VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A PROJECT REPORT **(15CSP85)** ON

"ROAD TRIP RECOMMENDATION SYSTEM"

Submitted in Partial fulfillment of the Requirements for the VIII Semester of the Degree of

Bachelor of Engineering in Computer Science & Engineering

By

ANJU K P (1CR16CS020)

BHAVYA R (1CR16CS033)

CHELSA DAPTRY (1CR16CS043)

SNEHA SINGH (1CR16CS185)

Under the Guidance of,

Dr. P Kavitha

Associate Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled **"Road Trip Recommendation System"** carried out by **Ms.ANJU K P**, USN **1CR16CS020**,**Ms.BHAVYA R**, USN **1CR16CS033**, **Ms.CHELSA DAPTRY**, USN **1CR16CS043, Ms.SNEHA SINGH**, USN **1CR16CS185,** bonafide students of CMR Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering** in Computer Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Dr. P Kavitha	**Dr. Prem Kumar Ramesh**	**Dr. Sanjay Jain**
Associate Professor	**Professor & Head**	**Principal**
Dept. of CSE, CMRIT	**Dept. of CSE, CMRIT**	**CMRIT**

External Viva

Name of the examiners — Signature with date

1. _____

2. _____

DECLARATION

We, the students of 8th semester of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the work entitled "**Road Trip Recommendation System**" has been successfully completed under the guidance of Dr. P Kavitha, Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This dissertation work is submitted in partial fulfillment of the requirements for the award of Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2019 - 2020. Further the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place:

Date:

Team members:

ANJU K P (1CR16CS020) _____

BHAVYA R (1CR16CS033) _____

CHELSA DAPTRY (1CR16CS043) _____

SNEHA SINGH (1CR16CS185) _____

ABSTRACT

The Road Trip Recommendation System aims to minimize the time taken by the user in planning for the trip, since road trips are quite common nowadays, but people tend to utilize most of the time in planning for the trip and hence compromise the time taken to do other task. It takes the source, destination and the types of places from the user as input and generates the most optimal path from the source to the destination considering most of the places which falls under the users interest along the way. The various categories given to the user to select from are :'Spiritual',' Heritage', 'Trekking', 'Rivers and Waterfall', 'Wildlife', 'Hill station' and 'Beach'. In addition to generating the most optimal path the system also predicts the best time to travel from the source to the destination so that the user will get the most out of the trip. For the prediction of the best time to travel the system is using decision tree CART algorithm which uses the concept of gini instead of entropy and information gain which greatly reduces the computation time which is crucial for our system.

By integrating both generation of the optimal path and prediction of the best months to travel this systems saves most of the time used by the user for planning for the trip. The system generates the path with considerably high accuracy and also the prediction of the best months to travel is also quite precise.

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing us a platform to pursue our studies and carry out our final year project

We have a great pleasure in expressing our deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for his constant encouragement.

We would like to thank **Dr. Prem Kumar Ramesh,** HOD, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

We consider it a privilege and honor to express our sincere gratitude to our guide **Dr. P Kavitha, Associate professor,** Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of this review.

We also extend our thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged us.

Finally, we would like to thank our family and friends for all their moral support they have given us during the completion of this work.

Anju K P (1CR16CS020)
Bhavya R(1CR16CS033)
Chelsa Daptry(1CR16CS043)
Sneha Singh(1CR16CS185)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CART	**Classification and Regression Tree**
ML	**Machine Learning**
POI	**Point Of Interest**

CHAPTER 1

INTRODUCTION

Great journeys are not planned, they are made. If you claim to be a traveller, you have to be a lover of the quiet trails and the footpaths, and really taste the travel, live the journey, and seek the new beyond. Imagine reading a book and not reading the footnotes.

Road trips are the best time to bond with family, and friends, and there's plenty that can be done. Moving away from a steady Internet connection means long conversations find a place to happen; songs that stir up nostalgia get a chance to be heard; embarrassing and funny old stories that you thought were buried deep get to relive (and admit it, you, kind of, like it); and more importantly, you get a chance to create new memories. Road trips give you the feeling that you are in control, which also means that you can practically wear anything you like – those torn jeans that mom wants to throw out; that favourite band t-shirt that has been living this long in your bottom shelf; or a pair of flip-flops or monkey slippers that, for obvious reasons, you can't wear to work. You can almost hear your toes breathing, can't you?

After all, on a road trip, the journey is always the real destination. And what's a little detour if you can end up having the time of your life in some town you would have otherwise never heard of? Taking your time means allowing for the spontaneous to happen, and allowing for some awesome unexpected experiences to take shape. Going on a road trip is really a very fascinating and fantastic experience in one's life. It gives the excitement and adventure of the place being travelled to as well as of the entire journey on the road.

However, the most common difficulty faced by people is to decide which would be the most optimal path to be taken to reach their destination while planning for a road trip. Also people would want to know whether there are any site-seeing places that could be visited on the way to their destination and what would be the best

time to travel to their destination. The existing works have been focusing on how to find routes that minimize a single kind of trip cost such as trip time or distance, amongst others. However, it has been noticed that the existing systems have not considered the user interest in their recommendation systems[1]. Some only suggest the best time to travel to the destinations[3],[5]. The users will have to browse through different sites to obtain all information required for planning their travel. Therefore, taking all these factors into consideration we have developed an application that provides suitable solutions for the above mentioned problems so that people can make their travel experience a memorable one. We have tried to efficiently plan a route that includes most of the sites of the user's interest based on the geographical location of these sites. We have focused on generating an aesthetic travel route, which covers the astounding tourist sites. We also focused on suggesting the best time to travel to the destinations so that the user will not have to browse through multiple sites to acquire all the information required to plan his/her travel. We have therefore designed an application where information about the route to be taken as well as the best time to travel is obtained at one place[4].

1.1 Relevance of the Project

The main purpose of this project is to suggest the most optimal routes to the specified destination considering the user's interests to visit multiple destinations on the way to their final destination. For the user to have the best travel experience, this application also suggests the best time to visit the destination.

This application also saves the user's time by providing all the necessary information required to travel to the destination in one place instead of the user spending his time on acquiring all this information from different sources.

1.2 Scope of the Project

The application takes the source and destination as input from the user. It also asks the user to select his/her interest. An optimal path from the source to the destination is created which covers all the site-seeing places according to the user's interest. It

displays this optimal path that is generated based on the user's interest on a map for better user readability. It displays a list of intermediate places that are selected according to the user's interest which was provided as input.

In addition to this, it also displays a directions panel where the distance from every source to every other destination is mentioned so that the user can have a clear idea of the distance he/she will have to travel between source and destination by covering the intermediate places. It also recommends the best time to travel by analysing the weather data. This information helps the user to have the best experience by visiting the destination during the best season possible and it also saves the user's time by avoiding further browsing to gather this particular information.

1.3 Machine Learning Algorithms

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

We are using Machine Learning to predict the best path to be taken by the user, taking the user's interest into consideration. The following machine learning algorithms were studied in order to select the best one for implementation of our project.

1.3.1 Decision Tree

Decision Tree Analysis is a general, predictive modelling tool that has applications spanning a number of different areas. In general, decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different

conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surfaces.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

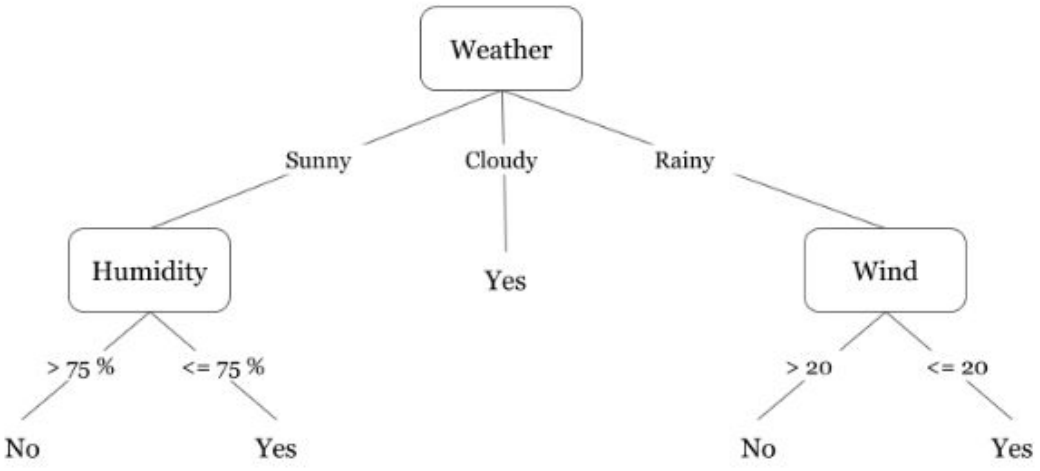

Fig 1.1 Decision tree

1.3.2 Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.

Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2 . x$$

Eqn 1.1

While training the model we are given :

x: input training data (univariate – one input variable(parameter))

y: labels to data (supervised learning) When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.

θ1: intercept

θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

1.3.3 Artificial Neural Networks

Artificial Neural networks (ANN) or neural networks are computational algorithms. It intended to simulate the behaviour of biological systems composed of "neurons". ANNs are computational models inspired by an animal's central nervous systems. It is capable of machine learning as well as pattern recognition. These are presented as systems of interconnected "neurons" which can compute values from inputs.

A neural network is an oriented graph. It consists of nodes which in the biological analogy represent neurons, connected by arcs. It corresponds to dendrites and synapses. Each arc associated with a weight while at each node. Apply the values received as input by the node and define Activation function along the incoming arcs, adjusted by the weights of the arcs.

A neural network is a machine learning algorithm based on the model of a human neuron. The human brain consists of millions of neurons. It sends and processes signals in the form of electrical and chemical signals. These neurons are connected with a special structure known as synapses. Synapses allow neurons to pass signals. From large numbers of simulated neurons neural networks form.

An Artificial Neural Network is an information processing technique. It works like the way the human brain processes information. ANN includes a large number of connected processing units that work together to process information. They also generate meaningful results from it. We can apply Neural networks not only for classification. It can also apply for regression of continuous target attributes.

Neural networks find great application in data mining used in sectors. For example, economics, forensics, etc. and for pattern recognition. It can be also used for data classification in a large amount of data after careful training. A neural network may contain the following 3 layers:

1. Input layer – The activity of the input units represents the raw information that can feed into the network.

2. Hidden layer – To determine the activity of each hidden unit. The activities of the input units and the weights on the connections between the input and the hidden units. There may be one or more hidden layers.

3. Output layer – The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

1.3.4 ID3 Algorithm

In decision tree learning, ID3 (Iterative Dichotomiser 3) is an algorithm invented by Ross Quinlan used to generate a decision tree from a dataset. ID3 is the precursor to the C4.5 algorithm, and is typically used in the machine learning and natural language processing domains. Most generally ID3 is only used for classification problems with nominal features only. ID3 uses a top-down greedy approach to build a decision tree. In simple words, the top-down approach means that we start building the tree from the top and the greedy approach means that at each iteration we select the best feature at the present moment to create a node. The ID3 algorithm selects the best feature at each step while building a Decision tree. It uses a metric known as Information Gain or just Gain to find the best feature.

Information Gain calculates the reduction in the entropy and measures how well a given feature separates or classifies the target classes. The feature with the highest Information Gain is selected as the best one.

Entropy is the measure of disorder and the Entropy of a dataset is the measure of disorder in the target feature of the dataset. In the case of binary classification (where the target column has only two types of classes) entropy is 0 if all values in the target column are homogenous(similar) and will be 1 if the target column has equal number values for both the classes.

