence the travel time.

Datapoints affecting travel time

1. Time of the day

The travel time is largely dependent on this as during busy hours the traffic density on the city roads is at its peak.

2. Day of the week

Weekdays always witness more traffic density due to the rush of office commuters as compared to weekends.

3. Weather conditions

Harsh weather conditions can affect road and public infrastructure so this has an adverse impact on travel time.

4. Temperature

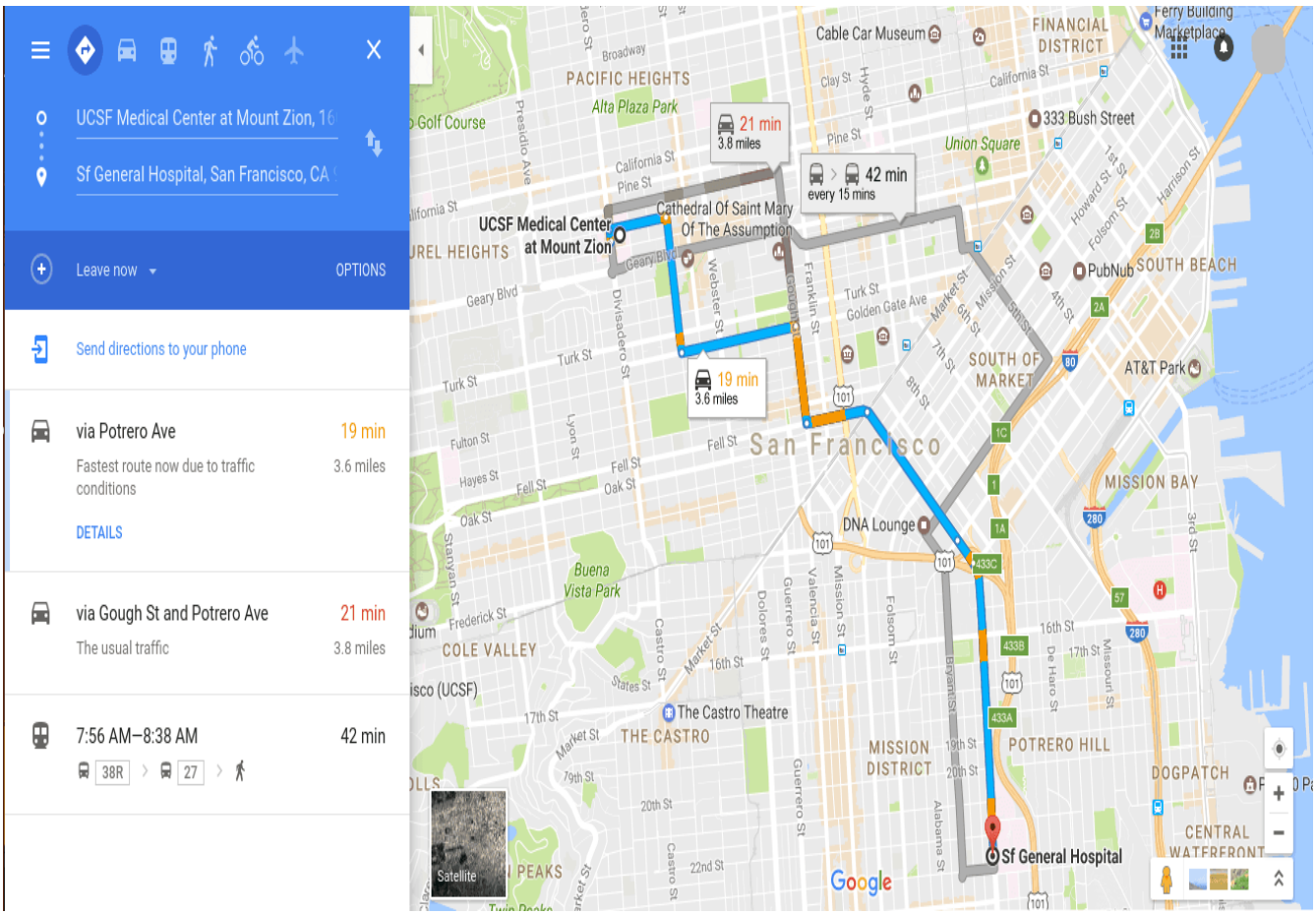
Extreme temperatures tend to keep people indoors which means less traffic density and faster travel.