Let dataset be denoted as S, entropy is calculated as:

$$\text{Entropy}(S) = -\sum p_i * \log_2(p_i);\ i = 1 \text{ to } n$$

Eqn 1.2

where, n is the total number of classes in the target column (in our case n = 2 i.e., I and NI) p_i is the probability of class 'i' or the ratio of "number of rows with class i in the target column" to the "total number of rows" in the dataset.

Information Gain for a feature column A is calculated as:

$$IG(S, A) = \text{Entropy}(S) - \sum((|S_v| / |S|) * \text{Entropy}(S_v))$$

Eqn 1.3

where S_v is the set of rows in S for which the feature column A has value v, $|S_v|$ is the number of rows in S_v and likewise |S| is the number of rows in S.

ID3 Steps:

1. Calculate the Information Gain of each feature.
2. Considering that all rows don't belong to the same class, split the dataset S into subsets using the feature for which the Information Gain is maximum.
3. Make a decision tree node using the feature with the maximum Information gain.
4. If all rows belong to the same class, make the current node as a leaf node with the class as its label.
5. Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

1.3.5 CART Algorithm

Classification and Regression Trees or CART for short is a term introduced by Leo Breiman to refer to Decision Tree algorithms that can be used for classification or regression predictive modeling problems. Classically, this algorithm is referred to as "decision trees", but on some platforms like R they are referred to by the more modern term CART. The CART algorithm provides a foundation for important algorithms like bagged decision trees, random forest and boosted decision trees. Creating a CART model involves selecting input variables and split points on those variables until a suitable tree is constructed. The selection of which input variable to use and the specific split or cut-point is chosen using a greedy algorithm to minimize a cost function. Tree construction ends using a predefined stopping criterion, such as a minimum number of training instances assigned to each leaf node of the tree.

Gini is used by the CART (classification and regression tree) algorithm for classification trees. Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. Gini impurity can be computed by summing the probability p_i of an item with label i being chosen times the probability $\sum_{k \neq i} p_k = 1 - p_i$ of a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

The Gini impurity is also an information theoretic measure and corresponds to Tsallis Entropy with deformation coefficient $q=2$ q, which in Physics is associated with the lack of information in out-of-equlibrium, non-extensive, dissipative and quantum systems. For the limit $q \to 1$ one recovers the usual Boltzmann-Gibbs or Shannon entropy. In this sense, the Gini impurity is but a variation of the usual entropy measure for decision trees.

To compute Gini impurity for a set of items with J classes, suppose $i \in \{1, 2, \ldots, J\}$, and let p_i be the fraction of items labeled with class i in the set.

$$I_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i (1 - p_i) = \sum_{i=1}^{J} (p_i - p_i^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2$$

Eqn 1.4

CHAPTER 2

LITERATURE SURVEY

The proposed project draws ideas and conclusions from several other papers published on topics concerning travel and prediction of prevailing as well as upcoming conditions of an area. The surface details of those papers which were used to derive the solutions and ideas related to various relevant concerns are described in the upcoming sections.

2.1 Exploring Personalized Travel Route Using POIs

Authors : Jie Li, Yujiu Yang, and Wenhuang Liu

The development of the travel industry gradually fosters personalized requirements of tourists, such as setting the start place, exploiting interesting activities and organizing the travel route. This paper takes advantage of two online websites to collect information of both scenic spots and real-world local activities, and three rule-based methods are proposed to recommend travel routes for tourists, which can take their instant location into account and satisfy their personalized demands. Specifically, scenery spots dataset is extracted from Flickr and popular activities from Douban-Event in line with the visiting city and date. Then these items are processed and the filtering results are taken as the recommendation candidates. Finally based on popularity and geographical location of these venues/activities, a colorful travel route is arranged, which covers not only the everlasting scenic spots, but also some temporary activities.The quality of travel routes produced is evaluated by our recommendation approaches with an alternative plan suggested by a professional travel site. The results justify the advantages of our approaches over the baseline method provided by travel experts.

This paper aims to offer the user a travel route covering a set of POIs (scenic spots or events) by giving consideration to both popularity and distance. Three approaches are proposed:

1. Distance-based recommendation (DR)
2. Popularity-based recommendation (PR)
3. Distance-Popularity-based recommendation (DPR).

The methods are designed to be deployed in an application scenario that is suitable for everyone, without limitation to the tourists who already have travel history in the given dataset or have been the registered members of some specific website. Moreover, it could also match the user's personalized demands, such as selecting the travel city/date and setting the start point for the route.

2.1.1 Contributions

- Popularity: in order to define the level of a POI being interested by users quantitatively, the concept of popularity is adopted and a new definition of it is applied, which considers not merely the number of unique visits made to those POIs but also the number of views/comments/favorites produced on the website. And the popularity is used to filter the locations and then rank the POIs in the PR/DPR method.
- Activity/event: The travel route incorporates both scenic spots and temporal activities. The common travel recommendation only takes attractions into account, which could be visited almost at any time. This paper suggested that, attending some popular online activities is a good way to understand local history and culture.

2.1.2 Framework

The architecture behind the approach is depicted in Fig. 2.1. It is configured into various modular tasks. From this, the process of recommendation can be understood.

At first, the tourist tells the system which city he wants to visit, the time he wants to come and the place he wants to set as the beginning point. The start point is determined by the user's manual input or delivered by a GPS module embedded in his smart phone. After receiving these initial requirements, the system collects the POI information. Since the number of cities is limited and the scenic spots information would not change frequently, so the scenic spots metadata of cities have been collected offline.

However, activities change every day, which means there are different activities on a separate day. So the system extracts activities information from DoubanEvent online. Then the system calculates the popularity of each POI, which makes up a POI database with other two attributes (name and geo-coordinates). With these preparations, the system could provide travel routes for the user.

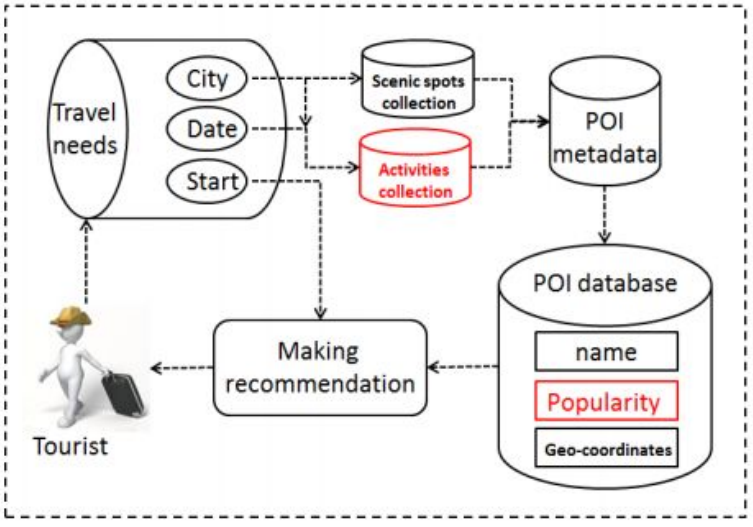

Fig 2.1 Architecture of personalized travel recommendation.

2.1.3 Conclusion

This paper addressed the question of real-life travel route recommendation. The user input their tourism requirements, then the system makes good use of POI information to get the popularity and geo-coordinates of them. Flickr provides the metadata of scenic spots and DoubanEvent is the source of activities information. Three rule-based

methods have been proposed to produce travel route recommendations and take experiments to evaluate the quality of these recommendations. The feedback of the survey questionnaire yields promising results and distance-based recommendation leads to the best travel route.

2.2 Machine Learning Applied to Weather Forecasting

Authors : Mark Holmstrom, Dylan Liu, Christopher Vo

Weather forecasting is the task of predicting the state of the atmosphere at a future time and a specified location. Since machine learning, on the contrary, is relatively robust to perturbations and doesn't require a complete understanding of the physical processes that govern the atmosphere. The scope of this paper was restricted to forecasting the maximum temperature and the minimum tem- perature for seven days, given weather data for the past two days. A linear regression model and a variation on a functional regression model were used, with the latter able to capture trends in the weather. The linear regression model outperformed the functional regression model, suggesting that two days were too short for the latter to capture significant weather trends, and perhaps basing our forecasts on weather data for four or five days would allow the functional regression model to outperform the linear regression model.

2.2.1 Methods

The first algorithm that was used was linear regression, which seeks to predict the high and low temperatures as a linear combination of the features. Only eight features were used: the maximum temperature, minimum temperature, mean humidity, and mean atmospheric pressure for each of the past two days. The cost function that linear regression seeks to minimize is :

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left\|h_\theta(x^{(i)}) - y^{(i)}\right\|^2$$

Eqn 2.1

where m is the number of training examples. The value of θ that minimizes the cost in equation 2.1 is

$$\theta = (X^T X)^{-1} X^T Y$$

Eqn 2.2

The second algorithm that was used was a variation of functional regression, which searches for historical weather patterns that are most similar to the current weather patterns, then predicts the weather based upon these historical patterns. The metric on the space of spectra is defined as:

$$d(f_1, f_2) = \sum_{j=1}^{2} [w_1 [f_1(j)_1 \neq f_2(j)_1] + \sum_{k=2}^{5} [w_k (f_1(j)_k - f_2(j)_k)^2]$$

Eqn 2.3

where w is a weight vector that assigns weight to each feature. Then, given the values f(1), f(2) of the first two days of a spectrum f, the remainder of the spectrum f(i) for i in the range 3 to 9 can be predicted as:

$$\hat{f}(i) = \frac{\Sigma_{j \epsilon neighk\,(f)} ker(d(f^{(j)}, f) / h) f^{(j)}(i)}{\Sigma_{j \epsilon neighk\,(f)} ker(d(f^{(j)}, f) / h)}$$

Eqn 2.4

The rms error of the estimator $\hat{f}$ is defined to be

$$Error_{rms} = \sqrt{\sum_{i=3}^{9} \frac{\|\hat{f}(i) - f(i)\|^2}{14}}$$

Eqn 2.5

2.2.2 Experimental and Result

The learning curves for the linear regression and functional regression models are also shown in Fig. 2.2 and Fig. 2.3 respectively.

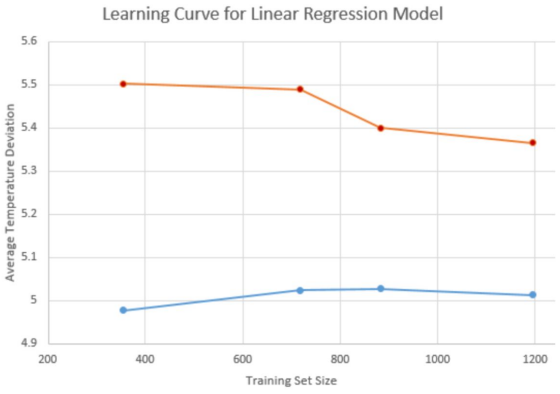

Fig. 2.2 The learning curve for the linear regression model, showing the rms error averaged across all seven days as a function of the training set size.

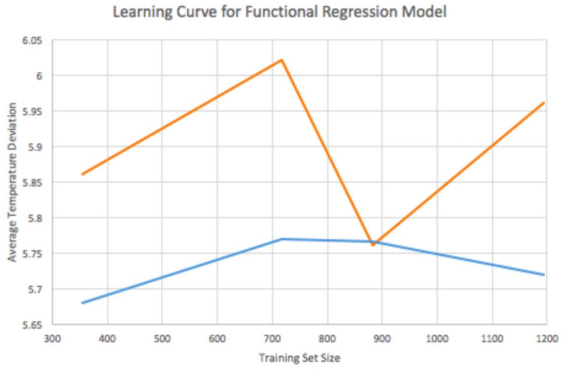

Fig. 2.3 The learning curve for the functional regression model, showing the rms error averaged across all seven days as a function of the training set size.

As expected, linear regression proved to be a low bias, high variance model. The relatively low errors in the learning curve in Fig. 2.2 indicates that the model is an unbiased estimator. More interestingly, functional regression proved to be a high bias, low variance mode. The relatively large errors in the learning curve in Fig. 2.3 indicates that the model is a biased estimator.

2.2.3 Conclusion

Both linear regression and functional regression were outperformed by professional weather forecasting services, although the discrepancy in their performance decreased significantly for later days, indicating that over longer periods of time, these models may outperform professional ones. Linear regression proved to be a low bias, high variance model whereas functional regression proved to be a high bias, low variance model. Linear regression is inherently a high variance model as it is unstable to outliers, so one way to improve the linear regression model is by collection of more data. Functional regression, however, was high bias, indicating that the choice of the model was poor and prediction cannot be improved by further collection of data.

2.3 Trip planning Route Optimization

Authors : Wai Chong Chia*, Lee Seng Yeong, Fennie Jia Xian Lee, Sue Inn Ch'ng

This paper proposes an offline approach to generating an optimal vacation routing plan to maximize the number of places to visit during the vacation duration. A traveling salesman problem algorithm results in a plan with the shortest path of travel given a number of stops. This solution is not suitable as a travel planner needs to take into account the operating hours and duration of stay at each stop to create a daily plan. The proposed approach will calculate the time of arrival and minimum time of stay to determine if it exceeds the operating hours of the stop. The most important criteria of this system is the need to calculate the time of travel from each location. This is done using two methods :

1) Using the latitudinal and longitudinal coordinates of places to calculate the straight-line distance and subsequently time between them, and

2) By constructing a look-up table consisting of the travel distance and travel time between places using the Google Maps Directions API.

The simulation results show that the total traveling time of the proposed system is within 15 minutes to online results which is optimized using data from the Google Maps Directions API while being able to satisfy both operating hours and duration of stay constraints.

2.3.1 Methodology

Testing of the approach comes in the form of a mobile application. To start, the user is required to first select an origin (typically the place of stay), the start time and a selection of up to six places to visit. All location selection is done by scanning a QR-code associated with those locations that will be published in guidebooks, magazines, brochures and websites. The contents of the QR code includes:

1) Coordinates of the place (latitude and longitude),

2) Operating hours, and

3) Recommended stay duration.

For places that are not listed, the user can enter the information that are contained in the QR-codes into the software. This can happen often as users may obtain information from their friends who have been there before. A maximum of six places per day was chosen as that was decided to be a comfortable number of places to visit. The results are shown where a round-trip is assumed (i.e. the start and end points are the same) but this does not have to be so. A flowchart of this operation is shown in Fig. 2.4.

To determine the total time for each permutation, the time between locations is first calculated. This is done in two ways:

1) Using the latitudinal and longitudinal coordinates of places to calculate the straight line surface distance (spherical distance) between them.

2) By constructing an LUT consisting of the travel distance and travel time between places using the Google Maps Directions API .

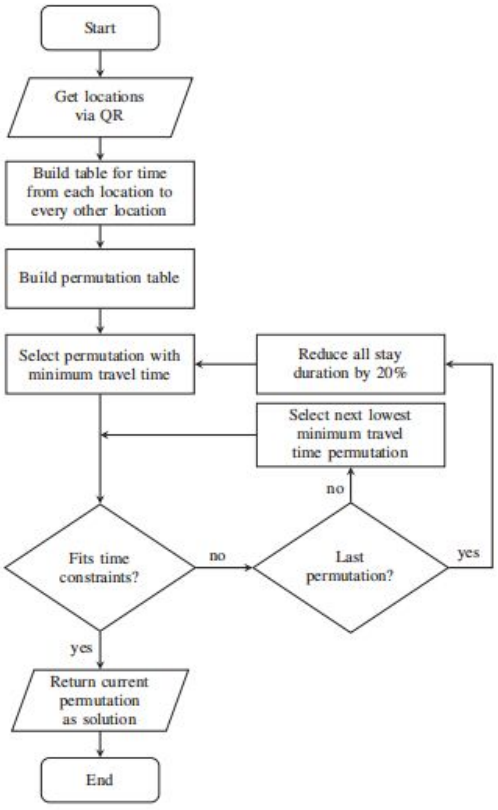

Fig 2.4 Trip planner system flowchart

2.3.2 Conclusion

This paper presented two offline approaches to calculating optimal daily sequences for the duration of a vacation given a daily maximum of six locations to visit without the need for maps. An optimal sequence is one with the shortest travel time and is determined by calculating the travel time of all permutations using two approaches, 1) using straight-line distance approximation and an average travelling speed, and 2) using information from Google Maps Directions API generated LUT, which must also fulfill two main criteria, 1) ensuring arrival during the operating hours of each location, and 2) ensuring sufficient time of stay at each location. The results show that sequences generated by the straight-line distance approximation and using the LUT are able to satisfy these criteria with a 15.1% time penalty compared to waypoint optimizations using API.

2.4 Adaptive Trip Recommendation System: Balancing Travellers Among POI's with MapReduce

Authors : Sara Migliorini, Damiano Carra and Alberto Belussi

This paper considers the trip planning problem that takes into account, besides the user preferences and the system constraints, the balancing of users among the different POIs. The recommendation engine needs to consider the prediction of the user presence at the POIs. The quality of the prediction determines the quality of the recommendation. The prediction should include historical data, as well as the recommendations made so far by the system itself.

2.4.1 Key Contributions

The contributions of this work are the following:

(1) Formulation of the online optimization problem, where the current estimation of the user visiting the different POIs is considered as part of the input of the recommendation system.

(2) Design and implementation of an efficient solution engine that works in near real-time. The solution is based on a parallel version of the Simulated Annealing approach, using the MapReduce programming framework.

(3) Evaluation of the trip recommendation system with a dataset collected from the tourist information office of the city of Verona. The dataset contains the visits of different POIs included in a set of city passes, whose duration is predetermined.

2.4.2 Methodology

Definition 1 (Point of interest). *A point of interest (POI) p represents an attraction reachable by users. It is characterized by several attributes, such as the admission fee, or the opening hours. Among these, we consider: the spatial coordinates defining its position on the Earth surface, which we denote with* p^c*, and the duration of a visit, denoted by* $p^v(t)$*, which depends on the time t when the visit starts.*

Definition 2 (Trip). *A trip* τ *is an ordered collection of POIs, i.e.,* $\tau = \langle p_1, p_2, \ldots, p_n \rangle$,

where n indicates the number of POIs contained in τ, |τ| = n.

The set T of all possible trips contains all the possible ordered combinations of POIs in P, for any cardinality of τ.

Definition 3 (Path). *Given any two spatial coordinates c_i and c_j, and a travel mode m (e.g., walking, public transportation), a path $\pi(c_i, c_j, m)$ is a continuous portion of a transport network that connects the points whose location is defined by ci and c_j. The path is characterized by the travel distance, $\pi_{td}(c_i, c_j, m)$, and by the travel time, $\pi_{tt}(c_i, c_j, m)$.*

Definition 4 (Recommendation query). *Users looking for a recommendation submit a query Q to the system by specifying the following constraints:*

- *the initial coordinates c_0 where the trip begins;*
- *the time t_0 at which the trip will start;*
- *the maximum trip duration TD_{max};*
- *the travel mode m.*

The total trip time λτ for a trip τ is defined as:

$$\lambda_T(c_0, t_0) = \sum_{i=1}^{|\tau|} (\pi_{tt}(p_{i-1}^c, p_i^c) + p_i^v(t_i))$$

Eqn 2.6

Definition 5 (Objective functions). *Given a trip τ , the objective functions f_n, f_{tt} and f_{td} denote the number of locations not visited during the trip, the estimated trip travel time, and the total distance travelled during the trip, respectively. They are computed as:*

$$f_n = |P| - |\tau|, \qquad f_{tt} = \sum_{i=1}^{|\tau|} \pi_{tt}(p_{i-1}^c, p_i^c),$$

$$f_{td} = \sum_{i=1}^{|\tau|} \pi_{td}(p_{i-1}^c, p_i^c)$$

Eqn 2.7

We would like to minimize is a composition of objective functions, and it can be defined as

$$Minimize_{\tau} \ < f_n, f_{tt}, f_{td} >,$$
$$\text{subject to } \lambda_T(c_0,\ t_0) \ < \ TD_{max}$$

Eqn 2.8

The proposed recommendation system (Fig. 2.5) has two main components: an offline analysis of the user presence in the different POIs, and a recommendation engine based on a parallel implementation divided into two main stages.

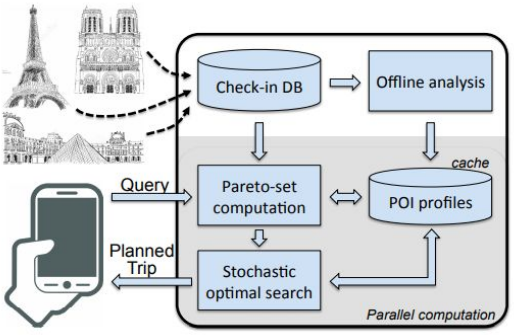

Fig. 2.5. Proposed System Architecture

Offline analysis: The offline analysis processes the records of the visits at the POIs. In particular, it considers the users with identifiers, i.e., the ones that use, for instance, a bundle offer. For these users, we can reconstruct the set of POIs they visited. We can also derive the set of popular trips that can be used by the recommendation engine as a starting point in the search of the optimal solution when replying to a query. The popular trips are stored back in the database, and accessed by the recommendation engine when it processes a query.

Exploration of the solution space: The exact solution of is computationally expensive, thus we resort to heuristics. Our solution builds trip recommendations using a dominance-based Multi-Objective Simulated Annealing (MOSA) technique.

Dynamic adaptation: During the computation of the solution for the user, the recommendation system needs to take into account the information about the visiting

time, its duration must be less than the maximum trip duration TDmax. The visiting time in turn depends on the POI occupancy. To adapt to the current occupancy, as the system issues recommendations, it records the choices of the users, and it updates in real-time the estimation of the POI occupancy.

2.4.3 Conclusion

Personalized trip recommendation systems tailor the suggestions to the users based on their constraints and requirements. Nevertheless, they do not consider the impact of the recommendations on the whole system. In this paper a step is taken to fill this gap. In particular, a system has been proposed that efficiently searches the solution space through a Map Reduce implementation of the multi-objects optimization problem and balances the users among different POIs by including the predicted level of crowding. The system is evaluated using a real dataset, showing consistent improvements over the paths usually followed by the tourists.

2.5 TripPlanner:Personalized Trip Planning Leveraging Heterogeneous Crowdsourced Digital Footprints

Authors: **Chao Chen, Daqing Zhang, Bin Guo, Xiaojuan Ma, Gang Pan, and Zhaohui Wu**

This paper proposes a novel framework called TRIPPLANNER, leveraging a combination of location-based social network (i.e., LBSN) and taxi GPS digital footprints to achieve personalized, interactive, and traffic-aware trip planning. First, a dynamic point-of-interest network model is constructed by extracting relevant information from crowdsourced LBSN and taxi GPS traces. Then a two-phase approach for personalized trip planning is proposed. In the route search phase, TRIPPLANNER works interactively with users to generate candidate routes with specified venues. In the route augmentation phase, TRIPPLANNER applies heuristic algorithms to add user's preferred venues iteratively to the candidate routes, with the

objective of maximizing the route score while satisfying both the venue visiting time and total travel time constraints.