Sample Data for the Problem Statement

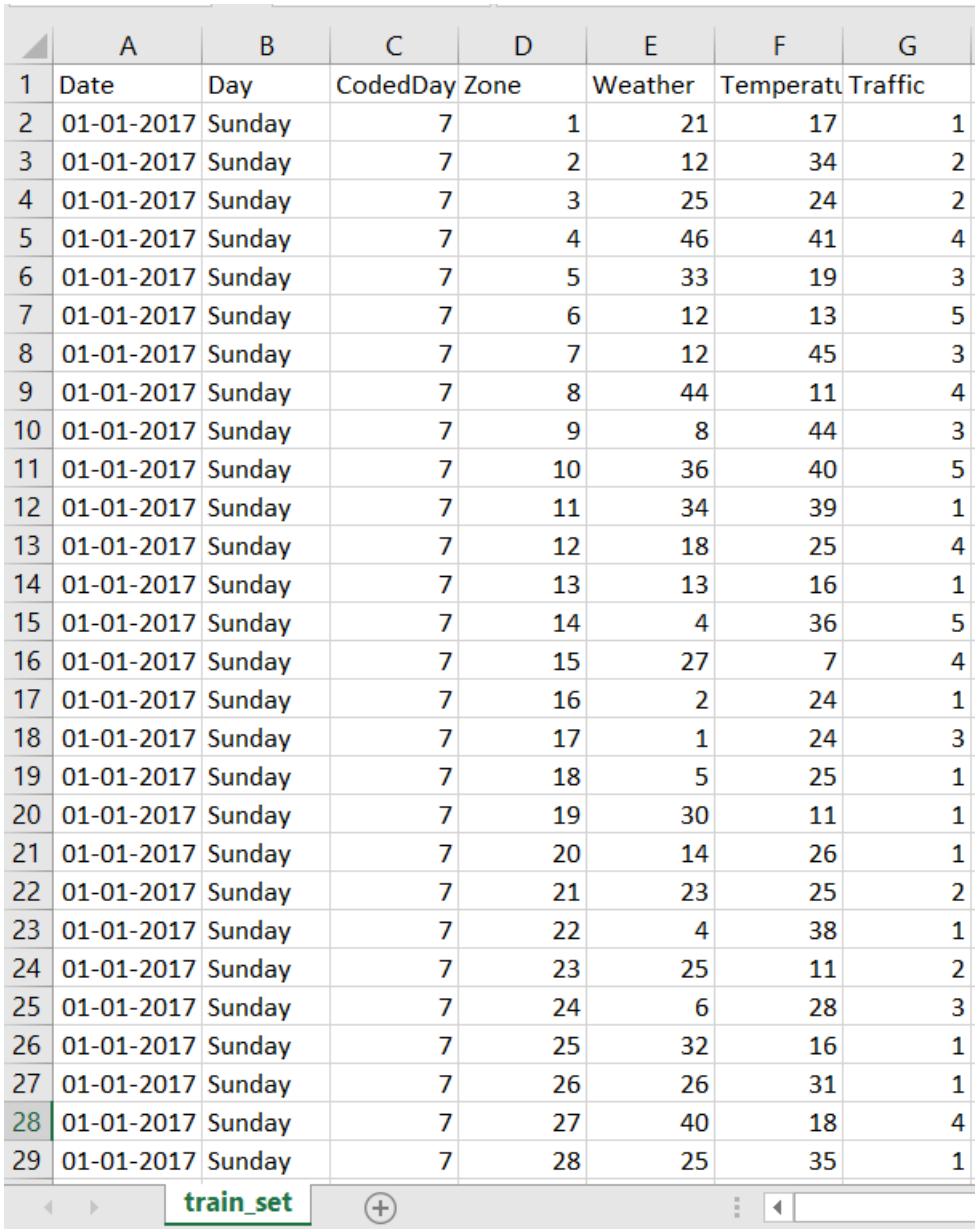
So we now have decided upon the four broad parameters that influence the travel time from a point A to point B. The next step is to collect some historical data that can aid in our prediction. This will be our training set to train our algorithm. It will be used for predicting current travel time, based on the known parameters contained in a testing set, for the same route between point A to point B.

Training Set Composition

This training set will consist of the above four parameters along with the actual travel times, recorded from the past. We have collected the data for San Francisco between the two hospitals shown in the map.



Here is how the sample training set looks like. This is real data as has been captured using the Mapquest API.