2.5.1 Contributions

- First a new trip planning problem is defined, which allows users to specify not only the must-visit venues but also optional venue categories if the time permits, given a total travel time budget. Further more realistic assumption is made about the transit time between venues that varies with time of the day and day of the week. In other words, the total travel time of the same route may be different.
- Second an attempt is made to construct a dynamic POI network model of a city, leveraging heterogeneous crowdsourced digital footprints (i.e., Foursquare check-ins and taxi GPS traces) to better utilize the strengths of each data source in characterizing the attributes of the nodes and links of the POI network.
- Third a two-phase approach is proposed for personalized, interactive, and traffic-aware trip planning. A new way to score an itinerary is also proposed, considering both the popularity and individual preference of venues.
- Finally, the efficiency and effectiveness of TRIPPLANNER is validated by extensive evaluations using large-scale real-world data sets.

2.5.2 Framework

As shown in Fig. 2.6, the proposed framework contains three components, i.e., the dynamic POI network model, the route search component, and the route augmentation component. While the dynamic POI network model is prebuilt and maintained offline, the route search and route augmentation components collaboratively answer users' trip queries in real time.

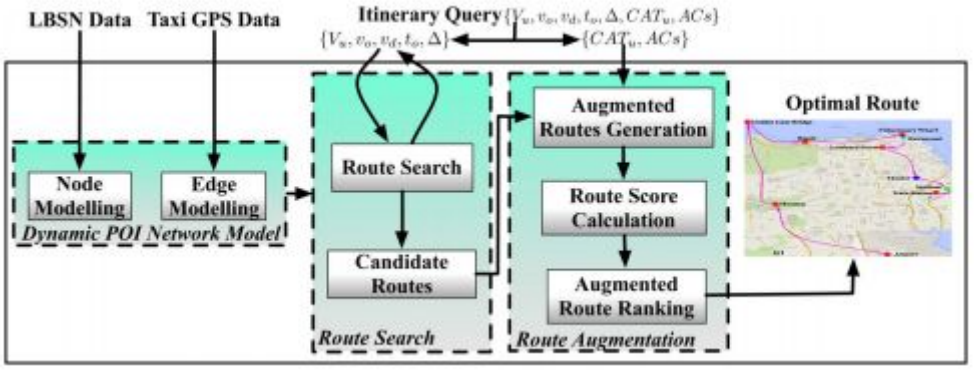

Fig 2.6 TRIPLANNER Framework

- Dynamic POI Network Model: The key problem of POI network model construction is to separately extract attributes of POI nodes from the Foursquare data set and information of the edges from the taxi GPS data set.
- Route Search: Given user-specified venues to visit, the starting time, and the time budget, the route search component returns routes that traverse all the intended venues from the starting location to the destination. In particular, the returned routes with a time margin greater than a user-determined threshold become candidate input to the route augmentation component. However, users might list too many venues to cover within the time constraint, or the planned visiting time does not agree with the operating hours of certain venues. If the TRIPPLANNER system detects any of those cases, it will interact with the user to manually modify the venue list.
- Route Augmentation: This component aims to augment the candidate generated from the route search module with user-preferred venues inferred from the intended venue categories in the query, maximizing the route score under the given travel time budget. It first pulls together all of the venues that belong to user-preferred venue categories as candidate venues. Then for each candidate route, it tries to insert venues in the pool into it to generate an augmented route without breaking any constraint. In the end, TRIPPLANNER presents the augmented routes to the user, in an order sorted according to their scores in the augmented route ranking module.

2.5.3 Conclusion

In this paper, a novel framework called TRIPPLANNER has been developed for personalized, interactive, and traffic-aware trip planning. It leverages two heterogeneous data sources and considers factors, including the varying transit time between POIs, user preferences, and the total travel time budget. First, the dynamic POI network model is constructed by extracting relevant information from crowdsourced Foursquare and taxi GPS traces. Then, a two-phase approach has been proposed for personalized trip planning with a comprehensive route scoring method and a novel route search-augmentation-ranking process.

2.6 A Decision Tree based Recommendation System for Tourists

Authors: **Pree Thiengburanathum , Shuang Cangb , Hongnian Yuc**

This paper proposes a novel human-centric TRS that recommends destinations to tourists in an unfamiliar city. It considers both technical and practical aspects using a real world data set we collected. The proposed TRS is processed offline using the Data Mining (DM) process. This includes data acquisition, variable selection by using feature selection methods, decision making by using decision tree C4.5, and interpretation of the decision tree. The proposed TRS has three main innovations.

Firstly, two feature selection methods are used to remove the unnecessary (both irrelevant and redundant) inputs into the system and to decrease the model complexity. Secondly, a decision tree C4.5 is used as a classifier to identify the tourist destination selection process. Lastly, the proposed system uses real world data that have been collected by us from Chiang Mai, Thailand.

2.6.1 Methodology

The proposed DM framework shown in Fig. 2.7 consists of four phases including data acquisition, data preprocessing, data analysis, and result interpretation.

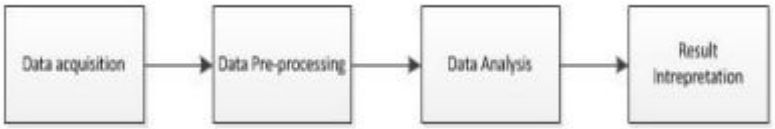

Fig 2.7 Data Mining Framework

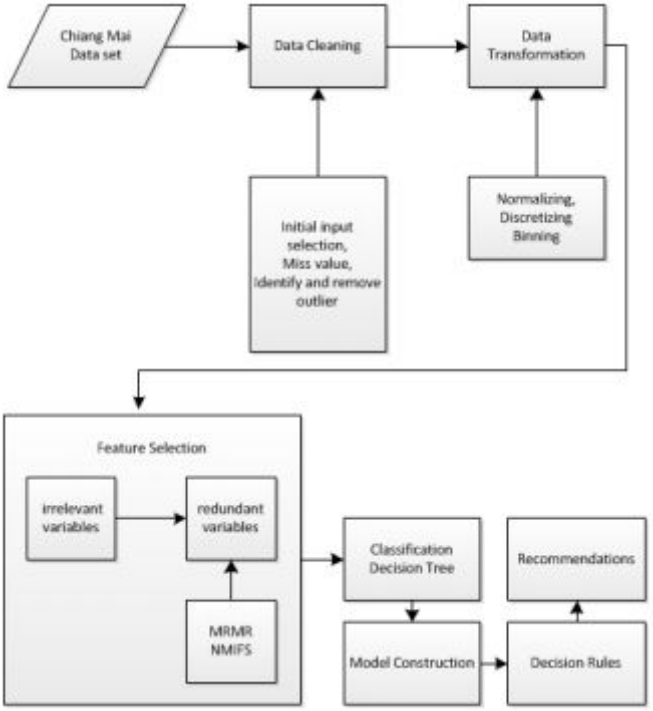

Fig 2.8 Methodology of proposed TRS

A. Data acquisition

To understand tourist's search behavior in assessing travel information and decision-making processing for destination choice, a questionnaire is used as a data collection method due to its effective mechanism for collecting information from tourists. The questionnaire design contains four parts: Trip characteristics, Tourist characteristics, Travel motivations, Tourist socio-demographic information. The proposed framework uses variables extracted from questionnaires as inputs for

classification of the tourist's preferred destination.

B. Data Pre-processing

Real world data are generally incomplete, noisy, and inconsistent. The best classification results require good quality data. To achieve this, the survey data is preprocessed through data integration, data cleaning, data transformation, and variable selection using feature selection methods.

C. Data Analysis

Decision tree is chosen as a classifier/model for the proposed TRS because it provides several benefits for decision maker such as simplicity, interpretability. Decision-making is easily understood due to its flowchart-like model. To recommend a destination to tourists, we must traverse the decision tree from the root to the leaf. C4.5 is an extension of ID3.

2.6.2 Conclusion

In this paper, a decision tree based tourist recommendation system has been presented in an attempt of solving the current challenge of the destination TRS. The data set has been decomposed into two sub data sets using relevant tourism domain knowledge. This was done to increase classification accuracy rate and to reduce the complexity of the decision tree. The optimal decision trees from NMIFS with the highest accuracy rate and simplicity (i.e. less number of leaf and tree size) have been constructed for destination choice. The decision rules from decision trees were extracted. It can be seen that NMIFS is the optimum method because it uses fewer numbers of features than MRMR for both of the data sets. Finally, the experimental results confirm the applicability of the proposed TRS. The proposed TRS satisfies the tourists' requirements who plan to visit or during their visit to the city of Chiang Mai.

2.7 Intelligent travel recommendation system by mining attributes from community contributed photos

Authors: Subramaniyaswamy V , Vijayakumar V , Logesh R and Indragandhi V

This paper proposes a system which helps users in finding tourist locations that he/she might like to visit a place from available user contributed photos of that place available on photo sharing websites. This paper describes methods used to mine demographic information and provide travel recommendations to users. This paper also describes an algorithm adaboost to classify data and Bayesian Learning model for predicting desired location to a user based on his/her preferences. The major differences between our work and other related studies are that we bring in concept people attributes such as gender, group type, travel season in the travel photos and consider demographic information with the movements of photographers into a personalized travel recommendation framework.

2.7.1 System Architecture

Architecture of System is shown in Fig 2.9, it has four main steps. First step is to collect photos from photo sharing websites such as Facebook, Flickr where people share and tag their photos. After collecting photos identify the destination in which photos are captured. In second step people attributes such as gender, race, age, and travel season are detected.

Group types (friends, family, couple, solo) are predicted from detected people attributes. Trips are generated from detected people attributes by sorting out photos of users according to captured date and time and building a travel recommendation model which is further assisted by a Probabilistic Bayesian model.

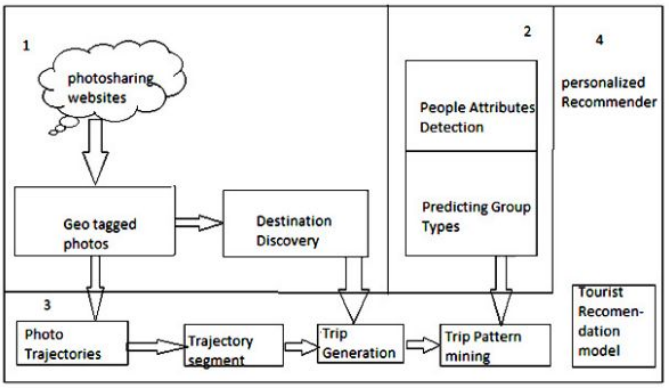

Fig. 2.9 System Architecture

2.7.2 Intelligent Travel Recommendation Model

The Bayesian learning model has been adopted as a recommendation model because Bayesian is simple and most effective in recommendation systems. Bayes theorem states that the probability that the location L_j is suggested destination given a start location Li and attribute value PRu of a specific user u is:

$$P(L_{i\to j}/\,PR_u) \;=\; P(L_{i\to j}/\,PR_u)\,/\,P(PR_u)$$

$$P(L_{i\to j}/\,PR_u) \;=\; P(L_{i\to j}) * P(\,PR_u/L_{i\to j})\,/\,P(PR_u)$$

Eqn 2.9

$L_{i->j}$ is a trip segment (L_i is starting location L_j is ending location).We need to predict the location L_j probabilities that the user might like to visit from a location L_i .To calculate L_j equation (1) can further be transformed into following equation

$$P(L_{i\to j}/\,PR_u) \;=\; P(L_i) * P(L_j/L_i) * P(\,PR_u/L_{i\to j})\,/\,P(PR_u)$$

Eqn 2.10

2.7.3 Conclusion

In this paper, a probabilistic travel recommendation model has been proposed which exploits automatically mined knowledge from user contributed photo tags as well as

the detected people attributes, travel group types and travel group season in photo contents.