	A	B	C	D	E	F	G
1	Date	Day	CodedDay	Zone	Weather	Temperatu	Traffic
2	01-01-2017	Sunday	7	1	21	17	1
3	01-01-2017	Sunday	7	2	12	34	2
4	01-01-2017	Sunday	7	3	25	24	2
5	01-01-2017	Sunday	7	4	46	41	4
6	01-01-2017	Sunday	7	5	33	19	3
7	01-01-2017	Sunday	7	6	12	13	5
8	01-01-2017	Sunday	7	7	12	45	3
9	01-01-2017	Sunday	7	8	44	11	4
10	01-01-2017	Sunday	7	9	8	44	3
11	01-01-2017	Sunday	7	10	36	40	5
12	01-01-2017	Sunday	7	11	34	39	1
13	01-01-2017	Sunday	7	12	18	25	4
14	01-01-2017	Sunday	7	13	13	16	1
15	01-01-2017	Sunday	7	14	4	36	5
16	01-01-2017	Sunday	7	15	27	7	4
17	01-01-2017	Sunday	7	16	2	24	1
18	01-01-2017	Sunday	7	17	1	24	3
19	01-01-2017	Sunday	7	18	5	25	1
20	01-01-2017	Sunday	7	19	30	11	1
21	01-01-2017	Sunday	7	20	14	26	1
22	01-01-2017	Sunday	7	21	23	25	2
23	01-01-2017	Sunday	7	22	4	38	1
24	01-01-2017	Sunday	7	23	25	11	2
25	01-01-2017	Sunday	7	24	6	28	3
26	01-01-2017	Sunday	7	25	32	16	1
27	01-01-2017	Sunday	7	26	26	31	1
28	01-01-2017	Sunday	7	27	40	18	4
29	01-01-2017	Sunday	7	28	25	35	1

train_set

This data is collected from 1st January 2017 to 31st December 2017. The captured datapoints are :

- **Time of the day (Zone column) :**

  A number code representing a 10 minute interval time-zone, splitting the 24 hours of a day into 144 zones, (For example, the 10 minute duration from 00:00 to 00:10 Hrs is coded as 1 and 00:10 to 00:20 Hrs is coded as 2, and so on)

- **Day of the week (CodedDay column) :**

  Week day in a coded number, 7 weekdays converted into to 7 numbers starting from 1(Sunday) to 7(Saturday).

- **Weather Conditions (CodedWeather column) :**

  Weather in a coded number. Check out the codes representing weather conditions that are used in this training set.

- **Temperature (Temperature column) :**

  Average temperature during the day, in Fahrenheit.

The training set also captures the actual time taken for travel in minutes (under the Realtime column) for each of the records that capture the four data points.

So in brief, here is the list of all six parameters that constitute one data record of the training set.

1. The starting time from the source or point A (represented by Date column)
2. Ten minute interval time zone of the day (represented by Zone column)
3. Day of the week (represented by Day and CodedDay column, both meaning the same)
4. Weather conditions on that day (represented by CodedWeather column)
5. The Temperature on that day (represented by Temperature column)

The actual traffic flow from A to reach destination, or point B, when someone started from the source at the time indicated in the Date column, (represented by Realtime column)

The Testing Set

The testing set looks like this.

	A	B	C	D	E	F	G
1	Date	Day	CodedDay	Zone	Weather	Temperatu	Traffic
2	01-06-2018	Wednesda	3	2	35	17	2
3	01-06-2018	Wednesda	3	3	36	16	3
4	01-06-2018	Wednesda	3	4	27	25	5
5	01-06-2018	Wednesda	3	5	23	23	3
6	01-06-2018	Wednesda	3	6	18	42	2
7	01-06-2018	Wednesda	3	7	11	14	2
8	01-06-2018	Wednesda	3	8	45	28	4
9	01-06-2018	Wednesda	3	9	39	18	5
10	01-06-2018	Wednesda	3	10	25	9	4
11	01-06-2018	Wednesda	3	11	39	7	5
12	01-06-2018	Wednesda	3	12	22	29	2

It contains records of the traffic flow dataset that has all the influencing factors that we explained earlier. The only thing left out is the traffic flow itself (the Realtime column from the training set). Why? Because this is the parameter that we will predict.

CHAPTER 5:

Model Construction

Importing Modules

To make use of the functions in a module, you'll need to import the module with an import statement. An import statement is made up of the import keyword along with the name of the module. In a Python file, this will be declared at the top of the code, under any shebang lines or general comments.