2. 8. Visiting Time Prediction Using Machine Learning Regression Algorithm

Authors : Indri Hapsari, Isti surjandari, Komarudin

Travellers need prediction of traveling and visiting time, to arrange their journey. If traveling time has been predicted accurately by Google Map using the location feature, visiting time has another issue. Until today, Google detects the user's position based on crowdsourcing data from customer visits to a specific location over the last several weeks. It cannot be denied that this method will give valid information for the tourists. However, because it needs a lot of data, there are many destinations that have no information about visiting time.

Multiple linear regression becomes the common model, because there are six factors that influenced the visiting time, i.e. access, government, rating, number of reviews, number of pictures, and other information. Those factors become the independent variables to predict dependent variables or visiting time.

This paper used the classification algorithm in machine learning by using 5 top algorithms which are Linear Regression, k-Nearest Neighbors, Decision Tree, Support Vector Machines, and Multi-Layer Perceptron. Based on maximum correlation coefficient and minimum root mean square error, Linear Regression with 6 independent variables has the best result with the correlation coefficient 20.41% and root mean square error 48.46%.

2.8.1 Methodology

In the beginning the relationship between dependent and independent variables was assumed to be linear. Model of predicting visiting time uses multiple regression because there are some independent variables from the Google Map feature that will

predict one dependent variable. For dependent variable x_j and m independent variables x_1, x_2, ..., x_k, x_m is the linear multiple regression of xj on the independent variables is given by the equation:

$$\hat{x_j} = \alpha_\tau + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + ... + \beta_m x_m$$

Eqn 2.11

Where the regression coefficient for any of the variables, say β_k for x_k is the linear regression coefficient for the least squares fit of x_i to x_k, holding the other independent variables constant. However, the model must be valid first before it will be used to predict the dependent variable. The requirements are:

1. The dependent variable is a linear function of the independent variables or linearity.

2. Each person (or other observation) should be drawn independently from the population.

3. The variance of the errors is not a function of any of the independent variables. This assumption is referred to as homoscedasticity.

4. The errors are normally distributed.

5. The dependent variable does not influence any of the independent variables.

6. The independent variables are measured without error, with perfect reliability and validity.

7. The regression must include all common causes of the presumed cause and the presumed effect.

After running the data, the correlation coefficient, and root mean squared error for 5 regression algorithms were compared (TABLE 2.1).

Method	Correlation	RMSE
LinearRegression	20.41	48.47
lazy.IBk	19.74	61.25
REPTree	13.36	51.39
SMOreg	14.85	49.66
MultilayerPerceptron	2.61	63.93

TABLE 2.1 Top 5 Algorithm Performance Comparison

2.8.2 Conclusion

Predicting the visiting time is a mandatory requirement for applications that have function to develop a good route for the traveller. Even though we can use information from Google crowdsourcing in the future, it still becomes a requirement to analyse all the factors that make tourists stay longer than other destinations. Because there are six independent variables that influence the dependent variable which is visiting time, the paper uses multiple linear regression to predict it.

2.9 Weather Prediction Based on Decision Tree Algorithm Using Data Mining Techniques

Authors : Siddharth S. Bhatkande, Roopa G. Hubballi2

Presently, weather forecasting is made through the application of science and technology. It is made by gathering quantitative information sets about the present condition of the environment through climate stations and deciphers by meteorologists. Decision tree is considered as the powerful solution to the classification problems and it is applied in many real world applications.

2.9.1 Workflow

In this approach paper is completed in four stages as shown in the Fig. 2.10. Data collection and data pre- processing, data cleaning, data selection and finally data transportation. Generally responsible parameters for the weather prediction are maximum temperature and minimum temperature. These are collected from weather departments like meteorologists centre and then perform the decision tree algorithm on available datasets and predict the future weather such as day wise or months or years.

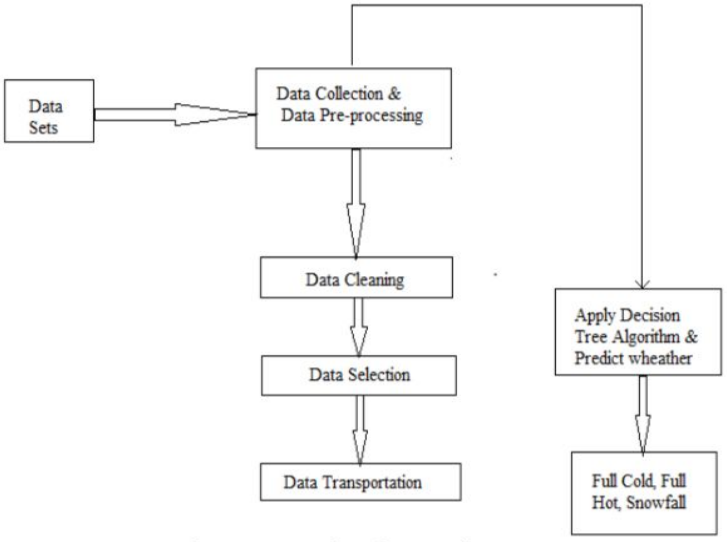

Fig. 2.10 Work flow diagram

- **Data Collection**: The data used for this work was collected from the meteorologist's centre. The case data covered the period of 2012 to 2015. The following procedures were adopted at this stage of the research: Data Cleaning, Data Selection, Data Transformation and Data Mining.

- **Data Cleaning**: In this stage, a consistent format for the data model was developed which is search for missing data, finding duplicated data, and weeding out bad data. Finally, system cleaned data were transformed into a format suitable for data mining.

- **Data Selection**: At this stage, data relevant to the analysis like decision tree was decided on and retrieved from the dataset. The Meteorological dataset had ten attributes that were using two attributes for future prediction.

- **Data Transformation**: "This is also known as data consolidation". It is the stage in which the selected data is transformed into forms appropriate for data mining. The data file was saved in Commas Separated Value (CVS) file format and the datasets were normalized to reduce the effect of scaling on the data.

- **Data Mining Stage**: The data mining stage was divided into three phases. At each phase all the algorithms were used to analyse the meteorological datasets. The testing method adopted for this research was percentage split that trained on a percentage of the dataset, cross validated on it and tested on the remaining percentage. There after interesting patterns representing knowledge were identified.

In order to enhance the precision and speculation of arrangement and relapse trees, different systems were presented like boosting and pruning. Decision tree construct arrangement or relapse models as a tree structure. It separates the datasets into littler and littler subsets while in the meantime a related decision tree is incrementally created. The last result is a tree with decision nodes and leaf nodes. The highest decision node in a tree which relates to the best indicator called root node. Decision trees can deal with both clear cut and numerical information. Fig. 2.11 shows a Decision Tree.

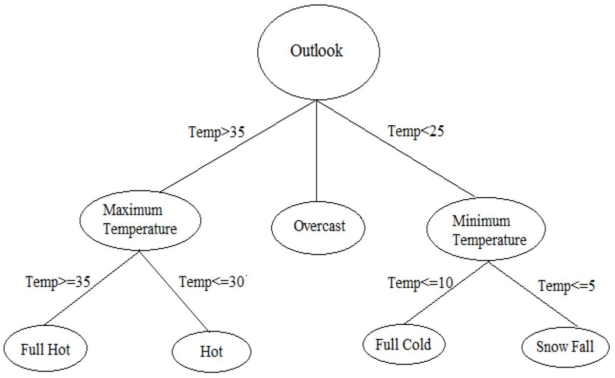

Fig 2.11 Decision Tree generated by Training data sets

2.9.2 Conclusion

In this paper a data mining technique has been used and a Decision tree algorithm for classifying weather parameters such as maximum temperature, minimum temperature in terms of the day, month and year. Decision trees prove as an effective method of Decision making in Weather prediction. As decision trees are ideal for multiple variable analyses, it is particularly important in current problem-solving tasks like weather forecasting. This work is important to climatic change studies because the variation in weather conditions in terms of temperature, rainfall and wind speed can be studied using these data mining techniques.

2.10 Smart Weather Forecasting using ML

Authors : A H M Jakaria, Md Mosharaf Hossain , Mohammad Ashiqur Rahman

This proposes a method to utilize the surrounding city's historical weather data along with a particular city's data to predict its weather condition. This data is combined and used to train simple machine learning models, which in turn, can predict correct weather conditions for the next few days. These simple models can be run on low cost and less resource intensive computing systems, yet can provide quick and accurate enough forecasts to be used in our day-to-day life.

2.10.1 Contributions

- The utilization of machine learning in prediction of weather conditions in short periods of time, which can run on less resource-intensive machines.
- Implementation of automated systems to collect historical data from a dedicated weather service.
- Thorough evaluation of the proposed technique and comparison of several machine learning models in the prediction of future weather conditions.

2.10.2 Methodology

In this paper, ML techniques are used to predict the temperature of the next day at any particular hour of the city. First, the weather observations at a particular timestamp from all the cities is combined that is considered to create a single record. That is, each record in the data will contain the temperature, humidity, wind direction, atmospheric pressure, condition, etc. data of all the cities. The target variable for this record is assigned as the temperature at the same timestamp of the next day. Thus, the temperature of the next day is predicted given the weather observation of the current day.

The predicted outcomes are continuous numeric values, temperature in this case, so regression technique is used. It is found that Random Forest Regression (RFR) is the superior regressor, as it ensembles multiple decision trees while making decisions. In addition, comparison of several other state-of-the-art ML techniques is shown with the RFR technique. The incorporated regression techniques are Ridge Regression (Ridge), Support Vector (SVR), Multi-layer Perceptron (MLPR), and Extra-Tree Regression (ETR).

After having the raw data from ‘wunderground’, each row (record) in the dataset has records for all ten cities for a particular timestamp. Any feature with empty or invalid data is eliminated while creating the dataset. Also, the categorical features in the dataset are converted, such as wind direction and condition, into dummy/indicator

variables using a technique called 'One Hot Encoding'. Furthermore, we perform mean scaling $x \leftarrow \frac{x-\mu}{\sigma}$ to all the continuous variables so that the variables possess approximately zero mean, which in practice reduces the computational cost while training the model.

2.10.3 Conclusion

This paper presented a technology to utilize machine learning techniques to provide weather forecasts. Machine learning technology can provide intelligent models, which are much simpler than traditional physical models. They are less resource-hungry and can easily be run on almost any computer including mobile devices. Evaluation results show that these machine learning models can predict weather features accurately enough to compete with traditional models. The historical data from surrounding areas was also utilized to predict weather of a particular area. It is shown that it is more effective than considering only the area for which weather forecasting is done.

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

A Software Requirement Specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. It may include the use cases of how user is going to interact with software system. The software requirements specification document consistent of all necessary requirements required for project development. To develop the software system, we should have clear understanding of software system. To achieve this, we have to gather information accurately and meticulously.