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Install and Load pandas Package

Pandas is a powerful data analysis package. It makes data exploration and manipulation easy. It has several functions to read data from various sources.

```
# Importing the training dataset
train_dataset = pd.read_csv('train_set.csv')
X_train = train_dataset.iloc[:, [2,3,4,5]].values
y_train = train_dataset.iloc[:, 6].values


# Importing the testing dataset
test_dataset = pd.read_csv('test_set.csv')
X_test = test_dataset.iloc[:, [2,3,4,5]].values
```

LabelEncoder and OneHotEncoder

In supervised learning, we usually deal with a variety of labels. These can be in the form of numbers or words. If they are numbers, then the algorithm can use them directly. However, a lot of times, labels need to be in human readable form. So, people usually label the training data with words. Label encoding refers to transforming the word labels into numerical form so that the algorithms can understand how to operate on them. Let's take a look at how to do this.

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X_train[:, 2] = labelencoder_X.fit_transform(X_train[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [2])
X_train = onehotencoder.fit_transform(X_train).toarray()

X_train[:, 3] = labelencoder_X.fit_transform(X_train[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X_train = onehotencoder.fit_transform(X_train).toarray()

X_train[:, 4] = labelencoder_X.fit_transform(X_train[:, 4])
onehotencoder = OneHotEncoder(categorical_features = [4])
X_train = onehotencoder.fit_transform(X_train).toarray()

X_train[:, 5] = labelencoder_X.fit_transform(X_train[:, 5])
onehotencoder = OneHotEncoder(categorical_features = [5])
X_train = onehotencoder.fit_transform(X_train).toarray()


# Encoding categorical data
# Encoding the Independent Variable
labelencoder_XY = LabelEncoder()
X_test[:, 2] = labelencoder_XY.fit_transform(X_test[:, 2])
onehotencoder = OneHotEncoder(categorical_features = [2])
X_test = onehotencoder.fit_transform(X_test).toarray()

X_test[:, 3] = labelencoder_XY.fit_transform(X_test[:, 3])
onehotencoder = OneHotEncoder(categorical_features = [3])
X_test = onehotencoder.fit_transform(X_test).toarray()

X_test[:, 4] = labelencoder_XY.fit_transform(X_test[:, 4])
onehotencoder = OneHotEncoder(categorical_features = [4])
X_test = onehotencoder.fit_transform(X_test).toarray()

X_test[:, 5] = labelencoder_XY.fit_transform(X_test[:, 5])
onehotencoder = OneHotEncoder(categorical_features = [5])
X_test = onehotencoder.fit_transform(X_test).toarray()
```

Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Fitting K-NN Classifier to the Training set

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

Fitting Random Forest Classifier to the Training set

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 1000, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

Accuracy Metrics

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

```
# Making the Confusion Matrix
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_test, y_pred)


# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

F1 SCORE

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. The F1 score is the harmonic average of the precision and

recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

```
from sklearn.metrics import f1_score
f1_score(y_test, y_pred, average='micro')
```

RESULT

Traffic Status parameter can have only values between 1 and 5, where 1 being “normal traffic”, and 5 being “heavy traffic with congestion”.

CHAPTER 6:

Conclusion

We have compared the performance of two machine learning methods (kNN and Random Forests) used for predicting traffic flow. The results show that simple naïve methods, such as historical average, are surprisingly effective when making long-term predictions (more than one hour into the future), while using current traffic measuremen -ts as naïve method for prediction works well when making more short term predictions (less than 1h). This is to be expected, since current traffic situation effect more on traffic in the nearby future, then on traffic in a few hours or days.

By using less complex models, optimal model parameters are found more readily, the models run a lot faster, and they are easier to understand and maintain. We also state that the main disadvantage of models presented in this research, is its inability to

predict unusual traffic events. Even though common traffic status is informative for a commuter in a new environment, unusual traffic is the most informative information for local commuter who is aware of usual traffic. The main reason for this disadvantage is that current models uses only historical traffic data. Since, some of unusually traffic events are caused by other related events (such as nearby traffic accidents, bad weather, holidays, etc.), we believe that by including additional data sources in the model, prediction of such events could be significantly improved.

Therefore, our future plan is to collect several quality traffic related data sources (such as traffic alerts, special days statuses, bigger social events, etc.) and fuse them with loop counters data in order to generate better traffic prediction models.

CHAPTER 7:

References

[1] https://www.mapquest.com/

[2] https://www.wikipedia.org /

[3] https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[4] https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html