3.1 Functional Requirements:

Functional requirement may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. The following are the functional requirement for our project: The user can enter the source and the destination.

- The application should generate common places of interest based on the region.
- The user can select his/her places of interest from the generated places of interest.
- The application should generate the most optimal path which includes almost all site-seeing spots of the user's interest.
- The application should suggest the best time to visit that region.
- A map should display the optimal path.

3.2 Non-Functional Requirements:

- Non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. The following are the non-functional requirements of our project:
- The optimal path must be generated without much delay.
- An accurate suggestion of the best time to visit the region must be made.
- The optimal path must be displayed on a map accurately.

3.3 Hardware Requirements:

Hardware requirement specifies the minimum hardware that is required for the application to run smoothly. The following are the hardware requirement for our project:

- Processor: Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
- Installed memory(RAM): 8.00GB
- System type: 64-bit Operating System, x64-based processor
- Internet connection

3.4 Software Requirements:

Software requirement specifies the minimum software that is required for the application to run smoothly. The following are the software requirement for our project:

- Django
- Text editor
- HTML

- CSS
- Bootstrap
- JavaScript
- Python

3.4.1 Django

Django is a Python-based free and open-source web framework that follows the model-template-view (MTV) architectural pattern. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Despite having its own nomenclature, such as naming the callable objects generating the HTTP responses "views", the core Django framework can be seen as an MVC architecture. It consists of an object-relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database ("Model"), a system for processing HTTP requests with a web templating system ("View"), and a regular-expression-based URL dispatcher ("Controller").

Also included in the core framework are:

- a lightweight and standalone web server for development and testing
- a form serialization and validation system that can translate between HTML forms and values suitable for storage in the database
- a template system that utilizes the concept of inheritance borrowed from object-oriented programming
- a caching framework that can use any of several cache methods

- support for middleware classes that can intervene at various stages of request processing and carry out custom functions
- an internal dispatcher system that allows components of an application to communicate events to each other via pre-defined signals
- an internationalization system, including translations of Django's own components into a variety of languages
- a serialization system that can produce and read XML and/or JSON representations of Django model instances
- a system for extending the capabilities of the template engine
- an interface to Python's built-in unit test framework
- Django REST framework is a powerful and flexible toolkit for building Web APIs.

3.4.2 HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets.

3.4.3 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

3.4.4 Bootstrap

Bootstrap is a powerful toolkit - a collection of HTML, CSS, and JavaScript tools for creating and building web pages and web applications. After its open source release in 2011, Bootstrap became popular very quickly, and not without reason. Web designers and web developers like Bootstrap because it is flexible and easy to work with.

Its main advantages are that it is responsive by design, it maintains wide browser compatibility, it offers consistent design by using re-usable components, and it is very easy to use and quick to learn. It offers rich extensibility using JavaScript, coming with built-in support for jQuery plugins and a programmatic JavaScript API. Bootstrap can be used with any IDE or editor, and any server side technology and language, from ASP.NET to PHP to Ruby on Rails.

With Bootstrap, web developers can concentrate on the development work, without worrying about design, and get a good looking website up and running

quickly. Conversely, it gives web designers a solid foundation for creating interesting Bootstrap themes.

We have used Bootstrap 4 in out project to design the welcome page carousel. The carousel is a slideshow for cycling through a series of content, built with CSS 3D transforms and a bit of JavaScript. It works with a series of images, text, or custom markup. It also includes support for previous/next controls and indicators.

3.4.5 JavaScript

JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.

JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behaviour.

Contrary to popular misconception, JavaScript is *not* "Interpreted Java". In a nutshell, JavaScript is a dynamic scripting language supporting prototype based object construction. The basic syntax is intentionally similar to both Java and C++ to reduce the number of new concepts required to learn the language. Language constructs, such as if statements, for and while loops, and switch and try ... catch blocks function the same as in these languages.

JavaScript can function as both a procedural and an object oriented language. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects at run time, as opposed to the syntactic class

definitions common in compiled languages like C++ and Java. Once an object has been constructed it can be used as a blueprint for creating similar objects.

3.4.6 Python

Python is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

Our trip recommendation system has been designed to provide recommendation to the user based on the user's interest. For the purpose of considering user's preference the system needs to take an input from the user. From this data, the system needs to suggest the most optimal routes to the specified destination considering the user's interests to visit multiple destinations on the way to their final destination. Our system should also suggest the best time to visit the places of interest. For suggesting best time, it needs to make weather prediction by considering past weather data and analysing it. So, the system will consist of several modules to generate optimal route and also provide best time to travel.

4.1 System Architecture

The system consists of several modules that are integrated together to provide recommendation to the user. Fig 4.1 depicts the architecture of our trip recommendation system. The main components of the system include:

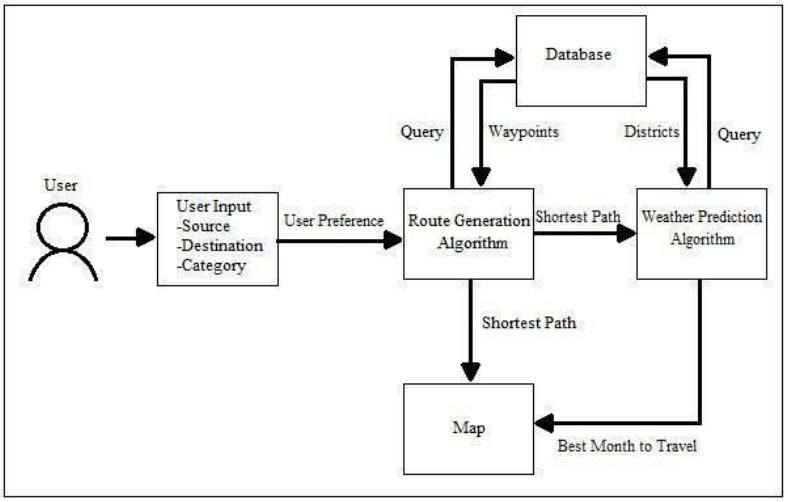

Fig 4.1 System Architecture

1. **User Input:** The user has to select the source and the destination from the drop-down consisting of 30 districts and also select the category of places they want to visit. The categories included are: ‘Spiritual’, ’Heritage’, ’Trekking’, ’Rivers and Waterfall’, ’Wildlife’, ’Hill station’ and ‘Beach’.

2. **Database:** The database contains waypoints along with the district they belong to and the category they fall under.

3. **Route Generation Algorithm:** The user input is given to the route generation algorithm. The algorithm queries the database to get the waypoints which fall under the selected categories, generates the optimal path from source to the destination considering the waypoints which fall under the users selected category along the way.

4. **Weather Prediction Algorithm:** The result obtained by route generation algorithm is given as input to the weather prediction algorithm which first determines the district under which the intermediate waypoints fall by querying the database and then predicts the best months to travel to those districts, and then gives the intersection as the final result.

5. **Map:** The output from the Route Generation Algorithm and the Weather Prediction Algorithm is given as input to the optimal route page. With the help of Google map API the path is plotted in the map and the other information along with the best months to travel from source to destination is also given in the side pane.

CHAPTER 5

IMPLEMENTATION

5.1 Generation of the Optimal Path

The optimal path was generated using permutation and combinations. Python has a lot of in-built packages to calculate permutations of an array of string or integer. We use itertools in our code to compute the combinations of places. This module implements a number of iterator building blocks inspired by constructs from APL, Haskell, and SML. Each has been recast in a form suitable for Python. Itertools standardizes a core set of fast, memory efficient tools that are useful by themselves or in combination. Together, they form an "iterator algebra" making it possible to construct specialized tools succinctly and efficiently in pure Python.

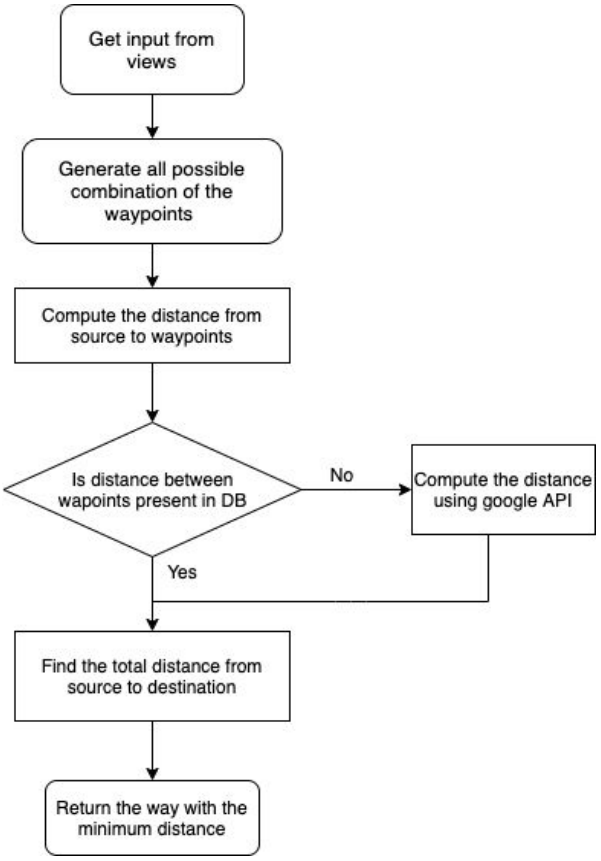

Fig 5.1 Flow Chart for Route Generation

In the database the preferences are stored on the basis on the category given as input. The input category are namely Heritage, Water, Wildlife, Beach, Spiritual, Trekking and Hill Station. All the places with the input preferences are computed using a DB query. The resultant QuerySet is a nested list of the places of the preferences. Using itertools the product of the preferences is computed; hence we have all the possible locations that can be there.

Distance between the two places is calculated by using Google API. The Distance Matrix API is a service that provides travel distance and time for a matrix of origins and destinations. The API returns information based on the recommended route between start and end points, as calculated by the Google Maps API, and consists of rows containing duration and distance values for each pair. All the combinations for the districts and the distance between them is already computed and stored in the DB. If the combination is new then the distance is calculated using API. This saves us a lot of time as computing every time can be expensive and time consuming.

Once we have all the combinations and distances we compare the distance amongst all and the routes and the one with the minimum distance is given as the output. To make the process faster we use multithreading. Multithreading is the ability of a central processing unit (CPU) (or a single core in a multi-core processor) to provide multiple threads of execution concurrently, supported by the operating system. This approach differs from multiprocessing.

5.2 Predicting The Best Months to Travel

The next part of implementation is to predict the best months to travel. The main objective here is to predict the most favourable months to travel from source to destination, for doing so past monthly forecast is required for each and every waypoint.

5.2.1 Construction of The Data-Set

The data-set required for the prediction of the best months to travel was prepared manually since no ready-made data-set was available to satisfy our requirement.

The monthly average for various places in Karnataka was taken from a website "www.timeanddate.com" which provides the average values for various parameters which is used to determine the weather. It contained the weather information for the past 5 years.

The data-set contained the past 5 years' monthly average of the various parameters used to determine the weather. Since the website provided the monthly averages for the districts, we determined the districts where the way points lie and then collected the past five years' monthly averages for those districts.

There were certain districts for which the weather data was not available for such places we took the weather information for the districts which was closest to the former districts, this will not result in huge increase in error for predicting the best months to travel since we are only concentrating places in Karnataka and the weather conditions do not vary drastically throughout Karnataka.

The data-set prepared for the prediction of the best months to travel is given in Fig 5.1

```
Place,Month,HighTemp,LowTemp,MeanTemp,Precipitation,Humidity,DewPoint,Wind,Pressure,Visibility
Bangalore,Jan,29,16,23,1.0,58,13,4,1013,6
Bangalore,Feb,31,18,24,7.5,49,12,4,1011,7
Bangalore,March,33,20,27,22.6,46,13,4,1009,7
Bangalore,April,34,22,28,85.6,53,17,3,1007,7
Bangalore,May,33,22,27,124.8,65,19,5,1006,7
Bangalore,June,30,21,25,94.8,73,19,8,1006,7
Bangalore,July,29,20,24,138.3,77,19,9,1006,6
Bangalore,August,28,20,24,170.0,78,19,7,1007,6
Bangalore,September,29,20,24,176.6,76,19,6,1008,7
Bangalore,October,29,20,24,116.9,74,18,4,1010,6
Bangalore,November,28,18,23,79.2,73,17,3,1011,6
Bangalore,December,28,17,22,11.2,68,15,4,1012,6
Mangalore,Jan,33,21,27,0.6,65,19,8,1011,4
Mangalore,Feb,34,22,28,5.0,66,20,8,1011,4
Mangalore,March,34,24,29,21.7,71,22,8,1010,4
Mangalore,April,34,25,29,38.0,72,24,8,1009,4
Mangalore,May,33,25,29,87.7,76,24,9,1008,4
Mangalore,June,30,24,27,414.9,87,24,9,1007,4
Mangalore,July,29,23,26,485.2,89,24,10,1008,4
Mangalore,August,29,23,26,364.4,89,24,9,1008,4
Mangalore,September,30,23,27,275.5,86,24,8,1009,4
Mangalore,October,32,23,27,150.8,83,24,7,1009,4
Mangalore,November,33,23,28,109.6,75,22,7,1009,4
Mangalore,December,33,22,28,12.1,67,20,8,1010,4
Mysore,Jan,34,22,28,85.6,53,17,3,1007,7
Mysore,Feb,31,18,24,7.5,49,12,4,1011,7
Mysore,March,33,20,27,22.6,46,13,4,1009,7
Mysore,April,34,22,28,85.6,53,17,3,1007,7
Mysore,May,33,22,27,124.8,65,19,5,1006,7
Mysore,June,30,21,25,94.8,73,19,8,1006,7
```

Fig 5.2 Data-Set

The data-set contained 276 instances i.e., it contained the monthly average for 23 districts. Each instance was the average monthly values for the past 5 years for various parameters that is used to determine the weather.

The data-set contained a total of 9 attributes:

1. HighTemp: the average highest temperature experienced in the area for the month.
2. LowTemp: the average lowest temperature experienced in the area for the month.
3. MeanTemp: the average temperature experienced in that area
4. Precipitation : Precipitation is a form of water from the atmosphere.
5. Humidity : Humidity means water vapour in the air
6. DewPoint: the dew point is the temperature where the water vapour condenses into liquid water.
7. Wind: the average measure of wind in the area.
8. Pressure: the average pressure in the area.
9. Visibility: the % of fog in the area.

There were no missing values the data-set since the data-set was created manually.

Training data-set was created and contained the target attribute "SuitableForTravelling" which could possess two possible value either 'yes' or 'no'.

The training data-set contained 48 instances.

```
Place,Month,HighTemp,LowTemp,MeanTemp,Precipitation,Humidity,DewPoint,Wind,Pressure,Visibility,SuitableForTravelling
Bangalore,Jan,29,16,23,1.0,58,13,4,1013,6,Yes
Bangalore,Feb,31,18,24,7.5,49,12,4,1011,7,Yes
Bangalore,March,33,20,27,22.6,46,13,4,1009,7,No
Bangalore,April,34,22,28,85.6,53,17,3,1007,7,No
Bangalore,May,33,22,27,124.8,65,19,5,1006,7,No
Bangalore,June,30,21,25,94.8,73,19,8,1006,7,Yes
Bangalore,July,29,20,24,138.3,77,19,9,1006,6,No
Bangalore,August,28,20,24,170.0,78,19,7,1007,6,No
Bangalore,September,29,20,24,176.6,76,19,6,1008,7,No
Bangalore,October,29,20,24,116.9,74,18,4,1010,6,Yes
Bangalore,November,28,18,23,79.2,73,17,3,1011,6,Yes
```

Fig 5.3 Training Data-Set

Testing data-set also contained the target attribute and contained 24 instances.

```
Place,Month,HighTemp,LowTemp,MeanTemp,Precipitation,Humidity,DewPoint,Wind,Pressure,Visibility,SuitableForTravelling
Udupi,Jan,33,21,27,0.6,65,19,8,1011,4,Yes
Udupi,Feb,34,22,28,5.0,66,20,8,1011,4,Yes
Udupi,March,34,24,29,21.7,71,22,8,1010,4,Yes
Udupi,April,34,25,29,38.0,72,24,8,1009,4,No
Udupi,May,33,25,29,87.7,76,24,9,1008,4,No
Udupi,June,30,24,27,414.9,87,24,9,1007,4,No
Udupi,July,29,23,26,485.2,89,24,10,1008,4,No
Udupi,August,29,23,26,364.4,89,24,9,1008,4,No
Udupi,September,30,23,27,275.5,86,24,8,1009,4,No
Udupi,October,32,23,27,150.8,83,24,7,1009,4,Yes
Udupi,November,33,23,28,109.6,75,22,7,1009,4,Yes
Udupi,December,33,22,28,12.1,67,20,8,1010,4,Yes
Kolar,Jan,29,16,23,1.0,58,13,4,1013,6,Yes
Kolar,Feb,31,18,24,7.5,49,12,4,1011,7,Yes
```

Fig 5.4 Testing Data-Set

5.2.2 Decision Tree Algorithm for The Prediction of The Best Months To Travel

Since the main objective involves the prediction of the best time to travel various machine learning algorithm such as regression, decision tree and Artificial Neural Network (ANN) were considered to be candidates for the same. Since decision tree is easy to understand and can be used to implement complex problem decision tree was selected to prepare the model to predict the best months to travel. Decision tree has a number of advantages such as:

- Comprehensive
- Specific
- Easy to use

- Versatile, etc.

Classic CART (**C**lassifications **A**nd **R**egression **T**rees) algorithm was used for the construction of the decision tree. The classic CART algorithm uses the Gini index for the construction of the decision tree.

Gini index is calculated by subtracting the sum of squared probabilities of each class from one. It favours larger partitions and is easy to implement whereas information gain favours smaller partitions with distinct values. A feature with a lower gini index is chosen for the split.

The attributes 'HighTemp' and 'LowTemp' does not contribute much in construction of the final decision tree as we have another attribute 'MeanTemp'. Hence the two attributes were not considered for the construction of the decision tree.

Then features used for the prediction of the best time to travel are : 'MeanTemp','Precipitation','DewPoint','Wind','Pressure' and 'Visibility'.

DecisionTreeClassifier from the library sklearn.tree was used for the construction of the decision to predict the best time to trave[4].

Fig 5.5 is the decision tree obtained after training. Hence this model could predict the best months to travel given any district whose weather data was present in the dataset. The total accuracy obtained is 91.67% which is considerable high and also overfitting is also negligible since the accuracy obtained for training set is 95.83%.

Since our main objective was to find the most favourable months to travel from source to the destination, the intermediate points obtained from the optimal path generation algorithm were taken. Once the intermediate points were obtained, the districts to which these points lie were determined. After obtaining the districts, the decision tree classifier was used to predict the best months to travel to each and every district separately and the result was saved as a list of list. Finally, the intersection of the list of list was found which would contain all the common favourable months to travel to

those places and this result was passed on to the UI where the best months to travel was displayed.

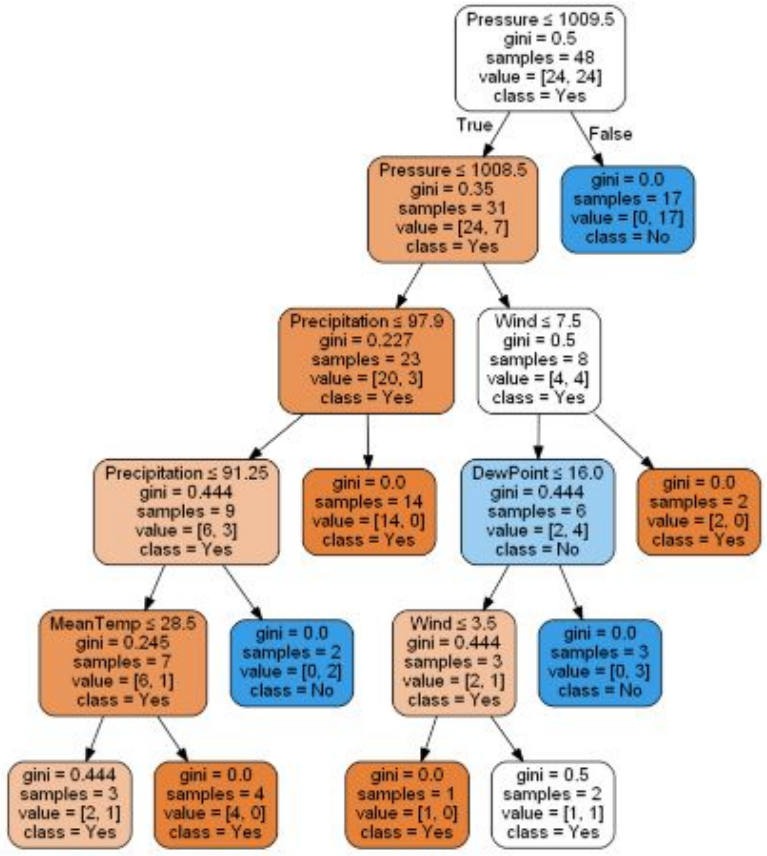

Fig 5.5 Decision tree

5.2.3 Code: Prediction Of The Best Months To Travel From Source To The Destination Using CART Algorithm

```
import pandas as pd

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

from sklearn import metrics
```

```
from website.models import Waypoints
from functools import reduce
def district_fetch(waypoints):
  district = []
  for way in waypoints:
    wp=Waypoints.objects.filter(name__iexact=way)
    district.append(wp[0].district)
  return district
def weather_prediction(way):
  d=pd.read_csv('website/training.csv')
  d_test = pd.read_csv('website/testing.csv')
  feature_cols = ['MeanTemp','Precipitation','DewPoint','Wind','Pressure','Visibility']
  X = d[feature_cols] # Features
  y = d.SuitableForTravelling # Target variable
  x_test = d_test[feature_cols]
  y_test = d_test.SuitableForTravelling
  clf = DecisionTreeClassifier()
  clf = clf.fit(X,y)
  testing = pd.read_csv('website/weather_dataset.csv')
  districts = district_fetch(way)
  print(districts)
```

```
months = []
for district in districts:
  new = testing.loc[testing['Place'] == district]
  a= []
  X = new[feature_cols]
  y_pred = clf.predict(X)
  for i in range(len(y_pred)):
    if y_pred[i] == 'Yes' :
      a.append(new['Month'].iloc[i])
  months.append(a)
favourable_for_travelling = list(reduce(lambda i, j: i & j, (set(x) for x in months)))
print(months)
print(favourable_for_travelling)
return  favourable_for_travelling
```

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Generation of the Optimal Path

The process was taking a lot of time if distance between two places was being calculated every time using Google API. The solution to this problem was that every time we get a new combination and we store it in the DB. This way we were able to process faster and more efficiently. Since permutations and combinations return a lot data it is inefficient to process the data in a single block. So multithreading was used. Blocks of capacity 500 were made and the single block was divided into parts.

Multiple preferences input are possible. For 1 possible input the time taken is 0.0978 seconds. For 2 possible inputs the time taken is 0.1068 seconds. For 3 possible inputs the time taken is 0.1178 seconds. For 4 possible inputs the time taken is 0.3033 seconds. For 5 possible inputs the time taken is 2.5878 seconds. For more than 5 preferences it takes more than 30 seconds time which is not optimal. The value grows exponentially and it not suitable for more than 5 preferences.

6.2 Predicting The Best Months to Travel

The classifier was tested using the testing data-set that contained 48 instances and the accuracy achieved by the classifier was 91.67% which is considerably good.

Overfitting was also negligible since the accuracy obtained for the training data-set was 95.83% and the accuracy obtained for the testing data-set was 91.67%. High accuracy was obtained in the prediction of the best months to travel since we are only considering the places in Karnataka and the weather wont vary drastically over the region.

There were districts whose weather data could not be found, for such places we considered the weather data of the nearest district and this did not lead to huge increase in the error since as mentioned before we are only considering places in Karnataka and the weather over Karnataka does not vary to great extent.

There is still room to improve in the prediction of the best months to travel since we are considering the data for the past 5 years only, accuracy can be further increased if we increase the amount of historic data. Also right now the data-set contains only the weather data for 23 districts of Karnataka, including more districts and also expanding the data-set is future scope of the project. Another challenge is the definition of the favourable weather can change according to the place you are visiting e.g. people tend to visit the beach even if the temperature is more than 28 degrees the same is not true for other places, hence this aspect could not be considered when predicting the best months to travel since we are considering the district wise weather data and not waypoint wise.

The classifier is efficient and also the accuracy is also high, there can be improvement in the classifier for predicting the best months to travel but for the current requirement of the application the classifier does a decent job.

6.3 Screenshots

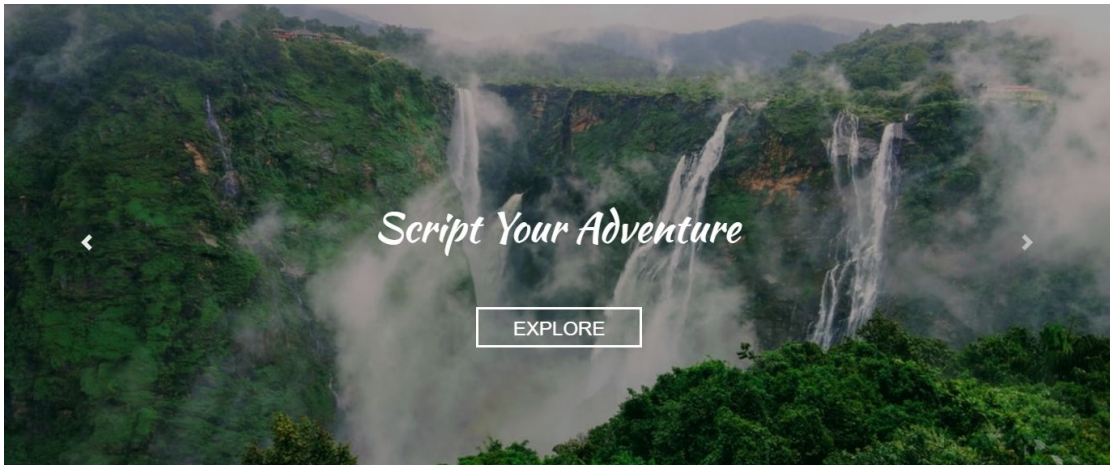

Fig 6.1 Welcome page

Fig 6.1 is the screenshot of the welcome page; this is the first page that the user encounters when visiting the site. Once the user clicks on the explore button he/she will be redirected to the choices page.

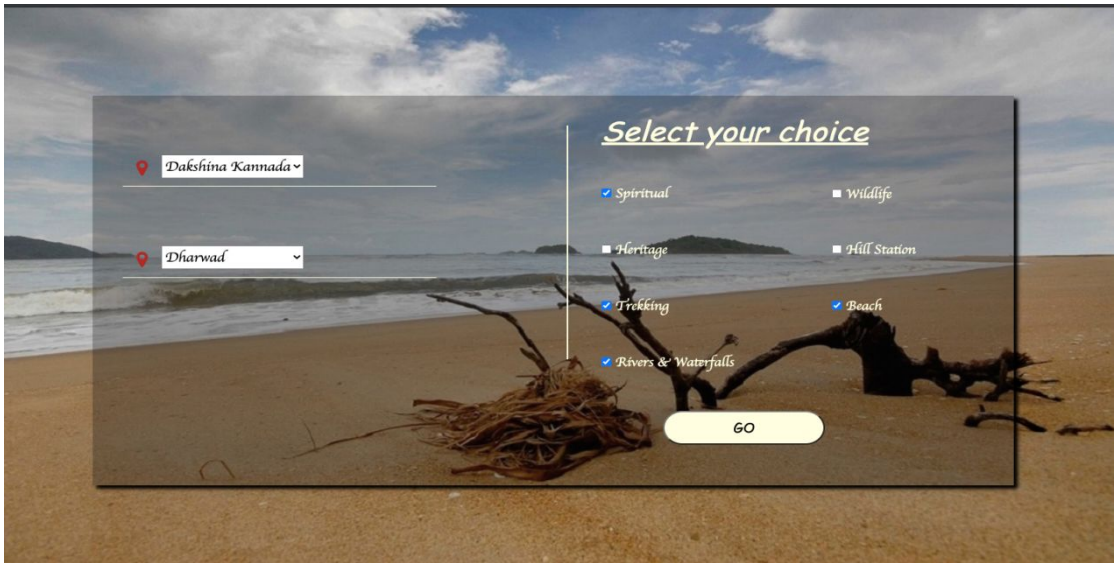

Fig 6.2 Choices page

Fig 6.2 is the screenshot of the choices page which allows the user to select the source and destination from the drop-down which contains a total of 30 districts. The user has to select the types of places that they want to visit. There is a total of 7 choices from which the user can select multiple choices, they are: 'Spiritual',' Heritage', 'Trekking', 'Rivers and Waterfall', 'Wildlife', 'Hill station' and 'Beach'.

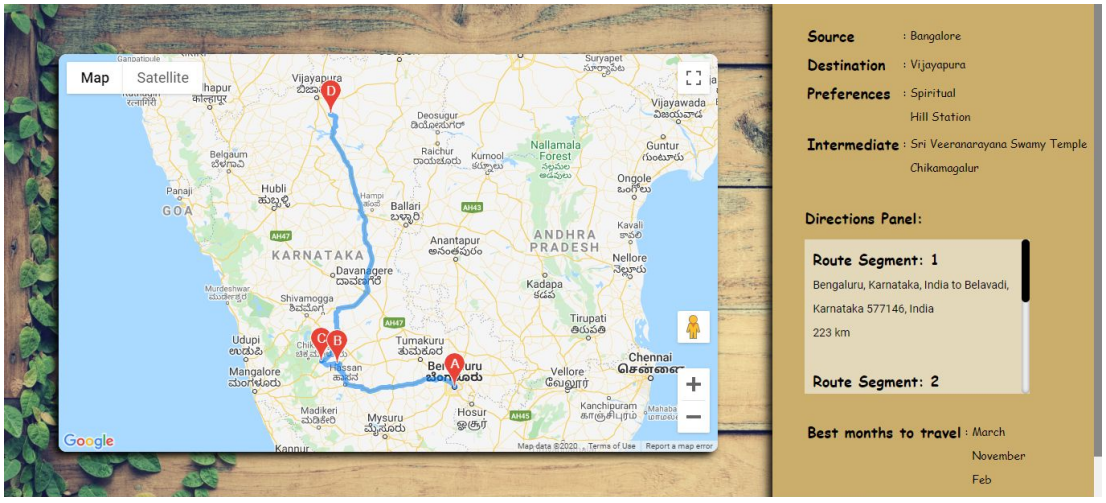

Fig 6.3 Optimal Route

Fig 6.3 is the screenshot of the optimal route that is generated. Once the user enters the source, destination and the type of places and submits the form, the optimal route generation algorithm will compute the route from source to the destination considering all the places which falls under the type of places the user is interested in as the intermediate points. Then this result is passed onto the optimal route page and also to the classifier to predict the best months to travel. The output from the classifier is sent to the optimal route page. The optimal route page plots the route in the map using the Google map API. The map is displayed in the left pane and the right pane displays 'Source', 'Destination', 'Preference', 'Intermediate places', 'Directions panel', and the best months to travel. The direction panel gives information regarding the route segment.

CHAPTER 7

TESTING

7.1. Usability Testing

Usability testing is performed, the application flow and the system navigation is checked, the web page content is checked for spelling and grammatical errors and it is found that it meets the criteria.

7.2. Functional Testing

Functional testing is performed, all the features and operational behaviour of the web page is tested. It is found that the web page meets the intended functional requirement.

7.3. Database Testing

Database testing is performed, the backend records are tested which have been through the web and it is found that the data displayed in the web page matches with the data stored in the database.

7.4. Interface Testing

Interface testing is performed and it is found that the the application, web server and the database server meets the intended requirement.

7.5. Integration Testing

Integration testing is performed, the individual units are combined and tested as a group and no fault was found in the interaction between the integrated units.

CHAPTER 8

CONCLUSION

The existing system[1],[3],[5] for generation of the optimal path does not consider the user preferences, the application that we developed is unique and also efficient as it incorporates the user's preference when generating the path from the source to the destination. Along with generating the optimal path from the source to the destination considering the waypoints that falls under the user's preference along the way it also predicts the best months to travel from the given source to the destination by using the past weather data of the districts under which the waypoints fall.

The application allows the user to enter multiple preferences. The accuracy of the path that is obtained by our algorithm is high and also the weather predicting classifier is able to predict the best months to travel from source to the destination with an accuracy of 91.7% and negligible percent of overfitting.

The application only considers the places within Karnataka currently, expanding the region covered is future work and also the accuracy of predicting the best months to travel can also be improved by including more instances and also by considering the weather data of the place instead of the district in which the place belongs.

Altogether the performance of the developed application is pretty good and it is quite different and unique since it incorporates more functions such as taking user preference into consideration and also predicting the best months to travel. Hence the application greatly reduces the total time spent by us in planning of the road-trip.

REFERENCES

[1] Jie Li, Yujiu Yang, and Wenhuang Liu, ”Exploring Personalized Travel Route Using POIs”, International Journal of Computer Theory and Engineering, Vol. 7, No. 2, April 2015.

[2] Dylan Liu, Christopher Vo and Mark Holmstrom, Stanford University, “Machine Learning Applied to Weather Forecasting”.

[3] “Visiting Time Prediction Using Machine Learning Regression Algorithm”, IEEE 2018 6th International Conference on Information and Communication Technology(ICoICT).

[4] Siddharth S. Bhatkande, Roopa G. Hubballi2, “Weather Prediction Based on Decision Tree Algorithm Using Data Mining Techniques”, International Journal of Advanced Research in Computer and Communication Engineering,Vol. 5, Issue 5, May 2016.

[5] “Trip Planning Route Optimization with Operating Hour and Duration of Stay Constraint”, The 11th International Conference on Computer Science & Education .

[6] Smart Weather Forecasting Using Machine Learning: A Case Studying Tennessee.

2009,” *Univ. Calif. Berkeley*, 2010.